



Simulador y visualizador de sistema de movilidad urbana distribuido, emergente, autónomo y enfocado en la sustentabilidad.

***AUTOR: Andrés Armstrong Cruz
PROFESOR GUÍA: Tomás Vivanco
Diciembre 2017. Santiago, CHILE***

Tesis presentada a la Escuela de Diseño de la Pontificia Universidad Católica de Chile para optar al título profesional de Diseñador.

FACULTAD DE ARQUITECTURA, ESTUDIOS URBANOS Y DISEÑO | UC

Pontificia Universidad Católica de Chile
Escuela de Diseño

Simulador y visualizador de sistema de movilidad urbana distribuido, emergente, autónomo y enfocado en la sustentabilidad.



PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

Simulador y visualizador de sistema de movilidad urbana distribuido, emergente, autónomo y enfocado en la sustentabilidad.

***AUTOR: Andrés Armstrong Cruz
PROFESOR GUÍA: Tomás Vivanco
Diciembre 2017. Santiago, CHILE***

Tesis presentada a la Escuela de Diseño de la Pontificia Universidad Católica de Chile para optar al título profesional de Diseñador.

FACULTAD DE ARQUITECTURA, ESTUDIOS URBANOS Y DISEÑO | UC

Pontificia Universidad Católica de Chile
Escuela de Diseño

AGRADECIMIENTOS

a

Paulina Diaz Schüssler,
Daniel Shiffman,
Tomás Vivanco,
Alessio Bellino,
Ricardo Vega,
Oscar Figueroa,
Anita Schacht,
Mis papás,
Santiago Armstrong,
Diego Cruz,
Felipe Vilches,
Los Cabros,
Mis amigos,
Tito Schneider,
La comunidad de 9gag,
y a Klaus Teuber

ÍNDICE

13	OPINIÓN DE BELTRÁN MENA
15	MARCO TEÓRICO
30	FORMULACIÓN
31	OBJETIVOS
32	CONTEXTO
33	USUARIOS
34	ANTECEDENTES
36	REFERENTES
41	PROCESO DE DISEÑO Y VALIDACIÓN
139	PRODUCTO FINAL: EL SIMULADOR
177	ANEXOS DIGITALES
178	REFERENCIAS



Ridículos

por BELTRÁN MENA

Diario El Mercurio,
Artes & Letras, Domingo 13
de noviembre de 2005

Nos detenemos en una esquina a ver pasar los autos.

Luego de un rato, así como una palabra pierde su sentido si la repetimos cien veces el conductor atrapado se nos aparece en toda su ridícula condición.

No importan las insignias del capó, siempre será ridículo ver a un hombre acomodado en el centro de una tonelada de fierro autopropulsado, controlando el ruidoso artefacto con ayuda de manubrios y palancas, detenido entre cientos de máquinas similares, esperando la señal luminosa que lo autorice a seguir.

No se salva ningún vehículo.
Todos ridículos.

Ridículo ir de pasajero en un taxi conducido por un desconocido, indigno para el chofer, indigno para el pasajero. Ridícula la micro, cuarenta cabezas saltando al unísono. Ni hablar de esa huincha transportadora de mano de obra que es el metro, que regurgita ciudadanos recién lavados a los pies de sus lugares de trabajo y los devuelve por la noche a sus casas, sucios y arrugados.

Mientras dura el viaje, se les intenta exprimir una última gota, avisos les ofrecen detergente, quitacallos o tumbas en el cementerio. ¿Cómo transportarse?

Porque no será la bicicleta la que devuelva al hombre su dignidad.

Ridícula la bicicleta, con su ser humano en equilibrio, pedaleando sobre un ingenio de poleas y engranajes. Peor si luce un coquete gorrito aerodinámico de polipropileno.

Desplazarse será siempre un acto ridículo. Excepto caminar. El único digno, el caminante que se desplaza con las piernas que Dios le dio.

El pobre caminante amenazado, cada año le quitan otro metro de vereda.

“¡Cómo se te ocurre atravesar aquí, cuando a sólo cinco cuadras te hemos construido un flamante túnel subterráneo!”

Peatón lo llaman, le gritan por el camino.

1 MARCO TEÓRICO

conjunto teórico y técnico de reflexiones, sistemas de orden, y buenas o malas tendencias sobre el transporte

1. CIUDAD Y TRANSPORTE

ORIGEN Y EVOLUCIÓN DE LA CIUDAD COMO SISTEMA COMPLEJO

La ciudad se originó cuando nos volvimos sedentarios; dominábamos el ganado y la agricultura y esta a su vez nos dominaba a nosotros con sus procesos cíclicos. Surgen tiempos de ocio y reflexión donde el hombre finalmente deposita sus intereses. Aparece el mercado, los templos, observatorios, entre otros espacios de recreación y encuentro.

Las personas comienzan a ir de manera cada vez más masiva a estos lugares, y van acercando también sus propias casas a estos lugares de interés; paulatinamente la migración del campo a los pueblos se va dando. Ahí comparten técnicas y conocimiento, intercambian bienes y se especializa el mercado, y lo que es más interesante, se van marcando las huellas por el paso de las personas. Así como las liebres repiten sus trayectos en los pastizales entre madrigueras: a medida que pasa el tiempo

y que se repiten las huellas nacen de manera casi fortuita los caminos de las personas. La primera red que nos une está hecha a partir de caminos de comercio. La ciudad es una red compuesta de nexos y nodos y los espacios determinados internos de estos. Los nexos son los caminos, los nodos las intersecciones, y los espacios dentro son aquellos habitables. Con el aumento gradual de la densidad de conexiones y nodos y el número de habitantes de pasamos de villa pueblo y de pueblo a ciudad y de ciudad a metrópolis. *La ciudad es un patrón en el tiempo. (John Holland)*

La ciudad se convirtió en red, y de red a sistema; un conjunto funcional que se regula a si mismo en base a un ordenado de reglas y relaciones y procedimientos. La ciudad contiene ya redes sobre redes. Sobre ella se despliega un conjunto de agentes que interactúan entre sí y con ella. Parecida a la de definición de ecosistema; “sistema biológico constituido por una comunidad de seres vivos y el medio natural en que viven” (Diccionario Oxford); la ciudad es un sistema dinámico relativamente autónomo formado por seres vivos que se relacionan sobre un medio físico y virtual, de índole natural y artificial. El hombre ha creado sobre esta; sistemas complejos de transporte, de comunica-

ción y de intercambio que hacen uso de las materias que nos entrega o arrancamos de la Tierra. Manteniendo de algún modo una conexión indirecta e insana con la naturaleza. (Richardson, 2017)

La inteligencia humana ha extendido sobre el mundo, en su entorno, su propia inteligencia. Hemos modificado nuestro espacio de habitar en base a criterios personales y sociales. El ecosistema urbano se va complejizando y especificando a medida que le damos más cualidades propias de la naturaleza inventiva humana. Pero el hombre, el único ser antropocéntrico, al fin y al cabo necesita a los recursos de la Tierra para poder sobrevivir. Es secundario modificar el entorno, adaptar el artificio racional que es una ciudad. Pero es solo el ecosistema de la naturaleza el único que posee procesos y complejidades tan sofisticadas como el ser humano, pero que se dan de una manera tan natural, valga la redundancia. El hombre intenta crear para la ciudad tecnología cada vez más funcional y compleja, siendo que esos sistemas ya están programados en el mismo código genético de la naturaleza, la cual logra hacer tareas con una ciencia sin precedentes.

EL COMPLEJO MUNDO INVENTADO DEL HOMBRE

La ciudad se compone de manera parecida a la Tierra: tiene un ecosistema biótico y otro abiótico, un conjunto vivo y otro inerte. Estos interactúan intercambiando energía activa o pasivamente a través de distintos medios. Es necesario aclarar que los objetos inertes, las cosas, a pesar de nos estar vivos si actúan como si la tuvieran.

Los objetos son nodos de relación y tan actores como las personas en la sociedad (Latour, 1980s). Ellos tienen capacidades de condicionar nuestro actuar; nos persuaden, nos explican, nos indican, nos manipulan, nos restringen. Nos movemos en la ciudad entre objetos, en objetos, por objetos. Nos comunicamos con y a través de ellos. Vivimos diariamente en una coreografía entre personas y objetos tan integrados que tiene una inercia imperceptible. La ciudad tiene elementos físicos definidos por el hombre, y estos a su vez nos definen.

Un trozo de metal con una forma particular cubierto de pigmentos rojos y nos encontramos frente a un objeto que nos obliga a detener nuestro auto y ser, a

ese lo llamamos disco pare. Los objetos nos modifican la naturaleza de nuestro andar generando nuevos patrones de comportamiento, porque elegimos por medio de consenso que así sea nuestra relación con ello.

Uno se pregunta, ¿será necesario? ¿Será necesario tanto artefacto reparado por la ciudad? Tanto semáforo, tanto ceda el paso, tanto lomo de toro, preferencia antes de doblar, prohibido doblar, prohibido camiones, prohibido doblar a la izquierda de 17 a 22hrs.

Nos creemos el animal con más capacidad y posibilidad de creación pero somos también el con más restricciones, y estas son lamentablemente autoimpuestas. ¿Qué clase de extraño simio decide inventar reglas para convivir sin libertad?



Signo pare internacional (HomeDepot, s.f.)

LA COMPLEJIDAD DEL PROBLEMA DEL TRANSPORTE URBANO

Nos movemos o transportamos, por la ciudad de formas muy extrañas, y para satisfacer necesidades muy humanas; y con humanas se refiere a poco animales. Viajo a pie para comprar las noticias, en bicicleta para pagar una deuda, en autobús para ver un amigo en el hospital, en auto para trabajar, o incluso viajo en avión miles y miles de kilómetros para solo obligarme descansar. La complejidad humana se ve reflejada en nuestras complejas y modernas necesidades. Si ya no tenemos el sentido de sobrevivir para vivir, solo sobrevivimos si buscamos otro sentido para vivir.

La realidad humana se ha acomplexado con el paso tiempo, así también la forma en la que nos movemos por las calles. La aceleración del desarrollo de la tecnología va de la mano con la forma en la que la humanidad crece por la información que va con ella. Al principio las personas y la información se demoraban meses en viajar de una gran ciudad a otra, con la ayuda de la fuerza animal de los caballo por ejemplo el tiempo pasó a ser semanas. Con la aparición de las carretas se redujo la

velocidad del transporte a la mitad. Y cuando estas evolucionaron a los autos viajar de una ciudad a otra toma solo unas horas o menos.

El transporte afecta toda nuestra vida. Uno crearía que un auto tiene varios componentes dentro de su sistema mecánico, pero estos no se comparan con la compleja red de influencia que despliega entorno al transporte en sí. El transporte es soporte de la economía: permite el transporte de información, de personas, de bienes, de productos; compramos en el extranjero, vendemos entre regiones del país, hay servicios de transporte a domicilio, si la pizza no llega en 30 minutos es gratis. El transporte afecta en los social: la posibilidad de transportarnos a mayor velocidad permite que nuestras necesidades queden a unos cuantos kilómetros pero a poco tiempo de distancia; hay familias viven separadas por cientos de kilómetros, hay quienes estudian a decenas de kilómetros de su casa, y nuestros alimentos o los servicios de salud quedan a algunos pocos. El transporte nos acerca temporalmente las cosas que nos quedan lejos y eso permite que podamos y tengamos que viajar gran parte de nuestras vidas de un lugar a otros, aun cuando estemos en la misma ciudad que aquello que busca-

mos. Ahora bien, aun que nos permita conectarnos con la gente que está a distancia, esto no significa que nos permita conectarnos más con quienes viven más cerca nuestro. Es más, sucede casi lo contrario, estudios sugieren que mientras más rápido se mueven los autos por las calles, menor va a ser la posibilidad de interacción entre las personas que circundan juntos por las calles. En al aún momento los vecinos que vivían separados por la calles Kennedy se conocían posiblemente, hoy con la moderna ampliación de 3 pistas por lado es casi imposible que aquellos vecinos se conozcan. El transporte afecta el medioambiental y la salud. Moverse por la ciudad por un modo motorizado significa contaminar con gases y con ruidos. Afectan la capa de ozono, la temperatura del globo terrestre, toda la biósfera. El mal aire afecta directamente nuestros corazones, vías respiratorias, pulmones, cerebro y gestación; respectivamente nos aumenta la presión sanguínea y da problemas cardiovasculares, se ve perjudicado nuestro sistema inmune y la irritación de las vías, nos da bronquitis y neumonía y cáncer de pulmón, el desarrollo cognitiva se ve afectado, y también el nacimiento bajo peso, prematuro y el aborto natural (UNICEF, 2016) (Union of Concerned Scientists, 2014).

Cada modo de transporte afecta toda nuestra vida de varias formas y de manera directa. Cada vez que decidimos andar en bicicleta estamos respirando aire contaminado por los autos, pero estamos ahorrando 1 tonelada de CO₂ anual que perjudica en la vida propia y la de los demás. (Ministerio del Medio Ambiente, 2016) ¿Qué preferimos, andar en bicicleta de la forma más económica, rápida, sana y que protege nuestro medio ambiente, o esperar sentados en un caro auto, engordando mientras destruimos nuestro ecosistema y el de los demás? La mitad de los viajes que se realizan en auto se realizan en auto no superan los 4 kilómetros, lo que es absolutamente cicleable. (EOD 2015) Cada decisión que tomamos de como movernos por la ciudad repercute en la salud, economía, medio ambiente y en lo social tanto a nivel particular como el de sociedad. ¿Cómo es posible que casi toda la infraestructura de la ciudad esta diseñada para fomentar el uso de los autos si estos nos traen infinitamente más desventajas? Andar en bus es 7 veces más seguro que en auto (BusAndCoach, s.f.) En bicicleta se gasta 35 veces menos energía para mover una persona. Una bicicleta usa unos justos y suficientes 1,25 metros cuadrados de la ciudad, los autos usan casi 14 metros cuadrados para trasladar a la misma

persona.(Allen, 2008) Da para pensar que el sobreuso de recursos, como lo es el espacio, esta lejos de promover la convivencia de una sociedad en especial una que se caracteriza por ser masiva. Cada modo de transporte está circunscrito dentro de la compleja red de relaciones que afectan directamente la vida urbana.

LA COMPLEJIDAD DE LOS SISTEMAS DE SIMULACIÓN

Dada la profunda cantidad de variables a la que se ve sometido el transporte estos se ven limitados a editar lo que van a simular. No se puede simular la realidad por que el costo computacional es inmenso, mas simular ciertos aspectos de la realidad si es posible, y se hace. Hay diversos softwares encargados de simular acontecimientos, trozos de ciudad, entre otros, para ayudar revelar, analizar y concluir información útil a la hora de tomar decisiones y planificar el entorno urbano.

El problema está en que estos programas se quedan en simular partes de la “realidad”, pero sin lograr proyectar ideas nuevas; no se da lugar para proponer, o en informar, en discutir, o en criticar; solo simular. No llevan un ideal de fondo, no se adelantan a su época si no que intentan resolver con la herramientas actuales problemas de planificaciones futuras. Es intentar solucionar las cosas con más de lo mismo. Cómo es posible que se sigan creando más calles y puentes para descongestionar una zona, esto sólo le permite a la industria automotriz crecer.

A continuación se presentan algunas de las funciones principales que desempeñan, los simuladores más reconocidos en el rubro, para graficar un poco el enfoque de estos mismos.

CORSIM (Corridor Simulator) Enfocado en examinar operaciones del control de tráfico y la optimización de señalética. Además permite la interfaz analizar en tiempo real el tráfico adaptable a al control de operaciones. (Perrin et al. , 2002).

SIMTRAFFIC tiene gran potencial a la hora de simular la fluencia, flujo, , seguridad de las rotondas, para luego comparar datos con lo que fueran inter-

secciones reales. El simulador SimTraffic se puede usar para generar tráfico y compararlo con los datos computados de choques de años pasados. (Drummond et al. , 2002).

AIMSUN (Advanced Interactive Microscopic Simulator for Urban and Non-Urban Networks) se especializa en los algoritmo de seguimiento de las líneas de tránsito y el algoritmo de cambio de pistas. Así también este software tiene un tráfico con la capacidad de responder al control de señalética y el poder cambiar prioridades entre vehículos. Según Barcelo (1999) el programa finalmente permite modelar el manejo de incidentes y estrategias de tráfico.

The screenshot shows the CORSIM software interface. On the left, there's a 'PRIMARY INPUT DATA' section with fields for 'Urban Street' (Speedway Boulevard), 'Intersection' (17th Street), 'Description' (No-Build Alternative), 'Data File' (Chapter17Automobile.xls), 'Forward Direction' (EB), 'Area Type' (Other), 'Segment Length' (1800), 'Duration' (0.25), and 'PHF' (0.52). In the center is a diagram of the intersection. On the right, there's a 'Phasing' section with 'Cycle' (100), 'Offset' (0), 'Phase 2 Direction' (EB), 'Phase 4 Direction' (SB), 'Reference Phase' (2), 'Reference Point' (End), 'Force Mode' (Fixed), and 'Allow Optimization' (checked). Below these are two tables: 'Traffic' and 'Timing'.

	EBL	EBT	EBR	WBL	WBT	WBR	NBL	NBT	NBR	SBL	SBT	SBR
Demand, veh/h	200	1000	10	200	1000	10	100	500	50	100	500	50
Lane Width, ft	12.0	12.0	12.0	12.0	12.0	12.0	12.0	12.0	12.0	12.0	12.0	12.0
Storage Length, ft	200	0	200	200	0	200	200	0	1000	200	0	1000
Saturation, pc/h/ln	1800	1800	1800	1800	1800	1800	1800	1800	1900	1800	1800	1900
Heavy Vehicles, %	0	0	0	0	0	0	0	0	0	0	0	0
Grade, %	0	0	0	0	0	0	0	0	0	0	0	0
Buses, per/h	0	0	0	0	0	0	0	0	0	0	0	0
Parking, per/h	0	N	0	0	N	0	0	N	0	0	N	0
Bicycles, per/h	0	0	0	0	0	0	0	0	0	0	0	0
Pedestrians, per/h	0	0	0	0	0	0	0	0	0	0	0	0
Arrival Type	3	4	3	3	4	3	3	3	3	3	3	3
Upstream Filtering (I)	I-EB	0.77	I-WB	1.00	I-NB	1.00	I-SB	1.00				
Initial Queue, veh	0	0	0	0	0	0	0	0	0	0	0	0

	EBL	EBT	WBL	WBT	NBL	NBT	SBL	SBT
Phase Split, s	20.0	35.0	20.0	35.0	20.0	25.0	20.0	25.0
Yellow Change, s	3.0	4.0	3.0	4.0	3.0	4.0	3.0	4.0
Red Clearance, s	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Minimum Green, s	5	5	5	5	5	5	5	5
Lag Phase	<input type="checkbox"/> EL	<input type="checkbox"/> ET	<input type="checkbox"/> WL	<input type="checkbox"/> WT	<input type="checkbox"/> NL	<input type="checkbox"/> NT	<input type="checkbox"/> SL	<input type="checkbox"/> ST
Passage Time, s	2.0	0.0	2.0	0.0	2.0	2.0	2.0	2.0

CORSIM posee gran variedad de variables a modificar para hacer una buena simulación, pero no para poder comunicar una nueva idea o sistema de transporte (mctrans.ce.ufl.edu, s.f.)

ESCENARIOS FUTUROS Y PROYECCIONES DE TRANSPORTE

El futuro estará dirigido por la sustentabilidad: por la adecuada y responsable extracción de materias primas, el lógico reciclaje, la proporcionada producción y el medido consumo. En vez de producir autos, o buses, o metro trenes hiperveloces se valorizará más la calidad de viaje, la consistencia y la estabilidad y la experiencia y el bienestar de las personas.

Como propone el Gobierno Australia para el año 2056 en su proyecto de nuevo sistema de transporte: *“We’re moving away from a simple view of transport as vehicles, infrastructure and services to seamless customer experiences”* (Australian Gov., 2016)

Existirá una verdadera sostenibilidad ecológica entre el hombre y la tecnología y la naturaleza que dará paso a una nueva era en la Tierra. Esta simbiosis será probablemente producto de una biomimesis: las tecnologías aprovecharán las ciencias y mecánicas ya presentes en la naturaleza. El hombre dejará de competir con la naturaleza y comenzará a aprovecharla haciendo

uso de nuevas tecnologías y nuevas formas de organizar la forma en que nos movemos. *“Multi-modal transportation will be prioritised upside down and enriched by new devices. Having all the tools and devices in place, it’s all going to be about connectivity”*. (Arje Van Berkum, 2017)

Surgirá una nueva ciudad con capacidades que favorecerán al ecosistema humano-natural. Esta ciudad inteligente tomará decisiones en conjunto con el hombre, configurada desde su core funcionará de manera viable, vivible y equitativa para acercarnos más al desarrollo sustentable. El nuevo enfoque de la sociedad le dará un giro a la vida humana que nos llevará hacia el verdadero progreso. Lo sano para nosotros será por fin entendido como lo ecológico para la Tierra.

En el futuro venidero el transporte será para la humanidad algo sin precedentes de nuestra historia; pues su fluidez, dinámica y lógica, dada su propia sustentabilidad, será digna de compararse con los sistemas de la naturaleza. El hombre vivirá en movimiento tanto al igual como lo hace en su hogar. Las familias tomarán desayuno mientras los autónomos cocinan dentro de una unidad también autónoma de transporte. Las calles y veredas serán una sola super-

ficie habitable donde autos, peatones, ciclistas y el transporte público conviven de forma armoniosa, respetándose y coordinándose de manera connatural. La ciencia estará medida bajo el prisma del bienestar social, el de la experiencia. El éxito se entenderá como una terna entre lo social, lo económico y lo ambiental.



El concepto de la ciudad bosque en Liuzhou China, por la firma Stefano Boeri Architetti (CNN Style, 2017)

2. SISTEMAS EMERGENTES

“El problema original de la descodificación: cómo pueden construirse patrones complejos a partir de reglas simples: ¿Cómo sabe una semilla construir una flor?” (Steven Johnson, 2003, p.40)

a. Del orden aparece el caos. “En estos sistemas, los agentes que residen en una escala comienzan a producir comportamientos que yacen en una escala superior a la suya: las hormigas crean colonias, los habitantes de una ciudad crean barrios, un software de reconocimiento de patrón simple aprende a recomendar libros. La evolución de reglas simples a complejas es lo que llamamos ‘emergencia’”. (Steven Johnson, 2003, p.19)

b. Alan Turing lo describe como la capacidad de todas las formas de vida de desarrollar cuerpos cada vez más complejos a partir de orígenes increíblemente simples.

c. “El comportamiento separado, individual, de cada uno de los agentes, al aumentar la escala comienza a producir un comportamiento colectivo propio

de un nivel de organización superior a pesar de la aparente carencia de organización en forma de leyes o instrucciones provenientes de una autoridad superior”. (Wikipedia, 2017)

d. *“Lo que une a estos distintos fenómenos es una misma forma y patrón: una red de autoorganización, de agentes dispares que crean un orden de un nivel superior sin proponérselo”. (Steven Johnson, 2003, p.22)*

La fuerza hace la unión. En el caso de las hormigas el estar organizadas de forma distribuida les permite generar sistemas que en su totalidad son complejos.

e. “Local parece ser el término clave para comprender el poder de la lógica del enjambre. Vemos conductas emergentes en sistemas como las colonias

de hormigas cuando los agentes individuales del sistema prestan atención a sus vecinos inmediatos y no esperan órdenes de arriba. Piensan localmente y actúan localmente, pero su acción colectiva produce comportamiento global”. (Steven Johnson, 2003, p.68)

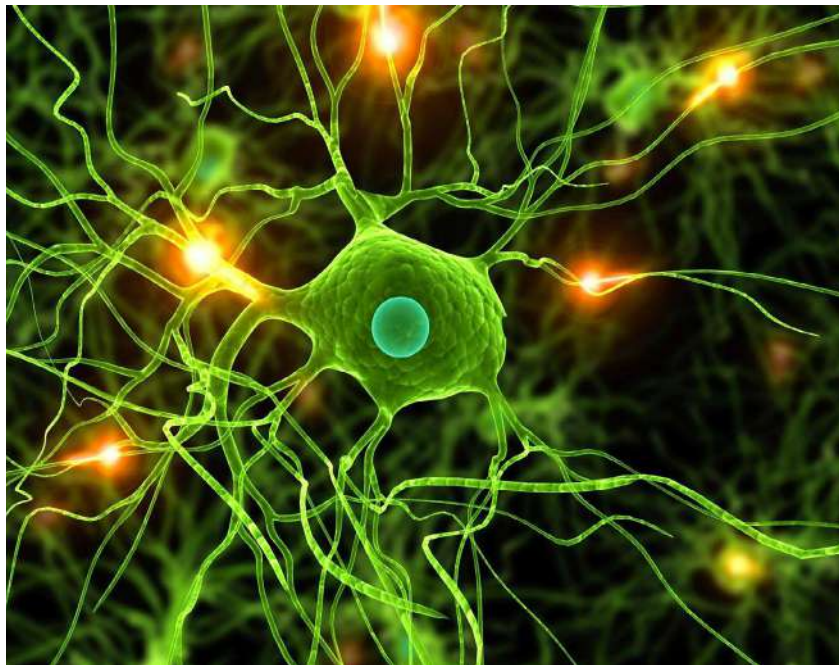
f. Los sistemas emergentes están distribuidos, descentralizados, se autorganizan, son escalables y a nivel de agentes son poco complejos.

The work on ants has profoundly affected the way I think about humans. E.O.Wilson

Our challenge for the future is that we realize we are very much a part of the earth's ecosystem, and we must learn to respect and live according to the basic biological laws of nature. Jim Fowler



Hormigas fuego formando un puente es un ejemplo de lo poderoso que es un sistema emergente (A. Bockoven)



Las neuronas son un sistema emergente que ocurre dentro nuestro (2.bp.blogspot, s.f.)



Los cardúmenes son un claro ejemplo de comportamiento emergente (dossiergf, 2012)



Las bandadas se mueven en conjunto gracias a la emergencia (marktlederschap, s.f.)



Las catedrales de termitas las construyen gracias a la emergencia (tendencias21, s.f.)

3. TENDENCIAS

MALAS COSTUMBRES Y TENDENCIAS NEGATIVAS

“La ciudad es un patrón en el tiempo”. (John Holland) La ciudad nace a partir de nuestras huellas de forma natural, ¿por qué no movernos a través de ella de la misma manera? La ciudad es un ecosistema orgánico lleno de mecanismos naturales con una eficiencia que supera toda invención de la humanidad. Como dijo *Manfredi Nicoletti*, “*La naturaleza no debe ser imitada, debe ser entendida*”. Nuestro error está en intentar regular más y más las cosas sin intentar dejar que fluyan por su propio curso. Agregamos señalética, separamos la calzada de la vereda, ponemos costosos semáforos en cada esquina y todo para frenar y encausar lo que se debiese estar moviendo, hablando tanto de autos como peatones. La industria automotriz junto a los gobiernos, movidos equivocadamente por la economía, por décadas han promovido la producción, venta y uso del auto, el símbolo del progreso americano, quien en realidad es absolutamente el modo menos sustentable.



(CarScoops, s.f.)

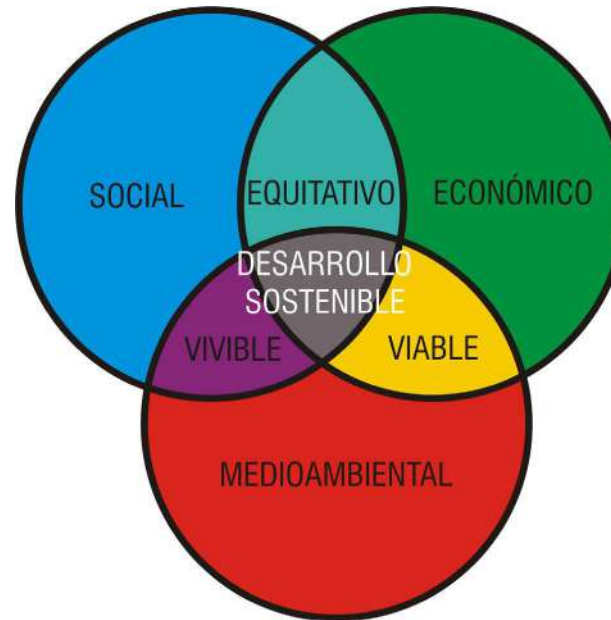
“El derecho a moverse por la ciudad es universal y no debiese reservarse a propietarios de automotores privados”. (ITPD, 2013) A estos últimos cada vez les entregamos más poder, espacio y beneficios. Entre todas las calles y estacionamientos el 60% de la ciudad está cubierto por asfalto dedicado para los autos. Nos traen congestión, contaminación, retención de altas temperatura, menos espacios de recreación; exactamente lo contrario sucede con las áreas verdes, que son un mísero 1 a 2% de la superficie de la ciudad. Las áreas verdes mejoran la calidad del aire, regulan el clima (en especial los árboles), mejoran el agua y tierra, son espacios útiles para eventos, recreación y fomentan el deporte. (Project EverGreen, 2017) Y para que hablar del tiempo que nos consume el andar en los autos, al día se anda en promedio 2 horas en auto, 5 veces a la semana, es decir un total de 40hrs al mes en tiem-

po perdido que pudo ser otra semana de trabajo o 3 días completos para estar con la familia. El ITPD (El instituto de transporte planificación y desarrollo de México) propone revertir las prioridades de los proyectos y la distribución de las calles de la ciudad privilegiando a los más sustentables por sobre los menos, que son a su vez los más indeseables por la sociedad.

I think that technology is the best thing that ever happened to mankind. It's an absurd notion that somehow, 'My God, what are we going to do when driverless cars come along?' It's going to save lives on the road. And maybe, one day, we'll all be working four days a week and not five or six days a week.
Jamie Dimon

Hay que revertir la situación actual, cambiar de la sociedad del consumo y desenfreno por la del equilibrio y la sustentabilidad. El desarrollo sustentable es un enfoque tridimensional: es la mezcla de lo económico, social y ambiental; lo justo, lo viable y lo vivible. (Peter Njikamp, sf)

“Un proceso es sostenible cuando ha desarrollado la capacidad para producir indefinidamente a un ritmo en el cual no agota los recursos que utiliza y que necesita para funcionar y no produce más contaminantes de los que puede absorber su entorno”. (Universidad Abierta Interamericana, & M. Calvente, Ing., 2007)



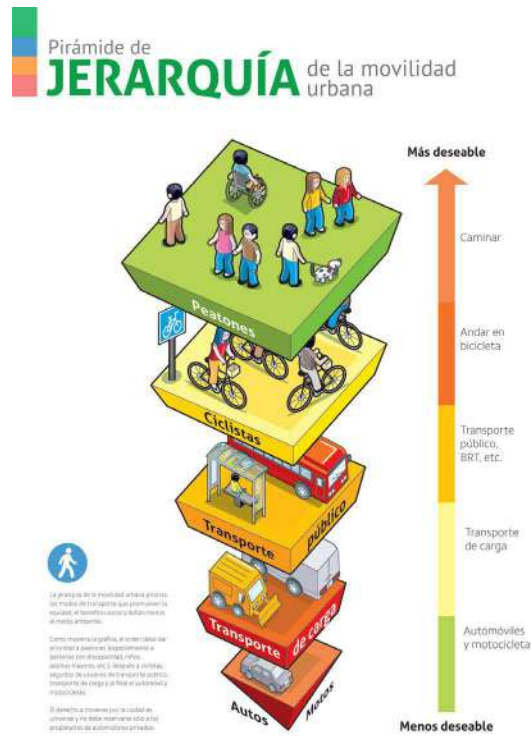
(DesarrolloSustentable.Co, 2013)

BUENAS PRÁCTICAS Y TENDENCIAS POSITIVAS

1. FUTURO SUSTENTABLE Y EQUITATIVO

a. Los derechos de movilidad en la ciudad son iguales para todos, se habla de un ciclista, peatón, conductor o chofer. El Instituto de Políticas para el Transporte y el Desarrollo propone una pirámide de jerarquía de la movilidad (figura 1) urbana según cuan deseable es cada modo de transporte según su impacto al medio ambiente, el beneficio social y cuanto promueven la equidad. *“El orden debe dar prioridad a los peatones (en especial a personas con discapacidad, niños, adultos mayores, etc.), después a ciclistas, seguidos de usuarios de transporte público, transporte de carga y al final al automóvil y motos.” (ITPD, 2013)* Este sistema de organización y jerarquización también se encuentra alrededor del mundo; desde los planes de infraestructura de Santiago de Chile hasta en el Bicycle Innovation Lab de Copenhagen por mencionar un par. (Arjen Van Berkum, 2017)

b. Ciudades fomentan uso de bici y de los modos de movilidad más sustentables. “De acuerdo a lo que señalan los



Ordenados según deseabilidad (ITPD, 2013)

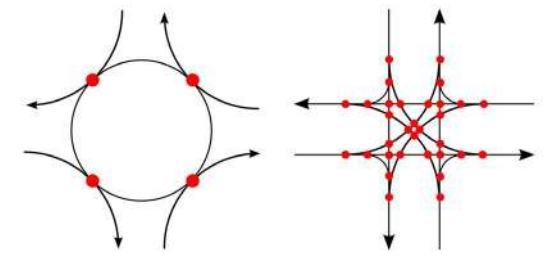


Santiago fomenta la movilidad verde (The Note, 2016)

expertos, Santiago se impuso frente a las otras ciudades gracias al mejoramiento integral para peatones, [...] la bicicleta y medios no motorizados y de medio motorizados con energía eléctrica”. (figura 2) (TheNote, 2016)

“Los cambios irán acompañados con medidas de movilidad que priorizan el transporte público y otras que fomentan el uso de la bicicleta a partir de la construcción de 300 kilómetros de ciclovías y tres mil estaciones públicas para bicicletas”. dijo Cristián Bowen (Ministerio del Medio Ambiente, 2016).

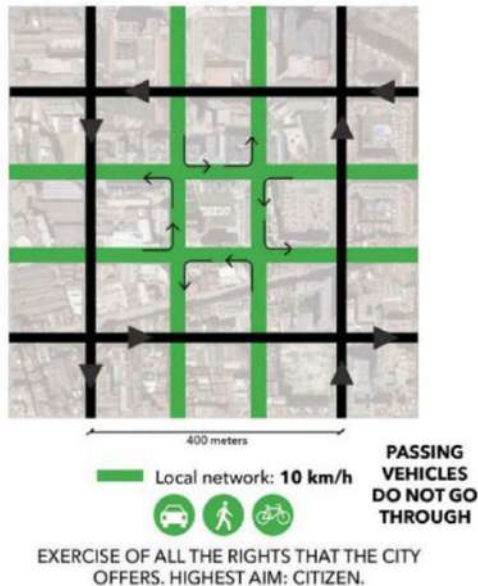
c. En varios lados de Europa se están eliminando intersecciones con semáforos y señalética para crear espacio compartidos entre autos, bicis, buses y peatones. El concepto es que la ausencia



Las rotondas permiten no solo reducir de forma efectiva la velocidad de los autos, si no también la cantidad de puntos de encuentros y de riesgo son menores. (Actam, 2008)

de separación mantendrá a todos más alerta, entonces todos irán más despacio, harán contacto visual y negociarán. En gran medida funcionan como funcionan las rotondas, pero sin las rotondas. (99% Invisible, 2017)

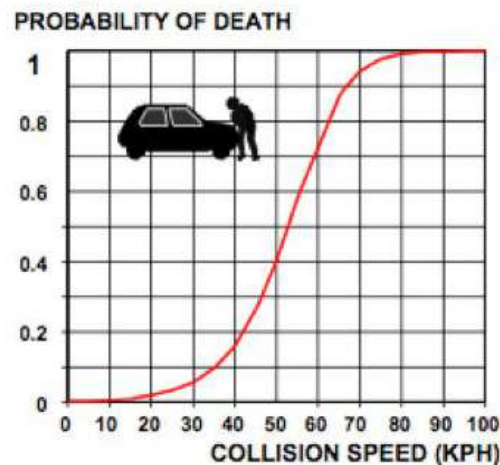
SUPERBLOCK



Supermanzanas en barcelona (99% Invisible, 2017)
En Barcelona se están creando las SuperManzanas, conjuntos de 3x3 bloques de edificios entre los cuales la velocidad máxima es de 10kph, permitiendo también que se compartan los espacios comunes. (99% Invisible, 2017)

En Chile, y particularmente en Santiago, la idea de las zonas 30 o las zonas pacíficas ha sido impulsada por dos

municipios, el de La Florida y el de Providencia. En cada una de ellas se escogieron ciertos tramos en los que se instalarán resaltos y señalética para lograr disminuir la velocidad de los autos y que más peatones y ciclistas transiten por el sector. (PlataformaUrbana.cl, 2014) En un impacto vehículo peatón promedio a 32kph el 5% de lo peatones muere, si fuese a 48kphr el 45% y a 64 kph el 85% . (Literature Review on



Razón muerte/velocidad (MountRoyalStation, 2013)

Vehicle Travle Speed and Pedestrian Injuries, 1999) Esta clase de datos debiesen ser el un jingle de las escuelas de manejo. ¿Cómo es posible que sea tan común que la gente maneje a 70kph en calles residenciales?

2. MEJORA DE TECNOLOGÍAS AUTÓNOMAS Y EL USO DE ENERGÍAS RENOVABLES

a. Las inversiones en el mundo automotriz autónomos crece vertiginosamente. Nos encontramos con toda la flota de prueba de Uber, o los testeos en California de Waymo que es de Google, o el accesorio para automatizar autos de Apple o el caso de nuTonomy en Singapur, que tiene una flota de taxis autónomos que ya están funcionando por completo. En Londres se presentaron los primeros proyectos de autos autónomos llamados Harry. En Alemania completa se legalizaron los autos autónomos y cuentan con un bus autónomo licenciado para manejar por carreteras y calles, el "Spirit of Berlin". Y esto es una buena señal, porque los autos autónomos manejan mucho mejor que los humanos. El 81% de los accidentes de autos son por error humano (Pony Parts, 2014). La superioridad de



(The Verge, 2016) Harry el podautónomo de Londres



(The News Wheel, 2017) Taxi autónomo de Singapur



(Mashable, 2016) El Bus autónomo de Berlín



(ABCNews, 2015) Ciega con auto autónomo Waymo

las computadoras reside en la capacidad de reacción, precisión y constancia. Los choques son el 2.2% de las muertes anuales del mundo. El 2013 solamente en Chile murieron 2179 personas en accidentes de tránsito (WHO, 2015). Otra tendencia positiva es el crecimiento del mercado de energías renovables y sostenibles. Noruega por ejemplo logró que el 25% de sus ventas de autos fueran eléctricos el año 2015. (Unicef, 2016, p.77) Holanda anunció un proyecto que eliminará los automóviles diésel y a gas para el 2025. (The Note, 2016) Volvo dejará de invertir en motores diésel, y se comprometió a convertirse en una empresa de únicamente autos eléctricos.

b. Están apareciendo también tecnologías más eficientes de almacenamiento y de carga rápida de energía. La Unión Europea está desarrollando el proyecto Zeus que promueve la movilidad eléctrica urbana, ha subvencionado 60 buses y cargadores eléctricos que cargan el 80% de las baterías de buses en 5 minutos. (The Note, 2016) En México se invirtió en la compra de 100 taxis híbrido y planean hacer un cambio total de los taxis de la capital en un período máximo de 15 años. (The Note, 2017)

c. La Fórmula E correrá en las calles de Santiago pasando por Plaza Italia el 3 de Febrero del 2018. (FIAformulaE, 2017) Para fomentar las fuentes de energía renovables la federación internacional de la fórmula E está generando campañas del deporte y asociándose con gobiernos para promocionar más este deporte sustentable.



(CodexVerde, 2017) Fórmula E llegará el 2018

d. Hay mejoras en dispositivos de automatismo y reconocimiento. No solo en los autos autónomos, si no en las cámaras actuales se pueden instalar *softwares* para que de forma automatizada se hagan mediciones en los peatones en intersecciones, para finalmente mejorar la seguridad y la movilidad todo esto sin requerir de la instalación de nuevos equipos. (Business Aire, 2016)

REFLEXIÓN Y CONCLUSIÓN

Falta un sistema de naturaleza emergente capaz de permitir el desarrollo sustentable y orgánico del ecosistema de transporte independiente de la infraestructura urbana. Uno que soporte un flujo de agentes en la ciudad respetando siempre todos los deseos de la sociedad por promover la equidad y la democratización de los modos de transporte. También debe integrar las virtudes de los autónomos como lo son la programabilidad y el uso de energías renovables.

Las tendencias actuales y los sistemas emergentes nos entregan varios atributos que llaman a ser explotados por el bien de la sociedad. Entre todas aquellas bondades se destacan las siguientes como las más relevantes:

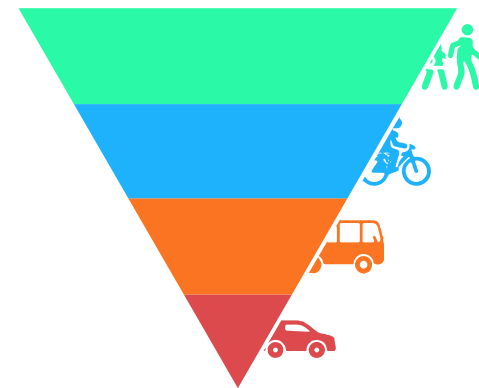
1. PRIORIDAD POR DESEABILIDAD

Hay una implícita referencia de unos modos por sobre otros según deseabilidad por beneficio social, equidad e impacto ambiental. Existe una tendencia por revertir la situación actual donde no se respeta este orden tácito. Se dará un reordenamiento proporcionado entre la sociedad y objetos y recursos que nos llevará a la reestructuración y

edición de la de la ciudad a nivel infraestructural. La oportunidad está en crear un sistema con orden jerarquizado basado en una preferencia consensuada y explícita que nace a partir de la deseabilidad de diferentes modos de transporte.

2. SURGIMIENTO DE UNA CULTURA SUSTENTABLE

El enfoque simultáneo y triple sobre la economía, la sociedad y el medioambiente será la norma. Nuevos modelos de negocios consideran el capital social y medioambiental por sobre el únicamente económico. Se buscará un equilibrio y proporción que estará inmerso en la esencia de los proyectos ciudadanos. Considerando el futuro



Pirámide de Jerarquía de movilidad (ITPD, 2013)

sustentable se pueden asumir la favorabilidad que tendrían los proyectos de la misma calidad social, económica y medioambiental. La naturaleza del nuevo proyecto debiese sostenerse en la integración de los nuevos sistemas actualizados e infraestructuras sustentables.

3. MEJORA AUTOMATISMO Y TECNOLOGÍAS DE RECONOCIMIENTO URBANO

Nuevas tecnologías darán paso a procesos más avanzados y mejores. Haremos uso de nuevas fuentes de energía y ya estarán adoptados nuevos sistemas de flujos computarizados que integraran a los vivos con los inertes. Quizás no nos moveremos más rápidos, pero si mejor. Se dará una automatización en la mayoría de los de procesos tecnológicos relacionados con el transporte y la energía.

Hay una gran oportunidad de acceso a fuentes de energía renovables y nuevas y más sofisticadas tecnologías. El proyecto en sí debe aprovechar la masificación de las nuevas capacidades técnicas y electrónicas del mundo del software y hardware.

4. ORDEN DISTRIBUIDO

En vez de haber una central o autoridad al mando todos los agentes tienen el mismo acceso a sistema, tienen propios roles y funciones y el poder se ve distribuido por medio de relaciones multidireccionales. El orden heterárquico permite la correcta integración de todos los actores involucrados, lo cual lleva a la unificación transversal y generalizado. Por medio de la programación física o tecnológica de cada individuo a nivel individual, con el uso de un mismo código “genético”; se podría lograr un sistema transversal adaptativo que resuelva de manera local los problemas de un sistema de mayor grado. Evitando así la necesidad de una central de comandos.

5. AUTORGANIZADO

Cada actor tiene influencia por sobre los otros. Estas se van dando por reglas de proxémica, por tipos de funciones y de diferentes configuraciones particulares de los agentes del ecosistema. Los grados de influencia funcionan sin importar la cantidad de número de actores. Todos estos factores llevan a la autorganización de un sistema de orden superior.

La capacidad de resolver problemas puede estar dada por las propiedades de cada individuo particular del transporte. Cada uno se relaciona con sus vecinos en base a situaciones locales y bajo el comportamiento emergente espontáneo. Autónomos saben cuan cerca y rápido deben moverse al lado de otros.

6. ESCALABILIDAD

Las diferentes repeticiones de las interacciones permiten el surgimiento de patrones y guras; de coreografías, que funcionan independiente de las escalas. Aparecen áreas de interacción de diferentes tamaños por movimiento y estática, y por presencia y ausencia. Estas zonas pueden establecerse premeditadamente o bien dejar que apareciesen de manera espontánea. Áreas o regiones de interacción pueden permitir diferentes espacios de interacción coincidentes con las jerarquías u ordenes programados.

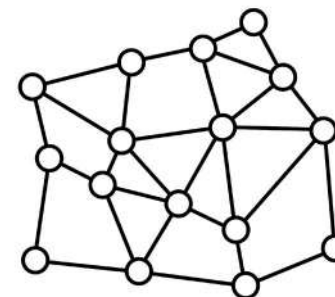


Diagrama de red distribuida. (Armstrong, 2017)

2 FORMULACIÓN

QUÉ

Propuesta de nuevo ecosistema emergente y distribuido de movilidad urbana para el futuro; a través de un simulador que ayuda en la toma de decisiones y en la planificación urbana enfocadas en la sustentabilidad.

POR QUÉ

El derecho a moverse por la ciudad es universal y no debe reservarse sólo a los propietarios de los autos¹: el modo menos económico, el de mayor impacto ambiental y menor beneficio social. La repartición de la superficie de la ciudad se aleja de la equidad y de fomentar los modos de movilidad más sustentables: el caminar y andar en bicicleta. Existe un entorno urbano en el cual no se logra la intermovilidad adecuada, ni sustentable ni tampoco segura. Entorno al transporte hay una desproporción en la producción, economía, impacto ambiental y el social.

No se están generando los suficientes cambios político o culturales para integrar las energías renovables o las de automatismo de lleno; ambas son necesidades indiscutible para nuestra sociedad.

1. (ITPD, 2013)

PARA QUÉ

Ayudar en la toma de decisiones y en la planificación urbana, reflexionar y especular con un especial enfoque en la sustentabilidad; para finalmente para mejorar el bienestar y la experiencia de las personas del futuro. El ecosistema de movilidad per se busca unificar y priorizar aquellos modos que promueven la equidad económica social y el beneficio ambiental.

CÓMO

Con el desarrollo de un programa interactivo digital que simule, explique e integre la esencia del ecosistema midiendo el impacto que estos tienen desde el prisma de la sustentabilidad.

Por medio de reglas de comportamiento se jerarquerizan los diferentes modos de locomoción los cuales actúan según cuán sustentables son para la ciudad. Los agentes móviles se identificarán entre sí de manera local para determinar cómo se mueven según al nivel que pertenecen ellos y sus 'vecinos' dentro la jerarquía. Resultando finalmente en una coreografía emergente de flujos y de quién le cede el paso a quien.

3 OBJETIVOS DEL PROYECTO

MICRO SIMULADOR DE FLUJOS, INTERACCIÓN Y DE MEDICIÓN DE IMPACTO

GENERAL

Ayudar en la toma de decisiones y en la planificación urbana.

ESPECÍFICOS

1. De un modo didáctico e intuitivo mostrar y medir el impacto en la ciudad en base a variables de datos reales.
2. Enfocado en la sustentabilidad y en la integración de las nuevas tendencias
3. Ser interactivo, controlable por un usuario en algún cierto grado.
4. Simular los flujos y encuentros de y entre los diferentes modos de movilidad urbana en un caso ciudadano real.
5. Orden emergente: distribuido, escalable y de naturaleza casi orgánica.

PROPUESTA DE UN FUTURO ECOSISTEMA DE TRÁNSITO EMERGENTE & DISTRIBUIDO

GENERAL

Mejorar la experiencia y el bienestar de las personas por medio de la integración distribuida, unificada y jerarquizada de los distintos modos de movilidad urbanos según su deseabilidad.

ESPECÍFICOS

1. Fomentar el uso de las tecnologías sustentables; aquellas que nutren lo social, económico y ambiental.
2. Redistribuir el espacio de la ciudad acorde a lo que la sociedad desea.
3. Mejorar la accesibilidad del transporte urbano al punto de que toda persona se sienta capaz de usar todo tipo de modo.
4. Aprovechar las tecnologías autónomas como soporte para crear un sistema emergente más orgánico, fluido, seguro y sostenible.

4 CONTEXTO

espacio y tiempo

DÓNDE

El ecosistema de transporte está basado en un orden emergente y distribuido; la jerarquización entre los agentes no requiere de una infraestructura para poder funcionar. El orden global aparece al haber varios conjuntos de agentes locales ordenados. Por esta misma razón la propuesta ecosistémica está pensada indistintamente de los elementos repartidos por la superficie de las ciudades: refiriéndose a semáforos, señalética, veredas, resaltos, calzada, entre otros. Sin embargo, a pesar de no requerir físicamente de una estructura, si tiene uno o varios lugares de aplicación: el simulador (o el ecosistema que emula) está pensado para ser principalmente usado por un funcionario de algún gobierno o municipalidad que esté encaminando sus decisiones hacia un futuro que incluya las tendencias del transporte. La propuesta se desenvolvería mejor en ciudades cuyo enfoque en fomentar radicalmente la sustentabilidad y el bienestar de las personas. Por ejemplo en Copenhagen, la ciudad más amigable para andar en bicicleta el año 2017, la jerarquía en pos de la sustentabilidad ya está dada por la infraestructura y la ley, lo único que falta es soportar tal jerarquía e independencia con un sistema orgánico y fluido.

CUÁNDO

Más importante que dónde puede ser implementado es cuándo. El programa para especular e investigar se necesita el día de hoy. Reflexionar sobre el cambio que se viene es una urgencia real actual. Mientras antes nos adaptemos según las tendencias tecnológicas y busquemos solucionar las problemáticas ambientales será mejor. La propuesta de transporte apunta a funcionar aproximadamente en 2 a 3 década en el futuro. Las tendencias ecológicas de los gobiernos por dejar de usar los combustibles fósiles, vender cada vez más autos con energías renovables, o por internalizar y fomentar las tecnologías autónomas es cada vez más común; es por esto que promover las conductas sustentables en la sociedad tiene que comenzar a ser un hábito lo antes posible.

La tecnología avanza más rápido de lo que la sociedad se puede adaptar, es por esto que tenemos que esforzarnos por dejar los malos hábitos de hoy, como imprimir 3 libros de 150 páginas que serán leídos una vez, y aprovechar las nuevas tecnologías para enfocarnos en vivir de un modo más sustentable.

5 USUARIOS

quiénes y para qué

FUNCIONARIOS DEL GOBIERNO Y MUNICIPALIDADES, URBANISTAS, INVESTIGADORES, EMPRESAS PRIVADAS, EMPLEADOS PÚBLICOS Y CIUDADANOS

Funcionarios del gobierno y/o municipalidades podrían ayudarse con el programa para medir la tasa de impacto de sustentabilidad, todo esto con el fin de facilitar la toma de decisiones y la planificación urbana orientada hacia el bienestar económico ambiental y social.

Profesionales encargados de proyectos y estudios a una mayor escala social podrían usar el programa para observar flujos, puntos de interés, o tendencias; para resolver, especular y proponer

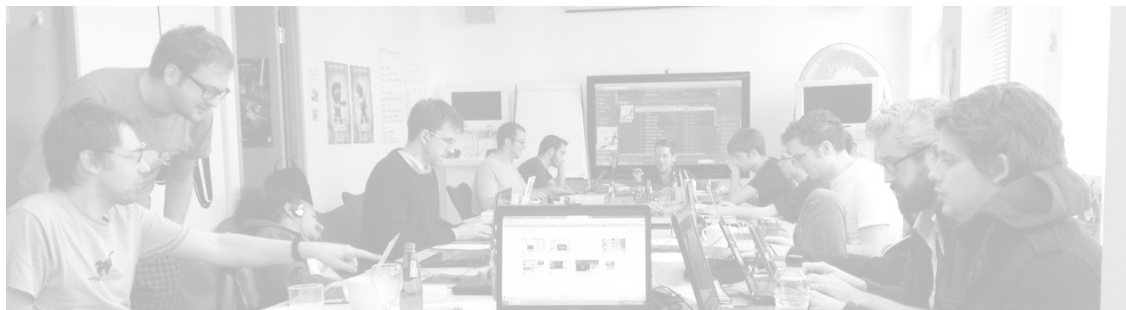
como nuevos ordenamientos de vehículo-persona, es decir, la redistribución de las calles.

De un modo parecido a de los urbanistas, los investigadores podrían usar la plataforma para comparar y analizar datos duros que se actualizan de forma interactiva y real, así como también para emular casos hipotéticos de una manera didáctica e interactiva.

Empresas privadas pueden usar los datos entregados por el programa interactivo como base para sus propias necesidades corporativas, como lo es probar sistemas de autos descentralizados y autónomos.

Ciudadanos comunes e interesadas en entender el impacto que puede tener un nuevo tipo de orden en su ciudad y en sus vidas; podrían usar el programa interactivo para entender un poco lo que depara el futuro.

Es necesario aclarar que dada la naturaleza del proyecto nos podemos encontrar varios tipos de destinatarios. Esto se debe a las distintas aristas por las que puede ser entendido y aprovechado el programa: como plataforma, por su mensaje, por solo la idea, como una visualización, entre otras. A continuación se encuentran los principales usuarios para quienes fue desarrollado esta el proyecto.

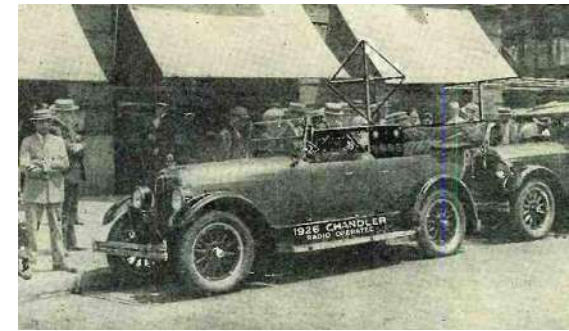


6 ANTECEDENTES LINRRICAN WONDER

Por Houdina Radio Control, año 1925 en Nueva York. Hizo demostraciones a través del tráfico de la 5ta avenida NY de su auto controlado de manera remota por un segundo auto que lo seguía. El auto era comandado bajo impulsos de radio que le enviaban a una antena de transmisión que tenía integrada a frenos, acelerador y volante. Todo movimiento lo hacía dirigido por aquellas ordenes electromagnéticas. (Time, 2017)

ZONAS 30

En México se están planificando nuevos tipos de infraestructura enfocadas en la pirámide invertida de movilidad. (ITPD, 2013) Tales estructuras promueven la equidad y la repartición proporcionada de espacio. Las Zonas 30 o zonas pacíficas, o los espacios urbanos compartidos por personas, bicicletas y conductores son cada vez más comunes. Hay en Chile, hay en Inglaterra, en Milán y España, por mencionar algunas. Estos le devuelven las calles a las personas, son más ecofriendly y seguros.



Linrrican wonder, el auto radiocontrolado (Wired, s.f.)

PARKSHUTTLE

The First Autonomous Car, Neatherland 1997. (Maurits Vink, 2005) El autobús de acercamiento usa puntos de referencia imantados para guiarse. Fue nombrado como el primer vehículo autónomo del mundo. Lleva más de una década funcionando y tienen gran número de reseñas positivas. (2Get ere, 2016) Usa puntos magnéticos de referencia incrustados en la superficie del camino para determinar su posición dentro de las pistas exclusivas. Se hicieron pilotos con público general en el Aeropuerto de Schipol 1997 y el parque.



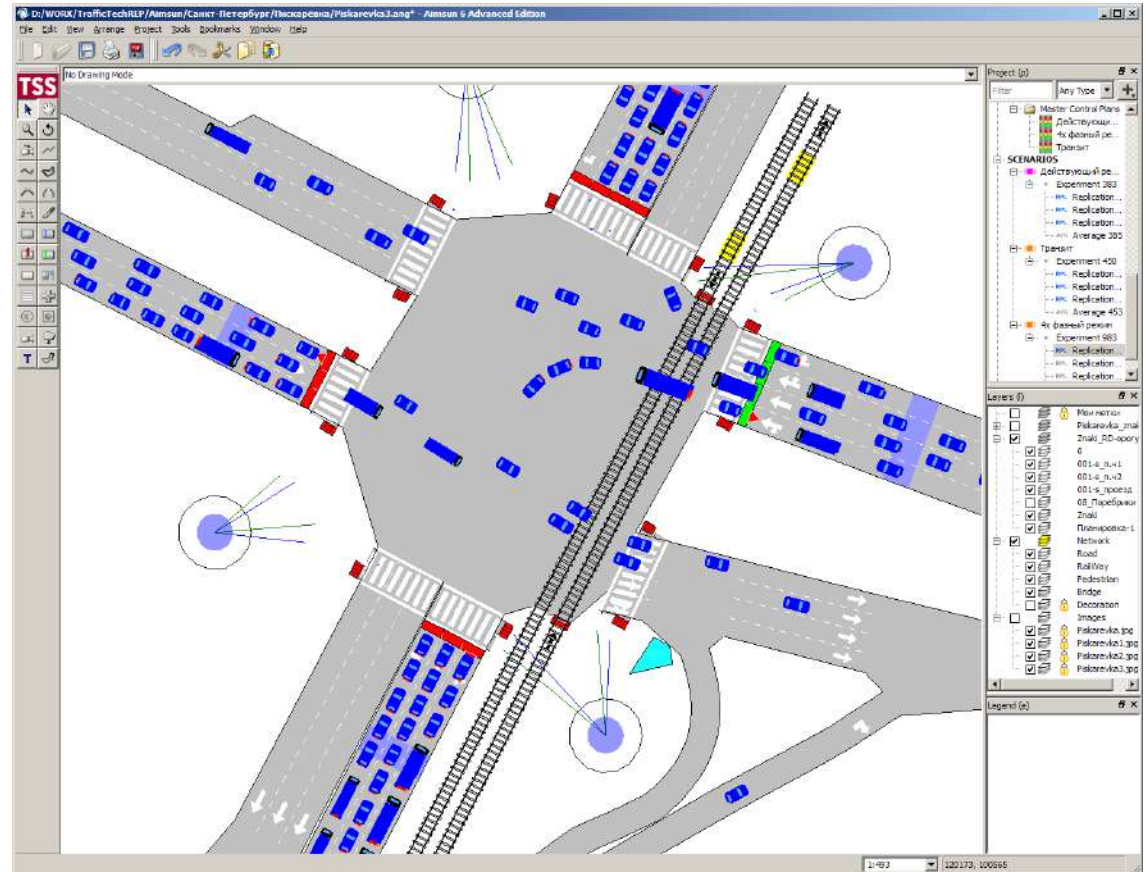
Décadas lleva en funcionamiento (Maurits Vink, 2005)

ALV DARPA

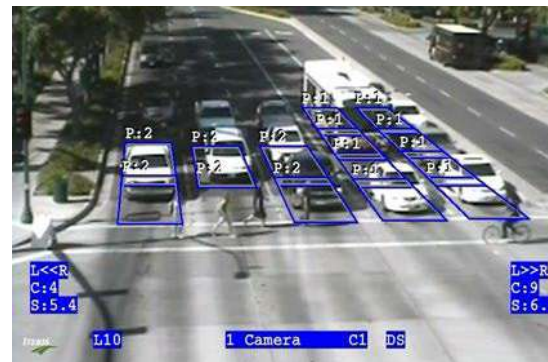
Autonomous Land Vehicle, DARPA, EEUU 1980. “The ALV project achieved the first road-following demonstration that used LIDAR, computer vision and autonomous robotic control to direct a robotic vehicle at speeds of up to 19 miles per hour (31 km/h).” Wikipedia 1980s (Wikipedia, 2017) Se pueden usar guías visuales o digitales sobre los agentes de móviles para guiar un robot con sensores LIDAR.

PEDTRAX

Iteris desarrollo una tecnología de reconocimiento de peatones que sería posible integrar la en todo auto autónomo para mejorar la seguridad vial. Es un software capaz de ser integrado en cámaras bastante básicas sin la necesidad de instalar dispositivos nuevos. (Business Wire, 2016) Mejorar la identificación entre todo elemento móvil podría ser la clave para soportar una red de interacción remota. En el caso de que existiesen chips de reconocimiento en cada cédula de indentidad, por dar un ejemplo, cada persona podría estar conectada de manera más confiable al sistema



De los más didácticos simuladores (Aimsun, 2012)



El aspecto del software en uso (Pedtrax Iteris, s.f)

AIMSUN NEXT

Este y varios simuladores ayudan en la toma de decisiones de planificación urbana. Algunos de forma más codificada que otros dan indicios sobre flujos o velocidades, o impactos sobre la economía. Aún falta un nuevo enfoque en estos programas de emulación vial urbana.

7 REFERENTES

REACTABLE

Creado por el Music Technology Group/IUA Universitat Pompeu Fabra, en Barcelona el año 2005, es un sintetizador de música en el cual los usuarios manipulan el sonido con unos bloques sobre una superficie redonda. Al rotar o acercar los bloques se pueden modificar y crear varios sonidos, beats, notas o pulsos. (Fuller, 2017)



Reactable en uso (Atelier Avant Austria CMS, 2011)

BLOCK'HOOD

De Plethora Project 2017.: “Es un juego de simulación en el que se construyen ciudades enfocadas en la ecología, la interdependencia y el deterioro. [...] El jugador es desafiado a mantener el balance ecológico a medida que cada bloque adherido va consumiendo y produciendo recursos de diferentes tipos. Aquellos bloques que no proveídos con el input requerido irán lentamente deteriorándose hasta colapsar” (Plethora, 2017)



Block'Hood busca el equilibrio (HeyPoorPlayer, 2017)

KILOBOTS

Son robots de baja complejidad que comparten un mismo código de programación y sistema de comunicación. (NewScientist, 2014) Generan figuras de manera espontánea y emergente; ninguno sabe a donde tiene que ir pero si saben que figura tienen que formar, por medio de una comunicación con luces RGB se informan entre vecinos cuales son los movimientos a seguir.



(dev.comofunciona.com.mx, 2014)

MALUUBA

Inteligencia artificial de Microsoft. Usando principios de castigo y recompensa determinaba de que manera se debía mover Mrs. Pacman para poder vencer el juego. (Weinberger, 2017) Básicamente una gran sumatoria vectorial de todas las influencias de los agentes por sobre Mrs. Pacman determinan el

vector de movimiento de esta misma; pudiendo así resolver con eficiencia el juego haciendo uso de una programación relativamente simple.

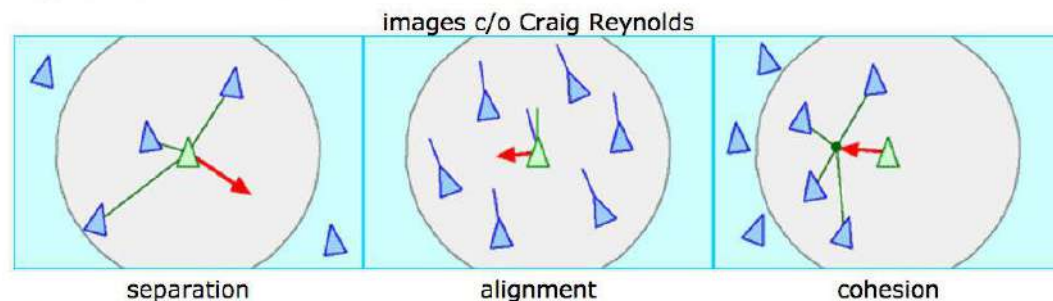


Inteligencia artificial marcó récord (Maluuba, 2017)

BODIDS DE CRAIG REYNOLDS

Expresiones matemáticas en la programación de agentes en interacciones como bandadas de pájaros. (Reynolds, s.f.) Craig Reynolds decidió hacer simulaciones visuales del comportamiento

de bandadas de pájaros por medios de la programación. Las líneas de códigos las basó en 3 expresiones matemáticas simples: una para la separación entre los agentes, otra para el alineamiento y otra para la cohesión. Para la separación de los agentes establece una máxima cercanía posible entre ellos. Para la alineación establece que los “pájaros” tiendan su alineación hacia el promedio de alineación general; es decir, la suma de todas las alineaciones dividida por la cantidad de agentes. Y por último la cohesión, es la tendencia de los pájaros para acercarse al promedio de posición de los pájaros vecinos; dentro de un radio perimetral. Las expresiones matemáticas simplifican el entendimiento de manera numérica parte de los comportamientos de los agentes. (n-e-r-v-o-u-s.com, s.f.)



Estas tres reglas de comportamiento permiten simular bandadas de pájaros (Craig Reynolds, s.f.)

BLOCKCHAIN DE BITCOIN

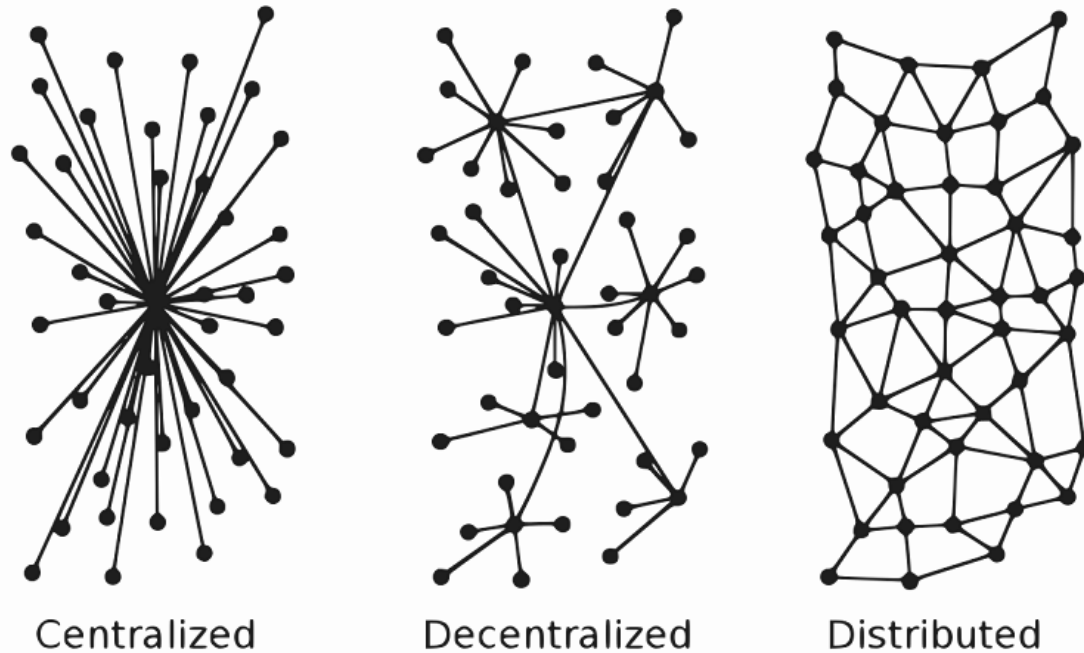
“La Blockchain o cadena de bloques es una base de datos pública y distribuida en la cual se registran todas las transacciones. Es una contabilidad pública que funciona a través de una red distribuida de ordenadores, es decir, no requiere ninguna autoridad central ni terceras partes que actúen como intermediarias”. (NoCreasNada.com, 2016) De la misma forma la se podría crear la red de los agentes móviles de la ciudad. Si cada uno de los ciudadanos tuviera un chip en su tarjeta de identificación a pocos metros de distancia, lo podrían reconocer autos, buses, incluso bicicletas inteligentes y formar un internet de los autos, en pos de la seguridad y sustentabilidad claro.

AIRBNB Y WAZE

Ambos funcionan gracias a la acumulación de las personas. Estas aplicaciones, servicios en realidad, por sí solas no son nada. Si lo usara una sola persona tampoco; el varlo de ellas está en que varias personas las usan y la potencian exponencialmente. “Son los usuarios los que potencian los mapas y la navegación. Cuantos más usuarios conduzcan con Waze abierto, mejor será la

navegación.” (Google Support, 2017)

La red al estar distribuida entre varios se sustenta en esas múltiples conexiones. Lo interesante es que cada individuo se limita a jugar con la reglas de cada juego para estar dentro del servicio; dispuestos a ser evaluados con “estrellas“ usan el servicio comportándose con cautela.



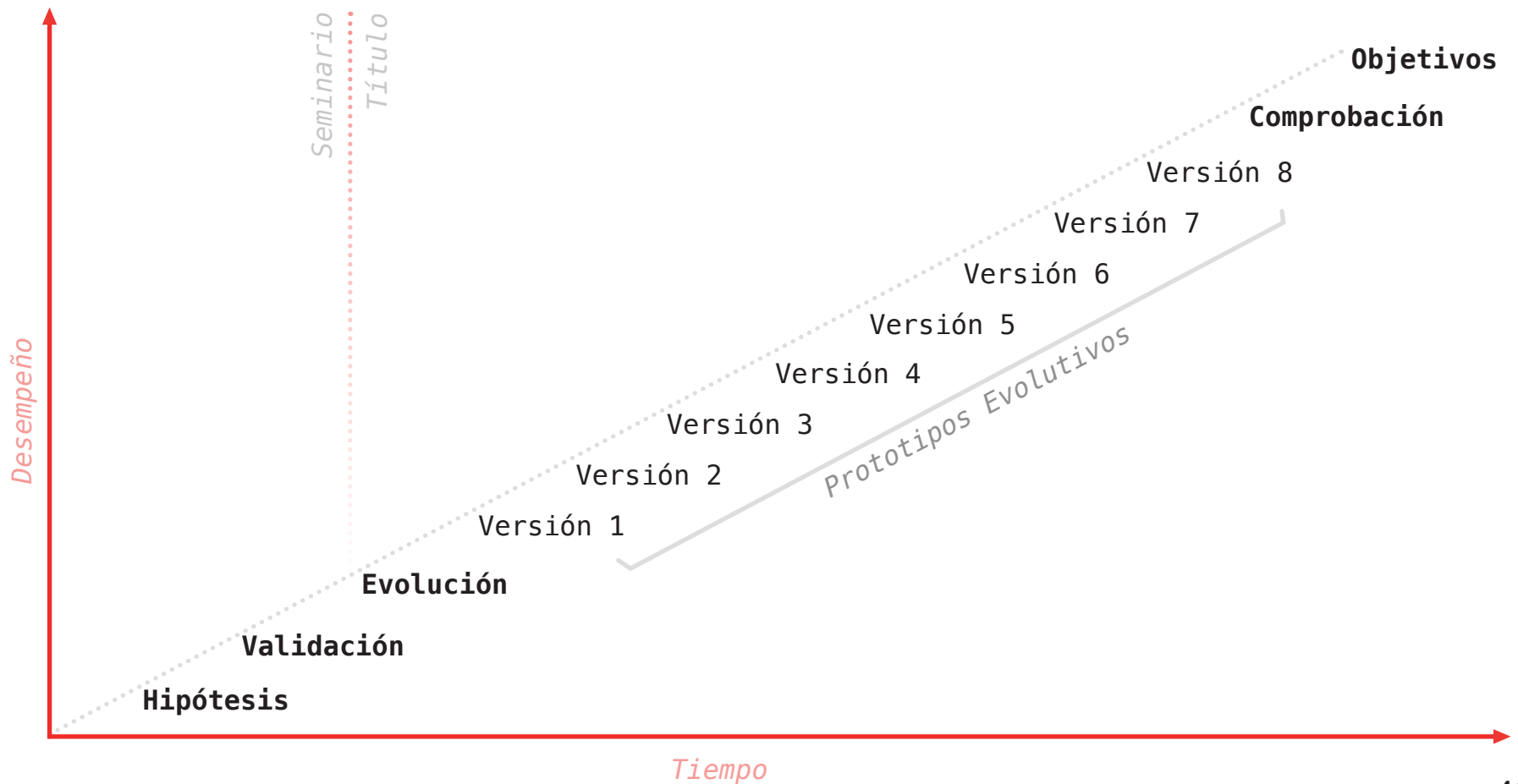
Los 3 tipos de redes según Paul Baran en 1964 (TruthCoin, 2017)

8 EL PROCESO DE DISEÑO

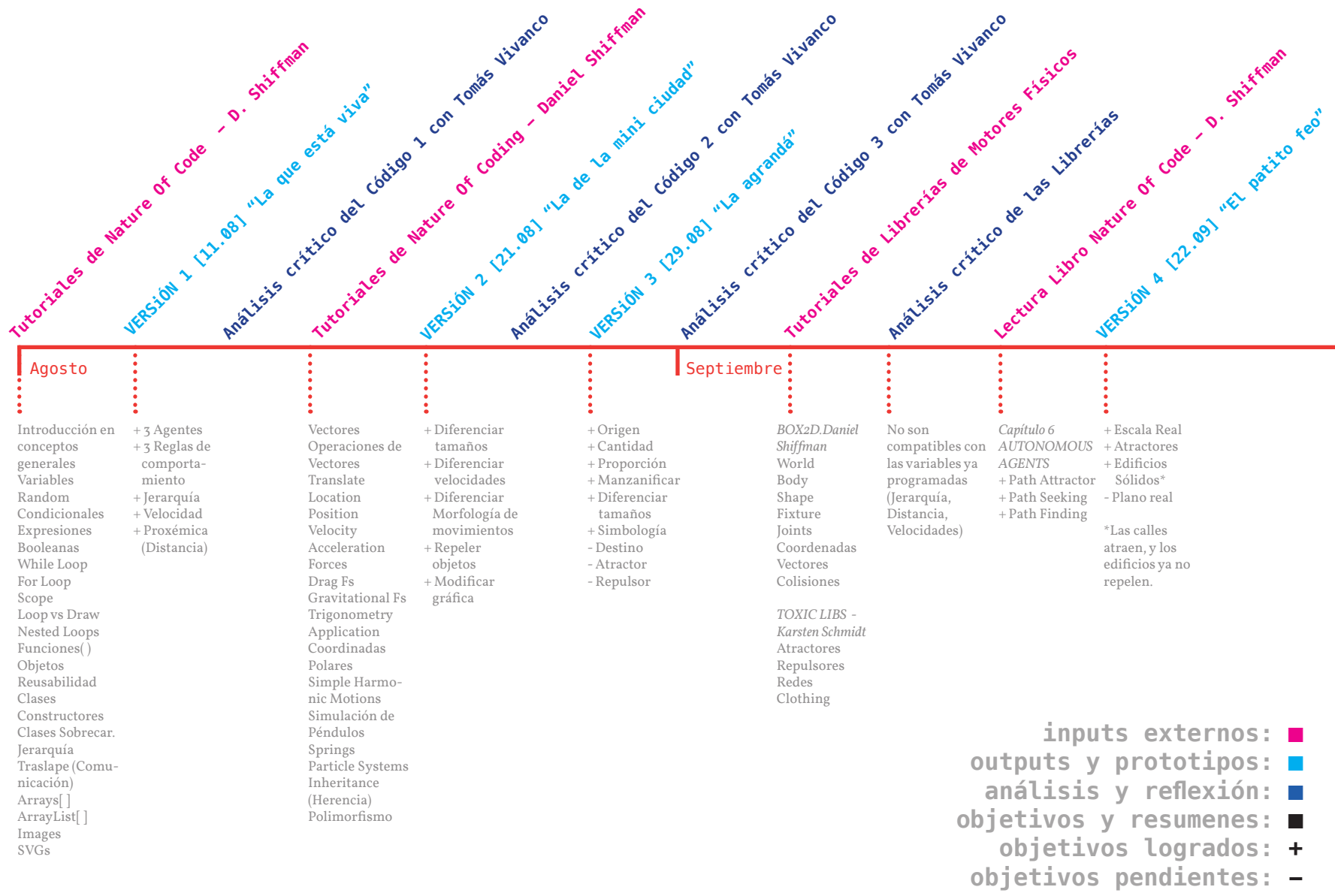
INTRODUCCIÓN GENERAL

Una vez validada la hipótesis, o bien llamada la formulación, el proceso de desarrollo del proyecto comenzó con la búsqueda y aprendizaje de un modo absolutamente nuevo de diseño: la programación.

Cada nuevo tutorial, cada libro o conversación con programadores se abrían nuevos campos de posibilidades para probar en el código del simulador. Con el tiempo este fue evolucionando y complejizándose, fallando demasiadas veces en el camino, pero finalmente llegando al desempeño deseado y lograr los objetivos del proyecto. (ver pág. 31)



LÍNEA DE TIEMPO DEL PROYECTO



Análisis crítico del Código 4 con Tomás Vivanco
 VERSIÓN 5 [02.10] "La que parece ser algo"
 Análisis crítico del Código 5 con Tomás Vivanco
 VERSIÓN 6 [16.10] "La con aspiraciones de grandeza"
 Entrevista y Análisis con PhD. Alessio Bellino
 Análisis crítico del Código 6 con Tomás Vivanco
 Entrevista y Análisis con Ricardo Vega
 VERSIÓN 7 [23.11] "La fea hija perdida"
 Análisis crítico del Código 7 con Tomás Vivanco
 Entrevista y Análisis con Oscar Figueroa
 VERSIÓN 8 [25.11] EL Simulador

Octubre

- + Plano real
- + Calles
- + Edificios
- + Mapa de coordenadas
- + Atractores de 3 tipos
- + Interfaz gráfica
- + Sliders
- + Simbología
- + Agentes ajuste de tamaño
- + Origenes controlados
- + Función fuera de mapa

- + Agregar Buses
 - + Simplificar el código
 - + Interfaz con 3 interacciones
 - Numerador de Superficie
 - Sistema actual interactivo
 - Separar Buses de Peatones
- Doctor en ciencias de la computación.
 Experto en la interacción hombre máquina

Noviembre

- + Indicadores
 - + Ajustes estéticos finales
 - + Naturaleza autoexplicativa
 - + Usabilidad
 - + Contraste de color
- Diseñador - Artista, experto en visualización de datos y programación creativa

- Economista
- Doctor en urbanismo
- experto en transporte

- + Indicadores
- + Ajustes estéticos finales
- + Ver referente visuales
- + Replantear usabilidad
- + Considerar orden de lectura
- + Mejorar visibilidad
- + Contraste y Paleta de colores
- + Ajustar formulas de indicadores
- + Mejorar jerarquía de composición.

8 análisis crítico proceso de diseño
3 entrevistas de validación con expertos
+20 hrs de tutoriales vistos de Daniel Shiffman sin contar repeticiones (LAS 8 HARRY POTTER DURAN 19.5 HRS)
8 versiones del código
+8200 líneas de código escritas solo en las versiones finales de los códigos

DESARROLLO Y DISEÑO DE LAS VERSIONES DEL SIMULADOR DESDE EL CÓDIGO

INTRODUCCIÓN

A continuación se presenta en detalle el proceso de desarrollo del código del simulador en Processing, la extensión del común lenguaje llamado JavaScript.

Cada versión del simulador fue capturado tanto en apariencia, como en función del programa de simulación, para criticar y establecer como se estaban desempeñando los objetivos del proyecto. También se capturaron las líneas de código a modo de registro, comprensión y transparencia del proceso. Cada evento, situación o apariencia fue desglosada en cada versión del simulador para concluir y formular los cambios futuros de este y para observar retrospectivamente la trayectoria de cambios sufridos por el prototipo.

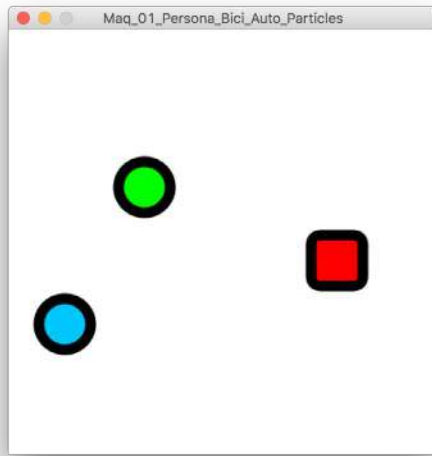
CITAS DE LOS CÓDIGOS

Para referirse a cada línea del código en bruto, al momento de describirlas, se usa una *P* para referirse a alguna pestaña en específico y seguido de *números* que corresponden a las líneas. Por ejemplo para decir que cierta línea de código está descrita desde la línea 23 hasta la 35 en la cuarta pestaña estará referenciado de esta manera: “*P4, 23-35*”

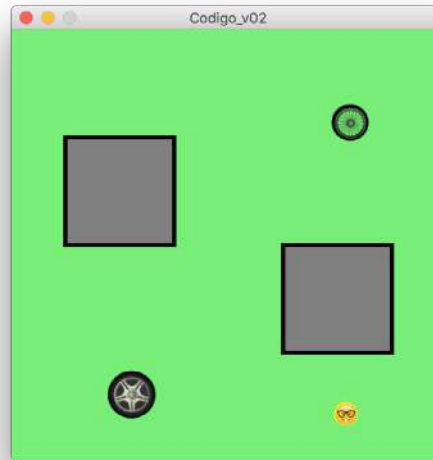
LA ÚLTIMA ITERACIÓN

En la página 139 se encuentra la versión final del proyecto (y su código) la cual incluye la mayor parte de todas las versiones anteriores, dado que fue un proceso adhitivo; cada nuevo código incluye el anterior como base a la cual se le van sumando grados de complejidad y más opciones que van acercandose más a los objetivos del proyecto.

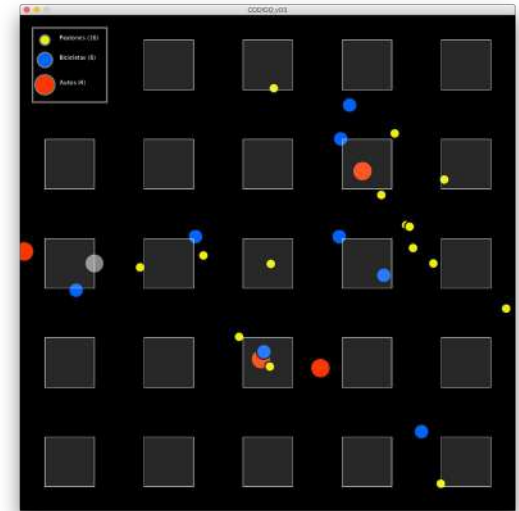
Cabe mencionar que se desarrolló el simulador sin nunca antes haber programado ni en Processing o otro lenguaje.



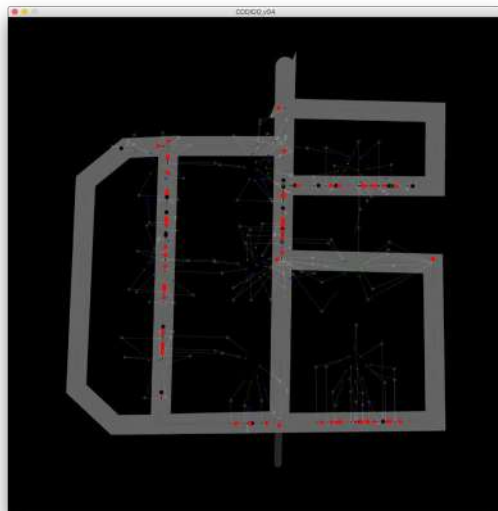
V01 "LA QUE ESTÁ VIVA"



V02 "LA DE LA MINI CIUDAD"



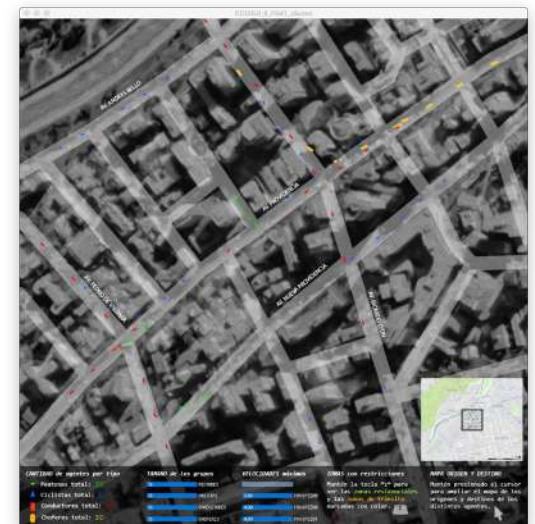
V03 "LA AGRANDÁ"



V04 "EL PATITO FE0"



V05 "LA QUE PARECE SER ALGO"

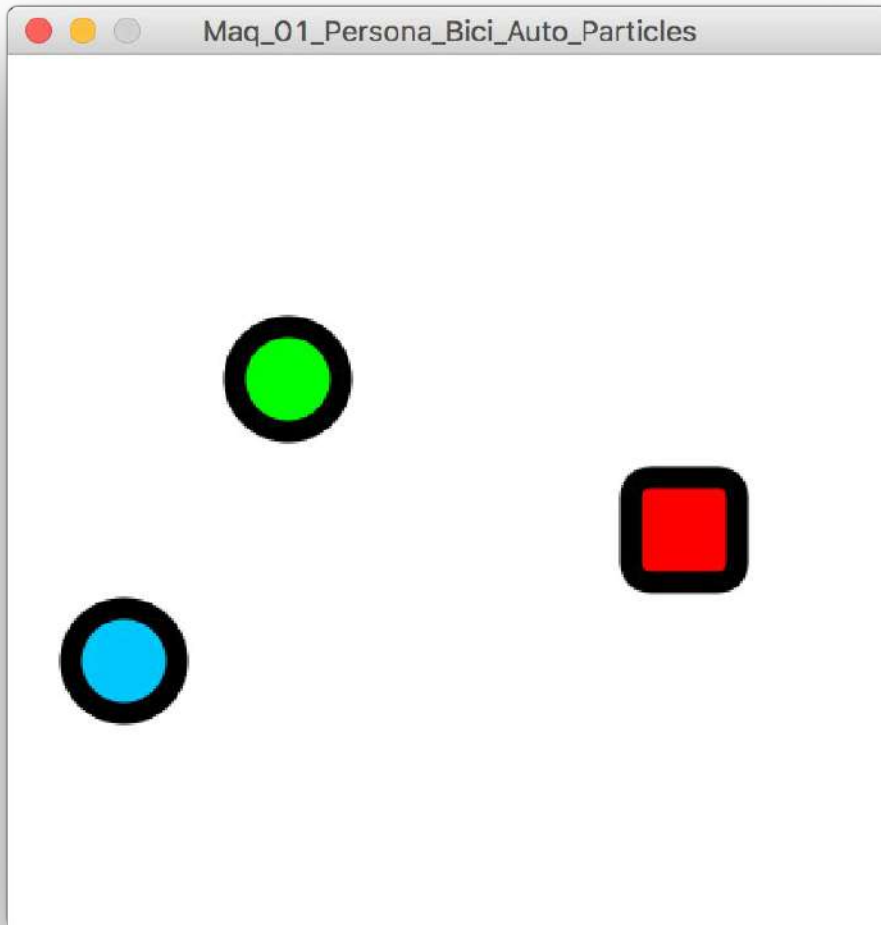


V06 "LA CON ASPIRACIONES DE GRANDEZA"

*La iteración última se encuentra en la página 139

VERSIÓN 01

la que "iestá viva!"



```
Codigo_v01 | Processing 3.3.5
Código_v01 Auto Bici
1 // DEFINIR CLASES
2 Persona p0;
3 Bici b0;
4 Auto a0;
5
6 // DEFINIR EL ESPACIO DE INTERACCIÓN
7 void setup() {
8   size(400, 400);
9 }
10 // CREAR AGENTES
11 p0 = new Persona();
12 b0 = new Bici();
13 a0 = new Auto();
14 }
15
16 void draw() {
17   background(255);
18   // ACTIVAR AGENTES Y SUS FUNCIONES
19   p0.run();
20   b0.run();
21   a0.run();
22
23 //EVENTOS DE PROXÉMICA
24 if (b0.choca(p0)) { //Bici choca con Persona
25   noStroke();
26   fill(255, 0, 0, 66);
27   rect(0, 0, 400, 400);
28   b0.velocity.x = b0.velocity.x * -1;
29   b0.velocity.y = b0.velocity.y * -1;
30 }
31 if (a0.choca(p0)) { //Auto choca con Persona
32   noStroke();
33   fill(255, 0, 0, 66);
34   rect(0, 0, 400, 400);
35   a0.velocity.x = a0.velocity.x * -1;
36   a0.velocity.y = a0.velocity.y * -1;
37 }
38 if (a0.choca(b0)) { //Auto choca con Bici
39   noStroke();
40   fill(255, 0, 0, 66);
41   rect(0, 0, 400, 400);
42   a0.velocity.x = a0.velocity.x * -1;
43   a0.velocity.y = a0.velocity.y * -1;
44 }
45 }
```

Pestaña 1: Programa

```

Codigo_v01 | Processing 3.3.5
Java
Codigo_v01 Auto Bici
1 class Auto extends Bici {
2
3 // POSICION DE ORIGEN Y
4 // VELOCIDAD DE DESPLAZAMIENTO
5 Auto() {
6     position = new PVector(random(100,300),
7         random(100,300));
8     velocity = new PVector(3.3, 2.2);
9 }
10 // APARIENCIA
11 void display() {
12     // Display circle at x position
13     stroke(0);
14     fill(255, 0, 0);
15     rectMode(CENTER);
16     rect(position.x, position.y, 48, 48,10);
17     rectMode(CORNER);
18 }
19 // INTERACCION CON PERSONA
20 boolean choca(Persona fulano) {
21     float d = dist(a0.position.x,
22         a0.position.y, p0.position.x,
23         p0.position.y);
24     if (d < 80) {
25         return true;
26     } else {
27         return false;
28     }
29 }
30 // INTERACCION CON BICI
31 boolean choca(Bici fulano) {
32     float d = dist(a0.position.x,
33         a0.position.y, fulano.position.x,
34         fulano.position.y);
35     if (d < 80) {
36         return true;
37     } else {
38         return false;
39     }
40 }
41 }
42
43
44
45

```

Pestaña 2: Auto

```

Codigo_v01 | Processing 3.3.5
Java
Codigo_v01 Auto Bici
1 class Bici {
2     PVector position; PVector velocity;
3 // POSICION DE ORIGEN Y
4 // VELOCIDAD DE DESPLAZAMIENTO
5 Bici() {
6     position = new PVector(random(100,300),
7         random(100,300));
8     velocity = new PVector(2.2, 3.3);
9 }
10 // FUNCION DE FUNCIONES
11 void run() {
12     update();
13     display();
14 }
15 // DESPLAZAMIENTO Y FRONTERAS
16 void update() {
17     // Add the current speed to the position.
18     position.add(velocity);
19     if ((position.x > width) ||
20         (position.x < 0)) {
21         velocity.x = velocity.x * -1;
22     }
23     if ((position.y > height) ||
24         (position.y < 0)) {
25         velocity.y = velocity.y * -1;
26     }
27 }
28 // APARIENCIA
29 void display() {
30     // Display circle at x position
31     stroke(0);
32     strokeWeight(10);
33     fill(0,200,255);
34     ellipse(position.x, position.y, 48, 48);
35 }
36 // INTERACCION CON PERSONA
37 boolean choca(Persona fulana) {
38     float d = dist(b0.position.x,
39         b0.position.y, fulana.position.x,
40         fulana.position.y);
41     if (d < 80) {
42         return true;
43     } else {
44         return false;
45     } } }

```

Pestaña 3: Bici

```

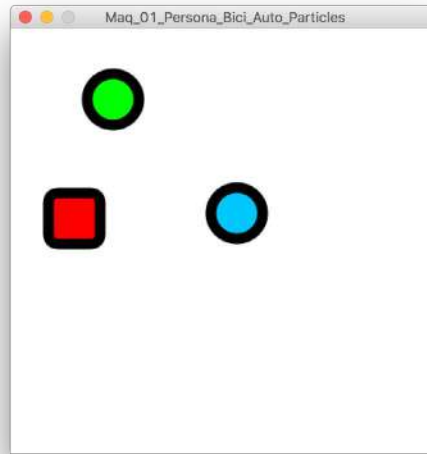
Codigo_v01 | Processing 3.3.5
Java
Codigo_v01
1 class Persona {
2     PVector position; PVector velocity;
3     PVector acceleration; float topspeed;
4
5 // POSICION DE ORIGEN, VELOCIDAD Y LIMITE DE V
6 Persona() {
7     position = new PVector(width/3, height/3);
8     velocity = new PVector(0, 0);
9     topspeed = 3;
10 }
11 // FUNCION DE FUNCIONES
12 void run() {
13     update();
14     display();
15     checkEdges();
16 }
17 // DESPLAZAMIENTO, VELOCIDAD Y ACELERACION
18 void update() {
19     acceleration = PVector.random2D();
20     acceleration.mult(random(2));
21     velocity.add(acceleration);
22     velocity.limit(topspeed);
23     position.add(velocity);
24 }
25 // APARIENCIA
26 void display() {
27     stroke(0);
28     strokeWeight(10);
29     fill(0,255,0);
30     ellipse(position.x, position.y, 48, 48);
31 }
32 // FRONTERAS (SIN FRONTERAS)
33 void checkEdges() {
34     if (position.x > width +24) {
35         position.x = 0 -24;
36     } else if (position.x < 0 -24) {
37         position.x = width +24;
38     }
39     if (position.y > height +24) {
40         position.y = 0-24;
41     } else if (position.y < 0 -24) {
42         position.y = height+24;
43     }
44 }
45 }

```

Pestaña 4: Persona

VERSIÓN 01

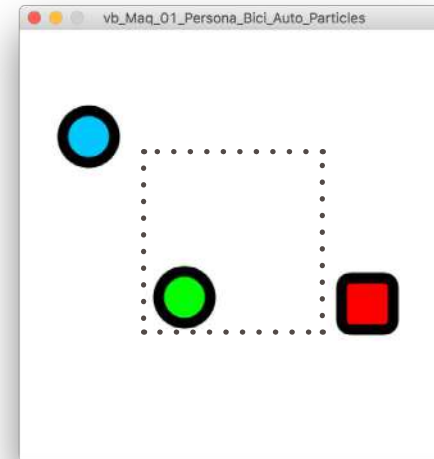
funcionamiento y
observaciones



1. APARIENCIA

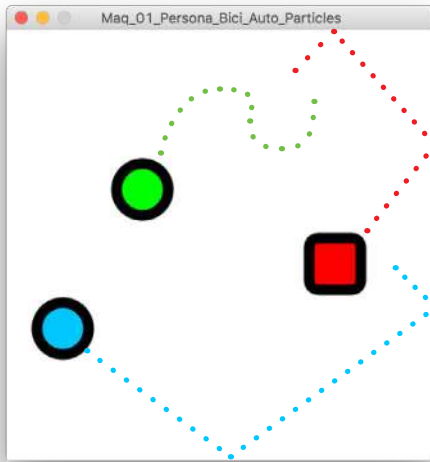
La apariencia de la “persona” está declarada en la Pestaña 4 de la línea 25 a 31. Es un círculo verde de trazo grueso negro de diámetro de 48 píxeles. La apariencia de la “bicicleta” está declarada en la Pestaña 4 de la línea 28 a 35. Es un círculo azul claro del mismo trazo y diámetro. El “auto” está declarado en la Pestaña 2 desde la línea 10 a la 18. Es un cuadrado rojo de lado igual al diámetro de la bici.

Las figuras a pesar de ser de diferente color no suelen distinguirse tanto entre sí, y también no representan con claridad ni intuitividad el modo de transporte respectivo pensando en los usuarios.



2. PUNTO DE ORIGEN

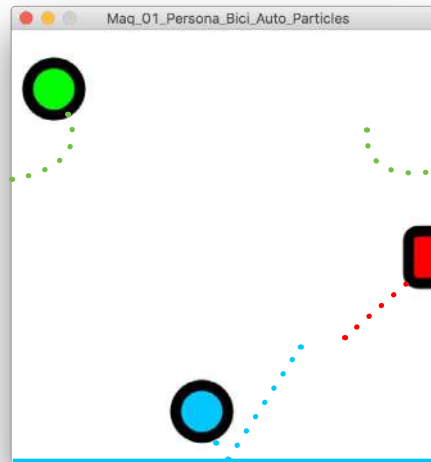
El origen de la persona está escrito en la Pestaña 4: línea 7; este indica que puede aparecer aleatoriamente dentro de los tercios de la ventana. La posición de origen de la bici y el auto está descrita en las líneas seis de la Pestaña 3 y 2 respectivamente: éstas definen la aparición de manera aleatoria entre los mismo tercios que en los que aparece la persona. La aparición de cada uno de los modos de transporte podría estar marcada desde algunos puntos específicos que remitan a algún punto de acceso o salida de estos mismo, ejemplo: una puerta para las personas.



3. DESPLAZAMIENTO

El desplazamiento de las personas (Pestaña 4, líneas 17-24) está pensado para ser aleatorio de manera constante y con velocidad variable entre 0 a 2. La bicicleta (P3, L8) se mueve 2.2 píxeles en x y 5.5 en y, es decir aproximadamente 4 px en diagonal. El auto (P2, L8) igual pero con lo valores invertidos.

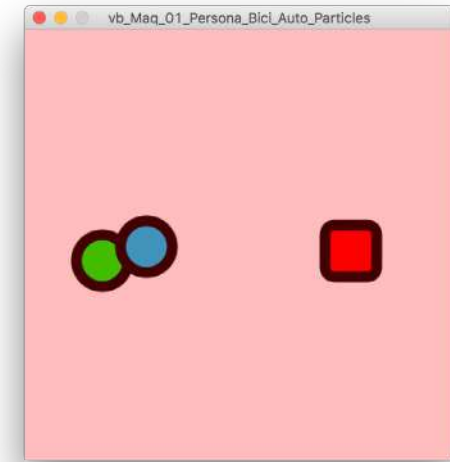
Los tipos de desplazamiento se podrían diferenciar en mayor medida para poder observar y distinguir con menor esfuerzo. Así también se podría aludir a los tipos de movimientos y velocidades más cercanos a la realidad permitiéndole a quien usa el simulador relacionar los modos digitales con los reales.



4. FRONTERAS

Las restricciones de movimiento para la persona (P4, L32-44) no existen, o bien, cada vez que está cruza un borde aparece por el otro lado a la misma velocidad. La bicicleta y el auto (P3, L15-27) comparten la misma función de las fronteras: si llegan al límite de arriba o abajo de la ventana cambian el sentido de su desplazamiento vertical, y si llegan al límite de los lados cambian su desplazamiento horizontal.

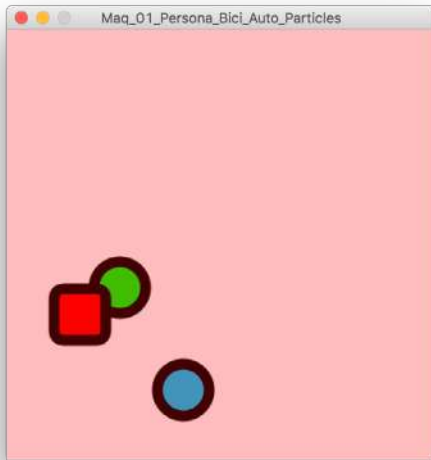
Se podrían establecer obstáculos, o bien “edificios” para poder hacer pruebas de como pueden aparecer naturalmente flujos y diferentes formas de interacción y movimiento.



5.1 PROXÉMICA

En caso de que una bicicleta entra los 80 píxeles de espacio personal (P3, L36-45) de la persona, valga la redundancia, cambia de dirección, pero no de sentido, se vuelve por donde venía. En cada caso de proxémica la pantalla se torna roja para indicar la “situación de riesgo”

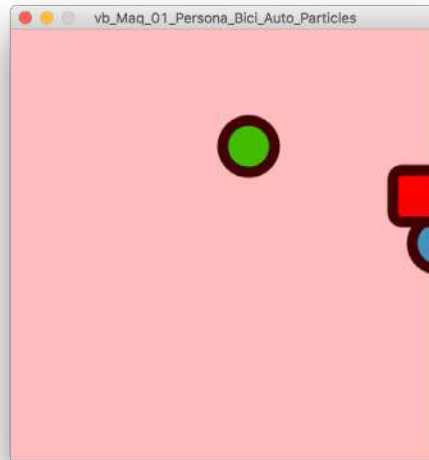
En vez de marcar con rojo cada situación donde se cruzan los rangos de espacio personal, se podría establecer un límite de interacción mayor para activar las funciones de cambio de dirección, y mantener el cambio de color en caso de haber contacto directo para diferenciar los “riesgos”.



5.2 PROXÉMICA

En caso de que un auto entra los 80 píxeles de espacio personal (P2, L19-29) de la persona, cambia de dirección, pero no de sentido. Al igual que la bicicleta se devuelve por donde venía.

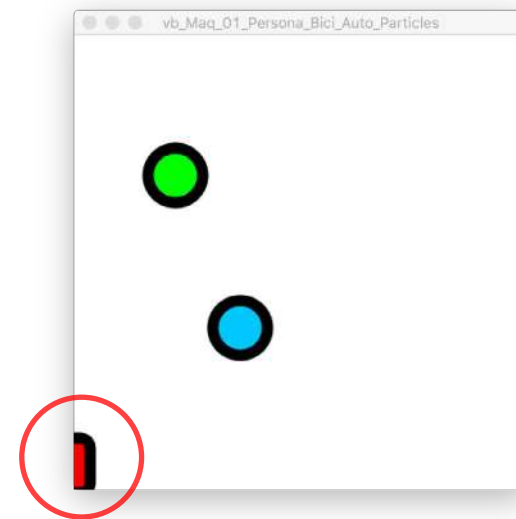
Así como la bicicleta logra mantener su distancia con las personas de manera natural, los autos debiesen respetar de la misma intuitiva manera tal movimiento, espera y distancia. En versiones futuras sería óptimo que el usuario pudiese manipular la distancia programada entre los agentes usando el simulador.



5.3 PROXÉMICA

En caso de que un auto entra los 80 píxeles de espacio personal (P2, L19-29) de la bicicleta, cambia de dirección, pero no de sentido. Al igual que su interacción con los autos.

Casos extraños de proxémica al usuario pueden parecerle extraños, sería bueno a medida que se van mejorando las versiones del simulador poder ampliar el mapa y el espacio simulado a uno incluso real.



6. ERRORES

En casos particulares donde las personas producto de su movimiento azaroso acorralan o al auto o a la bici en las esquinas y las expulsan del límite de la ventana y estas se quedan estancadas en el limbo. Otro evento particular es cuando la persona se queda circundando alrededor y encima de los otros agentes y estos cambian de dirección de manera continua quedándose en el mismo lugar vibrando hasta que la persona “decide” irse.

Con hacer a la persona más lenta quizás se podrían evitar aquellas expulsiones o movimientos estáticos de los otros agentes.

VERSIÓN 01

reflexión y proyección

CONCLUSIÓN GENERAL

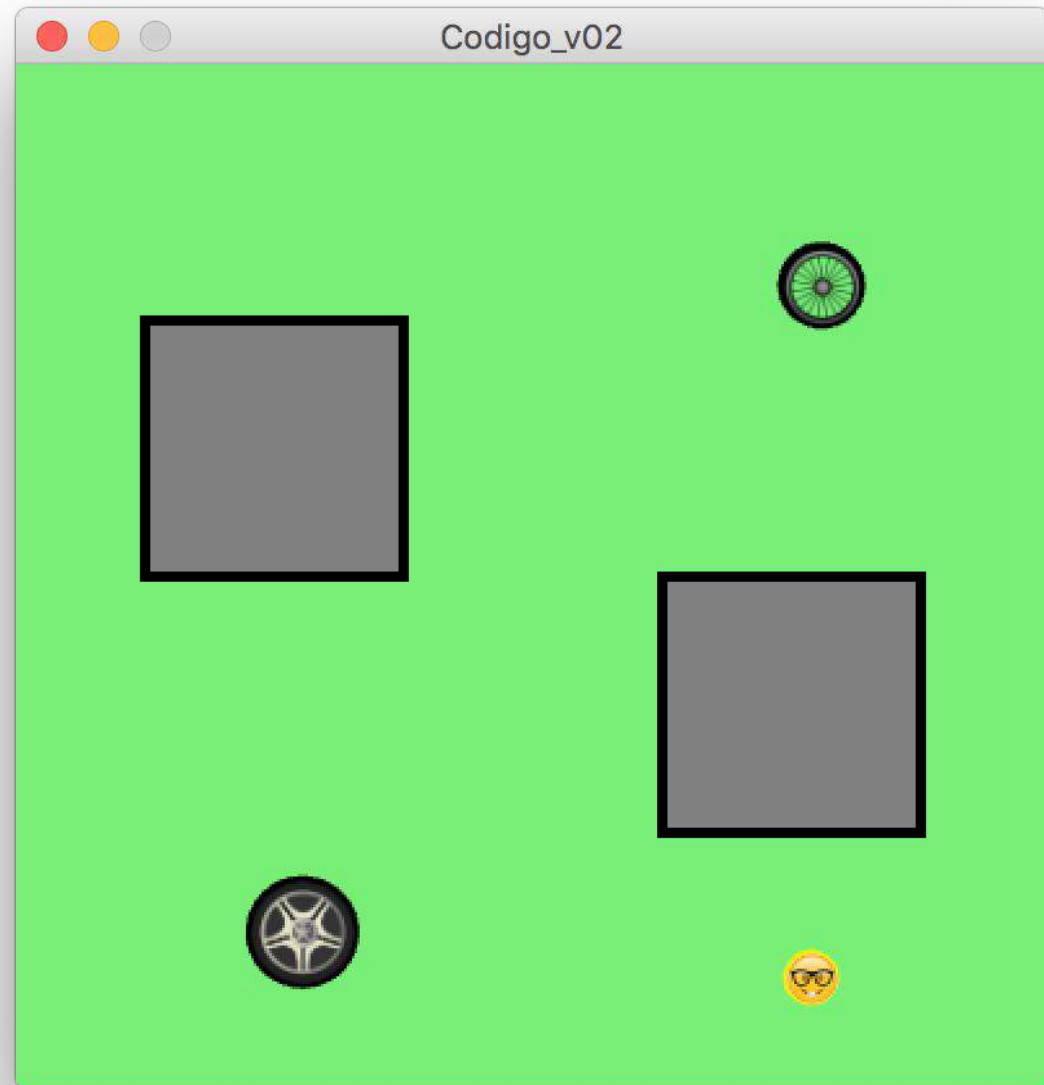
Las figuras pudiesen distinguirse más en pos de una mejor observación a través de cambios de aspecto, u apariencia, tipo de desplazamiento y también comportamiento entre sí y con otros objetos. Es también necesario tener en consideración que estos agentes debieran responder a los diferentes motivos de por qué se trasladan en un principio, es por esto que es pertinente, quizás no obligatoriamente necesario, agregar puntos de origen definidos y controlados para estos. Para aportar en la distinción de los movimientos se podría cambiar el tipo de movimiento de la bicicleta y el auto para que parezcan más naturales y muestren un poco más de impredecibilidad; así como lo hace la persona. Es para esto que se podrían eliminar las fronteras de la ventana del programa y agregar obstáculos u otros elementos que se condigan con la realidad de la infraestructura urbana. La reacción de las clases, el rebote de los agentes, se muestra brusca y poco natural, esta se podría ajustar por medio de lo nuevas morfologías de movimientos de estos.

PRÓXIMOS PASOS

1. Se diferenciarán más la velocidades de los distintos tipos de agentes.
2. La morfología, el tipo de movimiento, de estos mismos se distinguirá más por medio del factor controlado del azar. Cada detalle de movimiento, y velocidad le permitirán al usuario entender de forma intuitiva los movimientos de los agentes y que son por si mismos.
3. Se incluirán objetos a modo de obstáculos para acercarse más a distintas infraestructuras y los comportamientos que estos significan. Este paso es esencial para luego ampliar el espacio de simulación a uno real, un fragmento de la ciudad real
4. Modificar la gráfica; con esto se refiere a los tamaños y formas y aspectos. Por el momento el simulador solo logra comunicar la jerarquía, pero de forma anónima desde el punto de vista de los modos de transporte; los usuarios podrían confundir que agente digital responde a que tipo de modo de transporte real.

VERSIÓN 02

la de la mini ciudad



```

Codigo_v02 | Processing 3.3.5
Java
Codigo_v02
1 // DECLARACION DE LOS AGENTES
2 Persona p0;
3 Bici b0;
4 Auto a0;
5 MurallaH m0;
6 MurallaV m1;
7 MurallaH m2;
8 MurallaV m3;
9 MurallaH m4;
10 MurallaV m5;
11 MurallaH m6;
12 MurallaV m7;
13 // COORDS LAS MURALLAS
14 int ex1 = 50;
15 int ey1 = 100;
16 int ex2 = 150;
17 int ey2 = 200;
18 int ex3 = 250;
19 int ey3 = 200;
20 int ex4 = 350;
21 int ey4 = 300;
22
23 void setup() {
24   size(400, 400, P2D);
25   // ORIGEN, Posicion inicial
26   p0 = new Persona(0 + 30, height/2);
27   b0 = new Bici(width/2, height - 30);
28   a0 = new Auto(width - 30, height/2);
29
30   // COORDS MURALLA a MURALLA
31   m0 = new MurallaH(ex1, ey1, ex2, ey1);
32   m1 = new MurallaV(ex2, ey1, ex2, ey2);
33   m2 = new MurallaH(ex1, ey2, ex2, ey2);
34   m3 = new MurallaV(ex1, ey1, ex1, ey2);
35   m4 = new MurallaH(ex3, ey3, ex4, ey3);
36   m5 = new MurallaV(ex4, ey3, ex4, ey4);
37   m6 = new MurallaH(ex3, ey4, ex4, ey4);
38   m7 = new MurallaV(ex3, ey3, ex3, ey4);
39 }
40
41 void draw() {
42   background(120, 240, 120);
43   // COLOR DE LOS EDIFICIOS
44   fill(128);
45   rect(ex1, ey1, ex2 - ex1, ey2 - ey1);

```

Pestaña 1: Programa 1/2

```

Codigo_v02 | Processing 3.3.5
Java
Codigo_v02
42 background(120, 240, 120);
43 // COLOR DE LOS EDIFICIOS
44 fill(128);
45 rect(ex1, ey1, ex2 - ex1, ey2 - ey1);
46 rect(ex3, ey3, ex4 - ex3, ey4 - ey3);
47 // FUNCIONES DE AGENTES
48 b0.run();
49 p0.run();
50 a0.run();
51 // FUNCION DE LAS MURALLAS
52 m0.build(); m1.build(); m2.build();
53 m3.build(); m4.build(); m5.build();
54 m6.build(); m7.build();
55
56 //EVENTOS DE PROXEMICA ENTRE AGENTES
57 if (b0.choca(p0)) { //Bici choca con Persona
58   b0.velocity.x = b0.velocity.x * -0.3;
59   b0.velocity.y = b0.velocity.y * -0.3;
60 }
61 if (a0.choca(p0)) { //Auto choca con Persona
62   a0.velocity.x = a0.velocity.x * -0.3;
63   a0.velocity.y = a0.velocity.y * -0.3;
64 }
65 if (a0.choca(b0)) { //Auto choca con Bici
66   a0.velocity.x = a0.velocity.x * -0.3;
67   a0.velocity.y = a0.velocity.y * -0.3;
68 }
69 }
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86

```

Pestaña 1: Programa 2/2

```

Codigo_v02 | Processing 3.3.5
Java
Codigo_v02
1 class Auto extends Bici {
2   Auto(float x_, float y_) {
3     super(x_, y_);
4     topspeed = 4.5;
5     // TEXTURA DE RUEDA
6     foto = loadImage("auto.png");
7   } // APARIENCIA
8   void display() {
9     diam = 40;
10    stroke(0);
11    strokeWeight(4);
12    fill(255, 56, 0);
13    ellipse(position.x, position.y,
14            diam, diam);
15    // TEXTURA DE RUEDA
16    imageMode(CENTER);
17    image(foto, position.x, position.y,
18          diam, diam);
19  } // DESPLAZAMIENTO, VELOCIDAD Y ACELERACION
20  void update() {
21    acceleration = PVector.random2D();
22    acceleration.mult(0.2);
23    velocity.add(acceleration);
24    velocity.limit(topspeed);
25    position.add(velocity);
26  } // INTERACCION CON PERSONA
27  boolean choca(Persona fulano) {
28    float d = dist(a0.position.x,
29                 a0.position.y, p0.position.x,
30                 p0.position.y);
31    if (d < 60) {
32      return true;
33    } else {
34      return false;
35    }
36  } // INTERACCION CON BICI
37  boolean choca(Bici fulano) {
38    float d = dist(a0.position.x,
39                 a0.position.y, fulano.position.x,
40                 fulano.position.y);
41    if (d < 60) {
42      return true;
43    } else {
44      return false;
45    } }

```

Pestaña 2: Auto

```
Codigo_v02 | Processing 3.3.5
Java
Codigo_v02
1 class Bici extends Persona {
2
3   Bici(float x_, float y_) {
4     super(x_, y_);
5     topspeed = 2.5;
6     // TEXTURA DE RUEDA
7     foto = loadImage("bici.png");
8   }
9
10  // APARIENCIA
11  void display() {
12    diam = 30;
13    stroke(0);
14    strokeWeight(4);
15    noFill();
16    ellipse(position.x, position.y, diam, diam
17    // TEXTURA DE RUEDA
18    imageMode(CENTER);
19    image(foto, position.x, position.y, diam,
20  }
21
22  // DESPLAZAMIENTO, VELOCIDAD Y ACELERACION
23  void update() {
24    acceleration = PVector.random2D();
25    acceleration.mult(0.2);
26    velocity.add(acceleration);
27    velocity.limit(topspeed);
28    position.add(velocity);
29  }
30
31  // INTERACCION CON PERSONA
32  boolean choca(Persona fulana) {
33    float d = dist(b0.position.x,
34    b0.position.y, fulana.position.x,
35    fulana.position.y);
36    if (d < 60) {
37      return true;
38    } else {
39      return false;
40    }
41  }
42
43
44
45
```

Pestaña 3: Bici

```
Codigo_v02 | Processing 3.3.5
Java
Codigo_v02
1 class MurallaH {
2
3   PVector vertice1;
4   PVector vertice2;
5   PVector vertice3;
6   PVector vertice4;
7
8   MurallaH(int mxh1, int mhy1,
9     int mxh2, int mhy2) {
10    vertice1 = new PVector(mxh1, mhy1);
11    vertice2 = new PVector(mxh2, mhy1);
12    vertice3 = new PVector(mxh2, mhy2);
13    vertice4 = new PVector(mxh1, mhy2);
14  }
15
16  void build() {
17    strokeWeight(4);
18    stroke(0);
19    fill(128);
20    quad(
21    vertice1.x, vertice1.y,
22    vertice2.x, vertice2.y,
23    vertice3.x, vertice3.y,
24    vertice4.x, vertice4.y);
25  }
26 }
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
```

Pestaña 4: Muralla Hor.

```
Codigo_v02 | Processing 3.3.5
Java
Codigo_v02
1 class MurallaV extends MurallaH {
2
3   PVector vertice1;
4   PVector vertice2;
5   PVector vertice3;
6   PVector vertice4;
7
8   MurallaV(int mvx1, int mvy1,
9     int mvx2, int mvy2) {
10    super(mvx1, mvy1, mvx2, mvy2);
11    vertice1 = new PVector(mvx1, mvy1);
12    vertice2 = new PVector(mvx2, mvy1);
13    vertice3 = new PVector(mvx2, mvy2);
14    vertice4 = new PVector(mvx1, mvy2);
15  }
16 }
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
```

Pestaña 5: Muralla Ver.

```

Codigo_v02 | Processing 3.3.5
Java
Codigo_v02
1 class Persona {
2   PVector position;
3   PVector velocity;
4   PVector acceleration;
5   float topspeed;
6   int diam;
7   PImage foto;
8
9   // POSICION DE ORIGEN, VELOCIDAD
10  // Y LIMITE DE VELOCIDAD
11  Persona(float x_, float y_) {
12    float ppx = x_;
13    float ppy = y_;
14    position = new PVector(ppx, ppy);
15    velocity = new PVector(0, 0);
16    topspeed = 1.5;
17    foto = loadImage("persona.png");
18  }
19
20  // FUNCION DE FUNCIONES
21  void run() {
22    update();
23    display();
24    checkEdges();
25    choqueMurallas();
26  }
27
28  // DESPLAZAMIENTO, VELOCIDAD Y ACELERACION
29  void update() {
30    acceleration = PVector.random2D();
31    acceleration.mult(random(0.07));
32    velocity.add(acceleration);
33    velocity.limit(topspeed);
34    position.add(velocity);
35  }
36
37  // APARIENCIA
38  void display() {
39    diam = 20;
40    stroke(240,240,0);
41    strokeWeight(2);
42    fill(255);
43    ellipse(position.x, position.y,
44            diam, diam);
45    imageMode(CENTER);

```

Pestaña 6: Persona 1/3

```

Codigo_v02 | Processing 3.3.5
Java
Codigo_v02
37 // APARIENCIA
38 void display() {
39   diam = 20;
40   stroke(240,240,0);
41   strokeWeight(2);
42   fill(255);
43   ellipse(position.x, position.y,
44           diam, diam);
45   imageMode(CENTER);
46   image(foto, position.x, position.y,
47         diam, diam);
48 }
49
50 // FRONTERAS (SIN FRONTERAS)
51 void checkEdges() {
52   // Bordes de la ventana
53   if (position.x > width + diam/2) {
54     position.x = 0 - diam/2;
55   } else if (position.x < 0 - diam/2) {
56     position.x = width + diam/2;
57   }
58   if (position.y > height + diam/2) {
59     position.y = 0 - diam/2;
60   } else if (position.y < 0 - diam/2) {
61     position.y = height + diam/2;
62   }
63 }
64
65 void choqueMurallas() {
66   // EDIFICIO 1 (muralla a muralla)
67   // Choque murallas horizontales //
68   // BOUNCE c/ SENTIDOOOP = velocity.mult(-1)
69   if ((position.x > m0.vertice1.x - diam/2)
70       && (position.x < m0.vertice3.x + diam/2)
71       && (position.y > m0.vertice1.y -diam/2)
72       && (position.y < m0.vertice3.y + diam/2))
73     velocity.y = velocity.y * (-1);
74 }
75   if ((position.x > m2.vertice1.x - diam/2)
76       && (position.x < m2.vertice3.x + diam/2)
77       && (position.y > m2.vertice1.y -diam/2)
78       && (position.y < m2.vertice3.y + diam/2))
79     velocity.y = velocity.y * (-1);
80 }
81 // Choque muralla vertical

```

Pestaña 6: Persona 2/3

```

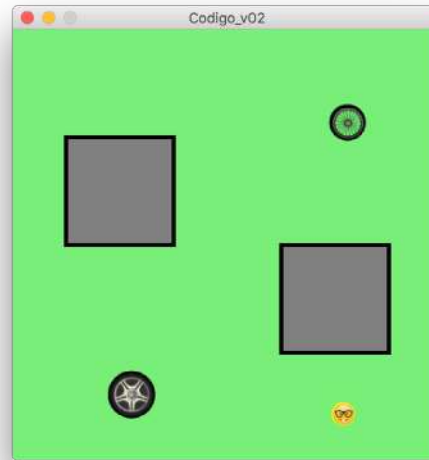
Codigo_v02 | Processing 3.3.5
Java
Codigo_v02
79   velocity.y = velocity.y * (-1);
80 }
81 // Choque muralla vertical
82 if ((position.x > m1.vertice1.x - diam/2)
83     && (position.x < m1.vertice3.x + diam/2)
84     && (position.y > m1.vertice1.y - diam/2)
85     && (position.y < m1.vertice3.y + diam/2))
86   velocity.x = velocity.x * (-1);
87 }
88   if ((position.x > m3.vertice1.x - diam/2)
89     && (position.x < m3.vertice3.x + diam/2)
90     && (position.y > m3.vertice1.y - diam/2)
91     && (position.y < m3.vertice3.y + diam/2))
92   velocity.x = velocity.x * (-1);
93 }
94 // EDIFICIO 2 (muralla a muralla)
95 // Choque murallas horizontales //
96 // BOUNCE c/ SENTIDOOOP = velocity.mult(-1)
97 if ((position.x > m4.vertice1.x - diam/2)
98     && (position.x < m4.vertice3.x + diam/2)
99     && (position.y > m4.vertice1.y -diam/2)
100    && (position.y < m4.vertice3.y + diam/2))
101   velocity.y = velocity.y * (-1);
102 }
103   if ((position.x > m6.vertice1.x - diam/2)
104     && (position.x < m6.vertice3.x + diam/2)
105     && (position.y > m6.vertice1.y -diam/2)
106     && (position.y < m6.vertice3.y + diam/2))
107   velocity.y = velocity.y * (-1);
108 }
109 // Choque muralla vertical
110 if ((position.x > m5.vertice1.x - diam/2)
111     && (position.x < m5.vertice3.x + diam/2)
112     && (position.y > m5.vertice1.y - diam/2)
113     && (position.y < m5.vertice3.y + diam/2))
114   velocity.x = velocity.x * (-1);
115 }
116   if ((position.x > m7.vertice1.x - diam/2)
117     && (position.x < m7.vertice3.x + diam/2)
118     && (position.y > m7.vertice1.y - diam/2)
119     && (position.y < m7.vertice3.y + diam/2))
120   velocity.x = velocity.x * (-1);
121 }
122 }
123 }

```

Pestaña 6: Persona 3/3

VERSIÓN 02

funcionamiento,
observaciones y
conclusiones

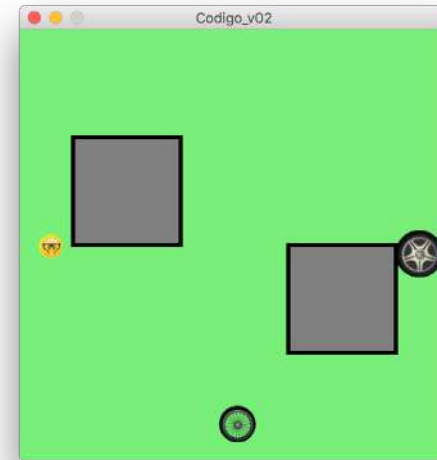


1. APARIENCIA

Color verde del fondo del programa: P1, 42. Color grisáceo de las manzanas: P1, 43-46. El tamaño de las manzanas está definido por las coordenadas cartesianas de cada muralla en P1, 13-21.

Para mejorar la distinción entre cada uno de los agentes se les incrustó a cada uno una imagen y además se los diferencié en tamaño. Aspecto de la persona: P6, 38-48. Aspecto de la bici: P3, 10-20. Aspecto del auto: P2, 8-17.

Se logra notar la diferencia de una mejor manera comparado con la versión anterior. Aun la forma es simbólica para un funcionario municipal.



2. PUNTO DE ORIGEN

En pos de futuras versiones del programa se decidió crear punto de origen y aparición para cada uno de los tipos de agentes, anticipando posibles orígenes respecto a la ciudad, y así también destinos. Los puntos de origen de cada uno de los agentes está definido en la Pestaña 1 de la línea 25 a 28.

El control del origen podría ser de utilidad en versiones posteriores. Podría el usuario quizás seguir de donde provienen y a donde van los agentes con el fin de representar: horarios punta, puntos de interés, horarios de salidas de colegio, eventos deportivos, etc.



3. DESPLAZAMIENTO

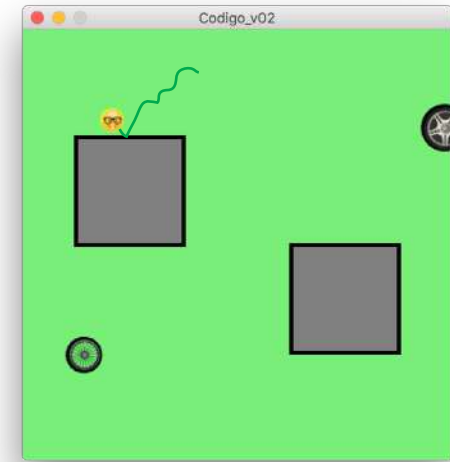
Se creó una clase para las personas (P6, 1), la cual incluye varios tipos de comportamientos comunes que heredan los otros agentes. La bici es una subclase que se extiende de la persona (P3, 1) y el auto una subclase de la bici (P2, 1), esto quiere decir que la bici hereda cualidades y funciones de la persona y el auto de la bici. En esta versión todos los agentes se mueven libremente (P6, 28-35), cambiando natural, continua y aleatóreamente de rumbo. La diferencia está en los límites de velocidad máxima de cada uno; la persona (P6, 16) es la más restringida, seguida de la bici (P3, 5) y finalmente el auto (P2, 4).



4. FRONTERAS

Así como la primera versión, las restricciones de movimiento para la persona (P6, 50-63) no existen, o bien, cada vez que está cruza un borde aparece por el otro lado a la misma velocidad, la diferencia esta vez está en que las bicicleta y el auto heredan esta función.

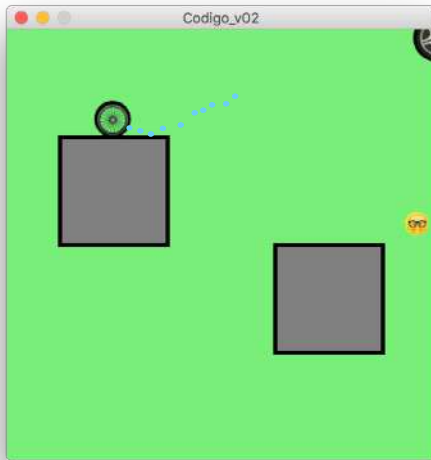
El atravesar las paredes crea da a entender que la ventana del programa es solo un fragmento de un espacio mayor. Esto es porque Processing tiene un espacio de programa infinito; y es esta una cualidad que se podría usar para referirse a un área mayor, como lo es una ciudad.



5.1 MURALLAS

En caso de que una persona se encuentra con una pared cambia de dirección en el eje en el que se aproximó (P6, 65-122); es decir, si toca alguna muralla horizontal (P4, 1-26) la persona rebota en el eje vertical, y si toca una muralla vertical en el eje horizontal. Las bicis y los autos heredan esta función de la persona.

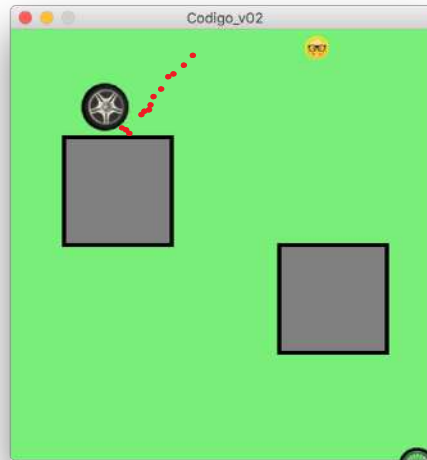
El lograr codificar la función de las murallas de manera programática dará paso a poder ingresar los mapas reales de forma vectorial y automática, acelerando el proceso técnico y la posibilidad de cambiar de locación en simulador.



5.2 MURALLAS

En la captura superior se ve como la función de los choques de murallas escritas en la clase de la persona afectan en la bicicleta, la subclase.

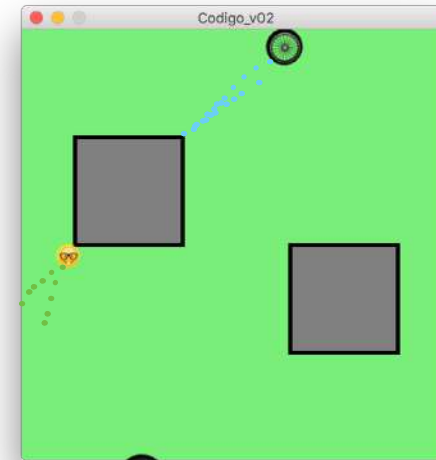
Para todos los agentes la función de las murallas podrían afectar de distintas manera, así como en la realidad, las personas pueden llegar a pasar más cerca de los murallas en comparación a las bicicletas, y los autos suelen avanzar aún más lejos que los dos. Detalles como estos pueden hacer de la experiencia del simulador algo más real y natural.



5.3 MURALLAS

Así como la bicicleta heredó la función de contacto de la persona con las murallas, el auto la heredó de la bici.

Estos conjuntos de murallas son edificios o manzanas que hacen aparecer naturalmente algunos flujos y diferentes formas de interacción y movimiento. Lograr introducirlos de una forma programática le permitiría al usuario poder recrear los flujos y movimientos reales de la ciudad según sus propias voluntades e intenciones como funcionarios de gobierno y/o planificadores urbanos.

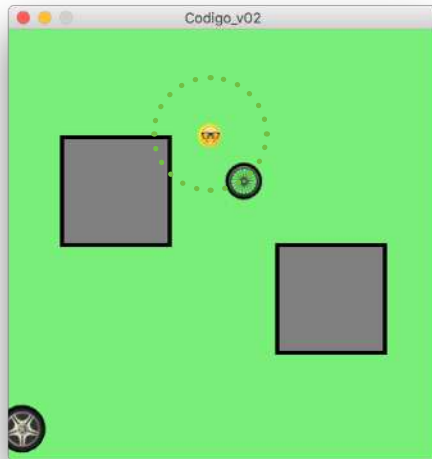


5.4 MURALLAS

En el caso de chocar contra una esquina se suman los rebotes verticales y horizontales de los agentes y resulta en una dirección revertida; rebota y se vuelve por el camino por el cual venía.

Este error de código de repulsión quizás podría evitarse al tener una calle con poder de atracción, en vez de muros con repulsión.

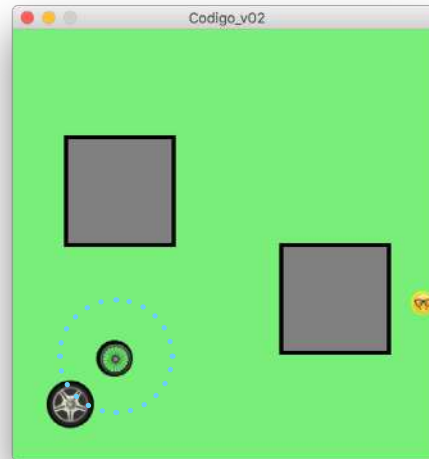
Hay que sopesar lo que se vea más natural para el usuario y lo que funcione de manera menos costosa para el computador que corre el simulador.



6.1 ENCUENTRO

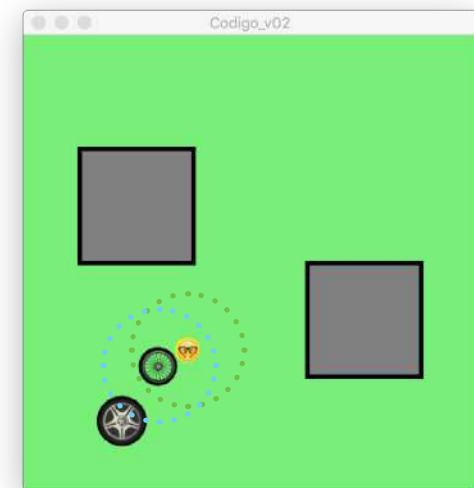
En la versión anterior las bicicletas al encontrarse con las personas ‘rebotaban’, en esta versión la bicicletas al entrar en los 80 píxeles de espacio personal (P3, 31-46) desaceleran y esperan que la persona siga su rumbo hasta quedar fuera del rango de riesgo.

Cómo fue mencionado anteriormente, tal distancia de encuentro podría ser un factor relevante a controlar por el usuario. Este valor variable e interactivo, permitiría marcar con mayor o menor fuerza las jerarquías revelando información crucial del ecosistema de transporte.



6.2 ENCUENTRO

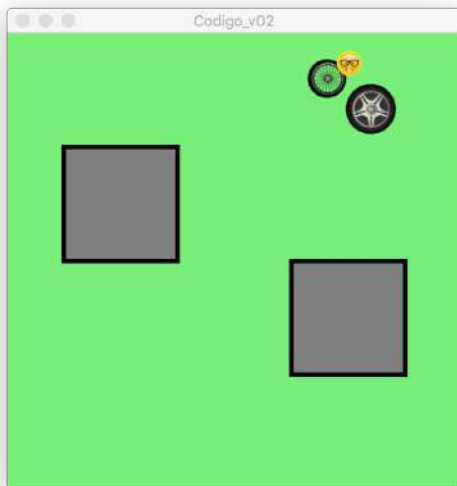
Las personas son a las bicis como las bicis a los autos; los autos al encontrarse con una bicicleta o una persona se detienen (P2, 56-68) y esperan a que estas avancen evitando así pasarlas a llevar. En el caso de arriba el auto espera que la bici se mueva para poder seguir moviéndose libremente.



6.3 ENCUENTRO

En casos donde se encuentran los tres se aplican las mismas condiciones de encuentro y que si estuvieran de a par. En el caso de arriba la bicicleta espera que la persona se mueva para moverse y el auto espera que la bici se mueva para moverse. En estos casos se clarifica la jerarquía entre cada una de las clases.

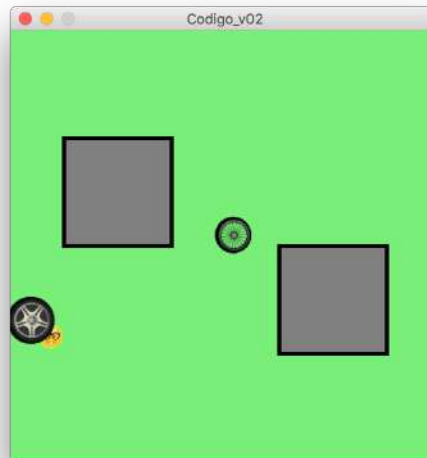
Quizás si se controlase de manera interactiva la distancia de “respeto” para cada tipo de agente se podría, a gusto del usuario, acentuar la jerarquía en pos de mejorar la sustentabilidad, la cual se debería poder medir con algún indicador.



6.4 ENCUENTRO

Las bicis no se pueden mover si está una persona muy cerca, pero lo que si puede hacer una persona es acercarse más e incluso pasar por 'encima'.

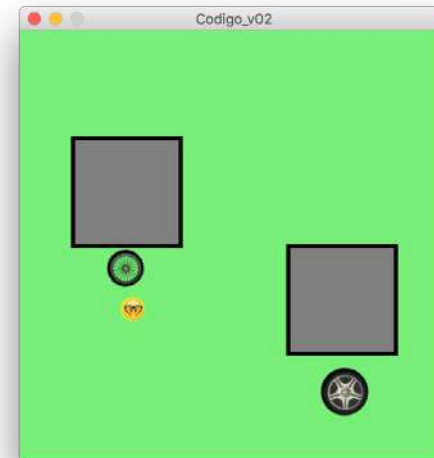
En este caso la persona imposibilita cualquier tipo de movimiento de los otros agentes por cercanía; una conducta antinatural. El usuario con un controlador debería poder elegir si al momento de violar el espacio personal de un agente de mayor rango fuese necesario detenerse, o tan solo restringir la velocidad máxima de el infractor.



6.5 ENCUENTRO

Parecida a la situación anterior el auto es en estos casos el que se queda esperando que la persona lo atraviese.

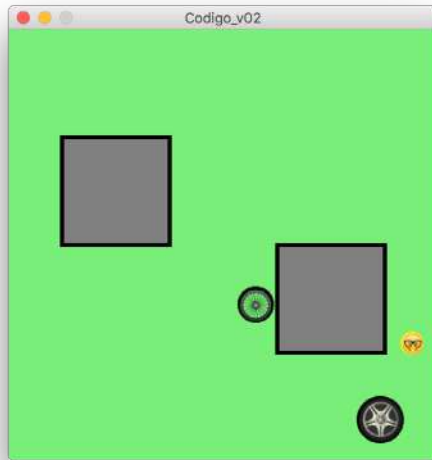
En estos casos donde los agentes se ven obligados a detenerse se ven limitados en 360° a moverse, sin importar la dirección en la que va tanto la persona como el auto. Quizás el usuario debería poder determinar el ángulo de visión de los agentes, permitiendo así que si todos se mueven en una misma dirección no se vean obligados a frenar por completo, si no que a unirse a la masa cual cardumen de peces.



7.1 MIX DE FUNCIONES

En este caso la bici queda 'chocando' de manera estática contra el muro debido a la presencia de la persona quien le quita la posibilidad de movimiento.

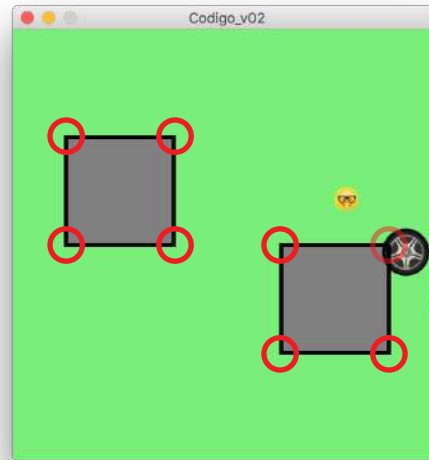
Está bien para el simulador lograr jerarquizar el sistema de tránsito y sus modos según sustentabilidad, más es un exceso detener los flujos, no es lo que se busca, ni lo que el usuario espera. Hay que encontrar el modo de mantener el flujo de usuarios sin que se vea privada la pirámide invertida de prioridad.



7.2 MIX DE FUNCIONES

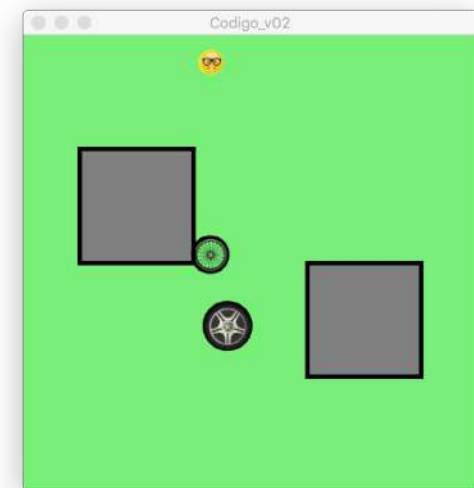
Las funciones están activas y 'corriendo' de manera simultánea, esto quiere decir que si hay un caso de encuentro no significa que el agente libre, como es el caso de arriba, no pueda estar chocando, o más ampliamente interactuando con otro elemento, como una muralla por ejemplo.

La naturaleza emergente del simulador debería poder permitirle a cada uno de los agentes actuar de manera local para determinar su siguiente movimiento. Solo de esta manera aparecerá el orden a nivel global que resulta ser más complejo comparado con la particularidad.



8.1 ERRORES

Hay 4 puntos muertos del tamaño de 1 píxel cada uno. Estos corresponden a las cuatro esquinas de las murallas. Si las personas u otro agente entra exactamente por aquellos puntos a los muros estos quedan atrapados dentro de los muros rebotando en el mismo lugar. Este error se da hasta que los agentes salen por donde entraron.



8.2 ERRORES

En el caso de arriba se aprecia una bicicleta atrapada en el muro que detuvo a un auto debido a lo mucho que se acercó.

La función de los muros parece ser compleja y si un agente no puede mantenerse fuera quizás al replicar la cantidad que son, los muros, el programa del simulador podría colapsar; es decir, posiblemente no es escalable, una propiedad clave de los sistemas emergentes.

VERSIÓN 02

reflexión y proyección
sobre simulador y
su programa de código

CONCLUSIÓN GENERAL

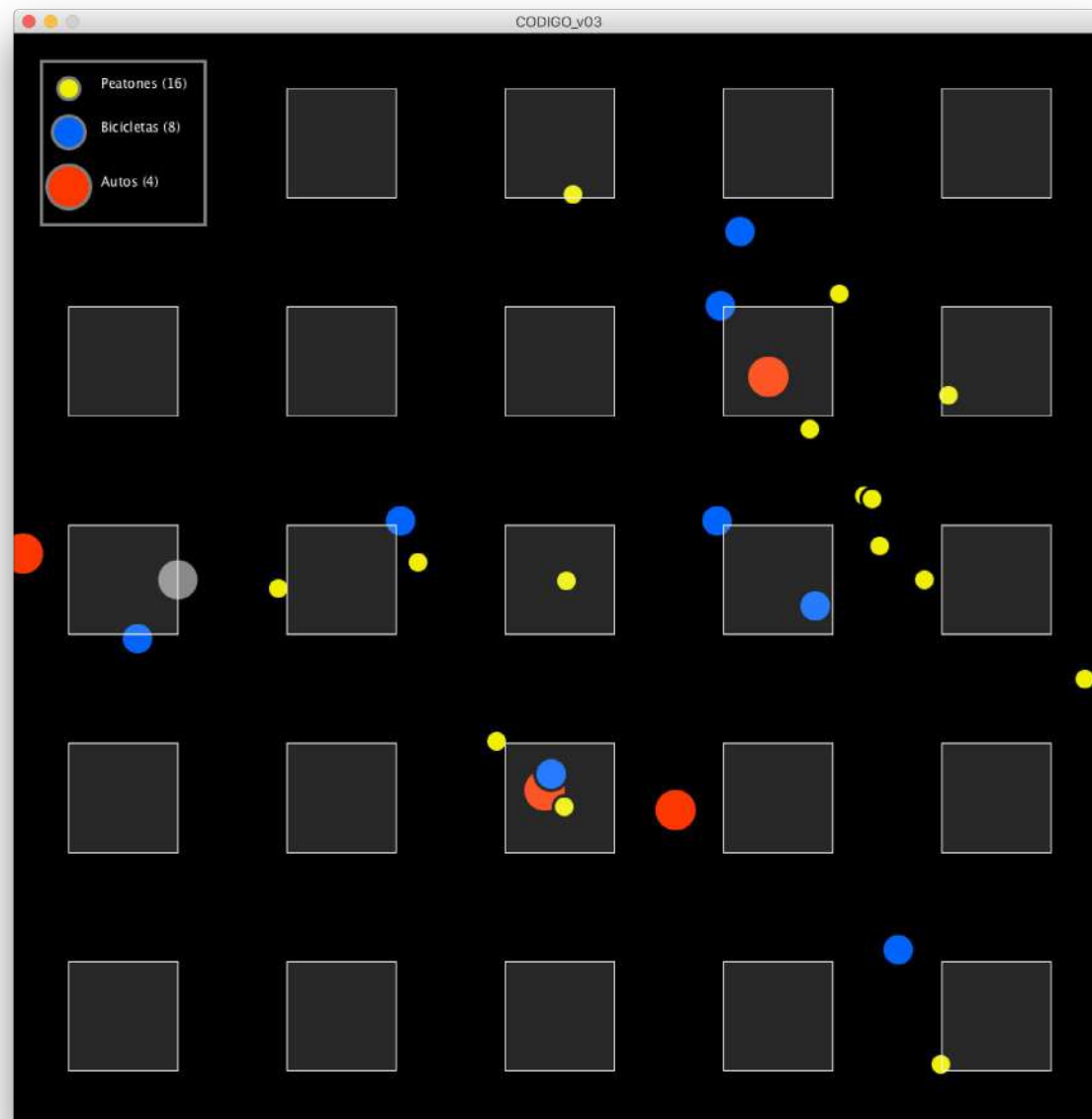
Las figuras pudiesen distinguirse más en pos de una mejor observación a través de cambios de aspecto, u apariencia, tipo de desplazamiento y también comportamiento entre sí y con otros objetos. Es también necesario tener en consideración que estos agentes debieran responder a los diferentes motivos de por qué se trasladan en un principio, es por esto que es pertinente, quizás no obligatoriamente necesario, agregar puntos de origen definidos y controlados para estos. Para aportar en la distinción de los movimientos se podría cambiar el tipo de movimiento de la bicicleta y el auto para que parezcan más naturales y muestren un poco más de impredecibilidad; así como lo hace la persona. Es para esto que se podrían eliminar las fronteras de la ventana del programa y agregar obstáculos u otros elementos que se condigan con la realidad de la infraestructura urbana. La reacción de las clases, el rebote de los agentes, se muestra brusca y poco natural, esta se podría ajustar por medio de lo nuevas morfologías de movimientos de estos.

PRÓXIMOS PASOS

1. Se diferenciarán más la velocidades de los distintos tipos de agentes para mejorar la distinción entre ellos para el usuario.
2. La morfología, el tipo de movimiento, de estos mismos se distinguirá más por medio del factor controlado del azar. Este factor puede ser controlado manualmente de manera interactiva en alguna futura versión.
3. Se incluirán objetos a modo de obstáculos para acercarse más a distintas infraestructuras y los comportamientos que estos significan; un paso esencial pensando en que en versiones venideras se usará un mapa real de la ciudad.
4. Modificar la gráfica; con esto se refiere a los tamaños y formas y aspectos. Puede ser más bello y autoexplicativo por medio de mejoras de composición
5. Agregar puntos de origen y de destino para recrear flujos urbanos reales.
6. Evitar que agentes crucen entre ellos. En caso de encuentros limitar las velocidades máximas, en vez de frenar, de los agentes de forma interactiva.

VERSIÓN 03

la agrandá'




```

CODIGO_v03 | Processing 3.3.5
CODIGO_v03
1 // LISTAS DE AGENTES //
2 ArrayList<Persona> peatones;
3 ArrayList<Bici> ciclistas;
4 ArrayList<Auto> conductores;
5 Edificio[][] block;
6 Atractor att;
7
8 //PARAMETROS DEL BLOCK //
9 int filas = 5;
10 int columnas = 5;
11 int calle = 2; // = ancho edificio
12 int x1Edif = 100;
13 int y1Edif = 100;
14 int x2Edif = 100;
15 int y2Edif = 100;
16
17 //////////////////////////////////////
18 /*** S E T U P ***/////////////////////////////////////
19 //////////////////////////////////////
20 void setup() {
21     size (1000, 1000);
22
23     // SETEO DE AGENTES //
24     peatones = new ArrayList<Persona>();
25     ciclistas = new ArrayList<Bici>();
26     conductores = new ArrayList<Auto>();
27     att = new Atractor();
28
29     // SETEO DEL BLOCK //
30     block = new Edificio[filas][columnas];
31     for (int i = 0; i < filas; i++) {
32         for (int j = 0; j < columnas; j++) {
33             block[i][j] = new
34                 Edificio(i*calle*x1Edif,
35                     j*calle*y1Edif, x2Edif, y1Edif);
36         }
37     }
38 }
39
40 //////////////////////////////////////

```

Pestaña 1: Programa 1/6

```

CODIGO_v03 | Processing 3.3.5
CODIGO_v03
40 //////////////////////////////////////
41 /*** D R A W ***/////////////////////////////////////
42 //////////////////////////////////////
43 void draw() {
44     background(0);
45
46     // FUNCION DEL ATRACTOR
47     att.display();
48
49     // ORIGENES DE PERSONAS,
50     // BICIS Y AUTOS
51     int r1 = 200*(int(random(2, 5)));
52     int r2 = 200*(int(random(2, 5)));
53     int randomstartxp = -100 +
54         int(random(r1, r1)); // Origen random
55     int randomstartyp = -100 +
56         int(random(r2, r2));
57
58     peatones.add(new Persona(randomstartxp
59                             randomstartyp))
60     ciclistas.add(new Bici(randomstartxp,
61                             randomstartyp));
62     conductores.add(new Auto(randomstartxp
63                               randomstartyp))
64
65     // CANTIDADES DE AGENTES
66     if (peatones.size() > 16) {
67         peatones.remove(16);
68     }
69     if (ciclistas.size() > 8) {
70         ciclistas.remove(8);
71     }
72     if (conductores.size() > 4) {
73         conductores.remove(4);
74     }
75
76     // CHOQUES ENTRE AGENTES
77     for (int i = peatones.size()-1;
78           i >= 0; i--) {
79         for (int j = ciclistas.size()-1;

```

Pestaña 1: Programa 2/6

```

CODIGO_v03 | Processing 3.3.5
CODIGO_v03
77     for (int i = peatones.size()-1;
78           i >= 0; i--) {
79         for (int j = ciclistas.size()-1;
80           j >= 0; j--) {
81             for (int k = conductores.size()-1;
82                   k >= 0; k--) {
83                 Persona p = peatones.get(i);
84                 Bici b = ciclistas.get(j);
85                 Auto a = conductores.get(k);
86
87                 // (BICI)
88                 if (b.choca(p)) { //Bici vs Peaton
89                     b.velocity.x =
90                         constrain(b.velocity.x, -0.2, 0.2);
91                     b.velocity.y =
92                         constrain(b.velocity.y, -0.2, 0.2);
93                 }
94
95                 // (AUTO)
96                 if (a.choca(p)) { //Auto vs Peaton
97                     a.velocity.x =
98                         constrain(a.velocity.x, -0.2, 0.2);
99                     a.velocity.y =
100                         constrain(a.velocity.y, -0.2, 0.2);
101                 }
102                 if (a.choca(b)) { //Auto vs Bici
103                     a.velocity.x =
104                         constrain(a.velocity.x, -0.2, 0.2);
105                     a.velocity.y =
106                         constrain(a.velocity.y, -0.2, 0.2);
107                 }
108             }
109         }
110     }
111
112     // FUNCIONES AUTOS
113     for (int i =
114           conductores.size()-1; i >= 0; i--) {
115         Auto a = conductores.get(i);
116         a.update();

```

Pestaña 1: Programa 3/6

```

CODIGO_v03 | Processing 3.3.5
CODIGO_v03
112 // FUNCIONES AUTOS
113 for (int i =
114 conductores.size()-1; i >= 0; i--) {
115     Auto a = conductores.get(i);
116     a.update();
117     a.display();
118
119     // TENDENCIA HACIA LA DERECHA
120     PVector inclinacion =
121     new PVector(100, 0);
122     // Fuerza del "Viento" (x,y)
123     a.applyForce(inclinacion);
124
125     // SI SALE DE LA VENTANA MUERE
126     if (a.outOfBounds()) {
127         conductores.remove(i);
128     }
129 }
130
131 // FUNCIONES BICIS
132 for (int i =
133 ciclistas.size()-1; i >= 0; i--) {
134     Bici b = ciclistas.get(i);
135     b.update();
136     b.display();
137
138     // TENDENCIA HACIA LA DERECHA
139     PVector inclinacion =
140     new PVector(100, 0);
141     // Fuerza del Viento (x,y)
142     b.applyForce(inclinacion);
143
144     // SI SALE DE LA VENTANA MUERE
145     if (b.outOfBounds()) {
146         ciclistas.remove(i);
147     }
148 }
149
150 // FUNCIONES PERSONAS
151 for (int i =

```

Pestaña 1: Programa 4/6

```

CODIGO_v03 | Processing 3.3.5
CODIGO_v03
148 }
149
150 // FUNCIONES PERSONAS
151 for (int i =
152 peatones.size()-1; i >= 0; i--) {
153     Persona p = peatones.get(i);
154     p.update();
155     p.display();
156
157     // TENDENCIA HACIA LA DERECHA
158     PVector inclinacion =
159     new PVector(100, 0);
160     // Fuerza del Viento
161     p.applyForce(inclinacion);
162
163     // HAMBRE, IR POR SNACK
164     PVector hambre = att.attract(p);
165     p.applyForce(hambre);
166
167     // SI SALE DE LA VENTANA MUERE
168     if (p.outOfBounds()) {
169         peatones.remove(i);
170     }
171 }
172
173 // FUNCIONES DE c/ EDIFICIO
174 for (int i = 0; i < filas; i++) {
175     for (int j = 0; j < columnas; j++) {
176         block[i][j].build();
177     }
178 }
179
180 // SIMBOLOGIA
181 pushMatrix();
182 translate(25, 25);
183 stroke(128);
184 strokeWeight(3);
185 fill(0);
186 rect(0, 0, 150, 150);
187 fill(240, 240, 0);

```

Pestaña 1: Programa 5/6

```

CODIGO_v03 | Processing 3.3.5
CODIGO_v03
161     p.applyForce(inclinacion);
162
163     // HAMBRE, IR POR SNACK
164     PVector hambre = att.attract(p);
165     p.applyForce(hambre);
166
167     // SI SALE DE LA VENTANA MUERE
168     if (p.outOfBounds()) {
169         peatones.remove(i);
170     }
171 }
172
173 // FUNCIONES DE c/ EDIFICIO
174 for (int i = 0; i < filas; i++) {
175     for (int j = 0; j < columnas; j++) {
176         block[i][j].build();
177     }
178 }
179
180 // SIMBOLOGIA
181 pushMatrix();
182 translate(25, 25);
183 stroke(128);
184 strokeWeight(3);
185 fill(0);
186 rect(0, 0, 150, 150);
187 fill(240, 240, 0);
188 ellipse(25, 25, 20, 20); // Persona
189 fill(0, 100, 255);
190 ellipse(25, 65, 30, 30); // Bici
191 fill(255, 56, 0);
192 ellipse(25, 115, 40, 40); // Auto
193 fill(255);
194 text("Peatones (16)", 55, 25);
195 text("Bicicletas (8)", 55, 65);
196 text("Autos (4)", 55, 115);
197 popMatrix();
198 } ////////////////////////////////////////////////// fin draw
199
200

```

Pestaña 1: Programa 6/6

```

CODIGO_v03 | Processing 3.3.5
CODIGO_v03
1 class Atractor {
2   float mass;
3   float G;
4   PVector posicion;
5
6   // CONSTRUCTOR DEL ATRACTOR
7   Atractor() {
8     posicion = new PVector(150,
9       height/2);
10    mass = 20; // TAMAÑO
11    G = 10;
12  }
13
14  // FUNCION DE ATRACCION PERSONAS
15  PVector attract(Persona fulana) {
16    // Calculate direction of force
17    PVector hambre =
18      PVector.sub(posicion,
19        fulana.position);
20    // Distance between objects
21    float d = hambre.mag();
22    // Limiting the distance
23    d = constrain(d, 5.0, 250.0);
24    hambre.normalize();
25    // Calculate gravitational force magni
26    float strength =
27      (G * mass * fulana.mass) / (d * d);
28    hambre.mult(100*(strength));
29    return hambre;
30  }
31
32  // FUNCION DE APARIENCIA
33  void display() {
34    ellipseMode(CENTER);
35    strokeWeight(4);
36    stroke(0);
37    fill(175, 200);
38    ellipse(posicion.x, posicion.y,
39      mass*2, mass*2);
40  }}

```

Pestaña 2: Atractor

```

CODIGO_v03 | Processing 3.3.5
CODIGO_v03
1 class Auto extends Bici {
2   Auto(float x_, float y_) {
3     super(x_, y_);
4     topspeed = 3;
5
6     // TEXTURA DE RUEDA
7     // HACE MUY PESADO EL PROGRAMA :)
8     //foto = loadImage("auto.png");
9   }
10
11  // APARIENCIA
12  void display() {
13    diam = 40;
14    stroke(0);
15    strokeWeight(3);
16    fill(255, 56, 0);
17    ellipse(position.x, position.y,
18      diam, diam);
19  }
20
21  // DESPLAZAMIENTO,
22  // VELOCIDAD Y ACELERACION
23  void update() {
24    acceleration = PVector.random2D();
25    acceleration.mult(0.2);
26    velocity.add(acceleration);
27    velocity.limit(topspeed);
28    position.add(velocity);
29  }
30
31  // CHOQUE CON PERSONA
32  boolean choca(Persona p) {
33    float d = abs(dist(position.x,
34      position.y, p.position.x,
35      p.position.y));
36    if (d <= 50) {
37      return true;
38    } else {
39      return false;
40    }
41  }
42
43  // CHOQUE CON BICI
44  boolean choca(Bici b) {
45    float d = abs(dist(position.x,
46      position.y, b.position.x,
47      b.position.y));
48    if (d <= 50) {
49      return true;
50    } else {
51      return false;
52    }
53  }
54 }
55
56

```

Pestaña 3: Auto 1/2

```

CODIGO_v03 | Processing 3.3.5
CODIGO_v03
17   ellipse(position.x, position.y,
18     diam, diam);
19 }
20
21 // DESPLAZAMIENTO,
22 // VELOCIDAD Y ACELERACION
23 void update() {
24   acceleration = PVector.random2D();
25   acceleration.mult(0.2);
26   velocity.add(acceleration);
27   velocity.limit(topspeed);
28   position.add(velocity);
29 }
30
31 // CHOQUE CON PERSONA
32 boolean choca(Persona p) {
33   float d = abs(dist(position.x,
34     position.y, p.position.x,
35     p.position.y));
36   if (d <= 50) {
37     return true;
38   } else {
39     return false;
40   }
41 }
42
43 // CHOQUE CON BICI
44 boolean choca(Bici b) {
45   float d = abs(dist(position.x,
46     position.y, b.position.x,
47     b.position.y));
48   if (d <= 50) {
49     return true;
50   } else {
51     return false;
52   }
53 }
54 }
55
56

```

Pestaña 3: Auto 2/2

```

CODIGO_v03 | Processing 3.3.5
class Bici extends Persona {
  Bici(float x_, float y_) {
    super(x_, y_);
    topspeed = 1.5;
  }

  // APARIENCIA
  void display() {
    diam = 30;
    stroke(0);
    strokeWeight(3);
    fill(0, 100, 255);
    ellipse(position.x, position.y,
            diam, diam);
  }

  // DESPLAZAMIENTO,
  // VELOCIDAD Y ACELERACION
  void update() {
    acceleration = PVector.random2D();
    acceleration.mult(0.2);
    velocity.add(acceleration);
    velocity.limit(topspeed);
    position.add(velocity);
  }

  // INTERACCION CON PERSONA
  boolean choca(Persona p) {
    float d = abs(dist(position.x,
                      position.y, p.position.x,
                      p.position.y));

    if (d <= 50) {
      return true;
    } else {
      return false;
    }
  }
}

```

Pestaña 4: Bici

```

CODIGO_v03 | Processing 3.3.5
class Edificio {
  int xA;
  int yA;
  int xB;
  int yB;

  // CONSTRUCTOR EDIFICIO
  Edificio(int x1_, int y1_,
          int x2_, int y2_) {
    xA = x1_;
    yA = y1_;
    xB = x2_;
    yB = y2_;
  }

  // FUNCION DE FUNCIONES
  void run() {
    build();
  }

  // APARIENCIA EDIFICIO
  void build() {
    pushMatrix();
    translate(50, 50);
    fill(255,40);
    stroke(255);
    strokeWeight(1);
    rect(xA,yA,xB,yB);
    popMatrix();
  }
}

```

Pestaña 5: Edificio

```

CODIGO_v03 | Processing 3.3.5
class Persona {
  PVector position;
  PVector velocity;
  PVector acceleration;

  float topspeed;
  int diam;
  float lifespan = 200;
  float mass;

  // POSICION DE ORIGEN,
  // VELOCIDAD Y LIMITE DE VELOCIDAD
  Persona(float x_, float y_) {
    float ppx = x_;
    float ppy = y_;
    position = new PVector(ppx, ppy);
    velocity = new PVector(random(0, 1),
                          random(-1, 1))
    topspeed = 1.5; // max. velocidad
    mass = 100; // peso
  }

  // APLICADOR DE FUERZA
  void applyForce(PVector force) {
    PVector f = PVector.div(force, mass)
    acceleration.add(f);
  }

  // FUNCION DE FUNCIONES
  void run() {
    update();
    display();
    //reboteEdificio();
  }

  // DESPLAZAMIENTO,
  // VELOCIDAD Y ACELERACION
  void update() {
    acceleration = PVector.random2D();
    acceleration.mult(random(0.07));
  }
}

```

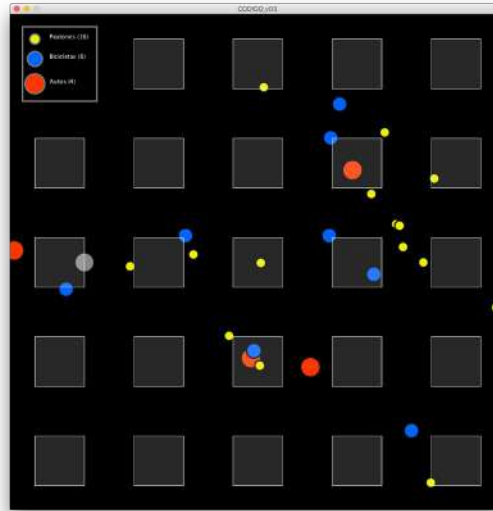
Pestaña 6: Persona 1/2

```
CODIGO_v03 | Processing 3.3.5
CODIGO_v03
30 void run() {
31   update();
32   display();
33   //reboteEdificio();
34 }
35
36 // DESPLAZAMIENTO,
37 // VELOCIDAD Y ACELERACION
38 void update() {
39   acceleration = PVector.random2D();
40   acceleration.mult(random(0.07));
41   velocity.add(acceleration);
42   velocity.limit(topspeed);
43   position.add(velocity);
44 }
45
46 // APARIENCIA
47 void display() {
48   diam = 20;
49   stroke(0);
50   strokeWeight(3);
51   fill(240, 240, 0);
52   ellipse(position.x, position.y,
53           diam, diam);
54 }
55
56 // SI SALE DE LA VENTANA MUERE
57 boolean outOfBounds() {
58   if ((position.x > width + diam/2)
59       || (position.x < 0 - diam/2)
60       || (position.y > height + diam/2)
61       || (position.y < 0 - diam/2)) {
62     return true;
63   } else {
64     return false;
65   }
66 }
67 }
68
69
```

Pestaña 6: Persona 2/2

VERSIÓN 03

funcionamiento,
observaciones y
conclusiones pensadas
en el uso y control
del usuario

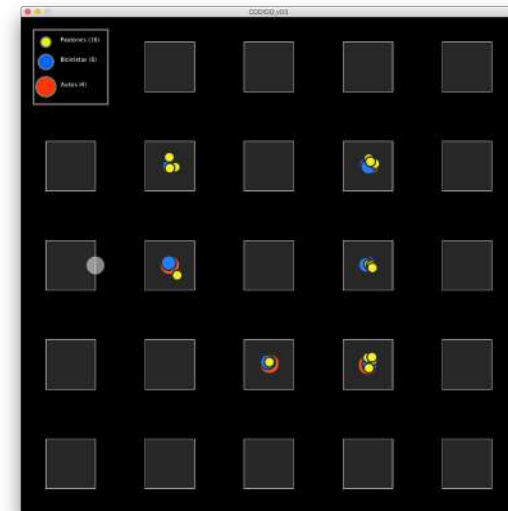


1. APARIENCIA

Con la intención de permitir mayor nitidez, o bien, detalle para por el programa se aumentó el tamaño de la ventana hasta **1000 x 1000**, el máximo reproducible en pantalla.

Además se cambió la paleta de claros y chillones a colores oscuros de fondo y colores chillones para los agentes quienes deben destacar.

Los agentes eran todos elipses debido a que aún no se lograba representar el giro y la dirección del agente. Característica clave para que el usuario logre entender el flujo y dirección de los agentes; de dónde vienen y a dónde van.

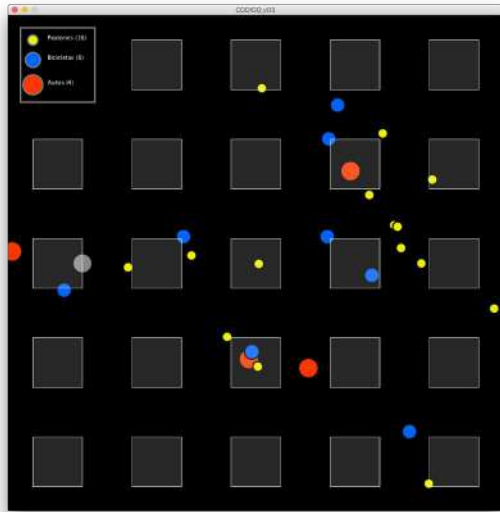


2. PUNTOS DE ORIGEN

Los puntos de origen fueron descritos en una grilla de coordenadas x e y que corresponden a los centros de las manzanas del centro del programa. Cada uno de los agentes tiene la misma probabilidad de aparición en cada uno de los puntos.

Los orígenes fueron agregados de manera programática en el centro de las manzanas de la grilla. Más no representan las posibilidades deseadas por el usuario de representar los flujos reales.

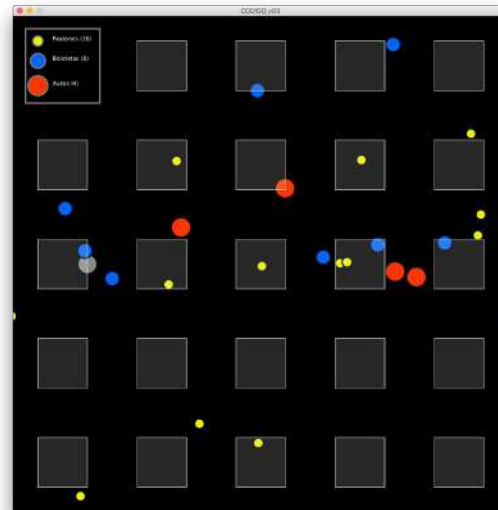
La forma de cuadrícula aun no tiene la adaptabilidad necesaria para que el usuario logre elegir otro lugar.



3. MUROS

Al crecer la cantidad de agentes por tipo cambió el modo interacción y programación entre cada uno de ellos y los obstáculos, las manzanas. En esta versión no se logró hacer que cada uno de los agentes lograra evadir o interactuar con la grilla de manzanas sin perjudicar la velocidad del programa. Al aumentar los obstáculos y los móviles aumenta exponencialmente la cantidad de procesos por los que pasa el computador.

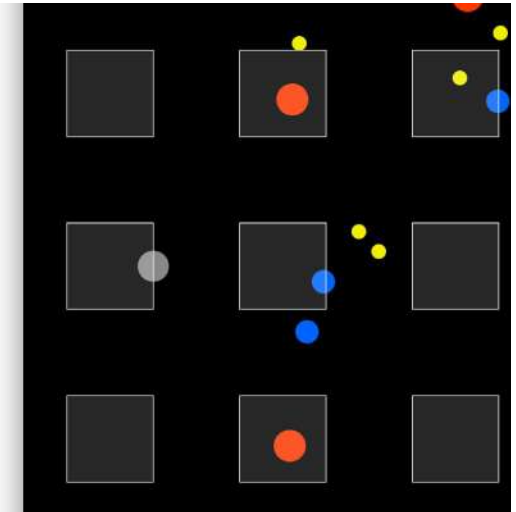
Hay que hallar una forma en la que sea posible importar mapas externos para facilitar el uso del programa en diferentes lugares deseados y controlados por el usuario.



4. FRONTERAS

En esta versión en vez de hacer que los agentes al salir por un lado entren por el otro, cual pasillo de PacMan, quienes salen son eliminados del programa, para aliviar a este último, y aparece otro nuevo agente de su mismo tipo (persona, ciclista, pasajero de auto) para reemplazarlo desde alguno de los puntos de origen.

Considerando el nivel de costo programático es una buena solución eliminar a los agentes que no vemos, solo falta que algunos de ellos puedan llegar desde fuera. El poder salir y entrar en del simulador sugiere un tamaño mayor de mapa; no son fronteras, si no un marco.

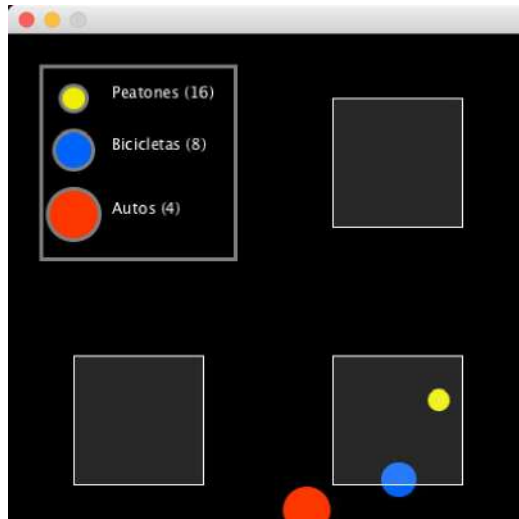


5. ATRACTOR

Haciendo uso de fórmulas de atracción gravitacional comunes, aparentemente lo más simple según los tutoriales de Daniel Shiffman, no se logró ningún tipo de movimiento de acercamiento al punto atractor que estaba en el mapa (la elipse gris).

Quizás la gravedad pudiese estar descrita en alguna librería de algún modo más fácil de aplicar en el programa.

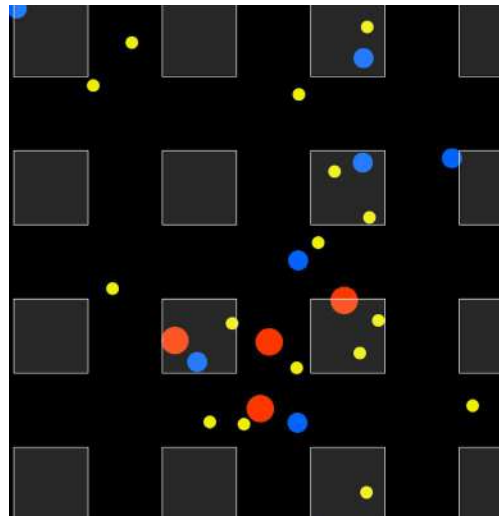
Los atractores, sean modificables o no por el usuario, son necesarios para: recrear flujos, darle dirección y sentido a los agentes, generar interacción y cruces, darles un objetivo, un propósito.



6. SIMBOLOGÍA

La simbología contenía pequeños dibujos representativos de los distintos tipos de agentes acompañados del nombre de su categoría y la cantidad total de agentes dispersos.

Es absolutamente necesario, en el estado actual del simulador, poner una simbología, dado que al ser solamente unas elipses la comprensión de que agente es que tipo es mejorable. El usuario en un futuro podría quizás editar la cantidad marcada en la simbología para elegir la cantidad x, y, z de agentes en el simulador. Modificando así las densidades, flujo y sustentabilidad de tal caso particular del ecosistema.



7. ENCUENTROS

Se logró limitar las velocidades máximas en caso de encuentro en vez de reducirlas a 0, sin embargo hay casos donde agentes, en vez de mantener una separación, se traspasan y cruzan sin criterio.

Falta, para lograr una mejor naturalidad en el comportamiento de los agentes una función de separación entre ellos, sean del mismo tipo de modo o no. Sería un error no reflejar las capacidades seguras de mantener distancia que tienen los autos autónomos; tecnología base que sostiene el sistema emergente propuesto en el simulador.

VERSIÓN 03

reflexión y proyección

CONCLUSIÓN GENERAL

Los puntos de origen ubicados bajo las manzanas es útil pero no extrapolable a otros tipos de mapas, hay que buscar una mejor manera de repartir a los agentes.

Gracias a las listas (arrays) se puede controlar perfectamente las cantidades y porcentajes de agentes que se desplazan por el programa, ahora habría que modificar cuantos grupos de cada tipos de agente podrían haber. Se destaca la flexibilidad de los arrays por que permiten la escalabilidad.

Se eliminaron texturas para hacer el programa mas rápido con éxito. Al simplificar paleta de colores también se mejoró la lectura de las situaciones.

En caso de encuentros en vez neutralizar a los agentes más bajos en categoría según la jerarquía, se limitaron sus velocidades máximas, cosa de desacelerar.

Se adhirió un recuadro con una pequeña simbología que permitía entender que color correspondía a tal agente Este sería uno de los primero pasos que encaminarían el programa hacia la usabilidad.

PRÓXIMOS PASOS

1. Agregar un destino, una dirección para los agentes. Esta función debe ser en las versiones siguientes interactiva.
2. Tales atractores debiesen ser de distintos tipos para cada agente: por la necesidad de moverse a diferentes niveles: cuadra, barrio y comuna, para los peatones, ciclistas y autos respectivamente. Las interacciones y flujos apareceran naturalmente al seguir objetivos.
3. Repulsores, objetos de colisión; pensando en los edificios, se podrían usar los códigos ya escritos en librerías de Processing, los cuales usan fuerza de gravedad o atracción por uniones tipo resortes.
4. Usar un mapa real de fondo, un contexto y una problemática real. Y agregar un mapa sobre la interfaz para entender la ubicación de los atractores. Usando como referencia el mini mapa de Google Maps, el usuario podría ubicarse geográficamente.
5. Modificar la escala a una real. Definir cuantos píxeles equivalen a cuantos metros. Adelantándose al uso de un mapa a escala real, se debe usar una escala versátil en el simulador.

LIBRERÍAS DE MOTORES FÍSICO

un largo desvío sin mucha ganancia

Creyendo que el próximo paso sería el uso de librerías se investigó sobre cuáles serían las más adecuadas para poder simular eventos físicos. Se estudiaron 2 principales librerías de motores físicos que podían satisfacer ciertas situaciones fundamentales que aun no se lograban desarrollar en el código; estas eran BOX2D y Toxic Libs (de Daniel Shifman y Karsten Schmidt respectivamente).

Al final uno de los aspectos más fundamentales respecto a las librerías era el grado de compatibilidad con lo ya desarrollado, y esta es la razón de por qué finalmente estas dos librerías a pesar de ser estudiadas por semanas no se usaron.

Cabe mencionar que se experimentaron varias funciones de las librerías dentro de programas, pero no va al caso entrar en detalles de como funcionaban por que de estas no se rescató casi nada para las siguientes versiones del programa del simulador.

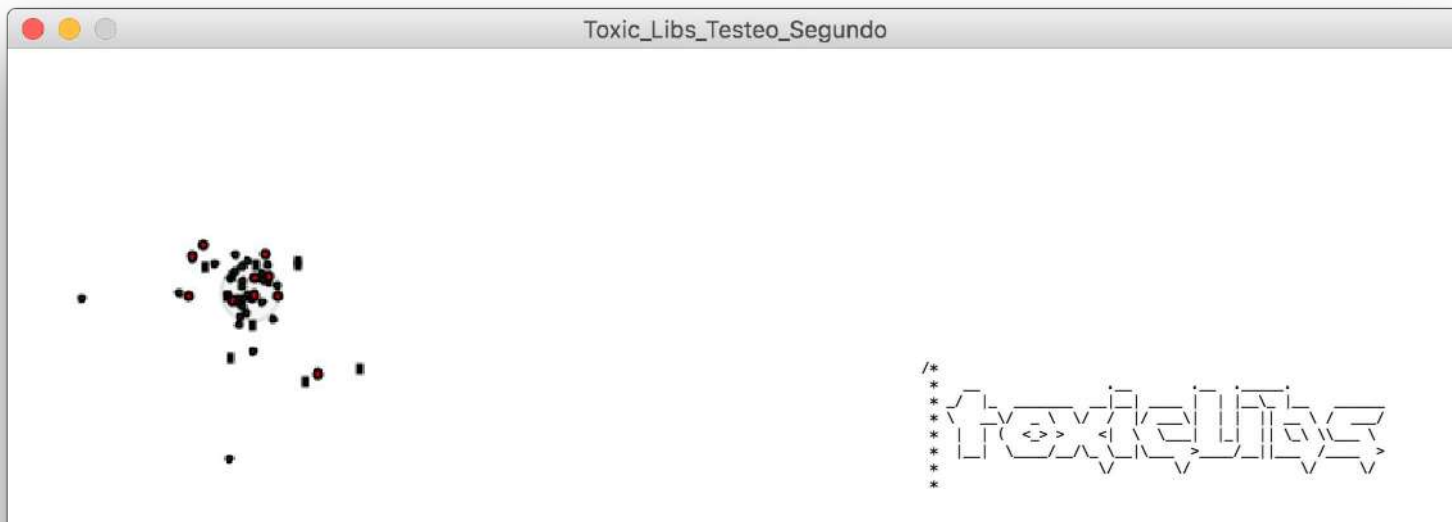
A continuación se presenta una tabla resumen que indica las fortalezas y bondades, así como también las faltas, de las librerías vistas.

Los criterios nacen a partir de lo que se aprendió y también de lo que le es útil al desarrollo del programa. A fin de cuentas la experiencia con cada una de las librerías se resume en que: **BOX2D** permitía un **buen sistema de colisiones y también podría programarse con escala métrica, pero carecía de atractores y repulsores: piezas fundamentales para el origen, destino y los edificios.** **TOXICLIBS** por su parte **si tenía funciones de atractores y repulsores, más no con las colisiones que poseía BOX2D** y que el programa requería.

Ni por si solas, ni la mezcla de ambas lograron satisfacer los requisitos del programa, ni tampoco compatibilizar con las variables ya determinadas: Velocidades, Jerarquía y proxémica. Las librerías están limitadas a hacer lo que hacen y no tiene muchas otras salidas, ni tampoco la cómoda posibilidad de mezclarse unas con otras.

PROS Y CONTRAS DE LAS LIBRERÍAS

	BOX2D	TOXIC-LIBS
Geometría de colisión	1	0
Física en tres dimensiones	0	1
Atracción y Repulsión	0	1
Springs (Resortes)	1	1
Conectores	1	0
Facilidad de uso (1 a 5)	3	5



VERSIÓN 04*

el "patito feo"



* SÁLTESE A LA PÁGINA 84 PARA VER LA SIGUIENTE Y MENOS ÁRIDA SECCIÓN

```

CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
1 // IMPORTACION DE LIBRERIAS
2 import geomerative.*;
3 RShape grp;
4 RPoint[][] pointPaths;
5 boolean ignoringStyles = false;
6
7 // Using this variable to decide
8 // whether to draw all the stuff
9 boolean debug = true;
10
11 // LISTA DE PATHS
12 //Path path1;
13 //Path path2;
14 Path pathLC; // Path de Lo Contador
15
16 // LISTAS DE AGENTES
17 ArrayList<Persona> peatones;
18 ArrayList<Persona> peatones2;
19 ArrayList<Persona> peatones3;
20 ArrayList<Bici> ciclistas;
21
22 // NUMERO MAXIMO DE AGENTES
23 int pMax = 30;
24 int bMax = 40;
25 int aMax = 30;
26
27 // SETUP //
28 void setup() {
29   size(1000, 1000, P2D);
30   smooth();
31   frameRate(25);
32
33   // VERY IMPORTANT: Always initialize
34   // the library before using it
35   RG.init(this);
36   RG.ignoreStyles(ignoringStyles);
37   RG.setPolygonizer(RG.ADAPTATIVE);
38   grp = RG.loadShape("LoContadorMapa.svg");

```

Pestaña 1: Programa 1/8

```

CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
40 grp = RG.loadShape("LoContadorMapa.svg");
41 grp.centerIn(g, 100, 1, 1);
42 pointPaths = grp.getPointsInPaths();
43
44 //newPath1();
45 //newPath2();
46 // CREACION PATH SEGUN FUNCION EN 257
47 newPathLC();
48
49 // SETEO DE AGENTES
50 peatones = new ArrayList<Persona>();
51 peatones2 = new ArrayList<Persona>();
52 peatones3 = new ArrayList<Persona>();
53 ciclistas = new ArrayList<Bici>();
54
55
56 //// SETEO DE ATRACTORES (COORDS)
57 // NO FUNCIONAN SIMULTANEAMENTE!!!
58 //diamA = 15;
59 //cuadra1 = new PVector(0, 0);
60 //barrio1 = new PVector(width, height)
61 //comunal = new PVector(0, height);
62 }
63
64 // D R A W //
65 void draw() {
66   background(0);
67
68   // IDENTIFICADOR POSICION DEL MOUSE
69   println(mouseX, mouseY);
70
71   // SVG SIMPLE, COLOR DE CALLES
72   pushMatrix();
73   translate(width/2, height/2);
74   for (int i = 0;
75     i < pointPaths.length; i++) {
76     if (pointPaths[i] != null) {
77       noFill();
78       stroke(255,40);
79

```

Pestaña 1: Programa 2/8

```

CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
79   stroke(255,40);
80   // % visibilidad Lo Contador
81   strokeWeight(15);
82   beginShape();
83   for (int j = 0;
84     j < pointPaths[i].length; j++) {
85     vertex(pointPaths[i][j].x,
86           pointPaths[i][j].y);
87   }
88   endShape();
89 }
90 }
91 popMatrix();
92
93 // FUNCIONES DEL PATH RANDOM
94 //path1.display();
95 //path2.display();
96 pathLC.display();
97
98 // PATH 1
99 for (Persona p : peatones) {
100   p.follow(pathLC);
101 }
102 for (Bici b : ciclistas) {
103   b.follow(pathLC);
104 }
105
106 // PATH 2
107 for (Persona p : peatones2) {
108   p.follow(pathLC);
109 }
110
111 // PATH 3
112 for (Persona p : peatones3) {
113   p.follow(pathLC);
114 }
115
116 // ORIGENES DE PERSONAS,
117 // Y BICIS Y AUTOS
118 int r1 = 200*(int(random(2, 5)));

```

Pestaña 1: Programa 3/8

```

CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
118 int r1 = 200*(int(random(2, 5)));
119 int r2 = 200*(int(random(2, 5)));
120 int randomstartxp = -100 +
121 int(random(r1, r1)); // random origen
122 int randomstartyp = -100 +
123 int(random(r2, r2));
124
125 peatones.add(new
126 Persona(new PVector(randomstartxp,
127 randomstartyp), 0.5, 0.01));
128 peatones2.add(new
129 Persona(new PVector(randomstartxp,
130 randomstartyp), 0.5, 0.01));
131 peatones3.add(new
132 Persona(new PVector(randomstartxp,
133 randomstartyp), 0.5, 0.01));
134
135 ciclistas.add(new Bici(new PVector(
136 randomstartxp, randomstartyp),
137 1.5, 0.03));
138
139 // LIMITADOR DE CANTIDADES
140 // DE AGENTES POR TIPO
141 if (peatones.size() > pMax) {
142 peatones.remove(pMax);
143 }
144 if (peatones2.size() > pMax) {
145 peatones2.remove(pMax);
146 }
147 if (peatones3.size() > pMax) {
148 peatones3.remove(pMax);
149 }
150 if (ciclistas.size() > bMax) {
151 ciclistas.remove(bMax);
152 }
153
154 // CHOQUES ENTRE AGENTES
155 for (int i = peatones.size()-1;
156 i >= 0; i--)

```

Pestaña 1: Programa 4/8

```

CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
155 // CHOQUES ENTRE AGENTES
156 for (int i = peatones.size()-1;
157 i >= 0; i--) {
158 for (int j = ciclistas.size()-1;
159 j >= 0; j--) {
160
161 Persona p = peatones.get(i);
162 Persona p2 = peatones2.get(i);
163 Persona p3 = peatones3.get(i);
164 Bici b = ciclistas.get(j);
165 //Auto a = conductores.get(k);
166
167 // (BICI)
168 if (b.choca(p)) { //Bici vs Peaton
169 b.velocity.x =
170 constrain(b.velocity.x, -0.2, 0.2);
171 b.velocity.y =
172 constrain(b.velocity.y, -0.2, 0.2);
173 }
174 if (b.choca(p2)) { //Bici vs Peaton
175 b.velocity.x =
176 constrain(b.velocity.x, -0.2, 0.2);
177 b.velocity.y =
178 constrain(b.velocity.y, -0.2, 0.2);
179 }
180 if (b.choca(p3)) { //Bici vs Peaton
181 b.velocity.x =
182 constrain(b.velocity.x, -0.2, 0.2);
183 b.velocity.y =
184 constrain(b.velocity.y, -0.2, 0.2);
185 }
186 }
187 }
188
189 // FUNCIONES DE PERSONAS 1
190 for (int i = peatones.size()-1; i >= 0
191 Persona p = peatones.get(i);
192 // F MAESTRA Y TARGET
193 p.run();
194

```

Pestaña 1: Programa 5/8

```

CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
195 // F DE BORDES
196 if (p.outOfBounds()) {
197 peatones.remove(i);
198 }
199 }
200 // F DE SEPARACION
201 for (Persona p : peatones) {
202 p.separateP(peatones);
203 p.separateP(peatones2);
204 p.separateP(peatones3);
205 //p.separateL(muro);
206 }
207
208 // FUNCIONES DE PERSONAS 2
209 for (int i =
210 peatones2.size()-1; i >= 0; i--) {
211 Persona p2 = peatones2.get(i);
212 // F MAESTRA Y TARGET
213 p2.run();
214
215 // F DE BORDES
216 if (p2.outOfBounds()) {
217 peatones2.remove(i);
218 }
219 }
220 // F DE SEPARACION
221 for (Persona p2 : peatones2) {
222 p2.separateP(peatones);
223 p2.separateP(peatones2);
224 p2.separateP(peatones3);
225 //p.separateL(muro);
226 }
227
228 // FUNCIONES DE PERSONAS 3
229 for (int i = peatones3.size()-1;
230 i >= 0; i--) {
231 Persona p3 = peatones3.get(i);
232 // F MAESTRA Y TARGET
233 p3.run();
234

```

Pestaña 1: Programa 6/8

```

CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
234 // F DE BORDES
235 if (p3.outOfBounds()) {
236     peatones3.remove(i);
237 }
238 }
239 }
240 // F DE SEPARACION
241 for (Persona p3 : peatones3) {
242     p3.separateP(peatones3);
243     p3.separateP(peatones2);
244     p3.separateP(peatones);
245     //p.separateL(muro);
246 }
247 }
248 // FUNCIONES DE BICIS
249 for (int i = ciclistas.size()-1;
250      i >= 0; i--) {
251     Bici b = ciclistas.get(i);
252     b.run();
253 }
254 // F DE BORDES
255 if (b.outOfBounds()) {
256     ciclistas.remove(i);
257 }
258 }
259 }
260 // F DE SEPARACION
261 for (Bici b : ciclistas) {
262     b.separateP(peatones);
263     b.separateB(ciclistas);
264 }
265 }
266 }
267 //FUNCION Y COORDENADAS MAPA LO CONTADOR
268 void newPathLC() {
269     // Path: a series of connected points
270     // A more sophisticated path: curve
271     pathLC = new Path();
272 }
273 pathLC.addPoint(560,100);

```

Pestaña 1: Programa 7/8

```

CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
267 //FUNCION Y COORDENADAS MAPA LO CONTADOR
268 void newPathLC() {
269     // Path: a series of connected points
270     // A more sophisticated path: curve
271     pathLC = new Path();
272 }
273 pathLC.addPoint(560,100);
274 pathLC.addPoint(548,814);
275 pathLC.addPoint(308,820);
276 pathLC.addPoint(324,260);
277 pathLC.addPoint(239,264);
278 pathLC.addPoint(157,331);
279 pathLC.addPoint(138,749);
280 pathLC.addPoint(216,822);
281 pathLC.addPoint(863,814);
282 pathLC.addPoint(858,497);
283 pathLC.addPoint(553,489);
284 pathLC.addPoint(556,339);
285 pathLC.addPoint(862,341);
286 pathLC.addPoint(863,192);
287 pathLC.addPoint(557,184);
288 pathLC.addPoint(559,179);
289 pathLC.addPoint(557,260);
290 pathLC.addPoint(314,260);
291 }
292 }
293 // FUNCION DE VISUALIZACION
294 // DEL CAMBIO DE DIRECCIÓN
295 public void keyPressed() {
296     if (key == ' ') {
297         debug = !debug;
298     }
299 }
300 }
301 // FUNCION PARA LA
302 // CREACION DE NUEVOS CAMINOS
303 public void mousePressed() {
304     //newPath1();
305     //newPath2();
306     newPathLC();

```

Pestaña 1: Programa 8/8

```

CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
1 class Bici extends Persona {
2     // ;La bici hereda las funciones y
3     // variables de la persona!
4
5     // POSICION DE ORIGEN,
6     // VELOCIDAD Y LIMITE DE VELOCIDAD
7     Bici(PVector l, float ms, float mf) {
8         super(l, ms, mf);
9         diam = 1.5;
10        maxspeed = ms; //1.5;
11        maxforce = mf; //0.03;
12    }
13
14    // FUNCION DE APARIENCIA
15    void display() {
16        float theta = velocity.heading()
17                    + PI/2;
18
19        fill(0, 0, 255);
20        noStroke();
21        pushMatrix();
22        translate(location.x, location.y);
23        rotate(theta);
24        beginShape();
25        vertex(0, -diam*2);
26        vertex(-diam, diam);
27        vertex(diam, diam);
28        endShape(CLOSE);
29        popMatrix();
30    }
31
32    // INTERACCION CON PERSONA
33    boolean choca(Persona p) {
34        float d = abs(dist(location.x,
35                          location.y, p.location.x,
36                          p.location.y));
37
38        if (d <= 2) {
39            return true;
40        } else {
41            return false;
42        }

```

Pestaña 2: Bici 1/2

```

CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
40 }
41 }
42
43 // FUNCION DE SEPARACION DE
44 // BICICLETAS CON BICICLETAS
45 void separateB (ArrayList<Bici> ciclistas) {
46 //Note how the desired separation
47 // is based on the Vehicle's size.
48 float desiredseparation = diam*20;
49 PVector sum = new PVector();
50 int count = 0;
51 for (Bici other : ciclistas) {
52 float d = PVector.dist(location,
53 other.location);
54 if ((d > 0) && (d < desiredseparation)) {
55 PVector diff = PVector.sub(location,
56 other.location); diff.normalize();
57 //What is the magnitude of the
58 //PVector pointing away from the
59 // other vehicle? The closer it is,
60 // the more we should flee.
61 // The farther, the less. So we
62 // divide by the distance to weight
63 // it appropriately.
64 diff.div(d);
65 sum.add(diff);
66 count++;
67 }
68 }
69 if (count > 0) {
70 sum.div(count);
71 sum.normalize();
72 sum.mult(maxspeed);
73 PVector steer =
74 PVector.sub(sum, velocity);
75 steer.limit(maxforce);
76 applyForce(steer);
77 }
78 }
79 }

```

Pestaña 2: Bici 2/2

```

CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
1 // Créditos de la función a
2 // Daniel Shiffman - The Nature of Code
3 // http://natureofcode.com
4
5 // Función de Path Following
6 // "Seguimiento de Caminos"
7
8 class Path {
9
10 // A Path is an arraylist of points...
11 // ...(PVector objects)
12 ArrayList<PVector> points;
13 // A path has a radius, i.e how far is
14 float radius;
15
16 // CONSTRUCTOR DEL CAMINO
17 Path() {
18 // Arbitrary radius of 20
19 radius = 20;
20 points = new ArrayList<PVector>();
21 }
22
23 // FUNCION PARA CREAR CAMINOS
24 // DE PUNTO A PUNTO
25 // Add a point to the path
26 void addPoint(float x, float y) {
27 PVector point = new PVector(x, y);
28 points.add(point);
29 }
30
31 PVector getStart() {
32 return points.get(0);
33 }
34
35 PVector getEnd() {
36 return points.get(points.size()-1);
37 }
38
39 // FUNCION DE APARIENCIA
40 // Draw the path

```

Pestaña 3: Path 1/2

```

CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
22
23 // FUNCION PARA CREAR CAMINOS
24 // DE PUNTO A PUNTO
25 // Add a point to the path
26 void addPoint(float x, float y) {
27 PVector point = new PVector(x, y);
28 points.add(point);
29 }
30
31 PVector getStart() {
32 return points.get(0);
33 }
34
35 PVector getEnd() {
36 return points.get(points.size()-1);
37 }
38
39 // FUNCION DE APARIENCIA
40 // Draw the path
41 void display() {
42 // Draw thick line for radius
43 stroke(100);
44 strokeWeight(radius*2);
45 noFill();
46 beginShape();
47 for (PVector v : points) {
48 vertex(v.x, v.y);
49 }
50 endShape();
51 // Draw thin line for center of path
52 stroke(0,20);
53 strokeWeight(1);
54 noFill();
55 beginShape();
56 for (PVector v : points) {
57 vertex(v.x, v.y);
58 }
59 endShape();
60 }
61 }

```

Pestaña 3: Path 2/2


```
CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
1 class Persona {
2
3 // VARIABLES
4 PVector location;
5 PVector velocity;
6 PVector acceleration;
7 float diam;
8 float maxforce;
9 float maxspeed;
10
11 // POSICION DE ORIGEN,
12 // VELOCIDAD Y LIMITE DE VELOCIDAD
13 Persona(PVector l, float ms, float mf) {
14     location = l.get();
15     diam = 1;
16     maxspeed = ms; //0.5
17     maxforce = mf; //0.01
18     acceleration = new PVector(0, 0);
19     velocity = new PVector(random(-1, 1)
20     random(-1, 1));
21     // = new PVector(maxspeed, 0);
22 }
23
24 // FUNCION MAESTRA
25 void run() {
26     update();
27     display();
28     //seek(new PVector(x_, y_));
29     //choqueMurallas();
30     //reboteEdificio();
31 }
32
33 // FUNCION DE SEGUIMIENTO
34 // This function implements Craig
35 // Reynolds' path following algorithm
36 // http://www.red3d.com/cwr/steer/
37 // PathFollow.html
38 void follow(Path p) {
39
40     // PREDICCION DEL MOVIMIENTO
```

Pestaña 4: Persona 1/8

```
CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
39
40 // PREDICCION DEL MOVIMIENTO
41 // Predict location 50 (arbitrary
42 // choice) frames ahead
43 // This could be based on speed
44 PVector predict = velocity.get();
45 predict.normalize();
46 predict.mult(50);
47 PVector predictpos =
48     PVector.add(location, predict);
49
50 // Now we must find the normal to the
51 // path from the predicted location
52 // We look at the normal for each
53 // line segment and pick out the
54 // closest one
55
56 PVector normal = null;
57 PVector target = null;
58 float worldRecord = 1000000;
59 // Start with a very high record
60 // distance that can easily be beaten
61
62 // Loop through all points of the path
63 for (int i = 0;
64     i < p.points.size()-1; i++) {
65
66     // Look at a line segment
67     PVector a = p.points.get(i);
68     PVector b = p.points.get(i+1);
69
70 // Get the normal point to that line
71 PVector normalPoint =
72     getNormalPoint(predictpos, a, b);
73 // This only works because we know
74 // our path goes from left to right
75 // We could have a more
76 // sophisticated test to tell if
77 // the point is in the line
78 // segment or not
```

Pestaña 4: Persona 2/8

```
CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
77 // the point is in the line
78 // segment or not
79 if (normalPoint.x < a.x ||
80     normalPoint.x > b.x) {
81     // This is something of a hacky
82     // solution, but if it's not
83     // within the line segment
84     // consider the normal to just
85     // be the end of the line
86     // segment (point b)
87     normalPoint = b.get();
88 }
89
90 // How far away are we from the
91 // path?
92 float distance =
93     PVector.dist(predictpos,
94                 normalPoint);
95 // Did we beat the record and find
96 // the closest line segment?
97 if (distance < worldRecord) {
98     worldRecord = distance;
99     // If so the target we want to
100 // steer towards is the normal
101     normal = normalPoint;
102
103 // Look at the direction of the
104 // line segment so we can seek a
105 // little bit ahead of the normal
106 PVector dir = PVector.sub(b, a);
107 dir.normalize();
108 // This is an oversimplification
109 // Should be based on distance
110 // to path & velocity
111 dir.mult(10);
112 target = normalPoint.get();
113 target.add(dir);
114 }
115 }
116
```

Pestaña 4: Persona 3/8

```

CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
116
117 // EN CASO DE DESVIARSE
118 // REDIRECCIONARSE
119 // Only if the distance is greater
120 // than the path's radius do we
121 // bother to steer
122 if (worldRecord > p.radius) {
123     seek(target);
124 }
125
126 // FUNCION DE DIBUJO DEL
127 // REDIRECCIONADOR
128 // Draw the debugging stuff
129 if (debug) {
130     // Draw predicted future location
131     stroke(255,40);
132     fill(0,40);
133     line(location.x, location.y,
134         predictpos.x, predictpos.y);
135     ellipse(predictpos.x,
136         predictpos.y, 4, 4);
137
138     // Draw normal location
139     stroke(255,40);
140     fill(0);
141     ellipse(normal.x, normal.y, 4, 4);
142     // Draw actual target
143     // (red if steering towards it)
144     line(predictpos.x, predictpos.y,
145         normal.x, normal.y);
146     if (worldRecord > p.radius)
147         fill(255, 0, 0);
148     noStroke();
149     ellipse(target.x, target.y, 8, 8);
150 }
151 }
152
153 // A function to get the normal point
154 // from a point (p) to a line
155 // segment (a-b)

```

Pestaña 4: Persona 4/8

```

CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
154 // from a point (p) to a line
155 // segment (a-b)
156 // This function could be optimized
157 // to make fewer new Vector objects
158 PVector getNormalPoint(PVector p,
159     PVector a, PVector b) {
160     // Vector from a to p
161     PVector ap = PVector.sub(p, a);
162     // Vector from a to b
163     PVector ab = PVector.sub(b, a);
164     ab.normalize(); // Normalize the
165     // line. Project vector "diff"
166     // onto line by using the dot
167     // product
168     ab.mult(ap.dot(ab));
169     PVector normalPoint =
170     PVector.add(a, ab);
171     return normalPoint;
172 }
173
174 ////////////////////////////////////////////////////////////////////
175
176 // DESPLAZAMIENTO,
177 // VELOCIDAD Y ACELERACION
178 void update() {
179     velocity.add(acceleration);
180     velocity.limit(maxspeed);
181     location.add(velocity);
182     acceleration.mult(0);
183 }
184
185 // APLICADOR DE FUERZAS
186 void applyForce(PVector force) {
187     acceleration.add(force);
188 }
189
190
191 // FUNCION DE APARIENCIA
192 void display() {
193     //Vehicle is a triangle pointing

```

Pestaña 4: Persona 5/8

```

CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
190
191 // FUNCION DE APARIENCIA
192 void display() {
193     //Vehicle is a triangle pointing
194     // in the direction of velocity;
195     // since it is drawn pointing up,
196     // we rotate it an additional 90°.
197     float theta = velocity.heading2D()
198     + PI/2;
199     fill(0, 255, 0);
200     stroke(0, 255, 0);
201     strokeWeight(1);
202     pushMatrix();
203     translate(location.x, location.y);
204     rotate(theta);
205     ellipse(0, 0, diam, diam/3);
206     popMatrix();
207 }
208
209 // FUNCION DE VENTANAS
210 boolean outOfBounds() {
211     if ((location.x > width + diam/2)
212         || (location.x < 0 - diam/2)
213         || (location.y > height + diam/2)
214         || (location.y < 0 - diam/2)) {
215         return true;
216     } else {
217         return false;
218     }
219 }
220
221 // FUNCION DE SEPARACION ENTRE
222 // PEATONES Y PEATONES
223 void separateP (ArrayList<Persona> peaton
224 //Note how the desired separation
225 //is based on the Vehicle's size.
226 float desiredseparation = diam*20;
227 PVector sum = new PVector();
228 int count = 0;
229 for (Persona other : peaton

```

Pestaña 4: Persona 6/8

```
CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
229 for (Persona other : peatones) {
230     float d = PVector.dist(location,
231         other.location); if ((d > 0) &&
232         (d < desiredseparation)) {
233         PVector diff =
234             PVector.sub(location,
235                 other.location);
236         diff.normalize();
237         //What is the magnitude of
238         // the PVector pointing away
239         // from the other vehicle? The
240         // closer it is, the more we
241         // should flee. The farther,
242         // the less. So we divide by
243         // the distance to weight it
244         // appropriately.
245         diff.div(d);
246         sum.add(diff);
247         count++;
248     }
249 }
250 if (count > 0) {
251     sum.div(count);
252     sum.normalize();
253     sum.mult(maxspeed);
254     PVector steer = PVector.sub(sum,
255         velocity);
256     steer.limit(maxforce);
257     applyForce(steer);
258 }
259 }
260 // FUNCION DE ATRACCION
261 void seek(PVector target) {
262     PVector desired = PVector.sub(target
263         location)
264     if (desired.mag() == 0) return;
265     desired.normalize();
266     desired.mult(maxspeed);
267     PVector steer = PVector.sub(desired,
```

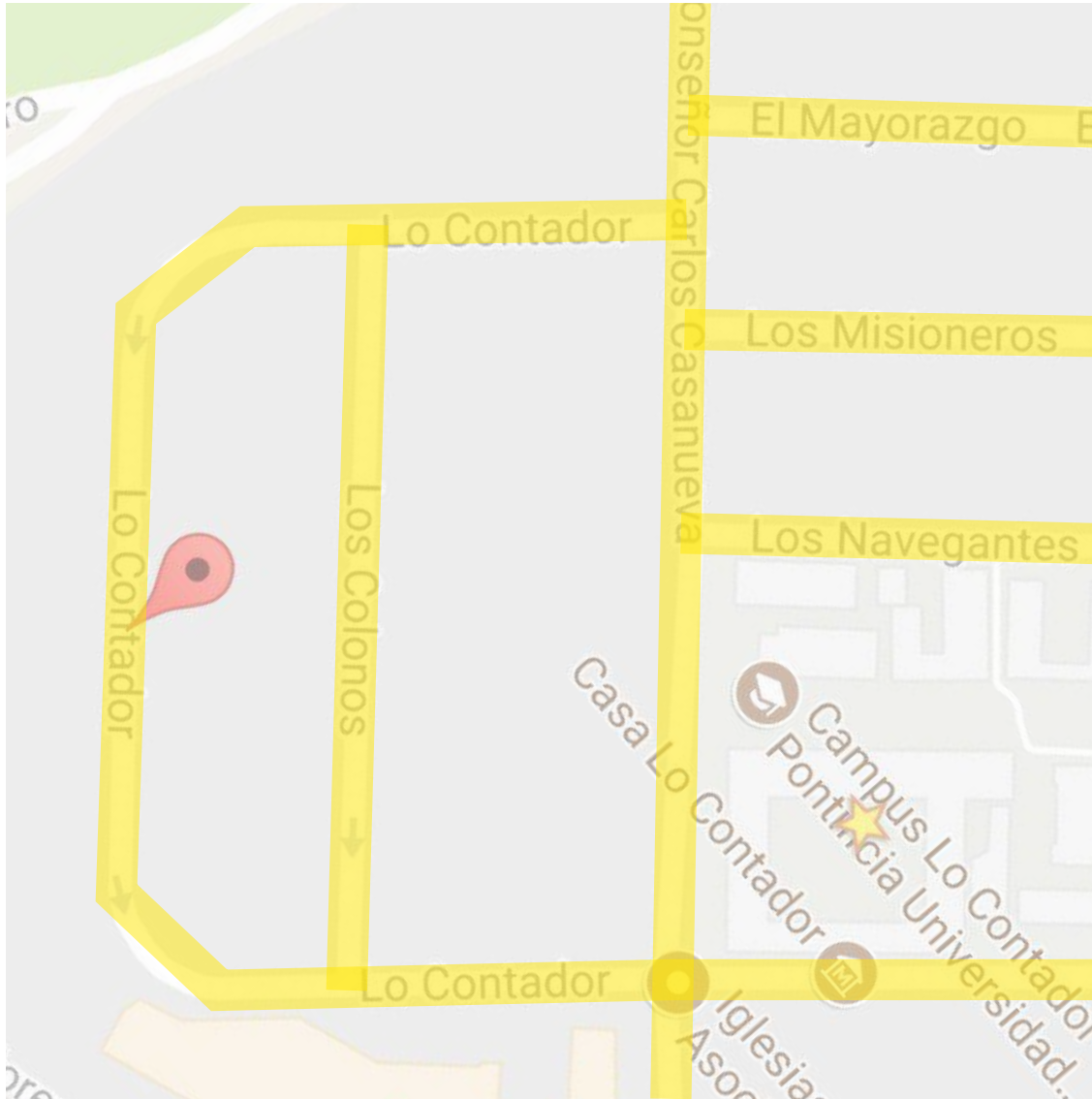
Pestaña 4: Persona 7/8

```
CODIGO_v04 | Processing 3.3.5
CODIGO_v04 Bici Path Persona
238 // the PVector pointing away
239 // from the other vehicle? The
240 // closer it is, the more we
241 // should flee. The farther,
242 // the less. So we divide by
243 // the distance to weight it
244 // appropriately.
245 diff.div(d);
246 sum.add(diff);
247 count++;
248 }
249 }
250 if (count > 0) {
251     sum.div(count);
252     sum.normalize();
253     sum.mult(maxspeed);
254     PVector steer = PVector.sub(sum,
255         velocity);
256     steer.limit(maxforce);
257     applyForce(steer);
258 }
259 }
260 // FUNCION DE ATRACCION
261 void seek(PVector target) {
262     PVector desired = PVector.sub(target
263         location)
264     if (desired.mag() == 0) return;
265     desired.normalize();
266     desired.mult(maxspeed);
267     PVector steer = PVector.sub(desired,
268         velocity);
269     steer.limit(maxforce);
270     applyForce(steer);
271 }
272 }
273 }
274 }
275 }
276 }
277 }
```

Pestaña 4: Persona 8/8

VERSIÓN 04

material complementario



MAPA

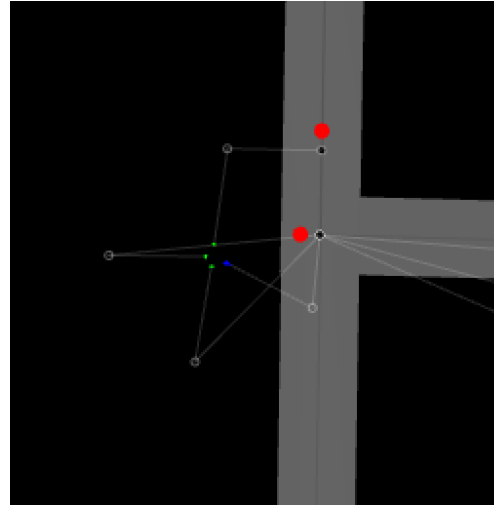
Se decidió usar un mapa de naturaleza casi ortogonal y de ubicación conocida, con el fin de poder reutilizar la parte del código encargada de la repulsión de los agentes como ya lo hacían las “manzanas” y también para entender la escala en la que se estaba desarrollando el simulador. Para nuestra suerte las cuadras cercanas al campus Lo Contador coincidían con tales criterios.

El desafío estaba en lograr encontrar el archivo de las calles en formato SVG para poder importarlos de forma nativa a Processing. Esto permitiría, si se lograba hacer de manera correcta, cambiar eventualmente el mapa, el contexto, de manera automática permitiendo así la proyección del simulador en otros países, pensando en otras necesidades de otros usuarios.

Finalmente se importó el SVG como una textura de fondo que ayudó en la escritura a mano de los Paths de manera manual. (Pi, 268-292)

VERSIÓN 04

funcionamiento,
observaciones y
conclusiones

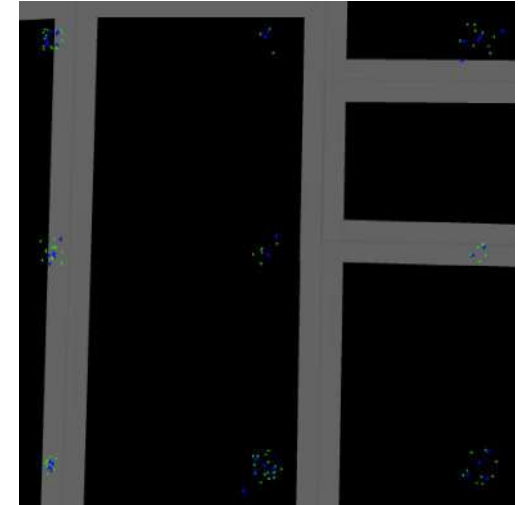


1. PREDICTOR

Esta versión del simulador estaba especializada en sólo 3 cosas, 1. Usar un mapa real, 2. Mantener a los agentes dentro de las calles (gracias a repulsores) y atraerlos a un atractor (un destino) y 3. Ajustar las dimensiones de los agentes a una escala real.

Para resolver el punto 2. se comenzó por aplicar (P4, 40-151) un extracto del código de Daniel Shifman¹ (escrito en base a los Boids de Craig Reynolds) el cual permitía determinar cual sería la futura posición del agente, según su velocidad y dirección, y con esto se decidía aplicar una fuerza 'x' la cual lo hacía redirigirse hacia y por la calle.

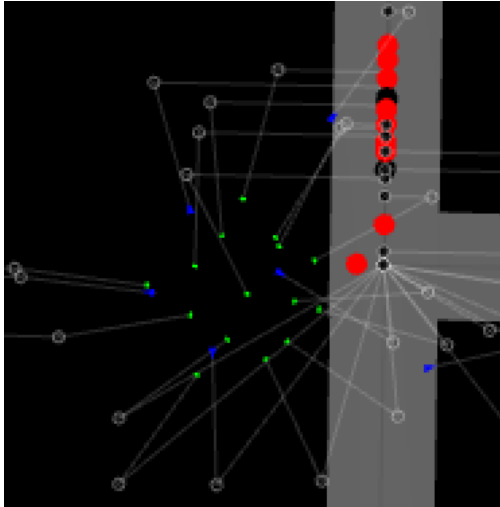
¹. (Shifman, 2006),
Nature Of Code. Capítulo 6.



2. ORÍGENES

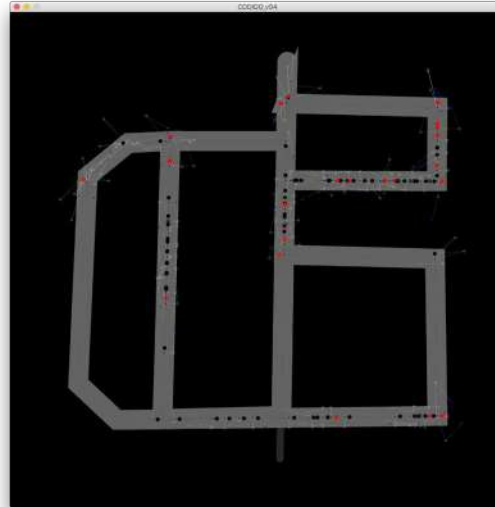
Se reutilizaron las ubicaciones de los puntos de origen del programa anterior, pero esta vez sin la intención de aparentar que los agentes estuviesen saliendo de edificios, si no para ver como se desenvolvían desde puntos repartidos relativamente al azar de la pantalla. (P1, 116-138)

Recordemos que los orígenes eventualmente debiesen poder imitar los puntos de origen reales de la ciudad, por ejemplo: un edificio de oficinas, una universidad, un supermercado al paso, entre otros.



3. SEPARACIÓN

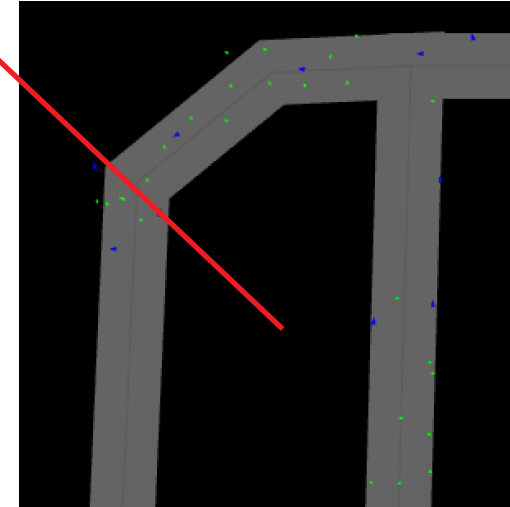
Para evitar el cruce entre agentes de su mismo tipo o cualquier otro se agregó una función de separación también descrita en el libro Nature Of Code para mantener los móviles a una distancia establecida. (P4, 221-260) Mejorando así la naturalidad del comportamiento emergente de los agentes y también sus flujos.



4. ATRACCIÓN CALLES

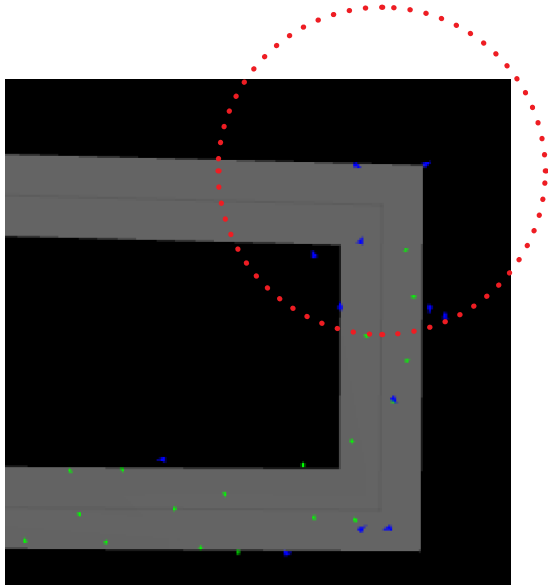
Con la ayuda del predictor de posición y la función de separación los agentes en vez de ser repelidos por los edificios estos eran atraídos por las calles. Esto sería un cambio de paradigma en el desarrollo del simulador para más adelante. (P4, 260-273)

De esta manera se logró revertir el sistema de flujo por las calles, anteriormente el simulador buscaba repeler a los agentes con las murallas de los edificios, pero de esta manera se mantienen a los agentes y su flujo dentro de las calles por atracción sin importar la infraestructura de alrededor. El usuario podría fácilmente importar vectores.



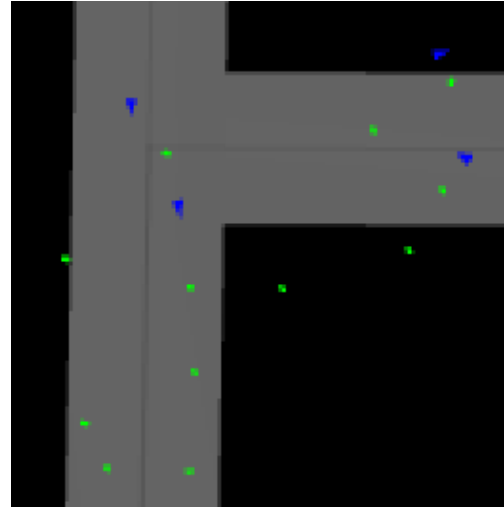
5.1 ERROR ATRACTORES

Al ser ya atraídos por las calles solo se faltaba crear atractores que generaran el flujo y la dirección de movimiento de las personas, más al usar la función de atracción de las calles de manera simultánea con la de los atractores, los agentes se movían de manera “confundida” o incierta. Es por esto que se generaban cambios de sentido repentinos. Se creó un solo Path, un camino formado por coordenadas de puntos, que daba vueltas por todas las calles, y algunas repitiéndose; quizás tales redundancias generaban una doble atracción para los agentes y por eso se quedaban estancados. Se puede separar en más paths, uno para cada calle e ir atrayéndolos.



5.2 ERRORES ATRACTORES

En ciertos puntos frecuentaba más aquel error y se formaban acumulaciones de agentes girando en círculos. La redundancia se daba en varios puntos del path debido a su estilo de construcción.



6. APARIENCIA

En esta versión se logró programar una pequeña función que permitía girar los agentes según la dirección en la que se dirigen. Esto permite entender con mayor claridad como se están moviendo y predecir como se van a encontrar.

Este detalle permite la personalización de cada tipo de agente por sí sólo. Esto da un generoso grado de naturalidad y de entendimiento del sentido, dirección, flujo e interacción entre los agentes para el usuario. Al reconocerse mejor las formas y tamaños no sería necesario quizás disponer de una simbología para que el usuario tenga que entender los tipos de agentes.

VERSIÓN 04

conclusión general,
reflexión y proyección

CONCLUSIÓN GENERAL

Los agentes siguen el camino, se mueven con decisión, mantienen la separación, se mueven hacia un objetivo, pero no uno real. Se debe mejorar esta situación, en especial por si se quiere imitar un flujo real, en un espacio real y para que sea usado en la toma de decisiones de planificación urbana.

Faltan atractores que fijen el flujo de manera correcta, además de agrandar el mapa para que se entienda como un fragmento de una parte más grande de la ciudad. La posibilidades de lograr hacer de el mapa una parte editable son bajas, pero si se pueden ingresar al menos los datos de las coordenadas de las calles directamente. Permitiendo de esta forma poder usar un mapa real, uno con el cual el usuario se sienta familiar y cómodo a la hora de interactuar.

La escala a pesar de ser la verosímil no permite la correcta lectura o entendimiento de las situaciones y de choques, se podría modificar un poco para generar el efecto contrario. Al usar la escala 1:1 (un pixel equivale a un metro) se pierde la referencia de escala y dependiendo de la pantalla se pierde nitidez.

PRÓXIMOS PASOS

1. Plano real de providencia: con calles, edificios, y un mapa de coordenadas de fondo. Que el mapa se entienda como un fragmento de un trozo más grande de la ciudad. Una versión del simulador futuro debería tener infaltablemente esta carecterística.

2. Atractores fuera del plano, y de tres tipos con distintos grados de incidencia para cada agente. Para marcar los flujos, los destinos y la dirección con la que se mueven los agentes que controla el usuario dependiendo de su voluntad, o la del gobierno/municipalidad.

3. Interfaz gráfica con sliders para modificar la cantidad de agentes por medio de la interacción. Primer gran paso que abre la posibilidad clara de interacción del usuario con el programa de simulación.

4. Agregar simbología dinámica. Con esto se refiere a los datos que se actualizan en forma real según la cantidad de agentes determinada de forma interactiva por el usuario con los sliders.

5. Mejorar la escala y los puntos de origen a algunos más verosímil y legible.

VERSIÓN 05*

la que parece ser algo



*** SÁLTESE A LA PÁGINA 102 PARA VER LA SIGUIENTE Y MENOS ÁRIDA SECCIÓN**

```
CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
CODIGO_5_04_RainbowPaths_y_3Atractores
1 // FUNCIONES DEL AUTO 2 y 3
2 // ESTAN DESACTIVADAS #MUY PESADO#
3
4 // IMPORTAR LIBRERIAS
5 import controlP5.*;
6 ControlP5 cp5;
7
8 // FONDO Y SIMBOLOGIA
9 PImage fondo;
10 PImage fondocolor;
11 boolean zonasclick = false;
12 PImage simbologia;
13
14 // FUNCION DEL PATH
15 boolean debug = true;
16
17 // LISTA DE PATHS
18 Path pathp1;
19 Path pathp2;
20 Path pathp3;
21 Path pathb1;
22 Path pathb2;
23 Path pathb3;
24 Path patha1;
25 Path patha2;
26 Path patha3;
27
28 // LISTAS DE AGENTES
29 ArrayList<Persona> peatones;
30 ArrayList<Persona> peatones2;
31 ArrayList<Persona> peatones3;
32 ArrayList<Bici> ciclistas;
33 ArrayList<Bici> ciclistas2;
34 ArrayList<Bici> ciclistas3;
35 ArrayList<Auto> conductores;
36 ArrayList<Auto> conductores2;
37 ArrayList<Auto> conductores3;
38
39 // NUMERO MAX. DE AGENTES
40 // Y PROPORCIOENS
```

Pestaña 1: Programa 1/18

```
CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
CODIGO_5_04_RainbowPaths_y_3Atractores
39 // NUMERO MAX. DE AGENTES
40 // Y PROPORCIOENS
41 int peatonesGrupo1 = 5;
42 int peatonesGrupo2 = 5;
43 int peatonesGrupo3 = 5;
44 int ciclistasGrupo1 = 5;
45 int ciclistasGrupo2 = 5;
46 int ciclistasGrupo3 = 5;
47 int conductoresGrupo1 = 5;
48 int conductoresGrupo2 = 5;
49 int conductoresGrupo3 = 5;
50
51
52 // COORDS MAPA PROVIDENCIA
53 PVector v1 = new PVector(374, 0);
54 PVector v2 = new PVector(426, 0);
55 PVector v3 = new PVector(576, 0);
56 PVector v4 = new PVector(705, 0);
57 PVector v5 = new PVector(800, 0);
58 PVector v6 = new PVector(853, 0);
59 PVector v7 = new PVector(390, 33);
60 PVector v8 = new PVector(813, 21);
61 PVector v9 = new PVector(746, 57);
62 PVector v10 = new PVector(1000, 36);
63 PVector v11 = new PVector(728, 107);
64 PVector v12 = new PVector(272, 125);
65 PVector v13 = new PVector(670, 147);
66 PVector v14 = new PVector(869, 117);
67 PVector v15 = new PVector(140, 213);
68 PVector v16 = new PVector(718, 221);
69 PVector v17 = new PVector(1000, 172);
70 PVector v18 = new PVector(42, 272);
71 PVector v19 = new PVector(349, 304);
72 PVector v20 = new PVector(597, 310);
73 PVector v21 = new PVector(937, 231);
74 PVector v22 = new PVector(0, 295);
75 PVector v23 = new PVector(0, 388);
76 PVector v24 = new PVector(279, 395);
77 PVector v25 = new PVector(465, 400);
78 PVector v26 = new PVector(805, 341);
```

Pestaña 1: Programa 2/18

```
CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
CODIGO_5_04_RainbowPaths_y_3Atractores
78 PVector v26 = new PVector(805, 341);
79 PVector v27 = new PVector(1000, 427);
80 PVector v28 = new PVector(406, 455);
81 PVector v29 = new PVector(352, 491);
82 PVector v30 = new PVector(647, 471);
83 PVector v31 = new PVector(54, 440);
84 PVector v32 = new PVector(1000, 435);
85 PVector v33 = new PVector(285, 556);
86 PVector v34 = new PVector(0, 538);
87 PVector v35 = new PVector(209, 623);
88 PVector v36 = new PVector(483, 606);
89 PVector v37 = new PVector(697, 630);
90 PVector v38 = new PVector(891, 477);
91 PVector v39 = new PVector(133, 698);
92 PVector v40 = new PVector(277, 824);
93 PVector v41 = new PVector(526, 683);
94 PVector v42 = new PVector(731, 739);
95 PVector v43 = new PVector(1000, 666);
96 PVector v44 = new PVector(0, 852);
97 PVector v45 = new PVector(598, 862);
98 PVector v46 = new PVector(170, 896);
99 PVector v47 = new PVector(324, 953);
100 PVector v48 = new PVector(638, 966);
101 PVector v49 = new PVector(789, 922);
102 PVector v50 = new PVector(0, 977);
103 PVector v51 = new PVector(335, 1000);
104 PVector v52 = new PVector(649, 1000);
105 PVector v53 = new PVector(811, 1000);
106
107
108 // COORDENADAS ATRACTORES
109 float diam = 15;
110 PVector cuadra1 =
111     new PVector(v53.x, v53.y+100);
112 PVector cuadra2 =
113     new PVector(v52.x, v52.y+100);
114 PVector cuadra3 =
115     new PVector(v17.x+100, v17.y);
116 PVector barrio1 =
117     new PVector(v17.x+100, v17.y);
```

Pestaña 1: Programa 3/18

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
Java
CODIGO_5_04_RainbowPaths_y_3Atractores
117 new PVector(v17.x+100, v17.y);
118 PVector barrio2 =
119 new PVector(v53.x, v53.y+100);
120 PVector barrio3 =
121 new PVector(v23.x-100, v23.y);
122 PVector comunal =
123 new PVector(v53.x, v53.y+100);
124 PVector comuna2 =
125 new PVector(v10.x+100, v10.y);
126 PVector comuna3 =
127 new PVector(v43.x+100, v43.y);
128
129 // SETUP //
130 void setup() {
131 size(1000, 1000, P2D);
132 smooth();
133 frameRate(25);
134 // IMAGENES
135 fondo =
136 loadImage("Satelital-PSD-calle.jpg")
137 fondocolor =
138 loadImage("Satelital-PSD-calleColor.");
139 simbologia =
140 loadImage("Simbologia-copia.png");
141
142 // SLIDERS (Cortesía de la librería)
143 cp5 = new ControlP5(this);
144 cp5.addSlider("peatonesGrupo1").setPos:
145 cp5.addSlider("peatonesGrupo2").setPos:
146 cp5.addSlider("peatonesGrupo3").setPos:
147 cp5.addSlider("ciclistasGrupo1").setPo:
148 cp5.addSlider("ciclistasGrupo2").setPo:
149 cp5.addSlider("ciclistasGrupo3").setPo:
150 cp5.addSlider("conductoresGrupo1").setI:
151 cp5.addSlider("conductoresGrupo2").setI:
152 cp5.addSlider("conductoresGrupo3").setI:
153
154 // SETEO de Paths
155 newPathp1();

```

Pestaña 1: Programa 4/18

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
Java
CODIGO_5_04_RainbowPaths_y_3Atractores
155 // SETEO de Paths
156 newPathp1();
157 newPathp2();
158 newPathp3();
159 newPathb1();
160 newPathb2();
161 newPathb3();
162 newPatha1();
163 newPatha2();
164 newPatha3();
165
166 // SETEO DE AGENTES
167 peatones = new ArrayList<Persona>();
168 peatones2 = new ArrayList<Persona>();
169 peatones3 = new ArrayList<Persona>();
170 ciclistas = new ArrayList<Bici>();
171 ciclistas2 = new ArrayList<Bici>();
172 ciclistas3 = new ArrayList<Bici>();
173 conductores = new ArrayList<Auto>();
174 conductores2 = new ArrayList<Auto>();
175 conductores3 = new ArrayList<Auto>();
176 }
177
178 // D R A W //
179 void draw() {
180 // FONDO Y ZONAS
181 // (TRÁNSITO Y RESIDENCIALES)
182 if (mousePressed == true) {
183 image(fondocolor, 0, 0);
184 } else {
185 image(fondo, 0, 0);
186 }
187 // INSTRUCCIONES
188 pushMatrix();
189 fill(255);
190 text("Manten presionado el cursor para
191 // "Manten presionado el cursor para
192 // ver las zonas Residenciales en
193 // verde y las de Transito en naranja"
194

```

Pestaña 1: Programa 5/18

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
Java
CODIGO_5_04_RainbowPaths_y_3Atractores
196 // SIMBOLOGIA DINAMICA
197 // Muestran la suma de personas
198 // ciclistas y conductores
199 image(simbologia, 10, 120, 100, 100);
200 text(peatonesGrupo1+peatonesGrupo2+peat:
201 text(ciclistasGrupo1+ciclistasGrupo2+c:
202 text(conductoresGrupo1+conductoresGrup:
203 popMatrix();
204
205 // COORDENADAS DEL MOUSE
206 // (x,y)s para orientar el mapa
207 println(mouseX, mouseY);
208
209 ////FUNCIONES DEL PATH
210 // Sirven para visualizar los caminos
211 //pathp1.display(0, 255, 0);
212 //pathp2.display(0, 255, 0);
213 //pathp3.display(0, 255, 0);
214 //pathb1.display(0, 0, 255);
215 //pathb2.display(0, 0, 255);
216 //pathb3.display(0, 0, 255);
217 //patha1.display(255, 0, 0);
218 //patha2.display(255, 0, 0);
219 //patha3.display(255, 0, 0);
220
221 // FUNCION AGENTE FOLLOWS PATH N°
222 for (Persona p : peatones) {
223 p.follow(pathp1);
224 }
225 for (Persona p2 : peatones2) {
226 p2.follow(pathp2);
227 }
228 for (Persona p3 : peatones3) {
229 p3.follow(pathp3);
230 }
231 for (Bici b : ciclistas) {
232 b.follow(pathb1);
233 }
234 for (Bici b2 : ciclistas2) {
235 b2.follow(pathb2);

```

Pestaña 1: Programa 6/18

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
CODIGO_5_04_RainbowPaths_y_3Atractores
235     b2.follow(pathb2);
236 }
237 for (Bici b3 : ciclistas3) {
238     b3.follow(pathb3);
239 }
240 for (Auto a : conductores) {
241     a.follow(patha1);
242 }
243 for (Auto a2 : conductores2) {
244     a2.follow(patha2);
245 }
246 for (Auto a3 : conductores3) {
247     a3.follow(patha3);
248 }
249
250 // ORIGENES DE PERSONAS Y
251 // BICIS Y AUTOS
252 //int r1 = 200*(int(random(2, 5)));
253 peatones.add(new
254     Persona(random(v34.x-10, v34.x+10),
255         random(v34.y-10, v34.y+10)));
256 peatones2.add(new
257     Persona(random(v2.x-10, v2.x+10),
258         random(v2.y-10, v2.y+10)));
259 peatones3.add(new
260     Persona(random(v50.x-10, v50.x+10),
261         random(v50.y-10, v50.y+10)));
262 ciclistas.add(new
263     Bici(random(v52.x-10, v52.x+10),
264         random(v52.y-10, v52.y+10)));
265 ciclistas2.add(new
266     Bici(random(v22.x-10, v22.x+10),
267         random(v22.y-10, v22.y+10)));
268 ciclistas3.add(new
269     Bici(random(v10.x-10, v10.x+10),
270         random(v10.y-10, v10.y+10)));
271 conductores.add(new
272     Auto(random(v1.x-10, v1.x+10),
273         random(v1.y-10, v1.y+10)));
274 conductores2.add(new

```

Pestaña 1: Programa 7/18

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
CODIGO_5_04_RainbowPaths_y_3Atractores
271 conductores.add(new
272     Auto(random(v1.x-10, v1.x+10),
273         random(v1.y-10, v1.y+10)));
274 conductores2.add(new
275     Auto(random(v44.x-10, v44.x),
276         random(v44.y-10, v44.y+10)));
277 conductores3.add(new
278     Auto(random(v30.x-10, v30.x+10),
279         random(v30.y-10, v30.y+10)));
280
281 // CANTIDADES DE AGENTES
282 if (peatones.size() > peatonesGrupo1)
283     peatones.remove(peatonesGrupo1);
284 }
285 if (peatones2.size() > peatonesGrupo2)
286     peatones2.remove(peatonesGrupo2);
287 }
288 if (peatones3.size() > peatonesGrupo3)
289     peatones3.remove(peatonesGrupo3);
290 }
291 if (ciclistas.size() > ciclistasGrupo1)
292     ciclistas.remove(ciclistasGrupo1);
293 }
294 if (ciclistas2.size() > ciclistasGrupo2)
295     ciclistas2.remove(ciclistasGrupo2);
296 }
297 if (ciclistas3.size() > ciclistasGrupo3)
298     ciclistas3.remove(ciclistasGrupo3);
299 }
300 if (conductores.size() > conductoresGrupo1)
301     conductores.remove(conductoresGrupo1);
302 }
303 if (conductores2.size() > conductoresGrupo2)
304     conductores2.remove(conductoresGrupo2);
305 }
306 if (conductores3.size() > conductoresGrupo3)
307     conductores3.remove(conductoresGrupo3);
308 }
309
310

```

Pestaña 1: Programa 8/18

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
CODIGO_5_04_RainbowPaths_y_3Atractores
311 // CHOQUES VEHICULOS CON PERSONAS
312 for (int i = peatones.size()-1; i >= 0)
313     for (int j = ciclistas.size()-1; j >= 0)
314         for (int k = conductores.size()-1; k >= 0)
315             for (int l = peatones2.size()-1; l >= 0)
316                 for (int m = peatones3.size()-1; m >= 0)
317                     for (int n = ciclistas2.size()-1; n >= 0)
318                         for (int o = ciclistas3.size()-1; o >= 0)
319                             Persona p = peatones.get(i); //
320                             Persona p2 = peatones2.get(l);
321                             Persona p3 = peatones3.get(m);
322                             Bici b = ciclistas.get(j);
323                             Bici b2 = ciclistas2.get(n);
324                             Bici b3 = ciclistas3.get(o);
325                             Auto a = conductores.get(k);
326
327 // (B1)
328 // Si choca con P: limitar velocidad
329 if (b.choca(p) || b.choca(p2)
330     || b.choca(p3)) {
331     b.velocity.x = constrain(b.velocity.x,
332     b.velocity.y = constrain(b.velocity.y,
333     }
334 // (B2)
335 // Si choca con P: limitar velocidad
336 if (b2.choca(p) || b2.choca(p2)
337     || b2.choca(p3)) {
338     b2.velocity.x = constrain(b2.velocity.x,
339     b2.velocity.y = constrain(b2.velocity.y,
340     }
341 // (B3)
342 // Si choca con P: limitar velocidad
343 if (b3.choca(p) || b3.choca(p2)
344     || b3.choca(p3)) {
345     b3.velocity.x = constrain(b3.velocity.x,
346     b3.velocity.y = constrain(b3.velocity.y,
347     }
348 // (AUTO) SOLAMENTE AUTO (1)
349 // Si choca con P: limitar velocidad
350 // Si choca con B: limitar velocidad

```

Pestaña 1: Programa 9/18

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
CODIGO_5_04_RainbowPaths_y_3Atractores
350 // Si choca con B: limitar velocidad
351 if (a.choca(p) || a.choca(p2)
352 || a.choca(p3) || a.choca(b)
353 || a.choca(b2) || a.choca(b3)) {
354 a.velocity.x = constrain(a.velocity.x,
355 a.velocity.y = constrain(a.velocity.y,
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 // FUNCIONES PERSONAS (1)
365 for (int i = peatones.size()-1;
366 i >= 0; i--) {
367 Persona p = peatones.get(i);
368 p.run(cuadra1.x, cuadra1.y);
369 fill(0, 255, 0);
370 ellipse(cuadra1.x, cuadra1.y, diam, c
371 // SI SALE DE LA VENTANA MUERE
372 if (p.outOfBounds()) {
373 peatones.remove(i);
374 }
375 }
376 for (Persona p : peatones) {
377 p.separateP(peatones);
378 p.separateP(peatones2);
379 p.separateP(peatones3);
380 }
381 // FUNCIONES PERSONAS (2)
382 for (int i = peatones2.size()-1;
383 i >= 0; i--) {
384 Persona p2 = peatones2.get(i);
385 p2.run(cuadra2.x, cuadra2.y);
386 fill(0, 255, 0);
387 ellipse(cuadra2.x, cuadra2.y, diam, c
388
389

```

Pestaña 1: Programa 10/8

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
CODIGO_5_04_RainbowPaths_y_3Atractores
389 ellipse(cuadra2.x, cuadra2.y, diam, c
390 // SI SALE DE LA VENTANA MUERE
391 if (p2.outOfBounds()) {
392 peatones2.remove(i);
393 }
394 }
395 for (Persona p2 : peatones2) {
396 p2.separateP(peatones);
397 p2.separateP(peatones2);
398 p2.separateP(peatones3);
399 }
400 // FUNCIONES PERSONAS (3)
401 for (int i = peatones3.size()-1;
402 i >= 0; i--) {
403 Persona p3 = peatones3.get(i);
404 p3.run(cuadra3.x, cuadra3.y);
405 fill(0, 255, 0);
406 ellipse(cuadra3.x, cuadra3.y, diam, c
407 // SI SALE DE LA VENTANA MUERE
408 if (p3.outOfBounds()) {
409 peatones3.remove(i);
410 }
411 }
412 for (Persona p3 : peatones3) {
413 p3.separateP(peatones);
414 p3.separateP(peatones2);
415 p3.separateP(peatones3);
416 }
417 // FUNCIONES BICIS (1)
418 for (int i = ciclistas.size()-1;
419 i >= 0; i--) {
420 Bici b = ciclistas.get(i);
421 b.run(barrio1.x, barrio1.y);
422 fill(0, 100, 255);
423 ellipse(barrio1.x, barrio1.y, diam, c
424 //SI SALE DE LA VENTANA MUERE
425 if (b.outOfBounds()) {
426
427
428

```

Pestaña 1: Programa 11/18

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
CODIGO_5_04_RainbowPaths_y_3Atractores
429 ciclistas.remove(i);
430 }
431 }
432 for (Bici b : ciclistas) {
433 b.separateP(peatones);
434 b.separateP(peatones2);
435 b.separateP(peatones3);
436 b.separateB(ciclistas);
437 b.separateB(ciclistas2);
438 b.separateB(ciclistas3);
439 }
440 // FUNCIONES BICIS (2)
441 for (int i = ciclistas2.size()-1;
442 i >= 0; i--) {
443 Bici b2 = ciclistas2.get(i);
444 b2.run(barrio2.x, barrio2.y);
445 fill(0, 100, 255);
446 ellipse(barrio2.x, barrio2.y, diam, c
447 //SI SALE DE LA VENTANA MUERE
448 if (b2.outOfBounds()) {
449 ciclistas2.remove(i);
450 }
451 }
452 for (Bici b2 : ciclistas2) {
453 b2.separateP(peatones);
454 b2.separateP(peatones2);
455 b2.separateP(peatones3);
456 b2.separateB(ciclistas);
457 b2.separateB(ciclistas2);
458 b2.separateB(ciclistas3);
459 }
460 // FUNCIONES BICIS (3)
461 for (int i = ciclistas3.size()-1;
462 i >= 0; i--) {
463 Bici b3 = ciclistas3.get(i);
464 b3.run(barrio3.x, barrio3.y);
465 fill(0, 100, 255);
466 ellipse(barrio3.x, barrio3.y, diam, c
467
468

```

Pestaña 1: Programa 12/18

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
468 ellipse(barrio3.x, barrio3.y, diam, c
469 //SI SALE DE LA VENTANA MUERE
470 if (b3.outOfBounds()) {
471     ciclistas3.remove(i);
472 }
473 }
474 for (Bici b3 : ciclistas3) {
475     b3.separateP(peatones);
476     b3.separateP(peatones2);
477     b3.separateP(peatones3);
478     b3.separateB(ciclistas);
479     b3.separateB(ciclistas2);
480     b3.separateB(ciclistas3);
481 }
482 // FUNCIONES AUTO SOLO (1)
483 for (int i = conductores.size()-1;
484     i >= 0; i--) {
485     Auto a = conductores.get(i);
486     a.run(comunal.x, comunal.y);
487     fill(255, 0, 0);
488     ellipse(comunal.x, comunal.y, diam, c
489 }
490 // SI SALE DE LA VENTANA MUERE
491 if (a.outOfBounds()) {
492     conductores.remove(i);
493 }
494 } //(1)
495 for (Auto a : conductores) {
496     a.separateP(peatones);
497     a.separateP(peatones2);
498     a.separateP(peatones3);
499     a.separateB(ciclistas);
500     a.separateB(ciclistas2);
501     a.separateB(ciclistas3);
502     a.separateA(conductores);
503     a.separateA(conductores2);
504     a.separateA(conductores3);
505 }
506 }
507

```

Pestaña 1: Programa 13/18

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
508 // FUNCIONES AUTO SOLO (2)
509 for (int i = conductores2.size()-1;
510     i >= 0; i--) {
511     Auto a2 = conductores2.get(i);
512     a2.run(comuna2.x, comuna2.y);
513     fill(255, 0, 0);
514     ellipse(comunal.x, comunal.y, diam, c
515 }
516 // SI SALE DE LA VENTANA MUERE
517 if (a2.outOfBounds()) {
518     conductores2.remove(i);
519 }
520 } //(1)
521 for (Auto a2 : conductores2) {
522     a2.separateP(peatones);
523     a2.separateP(peatones2);
524     a2.separateP(peatones3);
525     a2.separateB(ciclistas);
526     a2.separateB(ciclistas2);
527     a2.separateB(ciclistas3);
528     a2.separateA(conductores);
529     a2.separateA(conductores2);
530     a2.separateA(conductores3);
531 }
532 // FUNCIONES AUTO SOLO (3)
533 for (int i = conductores3.size()-1;
534     i >= 0; i--) {
535     Auto a3 = conductores3.get(i);
536     a3.run(comuna3.x, comuna3.y);
537     fill(255, 0, 0);
538     ellipse(comuna3.x, comuna3.y, diam, c
539 }
540 // SI SALE DE LA VENTANA MUERE
541 if (a3.outOfBounds()) {
542     conductores3.remove(i);
543 }
544 } //(1)
545 for (Auto a3 : conductores3) {
546     a3.separateP(peatones);
547

```

Pestaña 1: Programa 14/18

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
545 } //(1)
546 for (Auto a3 : conductores3) {
547     a3.separateP(peatones);
548     a3.separateP(peatones2);
549     a3.separateP(peatones3);
550     a3.separateB(ciclistas);
551     a3.separateB(ciclistas2);
552     a3.separateB(ciclistas3);
553     a3.separateA(conductores);
554     a3.separateA(conductores2);
555     a3.separateA(conductores3);
556 }
557 }
558 // FUNCIONES MACRO de los PATHS
559 void newPath1() {
560     // PATH PERSONA (1)
561     // Path: a series of connected points,
562     pathp1 = new Path();
563     pathp1.addPoint(v34.x, v34.y);
564     pathp1.addPoint(v39.x, v39.y);
565     pathp1.addPoint(v35.x, v35.y);
566     pathp1.addPoint(v40.x, v40.y);
567     pathp1.addPoint(v36.x, v36.y);
568     pathp1.addPoint(v41.x, v41.y);
569     pathp1.addPoint(v45.x, v45.y);
570     pathp1.addPoint(v48.x, v48.y);
571     pathp1.addPoint(v49.x, v49.y);
572     pathp1.addPoint(v53.x, v53.y);
573     pathp1.addPoint(cuadra1.x, cuadra1.y);
574 }
575 void newPath2() {
576     // PATH PERSONA (2)
577     pathp2 = new Path();
578     pathp2.addPoint(v7.x, v7.y);
579 }
580

```

Pestaña 1: Programa 15/18

```
CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
Java
CODIGO_5_04_RainbowPaths_y_3Atractores
584 pathp2.addPoint(v7.x, v7.y);
585 pathp2.addPoint(v12.x, v12.y);
586 pathp2.addPoint(v25.x, v25.y);
587 pathp2.addPoint(v28.x, v28.y);
588 pathp2.addPoint(v36.x, v36.y);
589 pathp2.addPoint(v41.x, v41.y);
590 pathp2.addPoint(v45.x, v45.y);
591 pathp2.addPoint(v48.x, v48.y);
592 pathp2.addPoint(v52.x, v52.y);
593 pathp2.addPoint(cuadra2.x, cuadra2.y);
594 }
595
596 void newPath3() {
597     // PATH PERSONA (2)
598     pathp3 = new Path();
599     pathp3.addPoint(v44.x, v44.y);
600     pathp3.addPoint(v39.x, v39.y);
601     pathp3.addPoint(v35.x, v35.y);
602     pathp3.addPoint(v33.x, v33.y);
603     pathp3.addPoint(v28.x, v28.y);
604     pathp3.addPoint(v36.x, v36.y);
605     pathp3.addPoint(v30.x, v30.y);
606     pathp3.addPoint(v26.x, v26.y);
607     pathp3.addPoint(v21.x, v21.y);
608     pathp3.addPoint(v17.x, v17.y);
609     pathp3.addPoint(cuadra3.x, cuadra3.y);
610 }
611
612 void newPathb1() { // PATH BICI
613     pathb1 = new Path();
614     pathb1.addPoint(v52.x, v52.y);
615     pathb1.addPoint(v41.x, v41.y);
616     pathb1.addPoint(v36.x, v36.y);
617     pathb1.addPoint(v30.x, v30.y);
618     pathb1.addPoint(v26.x, v26.y);
619     pathb1.addPoint(v21.x, v21.y);
620     pathb1.addPoint(v17.x, v17.y);
621     pathb1.addPoint(barrio1.x, barrio1.y);
622 }
623
```

Pestaña 1: Programa 16/18

```
CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
Java
CODIGO_5_04_RainbowPaths_y_3Atractores
623
624 void newPathb2() { // PATH BICI
625     pathb2 = new Path();
626     pathb2.addPoint(v22.x, v22.y);
627     pathb2.addPoint(v18.x, v18.y);
628     pathb2.addPoint(v15.x, v15.y);
629     pathb2.addPoint(v12.x, v12.y);
630     pathb2.addPoint(v7.x, v7.y);
631     pathb2.addPoint(v20.x, v20.y);
632     pathb2.addPoint(v30.x, v30.y);
633     pathb2.addPoint(v37.x, v37.y);
634     pathb2.addPoint(v42.x, v42.y);
635     pathb2.addPoint(v49.x, v49.y);
636     pathb2.addPoint(v53.x, v53.y);
637     pathb2.addPoint(barrio2.x, barrio2.y);
638 }
639
640 void newPathb3() { // PATH BICI
641     pathb3 = new Path();
642     pathb3.addPoint(barrio3.x, barrio3.y);
643     pathb3.addPoint(v23.x, v23.y);
644     pathb3.addPoint(v35.x, v35.y);
645     pathb3.addPoint(v33.x, v33.y);
646     pathb3.addPoint(v29.x, v29.y);
647     pathb3.addPoint(v28.x, v28.y);
648     pathb3.addPoint(v25.x, v25.y);
649     pathb3.addPoint(v20.x, v20.y);
650     pathb3.addPoint(v16.x, v16.y);
651     pathb3.addPoint(v14.x, v14.y);
652     pathb3.addPoint(v10.x, v10.y);
653 }
654
655 // AV. LYON
656 void newPatha1() { // PATH AUTO
657     patha1 = new Path();
658     patha1.addPoint(v1.x, v1.y);
659     patha1.addPoint(v7.x, v7.y);
660     patha1.addPoint(v20.x, v20.y);
661     patha1.addPoint(v30.x, v30.y);
662     patha1.addPoint(v37.x, v37.y);
663
```

Pestaña 1: Programa 17/18

```
CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
Java
CODIGO_5_04_RainbowPaths_y_3Atractores
661 patha1.addPoint(v30.x, v30.y);
662 patha1.addPoint(v37.x, v37.y);
663 patha1.addPoint(v49.x, v49.y);
664 patha1.addPoint(v53.x, v53.y);
665 patha1.addPoint(comuna1.x, comuna1.y);
666 }
667
668 // AV. PROVIDENCIA
669 void newPatha2() { // PATH AUTO
670     patha2 = new Path();
671     patha2.addPoint(v44.x, v44.y);
672     patha2.addPoint(v39.x, v39.y);
673     patha2.addPoint(v35.x, v35.y);
674     patha2.addPoint(v33.x, v33.y);
675     patha2.addPoint(v29.x, v29.y);
676     patha2.addPoint(v28.x, v28.y);
677     patha2.addPoint(v25.x, v25.y);
678     patha2.addPoint(v20.x, v20.y);
679     patha2.addPoint(v16.x, v16.y);
680     patha2.addPoint(v14.x, v14.y);
681     patha2.addPoint(v10.x, v10.y);
682     patha2.addPoint(comuna2.x, comuna2.y);
683 }
684
685 // AUTO RESIDENTE
686 void newPatha3() { // PATH AUTO
687     patha3 = new Path();
688     patha3.addPoint(v7.x, v7.y);
689     patha3.addPoint(v12.x, v12.y);
690     patha3.addPoint(v25.x, v25.y);
691     patha3.addPoint(v28.x, v28.y);
692     patha3.addPoint(v36.x, v36.y);
693     patha3.addPoint(v30.x, v30.y);
694     patha3.addPoint(v37.x, v37.y);
695     patha3.addPoint(v42.x, v42.y);
696     patha3.addPoint(v43.x, v43.y);
697     patha3.addPoint(comuna3.x, comuna3.y);
698 }
699
700
```

Pestaña 1: Programa 18/18


```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
Java
CODIGO_5_04_RainbowPaths_y_3Atractores
1 class Auto extends Bici {
2
3   Auto(float x_, float y_) {
4     super(x_, y_);
5     velocity = new
6       PVector(random(0,0), random(0,0));
7     diam = 3;
8     maxspeed = 4;
9     maxforce = 0.15; // 0.05
10  }
11
12  // APARIENCIA
13  void display() {
14    //Vehicle is a triangle pointing in
15    // the direction of velocity; since
16    // it is drawn pointing up, we
17    // rotate it an additional 90°.
18    float theta =
19      velocity.heading() + PI/2;
20    fill(255, 0, 0);
21    noStroke();
22    pushMatrix();
23    translate(location.x, location.y);
24    rotate(theta);
25    rect(0, 0, diam, diam*3);
26    //beginShape();
27    //vertex(0, -diam*2);
28    //vertex(-diam, diam*2);
29    //vertex(diam, diam*2);
30    //endShape(CLOSE);
31    popMatrix();
32  }
33
34  // DESPLAZAMIENTO, VELOCIDAD
35  // Y ACELERACION
36  void update() {
37    velocity.add(acceleration);
38    velocity.limit(maxspeed);
39    location.add(velocity);
40    acceleration.mult(0);

```

Pestaña 2: Auto 1/3

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
Java
CODIGO_5_04_RainbowPaths_y_3Atractores
49   acceleration.mult(0);
50 }
51
52 // FUNCION DE FUNCIONES
53 void run(float x_, float y_) {
54   update();
55   display();
56   seek(new PVector(x_, y_));
57   //choqueMurallas();
58   //reboteEdificio();
59 }
60
61 // INTERACCION CON PERSONA
62 boolean choca(Persona p) {
63   float d = abs(dist(location.x,
64     location.y, p.location.x,
65     p.location.y));
66
67   if (d <= 10) {
68     return true;
69   } else {
70     return false;
71   }
72 }
73
74 // INTERACCION CON BICI
75 boolean choca(Bici b) {
76   float d = abs(dist(location.x,
77     location.y, b.location.x,
78     b.location.y));
79   if (d <= 10) {
80     return true;
81   } else {
82     return false;
83   }
84 }
85
86 // SEPARACION CON AUTOS
87 void separateA (ArrayList<Auto>
88   conductores) {
89   //Note how the desired separation

```

Pestaña 2: Auto 2/3

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
Java
CODIGO_5_04_RainbowPaths_y_3Atractores
76 // SEPARACION CON AUTOS
77 void separateA (ArrayList<Auto>
78   conductores) {
79   //Note how the desired separation
80   // is based on the Vehicle's size.
81   float desiredseparation = diam*20;
82   PVector sum = new PVector();
83   int count = 0;
84   for (Auto other : conductores) {
85     float d = PVector.dist(location,
86       other.location);
87     if ((d > 0) && (d <
88       desiredseparation)) {
89       PVector diff =
90         PVector.sub(location,
91           other.location);
92       diff.normalize();
93       //What is the magnitude of
94       // the PVector pointing away
95       // from the other vehicle?
96       // The closer it is, the more
97       // we should flee. The farther,
98       // the less. So we divide by
99       // the distance to weight
100      // it appropriately.
101      diff.div(d);
102      sum.add(diff);
103      count++;
104    }
105  }
106  if (count > 0) {
107    sum.div(count);
108    sum.normalize();
109    sum.mult(maxspeed);
110    PVector steer =
111      PVector.sub(sum, velocity);
112    steer.limit(maxforce);
113    applyForce(steer);
114  }
115 }

```

Pestaña 2: Auto 3/3

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
Java
CODIGO_5_04_RainbowPaths_y_3Atractores
1 class Bici extends Persona {
2
3   Bici(float x_, float y_) {
4     super(x_, y_);
5     velocity = new
6       PVector(random(0,0), random(0,0));
7     diam = 2.5;
8     maxspeed = 3;
9     maxforce = 0.15;
10
11  }
12
13  // APARIENCIA
14  void display() {
15    //Vehicle is a triangle pointing
16    //in the direction of velocity;
17    //since it is drawn pointing up,
18    // we rotate it an additional 90°
19
20    float theta =
21      velocity.heading() + PI/2;
22    fill(0, 100, 255);
23    noStroke();
24    pushMatrix();
25    translate(location.x, location.y);
26    rotate(theta);
27    beginShape();
28    vertex(0, -diam*2);
29    vertex(-diam, diam);
30    vertex(diam, diam);
31    endShape(CLOSE);
32    popMatrix();
33  }
34
35  // DESPLAZAMIENTO, VELOCIDAD
36  // Y ACELERACION
37  void update() {
38    velocity.add(acceleration);
39    velocity.limit(maxspeed);
40    location.add(velocity);

```

Pestaña 3: Bici 1/3

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
Java
CODIGO_5_04_RainbowPaths_y_3Atractores
40    location.add(velocity);
41    acceleration.mult(0);
42  }
43
44  // FUNCION DE FUNCIONES
45  void run(float x_, float y_) {
46    update();
47    display();
48    seek(new PVector(x_, y_));
49    //choqueMurallas();
50    //reboteEdificio();
51  }
52
53  // INTERACCION CON PERSONA
54  boolean choca(Persona p) {
55    float d =
56      abs(dist(location.x, location.y,
57        p.location.x, p.location.y));
58    if (d <= 10) {
59      return true;
60    } else {
61      return false;
62    }
63  }
64  void separateB (ArrayList<Bici>
65    ciclistas) {
66    //Note how the desired separation is
67    // based on the Vehicle's size.
68    float desiredseparation = diam*20;
69    PVector sum = new PVector();
70    int count = 0;
71    for (Bici other : ciclistas) {
72      float d =
73        PVector.dist(location,
74          other.location);
75      if ((d > 0) &&
76        (d < desiredseparation)) {
77        PVector diff =
78          PVector.sub(location,
79            other.location);

```

Pestaña 3: Bici 2/3

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
Java
CODIGO_5_04_RainbowPaths_y_3Atractores
66    //Note how the desired separation is
67    // based on the Vehicle's size.
68    float desiredseparation = diam*20;
69    PVector sum = new PVector();
70    int count = 0;
71    for (Bici other : ciclistas) {
72      float d =
73        PVector.dist(location,
74          other.location);
75      if ((d > 0) &&
76        (d < desiredseparation)) {
77        PVector diff =
78          PVector.sub(location,
79            other.location);
80      diff.normalize();
81      //What is the magnitude of the
82      // PVector pointing away from
83      // the other vehicle? The closer
84      // it is, the more we should
85      // flee. The farther, the less.
86      // So we divide by the distance
87      // to weight it appropriately.
88      diff.div(d);
89      sum.add(diff);
90      count++;
91    }
92  }
93  if (count > 0) {
94    sum.div(count);
95    sum.normalize();
96    sum.mult(maxspeed);
97    PVector steer =
98      PVector.sub(sum, velocity);
99    steer.limit(maxforce);
100    applyForce(steer);
101  }
102 }
103 }
104 }
105 }

```

Pestaña 3: Bici 3/3

```
CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
CODIGO_5_04_RainbowPaths_y_3Atractores
1 // Créditos a Daniel Shiffman :)
2 // The Nature of Code
3 // http://natureofcode.com
4
5 // Path Following
6
7 class Path {
8
9     // A Path is an arraylist of points
10    // (PVector objects)
11    ArrayList<PVector> points;
12    // A path has a radius, i.e how far is
13    // it ok for the boid to wander off
14    float radius;
15
16    Path() {
17        // Arbitrary radius of 12 meters
18        radius = 12;
19        points = new ArrayList<PVector>();
20    }
21
22    // Add a point to the path
23    void addPoint(float x, float y) {
24        PVector point = new PVector(x, y);
25        points.add(point);
26    }
27
28    PVector getStart() {
29        return points.get(0);
30    }
31
32    PVector getEnd() {
33        return points.get(points.size()-1);
34    }
35
36    // Draw the path
37    void display(color r_, color g_, color
38        // Draw thick line for radius
39        color r = r_;
40
```

Pestaña 4: Path 1/2

```
CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
CODIGO_5_04_RainbowPaths_y_3Atractores
22 // Add a point to the path
23 void addPoint(float x, float y) {
24     PVector point = new PVector(x, y);
25     points.add(point);
26 }
27
28 PVector getStart() {
29     return points.get(0);
30 }
31
32 PVector getEnd() {
33     return points.get(points.size()-1);
34 }
35
36 // Draw the path
37 void display(color r_, color g_, color
38 // Draw thick line for radius
39 color r = r_;
40 color g = g_;
41 color b = b_;
42 stroke(r,g,b, 40);
43 strokeWeight(radius*2);
44 noFill();
45 beginShape();
46 for (PVector v : points) {
47     vertex(v.x, v.y);
48 }
49 endShape();
50 // Draw thin line for center of path
51 stroke(0, 50);
52 strokeWeight(1);
53 noFill();
54 beginShape();
55 for (PVector v : points) {
56     vertex(v.x, v.y);
57 }
58 endShape();
59 }
60 }
61
```

Pestaña 4: Path 2/2

```
CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
CODIGO_5_04_RainbowPaths_y_3Atractores
1 class Persona {
2
3     PVector location;
4     PVector velocity;
5     PVector acceleration;
6     float diam;
7     float maxforce;
8     float maxspeed;
9     //float mass;
10
11    // POSICION DE ORIGEN,
12    // VELOCIDAD Y LIMITE DE VELOCIDAD
13    Persona(float x_, float y_) {
14        acceleration = new PVector(0, 0);
15        velocity =
16        new PVector(random(0, 0),random(0, 0);
17        location = new PVector(x_, y_);
18        diam = 3;
19        maxspeed = 1.5; // 1
20        maxforce = 0.01; // 0.01
21        //mass = 100;
22    }
23
24    // DESPLAZAMIENTO,
25    // VELOCIDAD Y ACELERACION
26    void update() {
27        velocity.add(acceleration);
28        velocity.limit(maxspeed);
29        location.add(velocity);
30        acceleration.mult(0);
31    }
32
33    // APLICADOR DE FUERZA
34    // Newton's second law; we could
35    // divide by mass if we wanted.
36    void applyForce(PVector force) {
37        acceleration.add(force);
38    }
39
40    void seek(PVector target) {
```

Pestaña 5: Persona 1/6

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
CODIGO_5_04_RainbowPaths_y_3Atractores
40 void seek(PVector target) {
41     PVector desired =
42         PVector.sub(target, location);
43     desired.normalize();
44     desired.mult(maxspeed);
45     PVector steer =
46         PVector.sub(desired, velocity);
47     steer.limit(maxforce);
48     applyForce(steer);
49 }
50
51 void seekPath(PVector target) {
52     PVector desired =
53         PVector.sub(target, location);
54     desired.normalize();
55     desired.mult(maxspeed);
56     PVector steer =
57         PVector.sub(desired, velocity);
58     steer.limit(maxforce);
59     steer.x = steer.x*5;
60     steer.y = steer.y*5;
61     applyForce(steer);
62 }
63
64 // FUNCION DE FUNCIONES (F. MAESTRA)
65 void run(float x_, float y_) {
66     update();
67     display();
68     seek(new PVector(x_, y_));
69     //choqueMurallas();
70     //reboteEdificio();
71 }
72
73 // APARIENCIA
74 void display() {
75     // Vehicle is a triangle pointing in
76     // the direction of velocity; since
77     // it is drawn pointing up, we rotate
78     // it an additional 90 degrees.
79     float theta =

```

Pestaña 5: Persona 2/6

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
CODIGO_5_04_RainbowPaths_y_3Atractores
79     float theta =
80         velocity.heading() + PI/2;
81     fill(0, 255, 0);
82     stroke(0, 255, 0);
83     strokeWeight(1);
84     pushMatrix();
85     translate(location.x, location.y);
86     rotate(theta);
87     ellipse(0, 0, diam, diam/2);
88     popMatrix();
89 }
90
91 // SI SALE DE LA VENTANA MUERE
92 boolean outOfBounds() {
93     if ((location.x > width + 15)
94         || (location.x < 0 - 15)
95         || (location.y > height + 15)
96         || (location.y < 0 - 15)) {
97         return true;
98     } else {
99         return false;
100 }
101 }
102
103 void separateP (ArrayList<Persona>
104                 peatones) {
105     //Note how the desired separation is
106     //based on the Vehicle's size.
107     float desiredseparation = diam*100;
108     PVector sum = new PVector();
109     int count = 0;
110     for (Persona other : peatones) {
111         float d =
112             PVector.dist(location,
113                          other.location);
114         if ((d > 0) &&
115             (d < desiredseparation)) {
116             PVector diff =
117                 PVector.sub(location,
118                             other.location);

```

Pestaña 5: Persona 3/6

```

CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
CODIGO_5_04_RainbowPaths_y_3Atractores
118         other.location);
119         diff.normalize();
120         // What is the magnitude of the
121         // PVector pointing away from
122         // the other vehicle? The closer
123         // it is, the more we should
124         // flee. The farther, the less.
125         // So we divide by the distance
126         // to weight it appropriately.
127         diff.div(d);
128         sum.add(diff);
129         count++;
130     }
131 }
132 if (count > 0) {
133     sum.div(count);
134     sum.normalize();
135     sum.mult(maxspeed);
136     PVector steer =
137         PVector.sub(sum, velocity);
138     steer.limit(maxforce);
139     applyForce(steer);
140 }
141 }
142
143 void follow(Path p) {
144
145     PVector predict = velocity.get();
146     predict.normalize();
147     predict.mult(50);
148     PVector predictpos =
149         PVector.add(location, predict);
150     PVector normal = null;
151     PVector target = null;
152     float worldRecord = 1000000;
153     // Start with a very high record
154     // distance that can easily
155     // be beaten
156     for (int i = 0;

```

Pestaña 5: Persona 4/6

```
CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
CODIGO_5_04_RainbowPaths_y_3Atractores
157   for (int i = 0;
158       i < p.points.size()-1; i++) {
159
160       PVector a = p.points.get(i);
161       PVector b = p.points.get(i+1);
162
163       PVector normalPoint =
164         getNormalPoint(predictpos, a, b)
165       if (normalPoint.x < a.x ||
166         normalPoint.x > b.x) {
167         normalPoint = b.get();
168       }
169       float distance =
170         PVector.dist(predictpos,
171           normalPoint);
172       if (distance < worldRecord) {
173         worldRecord = distance;
174         normal = normalPoint;
175
176         PVector dir = PVector.sub(b, a);
177         dir.normalize();
178         dir.mult(10);
179         target = normalPoint.get();
180         target.add(dir);
181       }
182     }
183     if (worldRecord > p.radius) {
184       seekPath(target);
185     }
186   }
187   // A function to get the normal point
188   // from a point (p) to a line segment
189   // (a-b)
190   // This function could be optimized to
191   // make fewer new Vector objects
192   PVector getNormalPoint(PVector p,
193     PVector a, PVector b) {
194     // Vector from a to p
195     PVector ap = PVector.sub(p, a);
196     // Vector from a to b
```

Pestaña 5: Persona 5/6

```
CODIGO_5_04_RainbowPaths_y_3Atractores | Pr...
CODIGO_5_04_RainbowPaths_y_3Atractores
175
176     PVector dir = PVector.sub(b, a);
177     dir.normalize();
178     dir.mult(10);
179     target = normalPoint.get();
180     target.add(dir);
181   }
182 }
183 if (worldRecord > p.radius) {
184   seekPath(target);
185 }
186 }
187 // A function to get the normal point
188 // from a point (p) to a line segment
189 // (a-b)
190 // This function could be optimized to
191 // make fewer new Vector objects
192 PVector getNormalPoint(PVector p,
193   PVector a, PVector b) {
194   // Vector from a to p
195   PVector ap = PVector.sub(p, a);
196   // Vector from a to b
197   PVector ab = PVector.sub(b, a);
198   ab.normalize();
199   // Normalize the line
200   // Project vector "diff" onto line
201   // by using the dot product
202   ab.mult(ap.dot(ab));
203   PVector normalPoint =
204     PVector.add(a, ab);
205   return normalPoint;
206 }
207 }
208
209
210
211
212
213
214
```

Pestaña 5: Persona 6/6

VERSIÓN 05

material complementario

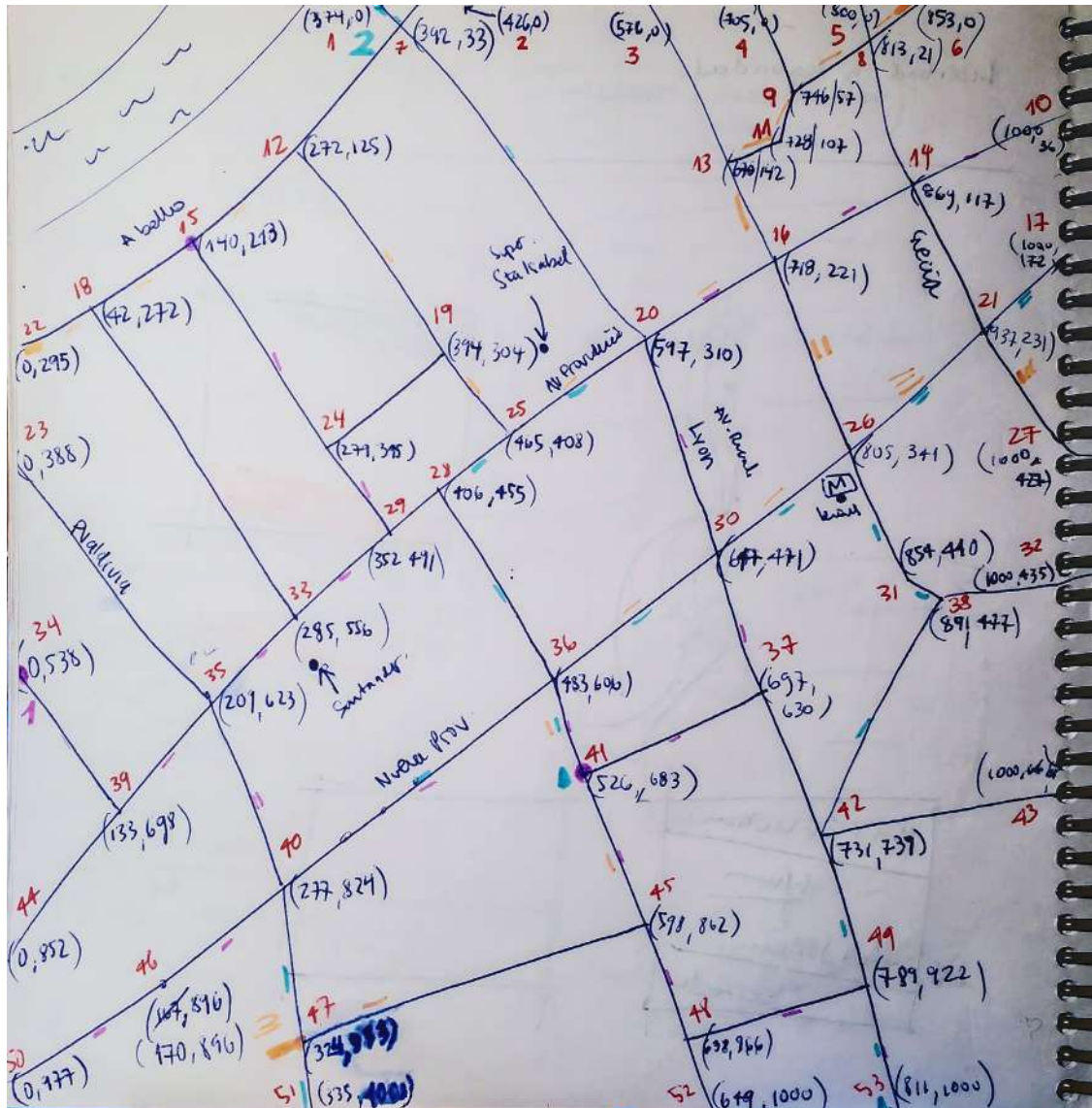


MAPA REAL SATELITAL

Para hacer más liviano el programa se rediseñaron las capas del mapa, zonas y la simbología. Al hacer más liviana la función este correría más rápido, en especial en el caso de la interacción del usuario. Como por ejemplo la aparición de la distinción entre zonas. (f.3)

La simbología (f.2) era una imagen sólida que solo carecía de los números identificadores de cantidades dado que estos al ser unos elementos variables tenían que ser reescritos en tiempo real.

Las calles/paths del mapa debido a que no se logró importarlos como vectores; fueron escritos a mano identificando cada una de las intersecciones o curvas pronunciadas. Resultó en una fidedigna interpretación del mapa, sin embargo no es el sistema más eficiente en caso de querer escalar el mapa, o bien cambiar de ubicación. (f.1) Todas las coordenadas (P1, 52-105) están numeradas de izquierda a derecha, de arriba hacia abajo. Fueron escritas usando la función llamada mouseX y mouseY, que dan la posición del cursor. (P1, 205-207)



f.1

mapa de coordenadas
producido artesanalmente



f.2



f.3

mapa coloreado
según zonas distintas

VERSIÓN 05

funcionamiento,
observaciones y
conclusiones de las
funciones más relevantes



1. APARIENCIA

El primer y más notorio cambio es el uso de un mapa real de fondo. Después de ser extraído de Google Maps el mapa fue ajustado y editado en Photoshop para acentuar más los edificios y calles. Luego se crearon calles más anchas e iluminadas para contrastar de mejor manera con los agentes. (PI, 9, 136-137, 181-187)

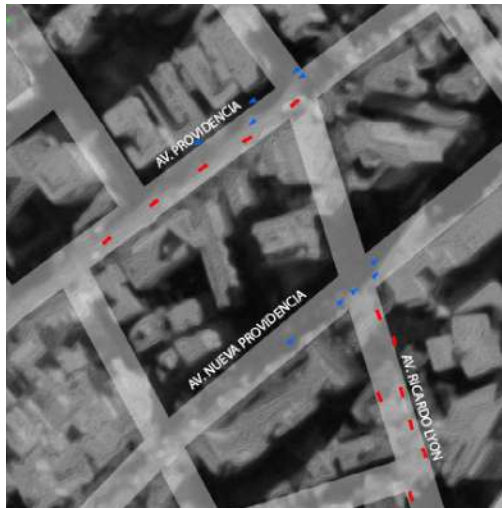
El mapa con escala real le permite al usuario comprender como está siendo redistribuido el espacio de la ciudad según la cantidad de agentes que este mismo decide ingresar. Entiende también de mejor manera la proxémica y el comportamiento intermodal en vivo.



2. PUNTOS DE ORIGEN

Usando las coordenadas del mapa hecho a mano se determinaron 3 puntos de origen para cada uno de los 3 grupos de agentes programados. (PI, 250-280)

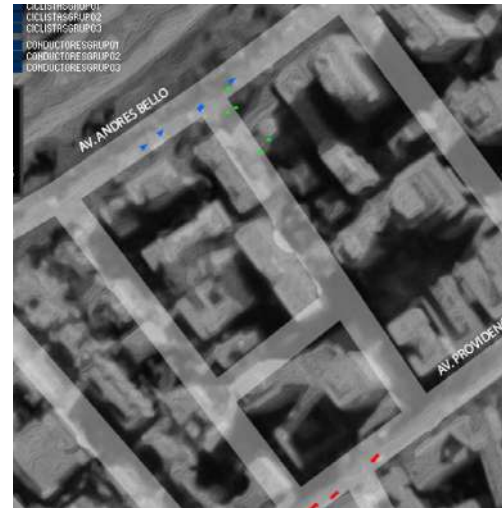
Los agentes aparecen azorosamente en un área de **10 x 10** píxeles alrededor del punto vector del inicio o fin de los path. Esto le da cierta controlada impredecibilidad que permite crear diferentes tiempos de partida para cada uno de los agentes que se encaminan a diferentes puntos en el mapa. Editar tal área de aparición es posible pero no acercaría al simulador a su desempeño deseado.



3. AGRUPACIONES

Los grupos de agentes se mueven como tal, parecen pequeños cardúmenes que se van encontrando con otras caravanas de agentes. La emergencia comienza a surgir en el comportamiento programado de los agentes. Le quita un grado de realismo entrópico, mas no es relevante para el fin último del simulador.

En la ciudad se espera que las personas compartan ciertas partes de sus caminos y algunas veces el destino, en este caso en pos de beneficiar la estructura y eficiencia del código del simulador se tratan las funciones de los agentes por distintas agrupaciones.



4. ENCUENTROS

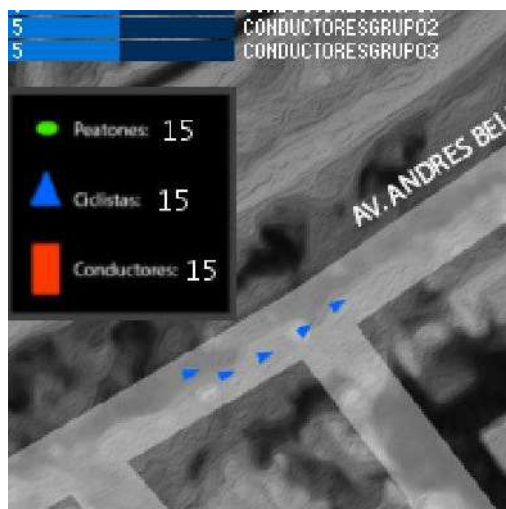
Se siguen respetando las mismas reglas de proxémica heredada del simulador anterior. La gran diferencia recae en que son grupos de agentes contra grupos de agentes, lo cual puede producir pequeños momentos de inmovilidad para un auto por ejemplo, por que este debe cederle la pasada a todo el conjunto de peatones o ciclistas. Sin embargo, esto podría simular situaciones extremas como lo son una maratón o un cruce peatonal, o un conjunto de escolares.



5. SLIDERS

Se importó la librería control.P5 (*PI*, 4-6) para poder hacer sliders que controlarían la cantidad de cada uno de los grupos de cada tipo de agente. Estos sliders están escritos en *PI*, 43-153

Considerando que eventualmente hay que agregar al transporte público a la fórmula, puede ser un poco excesivo tener controladores para cada uno de los grupos de agentes, en vez de poder controlarlos todos desde un slider por tipo. Uno para los peatones, otro para los ciclistas, otro para la cantidad de autos y otro para los futuros buses. Esto podría afectar la intuitividad a la hora de interactuar del usuario.



6. SIMBOLOGÍA

La simbología se actualizó debido al cambio de las figuras que tiene cada tipo de agentes. Al igual que la versión anterior, es un imagen de fondo a la cual se le agregan los números que se actualizan de manera dinámica en el programa. A medida que el funcionario mueve con el cursor el slider de cantidad de agentes el número se va actualizando. El número representa la cantidad total de agentes por tipo. (PI, 12,196-203)

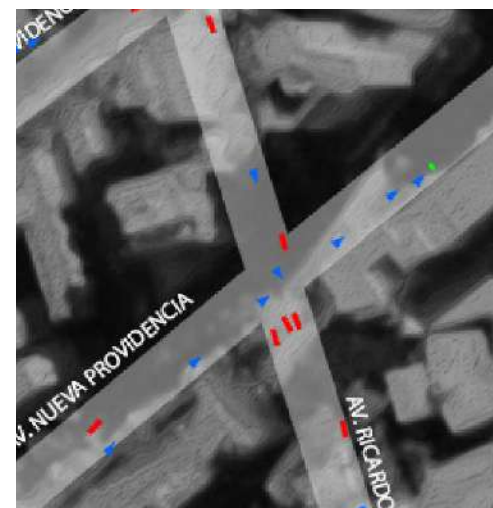
Si el slider marca 5 agentes el total será el triple debido a que es número en realidad domina la cantidad de agentes en cada uno de tres los grupos.



7. ZONAS

Al hacer click con el mouse en cualquier zona de la pantalla la imagen de fondo cambia por un fondo con las zonas de tránsito en naranja y o las residenciales en verde. (PI, 10-II,138-139,181-187)

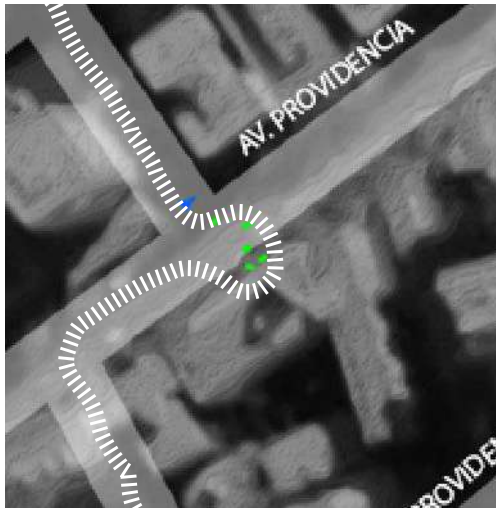
La interacción del programa se a reducido a el uso exclusivo del mouse para evitar hacer interferencias entre 2 o más funciones. Esto simplifica la intuitividad de uso del simulador.



7. ATRACTORES Y CALLES

Se crearon dos función de atracción, una con una fuerza fuerte, para las calles y una con una débil, para seguir el objetivo/destino.

Esto permitió que los agentes primero entrar a las calles para después moverse por ellas, así como también permitió omitir las líneas de código que le correspondía a los edificios como repulsores.



9. ERROR ACUMULACIÓN

A veces por curvas cerradas los agentes se mantienen estancados por más tiempo en las esquinas. Quizás, la atracción por la calles se duplicaba y superaba la atracción que tenía el atractor con creces, valga la redundancia.

La capacidad de giro de los agentes se ve dominada por el valor de la 'maxforce', una variable declarada al comienzo de cada una de las clases. Es posible hacer de esta una opción manipulable por el usuario, pero no va al caso controlar con tanto detalle el comportamiento.



10. ERROR CURVA AMPLIA

En algunos casos el predictor notaba que el agente iba en buena dirección a la vez que este estaba en una curva, entonces el movimiento terminaba siendo una curva más amplia y menos ortogonal como lo eran las calles. Básicamente se acortaba el camino de un modo no deseado.

Corregir programáticamente este detalle conlleva un costo mayor y no era una prioridad frente a otros objetivos como lo son la interacción y los indicadores dinámicos, u otra función orientada hacia la usabilidad del programa.

CONCLUSIÓN GENERAL

Lo más rescatable del programa es la doble función de seguimiento, con un seek fuerte para las calles y uno débil para seguir el objetivo/destino. Lo cual le permite a los agentes primero entrar a las calles y después moverse por ellas manteniendo la separación entre sí.

El fragmento elegido de providencia no es sólo un buen conjunto de intersecciones sino también un área de intersección de comunas, es decir de flujos. Esto lo convierte en un interesante caso de estudio y simulación dada la diversidad de agentes. Distinto atractores se repartieron por fuera del mapa para generar los flujos de los distintos agentes.

Las figuras pueden distinguirse más luego de los ajustes de escala. Los agentes son un poco más grandes que lo que deberían pero se leen correctamente como tal. Y los puntos de origen de estos fueron sacados de su grilla y puestos en lugares estratégicos por fuera de la ventana al inicio de paths.

Se agregaron sliders que permiten la directa interacción de un usuario. Estos controlan la cantidad de agentes lo cual repercute directamente en parámetros de impacto en la ciudad.

PRÓXIMOS PASOS

1. Agregar buses, esto podría significar un considerable ralentizamiento del programa, pero no se puede omitir una parte crucial del ecosistema de transporte como lo son los buses.

2. Separar peatones de buses por zonas de tránsito y residenciales. Aún falta comunicar de alguna forma que buses van por unas vías especializadas para ellos, en la cual ellos se llevan el “trono” en la jerarquía de sustentabilidad.

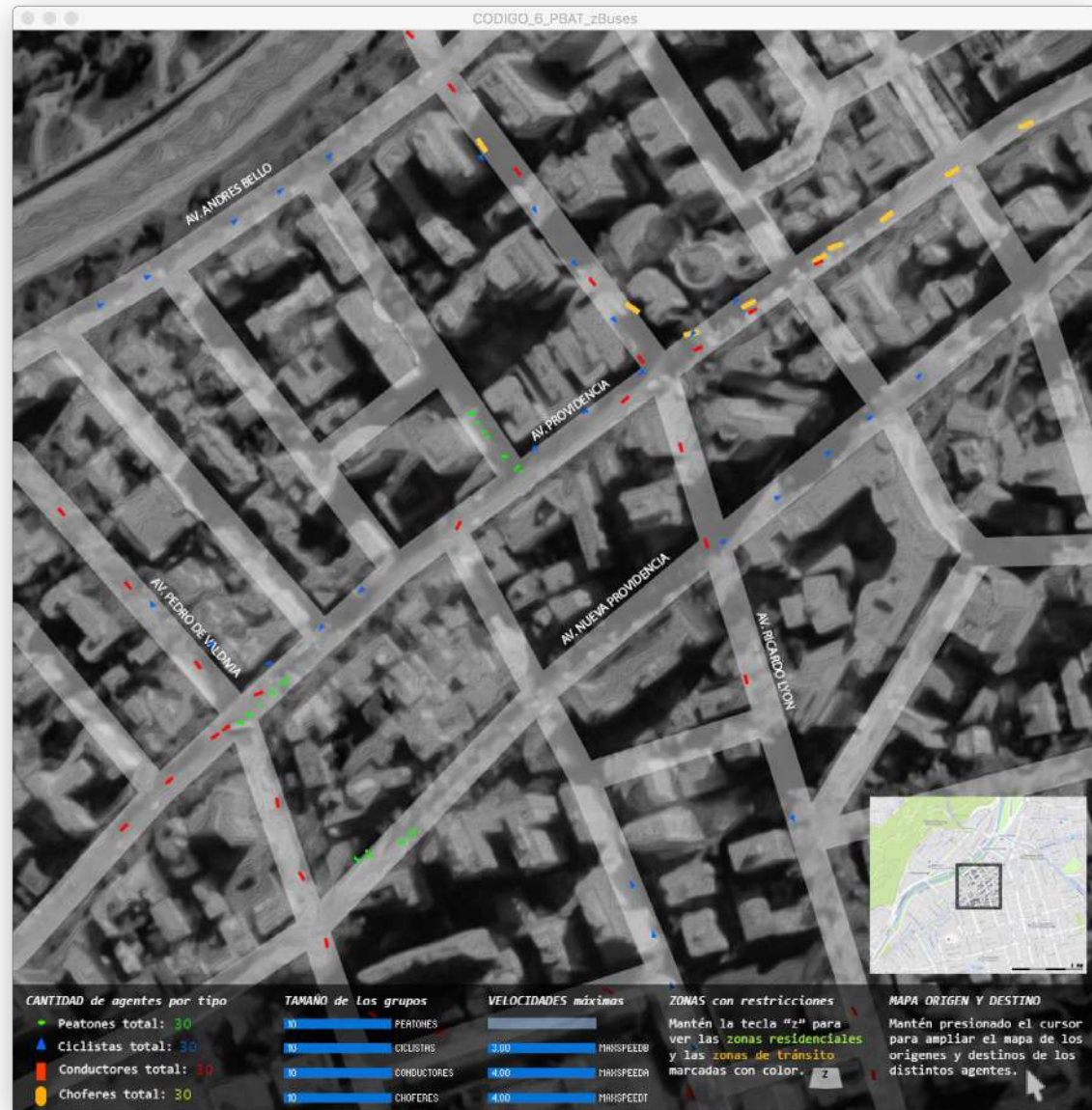
3. Mejorar interfaz: Cantidades, Velocidades, Agrupar agentes, Mejorar sistema de simbología, zonas versátiles, mapa de orígenes y atractores, y titular simulador. Mejorar la interacción del usuario desde la composición, orden de lectura, contraste y el equilibrio cromático.

4. Numerador de área de superficie, como indicador de impacto de sustentabilidad, valor totalmente necesario para la planificación urbana.

5. Simplificar código, hacerlo más eficiente a la hora de proceder por loops e iteraciones varias. Esto mejoraría la interacción con el usuario y los tiempos de espera. Además de liberarle costos computacionales al dispositivo en uso.

VERSIÓN 06*

la con aspiraciones
de grandeza



* SÁLTESE A LA PÁGINA 124 PARA VER LA SIGUIENTE Y MENOS ÁRIDA SECCIÓN

```
CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
1 // IMPORTACION DE LIBRERIA DE SLIDERS
2 import controlP5.*;
3 ControlP5 cp5;
4
5 // IMAGENES DE FONDO, MAPA Y SIMBOLOGIA
6 PImage fondo;
7 PImage fondoColor;
8 boolean zonasClick = false;
9 PImage simbologia;
10 PImage mapachico;
11 PImage mapagrande;
12
13 // INTERRUPTORES (FUNCIONES BOOLEANAS)
14 boolean mapClick = false;
15 boolean debug = true;
16
17 // LISTA DE PATHS
18 Path pathp1;
19 Path pathp2;
20 Path pathp3;
21 Path pathb1;
22 Path pathb2;
23 Path pathb3;
24 Path patha1;
25 Path patha2;
26 Path patha3;
27 Path pathT1;
28
29 // LISTAS DE AGENTES
30 ArrayList<Persona> peatones;
31 ArrayList<Persona> peatones2;
32 ArrayList<Persona> peatones3;
33 ArrayList<Bici> ciclistas;
34 ArrayList<Bici> ciclistas2;
35 ArrayList<Bici> ciclistas3;
36 ArrayList<Auto> conductores;
37 ArrayList<Auto> conductores2;
38 ArrayList<Auto> conductores3;
39 ArrayList<TransportePublico> choferes;
40
```

Guardado finalizado.

Pestaña 1: Programa 1/20

```
CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
41 // NUMERO MAX. DE AGENTES (PROPORCIONES)
42 int Peatones = 5;
43 int Ciclistas = 5;
44 int Conductores = 5;
45 int Choferes = 5;
46
47 // VELOCIDADES MAXIMAS DE AGENTES
48 float maxspeedP = 1.5;
49 float maxspeedB = 3;
50 float maxspeedA = 4;
51 float maxspeedT = 4;
52
53 //COORDS MAPA PROVIDENCIA
54 PVector v1 = new PVector(374, 0);
55 PVector v2 = new PVector(426, 0);
56 PVector v3 = new PVector(576, 0);
57 PVector v4 = new PVector(705, 0);
58 PVector v5 = new PVector(800, 0);
59 PVector v6 = new PVector(853, 0);
60 PVector v7 = new PVector(390, 33);
61 PVector v8 = new PVector(813, 21);
62 PVector v9 = new PVector(746, 57);
63 PVector v10 = new PVector(1000, 36);
64 PVector v11 = new PVector(728, 107);
65 PVector v12 = new PVector(272, 125);
66 PVector v13 = new PVector(670, 147);
67 PVector v14 = new PVector(869, 117);
68 PVector v15 = new PVector(140, 213);
69 PVector v16 = new PVector(718, 221);
70 PVector v17 = new PVector(1000, 172);
71 PVector v18 = new PVector(42, 272);
72 PVector v19 = new PVector(349, 304);
73 PVector v20 = new PVector(597, 310);
74 PVector v21 = new PVector(937, 231);
75 PVector v22 = new PVector(0, 295);
76 PVector v23 = new PVector(0, 388);
77 PVector v24 = new PVector(279, 395);
78 PVector v25 = new PVector(465, 400);
79 PVector v26 = new PVector(805, 341);
80 PVector v27 = new PVector(1000, 427);
```

Guardado finalizado.

Pestaña 1: Programa 2/20

```
CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
81 PVector v28 = new PVector(406, 455);
82 PVector v29 = new PVector(352, 491);
83 PVector v30 = new PVector(647, 471);
84 PVector v31 = new PVector(54, 440);
85 PVector v32 = new PVector(1000, 435);
86 PVector v33 = new PVector(285, 556);
87 PVector v34 = new PVector(0, 538);
88 PVector v35 = new PVector(209, 623);
89 PVector v36 = new PVector(483, 606);
90 PVector v37 = new PVector(697, 630);
91 PVector v38 = new PVector(891, 477);
92 PVector v39 = new PVector(133, 698);
93 PVector v40 = new PVector(277, 824);
94 PVector v41 = new PVector(526, 683);
95 PVector v42 = new PVector(731, 739);
96 PVector v43 = new PVector(1000, 666);
97 PVector v44 = new PVector(0, 852);
98 PVector v45 = new PVector(598, 862);
99 PVector v46 = new PVector(170, 896);
100 PVector v47 = new PVector(324, 953);
101 PVector v48 = new PVector(638, 966);
102 PVector v49 = new PVector(789, 922);
103 PVector v50 = new PVector(0, 977);
104 PVector v51 = new PVector(335, 1000);
105 PVector v52 = new PVector(649, 1000);
106 PVector v53 = new PVector(811, 1000);
107
108
109 // ATRACTORES Y SUS COORDENADAS
110 float diam = 15;
111 // ATRACTORES DE PERSONAS
112 PVector cuadra1 = new PVector(v53.x, v53
113 PVector cuadra2 = new PVector(v52.x, v52
114 PVector cuadra3 = new PVector(v17.x+100,
115 // ATRACTORES DE BICIS
116 PVector barrio1 = new PVector(v17.x+100,
117 PVector barrio2 = new PVector(v53.x, v53
118 PVector barrio3 = new PVector(v23.x-100,
119 // ATRACTORES DE AUTOS Y BUSES
120 PVector comuna1 = new PVector(v53.x, v53
```

Guardado finalizado.

Pestaña 1: Programa 3/20

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
121 PVector comuna2 = new PVector(v10.x+100,
122 PVector comuna3 = new PVector(v52.x, v52.y);
123
124
125 // SETUP //
126 void setup() {
127   size(1000, 1000, P2D);
128   smooth();
129   frameRate(25);
130   // CARGA DE IMAGENES EN PROGRAMA
131   fondo = loadImage("FondoGris.jpg");
132   fondocolor = loadImage("FondoColor.jpg");
133   simbologia = loadImage("SimbologiaBanner.png");
134   mapachico = loadImage("MapaChico.png");
135   mapagrande = loadImage("MapaGrande.png");
136
137   // SLIDERS INTERACTIVOS
138   // SLIDER DE CANTIDAD
139   cp5 = new ControlP5(this);
140   cp5.addSlider("Peatonales").setPosition(0, 100);
141   cp5.addSlider("Ciclistas").setPosition(0, 100);
142   cp5.addSlider("Conductores").setPosition(0, 100);
143   cp5.addSlider("Choferes").setPosition(0, 100);
144   // SLIDER DE VELOCIDAD
145   //cp5.addSlider("maxspeedP").setPosition(0, 100);
146   cp5.addSlider("maxspeedB").setPosition(0, 100);
147   cp5.addSlider("maxspeedA").setPosition(0, 100);
148   cp5.addSlider("maxspeedT").setPosition(0, 100);
149
150   // CREACION DE PATHS
151   newPath1();
152   newPath2();
153   newPath3();
154   newPathb1();
155   newPathb2();
156   newPathb3();
157   newPatha1();
158   newPatha2();
159   newPatha3();
160   newPatht1();

```

Guardado finalizado.

Pestaña 1: Programa 4/20

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
161 //newPatht2();
162 //newPatht3();
163
164 // SETED DE AGENTES
165 peatonales = new ArrayList<Persona>();
166 peatonales2 = new ArrayList<Persona>();
167 peatonales3 = new ArrayList<Persona>();
168 ciclistas = new ArrayList<Bici>();
169 ciclistas2 = new ArrayList<Bici>();
170 ciclistas3 = new ArrayList<Bici>();
171 conductores = new ArrayList<Auto>();
172 conductores2 = new ArrayList<Auto>();
173 conductores3 = new ArrayList<Auto>();
174 choferes = new ArrayList<TransportePublico>();
175 //choferes2 = new ArrayList<TransportePublico>();
176 //choferes3 = new ArrayList<TransportePublico>();
177 }
178
179 // D R A W //
180 void draw() {
181   // FONDO Y ZONAS DE TRANSITO Y RESIDENCIALES
182   if (keyPressed) {
183     if (key == 'z' || key == 'Z') {
184       image(fondocolor, 0, 0);
185     }
186   } else {
187     image(fondo, 0, 0);
188   }
189
190   // INFO COORDS MOUSE
191   println(mouseX, mouseY);
192
193   // FUNCIONES VISUALES DE LOS PATHS
194   //path1.display(0, 255, 0);
195   //path2.display(0, 255, 0);
196   //path3.display(0, 255, 0);
197   //pathb1.display(0, 0, 255);
198   //pathb2.display(0, 0, 255);
199   //pathb3.display(0, 0, 255);
200

```

Guardado finalizado.

Pestaña 1: Programa 5/20

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
201 //path1.display(255, 0, 0);
202 //path2.display(255, 0, 0);
203 //path3.display(255, 0, 0);
204
205 // FUNCION AGENTE FOLLOWS PATH N°
206 for (Persona p : peatonales) {
207   p.follow(pathp1);
208 }
209 for (Persona p2 : peatonales2) {
210   p2.follow(pathp2);
211 }
212 for (Persona p3 : peatonales3) {
213   p3.follow(pathp3);
214 }
215 for (Bici b : ciclistas) {
216   b.follow(pathb1);
217 }
218 for (Bici b2 : ciclistas2) {
219   b2.follow(pathb2);
220 }
221 for (Bici b3 : ciclistas3) {
222   b3.follow(pathb3);
223 }
224 for (Auto a : conductores) {
225   a.follow(patha1);
226 }
227 for (Auto a2 : conductores2) {
228   a2.follow(patha2);
229 }
230 for (Auto a3 : conductores3) {
231   a3.follow(patha3);
232 }
233 for (TransportePublico t : choferes) {
234   t.follow(patht1);
235 }
236
237 // ORIGENES DE PERSONAS, BICIS,
238 // AUTOS Y TRANSPORTE PUBLICO
239 peatonales.add(new Persona(random(v34.x-100, v34.x+100),
240   random(0, 255), 0));

```

Guardado finalizado.

Pestaña 1: Programa 6/20


```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
241 peatones3.add(new Persona(random(v50.x-
242 ciclistas.add(new Bici(random(v52.x-10
243 ciclistas2.add(new Bici(random(v22.x-10
244 ciclistas3.add(new Bici(random(v10.x-10
245 conductores.add(new Auto(random(v1.x-10
246 conductores2.add(new Auto(random(v44.x-
247 conductores3.add(new Auto(random(v23.x-
248 choferes.add(new TransportePublico(ran
249
250 // CANTIDADES DE AGENTES
251 if (peatones.size() > Peatones) {
252     peatones.remove(Peatones);
253 }
254 if (peatones2.size() > Peatones) {
255     peatones2.remove(Peatones);
256 }
257 if (peatones3.size() > Peatones) {
258     peatones3.remove(Peatones);
259 }
260 if (ciclistas.size() > Ciclistas) {
261     ciclistas.remove(Ciclistas);
262 }
263 if (ciclistas2.size() > Ciclistas) {
264     ciclistas2.remove(Ciclistas);
265 }
266 if (ciclistas3.size() > Ciclistas) {
267     ciclistas3.remove(Ciclistas);
268 }
269 if (conductores.size() > Conductores) {
270     conductores.remove(Conductores);
271 }
272 if (conductores2.size() > Conductores) {
273     conductores2.remove(Conductores);
274 }
275 if (conductores3.size() > Conductores) {
276     conductores3.remove(Conductores);
277 }
278 if (choferes.size() > Choferes) {
279     choferes.remove(Choferes);
280 }
Guardado finalizado.

```

Pestaña 1: Programa 7/20

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
281
282 // INTERACCION ENTRE AGENTES
283 // VERSIONE MUCHO MAS EFICIENTE!
284 for (int i = peatones.size()-1; i >= 0
285 for (int j = ciclistas.size()-1; j >= 0
286 for (int k = conductores.size()-1; k >= 0
287 for (int l = choferes.size()-1; l >= 0
288 Persona p = peatones.get(i);
289 Persona p2 = peatones2.get(i);
290 Persona p3 = peatones3.get(i);
291 Bici b = ciclistas.get(j);
292 Bici b2 = ciclistas2.get(j);
293 Bici b3 = ciclistas3.get(j);
294 Auto a = conductores.get(k);
295 Auto a2 = conductores2.get(k);
296 Auto a3 = conductores3.get(k);
297 TransportePublico t = choferes.get(l)
298
299 // (B1)
300 if (b.choca(p) || b.choca(p2)
301     || b.choca(p3) || b.choca(t)) {
302     b.velocity.x = constrain(b.velocity.x
303     b.velocity.y = constrain(b.velocity.y
304     }
305 // (B2)
306 if (b2.choca(p) || b2.choca(p2)
307     || b2.choca(p3) || b2.choca(t)) {
308     b2.velocity.x = constrain(b2.velocity
309     b2.velocity.y = constrain(b2.velocity
310     }
311 // (B3)
312 if (b3.choca(p) || b3.choca(p2)
313     || b3.choca(p3) || b3.choca(t)) {
314     b3.velocity.x = constrain(b3.velocity
315     b3.velocity.y = constrain(b3.velocity
316     }
317 // (A1)
318 if (a.choca(p) || a.choca(p2)
319     || a.choca(p3) || a.choca(b)
320     || a.choca(b2) || a.choca(b3)
Guardado finalizado.

```

Pestaña 1: Programa 8/20

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
321     || b.choca(t)) {
322     a.velocity.x = constrain(a.velocity.x
323     a.velocity.y = constrain(a.velocity.y
324     }
325 // (A2)
326 if (a2.choca(p) || a2.choca(p2)
327     || a2.choca(p3) || a2.choca(b)
328     || a2.choca(b2) || a2.choca(b3)
329     || b.choca(t)) {
330     a2.velocity.x = constrain(a2.velocity
331     a2.velocity.y = constrain(a2.velocity
332     }
333 // (A3)
334 if (a3.choca(p) || a3.choca(p2)
335     || a3.choca(p3) || a3.choca(b)
336     || a3.choca(b2) || a3.choca(b3)
337     || b.choca(t)) {
338     a3.velocity.x = constrain(a3.velocity
339     a3.velocity.y = constrain(a3.velocity
340     }
341     }
342     }
343     }
344 }
345
346 // FUNCIONES PERSONAS (1)
347 for (int i = peatones.size()-1;
348     i >= 0; i--) {
349     Persona p = peatones.get(i);
350     p.run(cuadral.x, cuadral.y);
351     fill(0, 255, 0);
352     ellipse(cuadral.x, cuadral.y, diam, c
353     // SI SALE DE LA VENTANA MUERE
354     if (p.outOfBounds()) {
355         peatones.remove(i);
356     }
357 }
358 }
359 // FUNCION DISTANCIA PERSONA 1
360 // CON PERSONA 2 y 3
Guardado finalizado.

```

Pestaña 1: Programa 9/20

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
361 for (Persona p : peatones) {
362     p.separateP(peatones);
363     p.separateP(peatones2);
364     p.separateP(peatones3);
365 }
366
367 // FUNCIONES PERSONAS (2)
368 for (int i = peatones2.size()-1;
369      i >= 0; i--) {
370     Persona p2 = peatones2.get(i);
371     p2.run(cuadra2.x, cuadra2.y);
372     fill(0, 255, 0);
373     ellipse(cuadra2.x, cuadra2.y, diam, c
374 // SI SALE DE LA VENTANA MUERE
375     if (p2.outOfBounds()) {
376         peatones2.remove(i);
377     }
378 }
379 // FUNCION DISTANCIA PERSONA 2
380 // CON PERSONAS 1 y 3
381 for (Persona p2 : peatones2) {
382     p2.separateP(peatones);
383     p2.separateP(peatones2);
384     p2.separateP(peatones3);
385 }
386 // FUNCIONES PERSONAS (3)
387 for (int i = peatones3.size()-1;
388      i >= 0; i--) {
389     Persona p3 = peatones3.get(i);
390     p3.run(cuadra3.x, cuadra3.y);
391     fill(0, 255, 0);
392     ellipse(cuadra3.x, cuadra3.y, diam, c
393 // SI SALE DE LA VENTANA MUERE
394     if (p3.outOfBounds()) {
395         peatones3.remove(i);
396     }
397 }
398 // FUNCION DISTANCIA PERSONA 3
399 // CON PERSONAS 1 y 2
400 for (Persona p3 : peatones3) {

```

Guardado finalizado.

Pestaña 1: Programa 10/20

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
401     p3.separateP(peatones);
402     p3.separateP(peatones2);
403     p3.separateP(peatones3);
404 }
405
406 // FUNCIONES TRANSPORTE PUBLICO
407 // FUNCIONES T.PUBLICICO (1)
408 for (int i = choferes.size()-1;
409      i >= 0; i--) {
410     TransportePublico t =
411         choferes.get(i);
412     t.run(comuna1.x, comuna1.y);
413     fill(200, 255, 0);
414     ellipse(comuna1.x, comuna1.y, diam, c
415 // SI SALE DE LA VENTANA MUERE
416     if (t.outOfBounds()) {
417         choferes.remove(i);
418     }
419 }
420 // FUNCION DISTANCIA T.PUBLICICO
421 for (TransportePublico t : choferes) {
422     t.separateP(peatones);
423     t.separateP(peatones2);
424     t.separateP(peatones3);
425     t.separateT(choferes);
426 }
427
428 // FUNCIONES BICIS (1)
429 for (int i = ciclistas.size()-1;
430      i >= 0; i--) {
431     Bici b = ciclistas.get(i);
432     b.run(barrio1.x, barrio1.y);
433     fill(0, 100, 255);
434     ellipse(barrio1.x, barrio1.y, diam, c
435 //SI SALE DE LA VENTANA MUERE
436     if (b.outOfBounds()) {
437         ciclistas.remove(i);
438     }
439 }
440 }

```

Guardado finalizado.

Pestaña 1: Programa 11/20

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
441 // FUNCION DISTANCIA BICI 1
442 // CON OTRAS BICIS Y PERSONAS
443 for (Bici b : ciclistas) {
444     b.separateP(peatones);
445     b.separateP(peatones2);
446     b.separateP(peatones3);
447     b.separateB(ciclistas);
448     b.separateB(ciclistas2);
449     b.separateB(ciclistas3);
450     b.separateT(choferes);
451 }
452
453 // FUNCIONES BICIS (2)
454 for (int i = ciclistas2.size()-1;
455      i >= 0; i--) {
456     Bici b2 = ciclistas2.get(i);
457     b2.run(barrio2.x, barrio2.y);
458     fill(0, 100, 255);
459     ellipse(barrio2.x, barrio2.y, diam, c
460 //SI SALE DE LA VENTANA MUERE
461     if (b2.outOfBounds()) {
462         ciclistas2.remove(i);
463     }
464 }
465 // FUNCION DISTANCIA BICI 2
466 // CON OTRAS BICIS Y PERSONAS
467 for (Bici b2 : ciclistas2) {
468     b2.separateP(peatones);
469     b2.separateP(peatones2);
470     b2.separateP(peatones3);
471     b2.separateB(ciclistas);
472     b2.separateB(ciclistas2);
473     b2.separateB(ciclistas3);
474     b2.separateT(choferes);
475 }
476
477 // FUNCIONES BICIS (3)
478 for (int i = ciclistas3.size()-1;
479      i >= 0; i--) {
480     Bici b3 = ciclistas3.get(i);

```

Guardado finalizado.

Pestaña 1: Programa 12/20

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
481 b3.run(barrio3.x, barrio3.y);
482 fill(0, 100, 255);
483 ellipse(barrio3.x, barrio3.y, diam, c
484 //SI SALE DE LA VENTANA MUERE
485 if (b3.outOfBounds()) {
486     ciclistas3.remove(i);
487 }
488 }
489 // FUNCION DISTANCIA BICI 3
490 // CON OTRAS BICIS Y PERSONAS
491 for (Bici b3 : ciclistas3) {
492     b3.separateP(peatonos);
493     b3.separateP(peatonos2);
494     b3.separateP(peatonos3);
495     b3.separateB(ciclistas);
496     b3.separateB(ciclistas2);
497     b3.separateB(ciclistas3);
498     b3.separateT(choferes);
499 }
500
501 // FUNCIONES AUTO SOLO (1)
502 for (int i = conductores.size()-1;
503      i >= 0; i--) {
504     Auto a = conductores.get(i);
505     a.run(comunal.x, comunal.y);
506     fill(255, 0, 0);
507     ellipse(comunal.x, comunal.y, diam, c
508
509     // SI SALE DE LA VENTANA MUERE
510     if (a.outOfBounds()) {
511         conductores.remove(i);
512     }
513 } //(1)
514 // FUNCION DISTANCIA AUTO 1
515 // CON OTROS AUTOS, PERSONAS Y BICIS
516 for (Auto a : conductores) {
517     a.separateP(peatonos);
518     a.separateP(peatonos2);
519     a.separateP(peatonos3);
520     a.separateB(ciclistas);

```

Guardado finalizado.

Pestaña 1: Programa 13/20

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
521 a.separateB(ciclistas2);
522 a.separateB(ciclistas3);
523 a.separateA(conductores);
524 a.separateA(conductores2);
525 a.separateA(conductores3);
526 a.separateT(choferes);
527 }
528
529 // FUNCIONES AUTO SOLO (2)
530 for (int i = conductores2.size()-1; i
531      Auto a2 = conductores2.get(i);
532     a2.run(comuna2.x, comuna2.y);
533     fill(255, 0, 0);
534     ellipse(comuna1.x, comuna1.y, diam, c
535
536     // SI SALE DE LA VENTANA MUERE
537     if (a2.outOfBounds()) {
538         conductores2.remove(i);
539     }
540 } //(2)
541 // FUNCION DISTANCIA AUTO 2
542 // CON OTROS AUTOS, PERSONAS Y BICIS
543 for (Auto a2 : conductores2) {
544     a2.separateP(peatonos);
545     a2.separateP(peatonos2);
546     a2.separateP(peatonos3);
547     a2.separateB(ciclistas);
548     a2.separateB(ciclistas2);
549     a2.separateB(ciclistas3);
550     a2.separateA(conductores);
551     a2.separateA(conductores2);
552     a2.separateA(conductores3);
553     a2.separateT(choferes);
554 }
555
556 // FUNCIONES AUTO SOLO (3)
557 for (int i = conductores3.size()-1;
558      i >= 0; i--) {
559     Auto a3 = conductores3.get(i);
560     a3.run(comuna3.x, comuna3.y);

```

Guardado finalizado.

Pestaña 1: Programa 14/20

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
561 fill(255, 0, 0);
562 ellipse(comuna3.x, comuna3.y, diam, c
563
564 // SI SALE DE LA VENTANA MUERE
565 if (a3.outOfBounds()) {
566     conductores3.remove(i);
567 }
568 } //(3)
569 // FUNCION DISTANCIA AUTO 3
570 // CON OTROS AUTOS, PERSONAS Y BICIS
571 for (Auto a3 : conductores3) {
572     a3.separateP(peatonos);
573     a3.separateP(peatonos2);
574     a3.separateP(peatonos3);
575     a3.separateB(ciclistas);
576     a3.separateB(ciclistas2);
577     a3.separateB(ciclistas3);
578     a3.separateA(conductores);
579     a3.separateA(conductores2);
580     a3.separateA(conductores3);
581     a3.separateT(choferes);
582 }
583
584 // INSTRUCCIONES
585 // "Mantén presionado la tecla z para
586 // revelar las zonas residenciales y
587 // de tránsito"
588 // "Mantén apretado space para marcar
589 // las rutas recorridas por los agentes"
590
591 // DATOS SIMBOLOGIA VARIABLES
592 image(simbologia, 0, 0);
593 // CANTIDAD DE AGENTES POR TIPO
594 fill(0, 255, 0);
595 text(Peatones*3, 149, 921);
596 fill(0, 100, 255);
597 text(Ciclistas*3, 154, 942);
598 fill(255, 0, 0);
599 text(Conductores*3, 169, 963);
600 fill(255, 255, 0);

```

Guardado finalizado.

Pestaña 1: Programa 15/20

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
601 text(Choferes*3, 149, 986);
602
603 // MAPA ZOOM OUT
604 // CONDICION DEL CLICK
605 if (mousePressed && (mouseY < 880)) {
606   image(mapagrande, 0, 0);
607 } else {
608   image(mapachico, 0, 0);
609 }
610 }
611
612 //////////////////////////////////////
613 ////////////////////////////////////// FIN DRAW //////////////////////////////////////
614 //////////////////////////////////////
615
616 // PATHS PERSONAS
617 void newPathp1() {
618   // PATH PERSONA (1)
619   pathp1 = new Path();
620   pathp1.addPoint(v34.x, v34.y);
621   pathp1.addPoint(v39.x, v39.y);
622   pathp1.addPoint(v35.x, v35.y);
623   pathp1.addPoint(v40.x, v40.y);
624   pathp1.addPoint(v36.x, v36.y);
625   pathp1.addPoint(v41.x, v41.y);
626   pathp1.addPoint(v45.x, v45.y);
627   pathp1.addPoint(v48.x, v48.y);
628   pathp1.addPoint(v49.x, v49.y);
629   pathp1.addPoint(v53.x, v53.y);
630   pathp1.addPoint(cuadra1.x, cuadra1.y);
631 }
632 void newPathp2() {
633   // PATH PERSONA (2)
634   pathp2 = new Path();
635   pathp2.addPoint(v7.x, v7.y);
636   pathp2.addPoint(v12.x, v12.y);
637   pathp2.addPoint(v25.x, v25.y);
638   pathp2.addPoint(v28.x, v28.y);
639   pathp2.addPoint(v36.x, v36.y);
640   pathp2.addPoint(v41.x, v41.y);

```

Guardado finalizado.

Pestaña 1: Programa 16/20

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
641 pathp2.addPoint(v45.x, v45.y);
642 pathp2.addPoint(v48.x, v48.y);
643 pathp2.addPoint(v52.x, v52.y);
644 pathp2.addPoint(cuadra2.x, cuadra2.y);
645 }
646 void newPathp3() {
647   // PATH PERSONA (3)
648   pathp3 = new Path();
649   pathp3.addPoint(v44.x, v44.y);
650   pathp3.addPoint(v39.x, v39.y);
651   pathp3.addPoint(v35.x, v35.y);
652   pathp3.addPoint(v33.x, v33.y);
653   pathp3.addPoint(v28.x, v28.y);
654   pathp3.addPoint(v36.x, v36.y);
655   pathp3.addPoint(v30.x, v30.y);
656   pathp3.addPoint(v26.x, v26.y);
657   pathp3.addPoint(v21.x, v21.y);
658   pathp3.addPoint(v17.x, v17.y);
659   pathp3.addPoint(cuadra3.x, cuadra3.y);
660 }
661
662
663 // PATHS BICI
664 void newPathb1() { // (1)
665   pathb1 = new Path();
666   pathb1.addPoint(v52.x, v52.y);
667   pathb1.addPoint(v41.x, v41.y);
668   pathb1.addPoint(v36.x, v36.y);
669   pathb1.addPoint(v30.x, v30.y);
670   pathb1.addPoint(v26.x, v26.y);
671   pathb1.addPoint(v21.x, v21.y);
672   pathb1.addPoint(v17.x, v17.y);
673   pathb1.addPoint(barrio1.x, barrio1.y);
674 }
675 void newPathb2() { // (2)
676   pathb2 = new Path();
677   pathb2.addPoint(v22.x, v22.y);
678   pathb2.addPoint(v18.x, v18.y);
679   pathb2.addPoint(v15.x, v15.y);
680   pathb2.addPoint(v12.x, v12.y);

```

Guardado finalizado.

Pestaña 1: Programa 17/20

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
681 pathb2.addPoint(v7.x, v7.y);
682 pathb2.addPoint(v20.x, v20.y);
683 pathb2.addPoint(v30.x, v30.y);
684 pathb2.addPoint(v37.x, v37.y);
685 pathb2.addPoint(v42.x, v42.y);
686 pathb2.addPoint(v49.x, v49.y);
687 pathb2.addPoint(v53.x, v53.y);
688 pathb2.addPoint(barrio2.x, barrio2.y);
689 }
690 void newPathb3() { // (3)
691   pathb3 = new Path();
692   pathb3.addPoint(barrio3.x, barrio3.y);
693   pathb3.addPoint(v23.x, v23.y);
694   pathb3.addPoint(v35.x, v35.y);
695   pathb3.addPoint(v33.x, v33.y);
696   pathb3.addPoint(v29.x, v29.y);
697   pathb3.addPoint(v28.x, v28.y);
698   pathb3.addPoint(v25.x, v25.y);
699   pathb3.addPoint(v20.x, v20.y);
700   pathb3.addPoint(v16.x, v16.y);
701   pathb3.addPoint(v14.x, v14.y);
702   pathb3.addPoint(v10.x, v10.y);
703 }
704
705 // AV. LYON
706 // PATHS AUTO
707 void newPatha1() {
708   patha1 = new Path();
709   patha1.addPoint(v1.x, v1.y);
710   patha1.addPoint(v7.x, v7.y);
711   patha1.addPoint(v20.x, v20.y);
712   patha1.addPoint(v30.x, v30.y);
713   patha1.addPoint(v37.x, v37.y);
714   patha1.addPoint(v49.x, v49.y);
715   patha1.addPoint(v53.x, v53.y);
716   patha1.addPoint(comunal1.x, comunal1.y);
717 }
718
719 // AV. PROVIDENCIA
720 void newPatha2() { // PATH AUTO

```

Guardado finalizado.

Pestaña 1: Programa 18/20

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
721 patha2 = new Path();
722 patha2.addPoint(v44.x, v44.y);
723 patha2.addPoint(v39.x, v39.y);
724 patha2.addPoint(v35.x, v35.y);
725 patha2.addPoint(v33.x, v33.y);
726 patha2.addPoint(v29.x, v29.y);
727 patha2.addPoint(v28.x, v28.y);
728 patha2.addPoint(v25.x, v25.y);
729 patha2.addPoint(v20.x, v20.y);
730 patha2.addPoint(v16.x, v16.y);
731 patha2.addPoint(v14.x, v14.y);
732 patha2.addPoint(v10.x, v10.y);
733 patha2.addPoint(comuna2.x, comuna2.y);
734 }
735
736 // AUTO RESIDENTE
737 void newPatha3() { // ZONA RESIDENTE
738 patha3 = new Path();
739 patha3.addPoint(v23.x, v23.y);
740 patha3.addPoint(v35.x, v35.y);
741 patha3.addPoint(v40.x, v40.y);
742 patha3.addPoint(v47.x, v47.y);
743 patha3.addPoint(v45.x, v45.y);
744 patha3.addPoint(v48.x, v48.y);
745 patha3.addPoint(comuna3.x, comuna3.y);
746 }
747
748 // PROVIDENCIA
749 // PATHS TRANSPORTE PUBLICO
750 void newPatht1() {
751 patht1 = new Path();
752 patht1.addPoint(v44.x, v44.y);
753 patht1.addPoint(v39.x, v39.y);
754 patht1.addPoint(v35.x, v35.y);
755 patht1.addPoint(v33.x, v33.y);
756 patht1.addPoint(v29.x, v29.y);
757 patht1.addPoint(v28.x, v28.y);
758 patht1.addPoint(v25.x, v25.y);
759 patht1.addPoint(v20.x, v20.y);
760 patht1.addPoint(v16.x, v16.y);

```

Guardado finalizado.

Pestaña 1: Programa 19/20

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
736 // AUTO RESIDENTE
737 void newPatha3() { // ZONA RESIDENTE
738 patha3 = new Path();
739 patha3.addPoint(v23.x, v23.y);
740 patha3.addPoint(v35.x, v35.y);
741 patha3.addPoint(v40.x, v40.y);
742 patha3.addPoint(v47.x, v47.y);
743 patha3.addPoint(v45.x, v45.y);
744 patha3.addPoint(v48.x, v48.y);
745 patha3.addPoint(comuna3.x, comuna3.y);
746 }
747
748 // PROVIDENCIA
749 // PATHS TRANSPORTE PUBLICO
750 void newPatht1() {
751 patht1 = new Path();
752 patht1.addPoint(v44.x, v44.y);
753 patht1.addPoint(v39.x, v39.y);
754 patht1.addPoint(v35.x, v35.y);
755 patht1.addPoint(v33.x, v33.y);
756 patht1.addPoint(v29.x, v29.y);
757 patht1.addPoint(v28.x, v28.y);
758 patht1.addPoint(v25.x, v25.y);
759 patht1.addPoint(v20.x, v20.y);
760 patht1.addPoint(v16.x, v16.y);
761 patht1.addPoint(v14.x, v14.y);
762 patht1.addPoint(v10.x, v10.y);
763 patht1.addPoint(comuna2.x, comuna2.y);
764 }
765
766 // FUNCION EN DESUSO
767 public void keyPressed() {
768 if (key == ' ') {
769 debug = !debug;
770 }
771 }
772
773
774
775

```

Guardado finalizado.

Pestaña 1: Programa 20/20

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
1 class Auto extends Bici {
2
3 Auto(float x_, float y_) {
4 super(x_, y_);
5 velocity =
6 new PVector(random(0, 0),
7 random(0, 0));
8 diam = 3;
9 maxspeedP = 4;
10 maxforce = 0.15; // 0.05
11 }
12
13 // APARIENCIA
14 void display() {
15 float theta =
16 velocity.heading() + PI/2;
17 fill(255, 0, 0);
18 noStroke();
19 pushMatrix();
20 translate(location.x, location.y);
21 rotate(theta);
22 rect(0, 0, diam, diam*3);
23 popMatrix();
24 }
25
26 // DESPLAZAMIENTO, VELOCIDAD
27 // Y ACELERACION
28 void update() {
29 velocity.add(acceleration);
30 velocity.limit(maxspeedA);
31 location.add(velocity);
32 acceleration.mult(0);
33 }
34
35 // FUNCION DE FUNCIONES
36 void run(float x_, float y_) {
37 update();
38 display();
39 seek(new PVector(x_, y_));
40 //choqueMurallas();

```

Pestaña 2: Auto 1/4

```
CODIGO_6_PBAT_zBuses | Processing 3.3.5
//reboteEdificio();
}

// INTERACCION CON PERSONA
boolean choca(Persona p) {
    float d =
        abs(dist(location.x, location.y,
            p.location.x, p.location.y));
    if (d <= 10) {
        return true;
    } else {
        return false;
    }
}

// INTERACCION CON BICI
boolean choca(Bici b) {
    float d =
        abs(dist(location.x, location.y,
            b.location.x, b.location.y));
    if (d <= 10) {
        return true;
    } else {
        return false;
    }
}

// INTERACCION CON TRANSPORTE PUBLICO
boolean choca(TransportePublico t) {
    float d =
        abs(dist(location.x, location.y,
            t.location.x, t.location.y));
    if (d <= 10) {
        return true;
    } else {
        return false;
    }
}

// SEPARACION CON AUTOS
```

Pestaña 2: Auto 2/4

```
CODIGO_6_PBAT_zBuses | Processing 3.3.5
void separateA (ArrayList<Auto>
    conductores) {
    //Note how the desired separation is
    float desiredseparation = diam*20;
    PVector sum = new PVector();
    int count = 0;
    for (Auto other : conductores) {
        float d =
            PVector.dist(location,
                other.location);
        if ((d > 0) &&
            (d < desiredseparation)) {
            PVector diff =
                PVector.sub(location,
                    other.location);
            diff.normalize();
            //What is the magnitude of the P
            diff.div(d);
            sum.add(diff);
            count++;
        }
        if (count > 0) {
            sum.div(count);
            sum.normalize();
            sum.mult(maxspeedA);
            PVector steer =
                PVector.sub(sum, velocity);
            steer.limit(maxforce);
            applyForce(steer);
        }
    }

    void separateT (ArrayList<TransporteP
        choferes) {
    //Note how the desired separation is
    float desiredseparation = diam*10;
    PVector sum = new PVector();
    int count = 0;
    for (TransportePublico other :
```

Pestaña 2: Auto 3/4

```
CODIGO_6_PBAT_zBuses | Processing 3.3.5
}
}

void separateT (ArrayList<TransporteP
    choferes) {
    //Note how the desired separation is
    float desiredseparation = diam*10;
    PVector sum = new PVector();
    int count = 0;
    for (TransportePublico other :
        choferes) {
        float d = PVector.dist(location,
            other.location);
        if ((d > 0) &&
            (d < desiredseparation)) {
            PVector diff =
                PVector.sub(location,
                    other.location);
            diff.normalize();
            diff.div(d);
            sum.add(diff);
            count++;
        }
    }
    if (count > 0) {
        sum.div(count);
        sum.normalize();
        sum.mult(maxspeedT);
        PVector steer =
            PVector.sub(sum, velocity);
        steer.limit(maxforce);
        applyForce(steer);
    }
}
}
```

Pestaña 2: Auto 4/4

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
1 class Bici extends Persona {
2
3   Bici(float x_, float y_) {
4     super(x_, y_);
5     velocity =
6       new PVector(random(0, 0),
7         random(0, 0));
8     diam = 2.5;
9     maxforce = 0.15;
10  }
11
12  // APARIENCIA
13  void display() {
14    //Vehicle is a triangle pointing in
15    float theta =
16      velocity.heading() + PI/2;
17    fill(0, 100, 255);
18    noStroke();
19    pushMatrix();
20    translate(location.x, location.y);
21    rotate(theta);
22    beginShape();
23    vertex(0, -diam*2);
24    vertex(-diam, diam);
25    vertex(diam, diam);
26    endShape(CLOSE);
27    popMatrix();
28  }
29
30  // DESPLAZAMIENTO,
31  // VELOCIDAD Y ACELERACION
32  void update() {
33    velocity.add(acceleration);
34    velocity.limit(maxspeedB);
35    location.add(velocity);
36    acceleration.mult(0);
37  }
38
39  // FUNCION DE FUNCIONES
40  void run(float x_, float y_) {

```

Pestaña 3: Bici 1/4

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
41  update();
42  display();
43  seek(new PVector(x_, y_));
44  //choqueMurallas();
45  //reboteEdificio();
46  }
47
48  // INTERACCION CON PERSONA
49  boolean choca(Persona p) {
50    float d =
51      abs(dist(location.x, location.y,
52        p.location.x, p.location.y));
53    if (d <= 10) {
54      return true;
55    } else {
56      return false;
57    }
58  }
59  // INTERACCION CON TRANSPORTE PUBLICO
60  boolean choca(TransportePublico t) {
61    float d =
62      abs(dist(location.x, location.y,
63        t.location.x, t.location.y));
64    if (d <= 10) {
65      return true;
66    } else {
67      return false;
68    }
69  }
70
71  // FUNCION SEPARACION BICI A BICI
72  void separateB (ArrayList<Bici>
73    ciclistas) {
74    //Note how the desired separation is
75    float desiredseparation = diam*20;
76    PVector sum = new PVector();
77    int count = 0;
78    for (Bici other : ciclistas) {
79      float d =
80        PVector.dist(location,

```

Pestaña 3: Bici 2/4

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
81    other.location);
82    if ((d > 0) &&
83      (d < desiredseparation)) {
84      PVector diff =
85        PVector.sub(location, other.
86        diff.normalize();
87        //What is the magnitude of the P
88        diff.div(d);
89        sum.add(diff);
90        count++;
91      }
92    }
93    if (count > 0) {
94      sum.div(count);
95      sum.normalize();
96      sum.mult(maxspeedB);
97      PVector steer =
98        PVector.sub(sum, velocity);
99      steer.limit(maxforce);
100     applyForce(steer);
101   }
102 }
103
104 void separateT (ArrayList<TransporteP
105   choferes) {
106   //Note how the desired separation is
107   float desiredseparation = diam*10;
108   PVector sum = new PVector();
109   int count = 0;
110   for (TransportePublico other :
111     choferes) {
112     float d =
113       PVector.dist(location,
114         other.location);
115     if ((d > 0) &&
116       (d < desiredseparation)) {
117       PVector diff =
118         PVector.sub(location,
119         other.location);
119       diff.normalize();
120

```

Pestaña 3: Bici 3/4

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
101 }
102 }
103
104 void separateT (ArrayList<TransporteI
105                 choferes) {
106     //Note how the desired separation is
107     float desiredseparation = diam*10;
108     PVector sum = new PVector();
109     int count = 0;
110     for (TransportePublico other :
111         choferes) {
112         float d =
113             PVector.dist(location,
114                 other.location);
115         if ((d > 0) &&
116             (d < desiredseparation)) {
117             PVector diff =
118                 PVector.sub(location,
119                     other.location);
120             diff.normalize();
121             diff.div(d);
122             sum.add(diff);
123             count++;
124         }
125     }
126     if (count > 0) {
127         sum.div(count);
128         sum.normalize();
129         sum.mult(maxspeedT);
130         PVector steer =
131             PVector.sub(sum, velocity);
132         steer.limit(maxforce);
133         applyForce(steer);
134     }
135 }
136 }
137
138
139
140

```

Pestaña 3: Bici 4/4

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
1 // The Nature of Code
2 // Daniel Shiffman
3 // http://natureofcode.com
4
5 // Path Following
6
7 class Path {
8
9     // A Path is an arraylist of points (P
10    ArrayList<PVector> points;
11    // A path has a radius, i.e how far is
12    float radius;
13
14    Path() {
15        // Arbitrary radius of 12 meters
16        radius = 12;
17        points = new ArrayList<PVector>();
18    }
19
20    // Add a point to the path
21    void addPoint(float x, float y) {
22        PVector point = new PVector(x, y);
23        points.add(point);
24    }
25
26    PVector getStart() {
27        return points.get(0);
28    }
29
30    PVector getEnd() {
31        return points.get(points.size()-1);
32    }
33
34
35    // Draw the path
36    void display(
37        color r_,
38        color g_,
39        color b_) {
40        // Draw thick line for radius

```

Pestaña 4: Path 1/2

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
23     points.add(point);
24 }
25
26 PVector getStart() {
27     return points.get(0);
28 }
29
30 PVector getEnd() {
31     return points.get(points.size()-1);
32 }
33
34
35 // Draw the path
36 void display(
37     color r_,
38     color g_,
39     color b_) {
40     // Draw thick line for radius
41     color r = r_;
42     color g = g_;
43     color b = b_;
44     stroke(r,g,b, 40);
45     strokeWeight(radius*2);
46     noFill();
47     beginShape();
48     for (PVector v : points) {
49         vertex(v.x, v.y);
50     }
51     endShape();
52     // Draw thin line for center of path
53     stroke(0, 50);
54     strokeWeight(1);
55     noFill();
56     beginShape();
57     for (PVector v : points) {
58         vertex(v.x, v.y);
59     }
60     endShape();
61 }
62 }

```

Pestaña 4: Path 2/2


```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
1 class Persona {
2
3   PVector location;
4   PVector velocity;
5   PVector acceleration;
6   float diam;
7   float maxforce;
8
9   //float mass;
10
11  // POSICION DE ORIGEN, VELOCIDAD
12  // Y LIMITE DE VELOCIDAD
13  Persona(float x_, float y_) {
14    acceleration = new PVector(0, 0);
15    velocity = new PVector(0,0);
16    location = new PVector(x_, y_);
17    diam = 3;
18    maxforce = 0.15; // 0.01
19    maxspeedP = 1.5;
20    //mass = 100;
21  }
22
23  // DESPLAZAMIENTO,
24  // VELOCIDAD Y ACELERACION
25  void update() {
26    velocity.add(acceleration);
27    velocity.limit(maxspeedP/5);
28    location.add(velocity);
29    acceleration.mult(0);
30  }
31
32  // APLICADOR DE FUERZA //Newton's seco
33  void applyForce(PVector force) {
34    acceleration.add(force);
35  }
36
37  // FUNCION DE SEGUIMIENTO DE ATRACTOR
38  void seek(PVector target) {
39    PVector desired =
40      PVector.sub(target, location);

```

Pestaña 5: Persona 1/6

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
41   desired.normalize();
42   desired.mult(maxspeedP);
43   PVector steer =
44     PVector.sub(desired, velocity);
45   steer.limit(maxforce);
46   applyForce(steer);
47 }
48
49 // FUNCION DE SEGUIMIENTO DE CALLES
50 void seekPath(PVector target) {
51   PVector desired =
52     PVector.sub(target, location);
53   desired.normalize();
54   desired.mult(maxspeedP);
55   PVector steer =
56     PVector.sub(desired, velocity);
57   steer.limit(maxforce);
58   steer.x = steer.x*5;
59   steer.y = steer.y*5;
60   applyForce(steer);
61 }
62
63 // FUNCION DE FUNCIONES
64 void run(float x_, float y_) {
65   update();
66   display();
67   seek(new PVector(x_, y_));
68   //choqueMurallas();
69   //reboteEdificio();
70 }
71
72 // APARIENCIA
73 void display() {
74   //Vehicle is a triangle pointing in
75   float theta =
76     velocity.heading() + PI/2;
77   fill(0, 255, 0);
78   stroke(0, 255, 0);
79   strokeWeight(1);
80   pushMatrix();

```

Pestaña 5: Persona 2/6

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
81   translate(location.x, location.y);
82   rotate(theta);
83   ellipse(0, 0, diam, diam/2);
84   popMatrix();
85 }
86
87 // SI SALE DE LA VENTANA MUERE
88 boolean outOfBounds() {
89   if ((location.x > width + 15)
90     || (location.x < 0 - 15)
91     || (location.y > height + 15)
92     || (location.y < 0 - 15)) {
93     return true;
94   } else {
95     return false;
96   }
97 }
98
99 // FUNCION DE SEPARACION ENTRE PERSONAS
100 void separateP (ArrayList<Persona>
101   peatones) {
102   //Note how the desired separation is
103   float desiredseparation = diam*1;
104   PVector sum = new PVector();
105   int count = 0;
106   for (Persona other : peatones) {
107     float d =
108       PVector.dist(location, other.l
109     if ((d > 0) &&
110       (d < desiredseparation)) {
111       PVector diff =
112         PVector.sub(location,
113           other.location);
114       diff.normalize();
115       //What is the magnitude of the P
116       diff.div(d);
117       sum.add(diff);
118       count++;
119     }
120 }

```

Pestaña 5: Persona 3/6

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
Java
CODIGO_6_PBAT_zBuses
121   if (count > 0) {
122       sum.div(count);
123       sum.normalize();
124       sum.mult(maxspeedP);
125       PVector steer =
126           PVector.sub(sum, velocity);
127       steer.limit(maxforce);
128       applyForce(steer);
129   }
130 }
131
132 void follow(Path p) {
133
134     PVector predict = velocity.get();
135     predict.normalize();
136     predict.mult(50);
137     PVector predictpos =
138         PVector.add(location, predict);
139     PVector normal = null;
140     PVector target = null;
141     float worldRecord = 1000000; // Sta
142
143     for (int i = 0;
144         i < p.points.size()-1; i++) {
145
146         PVector a = p.points.get(i);
147         PVector b = p.points.get(i+1);
148
149         PVector normalPoint =
150             getNormalPoint(predictpos, a, b);
151         if (normalPoint.x < a.x ||
152             normalPoint.x > b.x) {
153             normalPoint = b.get();
154         }
155         float distance =
156             PVector.dist(predictpos,
157                 normalPoint); // How
158         if (distance < worldRecord) {
159             worldRecord = distance;
160             normal = normalPoint;

```

Pestaña 5: Persona 4/6

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
Java
CODIGO_6_PBAT_zBuses
121   if (count > 0) {
122       sum.div(count);
123       sum.normalize();
124       sum.mult(maxspeedP);
125       PVector steer =
126           PVector.sub(sum, velocity);
127       steer.limit(maxforce);
128       applyForce(steer);
129   }
130 }
131
132 void follow(Path p) {
133
134     PVector predict = velocity.get();
135     predict.normalize();
136     predict.mult(50);
137     PVector predictpos =
138         PVector.add(location, predict);
139     PVector normal = null;
140     PVector target = null;
141     float worldRecord = 1000000; // Sta
142
143     for (int i = 0;
144         i < p.points.size()-1; i++) {
145
146         PVector a = p.points.get(i);
147         PVector b = p.points.get(i+1);
148
149         PVector normalPoint =
150             getNormalPoint(predictpos, a, b);
151         if (normalPoint.x < a.x ||
152             normalPoint.x > b.x) {
153             normalPoint = b.get();
154         }
155         float distance =
156             PVector.dist(predictpos,
157                 normalPoint); // How
158         if (distance < worldRecord) {
159             worldRecord = distance;
160             normal = normalPoint;

```

Pestaña 5: Persona 5/6

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
Java
CODIGO_6_PBAT_zBuses
152     normalPoint.x > b.x) {
153         normalPoint = b.get();
154     }
155     float distance =
156         PVector.dist(predictpos,
157             normalPoint); // How
158     if (distance < worldRecord) {
159         worldRecord = distance;
160         normal = normalPoint;
161
162         PVector dir = PVector.sub(b, a);
163         dir.normalize();
164         dir.mult(10);
165         target = normalPoint.get();
166         target.add(dir);
167     }
168 }
169 if (worldRecord > p.radius) {
170     seekPath(target);
171 }
172 }
173 // A function to get the normal point
174 // This function could be optimized to
175 PVector getNormalPoint(PVector p, PVec
176 // Vector from a to p
177 PVector ap = PVector.sub(p, a);
178 // Vector from a to b
179 PVector ab = PVector.sub(b, a);
180 ab.normalize(); // Normalize the line
181 // Project vector "diff" onto line b
182 ab.mult(ap.dot(ab));
183 PVector normalPoint =
184     PVector.add(a, ab);
185 return normalPoint;
186 }
187 }
188
189
190
191

```

Pestaña 5: Persona 6/6

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
1 class TransportePublico extends Persona
2
3 TransportePublico(float x_, float y_)
4     super(x_, y_);
5     velocity =
6     new PVector(random(0,0), random(0,0));
7     diam = 5;
8     maxforce = 0.15;
9
10 }
11
12 // APARIENCIA
13 void display() {
14     //Vehicle is a triangle pointing in
15     float theta =
16     velocity.heading() + PI/2;
17     fill(255, 200, 55);
18     noStroke();
19     pushMatrix();
20     translate(location.x, location.y);
21     rotate(theta);
22     rect(0,0,diam,diam*3,2);
23     popMatrix();
24 }
25
26 // DESPLAZAMIENTO,
27 // VELOCIDAD Y ACELERACION
28 void update() {
29     velocity.add(acceleration);
30     velocity.limit(maxspeedT);
31     location.add(velocity);
32     acceleration.mult(0);
33 }
34
35 // FUNCION DE FUNCIONES
36 void run(float x_, float y_) {
37     update();
38     display();
39     seek(new PVector(x_, y_));
40 }

```

Pestaña 6: T.Público 1/2

```

CODIGO_6_PBAT_zBuses | Processing 3.3.5
CODIGO_6_PBAT_zBuses
42 // INTERACCION CON PERSONA
43 boolean choca(Persona p) {
44     float d =
45     abs(dist(location.x, location.y,
46     p.location.x, p.location.y));
47     if (d <= 10) {
48         return true;
49     } else {
50         return false;
51     }
52 }
53 void separateT (ArrayList<TransportePul
54     choferes) {
55     //Note how the desired separation is
56     float desiredseparation = diam*3;
57     PVector sum = new PVector();
58     int count = 0;
59     for (TransportePublico other :
60     choferes) {
61         float d =
62         PVector.dist(location,
63         other.location);
64         if ((d > 0) &&
65         (d < desiredseparation)) {
66             PVector diff =
67             PVector.sub(location,
68             other.location);
69             diff.normalize();
70             diff.div(d);
71             sum.add(diff);
72             count++;
73         } }
74     if (count > 0) {
75         sum.div(count);
76         sum.normalize();
77         sum.mult(maxspeedT);
78         PVector steer =
79         PVector.sub(sum, velocity);
80         steer.limit(maxforce);
81         applyForce(steer);

```

Pestaña 6: T.Público 2/2

VERSIÓN 06

material
complementario



MAPA

Se reutilizaron ambos mapas de la versión 5, el de fondo gris en reposo y el con colores para marcar las zonas de tránsito y las zonas residenciales.

Lo que si cambió fue la forma de interactuar con estas opciones; anteriormente al hacer click se cambiaba el mapa de fondo, el problema estaba en que al presionar el cursor incluso sobre los sliders activaba esta función. Es por eso que se activó la tecla “z” (y su versión en mayúsculas “Z”) para activar las zonas, para así liberar la función del click del cursor, la cual sería utilizada en los sliders.

Se agregó en el programa una zona interactiva (f.1) que redistribuye la sibología, incluyendo a los “choferes”, también tiene un espacio marcado para sliders que modifican los tamaños de lo grupos, y se agregó la opción de modificar las velocidades máximas.

En el extremo derecho del banner se encuentran las instrucciones de las funciones para marcar las zonas de distintas jerarquías. Y también están las indicaciones para ampliar el mapa general de la zona de providencia.



<i>CANTIDAD de agentes por tipo</i>	<i>TAMAÑO de los grupos</i>	<i>VELOCIDADES máximas</i>	<i>ZONAS con restricciones</i>	<i>MAPA ORIGEN Y DESTINO</i>
<ul style="list-style-type: none"> ● Peatones total: ▲ Ciclistas total: ■ Conductores total: ● Choferes total: 	<input type="text"/> <input type="text"/> <input type="text"/> <input type="text"/>	<input type="text"/> <input type="text"/> <input type="text"/>	<p>Mantén la tecla "z" para ver las zonas residenciales y las zonas de tránsito marcadas con color.</p> <p style="text-align: center;">z</p>	<p>Mantén presionado el cursor para ampliar el mapa de los orígenes y destinos de los distintos agentes.</p>

f.1

VERSIÓN 06

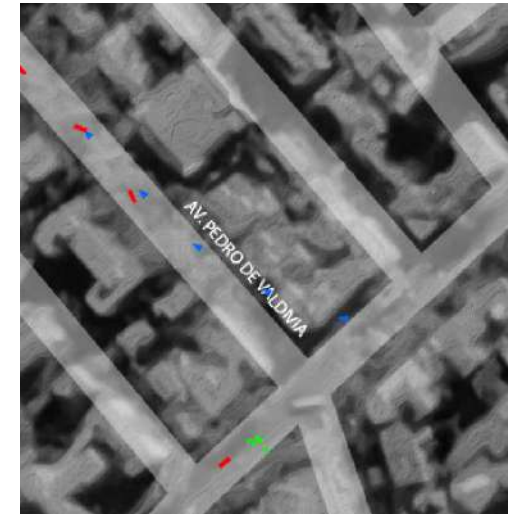
funcionamiento,
observaciones y
conclusiones



1. APARIENCIA

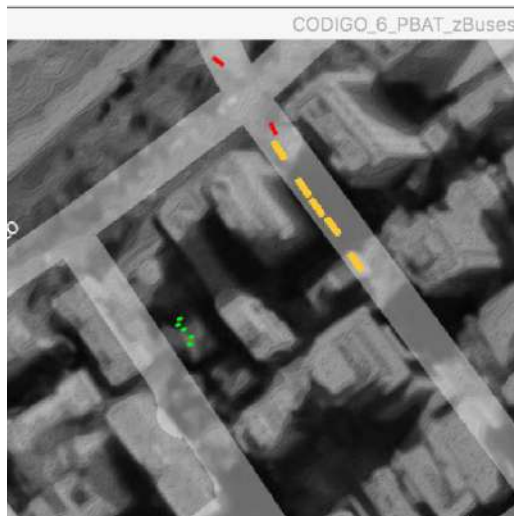
El mayor cambio sería la redistribución e inclusión de nuevos sliders, (PI, 127-148), la inclusión del transporte público (Pestaña 6), la inclusión del mapa interactivo (PI, 182-190) y la mayor fluidez del programa general; dada por una optimización de los “loops” internos escritos los cuales están encargados de comprobar si hay interacción o no entre todos los usuarios.

El comportamiento emergente y la jerarquización en base a la sustentabilidad se mantienen en esta versión incluyendo al nuevo tipo de agente: el bus del transporte público.



2.1 ENCUENTROS

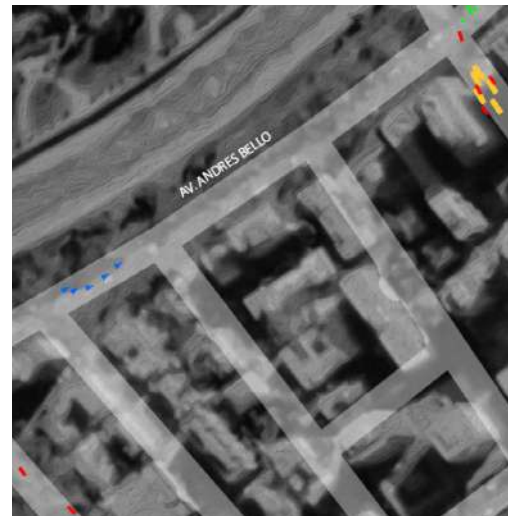
En la versión quinta del programa habían unas líneas que hacían correr muchas iteraciones para probar que cada uno de los agentes estaba o no en contacto con otro agente; cada persona por separado de cada grupo se comprobaba que no estuviera en contacto con, personas, ciclistas y autos. Era un bucle de programación de 9 niveles que corría para cada uno de los móviles en pantalla, un proceso muy costoso a nivel de procesador. Para simplificar esto se decidió controlar lo grupos por tipo de manera simultánea, es decir, hacer crecer la cantidad de las personas, es hacer crecer todos lo grupos, al final tratándolos como uno solo.



2.2 ENCUENTROS

Al tratar cada grupo como un conjunto para el programa recorrer 4 veces el conjunto de agentes es exponencialmente menos costoso que las 9 dimensiones que trataba antes.

Los caminos por los que se encuentran los agentes están escritos de forma manual entre las líneas 615 y 765 de la primera pestaña. Usan como referencia los puntos vectoriales (coordenadas x,y) del mapa hecho a mano de la versión 5 del simulador. Lo ideal, como ya se mencionó, sería no verse obligado a introducir las coordenadas de forma manual, si no que importarlas desde un software automático.



3. ORÍGENES

La pequeña gran diferencia comparado con el programa previo es la inclusión de los buses, o mejor llamarlo, transporte público, para no encasillar su forma futura.

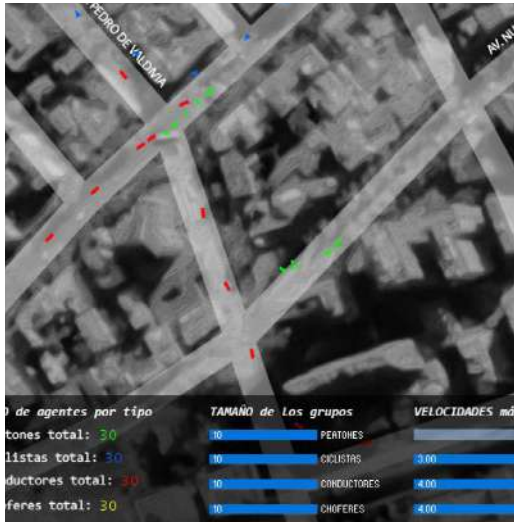
En la *pestaña 6* se puede apreciar que la clase del transporte público es una extensión de la persona. Recordemos que eso significa heredar sus características y funcionalidades predeterminadas, mas no pierde las capacidades de personalización. Esta nueva clase va incluida en el set de encuentro, no olvidemos que es el transporte público el otro “rey” de las jerarquías.



4. PATHS OCULTOS

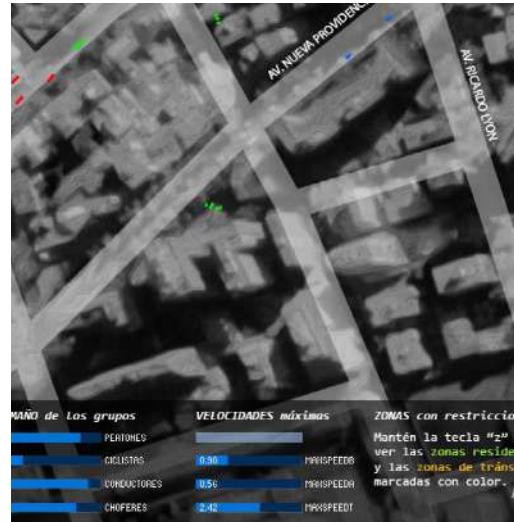
Lo mejor y última versión del programa le permitiría a las personas (óptimamente) elegir por dónde y cómo ir según una experiencia previa; “si me demoré ‘x’ por este camino con este ‘y’ modo de transporte: entonces me voy de esta manera ‘z’”. Este sistema generacional le permitiría al programa general su propio porcentaje y equilibrio de cantidades de tipos de agentes.

Aquel sistema se complementaría bien con el actual en el que los modos de transporte son usados para evaluar niveles de impacto y las interacciones y finalmente la tasa de sustentabilidad. (PI, 193-203)



5.1 SLIDERS TAMAÑO

Se modificaron los controladores para poder interactuar con las cantidad de agentes por tipo, agrandando simultáneamente los 3 grupos en vez de cada uno por separado. Eso quiere decir que si en slider de peatones uno aumenta el valor a 10, la cantidad total de será 30, como se muestra en el caso de arriba.



5.2 SLIDERS VELOCIDAD

Para un mayor control de las interacciones entre agentes, y proyectando la funcionalidad tanto como para el usuario como para los valores indicadores del impacto del ecosistema de tránsito se agregaron sliders para controlar las velocidades máximas. Cada uno de los controladores modifica la velocidad mayor a la que puede llegar cada tipo de agentes, excluyendo a los peatones, por su forma libre de moverse. (PI, 137-149)

Con la cantidad de agentes y la de las velocidades en control se puede crear un indicador sobre la seguridad o el nivel de interacción de los peatones.



6. ZONAS

Como fue ya mencionado, se modificó la interacción con el programa para ver las zonas con colores. Cambió de hacer click, lo cual generaba problemas a la hora de interactuar con los sliders, y pasó a ser la tecla Z la que activa la función del cambio de fondo. (PI, 181-189)

Al apretar botones y el cursor por separado funcionaba sin problemas, el problema estaba que al apretar combinaciones aparecían los trayectos de por donde pasaban los agentes. Quizás volver a como era en la versión anterior, en donde todo se controlaba con el mouse, se evitaría la sobrecarga de inputs y errores que estos producen.



7. MAPA

Al hacer click sobre el programa y más arriba del banner se amplía el mapa que revela la ubicación del extracto a la que pertenece el mapa. (PI, 603-610)

Este espacio podría usarse también para definir la ubicación aproximada de donde quedan los atractores / destinos de cada uno de los agentes.

La interacción del click de para ampliar solo ocurre cuando el mouse está sobre el mapa. En una versión intermedia funcionaba en cualquier lugar del simulador, incluso cuando se hacía click para modificar los sliders. Un desagrado y gran distracción en flujo de uso.

```

16
17
18 void work() {
19   this.display();
20   this.takeInput();
21 }
22
23 void display() {
24   fill(255);
25   strokeWeight(2);
26   stroke(0);
27   rect(x, y, abs(x2 - x), abs(y2 - y));
28   textSize(25);
29   fill(0);
30   if (drawLoop % 40 > 20 && isThisSelected) { //
31     text(text + '|', x, y, abs(x2 - x), abs(y2 - y));
32   } else { //other half the time
33     println(text, x, y, abs(x2 - x), abs(y2 - y));
34     text(text, x, y, abs(x2 - x), abs(y2 - y));
35   }
36 }
37
38 void takeInput() {
39   if (mousePressed && x < mouseX && mouseY < y2 &&
40     isThisSelected = true;
41   else if (mousePressed)
42     isThisSelected = false;

```

8. ERRORES

En algunos casos particulares se produce error cuando se reduce la cantidad de agentes el programa intenta corroborar la posición de dos agentes, para ver si hay o no un encuentro, por que estos desaparecen justo. Entonces se da una función vacía. Básicamente es preguntar donde está un agente que ya no existe.

Otro error que se produce al reducir la cantidad de agentes del programa es que el radio de aparición alrededor del punto de origen pareciera crecer, y esto resulta en la aparición y desaparición rápida de agentes, un parpadeo constante.

ENTREVISTA Y VALIDACIÓN CON ALESSIO BELLINO

doctor en ciencias de
la computación



Alessio Bellino tiene un doctorado en ciencias de la computación (Università di Milano-Bicocca) y tiene diferentes publicaciones internacionales. Trabaja en el ámbito de la interacción hombre-maquina, diseño de la interacción y computación ubicua. www.fundaciondid.cl Transcripción de la entrevista adjunta al final del libro.

RESUMEN

Partiendo por lo más importante; Alessio cree que *“tu modelo es más un proyecto de comunicación”*, que una simulación, dado que no representa una realidad, si no más bien un imaginario futuro, una idea. Esto valida la idea el simulador debe ser una herramienta de comunicación y propuesta, y no solo tener la capacidad de emular tal realidad.

Él concuerda que el proceso de cambio social y de manera de actuar, para hacer una realidad este ecosistema vial, sería un proceso lento y de generaciones; más aun si la situación chilena es la contraria: los choferes cruzan sin respeto por sobre los vulnerables peatones.

Respecto a tecnicismos del programa Alessio Bellino cree que, como se aplica a mucho diseño, se deben simplificar muchas cosas, remover lo innecesario, hacer el programa más autoexplicativo; convertir el programa en una experiencia visual. Introducir brevemente las jerarquías y dejar que el programa se despliegue por si solo y que se entienda con solo mirarlo.

VERSIÓN 06

proyección y reflexión

CONCLUSIÓN GENERAL

A pesar de verse más robustecido al programa le faltan algunos ajustes para poder responder con los objetivos iniciales del proyecto.

Los casos de encuentros, lo orígenes, sliders y lo destinos ya están resultados, pero la separación de las zonas de tránsito y las residenciales podría estar mejor representado y explicado. Por ahora la versión carece de una explicación de como funcionan las jerarquías y el por qué. Para complementar estos componentes del simulador se podrían desarrollar ventanas popups que contengan información complementaria que vaya relacionada con la propuesta de sistema emergente, la jerarquía y la tendencia en pos de la sustentabilidad y las tecnologías autónomas.

El gráfico con indicadores será una pieza fundamental, dado su dinamismo en tiempo real de como afecta la ciudad las distintas cantidades de agentes, las proporciones y las relaciones entre ellos. Estos valores son útiles para comparar el impacto con o sin factores sustentables.

PRÓXIMOS PASOS

1. Hacer del programa una experiencia más auto-explicativo. Ensachar calles, explicar jerarquías y quizás eliminar la leyenda y agregar los popups que explican en profundidad algunas funciones u orígenes de los sliders o indicadores.

2. Retomando los objetivos generales y específicos de la etapa de investigación; definir cuales son los indicadores que a partir de cruces de datos concretos permiten demostrar el impacto de la propuesta ecosistémica de transporte. Para luego plasmar dichos datos en un gráfico dinámico dentro del programa. Tales datos variables le indicarán al usuario la tasa de impacto de sustentabilidad final, un valor útil a la hora de tomar decisiones de planificación.

3. Separar zonas de tránsito y las zonas de residencia y lo agentes correspondientes de una forma gráfica intuitiva y autoexplicativa. Se seguirá la idea sugerida por Alessio de ensachar las calles de tránsito y reducir las residenciales. Con el fin de obviar la tipología de las calles en pos de la interacción fluida del usuario.

INDICADORES

a partir de cruce de datos

Con diversos artículos se determinó la naturaleza y origen de indicadores que nacen a partir del cruce de datos que están presente en el programa y en diversos estudios. Estos indicadores permiten comprender el impacto que podría llegar a tener el nuevo sistema de transporte propuesto, así como también, sin importar el grado de innovación que este puede tener. Estas fórmulas, o bien relaciones de datos, fueron aprobadas teóricamente por Oscar Figueroa (p.117) y Tomás Vivanco. Y desde el criterio matemático fueron modificadas con el futuro Ingeniero Civil Matemático Santiago Armstrong.

% SEGURIDAD VIAL

Es la relación que hay entre la distancia D a la que los agentes reaccionan entre sí y la velocidades máximas a las que puede llegar cada uno. La World Health Organization sostiene que estudios han demostrado que los peatones tienen un 90% de chance de sobrevivencia al ser chocados por un auto viajando a una velocidad de 30kph o menos, pero si el auto va a 45kph el porcentaje de sobrevivencia se reduce a solo 50%. "Studies suggest that a 1 km/h decrease in travelling speed would lead to a 2-3% reduction in road crashes". (WHO, 2004)

% INTERACCIÓN PEATONAL

Es la razón entre las posibles interacciones de todas las personas Q , multiplicado por la distancia D y las velocidades y cantidades de las bicicletas y autos Q . "Respecto de los contactos sociales, los estratos socioeconómicos de menores ingresos siguen vinculados a sus redes a partir del espacio físico, por cercanía y por el contacto diario en la calle. De este modo es posible encontrar en estos sectores un vínculo afectivo con el entorno mucho más arraigado que en los sectores de altos ingresos, lo que además es reforzado por el hecho de que parte de la red personal afectiva y familiar también vive en las cercanías físicas. Por estas mismas razones, el espacio de actividades por donde se mueven e interactúan es más concentrado, usualmente con una base muy marcada al interior de los propios sectores." (Garcia, Carrasco & Rojas, 2014)

% NIVEL DE SILENCIO

Corresponde a la suma lineal de los decibeles R variables de todos los buses y autos Q . Las bicicletas y las personas son omitidas dado que sus 25dB y 20dB son casi imperceptibles. para el ser humano, y no molestos para las demás

creaturas. 20dB es lo más despacio que puede escuchar un humano. Datos extraídos de AmegoEV, 2017: Bus: 100 dB, Auto: 75d, Bici: 25dB y Peatones: 20dB.

% ESPACIO LIBERADO

A la superficie total de calles se le resta la suma de todo el espacio S usado por cada uno de los agentes Q. (Allen, 2008)
 Bus: 0,4 mts², Auto: 13,8 mts², Bici: 1,25mts², Persona: 0,5mts²

% EMISIONES DE CO2

Corresponde a la suma de las emisiones P variables de kilogramos de CO2 de todos los buses y autos Q. Bus: 1900kg CO2, Auto: 410kgs, Bici: 0kg, Peatones: 0kg (AmegoEV, 2017)

FÓRMULAS Y ESQUEMAS

SEGURIDAD

$$\frac{D}{Vc \times Vb \times Va}$$

INTERACCIÓN

$$\frac{(Qp - 1) \times Qp \times D}{Vc \times Qc \times Va \times Qa}$$

SILENCIO

$$Qb \times Rb + Qa \times Ra$$

ESPACIO

$$Qp \times Sp + Qc \times Sc + Qb \times Sb + Qa \times Sa$$

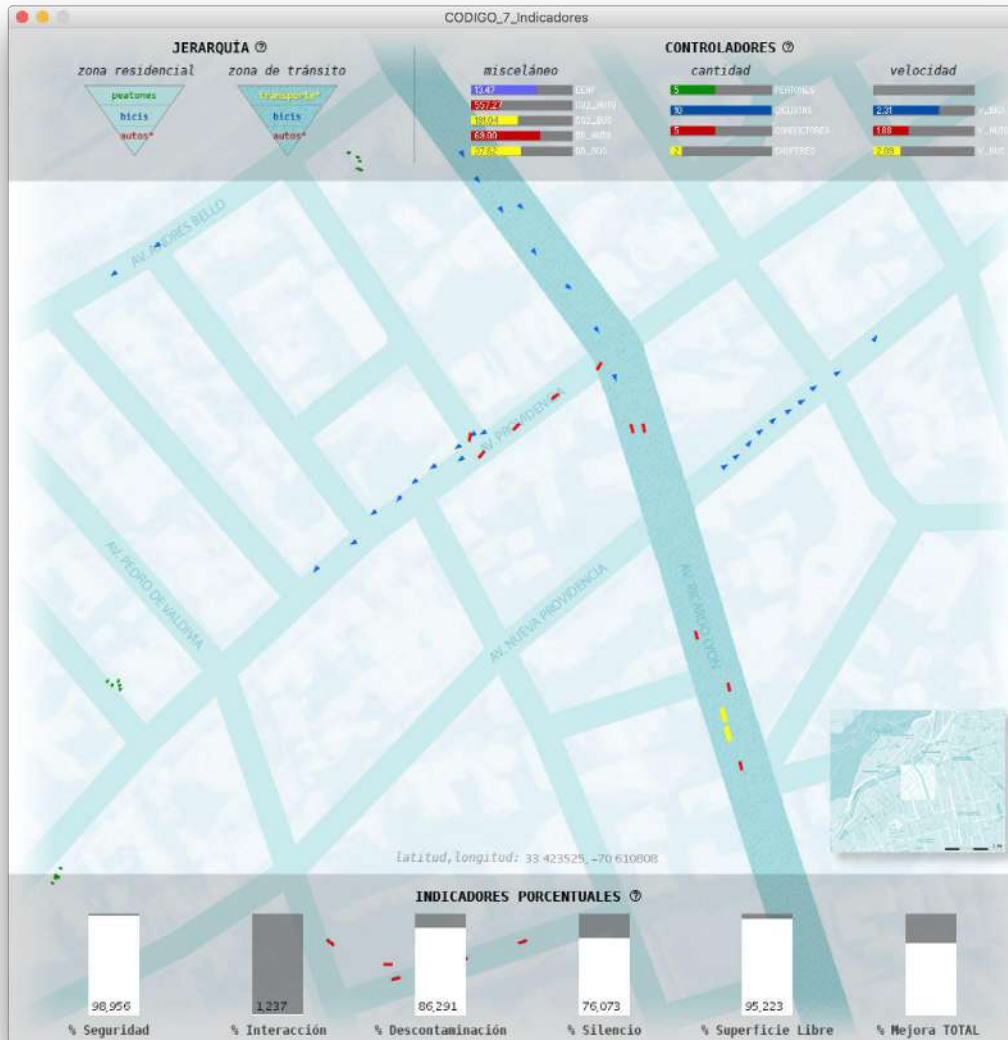
EMISIÓN

$$Qb \times Pb + Qa \times Pa$$



VERSIÓN 07

la fea hija perdida



COMENTARIOS GENERALES

Se intentó hacer del programa una experiencia más intuitiva. Ensanchando la calle de los buses se volvía innecesario tener que hacer una función específica para revelar las zonas de transporte y las de residencia.

Para explicar las jerarquías se generaron dos pirámides invertidas, una para cada zona.

Los porcentajes de los indicadores nacen a partir de cruces de datos concretos y permiten demostrar el impacto de la propuesta ecosistémica de transporte de manera dinámica.

Separar zonas de tránsito y las zonas de residencia y lo agentes correspondientes se logró al forzar los paths de cada tipo de agente, pero no se pudo hacer una función para resolver tal objetivo.

Cabe mencionar que la estética, diagramación y lectura dejan que desear en esta versión.

Este código se parece demasiado a nivel de programación al código 8, el final, es por esto que sólo se analizará el código 8 para no caer en la redundancia.

VERSIÓN 07

reflexión y proyección

CONCLUSIÓN GENERAL

Se lograron resolver los encuentros, el uso apropiado de los sliders, la interfaz dinámica para los indicadores y su traducción matemática a la de la programación. Es un proyecto de programación más cerrado y redondo. Pero, carece de usabilidad, buena lectura, composición, contraste, buen uso del color, entre otros detalles que lo alejan de ser un simulador como tal. Es absolutamente necesario hacer una mejora en lo estético, intuitivo y comunicativo.

Fue una versión más bien de prueba de una nueva estética, pero sin duda le queda bastante para mejorar su calidad de experiencia y uso.

PRÓXIMOS PASOS

1. Mejorar la estética general con la ayuda de referentes de plataformas digitales para entender como componen desde lo más general a los detalles que lo distinguen de otros programas.
2. Replantearse la usabilidad, el control y el uso de múltiples capas. Esto incluye la composición, orden de lectura y las proporciones del simulador.
3. Mejorar la visibilidad por color. Crear una paleta de colores suficientemente versátil como para desempeñarse en cualquier pantalla sin dejar de mantener una identidad particular y propia.
4. Repensar las variables de control y su influencia con los indicadores de porcentaje. Ajustar y afinar la capacidad de interacción con los sliders, pensando en los usuarios, y mejorar la incidencia de estos valores en el futuro gráfico dinámico de los indicadores y la tasa de sustentabilidad.
5. Mejorar la jerarquía visual general para lograr una equilibrada composición entre imagen y controles. Respetando los cánones implícitos de los juegos, programas y páginas web.

ENTREVISTA Y VALIDACIÓN CON RICARDO VEGA

diseñador – artista
experto en visualización
de información y
programación aplicada



Es Diseñador y artista, MFA Technology (Parsons, The New School, Nueva York), trabaja en temas relativos a la visualización de información, programación aplicada en el diseño y el arte. Tiene particular interés en temas relativos a la tecnología y sus implicancias sociales, culturales y artísticas. (diseno.uc.cl) Transcripción de la entrevista adjunta al final del libro.

REFLEXIÓN Y APORTES

Al comienzo hizo breves y precisos comentarios sobre la estética, como bajarle la opacidad al fondo, generar más contraste entre la información que importa y la que no; para lograr una jerarquía visual.

Comparó el trasfondo del sistema con los programas de NetLogo, unos simuladores que permitían emular ecosistemas de predador - presa el cual ajustando algunos parámetros lograban encontrar el equilibrio, todo en base a la relación *Lotka-Volterra*. Hizo cuestionarme dónde está el equilibrio en mi programa, o en que umbral se gatilla el caos.

Me hizo cuestionar cuál es el medio de comparación entre la ciudad con y la ciudad el sistema distribuido autónomo. Con el modelo de una manera simple **“puedes medir o tasar el impacto de las diferentes condiciones que controlas”**. Este punto fue clave para revalidar la idea de disponer con indicadores y una tasa global de la sustentabilidad se podría decir.

ENTREVISTA Y VALIDACIÓN CON OSCAR FIGUEROA

economista, Doctor en urbanismo y experto en transporte



Economista e Ingeniero Comercial, Universidad de Chile, Doctor en Urbanismo, Universidad de París. Profesor del Instituto de Estudios Urbanos y Territoriales de la Universidad Católica de Chile. Fue, durante 10 años, director del programa de posgrado en Desarrollo Urbano de la misma Universidad. Además posee, más de 30 años de experiencia en temas de movilidad, transporte e infraestructura, como académico y como profesional. Ha sido consultor del Banco Mundial, del BID, de la Corporación Andina de Fomento y de Naciones Unidas; y ha asesorado o realizado consultoría en prácticamente todos los países de la región latinoamericana. www.dese.cl Transcripción de la entrevista adjunta al final del libro.

REFLEXIÓN Y APORTES

Me comentó que era no solo visualmente atractivo y altamente didáctico, si no también *“es de gran interés para municipalidades como un micro simulador a la hora de tener que tomar decisiones y planificar.”*

Me comentaba que los simuladores actuales era planos blancos con una infinidad de flechas apuntando hacia todos lados que lo volvían inentendibles; *“Tu modelo es didáctico y es menos costoso (que no simular)”*.

Luego de aprobar mis variables controladas y el marco teórico y las tendencias de transporte reconocidas me hizo preguntarme cuáles son las variables que se le pueden ir agregando en un futuro, si colisiones, encuentros, direcciones, tipos de calle o vehículos, memoria, entre otras. Recalcó después que la pirámide invertida de la ITPD está reconocida en casi todos lados incluido Chile. Y agregó que *“el transporte del futuro va a perseguir la independencia y maximizar la experiencia”* por medio de la tecnología.

***SIMULADOR DE
FLUJOS E IMPACTO DE
TRANSPORTE AUTÓNOMO,
EMERGENTE Y DISTRIBUIDO***

EL PROTOTIPO FINAL
VERSIÓN 08

ASPECTO FINAL Y COMPOSICIÓN DEL SIMULADOR POR CAPAS

El aspecto general del simulador está inspirado en el proyecto Senseable del MIT (.edu) y la plataforma de Smart-citizen (.me). Cargadas con colores oscuros de fondo y con una saturada paleta para los elementos interactivos y dinámicos*.

Se aprovecha la plataforma digital para crear diferentes capas de información, permitiéndole al usuario ocultar o revelar diferentes capas para aprovechar de mejor manera el espacio de pantalla y para jerarquizar la información según importancia.

Esta naturaleza intuitiva del simulador encontrada en las capas de información está inspirada en juegos como el Plethora Project o PaintShop del GTAV, donde un espacio reducido de la pantalla está dedicado a detalles que afectan la interacción importante y móvil, que en este caso es el mapa con los agentes.

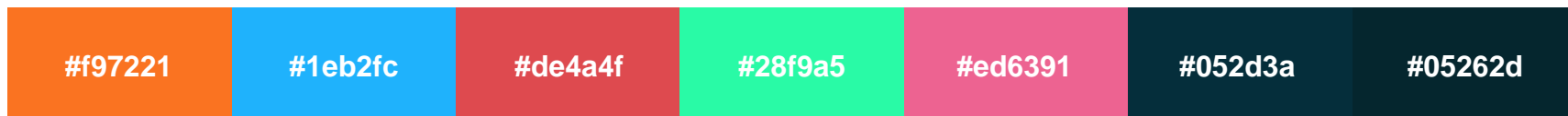
En el fondo está el mapa de Google Maps adaptado a la paleta de colores y es la capa de más abajo del simulador

La siguiente capa es la de los agentes, esta no fue digitalmente compuesta; fue programáticamente diseñada.

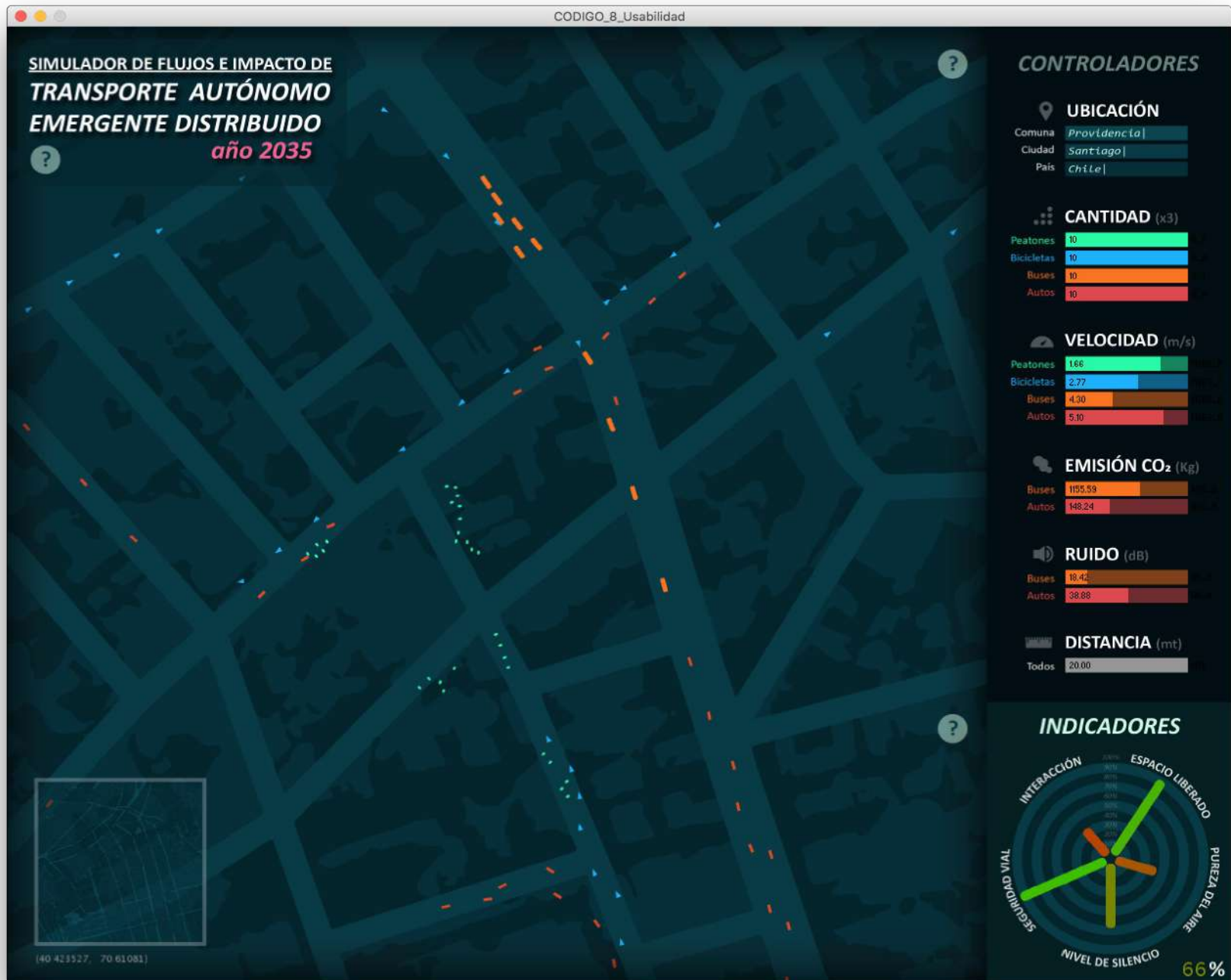
La capa que sigue la de la interfaz, esta contiene datos inmóviles que van por sobre los agentes móviles y el mapa. Esta contiene los signos de interrogación y al costado derecho se encuentra la barra interactiva con las bases para 13 sliders, que permiten la manipulación de datos variables que están relacionado con el funcionamiento del programa en sí. Tales datos afectan de manera dinámica y en tiempo real el gráfico de los “Indicadores” que se encuentra en la esquina inferior derecha. Los sliders y el gráfico dinámico son la última capa de información.

Los tres signos de interrogación son botones interactivos que revelan más información sobre los elementos básicos de la interfaz.

En las páginas siguientes se encuentran las capas de información desglosadas y explicadas.



**Los colores varían entre la pantalla digital y la impresión*




Aspecto final del proyecto. Esta captura es del simulador en funcionamiento y con los controladores ya alterados.



El Mapa es la base visual del programa y la base de los puntos de coordenadas de los paths.

SIMULADOR DE FLUJOS E IMPACTO DE TRANSPORTE AUTÓNOMO EMERGENTE DISTRIBUIDO año 2035



CONTROLOADORES

UBICACIÓN

Comuna:

Ciudad:

País:

CANTIDAD (x3)

Peatones:

Bicicletas:

Buses:

Autos:

VELOCIDAD (m/s)

Peatones:

Bicicletas:

Buses:

Autos:

EMISIÓN CO₂ (Kg)

Buses:

Autos:

RUIDO (dB)

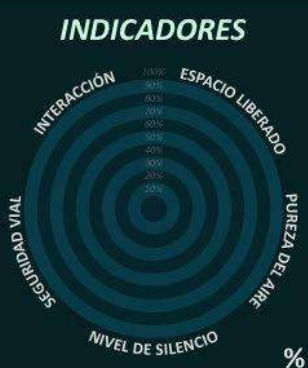
Buses:

Autos:

DISTANCIA (mt)

Todos:

INDICADORES



La Interfaz, la segunda capa, es la imagen base que soporta todos los elementos dinámicos como lo son los sliders e indicadores.

SIMULADOR DE FLUJOS E IMPACTO DE TRANSPORTE AUTÓNOMO EMERGENTE DISTRIBUIDO

año 2035



Las ciudades del mundo ocupan 3% del suelo de la Tierra y producen 75% de las millones de toneladas de las emisiones de carbono. Se estima que para el año 2030 el 60% de la población mundial vivirá en zonas urbanas. *UN.org*

Es por esto que las ciudades deben prepararse para el desarrollo sustentable tanto económica, como social y ambientalmente. Para esto es clave el desarrollo de un SISTEMA DE TRANSPORTE que integre en su esencia:

1. **JERARQUIZACIÓN** entre distintos tipos de movilidad según la propia sustentabilidad. *MexicoITPD.org*



2. **ORGANIZACIÓN EMERGENTE:** sistema distribuido, escalable y orgánico. *Sistemas Emergentes; S. Johnson*



3. **INTEGRACIÓN INTERMODAL:** tecnologías autónomas soportan una red sofisticada de identificación abastecida por energías renovables. *Acuerdo de París*



¿Qué hacen los sliders?



UBICACIÓN

Se puede ingresar una ubicación específica para cambiar de mapa en el mundo. *(Versión futura)*

CANTIDAD

Con estos controles se puede variar la cantidad de agentes móviles por tipo. Cada tipo de agente está agrupado por sus distintos puntos de origen. El valor del control es un tercio del total. La cantidad de agentes afectan todos los indicadores que inciden en la tasa de sustentabilidad.

VELOCIDAD

Sirven para controlar la velocidad máxima a la que pueden llegar cada tipo de movilidad. Estos valores afectan a los indicadores de la seguridad vial e interacción urbana.

EMISIÓN DE CO₂

La cantidad de kilogramos que contaminan ambos agentes motorizados son anuales. Suponiendo las nuevas tecnologías y fuentes de energía renovable se puede controlar las cantidades de contaminantes al cambiar las tecnologías del transporte. Afectan la pureza del aire.

RUIDO

Dado el cambio de tecnologías de combustión por las de transmisión eléctrica los motores son menos ruidos y algunos casi imperceptibles. Estos factores afectan la percepción sonora y la contaminación acústica final.

DISTANCIA

Es el tamaño del radio perceptivo sensible por el cual reaccionan los agentes unos con otros. Afectan directamente la seguridad vial e interacción.



CONTROLADORES

UBICACIÓN

Comuna:
 Ciudad:
 País:

CANTIDAD (x3)

Peatones:
 Bicicletas:
 Buses:
 Autos:

VELOCIDAD (m/s)

Peatones:
 Bicicletas:
 Buses:
 Autos:

EMISIÓN CO₂ (Kg)

Buses:
 Autos:

RUIDO (dB)

Buses:
 Autos:

DISTANCIA (mt)

Todos:

INDICADORES



%

Los pop ups superiores explican por separado el contexto del transporte sustentable futuro y la utilidad de los sliders interactivos.

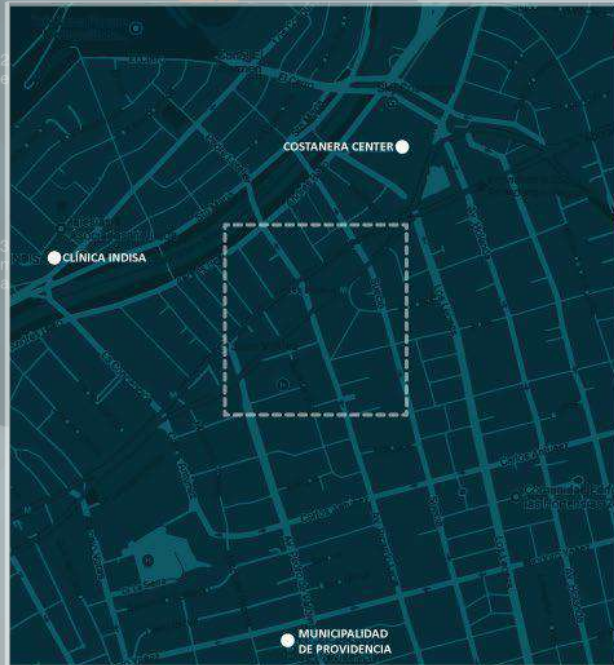
SIMULADOR DE FLUJOS E IMPACTO DE TRANSPORTE AUTÓNOMO EMERGENTE DISTRIBUIDO

año 2035

Las ciudades del mundo ocupan 3% del suelo de la Tierra y producen 75% de las millones de toneladas de las emisiones de carbono. Se estima que para el año 2030 el 60% de la población mundial vivirá en zonas urbanas. [UN.org](#)

Es por esto que las ciudades deben prepararse para el desarrollo sustentable tanto económica, como social y ambientalmente. Para esto es clave el desarrollo de un SISTEMA DE TRANSPORTE que integre en su esencia:

1. JERARQUIZACIÓN entre distintos tipos de movilidad según la propia sustentabilidad. [MexicoITPD.org](#)



¿Qué hacen los sliders?

UBICACIÓN
Se puede ingresar una ubicación específica para cambiar de mapa en el mundo. *(Versión futura)*

CANTIDAD
Se puede ingresar la cantidad de agentes de cada tipo.

Fórmulas de los Indicadores

ESPACIO LIBERADO
A la superficie total de calles se le resta la suma de todo el espacio S usado por cada uno de los agentes Q .

$$Qp \times Sp + Qc \times Sc + Qb \times Sb + Qa \times Sa$$

PUREZA DEL AIRE
Corresponde a la suma de las emisiones P variables de CO2 de todos los buses y autos Q .

$$Qb \times Pb + Qa \times Pa$$

NIVEL DE SILENCIO
Corresponde a la suma lineal de los decibeles R variables de todos los buses y autos Q .

$$Qb \times Rb + Qa \times Ra$$

SEGURIDAD VIAL
Es la relación que hay entre la distancia D a la que los agentes reaccionan entre sí y la velocidades máximas a las que puede llegar cada uno.

$$\frac{D}{Vc \times Vb \times Va}$$

INTERACCIÓN PEATONAL
Es la razón entre las posibles interacciones de todas las personas Q , multiplicado por la distancia D y las velocidades y cantidades de las bicicletas y autos Q .

$$\frac{(Qp - 1) \times Qp \times D}{Vc \times Qc \times Va \times Qa}$$

Tasa de Sustentabilidad

PROMEDIO
De todos los indicadores y valor indicador final del impacto. Expresado como porcentaje se encuentra en la esquina inferior derecha.

CONTROLADORES

UBICACIÓN
Comuna:
Ciudad:
País:

CANTIDAD (x3)
Peatones:
Bicicletas:
Buses:
Autos:

VELOCIDAD (m/s)
Peatones:
Bicicletas:
Buses:
Autos:

EMISIÓN CO2 (Kg)
Buses:
Autos:

RUIDO (dB)
Buses:
Autos:

DISTANCIA (mt)
Todos:

INDICADORES



Los pop ups de abajo muestran el mapa ampliado de la ubicación y fórmulas detrás las barras interactivas de los indicadores.

FUNCIONAMIENTO, COMPONENTES Y USO DEL SIMULADOR

DESCRIPCIÓN GENERAL



Mientras los agentes comienzan a invadir de afuera hacia adentro, con una grata ansiedad el espectador espera el cruce de agentes en las grandes intersecciones. Algunos más frenéticos que otros se desplazan con un destino claro por la ciudad.

A la hora de llegar a las intersecciones sorprendentemente las velocidades cambian. Al pasar un peatón todos se detienen sin excepción. Bicicletas y autos respetan su velocidad y su espacio personal. Y una vez que se esfuma, el auto esperará paciente (y programadamente) su turno de partir libre por la calzada.

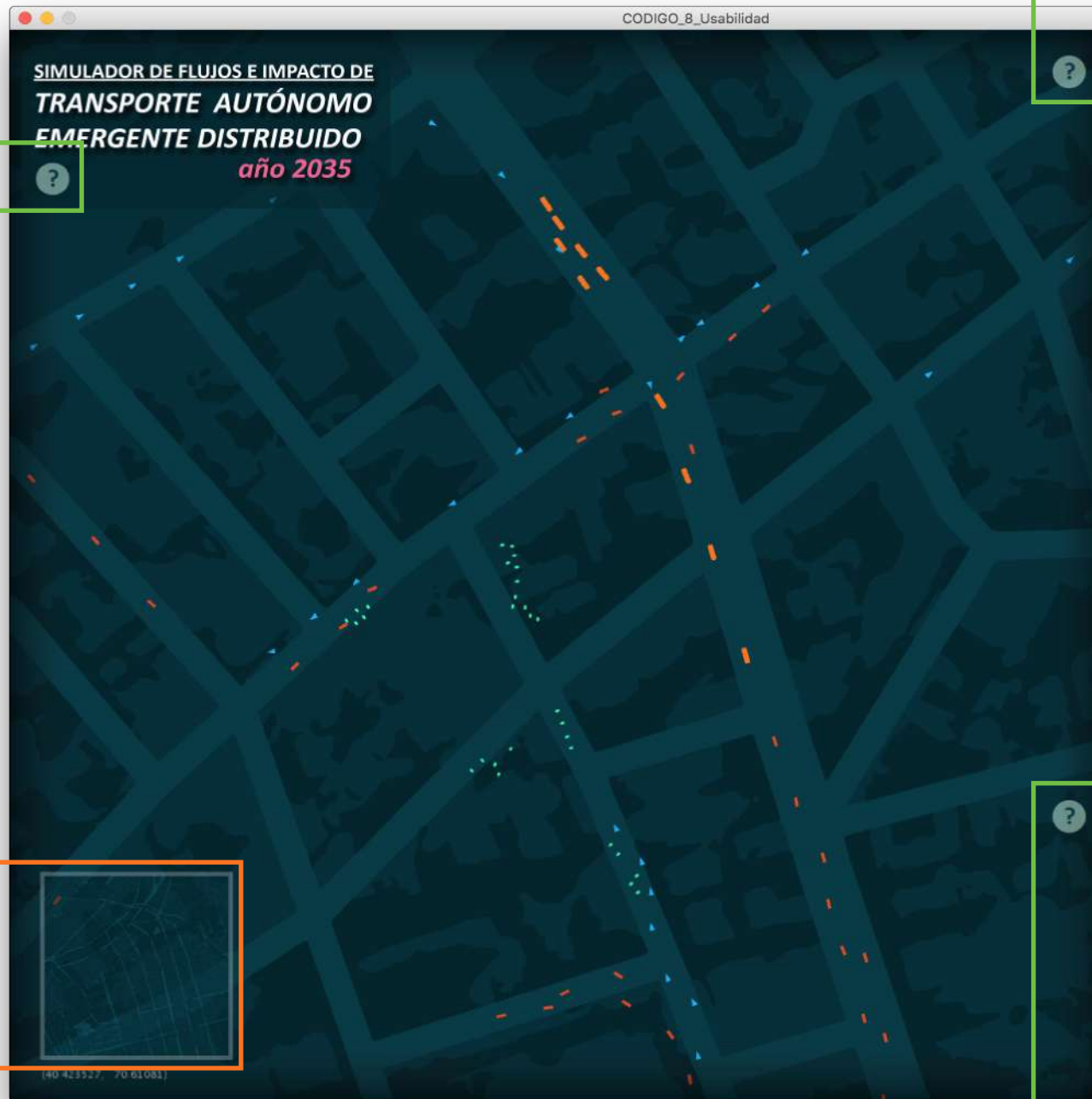
Tomando decisiones locales de vecino a vecino comienza a aparecer un orden global, un flujo y un ritmo de ciudad. Todas las funciones están en la pág. 158.

INTERACCIÓN INTUITIVA

Hay *17 botones interactivos* en el proyecto los cuales se usan con el cursor. El conjunto más llamativo el de la barra lateral con los *controladores* de variables (*figuraC*) Las bien comunes y silvestres barras interactivas, decoran como un árbol de navidad al modelo de transporte. Nace la curiosidad y las ganas de mover tales parámetros disponibles. - ¿Más autos? ¿Más emisiones? ¿Menos distancia? - Espera. ¿Por qué se está moviendo el gráfico de la parte de abajo?

Las variables controladas revela la capacidad dinámica del gráfico de abajo; titulado *Indicadores. (Figura i)* Este mide la interacción entre las personas, la superficie utilizada, la contaminación del aire o acústica y la seguridad. Entre cruces de datos y relaciones de sentido común permite que las barras del gráfico se muevan con fluidez y cambien de manera intuitiva de color.

Tres botones con *signos de interrogación* y un *mapa* esperan ser apretados por el cursor: Las figuras 1 a 4 vistas de izquierda a derecha y de arriba hacia abajo. El usuario busca descubrir su contenido.

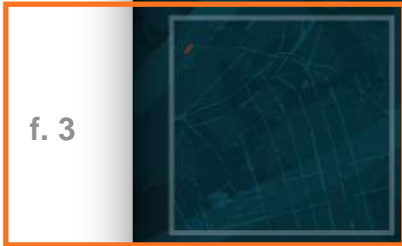


f. 1

f. 2



f. c



f. 4



f. i

POPUP 1 – CONTEXTO

A modo de introducción en el primero popup se introducen los antecedentes reales y la esencia del transporte: jerarquía, emergencia y automatismo.

Las ciudades del mundo ocupan 3% del suelo de la Tierra y producen 75% de las emisiones de carbono. Se estima que para el año 2030 el 60% de la población mundial vivirá en zonas urbanas. UN.org

Es por esto que las ciudades deben prepararse para el desarrollo sustentable tanto económica, como social y ambientalmente. Para esto es clave el desarrollo de un Sistema de Transporte que integre en su esencia:

1. Jerarquización entre distintos tipos de movilidad según la propia sustentabilidad. Mexico.ITPD.org

2. Organización emergente: sistema distribuido, escalable y orgánico. Sistemas Emergentes, S. Johnson

3. Integración intermodal: tecnologías autónomas soportan una red sofisticada de identificación abastecida por energías renovables. Acuerdo de París

SIMULADOR DE FLUJOS E IMPACTO DE TRANSPORTE AUTÓNOMO EMERGENTE DISTRIBUIDO
año 2035

Las ciudades del mundo ocupan 3% del suelo de la Tierra y producen 75% de las emisiones de carbono. Se estima que para el año 2030 el 60% de la población mundial vivirá en zonas urbanas. *UN.org*

Es por esto que las ciudades deben prepararse para el desarrollo sustentable tanto económica, como social y ambientalmente. Para esto es clave el desarrollo de un SISTEMA DE TRANSPORTE que integre en su esencia:

- 1. JERARQUIZACIÓN** entre distintos tipos de movilidad según la propia sustentabilidad. *Mexico.ITPD.org*
- 2. ORGANIZACIÓN EMERGENTE:** sistema distribuido, escalable y orgánico. *Sistemas Emergentes, S. Johnson*
- 3. INTEGRACIÓN INTERMODAL:** tecnologías autónomas soportan una red sofisticada de identificación abastecida por energías renovables. *Acuerdo de París*

(88 423527, 71 61081)

¿Qué hacen los sliders?

UBICACIÓN
Se puede ingresar una ubicación específica para cambiar de mapa en el mundo. *(Versión futura)*

CANTIDAD
Con estos controles se puede variar la cantidad de agentes móviles por tipo. Cada tipo de agente está agrupado por sus distintos puntos de origen. El valor del control es un tercio del total. La cantidad de agentes afectan todos los indicadores que inciden en la tasa de sustentabilidad.

VELOCIDAD
Sirven para controlar la velocidad máxima a la que pueden llegar cada tipo de movilidad. Estos valores afectan a los indicadores de la seguridad vial e interacción urbana.

EMISIÓN DE CO2
La cantidad de kilogramos que contaminan ambos agentes motorizados son anuales. Suponiendo las nuevas tecnologías y fuentes de energía renovable se puede controlar las cantidades de contaminantes al cambiar las tecnologías del transporte. Afectan la pureza del aire.

RUIDO
Dado el cambio de tecnologías de combustión por las de transmisión eléctrica los motores son menos ruidos y algunos casi imperceptibles. Estos factores afectan la percepción sonora y la contaminación acústica final.

DISTANCIA
Es el tamaño del radio perceptivo sensible por el cual reaccionan los agentes unos con otros. Afectan directamente la seguridad vial e interacción.

CONTROLADORES

UBICACIÓN

Comuna:

Ciudad:

País:

CANTIDAD (x3)

Peatones	10
Bicicletas	10
Buses	10
Autos	10

VELOCIDAD (m/s)

Peatones	1.66
Bicicletas	2.77
Buses	4.30
Autos	5.10

EMISIÓN CO₂ (Kg)

Buses	1155.59
Autos	148.24

RUIDO (dB)

Buses	18.42
Autos	38.88

DISTANCIA (mt)

Todos	20.00
-------	-------

INDICADORES

66%

POPUP 2 – SLIDERS

Estos controladores permiten manipular todas las variables más relevantes a la hora de controlar el ecosistema de transporte. La modificación de estas variables tiene una reacción inmediata y una relación directa con los valores de los indicadores (p. 132). Si se agregan personas o vehículos además de ver como van entrando más y más agentes al mapa desde afuera se alteran los valores de interacción, seguridad, espacio, silencio y pureza del aire simultáneamente.

Las 14 variables que se pueden modificar son: la cantidad de agentes, la velocidad máxima a la que pueden llegar, la distancia de sensibilidad y reacción (lo relacionado con la próxemica), la cantidad de kilogramos de CO₂ que emiten anualmente y el nivel de ruido en dB que producen los motores. Ambas últimas variables afectan solo a los buses y autos.

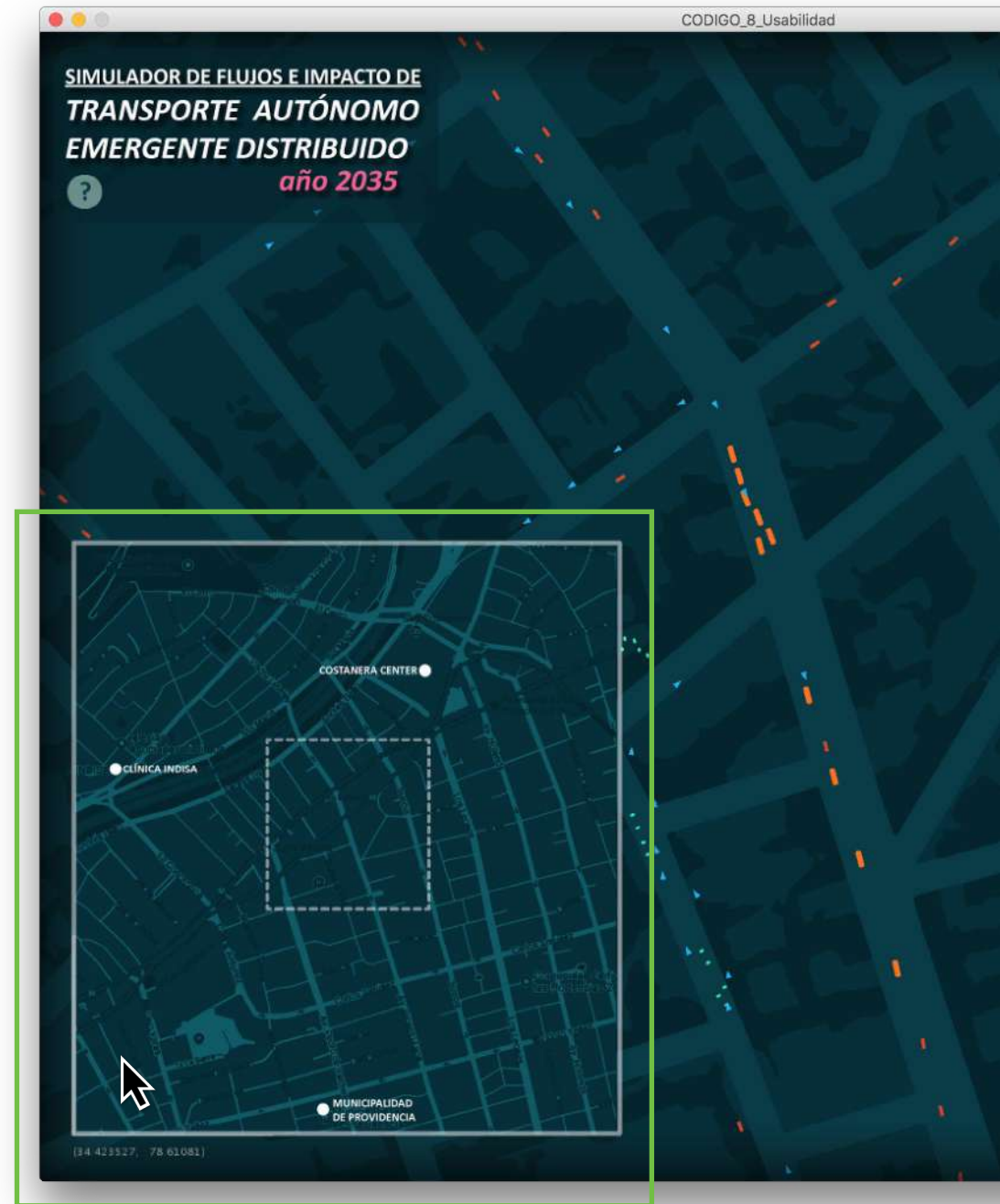
El recuadro para cambiar la ubicación es una maqueta que prototipa una función que debiese tener el simulador.

POPUP 3 – MAPA

Con el mismo cursor, al posarlo sobre el mini mapa, se despliega un mapa más general para ubicar el fragmento del modelo dentro de la ciudad.

Para una versión futura lo ideal sería poder arrastrar el mapa del modelo para cambiar de ubicación, así como se puede con Google Maps, o bien ingresar la ubicación deseada en la columna de los controladores.

Para esto debiese existir una correcta integración entre los vectores de los mapas y los paths. En el caso del prototipo, recordemos, están ingresadas cada una de las coordenadas manualmente.



CONTROLES

UBICACIÓN
 Comuna: Providencia |
 Ciudad: Santiago |
 País: Chile |

CANTIDAD (x3)

Peatones	10
Bicicletas	10
Buses	10
Autos	10

VELOCIDAD (m/s)

Peatones	1.66
Bicicletas	2.77
Buses	4.30
Autos	5.10

EMISIÓN CO₂ (Kg)

Buses	155.59
Autos	148.24

RUIDO (dB)

Buses	18.42
Autos	38.88

DISTANCIA (mt)

Todos	20.00
-------	-------

INDICADORES

INTERACCIÓN PEATONAL
 ESPACIO LIBERADO
 PUREZA DEL AIRE
 NIVEL DE SILENCIO

66%

Fórmulas de los Indicadores

ESPACIO LIBERADO
 A la superficie total de calles se le resta la suma de todo el espacio S usado por cada uno de los agentes Q .

$$Qp \times Sp + Qc \times Sc + Qb \times Sb + Qa \times Sa$$

PUREZA DEL AIRE
 Corresponde a la suma de las emisiones P variables de CO₂ de todos los buses y autos Q .

$$Qb \times Pb + Qa \times Pa$$

NIVEL DE SILENCIO
 Corresponde a la suma lineal de los decibeles R variables de todos los buses y autos Q .

$$Qb \times Rb + Qa \times Ra$$

SEGURIDAD VIAL
 Es la relación que hay entre la distancia D a la que los agentes reaccionan entre sí y la velocidades máximas a las que puede llegar cada uno.

$$\frac{D}{Vc \times Vb \times Va}$$

INTERACCIÓN PEATONAL
 Es la razón entre las posibles interacciones de todas las personas Q , multiplicado por la distancia D y las velocidades y cantidades de las bicicletas y autos Q .

$$\frac{(Qp - 1) \times Qp \times D}{Vc \times Qc \times Va \times Qa}$$

Tasa de Sustentabilidad

PROMEDIO
 De todos los indicadores y valor indicador final del impacto. Expresado como porcentaje se encuentra en la esquina inferior derecha.

POPUP 4 – INDICADORES

Los indicadores porcentuales (p.132) son los valores que permiten la comparación entre cruces de datos variables. Tales cruces de datos aparecen en el programa expresados como las fórmulas usadas en el código. Están graficados en la parte inferior derecha del simulador en un diagrama de barras representado de manera radial.

Nótese que todos los valores fueron normalizados en escala de porcentaje; es menos relevante saber cuantos kilogramos de CO₂ contaminan los agentes exactamente, que cuánto reduciría la contaminación en relación al valor máximo: lo que ya se contamina hoy.

Los datos que generan los porcentajes tienen relación directa con mediciones de estudios de casos actuales, siendo estos últimos un conjunto de valores de referencia que permiten estimar el nivel de impacto total.

Los valores comandados por los sliders (inputs) son promediados y resultan en la tasa de sustentabilidad: el indicador general que permitiría ayudar en la toma de decisiones. Este valor esta representado en la esquina derecha abajo.

DECISIONES GRÁFICAS DINÁMICAS E INTERACTIVAS

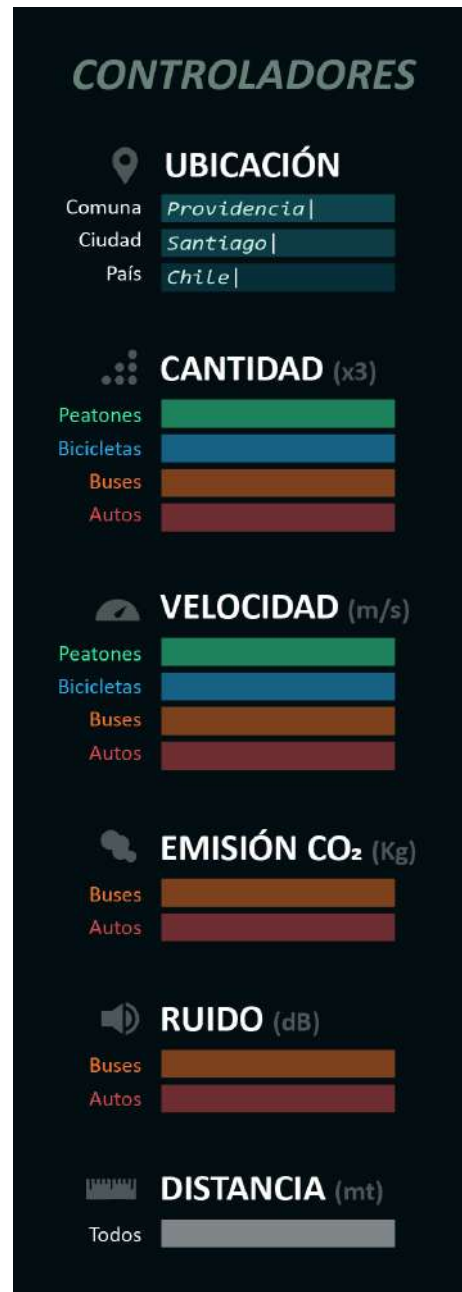
Las barras del gráfico de indicadores (f.1) al igual que el porcentaje de tasa de sustentabilidad a medida que reduce su valor cambia su tonalidad hacia el rojo, y si sube hacia el verde. (f.2) Así también lo hacen los buses (f.3) De esta manera se relaciona con mayor facilidad el impacto que están teniendo los valores concebidos por los controladores.

Los sliders cambian de color al estar activos: al estar el cursor encima de ellos, y también cuando se les pincha'. (f.4)

Al costado de cada slider está anotada la unidad de medida correspondiente a lo que representan los valores. (f.5) Y antes de cada título de los controladores hay iconos para identificar con mayor velocidad lo que controlan. (f.6)

Bajo el mapa están los valores de la latitud y longitud (f.7), estos números cambian dependiendo de la ubicación del mouse de manera instantánea.

Las fórmulas fueron escritas en LaTeX, un lenguaje abierto para escribir con las sutilezas que las fórmulas matemáticas tienen.



A la izquierda se encuentra la imagen 'pura' usada de fondo para la columna de los sliders. A la derecha se encuentra la versión del programa con sus sliders activos y funcionando.

f. 4

f. 5

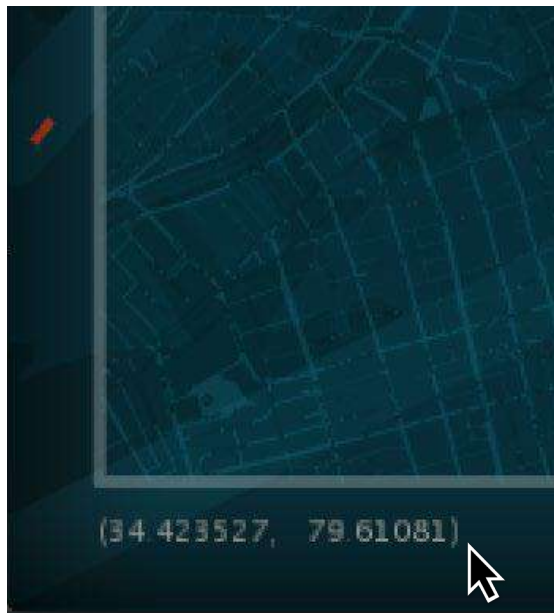
f. 6



f. 1



f. 2



f. 8



f. 3



REFLEXIÓN DEL SIMULADOR

Del proyecto alguna de las bondades más destacables fueron verbalizadas por Óscar Figueroa, el economista y doctor en urbanismo. Refiriéndose a la atingencia, utilidad y potencial que tiene un modelo así de didáctico como herramienta de ayuda para la toma de decisiones y planificar según las distintas voluntades que puede tener un funcionario del gobierno o municipalidad.

Siendo punto de referencia los actuales modelos de simulación de transporte, Óscar que tiene más de 10 años de experiencia en el tema del transporte encontraba sumamente interesante el concepto de un micro-simulador como una guía para resolver casos a escalas menores y medianas. Bastaría incluir algunas variables más en el *backend*, con la ayuda de un ingeniero en programación, para poder complejizar de buena manera y explotarle así el potencial a la plataforma actual.

Las tendencias de transporte y la de las plataformas didácticas se ven culminadas en el simulador. Incluir desde el comienzo la jerarquía de la deseabilidad, la automatización de los móviles y la tasa de la sustentabilidad (con los indicadores respectivos) fuerzan la lectura crítica respecto a lo económico, ambiental y social; o sea lo sustentable.

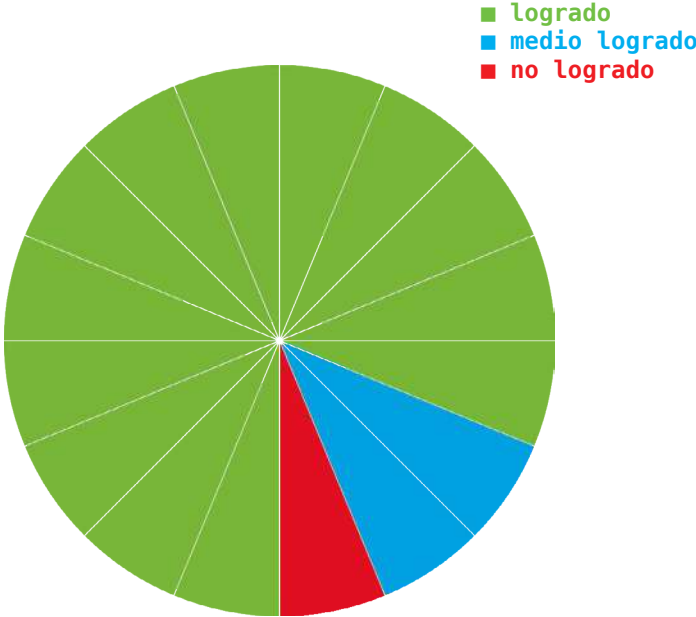
Así como los simuladores de ecosistemas entre animales de NetLogo, este modelo es un resumido sistema de ecología de transporte que reduce su cantidad de variables controlas al mínimo sin perder la esencia del mensaje que se busca por atrás, que es el equilibrio en la sustentabilidad.

Hay una particularidad de la relación de los datos, todos los indicadores se ven afectados aparentemente de manera lineal, pero al llevar algún parámetro al extremo los otros dejando de afectar tanto en el gráfico, esto sucede por las relaciones no lineales. Y con esto se quiere decir que hay umbrales en lo que puede quedar el equilibrio ecológico o bien encontrarse con un punto de no retorno.

Algunos beneficios claros que posee el modelo son: el considerar el sistema como un todo conectado, asumiendo las complicaciones que puede tener la inter modalidad; el estar enfocado en la sustentabilidad, un objetivo de desarrollo común y global; considerar los autos autónomos como una realidad próxima; y ser de bajo costo de producción y uso.

PORCENTAJE DE LOGRO DE LOS OBJETIVOS DEL PROYECTO

AYUDAR EN LA TOMA DE DECISIONES
 DIDÁCTICO E INTUITIVO
 QUE PERMITA MEDIR IMPACTO CON VARIABLES ATINGENTES
 ENFOCADO EN LA SUSTENTABILIDAD
 QUE INTEGRE LAS NUEVAS TENDENCIAS
 NATURALEZA INTERACTIVA
 CON DATOS MANIPULABLE QUE AFECTEN EN TIEMPO REAL
 SIMULAR LOS FLUJOS DE LOS AGENTES
 ESTAR UBICADO EN UN ESPACIO DE CASO URBANO REAL
 DE ORDEN EMERGENTE: DISTRIBUIDO Y ESCALABLE
 QUE LOGRE UNIFICAR Y JERARQUIZAR LA MOVILIDAD
 REDISTRIBUIR EL ESPACIO DE LA CIUDAD
 USAR AUTOMATISMO COMO SOPORTE DEL SISTEMA EMERGENTE
 SIMULAR EL COMPORTAMIENTO INTERMODAL
 FOMENTAR EL USO DE ENERGÍAS SUSTENTABLES
 MEJORAR LA ACCESIBILIDAD DE TRANSPORTE



LOGRO TOTAL: **88%**

PROYECCIONES Y MEJORAS FUTURAS DEL SIMULADOR

hacia dónde debería evolucionar el simulador

En futuras versiones se podría considerar agregar una opción para visualizar el modelo en 3 dimensiones, para poder comunicar el sistema desde puntos de vista más comunes y familiares para quien no tiene la agudeza técnica.

Pensando en los investigadores que usan el programa sería bueno poder exportar los parámetros manipulados en el simulador así como también los datos que se están generando de forma continua.

También sería menester encontrar la forma de incluir más variables controlables relevantes que afectan directamente el ecosistema, como lo son: las colisiones, los puntos calientes (según el número de intersecciones), tipo de calles, tipo de personalidad pensando en la toma de decisiones de los agentes.

Lo ideal en interacción de los agentes sería agregar la capacidad de almacenar la experiencia de viajes pasados para decidir por sí solos cuál modo de transporte prefieren, evaluando ellos entre sustentabilidad, experiencia y tiempo. Todo esto además de afinar los encuentros y las reacciones de y entre los agentes móviles.

Por último y no menos importante sería esencial poder integrar, ya sea con una librería u otro proceso programático, distintos mapas reales construidos a partir de vectores o grafos permitiendo así el uso del simulador en diferentes partes del mundo. Esto da pie para pensar que quizás el español no debiese ser el idioma predeterminando del simulador, dado que no es el idioma más hablado en el mundo digital a diferencia del inglés, que si se encuentra en otros simuladores.

EL CÓDIGO DE LA VERSIÓN FINAL DEL SIMULADOR

tras bambalinas

En las dos páginas siguientes se encuentra el desglose y explicación de la última versión lograda del simulador. Escrita de manera tal que permite entender que está realmente haciendo bloque por bloque, y presentado en el mismo orden en el que se desarrolló el código.

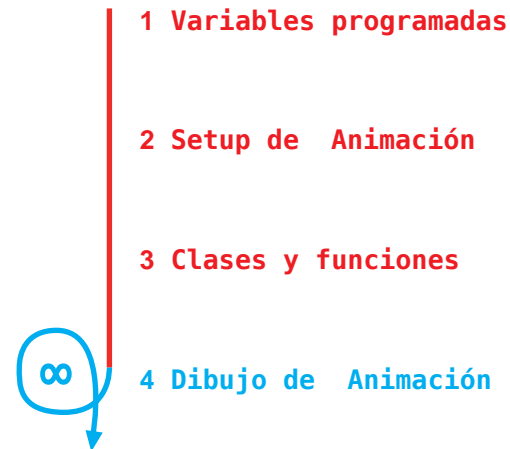
Seguido del gran esquema está un pequeño mapa de flujos que permite entender de que manera el computador procesa la información escrita en el sketch de Processing.

Más adelante está la captura de todo el código, incluido tanto el programa como las clases de los agentes. Tal desglose está completo con el fin de poder acceder a él con total transparencia y facilidad, pensando sobre todo en el caso de querer recurrir a él para explicarlo o bien copiarlo o citarlo.

LAS FUNCIONES DEL CÓDIGO FINAL EN CASTELLANO

la misteriosa secuencia de instrucciones que ocurre detrás

La mayor parte de los códigos de diferentes lenguajes de programación, incluido Processing P3, comparten una meta estructura compuesta por 4 grandes partes.



1 La primera es donde se incluyen las variables globales que se usarán a lo largo de todo el programa.

2 La segunda parte, el void setup, corre una sola vez y esta carga toda la información que se usará en el programa y define el espacio de trabajo y el modo en el que se visualizarán los datos.

3 La tercera parte corresponde a las clases: objetos que poseen funciones propias, que son llamados desde el void draw para ser ejecutados.

4 La cuarta y última parte en correr se llama void draw; y es la encargada de la parte animada y dinámica del programa. Esta contiene y ejecuta todas las funciones de acción, reacción, comunicación de los datos ingresados desde las otras partes del código. Esta parte del código continúa corriendo infinitamente en loop * de arriba hacia abajo continuamente hasta que alguien detenga el programa o se produzca algún error en la lógica del código.

1 // ESTRUCTURA CÓDIGO VERSIÓN 08 //

Importación de
librería de sliders P5

Declaración de
imágenes
interruptores
paths (calles)
listas de agentes
cantidades de agentes
velocidades máximas
coordenadas intersecciones
diámetro atractores
ubicación de tractores
distancia de encuentro
superficie por agente
emisión de co2
decibeles de ruido

Función de tamaño de ventana

2

VOID SETUP () {
smooth rendering
frame rate fijo en 25
cargar imágenes ya declaradas
cargar sliders de librería
usar sliders y posicionarlos
crear paths*
creación de listas de agentes
}

4

VOID DRAW
() {
uso de imagen de Fondo
función de seguimiento de paths
agregar agentes de cada tipo
orígenes de agentes por grupo
condicional de límite de agentes
loop de todos con todos
si choca A con B limitar velocidad
funciones de personas (x3)
seguir Atractor
si sale de ventana desaparece
separarse de otras personas
}

//

funciones Transporte Público
seguir Atractor
si sale de ventana desaparece
separarse de Personas
separarse de otros Buses

funciones Bicicletas
seguir Atractor
si sale de ventana desaparece
separarse de Personas
separarse de otras Bicis
separarse de Buses

funciones Autos
seguir Atractor
si sale de ventana desaparece
separarse de Personas
separarse de Bicis
separarse de Buses
separarse de otros Autos

uso de imagen de Interfaz
declaración y fórmula de seguridad
declaración y fórmula de interacción
declaración y fórmula de polución
declaración y fórmula de ruido
declaración y fórmula de superficie
declaración y fórmula del % total
construcción del gráfico dinámico
declaración de
latitud y longitud dinámica
condicionales del cursor de los Pop Up
MAPA grande
POP 1 contexto
POP 2 controladores
POP 3 formulas

* }

3 **SECUENCIA DE COORDS DE PATHS () {**
Paths para cada grupo de agentes
}

**Los buses, autos y bicicletas son una extensión de la clase Persona; heredan todas sus funciones y variables básicas enlistadas a continuación. Y se les agrega a cada uno de los 3 la función de encuentro Jerarquizado.*

CLASE PERSONA {
Declaración
ubicación
velocidad
aceleración
diámetro
fuerza de virado
Constructor
seteo de variables
Función de velocidad y aceleración
Función de seguimiento de Atractor
Función de seguimiento de Paths
Función de funciones
Función de apariencia
Función de límites de ventana
Función de separación según diámetro
Función de virado hacia y por Paths
}

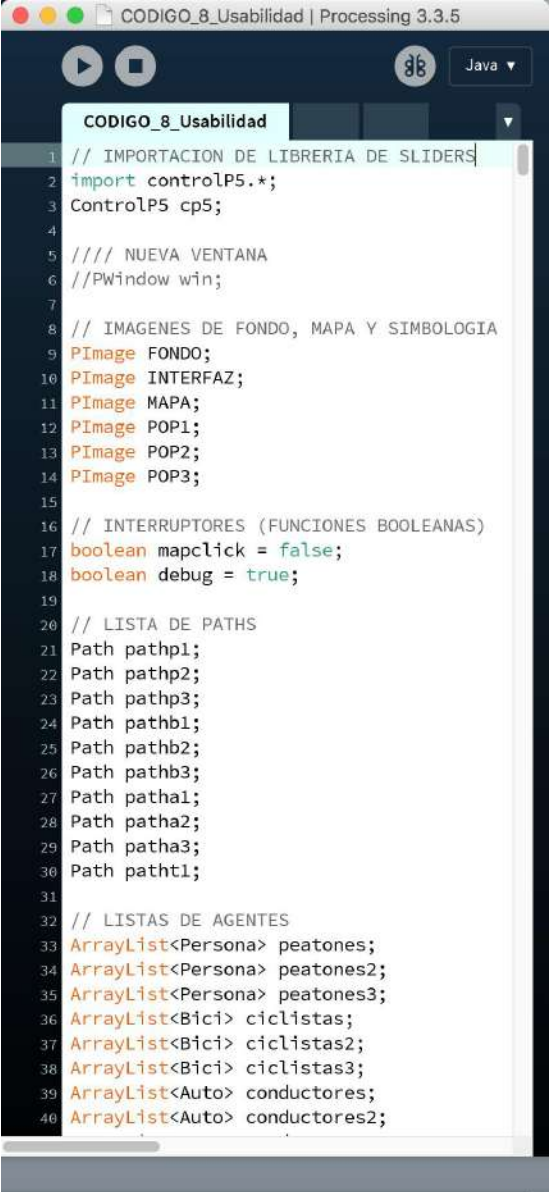
CLASE PATH () {
Declaración de conjunto de puntos
Declaración de Inicio
Identificación de puntos intermedios
Declaración de final
Función de apariencia según puntos
}

VERSIÓN 08

todas las líneas de código de la versión

Estas son las líneas de código de la última versión del simulador. Al igual que las otras versiones están presentadas en el orden en el que están escritas las pestañas del sketch de processing.

Esta versión tiene en total más de **1800 líneas de código.**

A screenshot of the Processing IDE interface. The window title is 'CODIGO_8_Usabilidad | Processing 3.3.5'. The code editor shows the following code:

```
1 // IMPORTACION DE LIBRERIA DE SLIDERS
2 import controlP5.*;
3 ControlP5 cp5;
4
5 //// NUEVA VENTANA
6 //PWindow win;
7
8 // IMAGENES DE FONDO, MAPA Y SIMBOLOGIA
9 PImage FONDO;
10 PImage INTERFAZ;
11 PImage MAPA;
12 PImage POP1;
13 PImage POP2;
14 PImage POP3;
15
16 // INTERRUPTORES (FUNCIONES BOOLEANAS)
17 boolean mapclick = false;
18 boolean debug = true;
19
20 // LISTA DE PATHS
21 Path pathp1;
22 Path pathp2;
23 Path pathp3;
24 Path pathb1;
25 Path pathb2;
26 Path pathb3;
27 Path patha1;
28 Path patha2;
29 Path patha3;
30 Path patht1;
31
32 // LISTAS DE AGENTES
33 ArrayList<Persona> peatones;
34 ArrayList<Persona> peatones2;
35 ArrayList<Persona> peatones3;
36 ArrayList<Bici> ciclistas;
37 ArrayList<Bici> ciclistas2;
38 ArrayList<Bici> ciclistas3;
39 ArrayList<Auto> conductores;
40 ArrayList<Auto> conductores2;
```

Pestaña 1: Programa 1/29


```
CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
41 ArrayList<Auto> conductores3;
42 ArrayList<TransportePublico> choferes;
43
44 // NUMERO MAX. DE AGENTES (PROPORCIONES)
45 int Q_p = 5;
46 int Q_b = 5;
47 int Q_a = 5;
48 int Q_t = 5;
49
50 // VELOCIDADES MAXIMAS DE AGENTES
51 float mseg_p = 1;
52 float mseg_c = 2;
53 float mseg_a = 3;
54 float mseg_b = 4;
55
56 //COORDS MAPA PROVIDENCIA
57 PVector v1 = new PVector(374, 0);
58 PVector v2 = new PVector(426, 0);
59 PVector v3 = new PVector(576, 0);
60 PVector v4 = new PVector(705, 0);
61 PVector v5 = new PVector(800, 0);
62 PVector v6 = new PVector(853, 0);
63 PVector v7 = new PVector(390, 33);
64 PVector v8 = new PVector(813, 21);
65 PVector v9 = new PVector(746, 57);
66 PVector v10 = new PVector(1000, 36);
67 PVector v11 = new PVector(728, 107);
68 PVector v12 = new PVector(272, 125);
69 PVector v13 = new PVector(670, 147);
70 PVector v14 = new PVector(869, 117);
71 PVector v15 = new PVector(140, 213);
72 PVector v16 = new PVector(718, 221);
73 PVector v17 = new PVector(1000, 172);
74 PVector v18 = new PVector(42, 272);
75 PVector v19 = new PVector(349, 304);
76 PVector v20 = new PVector(597, 310);
77 PVector v21 = new PVector(937, 231);
78 PVector v22 = new PVector(0, 295);
79 PVector v23 = new PVector(0, 388);
80 PVector v24 = new PVector(279, 395);
```

Pestaña 1: Programa 2/29

```
CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
81 PVector v25 = new PVector(465, 400);
82 PVector v26 = new PVector(805, 341);
83 PVector v27 = new PVector(1000, 427);
84 PVector v28 = new PVector(406, 455);
85 PVector v29 = new PVector(352, 491);
86 PVector v30 = new PVector(647, 471);
87 PVector v31 = new PVector(54, 440);
88 PVector v32 = new PVector(1000, 435);
89 PVector v33 = new PVector(285, 556);
90 PVector v34 = new PVector(0, 538);
91 PVector v35 = new PVector(209, 623);
92 PVector v36 = new PVector(483, 606);
93 PVector v37 = new PVector(697, 630);
94 PVector v38 = new PVector(891, 477);
95 PVector v39 = new PVector(133, 698);
96 PVector v40 = new PVector(277, 824);
97 PVector v41 = new PVector(526, 683);
98 PVector v42 = new PVector(731, 739);
99 PVector v43 = new PVector(1000, 666);
100 PVector v44 = new PVector(0, 852);
101 PVector v45 = new PVector(598, 862);
102 PVector v46 = new PVector(170, 896);
103 PVector v47 = new PVector(324, 953);
104 PVector v48 = new PVector(638, 966);
105 PVector v49 = new PVector(789, 922);
106 PVector v50 = new PVector(0, 977);
107 PVector v51 = new PVector(335, 1000);
108 PVector v52 = new PVector(649, 1000);
109 PVector v53 = new PVector(811, 1000);
110
111
112 // ATRACTORES Y SUS COORDENADAS
113 // Tamaño de atractores
114 float diam = 15;
115 // ATRACTORES DE PERSONAS
116 PVector cuadra1 = new
117     PVector(v53.x, v53.y+100);
118 PVector cuadra2 = new
119     PVector(v52.x, v52.y+100);
120 PVector cuadra3 = new
```

Pestaña 1: Programa 3/29

```
CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
121 PVector(v17.x+100, v17.y);
122 // ATRACTORES DE BICIS
123 PVector barrio1 = new
124     PVector(v17.x+100, v17.y);
125 PVector barrio2 = new
126     PVector(v53.x, v53.y+100);
127 PVector barrio3 = new
128     PVector(v23.x-100, v23.y);
129 // ATRACTORES DE AUTOS Y BUSES
130 PVector comuna1 = new
131     PVector(v53.x, v53.y+100);
132 PVector comuna2 = new
133     PVector(v10.x+100, v10.y);
134 PVector comuna3 = new
135     PVector(v52.x, v52.y+100);
136
137 // DISTANCIA DE ENCUENTRO
138 // (Proxémica y seguridad)
139 float mts = 20;
140 // cambió a float con slider de 5 a 20)
141
142 // INTERACCION
143 // *nace a partir de velocidad, cantidad
144 // y distancia; valores que ya están
145 // declarados en el programa
146
147 // SUPERFICIE (de Modos por Persona)
148 float srf_p = 0.5;
149 float srf_b = 1.25;
150 float srf_a = 13.8;
151 float srf_t = 0.4;
152 float srf_total = 1000; // ****
153
154 // CONTAMINACION (AIRE Y RUIDO)
155 // SLIDER: uso de energías renovables
156 // Emisión de CO2
157 float Kgs_b = 1900; //kgs anuales
158 float Kgs_a = 410; //kgs anuales
159
160 // Contaminación Acústica
```

Pestaña 1: Programa 4/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
161 float dB_b = 50; // en decibeles
162 float dB_a = 37.5; // en decibeles
163
164
165 ////////////////////////////////////////////////////
166 //////////////////////////////////////////////////// S E T U P ////////////////////////////////////////////////////
167 ////////////////////////////////////////////////////
168 public void settings() {
169     size(1250, 1000, P2D);
170     // TAMAÑO VENTANA
171 }
172
173
174 void setup() {
175     //win = new PWindow();
176     smooth();
177     frameRate(25);
178     // CARGA DE IMAGENES EN PROGRAMA
179     FONDO = loadImage("FONDO.png");
180     INTERFAZ = loadImage("INTERFAZ.png");
181     MAPA = loadImage("MAPA.png");
182     POP1 = loadImage("POP1.png");
183     POP2 = loadImage("POP2.png");
184     POP3 = loadImage("POP3.png");
185
186     // SLIDERS INTERACTIVOS
187
188     cp5 = new ControlP5(this);
189
190     // DISTANCIA
191     cp5.addSlider("mts").
192         setPosition(1080, 660).
193         setSize(125,15).
194         setRange(3, 20).
195         setColorForeground(color(150)).
196         setColorActive(color(200)).
197         setColorValue(color(0)).
198         setColorBackground(color(129,135,136)).
199         setColorLabel(color(0));
200

```

Pestaña 1: Programa 5/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
201
202 // EMISIONES
203 cp5.addSlider("Kgs_b").
204     setPosition(1080, 476).
205     setSize(125,15).
206     setRange(1, 1900).
207     setColorForeground(color(251,116,34)).
208     setColorActive(color(251,136,34)).
209     setColorValue(color(0)).
210     setColorBackground(color(127,65,26)).
211     setColorLabel(color(0));
212
213 cp5.addSlider("Kgs_a").
214     setPosition(1080, 494).
215     setSize(125,15).
216     setRange(1, 410).
217     setColorForeground(color(221,74,78)).
218     setColorActive(color(241,74,78)).
219     setColorValue(color(0)).
220     setColorBackground(color(112,44,48)).
221     setColorLabel(color(0));
222
223 // RUIDO
224 cp5.addSlider("dB_b").
225     setPosition(1080, 568).
226     setSize(125,15).
227     setRange(1, 100).
228     setColorForeground(color(251,116,34)).
229     setColorActive(color(251,136,34)).
230     setColorValue(color(0)).
231     setColorBackground(color(127,65,26)).
232     setColorLabel(color(0));
233
234 cp5.addSlider("dB_a").
235     setPosition(1080, 587).
236     setSize(125,15).
237     setRange(1, 75).
238     setColorForeground(color(221,74,78)).
239     setColorActive(color(241,74,78)).
240     setColorValue(color(0)).

```

Pestaña 1: Programa 6/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
241     setColorBackground(color(112,44,48)).
242     setColorLabel(color(0));
243
244
245 //SLIDER DE CANTIDAD
246 cp5.addSlider("Q_p").
247     setPosition(1080, 215).
248     setSize(125,15).
249     setRange(1, 10).
250     setColorForeground(color(42,249,167)).
251     setColorActive(color(42,249,187)).
252     setColorValue(color(0)).
253     setColorBackground(color(22,132,92)).
254     setColorLabel(color(0));
255
256 cp5.addSlider("Q_b").
257     setPosition(1080, 234).
258     setSize(125,15).
259     setRange(1, 10).
260     setColorForeground(color(30,178,252)).
261     setColorActive(color(30,198,252)).
262     setColorValue(color(0)).
263     setColorBackground(color(16,96,135)).
264     setColorLabel(color(0));
265
266 cp5.addSlider("Q_t").
267     setPosition(1080, 253).
268     setSize(125,15).
269     setRange(1, 10).
270     setColorForeground(color(251,116,34)).
271     setColorActive(color(251,136,34)).
272     setColorValue(color(0)).
273     setColorBackground(color(127,65,26)).
274     setColorLabel(color(0));
275
276 cp5.addSlider("Q_a").
277     setPosition(1080, 272).
278     setSize(125,15).
279     setRange(1, 10).
280     setColorForeground(color(221,74,78))

```

Pestaña 1: Programa 7/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
281 setColorActive(color(241,74,78)).
282 setColorValue(color(0)).
283 setColorBackground(color(112,44,48))
284 setColorLabel(color(0));
285
286 //SLIDER DE VELOCIDADES
287 cp5.addSlider("mseg_p").
288 setPosition(1080, 345).
289 setSize(125,15).
290
291 setRange(mseg_p/2, mseg_p*2).
292 setColorForeground(color(42,249,167)).
293 setColorActive(color(42,249,187)).
294 setColorValue(color(0)).
295 setColorBackground(color(22,132,92))
296 setColorLabel(color(0));
297
298 cp5.addSlider("mseg_c").
299 setPosition(1080, 364).
300 setSize(125,15).
301 setRange(mseg_c/2, mseg_c*2).
302 setColorForeground(color(30,178,252)).
303 setColorActive(color(30,198,252)).
304 setColorValue(color(0)).
305 setColorBackground(color(16,96,135))
306 setColorLabel(color(0));
307
308 cp5.addSlider("mseg_b").
309 setPosition(1080, 382).
310 setSize(125,15).
311 setRange(mseg_b/2, mseg_b*2).
312 setColorForeground(color(251,116,34)).
313 setColorActive(color(251,136,34)).
314 setColorValue(color(0)).
315 setColorBackground(color(127,65,26))
316 setColorLabel(color(0));
317
318 cp5.addSlider("mseg_a").
319 setPosition(1080, 401).
320 setSize(125,15).

```

Pestaña 1: Programa 8/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
321 setRange(mseg_a/2, mseg_a*2).
322 setColorForeground(color(221,74,78))
323 setColorActive(color(241,74,78)).
324 setColorValue(color(0)).
325 setColorBackground(color(112,44,48))
326 setColorLabel(color(0));
327
328
329
330
331 // CREACION DE PATHS
332 newPath1();
333 newPath2();
334 newPath3();
335 newPathb1();
336 newPathb2();
337 newPathb3();
338 newPatha1();
339 newPatha2();
340 newPatha3();
341 newPatht1();
342 //newPatht2();
343 //newPatht3();
344
345 // SETEO DE AGENTES
346 peatones = new ArrayList<Persona>();
347 peatones2 = new ArrayList<Persona>();
348 peatones3 = new ArrayList<Persona>();
349 ciclistas = new ArrayList<Bici>();
350 ciclistas2 = new ArrayList<Bici>();
351 ciclistas3 = new ArrayList<Bici>();
352 conductores = new ArrayList<Auto>();
353 conductores2 = new ArrayList<Auto>();
354 conductores3 = new ArrayList<Auto>();
355 choferes = new ArrayList<TransportePub
356 //choferes2 = new ArrayList<Transporte
357 //choferes3 = new ArrayList<Transporte
358 // Proximamente. . .
359 }
360

```

Pestaña 1: Programa 9/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
361
362 // D R A W //
363 void draw() {
364 // FONDO
365 image(FONDO, 0, 0);
366
367 // INFO COORDS MOUSE
368 println(mouseX, mouseY);
369
370 // Herramienta temporal usada para
371 // encontrar las coordenadas posición
372 // de los sliders según el fondo.jpg
373 // rect(mouseX,mouseY,100,10);
374
375 // FUNCIONES VISUALES DE LOS PATHS
376 //path1.display(0, 255, 0);
377 //path2.display(0, 255, 0);
378 //path3.display(0, 255, 0);
379 //pathb1.display(0, 0, 255);
380 //pathb2.display(0, 0, 255);
381 //pathb3.display(0, 0, 255);
382 //patha1.display(255, 0, 0);
383 //patha2.display(255, 0, 0);
384 //patha3.display(255, 0, 0);
385
386 // FUNCION AGENTE FOLLOWS PATH N°
387 for (Persona p : peatones) {
388 p.follow(pathp1);
389 }
390 for (Persona p2 : peatones2) {
391 p2.follow(pathp2);
392 }
393 for (Persona p3 : peatones3) {
394 p3.follow(pathp3);
395 }
396 for (Bici b : ciclistas) {
397 b.follow(pathb1);
398 }
399 for (Bici b2 : ciclistas2) {
400 b2.follow(pathb2);

```

Pestaña 1: Programa 10/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
401 }
402 for (Bici b3 : ciclistas3) {
403     b3.follow(pathb3);
404 }
405 for (Auto a : conductores) {
406     a.follow(patha1);
407 }
408 for (Auto a2 : conductores2) {
409     a2.follow(patha2);
410 }
411 for (Auto a3 : conductores3) {
412     a3.follow(patha3);
413 }
414 for (TransportePublico t : choferes) {
415     t.follow(patht1);
416 }
417
418 // ORIGENES DE PERSONAS, BICIS,
419 // AUTOS Y TRANSPORTE PUBLICO
420 peatones.add(new Persona(random(v34.x-
421 peatones2.add(new Persona(random(v2.x-
422 peatones3.add(new Persona(random(v50.x-
423 ciclistas.add(new Bici(random(v52.x-10
424 ciclistas2.add(new Bici(random(v22.x-10
425 ciclistas3.add(new Bici(random(v10.x-10
426 conductores.add(new Auto(random(v1.x-20
427 conductores2.add(new Auto(random(v44.x-
428 conductores3.add(new Auto(random(v23.x-
429 choferes.add(new TransportePublico(ran
430
431 // CANTIDADES DE AGENTES
432 if (peatones.size() > Q_p) {
433     peatones.remove(Q_p);
434 }
435 if (peatones2.size() > Q_p) {
436     peatones2.remove(Q_p);
437 }
438 if (peatones3.size() > Q_p) {
439     peatones3.remove(Q_p);
440 }

```

Pestaña 1: Programa 11/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
441 if (ciclistas.size() > Q_b) {
442     ciclistas.remove(Q_b);
443 }
444 if (ciclistas2.size() > Q_b) {
445     ciclistas2.remove(Q_b);
446 }
447 if (ciclistas3.size() > Q_b) {
448     ciclistas3.remove(Q_b);
449 }
450 if (conductores.size() > Q_a) {
451     conductores.remove(Q_a);
452 }
453 if (conductores2.size() > Q_a) {
454     conductores2.remove(Q_a);
455 }
456 if (conductores3.size() > Q_a) {
457     conductores3.remove(Q_a);
458 }
459 if (choferes.size() > Q_t) {
460     choferes.remove(Q_t);
461 }
462
463 // INTERACCION ENTRE AGENTES
464 // VERSIONE MUCHO MAS EFICIENTE!
465 for (int i = peatones.size()-1; i >= 0
466     for (int j = ciclistas.size()-1; j >=
467         for (int k = conductores.size()-1;
468             for (int l = choferes.size()-1;
469                 Persona p = peatones.get(i);
470                 Persona p2 = peatones2.get(i);
471                 Persona p3 = peatones3.get(i);
472                 Bici b = ciclistas.get(j);
473                 Bici b2 = ciclistas2.get(j);
474                 Bici b3 = ciclistas3.get(j);
475                 Auto a = conductores.get(k);
476                 Auto a2 = conductores2.get(k);
477                 Auto a3 = conductores3.get(k);
478                 TransportePublico t = choferes
479
480 // (B1)

```

Pestaña 1: Programa 12/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
481 if (b.choca(p) || b.choca(p2)
482 || b.choca(p3) || b.choca(t)) {
483     b.velocity.x =
484     constrain(b.velocity.x, -0.2, 0.2);
485     b.velocity.y =
486     constrain(b.velocity.y, -0.2, 0.2);
487 }
488 // (B2)
489 if (b2.choca(p) || b2.choca(p2)
490 || b2.choca(p3) || b2.choca(t)) {
491     b2.velocity.x =
492     constrain(b2.velocity.x, -0.2, 0.2);
493     b2.velocity.y =
494     constrain(b2.velocity.y, -0.2, 0.2);
495 }
496 // (B3)
497 if (b3.choca(p) || b3.choca(p2)
498 || b3.choca(p3) || b3.choca(t)) {
499     b3.velocity.x =
500     constrain(b3.velocity.x, -0.2, 0.2);
501     b3.velocity.y =
502     constrain(b3.velocity.y, -0.2, 0.2);
503 }
504 // (A1)
505 if (a.choca(p) || a.choca(p2)
506 || a.choca(p3) || a.choca(b)
507 || a.choca(b2) || a.choca(b3)
508 || b.choca(t)) {
509     a.velocity.x =
510     constrain(a.velocity.x, -0.2, 0.2);
511     a.velocity.y =
512     constrain(a.velocity.y, -0.2, 0.2);
513 }
514 // (A2)
515 if (a2.choca(p) || a2.choca(p2)
516 || a2.choca(p3) || a2.choca(b)
517 || a2.choca(b2) || a2.choca(b3)
518 || b.choca(t)) {
519     a2.velocity.x =
520     constrain(a2.velocity.x, -0.2, 0.2);

```

Pestaña 1: Programa 13/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
521 a2.velocity.y =
522   constrain(a2.velocity.y, -0.2, 0.2);
523   }
524 // (A3)
525 if (a3.choca(p) || a3.choca(p2)
526 || a3.choca(p3) || a3.choca(b)
527 || a3.choca(b2) || a3.choca(b3)
528 || b.choca(t)) {
529 a3.velocity.x =
530   constrain(a3.velocity.x, -0.2, 0.2);
531 a3.velocity.y =
532   constrain(a3.velocity.y, -0.2, 0.2);
533   }
534 }
535 }
536 }
537 }
538 }
539
540 // FUNCIONES PERSONAS (1)
541 for (int i = peatones.size()-1;
542 i >= 0; i--) {
543   // normal loop can be affected
544   Persona p = peatones.get(i);
545   p.run(cuadra1.x, cuadra1.y);
546   fill(0, 255, 0);
547   ellipse(cuadra1.x, cuadra1.y, diam, diam);
548
549   // SI SALE DE LA VENTANA MUERE
550   if (p.outOfBounds()) {
551     peatones.remove(i);
552   }
553 }
554 // FUNCION DISTANCIA PERSONA 1
555 // CON PERSONA 2 y 3
556 for (Persona p : peatones) {
557   p.separateP(peatones);
558   p.separateP(peatones2);
559   p.separateP(peatones3);
560 }

```

Pestaña 1: Programa 14/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
561
562 // FUNCIONES PERSONAS (2)
563 for (int i = peatones2.size()-1;
564 i >= 0; i--) { // normal loop can be affected
565   Persona p2 = peatones2.get(i);
566   p2.run(cuadra2.x, cuadra2.y);
567   fill(0, 255, 0);
568   // Estética del Atractor
569   // dado que está fuera de la ventana
570   // nunca se ve.
571   ellipse(cuadra2.x, cuadra2.y,
572           diam, diam);
573
574   // SI SALE DE LA VENTANA MUERE
575   if (p2.outOfBounds()) {
576     peatones2.remove(i);
577   }
578 }
579 // FUNCION DISTANCIA PERSONA 2
580 // CON PERSONAS 1 y 3
581 for (Persona p2 : peatones2) {
582   p2.separateP(peatones);
583   p2.separateP(peatones2);
584   p2.separateP(peatones3);
585 }
586 // FUNCIONES PERSONAS (3)
587 for (int i = peatones3.size()-1;
588 i >= 0; i--) {
589   // normal loop can be affected
590   Persona p3 = peatones3.get(i);
591   p3.run(cuadra3.x, cuadra3.y);
592   fill(0, 255, 0);
593   // Estetica del atractor
594   ellipse(cuadra3.x, cuadra3.y,
595           diam, diam);
596   // SI SALE DE LA VENTANA MUERE
597   if (p3.outOfBounds()) {
598     peatones3.remove(i);
599   }
600 }

```

Pestaña 1: Programa 15/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
601 // FUNCION DISTANCIA PERSONA 3
602 // CON PERSONAS 1 y 2
603 for (Persona p3 : peatones3) {
604   p3.separateP(peatones);
605   p3.separateP(peatones2);
606   p3.separateP(peatones3);
607 }
608
609 // FUNCIONES TRANSPORTE PUBLICO
610 // FUNCIONES T.PUBLICICO (1)
611 for (int i = choferes.size()-1;
612 i >= 0; i--) {
613   TransportePublico t =
614     choferes.get(i);
615   t.run(comunal.x, comunal.y);
616   fill(200, 255, 0);
617   // Estetica del atractor
618   ellipse(comunal.x, comunal.y,
619           diam, diam);
620   // SI SALE DE LA VENTANA MUERE
621   if (t.outOfBounds()) {
622     choferes.remove(i);
623   }
624 }
625 // FUNCION DISTANCIA T.PUBLICICO
626 for (TransportePublico t : choferes) {
627   t.separateP(peatones);
628   t.separateP(peatones2);
629   t.separateP(peatones3);
630   t.separateT(choferes);
631 }
632
633
634 // FUNCIONES BICIS (1)
635 for (int i = ciclistas.size()-1;
636 i >= 0; i--) {
637   // normal loop can be affected
638   Bici b = ciclistas.get(i);
639   b.run(barrio1.x, barrio1.y);
640   fill(0, 100, 255);

```

Pestaña 1: Programa 16/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
641 // Estetica del atractor
642 ellipse(barrio1.x, barrio1.y,
643         diam, diam);
644 //SI SALE DE LA VENTANA MUERE
645 if (b.outOfBounds()) {
646     ciclistas.remove(i);
647 }
648 }
649 // FUNCION DISTANCIA BICI 1
650 // CON OTRAS BICIS Y PERSONAS
651 for (Bici b : ciclistas) {
652     b.separateP(peatones);
653     b.separateP(peatones2);
654     b.separateP(peatones3);
655     b.separateB(ciclistas);
656     b.separateB(ciclistas2);
657     b.separateB(ciclistas3);
658     b.separateT(choferes);
659 }
660
661 // FUNCIONES BICIS (2)
662 for (int i = ciclistas2.size()-1;
663      i >= 0; i--) {
664     // normal loop can be affected
665     Bici b2 = ciclistas2.get(i);
666     b2.run(barrio2.x, barrio2.y);
667     fill(0, 100, 255);
668     // Estetica del atractor
669     ellipse(barrio2.x, barrio2.y,
670            diam, diam);
671     //SI SALE DE LA VENTANA MUERE
672     if (b2.outOfBounds()) {
673         ciclistas2.remove(i);
674     }
675 }
676 // FUNCION DISTANCIA BICI 2
677 // CON OTRAS BICIS Y PERSONAS
678 for (Bici b2 : ciclistas2) {
679     b2.separateP(peatones);
680     b2.separateP(peatones2);

```

Pestaña 1: Programa 17/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
681 b2.separateP(peatones3);
682 b2.separateB(ciclistas);
683 b2.separateB(ciclistas2);
684 b2.separateB(ciclistas3);
685 b2.separateT(choferes);
686 }
687
688 // FUNCIONES BICIS (3)
689 for (int i = ciclistas3.size()-1;
690      i >= 0; i--) {
691     // normal loop can be affected
692     Bici b3 = ciclistas3.get(i);
693     b3.run(barrio3.x, barrio3.y);
694     fill(0, 100, 255);
695     // Estetica del atractor
696     ellipse(barrio3.x, barrio3.y,
697            diam, diam);
698     //SI SALE DE LA VENTANA MUERE
699     if (b3.outOfBounds()) {
700         ciclistas3.remove(i);
701     }
702 }
703 // FUNCION DISTANCIA BICI 3
704 // CON OTRAS BICIS Y PERSONAS
705 for (Bici b3 : ciclistas3) {
706     b3.separateP(peatones);
707     b3.separateP(peatones2);
708     b3.separateP(peatones3);
709     b3.separateB(ciclistas);
710     b3.separateB(ciclistas2);
711     b3.separateB(ciclistas3);
712     b3.separateT(choferes);
713 }
714
715 // FUNCIONES AUTO SOLO (1)
716 for (int i = conductores.size()-1;
717      i >= 0; i--) {
718     // normal loop can be affected
719     Auto a = conductores.get(i);
720     a.run(comuna1.x, comuna1.y);

```

Pestaña 1: Programa 18/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
721 fill(255, 0, 0);
722 // Estetica del atractor
723 ellipse(comuna1.x, comuna1.y,
724         diam, diam);
725
726 // SI SALE DE LA VENTANA MUERE
727 if (a.outOfBounds()) {
728     conductores.remove(i);
729 }
730 } //(1)
731 // FUNCION DISTANCIA AUTO 1
732 // CON OTROS AUTOS, PERSONAS Y BICIS
733 for (Auto a : conductores) {
734     a.separateP(peatones);
735     a.separateP(peatones2);
736     a.separateP(peatones3);
737     a.separateB(ciclistas);
738     a.separateB(ciclistas2);
739     a.separateB(ciclistas3);
740     a.separateA(conductores);
741     a.separateA(conductores2);
742     a.separateA(conductores3);
743     a.separateT(choferes);
744 }
745
746 // FUNCIONES AUTO SOLO (2)
747 for (int i = conductores2.size()-1;
748      i >= 0; i--) {
749     // normal loop can be affected
750     Auto a2 = conductores2.get(i);
751     a2.run(comuna2.x, comuna2.y);
752     fill(255, 0, 0);
753     // Estetica atractor
754     ellipse(comuna1.x, comuna1.y,
755            diam, diam);
756
757 // SI SALE DE LA VENTANA MUERE
758 if (a2.outOfBounds()) {
759     conductores2.remove(i);
760 }

```

Pestaña 1: Programa 19/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
761 } //(2)
762 // FUNCION DISTANCIA AUTO 2
763 // CON OTROS AUTOS, PERSONAS Y BICIS
764 for (Auto a2 : conductores2) {
765     a2.separateP(peatones);
766     a2.separateP(peatones2);
767     a2.separateP(peatones3);
768     a2.separateB(ciclistas);
769     a2.separateB(ciclistas2);
770     a2.separateB(ciclistas3);
771     a2.separateA(conductores);
772     a2.separateA(conductores2);
773     a2.separateA(conductores3);
774     a2.separateT(choferes);
775 }
776
777 // FUNCIONES AUTO SOLO (3)
778 for (int i = conductores3.size()-1;
779      i >= 0; i--) {
780     // normal loop can be affected
781     Auto a3 = conductores3.get(i);
782     a3.run(comuna3.x, comuna3.y);
783     fill(255, 0, 0);
784     // Estetica del atractor
785     ellipse(comuna3.x, comuna3.y,
786            diam, diam);
787
788     // SI SALE DE LA VENTANA MUERE
789     if (a3.outOfBounds()) {
790         conductores3.remove(i);
791     }
792 } //(3)
793 // FUNCION DISTANCIA AUTO 3
794 // CON OTROS AUTOS, PERSONAS Y BICIS
795 for (Auto a3 : conductores3) {
796     a3.separateP(peatones);
797     a3.separateP(peatones2);
798     a3.separateP(peatones3);
799     a3.separateB(ciclistas);
800     a3.separateB(ciclistas2);

```

Pestaña 1: Programa 20/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
801     a3.separateB(ciclistas3);
802     a3.separateA(conductores);
803     a3.separateA(conductores2);
804     a3.separateA(conductores3);
805     a3.separateT(choferes);
806 }
807
808 // INTERFAZ
809 image(INTERFAZ, 0, 0);
810
811
812 // GRAFICO DE BARRAS INDICADORES
813 // FORMULAS EXPRESADAS EN CÓDIGO
814
815 // formula SEGURIDAD
816 fill(255);
817 float Seguridad =
818     (mseg_c*mseg_a*mseg_b)/mts;
819 // Conversion a porcentaje
820 float S = map(Seguridad, 60, 0.05,
821              0, 100);
822 //rect(79, 970, 50, -S);
823 fill(0);
824 int Si = round(S);
825 //text(Si, 79, 965);
826
827 // formula INTERACCION entre vecinos
828 fill(255);
829 float Interaccion =
830     ((Q_p-1)*(Q_p)*(mts))/(mseg_a*Q_a);
831 if (Interaccion > 100) {
832     Interaccion = 100;
833 }
834 // Conversion a porcentaje
835 float I = map(Interaccion, 0, 100,
836              0, 100);
837 //rect(240, 970, 50, -I);
838 fill(0);
839 int Ii = round(I);
840 //text(Ii, 240, 965);

```

Pestaña 1: Programa 21/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
841
842 // formula CONTAMINACION CO2 kg
843 float Polucion =
844     (Q_t*Kgs_b)+(Q_a*Kgs_a);
845 // Conversion a porcentaje
846 float P = map(Polucion, 2, 23100, 100,
847              fill(255));
848 //rect(400, 970, 50, -P);
849 fill(0);
850 int Pi = round(P);
851 //text(Pi, 400, 965);
852
853 // formula CONTAMINACION RUIDO dB
854 float Ruido = (Q_t*dB_b)+(Q_a*dB_a);
855 // Conversion a porcentaje invertido
856 float R = map(Ruido, 2, 1750, 100, 0);
857 fill(255);
858 //rect(561, 970, 50, -R);
859 fill(0);
860 int Ri = round(R);
861 //text(Ri, 561, 965);
862
863 // formula SUPERFICIE EN USO mt2
864 float Superficie =
865     (Q_t*srf_t)+(Q_a*srf_a)+
866     (Q_b*srf_b)+(Q_p*srf_p);
867 // Conversion a porcentaje
868 float Srf = map(Superficie, 15.95,
869                159.5, 0, 9.96);
870 fill(255);
871 //rect(721, 970, 50, -(100-Srf));
872 fill(0);
873 int Srfi = round(Srf);
874 //text(100-Srfi, 721, 965);
875 int Srfi2 = 100-Srfi;
876
877 // formula Mejora TOTAL
878 float Total =
879     (Si*0.2 +Ii*0.2 +Pi*0.2
880     +Ri*0.2 +Srfi2*0.2);

```

Pestaña 1: Programa 22/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
881
882 int TotalI = round(Total);
883 textSize(20);
884 fill(255-TotalI*2,2*TotalI,0);
885 text(TotalI, 1199,993);
886 // TOTAL % TASA SUSTENTABILIDAD
887 // rect(882, 970, 50, -Total);
888
889
890 // GRÁFICO DE 5 BARRAS
891 // TASA DE SUSTENTABILIDAD
892 pushMatrix();
893
894
895 translate(1126,870);
896 //ellipse(0,0, 199, 199);
897
898 pushMatrix();
899 rotate(0);
900 noStroke();
901 fill(255-2*R,2*R,0);
902 rect(-5, 5, 10, R, 5);
903 popMatrix();
904
905 pushMatrix();
906 noStroke();
907 fill(255-2*P,2*P,0);
908 rotate(5);
909 rect(-5, 5, 10, P, 5);
910 popMatrix();
911
912 pushMatrix();
913 noStroke();
914 fill(255-2*(100-Srf),2*(100-Srf),0);
915 rotate(10);
916 rect(-5, 5, 10, 100-Srf*0.8, 5);
917 popMatrix();
918
919 pushMatrix();
920 noStroke();

```

Pestaña 1: Programa 23/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
921 fill(255-2*I,2*I,0);
922 rotate(15);
923 rect(-5, 5, 10, I, 5);
924 popMatrix();
925
926 pushMatrix();
927 noStroke();
928 fill(255-2*S,2*S,0);
929 rotate(20);
930 rect(-5, 5, 10, S, 5);
931 popMatrix();
932
933 popMatrix();
934
935
936
937
938 // LATITUD LONGITUD
939 pushMatrix();
940 fill(150);
941 textSize(10);
942 text("(" + (33.423525+mouseX/100) +
943      ", "+(70.610808+mouseY/100) +
944      ")",29,978);
945 popMatrix();
946
947 // POP UPS OCULTOS
948 // INTERACCION USUARIO MOUSE
949 // MAPA ZOOM
950 if ((mouseX > 29) && (mouseX < 201) &&
951     (mouseY > 787) && (mouseY < 959)) {
952   image(MAPA, 0, 0);
953 }
954
955 // POP1
956 if ((mouseX > 24) && (mouseX < 53)&&
957     (mouseY > 126) && (mouseY < 156)) {
958   image(POP1, 0, 0);
959 }
960

```

Pestaña 1: Programa 24/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
961 // POP2
962 if ((mouseX > 950) && (mouseX < 980)&&
963     (mouseY > 26) && (mouseY < 56)) {
964   image(POP2, 0, 0);
965 }
966
967 // POP3 // TASA DE SUSTENTABILIDAD
968 // (FORMULAS)
969 if ((mouseX > 950) && (mouseX < 980)&&
970     (mouseY > 721) && (mouseY < 750)) {
971   image(POP3, 0, 0);
972 }
973 }
974
975 ///////////////////////////////////////////////////
976 /////////////////////////////////////////////////// F I N D R A W ///////////////////////////////////////////////////
977 ///////////////////////////////////////////////////
978
979 // CONSTRUCCION MANUAL DE PATHS
980 // SEPARADAS POR GRUPO DE PERSONAS
981
982 // PATHS PERSONAS
983 void newPathp1() {
984   // PATH PERSONA (1)
985   pathp1 = new Path();
986   pathp1.addPoint(v34.x, v34.y);
987   pathp1.addPoint(v39.x, v39.y);
988   pathp1.addPoint(v35.x, v35.y);
989   pathp1.addPoint(v40.x, v40.y);
990   pathp1.addPoint(v36.x, v36.y);
991   pathp1.addPoint(v41.x, v41.y);
992   pathp1.addPoint(v45.x, v45.y);
993   pathp1.addPoint(v48.x, v48.y);
994   pathp1.addPoint(v49.x, v49.y);
995   pathp1.addPoint(v53.x, v53.y);
996   pathp1.addPoint(cuadra1.x, cuadra1.y);
997 }
998
999 void newPathp2() {
1000 // PATH PERSONA (2)

```

Pestaña 1: Programa 25/29


```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
1001 pathp2 = new Path();
1002 pathp2.addPoint(v7.x, v7.y);
1003 pathp2.addPoint(v12.x, v12.y);
1004 pathp2.addPoint(v25.x, v25.y);
1005 pathp2.addPoint(v28.x, v28.y);
1006 pathp2.addPoint(v36.x, v36.y);
1007 pathp2.addPoint(v41.x, v41.y);
1008 pathp2.addPoint(v45.x, v45.y);
1009 pathp2.addPoint(v48.x, v48.y);
1010 pathp2.addPoint(v52.x, v52.y);
1011 pathp2.addPoint(cuadra2.x, cuadra2.y);
1012 }
1013 void newPath3() {
1014     // PATH PERSONA (3)
1015     pathp3 = new Path();
1016     pathp3.addPoint(v44.x, v44.y);
1017     pathp3.addPoint(v39.x, v39.y);
1018     pathp3.addPoint(v35.x, v35.y);
1019     pathp3.addPoint(v33.x, v33.y);
1020     pathp3.addPoint(v28.x, v28.y);
1021     pathp3.addPoint(v36.x, v36.y);
1022     pathp3.addPoint(v30.x, v30.y);
1023     pathp3.addPoint(v26.x, v26.y);
1024     pathp3.addPoint(v21.x, v21.y);
1025     pathp3.addPoint(v17.x, v17.y);
1026     pathp3.addPoint(cuadra3.x, cuadra3.y);
1027 }
1028
1029
1030 // PATHS BICI
1031 void newPathb1() { // (1)
1032     pathb1 = new Path();
1033     pathb1.addPoint(v52.x, v52.y);
1034     pathb1.addPoint(v41.x, v41.y);
1035     pathb1.addPoint(v36.x, v36.y);
1036     pathb1.addPoint(v30.x, v30.y);
1037     pathb1.addPoint(v26.x, v26.y);
1038     pathb1.addPoint(v21.x, v21.y);
1039     pathb1.addPoint(v17.x, v17.y);
1040     pathb1.addPoint(barrio1.x, barrio1.y);

```

Pestaña 1: Programa 26/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
1041 }
1042 void newPathb2() { // (2)
1043     pathb2 = new Path();
1044     pathb2.addPoint(v22.x, v22.y);
1045     pathb2.addPoint(v18.x, v18.y);
1046     pathb2.addPoint(v15.x, v15.y);
1047     pathb2.addPoint(v12.x, v12.y);
1048     pathb2.addPoint(v7.x, v7.y);
1049     pathb2.addPoint(v20.x, v20.y);
1050     pathb2.addPoint(v30.x, v30.y);
1051     pathb2.addPoint(v37.x, v37.y);
1052     pathb2.addPoint(v42.x, v42.y);
1053     pathb2.addPoint(v49.x, v49.y);
1054     pathb2.addPoint(v53.x, v53.y);
1055     pathb2.addPoint(barrio2.x, barrio2.y);
1056 }
1057 void newPathb3() { // (3)
1058     pathb3 = new Path();
1059     pathb3.addPoint(barrio3.x, barrio3.y);
1060     pathb3.addPoint(v23.x, v23.y);
1061     pathb3.addPoint(v35.x, v35.y);
1062     pathb3.addPoint(v33.x, v33.y);
1063     pathb3.addPoint(v29.x, v29.y);
1064     pathb3.addPoint(v28.x, v28.y);
1065     pathb3.addPoint(v25.x, v25.y);
1066     pathb3.addPoint(v20.x, v20.y);
1067     pathb3.addPoint(v16.x, v16.y);
1068     pathb3.addPoint(v14.x, v14.y);
1069     pathb3.addPoint(v10.x, v10.y);
1070 }
1071
1072 // AV. LYON
1073 // PATHS AUTO
1074 void newPatha1() {
1075     patha1 = new Path();
1076     patha1.addPoint(v1.x, v1.y);
1077     patha1.addPoint(v7.x, v7.y);
1078     patha1.addPoint(v20.x, v20.y);
1079     patha1.addPoint(v30.x, v30.y);
1080     patha1.addPoint(v37.x, v37.y);

```

Pestaña 1: Programa 27/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
1081 patha1.addPoint(v49.x, v49.y);
1082 patha1.addPoint(v53.x, v53.y);
1083 patha1.addPoint(comuna1.x, comuna1.y);
1084 }
1085
1086 // AV. PROVIDENCIA
1087 void newPatha2() { // PATH AUTO
1088     patha2 = new Path();
1089     patha2.addPoint(v44.x, v44.y);
1090     patha2.addPoint(v39.x, v39.y);
1091     patha2.addPoint(v35.x, v35.y);
1092     patha2.addPoint(v33.x, v33.y);
1093     patha2.addPoint(v29.x, v29.y);
1094     patha2.addPoint(v28.x, v28.y);
1095     patha2.addPoint(v25.x, v25.y);
1096     patha2.addPoint(v20.x, v20.y);
1097     patha2.addPoint(v16.x, v16.y);
1098     patha2.addPoint(v14.x, v14.y);
1099     patha2.addPoint(v10.x, v10.y);
1100     patha2.addPoint(comuna2.x, comuna2.y);
1101 }
1102
1103 // AUTO RESIDENTE
1104 void newPatha3() { // ZONA RESIDENTE
1105     patha3 = new Path();
1106     patha3.addPoint(v23.x, v23.y);
1107     patha3.addPoint(v35.x, v35.y);
1108     patha3.addPoint(v40.x, v40.y);
1109     patha3.addPoint(v47.x, v47.y);
1110     patha3.addPoint(v45.x, v45.y);
1111     patha3.addPoint(v48.x, v48.y);
1112     patha3.addPoint(comuna3.x, comuna3.y);
1113 }
1114
1115 // PROVIDENCIA
1116 // PATHS TRANSPORTE PUBLICO
1117 void newPatht1() {
1118     patht1 = new Path();
1119     patht1.addPoint(v44.x, v44.y);
1120     patht1.addPoint(v39.x, v39.y);

```

Pestaña 1: Programa 28/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
1106 patha3.addPoint(v23.x, v23.y);
1107 patha3.addPoint(v35.x, v35.y);
1108 patha3.addPoint(v40.x, v40.y);
1109 patha3.addPoint(v47.x, v47.y);
1110 patha3.addPoint(v45.x, v45.y);
1111 patha3.addPoint(v48.x, v48.y);
1112 patha3.addPoint(comuna3.x, comuna3.y);
1113 }
1114
1115 // PROVIDENCIA
1116 // PATHS TRANSPORTE PUBLICO
1117 void newPatht1() {
1118   patht1 = new Path();
1119   patht1.addPoint(v44.x, v44.y);
1120   patht1.addPoint(v39.x, v39.y);
1121   patht1.addPoint(v35.x, v35.y);
1122   patht1.addPoint(v33.x, v33.y);
1123   patht1.addPoint(v29.x, v29.y);
1124   patht1.addPoint(v28.x, v28.y);
1125   patht1.addPoint(v25.x, v25.y);
1126   patht1.addPoint(v20.x, v20.y);
1127   patht1.addPoint(v30.x, v30.y);
1128   patht1.addPoint(v37.x, v37.y);
1129   patht1.addPoint(v42.x, v42.y);
1130   patht1.addPoint(v49.x, v49.y);
1131   patht1.addPoint(v53.x, v53.y);
1132   patht1.addPoint(comuna1.x, comuna1.y);
1133 }
1134
1135 // FUNCION EN DESUSO
1136 public void keyPressed() {
1137   if (key == ' ') {
1138     debug = !debug;
1139   }
1140 }
1141
1142
1143
1144
1145 /// F I N ///

```

Pestaña 1: Programa 29/29

```

CODIGO_8_Usabilidad | Processing 3.3.5
1 class Auto extends Bici {
2
3   Auto(float x_, float y_) {
4     super(x_, y_);
5     velocity =
6       new PVector(random(0, 0),
7         random(0, 0));
8     diam = 3;
9     mseg_p = 4;
10    maxforce = 0.15; // 0.05
11  }
12
13  // APARIENCIA
14  void display() {
15    float theta =
16      velocity.heading() + PI/2;
17    fill(221, 74, 39); // 221, 74, 78
18    // 42,249,167
19    // Color cambiante cual bus
20    //fill(221*((Kgs_a/410+dB_a/8)/2),251
21    // (251*((Kgs_b/1000+dB_b/10)/2),116
22    noStroke();
23    pushMatrix();
24    translate(location.x, location.y);
25    rotate(theta);
26    rect(0, 0, diam, diam*3);
27    popMatrix();
28  }
29
30  // DESPLAZAMIENTO, VELOCIDAD
31  // Y ACELERACION
32  void update() {
33    velocity.add(acceleration);
34    velocity.limit(mseg_a);
35    location.add(velocity);
36    acceleration.mult(0);
37  }
38
39  // FUNCION DE FUNCIONES
40  void run(float x_, float y_) {

```

Pestaña 2: Auto 1/4

```

CODIGO_8_Usabilidad | Processing 3.3.5
41   update();
42   display();
43   seek(new PVector(x_, y_));
44   //choqueMurallas();
45   //reboteEdificio();
46 }
47
48 // INTERACCION CON PERSONA
49 boolean choca(Persona p) {
50   float d =
51     abs(dist(location.x, location.y,
52       p.location.x, p.location.y));
53   if (d <= mts) {
54     return true;
55   } else {
56     return false;
57   }
58 }
59
60 // INTERACCION CON BICI
61 boolean choca(Bici b) {
62   float d =
63     abs(dist(location.x, location.y,
64       b.location.x, b.location.y));
65   if (d <= mts) {
66     return true;
67   } else {
68     return false;
69   }
70 }
71
72 // INTERACCION CON TRANSPORTE PUBLICO
73 boolean choca(TransportePublico t) {
74   float d =
75     abs(dist(location.x, location.y,
76       t.location.x, t.location.y));
77   if (d <= mts) {
78     return true;
79   } else {
80     return false;

```

Pestaña 2: Auto 2/4

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
81 }
82 }
83
84 // SEPARACION CON AUTOS
85 void separateA (ArrayList<Auto>
86     conductores) { //No
87     float desiredseparation = diam*20;
88     PVector sum = new PVector();
89     int count = 0;
90     for (Auto other : conductores) {
91         float d =
92             PVector.dist(location,
93                 other.location);
94         if ((d > 0) &&
95             (d < desiredseparation)) {
96             PVector diff =
97                 PVector.sub(location,
98                     other.location);
99             diff.normalize(); //wha
100             diff.div(d);
101             sum.add(diff);
102             count++;
103         }
104     }
105     if (count > 0) {
106         sum.div(count);
107         sum.normalize();
108         sum.mult(mseg_a);
109         PVector steer =
110             PVector.sub(sum, velocity);
111         steer.limit(maxforce);
112         applyForce(steer);
113     }
114 }
115
116 void separateT
117 (ArrayList<TransportePublico>
118     choferes) {
119     //Note how the desired separation is
120     float desiredseparation = diam*10;

```

Pestaña 2: Auto 3/4

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
109     PVector steer =
110         PVector.sub(sum, velocity);
111     steer.limit(maxforce);
112     applyForce(steer);
113 }
114 }
115
116 void separateT
117 (ArrayList<TransportePublico>
118     choferes) {
119     //Note how the desired separation is
120     float desiredseparation = diam*10;
121     PVector sum = new PVector();
122     int count = 0;
123     for (TransportePublico other :
124         choferes) {
125         float d = PVector.dist(location,
126             other.location);
127         if ((d > 0) &&
128             (d < desiredseparation)) {
129             PVector diff =
130                 PVector.sub(location,
131                     other.location);
132             diff.normalize();
133             diff.div(d);
134             sum.add(diff);
135             count++;
136         }
137     }
138     if (count > 0) {
139         sum.div(count);
140         sum.normalize();
141         sum.mult(mseg_c);
142         PVector steer =
143             PVector.sub(sum, velocity);
144         steer.limit(maxforce);
145         applyForce(steer);
146     }
147 }
148 }

```

Pestaña 2: Auto 4/4

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
1 class Bici extends Persona {
2
3     Bici(float x_, float y_) {
4         super(x_, y_);
5         velocity =
6             new PVector(random(0, 0),
7                 random(0, 0));
8         diam = 2.5;
9         maxforce = 0.15;
10    }
11
12    // APARIENCIA
13    void display() {
14        //Vehicle is a triangle pointing in
15        float theta =
16            velocity.heading() + PI/2;
17        fill(30,178,252);
18        noStroke();
19        pushMatrix();
20        translate(location.x, location.y);
21        rotate(theta);
22        beginShape();
23        vertex(0, -diam*2);
24        vertex(-diam, diam);
25        vertex(diam, diam);
26        endShape(CLOSE);
27        popMatrix();
28    }
29
30    // DESPLAZAMIENTO,
31    // VELOCIDAD Y ACELERACION
32    void update() {
33        velocity.add(acceleration);
34        velocity.limit(mseg_c);
35        location.add(velocity);
36        acceleration.mult(0);
37    }
38
39    // FUNCION DE FUNCIONES
40    void run(float x_, float y_) {

```

Pestaña 3: Bici 1/4

```

CODIGO_8_Usabilidad | Processing 3.3.5
41 update();
42 display();
43 seek(new PVector(x_, y_));
44 //choqueMurallas();
45 //reboteEdificio();
46 }
47
48 // INTERACCION CON PERSONA
49 boolean choca(Persona p) {
50     float d =
51         abs(dist(location.x, location.y,
52                 p.location.x, p.location.y));
53     if (d <= mts) {
54         return true;
55     } else {
56         return false;
57     }
58 }
59 // INTERACCION CON TRANSPORTE PUBLICO
60 boolean choca(TransportePublico t) {
61     float d =
62         abs(dist(location.x, location.y,
63                 t.location.x, t.location.y));
64     if (d <= mts) {
65         return true;
66     } else {
67         return false;
68     }
69 }
70
71 // FUNCION SEPARACION BICI A BICI
72 void separateB (ArrayList<Bici>
73                 ciclistas) { //Nc
74     float desiredseparation = diam*20;
75     PVector sum = new PVector();
76     int count = 0;
77     for (Bici other : ciclistas) {
78         float d =
79             PVector.dist(location,
80                 other.location);

```

Pestaña 3: Bici 2/4

```

CODIGO_8_Usabilidad | Processing 3.3.5
81 if ((d > 0) &&
82     (d < desiredseparation)) {
83     PVector diff =
84         PVector.sub(location, other.location);
85     diff.normalize(); //wl
86     diff.div(d);
87     sum.add(diff);
88     count++;
89 }
90
91 if (count > 0) {
92     sum.div(count);
93     sum.normalize();
94     sum.mult(mseg_c);
95     PVector steer =
96         PVector.sub(sum, velocity);
97     steer.limit(maxforce);
98     applyForce(steer);
99 }
100
101 void separateT
102 (ArrayList<TransportePublico>
103     choferes) { //Nc
104     float desiredseparation = diam*10;
105     PVector sum = new PVector();
106     int count = 0;
107     for (TransportePublico other :
108         choferes) {
109         float d =
110             PVector.dist(location,
111                 other.location);
112         if ((d > 0) &&
113             (d < desiredseparation)) {
114             PVector diff =
115                 PVector.sub(location,
116                     other.location);
117             diff.normalize();
118             diff.div(d);
119             sum.add(diff);
120

```

Pestaña 3: Bici 3/4

```

CODIGO_8_Usabilidad | Processing 3.3.5
95     PVector steer =
96         PVector.sub(sum, velocity);
97     steer.limit(maxforce);
98     applyForce(steer);
99 }
100 }
101
102 void separateT
103 (ArrayList<TransportePublico>
104     choferes) { //Nc
105     float desiredseparation = diam*10;
106     PVector sum = new PVector();
107     int count = 0;
108     for (TransportePublico other :
109         choferes) {
110         float d =
111             PVector.dist(location,
112                 other.location);
113         if ((d > 0) &&
114             (d < desiredseparation)) {
115             PVector diff =
116                 PVector.sub(location,
117                     other.location);
118             diff.normalize();
119             diff.div(d);
120             sum.add(diff);
121             count++;
122         }
123     }
124     if (count > 0) {
125         sum.div(count);
126         sum.normalize();
127         sum.mult(mseg_b);
128         PVector steer =
129             PVector.sub(sum, velocity);
130         steer.limit(maxforce);
131         applyForce(steer);
132     }
133 }
134 }

```

Pestaña 3: Bici 4/4

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
1 // Créditos a Daniel Shiffman
2 // Y su libro The Nature of Code
3 // http://natureofcode.com
4
5 // Path Following
6 // Basado en las investigaciones
7 // de Craig Reynolds
8
9 class Path {
10
11 // A Path is an arraylist of points
12 ArrayList<PVector> points;
13 // A path has a radius,
14 // i.e how far is it ok for the
15 // boid to wander off
16 float radius;
17
18 Path() {
19 // Arbitrary radius of 12 meters
20 radius = 12;
21 points = new ArrayList<PVector>();
22 }
23
24 // Add a point to the path
25 void addPoint(float x, float y) {
26 PVector point = new PVector(x, y);
27 points.add(point);
28 }
29
30 PVector getStart() {
31 return points.get(0);
32 }
33
34 PVector getEnd() {
35 return points.get(points.size()-1);
36 }
37
38 // Draw the path
39 void display(
40

```

Pestaña 4: Path 1/2

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
28 }
29
30 PVector getStart() {
31 return points.get(0);
32 }
33
34 PVector getEnd() {
35 return points.get(points.size()-1);
36 }
37
38 // Draw the path
39 void display(
40 color r_,
41 color g_,
42 color b_) {
43 // Draw thick line for radius
44 color r = r_;
45 color g = g_;
46 color b = b_;
47 stroke(r,g,b, 40);
48 strokeWeight(radius*2);
49 noFill();
50 beginShape();
51 for (PVector v : points) {
52 vertex(v.x, v.y);
53 }
54 endShape();
55 // Draw thin line for center of
56 // the path
57 stroke(0, 50);
58 strokeWeight(1);
59 noFill();
60 beginShape();
61 for (PVector v : points) {
62 vertex(v.x, v.y);
63 }
64 endShape();
65 }
66 }
67 }

```

Pestaña 4: Path 1/2

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
1 class Persona {
2
3 PVector location;
4 PVector velocity;
5 PVector acceleration;
6 float diam;
7 float maxforce;
8
9 //float mass;
10
11 // POSICION DE ORIGEN, VELOCIDAD
12 // Y LIMITE DE VELOCIDAD
13 Persona(float x_, float y_) {
14 acceleration = new PVector(0, 0);
15 velocity = new PVector(0,0);
16 location = new PVector(x_, y_);
17 diam = 3;
18 maxforce = 0.15; // 0.01
19 mseg_p = 1.5;
20 //mass = 100;
21 }
22
23 // DESPLAZAMIENTO,
24 // VELOCIDAD Y ACELERACION
25 void update() {
26 velocity.add(acceleration);
27 velocity.limit(mseg_p/5);
28 location.add(velocity);
29 acceleration.mult(0);
30 }
31
32 // APLICADOR DE FUERZA
33 // Newton's second law; we could
34 // divide by mass if we wanted.
35 void applyForce(PVector force) {
36 acceleration.add(force);
37 }
38
39 // FUNCION DE SEGUIMIENTO DE ATRACTOR
40 void seek(PVector target) {

```

Pestaña 5: Persona 1/6

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
41 PVector desired =
42   PVector.sub(target, location);
43 desired.normalize();
44 desired.mult(mseg_p);
45 PVector steer =
46   PVector.sub(desired, velocity);
47 steer.limit(maxforce);
48 applyForce(steer);
49 }
50
51 // FUNCION DE SEGUIMIENTO DE CALLES
52 void seekPath(PVector target) {
53   PVector desired =
54     PVector.sub(target, location);
55   desired.normalize();
56   desired.mult(mseg_p);
57   PVector steer =
58     PVector.sub(desired, velocity);
59   steer.limit(maxforce);
60   steer.x = steer.x*3;
61   steer.y = steer.y*3;
62   applyForce(steer);
63 }
64
65 // FUNCION DE FUNCIONES
66 void run(float x_, float y_) {
67   update();
68   display();
69   seek(new PVector(x_, y_));
70   //choqueMurallas();
71   //reboteEdificio();
72 }
73
74 // APARIENCIA
75 void display() {
76   //Vehicle is a triangle pointing
77   // in the direction of velocity;
78   // since it is drawn pointing up,
79   // we rotate it an additional 90°
80   float theta =

```

Pestaña 5: Persona 2/6

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
81   velocity.heading() + PI/2;
82   fill(42,249,167);
83   stroke(42,249,167);
84   strokeWeight(1);
85   pushMatrix();
86   translate(location.x, location.y);
87   rotate(theta);
88   ellipse(0, 0, diam, diam/2);
89   popMatrix();
90 }
91
92 // SI SALE DE LA VENTANA MUERE
93 boolean outOfBounds() {
94   if ((location.x > 1000 + 15)
95       || (location.x < 0 - 15)
96       || (location.y > height + 15)
97       || (location.y < 0 - 30)) {
98     return true;
99   } else {
100     return false;
101   }
102 }
103
104
105 // FUNCION SEPARACION ENTRE PERSONAS
106 void separateP (ArrayList<Persona>
107   peatones) {
108   //Note how the desired separation
109   // is based on the Vehicle's size.
110   float desiredseparation = diam*3;
111   PVector sum = new PVector();
112   int count = 0;
113   for (Persona other : peatones) {
114     float d =
115       PVector.dist(location, other.location)
116     if ((d > 0) &&
117         (d < desiredseparation)) {
118       PVector diff =
119         PVector.sub(location,
120           other.location);

```

Pestaña 5: Persona 3/6

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
121   diff.normalize();
122   //What is the magnitude of the
123   // PVector pointing away from
124   // the other vehicle? The closer
125   // it is, the more we should flee
126   // The farther, the less. So we
127   // divide by the distance to
128   // weight it appropriately.
129   diff.div(d);
130   sum.add(diff);
131   count++;
132 }
133
134 if (count > 0) {
135   sum.div(count);
136   sum.normalize();
137   sum.mult(mseg_p);
138   PVector steer =
139     PVector.sub(sum, velocity);
140   steer.limit(maxforce);
141   applyForce(steer);
142 }
143 }
144
145 void follow(Path p) {
146
147   PVector predict = velocity.get();
148   predict.normalize();
149   predict.mult(50);
150   PVector predictpos =
151     PVector.add(location, predict);
152   PVector normal = null;
153   PVector target = null;
154   float worldRecord = 1000000;
155   // Start with a very high record
156   // distance that can easily be
157   // beaten
158
159   for (int i = 0;
160     i < p.points.size()-1; i++) {

```

Pestaña 5: Persona 4/6

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
161
162   PVector a = p.points.get(i);
163   PVector b = p.points.get(i+1);
164
165   PVector normalPoint =
166   getNormalPoint(predictpos, a, b);
167   if (normalPoint.x < a.x ||
168       normalPoint.x > b.x) {
169       normalPoint = b.get();
170   }
171   float distance =
172   PVector.dist(predictpos,
173               normalPoint);
174   // How far away are we from the
175   // path's center?
176   if (distance < worldRecord) {
177       worldRecord = distance;
178       normal = normalPoint;
179   }
180   PVector dir = PVector.sub(b, a);
181   dir.normalize();
182   dir.mult(10);
183   target = normalPoint.get();
184   target.add(dir);
185   }
186   }
187   if (worldRecord > p.radius) {
188       seekPath(target);
189   }
190   }
191   // A function to get the normal
192   // point from a point (p) to a line
193   // segment (a-b)
194   // This function could be optimized
195   // to make fewer new Vector objects
196   PVector getNormalPoint(
197   PVector p, PVector a, PVector b) {
198   // Vector from a to p
199   PVector ap = PVector.sub(p, a);
200   // Vector from a to b

```

Pestaña 5: Persona 5/6

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
185   }
186   }
187   if (worldRecord > p.radius) {
188       seekPath(target);
189   }
190   }
191   // A function to get the normal
192   // point from a point (p) to a line
193   // segment (a-b)
194   // This function could be optimized
195   // to make fewer new Vector objects
196   PVector getNormalPoint(
197   PVector p, PVector a, PVector b) {
198   // Vector from a to p
199   PVector ap = PVector.sub(p, a);
200   // Vector from a to b
201   PVector ab = PVector.sub(b, a);
202   ab.normalize();
203   // Normalize the line
204   // Project vector "diff" onto
205   // line by using the dot product
206   ab.mult(ap.dot(ab));
207   PVector normalPoint =
208   PVector.add(a, ab);
209   return normalPoint;
210   }
211   }
212   }
213   }
214   }
215   }
216   }
217   }
218   }
219   // B O I D S H A P E : )
220   //beginShape();
221   //vertex(0, -diam*2);
222   //vertex(-diam, diam*2);
223   //vertex(diam, diam*2);
224   //endShape(CLOSE);

```

Pestaña 5: Persona 6/6

```

CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
1 class TransportePublico extends Persona {
2
3   TransportePublico(float x_, float y_){
4     super(x_, y_);
5     velocity =
6     new PVector(random(0,0),random(0,0));
7     diam = 5;
8     maxforce = 0.15;
9
10  }
11
12  // APARIENCIA
13  void display() {
14    // Vehicle is a triangle pointing in
15    // the direction of velocity; since
16    // it is drawn pointing up, we
17    // rotate it an additional 90°
18    float theta =
19    velocity.heading() + PI/2;
20    // Este fill controla el color
21    // del bus segun cuan contaminante
22    // es para la ciudad
23    fill(251*((Kgs_b/1000+dB_b/10)/2),
24         116,34);
25
26    noStroke();
27    pushMatrix();
28    translate(location.x, location.y);
29    rotate(theta);
30    rect(0,0,diam,diam*3,2);
31    popMatrix();
32  }
33
34  // DESPLAZAMIENTO,
35  // VELOCIDAD Y ACELERACION
36  void update() {
37    velocity.add(acceleration);
38    velocity.limit(mseg_b);
39    location.add(velocity);
40    acceleration.mult(0);

```

Pestaña 6: T. Publico 1/3

```
CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
41 // FUNCION DE FUNCIONES
42 void run(float x_, float y_) {
43   update();
44   display();
45   seek(new PVector(x_, y_));
46 }
47
48 // INTERACCION CON PERSONA
49 boolean choca(Persona p) {
50   float d =
51     abs(dist(location.x, location.y,
52       p.location.x, p.location.y));
53   if (d <= mts) {
54     return true;
55   } else {
56     return false;
57   }
58 }
59
60 void separateT
61 (ArrayList<TransportePublico>
62   choferes) {
63   //Note how the desired separation
64   // is based on the Vehicle's size.
65   float desiredseparation = diam*4;
66   PVector sum = new PVector();
67   int count = 0;
68   for (TransportePublico other :
69     choferes) {
70     float d =
71       PVector.dist(location,
72         other.location);
73     if ((d > 0) &&
74       (d < desiredseparation)) {
75       PVector diff =
76         PVector.sub(location,
77           other.location);
78       diff.normalize();
79       diff.div(d);
80       sum.add(diff);
81     }
82   }
83   if (count > 0) {
84     sum.div(count);
85     sum.normalize();
86     sum.mult(mseg_b);
87     PVector steer =
88       PVector.sub(sum, velocity);
89     steer.limit(maxforce);
90     applyForce(steer);
91   }
92 }
93 }
```

Pestaña 6: T. Publico 2/3

```
CODIGO_8_Usabilidad | Processing 3.3.5
CODIGO_8_Usabilidad
54   if (d <= mts) {
55     return true;
56   } else {
57     return false;
58   }
59 }
60 void separateT
61 (ArrayList<TransportePublico>
62   choferes) {
63   //Note how the desired separation
64   // is based on the Vehicle's size.
65   float desiredseparation = diam*4;
66   PVector sum = new PVector();
67   int count = 0;
68   for (TransportePublico other :
69     choferes) {
70     float d =
71       PVector.dist(location,
72         other.location);
73     if ((d > 0) &&
74       (d < desiredseparation)) {
75       PVector diff =
76         PVector.sub(location,
77           other.location);
78       diff.normalize();
79       diff.div(d);
80       sum.add(diff);
81       count++;
82     } }
83   if (count > 0) {
84     sum.div(count);
85     sum.normalize();
86     sum.mult(mseg_b);
87     PVector steer =
88       PVector.sub(sum, velocity);
89     steer.limit(maxforce);
90     applyForce(steer);
91   }
92 }
93 }
```

Pestaña 6: T. Publico 3/3

10 ANEXOS

Por una sensación de culpa y remordimiento por lo largo de las tesis se ha decidido no agregar las transcripciones de las entrevistas para salvar hojas.

Las tres entrevistas hechas con los expertos se pueden encontrar digitalmente en el siguiente enlace de SoundCloud:

<https://soundcloud.com/andres-armstrong-cruz/sets/entrevistas-proyecto-de-titulacion-simulador-de-transporte-andres-armstrong-cruz>



11 REFERENCIAS Y BIBLIOGRAFÍA

A

ABC Motor (2016). La bicicleta es el vehículo más eficiente en los trayectos inferiores a 5 kilómetros. [online] ABC.es. Disponible en: http://www.abc.es/motor/reportajes/abci-bicicleta-vehiculo-mas-e-eficiente-trayectos-inferiores-5-kilometros-201609151756_noticia.html [Revisada el 26 Jun. 2017].

Aimsun. (2012). Cite a Website - Cite This For Me. Aimsun.com. Retrieved 1 December 2017, from <https://www.aimsun.com/wp/wp-content/uploads/2012/04/Modelled-intersection-with-real-TLC-connected.png>

Allen, J. (2008). Muenster road space poster — check the numbers!. [online] John-s-allen.com. Disponible en: <http://john-s-allen.com/blog/?p=7> [Revisada el 26 Jun. 2017].

AmegoEV. (2017). Electric Bikes, Electric Scooters & Electric Bicycles. Amegoev.com. Retrieved 8 November 2017, from https://www.amegoev.com/electric_bike_and_scooter_benefits

Amelie, P. (2013). Cities with the most % of public green space (parks and gardens). Skyscrapercity.com. Recuperado el 26 de Junio 2017, de <http://www.skyscraper-city.com/showthread.php?t=1660203>

Berkum, A. (2017). How future transport will impact society. Arjen van Berkum | Blog. Retrieved 4 December 2017, from <http://www.arjenvanberkum.nl/innovation/transport-of-the-future/>

ASIRT (2017). Road Crash Statistics. [onli-

ne] Asirt.org. Disponible en: <http://asirt.org/initiatives/informing-road-users/road-safety-facts/road-crash-statistics> [Revisada el 26 Jun. 2017].

Auto Insurance Center (2017). Top 20 Pros and Cons Associated With Self-Driving Cars. [online] Autoinsurancecenter.com. Disponible en: <http://www.autoinsurancecenter.com/top-20-pros-and-cons-associated-with-self-driving-cars.htm> [junio 2017].

Australian Government (2016). The Home of Future Transport. Future Transport. Retrieved 4 December 2017, from <https://future.transport.nsw.gov.au/>

Auto-ABC (2003). Hyundai Accent Sedan 2003 - 2005 reviews, technical data, prices. [online] Auto-abc.eu. Disponible en: <http://www.auto-abc.eu/Hyundai-Accent/g1408-2003> [Revisada el 26 Jun. 2017].

B

Badger, E. (2013). How Traffic Congestion Affects Economic Growth. CityLab. Recuperado el 26 de Junio 2017, de <https://www.citylab.com/transportation/2013/10/how-traffic-congestion-affects-economic-growth/7310/>

Barcelo, J., Ferrer, J. and Martin, R. (1999) "Simulation Assisted Design and Assessment of Vehicle Guidance Systems" International Transactions on Operations Research, Vol. 6, No. 1, pp. 123-143.

Batra, A. (2014). Community Benefits of

Green Areas & Parks. LinkedIn. Recuperado el 26 de Junio 2017, de <https://www.linkedin.com/pulse/20140922075014-150737338-architecture-and-urban-planning>

Bartz, D. (2009). Autonomous Cars Will Make Us Safer | WIRED. [online] Wired.com. Disponible en: <https://www.wired.com/2009/11/autonomous-cars/> [Revisado el 27 Jun. 2017].

Bosch. (2016). Bosch Mobility Solutions. Products. products.bosch-mobility-solutions.com. Recuperado el 26 de Junio 2017, de <http://products.bosch-mobility-solutions.com> Busandcoach. (s.f.). Buses and coaches for safer mobility!. [online] Available at: http://www.busandcoach.travel/en/bptest/buses_and_coaches_for_safer_mobility.htm [Accessed 29 Jun. 2017].

Business Wire (2016). Iteris Launches PedTrax Computer Vision for Smarter Crosswalks. [online] Finance.yahoo.com. Available at: <https://finance.yahoo.com/news/iteris-launches-pedtrax-computer-vision-150000826.html> [Accessed 27 Jun. 2017].

C

CBS DC. (2013). Study: Self-Driving Cars Would Eliminate Majority Of Traffic Deaths, Congestion. Washington.cbslocal.com. Recuperado el 26 de Junio 2017, de <http://washington.cbslocal.com/2013/10/23/study-self-driving-cars-would-eliminate-majority->

[ty-of-traffic-deaths-congestion/](http://washington.cbslocal.com/2013/10/23/study-self-driving-cars-would-eliminate-majority-ty-of-traffic-deaths-congestion/)
Centers for Disease Control and Prevention (2017). Teen Drivers: Get the Facts | Motor Vehicle Safety | CDC Injury Center. [online] Cdc.gov. Disponible en: https://www.cdc.gov/motorvehiclesafety/teen_drivers/teendrivers_factsheet.html [Revisada el 26 Jun. 2017].

Ciancio, A. (2008). Milan drivers face trend-setting pollution charge. [online] Reuters UK. Available at: <http://uk.reuters.com/article/autos-milan-idUKNOA22930620080102?pageNumber=1&virtualBrandChannel=0> [Accessed 27 Jun. 2017].

Coznect (2017). Calculation of emissions. [online] Coznect.org. Disponible en: http://www.coznect.org/help_sheets/?op_id=602&opt_id=98 [Revisada el 26 Jun. 2017].

CodeXVerde. (2016). Gobierno anuncia fecha de la Fórmula-E en Chile - CodeXVerde. CodeXVerde. Retrieved 1 December 2017, from <http://codexverde.cl/gobierno-anuncia-fecha-de-la-formula-e-en-chile/>

Conly. (2013). Is 30% of traffic actually searching for parking?. Reinventingparking.org. Retrieved 27 June 2017, from <http://www.reinventingparking.org/2013/10/is-30-of-traffic-actually-searching-for.html>

Collaborative innovation network. (2017). En.Wikipedia.org. Recuperado el 26 de Junio 2017, de [https://en.wikipedia.org/wiki/Collaborative_innovation_net-](https://en.wikipedia.org/wiki/Collaborative_innovation_network#-)

[cite_note-1-3](https://en.wikipedia.org/wiki/Collaborative_innovation_network#-cite_note-1-3)
Cox, W. (2009). Traffic Congestion, Time, Money & Productivity. Newgeography.com. Recuperado el 26 de Junio 2017, de <http://www.newgeography.com/content/001044-traffic-congestion-time-money-productivity>

CTL del MIT. (2017). Self-Driving Cars Could Be Boon for Aged, After Initial Hurdles | Center for Transportation and Logistics. ctl.mit.edu. Recuperado el 26 de Junio 2017, de <http://ctl.mit.edu/news/self-driving-cars-could-be-boon-aged-after-initial-hurdles>

D

Della Cava, M. (2016). LA to SF in 30 min: the hyperloop wars are on. USA TODAY. Recuperado el 26 de Junio 2017, de <https://www.usatoday.com/story/tech/news/2016/05/09/la-sf-30-min-hyperloop-wars/84137224/>

DesarrolloSustentable.co. (2013). ¿Qué es el desarrollo sustentable?. DesarrolloSustentable.co. Retrieved 1 December 2017, from <http://www.desarrollosustentable.co/2013/04/que-es-el-desarrollo-sustentable.html>

Downs, A. (2004). Traffic: Why It's Getting Worse, What Government Can Do | Brookings Institution. Brookings. Recuperado el 26 de Junio 2017, de <https://www.brookings.edu/research/traffic-why-its-getting-worse-what-government-can-do/>

Drummond, K., Hoel, L., and Miller, J. (2002) "Using Simulation to Predict Safety and Operational Impacts of Increasing Traffic Signal Density," *Transportation Research Record*, No. 1784, Transportation Research Board, Washington, DC.

Dutton, J. (2014). Google's Self-Driving Cars Won't Work In Heavy Rain Or On Most Roads. *Business Insider*. Recuperado el 26 de Junio 2017, de <http://www.businessinsider.com/googles-self-driving-cars-dont-work-in-rain-or-on-roads-2014-9>

E

Ecoosfera. (2014). Las bicicletas como el medio de transporte más eficiente de todos - Ecoosfera. *Ecoosfera*. Recuperado el 26 de Junio 2017, de <http://ecoosfera.com/2014/11/las-bicicletas-como-el-medio-de-transporte-mas-eficiente-de-todos/>

Edmonton Trolley Coalition (2016). Edmonton Trolley Coalition. [online] [trolleycoalition.org](http://www.trolleycoalition.org). Disponible en: <http://www.trolleycoalition.org/noise.html> [Revisada el 26 Jun. 2017].

Elswick, F. (2016). How Much Does It Cost To Build A Mile Of Road?. *Blog.midwestind.com*. Recuperado el 26 de Junio 2017, de <http://blog.midwestind.com/cost-of-building-road/>

Eltis.org. (2015). Area C in Milan: from pollution charge to congestion charge (Italy) | Eltis. [online] Available at: <http://www.eltis.org/discover/case-studies/area-c-milano-pollution-charge-congestion-charge-italy> [Accessed 27 Jun. 2017].

F

FIAFormulaE. (2017). Formula E set for Santiago's streets - Formula E. *Fiaformulae.com*. Retrieved 1 December 2017, from <http://www.fiaformulae.com/en/news/2017/october/formula-e-set-for-santiagos-streets/>

Financial Times (s.f.) Secrets of the sidewalk — what pavements say about a city. *Ft.com*. Recuperado el 26 de Junio 2017, de <https://www.ft.com/content/83bf6932-6f9c-11e6-a0c9-1365ce54b926>

Friedman, Y. (1977). *Utopías realizables* (1st ed.). Barcelona: Gili.

Fuller, J. (2017). How the Reactable Works. [online] *HowStu Works*. Available at: <http://electronics.how-stu-works.com/gadgets/audio-music/reactable.htm> [Accessed 27 Jun. 2017].

Fundación Mi Parque. (2014). ¿Cuál es el costo de mantención de áreas verdes?. *Fundación Mi Parque*. Recuperado el 26 de Junio 2017, de <http://www.miparque.cl/cuales-el-costo-de-mantenion-de-areas-verdes/>

G

Gallagher, D. (2017). *Transportation Systems - MIT Portugal*. MITportugal.org. Recuperado el 26 de Junio 2017, de <http://www.mitportugal.org/about/overview/>

transportation-systems

García, C., Carrasco, J., & Rojas, C. (2014). El contexto urbano y las interacciones sociales: dualidad del espacio de actividades de sectores de ingresos altos y bajos en Concepción, Chile (40th ed.). Santiago: EURE. Retrieved from http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0250-71612014000300004

Gardner, C. (2011). *We Are the 25%: Looking at Street Area Percentages and Surface Parking*. *Oldurbanist.blogspot.cl*. Recuperado el 26 de Junio 2017, de <http://oldurbanist.blogspot.cl/2011/12/we-are-25-looking-at-street-area.html>

Geddes and Bel (1893). *Magic motorways: Geddes, Norman Bel, 1893-1958: Free Download & Streaming: Internet Archive*. [online] *Internet Archive*. Disponible en: <https://archive.org/details/magic-motorways00geddrich> [Revisado el 27 Jun. 2017].

Ghadge, D. (2015). *Traffic congestion and its impact on employees work life*. *Linkedin*. Recuperado el 26 de Junio 2017, de <https://www.linkedin.com/pulse/traffic-congestion-its-impact-employees-work-life-deepak-ghadge>

Gloor, P. (2006). *Swarm creativity* (1st ed.). Oxford [etc.]: Oxford University Press.

Google. (2017). *A new way forward for mobility - Waymo*. *Waymo*. Recuperado el 26 de Junio 2017, de <https://www.google.com/selfdrivingcar/faq/>

Google Support. (2017). ¿Cómo funciona Waze? - Ayuda de Waze. Support. google.com. Retrieved 1 December 2017, from <https://support.google.com/waze/answer/6078702?hl=es>

H

Hall, E. (2017). Self-Driving Cars: Can We Really Trust em?. Esurance On. Recuperado el 26 de Junio 2017, de http://blog.esurance.com/self-driving-cars-can-we-really-trust-them/#.VGNVTPnF_I

Hendry, R., Salehi, M., Kingsbury, D., & Ortiz, B. (2003). Is there a rough dollar gure for parking space costs? - Knowledge base for National TDM and Tele- work Clearinghouse and Best Workplaces for Commut- ers. Usf-cutr.custhelp.com. Recuperado el 26 de Junio 2017, de http://usf-cutr.custhelp.com/app/answers/detail/a_id/2161/~is-there-a-rough-dollar-gure-for-parking-space-costs%3F

Hern, A. (2014). Self-driving cars irresistible to hackers, warns security executive. the Guardian. Recuperado el 26 de Junio 2017, de <https://www.theguardian.com/technology/2014/jan/28/self-driving-cars-irresistible-hackers-security-executive>

HeyPoorPlayer.com. (2017). Cite a Website - Cite This For Me. Heypoorplayer.com. Retrieved 1 December 2017, from https://www.heypooplayer.com/wp-content/uploads/2017/05/20170511202044_I

-eI4947384666II.jpg

Hobson, B. (2017). Carlo Ratti aims to save energy with personalised heating and cooling. [online] Dezeen. Available at: https://www.dezeen.com/2017/05/11/video-carlo-ratti-reduce-energy-use-personalised-heat-ing-cooling-buildings-movie/?li_source=LI&li_medi-um=recommended_movies_block [Accessed 27 Jun. 2017].

I

IAC Library (2017). Comparative Examples of Noise Levels | Industrial Noise Control. [online] Industrial-noisecontrol.com. Disponible en: <http://www.industrial-noisecontrol.com/comparative-noise-examples.htm> [Revisada el 26 Jun. 2017].

J

Kahane C.J., National Highway Traffic Safety Administration (2013). Injury Vulnerability and Effectiveness of Occupant Protection Technologies for Older Occupants and Women. Estados Unidos.p. 216. 2013. [cited 2015 Oct 9] Recuperado de <https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811766>

Kendall, G. (2017). UPS drivers don't turn le —and it saves them 10 million gallons of gas a year. Quartz. Re- cuperado el 26 de Junio 2017, de <https://qz.com/895691/ups-drivers-dont-turn-le-and-it-saves-them-10-million-gallons-of-gas-a-year/>

Kloub, M. (2016). Not an easy job, but Amman's tra c headache curable — experts. Jordan Times. Recuperado el 26 de Junio 2017, de <http://jordantimes.com/news/local/not-easy-job-amman%E2%80%99s-tra-c-headache-curable-%E2%80%94-experts>

Kopyto, V. (2014). Hot commodity: Self-driving car permits in California. Fortune.com. Recuperado el 26 de Junio 2017, de <http://fortune.com/2014/11/11/cali-for-nia-permits-autonomous-vehicles/>

L

Law O ces of Michael Pines (2017). Top 25 Causes of Car Accidents. [online] Law O ces of Michael Pines, APC. Disponible en: <https://seriousaccidents.com/le-gal-advice/top-causes-of-car-accidents/> [Revisada el 26 Jun. 2017].

Lohmann, R. (2009). Park shuttle automated driverless vehicle pilot project - Netherlands. [online] Faculty. washington.edu. Available at: <http://faculty.washington.edu/jbs/itrans/parkshut.htm> [Accessed 27 Jun. 2017].

M

MctTrans.ce.ufl.edu. (s.f) Corsim Traffic Software - Image Mag. Imagemag.ru. Retrieved 4 December 2017, from https://imagemag.ru/img-ba_corsim-traffic-software.html

Madrid, E. (2016). La bici, el medio más eficiente para trayectos de hasta cinco kilómetros. [online] *Heraldo. es*. Disponible en: <http://www.heraldo.es/noticias/sociedad/2016/09/15/la-bici-medio-mas-eficiente-para-trayectos-hasta-kilometros-1061336-310.html> [Revisada el 26 Jun. 2017].

Martabit, P. (2016). Parque automotor se septuplica en 15 años: llega hasta los 7,3 millones de vehículos. *Emol*. Recuperado el 26 de Junio 2017, de <http://www.emol.com/noticias/Economia/2016/03/15/793101/Parque-automotor-se-septuplica-15-anos-llego-hasta-los-73-mil-lones-vehiculos-en-2015.html>

Mercedes-Benz USA. (2017). DIS-TRONIC PLUS®. Mercedes-Benz USA. Recuperado el 26 de Junio 2017, de <https://www.mbusa.com/mercedes/owners/videos/detail/videoId-6f31735fd7cee310VgnVCM1000007c184335RCRD>

Ministerio de Transportes y Telecomunicaciones (2015). Presentamos resultados de la Encuesta Origen Destino de Santiago. Santiago, Chile, p.1. Recuperado de <http://www.mtt.gob.cl/archivos/10194>

Ministerio del Medio Ambiente. (2016). La bicicleta fue el medio más rápido de movilización en la Medición de E ciencia de Modos de Transporte 2016 (p. 1). Santiago, Chile: MMA. Recuperado de <http://portal.mma.gob.cl/la-bicicleta-fue-el-medio-mas-rapido-de-moviliza->

[cion-en-la-medicion-de-e-ciencia-de-modos-de-trans-porte-2016/](http://portal.mma.gob.cl/cion-en-la-medicion-de-e-ciencia-de-modos-de-trans-porte-2016/)

Ministerio del Medio Ambiente. (2016). *Prevén auge de vehículos eléctricos, bicicletas y un retorno al transporte público (p.1)*. Santiago, Chile: El Mercurio. Recuperado de <http://portal.mma.gob.cl/preven-auge-de-vehiculos-electricos-bicicletas-y-un-retorno-al-trans-porte-publico/>

MobilityAuthority. (2011). AECOM Final Presentation for the Green Mobility Challenge. Slideshare. net. Recuperado el 26 de Junio 2017, de <https://www.slideshare.net/MobilityAuthority/aecom-nal-presentation-for-the-green-mobility-challenge>

Mount Royal. (2013). Speed & Pedestrian Death or Injury - Mount Royal Community Association. Mount Royal Community Association. Retrieved 1 December 2017, from <http://mountroyalstation.ca/events/traffic/speed-pedestrian-death-or-injury/>

N

NewScientist. (2014). Swarm of 1024 robots forms shapes on its own. YouTube. Retrieved 27 June 2017, from <https://www.youtube.com/watch?v=ZDIH-70aQc7U>

NoCreasNada. (2016). ¿Cómo funciona Blockchain? La guía definitiva - NoCreasNada. NoCreasNada. Retrieved 1 December 2017, from <https://www.nocreasnada.com/como-funciona-blockchain/>

Note, T. (2016). *Ámsterdam prueba el primer bus urbano que reduce emisiones y*

recorre las calles sin conductor. eNote.cl. Recuperado el 26 de Junio 2017, de <http://www.thenote.cl/category/amsterdam-prueba-el-primer-bus-urbano-que-reduce-emisiones-y-recorre-las-calles-sin-conductor/>

Note, T. (2016). *Aprueban plan para bajar emisiones en Valdivia | e Note.* eNote.cl. Recuperado el 26 de Junio 2017, de <http://www.thenote.cl/category/aprueban-plan-para-bajar-emisiones-en-valdivia/>

Note, T. (2016). *El avanzado sistema de buses eléctricos de Barcelona: se cargan en cinco minutos y funcionan 14 horas.* eNote.cl. Recuperado el 26 de Junio 2017, de <http://www.thenote.cl/category/el-avanzado-sistema-de-buses-electricos-de-barcelona-se-cargan-en-cinco-minutos-y-funcionan-14-horas/>

Note, T. (2016). *Grandes capitales del mundo buscan erradicar el uso de vehículos diesel desde 2025.* e-Note.cl. Recuperado el 26 de Junio 2017, de <http://www.thenote.cl/category/grandes-capitales-del-mundo-buscan-erradicar-el-uso-de-vehiculos-diesel-desde-2025/>

Note, T. (2016). *Holanda prohibirá en 2025 los automóviles que ocupen gasolina y diésel.* eNote.cl. Recuperado el 26 de Junio 2017, de <http://www.thenote.cl/category/holanda-prohibira-en-2025-los-automoviles-que-ocupen-gasolina-y-diesel/>

Note, T. (2016). *La apuesta de Enersis Chile por la movilidad sustentable.* eNote.cl. Recuperado el 26 de Junio 2017, de <http://www.>

thenote.cl/category/la-apues-ta-de-ener-sis-chile-por-la-movilidad-sustentable/

Note, T. (2016). Santiago gana el premio internacional de Transporte Sustentable 2017. eNote.cl. Recuperado el 26 de Junio 2017, de <http://www.thenote.cl/category/santiago-gana-el-premio-internacional-de-transporte-sustentable-2017/>

Note, T. (2016). Sepa cuáles son las ciudades más contaminadas del mundo. eNote.cl. Recuperado el 26 de Junio 2017, de <http://www.thenote.cl/category/sepa-cuales-son-las-ciudades-mas-contaminadas-del-mundo-2/>

Note, T. (2017). 'Harry', el pequeño autobús público y autónomo que comenzará a operar en Londres. eNote.cl. Recuperado el 26 de Junio 2017, de <http://www.thenote.cl/category/harry-el-pequeno-autobus-publico-y-autonomo-que-comenzara-a-operar-en-londres/>

Note, T. (2017). Conoce el primer autobus con paneles solares en el techo y los laterales. eNote.cl. Recuperado el 26 de Junio 2017, de <http://www.thenote.cl/category/conoce-el-primer-autobus-con-paneles-solares-en-el-techo-y-los-laterales/>

Note, T. (2017). Inauguran primer bus eléctrico del país que operará de forma gratuita en el centro de Santiago. [online] eNote.cl. Disponible en: <http://www.thenote.cl/category/inauguran-primer-bus-electrico-del-pais-que-operara-de-forma-gratuita-en-el-centro-de-santiago/> [26 Jun. 2017].

Note, T. (2017). México estrena taxis híbridos para reducir la contaminación. eNote.cl. Recuperado el 26 de Junio 2017, de <http://www.thenote.cl/category/mexico-estrena-taxis-hibridos-para-reducir-la-contaminacion/>

Novak, M. (2017). GM Car of the Future (1962). [online] Paleofuture.gizmodo.com. Available at: <http://paleofuture.gizmodo.com/gm-car-of-the-future-1962-512629798> [Accessed 27 Jun. 2017].

O

OfficialDeathCubeK (2008). Reactable - How it works (High Quality) - Bjork. [online] YouTube. Available at: <https://www.youtube.com/watch?v=hNeCHI4NAzw> [Accessed 27 Jun. 2017].

Ozimek, A. (2014). Forbes Welcome. [online] Forbes.com. Disponible en: <https://www.forbes.com/sites/mod-elledbehavior/2014/11/08/the-massive-economic-benefits-of-self-driving-cars/#158d57e83273> [Revisada el 26 Jun. 2017].

P

Perrin, J., Martin, P., and Jhaveri, C. (2002) "Interfacing SCOOT and CORSIM: Three Salt Lake City Networks Modeled," Compendium of Technical Papers from the ITE 2002 Annual Meeting and Exhibit, Philadelphia, PA, Institute of Transportation Engineers.

Pino, P. (2014). En 10 años habrá el doble

de autos en Santiago: Gobierno crea «comité pro movilidad» para salir de los tacos. LaSegunda.com. Recuperado el 26 de Junio 2017, de <http://www.lasegunda.com/Noticias/Impreso/2014/06/939367/en-10-anos-habra-el-doble-de-autos-en-santiago-gobierno-crea-comite-pro-movilidad-para-salir-de-los-tacos>

Plataforma Urbana (2017). Las "Zonas 30" comienzan a volverse realidad en Chile por Gaete C. Plataforma Urbana. Retrieved 1 December 2017, from <http://www.plataformaurbana.cl/archive/2014/01/27/las-zonas-30-comienzan-a-volverse-realidad-en-chile/>

Plethora (2017). Block'hood. [online] Plethora Project. Available at: <https://www.plethora-project.com/block-hood/> [Accessed 27 Jun. 2017].

Pony Parts, C. (2014). Are We On e Road To Self-Driving Cars? [Infographic] | Mustang News Blog | [online] Blog.cjponyparts.com. Disponible en: <http://blog.cjponyparts.com/2014/08/are-we-road-self-driving-cars-infographic/> [Revisada el 26 Jun. 2017].

Project Evergreen. (2017). Environmental Benefits of Green Space. Project Evergreen. Recuperado el 26 de Junio 2017, de <http://projectevergreen.org/resources/environmental-benefits-of-green-space/>

PTV VIC (2013). Benefits of public transport - Public Transport Victoria. [online] Ptv.vic.gov.au. Disponible en: <https://www.ptv.vic.gov.au>

gov.au/about-ptv/media-centre/student-media-enquiries/benefits-of-public-transport/ [Revisada el 26 Jun. 2017].

R

Reynolds, C. (s.f.). *Simulation and Nature in Design | Nervous System.* [online] *Nervo-u-s.com*. Disponible en: <http://nervo-u-s.com/education/simulation/week4.php> [Revisado el 27 Jun. 2017].

Roizman, T. (2015). *Advantages & Disadvantages of Cycling.* *LIVESTRONG.COM*. Recuperado el 26 de Junio 2017, de <http://www.livestrong.com/article/438688-advantages-disadvantages-of-cycling/>

Richardson, D. (2017). *Why Our Connection With Nature Matters | Psychreg.* *Psychreg*. Retrieved 6 December 2017, from <http://www.psychreg.org/connection-nature-matters/>

Roundabout. (2017). *En.Wikipedia.org*. Recuperado el 26 de Junio 2017, de <https://en.wikipedia.org/wiki/Roundabout>

S

SAE. (2017). *Automated Driving Levels Of Driving Automation Are Defined In New SAE International Standard J3016. (1st ed.).* Recuperado de https://www.sae.org/misc/pdfs/automated_driving.pdf

Schiller, B. (2013). *Volvo Tests A Road at Can Charge Cars And Trucks.* *Fast Com-*

pany. Recuperado el 26 de Junio 2017, de https://www.fastcompany.com/3016069/futurist-forum/volvo-tests-a-road-that-can-charge-cars-and-trucks?show_rev_content
Señalética. (2017). *Es.Wikipedia.org*. Recuperado el 26 de Junio 2017, de <https://es.wikipedia.org/wiki/Se%C3%B1al%C3%A9tica>
Sipes, J., & Sipes, M. (2013). *Creating Green Roadways* (pp. 3, 46). Washington, DC: Island Press/Center for Resource Economics. SMART. *Future Urban Mobility | Research.* *fm.smart.MIT.edu*. Recuperado el 12 April 2017, de <http://smart.mit.edu/research/future-urban-mobility/future-urban-mobility.html>

Stenquist, P. (2014). *In Self-Driving Cars, a Potential Lifeline for the Disabled.* *NYTimes.com*. Recuperado el 26 de Junio 2017, de https://www.nytimes.com/2014/11/09/automobiles/in-self-driving-cars-a-potential-lifeline-for-the-disabled.html?_r=1

T

Team Treehugger (2014). *10 reasons why you should use public transport.* [online] *TreeHugger*. Disponible en: <https://www.treehugger.com/htgg/how-to-go-green-public-transportation.html> [26 Jun. 2017].

TIME (2017). *Science: Radio Auto.* [online] *TIME.com*. Available at: <http://content.time.com/time/magazine/article/0,9171,720720,00.html> [27 Jun. 2017].

Transportenvironment.org.

(2008). *'Eco-Pass' begins as Milan fights city pollution | Transport & Environment.* [online] Available at: <https://www.transportenvironment.org/news/%E2%80%98eco-pass%E2%80%99-begins-milan-fights-city-pollution> [Accessed 27 Jun. 2017].

TruthCoin. (2017). *Centralizado, Descentralizado y Distribuido.* *Truthcoin.info*. Retrieved 1 December 2017, from <http://www.truthcoin.info/images/cent-dec-dist.jpg>

Tucker, E. (2017). *Carlo Ratti designs graffiti-painting drones to safely make multistorey artworks.* [online] *Dezeen*. Available at: <https://www.dezeen.com/2017/05/15/carlo-ratti-graffiti-painting-drones-multistorey-artworks-design-products-technology-robots/> [Accessed 27 Jun. 2017].

U

UNICEF. (2016). *Clear the air for children.* Nueva York. Recuperado de https://www.unicef.org/publications/files/UNICEF_Clear_the_Air_for_Children_30_Oct_2016.pdf

Union of Concerned Scientists

(2014). *Cars, Trucks, and Air Pollution.* [online] *Union of Concerned Scientists*. Disponible en: <http://www.ucsusa.org/clean-vehicles/vehicles-air-pollution-and-human-health/cars-trucks-air-pollution#.WQJLiVM19E4> [Revisada el 26 Jun. 2017].

Universidad Abierta Interamericana, & M. Calvente, Ing., A. (2007).

El concepto moderno de sustentabilidad (1st ed.). Recuperado de <http://www.sustentabilidad.uai.edu.ar/pdf/sde/uaisds-100-002%20-%20sustentabilidad.pdf>
University Of Virginia. (2016). What Does a Driverless Future Look Like?. UVA Today. Recuperado el 26 de Junio 2017, de <https://news.virginia.edu/content/what-does-driverless-future-look>

V

Vera, J. and Vera, J. (2010). Sostenibilidad económica y social como prioridad para la.... [online] GestioPolis - Conocimiento en Negocios. Disponible en: <https://www.gestiopolis.com/sostenibilidad-economica-social-prioridad-sustentabilidad-ambiental/> [Revisada el 26 Jun. 2017].

Vox. (2016). Superblocks: How Barcelona is taking city streets back from cars. YouTube. Retrieved 27 June 2017, from https://www.youtube.com/watch?v=-ZORzsubQA_M

W

Walker, J. (2016). Can public transport investment really x tra c congestion?. CityMetric. Recuperado el 26 de Junio 2017, de <http://www.citymetric.com/transport/can-public-transport-investment-really-x-tra-c-con-gestion-1870>

Weinberger, M. (2017). A Microsoft robot got the highest all-time score in 'Ms. Pac-Man'. [online] Business Insider. Available

at: <http://www.businessinsider.com/microsoft-ai-pacman-highest-all-time-score-2017-6> [Accessed 27 Jun. 2017].

WHO. (2004). Road safety - Speed (1st ed., pp. 1-2). World Health Organization. Retrieved from http://www.who.int/violence_injury_prevention/publications/road_traffic/world_report/speed_en.pdf

Wikipedia. (2017). Bicicleta. Recuperado el 26 de Junio 2017, de https://es.wikipedia.org/wiki/Bicicleta#Origen_de_la_bicicleta

Wikipedia. (2017). Emergence. [online] En.wikipedia.org. Disponible en: https://en.wikipedia.org/wiki/Emergence#Emergent_structures_in_nature [Accessed 27 Jun. 2017].

Wikipedia (2017). History of autonomous cars. [online] En.wikipedia.org. Available at: https://en.wikipedia.org/wiki/History_of_autonomous_cars# [Accessed 27 Jun. 2017].

Wikipedia.org. (2017). Platoon (automobile). [online] Disponible en: [https://en.wikipedia.org/wiki/Platoon_\(automobile\)](https://en.wikipedia.org/wiki/Platoon_(automobile)) [Revisada el 26 Jun. 2017].

Wikipedia (2017). Sistema emergente. [online] Es.wikipedia.org. Disponible en: https://es.wikipedia.org/wiki/Sistema_emergente [Revisado el 27 Jun. 2017].

Woon, C. (1999). Size, Sprawl, Speed and the Efficiency of Cities Urban Studies - Remy Prud'homme, Chang-Woon Lee, 1999. Journals.sagepub.com. Recuperado el 26 de Junio 2017, de <http://journals.sagepub.com/doi/abs/10.1080/0042098992638>

World Health Organization. (2015). Road Traffic Death And Proportion of Road Users By Country/Area. 1st ed. [ebook] p.264. Disponible en: http://www.who.int/violence_injury_prevention/road_safety_status/2015/TableA2.pdf?ua=1 [Revisada el 26 Jun. 2017].

Y

Yuniar, R. (2016). Grab Joins nuTonomy to Offer Self-Driving Taxis in Singapore. WSJ. Recuperado el 26 de Junio 2017, de <https://www.wsj.com/articles/grab-joins-nutonomy-to-offer-self-driving-taxis-in-singapore-1474598345>

#

2Get there (2016). Driverless Parkshuttle - 2getthere. [online] 2getthere. Available at: <http://www.2getthere.eu/driverless-parkshuttle/> [Accessed 27 Jun. 2017].

99% Invisible (2017). 'Shared Space' Design: Road Signs Suck. What if We Got Rid of Them All? - 99% Invisible. 99% Invisible. Retrieved 1 December 2017, de 99percentinvisible.org/article/shared-space-design-road-signs-suck-got-rid/

99% Invisible (2017). Civic Superblocks: Barcelona's Urban Redesign Returns Streets to Residents - 99% Invisible by Kurt Kohlstedt. Visto el 1 Diciembre 17, de 99percentinvisible.org/article/civic-superblocks-barcelonas-urban-redesign-returns-streets-residents/

