TC2008B.5 Modelación de sistemas multiagentes con gráficas computacionales (Gpo 5)

Reto: Movilidad urbana

Documentación Final

"*Lorem Ipsum*"

G. Andrés Castillo	A01720388
Eduardo Antonio Beitia	A00829121
Alfonso Villarreal Galindo	A00828725
Sergio Lopez Urzaiz	A00827462

Profesoras: Dra. Lorena Martínez Elizalde, Dra. Raquel Landa Cavazos

Socio Formador: CII.IA

Agosto 12, 2021

Semestre Agosto-Diciembre 2021

Tec de Monterrey, Campus Monterrey

Índice General

Introducción 3

Créditos 3

Problema 5

Objetivos Generales 5

Restricciones 6

Requerimientos 6

- **Funcionales** 6
- **No Funcionales** 7

Descripción del sistema Multiagente 8

- **Diagramas de agente estacionamiento** 8
- **Diagramas de Agente Coche** 9
- **UML** 9
- **Modelo de los Agentes** 10
- **Modelo del Escenario** 17
- **Modelo de la Interacción** 18

Modelación 3D 18

- **Modelos Individuales** 18
- **Modelos Importados** 20
- **Escenario de Simulación** 22

Apéndice 1 23

Referencias 28

Introducción

El tráfico vehicular en los estacionamientos de las ciudades es algo que escuchamos seguido en cualquier estacionamiento al que vamos, esto gracias a la gran cantidad de vehículos que asisten a las tiendas, restaurantes, cines, apartamentos, etc. Esto ha causado no solamente la congestión de vehículos dentro de los estacionamientos sino también de las vías que se encuentran alrededor del alrededor de ellos afectando la vialidad de las ciudades.

Nuestro equipo propondrá una solución con ayuda de un sistema multiagente para poder simular cómo mejora el tráfico dentro del estacionamiento de una ciudad una vez que los agentes adopten comportamiento que permita el flujo constante y ordenado de los vehículos que buscan salir y entrar del estacionamiento.

Créditos

Nombre	Sergio López Urzaiz
Rol	Desarrollador
Categoría profesional	Diseño de agentes y entorno
Responsabilidades	• Programar agentes • Diseño de ambientes
Información de contacto	Cel: 9993607437 Mail: a00827462@itesm.mx

Nombre	Guillermo Andres Castillo Chapa
Rol	Programador

Categoría profesional	De Agentes y Gráficos
Responsabilidades	• Hablar con el cliente • Programar agentes • Ayudar en la creacion de graficos
Información de contacto	Cel: 8116367290 Mail: A01720388@itesm.mx

Nombre	Alfonso Villarreal Galindo
Rol	Desarrollador
Categoría profesional	Diseño de entorno
Responsabilidades	• Testear las soluciones • Documentación
Información de contacto	Cel: 8211111822 Mail: A00828725@itesm.mx

Nombre	Eduardo Antonio Beitia Tristan
Rol	Programador
Categoría profesional	Programador de algoritmos
Responsabilidades	• Programar algoritmos de aprendizaje • Programar servidor
Información de contacto	Cel: 81 35822934 Mail: A00829121@itesm.mx

Problema

Queremos resolver una problemática de congestión en el estacionamiento simulando cómo se comporta la salida y entrada de autos en el mismo. Dicho estacionamiento en la problemática tiende a estar muy lleno debido a su cercanía con la ciudad y su ingreso de autos, lo cual causa que se generen congestionamientos en el estacionamiento los cuales dependiendo del tipo y logística de entrada y salida puede causar tiempos de espera de hasta 10 minutos o más para poder encontrar estacionamiento o para poder salir de este mismo. Es importante buscar una solución a este problema ya que es posible que el estancamiento en los estacionamientos ocasione accidentes de tránsito dentro y fuera de él. Además se busca generar un sistema multiagentes el cual nos ayude a simular correctamente el comportamiento para tener una mayor exactitud en las simulaciones.

Objetivos Generales

Nuestro principal objetivo es proponer una solución para aminorar los tiempos de entrada y salida de autos en un estacionamiento, ya que como se ha comprobado en la vida real hay diversos tipos de logística que en vez de mejorar la funcionalidad de este la empeoran además de provocar embotellamientos y en su caso accidentes automovilísticos dentro del estacionamiento.

Con nuestra propuesta buscamos mejorar los tiempos de espera en un 15%, esto será comprobable ya que se medirán los respectivos tiempos en diversas simulaciones con y sin la solución propuesta y así poder corroborar nuestra hipótesis.

Restricciones

Una de las principales restricciones es que no habrá algunas variables que simulen algún tipo de comportamiento climatológico que afecte a la velocidad de los agentes, no habrá peatones que también afecten la velocidad y tampoco habrá variables que simulan iluminación dentro del estacionamiento. Estos factores pueden llegar a limitar la exactitud de la simulación ya que estas condiciones previamente mencionadas afectan el comportamiento y tiempos de espera.

Por cuestiones de simplificación del concepto la modelación y la implementación de los agentes se hará de un estacionamiento simple de 20 cajones lo cual va a funcionar como prueba de nuestro concepto. Se considera estacionamiento simple ya que la mayoría de los estacionamientos realistas, contienen una mayor cantidad de cajones (mínimo 75) y tienen un diseño más admirado a la arquitectura del edificio o lugar. Si en un futuro se busca crear una simulación de algo más realista se tendría que modificar y expandir los modelos de estacionamiento.

Requerimientos

Funcionales

RF-001 Los agentes simularan el flujo de entrada y salida del estacionamiento, así también como el comportamiento en el cual el agente coche reconocerá cuál es el estacionamiento que el agente estacionamiento le ha asignado y el valor o los puntos de ese estacionamiento.

RF-002 Los agentes se moverán por los espacios de las calles y por los espacios de estacionamiento de manera animada simulando como es la aceleración y el manejo de un vehículo en un estacionamiento.

RF-003 Diseño del estacionamiento tendrá en donde los agentes actuarán en el ambiente interactuando con las rutas direccionadas, entradas, espacios de estacionamiento y salidas.

RF-004 El sistema de estacionamiento debe de mejorar la eficiencia del estacionamiento.

RF-005 El sistema de estacionamiento debe de mejorar el tiempo de espera para que los autos se tarden menos tiempo en estacionarse.

No Funcionales

RN-001 Se programará en lenguaje c# todo lo relacionado con el movimiento y las traslaciones que se verán en los agentes coches. Para que sea posible aplicaremos a los objetos mallas que nos permitan poder mover el objeto como muchas mayas juntas.

RN-002 Se programara con python la librería agentpy y seaborn los modelos y agentes que tendremos en nuestro sistema multiagente. Estos le darán el comportamiento a los agentes dentro de la escena de Unity.

RN-003 Se creará un servidor de python que nos permita conectar el motor de Unity con el modelo creado en agentpy para poder darle a nuestros assets los comportamientos de los agentes que creamos.

Descripción del sistema Multiagente

- Automóviles
    - Habrá una numerosa cantidad de agentes automóviles entrando y buscando espacio de estacionamiento e igual que liberando espacio de estacionamiento y saliendo todos al mismo tiempo estarán realizando acciones diferentes pero siempre bajo las entradas que reciban sus sensores para permitir que no se altere el flujo del estacionamiento.

- Estacionamiento individual (cajón)
    - El sistema estará constantemente revisando que los lugares sean llenados de forma correcta por los estudiantes, sin embargo en el caso de que no sea así se está revisando constantemente que cajones están o no disponibles para evitar contratiempos.

Diagrama de Actividades: Agente Estacionamiento

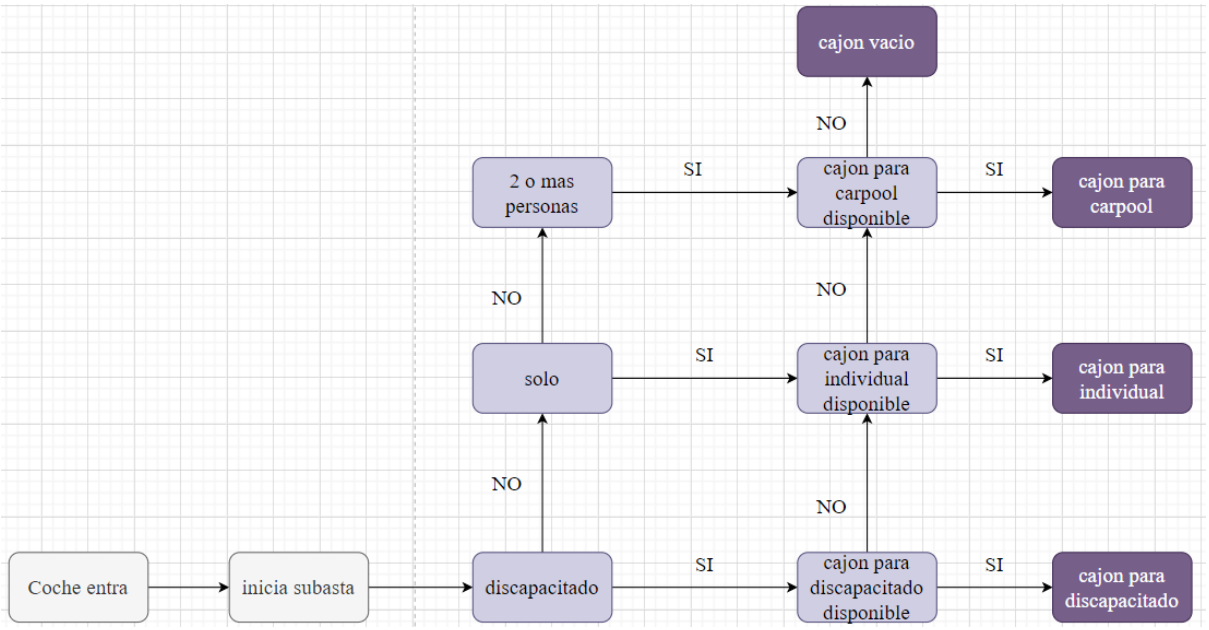


Diagrama de Actividades: Agente Coche

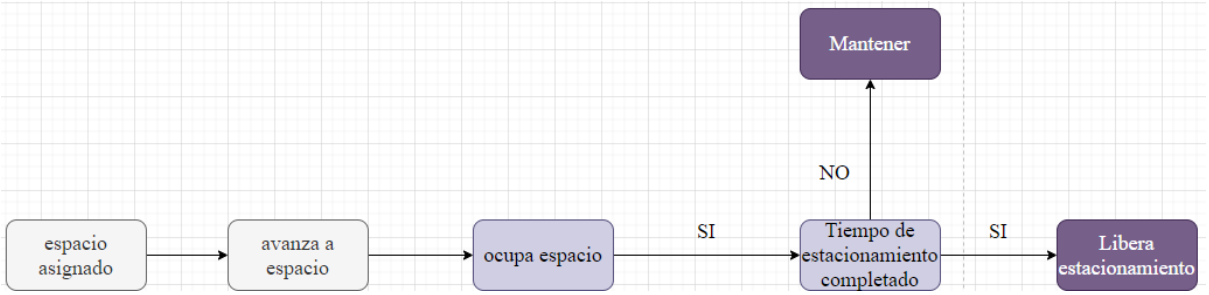
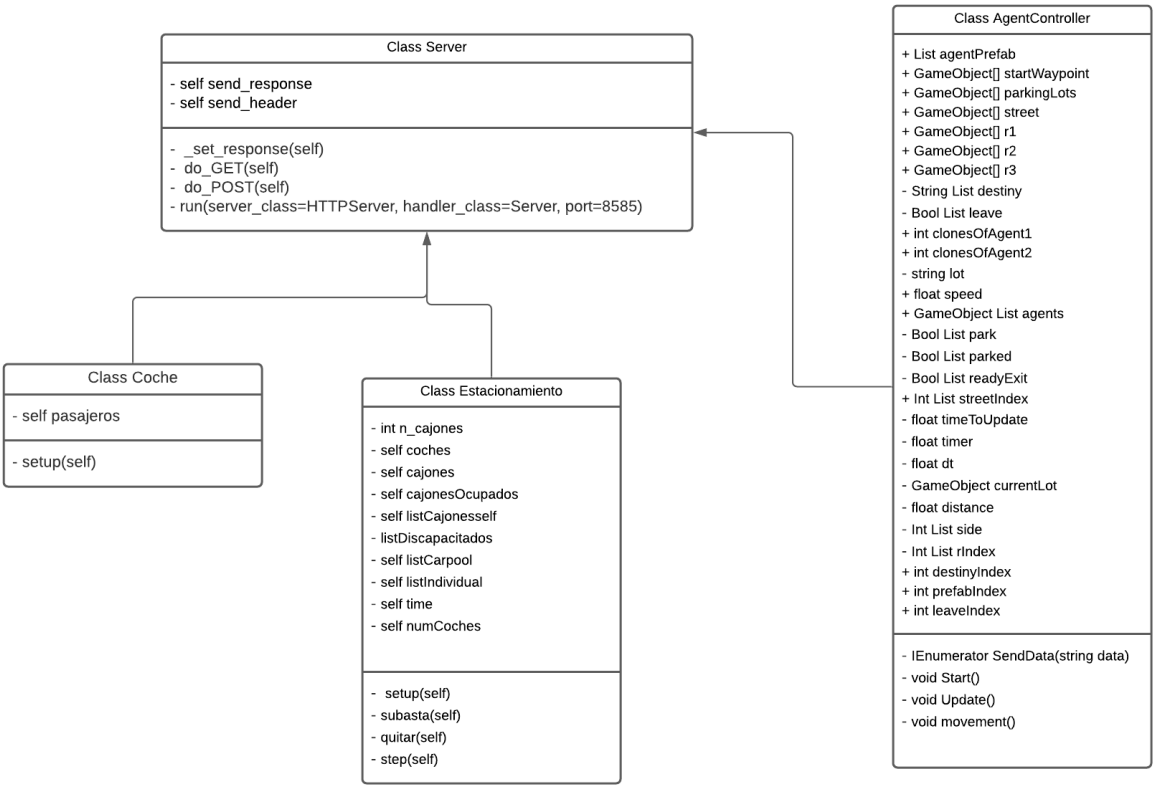


Diagrama UML



Modelo de los Agentes

<u>Sistema de estacionamiento:</u>

- Planes: El sistema de estacionamiento va a funcionar asignando a los vehículos que entran mediante las prioridades de las especificaciones del agente coche, implementaremos una función de subasta para que sea capaz de pensar y decidir cual es el mejor estacionamiento para los agentes coche que ingresen al estacionamiento.

- Creencias: El agente toma ciertos cajones como más valiosos que otros con base en la cercanía de las cosas del ambiente, entre más cerca más valor asignado tiene el cajón. El sistema toma en consideración varios de los autos como por ejemplo si es discpaciatado o el número de pasajeros. Se le dará prioridad a los autos que presenten la característica de discapacidad y después de eso se le dará prioridad a los autos que hagan carpooling (mayor cantidad de pasajeros en el carro). Entre más personas se les dará un lugar más prioritario.

- Cooperación: Como se mencionó anteriormente el sistema funcionará con un algoritmo que le permitirá subastar las prioridades para los agentes coches, almacenará estados para saber si tiene disponibilidad o no y se le asignará una posición en el plano para que el agente coche sepa donde esta ubicado en el grid o en el espacio de unity.

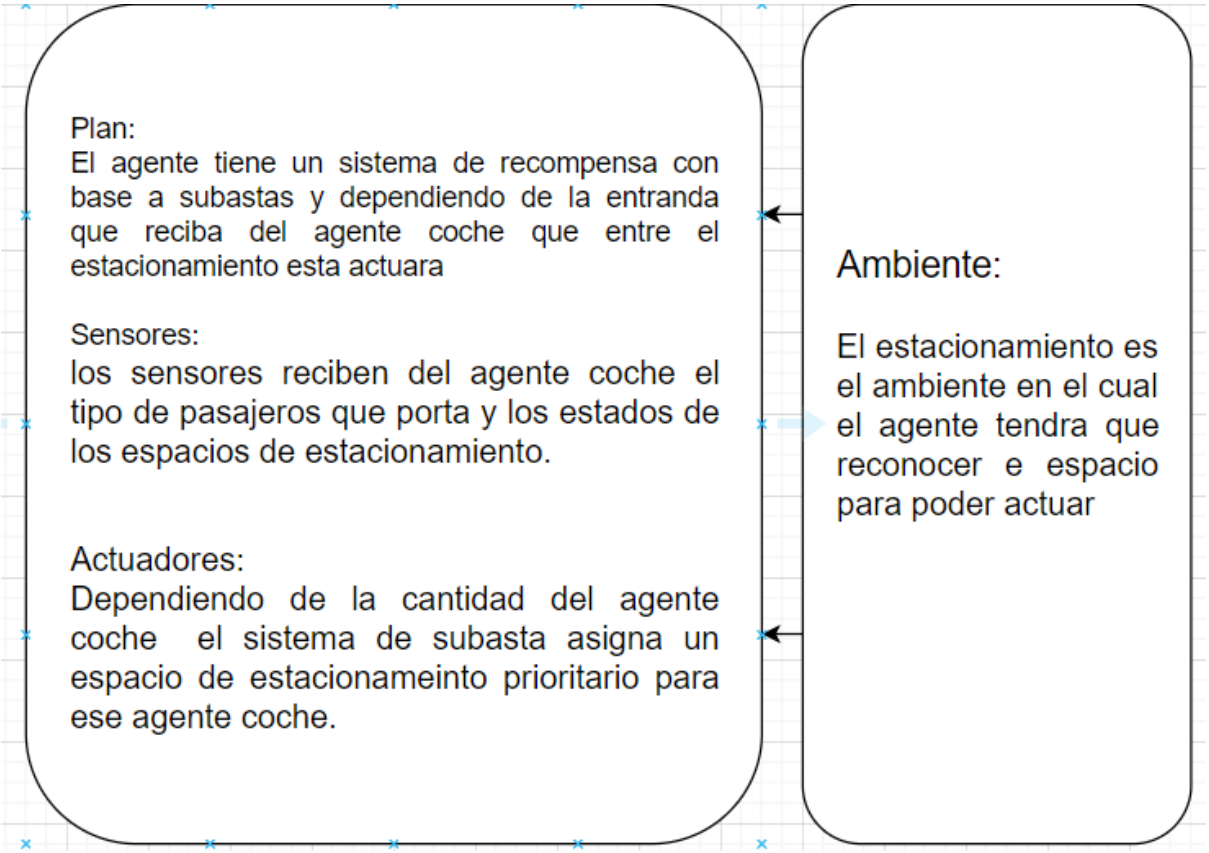


Diagrama de Estados

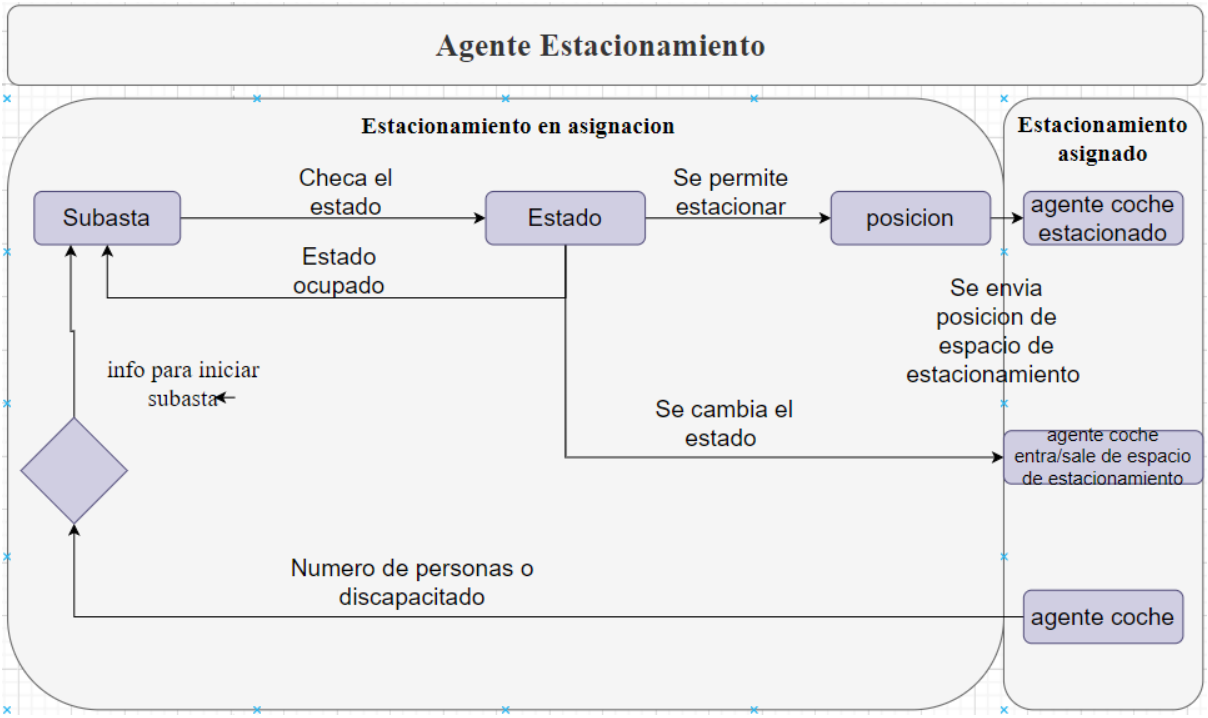
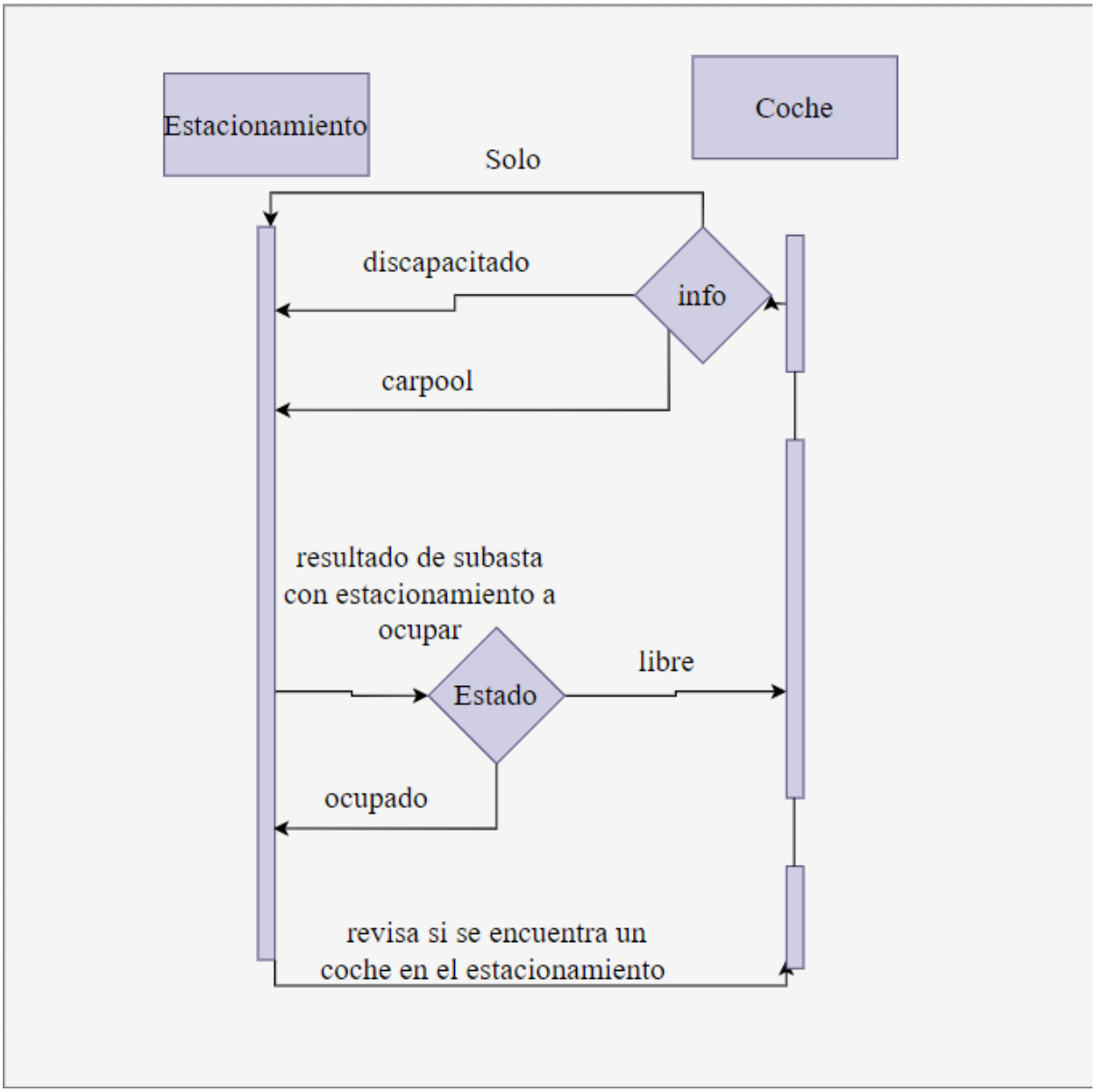


Diagrama de protocolo



Coche/Vehículo:

- Planes: El coche recorre el estacionamiento asignado. Es entonces que este agente trabajará con un modelo reactivo debido a que su movimiento se basará en los “inputs” realizados por sus sensores.

- Creencias: El agente entra al estacionamiento y sabrá hacia dónde avanzar gracias a la indicación del otro agente. Después que pasa el tiempo de

ocupar el estacionamiento el agente libera el espacio y sale de estacionamiento.

- Cooperación: El coche trabajará en conjunto con el agente estacionamiento para poder saber qué cajón está disponible para ser usado y cómo llegar a este. También el coche identificará el camino dentro del estacionamiento para poder hacer el recorrido al cajón de estacionamiento.

- Aprendizaje: Cada vez que el coche entra al estacionamiento, este recibirá instrucciones de donde está su cajón e irá a estacionarse a ese cajón sin problema. Sin embargo después de un tiempo el coche podrá buscar por sí solo un cajón disponible.

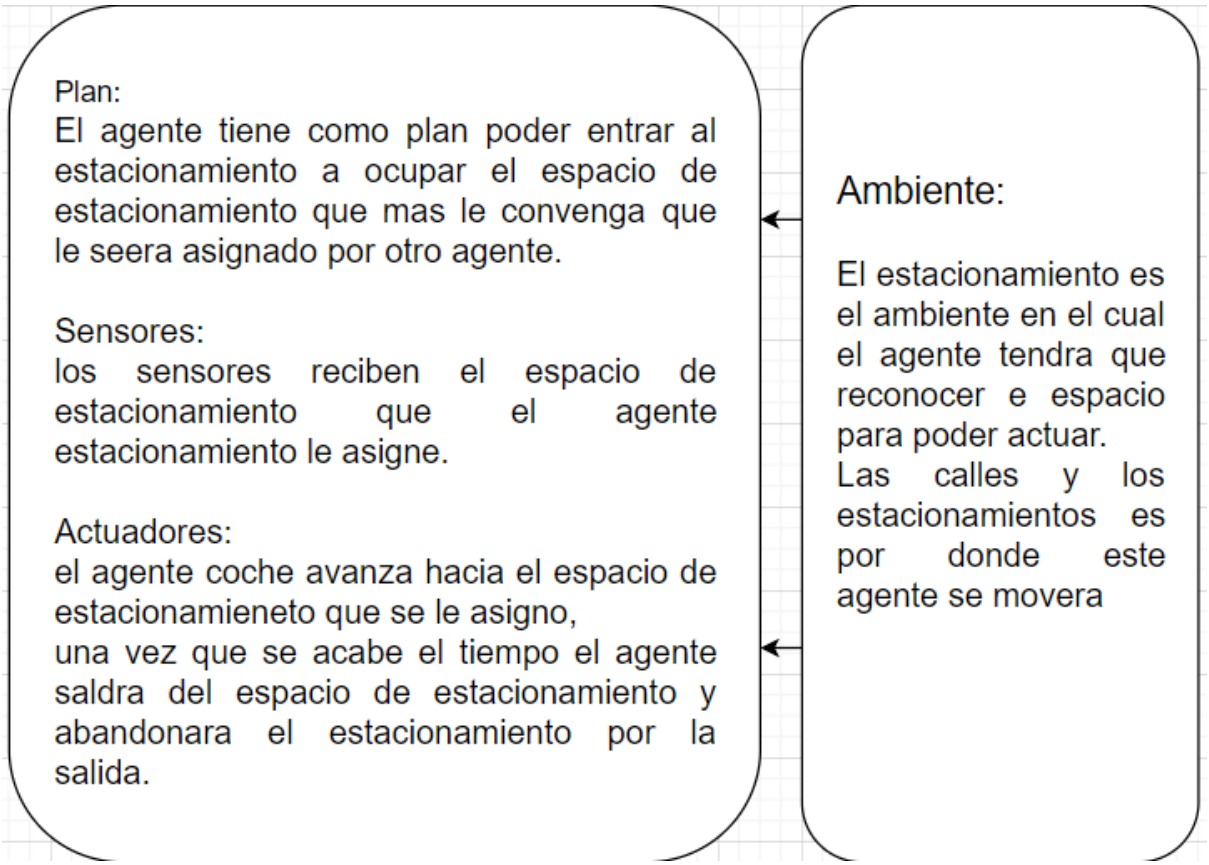


Diagrama de Estados

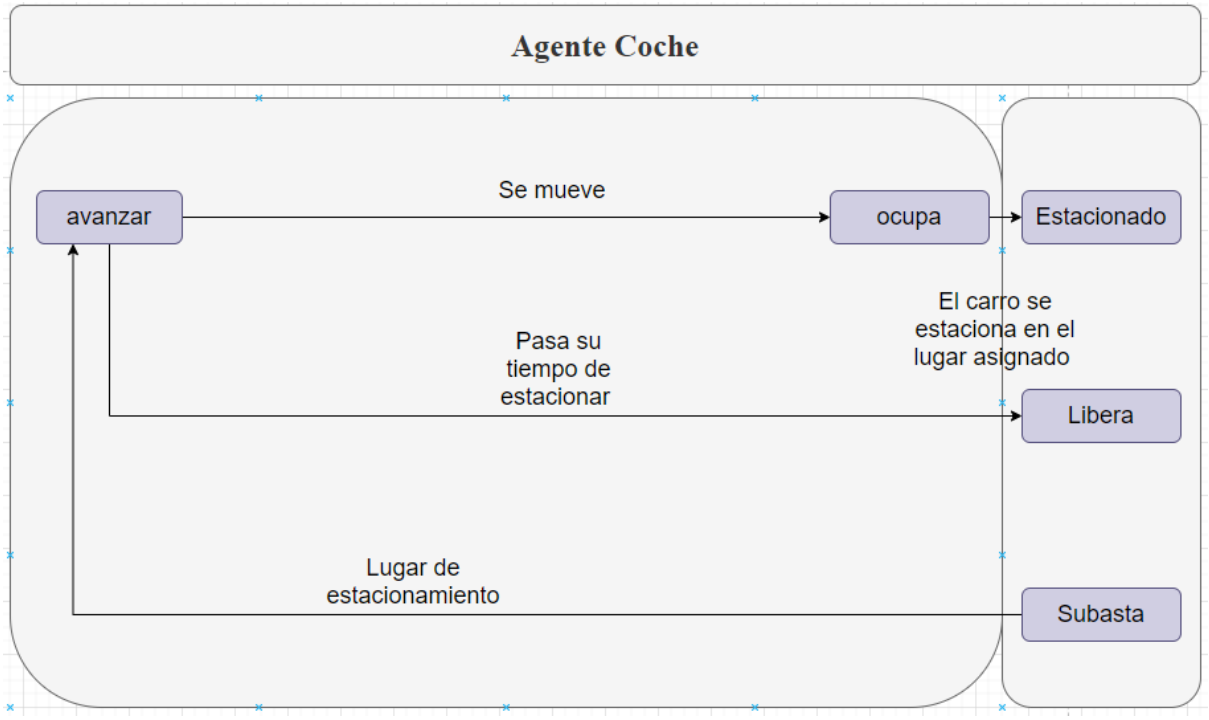
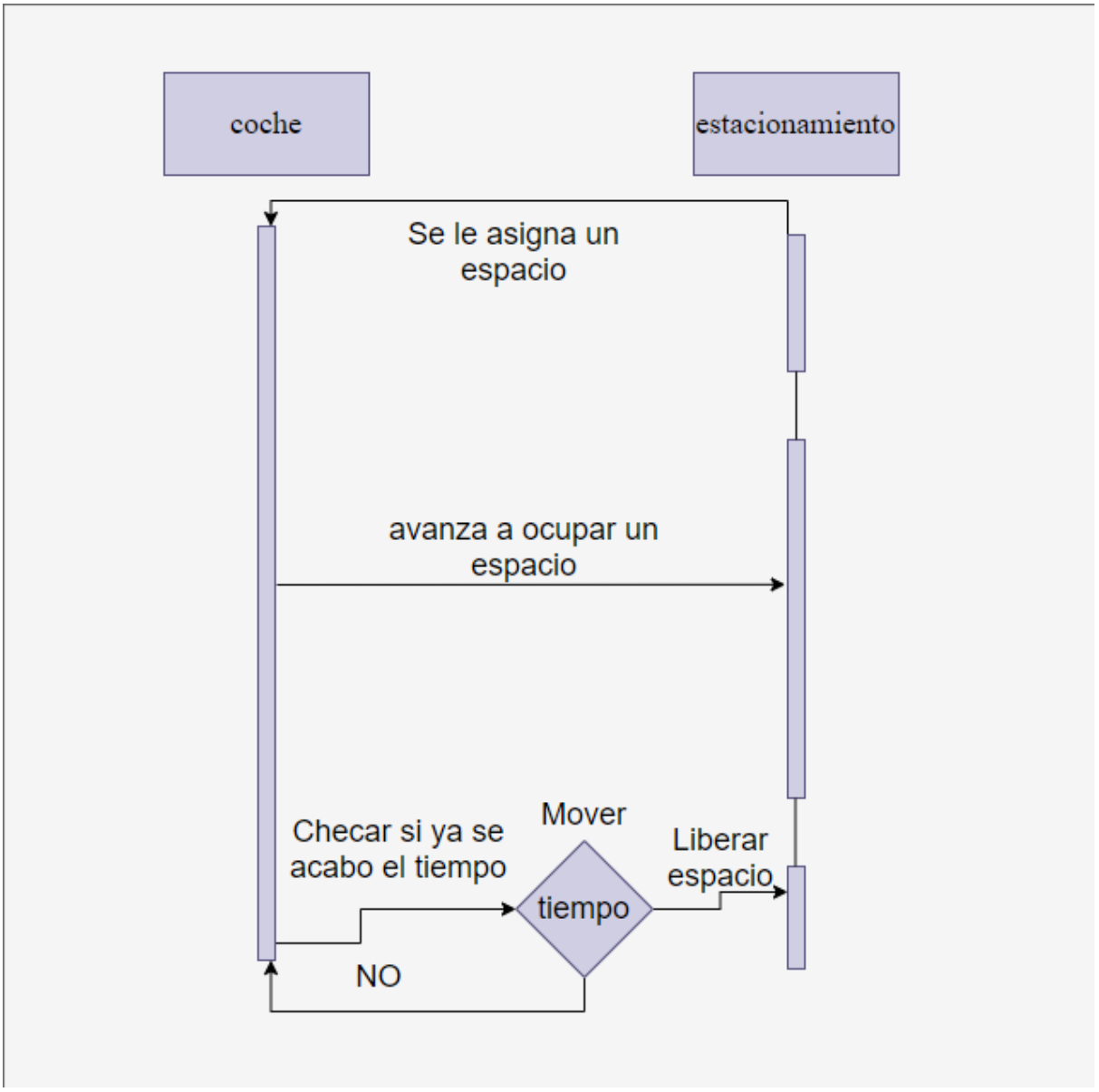
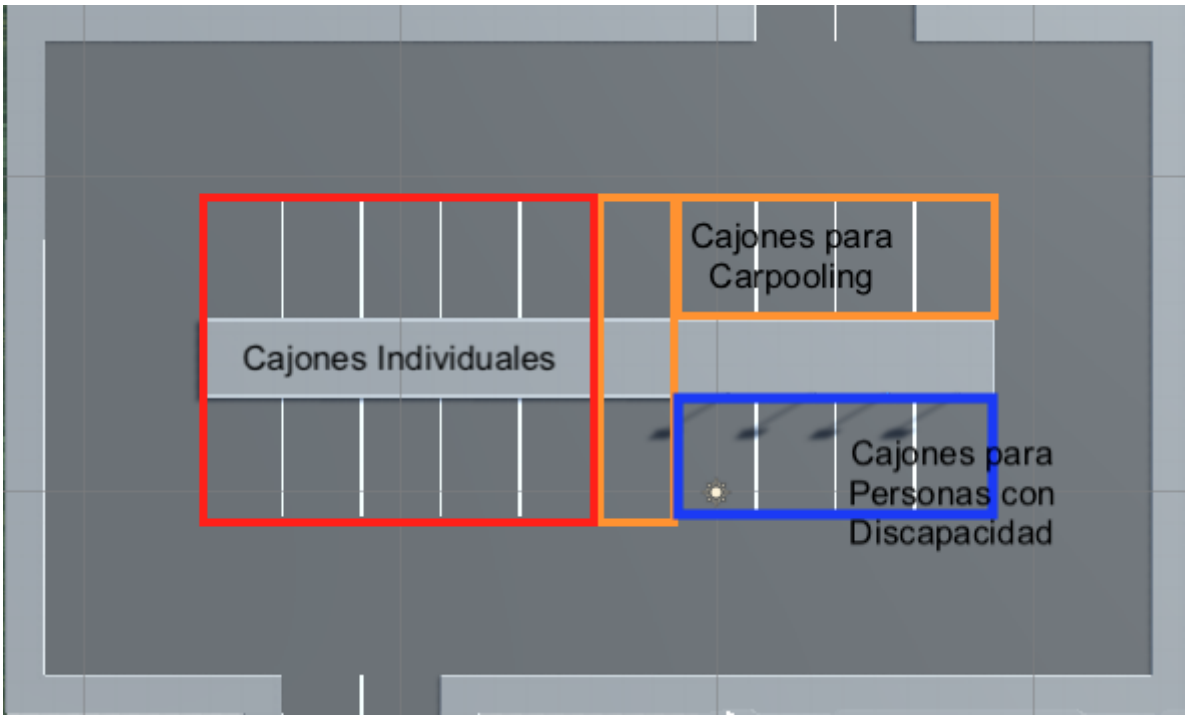


Diagrama de protocolo



Modelo del Escenario

El escenario que estaremos modelando será un estacionamiento con 20 cajones. El principio de este proyecto es poder simular el proceso de estacionamiento de los autos por medio de un agente de subastas el cual asigna los cajones en base a distintas características de cada auto de esta forma se le da prioridad a los autos con personas discapacitadas y que hacen carpooling, además de que agiliza el proceso de búsqueda de un cajón de estacionamiento para los autos.



El ambiente de este escenario se conforma de un ambiente totalmente observable, determinístico ya que se le están dando instrucciones al auto, todo esto es de manera secuencial, el ambiente es totalmente estático, continuo y especialmente multiagente.

Para el ambiente se creó un grid en el cual se definieron posiciones para los cajones y para el camino para llegar al cajón asignado. La medición de tiempo se hará por medio de un contador de steps en base al movimiento del agente auto en el grid y se ejecuta mediante la función de update de unity en conjunto con la de steps de agentPy. *Para más información consultar apéndice 1*

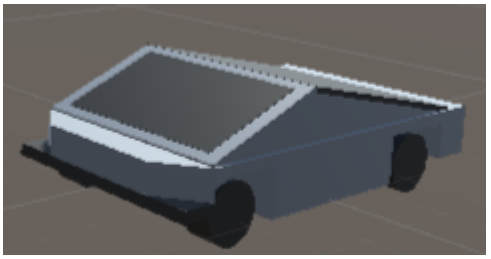
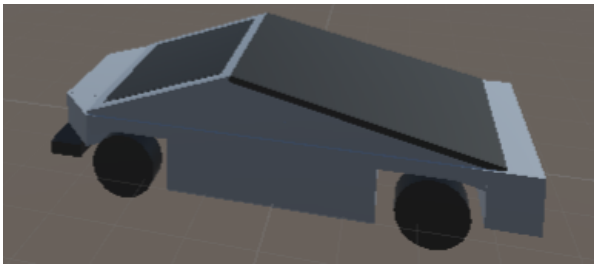
Modelo de la Interacción

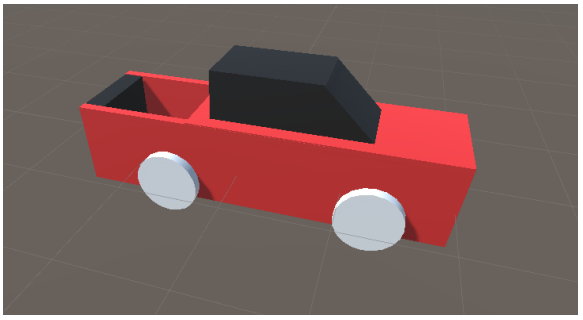
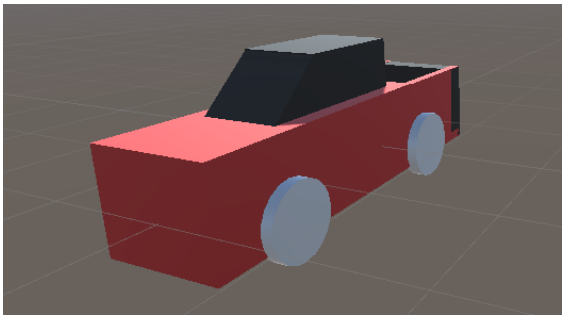
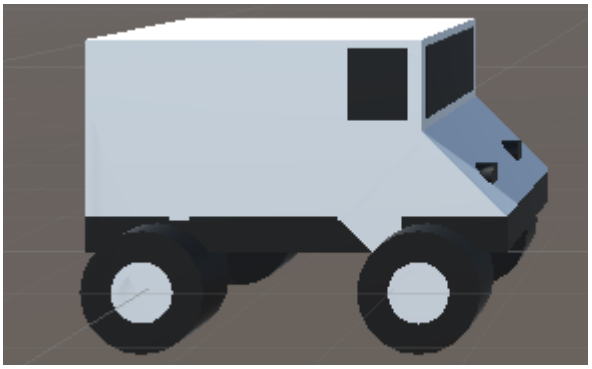
Nosotros para nuestro modelo tendremos una interacción principal entre el sistema de estacionamiento y el coche que está entrando. Dependiendo de si el coche tiene un discapacitado, está haciendo carpool o si va solo, se le asignará un lugar por medio de una *subasta* la cual dará prioridad a los casos mencionados respectivamente. Después, cada vez que tenga una salida un coche, el sistema reconocerá que su lugar ha sido desocupado y que lo puede volver a entregar.

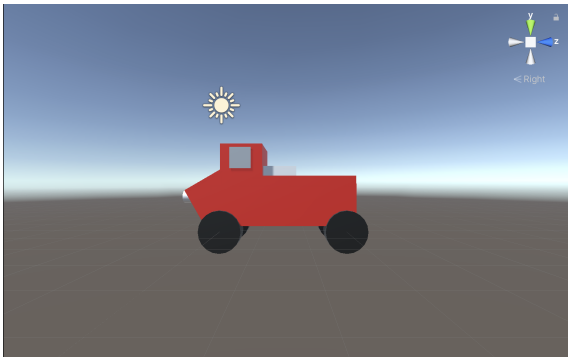
Modelación 3D

Modelos Individuales

A continuación se muestran los autos que fueron modelados por los integrantes del equipo:

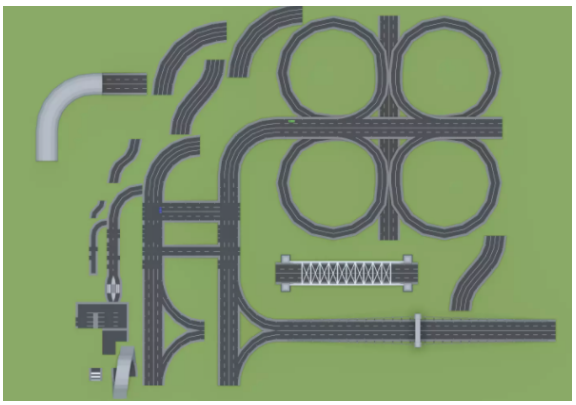
Nombre:	**Modelo:**
Sergio	 

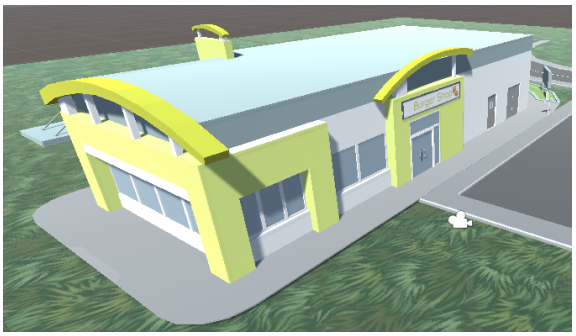
Eduardo	 
Andrés	 

Alfonso	 

Modelos Importados

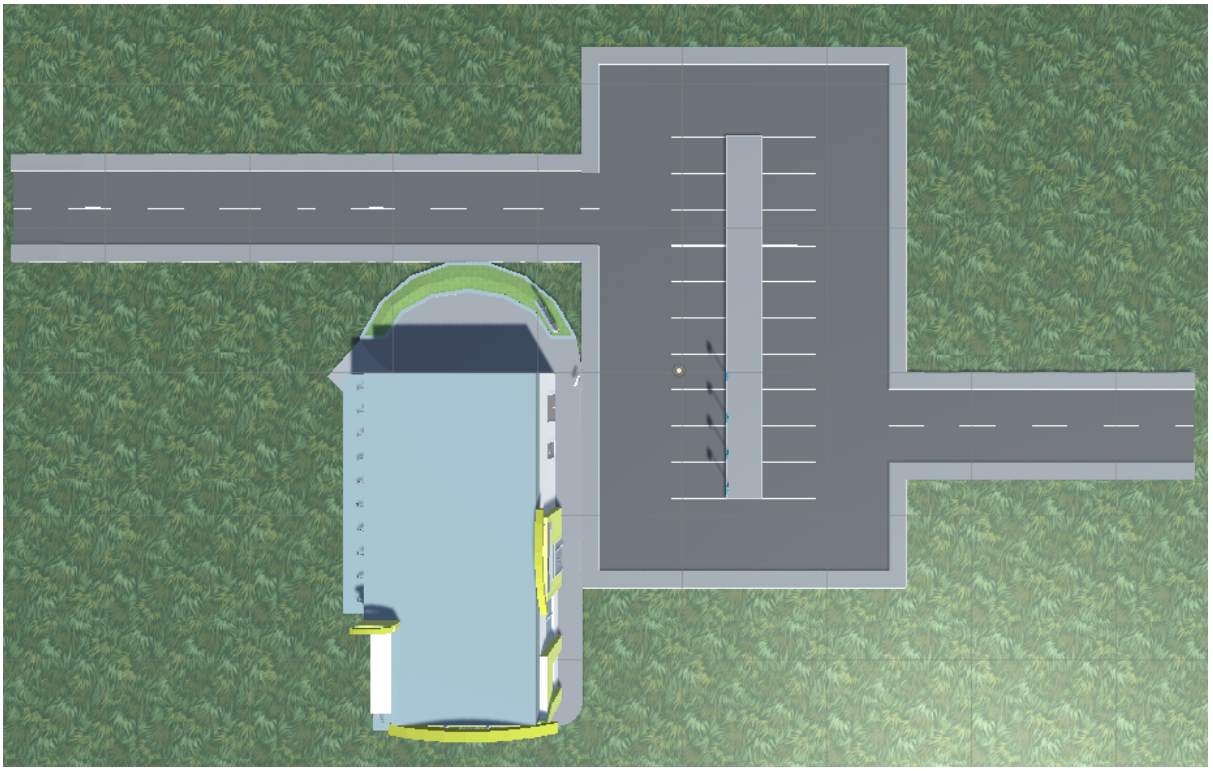
A continuación se muestran los assets que fueron importados del unity asset store:

Descripción del Asset	Imagen	Referencia
Paquete de texturas y asfalto		https://assetstore.unity.com/packages/3d/environments/roadways/low-poly-road-pack-67288

Paquete de señalamientos de calle		https://assetstore.unity.com/packages/3d/props/exterior/old-road-signs-pbr-170952
Textura de Pasto		https://assetstore.unity.com/packages/2d/textures-materials/glass/stylized-grass-texture-153153
Tienda		https://assetstore.unity.com/packages/3d/environments/urban/city-package-107224

Escenario de Simulación

La escena elaborada en unity consiste en un estacionamiento de 20 cajones de los cuales 4 son para personas con discapacidad, 6 son para carpooling y 10 son cajones individuales para autos que no hacen carpooling. El estacionamiento sigue el mismo diagrama mostrado anteriormente (hoja 17) y está colocado de forma que los cajones prioritarios (discapacitados y carpool) tengan su lugar más cerca de la tienda.



Apéndice 1: Explicación de AgentPy

Diseño de Clase Modelo

CLASE: AGENTE ESTACIONAMIENTO

CARACTERÍSTICAS:

- POSICIONAMIENTO
    - Dependiendo de la posición de los grids se le asignará un color para que se distinga cuáles son las calle, limites y cajones de estacionamiento por donde transita el agente coche

- ESTADO
    - El agente estacionamiento tendrá estados de libre u ocupado dependiendo de si un agente coche se encuentra en él o no. En caso de que el estado sea ocupado no se le podrá asignar a otro agente coche ese espacio de estacionamiento solo será posible asignar los que se encuentren en el estado libre.

- SUBASTA
    - El agente estacionamiento contará con un sistema de subastas en donde subastará el estacionamiento más cercano a la salida dependiendo de cuál es la información de pasajeros del agente coche.

CLASE: AGENTE COCHE

CARACTERÍSTICAS:

- VELOCIDAD
    - Tendremos un mínimo (0) y un máximo (TBD) de velocidad para cada coche. Esto se debe a que el coche frenará cuando el coche de

adelante frene o se estacione. El coche entonces esperará a que haya una distancia mínima (TBD) y volverá a acelerar hasta llegar a la velocidad máxima permitida.

- POSICIÓN
    - La posición de los coches será asignada en 2 secciones que representan ambos lados de la calle. Las direcciones de las calles serán opuestas una a otra, por lo que los coches también tendrán una dirección opuesta a la de los coches en la otra calle.

- SEPARACIÓN
    - Queremos que siempre haya una distancia mínima entre coche y coche para poder simular la distancia mínima que se debe tener para evitar coches. Esta separación afecta a la velocidad que tendrá el coche ya que esta determinará si el coche tiene que frenar o si el coche puede acelerar después de haber frenado.

FUNCIONES:

- MODEL.SETUP
    - Con esta función se iniciará el agente con todos sus elementos se le asignará una posición, orientación y velocidad inicial.

- MODEL.STEP
    - Con el step se cambiará el movimiento del agente por cada paso de tiempo que al cambiarlo será actualizado por todas nuestras funciones update. El comportamiento en el cambio del tiempo va a depender de los estímulos que reciba el agente en el ambiente.

- UPDATE_VELOCITY:
    - Esta función la estaremos llamando constantemente para actualizar la velocidad en todo momento. Como mencionamos anteriormente, tendremos un máximo de velocidad al cual se podrá llegar. Es entonces que esta función trabajara en conjunto con la característica

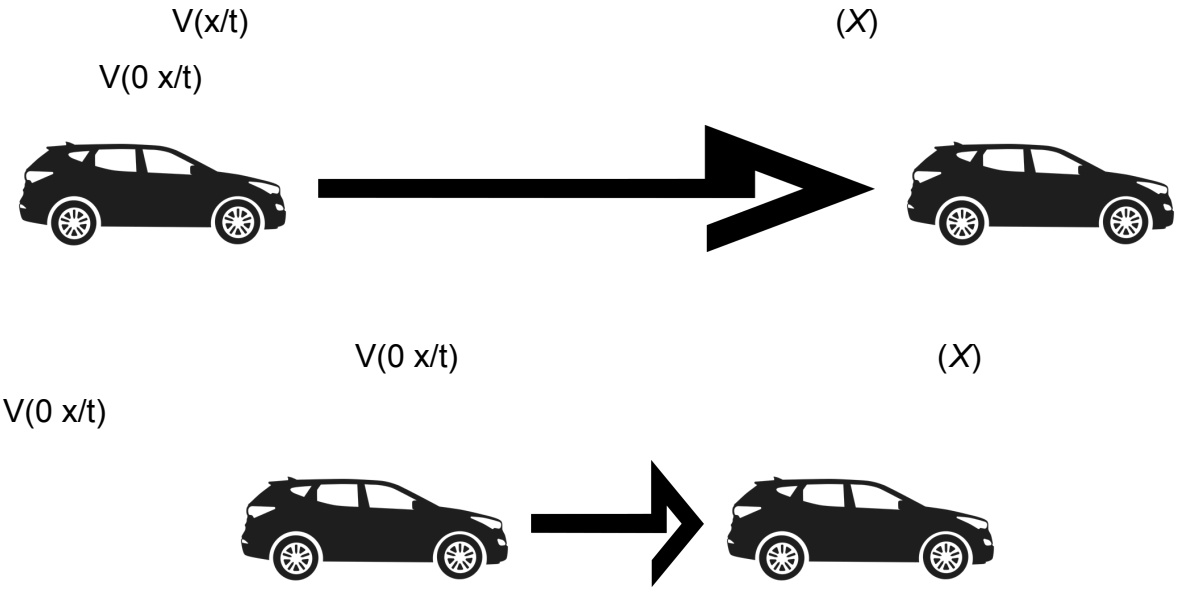
de separación de nuestro coche para determinar cuál es el cambio de velocidad (si es que se necesita uno) que se le debe hacer al coche.

- UPDATE_POS:
    - La función de actualizar posición también se estará llamando todo momento después de llamar a la función de actualizar velocidad. Esta función será con la cual podremos calcular cuánta distancia (*x*) debe de recorrer nuestro coche en el tiempo (*t*) transcurrido.

- UPDATE_SEP:
    - La función de actualizar separación se estará llamando en todo momento después de llamar la función de actualizar posición. Se le asigna un valor mínimo a X (distancia entre autos) y cuando se cumpla este valor el auto debe de tener una velocidad de cero. Esto evitará que los autos (agentes) colisionen con el otro.

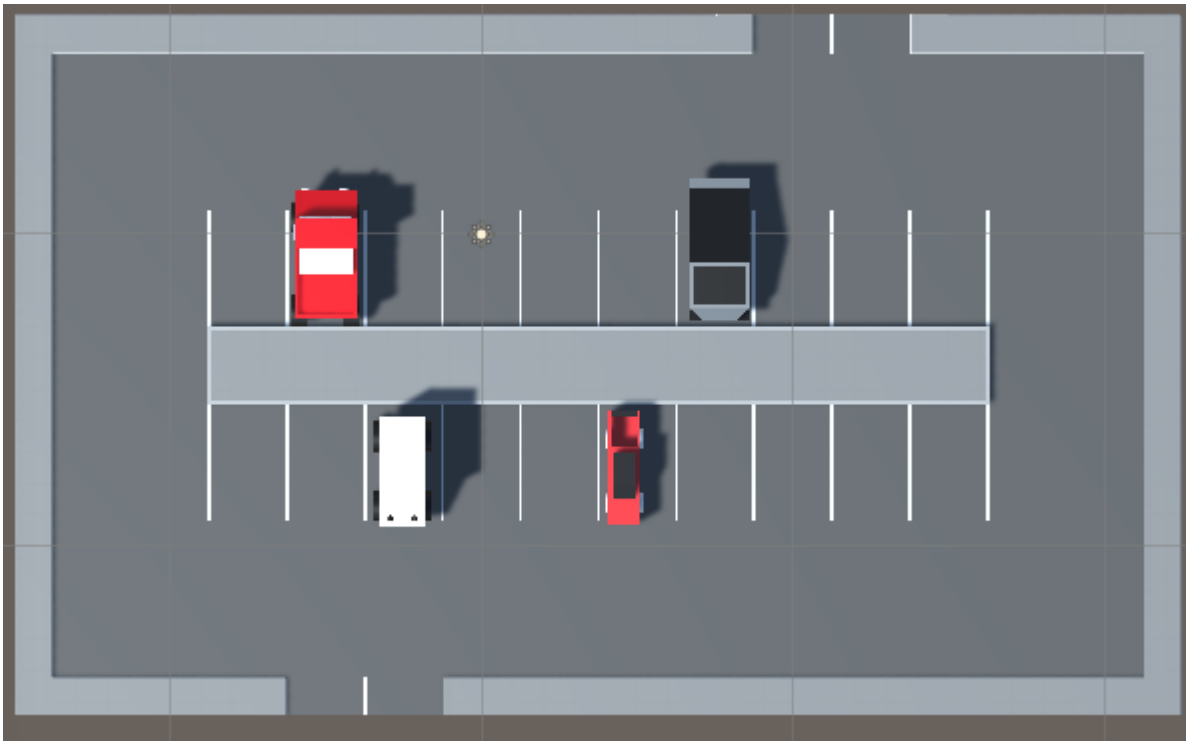
Visualización

Separación:

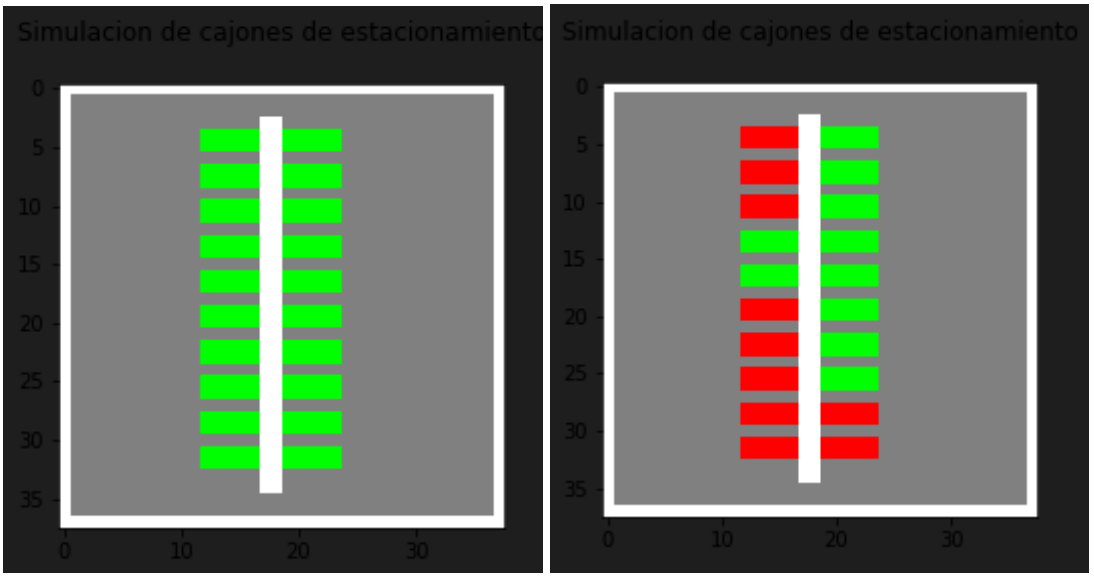


Modelo de Simulación

Para el ambiente de simulación que generamos y mostraremos en Unity se hizo la modelación de un estacionamiento de 20 cajones los cuales están modelados de la misma manera que se encuentran en el agentpy y de igual manera funcionan con el sistema de subastas dependiendo de las características o propiedades de cada auto.



Para el lado de Python, también se modelo un pseudo-estacionamiento y una pequeña visualización con la cual podremos estar viendo que se cumple el ocupado y la liberación de cajones de estacionamiento. Para esto utilizamos la función Grid de la librería de Agentpy, con la cual obtuvimos un espacio deseado y pudimos ir asignando paredes, cajones y disponibilidad. En el siguiente ejemplo se verá como se muestran todos los cajones disponibles y como se muestra cuando uno se ocupa.


Simulacion de cajones de estacionamiento
0
5
10
15
20
25
30
35
0
10
20
30
Simulacion de cajones de estacionamiento
0
5
10
15
20
25
30
35
0
10
20
30

Referencias

“Low poly Road PACK: 3d roadways: Unity asset store,” 3D Roadways | Unity Asset Store. [Online]. Available: https://assetstore.unity.com/packages/3d/environments/roadways/low-poly-road-pack-67288. [Accessed: 07-Sep-2021].

“Old road Signs PBR: 3d Exterior: Unity asset store,” 3D Exterior | Unity Asset Store. [Online]. Available: https://assetstore.unity.com/packages/3d/props/exterior/old-road-signs-pbr-170952. [Accessed: 07-Sep-2021].

“Stylized grass Texture: 2D Glass: Unity asset store,” 2D Glass | Unity Asset Store. [Online]. Available: https://assetstore.unity.com/packages/2d/textures-materials/glass/stylized-grass-texture-153153. [Accessed: 07-Sep-2021].

“CITY package: 3D urban: Unity asset store,” 3D Urban | Unity Asset Store. [Online]. Available: https://assetstore.unity.com/packages/3d/environments/urban/city-package-107224. [Accessed: 07-Sep-2021].