UNIVERSIDAD DEL BÍO-BÍO

UNIVERSIDAD DEL BÍO-BÍO

FACULTAD DE CIENCIAS EMPRESARIALES

DEPARTAMENTO DE SISTEMAS DE INFORMACIÓN

"SIMULACIÓN DIGITAL DE TRÁFICO PARA INTERSECCIONES SEÑALIZADAS POR SEMÁFORO, BAJO AMBIENTE TRIDIMENSIONAL"

AUTOR: ALMONACID MANSILLA, OSCAR MANUEL

PROFESOR GUÍA: Parra Márquez, Juan Carlos

Proyecto de Título presentado en conformidad a los requisitos para obtener el Título de Ingeniero Civil en Informática

Concepción, septiembre 2007

Resumen

El objetivo general de este proyecto es la construcción de una aplicación que permita realizar simulaciones en tres dimensiones del flujo de tráfico urbano sobre intersecciones señalizadas por semáforos (flujo interrumpido) y obtener como resultado de estas simulaciones los tiempos de espera a los que se ven sujetos los conductores en dichas intersecciones así como también observar visualmente el comportamiento de los vehículos de forma individual y como conjunto, el sistema en sus partes y como un todo.

El presente informe se divide en dos partes, la primera parte aborda la investigación sobre los temas del proyecto, con la finalidad de tener un conocimiento más acabado del problema. La segunda parte del informe trata sobre la definición de requerimientos y diseño de la aplicación que se construirá.

Para poder abordar el tema de la simulación de tráfico urbano en 3d es necesario tener conocimiento sobre las materias de fondo que atañen al problema. Para el caso puntual del proyecto, las materias caen dentro de las disciplinas de la *Ingeniería de Tráfico* o I*ngeniería del Transito*, la *Dinámica de Sistemas* y la S*imulación*, estas dos últimas muy interrelacionadas entre si.

En las primeras secciones del documento, se tratarán muy resumidamente las áreas dentro de estas disciplinas las cuales tocan directamente los objetivos del proyecto. Esto último con el afán de formalizar la investigación y crear una base sólida para el desarrollo del trabajo. Es así como se tratará brevemente el estudio del flujo tráfico, que comprende un área de análisis dentro de la Ingeniería de Tráfico, junto con el diseño y control del tráfico en las intersecciones. Así también la Dinámica de Sistemas aporta sus tecnicismos y formalismos en la creación del modelo que representa los factores más importantes en el comportamiento del flujo interrumpido. Modelo que es la base para lograr la simulación.

Luego de formalizar definiciones y terminologías se definen los factores o variables que se consideran de importancia para representar adecuadamente el comportamiento del flujo de tráfico, todo esto último pensado en la aplicación que se desarrollará durante la habilitación profesional (HP) y tomando como base los parámetros de entrada ocupados por un software comercial de simulación de tráfico llamado AISUM.

También como parte preliminar del presente informe se describen los modelos matemáticos que dan vida a la simulación digital, a saber, modelo de Gipps de "Vehículo Siguiente" y modelo "Influencia Señal-Intersección".

Un factor fundamental, sin considerar los modelos para la simulación, es la determinación y selección de las tecnologías que se integrarán para lograr concretar el proyecto, esto es también abordado en la parte primera del informe. Se realiza una sobrevista muy general sobre estas tecnologías o herramientas para tratar de comprender sus objetivos y aportes al proyecto. Es así como se trata brevemente Ogre3d un motor gráfico que soporta la tarea de la visualización tridimensional, PhysX utilizado para las simulaciones al nivel de física newtoniana para dar a la simulación mayor realismo, NxOgre que es el envoltorio encargado de unir el poder entregado por Ogre3D y PhysX de forma transparente para el desarrollador.

La segunda parte del informe de habilitación profesional considera netamente la ingeniería de software. La definición de requerimientos basado en el estándar de definición de requerimientos ERS, el diseño del simulador basado en el estándar de diseño, utilizando una metodología hibrida entre cascada con reducción de riesgos en las etapas de diseño y construcción y OMT (en la fase de diseño).

Como es de esperar las conclusiones sobre el desarrollo del proyecto cierran el informe de HP.

TABLA DE CONTENIDOS

PARTE I GENERALIDADES

CAPITULO 1. Introducción

En la búsqueda de temas para realizar la propuesta del Proyecto de Título y Habilitación Profesional, en áreas relacionadas con la dinámica de sistemas y simulación, se encontró en la Red una propuesta elaborada por el departamento de "Lenguajes y Sistemas Informáticos" de la Universidad de Granada España, también en la línea proyecto de titulación para sus alumnos informáticos, referente a la simulación en tridimensional del tráfico urbano. Esta misma propuesta fue presentada al profesor Juan Carlos Parra, docente jornada completa de nuestro departamento, quien estimó que se podía desarrollar como Proyecto de Título (PT) y posterior Habilitación Profesional (HP).

1.1 Justificación

Los factores de estabilidad económica que se viven en la actualidad en Chile, el crecimiento del poder adquisitivo de sus habitantes, sumado a la baja registrada en los precios de automóviles nuevos y sobre todo en los usados, hacen que el parque automotriz nacional crezca año a año de manera sostenida. Según datos del I.N.E. el comportamiento del parque automotriz (vehículos motorizados) para los periodos comprendidos entre los años 2002 y 2005 es el que se resume en la tabla siguiente.

Tabla N° 1.1 Comportamiento parque automotriz

Año	Parque Automotriz	Crecimiento (%)
2002	2.164.540	-
2003	2.195.878	1.43
2004	2.298.620	4.47
2005	2.444.571	5.97

Fuente: Adaptado de INE "Parque de Vehículos en Circulación 2005"

La Octava Región del Bío-Bío ocupa el tercer lugar a nivel nacional en cuanto a la cantidad de vehículos en circulación con cerca del 10% del total después de la Región de Valparaíso que posee aproximadamente un 11.1%, siendo la Región Metropolitana, como era de esperar, la región con mayor número de vehículos con un 42.7% del total nacional de vehículos en circulación.

A pesar de los esfuerzos que se realizan para modificar la infraestructura vial para que sea capaz de soportar este crecimiento y entregar un flujo expedito en la circulación de vehículos, muchas de las ciudades sufren del colapso en sus sistemas viles producto del alto flujo vehicular, en el mejor de los casos sólo en ciertos horarios, y la insuficiente capacidad soportada por la infraestructura existente. Todo lo cual se traduce en tiempos de espera y recorridos muy amplios por parte de los conductores quienes se ven demorados en sus trayectos. Los ya conocidos y no deseables “Tacos”.

Por lo anteriormente expuesto sería de utilidad contar con una herramienta que pudiera simular el flujo de vehículos en intersecciones complejas en donde se registran atochamientos. Y poder ver cuales son los resultados de la aplicación de planes dirigidos a la solución de estos problemas. Poder identificar por ejemplo como afecta al tiempo de espera, de los conductos que circulan por la intersección, una determinada configuración en los tiempos de los semáforos.

Si a esto se agrega un entorno tridimensional en donde se pueda observar la interacción de los vehículos con el medio y el comportamiento del flujo vehicular de una manera aproximada a la realidad, las personas encargadas de tomar y aplicar las decisiones de los planes de mejora tendrían una mayor comprensión producto de un mejor análisis del problema y sus soluciones.

1.2 Objetivos de Estudios Preliminares

1.2.1 Objetivo General

Desarrollar una aplicación que permita realizar simulaciones tridimensionales del comportamiento del tráfico vehicular en zonas urbanas, intersecciones señalizadas.

1.2.2 Objetivos del Estudio

- Investigar sobre las áreas que involucra el objetivo del proyecto.
- Conocer las distintas dinámicas del tráfico urbano en intersecciones señalizadas, como comportamiento y componentes.
- Seleccionar los modelos de simulación necesarios que permitan el comportamiento deseado de la aplicación (simulaciones apropiadas).
- Definir los parámetros principales con los cuales trabajará la aplicación.
- Seleccionar las tecnologías adecuadas para el desarrollo del simulador.

CAPITULO 2. Resumen Teórico De Las Disciplinas Involucradas

2.1 Ingeniería de Tráfico

La ingeniería del tráfico es una sub-área de la *Ingeniería de Transporte*.

Ingeniería de Transporte: Es la aplicación de tecnologías y principios científicos a la planificación, diseño funcional, operación y manejo de facilidades para cualquier modo de transporte en orden para proveer seguridad, rapidez, comodidad, oportunidad, económica y ambientalmente compatible con el movimiento de personas y bienes [Ref01].

Ingeniería de Tráfico: Según W. R. Blunden [Ref01]. Es la ciencia de medir el tráfico y viajes, el estudio de las leyes básicas relativas a la generación de flujo de tráfico, y la aplicación de este conocimiento a la práctica profesional de la planificación, diseño, y operación de sistemas de tráfico para lograr seguridad y eficiencia en el movimiento de personas y bienes.

La práctica de la ingeniería de tráfico puede ser dividida en cinco grandes áreas funcionales, estas son.

i) Estudio de las características del Tráfico. Estos estudios son conducidos para obtener datos de transporte y tendencias de tráfico para regiones completas, y condiciones de tráfico para localidades específicas. Los estudios más comunes cubren las siguientes áreas.

- Factores humanos y vehiculares.
- Indicadores del flujo de tráfico (volumen), velocidad, tiempo de viaje y demora.

- Patrones de flujo de peatones.
- Patrones de viaje, factores generadores de viajes, origen y destino.
- Demanda y uso de estacionamientos y facilidades de cargado de camiones.
- Análisis de accidentes.

ii) Planificación de transporte. La aplicación de los conocimientos a la planificación.

Continua y comprensiva planificación de estudios de transporte para guiar el desarrollo de las facilidades de transporte que encontraran los objetivos y estándares deseados por la comunidad.

- Planes para especificar las facilidades de desarrollo o mejoras.
- Estudios de impacto medioambiental.
- Investigación de los factores subyacentes del sistema de transporte y el comportamiento de los usuarios de ese sistema.

iii) Diseño de Geometrías. La aplicación del conocimiento al diseño.

Diseño de características geométricas para nuevas autopistas, basado en el análisis de la ingeniería de tráfico.

- Rediseño de las carreteras e intersecciones existentes para incrementar sus capacidades.
- Diseño de estacionamientos y terminales.
- Establecimiento de estándares para la subdivisión de diseño de entrada de autos, controles de acceso.

iv) Operación y Control de Tráfico. El control de tráfico se dirige a través de las leyes del

transito nacionales y ordenanzas locales.

v) Administración

- Mantener el inventario de la infraestructura de transporte y control de dispositivos.
- Planificación administrativa, presupuestos, necesidades del personal, estructura organizacional.

Después de conocer los cinco ámbitos de estudio de la ingeniería de tráfico, se puede apreciar que los temas tocantes al proyecto están dentro de dos de estos, *el estudio de las características del tráfico* y *el diseño de geometrías*. Más puntualmente en:

- Factores humanos y vehiculares.
- Indicadores del flujo de tráfico (volumen), velocidad, tiempo de viaje y demora.
- Rediseño de las carreteras e intersecciones existentes para incrementar sus capacidades.

2.1.1 Características y Componentes del flujo de Tráfico.

Existen dos tipos de flujo, el flujo ininterrumpido y el flujo interrumpido.

Flujo Ininterrumpido: Corresponde al flujo en que los vehículos que atraviesan una sección de camino no requieren detenerse por ninguna causa externa al flujo de tráfico, como los dispositivos de control de tráfico. Los principales análisis de este tipo de flujo son: El flujo, la velocidad, la densidad, el flujo en los embotellamientos, apelotonamiento que es cuando uno o más vehículos se desplazan a muy baja velocidad.

Flujo Interrumpido: Este es el tipo de flujo que se busca simular. Es aquel flujo interrumpido constante y periódicamente por características externas, principalmente por los dispositivos de control de tráfico. Los principales análisis para este tipo de flujo son sobre: Intersecciones controladas con señales, aplicación de teoría de colas,

apelotonamiento cuando los vehículos salen de una intersección.

2.1.1.1 Características del Flujo de Tráfico

Los factores críticos del sistema de tráfico son: Los conductores y peatones, los vehículos, los caminos y los dispositivos de control. Para una mejor comprensión de esto último se procederá a dar una breve descripción de cada uno de estos factores.

Características Humanas.

Los conductores y peatones son el principal elemento en el tráfico de carretera y tienen que ser comprendidos para que posteriormente puedan ser guiados y controlarlos de la mejor manera. La conducta individual de un conductor en el flujo de tráfico es frecuentemente el factor que determina las características de ese tráfico.

***Tiempo de percepción-reacción*.** Es el factor más usado para caracterizar a los conductores. El tiempo de percepción incluye la detección, identificación, y elementos de decisión comprendidos en respuesta a los estímulos. El tiempo de respuesta es el tiempo que lleva iniciar la respuesta física.

El procesamiento humano de la información puede ser visto como un sistema computacional en donde las vías de entrada (input) son los sentidos, encargados de la recepción de la información. El cerebro correspondería a la unidad central de procesamiento que a través de la percepción interpreta la información que capturan los sentidos, para luego dar pasó al intelecto que razona y resuelve el problema y por último al control de movimiento que son las instrucciones dadas por el cerebro a las partes del cuerpo como respuesta al problema (output).

La información que llega a través de los sentidos, puede hacerlo por medio de visión (agudeza visual), audición, y la estabilidad sensorial que es cuando los conductores reaccionan al sentir inestabilidad como caminos ásperos, curvas agudas o cuestas

pronunciadas.

Según Francisco Navarro [Ref12], el tiempo de percepción para un conductor normal en Chile es de ¾ de segundo al igual que el tiempo de reacción o respuesta, lo que hace un tiempo de percepción-reacción de 1,5 segundos, para un conductor normal en Chile.

Características de los Vehículos.

Las características y desempeño de los vehículos juegan un rol fundamental en la ingeniería de tráfico. Las dimisiones determinan el diseño de las geometrías y estructuras de las carreteras y estacionamientos. El rendimiento de los vehículos considerados en conjunto con el desempeño de los conductores determinan las características del flujo de tráfico y su seguridad.

Rendimiento de los vehículos esta dado por:

***Potencia*.** Define la habilidad del vehículo para acelerar, mantener la velocidad y subir pendientes.

Aceleración. Los vehículos se diferencian notablemente en sus relaciones peso energía y de esta manera en su aceleración. Esto se ve mas claramente entre los autos de pasajeros y los camiones.

Frenado. Cuando un vehículo desacelera, este pierde energía cinética. La energía esta siendo expendida por la fricción dentro de los frenos.

Desaceleración. Corresponde a la acción contraria a la aceleración.

En Gonzáles [Ref02], se encontró información sobre los datos de aceleración, desaceleración y dimensiones de los vehículos agrupados en tres conjuntos que son: los vehículos livianos, camiones simples, camiones remolques y semiremolques. La

información se muestra agrupada por las categorías en las siguientes tablas.

Tabla N° 2.1Características de los Vehículos Livianos

Parámetro	Media	Desviación	Mínimo	Máximo
Longitud (m)	4.32	0.30	3.73	4.98
Ancho (m)	2.00	0.00	2.00	2.00
Máx. Aceleración (m/s2)	2.72	0.34	2.13	3.61
Normal desaceleración (m/s2)	4.00	0.00	4.00	4.00
Máx. Desaceleración (m/s2)	8.00	0.00	8.00	8.00

Tabla N° 2.2 Características de los Camiones Simples

Parámetro	Media	Desviación	Mínimo	Máximo
Longitud (m)	7.50	2.00	6.00	10.00
Ancho (m)	2.300	0.50	1.90	3.00
Máx. Aceleración (m/s2)	1.00	0.50	0.60	1.80
Normal deceleración (m/s2)	3.50	1.00	2.50	4.80
Máx. Deceleración (m/s2)	7.00	1.00	5.50	8.00

Tabla N° 2.3 Características de los Semiremolques y Remolques

Parámetro	Media	Desviación	Mínimo	Máximo
Longitud (m)	15.00	2.00	8.00	20.00
Ancho (m)	2.300	0.50	1.90	3.00
Máx. Aceleración (m/s2)	0.50	0.80	0.40	1.80
Normal deceleración (m/s2)	2.00	2.00	1.50	4.80
Máx. Deceleración (m/s2)	5.00	2.00	4.50	8.00

C) Caminos

Las características de los caminos más relevantes están relacionadas con la detención y el progreso.

Visión de distancia. Es la longitud de un camino en la cual el conductor puede ver en

cualquier tiempo dado un objeto sobre su ruta de recorrido.

Visión distancia de frenado. La visión de la distancia de frenado, para propósitos de diseño, es usualmente tomada como el minimito de visión de distancia que requiere un conductor para detener el vehículo después de observar un objeto en su ruta sin impactarlo.

Visón distancia de decisión. Esta vista de distancia es la más extensa. Se define como "La distancia requerida por un conductor para detectar lo inesperado, las situaciones difíciles en el camino, amenazas potencial, y seleccionar una velocidad y trayectoria adecuadas para terminan de ejecutar las maniobras con seguridad".

Como el proyecto está enfocado a intersecciones de calles señalizadas por semáforo, las características de los caminos que son relevantes son las que comprenden el diseño de las intersecciones. Para esto, se utiliza como referencia el manual de demarcaciones de Vialidad [Ref13] que define las características y normas bajo las cuáles deben diseñarse las intersecciones en cuanto a señalización. Según este manual y de manera muy resumida las líneas longitudinales que separan las pistas de una vía, deben seguir las siguientes consideraciones.

"Las líneas longitudinales se emplean para delimitar pistas y calzadas; para indicar zonas con y sin prohibición de adelantar; zonas con prohibición de estacionar; y, para delimitar pistas de uso exclusivo de determinados tipos de vehículos, por ejemplo, pistas exclusivas de bicicletas o buses".

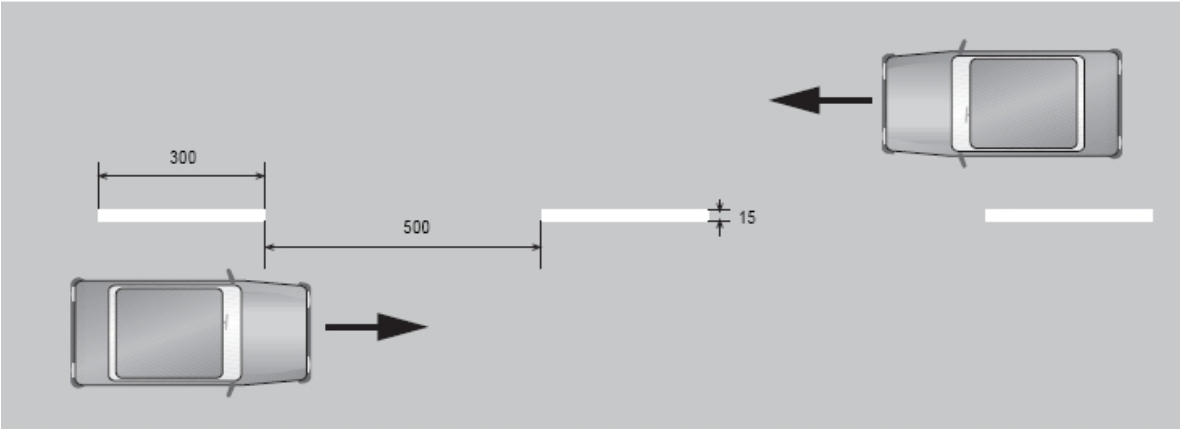


Figura Nº 2.1 Marcación Longitudinal - *Dimensiones en centímetros*

Tabla N° 2.4 Consideraciones para la demarcación longitudinal de las pistas

Velocidad Máxima de la Vía (km/h)	Ancho de la Línea (cm)	Patrón Relación (m)	Demarcación Brecha
Mayor a 60	15 mínimo	8 ó12	1 a 3 ó 3 a 5
Menor o igual a 60	10 mínimo	5 u 8	2 a 3 ó 3 a 5

La tabla anterior muestra las consideraciones que se deben tomar al momento de demarcar longitudinalmente las pistas de una calle, esto depende principalmente de la velocidad con la que circulan los vehículos por la pista. Como el proyecto aborda una intersección urbana la velocidad máxima permitida en Chile para vehículos livianos particulares es de 60 Km/h. Se considera entonces una demarcación para velocidades menores o iguales a 60 Km/h.

Según el mismo manual de demarcaciones, una intersección señalizada por semáforo debería tomar las siguientes consideraciones en su señalización.

"La demarcación transversal de un cruce regulado por semáforo esta compuesta por una Línea de Detención Continua y un Paso Peatonal.

La línea de detención indica al conductor que enfrenta la luz roja de un semáforo, el lugar más próximo al cruce donde el vehículo debe detenerse. Deben ubicarse a no más de 2 m del lugar donde se ubica el poste que sustenta la lámpara del semáforo".

La siguiente figura proporcionada por el manual de demarcación grafica lo dicho en el párrafo anterior y muestra como debería lucir una intersección, considerando la ubicación del semáforo, las líneas longitudinales y transversales.

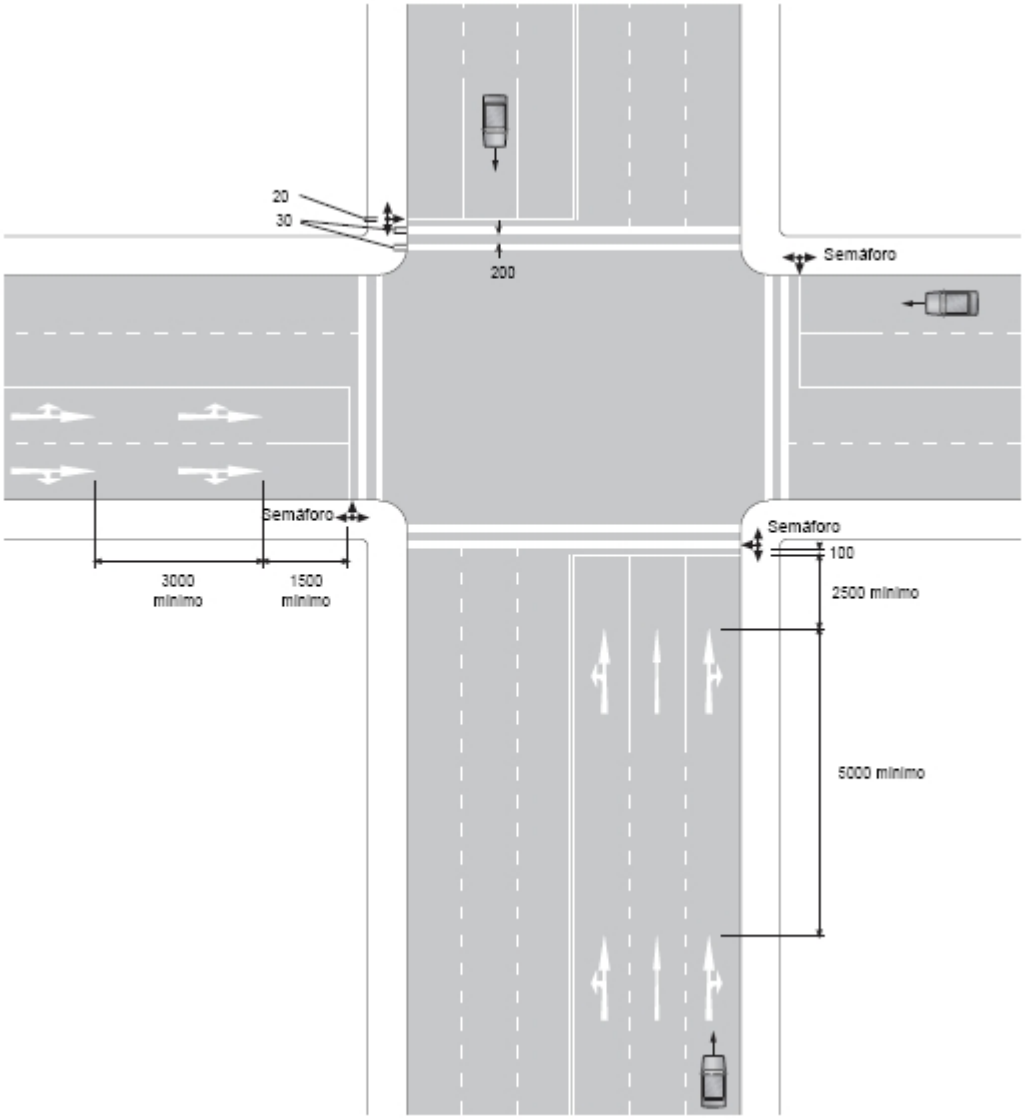


Figura Nº 2.2 Intersección regulada por semáforo - *Dimensiones en centímetros*

Siguiendo con la referencia al manual de demarcaciones el ancho de las pistas debe considerar la siguiente tabla.

Tabla N° 2.5 Consideraciones para los anchos de pistas

Velocidad Máxima de la Vía (km/h)	**Ancho de la Pista (m)**
Igual o superior a 70	Entre 3,5 y 3,8
50 ó 60	Entre 3,0 y 3,5
Menor a 50	Entre 2,5 y 3,0

E) Dispositivos de control

Los dispositivos de control de tráfico se comunican con los conductores, y deben seguir las normativas y reglamentaciones definidas por las autoridades. Muchos de estos controles son simples (marcas en el camino) a pesar de aquello los conductores pueden recibir el mensaje que se les esta transmitiendo a través de los colores y modelos de estos controles.

Las señales están estandarizadas según: color, forma, figura, símbolos, ubicación, tamaño y formato del mensaje de modo que los conductores tengan un alto grado de atención y comodidad en su lectura. Así también los dispositivos de señalización se estandarizan de la misma manera, incluyendo las secuencias de sus indicaciones.

2.1.1.2 Componentes del Flujo de Tráfico

La teoría de flujo de tráfico implica el desarrollo de relaciones matemáticas entre los principales componentes del flujo de tráfico que corresponden a: flujo, densidad y velocidad (otro elemento asociado con la densidad es el gap o headway (avance) entre dos vehículos en un flujo de tráfico). Estas relaciones ayudan a la ingeniera de tráfico en la planificación, diseño y evaluación de las medidas implementadas en los sistemas carreteros. La teoría de flujo de tráfico es utilizada para diseñar los largos adecuados de las pistas, los tiempos promedios de espera en las autopistas e intersecciones, para mejorar el funcionamiento del sistema existente, realizar mejoras en los dispositivos de control entre otras.

Otra aplicación importante de la teoría del flujo de tráfico es la simulación, en donde los algoritmos matemáticos son utilizados para estudiar la compleja interrelación que existe entre los componentes del flujo tráfico y para estimar los efectos de los cambios en estos.

A continuación se definen los componentes del flujo de tráfico antes mencionados.

***Flujo*:** *q* es definido simplemente como el flujo de vehículos, *n*, que pasa por un punto en una carretera en un intervalo o duración de tiempo *t.*

$$q = \frac{n}{t} \quad \text{(Ec. 2.1)}$$

Un ejemplo de unidades para *q* puede ser vehículos por hora.

El promedio de la velocidad del tráfico esta definido dedos maneras:

Velocidad media-tiempo: Que corresponde a la media aritmética de las velocidades observadas en algún punto designado de la carretera. Esta se conoce como velocidad media-tiempo, $\overline{u}_t$, y se expresa como:

$$\overline{u_t} = \frac{1}{n}\sum_{i=1}^{n} u_i \quad \text{(Ec. 2.2)}$$

Donde u_i es la velocidad del iésimo vehículo.

Velocidad media-espacio: Es la segunda forma de definir la velocidad media y es la más útil en el contexto del análisis de tráfico, se determina en base al tiempo necesario de un vehículo para atravesar una sección o longitud conocida de la carretera, *l.* Esta medida se denota por *u* y se expresa de la siguiente manera:

$$u = \frac{(1/n)\sum_{i=1}^{n} l_i}{\overline{t}} \quad \text{(Ec. 2.3)}$$

Donde, l_i es el largo usado de la carretera para medir la velocidad del vehículo *i*, y

$$q = \frac{n}{\sum_{i=1}^{n} u_f} \quad \text{(Ec. 2.4)}$$

Donde $t_n(l_n)$ es el tiempo necesario para que el vehículo n atraviese la sección de la carretera de largo *l*. Si todas las velocidades de los vehículos son medidas sobre el mismo largo de carretera, L = $l_1 = l_2 = \ldots l_n$. En tal caso la ecuación quedaría como:

$$u = \frac{1}{(1/n)\sum_{i=1}^{n}\left[1/(L/t_i)\right]} \quad \text{(Ec. 2.5)}$$

La que corresponde a la media armónica de la velocidad usada en los modelos de tráfico.

***Densidad*:** La densidad del tráfico, *k*, se refiere al número de vehículos ocupando algún largo de la carretera en algún momento específico, y se define como:

$$k = \frac{n}{l} \quad \text{(Ec. 2.6)}$$

Tiempo de Headway (tiempo de avance): Es el tiempo entre al paso de la parte frontal de un vehículo por un punto designado en la carretera y el paso de la parte frontal del vehículo siguiente por el mismo punto. Se denota por *t*.

$$t = \sum_{i=1}^{n} h_i \quad \text{(Ec. 2.7)}$$

Donde h_i es el tiempo headway para el vehículo pésimo (el tiempo que transcurre entre el arribo del vehículo *i* e *i-1*). Substituyendo en Ec.1

$$q = \frac{n}{\sum h_i} \quad \text{(Ec. 2.8)}$$

o

$$u = u_f \left(1 - \frac{k}{k_j} \right) \quad \text{(Ec. 2.9)}$$

Donde $\bar{h}$ es el promedio de headway ($\sum h_i / n$).

Espacio Headway (espacio de avance)*: d*, es la distancia entre el frente de un vehículo y el frente del vehículo siguiente.

Gap: Es el avance en una pista con mayor flujo, el cual es evaluado por el conductor del vehículo que se encuentra en un flujo menor y que desea unirse a la pista con mayor flujo. El gap es expresado en unidades de tiempo (tiempo gap) y en unidades de espacio (espacio gap).

2.1.2 Control de Intersecciones

El propósito del control de tráfico es asignar la derecha del camino al conductor, y así de esta forma facilitar la seguridad vial asegurando el movimiento ordenado y fiable de todo el tráfico en la carretera. El control puede ser logrado usando señales de tráfico, signos o marcas que regulen, guíen el tráfico.

Para el control de los conflictos en el flujo vehicular en las intersecciones se utilizan varios métodos diferentes. La selección de uno de estos métodos depende del tipo de intersección y el volumen del tráfico de cada flujo conflictivo. A continuación se describen los tipos de control de intersecciones más importantes.

Signo Ceda el Paso: Los conductores que se aproximen a este signo, necesitan disminuir la velocidad. La detención para este signo no es mandataria pero deben hacerlo cuando sea necesario para no interferir con el tráfico que tiene el camino.

Signo Pare: Este signo es usado cuando se requiere que un vehículo que se aproxima a una intersección se detenga. Este signo pare no debe ser utilizada en autopistas. Pueden ser utilizados en caminos menores que tengan intersección con caminos más grandes (de alta velocidad).

Signo Canalización de Intersecciones: Es usado para separar los carriles de los caminos, aquellos que son utilizados para doblar o seguir la misma dirección.

2.1.2.1 Semáforos

Los semáforos son parte importante dentro del sistema del flujo vehicular interrumpido ya que una adecuada programación o configuración de los semáforos ayuda a conseguir un flujo relativamente expedito y mantiene los tiempos de espera de los conductores en niveles aceptables, mientras que una inadecuada configuración produce congestiones y tiempos de espera elevados. El funcionamiento eficiente de los semáforos requiere de una adecuada temporización de las diferentes indicaciones de colores (luces de semáforos), esto se obtiene a través de la implementación de modelos para configuración de semáforos. No es un objetivo del proyecto crear modelos o realizar optimizaciones en la programación de los semáforos, sin embargo los resultados entregados por modelos que ya desarrollados o estén bajo desarrollo pueden ser simulados por la aplicación para observar cual es el comportamiento del sistema.

A continuación algunas definiciones necesarias para entender el funcionamiento de semáforos y poder trabajar con ellos.

Controlador (semáforo): Es un dispositivo de señalización de tráfico, que cambia los colores de las luces señalizadoras de tráfico según un plan fijo o variable.

Ciclo (duración del ciclo): Es el tiempo en segundos requerido para completar una secuencia de señales indicadoras de colores.

Fase (fase de señal): Es la parte de un ciclo asignada a una corriente de tráfico, o a una combinación de una o más corrientes de tráfico**.**

Intervalo: Es cualquier parte en la duración de ciclo en la cual la señal indicadora no cambia.

Desplazamiento: Es el lapso de tiempo en segundos o el porcentaje de la duración del ciclo entre el comienzo de una fase de luz verde en una intersección y el comienzo de la correspondiente fase de luz verde en la siguiente intersección. Este es el tiempo base de los sistemas de control.

Intervalo de cambio y despeje: Es la duración total del tiempo en segundos de las señales indicadoras de luz amarilla y roja. Es el tiempo que se provee a los vehículos para despejar la intersección después del intervalo de luz verde, antes de que se lance en movimiento el flujo en espera.

Intervalo de luz roja: Es el tiempo desplegado en una indicación de rojo para todos los que se aproximan. Es algunas veces usado exclusivamente para el cruce de peatones o para permitir a vehículos y peatones limpiar intersecciones muy largas antes de dar la indicación verde al flujo opuesto.

Factor de hora peak (FHP): Es la medida de la variabilidad en la demanda durante la hora peak. Es el indicador del volumen durante la hora peak para la máxima tasa de flujo durante un periodo de tiempo dado dentro de la hora peak. Para las intersecciones el periodo de tiempo usado es de 15 minutos y el FHP es dado como:

$$FHP = \frac{vhp}{4 * vdp} \quad \text{(Ec.2.10)}$$

Donde:

vhp: volumen durante la hora peak.

vdp: volumen durante peak en 15 minutos dentro de la hora peak

2.1.2.2 Temporización de Señales para Intersecciones Incomunicadas

Una intersección incomunicada es aquella en la cual los tiempos de las señales no están coordinados con las otras intersecciones y por lo tanto opera de forma independiente. La duración del ciclo para una de estas intersecciones debe ser corto, preferentemente entre 35 y 60 segundos, aunque puede ser necesario usar ciclos largos cuando los volúmenes de aproximación son muy altos. Si embargo la duración del ciclo debe ser mantenida bajo los 120 segundos, ya que ciclos muy largos resultaran en excesivas demoras. Varios métodos se han desarrollado para de terminar el ciclo de una intersección, y en la mayoría de los casos, el intervalo de amarillo se considera como componente del tiempo de intervalo verde.

Intervalo Amarillo. El propósito principal de la indicación de luz amarilla después de la luz verde es alertar a los motoristas del hecho que esta cercano el cambio a la luz roja y permitir a los vehículos estar alerta para cruzar la intersección. Una mala elección del intervalo de las luz amarilla puede llevar a la creación de una *zona de dilema*, que es un área cercana a una intersección en la cual ningún vehículo puede detenerse en forma segura antes de una intersección ni aclarar, o despejar, la intersección sin acelerar antes de llegada de la luz roja. El intervalo de luz amarilla requerido es el periodo de tiempo que garantiza a un vehículo que se aproxima detenerse de forma segura o continuar a través la intersección sin acelerar.

2.2 Dinámica De Sistemas

2.2.1 Definiciones

Antes de introducir a la dinámica de sistemas es importante dar claridad al significado de las palabras que la componen, para esto, a continuación se definen brevemente.

Sistema: Una definición aceptable para sistema puede ser dada como: Un objeto dotado de alguna complejidad, formado por partes coordinadas, de modo que el conjunto posea una unidad que es justamente el sistema. Se entiende como sistema a una unidad cuyos elementos interaccionan juntos, ya que continuamente se afectan uno a otros, de modo que operan hacia una meta común. El sistema es algo que se percibe como una entidad que se distingue de lo que lo rodea y es capaza de mantener esa identidad a lo largo del tiempo y bajo entornos cambiantes.

Para que un sistema pueda ser definido como tal debe cumplir las siguientes condiciones.

- Existe, o tiene, un fin.
- Existe un conjunto de cosas o normas.
- Tal conjunto esta ordenado.

Dinámica: El término dinámica es empleado en oposición al término estática, y se quiere con él expresar el carácter cambiante de aquello a lo que es referido con este término.

Al hablar entonces de la dinámica de un sistema, se quiere hablar de los cambios que se producen a lo largo del tiempo en las distintas variables que pueden ser asociadas a sus partes (sistema), como consecuencia de las interacciones que se producen entre ellas.

La dinámica de sistemas se utiliza para analizar como las relaciones en el seno de un

sistema permiten explicar su comportamiento. Ya que los cambios producidos en un sistema son el reflejo, en alguna medida, de las interacciones que tienen en su seno.

2.2.2 Lenguaje Elemental Para Describir Sistemas

La descripción mínima de un sistema viene dada por la especificación de las distintas partes que lo forman. De este modo se obtiene la descripción más elemental que se pueda tener de ese sistema, que se limita a establecer que partes lo forman y cuales de ellas se influyen entre si.

El conjunto de relaciones entre los elementos de un sistema recibe la denominación de *estructura del sistema* y se representa mediante el *diagrama de influencia o causal*.

En su forma más simple el diagrama causal esta formado por lo que se conoce como grafo orientado. En donde los nodos representan a los elementos y las flechas o aristas representan la influencia de un elemento sobre otro. A estas aristas se le puede asociar un signo. Este signo indica si las variaciones del antecedente y del consecuente son, o no, del mismo signo. A modo de ejemplo ver la siguiente figura.

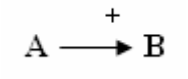


Figura N° 2.1 Lenguaje elemental de sistemas

El signo positivo sobre la arista de la figura 2.1 quiere decir que si **A** se incrementa, lo mismo sucederá con **B.** Y en caso contrario si **A** disminuye así mismo lo hará **B**. Por otra parte si la influencia entre los dos elementos **A, B** fuese negativa, un incremento en el primero implicaría de una disminución en el segundo, y viceversa.

2.2.2.1 Bucle de Realimentación Negativa

Un bucle de realimentación negativa, representa un tipo de situación muy frecuente en el que se trata de decidir acciones para modificar el comportamiento con el fin de alcanzar un determinado objetivo.

No se profundizará más sobre el tema de los bucles de realimentación negativa, ya que estos como se mencionó anteriormente buscan realizar modificaciones o implementar controles para lograr una meta u objetivo, que puede verse como optimización. Esto va mas allá de los objetivos del tema de proyecto, que no busca optimizar los flujos de tráfico, encontrar soluciones de control, sino que busca analizar el comportamiento de este en diferentes condiciones. Solo se mencionaron los bucles de retroalimentación negativa para dar más consistencia a la investigación.

2.2.2.2 Bucle de Realimentación Positiva

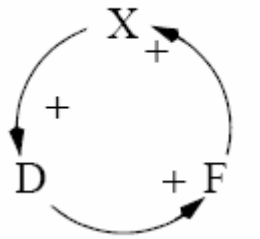


Figura N° 2.2 Realimentación positiva

La figura muestra una forma que puede adoptar un bucle de realimentación positiva. Esta figura muestra un bucle en que todas las influencias son positivas, pero pueden ser también negativas y en tal caso compensarse por pares. Esta figura representa un proceso en el que un estado determina una acción, que a su vez refuerza este estado, y así indefinidamente.

Si cualquiera de los elementos de este sistema, **X, D, F**, sufre una perturbación, ésta se propaga, reforzándose a lo largo del bucle. En efecto si **X** crece, entonces, en virtud del signo de influencia, lo hará **D**, lo que a su vez determinará el crecimiento de **F**, y de nuevo el de **X.**

Por lo tato la misma estructura del sistema hace que el crecimiento inicial de **X**, vuelva reforzado a **X**, iniciándose de este modo un proceso sin fin que determinará el crecimiento de **X.** esto es conocido como círculo vicioso o bola de nieve. El cambio se

amplifica produciendo más cambio.

2.2.2.3 Retrasos

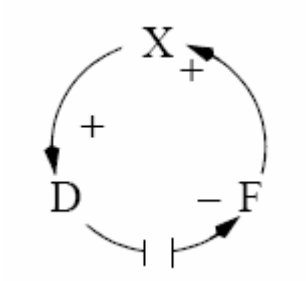


Figura N° 2.3 Retrasos

Las influencias de los elementos en un sistema o sobre otros elementos puede producirse de forma instantánea o de forma menos instantánea, como influencias que tardan mucho tiempo en manifestarse. En este último caso se tienen influencias a las que se asocian retrasos. Esto puede ser representado por la figura mostrada arriba, en donde la influencia de **D** sobre **F** tiene un retraso que se representa con una línea dividida en dos trozos.

2.2.3 Modelos de Sistemas

Uno de los objetivos principales de la dinámica de sistemas es la construcción de modelos precisamente para realizar posteriores simulaciones con ellos. Es por esto que se menciono anteriormente la cercanía entre estas dos disciplinas, ya que una constituye la base de la otra.

El término modelo tiene muchas acepciones, la acepción que interesa para la dinámica de sistemas es el modelo como sinónimo de representación. Más claramente, como la representación de un cierto aspecto de la realidad.

La dinámica de sistemas divide la construcción o desarrollo de modelos en tres fases. Fase de ***Conceptualización, Formulación*** y fase de ***Análisis y Evaluación.*** En la fase de conceptualización se describe el sistema en forma verbal, se define el modelo en el tiempo y se crean los diagramas causales mostrando como se influencias los diferentes

componentes del sistema. En la fase de formulación se crean los modelos matemáticos (ecuaciones) que representan el comportamiento del sistema en el tiempo. Y por último en la fase de análisis y evaluación, como su nombre lo indica se realizan análisis comparativos, de sensibilidad y se evalúa la implementación del modelo.

2.3 Simulación

La simulación es una área muy cercana a la dinámica de sistemas, que se describió brevemente en la sección anterior, es más, la dinámica es un componente fundamental de esta área de investigación ya que es a través de ella que se logran formular y validar los modelos que serán la representación de la realidad para el problema en estudio.

Para formalizar el concepto de simulación, se toma la siguiente definición.

Simulación: La simulación es la representación de un proceso o fenómeno mediante otro más simple, que permite analizar sus características. A través de la simulación se generan los posibles estados del sistema. La simulación tiene como objetivo predecir el comportamiento de los sistemas.

Las simulaciones nos entregan una serie de ventajas, entre las cuales se pueden destacar, tiempos bajísimos en el análisis del sistema, es decir, simular en segundos o minutos (con la ayuda de computadoras) el comportamiento de horas, días o meses.

En síntesis una de las mayores virtudes de los modelos de simulación es lograr entregar información detallada frente a los posibles cambios del sistema permitiendo anticipar soluciones correctas a los problemas reales [Ref02].

2.3.1 Simulación del Transporte (Tráfico)

La simulación de los modelos de transporte o de tráfico, esta clasificada en tres tipos de simulaciones, cada tipo se define según el nivel de detalle que se le quiera dar a la simulación y que se proceden a describir a continuación.

Simulación Macroscópica: Describe entidades y sus actividades e interacciones con bajo nivel de detalle. Ejemplos de esto, puede ser el que los flujos son representados como un promedio, considerando densidades y velocidades. Las maniobras de cambio de pista no son consideradas, sino que son incorporados los vehículos a los arcos suponiendo una correcta distribución [Ref02].

Simulación Mesoscópica: En general representa las entidades con un alto nivel de detalle, pero describe sus interacciones y actividades con un detalle mayor que el anterior, pero menor que el siguiente (pelotones) [Ref02].

Simulación Microscópica: Considera las entidades del sistema así como sus relaciones detalladas. Este tipo de simulación llamada a veces la teoría del auto siguiente o la teoría del vehículo siguiente ("car following"), considera los espacios y velocidades entre los vehículos individuales. El líder y su seguidor.

El tipo de simulación que se abordara en el proyecto es la simulación microscópica ya que se pretende considerar o manejar las distancias y las velocidades de cada vehículo en relación a las condiciones del flujo de tráfico en que se estén trasladando.

2.3.2 Ventajas y desventajas de la Simulación

Como en todo orden de cosas existen ventajas y desventajas a la hora de aplicar un cierto enfoque en un estudio o un determinado plan de acción para conseguir mejoras en situaciones problemáticas. También la simulación tiene ventajas y desventajas cuando se pretende aplicar sobre un sistema complejo para estudiar su comportamiento. Las siguientes ventas y desventajas fueron tomadas de Dextre [Ref03].

Desventajas

- Los investigadores de modelos de simulación requieren conocimientos de muchas disciplinas (programación, estadística, ingeniería, etc.).

- La simulación no es posible si el investigador no entiende totalmente el sistema.
- Los modelos de simulación trabajan sobre supuestos y limitaciones preestablecidas.
- Si los modelos de simulación no son representaciones válidas del sistema bajo estudio, proporcionarán resultados con muy poca información aprovechable.

Ventajas

- Los sistemas más complejos del mundo real, no pueden ser descritos minuciosamente por un modelo matemático que pueda ser evaluado analíticamente. En estos casos, la simulación es el único tipo de investigación posible.
- Los modelos de simulación pueden ser usados con nuevas situaciones que actualmente no existen en el mundo real.
- Un sistema de simulación puede ser estudiado en tiempo real, de manera lenta o en forma acelerada, dependiendo de los requerimientos del investigador.
- En la simulación, se pueden comparar diferentes escenarios, con la finalidad de ver cuál es el que mejor se ajusta a los requerimientos.

CAPITULO 3. Modelos De Flujo Tráfico Para Simulación Por Computadora

La Ingeniería de Tráfico ya se ha encargado de desarrollar modelos para simular el complejo sistema de flujo vehicular. Existen varios modelos publicados por distintos expertos en el área de transporte algunos de estos modelos fueron desarrollados pensando en la realización de las simulaciones a través de computadoras. Como se mencionó al comienzo del documento, se considera como referencia para el desarrollo del proyecto a AISUM que es un software comercial para la simulación de tráfico. Este SW utiliza como modelo para sus simulaciones al "modelo de Gipps de vehículo siguiente" (*Peter Gipps*) publicado en 1981 para realizar simulaciones digitales. Sin embargo este modelo no considera el flujo vehicular interrumpido constantemente por las intersecciones señalizadas por lo que es necesario integrar un modelo complementario que cubra esta necesidad.

Los modelos de vehículos siguientes pueden ser separados en dos grupos de modelos, los modelos de "Respuesta a Estímulos" y los modelos de "Distancia Segura" o "Distancia de Seguridad". El primer grupo considera la aceleración del vehículo siguiente como una función entre la diferencia de velocidades, entre vehículo líder y el vehículo siguiente, y el tiempo de reacción del conductor. Los modelos de distancia segura consideran la aceleración del vehículo siguiente como una relación entre las diferencias de velocidades, del líder y perseguidor, y la distancia que hay entre ambos procurando siempre mantener una distancia mínima que evite la colisión entre ambos vehículos en caso de frenado brusco.

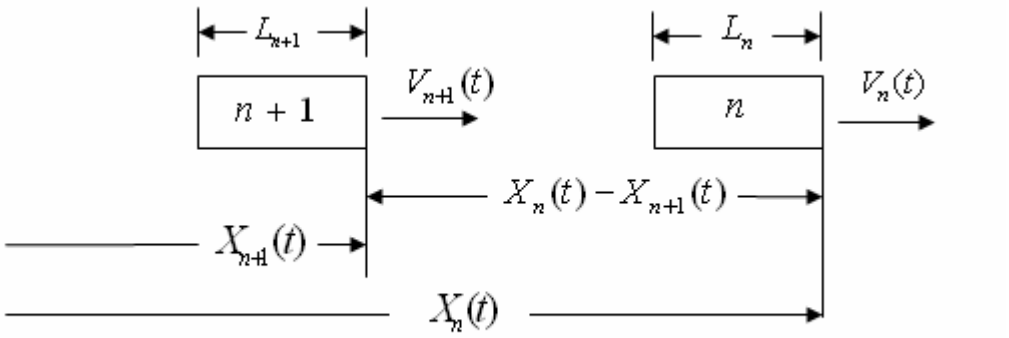
El modelo de Gipps corresponde al segundo de los grupos mencionados en el párrafo anterior.

3.1 Modelo De Gipps De Vehículo Siguiente

El modelo de Gipps de vehículo siguiente corresponde a un modelo de microsimulación que trata de estimar el comportamiento de los vehículos de manera individual y bajo las condiciones impuestas por el resto de los elementos que forman parte del sistema. Más puntualmente trata de estimar el comportamiento del vehículo siguiente según comportamiento del vehículo que viaja directamente delante de él, el vehículo líder.

El modelo de Gipps tiene dos variantes, una para flujo libre en donde no existe un vehículo líder, y una para flujo condicionado en donde si existe la presencia influyente de un vehículo líder.

Con el objetivo de dar claridad en la exposición del modelo servirá de apoyo la siguiente figura.



FiguraN° 3.1 Componentes Modelo de Gipps

Considerando que la dirección del flujo es de izquierda a derecha, n representa al vehículo líder con un largo de L_n y una velocidad en el instante t de $V_n(t)$. Mientras que $n+1$ representa al vehículo siguiente que posee un largo de L_{n+1} y lleva una velocidad $V_{n+1}(t)$ para el mismo instante de tiempo t. $X_n(t)$ es la posición del vehículo líder en el instante t, y así también $X_{n+1}(t)$ corresponde a la posición del vehículo siguiente para un

mismo t. La diferencia $X_n(t) - X_{n+1}(t)$ es la distancia existente entre los dos vehículos sin considerar el largo del vehículo líder. Teniendo en mente las consideraciones anteriores se expone a continuación el modelo de Gipps.

3.1.1 Gipps Flujo Libre

Esta variante del modelo corresponde al caso en el cual no existe un vehículo líder que vaya por delante o de existir un líder, la distancia que lo separa del vehículo siguiente es muy grande como para influir en el comportamiento de éste. De esta manera la aceleración que el vehículo siguiente tomará en un determinado instante de tiempo depende solo de sus propias capacidades y restricciones. Y está dada por el siguiente modelo matemático.

$$V_{n+1}(t+T) = V_{n+1}(t) + 2.5a_{n+1} * T * \left(1 - \frac{V_{n+1}(t)}{V_{n+1}^*}\right) * \sqrt{0.025 + \frac{V_{n+1}(t)}{V_{n+1}^*}}$$

(Ec. 3.1)

Donde:

$V_{n+1}(t)$: Velocidad del vehículo $n+1$ en el instante t.

a_{n+1} : Máxima aceleración del vehículo $n+1$. Esta dada por la capacidad del vehículo.

V_{n+1}^* : Velocidad de viaje deseada por el conductor del vehículo $n+1$.

T : Tiempo de reacción del conductor o intervalo de simulación. Para el proyecto en particular, se considera a T como tiempo de reacción siempre y cuando ocurra algún evento al cual reaccionar sino T toma el valor de tiempo del intervalo de simulación.

De esta manera entonces y suponiendo que el conductor no excede la velocidad

deseada de viaje (como se menciono anteriormente los modelos trabajan con supuestos y limitaciones preestablecidas en donde las reglas se cumplen), la velocidad del vehículo siguiente se incrementará para luego hacer a su aceleración más pequeña a medida que se alcanza la velocidad deseada, finalmente la aceleración se vuelve cero al lograrse ésta velocidad deseada permaneciendo en un estado de equilibrio [Ref03].

La máxima aceleración del vehiculo al igual que la velocidad de viaje deseada son constantes, de acuerdo con esto el modelo de Gipps libre podría ser representado por un diagrama causal simple.

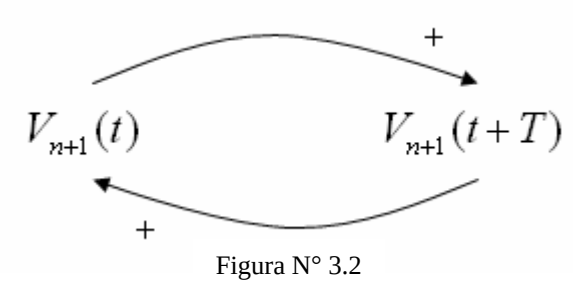


Figura N° 3.2

Según el diagrama, las velocidades tienen una influencia positiva sobre su consecuente. Al aumentar la velocidad del vehículo siguiente en un instante de tiempo t, se provoca como consecuencia un aumento de su velocidad para el instante de tiempo $t+T$. Si consideramos a los instantes de tiempo t y $t+T$ como estados inicial y final respectivamente y con un comportamiento cíclico en donde $t+T$ pasa a ser t en una nueva iteración del sistema, el aumento de la velocidad $V_{n+1}(t+T)$ conlleva también un aumento en $V_{n+1}(t)$. Cuando las velocidades no registran variaciones, no provocan tampoco variaciones en su consecuente.

3.1.2 Gipps Flujo Condicionado

El modelo de Gipps condicionado considera la presencia de un vehículo líder directamente delante del vehículo siguiente y que condiciona el comportamiento de éste último. Factores como la velocidad del líder, su desaceleración, la distancia entre ambos vehículos y el tiempo de reacción condicionan el comportamiento del vehículo siguiente y determinan la velocidad que éste desarrollará. El modelo matemático es el siguiente.

$$V_{n+1}(t+T) = d_{n+1}(t) + \sqrt{d_{n+1}^2 * T^2 - d_{n+1}\left[2*\{X_n(t) - S_n - X_{n+1}(t)\} - V_{n+1}(t)*T - \left(\frac{V_n^2(t)}{d_n}\right)\right]}$$

(Ec. 3.2)

Donde:

d_{n+1} : Es la máxima desaceleración que desea aplicar el conductor del vehículo siguiente. $d_{n+1} < 0$.

$X_n(t)$: Es la posición del vehículo n en tiempo t.

S_n : Corresponde al tamaño efectivo del vehículo n, esto es, el largo del vehículo líder L_n (ver figura 1) más una distancia o espacio que el vehículo siguiente no podrá invadir (una distancia de seguridad).

$X_{n+1}(t)$: Es la posición del vehículo siguiente en el momento t.

d_n : Es la desaceleración deseada por el vehículo líder. $d_n < 0$.

La diferencia bajo la raíz cuadrada y entre llaves $\{X_n(t) - S_n - X_{n+1}(t)\}$ corresponde a la distancia que existe en el momento t entre los dos vehículos considerando el tamaño efectivo del líder. Es ésta distancia la que determinará si el vehículo siguiente obtiene su velocidad desde la variante de flujo libre o de la condicionada. La interacción de

los elementos más importantes de este modelo se puede apreciar en el siguiente diagrama causal.

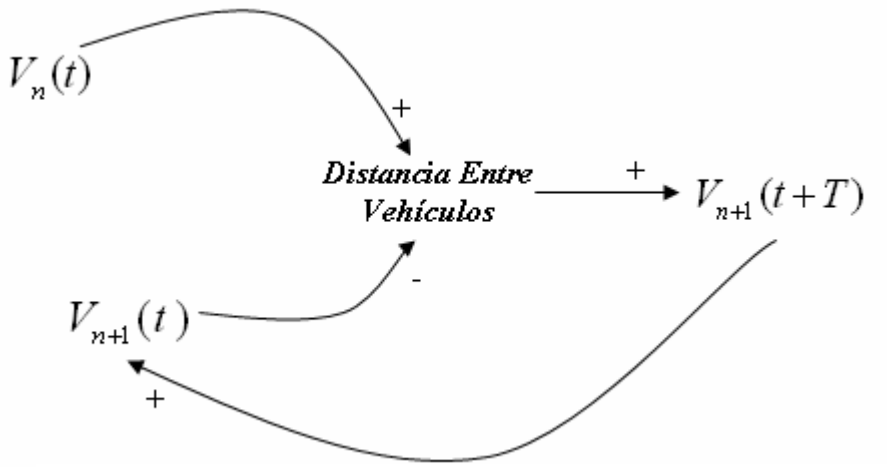


Figura N° 3.3 Elementos más importantes del modelo condicionado

Considerando las desaceleraciones de ambos vehículos como constantes se puede realizar un análisis simplificado del comportamiento del sistema.

Cuando la distancia entre ambos vehículos aumenta se produce un aumento de la velocidad del vehículo siguiente para el instante de tiempo $t+T$, si la distancia decrece también lleva a un decremento de la velocidad $V_{n+1}(t+T)$. El aumento de la distancia entre el líder y el vehículo perseguidor se provoca a causa de un aumento en la velocidad $V_n(t)$ del líder y/o también se produce por una disminución de velocidad por parte del vehículo siguiente. La interacción de los elementos de velocidad $V_{n+1}(t)$ y "Distancia Entre Vehículos" es de carácter negativa o inversa ya que al aumentar el valor del elemento antecedente, se gatilla una disminución en el consecuente y en el caso contrario al experimentarse una disminución en el antecedente, ocurre un aumento en el consecuente "Distancia Entre Vehículos".

3.1.3 Velocidad Definitiva Del Vehículo Siguiente Según Modelo De Gipps

Hasta aquí se ha expuesto que el modelo de Gipps posee dos variantes, una para el flujo libre y otra para el flujo condicionado. Las dos variantes calculan la velocidad que el vehículo siguiente tendrá en el instante de tiempo $t+T$, es decir, en el instante de tiempo en que el conductor comience a reaccionar después de ocurrido un evento en el momento t. Entonces, la velocidad definitiva que el vehículo siguiente tendrá en el instante de tiempo $t+T$ deberá satisfacer las dos ecuaciones presentadas, para el modelo en flujo libre y el modelo en flujo condicionado. De esta manera entonces:

$$V_{n+1}(t+T) = \min\left[V_{n+1}^{libre}(t+T), V_{n+1}^{condic}(t+T)\right] \quad \text{(Ec.3.3)}$$

La velocidad del vehículo siguiente $V_{n+1}(t+T)$ corresponderá al mínimo entre las velocidades en flujo libre y condicionado.

Se puede apreciar en este punto que el mayor discriminador, entre que variante del modelo predominará, es la distancia entre los vehículos. Cuando existe una distancia muy grande entre ambos el valor de $V_{n+1}^{condic}(t+T)$ será muy alto debido a que los valores bajo la raíz cuadrara especialmente las distancias elevaran el resultado de la ecuación y la velocidad que tomará entonces el vehículo siguiente será el valor de $V_{n+1}^{libre}(t+T)$. Viéndolo de otro modo cuando un vehículo se desplaza por una vía y lleva otro delante de él a una distancia muy grande, éste no se deja de influenciar por el comportamiento del vehículo líder y se desplaza libremente, es esto entonces lo que se refleja con la elección de la velocidad mínima. Por otro lado cuando la distancia entre los vehículos se vaya acortando los valores de las velocidades en flujo libre y condicionado se irán equiparando hasta llegar un momento en que la distancia es tal que $V_{n+1}^{condic}(t+T)$ será menor que la velocidad que desearía llevar el auto siguiente en un flujo libre sin condicionamientos y de esta manera comenzaría a comportarse de acuerdo a la influencia ejercida por el vehículo líder.

Se asume tan bien que la velocidad del vehículo siguiente en el intervalo de tiempo entre que ocurre un evento y cuando el conductor comienza a reaccionar, es decir, la duración del tiempo entre el instante t y el instante $t+T$, será el promedio de las velocidades registradas para al inicio y termino del periodo. A manera de ejemplo si un conductor tiene un tiempo de reacción de 1,5 segundos y el vehículo que conduce lleva una velocidad de 15km/h en un momento de tiempo 10 segundos cuando ocurre un evento, el conductor comenzará a reaccionar a el evento en el instante de tiempo 11,5 segundos con una velocidad calculada para el vehículo de 13km/h. La velocidad entonces que llevará el vehiculo siguiente en el intervalo de tiempo entre el instante 10 segundos y el instante 11,5 segundos será de 14km/h. (Los valores son solo de ejemplo).

3.2 Modelo De Aceleración De Influencia Señal-Intersección

Como el objetivo del proyecto es crear un simulador de tráfico para intersecciones señalizadas por semáforos y el modelo de Gipps no considera esta interrupción en el flujo vehicular, es necesaria la integración de otro modelo que si considere ésta característica y que pueda representar la detención del flujo vehicular en las intersecciones en donde la luz roja del semáforo lo indique. Racero [Ref05], en una exposición de modelos de aceleración para microsimulación de tráfico urbano, expone el modelo de “Aceleración De influencia Señal-Intersección”, que representa el modelo que sigue un vehículo para detenerse totalmente en la línea de detención de una intersección. Antes de exponer el modelo es necesario definir dos conceptos importantes, distancia influencia y distancia de seguridad.

Distancia De Seguridad. La distancia de seguridad es el espacio mínimo que debe existir entre un vehículo y cualquier objeto delante de él de tal manera que en caso de frenado total del vehículo no exista impacto entre éste y el objeto.

Si el objeto es estático la distancia de seguridad en el instante t esta dada por:

$$D_{n+1}^{seg} = \frac{V_{n+1}(t)}{2 * d_{n+1}^{\max}} \quad \text{(Ec.3.4)}$$

Siguiendo las notaciones utilizadas en el modelo de Gipps y presentadas en la figura 1, la distancia de seguridad D_{n+1}^{seg} del vehículo siguiente en el instante de tiempo t va a ser igual a la velocidad de éste, en igual momento, divida por el doble de su máxima desaceleración.

Si el objeto delante del vehículo no estático si no más bien otro vehículo en movimiento, la distancia de seguridad esta dada por:

$$D_{n+1}^{seg} = \left| \frac{V_n(t)}{2 * d_n^{\max}} - \frac{V_{n+1}(t)}{2 * d_{n+1}^{\max}} \right| \quad \text{(Ec. 3.5)}$$

La distancia de seguridad en este caso es el valor absoluto de la diferencia entre las velocidades divididas por el doble de su máxima desaceleración, respectivamente para los vehículos líder y siguiente.

Distancia De Influencia. La distancia de influencia como su nombre lo indica es el espacio dentro del cual cualquiera sea el evento que ocurra ocasiona la reacción del conductor del vehículo ante el cual ocurre el evento. Si un evento ocurriese fuera de ésta distancia o espacio no provocaría ninguna reacción de parte del conductor. La distancia de influencia esta dada por la siguiente función.

$$D_{n+1}^{\inf} = D_{n+1}^{seg}(t) + V_{n+1}(t) * T + \frac{a_{n+1}(t) + a_{n+1}}{2} * T^2 \quad \text{(Ec. 3.6)}$$

Donde:

$a_{n+1}(t)$: Es la aceleración del vehículo $n+1$ en el instante de tiempo t.

a_{n+1} : Es aceleración máxima del vehículo $n+1$.

La distancia de influencia es directamente proporcional a la velocidad del vehículo, al aumentar ésta aumenta la distancia de influencia y siempre será mayor que la distancia de seguridad.

Estando definidos los conceptos de distancia de seguridad y distancia de influencia corresponde presentar el modelo de influencia señal-intelección.

Este modelo como se mocionó en párrafos anteriores representa la detención total de un vehículo en la línea de detención de una intersección y empieza a actuar una vez que la distancia de influencia es mayor que la distancia entre el vehículo y la línea de detención. La representación matemática del modelo es la siguiente.

$$V_{n+1}(t+T) = V_{n+1}(t) * \left(\frac{1 - e^{X_f - X_{n+1}(t)}}{1 - e^{X_f - D_{n+1}^{\inf}}} \right)^g \quad \text{(Ec. 3.7)}$$

Donde:

X_f : Longitud del tramo por donde se desplaza el vehículo $n+1$.

$X_{n+1}(t)$: Posición del vehículo $n+1$.

g : Factor de corrección. Representa las características del conductor.

Considerando a g como una constante y simplificando el modelo, éste se puede representar por el siguiente diagrama causal.

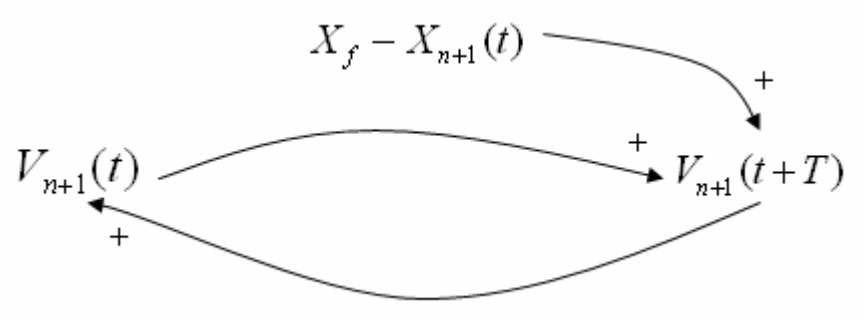


Figura N° 3.4 Diagrama causal del modelo distancia Influencia

La distancia $X_f - X_{n+1}(t)$ corresponde a la distancia entre el vehículo siguiente y la línea de detención de la intersección cuando esta disminuye provoca que se disminuya la velocidad del vehículo en el instante de tiempo $t+T$. Cuando esta distancia se vuelve cero hace que la velocidad del vehículo siguiente sea cero también. Independiente del valor que tome la distancia de influencia al disminuir la distancia hacia la línea de detención se provocara una disminución en la velocidad del vehículo (asumiendo que el vehículo debe detenerse por una señal de luz roja).

En el caso que la desaceleración aplicada al vehiculo entre los intentes de tiempo t y $t+T$ sea mayor que la máxima desaceleración posible del vehículo, se aplica esta última.

3.3 Integración De Los Modelos

Para el desarrollo del proyecto se utilizaran los dos modelos presentados en las secciones anteriores, el modelo de Gipps de seguimiento vehicular y el modelo de influencia señal-intersección. Para integrar y complementar los dos modelos se utiliza el semáforo como seleccionador de uno o de otro dependiendo de si está indicando luz roja o verde (amarilla). Si el semáforo esta con luz roja y además se cumplen las condiciones de

que la distancia de influencia sea mayor a la distancia entre el vehículo siguiente y la línea de detención y que no exista un vehículo líder o este ya cruzó la intersección, se utiliza el modelo de influencia señal-intersección. En Caso contrario de no cumplirse estas condiciones se utiliza el modelo de Gipps. El algoritmo encargado de implementar esto será descrito en detalle en posteriores capítulos.

CAPITULO 4. Selección De Tecnologías

Una ves claros los objetivos del proyecto, es decir, “Qué” se quiere conseguir con él y entendiendo el problema en sí, llega el momento de abordar el “Como” lograrlo. La selección entonces de las tecnologías o herramientas para el desarrollo de la aplicación tienen que soportar por un lado el ambiente tridimensional que se pretende dar al simulador, la parte gráfica que permita observar visualmente el comportamiento del sistema de flujo vehicular. Por otro lado para lograr una representación aceptable de la realidad es necesaria una herramienta que permita obtener comportamientos a nivel físico de los vehículos. Además es necesario seleccionar un lenguaje de programación que permita implementar los modelos de microsimulación que se utilizan y que finalmente otorgan el comportamiento al sistema.

4.1 Motor Gráfico.

La parte visual es una parte importante dentro del proyecto ya que comprende un objetivo que el usuario del simulador pueda ver a través de una interfaz de salida gráfica como se comporta el flujo vehicular en una intersección, cual es el comportamiento de un vehículo en particular dependiendo de las condiciones del medio, la presencia de una luz roja en el semáforo o la detención del vehículo que viaja directamente delante de él.

Inicialmente cuando no se tenían conocimientos relacionados con la programación gráfica menos aun en tres dimensiones se pretendió desarrollar la aplicación programando directamente la parte visual con las primitivas de OpenGL (librerías gráficas abiertas), es decir, la parte elemental de toda programación gráfica y creación de video juegos a nivel de líneas de código. Pero la falta de experiencia en este ámbito de programación no entregó un control adecuado sobre los objetos creados para darles el comportamiento que se requiere y si se agrega que el número de objetos a controlar es significativo la complejidad del

proyecto aumenta radicalmente, lo que llevo a desestimar esta vía de implementación. En su lugar se encontró la posibilidad de trabajar con motores gráficos, que permiten abstraerse de la programación a nivel elemental y entregan total control sobre los objetos creados.

En definitiva la primera vía que se intento explotar para la implementación del simulador traía consigo un trabajo no considerado y de gran envergadura que era la creación de un motor gráfico propio para el simulador, lo que creaba un alto riesgo para el proyecto. Sin embargo el haber optado inicialmente por esta alternativa entregó el conocimiento base de cómo trabajan los motores gráficos en relación con el renderizado de los objetos especialmente con OpenGL. *En términos simples renderizar comprende todo el proceso o los procesos desde que se crea un objeto hasta que se muestra por pantalla*.

Todos los video juegos y la mayor parte de las aplicaciones gráficas 3d, como es el caso del proyecto, están basadas en el uso de motores gráficos que permiten alejarse del uso directo de las librerías gráficas a nivel de primitivas. Permitiéndole a los desarrolladores concentrar sus esfuerzos en otras áreas como funcionalidad, rendimiento, apariencia entre otras sin invertir en el alto costo de tener que implementar a un nivel inferior manejando directamente cosas como listas de vértices, transformaciones matriciales para la rotación de los objetos etc. Es mucho más sencillo rotar un objeto en un ambiente tridimensional llamando a un método “rotar_Y(90°)” que cumpla la función de rotarlo noventa grados en el eje Y (como es el caso de los motores), a tener que lidiar directamente con la lista de vértices del objeto y con las matrices y cálculos necesarias para lograr la rotación en 3d.

Existen motores gráficos pagados y afortunadamente también de fuente abierta. A la hora de seleccionar un motor para integrarlo al proyecto se siguieron los criterios de: gratuidad, especialmente escritos en lenguaje de programación C o C++ y basados en la experiencia de terceros ya que experiencia propia no existe al ser un área nueva de desarrollo. Quien cumplió con estos criterios de selección fue Ogre, que es por hoy el motor gráfico de fuente abierta más popular. Esto es también corroborado por el ranking de motores pagados y libres disponible en http://www.devmaster.net/engines , donde Ogre3D

ocupa el primer lugar en una muestra de 10 motores gratuitos.

4.1.1 Ogre 3D

Ogre es un motor gráfico de fuente abierta orientado a objetos escrito en C++. Tiene licencia GNU Licencia Pública General Menor (LGPL) que permite su uso libre con algunas pequeñas restricciones. Fue diseñado con el objetivo de facilitar el trabajo de los desarrolladores que producen aplicaciones basadas en hardware de aceleración gráfica 3D. Utiliza librerías gráficas OpenGL y Direct3D de Microsoft, lo que permite la portabilidad del código a diferentes plataformas como Linux, Windows, Mac OS. Posee una comunidad altamente activa que a través de su foro participa desarrollando plugins que son capaces de integrar a Ogre con una gran cantidad de herramientas gráficas y da soporte continuo tanto a iniciados como a desarrolladores expertos. Lo que lo convierte sin duda en una atractiva opción para integrar al simulador.

Ogre tiene una gran cantidad de clases y subclases que le permiten realizar su trabajo, para entender a grandes rasgos como trabaja Ogre es útil el siguiente diagrama UML que incorpora las clases más importantes.

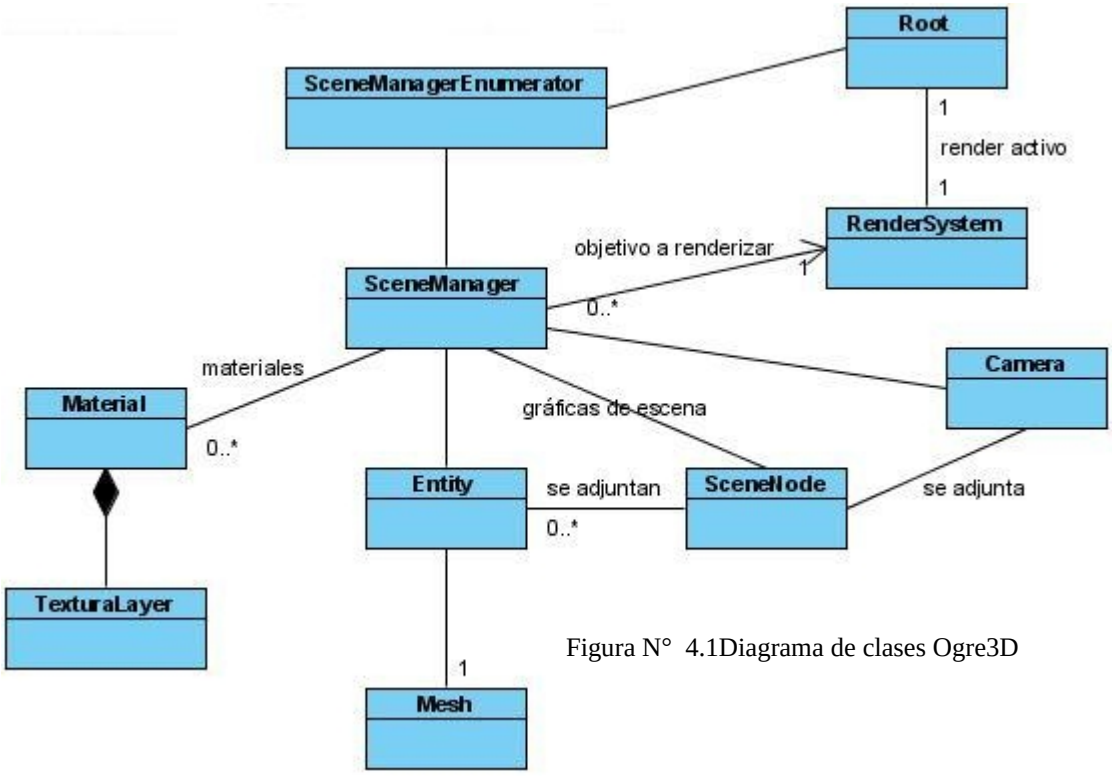


Figura N° 4.1Diagrama de clases Ogre3D

Este diagrama de clases fue desarrollado por Ogre3D para entregar una sobrevista muy general de las clases más importantes del motor y las relaciones que existen entre ellas. Cabe mencionar que la versión de Ogre utilizada en el proyecto corresponde a la versión 1.2.4 “Dagon”, ya que cuando se inicio el entrenamiento con esta herramienta correspondía a la ultima versión estable disponible. Finalizando el mes de marzo de este año 2007 se lanzo una nueva versión del motor gráfico con lo que la actual versión estable corresponde a 1.4.0 “Eihort”.

Volviendo al diagrama de clases. ***La instancia de la clase Root***, es decir, el objeto “Root” es el punto de entrada al sistema de Ogre. Este debe siempre ser el primer objeto que se crea y el último que se destruye. Es el que permite configurar el sistema, selecciona (o da la posibilidad de seleccionar) el sistema para el renderizado de los objetos que puedan haber en una escena, es decir, selecciona las librerías de OpenGL o Direct3D dependiendo de que estas existan y/o de las preferencias del usuario. Inicializa también al SceneManager y posee un método llamado “startRendering()” que es el encargado de iniciar el renderizado de los objetos y mantener el renderizado en cada frame mediante un loop hasta que se le dé la orden de finalizar, esto es terminando la aplicación. Un frame puede considerarse como una imagen independiente, una sucesión de frames dan la sensación de movimiento o animación, la fluidez del movimiento o la animación dependerá de los frames que se alcancen a desarrollar en un segundo. El cine tradicional genera 24 frames por segundo para dar la sensación al ojo humano de movimiento fluido. Los frames por segundo que desarrolle una aplicación por computadora dependerán de los procesos y/o cálculos que se tengan que realizar en cada frame (en cada iteración dentro del loop) además de las capacidades de hardware (procesador, aceleradora de video).

La clase RenderSystem es la que define las interfaces entre Ogre y las API 3D subyacentes (OpenGL, Direct3D). Una ves inicializado el sistema por el objeto “Root”, este selecciona el API 3D y le indica a RenderSystem cual es la API seleccionada para realizar el renderizado.

SceneManager es considerada la segunda clase más importante después de la clase Root. Es la clase encargada de organizar todos los elementos que están en una escena para ser renderizados.

Una escena puede ser definida como la suma de todos los objetos que serán desplegados por pantalla.

El objeto SceneManager crea las cámaras, luces, objetos (entidades) de una escena y mantiene la pista de todos estos para poder acceder a ellos cuando se requiera para dar la posibilidad de manipularos según convenga. Cuando llega el momento de renderizar una escena SceneManager envía a RenderSystem todos los objetos que se deben mostrar.

Existen diferentes tipos de "secene manager", algunos manejan mejor escenas en espacios cerrados, dentro de habitaciones o pasillos como por ejemplo juegos de primera persona tipo Doom. Otros están optimizados para escenas al aire libre. Pueden haber activos más de un "scene manager" al mismo tiempo para una misma aplicación. La clase ***SceneManagerEnumerator*** es la que tiene conocimiento de todos los "escene manager" disponibles y activos.

La clase ***Mesh*** representa un modelo discreto, un set de geometrías autónomo que generalmente es más pequeño que el mundo que integra [Ref06]. Un Objeto Mesh se utiliza para representar objetos móviles dentro de la escena, generalmente son creados en herramientas de modelado 3d y luego son exportados a un archivo .mesh que utiliza Ogre para recrear el objeto y desplegarlo por pantalla. También se pueden crear directamente en Ogre de forma manual realizando llamadas a métodos.

Los objetos de la clase ***Entity*** son instancias de un objeto móvil en la escena, estos pueden ser por ejemplo una persona, un perro o más puntualmente para nuestras necesidades un vehículo. Una entidad puede ser cualquier cosa y se basa en un set de geometrías, es decir, en los objetos Mesh. Una entidad entonces puede ser un vehículo modelado en una herramienta 3d, que es exportada a un archivo .mesh que es a su vez

considerado por Ogre como un objeto Mesh. Una entidad puede poseer sub-entidades, como en el caso del proyecto en donde una entidad vehículo posee cuatro sub-entidades rueda y cada rueda a su vez corresponde a un objeto Mesh separado.

La clase ***SceneNode***. Todos las entidades y opcionalmente cámaras y luces, son enlazadas o adjuntadas a un nodo, la forma de trabajar con una entidad u objeto de la escena es a través del nodo al que el objeto esta enlazado. Las trasformaciones tales como rotación, traslación, orientación, movimiento, no se aplican directamente a la entidad sino que al nodo que posee, es el nodo el que se rota o mueve a una determinada posición dentro del espacio tridimensional. Los nodos poseen una estructura jerarquizada que es manejada por la clase SceneManager, quien los crea y los destruye, en donde un nodo puede tener un nodo padre (solo uno) y muchos nodos hijos. Al inicializar el sistema a través de Root y crear un objeto SceneManager se crea automáticamente con este último un nodo llamado "nodo Root" que es el nodo principal del cual se crean (ramifican) el resto de los nodos a los cuales serán enlazados los diferentes objetos que tenga una escena. La siguiente gráfica permite aclarar esta jerarquía.

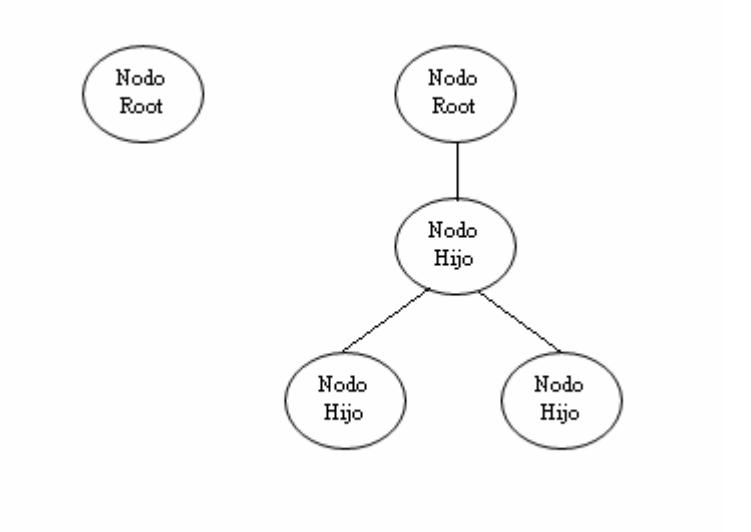


Figura N° 4.2 Ejemplo jerarquía de nodos

Una entidad solo puede ser enlazada a un solo nodo pero un nodo puede tener enlazada a él más de una entidad. El contenido de la escena que se renderizará en definitiva es el contenido de los nodos (o escenenodes) que en su conjunto conforma la estructura de la escena gráfica o "gráficas de escena" que es enviada a la clase SceneManager para que esta a su vez la envíe a RenderSystem para ser finalmente desplegado por pantalla. Así como un objeto puede enlazarse a un nodo también puede desligarse de él, si así fuera este objeto no será renderizado.

La clase ***Camera*** es la que, como su nombre lo describe, define los atributos y propiedades de las cámaras que serán las encargadas de entregar el punto de vista desde el cual se observará la escena renderizada. Las cámaras pueden ser rotadas, movidas directamente o a través de un nodo al que se puede enlazar. Una de las características más significativas de las cámaras es que definen una vista llama frustum que contiene todas las cosas que el ojo podrá ver.

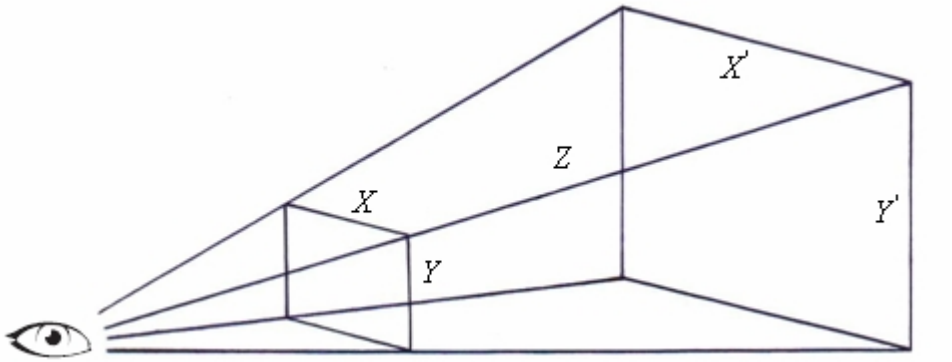


Figura N° 4.3 Frustum

El espacio contenido dentro de los seis planos que muestra la figura 4.3 es lo que se verá cuando la escena sea renderizada y es lo que se conoce como frustum, todo lo que quede fuera de este espacio no se mostrara por pantalla. Los Planos XY e $X'Y'$ limitan la visión cercana y lejana de la cámara respectivamente. Es decir, si un objeto se encuentra a cualquier distancia entre el ojo y el plano XY (en la figura) éste no será mostrado por la

cámara, así también si un objeto se encuentra a una distancia mayor que la contenida entre el ojo y el plano $X'Y'$ tampoco será renderizado. Z representa la profundidad, donde el plano $X'Y'$ se interna hacia dentro de la pantalla con Z^-. Una cámara es creada por un objeto SceneManager que mantiene la pista de esta durante la ejecución de la aplicación, a una cámara se le debe entregar información tal como posición, dirección, distancia al plano cercano (XY) y distancia al plano lejano ($X'Y'$) entre los más importante.

La clase ***Material*** controla todo lo relacionado con el aspecto de los objetos, sin considerar su forma. Un objeto de la clase Material controla como los objetos en la escena son renderizados desde el punto de vista de la apariencia, especifica las propiedades básicas de la superficie de los objetos como la reflexión de los colores, el brillo, las capas de texturas, los efectos que son aplicados. Los materiales pueden ser aplicados programaticamente llamando al método createMaterial de la clase SceneManager o pueden ser cargados por la aplicación en tiempo de ejecución a través de un script que contiene toda la información necesaria del material, este script es un archivo en lenguaje intuitivo con extensión .material. Generalmente cuando se modela un objeto 3d en una herramienta modeladora se le definen también los materiales (color, texturas), al momento de exportar el modelo 3d a un formato que Ogre puede interpretar se crean dos archivos uno con extensión .mesh que contiene la geometría del objeto 3d y otro con extensión .material que contiene la información de los materiales del objeto y las características de cómo serán aplicados. De esta manera cuando en Ogre se crea una “entidad” basado en una mesh se cargan automáticamente junto con ella los materiales, que son leídos desde el archivo .material.

Ogre provee un SDK, kit de desarrollo de software, para desarrollar aplicaciones basadas en su motor gráfico que se puede descargar libremente desde su pagina web www.ogre3d.org.

4.2 Motor Físico

Con el objetivo de dar mayor realismo al movimiento de los objetos, más puntualmente a los vehículos del simulador, se descubre la necesidad de integrar al proyecto un motor físico que permita simular el comportamiento de los vehículos a nivel de la física newtoniana.

Se puede definir a un motor físico como "Un programa por computadora que simula modelos físicos newtonianos, usando variables como masa, velocidad, fricción, peso y resistencia. Este puede simular y predecir efectos bajo diferentes condiciones que se aproximan a lo que ocurre en la realidad" [Ref07].

Al igual que con los motores gráficos existen motores físicos gratuitos y pagados. Los criterios de selección a la hora de determinar que motor físico integrar al proyecto son los mismos utilizados en la selección de Ogre3d más la restricción que se aplica por la elección de este último, es decir, que sea compatible con Ogre.

De los motores físicos que comúnmente se integran a aplicaciones desarrolladas con Ogre destaca por su rapidez y robustez, según la opinión de los propios usuarios recogida en los foros de Ogre [Ref08], PhysX.

4.2.1 PhysX

Se le da el nombre de PhysX a un chip y a un SDK desarrollado por AGEIA [Ref09] para realizar cálculos físicos complejos. El chip es un PPU, unidad de procesamiento físico, que entrega mayor rapidez al procesamiento de cálculos físicos ya que releva al procesador de esta tarea haciéndola él directamente. El SDK puede trabajar independientemente con la existencia o ausencia del chip ya que en caso de esto último los cálculos son realizados por el procesador. PhysX es un producto comercial pero de uso libre en proyectos no comerciales como es nuestro caso, está escrito en C++ soportado inicialmente en Windows pero recientemente se ha expandido su portabilidad a Linux.

PhysX es utilizado comercialmente por SONY en su consola de juegos PlayStation3.

A diferencia de Ogre no fue posible encontrar un esquema o diagrama que represente las clases definidas para la implementación de PhysX, y la relación que existe entre estas clases. Para dar un poco de claridad sobre los elementos (clases) más significativos del motor y su funcionamiento se realizó un diagrama de clases muy resumido y simplificado, rescatando la clases más significativas, basado en la experiencia adquirida que no pretende ser una formalización de un diagrama de clases para PhysX.

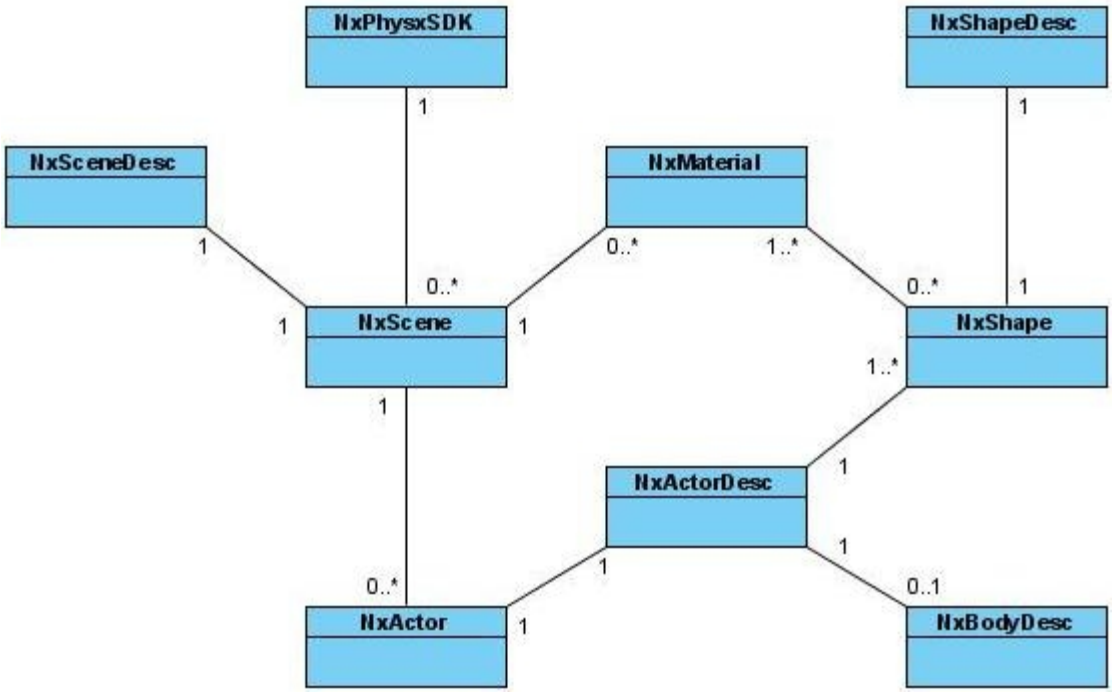


Figura N° 4.4 Diagrama de clases PhysX

El prefijo “Nx” que se utiliza en todos los nombres de las clases se debe a que anteriormente el SDK era conocido con el nombre de “**N**ovode**x**”, de donde heredó el prefijo.

Al igual que Ogre PhysX tiene un punto de entrada para inicializar su motor, este

punto de entrada es la clase ***NxPhysxSDK***. Lo primero entonces antes de realizar un simulación es crear una instancia de la clase NxPhysxSDK para inicializar el sistema, a través de este objeto se pueden ajustar parámetros que por ejemplo puedan dar la opción de visualizar los ejes de un actor (XYZ) para saber como esta orientado o hacia donde se está aplicado una fuerza, también permite ajustar un parámetro llamado "skin width" que puede ser traducido como "ancho de piel" que indica al sistema cuanto penetrará un objeto dentro de otro cuando ocurra una colisión entre estos. Una vez que el sistema se ha inicializado y ajustado según convenga a la simulación el objeto (instancia) NxPhysxSDK está encargado de crear una instancia de la clase ***NxScene***.

La clase ***NxScene*** en forma similar a Ogre define una escena en donde se encuentran objetos, en este caso **Actores**, en un ambiente de tres dimensiones que permite observar el comportamiento de estos objetos según las condiciones físicas a las que se sometan. Al momento de crear una escena, por parte de NxPhysxSDK, es necesario especificar la gravedad que afectará a los objetos en la escena además de si los cálculos físicos se realizan por hardware (PPU) o software, esto se hace a través de un descriptor de escena que es una instancia de la clase ***NxSceneDesc*.** Los descriptores son muy usados por PhysX y son estructuras que contienen toda la información sobre las características que se desea dar a los objetos al momento de crearlos.

La clase ***NxActor*** comprende los elementos básicos de una escena, es decir, las instancias de la clase NxActor, mejor dicho los actores, son los objetos que interactúan unos con otros dentro de la escena. Los actores dentro de una escena son creados por esta misma, es decir, la instancia de la clase NxScene que define la escena. PhysX emplea tres tipos de actores: los estáticos, dinámicos y cinemáticos. Un actor estático como su nombre lo indica no se moverá nunca, puede ser utilizado para representar suelos, montañas, una intersección etc; se considera que estos actores poseen una masa infinita y no son afectados por fuerzas. Un actor dinámico se moverá y actuará bajo la influencia de las distintas situaciones a las que se someta, fuerzas o velocidades, poseen una masa finita. Los actores

cinemáticos son objetos estáticos movibles, no se moverán por la aplicación de una fuerza o impacto pero pueden ser posicionados dentro de la escena según convenga. Los objetos dinámicos pueden ser convertidos en cinemáticos y viceversa, no así los estáticos que no pueden ser modificados en tipo. Al igual que con una escena al momento de crear un actor hay que especificar todas las características que se desea tenga este actor (entre ellas su tipo, cuerpo, forma, densidad) esto se hace a través de un *descriptor de actor,* instancia de la clase ***NxActorDesc***.

Un actor esta conformado por un cuerpo rígido, lo que PhysX denomina "Body", una forma de colisión, "Shape", que como su nombre lo indica es la forma que tomará un actor y se comportará en consecuencia. Es decir, si se crea un actor con forma de esfera este actor se verá y comportará como lo hace una esfera, así también si se crea otro actor con forma de cubo (caja) este se verá y comportará como tal. Como se mencionó anteriormente toda esta información está contenida dentro del descriptor de actor, si a un actor no se le asigna un cuerpo rígido (body) se considera como de tipo estático. La manera de asignar un cuerpo rígido a un actor es a través de la creación de un *descriptor de cuerpo,* esto es, la creación de una instancia de la clase ***NxBodyDesc***.

La clase ***NxShape*** define las formas con las que trabaja PhysX, estas pueden ser: planos, cajas, esferas, capsulas, ruedas, triangulo mesh. Para crear cualquiera de las formas anteriores se deben especificar características como dimensiones y materiales entre otras, esto se hace, al igual como en los casos anteriores, a través de descriptores y en este caso *descriptores de forma* que son instancias de la clase ***NxShapeDesc.*** Un actor puede tener varias formas asociadas como es el caso de un vehículo que en sí corresponde a un actor pero con una forma asociada para representar el chasis y otras cuatro formas para representar cada una, una rueda. La forma "triangulo mesh" da la posibilidad de crear formas más complejas, especialmente en herramientas modeladoras desde donde posteriormente se pueden exportar a un formato que puede ser leído por PhysX.

La clase ***NxMaterial***. A diferencia de Ogre en donde los materiales están más

relacionados con la apariencia, en PhysX son considerados desde el punto de vista físico. Los materiales por lo tanto son la sustancia física de la que están hechos los objetos (actores o más directamente las formas), definen propiedades internas como restitución y propiedades de la superficie del objeto como fricciones. Los materiales son creados por el objeto escena y es necesario ajustar sus propiedades dando valores a su restitución, fricción estática y fricción dinámica. Dar un valor alto de restitución, por ejemplo el máximo que es 1, significa que en un impacto el objeto perderá poca energía lo que llevará a que este tenga un rebote amplio, en el caso contrario si se le asigna un coeficiente de restitución bajo, por ejemplo de 0,15 el objeto perderá mucha energía en el impacto y rebotará poco. Si un material se ajusta con un coeficiente de fricción estática elevado provocará que el objeto tenga dificultades en desplazarse al intentar salir del reposo, esta dificultad disminuirá si disminuye el coeficiente de fricción. Un coeficiente de fricción dinámica pequeño permitirá a un objeto en movimiento deslizarse suavemente mientras que uno elevado tenderá a detenerlo de manera más rápida o más brusca. Una forma puede usar un material creado por la escena así como también puede dejar de usarlo y cambiarlo por otro en cualquier momento.

La versión del SDK PhysX utilizada para el desarrollo del proyecto es la 2.6.2, la última versión lanzada para el sdk corresponde a la 2.7. Se utiliza la versión 2.6.2 por que cuando se comenzó el entrenamiento con la herramienta esta correspondía a la ultima versión lanzada por Ageia.

Como se mocionó anteriormente PhysX se encarga de generar una escena en tres dimensiones que permite observar el comportamiento de los objetos dentro de la simulación, esto lo hace a través de OpenGL usando directamente sus primitivas. Es bueno recordar que PhysX es un motor físico y no gráfico, por lo tanto sus funcionalidades gráficas son mucho más limitadas que Ogre o cualquier otro motor gráfico, por lo que en general en el desarrollo de aplicaciones gráficas (juegos, simuladores) en donde se necesite simulación a nivel de física, son integrados ambos motores el físico y el gráfico cada uno

en lo que sabe hacer mejor.

La estructura general, y a grandes rasgos, de una simulación física usando PhysX es la siguiente.

```
int  main(int argc, char** argv){

InitGlut(argc, argv);
      InitNx();
      glutMainLoop();
      ReleaseNx();
      return 0;
}
```

La función principal comienza inicializando “Glut”, que es una caja de herramientas basada en OpenGl utilizada para administar de mejor manera el sistema de ventanas ya que es independiente de éste. Luego de inicializar Glut se inicializa el motor mediante la función InitNx(), ésta función inicializa el SDK, ajustas los parámetros que se necesiten ajustar, crea la escena, los materiales, los objetos dentro de la escena e inicia la simulación. Desde ahí la simulación entra en un loop hasta que se le ordene detener, el loop se obtiene a traves de la llamada a función glutMainLoop() que realiza una y otra vez llamadas a funciones tipo “CallBack” como por ejemplo a la función RenderCallback().

```
void RenderCallback()
{
      GetPhysicsResults();
      ProcessInputs();
      StartPhysics();
}
```

Dentro de las acciones más importantes que realiza esta función, desde el punto de vista de la simulación física, está obtener los resultados de la simulación en la iteración anterior llamando a la función GetPhysicsResults(), procesar las diferentes entradas (teclado por ejemplo) llamando a ProcessInputs() e iniciar nuevamente la simulación para la actual iteración (frame) en curso, mediante la llamada a StartPhysics(). Esto se repite hasta que se

deseea concluir la aplicación, generalmente procesando algún tipo de entrada.

La siguiente imagen muestra el entorno de trabajo utilizado por PhysX en los turoriales que tiene disponibles para el aprendizaje del motor.

Figura N° 4.5 Entorno trabajo PhysX

En la imagen se puede apreciar una escena donde un actor con forma de esfera cae por acción de la fuerza de gravedad, golpeando una pila de actores con forma de cajas.

4.2.2 El Envoltorio NxOgre

NxOgre en un envoltorio creado por Robin Southern [Ref10] que integra a Ogre con PhysX. De ésta manera al trabajar con NxOgre es posible acceder, a través de él, a ambos motores, gráfico y físico, dando la posibilidad de desarrollar aplicaciones con todo el potencial que posee cada uno. Al igual que con Ogre y PhysX, NxOgre esta escrito en lenguaje de programación C++, es orientado a objetos y tiene licencia open source pero está limitada por el uso de PhysX que como se mencionó anteriormente es pagada pero de libre uso en proyectos no comerciales.

El hecho de seleccionar PhysX como motor físico para el desarrollo del simulador de tráfico se basó fundamentalmente en la posibilidad de trabajar con NxOgre, ya que con éste se logró el primer acercamiento a la simulación de la física por recomendación de usuarios expertos de Ogre y disponibilidad de soporte en el aprendizaje, a través de tutoriales y un foro muy activo y cooperativo.

NxOgre posee muchas clases que envuelven a las clases definidas para Ogre y PhysX. Con el objetivo dar una visión más clara de como trabaja NxOgre se desarrolló un pseudo diagrama de clases con las características más relevantes y significativas que permitan entender al envoltorio, ya que no fue posible encontrar en su documentación un diagrama con estas características.

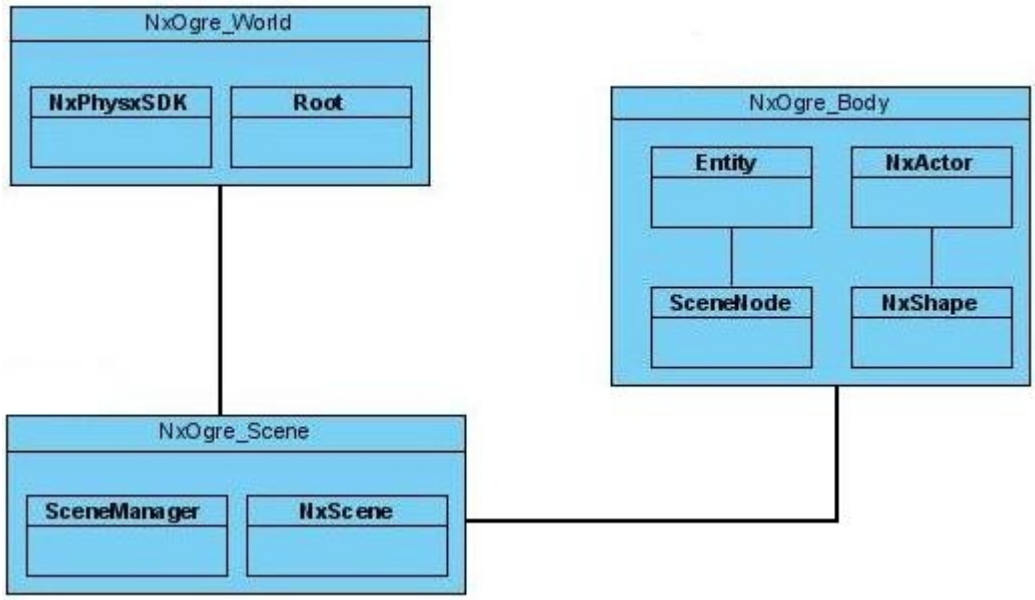


Figura N° 4.6 Pseudo diagrama de clases NxOgre

Como se mencionó con anterioridad Ogre y PhysX necesitan partir inicializando sus sistemas, esto lo consiguen mediante la clase Root y NxPhysxSDK respectivamente. NxOgre realiza este trabajo a través de la clase ***NxOgre_World***, el objeto NxOgre_World

toma como parámetro el objeto Root de Ogre ya inicializado y se encarga de inicializar el SDK PhysX mediante la creación de un objeto NxPhysxSDK. A través del objeto NxOgre_World se puede acceder a todos los métodos disponibles para los objetos de las clases inicializadoras de ambos motores, además el objeto NxOgre_World es el responsable de crear la escena de NxOgre.

La creación de una instancia de la clase NxOgre_Scene lleva implícita la creación de una escena tanto para Ogre, mediante un objeto de la clase SceneManager, como para PhysX, con la creación de un objeto de la clase NxScene. Al igual que ocurre con el objeto NxOgre_World, la escena de NxOgre otorga acceso a todos los métodos disponibles para las instancias de SceneManager y NxScene, permitiendo obtener toda la funcionalidad de las escenas de Ogre y PhysX. Como se puede apreciar hasta aquí el trabajo de los motores se realiza en paralelo, siendo NxOgre el nexo entre los dos.

Ogre define los objetos dentro de su escena como “Entidades”, PhysX por su lado los llama “Actores”. NxOgre a través de la clase ***NxOgre_Body*** integra los dos conceptos de objetos dentro de una escena, tanto el de Ogre como el de PhysX, y les da el nombre de “Bodies” o “Body”. Cuando NxOgre crea un Body mediante y dentro de su escena está creando al mismo tiempo un actor para PhysX y una entidad para Ogre, así logra realizar las simulaciones físicas con el actor y representar esto gráficamente a través de la entidad creada para Ogre. A los que se puede llamar también “Modelo de colisión” y “Modelo Gráfico” respectivamente. En donde el modelo de colisión corresponde a la forma (shape) sobre la cual se realizan los caculos físicos y el modelo gráfico corresponde a la entidad o mesh sobre la cual se basa la representación gráfica del objeto. Mediante un objeto NxOgre_Body se puede tener acceso a todos los métodos de la clase Entity y todos los métodos de la clase NxActor. También al crear un objeto NxOgre_Body se enlaza la entidad creada para Ogre a un nodo de la clase SceneNode que permite tener control sobre la entidad.

NxOgre utiliza, a través del uso de su escena (clase NxOgre_Scene), la escena de la

clase SceneManager para trabajar con las entidades y la escena de la clase NxScene para trabajar con los actores. Cada movimiento realizado por un actor (modelo de colisión) es replicado por la entidad (mesh) de esta manera se puede observar gráficamente a través de Ogre 3d el comportamiento de los objetos dentro de una simulación física realizada por PhysX. Por último, si se crea un "Body" con forma de esfera (shape) y una apariencia gráfica (entity o mesh) de una jirafa, éste Body no se comportará como jirafa sino que como esfera y lo más probable que la jirafa ruede antes que caminar.

El desarrollo del simulador se realizará sobre la base de las clases definidas por NxOgre, de ésta manera se logra integrar al proyecto el motor gráfico Ogre3d y el motor físico PhysX. Cualquier funcionalidad no cubierta por NxOgre tiene la facilidad de ser salvada interactuando directamente con cualquiera de los dos motores, ésta es una ventaja que posee NxOgre.

4.3 Modelador 3D

Los modeladores 3d son herramientas, software, dedicadas al modelado y creación de gráficos tridimensionales. Permiten modelar objetos 3d individuales, que son básicamente una representación de coordenadas, vértices y caras, para posteriormente integrarlos en alguna escena. La representación del objeto modelado también se conoce con el nombre de "malla" o en ingles "mesh".

Un proceso básico en la creación de gráficos 3d por computadora comprende 3 pasos: *El Modelado*, *La Composición de la Escena* y *El Rénder*. Estas etapas son cubiertas en su totalidad por las herramientas de modelado. La composición de la escena o la creación de la escena involucra la distribución de todos los objetos, luces, cámaras y otras entidades que serán utilizadas para producir una imagen estática o dinámica. El Rénder por su parte es el proceso final encargado de generar la imagen en 2d o animación a partir de la escena creada.

Para poder obtener los objetos que formarán parte de las simulaciones de tráfico, es

decir, los vehículos, las ruedas de estos, los semáforos y hasta la misma intersección es necesario modelar dichos objetos, complejos, en una herramienta modeladora 3d que permita posteriormente integrarlos al simulador.

Los criterios de selección utilizados para determinar que modelador utilizar fueron los de gratuidad y libertad de uso, disponibilidad de recursos en el aprendizaje de la herramienta, compatibilidad con el motor gráfico (que los objetos modelados puedan ser posteriormente trabajados con Ogre) y que siga manteniendo la portabilidad entregada por las herramientas ya seleccionadas. El modelador que cumplió con estos criterios es ***Blender***.

4.3.1 Blender

Blender es un software libre multiplataforma con licencia GPL, creado para modelar y crear gráficos tridimensionales. En la actualidad la última versión disponible de este modelador es la 2.44 que se puede obtener gratuitamente desde su sitio en Internet www.blender.org. La versión de blender utilizada para la creación de los vehículos, ruedas, semáforos e intersección utilizados en el simulador, es la 2.42a.

Blender tiene el mismo potencial para desarrollar productos de nivel profesional que otras herramientas modeladoras pagadas, incluso ya ha sido utilizado en el desarrollo de películas animadas como "Elephant´s Dream".

4.4 Manipulador De Imágenes

Como su nombre lo indica, las herramientas de este tipo permiten manipular imágenes digitales (gráficos, fotografías). Los usos más comunes que se le dan ha este tipo de aplicaciones son entre otros la creación de logos, cambio o recorte de fotografías, cambio de colores y brillo de fotografías, combinación de imágenes a través de uso de capas, eliminación de elementos no deseados en una imagen, conversión entre los distintos formatos de imágenes existentes etc. Otro uso común dentro de los desarrolladores de aplicaciones gráficas (juegos, simuladores) tridimensionales es la creación de texturas para cambiar o mejorar la apariencia de los objetos modelados. Este es el motivo por el cual se

integra un manipulador de imágenes al proyecto, la necesidad de crear texturas para los diferentes objetos 3d utilizados en el simulador.

Los criterios utilizados a la hora de seleccionar un programa manipulador de imágenes son básicamente los mismos utilizados en la selección de las otras herramientas, es decir, que sea un software open source, exista disponibilidad de recursos en el aprendizaje de la herramienta y sea multiplataforma.

4.4.1 Gimp

Gimp es un software para la manipulación de imágenes digitales y es integrado al proyecto principalmente por la necesidad de crear texturas para los objetos tridimensionales creados para usar en las simulaciones de tráfico. Gimp es la sigla de **P***rograma de* ***M****anipulación de* ***I****magen* ***G****NU,* es gratuito posee licencia GNU (**G**NU **N**o es **U**nix) GPL (**L**icencia **P**ública **G**eneral). Inicialmente fue desarrollado para sistemas Unix pero en la actualidad soporta múltiples plataformas lo que le permite ser portado a Windows, Linux, Mac OS. La versión utilizada de Gimp en el proyecto corresponde a la 2.2, siendo la actual versión lanzada a la fecha de presentación de este informe la 2.3.

4.5 Lenguaje de Programación y Compilador

Como ambos motores están escritos en C++, la codificación, es decir, la implementación de los modelos de simulación y las distintas funcionalidades del software también serán realizadas en este lenguaje de programación siguiendo el enfoque orientado a objetos.

Para generar y compilar el código se utilizará el compilador Microsoft Visual C++ 2005, en su edición Express que es de uso gratuito. Esta versión de Visual C++ es limitada en cuanto a su potencial de desarrollo pero las funcionalidades provistas por la edición Express son más que suficientes para desarrollar el proyecto en lo que a codificación respecta ya que las restricciones son referentes al uso las librerías de clases de Microsoft o

clases Microsoft Fundación (MFC) que no son utilizadas.

La razón de por que utilizar Visual C++ obedece simplemente a que NxOgre utiliza este compilador en sus tutoriales, en los códigos fuentes y ejecutables (archivos con formato de proyectos Visual C++), y provee de una serie de pasos para conseguir ajustar NxOgre con Visual C++. Aunque si se desea, claro está, se puede utiliza cualquier otro compilador de C++.

PARTE II. Desarrollo De La Aplicación

CAPITULO 5. Integración de las Herramientas en el Desarrollo de la Aplicación

Las herramientas descritas en el capitulo anterior se van integrado al proyecto a medida que van siendo necesarias. En nuestro caso en particular se siguió el siguiente flujo de trabajo agrupando las herramientas en dos grupos como se grafica en la siguiente figura.

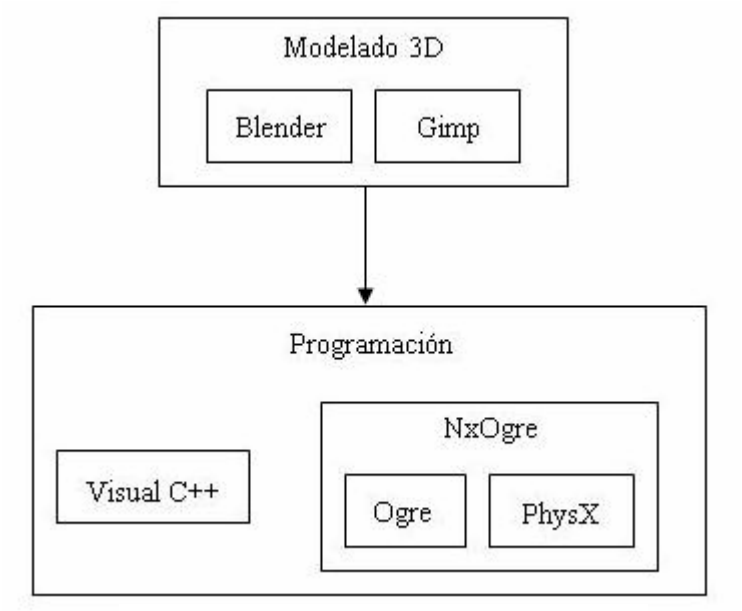


Figura N° 4.7 Flujo integración herramientas

Es bueno destacar que este proceso de integración se enmarca dentro de la fase de construcción o codificación del simulador y no engloba a todo el proceso de desarrollo del SW.

A continuación se detalla paso a paso como es abordado el proceso de construcción.

5.1 Modelado 3D

La decisión es primero modelar los objetos tridimensionales ocupados en el simulador. Estos son, como se mocionó anteriormente, los vehículos, los semáforos y la intersección.

Para crear los objetos tenemos a blender, expondremos aquí como modelar un vehículo liviano tipo sedán.

5.1.1 Modelado del Vehículo

Lo primero es disponer de un plano del vehículo que sirva como guía en el proceso de modelado. A este plano se le conoce como *blueprint* y generalmente provee de todas las vistas del objeto para facilitar su modelado en un ambiente de tres dimensiones. El vehículo seleccionado como modelo para utilizar en el simulador es el Audi A4 (año 2004) cuyo blueprint esta disponible en la pagina dedicada al tema www.the-blueprints.com.

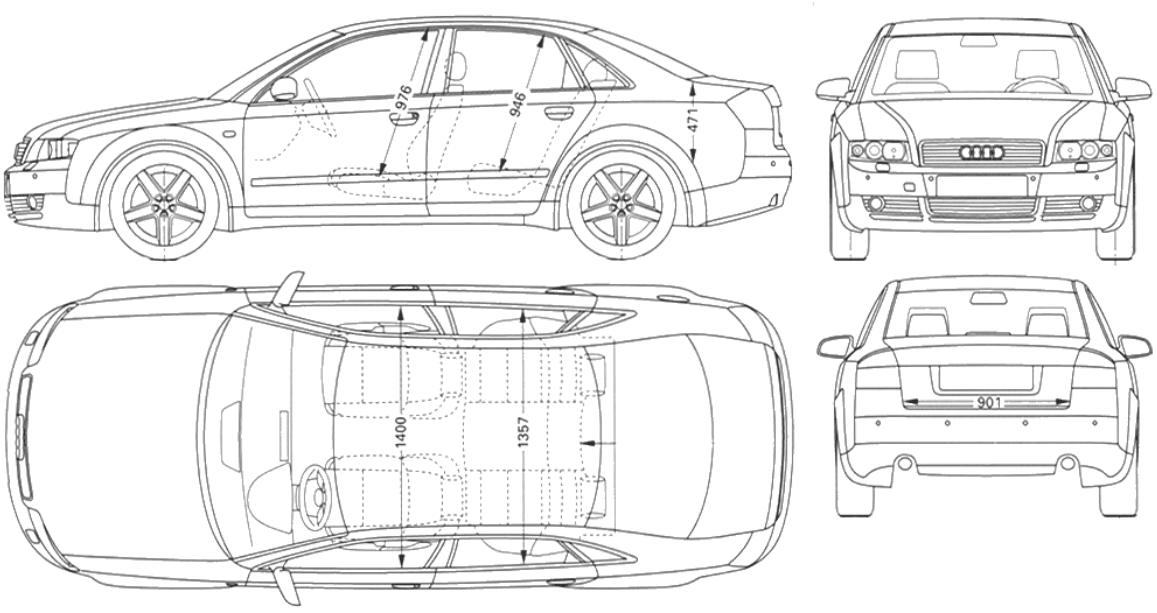


Figura N° 4.8 Blueprints vehículo

La imagen anterior corresponde al blueprint de un Audi A4, como se puede apreciar provee las cuatro vistas fundamentales para modelar el vehículo.

Lo siguiente a realizar una vez conseguido el blueprint es separar las cuatro vistas en un archivo diferente cada una, de tal manera de poder disponer de estas vistas por separado, esto se realiza con Gimp y como es de suponer no es un trabajo complicado realizar.

Ahora con las vistas separas cada una en un archivo hay que utilizarlas como guía para comenzar a realizar el modelado. Esto se consigue separando la ventana de trabajo de Blender en 4 partes y asignando a cada parte de ventana (o subventana) una vista del blueprint. La imagen siguiente muestra el resultado de este proceso.

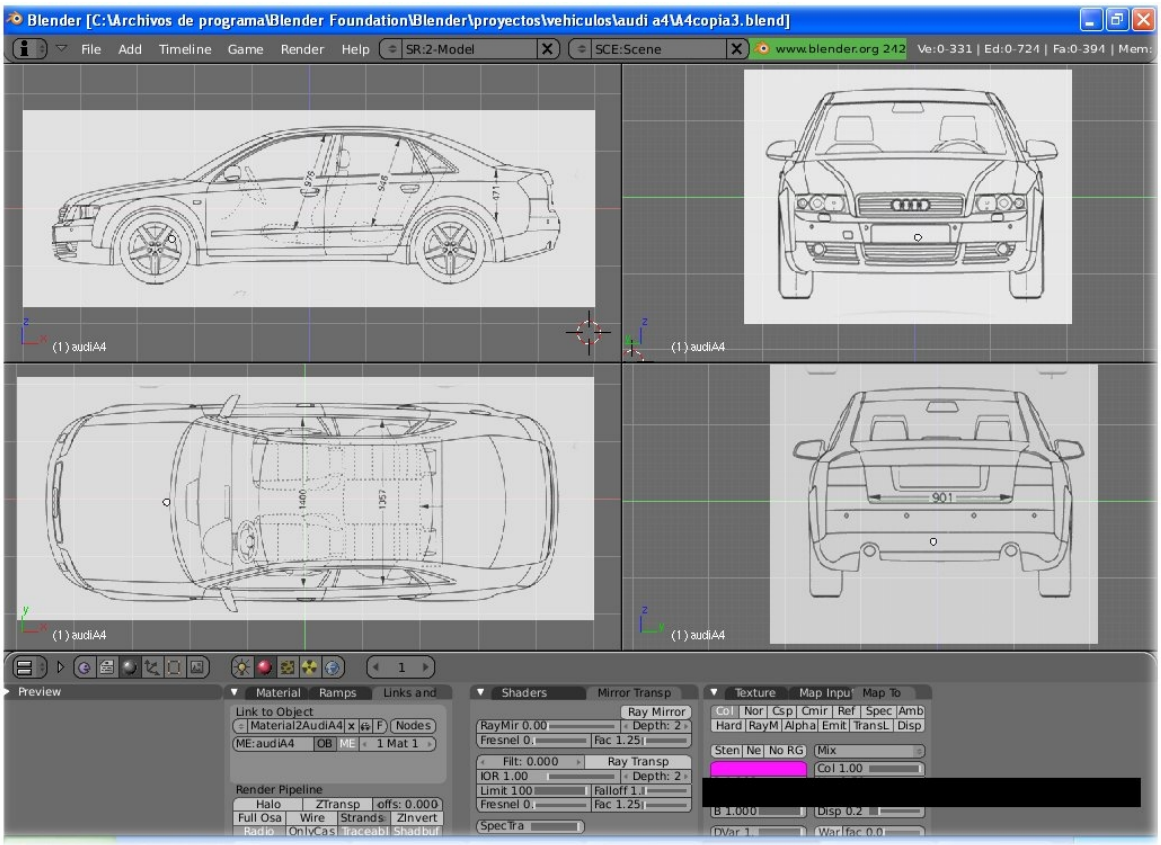


Figura N° 4.9 Entorno de trabajo Blender

En la imagen se puede apreciar el entorno de trabajo de Blender, además de las cuatro divisiones realizadas para albergar en cada una las distintas vistas del blueprint. Cada subventana posee un plano de trabajo distinto, no se puede apreciar muy claramente en la imagen pero la subventana superior izquierda visualiza el plano ZX (Z en color azul, X en color rojo), la ventana contigua y a la derecha visualiza el plano ZY (Y en verde) en su vista opuesta con la guía de la parte frontal del vehículo, por último la ventana inferior derecha visualiza también el plano ZY desde su vista normal con la imagen del blueprint de la parte posterior del vehículo.

Los planos de trabajo se asocian a una imagen de tal manera de tener correspondencia en cada uno de los otros planos, más claramente, por ejemplo si en la vista superior izquierda se realiza un desplazamiento en X positivo (hacia la cola del vehículo), en la vista del plano YX (vista inferior izquierda) el desplazamiento también será hacia la cola del vehículo. Así también en la vista del plano ZY opuesto, el movimiento será hacia adentro de la pantalla del monitor (cola del vehículo), y en la vista normal del plano ZY, el desplazamiento será hacia fuera de la pantalla del monitor. O sea la disposición de las vistas de trabajo y las imágenes debe ser coherente. Es importante mencionar también que la disposición de las diferentes vistas depende de las características personales de cada modelador y puede variar entre uno y otro.

Ahora con las vistas ubicadas de manera correcta solo queda seguir las imágenes guías para dar forma al vehículo. Esto se consigue de muchas maneras distintas, por nombrar algunas, se puede crear un gran cubo que cubra todo el vehículo, es decir, el blueprint y después ir ajustando este cubo hasta dar la forma del vehículo. También se pueden ir creando puntos separados que posteriormente se van uniendo y van dando forma a la malla del Audi. También se utilizada la creación de planos que se van ajustando a las distintas vistas del vehículo y conformando la maya de éste. Para el caso particular del proyecto se aplica una mezcla de estas dos últimas técnicas de modelado mencionadas, o sea a través de la creación de planos y puntos separados. Como el objetivo no es desarrollar

un tutorial para el modelado de un vehículo no se entrará en los detalles que esto llevaría.

A modo de ejemplo la imagen siguiente muestra un *plano* (figura geométrica) creado en la visualización ZX que cubre las puertas del lado derecho del vehículo. El plano creado se visualiza por completo en la vista ZX, mientras que en el resto de las vistas solo se ven sus puntos, como es lógico de pensar.

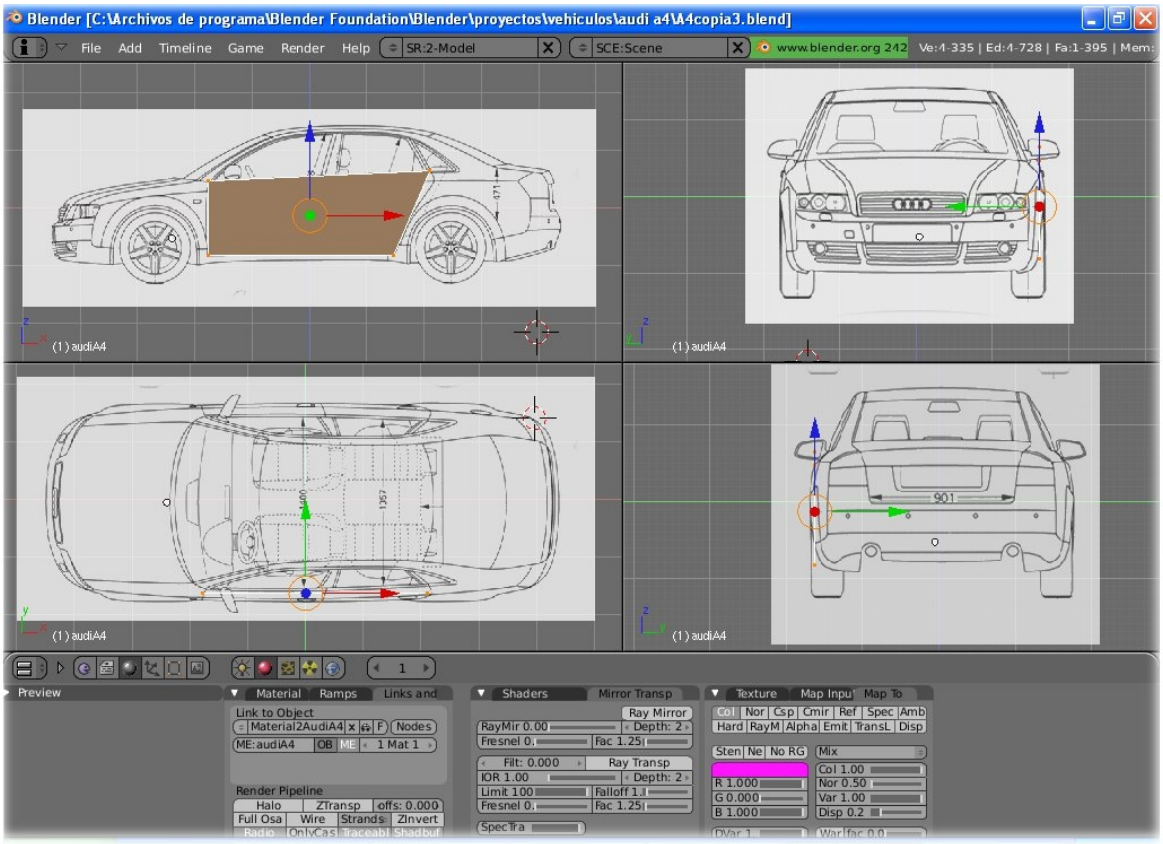


Figura N° 4.10 Modelamiento vehículo

Se ajustan los vértices del plano creado para seguir la geometría del vehículo y se realizan todas las subdivisiones del plano (creación de más vértices) que sean necesarias para conseguir este ajuste. Cada vértice debe ser ajustado en las cuatro vistas para que el modelo tome la forma correcta en las tres dimensiones. Este proceso se repite hasta obtener

la malla completa del vehículo.

La gran mayoría de los blueprints vienen diseñados de tal manera que todas las vistas sean equivalentes, es decir, por ejemplo que los espejos retrovisores estén a la misma altura en todas las vistas, de esta manera se pueden ajustar los vértices para que en cada plano de trabajo tengan la misma posición sobre el blueprint, independiente de la vista. Sin embargo en muchas ocasiones los blueprint no vienen tan exactos y es muy difícil hacer concordar un vértice sobre el mismo lugar de la imagen guía en las diferentes vistas.

A pesar de que cada ventana esta determinada para visualizar un plano de trabajo en particular, se tiene la libertar de rotar esta vista como se desee.

Como un vehículo es un objeto simétrico, es suficiente con modelar una mitad de éste ya que la otra mitad se consigue duplicando lo ya modelado.

La imagen siguiente muestra la malla completa

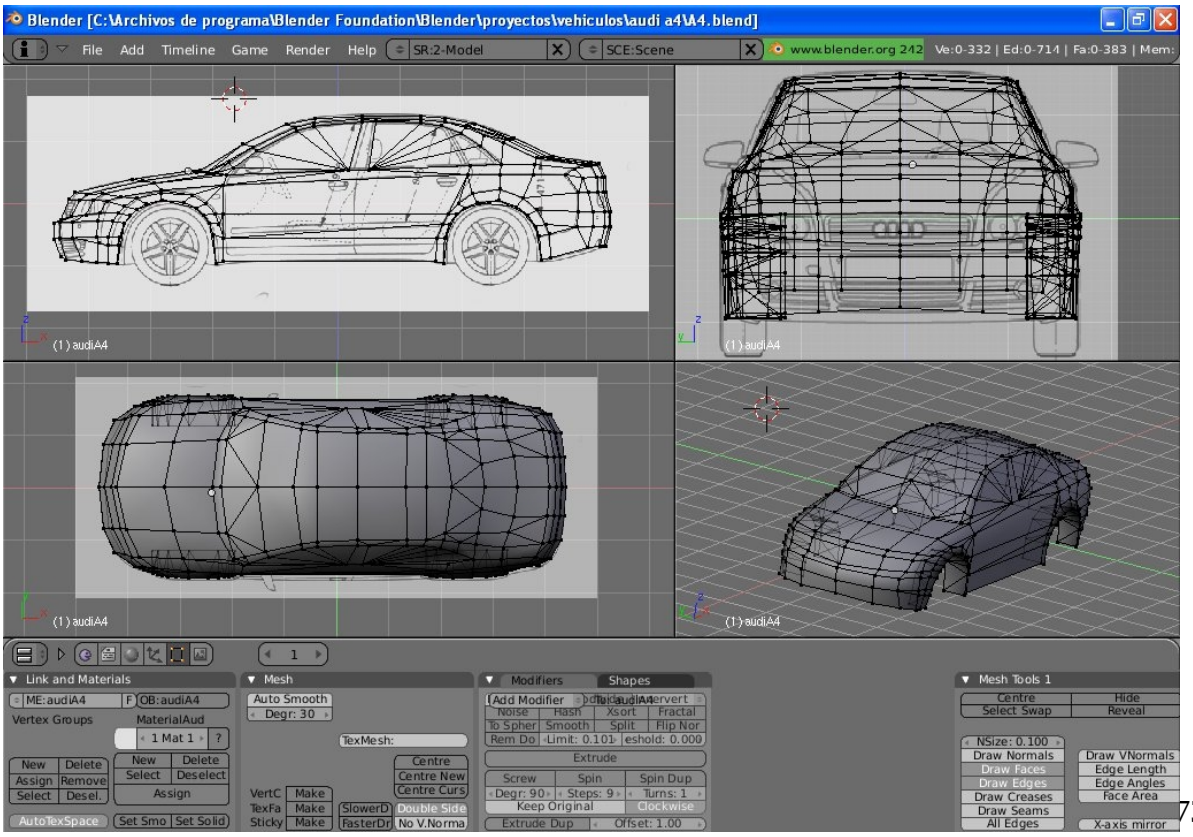


Figura N° 4.11 Malla terminada

Las dos vistas superiores muestran el vehículo en modo alambre, mientras que las inferiores lo hacen en modo sólido. El modelo construido es muy básico con solo 332 vértices, tiene la forma general del Audi y no consideran detalles como la separación entre puertas, capo, relieves entre los vidrios y los marcos, focos, mascarilla delantera etc. El objetivo fue tratar de construir un modelo económico en cuanto a número de polígonos utilizados, lo que se conoce en el mundo del modelado 3d como un modelo *low poly*, es por que mientras mayor sea el número de polígonos utilizados en el modelo mayor será el costo en tiempo de procesamiento del objeto. Como se estima que el simulador tendrá muchos vehículos interactuando en el sistema en un mismo instante la sobrecarga de procesamiento con objetos que contengan un elevado numero de polígonos (un modelo detallado puede tener sobre 20.000 vértices), modelo *high poly*, sería contraproducente con los objetivos del proyecto ya que se perdería la fluidez y entorpecería la visualización de la interacción de los autos dentro de la simulación.

Para tratar de mejorar el aspecto del modelo creado, se utilizan texturas que puedan ayudar en parte a dar un poco más de similitud con un vehículo del mundo real. La gran parte del peso en cuanto a la apariencia del modelo recae entonces sobre las texturas que se le aplicarán.

Para crear las texturas se utiliza el programa de manipulación de imágenes Gimp. Como la idea de aplicar texturas es dar un poco más de realismo al modelo, la mejor manera de lograrlo es a través de la utilización de imágenes de vehículos reales.

La pagina Web de Audi www.audi.com, provee imágenes de cada uno de sus modelos comercializados en vistas de 360°, es decir, imágenes de todos los posibles ángulos de vista. Para crear las texturas se capturaron imágenes de 3 vistas (Audi A4), a saber, la vista lateral, frontal y posterior.

Las imágenes capturadas se procesan en Gimp, eliminado lo que no es necesario, oscureciendo los vidrios para eliminar los reflejos y el interior del vehículo, juntando las

tres vistas en un solo archivo de formato “jpg” y retocando las imágenes en donde sea necesario. Con Gimp también se cambia el color de las texturas para poder crear vehículos de distintos colores. De esta manera se obtienen las texturas y se regresa a Blender para aplicarlas sobre la malla, lo que se conoce como *mapeado UV*.

El mapeado UV es una manera de mapear texturas de tipo imagen, 2D, sobre modelos tridimensionales siguiendo las coordenadas de la superficie (coordenadas UV) [Ref11].

La siguiente imagen muestra el modelo finalizado con la textura aplicada.

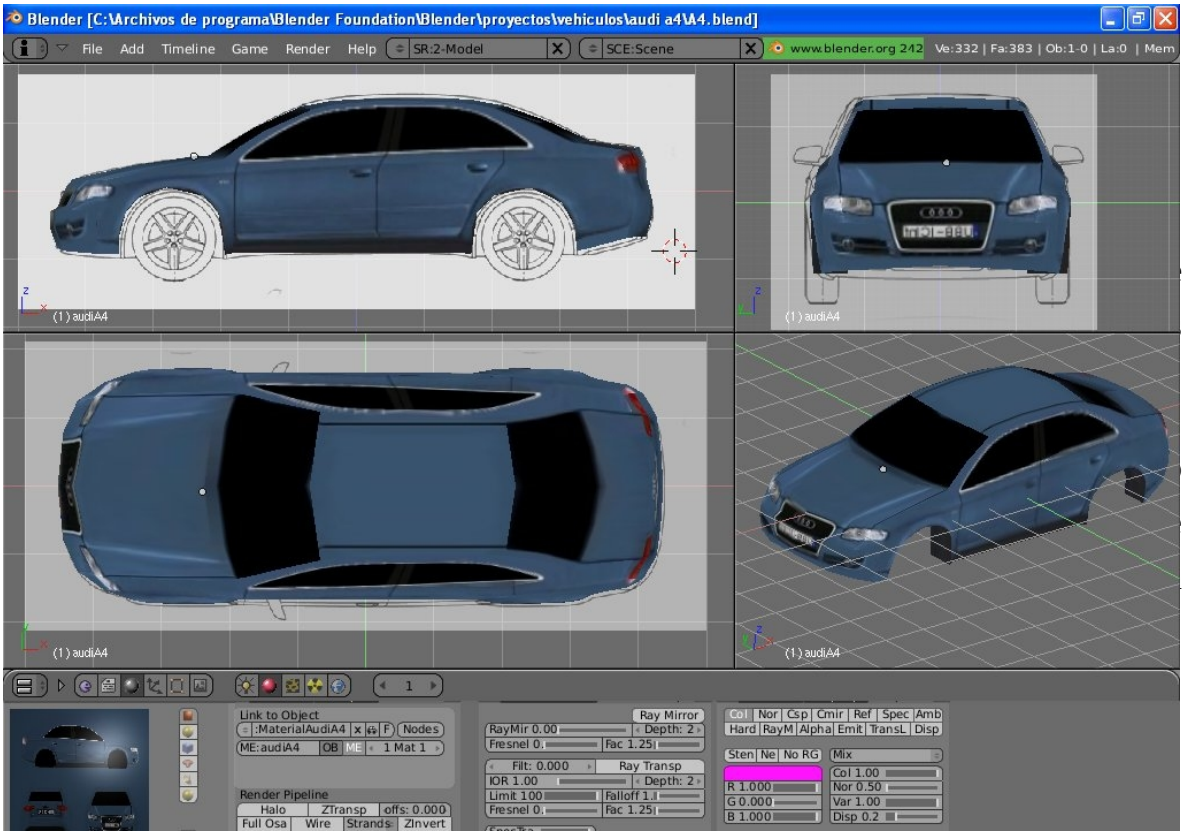


Figura N° 1.12 modelo con textura aplicada

Por último una vez que se tiene el modelo finalizado este se escala según la

información entregada por las tablas expuestas en la primera parte del informe. Escalando el modelo a la longitud, altura y ancho de un vehículo de tipo liviano. En donde una unidad de blender se considera como un metro de longitud al igual que en Ogre (NxOgre), por lo tanto se mantiene la equivalencia de unidades entre la herramienta modeladora y el motor gráfico.

Para que el modelo creado pueda ser utilizado con Ogre es necesario exportarlo desde Blender a un archivo de formato .xml.mesh. El plugin que realiza este trabajo de exportación se llama *"Ogre Meshes"* y fue desarrollado por miembros de la comunidad de Ogre. El archivo generado contiene información de la malla, puntos tridimensionales, vértices, caras entre otros, en formato XML. Este archivo en formato XML debe ser convertido a un archivo binario que pueda ser leído por Ogre, esto se hace a través de una herramienta llamada *"OgreXMLConverter"* que es un ejecutable manejado a través de línea de comandos, este programa convertidor viene con el SDK de Ogre y genera como resultado de la conversión un archivo con extensión .mesh.

Al momento de exportar el modelo desde Blender a un archivo .xml.mesh, el plugin "Ogre Meshes" crea otro archivo con extensión .material, que contiene toda información sobre los materiales aplicados al vehículo, en este caso la textura que se creó con Gimp. De esta manera es posible que Ogre reproduzca el vehículo con su textura, ya que recupera toda la información desde el archivo .material. Es necesario, claro esta, tener disponible la textura para que Ogre pueda acceder a ella y aplicarla al vehículo.

5.1.2 Modelado de la Intersección

Para el modelado de la intersección se toma como referencia el manual de demarcaciones de Vialidad, en las consideraciones esenciales, expuesto muy resumidamente en el capitulo 2. Como la intersección utilizada para prueba de simulador es genérica se utiliza un diseño típico de una intersección con dos flujos distintos. Si se deseara diseñar una intersección real y particular se puede tomar como referencia una imagen aérea o satelital y modelarla tal como se describió en la sección anterior el modelamiento de un vehículo, la diferencia estaría en que la intersección se modela desde una sola vista de la imagen de referencia.

La siguiente imagen muestra la intersección terminada en Blender.

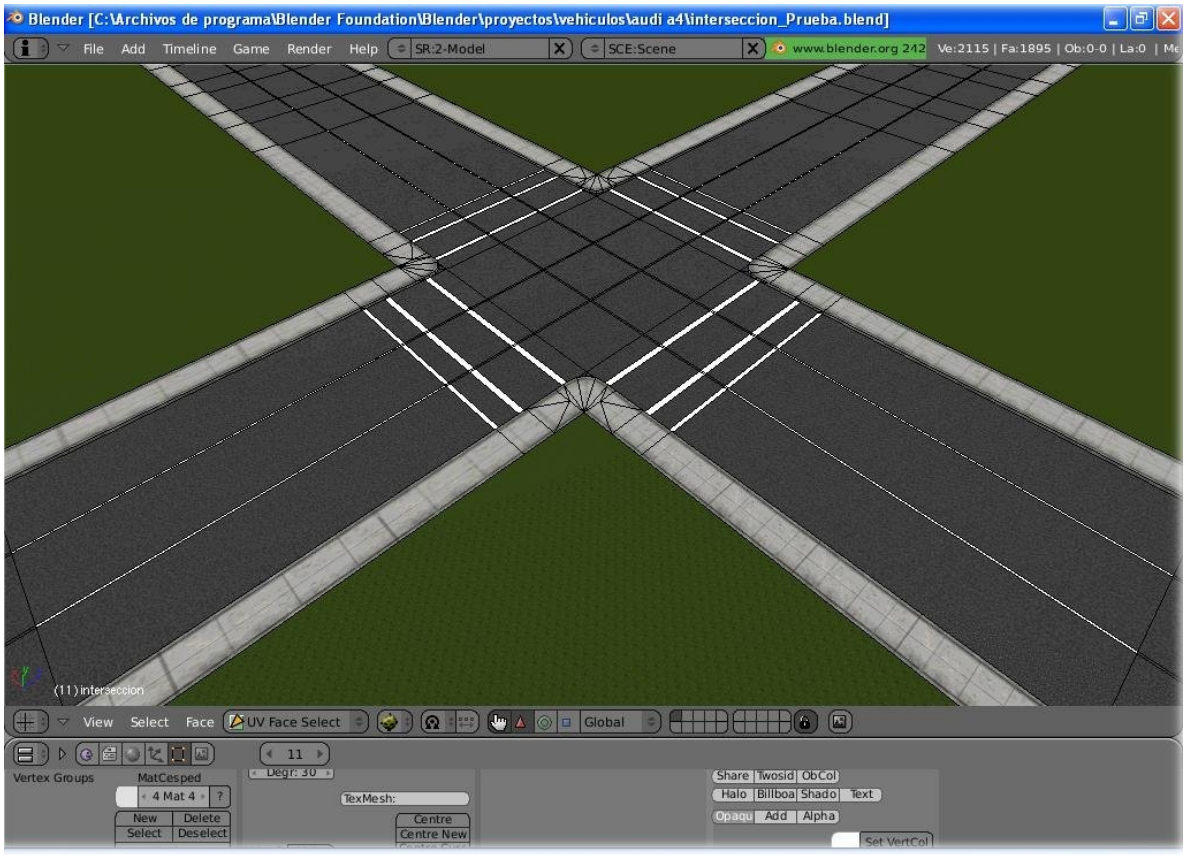


Figura N° 4.13 Intersección Modelada

5.2 Programación

En el grupo de las herramientas de programación se encuentran el compilador, los motores, gráfico y fisco, además del envoltorio que permite trabajar de forma transparente con ambos. Todas herramientas de programación se integran al proceso de desarrollo al mismo tiempo cuando se inicia la etapa de codificación del simulador, al trabajar con el envoltorio NxOgre se trabaja paralelamente con Ogre y PhysX y para trabajar con NxOgre se utiliza el compilador Visual C++:

La implementación del simulador se realiza trabajando directamente con la API de NxOgre, dejando en un segundo plano a Ogre y PhysX (sin olvidar que son ellos los realizan el trabajo) aunque en algunas oportunidades es necesario trabajar directamente con los motores, afortunadamente NxOgre provee estas facilidades.

Veamos a manera ejemplo cuales serían las líneas de código (en pseudo código C++) necesarias para crear un cuerpo en NxOgre, que podría ser un vehículo.

```
Inicio {
//Declaración del mundo, la escena y el cuerpo en este caso un vehículo
world *mMundo;
scene *mEscena;
body  *mVehiculo;
//creación del mundo
mMundo = new world(mRoot);
//creación de la escena
mEscena = mWorld->createScene("myEscena", mSceneMgr);
//aplicar gravedad a la escena
mScene->hasGravity();
//crear un suelo para la escena
mScene->hasFloor();
//crear el cuerpo
mVehiculo      =      mEscena->createBody("mVehiculo","vehiculo.mesh",new
cubeShape(Vector3(4.7,1.4,2.0)), 1500.0f, Vector3(0,0.7,0));
}Fin
```

Lo primero es crear el mundo “mMundo”, en el cual ocurrirán las cosas, el constructor recibe como parámetro un objeto mRoot de la clase Root de Ogre para poder integrar al motor gráfico en el programa, en capitulos anterios se explicó la funcionalidad

de este objeto. Lo segundo es la creación de la escena “mEscena” dentro del mundo (en un mundo pueden haber muchas escenas), el constructor recibe como parametro un string con el nombre de la escena que se esta creando y un objeto mSceneMgr de la clase SceneManager de Ogre, esto para crear de forma paralela una escena de Ogre y una de PhysX. Terminado esto se crea o aplica gravedad a la escena junto son un suelo para que los objetos no caigan al vació.

Recien despues de la creación del mundo y la escena, junto con la gravedad y el suelo, se puede crear un cuerpo dentro de la escena. Para hacerlo es necesario dar un nombre al cuerpo, en este caso un string “mVehiculo” que corresponde al primer parámetro de la función “createBody”, una malla o mesh del objeto “vehiculo.mesh” que como se mecionó anteriormente es un archivo binario que contiene la información de la geometria del vehículo (la mesh) que utiliza Ogre para reproducirlo en pantalla.

El tercer parametro que recibe la función “createBody” es la forma (shape) que utiliza PhysX para realizar los calculos físicos sobre el cuerpo, en este caso a pesar que la malla del objeto es la representación de un vehículo, la forma física (modelo de colisión) que se asocia al cuerpo es un cubo de 4.7 metros de largo, 1.4 de alto y de 2 metros ancho. Por lo tanto, por muy realista que sea la mesh en la respresetanción del vehículo, el cuerpo se comportará como un cubo de las dimensiones antes dichas. El cuarto parámentro que recibe la función es la masa del cuerpo, en este caso se le da una masa de 1500 Kg. El útimo parametro es la posicón inicial del cuerpo dentro de la escena.

La clase NxOgre_Body (o body) posee dentro de sus miembros 2 atributos, el objeto mActor de la clase NxActor en ***PhysX*** y el objeto ***mEntity*** de la clase Entity en Ogre, a través de estos dos objetos es posible acceder a las todos los atributos y funcionalidades de los objetos de Ogre (entidades) y PhysX (actores).

CAPITULO 6. Definición De Requerimientos

En la siguiente sección se realizará la especificación de los requerimientos del simulador, utilizando como formato el Estándar de Requerimientos de Software ERS.

6.1 Descripción del Software a Desarrollar

El software a desarrollar es un simulador de tráfico bajo un ambiente tridimensional. Más puntualmente para simular el flujo vehicular urbano en intersecciones señalizadas por semáforos. La aplicación SIMTIS, ***Sim**ulador **T**ráfico **I**ntersecciones **S**eñalizadas*, permitirá observar visualmente en tres dimensiones el comportamiento del flujo de vehículos al entrar y salir de una intersección, así como también el comportamiento individual de cada vehículo. El simulador dará la posibilidad de ajustar diferentes tipos de parámetros relacionados con el comportamiento del sistema (flujo de tráfico) y entregará un reporte al finalizar la simulación con velocidades promedio, tiempos de demora a que fueron sometidos los vehículos en espera de cruzar la intersección.

6.1.1 Objetivo

6.1.1.1 Objetivo General

Construir un software que realice una simulación digital en tres dimensiones del comportamiento del flujo de tráfico en intersecciones señalizadas por semáforos.

6.1.1.2 Objetivos Específicos

- Implementar modelos de micro simulación de tráfico para conseguir un comportamiento del sistema (flujo vehicular) adecuado, cercano a la realidad. Los modelos de micro simulación a implementar son: Modelo de Gipps de vehículo siguiente y Modelo de aceleración de influencia señal-intersección.

- Integrar a la aplicación un motor gráfico que permita obtener la representación tridimensional de los objetos dentro de la simulación.

- Integrar a la aplicación un motor físico para realizar todos los cálculos físicos necesarios en la simulación de tal manera de entregar a los vehículos en su dinámica comportamientos cercanos a la realidad.

- Recibir como información de entrada todos los parámetros necesarios para la realización de las simulaciones.

- Construir una interfaz gráfica de usuario amigable fácil de manejar y entender, para permitir al usuario realizar todos los ajustes necesarios y deseables de los parámetros sobre los cuales se basarán las simulaciones.

- Entregar información por pantalla sobre resultados de simulaciones y almacenar en formato persistente.

- Permitir a la aplicación trabajar con diferentes intersecciones o variaciones de una misma intersección.

6.1.1.3 Alcances y Limites

Quizás la característica que diferencia este simulador de otros relacionados con el flujo vehicular es su capacidad de mostrar el comportamiento del sistema, y sus componentes por separado, de una forma amigable en un escenario tridimensional. Ya que la gran mayoría de los simuladores en el área se limitan a entregar resultados finales como números y estadísticas. La posibilidad de observar el flujo de tráfico y a los vehículos en forma individual puede entregar información de carácter relevante a los usuarios que buscan soluciones a problemas de congestión en intersecciones problemáticas y facilitar el análisis del los resultados obtenidos de cada simulación mejorando el proceso en la toma de decisiones.

El simulador no considera implementar modelos de adelantamiento o cambios de carril, así como tampoco el manejo de colisiones entre vehículos. También realiza las abstracciones necesarias en la dinámica de vehículos como por ejemplo el proceso de aplicación de torque a las ruedas.

6.1.1.4 Definiciones, Siglas y Abreviaciones

R_SW_99 : Requerimientos de interfaz de usuario, número secuencial desde 0 a 99.

FUN_99 : Función del software, número secuencial desde 0 a 99.

R_FUN_99 : Requerimientos funcionales del sistema, número secuencial desde 0 a 99.

DE_99 : Datos de entrada, número secuencial desde 0 a 99.

IS_99 : Información de salida, número secuencial desde 0 a 99.

6.2 Descripción Global del Producto

6.2.1 Interfaz de Software

Nombre	Tipo	Descripción	Versión	Rol
Ogre3D	Motor Gráfico	Entrega las capacidades para trabajar con la parte gráfica 3D.	1.2.4	Crear y controlar los objetos gráficos, a nivel gráfico.
PhysX	Motor Físico	Simulador físico	2.6.2	Realizar todos los cálculos físicos de la simulación.
NxOgre	Envoltorio	Envoltorio del motor gráfico y físico	0.4RC2	Permitir el uso transparente del motor gráfico y físico al mismo tiempo.
CEGUI	GUI	API´s, para construcción de GUI's.	0.5-0-RC2	Crear y controlar la interfaz de usuario del simulador.

6.2.2 Clasificación de Usuarios

Usuario	Nivel Acceso Sistema	Nivel Acceso Información	Nivel de Conocimiento
Usuario SW	Total	Total	Medio

6.2.3 Interfaz de Usuario

Número Código	Grupo Usuario	Descripción
R_SW_01	Usuario SW	Utilización de combo box para selección de intersecciones disponibles para simular.
R_SW_02	Usuario SW	Utilización de botones para navegar por la GUI.
R_SW_03	Usuario SW	Utilización de Scroll bar para controlar el despliegue de información de entrada y salida.

6.2.4 Interfaz de Hardware

- Tarjeta de video SVGA 128 MB, monitor de 17", o más.
- Teclado estándar, o más.
- Mouse 2 botones, o más.

6.3 Requerimientos Específicos

6.3.1 Funciones del Producto

Principales funciones del software. Descomposición funcional global.

FUN_01 : Manejar interfaz de usuario.

FUN_02 : Crear la escena (intersección, vehículos).

FUN_03: Iniciar la simulación.

FUN_04 : Calcular la velocidad de los vehículos.

FUN_05: Dirigir el vehículo por su ruta de viaje.

FUN_06: Controlar los semáforos.

FUN_07: Detener la simulación.

FUN_08 : Generar informes.

6.3.2 Requerimientos Funcionales del Sistema

Número	Función del SW al que pertenece	Nombre o especificación	Descripción
R_FUN_01	FUN_04	Determinar Modelo de aceleración	Para un vehículo determinar que modelo de aceleración o micro simulación aplicar.
R_FUN_02	FUN_04	Determinar la existencia de un evento	Determinar si ocurrió algún evento dentro de la simulación al cual un conductor deba reaccionar.
R_FUN_03	FUN_04	Calcular velocidad	Calcular la velocidad que se debe aplicar a un vehículo.
R_FUN_04	FUN_04	Obtener velocidad	Obtener la velocidad actual que lleva un vehículo en un determinado momento.
R_FUN_05	FUN_07	Aplicar velocidad	Aplicar la velocidad calculada para un vehículo.
R_FUN_06	FUN_05	Conducir vehículo	Guiar el vehículo por su ruta de viaje predefinida.
R_FUN_07	FUN_02	Cargar vehículos	Cargar un nuevo vehículo para integrarlo a la simulación.
R_FUN_08	FUN_02	Cargar intersección	Cargar la intersección sobre la cual se realizará la simulación.
R_FUN_09	FUN_02	Cargar semáforos	Cargar los semáforos y ubicarlos en cada calle de la intersección.
R_FUN_10	FUN_03	Iniciar simulación	Dar inicio a la simulación.
R_FUN_11	FUN_02	Destruir vehículos	Destruir vehículos salen del sistema
R_FUN_12	FUN_06	Controlar semáforos	Controlar y sincronizar los semáforos en relación al tiempo transcurrido y realizar los cambios de luces cuando corresponda.

Número	Función del SW al que pertenece	Nombre o especificación	Descripción
R_FUN_13	FUN_04	Controlar los tiempos de intervalo de simulación	Controlar los tiempos de cada intervalo de simulación para determinar cuando se debe volver a calcular y aplicar una nueva velocidad. Dar inicio a un nuevo intervalo.
R_FUN_14	FUN_07	Controlar término de la simulación.	Controlar el tiempo transcurrido desde el inicio de la simulación para determinar la su finalización.
R_FUN_15	FUN_01	Capturar parámetros de entrada	Obtener los parámetros de entrada ingresados por el usuario, para aplicar en la simulación.
R_FUN_16	FUN_01	Validar parámetros de entrada	Validar los datos de entrada que ingresa el usuario según el tipo de dato que se pretenda conseguir.
R_FUN_17	FUN_02	Configurar parámetros	Configurar los parámetros obtenidos del usurario. Asignar tiempos a semáforos, asignar tiempos de reaccionar a conductores, asignar grado de aceptación a las normas. Asignar ruta de viaje.
R_FUN_18	FUN_02	Recuperar información sobre intersección	Recuperar información desde un archivo de texto sobre las características de la intersección a simular. Calles, pistas, rutas de viaje por pistas.
R_FUN_19	FUN_08	Generar informe tiempos de espera	Desplegar por pantalla los tiempos de espera promedio de los vehículos resultantes.
R_FUN_20	FUN_08	Generar informe cantidad de vehículos en el sistema	Desplegar información sobre el número de vehículos que ingresaron a sistema durante la simulación.
R_FUN_21	FUN_08	Generar informe velocidades promedio	Desplegar información sobre las velocidades promedios que registraron los vehículos al transitar por al intersección
R_FUN_22	FUN_08	Generar informe flujo vehicular	Desplegar información sobre los flujos de tráfico registrados en la intersección
R_FUN_23	FUN_08	Almacenar resultados simulación	Almacenar en un archivo de texto plano los resultados de la Simulación.

6.3.3 Interfaces Externas, Requerimientos de Información

Datos de entrada.

Código	Nombre Ítem	Detalle datos contenidos en ítem	Medio de entrada	Rango valido exactitud y/o tolerancia	Unidades de medida	Formato de datos
DE_01	Información General	Nombre simulación	Teclado	NO APLICABLE	NO APLICABLE	Texto largo no definido
		Tiempo simulación	Teclado	Valor positivo entero	Minutos	99
		Intersección a simular	Mouse	Intersecciones disponibles	NO APLICABLE	Texto largo no definido
DE_02	Parámetros Globales	Ciclo semáforo	Teclado	Valor positivo entero	Segundos	999
		Tiempo luz amarilla	Teclado	Valor positivo No especificado	Segundos	99,99
		Calle	Mouse	NO APLICABLE	NO APLICABLE	Texto largo no definido
		Tiempo luz verde	Teclado	Valor positivo No especificado	Segundos	99,99
		Tiempo luz roja	Teclado	Valor positivo No especificado	Segundos	99,99
		Velocidad máxima Vehículo Liviano	Teclado	Valor positivo No especificado	Km/h	99,99
DE_03	Parámetros por Pista, Tiempo de entrada al sistema	Tiempo de entrada constante	Teclado	Valor positivo No especificado	Segundos	99,99
		Tiempo de entrada aleatorio	Teclado	Valor positivo entero [1-99]	Segundos	99
		Tiempo de entrada según secuencia	Teclado	Lista de valores positivos	Segundos	99,99
DE_04	Parámetros Conductor	Tiempo de reacción	Teclado	Valor positivo No especificado	Segundos	9,99
		Grado acatamiento a normas	Teclado	0 <= grado >= 1	NO APLICABLE	0,99/1,0
		Distancia entre vehículos	Teclado	Valor positivo No especificado	Metros	99,9
		Velocidad Deseada	Teclado	Valor positivo No especificado	Km/h	99,99

Código	Nombre Ítem	Detalle datos contenidos en ítem	Medio de entrada	Rango valido exactitud y/o tolerancia	Unidades de medida	Formato de datos
DE_05	Características Intersección	Número de Calles	Archivo	Valor entero positivo No especificado	NO APLICABLE	9
		Número de Pistas	Archivo	Valor entero positivo No especificado	NO APLICABLE	9
		Ruta de viaje por la pista	Archivo	puntos en el espacio 3d, No especificado	NO APLICABLE	Arreglo de puntos, Vector3 (x,y,z)
		Línea de detención	Archivo	puntos en el espacio 3d, No especificado	NO APLICABLE	Arreglo de puntos, Vector3 (x,y,z)

Datos de Salida.

Código	Nombre del Ítem	Detalles de datos contenidos en ítem	Medio de salida
IS_01	Informe Simulación	Intersección, Calle, Pista, tiempo espera promedio, Flujo de vehículos, Velocidades promedio, Ciclo semáforo, Tiempo luz verde, Tiempo luz amarilla, Tiempo luz roja.	Por pantalla, archivo de texto

6.3.4 Requerimientos de Hardware

Tipo	Características Técnicas mínimas	Requisitos mínimos de desempeño	Rol
PC	Procesador: 2.5Ghz MRAM: 512 MB Tarjeta de Video: 128 MB Disco Duro: 10 GB Monitor: 17”	Entregar una simulación con movimiento de objetos fluidos. Considerando el procesamiento de la parte gráfica y cálculos físicos además de procesos de control.	Ejecutar las simulaciones

6.3.5 Restricciones Operacionales

Código	Descripción	Responsable	Ref. a req. Funcional e interfaces externas
REST_O_01	Las velocidades máximas permitidas a los vehículos livianos particulares en espacios urbanos debe ser considerara.	Reglamentación de tránsito vigente	FUN_04 DE_02
REST_O_02	Respetar la secuencia de señalización de semáforos y el significado de cada señalización.	Reglamentación de tránsito vigente	FUN_06 DE_02

CAPITULO 7. Diseño Global del Software

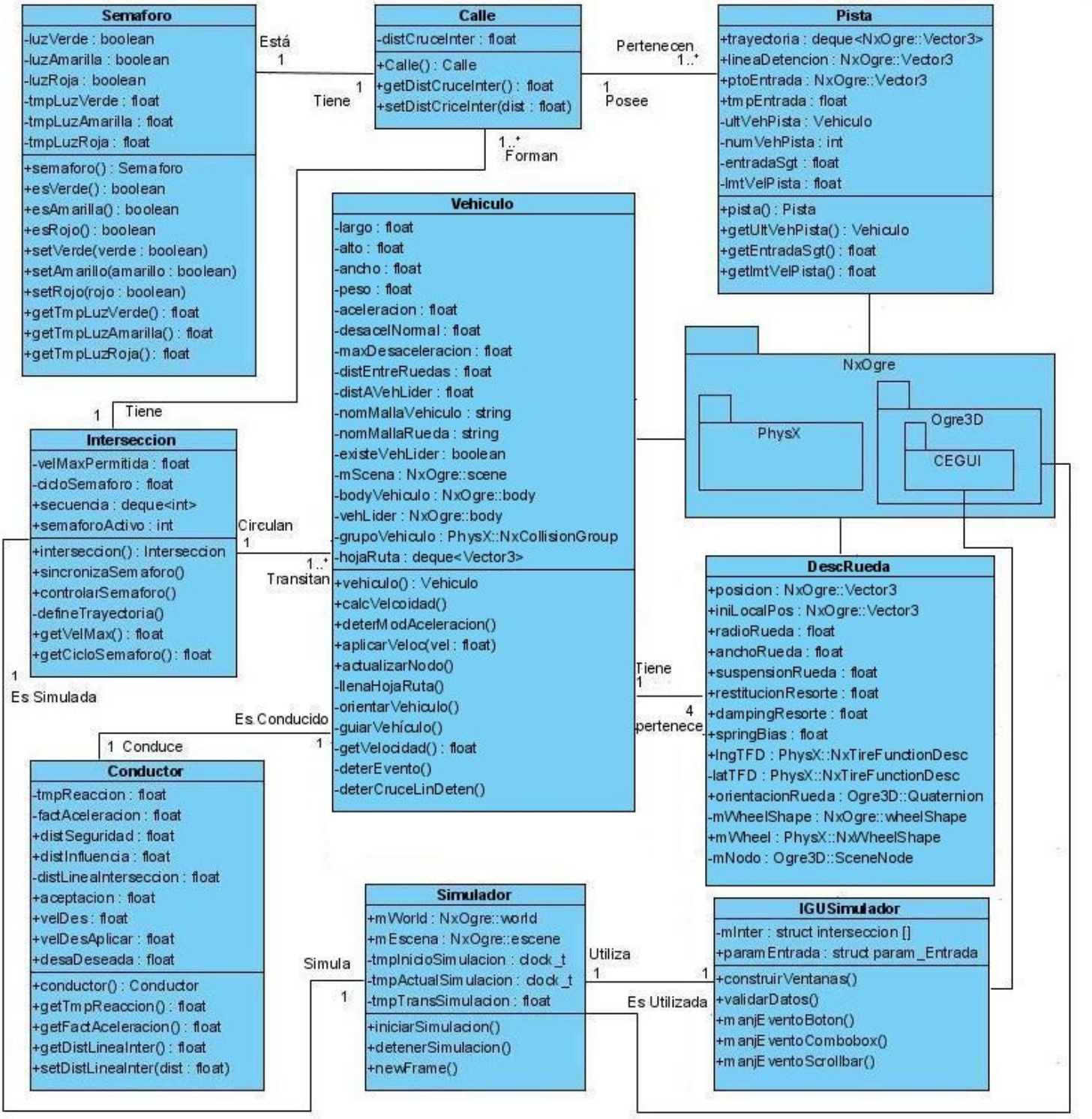
Para documentar el diseño del simulador se toma como base el estándar de diseño con algunas modificaciones. Estas modificaciones son la integración al estándar de diseño, en su sección inicial, de la fase de Análisis de la metodología de modelamiento orientado a objetos OMT, ya que esta entrega una clara referencia a la utilización de modelos de objeto y modelos dinámicos.

7.1 Diseño Funcional del Software

7.1.1 Modelo de Objeto

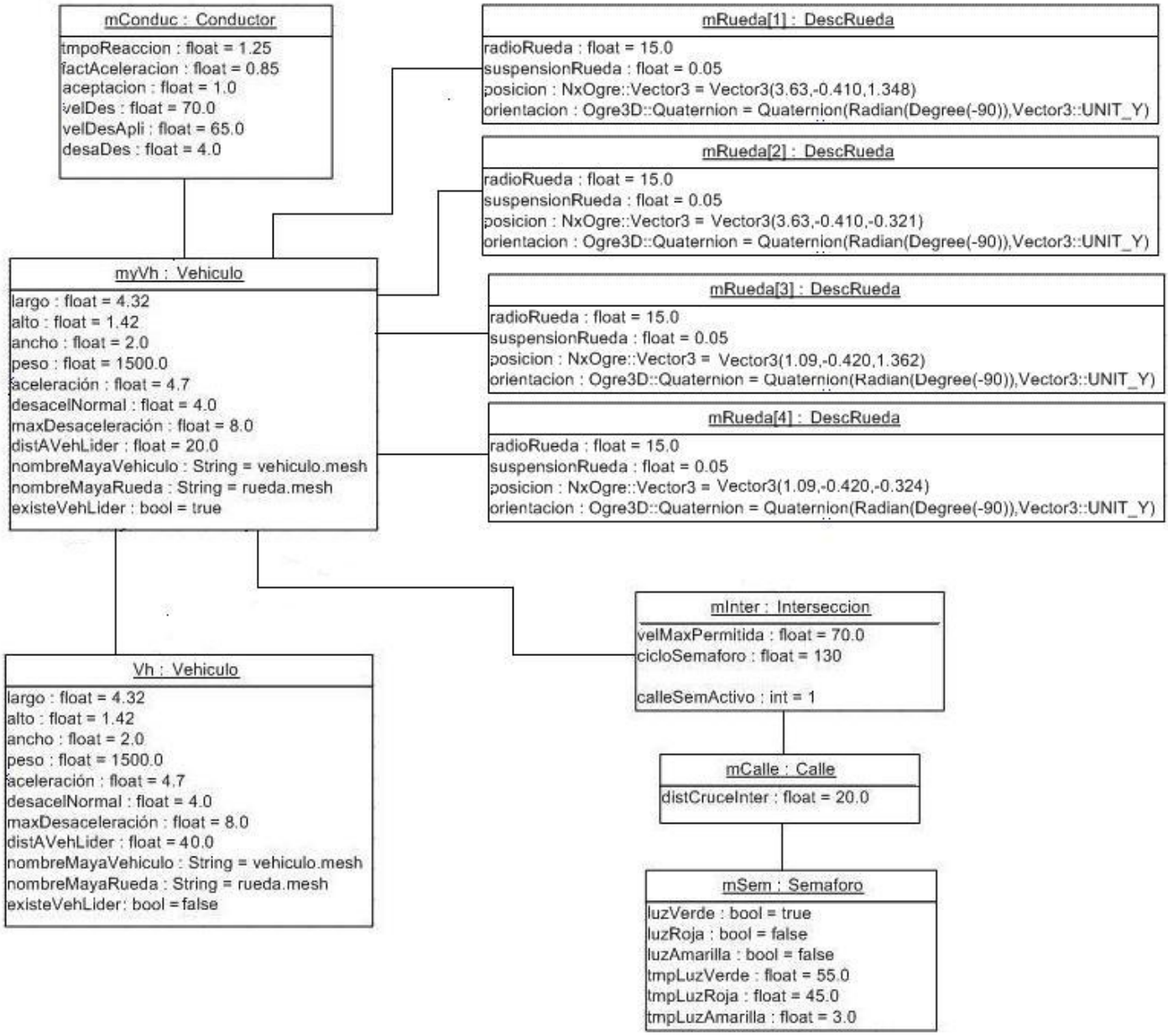
Para el modelo de objeto se realizan dos diagramas. El diagrama de clases, el diagrama de objetos. Junto con un diccionario de datos para formalizar el significado de las clases sus atributos y métodos.

7.1.1.1 Diagrama de clases



7.1.1.2 Diagrama de Objetos

Diagrama de objetos con las instancias y los atributos más significativos.



7.1.1.3 Diccionario de datos

Clase	Atributos	Operaciones
Interseccion: Agrupa toda la información concerniente a la intersección que será el escenario para simulación	***velMaxPermitida:*** Velocidad máxima permitida, por la reglamentación de tránsito vigente, para el desplazamiento de vehículos. ***cicloSemaforo:*** Tiempo que se demora en completarse un ciclo completo de semáforos en una intersección. ***secuencia:*** Orden en la secuencia de los semáforos, quien va primero, segundo, tercero. ***semaforoActivo:*** Almacena información del semáforo que esta con flujo activo. Luz verde.	***interseccion():*** Constructor de la clase. ***sincronizaSemaforo():*** Define la secuencia de los semáforos, asigna los tiempos para cada intervalo. ***controlarSemaforo():*** Controla los tiempos de los semáforos, realiza los cambios de luces y de semáforos (activo/inactivo). ***defineTrayectoria():*** Define la trayectoria o ruta de viaje, que sigue un vehículo, para cada una de las calles y pistas. ***getVelMax():*** Obtiene la velocidad máxima ***getCicloSemaforo():*** Obtiene el ciclo de semáforo.
IGUSimulador: Clase encargada de manejar la interfaz gráfica de usuario.	***mInter:*** Arreglo que almacena todas las intersecciones , y sus características, disponibles para simular. ***paramEntrada:*** Estructura que registra todos los parámetros ingresados por el usuario.	***iguSimulador():*** Constructor de la clase. ***construirVentanas():*** Construye las ventanas de la interfaz gráfica. ***validarDatos():*** Valida los datos ingresados por el usuario para que correspondan con los tipos adecuados. ***manjEventoBoton():*** Maneja los eventos relacionados a los botones utilizados en las ventanas de la interfaz. ***manjEventoCombobox():*** Maneja los eventos gatillados por el uso de los combobox utilizados en la interfaz. ***manjEventoScrollbar():*** Maneja los eventos gatillados por el uso del scrollbar en la interfaz .

Clase	Atributos	Operaciones
Simulador: Clase controladora de la simulación.	***mWorld:*** Contiene el mundo desde el punto de vista de NxOgre en donde suceden las cosas (tiene lugar la simulación). ***mEscena:*** Escena dentro del mundo donde se crearan los objetos 3d. ***tmpInicioSimulacion :*** Tiempo (hora) en el que se inicia la simulación. ***tmpActualSimulacion :*** Tiempo actual dentro de la simulación. ***tmpTransSimulacion:*** Tiempo transcurrido desde el inicio de la simulación.	***iniciarSimulacion():*** Da inicio a la simulación. ***detenerSimulacion():*** Detiene la simulación cuando el tiempo especificado haya terminado o se cancele la simulación. ***newFrame():*** Inicia repetitivamente un nuevo marco (iteración) para la simulación.
Conductor: Clase que contiene toda la información y operaciones necesaria sobre un conductor para realizar la simulación.	***tmpReaccion :*** Tiempo de reacción del conductor, engloba el tiempo de percepción-reacción. **distSeguridad:** Contiene la distancia de seguridad para detención segura. ***distInfluencia:*** Almacena la distancia de influencia para un conductor. ***distLineaInterseccion:*** Almacena la distancia hacia la línea de detención desde la posición donde se encuentra el vehículo ***aceptacion :*** Contiene el grado de aceptación a las normas del conductor. ***velDes:*** Contiene la velocidad deseada de desplazamiento del conductor. ***velDesAplicar:*** Almacena la velocidad deseada que se aplicará realmente (depende de la aceptación). ***desaDeseada:*** Almacena la desaceleración que desearía aplicar el conductor.	***conductor():*** Constructor de la clase ***getTmpReaccion():*** Obtiene el tiempo de reacción del conductor. ***getDistLineaInter():*** Consigue la distancia hacia la línea de detención. ***setDistLineaInter():*** Ajusta la distancia hacia la línea de detención

Clase	Atributos	Operaciones
DescRueda: Clase que representa la rueda de un vehículo, contiene todas las características necesarias para simular una rueda	***posición:*** Contiene la posición, como punto tridimensional, de la rueda en relación con el chasis del vehículo. ***radioRueda:*** Radio de la rueda en metros. ***anchoRueda:*** Ancho de la rueda en metros. ***suspensionRueda:*** Largo de la suspensión en metros. ***restitucionResorte:*** Fuerza con la que se restituye el resorte de la suspensión. ***dampingResorte:*** Resistencia a la restitución del resorte. ***springBias:*** Inclinación de la suspensión. ***lngTFD:*** Fricción longitudinal de la rueda. ***latTFD:*** Fricción lateral de la rueda. ***orientacionRueda:*** Almacena la dirección hacia donde esta mirando la rueda. ***mWheelShape:*** Contiene la forma (modelo de colisión) para la rueda dada por NxOgre. ***mWheel:*** Contiene la forma (modelo de colisión) para la rueda dada por PhysX. ***mNodo:*** Almacena el nodo que controla la representación gráfica de la rueda.	

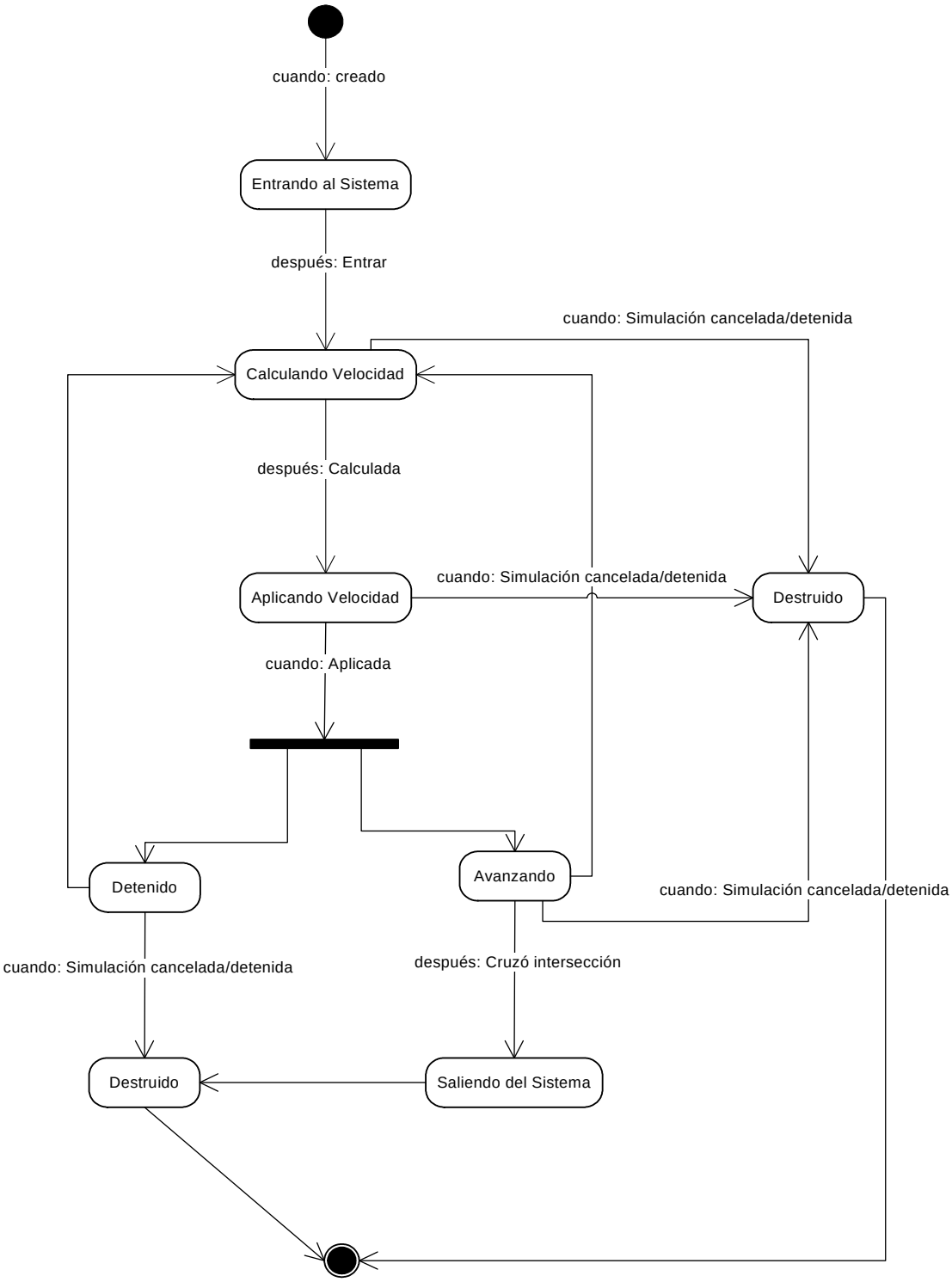
Clase	Atributos	Operaciones
Vehiculo: Representa y contiene toda la información necesaria sobre un vehículo.	***largo:*** Largo del vehículo en metros. ***ancho:*** Ancho del vehículo en metros. ***alto:*** Alto del vehículo en metros. ***peso:*** Peso del vehículo en KG. ***aceleración:*** Aceleración del vehículo. ***desacelNormal:*** Desaceleración normal del vehículo. ***maxDesaceleracion:*** Máxima desaceleración del vehículo. ***distEntreRuedas:*** Distancia entre las ruedas delanteras. ***distAVehLider:*** Distancia entre el vehículo y su vehículo líder. ***nomMallaVehiculo:*** Nombre del archivo que contiene la malla 3D del chasis del vehículo. ***nomMallaRueda:*** Nombre del archivo que contiene la malla 3D de la rueda. ***existeVehLider:*** Almacena la información de existencia o no de un vehículo líder. ***mEscena:*** Escena en la que el vehículo es creado. ***bodyVehiculo:*** Cuerpo del vehículo, el cuerpo contiene una forma y una malla. ***vehLider:*** Contiene el vehículo líder. ***grupoVehiculo:*** Contiene el grupo de body's al que pertenece el vehículo. ***hojaRuta:*** Contiene la hoja de ruta que debe seguir el vehículo.	***vehiculo():*** Constructor de la clase. ***calcVelocidad():*** Calcula la velocidad que el vehículo debe desarrollar en el siguiente intervalo de simulación. ***deterModAceleracion():*** Determina el modelo de aceleración o el modelo de micro simulación que se debe aplicar al vehículo. ***aplicarVeloc:*** Aplica la velocidad calculada al vehículo. ***actualizarNodo():*** Actualiza los nodos de las ruedas, para que la representación gráfica (malla) "mesh" siga los movimientos del modelo de colisión física "whellshape". ***llenaHojaRuta():*** Llena la hoja de ruta de vehículo. ***orientarVehiculo() :*** Orienta al vehículo después de crearlo para que éste mire hacia la intersección. ***guiarVehículo():*** Conduce al vehículo por su trayectoria, hoja de ruta. ***getVelocidad():*** Obtiene la velocidad de un vehículo. ***deterEvento():*** Determina si ha ocurrido algún evento al cual el conductor deba reaccionar. ***deterCruceLinDeten():*** Determina si el vehículo ha cruzado o no la línea de detención de la calle.

Clase	Atributos	Operaciones
Semaforo: Representa un semáforo para una calle con todas sus características.	***luzVerde:*** Almacena la información de si un semáforo se encuentra o no con luz verde. ***luzAmarilla:*** Almacena la información de si un semáforo se encuentra o no con luz amarilla. ***luzRoja:*** Almacena la información de si un semáforo se encuentra o no con luz roja. ***tmpLuzVerde:*** Tiempo asignado para luz verde. ***tmpLuzAmarilla:*** Tiempo asignado para luz amarilla. ***tmpLuzRoja:*** Tiempo asignado para luz roja.	***semaforo():*** Constructor de la clase. ***esVerde():*** Retorna si un semáforo está o no con luz vede. ***esAmarilla():*** Retorna si un semáforo esáa o no con luz amarilla. ***esRojo():*** Retorna si un semáforo está o no con luz roja. ***setVerde():*** Ajusta el estado del atributo luzVerde. ***setRojo():*** Ajusta el estado del atributo luzRoja. ***setAmarillo():*** Ajusta el estado del atributo luzAmarilla. ***getTmpLuzVerde():*** Obtiene el tiempo asignado a la luz verde. ***getTmpLuzRoja():*** Obtiene el tiempo asignado a la luz roja. ***getTmpLuzAmarila():*** Obtiene el tiempo asignado a la luz amarilla.
Calle: Representa una calle de la intersección	***distCruceInter:*** Distancia entre la línea de detención de una calle y el término del cruce de la intersección. El espacio propiamente tal de cruce.	***Calle():*** Constructor de la clase ***getDistCruceInter():*** Obtiene la distancia de cruce de la intersección. ***setDistCruceInter():*** Ajusta la distancia de cruce.

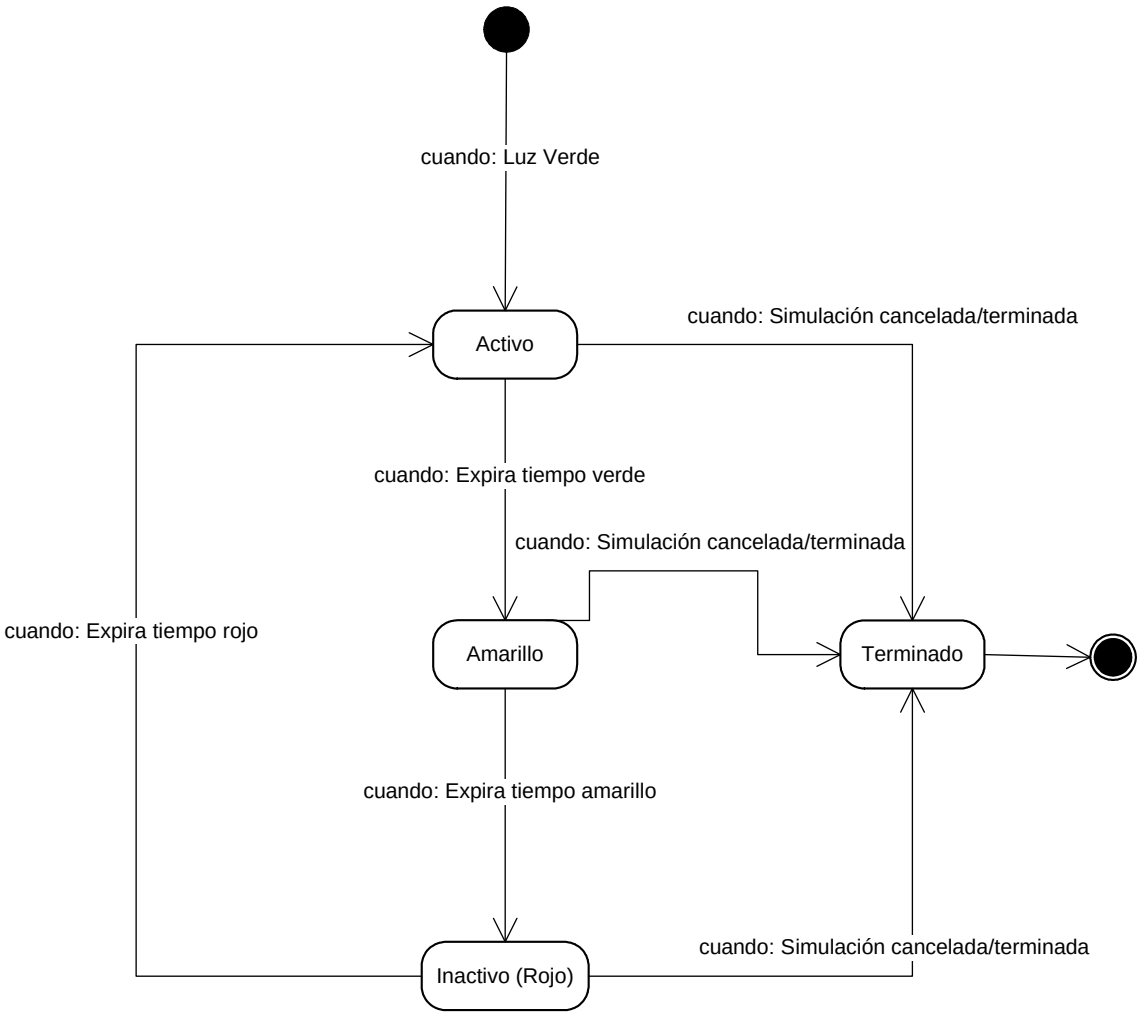
Clase	Atributos	Operaciones
Pista: Agrupa toda la información sobre una pista de una de la calle	***trayectoria:*** almacena la trayectoria que debe seguir un vehículo que viaje por esta pista. ***lineaDetencion :*** Contiene el punto de la línea de detención de la calle donde deben detenerse los vehículos. ***ptoEntrada:*** Almacena el punto de entrada de los vehículos al sistema en esta pista. ***tmpEntrada:*** El tiempo que debe esperar un vehículo para entrar al sistema después crear el último vehículo en la pista. Cada cuanto tiempo se crea un vehículo. ***ultVehPista:*** Mantiene registro del ultimo vehículo creado en la pista. ***numVehPista:*** Contador del número de vehículos creados en la pista. ***entradaSgt:*** Tiempo faltante para crear el próximo vehículo en la pista. ***lmtVelPista:*** Almacena la velocidad limite, en caso de existir, para la pista .	***pista():*** Constructor de la clase ***getUltVehPista():*** Consigue el último vehículo creado en la pista. ***getEntradaSgt():*** Consigue el tiempo faltante para crear un nuevo auto en la pista. ***getlmtVelPista():*** Obtiene la velocidad límite de la pista.

7.1.2 Modelo Dinámico

7.1.2.1 Diagrama De Transición De Estados Para Una Instancia Vehiculo.

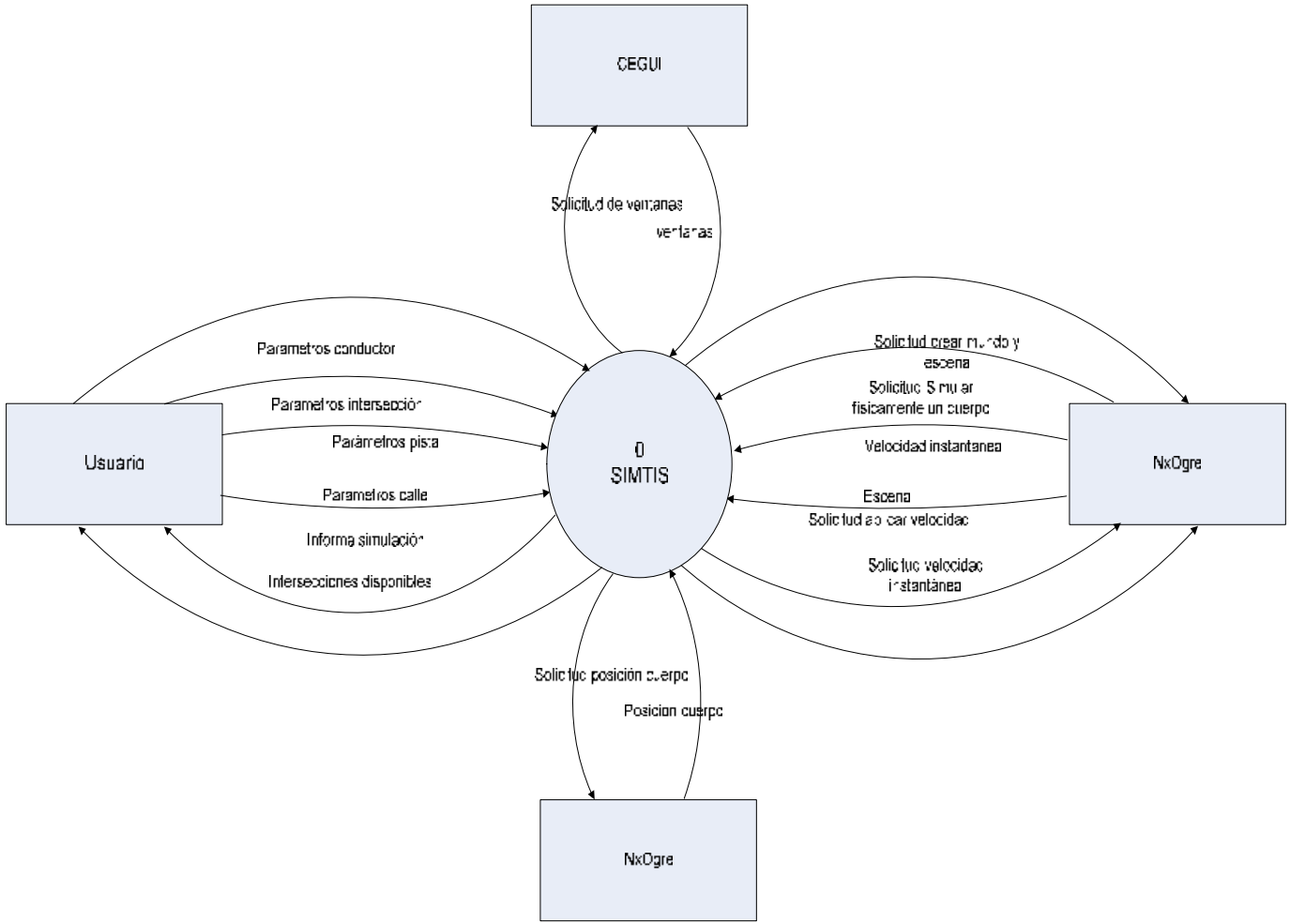


7.1.2.2 Diagrama De Transición De Estados Para Una Instancia Semáforo.

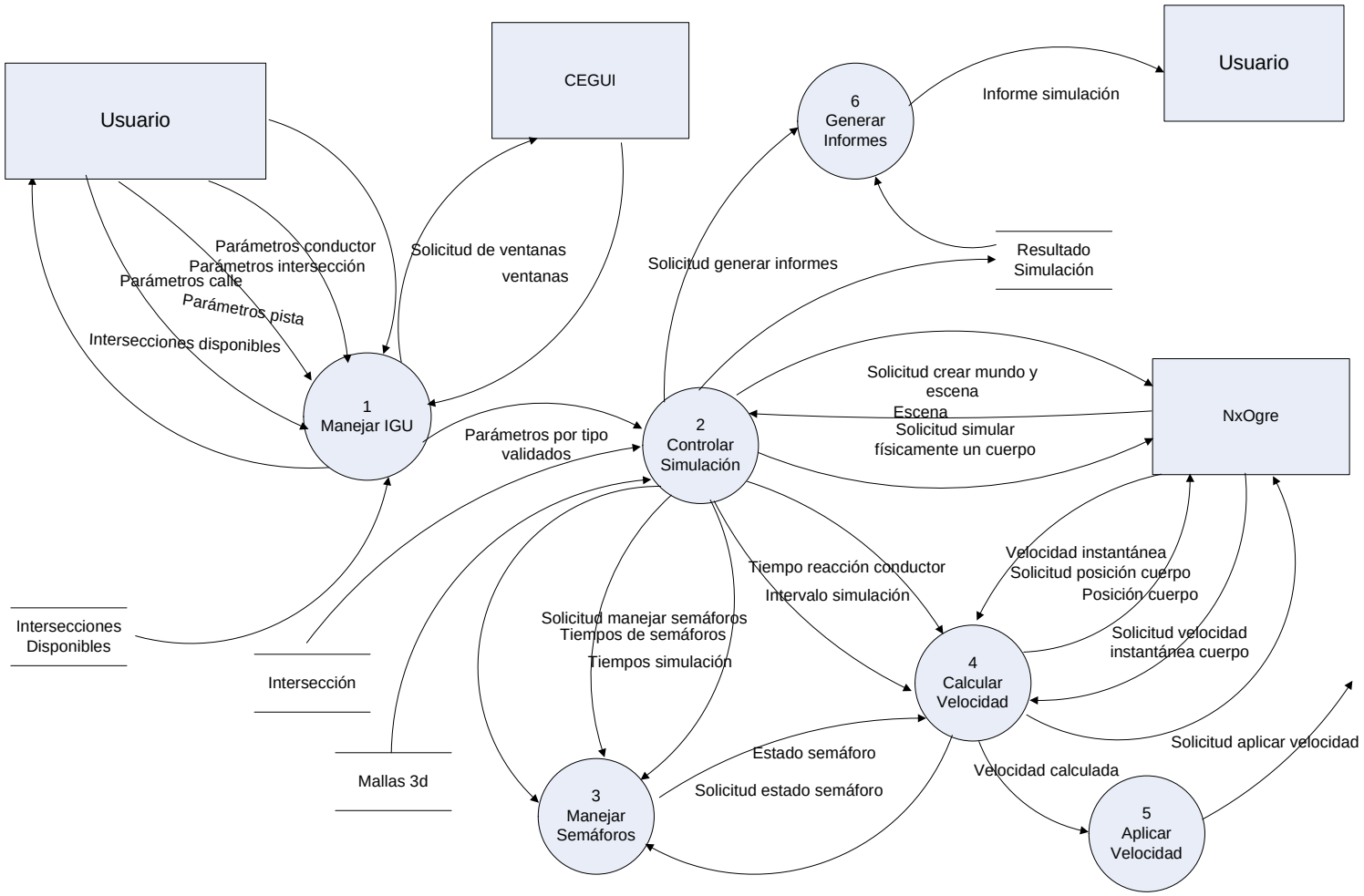


7.1.3 Modelo Funcional

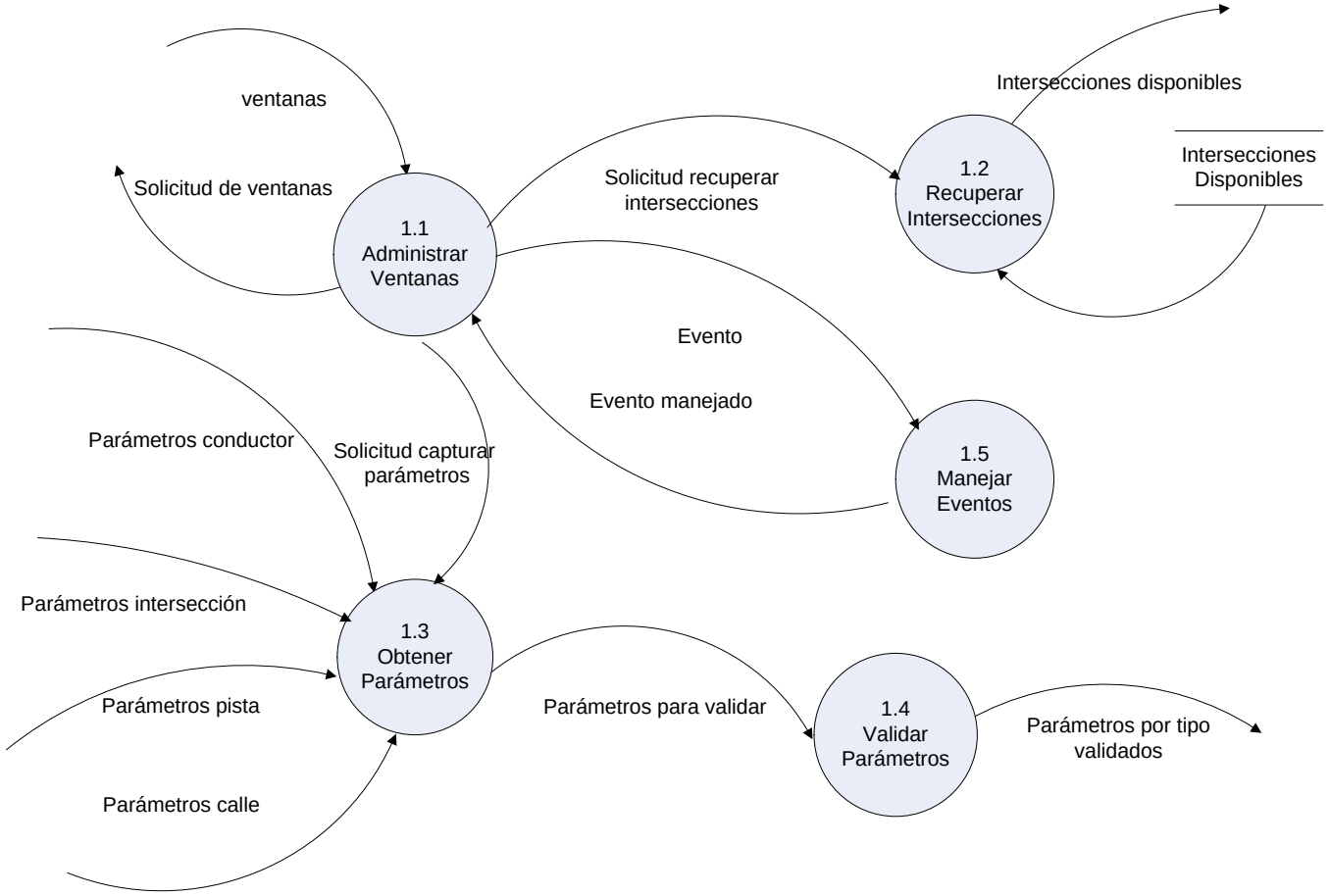
7.1.3.1 Diagrama de Contexto



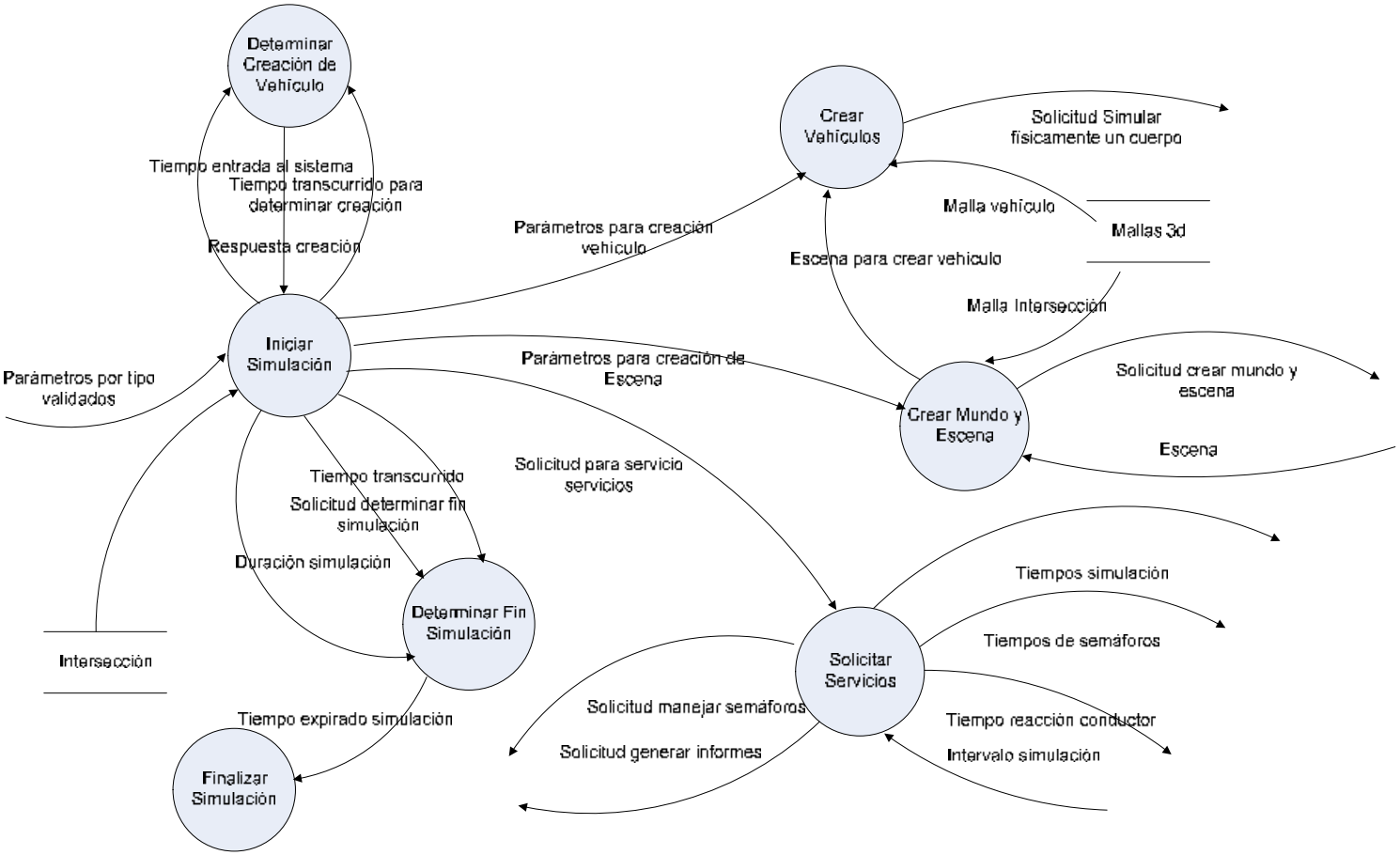
7.1.3.2 Diagrama De Flujo de Datos Nivel Superior



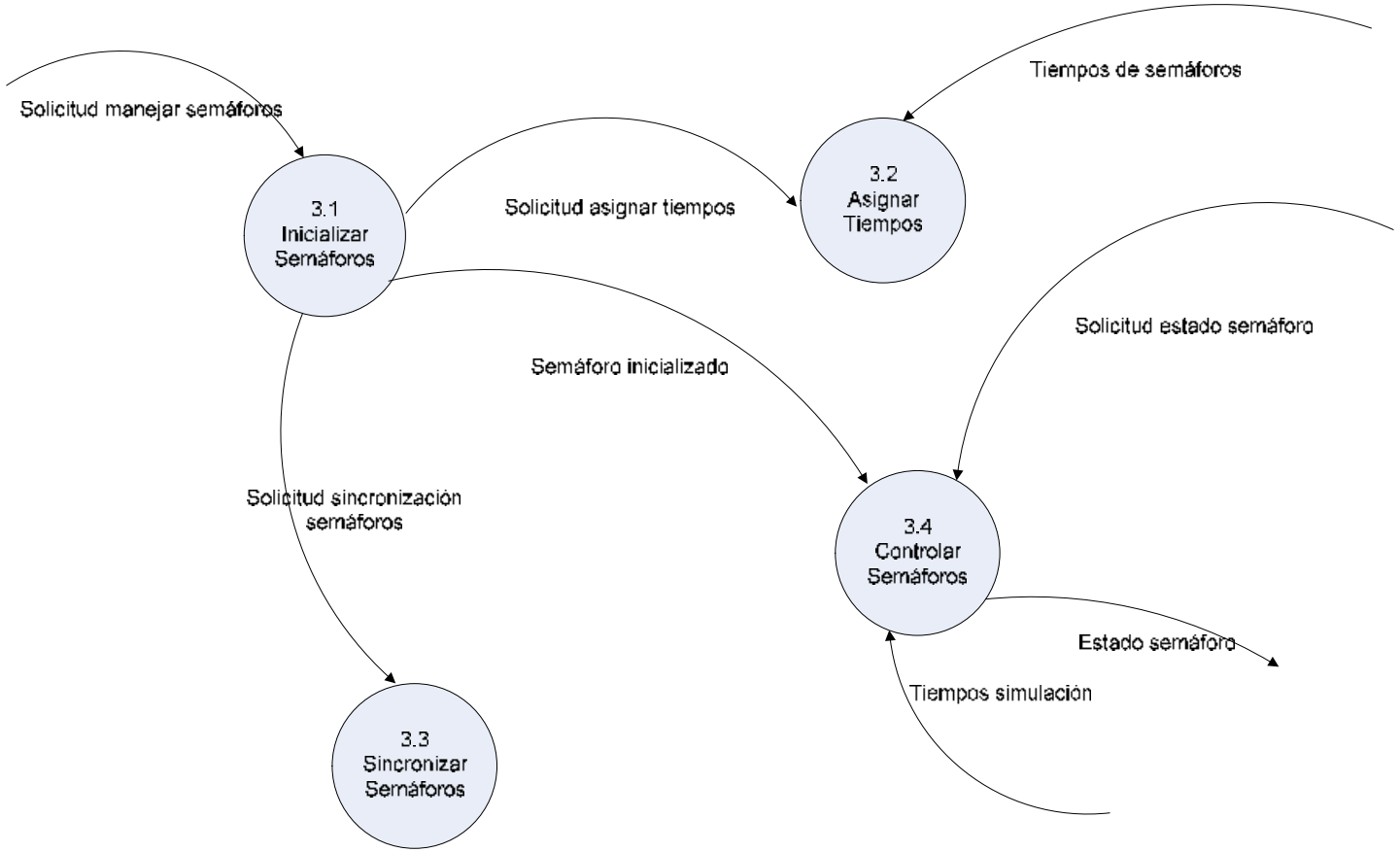
7.1.3.3 Diagrama De Flujo de Datos Nivel De Detalle “Manejar IGU”



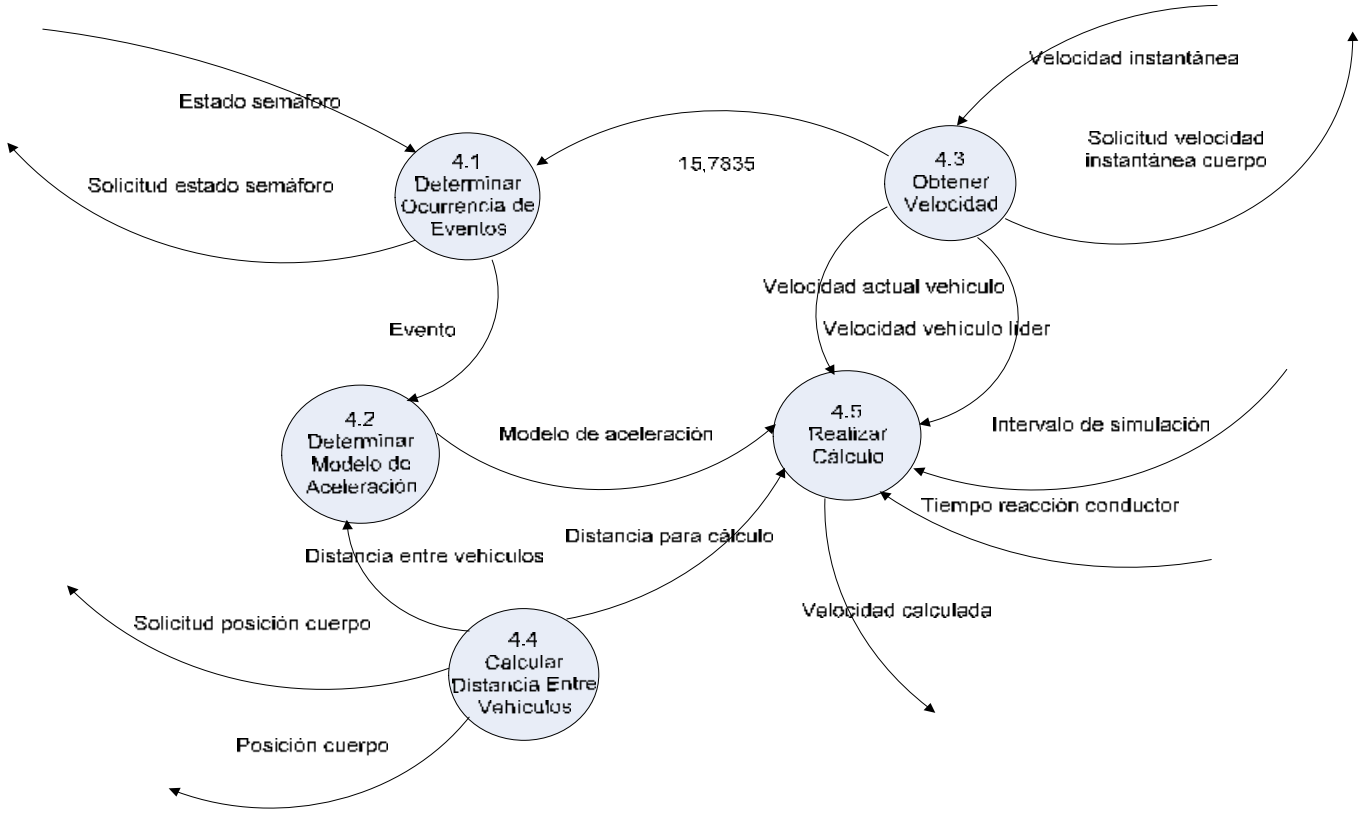
7.1.3.4 Diagrama De Flujo de Datos Nivel De Detalle “Controlar Simulación”



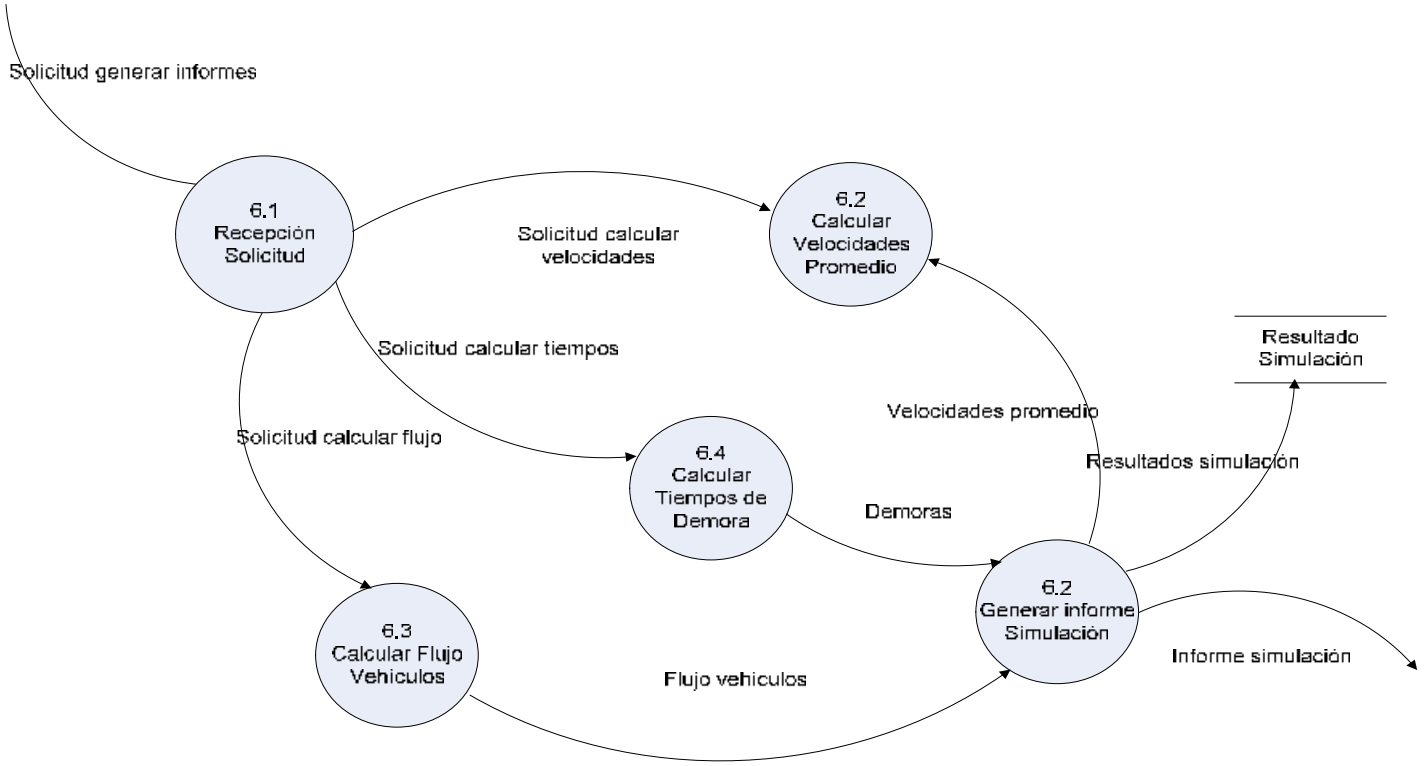
7.1.3.5 Diagrama De Flujo de Datos Nivel De Detalle “Manejar Semáforos”



7.1.3.6 Diagrama De Flujo de Datos Nivel De Detalle “Calcular Velocidad”



7.1.3.7 Diagrama De Flujo de Datos Nivel De Detalle “Generar Informes”



7.1.4 Diccionario De Datos

Nombre Entidad	Descripción
Usuario	Representa al usuario operador de la aplicación, el que ingresa todos los datos necesarios para realizar las simulaciones y al que se le entregan los informes con los resultados de las simulaciones.
NxOgre	Representa al sistema envoltorio NxOgre encargado de integrar los motores físicos y gráficos. A través de este envoltorio se tiene acceso a las capacidades de ambos motores para realizar las simulaciones físicas y su representación gráfica.
CEGUI	Representa el sistema utilizado para crear y manipular la interfaz gráfica de usuario.

N°	Nombre Función	Descripción
0	SIMTIS	Representa al sistema definido por la aplicación, "Simulador Tráfico Intersecciones Señalizadas", y todas sus características y operaciones. Interactúa con el Usuario que le suministra la información necesaria para operar y con los sistemas NxOgre, para realizar la simulación física y su representación gráfica, y con CEGUI para conseguir operar su interfaz gráfica.
1	Manejar IGU	Función encargada de manejar la interfaz gráfica de usuario, crear las ventanas, maneja los eventos de las ventanas, captura los parámetros ingresados por el usuario, valida los datos. Se comunica con el sistema de ventanas CEGUI para realizar su tarea.
2	Controlar Simulación	Función encargada de controlar la simulación: Inicia la simulación, crea la escena, crea los objetos de la escena, determina cuando crear un nuevo vehículo para ingresarlo al sistema, solicita servicios para controlar los semáforos, calcular las velocidades de los vehículos.
3	Manejar Semáforos	Función encargada de inicializar, los semáforos y controlar su funcionamiento. Asigna los tiempos a los semáforos, su secuencia de rotación, los cambios de luces.
4	Calcular Velocidad	Calcula la velocidad que deberá desarrollar un vehículo en la nueva iteración de la simulación. Determina la existencia de un vehículo líder, la distancia entre el vehículo y su líder, la velocidad propia y la del vehículo líder (solicita las velocidades a NxOgre), la existencia de algún evento al que deba reaccionar un conductor. Determina el modelo de aceleración o micro simulación que se utilizará para calcular la velocidad dependiendo de todas las características mencionadas anteriormente.
5	Aplicar Velocidad	Función encargada de aplicar al vehículo la velocidad calculada. Envía un mensaje a NxOgre para que finalmente este aplique y simule físicamente la nueva velocidad.

6	Generar Informes	Genera los informes con los resultados finales de la simulación. El informe generado contiene información de s velocidades promedios de los vehículos, sus tiempos de espera y el flujo vehicular.

Nombre Almacén	Descripción	Datos que lo componen
Intersecciones Disponibles	Archivo que contiene todas las intersecciones disponibles para ser utilizadas en las simulaciones.	Nombre intersección, Número de calles, Nombre calle, número de pistas, Nombre archivo intersección, Nombre archivo malla 3d.
Intersección	Archivo que contiene toda la información sobre una intersección	Punto ubicación de los semáforos, punto ubicación de la línea de detención, largo del cruce de la intersección, Calles, Hoja de ruta por pistas.
Mallas 3d	Contiene las mayas de los objetos tridimensionales modelados, además de las texturas utilizadas por estos.	Malla intersecciones, Malla chasis vehículo, malla rueda, texturas.

Nombre Flujo	Descripción	Datos que lo componen	Origen	Destino
Parámetros Conductor	Los parámetros que se necesitan para simular el comportamiento del conductor.	Tiempo de Reacción, Grado aceptación a las normas, Velocidad deseada para viajar de parte del conductor	Usuario	Manejar IGU
Parámetros intersección	Los parámetros necesarios para representar una intersección.	Velocidad máxima, ciclo de los semáforos,	Usuario	Manejar IGU
Parámetros calle	Los parámetros necesarios para trabajar con una calle.	Tiempo semáforos, tiempos en cada luz de semáforo.	Usuario	Manejar IGU
Parámetros pista	Los parámetros necesarios para trabajar con una pista de una calle.	Tiempo de entrada de vehículos a la pista. Cada cuanto tiempo se crea un vehículo en la pista	Usuario	Manejar IGU
Intersecciones disponibles	Una lista con todas las intersecciones disponibles para utilizar en una simulación. Para ser seleccionada por el usuario	Nombre intersección	Manejar IGU	Usuario

Solicitud de Ventanas	solicitud de gestar una ventana	Nombre ventana, posición, tamaño, tipo apariencia, artefactos.	Manejar IGU	CEGUI
Ventana	Retorno de la ventana gestada.	Dirección de la ventana	CEGUI	Manejar IGU
Nombre Flujo	Descripción	Datos que lo componen	Origen	Destino
Solicitud Generar Informes	Solicitud para generar informes después de finalizar la intersección.	mensaje	Controlar Simulación	Generar Informe
Solicitud manejar semáforos	Solicitud para manejar y controlar los semáforos.	mensaje	Controlar Simulación	Manejar Semáforos
Tiempos de semáforos	Tiempos asignados a cada semáforo.	Ciclo semáforo, tiempo luz roja, tiempo luz verde, tiempo luz amarilla.	Controlar Simulación	Manejar Semáforos
Tiempo simulación	Tiempos utilizados en la simulación.	Tiempo actual dentro de la simulación, tiempo transcurrido de la simulación.	Controlar Simulación	Manejar Semáforos
Tiempo reacción conductor	Tiempo de reacción del conductor ante la ocurrencia de algún evento.	Tiempo reacción	Controlar Simulación	Calcular Velocidad
Intervalo de simulación	Tiempo que toma realizar una iteración de la simulación de tráfico.	Intervalo simulación	Controlar Simulación	Calcular Velocidad
Solicitud crear mundo y escena	Solicitud a NxOgre para que cree el mundo y la escena donde ocurrirán las cosas, la simulación.	Tipo de escena	Controlar Simulación	NxOgre
Escena	Escena retorna por NxOgre.	Escena	NxOgre	Controlar Simulación
Solicitud simular físicamente un cuerpo	Solicitar a NxOgre que realice todos los cálculos físicos para el cuerpo y los replique en el modelo gráfico	El cuerpo del vehículo, es decir, “body” que está compuesto por una forma (modelo físico) y una malla (modelo gráfico).	Controlar Simulación	NxOgre
Solicitud estado semáforo	Solicitud para tener conocimiento de estado de un semáforo.	mensaje	Calcular Velocidad	Manejar Semáforos
Estado semáforo	Contiene el estado del semáforo	Estado del semáforo	Manejar Semáforos	Calcular Velocidad

Nombre Flujo	Descripción	Datos que lo componen	Origen	Destino
Solicitud posición cuerpo	Solicitud a NxOgre para conocer la posición de un cuerpo.	El cuerpo (body)	Calcular Velocidad	NxOgre
Posición cuerpo	Respuesta a solicitud posición de un cuerpo con la posición del cuerpo.	Posición del cuerpo.	NxOgre	Calcular Velocidad
Solicitud aplicar velocidad	Solicitud a NxOgre para que aplique la velocidad sobre el cuerpo y simule su comportamiento.	Cuerpo, velocidad	Aplicar Velocidad	NxOgre
Informe simulación	Informe sobre los resultados obtenidos por la simulación	Tiempos de demora, velocidades promedio, ciclo de semáforo, tiempos de semáforos, número de vehículos en la simulación.	Generar Informes	Usuario
Parámetros por tipo validados	Parámetros capturados desde la interfaz gráfica ya validados.	Parámetros por conductor, por intersección, por calle, por pista	Manejar IGU	Controlar Simulación
Velocidad calculada	Contiene la velocidad calculada para un vehículo.	Velocidad calculada	Calcular Velocidad	Aplicar Velocidad
Solicitud velocidad instantánea de un cuerpo	Solicitud para saber cual es la velocidad instantánea de un cuerpo.	El cuerpo (body)	Calcular Velocidad	NxOgre
velocidad instantánea	Velocidad instantánea de un cuerpo.	Velocidad instantánea	NxOgre	Calcular Velocidad

7.2 Especificación De Procesos

<table>
<tr><td>N° Proceso: 2</td><td>Nombre Proceso: Controlar Simulación</td><td>Proceso Padre:
Manejar IGU</td></tr>
<tr><td>Flujo de datos de entrada</td><td rowspan="11">/INICIO

Crear el mundo y la escena;
Agregar gravedad a la escena;

Crear una instancia clase Intersección;

Marcar hora inicio simulación;
Iniciar simulación;

Repetir hasta finalizar la simulación

Conseguir Tiempo Actual;
Determinar el tiempo transcurrido;
Controlar semáforos utilizando tiempos;

Si tiempo transcurrido > tiempo simulación
terminar simulación;
destruir objetos en la escena;
destruir escena;
guardar resultados;
mostrar informes;
Si no
Continuar;

Si tiempo transcurrido > tiempo entrada
Crear nuevo vehículo;
Si no
Continuar;

Repetir para cada vehículo creado
Calcular velocidad;
Aplicar velocidad;
Guiar vehículo;
Fin Repetir para cada vehículo

Fin Repetir hasta finalizar simulación

/FIN</td><td>Flujo de datos de salida</td></tr>
<tr><td>Parámetros por tipo validados</td><td>Solicitud generar informes</td></tr>
<tr><td>Escena</td><td>Solicitud simular físicamente un cuerpo</td></tr>
<tr><td></td><td>Solicitud crear mundo y escena</td></tr>
<tr><td></td><td>Intervalo simulación</td></tr>
<tr><td></td><td>Tiempo reacción conductor</td></tr>
<tr><td></td><td></td></tr>
<tr><td rowspan="4"></td><td rowspan="4"></td></tr>
<tr><td>Almacenes de Datos referenciados</td></tr>
<tr><td>Resultado simulación</td></tr>
<tr><td></td></tr>
<tr><td></td><td></td><td></td></tr>
</table>

N° Proceso: 3	**Nombre Proceso:** Manejar Semáforos	**Proceso Padre:** Controlar Simulación
Flujo de datos de entrada	/INICIO	**Flujo de datos de salida**
Solicitud manejar semáforos	Asignar tiempos a semáforo Repetir para cada semáforo Asignar tiempo luz verde; Asignar tiempo luz roja; Asignar tiempo luz amarilla; Fin repetir Fin Asignar tiempos	Estado semáforo
Tiempos simulación		
Tiempos de semáforos		
Solicitud estado semáforo		
	Sincronizar secuencia semáforos Repetir por cada semáforo Poner semáforo en la pila Asignar primer semáforo en pila activo Fin repetir Fin Sincronizar Controlar semáforo Determinar semáforo activo; Determinar expiración tiempo en luz; Si tiempo expiro Determinar luz activa de semáforo; Si luz verde Cambiar a luz amarilla; Fin si luz verde Si luz amarilla Cambiar a luz roja; Cambiar de semáforo activo Sacar semáforo activo de la pila; Poner semáforo al final de la pila Activar siguiente semáforo en pila Fin cambio semáforo Fin si luz amarilla Si tiempo no expiro Continuar; Fin Controlar semáforo /FIN	
	Almacenes de Datos referenciados	
	No Aplicable	

N° Proceso: 4	**Nombre Proceso:** Calcular Velocidad	**Proceso Padre:** Controlar Simulación
Flujo de datos de entrada	/ INICIO Determinar tiempo intervalo simulación expiró Si tiempo expiró (“nuevo intervalo de simulación”) Determinar si tiene vehículo líder; Si tiene Calcular distancia entre vehículos; Conseguir velocidad líder; Si no tiene Continuar; Determinar modelo de aceleración Si no cruzó línea de detención Si semáforo esta en rojo Si tiene vehículo líder Si distancia a vehículo líder < Distancia a línea de detención “Aceleración vehículo siguiente”; Si no “Aceleración influencia señal”; Si no tiene líder Si distancia de influencia > Distancia a Línea de intersección “Aceleración influencia señal”; Si no “Aceleración vehículo siguiente”; Si no esta en rojo Si semáforo esta en verde “Aceleración influencia señal”; Si no verde ni rojo (“amarillo”) Si el tiempo que queda amarillo alcanza para cruzar la intersección “Aceleración vehículo siguiente”; Si no alcanza cruzar Si distancia de influencia > Distancia a Línea de intersección “Aceleración influencia señal”; Si no “Aceleración vehículo siguiente”; Si cruzo la línea de detención “Aceleración vehículo siguiente”; Fin determinar modelo aceleración Calcular la velocidad según modelo de aceleración Determinar existencia de eventos; Si hay evento Calcular velocidad usando tiempo reacción del conductor; TmpIntSim = TmpReac; Si no hay evento Calcular velocidad utilizando tiempo definido de intervalote simulación; TmpIntSim = 0.3 segundos Si tiempo no expiro (“simulando intervalo actual”) Continuar; /FIN	**Flujo de datos de salida**
Tiempo reacción conductor		Solicitud estado semáforo
Intervalo simulación		Solicitud posición cuerpo
Estado semáforo		Solicitud velocidad instantánea cuerpo
Velocidad instantánea		Velocidad calculada
	Almacenes de Datos referenciados	
	No Aplicable	

<table>
<tr><td>N° Proceso: 5</td><td>Nombre Proceso: Aplicar velocidad</td><td>Proceso Padre:
Controlar Simulación</td></tr>
<tr><td>Flujo de datos de entrada</td><td rowspan="7">/INICIO

Conseguir el vehículo;
Conseguir las ruedas del vehículo;
Para cada una de las 4 ruedas
   Aplicar velocidad calculada;

/FIN</td><td>Flujo de datos de salida</td></tr>
<tr><td>Velocidad calculada</td><td>Solicitud aplicar velocidad</td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td rowspan="2"></td><td>Almacenes de Datos referenciados</td><td rowspan="2"></td></tr>
<tr><td>No Aplicable</td></tr>
</table>

7.3 Modelamiento De Datos

La aplicación utiliza dos tipos de archivos, en texto plano, diferentes para cargar información sobre las intersecciones que se utilizan como escenario de las simulaciones. Un archivo corresponde a un resumen de todas las intersecciones que están disponibles para ser utilizadas por el simulador, el otro tipo de archivo contiene toda la información detallada sobre una intersección, esto se hace para evitar saturar al usuario solicitándole demasiada información como puede ser la ruta que debe seguir un vehículo dentro de una calle, pensando que esa ruta es una colección de puntos tridimensionales y cada calle puede tener varias rutas (pistas).

El archivo de resumen de las intersecciones se llama “interseccion.rsm”, y debe tener la siguiente estructura para que la aplicación lo lea correctamente y pueda ejecutarse.

```
#indica comentario, esta línea no será considerada

# nombre  de la intersección
<interseccion>
Nombre : String_nombre_intersección
#archivo en donde se encuentra detallada la intersección
Archivo : nombre_archivo.int
#numero de calles que tiene la intersección, valor entero positivo
Numero calles : 99
#resumen de las calles, según el numero de las calles anterior
<calle>
Nombre : String_nombre_calle
Numero de Pistas : 99
</calle>
#.
#.
#.
<calle>
Nombre : String_nombre_calle
Numero de Pistas : 99
</calle>

</interseccion>
```

Este archivo es leído al iniciar la IGU, en donde se le da la opción al usuario de elegir las intersecciones disponibles. Dependiendo de su elección el resto de las ventanas se crean dinámicamente según el número de calles y pistas que tenga la intersección, ya que es necesario conocer los tiempos de semáforo para cada calle y los tiempos de ingreso de vehículos al sistema para cada pista.

Una vez que el usuario seleccionó la intersección, esta debe ser cargada con todas sus características. Para esto el simulador lee un archivo de texto plano con el siguiente formato.

```
#este es un comentario
#archivo que contiene la maya 3d ".mesh"
Malla : String_nombreMalla.mesh
#nombre de la malla del semáforo
Malla semaforo: String_nombreMalla_Semaforo.mesh
#información por calles
<calle>
Nombre : String_nombre_calle
#posición del semáforo en la calle, un punto 3d
Posicion semaforo :  x = 99.99; y = 99.99; z = 99.99;
Rotacion semaforo:  x = 99.99; y = 99.99; z = 99.99;
Numero pistas : 9
    <pista>
     Nombre : String_pista1
     #punto donde serán creados los vehículos
     Punto entrada : x = 99.99; y = 99.99; z = 99.99;
     Linea detencion: x = 99.99; y = 99.99; z = 99.99;
       <ruta>
          x = 99.99; y = 99.99; z = 99.99;
          x = 99.99; y = 99.99; z = 99.99;
          x = 99.99; y = 99.99; z = 99.99;
           .
           .
          x = 99.99; y = 99.99; z = 99.99;
       </ruta>
     </pista>
     .
     .
    <pista>
     Nombre : String_pistaN
     #punto donde serán creados los vehículos
     Punto entrada : x = 99.99; y = 99.99; z = 99.99;
       Linea detencion: x = 99.99; y = 99.99; z = 99.99
       <ruta>
          x = 99.99; y = 99.99; z = 99.99;
          x = 99.99; y = 99.99; z = 99.99;
          x = 99.99; y = 99.99; z = 99.99;
           .
           .
          x = 99.99; y = 99.99; z = 99.99;
       </ruta>
     </pista>
</calle>
```

7.4 Diseño de Entradas

Documento : [F01] **Nombre : [Formulario Información General]** **Descripción : Información general sobre la simulación que se realizará**			
Campo	Tipo	Longitud	Formato
Nombre Simulación	alfanumérico	30	X----------X
Tiempo Simulación	numérico	2	99
Nombre Intersección	alfanumérico	30	X----------X

Documento : [F02] **Nombre : [Formulario Parámetros Globales]** **Descripción : Parámetros globales que afectan a todos los vehículos por igual dentro de la simulación.**			
Campo	Tipo	Longitud	Formato
Ciclo Semáforo	numérico	5	999.99
Tiempo Luz Amarilla	numérico	5	999.99
Nombre Calle	alfanumérico	30	X----------X
Tiempo luz Verde	numérico	5	999.99
Tiempo luz Roja	numérico	5	999.99
Velocidad Máxima	numérico	numérico	999.99

Documento : [F03] **Nombre : [Formulario Parámetros por Pista]** **Descripción : Frecuencia con la que entrarán los vehículos al sistema, por pista**			
Campo	Tipo	Longitud	Formato
Nombre Calle	alfanumérico	30	X----------X
Nombre Pista	alfanumérico	30	X----------X
Tiempo entrada Constante	numérico	4	99.99
Tiempo Entrada Aleatorio	numérico	4	99.99 – 99.99
Tiempo Entrada Según Secuencia	numérico	4	99.99;99.99;...;99.99

7.5 Diseño de informes

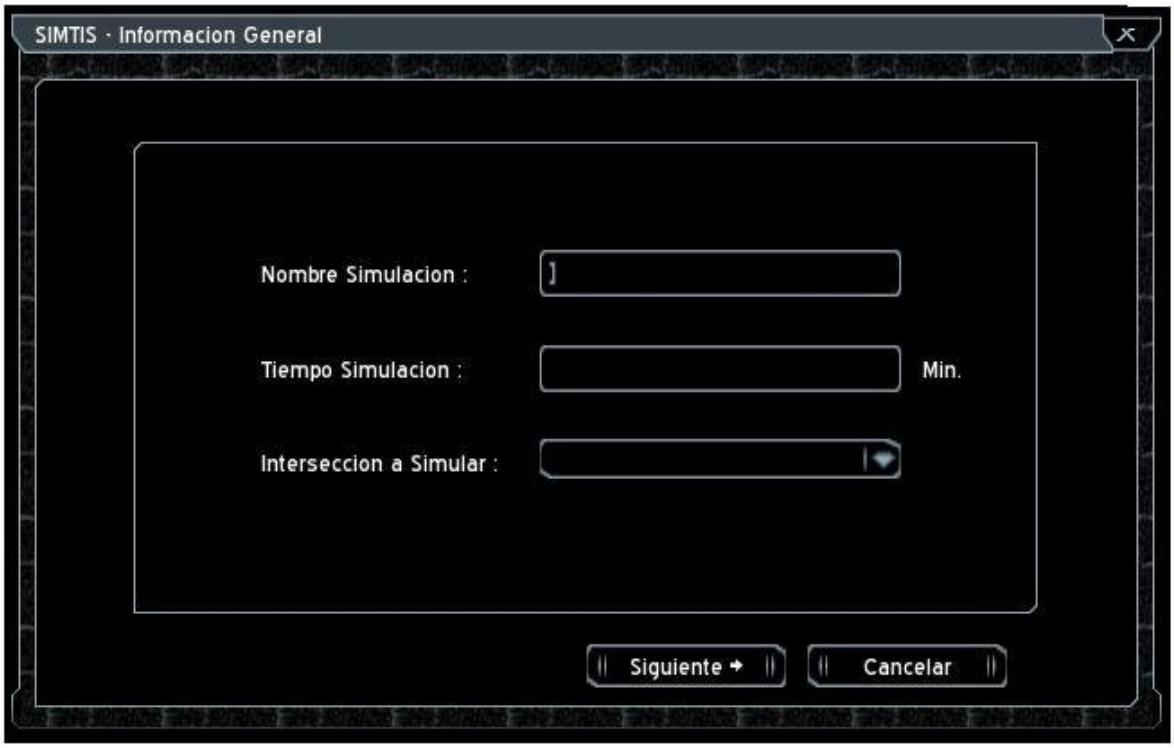
Documento : [S01]
Nombre : [Informe Simulación]
Salida : pantalla/archivo
Descripción : Informe que contiene los resultados de la simulación realizada

Campo	Tipo	Longitud	Formato
Intersección	alfanumérico	30	X----------X
Ciclo Semáforo	numérico	5	999.99
Tiempo Luz Amarilla	numérico	5	999.99
Nombre Calle	alfanumérico	30	X----------X
Nombre Pista	alfanumérico	30	X----------X
Tiempo luz Verde	numérico	5	999.99
Tiempo luz Roja	numérico	5	999.99
Tiempo de Espera	numérico	5	999.99
Numero de vehículos	numérico	6	9999.99

7.6 Pantallas

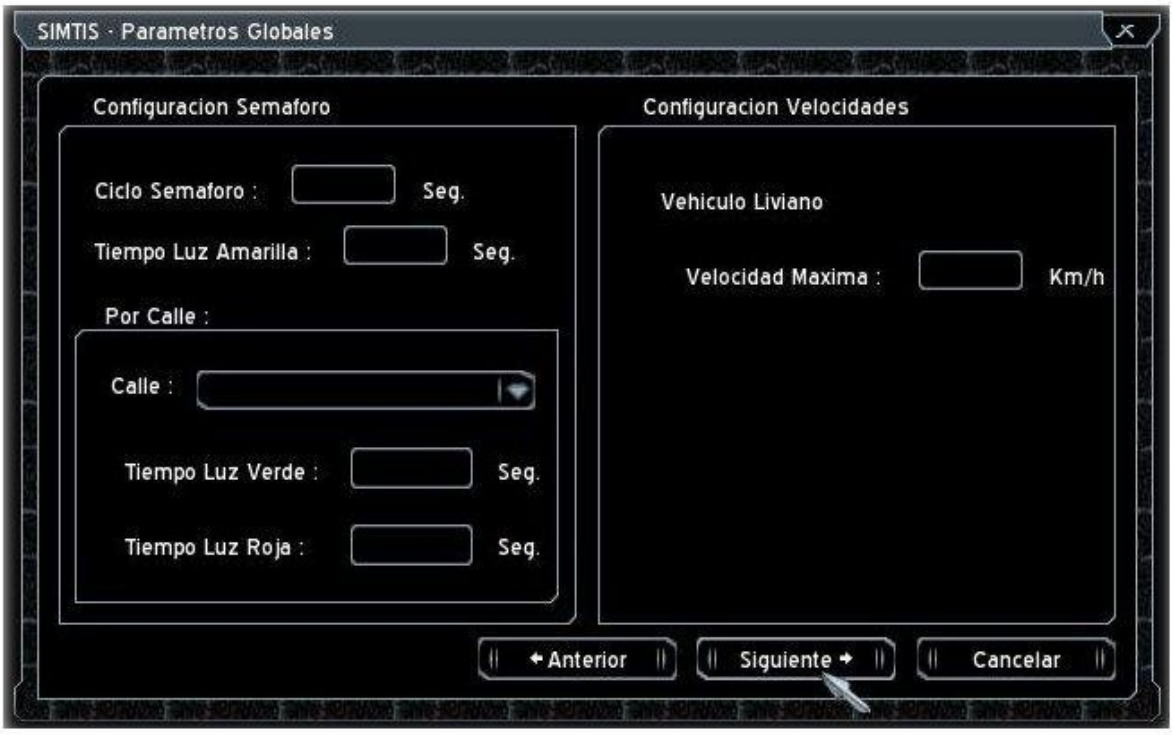
Las pantallas son presentadas en orden a como van apareciendo ante el usuario, con una breve descripción de cada una de ellas.

Pantalla información general.



Corresponde a la pantalla inicial y captura la información sobre las características generales de la simulación. Tales como un nombre para identificar la simulación, al finalizar esta se creará un archivo de resultados con el nombre que ingrese el usuario en este campo. Un tiempo de duración de la simulación, y la elección de la intersección que se simulará, esto dentro de las selecciones disponibles.

Ventana Parámetros Globales

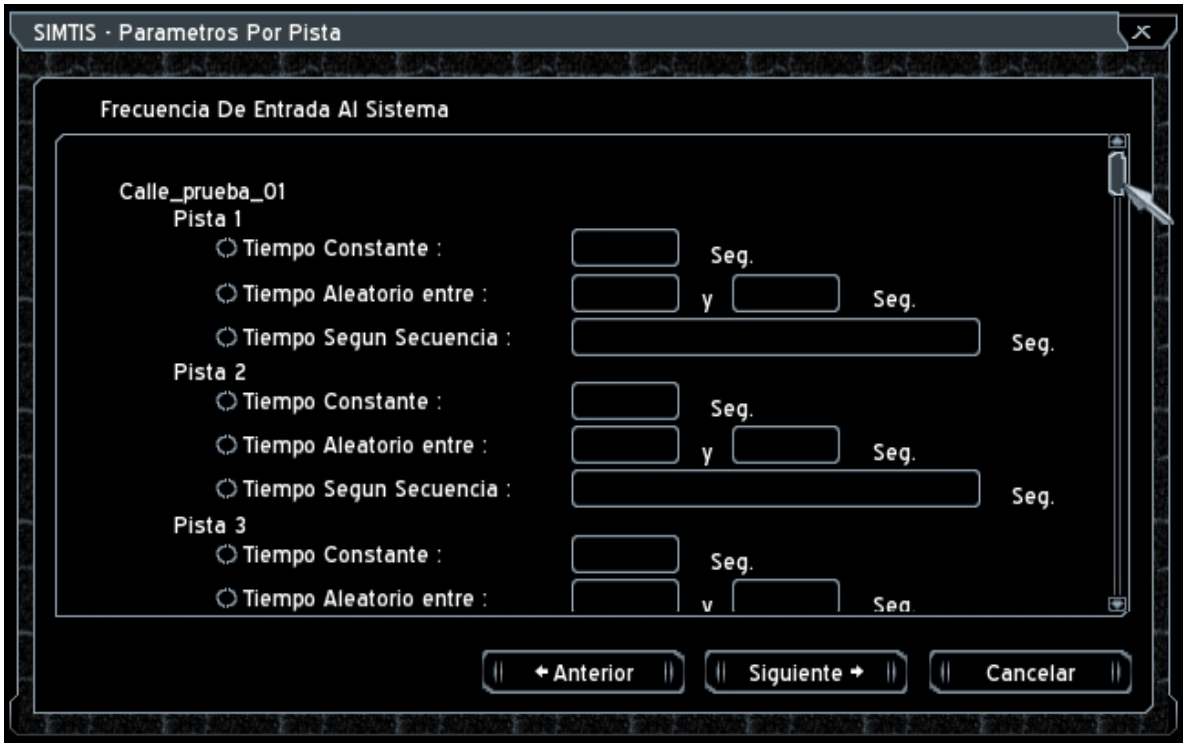


Es la segunda pantalla en el orden de aparición ante el usuario y es la encargada de capturar la información global para la simulación, es decir, la información que afectará a todos los vehículos dentro de la intersección. Esta pantalla solicita al usuario que ingrese un *"Ciclo de Semáforo"* para la intersección junto con un tiempo designado para la luz amarilla. Siguiendo con la configuración de semáforo ahora se divide el tiempo asignado a las luces verde y roja por calles, de esta manera el usuario debe seleccionar en el combobox la calle y seguidamente asignarle un tiempo de luz verde y luz roja. En la sección de "Configuración de Velocidades", se debe ingresar la velocidad máxima permitida para el desplazamiento de los vehículos en este caso vehículos livianos.

En el caso que el usuario ingrese una configuración de semáforo errónea, se

desplegará la siguiente pantalla de advertencia señalando el acontecimiento.

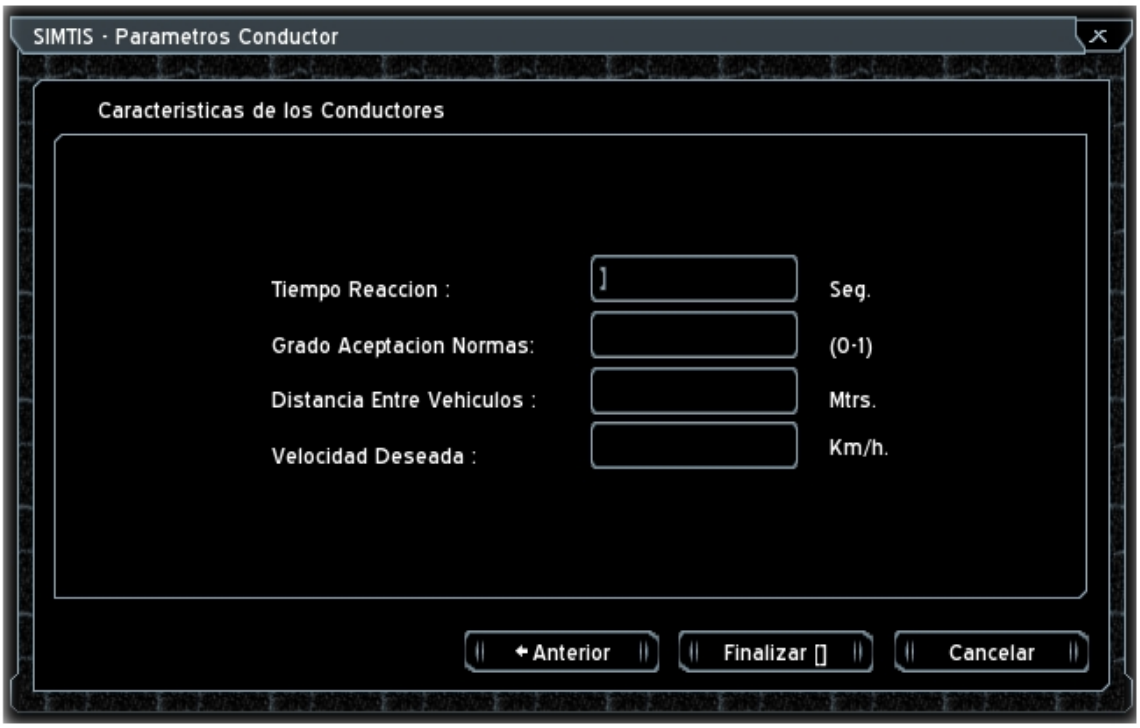
Ventana Parámetros por Pista.



La ventana “Parámetros Por Pista” recoge la información de cada cuanto tiempo se creará un nuevo vehículo que ingresará a la intersección. Esta información se divide por calles de la intersección y por pistas de la calle, de esta manera se ingresa el tiempo en

segundos que debe pasar hasta para crear un nuevo vehículo en la pista. El usuario dispone de tres modos de tiempo de ingreso, uno constante otro *aleatorio* y uno según alguna *secuencia*. El tiempo constante como su nombre lo indica es inalterable dentro de la simulación, el tiempo aleatorio se mueve dentro de un rango de dos valores y la secuencia es un conjunto de tiempos separados por guiones ej. 10-12-14-8-6. La secuencia se implementa como una cola circular sin fin, por lo tanto cuando se llega al fin de la secuencia se comienza nuevamente con el primer valor, así hasta terminar la simulación.

Ventana Parámetros Conductor

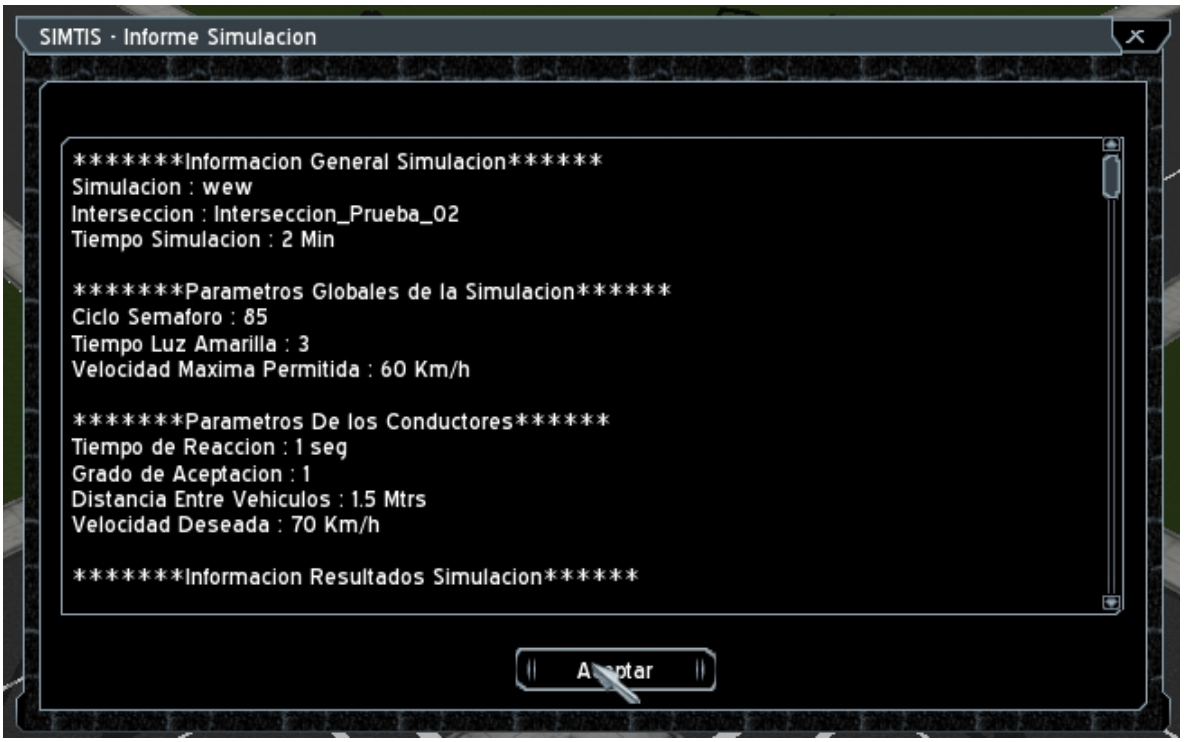


"Parámetros Conductor" es la ultima ventana de configuración para la simulación, indica al usuario que debe ingresar el "Tiempo de Reacción" que le toma al conductor enfrentar un evento, este tiempo en realidad es el tiempo percepción-reacción descrito en los primeros capítulos. El "Grado Aceptación normas" es un valor entre 0 y 1 que dice cuan

propenso o no es el conductor a adoptar las normas impuestas (velocidad máxima), siendo 0 ninguna aceptación y 1 una total aceptación. La “Distancia entre Vehículos” es la distancia que un vehículo nunca sobrepasará como espacio entre el y su vehículo líder (ver modelo de Gipps).

“Velocidad deseada” es la velocidad con la cual el conductor desearía viajar en condiciones ideales sin ninguna limitante por delante de él. Esta velocidad es contrastada con la velocidad máxima permitida y con el grado de aceptación para obtener una velocidad *“deseada con condicionamiento”* por decirlo de alguna manera y utilizarla en los modelos de simulación como la velocidad deseada de viaje.

Ventana Informe Simulación

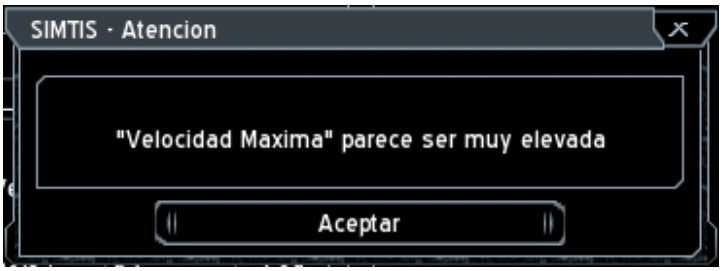


Al expirar el tiempo asignado a la simulación se despliega la ventana “Informe

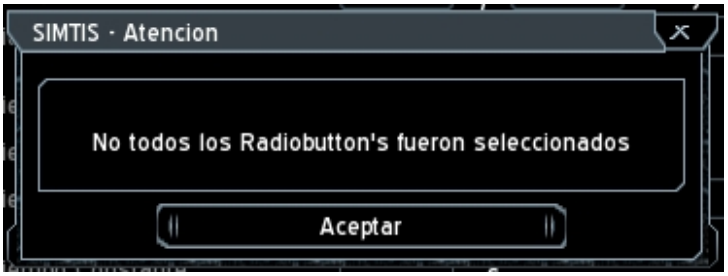
Simulación" que corresponde a la última ventana de la interfaz gráfica de usuario. Contiene los datos capturados durante la ejecución de la simulación, información como velocidades promedio, tiempos de demora y flujos vehiculares. Al aceptar la ventana se finaliza la aplicación.

Ventanas de Advertencia

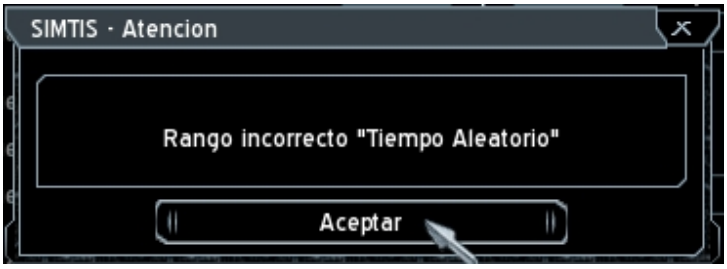
En ocasiones cuando el usuario ingresa un valor erróneo o dejo alguna información sin completar se despliega una ventana de advertencia similar a la que se muestra en la siguiente captura.



Cuando una velocidad máxima parece ser muy elevada al sentido común para una intersección urbana.



Cuando falta asignar tiempos de entrada a alguna pista.



Cuando se asigna un rango incorrecto a un tiempo de entrada aleatorio.

El acento de las palabras en la interfaz gráfica fue omitido deliberadamente ya que CEGUI, encargado de gestar y administrar las ventanas, integrado a Ogre no acepta acentos.

7.7 Casos de Prueba

Para observar el comportamiento del simulador y analizar los resultados que entrega se exponen 3 casos de prueba realizados con diferentes configuraciones cada uno. Los resultados fueron los siguientes.

Caso 1

Tabla N° 7.1: Parámetros entrada Caso Prueba 1

Nombre Simulación		: Simulación_Prueba_01			
Tiempo Simulación		: 05 minutos			
Intersección a Simulador		: Interseccion_Prueba_2			
Ciclo de Semáforo		: 95 segundos			
Tiempo Asignado a Luz Amarilla		: 03 segundos			
Calle	**Tiempo Luz Verde (seg)**	**Tiempo Luz Roja (seg)**	**Tipo entrada**	**Tiempo Entrada (seg)**	
Calle_Prueba01	58	34	Constante	Pista 1	12
			Constante	Pista 2	8
			Constante	Pista 3	6
Calle_Prueba02	31	61	Constante	Pista 1	3
			Constante	Pista 2	4
			Constante	Pista 3	2

Tabla N° 7.2: Resultados simulación Caso Prueba 1

	Calle_Prueba01		Calle_Prueba02	
Flujo Vehicular (Veh/h)	Pista 1	276	Pista 1	792
	Pista 2	396	Pista 2	744
	Pista 3	540	Pista 3	636
	Total	**1212**	**Total**	**2172**
Velocidad Promedio/Tiempo (Km/h)	Pista 1	36.56	Pista 1	17.27
	Pista 2	36.75	Pista 2	18.45
	Pista 3	36.72	Pista 3	20.71
	Total	**36.68**	**Total**	**18.81**
Velocidad Promedio/Espacio (Km/h)	Pista 1	27.48	Pista 1	12.46
	Pista 2	27.77	Pista 2	13.49
	Pista 3	28.07	Pista 3	14.68
	Total	**27.77**	**Total**	**13.54**
Tiempo Demora Promedio (seg)	Pista 1	22.45	Pista 1	49.52
	Pista 2	22.19	Pista 2	45.71
	Pista 3	21.98	Pista 3	41.99
	Total	**22.21**	**Total**	**45.74**

Como era de esperar la calle “Calle_Prueba02” es la que registra mayor flujo de vehículos en la simulación esto debido a que los tiempos de entrada asignado a sus pistas son menores que los asignados a la calle “Calle_Prueba01”. En este caso de prueba la configuración de semáforos realizada para la intersección no es adecuada ya que se está dando preferencia en tiempo a la calle que aporta el menor flujo de vehículos, considerablemente menor casi la mitad del flujo que aporta la calle “Calle_Prueba02”. Esta configuración de semáforo hace que la calle con mayor flujo aumente sus tiempos de demora y disminuya sus velocidades de viaje principalmente por que no todos los vehículos que esperan por cruzar la intersección pueden hacerlo.

Caso 2

Tabla N° 7.3: Parámetros entrada Caso Prueba 2

Nombre Simulación			: Simulación_Prueba_02		
Tiempo Simulación			: 05 minutos		
Intersección a Simulador			: Interseccion_Prueba_2		
Ciclo de Semáforo			: 95 segundos		
Tiempo Asignado a Luz Amarilla			: 03 segundos		
Calle	**Tiempo Luz Verde (seg)**	**Tiempo Luz Roja (seg)**	**Tipo entrada**	**Tiempo Entrada (seg)**	
Calle_Prueba01	31	61	Constante	Pista 1	12
			Constante	Pista 2	8
			Constante	Pista 3	6
Calle_Prueba02	58	34	Constante	Pista 1	3
			Constante	Pista 2	4
			Constante	Pista 3	2

Tabla N° 7.4: Resultados simulación Caso Prueba 2

	Calle_Prueba01		Calle_Prueba02	
Flujo Vehicular (Veh/h)	Pista 1	276	Pista 1	996
	Pista 2	372	Pista 2	792
	Pista 3	468	Pista 3	996
	Total	**1116**	**Total**	**2784**
Velocidad Promedio/Tiempo (Km/h)	Pista 1	24.91	Pista 1	32.52
	Pista 2	24.60	Pista 2	34.38
	Pista 3	22.15	Pista 3	32.42
	Total	**23.89**	**Total**	**33.10**
Velocidad Promedio/Espacio (Km/h)	Pista 1	16.17	Pista 1	25.02
	Pista 2	15.90	Pista 2	26.60
	Pista 3	14.48	Pista 3	24.84
	Total	**15.52**	**Total**	**25.49**
Tiempo Demora Promedio (seg)	Pista 1	38.24	Pista 1	24.66
	Pista 2	38.85	Pista 2	23.19
	Pista 3	42.58	Pista 3	24.81
	Total	**39.89**	**Total**	**24.22**

Para este caso de prueba se conservan los mismos parámetros de entrada que en el caso anterior la única excepción es que los tiempos de semáforo se intercambiaron entre las calles, es decir, los tiempos de la calle "Calle_prueba01" son ahora los de la calle "Calle_Prueba02" y viceversa. Por lo tanto la calle "Calle_Prueba02" sigue aportando el mayor flujo de vehículos a la intersección sin embargo sus tiempos de espera disminuyeron considerablemente casi a la mitad. Esto se debe principalmente al aumento en el tiempo de luz verde para su semáforo, con esto ahora todos los vehículos detenidos que esperan cruzar la intersección lo consiguen en el intervalo asignado a la luz verde.

La calle "Calle_Prueba02" además aumentó su flujo de vehículos esto también debido a la nueva configuración de semáforo, ahora puede poner más vehículos en la intersección. La calle "Calle_Prueba01" entrega una pequeña disminución en su flujo y un aumento en sus tiempos de demora por que ya no tiene flujo tan expedito producto de la disminución de su tiempo en luz verde y aumento en luz roja. Sin embargo ahora la preferencia la tiene la calle que aporta, un considerable, mayor flujo a la intersección.

Caso 3

Tabla N° 7.5: Parámetros entrada Caso Prueba 3

Nombre Simulación		: Simulación_Prueba_03			
Tiempo Simulación		: 05 minutos			
Intersección a Simulador		: Interseccion_Prueba_2			
Ciclo de Semáforo		: 85 segundos			
Tiempo Asignado a Luz Amarilla		: 03 segundos			
Calle	**Tiempo Luz Verde (seg)**	**Tiempo Luz Roja (seg)**	**Tipo entrada**	**Tiempo Entrada (seg)**	
Calle_Prueba01	37	45	Constante	Pista 1	12
			Constante	Pista 2	8
			Constante	Pista 3	6
Calle_Prueba02	42	40	Constante	Pista 1	3
			Constante	Pista 2	4
			Constante	Pista 3	2

Tabla N° 7.6: Resultados simulación Caso Prueba 3

	Calle_Prueba01		Calle_Prueba02	
Flujo Vehicular (Veh/h)	Pista 1	276	Pista 1	852
	Pista 2	408	Pista 2	708
	Pista 3	552	Pista 3	864
	Total	**1236**	**Total**	**2424**
Velocidad Promedio/Tiempo (Km/h)	Pista 1	34.61	Pista 1	26.67
	Pista 2	31.34	Pista 2	30.46
	Pista 3	30.64	Pista 3	25.99
	Total	**32.19**	**Total**	**27.71**
Velocidad Promedio/Espacio (Km/h)	Pista 1	24.71	Pista 1	20.14
	Pista 2	21.95	Pista 2	22.86
	Pista 3	21.41	Pista 3	19.93
	Total	**22.69**	**Total**	**20.98**
Tiempo Demora Promedio (seg)	Pista 1	24.96	Pista 1	30.63
	Pista 2	28.11	Pista 2	26.97
	Pista 3	28.78	Pista 3	30.92
	Total	**27.28**	**Total**	**29.51**

Para el caso de prueba 3 se cambió la configuración de semáforos para la intersección, asignando un ciclo de 85 segundos con un tiempo de 3 segundos para la luz amarilla, el resto de los parámetros se mantuvo igual que en los dos casos anteriores.

En éste caso de prueba los tiempos de demora de las dos calles se aproximan bastante el uno del otro habiendo una diferencia entre ellos de sólo dos segundos y fracción. El mayor flujo de vehículos lo sigue aportando la calle "Calle_Prueba02" con una disminución pequeña en su flujo. La calle "Calle_Prueba01" en tanto experimenta un leve aumento en su flujo. Esta nueva configuración de semáforo consigue la menor demora para la intersección, sumadas las dos calles, de los tres casos de prueba expuestos lo que quiere decir que al disminuir el ciclo de semáforo en 10 segundos y tratando de equiparar los tiempos en los dos semáforos se logran demoras aceptables para las dos calles. Este no

significa que la nueva configuración de tiempos de semáforo sea el optimo para la intersección de prueba, con los parámetros de de entrada asignados. Sólo muestra los impactos que producen las diferentes combinaciones de tiempos en los semáforos de la intersección.

Tabla N° 7.6: Parámetros de entrada para conductores

Tiempo de Reacción (Seg)	1.0
Grado de Aceptación [0-1]	1
Distancia entre Vehículos (m)	1.5
Velocidad Deseada (Km/h)	70

Los parámetros de entrada para los conductores se mantuvieron sin alteración en los tres casos de prueba descritos.

7.7.1 Capturas de Simulaciones

Conclusión

Las conclusiones al término del proyecto de titulación se desprenden de varios aspectos diferentes del mismo, los relacionados al software libre, la necesidad y potencialidad de la integración de varias herramientas en el desarrollo y los resultados sobre el problema principal que da origen al proyecto.

La existencia del software libre en el desarrollo de aplicaciones es cada día más significativa. Las potencialidades entregadas por el software libre, en sus mayores exponentes, prácticamente no tienen diferencias con las potencialidades entregadas por las herramientas de pago de similares características. Una gran limitación a la hora de emprender un proyecto sobre todo a nivel de investigación con fines académicos es la utilización y disponibilidad de licencias que permitan desarrollar. El acierto al seleccionar herramientas “open source” para integrarlas en el desarrollo del proyecto superó estas limitaciones y además entregó un valor agregado impensado, que esta relacionado con el aporte de las comunidades que se crean entorno al software libre, principalmente el soporte que entregan los foros que son el repositorio de conocimientos más grande que una herramienta pueda entregar. Para poder hoy día exponer el proyecto realizado fue necesario invertir muchas horas dentro de estas comunidades para poder crear las competencias y habilidades necesarias para el logro de los objetivos. Los foros de Blender, Ogre3D, NxOgre, CEGUI y en alguna medida PhysX, (auque este último es pagado pero de libre uso para su utilización en proyectos de carácter académico no comercial), fueron los fuentes que sin duda marcaron el éxito en el cumplimiento de los objetivos.

Por otro lado, el concepto fundamental que queda en mente después de integrar las distintas herramientas es la modularización, especialización y compatibilidad de estas. En cuanto a la modularización se puede ver a cada herramienta como un componente separado pero fundamental en el funcionamiento del simulador, la especialización de las

herramientas permite obtener las mejores prestaciones que otorga cada una de ellas. Por una parte está el motor gráfico como un subsistema del cual se pueden aprovechar sus características para trabajar de mejor manera la parte visual del simulador. Por otro lado está el motor físico que se puede ver de igual manera que el anterior pero enfocado a conseguir las simulaciones físicas de los objetos, los vehículos. La compatibilidad en las herramientas utilizadas es importante, ya que es gravitante que se puedan comunicar entre ellas, que el comportamiento físico de los vehículos pueda observarse gráficamente de una forma amigable. No es sorprendente encontrar que la gran mayoría de las herramientas open source son compatibles entre ellas (si sus áreas están relacionadas), ya que su carácter de libertad permite a cualquiera que lo necesite desarrollar aplicaciones o envoltorios que se encarguen de comunicar los sistemas, el envoltorio que se encarga de comunicar los dos motores utilizados en el simulador es NxOgre, su presencia en el proyecto es muy significativa.

Que cada cosa se dedique a lo suyo y lo haga de la mejor manera, y que exista la posibilidad de integrar sus funcionalidades a terceros proyectos resulta sin duda muy beneficioso.

Al inicio del proyecto solo se tenía la idea de que era lo que se deseaba desarrollar, pero ningún conocimiento acerca de cómo realizarlo. Afortunadamente la Red provee de todo lo necesario para lograr el desarrollo, solo hay que invertir tiempo buscando orientación, alternativas y especializándose en el uso de éstas. Quizás el tamaño del proyecto resulta ser un poco grande para un solo integrante, integrar a más personas y especializarlas en algún ámbito determinado puede resultar más beneficioso.

Finalmente, terminada la aplicación, es posible realizar simulaciones de tráfico sobre intersecciones señalizadas, obteniendo como resultado los tiempos de espera que lleva cruzar por la intersección así como el número de vehículos que usaron el sistema. Es posible configurar las simulaciones con diferentes parámetros y ver como afectan estos. Lo que permite saber con cierto grado de aproximación a la realidad como podría comportarse

una intersección verdadera a cambios en los tiempos de semáforo, tiempos de reacción de conductores, limites de velocidades, permitiendo a las personas encargadas de tomar decisiones sobre el control de tráfico urbano en intersecciones señalizadas mejorar sus procesos y no aventurar con los resultados de sus decisiones. Esto último es característica fundamental de toda simulación o simulador, abaratar costos o evitar implantar decisiones equivocadas con resultados altamente costosos.

Bibliografía

1. Nicholas J. Garber, Lester A. Hoel. 2001.Traffic and Highway Engineering.
2. Principles of Highway Engineering and Traffic Analysis. *Fred L. Mannering*
3. Roger P. Roess, William R. McShane, Elena S. Prassas. 1997 2da Ed. Traffic Engineering.
4. Wolfgang S. Homburger, Jerome W. Hall, Roy C. Loutzenheiser, William R. Reilly. 2001 15 Ed. Fundamentals of Traffic Engineering. *Wolfgang S. Homburger*
5. *JavierAracil.* 1997. Dinámica de Sistemas.
6. Física Volumen I: MECANICA. *Marcelo Alonso Edward*
7. Matematicas. *Luis Postigo*

Referencias

[Ref01]: Wolfgang S. Homburger, “Fundamentals of Traffic Engineering”, 15 edición 2001.

[Ref02]: Rodrigo González Rivera, “Calibración del Microsimulador AIMSUN para flujo ininterrumpido en la ciudad de Concepción”, octubre del 2005.

[Ref03]: Juan Carlos Dextre, “Modelo de Simulación de Tráfico Vehicular“, Congreso Nacional de Ingeniería Civil – Trujillo 1998.

[Ref04]: Juan Carlos Dextre, “Modelo de Simulación de Tráfico Vehicular“, Congreso Nacional de Ingeniería Civil – Trujillo 1998.

[Ref05]: Jesús Racero Moreno, “Modelo de Cambio de Carril para un Simulador Microscópico de Tráfico Urbano”

[Ref06]: Ogre3D manual, http://www.ogre3d.org/docs/manual/manual_4.html#SEC4

[Ref07]: Wikipedia, http://en.wikipedia.org/wiki/Physics_engine.

[Ref08]: Ogre3D foro, http://www.ogre3d.org/phpBB2/.

[Ref09]: Wikipedia, http://es.wikipedia.org/wiki/PhysX.

[Ref10]: Robin Southern, http://nxogre.org.

[Ref11]: WikiBlender, http://wiki.blender.org/index.php/Manual.es/PartIV/UV_Mapping.

[Ref12]: Francisco Navarro, http://www.escuelaconductores.cl/?showthread=3&idnot=5.

[Ref13]: Manual de Demarcaciones Vialidad,
http://www.vialidad.gov.cl/areasde_vialidad/seguridad_vial/normas/manuales.htm#dem.