PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

Integración de un Simulador Tridimensional a un Sistema Multi-Agente Aplicado a Berth Allocation Problem

Carlos Javier Mujica Ayala

INFORME FINAL PROYECTO PARA
OPTAR AL TÍTULO PROFESIONAL
INGENIERO EJECUCIÓN EN INFORMATICA

JUNIO 2014

Integración de un Simulador Tridimensional a un Sistema Multi-agente Aplicado a Berth Allocation Problem

CARLOS JAVIER MUJICA AYALA

Profesor Guía: **Claudio Cubillos Figueroa**

Profesor Co-referente: **Aldo Migliaro Osorio**

Carrera: **Ingeniería de Ejecución en Informática**

JUNIO 2014

Agradezco a mis padres, a mi hermano y a mi pareja por el apoyo otorgado durante este camino para llegar a ser un profesional. Todo mi esfuerzo va dedicado a ellos y especialmente a mi hijo.

Índice

Resumen

El presente trabajo consiste en el desarrollo de un simulador 3D a un programa que soluciona el problema de asignación de barcos a muelles denominado Berth Allocation Problem. El programa mencionado utiliza la tecnología multiagente para el desarrollo de la solución. Berth Allocation Problem consiste en el problema de asignación de los barcos en forma óptima con los diferentes puntos de atraques, de manera que el tiempo que se encuentre el barco en el muelle sea mínimo, y así utilizar el tiempo y los recursos establecidos de forma correcta.

Para cumplir el objetivo es utilizada la tecnología multiagente en un entorno Java, utilizando la plataforma JADE para la integración del programa con el simulador 3D y la herramienta JMonkey para el desarrollo de este.

El fin del proyecto consiste en que el usuario pueda comprender y visualizar de mejor manera las asignaciones de las embarcaciones a los distintos puntos de atraque, así permitirá al usuario analizar y monitorear de mejor manera el trabajo realizado por el software.

Palabras-claves: Berth Allocation Problem, simulador tridimensional, Tecnología multiagente, Java, JADE, JMonkey.

Abstract

The present work is the development of a 3D simulator program that solves the Berth Allocation Problem. The above program uses multi-agent technology for the development of the solution. Berth Allocation Problem consists in the assignment problem optimally boats with different mooring points, so long as the ship is in the dock is minimal , and so use the time and resources set correctly .

To meet the target multi-agent technology is used in a Java environment, using the JADE platform for integration with 3D simulator program and jMonkey tool to develop this.

The purpose of the project is that the user can understand and visualize better assignments to different craft moorings and allow the user to analyze and better monitor the work done by the software.

Key words: Berth Allocation Problem, three-dimensional simulation, multi-agent technology, Java, JADE , jMonkey .

Índice de Figuras

Capítulo 1: Descripción Del Tema

1.1 Introducción

Las embarcaciones siempre han sido muy importantes en el desarrollo de la economía de la mayoría de los países, ya que desde muchos años se han utilizado para la exportación e importación de productos. Hoy en día la creciente demanda de transporte de productos hace que el número de barcos aumente y sea más difícil organizar los tiempos y recursos necesarios para cada barco, por lo que se necesita un sistema de organización para optimizar de mayor manera cada recurso que se posee.

Beth Allocation Problem consiste en el problema de asignación de barcos a muelles para utilizar de forma óptima los recursos y el tiempo que se dispone. Ya existen programas que solucionan este problema con ayuda de sistemas multiagente. En este proyecto se seleccionará un software en particular con el objetivo de desarrollar una interfaz gráfica Tridimensional. El software se encarga de manera óptima de ordenar la llegada de los barcos y como se ordenarán en el puerto, para maximizar el transporte de los containers y minimizar el tiempo de los barcos en el puerto. Mientras que el simulador 3D tiene la misión de mostrar en forma dinámica y en tiempo real como se asignan los barcos a los puntos de atraque y si ocurre algún inconveniente en alguno de ellos, demostrar como los barcos son asignados nuevamente en otro punto de atraque.

En este proyecto es utilizado un entorno JAVA, donde a través de agentes se realizará la integración del sistema existente creado en JADE y la interfaz en 3D desarrollada en JMonkey Engine.

1.2 Definición De Objetivos

1.2.1 Objetivo General

Desarrollar una interfaz tridimensional a un sistema que resuelve el problema de asignación de barcos a muelles denominado Beth Allocation Problem.

1.2.2 Objetivo Específicos

- Estudiar en qué consiste BAP y el funcionamiento del Sistema multiagente que se utiliza para resolverlo.
- Investigar cómo se utilizan las herramientas de trabajo de modelado en 3D y la utilización del motor gráfico.
- Realizar y comunicar la Interfaz de simulación tridimensional con el Sistema multiagente.
- Realizar las respectivas pruebas del sistema.

1.3 Planificación Del Proyecto

	Nombre de tarea	Duración	Comienzo	Fin
1	**Etapa 1**	**44 días**	**mar 05/03/13**	**vie 03/05/13**
2	Investigar sobre Berth Allocation Problem(BAP)	24 días	mar 05/03/13	vie 05/04/13
3	Estudiar el programa de modelado 3D	15 días	lun 25/03/13	vie 12/04/13
4	Estudiar el motor grafico seleccionado	35 días	lun 18/03/13	vie 03/05/13
5	Comprender el codigo del BAP del Proyecto Anterior	17 días	mar 09/04/13	mié 01/05/13
6	Confección de Informe de avance	18 días	lun 25/03/13	mié 17/04/13
7	Entrega Informe de avance	1 día	vie 19/04/13	vie 19/04/13
8	**Etapa 2**	**72 días**	**jue 02/05/13**	**vie 09/08/13**
9	corregir error del codigo del proyecto anterior	11 días	jue 02/05/13	jue 16/05/13
10	creación del primer proptotipo	49 días	mar 04/06/13	vie 09/08/13
11	Confeccionde modelos PASSI	9 días	lun 29/07/13	jue 08/08/13
12	Confeccion y corrección informe final	6 días	jue 01/08/13	jue 08/08/13
13	entrega Informe final	1 día	vie 09/08/13	vie 09/08/13

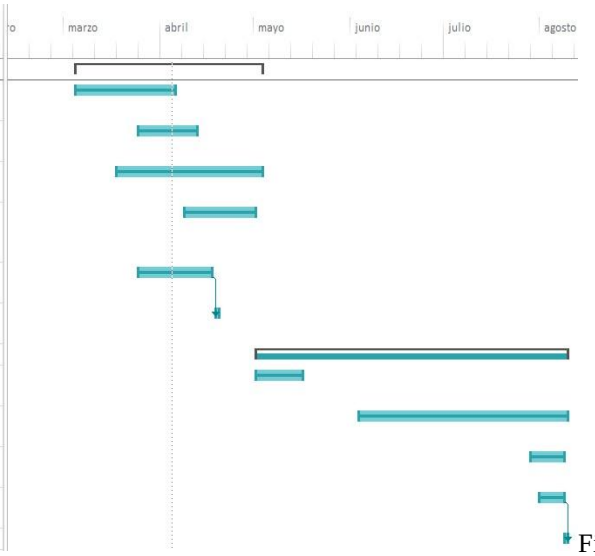


Figura 1.1
Carta Gantt, primeras etapas

14		Etapa 3	50 días	lun 02/09/13	vie 08/11/13
15		Corregir los últimos inconvenientes del código del proyecto anterior	21 días	lun 02/09/13	lun 30/09/13
16		Mejorar prototipo anterior	25 días	lun 07/10/13	vie 08/11/13
17		Sincronizar tiempos entre el sistema y la interfaz	5 días	lun 07/10/13	vie 11/10/13
18		Mejorar interfaz 3D	8 días	mar 15/10/13	jue 24/10/13
19		Añadir información a los barco y berths en la interfaz	11 días	jue 10/10/13	jue 24/10/13
20		Simular que un barco esta siendo atendido	4 días	lun 21/10/13	jue 24/10/13
21		Representar eventos de los barcos y berths	10 días	lun 28/10/13	vie 08/11/13
22		Corregir informe	7 días	mié 16/10/13	jue 24/10/13
23		Entrega informe de avance	1 día	vie 25/10/13	vie 25/10/13
24		Etapa 4	39 días	mar 05/11/13	vie 27/12/13
25		Confección del ultimo prototipo	24 días	mar 05/11/13	vie 06/12/13
26		Crear modelos finales	1 día	mar 12/11/13	mar 12/11/13
27		Crear menús	8 días	mié 13/11/13	vie 22/11/13
28		Crear entorno	6 días	lun 25/11/13	sáb 30/11/13
29		Realizar pruebas	4 días	mar 03/12/13	vie 06/12/13
30		Entrega de software	1 día	sáb 14/12/13	sáb 14/12/13
31		Corregir informe	6 días	lun 16/12/13	lun 23/12/13
32		Entrega Informe	1 día	vie 27/12/13	vie 27/12/13

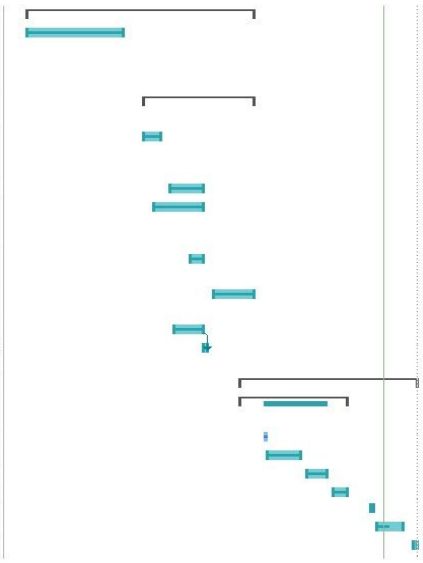


Figura 1.2 Carta Gantt, últimas etapas

1.4 Metodología De Trabajo

1.4.1 Proceso Unificado de Desarrollo de Software (UP)

El Proceso Unificado no es simplemente un proceso, sino un marco de trabajo extensible que puede ser adaptado a organizaciones o proyectos específicos. El Proceso Unificado de Rational se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental [1].

1.4.2 Proceso Unificado Ágil

Es una versión simplificada del Proceso Unificado de Rational (RUP) [2]. Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El Ágil UP aplica técnicas ágiles incluyendo Desarrollo Dirigido por Pruebas. Esta será la metodología que se utilizará en el proyecto [3].

1.4.3 Proceso Unificado Ágil Propio

El desarrollo de este proyecto lo componen 4 fases, la fase de inicio consta de la definición del problema y de que trata el proyecto, ya sea su objetivo principal como específicos. En la fase de elaboración se realiza la planificación del proyecto (carta Gantt), exponer las tecnologías de trabajo a utilizar y el marco teórico del proyecto. En la fase de construcción se comienza a realizar el proyecto, comenzando con un pequeño prototipo para luego ir realizando nuevas iteraciones y pruebas. Las iteraciones consisten en ir agregando nuevas funcionalidades como también ir mejorando la imagen 3D. A continuación se describen las iteraciones principales:

1. Se comenzará con simples modelos ubicados en el plano, para comprobar que la sincronización del sistema multiagente con el simulador 3D sea el correcto.

2. Luego se agregaran los movimientos y rutas a los barcos, cuando estos puntos están listos se continúa con mejorar los modelos tridimensionales.
3. Después agregar los menús desplegables dentro de la simulación y agregar los eventos que pueden ocurrir en la simulación, representando cuando un berth deja de funcionar, y la reasignación de un barco de un berth a otro, entre otros eventos.
4. Por ultimo agregar las texturas a los modelos y crear el espacio que rodea el mundo 3D.

Para terminar la fase final que corresponde a la fase de pruebas, donde se realizaran las pruebas correspondientes al software final.

1.5 Estructura del Documento

En este apartado se dará a conocer los diversos capítulos que contiene este informe y de que trata cada uno con una breve descripción. Cada capítulo se refiere a una temática diferente, pero siempre centrado en el trabajo que se quiere realizar. Los capítulos son:

Capítulo 1 Descripción del tema: Este capítulo explica de que trata el proyecto, su objetivo principal y objetivos específicos, la planificación del proyecto y la metodología de trabajo.

Capítulo 2 Marco de trabajo: en este capítulo se presentan las tecnologías que se utilizaran para desarrollar el proyecto. En primer lugar se refiere a la tecnología multiagente, luego al modelado 3D y por ultimo al motor gráfico 3D.

Capítulo 3 definición del problema: este capítulo se centra en el problema de Berth Allocation Problem, con sus restricciones, fórmula matemática, algoritmos que lo resuelven y como se lleva a cabo en los proyectos anteriores, que hace referencia este informe.

Capítulo 4 Desarrollo del proyecto: en este capítulo se da a conocer el desarrollo del proyecto, la presentación del caso de estudio y el desarrollo del caso de estudio. También se presentan los distintos modelos y diagramas para una mayor comprensión de lo que se quiere lograr.

Capítulo 5 Pruebas de software: en este capítulo se describirán los tipos de pruebas que se aplicarán al sistema. Estas pruebas serán esenciales para el buen funcionamiento del software.

Capítulo 6 Implementación: este capítulo hace referencia a lo que se ha realizado respecto al diseño visto en el capítulo 5. Explica cómo funciona el software desarrollado y sus características.

Capítulo 7 Futuro del Proyecto: este capítulo consiste en indicar como fue el resultado del proyecto y que cosas faltaron o se puede agregar a futuro para mejorarlo.

Conclusiones: Por último se encuentra las conclusiones, donde se realiza una comprensión general de los conceptos vistos anteriormente y se presentan las diversas conclusiones del proyecto a lograr.

Capítulo 2: Marco De Trabajo

En este capítulo se resumen las tecnologías utilizadas para el desarrollo de este proyecto. Las tecnologías utilizadas son: Agentes Inteligentes y Sistemas Multiagente, Modelado 3D, Motor Gráfico 3D.

2.1 Agentes Inteligentes y Sistemas Multiagente

A continuación se definirá en forma breve en que consiste la tecnología de agentes y sistemas multiagente utilizada para la realización del proyecto.

2.1.1 Definición de Agente

Existen variadas definiciones para el concepto de agente, algunas de estas son: "Un agente es cualquier cosa que pueda ver en su entorno a través de sensores y actuar en su entorno a través de efectores" [4], en esta definición describe una de las características de un agente computacional que es poder percibir y actuar en un entorno determinado.

Otra definición de agente es la propuesta por FIPA (Foundation for Intelligent Physical Agents): "Un agente es una entidad de software encapsulado con su propio estado, conducta, hilo de control y la habilidad para interactuar y comunicarse con otras entidades (gente, otros agentes o sistemas legados)", en esta definición se destaca la capacidad de un agente que no solo se comunica con el medio en el cual se encuentra, sino que también con otros agentes del sistemas y entidades externas como usuarios.

Entre otras definiciones también se encuentra: "Un agente es un sistema computacional que está situado en algún ambiente, y que es capaz de actuar autónomamente en dicho ambiente con el fin de cumplir sus objetivos" [5], donde se destaca que un agente posee autonomía, y es capaz de cumplir sus objetivos por sus propios medios.

2.1.2 Agentes Inteligentes

Para que un agente sea inteligente debe cumplir con ciertas características, las cuales son:

- Autonomía: Un agente a partir de sus propios conocimientos debe ser capaz de alcanzar sus objetivos, sin necesidad que un usuario le guíe.
- Sociabilidad: Los Agentes son capaces de comunicarse entre sí (Sistema Multiagente) y colaborar entre ellos para lograr un objetivo en común, como también interactuar con entidades externas al propio sistema, como es el caso del usuario.
- Reactividad: Un agente debe ser capaz de percibir estímulos, tanto de su ambiente como del mundo externo, estos estímulos afectan las acciones realizadas por el agente para alcanzar sus objetivos.
- Pro-actividad: Un agente no solo actúa en función de los estímulos, sino que puede realizar acciones como resultado de sus propias decisiones.

Aquellos agentes que posean estos atributos: autonomía, sociabilidad, reactividad y pro-actividad se clasifican en la noción débil de agente. Un agente es inteligente si es racional, coherente y adaptable, en mayor o menor medida. Un agente será más inteligente cuánto más

desarrolladas tenga estas características de inteligencia, racionalidad, coherencia y adaptación [6].

2.1.3 Sistemas Multiagente

Una de las definiciones de Sistema Multiagente es: "Un sistema multiagente es una red de problem-solvers que trabajan conjuntamente para en encontrar respuestas a problemas que ven más allá de las capacidades o conocimiento individuales de cada entidad" [7].

En general los sistemas multiagente son una sociedad de agentes que por medio de la interacción y colaboración entre los agentes se llega a una solución del problema. El éxito de la interacción de los agentes requiere que cada uno de estos tenga la capacidad de cooperar, coordinar y negociar con los demás agentes como también con los usuarios del sistema [8].

2.1.4 Comunicación de los Agentes

Para que los agentes puedan colaborar entre ellos necesitan un lenguaje de comunicación. FIPA ACL es uno lenguaje que está asociado con la arquitectura abierta de FIPA, el cual se basa en los actos de habla. Al igual que KQML, posee una sintaxis similar y está diseñado para trabajar con cualquier lenguaje y especificación de ontología. La diferencia de FIPA ACL con KQML es que poseen un grupo de performatives distintos [9].

2.1.5 Plataforma Multiagente: JADE

JADE (Java Agent Development Environment) es una plataforma de software completamente implementada en Java. Esta plataforma facilita el desarrollo de Sistemas Multiagente a través de un framework que cumple con las especificaciones FIPA y una serie de herramientas gráficas para administrar y monitorear la ejecución de los agentes. El objetivo de JADE es simplificar el desarrollo de agentes y a su vez garantizar el cumplimiento del estándar FIPA[10].

2.1.6 Metodología De Desarrollo De Agentes: PASSI

PASSI (Process of Agent Societies Specification and Implementation) es una metodología paso a paso para el diseño y desarrollo de sistemas Multiagente. PASSI integra los modelos de diseño y conceptos de la Ingeniería de Software Orientada a Objetos y los enfoques de la Inteligencia Artificial. Se utiliza UML como lenguaje de modelado, principalmente por su amplia aceptación y capacidad de extensión. El proceso de diseño de PASSI se compone de cinco modelos, los cuales pueden ser divididos en fases de construcción de Sistemas Multiagente [11].

2.2 Modelado 3D

El modelado 3D es un proceso en el cual por medio de una o varias técnicas de modelado se realizan los distintos objetos tridimensionales que se utilizan en los desarrollos de Sistemas en 3D. Hay distintas técnicas de modelado que se pueden utilizar por si solas o en conjunto. A continuación se describirán algunas técnicas para la creación de modelos 3D.

2.2.1 Modelado Poligonal o Modelado de Subdivisiones (Box Modeling)

Es una técnica de modelado poligonal en la que se comienza con una primitiva geométrica (cubo, esfera, cilindro, cono) y luego refina su forma hasta la apariencia deseada. Como se muestra en la figura 2.1. A menudo en esta técnica se trabaja por etapas, comenzando con una malla de baja resolución, a continuación, se va subdividiendo la malla para suavizar los bordes duros y agregar detalles, refinando la forma. El proceso de subdivisión y refinar se repite hasta que la malla contiene suficientes detalles poligonal para transmitir adecuadamente el concepto deseado [12].

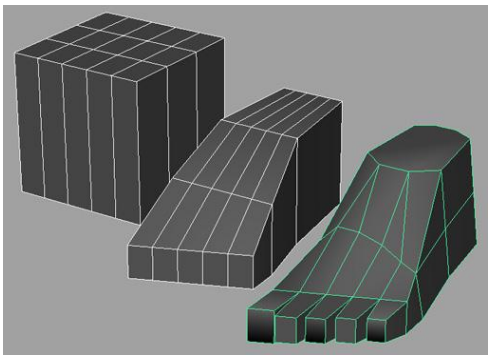


Figura 2.1 Ejemplo de Box Modeling

2.2.2 Modelado de Bordes (Edge Modeling)

Es otra técnica poligonal, aunque fundamentalmente diferente del box modeling. En el modelado de borde, en vez de comenzar con una forma primitiva y su refinamiento, el modelo es esencialmente construido pieza por pieza, mediante la colocación de conexiones entre las caras poligonales [13].

2.2.3 Modelado Mediante Splines (NURBS Modeling)

Consiste en la creación de modelos 3D a partir del uso de líneas vectoriales en 2D (splines) ubicadas en un espacio en 3D, a las cuales posteriormente se le aplican transformadores de malla para generar la superficie de la figura deseada, Esta técnica, a diferencia del Box Modeling, es mucho más adecuada para la creación de superficies orgánicas, es decir, modelos que posean una gran cantidad de superficies curvadas o con una complejidad media [14].

2.2.4 Escultura Digital (Digital Sculpting)

En esta técnica las herramientas utilizadas para modelar, dan la sensación de estar trabajando directamente con arcilla. Esta técnica de modelado principalmente trabaja con mallas de muy alta resolución a la que se le aplican distintos tipos de herramientas para ir dándole forma. Es muy directa en lo que se refiere a llegar a la forma final. Si el objeto generado a partir de esta técnica quiere ser utilizado con otros fines como animación o juegos, se debe realizar un proceso de retopología para bajar la cantidad de polígonos y hacerlo más manejable [13].

2.2.5 Herramienta De Modelado 3D, Blender

Es un programa informático multiplataforma, dedicado especialmente al modelado, animación y creación de gráficos tridimensionales. Inicialmente era un programa gratuito, pero sin código fuente, con una manual a la venta, luego paso a ser software libre [15]. En el área del modelado permite utilizar los operadores clásicos, en los modos de aristas, vértices y caras. Además admite técnicas de subdivisión de Catmull-Clark, mayas de resolución adaptativa, metabolas y metasuperficie, entre otros. Incluye además un modo esculpir que funciona como una metáfora de pincel 3D.

2.3 Formas de Texturizado

La fase de texturizado es tan importante como la de modelado, sobre todo si lo que se busca es realismo. El texturizado no sólo permite añadir color al modelo, sino que también permite simular diferentes materiales, por ejemplo, metal, madera, etc. Las texturas pueden pintarse en un software de creación de imágenes digitales o puede extraerse de fotografías de texturas reales [16].

2.3.1 Texturizado por Imagen

Esta técnica se basa en la utilización de imágenes para generar la textura de un modelo 3D. Debido a que el tamaño de la textura influye directamente en el rendimiento que tendrá posteriormente la aplicación, es necesario que esta tenga un tamaño lo más pequeño posible, considerando una proporcionalidad con el número de pixeles que se ocupará en la escena. Para el uso de memoria más eficiente, es recomendable utilizar texturas cuadradas, como 32x32, 64x64, 128x128, etc [17].

Existen muchas formas de unir la imagen al modelo, una de las más utilizadas es el *UV Mapping*. Esta técnica consiste en la asignación de pixeles de la imagen 2D a la superficie del polígono 3D, mediante coordenadas XY. Cuando un modelo es creado como una malla poligonal utilizando un modelador 3D, las coordenadas UV pueden ser generadas para cada uno de los vértices de la malla, si luego el modelo quiere pasarse a un motor gráfico, es necesario que la geometría posea sus respectivas coordenadas XY [17].

2.4 Motores Gráficos 3D

Los motores gráficos son la herramienta fundamental de las aplicaciones que trabajan con gráficos tridimensionales. Estos manipulan los distintos modelos 3D, lo que permiten crear un mundo tridimensional muy detallado por medio de técnicas de programación, permitiendo una sensación de realismo.

2.4.1 APIs Graficas

Las APIs graficas son un conjunto de funciones que actúan como capa intermedia que permite la comunicación entre el motor 3D con el hardware del sistema. Están destinadas únicamente al manejo de gráficos representados por pantalla, estas interfaces se encuentran limitadas a las capacidades de hardware que posea cada computador en el que se desarrolla alguna aplicación. A continuación se describirán las dos APIs más importantes que existen actualmente.

2.4.1.1 OpenGL

Es una interfaz de software para hardware gráfico. Consta de una biblioteca de modelado y graficas 3D, es portable y muy rápida. Fue desarrollado por Silicon Graphics, luego paso a convertirse en un estándar cuando lo adapto Microsoft para su sistema operativo Windows. Esta API posee una gran cantidad de funcionalidades, como el soporte de luces y sombras, transparencia, mapeado de textura, animación de modelos, soporte de nieblas, etc [18].

2.4.1.2 Direct3D

Es una API que posee DirectX, se encarga de todo el procesamiento grafico para poder visualizar imágenes 3D por pantalla. Esta API fue creada por Microsoft y permite a los programadores de juegos programar eficientemente en Windows. Direct3D aumenta de forma considerable el rendimiento ya que trabaja directamente con el hardware gráfico [19].

2.4.2 Técnicas utilizadas por motores gráficos

Las distintas técnicas que utiliza un motor gráfico son principalmente para darle un mayor número de detalle a la imagen y así ofrecer un mayor realismo. A continuación se describen algunas de las técnicas utilizadas por los motores gráficos.

2.4.2.1 Antialiasing

Funciona renderizando la imagen a una resolución mayor a la deseada, para eliminar el efecto de dentado llamado Aliasing. Actualmente existen técnicas mucho más eficientes como Quincuix, Antialiasing 2X, 4X, 6X, 8X, entre otras [20].

2.4.2.2 VSync

También llamada Sincronización Vertical, es una función que sincroniza el número de frame generados por la tarjeta gráfica con la taza de refresco del monitor. Esto previene la aparición de imágenes erróneas y movimientos bruscos que son producidos debido a que las tarjetas gráficas producen gran cantidad de imágenes por segundo superando a la cantidad generada por los monitores [21].

2.4.2.3 Iluminación

En la computación grafica existen dos tipos de luz: luz ambiental que está presente en cada punto de la escena 3D y posee una entidad constante y aparenta venir de todas direcciones. El otro tipo de luz es la directa que proviene de una fuente de luz que es posible encontrar. Hay tres tipos de luz directa [22]:

- Luz direccional: Representaría a la luz del sol, ya que viene de una fuente de luz distante y no posee cambios de ángulo, intensidad o color.
- Punto de luz: Punto en el espacio que emite luz en todas direcciones, por lo que para calcular la luz en un objeto se necesita saber la posición, color e intensidad.
- Foco de luz: este tipo de luz requiere más cálculos, ya que es similar al punto de luz, pero este emite luz en un ángulo y dirección especifico.

2.4.2.4 Sombreado

Las sombras se utilizan para dar mayor realismo a una imagen debido a que se revela la posición de la fuente de luz su intensidad y la profundidad de los objetos. Una de las técnicas más utilizadas en este ámbito es la llamada "Shadow maps", esta utiliza algoritmos de superficie visibles en el Z-buffer, en el cual se guardan valores de profundidad para cada punto de la escena con respecto a la fuente de luz [23].

2.4.3 Motor 3D seleccionado

El motor gráfico utilizado para este proyecto es Jmonkey Engine (JME). Es un motor 3D basado en java, de código abierto bajo licencia de BSD. Está orientado al desarrollo de videojuegos en tres dimensiones. La API gráfica utilizada por este motor es OpenGL, pero debido a su capa de abstracción, permite que cualquier API de renderizado pueda incorporarse como un plugin. LWJGL y JOGL ya están incorporadas.

Se basa en estructura de árbol de nodos, esto permite la organización de los datos del sistema tridimensional en un grafo de nodos, donde cada nodo puede tener muchos hijos pero provenientes de un solo padre.

En JME pueden usarse muchos tipos de geometrías, como: líneas, puntos, modelos, terrenos, niveles de detalles y más. También se puede hacer uso de muchos efectos de alto nivel como: renderizado en textura, mapeo del entorno, lentes, tintado, sistema de partículas, entre otros efectos.

El SDK JMonkey Engine, viene en forma de jMonkey Plataform, un IDE completo basado en la plataforma NetBeans con editores gráficos y las capacidades plugin, jMonkey Platform incluye su propio plugin repositorio SVN, que en comparación con otras aplicaciones como JAVA 3D vemos que esta herramienta facilita el trabajo gracias a su entorno de diseño integrado [24].

Capítulo 3: Definición Del Problema

El problema consiste en desarrollar un simulador 3D a un Sistema multiagente ya existente que da solución al problema de asignación de barcos a muelles denominado Berth Allocation Problem (BAP), y al mismo tiempo poder corregir y modificar si es necesario el código del sistema para llegar a un mejor resultado.

El objetivo principal del proyecto es poder representar de manera fácil al usuario el trabajo de asignación del SMA y cómo actúa frente a situaciones comunes, como el atraso de un barco, la cancelación de este, o si ocurre algún problema con un punto de atraque.

3.1 Berth Allocation Problem

BAP es un problema que se cuestiona qué localización en el muelle tomarán un conjunto de barcos y como organizar el tiempo de servicio de cada uno, de manera de optimizar alguna función objetivo. Para ello se deben considerar ciertas características, como el tiempo de servicio que tendrá el barco, tiempo de llegada, tiempo de salida, entre otros. Con el fin de que el tiempo que debe permanecer un barco en el puerto sea mínimo, como también el tiempo de espera de los puntos de atraque.

3.1.1 Restricciones Espaciales

Existen variados estudios realizados por distintos grupos para resolver la problemática de Berth Allocation problem, pero básicamente se divide en los siguientes enfoques [25]:

- Discreto: en este caso el muelle se divide en un número finito de secciones de igual longitud denominados puntos de atraque, donde cada punto de atraque está destinado a recibir un único barco a la vez. Este método posee ciertas complicaciones, ya que si se considera espacios muy pequeños se dificulta la búsqueda de una solución, y si por el contrario se segmenta en secciones muy grandes, se desperdicia el espacio, debido a lo variable que son los tamaños de los buques.
- Continuo: En el caso continuo el muelle no está dividido en secciones, por lo que la embarcación puede atracar en cualquier lugar del muelle, dependiendo solo de la posición de los demás barcos que están atracados.
- Híbrido: como en el caso discreto, el muelle está dividido en punto de atraque, pero en este caso los barcos grandes pueden ocupar más de un atraque así como los barcos más pequeños pueden compartir un punto de atraque.

3.1.2 Restricciones Temporales

También existen restricciones temporales para medir el tiempo de atraque de la embarcación y para los horarios de llegada y salida del muelle. Para ello se distinguen dos modelos:

- Estático: No hay tiempo de llegada para los barcos ni tampoco restricción en el tiempo que se mantiene un barco atracado en el muelle. Todos los barcos se encuentran en el muelle y pueden ser asignados a un atraque inmediatamente.
- Dinámico: Se considera el tiempo, los horarios de llegada que se le asignan a los barcos son fijos, por lo tanto no se le permite que atraquen antes de la hora señalada.

Esto permite que se mantengan los horarios previstos. En este modelo todo el servicio del barco debe ser ejecutado en una ventana de tiempo.

3.2 Solución Del Problema

En este proyecto se tomará el trabajo realizado por Romina López e Ivonne Ramírez para la realización del simulador 3D. Es por ello que a continuación se demostrará la solución al problema de BAP propuesta en su trabajo [25].

En la solución propuesta se consideró el BAP en forma dinámica como restricción temporal, y se implementa el modelo discreto como restricción espacial. La metodología utilizada para el desarrollo del sistema multiagente es la metodología PASSI.

3.2.1 Formulación Matemática

Como se mencionó anteriormente, en el proyecto se tratará la versión discreta del BAP. Para la creación de la solución se utilizó la siguiente formulación matemática.

Para la solución de la versión discreta del BAP el problema debe ser modelado como un problema de Multi-Depot Vehicle Routing Problem with Time Windows (MDVRPTW). En este modelo los barcos son vistos como clientes y los puntos de atraque como depósitos de vehículos, donde están ubicados cada uno. Existen M vehículos, uno para cada depósito. Además, cada vehículo parte y termina el recorrido en su propio depósito. Los barcos son modelados como vértices de un multigrafo. Cada depósito es dividido en vértices de origen y destino y ventanas de tiempo pueden ser consideradas en cada vértice. En el origen y destino, la ventana corresponde al período disponible de determinado punto de atraque [25].

Las variables son:

ai: Límite inferior de la ventana de tiempo de servicio del barco i.

bi: Límite superior de la ventana de tiempo de servicio del barco i.

vi: el valor, o costo, del tiempo de servicio para el barco i.

El problema se modela como un multigrafo $G_k = (V_k, A_k), \forall k \in M,$ donde $Vk = N \cup \{o(k), d(k)\}$ y $A_k \subseteq V_k \, x \, V_k.$

Las siguientes variables y constantes son definidas:

- $X_{ijk} \in \{0.1\}$ $k \in M$, (i, j) A_k, $X_{ipk} = 1$, sí y solo sí el barco *j* está programado después del barco *i* en el punto de atraque *k*;
- T_{ik} $k \in M$, $i \in N$: el tiempo de atraque del barco *i* en el punto de atraque *k*.
- $T_{o(k)-k}$, $k \in M$: el tiempo de comienzo de operaciones del punto de atraque *k*, dado por el primer barco que atraca en ese punto.
- $T_{d(k)-k}$, $k \in M$ *:* el tiempo de cese de operaciones del punto de atraque *k*, dado por el tiempo de salida del último barco en ese punto.
- M_{ijk} : max $\{ b_i + t_{ik} - a_j, 0 \}$, $k \in M$, *i* y $j \in N$

El MDVRPTW se modela de la siguiente manera:

Minimizar

$$\sum_{i \in N} \sum_{k \in M} v_i \left[T_i^k - a_i + t_i^k \sum_{j \in N \cup \{d(k)\}} x_{ij}^k \right] \tag{1}$$

$$\sum_{k \in M} \sum_{j \in N \cup \{d(k)\}} x_{ij}^k = 1 \qquad \forall i \in N \tag{2}$$

$$\sum_{j \in N \cup \{d(k)\}} x_{o(k),j}^k = 1 \qquad \forall k \in M \tag{3}$$

$$\sum_{i \in N \cup \{o(k)\}} x_{i,d(k)}^k = 1 \qquad \forall k \in M \tag{4}$$

$$\sum_{j \in N \cup \{d(k)\}} x_{ij}^k - \sum_{i \in N \cup \{o(k)\}} x_{ji}^k = 0 \qquad \forall k \in M, \forall i \in N \tag{5}$$

$$T_i^k + t_i^k - T_j^k \leq (1 - x_{ij}^k) M_{ij}^k \qquad \forall k \in M, \forall (i,j) \in A^k \tag{6}$$

$$a_i \leq T_i^k \qquad \forall k \in M, \forall i \in N \tag{7}$$

$$T_i^k + t_i^k \sum_{j \in N \cup \{d(k)\}} x_{ij}^k \leq b_i \qquad \forall k \in M, \forall i \in N \tag{8}$$

$$s^k \leq T_{o(k)}^k \qquad \forall k \in M \tag{9}$$

$$T_{d(k)}^k \leq e^k \qquad \forall k \in M \tag{10}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall k \in M, \forall (i,j) \in A^k. \tag{11}$$

La función objetivo es la minimización de la suma de los tiempos de servicio con un cierto peso asignado. Cuando el barco i no está asignada al punto de atraque *k*, el término correspondiente en la función objetiva es 0 puesto que $\sum_{j \in n \cup d(k)} x_{ij}^k = 0$, por lo tanto, el objetivo es minimizado. La restricción (2) indica que para cada barco *i* existe exactamente un arco activo$(i;\ j)\ \in\ A_k., \forall k\ \in\ M$. Las restricciones (3) y (4) definen el grado de los depósitos, mientras que la conservación del flujo para los restantes vértices está asegurada por la restricción (5). La consistencia de las *Tk* variables con la secuencia en el punto de atraque se logra mediante la restricción (6). Las ventanas de tiempo de servicio en los barcos son fijadas por las restricciones (7) y (8), y las ventanas de tiempo de la disponibilidad en los puntos de atraque por (9) y (10) [25].

3.2.2 Arquitectura MADARP

La arquitectura multiagente escogida para la solución del problema es MADARP, esta arquitectura es utilizada para resolver el Dial a Ride Problem (DARP) y fue diseñada para el desarrollo de sistemas multiagente enfocados en transporte de pasajeros, pero se puede adaptar fácilmente para utilizarla en el BAP.

A continuación se definirán los distintos agentes que participan para comprender con mayor claridad en que consiste la estructura MADARP. Es importante destacar que esta estructura está realizada para su utilización en el transporte de pasajeros, pero fue modificada para utilizarla en el desarrollo de BAP.

- **Agente Cliente**: es el responsable de obtener todos los requerimientos de los clientes respecto al transporte que desean.
- **Agente Trip-request**: toma los requerimientos y preferencias de los clientes para actuar a favor de él, y puede actuar como un asistente personal de viajes. Tanto el agente Cliente como el Trip-request hacen que haya una total comunicación con el usuario.
- **Agente Vehículo**: es visto como interfaz, con ello se puede comunicar el vehículo y su conductor, entregando información sobre contingencia o cambio de planes, además monitorea el estado y progreso de los vehículos.
- **Agente Schedule**: maneja la ruta del vehículo y realiza cualquier ajuste de la ruta o en la programación frente a eventualidades.
- **Agente Planner**: hace el servicio de transporte a los clientes.
- **Agente Broker y agente Map**: colaboran para manejar bien la planificación y control, como por ejemplo acceso a datos o rutas.

En general los agentes de interfaz entregan la entrada y la señal para que los agentes de planeamiento puedan cumplir con su labor de planear la ruta de los vehículos; por su parte, los agentes de servicio ofrecen y dan la información necesaria.

3.2.3 Arquitectura MABAP

Existe una alternativa para la aplicación de agentes que puede dar solución al BAP. Esta arquitectura multiagente es específicamente diseñada para solucionar el problema de asignación de barcos a muelles. Esta arquitectura permite dividir el problema en varios problemas de menor magnitud, para disminuir la complejidad inicial y cada uno de los problemas es resuelto por agentes específicos. Esta arquitectura fue propuesta por Rene Díaz en su proyecto de solución al problema de Berth Allocation Problem [26].

3.2.4 Diseño de la Arquitectura Propuesta

Dentro de las arquitecturas mencionadas y definidas con anterioridad, en el proyecto de Romina e Ivonne se implementa la arquitectura MADARP y también la propuesta por Rene Díaz. Sin embargo, la arquitectura especificada no cumple totalmente con lo que se quería plantear como solución dinámica del BAP en el trabajo de Ivonne y Romina, por lo que se realizó una adaptación tratando de mantener la arquitectura. Se integró una nueva capa de monitoreo quedando la arquitectura propuesta por 5 capas. Esto se debió a que el problema se plantea dinámicamente, y eso implica considerar la variable de tiempo para la ejecución de todo el proceso en tiempo real.

La funcionalidad de la nueva capa de monitoreo es permitir observar el comportamiento del sistema, respecto a los eventos que sucedan en los barcos y en el berth, logrando simular las acciones de cada uno de ellos mediante los agentes simuladores, y tener un control del

proceso de planificación del BAP, esto permite modificaciones y planificaciones en tiempo real.

Hay que entender algunos conceptos referidos al DARP, que son utilizados de manera distinta en el BAP. Estos se explican a continuación:

- Los clientes corresponden a los barcos y los vehículos corresponden a los berth.
- Los clientes de entrada y/o salida, en el BAP son ambos a la vez, ya que se requiere una hora determinada para llegar y una hora determinada para salir del berth.
- Un slack corresponde al tiempo desde que el berth está listo para recibir una nueva embarcación hasta que un nuevo barco llegue definitivamente.
- En un puerto existen múltiples berth que llevan a cabo el servicio propuesto por un barco, pero cada punto de atraque puede atender a un solo barco a la vez, a diferencia de los vehículos que pueden llevar varios clientes a la vez.
- En la planificación se considera los periodos de utilización del berth y el tiempo de slack, esta será considerada día a día.
- Un bloque de planificación corresponde a todas las llegadas y requerimientos de un barco dentro de un berth en específico y en un determinado tiempo.
- La ventana de tiempo es un rango de tiempo dado para que el barco llegue y se retire del berth, y también donde se define el tiempo más temprano (ET) y más tardío (LT).

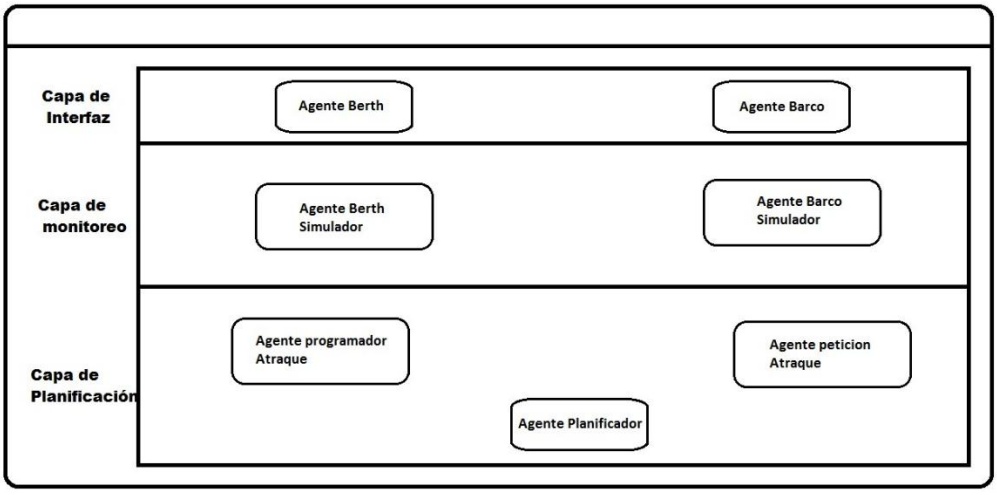


Figura 3.2 Arquitectura MABAP propuesta [25]

Finalmente en la figura 3.2 se muestra la arquitectura propuesta en el proyecto de Ivonne y Romina. A continuación se describen los agentes que se desarrollan en el sistema multiagente para solucionar el BAP.

Es importante destacar que los nombres de los agentes mencionados en la figura 3.2 sólo son una referencia y no son necesariamente los verdaderos nombres de los agentes. A continuación se describe cada uno de ellos con sus nombres respectivos.

- **Agente Barco (TripGen):** Es el responsable de obtener todos los requerimientos de los clientes reales, ya sean peticiones, cancelaciones, preferencias, etc. Como también Informa acerca del estado de la petición y eventos que puedan ocurrir al cliente real.
- **Agente Berth (SchedGen):** Es el encargado de brindar toda la información necesaria respecto a la asignación, situaciones de contingencia, cambios etc. También se encarga de informar el estado del berth, como también acerca de su progreso durante un periodo de tiempo.
- **Agente Programador de Atraque (Scheduler**): Es el responsable de la comunicación entre el agente Beth y el agente Planificador. Este agente genera una propuesta de atraque, la cual indica todas las posibilidades que se tienen para que el barco atraque dentro del berth, según su propia petición. Además lleva a cabo la programación de la secuencia de barcos en un berth, gestiona las eventualidades que pueden suceder dentro del este, como las eventualidades que se informan acerca de una embarcación.
- **Agente Planificador (Planner**): Gestiona todas las eventualidades, tanto de los berth como de los barcos y luego las informa. Además gestiona las peticiones del barco que son enviadas a las propuestas, como también las propuestas del berth, eligiendo la mejor y enviando la respuesta al barco de su petición.
- **Agente Petición Atraque (TripRequest):** Sirve de comunicación entre el agente barco y el agente planificador. Este Debe ser capaz de negociar la propuesta de atraque brindada por el agente Programador Atraque, para ello toma en cuenta la petición realizada por el barco, en el cual se encuentran sus preferencias.
- **Agente Simulador Barco (VesselEvent) y Agente Simulador Berth (BerthEvent):** Estos agentes están encargados de simular el momento en que, por ejemplo el barco llega al berth en la hora indicada y luego su descarga y salida, o también eventualidades como retrasos, cambios de horario, o cuando un berth deja de funcionar, entre otras funciones relacionadas con las embarcaciones y los berth. Todo esto para tratar de simular lo mejor posible a lo que realmente pude pasar en la realidad.

3.2.5 Algoritmo de Inserción para el BAP

En el proyecto de Romina e Ivonne se originó una nueva heurística que abarca el problema de asignación de barcos a muelles, para ello se realizaron modificaciones a la heurística creada en el DARP, la cual fue tomada como base para lograr el algoritmo de inserción para el BAP.

Un cambio del BAP respecto al DARP es que el evento de carga y descarga se realiza dentro del mismo berth y necesariamente junto, ya que en el momento en que el barco atraca se le entrega el servicio y se va del muelle quedando el berth disponible para atender otro barco. El tiempo de slack se produce al momento de comenzar cada sub-bloque, por lo tanto, el pase de un sub-bloque a otro siempre genera un tiempo de inactividad (estado en el que el berth se encuentra sin servicio de un barco).

La figura 3.3 demuestra el evento de llegada y salida de un barco, A+ y A-, respectivamente, luego B+ y B-, C+ y C-, y así sucesivamente; también se demuestran los slack producidos antes del evento de llegada y un horizonte de planificación de un berth especifico.

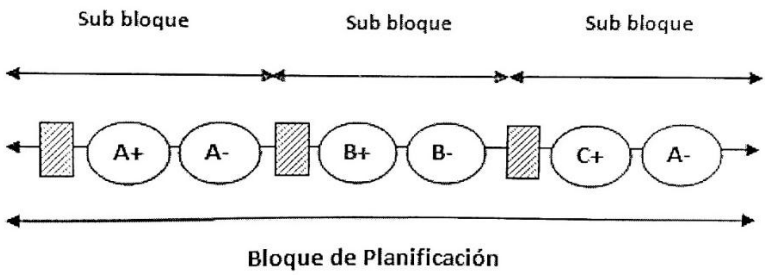


Figura 3.3 Bloque de planificación para el BAP [25]

Otro cambio realizado es en las ventanas de tiempo. Existen dos ventanas de tiempo asociadas a un barco, estas afectan a la llegada o arribo del barco al berth, una es a su tiempo más temprano (EAT) y tardío (LAT) de llegada y a su tiempo más temprano (EDT) y tardío (LDT) de su partida. Ver figura 3.4.

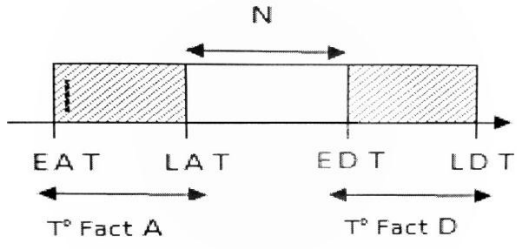


Figura 3.4 Ventanas de Tiempo para el BAP [25]

Como se muestra en la figura 3.5 las ventanas de tiempo (EAT, LAT) y (EDL, LDT) se van juntando hasta formar una sola, esto se debe a que el tiempo más tardío de llegada será el mismo que el tiempo más temprano de partida (LAT=EDT).

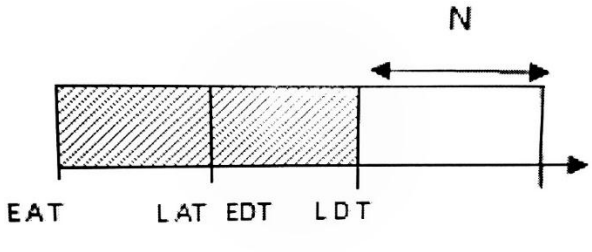


Figura 3.5 Formación de la nueva ventana de tiempo [25]

Por lo tanto queda una sola ventana de tiempo (EAT, LDT), en donde se obtiene una ventana de tiempo factible, la cual corresponde al tiempo más temprano de llegada (EAT) y al tiempo más tardío de partida (LDT), menos el tiempo de servicio denotado N, que necesita el barco. Este tiempo es el que servirá para la intersección. En la figura 3.6 se muestra la ventana de tiempo resultante para resolver el BAP.

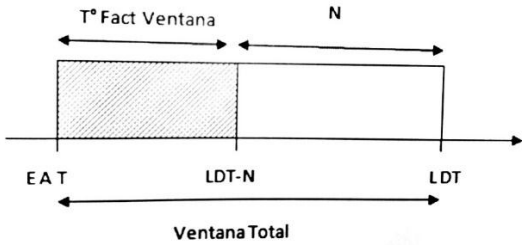


Figura 3.6 Ventana de tiempo resultante para el BAP [25]

3.2.6 Diseño De La Interfaz 3D

Esta parte del proyecto se basará en el trabajo realizado por Gerardo Urbina [27], quien en su trabajo realizó una interfaz 3D al problema de transporte de pasajeros (DARP). Para la visualización de la interfaz en tres dimensiones se utilizó el motor JMonkey y como herramienta de modelado 3D, se utilizó Blender.

La interfaz 3D realizada en el trabajo de Urbina consiste en crear una clase llamada interfaz3D, la cual contiene todas las librerías y métodos encargados de inicializar y poner en marcha la simulación 3D, contando también con las clases Vehiculo3D y Cliente3D, las cuales entregan más información a la interfaz. Por otro lado para comunicar el sistema multiagente que resuelve el problema del DARP con la interfaz tridimensional se utiliza un agente llamado Manager3D, el cual comunica las acciones que se realizan en el sistema multiagente a la interfaz.

A continuación, en el siguiente capítulo se describirá el desarrollo de la interfaz tridimensional para el presente trabajo.

Capítulo 4: Desarrollo del Proyecto

4.1 Presentación del caso de estudio

Como se ha mencionado anteriormente el objetivo del proyecto es realizar una interfaz tridimensional con el propósito de simular los eventos que ocurren en el sistema, para su posterior análisis y comprensión. Lo principal que se quiere lograr con este proyecto es mostrar al usuario de manera fácil la asignación de barcos a muelles que realiza el SMA. También se quiere demostrar como el SMA actúa frente algún evento, por ejemplo un cambio de hora de un barco o la inactividad de un punto de atraque, debiendo reprogramar los barcos en tiempo real. Todos estos eventos se quieren representar en la simulación 3D, algunos ejemplos son:

- Llegada del barco al puerto.
- Instalación del barco en el berth que le fue otorgado anteriormente en la fase de planificación de barcos a berths.
- Momento en que el barco está recibiendo el servicio, ya sea cargando o descargando en el puerto.
- Retirada del barco al momento de terminar su servicio
- La inactividad de un berth por alguna imperfección.
- Retraso de un barco.
- Cancelación de la solicitud por parte de un barco.

Todos estos eventos de alguna forma deben ser representados en la simulación 3D, para esto se debe realizar una comprensión previa de cómo funciona el software encargado de realizar la planificación y la simulación de los eventos de cada barco y punto de atraque. A continuación se detalla cómo funciona el sistema, y cómo éste se comunicará con la interfaz 3D.

4.1.1 Desarrollo del caso de estudio

La interfaz tridimensional estará compuesta por un escenario de tierra y mar. En la tierra estará ubicado un puerto cualquiera con un número determinado de puntos de atraque (berths), donde los barcos deberán atracar. Por otra parte, por el mar llegarán los barcos a la hora y al berth indicado previamente en la planificación, para ser atendidos.

Los modelos tridimensionales principales que participan en la simulación 3D son los barcos y los berth, ya que los demás modelos sólo serán para proporcionar mayor realismo a la simulación. Al momento de comenzar con la simulación 3D los elementos serán cargados, donde se podrá apreciar los distintos berths en el puerto. Los barcos comenzaran a llegar al puerto a la hora que les fue indicada, para luego cargar o descargar y retirarse.

Por otro lado, el sistema multiagente ya implementado, quien tiene toda la lógica del problema se comunicará con la interfaz tridimensional por medio de un agente llamado Manager3D. Este será encargado de recibir la información del sistema multiagente e informársela a la interfaz 3D, para luego ser representada en la simulación. Para aquello es necesario que tanto el sistema multiagente como la interfaz 3D tengan una perfecta

sincronización, ya que puede suceder que el sistema vaya más rápido que la simulación, haciendo el proceso difícil de entender para el usuario.

En la interfaz tridimensional además se agregará información a cada berth y a cada barco, donde en cualquier momento de la simulación se podrá acceder a datos, como por ejemplo sobre el barco, su nombre, hora de llegada, tiempo de servicio y berth que se le fue asignado; al igual que a los berth, su nombre, barcos que tiene que atender, etc.

4.2 Diseño del sistema

Como se ha mencionado en el capítulo 3 de este informe, se realizará una interfaz tridimensional a un sistema multiagente ya existente, el cual fue abordado por López y Ramírez en su proyecto. Por lo mencionado anteriormente, antes de realizar la interfaz3D se debe comprender el trabajo realizado en el proyecto que se mencionó. En el anexo A.1 se muestra el diagrama de identificación de roles, que muestra cómo está constituido el sistema multiagente de dicho proyecto.

A continuación se describe la información que los agentes entregan al agente Manager 3D para luego informar a la nueva interfaz 3D.

- **Generar Berths:** Se generan los berths, donde luego serán asignados los barcos. La información es enviada a la interfaz 3D por medio del agente Manager 3D para representar los berth creados.
- **Generar solicitudes de Barcos:** Se generan las solicitudes de los barcos, luego de esto el programa elige el mejor escenario para asignar cada barco a un berth determinado.
- **Generar eventos del Berth:** Se generan eventos del berth como por ejemplo, que el berth ha dejado de funcionar.
- **Generar eventos del Barco:** Al igual que el anterior genera eventos, pero sobre el barco, como por ejemplo, cambio de hora de llegada de un barco o cancelación de la solicitud.
- **Obtener información del Berth:** Generar información del berth para que el usuario pueda obtenerla a partir de la interfaz 3D.
- **Obtener información del Barco**: Al igual que el anterior, se genera información del barco, para que el usuario pueda tenerla a disposición.
- **Inducir evento del Berth**: En este caso, es el usuario quien genera un evento a partir de la interfaz 3D, por lo que el agente Manager3D debe informarle al SMA.

En todas las funcionalidades descritas anteriormente el agente Manager 3D tiene que participar para poder comunicárselas a la interfaz 3D o viceversa. A continuación se representaran los diferentes modelos PASSI que representará de mejor manera la participación del agente Manager 3D.

4.2.1 Modelos PASSI

El proceso de diseño de PASSI se compone de cinco modelos, pero en este trabajo solo serán representado tres, los cuales son: Modelo de requerimientos del sistema, Modelo de sociedad de agentes y Modelo de implementación de agentes. Estos a su vez se dividen en fases de construcción de sistemas multiagente.

A continuación se representará los llamados modelos de requerimientos del sistema, en el cual se encuentran los diagramas de identificación de agentes (figura 4.1), identificación de roles (figura 4.2) e identificación de tareas (ver anexo B.4), donde participa el agente Manager 3D y los agentes que interactúan con él.

4.2.1.1 Diagrama de Identificación de Agentes

En la figura 4.1 se puede observar los agentes principales que participan en el sistema multiagente con sus funcionalidades y cómo interactúan entre ellos. Los agentes que se comunican directamente con el agente Manager3D son el agente SchedGen, TripGen, VesselEvent y BerthEvent como muestra la imagen.

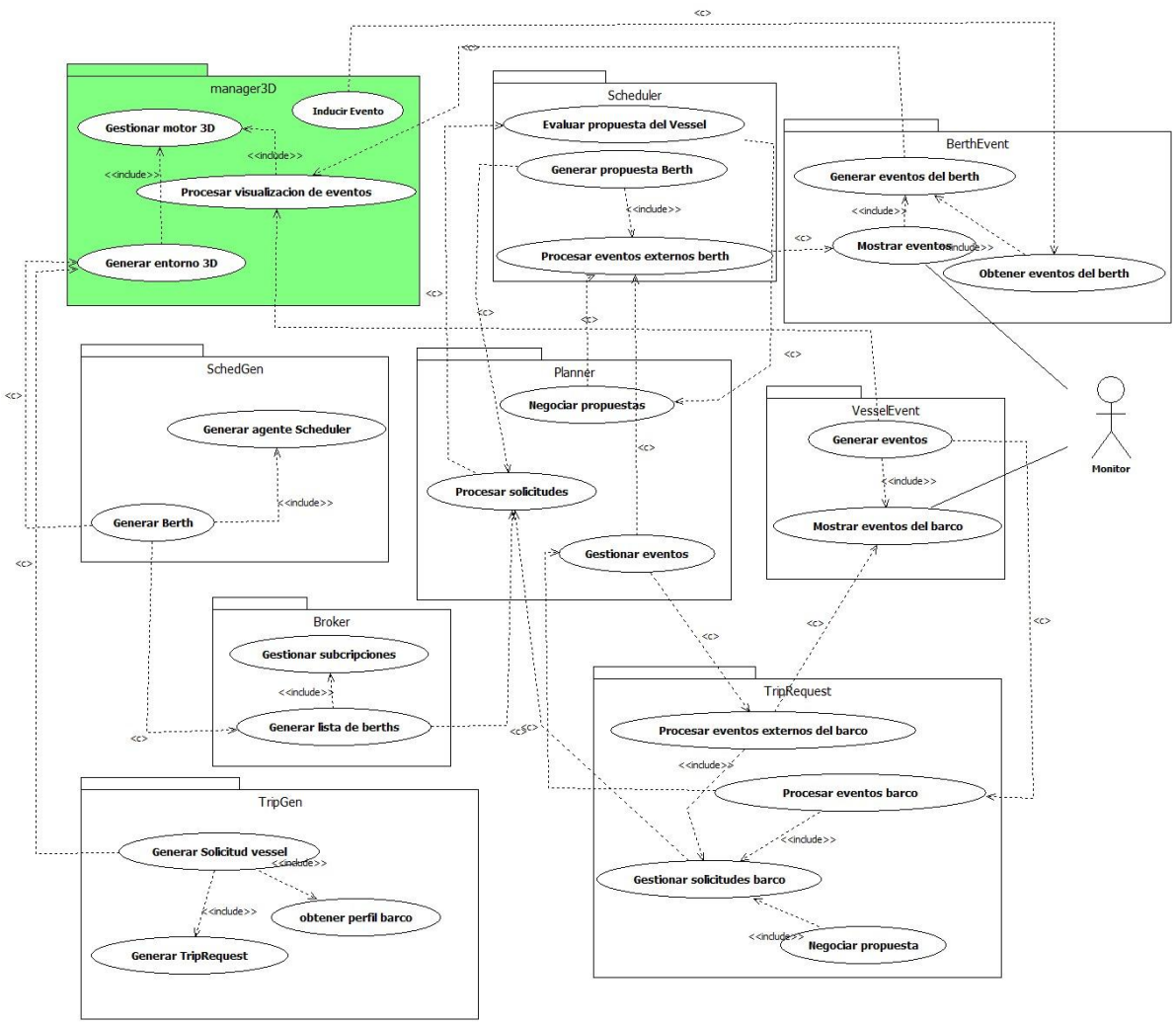


Figura 4.1 Diagrama de identificación de agentes con agente Manager3D

4.2.1.2 Diagrama de Identificación de Roles

En la figura 4.2 se muestra un diagrama de identificación de roles que representa el momento de inicio de la interfaz3D, donde el agente "Main" crea los agentes Manager 3D, SchedGen y TripGen, y estos dos últimos le informan al agente Manager 3D la creación de los

berths y solicitudes respectivamente, para después informar a la interfaz 3D que comience la simulación. En los anexos B.1, B.2 y B.3 se representan las figuras de otros diagramas de identificación de roles, donde se puede apreciar distintos agentes interactuando con el agente Manager 3D.

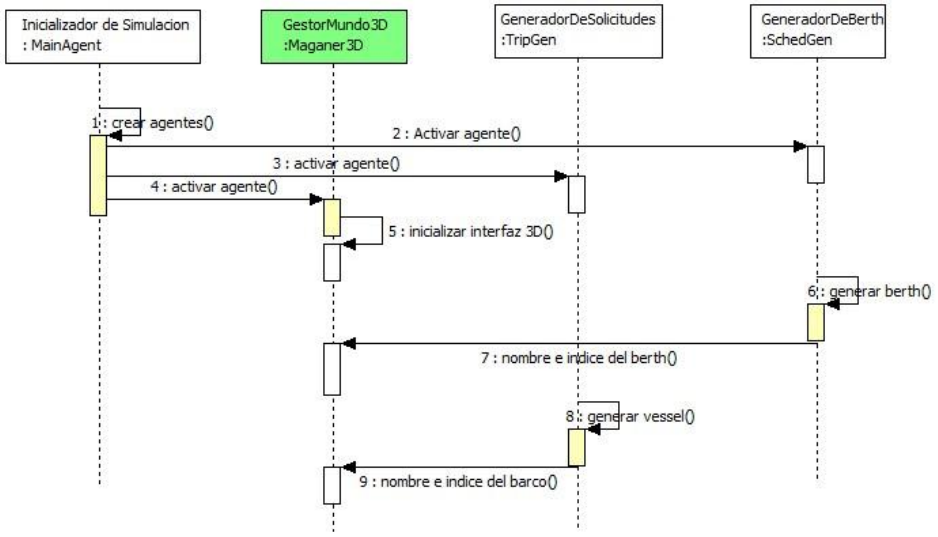


Figura 4.2 Diagrama de identificación de roles

Otros tipos de diagramas que representan el comportamiento y comunicación de los distintos agentes con el agente manager 3D son:

Modelo de Sociedad de Agentes: Entre estos modelos se encuentra el diagrama de descripción de ontología de dominio (ver anexo C.1), de descripción de ontología de comunicación (ver anexo C.2) y descripción de roles (ver anexo C.3). Igualmente que los diagramas anteriores, están centrados en el comportamiento del agente Manager 3D.

Modelo de Implementación de Agentes: En este tercer Modelo de PASSI contiene los diagramas que definen la estructura multiagente del sistema (ver anexo D.1) y del agente individual, en este caso el agente Manager 3D (ver anexo D.2).

4.2.3 Datos de entrada

El programa consta de un archivo de texto de entrada llamado “Adartw.ini” donde contiene los datos de inicialización del programa. Algunos de los campos del archivo, por ejemplo contienen información del modo en que se ejecutará la simulación, indicación de directorios, entre otros.

Al momento de realizar la interfaz 3D, se agrega otro campo al archivo llamado “INTERFACE_3D”, que da la opción de realizar la simulación 3D o no realizarla, ya sea con un 1 o un 0 respectivamente.

4.2.4 Diseño Simulador Grafico

Anteriormente se refirió al diseño de la solución entre la comunicación de agente Manager 3D y el sistema multiagente que resuelve el problema de Berth Allocation Problem. En este apartado se centrara en el diseño del motor gráfico y su comunicación con el agente Manager 3D.

El Simulador 3D está compuesto por 3 clases, que serán nombradas y descritas a continuación:

- **Interfaz3D:** Se encarga de inicializar la interfaz 3D, crea las instancias de las clases Barco3D y Berth3D que serán descritas posteriormente. El agente Manager 3D contiene una instancia de esta clase, y es por la cual llama a los métodos que contiene, dependiendo de la información que le llega del sistema multiagente. La Iterfaz3D es encargada de procesar toda la información y actualizarla por pantalla.
- **Barco3D:** se encarga de crear los barcos que participarán en la simulación y guardar su información.
- **Berth3D:** se encarga de crear los berth en la simulación 3D y guardar su información

En la figura E.1 en Anexos, se representa lo mencionado anteriormente en un diagrama de clases.

4.3 Modificaciones al Código del Sistema

Antes de dar inicio al capítulo 5, que se refiere a la implementación de la interfaz tridimensional, es necesario indicar las principales modificaciones que se realizaron al código del sistema multiagente, para que funcionara de forma correcta. Cabe destacar que este proyecto no solo implicó realizar una interfaz tridimensional, sino que también requirió entender el sistema multiagente y modificar algunas irregularidades para darle un mejor funcionamiento. A continuación se darán a conocer las irregularidades del sistema y la solución:

- **Los berths reciben a más de un barco a la vez:** La manera correcta de insertar los barcos en un berth es con los eventos “llegada” y “salida” del barco juntos, ya que un berth puede recibir a un barco a la vez, el inconveniente consiste en que se inserta el evento “llegada” de un barco y luego se inserta otro barco antes del evento “salida” del barco anterior.
- **Información incompleta en la interfaz:** No se muestra completamente en la interfaz todos los eventos en el momento de la simulación, ya sea las llegadas y las salidas de los barcos en cada berth, solo se muestran algunos.
- **Barcos sin tiempo de servicio:** No se toma en cuenta el tiempo de servicio de cada barco, solo se toma en cuenta el primer barco, y en los demás el tiempo de llegada con el tiempo de salida son el mismo.
- **Los barcos vuelven a ser atendidos en caso de Breakdown**: El evento Breakdown se refiere a que el berth no puede seguir operativo, por lo que los barcos que están asignados a él deben ser reprogramados en otros berth. Al ocurrir un evento Breakdown, lo barcos que ya fueron atendidos por el berth descompuesto vuelven a ser asignados a otro berth y atendidos nuevamente.

Para la solución de estos inconvenientes se debió analizar y comprender el código en su totalidad, para luego solucionar las irregularidades mencionadas anteriormente. A continuación se muestra la solución para cada una de los problemas:

- **Los berths reciben a más de un barco a la vez:** se cambió el orden de llegada de los parámetros en el método *"GenConstrProps( )"* de la clase ClientEventsAgent, como se muestra en la figura 4.3, ya que los parámetros llegaban en forma intercambiada.

```
    private Vector GenConstrProps(long ept, long nfac, long ldt,//se cambio la posicion del nfact con el ldt
242                               long drt,long mrt, long pickupST) {//y la de mrt por drt
243         //generate the properties list
            Vector constraintProps = new Vector();
245
246         constraintProps.add(new ConstraintProperty("PickupTime", new AndLOC(
```

Figura 4.3 Modificación de código (1)

- **Información incompleta en la interfaz:** en el método *"getNextEvent()"*, en la clase VehicleEventsAgent se cambió la condición de la sentencia "if" donde ahora se compara el nombre del barco y el evento. Figura 4.4, esto se realiza, ya que existe un vector de eventos y la función de este método es retornar el evento siguiente del vector, por lo que compara el nombre del barco y el nombre del evento del vector, con el nombre del barco y el nombre del evento actual, para luego retornar el evento que sigue.

```
466     //se compara el nombre del barco y el evento si es igual
467     if (((Event)meAgent.mySchedule.vehicle.trips.get(k)).name_User().equals(actual.name_User())&&
468             ((Event)meAgent.mySchedule.vehicle.trips.get(k)).evType.equals(actual.evType)){
                    if(actual.evType.equals("etStopDepot")) return(null); // el ultimo evento de cada berth
470 //                  System.out.println("###"+meAgent.getName()+"####GET_NEXT_EVENT#####->"+((Event)meAgent.myS
471                  return (Event)(meAgent.mySchedule.vehicle.trips.get(k + 1));//retorna el evento siguiente
472     }
```

Figura 4.4 Modificación de código (2)

- **Barcos sin tiempo de servicio:** En el método *"firstClientOnBus()",* en la clase CustomerInsertion se des comentó una línea de código, la cual era importante para el cálculo del tiempo de servicio de todos los barcos al momento de ser insertarlos en cada berth. Figura 4.5 La variable DRT es quien contiene el tiempo de servicio del barco, la cual es asignada a la variable T2Node, y esta será utilizada en el cálculo de la inserción del barco al berth.

```
137   ////    up.T2Node.setTime(dist_start_up.getTime());
138           down.T2Node.setTime(down.user.DRT.getTime());
139   ////    dep_stop.T2Node.setTime(dist_down_stop.getTime());
140
```

Figura 4.5 Modificación del código (3)

- **Los barcos vuelven a ser atendidos en caso de Breakdown**: Para solucionar este problema se creó una variable en la clase "Event" llamada estado, la cual es de tipo entero y al momento de que el barco es atendido cambia a valor uno, por el contrario sigue con el valor por defecto, el cual es cero. Así al momento de ocurrir un evento Breakdown se consulta a esta variable de cada evento de los barcos para ver si es necesario asignarlo a un nuevo berth o no.

Capítulo 5: Pruebas de Software

Las pruebas son esenciales para verificar que el funcionamiento del sistema sea el correcto. Hay distintos tipos de pruebas, como son las de caja negra, de caja blanca, de rendimiento, de usabilidad, de seguridad, entre otras. A continuación se describirá la planificación, el diseño de las pruebas y los datos de entrada utilizados.

5.1 Planificación de las Pruebas

En este apartado se describirán los tipos de prueba que se aplicarán al software, en este caso a la interfaz tridimensional. Hay distintos tipos de prueba, pero en este proyecto se han escogido las siguientes:

- **Pruebas de caja negra:** Se les hace énfasis a los datos de entrada y de salida, verificando que según los datos que entren, el resultado de las salidas debiera ser consistente y correcto. Se consideró esta prueba, ya que se necesita evaluar la interacción que tiene la interfaz 3D con el medio que la rodea, esto es porque la mayoría de la información llega de parte externa a la interfaz.
- **Pruebas de caja blanca:** Se prueba el funcionamiento interno del sistema, revisando cada método que tiene, verificando que tengan un buen funcionamiento. Se escogió este método ya que se necesita verificar que todas las funcionalidades del sistema cumplan bien su objetivo.
- **Pruebas de rendimiento:** Las cuales está enfocada al rendimiento del sistema, dependiendo en donde sea utilizado. En este caso se quiere evaluar cómo funciona el simulador, con respecto a las actualizaciones de pantalla que puede realizar por segundo. Buscando un equilibrio en donde el software funcione correctamente, dependiendo de las capacidades del hardware en el cual se está utilizando.

Para comenzar se probarán los elementos que constituyen a la creación del simulador. En esta etapa se evaluarán la correcta resolución de la pantalla, la activación de las técnicas de antialiasing y sincronización vertical, etc. Luego se probara la inicialización de la interfaz, acá se evaluará la creación del escenario 3D, la integración de algunos modelos y texturas a la simulación, como los efectos de sombra y luz. Posteriormente se evaluara la comunicación del sistema multiagente con el simulador, verificando que la información llegue de forma correcta a la interfaz, por medio del agente Manager3D.

5.2 Diseño de Pruebas

En la primera parte, como se mencionó en la planificación, se evaluara la resolución de la pantalla, la activación y desactivación de la técnica VSync y el aumento o disminución de la calidad del antialiasing. A continuación se describirán los problemas mencionados anteriormente y lo que se espera conseguir.

- **Resolución de la pantalla:** Se probará el tamaño de la ventana en la cual se verá la simulación. La ventana deberá ser creada respecto a los valores asignados a su anchura y altura.
- **Activación y desactivación de VSync:** Esta técnica al ser activada, permite que la cantidad de frame por segundos del motor gráfico se sincronice con los frame que puede renderizar la pantalla. En este caso al no activarla, podrían ocurrir algunos errores gráficos, como desaparición de geometría, desfases de la pantalla, etc.
- **Aumento o disminución de la calidad del antialiasing:** esta técnica consiste en eliminar el borde dentado de las imágenes, entonces dependiendo del valor que se asigne a esta técnica será la calidad del filtro. Dependiendo del valor se percibirá más o menos el dentado de las imágenes.

  En la segunda etapa de pruebas se evaluara la inicialización de la interfaz 3D, se revisaran la carga e integración de los modelos, los efectos de luz y sombra; y el camino que recorrerán los barcos dependiendo del berth que se les haya asignado.

- **Integración de los modelos:** En esta prueba se verifica que el modelo se haya cargado de forma correcta, ya sea en la posición indicada, con su textura correcta y que posea todos los polígonos que se le hayan especificado.
- **Efectos de luz y sombra:** Se evaluará que tanto las sombras como la luz afecte a todas las figuras integradas a la interfaz, que cada una proyecte su sombra y que reciba la luz, ya sea ambiental como direccional.
- **Recorrido de los barcos:** Se evaluará que los recorridos previamente creados sean asignados a los barcos según el berth al cual fueron asignados, esto quiere decir que cada barco haga su recorrido correcto al llegar al berth y al retirarse.

Por último se evaluará la comunicación del Sistema multiagente con el simulador, verificando que le llegue toda la información necesaria de los eventos que ocurren en el sistema multiagente. El correcto funcionamiento de este punto, depende principalmente del agente Manager 3D, el cual es el encargado de la comunicación entre el sistema y la interfaz 3D.

- Captura de eventos: Se verifica que la información de los eventos llegue en forma correcta al simulador, como también en el momento adecuado. El funcionamiento se verá reflejado en las acciones de los barcos y los berth en la simulación.

5.3 Datos de Entrada

Los datos de entrada utilizados por el sistema para estas pruebas son dos archivos de texto. Uno corresponde a los datos de los barcos, y el otro a los datos de los berth. Se utilizaron 30 barcos y 5 berth específicamente.

Por ejemplo en el archivo de texto de los barcos, la primera columna se encuentra un identificador general de barco (V), en la segunda columna un nombre para el barco, en la tercera el tiempo de llegada, en la cuarta el tiempo que demorará el barco en ser atendido y la última columna, la cual dependiendo si es un 1, el tiempo de llegada es el tiempo más temprano que puede llegar el barco; o si es un 0, el tiempo de llegada es el más tardío que puede llegar el barco.

Capítulo 6: Implementación

En este capítulo se explicará cómo se implementa lo diseñado en el capítulo 4, ya sea la comunicación del sistema multiagente (SMA) con el simulador 3D, el proceso de inicialización de la interfaz 3D, como actúan los distintos modelos al momento de la simulación, entre otras.

6.1 Comunicación del SMA con el Simulador 3D

Como se explicó en el capítulo 4, para que exista comunicación entre el SMA con el simulador 3D es necesaria la participación del agente Manager3D, el cual se encarga de informar al simulador lo que ocurre en el SMA.

La forma en que el SMA se comunica con el Manager3D es a través de un método llamado *informarActualizacion3D()*, el cual es el encargado de enviar el mensaje al agente Manager3D, este agente dependiendo del contenido del mensaje llamará al método correspondiente de la clase Interfaz3D quien es la encargada de representar la información en el simulador.

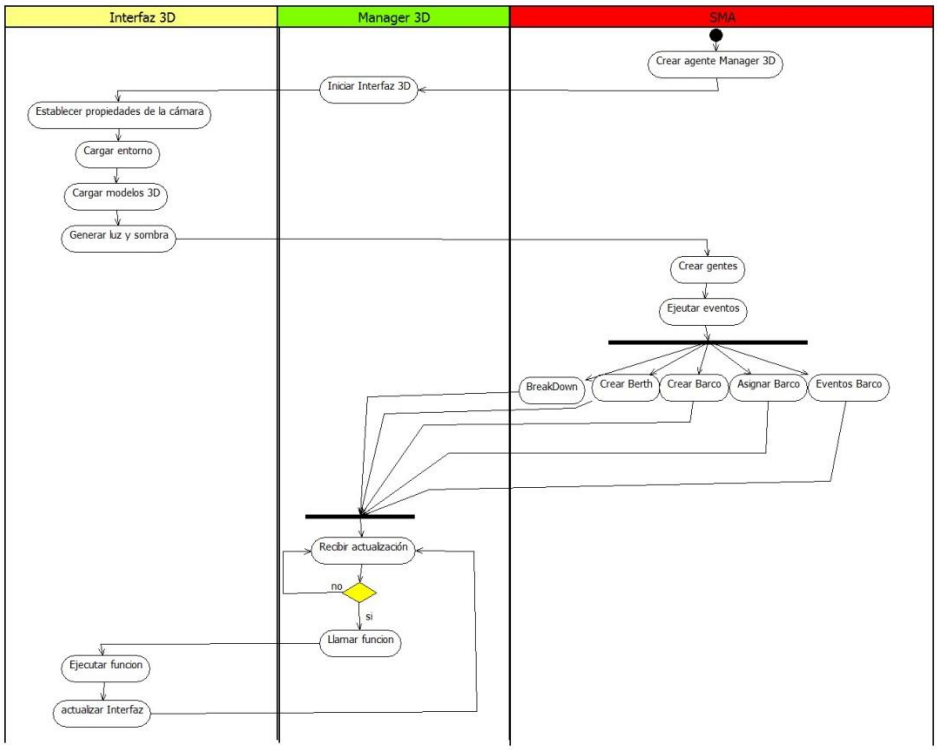


Figura 6.1 Diagrama de actividad ciclo de la interfaz 3D

La figura 6.1 describe el ciclo infinito que realiza la interfaz 3D con ayuda del Manager 3D. Se comienza por la creación de los agentes en el SMA. Al crear El agente Manager 3D, este inmediatamente crea la interfaz inicializando el escenario 3D y los modelos. Luego los distintos agentes creados (BerthEvent, VesselEvent, TripGen, Schedgen) en algún momento de la ejecución del programa ejecutarán sus respectivas funciones, enviando la información al

Agente Manager 3D, este recibirá el mensaje y llamará a la función correspondiente de la interfaz 3D, la cual se actualizará para representar lo realizado en el SMA. El agente Manager 3D quedará esperando en un ciclo infinito la llegada de un mensaje enviado desde el SMA, al momento de que llega se lo informa a la interfaz tridimensional. Esto permite que la interfaz 3D esté en una constante actualización.

6.2 Desarrollo e Integración de Modelos

En este apartado se describirán los distintos modelos utilizados en el mundo tridimensional. Qué técnicas se utilizaron para crearlos, como también en qué momentos son integrados en la interfaz 3D.

El ambiente en que se sitúa la simulación, es en el puerto. Gran parte de lo que se muestra en la simulación es el mar, donde comienzan a llegar los barcos. Estos llegan al punto de atraque al cual fueron asignados por el SMA.

Para la fabricación de los berths y los barcos se utilizaron las técnicas de modelado poligonal y modelado de bordes, descritas en el capítulo 2.

Sobre la integración de los modelos a la simulación, se comienza con la inicialización de la interfaz3D al momento que se crea el agente Manager3D. Al iniciar la interfaz 3D cargan los modelos relacionados con el ambiente, los cuales son el puerto, el mar y el entorno (cielo). A medida de que el SMA le envía información a la interfaz 3D por medio del agente Manager3D, se van incorporando los berths y también los barcos, pero estos últimos aun no aparecerán en la interfaz, solo lo harán en el momento en que llega su hora para ser atendidos por un berth.

Por otro lado también se han utilizado técnicas respecto al motor gráfico, como la utilización de Shadows maps para la creación de las sombras, el uso de luces difusas para la creación de la luz ambiental, y luces direccionales para la luz solar.

6.3 Visualización del Escenario Tridimensional

La simulación comienza con la cámara enfocando de tal modo que se pueden ver todos los berths, pero el simulador le da al usuario la capacidad de poder mover la cámara a su gusto, como también poder acercar o alejar la imagen, siempre con un límite. De esta manera, el usuario puede apreciar en mayor detalle los distintos modelos, si así lo desea.

6.4 Sincronización de los Tiempos de Llegada

Cuando el SMA termine de realizar las asignaciones de las diferentes solicitudes a los respectivos berths, se comienza con la simulación en tiempo real, la cual consiste que al llegar el tiempo cumplido para cada barco, estos lleguen al puerto y reciban el servicio dado por el berth. Para que esto ocurra debe existir una sincronización entre el SMA y el simulador3D, de manera que el barco llegue al berth justo a la hora indicada por el SMA y no antes ni después, para que se cumplan los tiempos de servicio correctamente.

Para que ocurra lo anterior, existe una variable llamada delta, la cual es el tiempo en que tarda el barco en posicionarse en el berth. Esta variable es sumada al tiempo de servicio del barco. La variable representa 5 segundos, este será el tiempo que se demore el barco en llegar

a su destino. En la Imagen 6.4 en la cuarta línea de código se encuentra esta variable llamada *"tiempo"* en la interfaz 3D.

6.5 Recorrido de los barcos

Como se explicó anteriormente cada barco es asignado a un solo berth, y para que en la simulación el barco llegue al berth al cual se le asigno, previamente se debe construir una ruta. Hay una ruta para cada berth, por lo cual dependiendo a cual fue asignado el barco, se le asigna dicha ruta de este berth. La ruta consiste en la llegada y retirada del barco del berth.

También se creó otra ruta alternativa, la cual solo es usada en casos de que el barco esté en un berth y tenga que posicionarse en otro. Para esto al barco se le asigna un vector de rutas para cada berth, y luego utiliza la ruta correspondiente al berth que debe llegar. En la figura 6.2 se muestra el recorrido que realizan los barcos, y el recorrido alternativo en caso de BreakDown.

Figura 6.2 Recorrido de los barcos

Para la realización de las rutas, se crea un arreglo, donde en cada casilla del arreglo se encuentra la ruta para cada berth. Al momento de que el barco debe aparecer en la escena y moverse hacia el berth, se le asigna dicha ruta al berth que le corresponda como destino. La figura 6.3 muestra el código que se utiliza para crear la ruta y cómo se van agregando cada punto de llegada a la ruta. La figura 6.4 muestra el momento en el cual se llama a la ruta para que el barco proceda a realizarla.

```
arr3 = new MotionPath[5];
for(int i=0;i<5;i++){
   arr3[i] = new MotionPath();
   arr3[i].setPathSplineType(SplineType.Linear);
   arr3[i].addListener(new MotionPathListener(){
        public void onWayPointReach(MotionEvent control, int wayPointIndex){
                //que se hace si llega al punto final
                if(control.getPath().getNbWayPoints() == wayPointIndex + 1){

arr3[0].addWayPoint(new Vector3f(actualPos.x + desfaceB,actualPos.y,actualPos.z));
arr3[0].addWayPoint(new Vector3f(actualPos.x + desfaceB,actualPos.y,actualPos.z-desfaceZ));
arr3[0].addWayPoint(new Vector3f(30f + desfaceB, actualPos.y,actualPos.z-desfaceZ));
arr3[0].addWayPoint(new Vector3f(30f + desfaceB, actualPos.y,240f));

distActual = 30 + distanciaBerth;

arr3[1].addWayPoint(new Vector3f(actualPos.x + desfaceB,actualPos.y,actualPos.z));
arr3[1].addWayPoint(new Vector3f(actualPos.x + desfaceB,actualPos.y,actualPos.z-desfaceZ));
arr3[1].addWayPoint(new Vector3f(distActual + desfaceB, actualPos.y,actualPos.z-desfaceZ));
arr3[1].addWayPoint(new Vector3f(distActual + desfaceB, actualPos.y,240f));
```

Figura 6.3 Creación de la ruta

```
MotionEvent eventoLlegada = new MotionEvent(berths[j].getBarco(i).getNodoSpatial(),arr1[j]);
eventoLlegada.setDirectionType(MotionEvent.Direction.PathAndRotation);
eventoLlegada.setRotation(new Quaternion().fromAngleNormalAxis(FastMath.DEG_TO_RAD, Vector3f.UNIT_Y));
eventoLlegada.setInitialDuration(tiempo);
//eventoLlegada.setSpeed(15f);
eventoLlegada.play();
```

Figura 6.4 Llamada a la ruta

6.6 Interfaz Gráfica de Usuario de la Interfaz 3D

Para que el usuario pueda adquirir más información a partir del simulador 3D, se integraron ventanas en la simulación, las cuales contienen información, ya sea de los berth como de los barcos. Estas ventanas aparecen al momento de hacer clic derecho a un berth o barco. La información que se muestra de un berth es el ID, el nombre, el estado (activo o inactivo); y los nombres de los barcos que va a recibir, el ultimo barco en servicio y los barcos listos. Por parte de los barcos, la información de su ID, su nombre, su estado y el berth destino.

Para un mayor entendimiento del usuario se incorporó más información respecto a los barcos de cada berth. La información completa del barco en un berth consta de su hora aproximada de llegada, su nombre, y una etiqueta que representa si el barco fue asignado en tiempo de simulación *(nuevo/nuevo tiempo/reasignado)*, si canceló su solicitud *(cancelado)*, si tuvo un retraso *(cambió hora)* o si no pudo ser atendido por un berth *(X).* Como se indica en la figura 6.5 donde un barco nuevo es asignado al berth.

Figura 6.5 Datos del Berth y Barco

6.7 Estado Ocupado o Libre del Berth

Para saber si un berth está atendiendo a un barco se incorporó a cada berth un color, el cual sí, es verde, el berth está libre para recibir un barco; si es rojo el berth está ocupado atendiendo a un barco. Esto se realizó creando una geometría (Cubo) en el motor gráfico jMonkey para cada berth, y dos materiales, uno rojo y uno verde. Al momento del berth estar libre u ocupado se asignaba el material a la geometría, ya sea verde o rojo respectivamente. En la figura 6.6 se muestra el código que crea los materiales y la figura geometrica, para luego añadir el material a la figura, en este caso por defecto comienza con el color verde.

```
60          mat1 = new Material(assetManager,"Common/MatDefs/Misc/Unshaded.j3md");
61          mat1.setColor("Color", ColorRGBA.Green);
62
63          mat2 = new Material(assetManager,"Common/MatDefs/Misc/Unshaded.j3md");
64          mat2.setColor("Color", ColorRGBA.Red);
65
66
67          //cargar etados del berth
68          Box box2 = new Box( Vector3f.ZERO, 3,3,3);
69          Geometry semaforo = new Geometry("Box", box2);
```

Figura 6.6 Creación y carga de materiales

6.8 Berth Inhabilitado

Los berths pueden quedar deshabilitados ya sea porque el SMA lo indica o también si el usuario lo desea, dando clic derecho al berth y presionando la opción *"Inducir BreakDown"*. La interfaz 3D representa esta situación con una boya impidiendo el paso de los barcos al berth. En la figura 6.7 se muestra como la interfaz 3D representa cuando un berth esta deshabilitado.

Lo anteriormente mencionado se realiza comunicando la acción en la interfaz 3D al agente manager3D, el cual le informa al SMA que debe inhabilitar el berth indicado, para esto se manda un mensaje con el nombre del berth.

El usuario puede deshabilitar un berth en cualquier momento que desee, por lo que si el barco está llegando al berth y este se inhabilita, el barco de igual manera llegará al berth, pero este no será atendido.

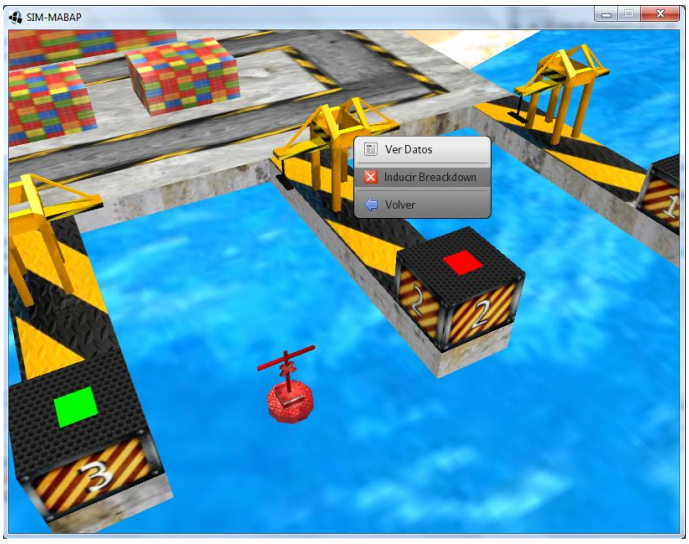


Figura 6.7 Berth deshabilitado

6.9 Movimiento de la Cámara

La cámara en la simulación se puede mover libremente como lo desee el usuario, pero existe un límite, ya que la cámara no puede atravesar por ejemplo el mar, o seguir infinitamente en dirección a un punto. Esto se realizó creando condiciones en el método *"simpleUpdate ( )",* donde la cámara no puede moverse más de una dicha ubicación. Por ejemplo en la figura 6.8 se muestra que la cámara no puede pasar más o menos del límite respecto al eje "y".

```
450     public void simpleUpdate(float tpf){
451         //determinar el limite de la camara
452         if(cam.getLocation().y<40){
453             cam.getLocation().y=40;
454             cam.update();
455         }
456
457         if(cam.getLocation().y>450){
458             cam.getLocation().y=450;
459             cam.update();
460         }
```

Figura 6.8 Limites de la cámara

También se agregó cuatro posiciones de la cámara, la cual el usuario puede acceder presionando las flechas del teclado, accediendo así a una vista aérea, a una vista desde el puerto, o por los costados, centrada siempre en el puerto con sus puntos de atraque y los barcos. En la figura 6.9 se representan las distintas formas de posición de la cámara. Esto se logra modificando directamente la ubicación de la cámara.

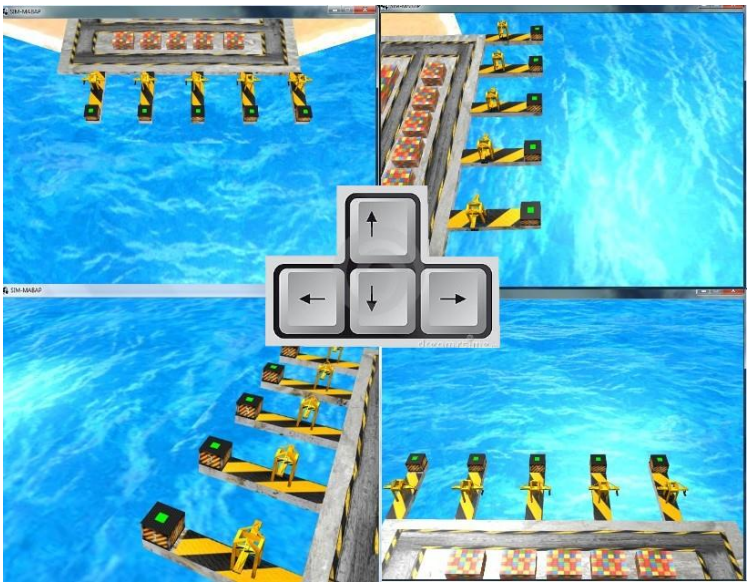


Figura 6.9 Posiciones de la cámara

6.10 Identificador de Barcos y Berths

Por último se quiso incorporar a la interfaz un identificador, donde arriba de cada barco y berth aparezca su nombre. Esto se realizó, ya que se le facilita al usuario identificar cada barco y cada berth, sin tener que hacer clic derecho a cada objeto para ver su nombre. También se agregó en la parte superior izquierda de la pantalla el nombre de cada berth y el último barco que atendió. En la figura 6.10 se muestra el identificador de los barcos y los berths en la interfaz 3D.

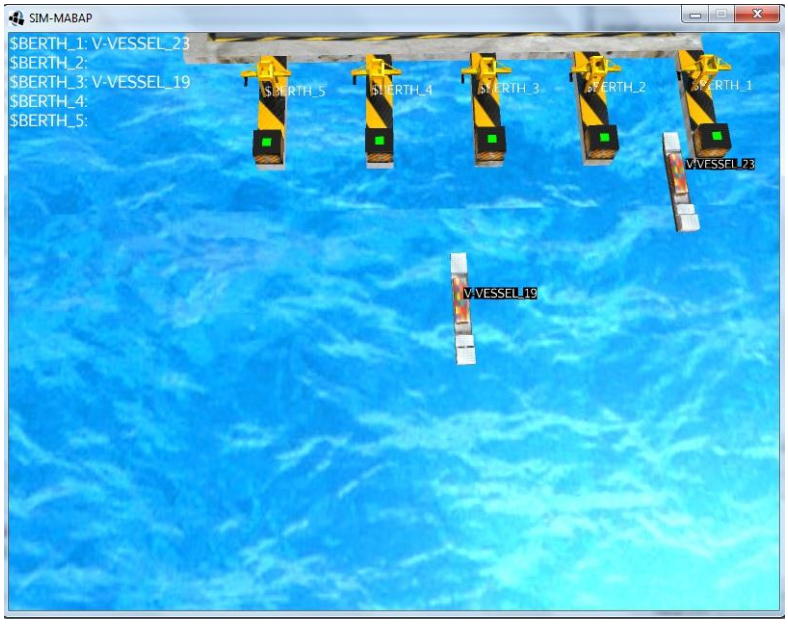


Figura 6.10 Identificador de Barcos y Berths

Para poder Incorporar texto en el mundo 3D se debió crear una variable de tipo BitMapFont, a la cual se le cargará la fuente que viene por default en JME. Luego se crea otra variable de tipo BitMapText, la cual contendrá el texto que se desea mostrar. En la figura 6.11 se muestra el código que se utilizó.

```
72          BitmapFont font = app.getAssetManager().loadFont("Interface/Fonts/Default.fnt");
73          BitmapText bmText = new BitmapText(font,false);
74          bmText.setSize(8);
75          bmText.setText(nombre);
76          bmText.setColor(ColorRGBA.White);
77          // se va alineando mientras se mueve la camara
78          BillboardControl bbControl = new BillboardControl();
79          bbControl.setAlignment(BillboardControl.Alignment.Screen);
80
81          textNode = new Node();
82          textNode.setLocalTranslation(30,-7,240);
83          textNode.setCullHint(Spatial.CullHint.Never);
84          textNode.attachChild(bmText);
85          textNode.addControl(bbControl);
86
87           this.app.getRootNode().attachChild(textNode);
```

Figura 6.11 creación de texto en la interfaz 3D

Capítulo 7: Futuro del Proyecto

Este capítulo se refiere al futuro del proyecto, es decir, que elementos se le puede integrar a futuro o qué cosas poder mejorar, ya que siempre un software necesitará de alguna mantención como también de mejoras o modificaciones en algunos ámbitos. A continuación se indican algunos puntos que abordaron en este periodo de proyecto, pudiéndolos realizar en un futuro.

- La cantidad de los berth como de los barcos en la interfaz 3D es estática, por lo que si fuera dinámica, se le daría al usuario la capacidad de que pueda elegir qué cantidad de berths y barcos participarán en la simulación antes de que comience.
- Agregar más animación a la simulación, como por ejemplo que las grúas de los berths se muevan, representando así que está atendiendo a un barco, siendo así la simulación más realista.
- Poder agregar a los barcos algún tipo de prioridad al momento de asignar, ya que el principal problema en que se centra el software es en la asignación de los barcos a los muelles de manera óptima. Así por ejemplo representar que un barco con mayor prioridad que otros se atiende en primer lugar, reprogramando así los que vienen a continuación.

Conclusiones

Al dar por terminada la memoria de título, se espera en este presente informe haber entregado toda la información necesaria de las ideas planteadas como también del problema y la solución que se llevó a cabo.

Se logró comprender en la etapa de proyecto 1, el problema en cuestión y la solución que se debía entregar, quedando así los objetivos claros. También quedaron claros los métodos de solución al problema de Berth Allocation Problem del proyecto relacionado [13], como también del código del SMA que le daba solución. En esta etapa también se llegó comprender las tecnologías que se utilizarían para realizar el proyecto, para así en la siguiente etapa solo aplicar los conocimientos.

En la fase de proyecto 2, se aprendió mucho más sobre las tecnologías utilizadas, ya que en este caso se estaban utilizando con mayor frecuencia, logrando perfeccionarse a medida que se iban creando los prototipos y mejorándolos cada cierto tiempo. También a principios de esta etapa se lograron encontrar más irregularidades del resultado del código del proyecto relacionado, corrigiendo así algunas, para lograr un mejor resultado al final del proceso.

Respecto a las tecnologías utilizadas, ya se tenía un conocimiento previo de la tecnología multiagente, por lo que no tuvo dificultad entender el SMA del proyecto. Por otro lado, el software de modelado 3D Blender, no es un programa difícil de usar, ya que se puede manipular fácilmente los modelos, aun así tiene muchas funciones y herramientas, pero cuenta con una variedad de tutoriales en la red.

JMonkey Engine gracias a que es 100% java no fue difícil de entender, aunque si se necesitó gran cantidad de tiempo en tutoriales y observando proyectos anteriores que lo utilizaron. Aun así se logró comprender y llevar a cabo el objetivo.

Como conclusión al final de este proceso se dio por logrado el objetivo al cual se quería llegar, ya que se cumplieron la mayoría de los requerimientos que demandaba. Con el resultado obtenido se facilita al usuario poder entender los resultados que arroja el sistema, ya que al representar la simulación de manera gráfica el usuario puede comprender y llegar a mejores conclusiones y decisiones. Aun así este proyecto puede ser retomado para seguir mejorando y agregarle nuevas características que sean necesarias.

Referencias

[1] JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. El Proceso Unificado de Desarrollo de Software. Pearson Addisson-Wesley. Año 2000, capítulo 1.1.

[2] http://www.utvm.edu.mx/OrganoInformativo/orgJul07/RUP.htm. Referenciado el 1 de abril del 2013.

[3] http://es.scribd.com/doc/59392631/Metodologias-de-Desarrollo-del-Software-word. Referenciado el 1 de abril, 2013

[4] Stuart Rusell, Peter Norvig, *"Artificial Intelligence: A Modern Approach"*, Prentice Hall, 1st Edition, ISBN 978-0131038059, 1995, pp 30-45.

[5] Wooldridge & Jennings (1995), Intelligent Agents: Theory and Practice, Queen Mary & Westfield College University of London, capítulo 1.1.

[6] Ana Mas, *"Agentes Software y Sistemas Multi-Agente: Conceptos, arquitecturas y aplicaciones"*, Prentice Hall, ISBN 84-205-4367-5, 2005, pp 10-160.

[7] Weiss (1999), Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, Massachusetts Institute of Technology, England, edited by Gerhard Weiss, pp 79-85.

[8] Michael Wooldridge (2002), An Introduction to MultiAgent Systems, Department of Computer Science, University of Liverpool, UK, Wiley, 2nd Edition, ISBN 978-0470519462, 1996, pp 20-101.

[9] Fundation for intelligent physical agents, "FIPA ACL Message structure presentation," 2002.

[10] Fabio Bellifemine, Agostino Poggi, Giovanni Rimassa, *"Developing Multi-agent Systems with JADE"*, Wiley Editorial, ISBN 978-0-470-05747-6, 2001, pp 30-32.

[11] Piermarco Burrafato, Massimo Cossentino, *"Designing a multi-agent solution for a bookstore with the PASSI methodology"*, Dipartimento di Ingegneria Informatica, Centro di studio sulle Reti di Elaboratori-Consiglio Nazionale delle Ricerche c/o CUC, CiteSeer Computer and Information Science Conference, In Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems AOIS, 2002, pp 3-17.

[12] Box Modeling, Página Web: http://wiki.cgsociety.org/index.php/Modeling_a_foot, 2006. Última vez visitada: 02/04/2013.

[13]http://www.test.blenderchevere.org.ve/2012/07/09/tecnicas-de-modelado-3d/ Referenciado el 8 de abril, 2013.

[14] NURB Modeling, Página Web: http://www.3dmax-tutorials.com/_2_Rail_Sweep_Surface.html, 2004. Referenciado el 9 de abril del 2013.

[15] http://es.wikipedia.org/wiki/Blender.Referenciado el 9 de abril del 2013.

[16] Marta Fernández Ruiz (2011) Modelado, texturizado y ajustes de malla, Universidad Carlos III de Madrid, España, Trabajo de investigación, pp 16-27.

[17] UV Map, Página Web: http://www.members.shaw.ca/nickyart/Comp/Camel.htm, 2006. Referenciado el 18 de abril del 2013.

[18] Richard S. Wright, Jr., Benjamin Lipchak, Nicholas Haemel, "*OpenGL SuperBible 4th Edition"*, Computer Graphics, ISBN 0-321-49882-8, 2007, pp 1-23.

[19] Richard Thomson, *"The Direct3D Graphics Pipeline"*, Guide to Learn Direct3D, 2006, pp 43-58.

[20] Mario Ezequiel, *"Uso de tecnologías de hardware gráfico en el apoyo al realismo en entornos virtuales arquitectónicos"*, Universidad de Colima, Colima, México, Tesis Posgrado Telemática, 2004, pp 36-39.

[21] Koroush Ghazi (2012) Tweak Guides - Vertical Synchronization. [Online]. http://www.tweakguides.com/Graphics_9.html

[22] Mario Ezequiel, *"Uso de tecnologías de hardware gráfico en el apoyo al realismo en entornos virtuales arquitectónicos"*, Universidad de Colima, Colima, México, Tesis Posgrado Telemática, 2004, pp 40-45.

[23] Timo Ahokas, *"Shadow Maps"*, Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory, HUT TML 2002, Tik-110.500 Seminar on Computer Graphics, 2002, pp 3-11.

[24] jMonkeyEngine, Página Web: http://jmonkeyengine.com/, 2012._Referenciado el 20 de abril, 2013

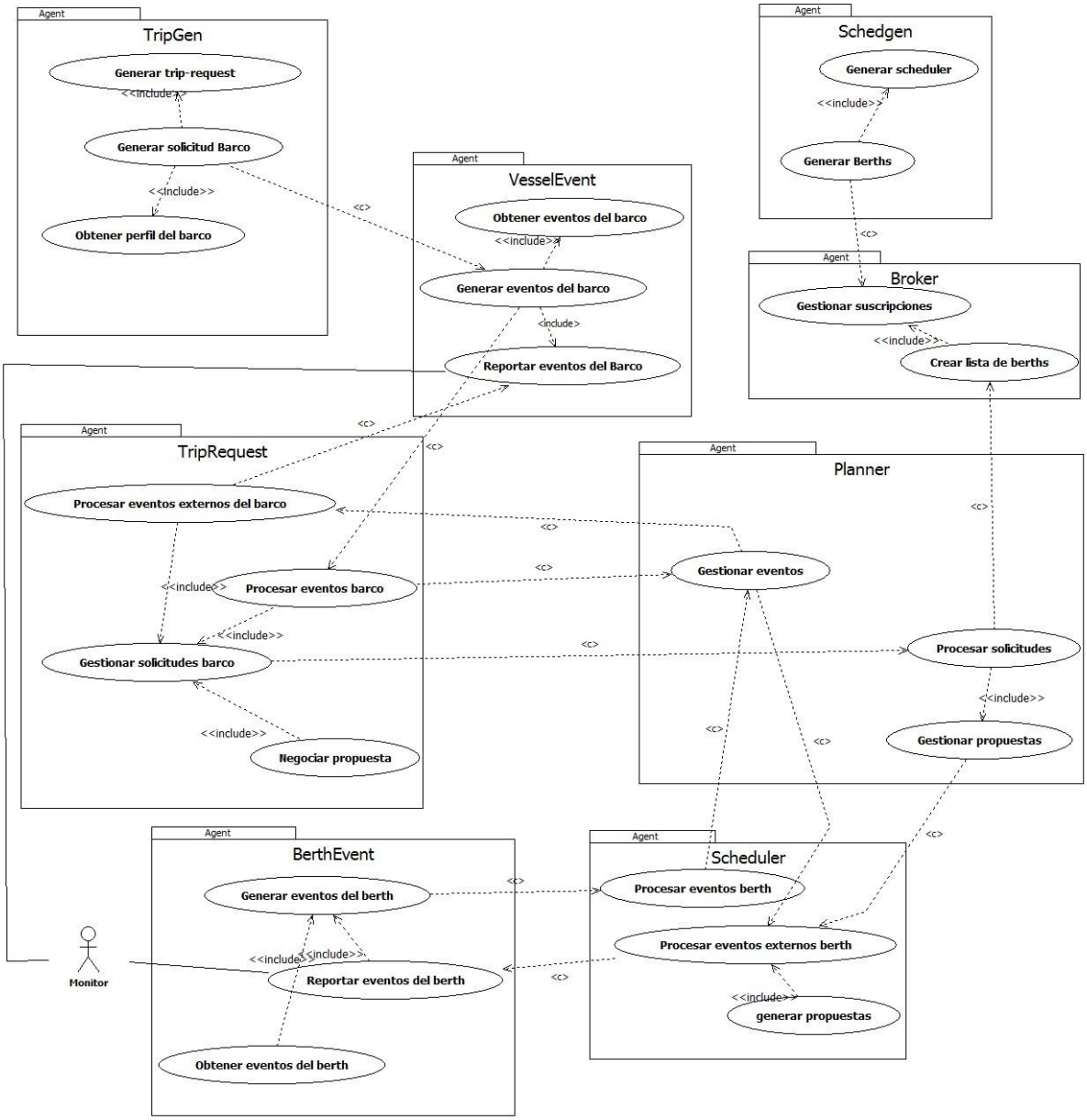
[25] Romina López e Ivonne Ramírez (2012), Berth Allocation problema con Sistemas Multiagente, Memoria de título, Pontificia Universidad Católica de Valparaíso, Chile.

[26] René Díaz (2007), Desarrollo de un sistema multiagente para el Berth Allocation Problem, Memoria de título, Pontificia Universidad Católica de Valparaíso, Chile.

[27]Gerardo Urbina (2013), Visualizador Tridimensional para un Sistema Multiagente aplicado al Problema de Transporte de Pasajeros, Memoria de título Pontificia Universidad Católica de Valparaíso, Chile.

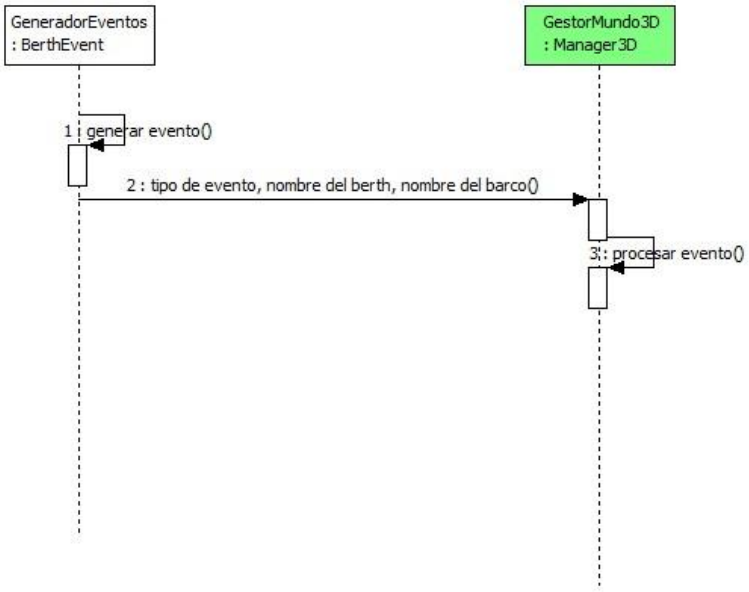
Anexos

A: Diagrama de Identificación de Agentes, López y Ramírez

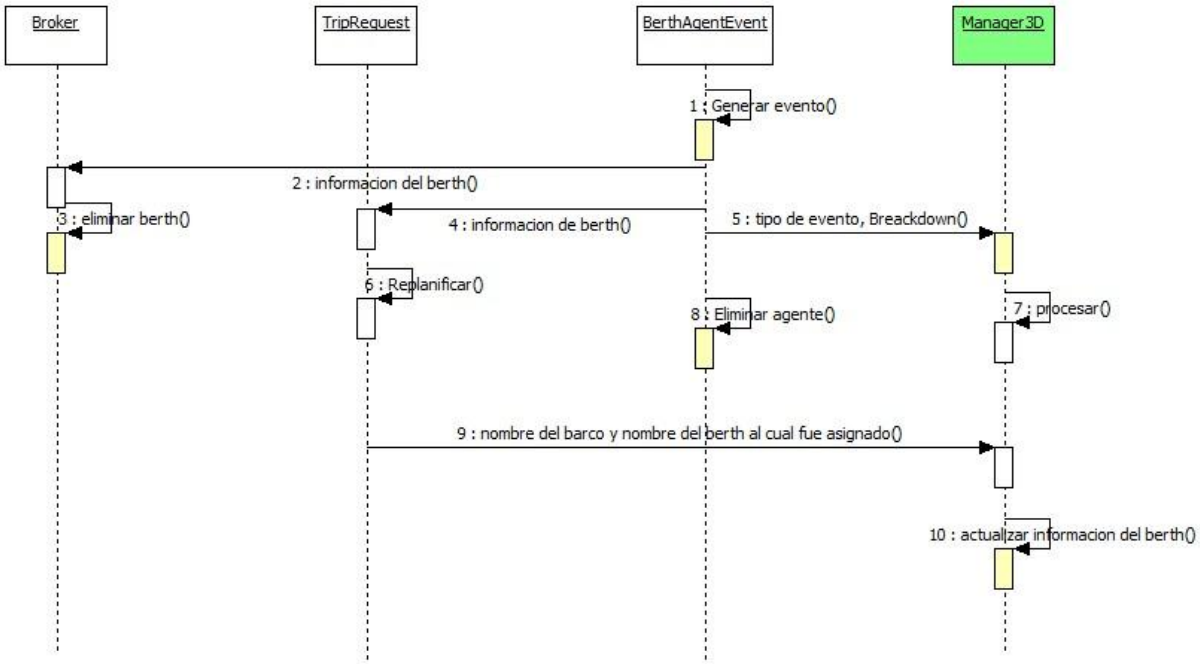


Anexo A.1 Diagrama de identificación de agentes

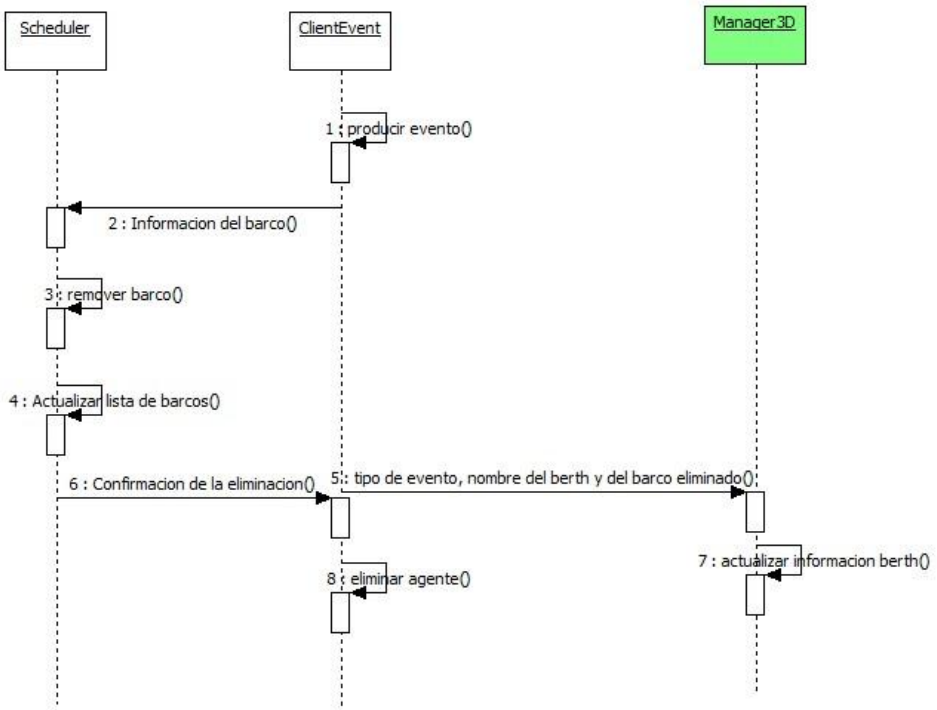
B: Modelos de requerimientos del sistema



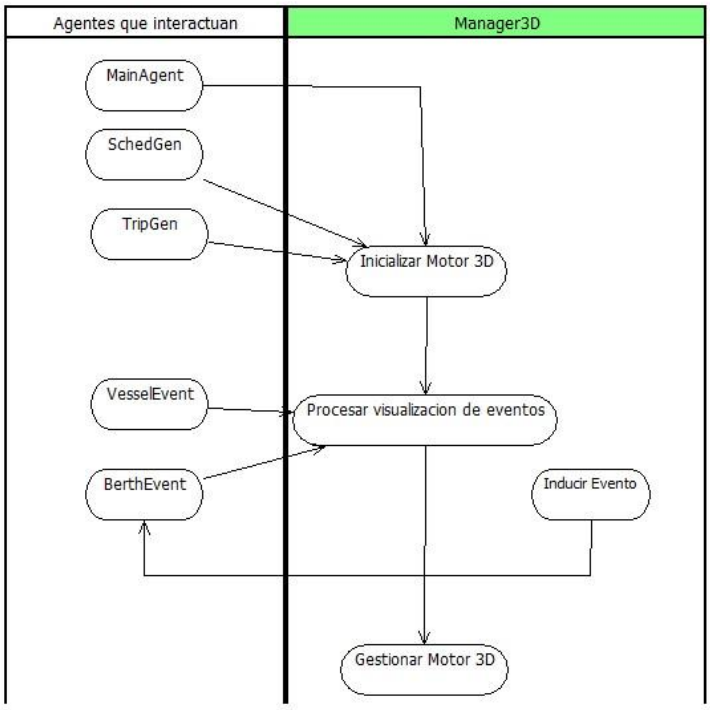
Anexo B.1 Identificación de roles: Generar eventos



Anexo B.2 Identificación de roles: Evento Breakdown

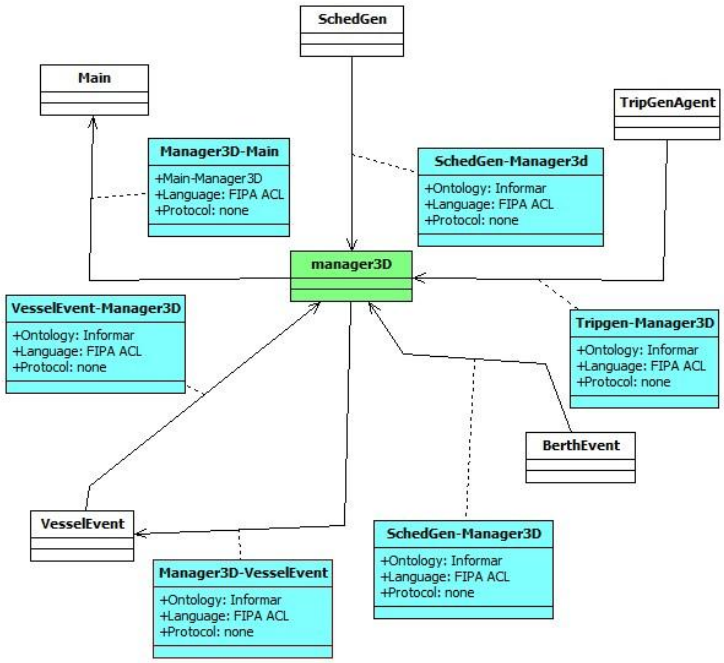


Anexo B.3 Identificación de roles: Evento cancelar solicitud

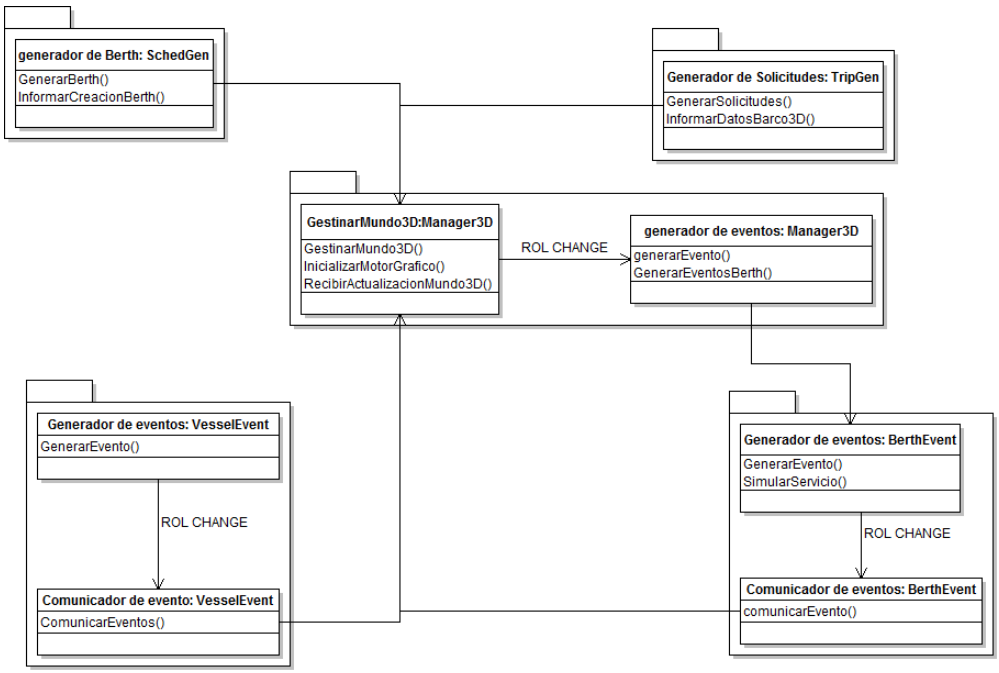


Anexo B.4 Diagrama de identificación de tareas: Agente Manager3D

C: Modelos de Sociedad de Agentes

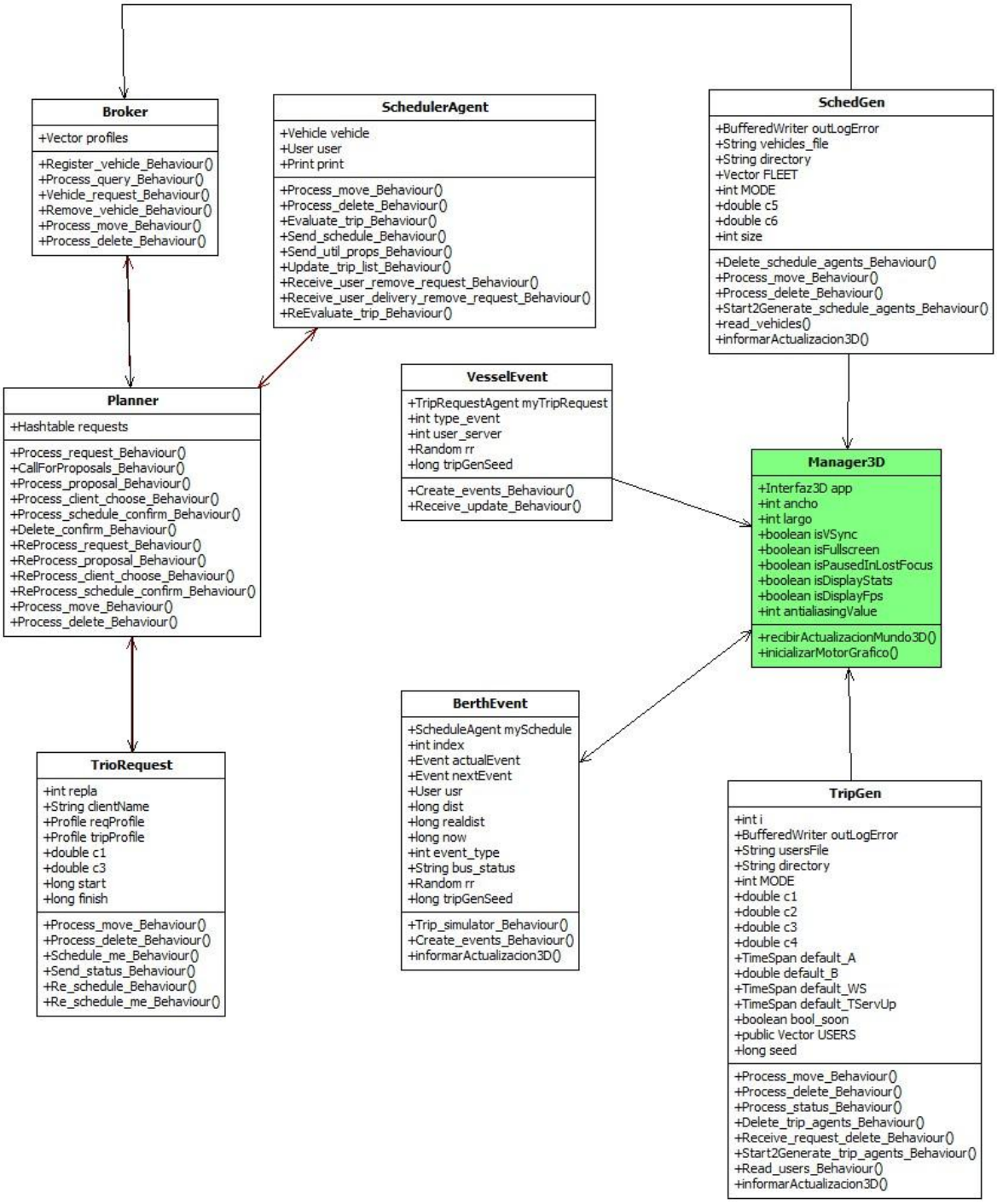


Anexo C.1. Diagrama de descripción de ontología de comunicación

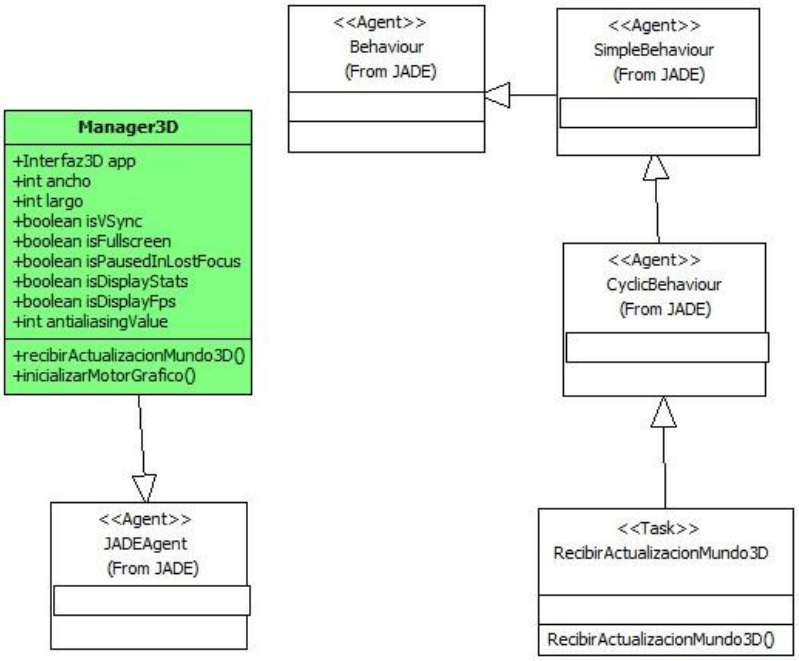


Anexo C.2 Diagrama de descripción de roles

D: Modelos de Implementación de Agentes

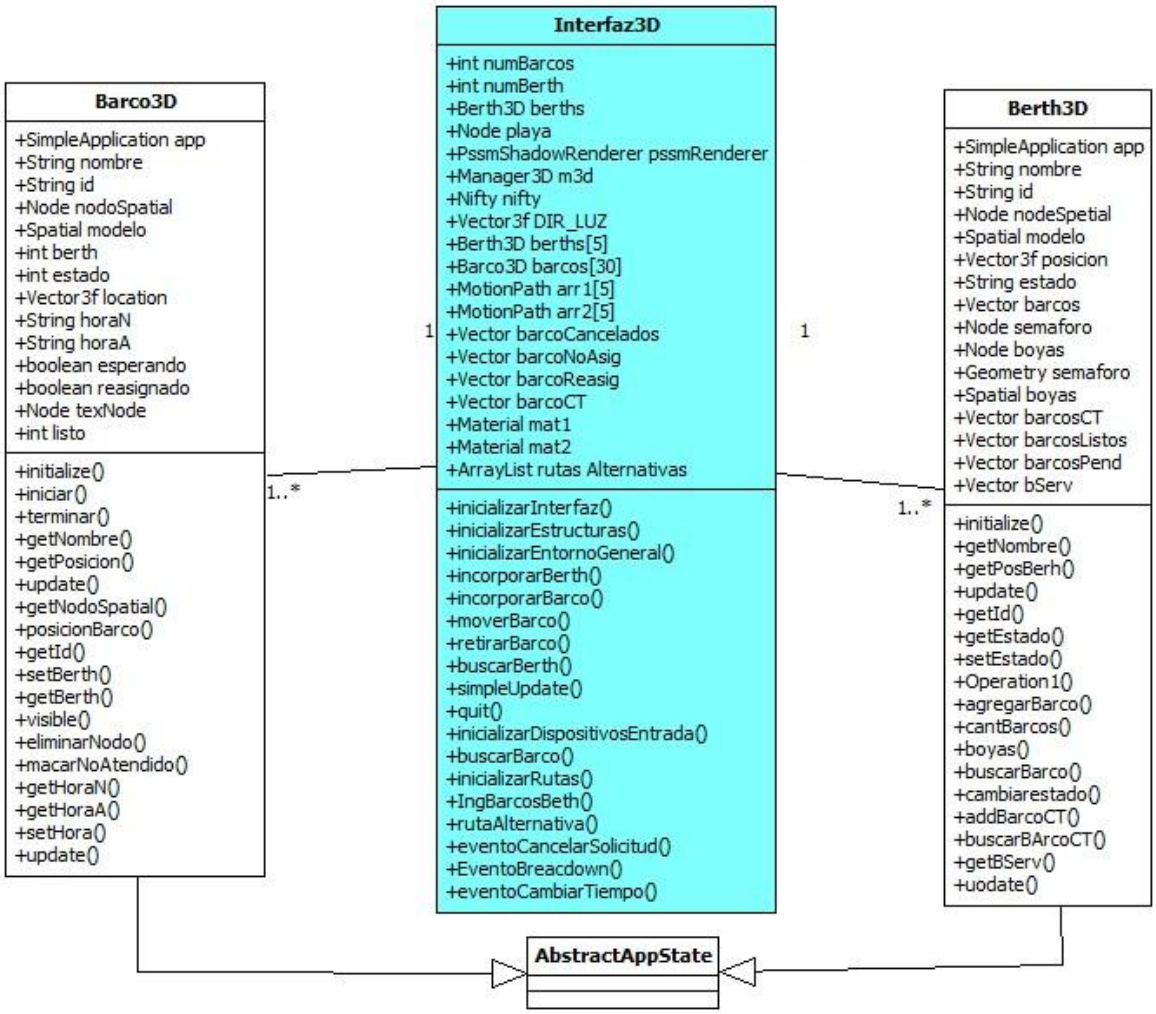


Anexo D.1 Diagrama de definición de la estructura multiagente



Anexo D.2 Diagrama de definición de la estructura multiagente individual

E: Diagrama de Clases



Anexo E.1 Diagrama de clases del motor gráfico 3D