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

SIMULADOR 3D PARA BERTH ALLOCATION PROBLEM, BASADO EN AGENTES.

EDUARDO JAVIER URQUETA ROJAS

Profesor Guía: **Claudio Cubillos Figueroa**
Profesor Co-referente: **Cristian Alexandru Rusu**

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

ABRIL 2013

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

SIMULADOR 3D PARA BERTH ALLOCATION PROBLEM, BASADO EN AGENTES.

EDUARDO JAVIER URQUETA ROJAS

Profesor Guía: **Claudio Cubillos Figueroa**
Profesor Co-referente: **Cristian Alexandru Rusu**

Carrera: Ingeniería Ejecución en Informática

ABRIL 2013

A mi padre que me apoyó y seguirá apoyando siempre y a mi madre que gracias a su esfuerzo y cariño volvió esto posible.

Índice

Resumen

El trabajo presentado a continuación corresponde a la realización de un simulador 3D para un sistema existente, que busca una solución al Berth Allocation Problem utilizando la tecnología multiagente. Por lo tanto se analizan distintas alternativas y aspectos fundamentales para el desarrollo de la solución. Berth Allocation Problem trata acerca del problema de la incapacidad de los puertos de crecer a la misma velocidad a la que aumenta la demanda de transporte de containers por barco, por lo que de los distintos problemas asociados, este se encarga de la manera más óptima de ordenar la llegada de los barcos y su ordenación en el puerto, para maximizar el transporte de los containers.

Es por ello que fue utilizado un entorno JAVA para la integración del sistema existente creado en JADE, el cual por medio de los agentes se comunica con el nuevo sistema de 3D desarrollado en JMonkey Engine, mostrando el ordenamiento de los barcos de acuerdo a la solución creada por los agentes, permitiendo obtener un mejor entendimiento del problema y su desarrollo.

Por medio de este sistema se puede apreciar como de manera más intuitiva que el sistema anterior, se realiza la solución a la problemática, así como las distintas variables que lo afectan y obtener información de ellos.

Palabras-claves: JAVA, JADE, JMonkey Engine, Berth Allocation Problem, Agentes, Simulador 3D.

Abstract

The work presented below regards the creation of a 3D simulator for an existing system, looking for a solution to the Berth Allocation Problem using multi-agent technology. So diverse alternatives and key issues are analyzed for the development of the solution. Berth Allocation Problem discusses about the problem of the ports inability to grow at the same speed of the increasing demand for container transportation by ship, so that from the various problems associated, this ensures the most optimal sequence for the arrival of the ships into the port, to maximize the transport of containers.

A JAVA environment was used for integrating the existing system created in JADE, through which agents communicate with the new 3D system developed in JMonkey Engine, showing the arrangement of the vessels according to the solution created by the agents, allowing a better understanding of the problem and its development.

By means of this system can be seen how in a more intuitive way than in the previous system, it performs the solution to the problem, as it affects the different variables and get information from them.

Keywords: JAVA, JADE, JMonkey Engine, Berth Allocation Problem, Agents, 3D Simulator.

Índice de ilustraciones

Índice de tablas

1 Introducción

Con la cada vez más creciente necesidad de importar y exportar productos, el uso del transporte marítimo sigue teniendo una mayor importancia en la movilización de productos por medio de contenedores, esta creciente demanda producida por la globalización ha generado problemas en puertos de todo el mundo, debido a que estos no pueden crecer en la misma medida. Es por ello que se buscan medidas que permitan optimizar de mejor manera cada uno de los recursos que se poseen, que son principalmente los espacios de atraques que poseen las distintas grúas y espacios de almacenaje. Este proyecto abarca principalmente el problema de atraque de los barcos (Berth allocation Problem), en donde se decide el orden en que los distintos barcos atracarán y a qué horas determinadas, para maximizar el uso de los distintos espacios que ofrece el puerto.

El presente trabajo se centra en la creación de un Simulador 3D que permitirá visualizar la solución del problema de atraque, que ayudará a los administradores de los puertos organizar de manera eficiente los distintos barcos que ingresan al puerto. Para la resolución del problema se usará un sistema existente basado en tecnología multiagente, que será modificado para aceptar los nuevos agentes que se encargarán del ambiente gráfico.

2 Definición de objetivos

2.1 Objetivo principal

- Desarrollar un Simulador 3D para visualizar la operación de un sistema multiagente para el Berth Allocation Problem (BAP).

2.2 Objetivos específicos

- Analizar el problema desde la perspectiva de un sistema multiagente.
- Comprender el funcionamiento de los entornos 3D.
- Desarrollar los diferentes modelos 3D necesarios para el Simulador 3D.
- Integrar el entorno 3D con el sistema multiagente.
- Validar el simulador 3D.
-

2.3 Metodología

Para la creación del plan de trabajo se analizarán distintas metodologías, con el fin de encontrar la manera más óptima de realizar el proyecto.

2.3.1 Cascada

Es un modelo de desarrollo de software creado por Royse en 1970. Este modelo es un derivado de otros procesos de ingeniería, por su filosofía también en es llamado ciclo de vida del software, se caracteriza por poseer cada una de sus fases muy diferenciadas y separadas, las cuales son: definición de requerimientos, diseño de sistemas y de software, implementación y prueba de unidades, integración y prueba del sistema, operación y mantenimiento. Los problemas asociados a esta metodología es que no son fácilmente aplicados a la práctica debido a que en la realidad las etapas interaccionan e intercambian informaciones, además de que el proceso no es un modelo lineal simple, ya que muchas etapas deben ser repetidas y es por este motivo que cada iteración se vuelve muy costosa. Las ventajas correspondientes a este modelo son la consistencia con los procesos de ingeniería tradicionales y que la documentación generada corresponde a cada fase terminada. Es recomendado usar este modelo solo cuando los requerimientos están muy claros y existe una baja posibilidad de que cambien con el tiempo.

2.3.2 Modelo Evolutivo (prototipos)

Este modelo contempla el desarrollo de una implementación inicial, exponiéndola a los comentarios del usuario y redefiniéndola a través de las diferentes versiones, donde luego las actividades de especificación, desarrollo y validación se llevan a cabo concurrentemente, y tiene retroalimentación rápida a lo largo del proceso. Esta metodología permite desarrollar inicialmente bajo especificaciones abstractas, que luego serán refinadas con el cliente. Los problemas asociados a esta metodología son su mayor costo en producir la gran cantidad de documentación necesaria y se suele producir una estructura deficiente del sistema. Sus ventajas son la mejor capacidad de captar requerimientos y posibilidades para el cliente de visualizar el software. Este sistema solo es recomendado para sistemas pequeños, en el caso de sistemas grandes, es recomendado un sistema mixto con la metodología cascada.

2.3.3 Proceso unificado de desarrollo de software (UP)

Es un proceso de desarrollo de software donde se asigna de manera disciplinada las tareas y responsabilidades en una empresa de desarrollo. Se centra en la producción y mantenimiento de modelos del sistema, y además es una guía de cómo usar UML de forma más efectiva. Incluye herramientas de apoyo como modelamiento visual, programación, pruebas, etc.

Utiliza un desarrollo iterativo, debido a que no es realista usar un modelo como la cascada, por lo que permite poseer una comprensión creciente de los requerimientos, que a su vez hace crecer el sistema. Esto permite abordar las tareas más riesgosas primero, por lo que permite reducir riesgos y poseer un subsistema ejecutable prontamente. Es por ello que son usados los casos de uso para captar los requerimientos, guiar el diseño, la implementación y las pruebas. Unas de las ventajas que posee esta metodología es que también es incremental, por lo que existe mayor tiempo para investigación del tema y lo cual permite tomar mejores decisiones que permiten llevar el proyecto de mejor manera.

En la Ilustración 2.1 se visualiza como cada una de las etapas: inicio, elaboración, construcción y transición se complementan a lo largo de toda la vida del proyecto, donde cada etapa posee un mayor énfasis según el proceso que se esté desarrollando.

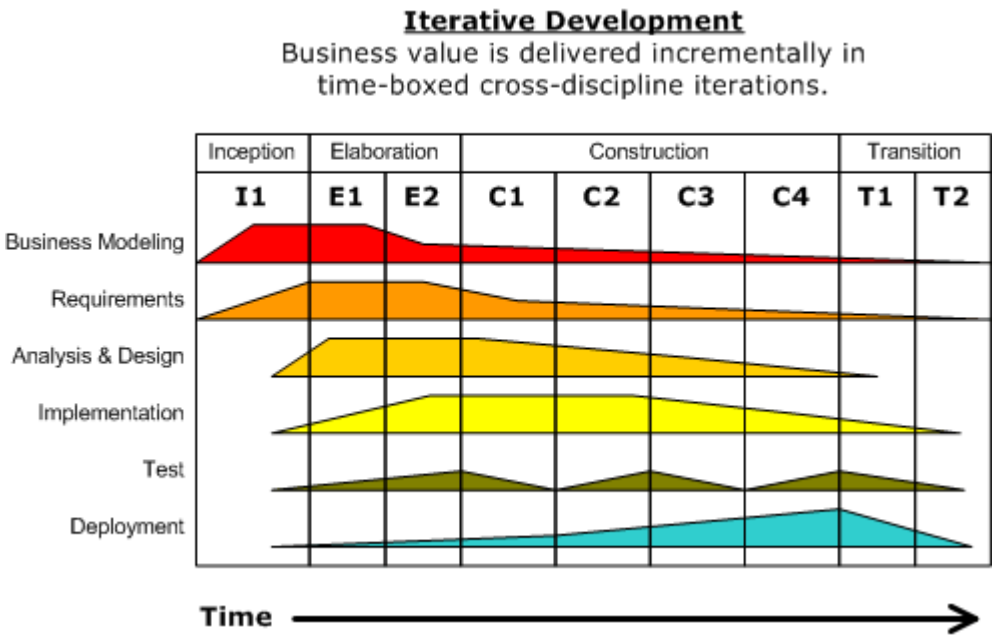


Ilustración 2-1 Ejemplo Desarrollo UP [1]

2.3.4 Modelo elegido

Debido a las claras ventajas del proceso unificado de desarrollo de software (UP), es el elegido en este proyecto, especialmente a la gran cantidad de investigación necesaria para lograr un buen resultado y su mayor flexibilidad para realizar cada una de las tareas.

2.4 Plan de trabajo

Para Proyecto es utilizado el diagrama propuesto en la Ilustración 2.2, por lo que se enfoca en las 4 fases correspondientes a construcción y transición del desarrollo UP.

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Inicio**	**35 días**	**lun 05-03-12**	**vie 20-04-12**
2	Investigar Tema	6 días	lun 05-03-12	lun 12-03-12
3	Establecer alcance del proyecto	3 días	mar 13-03-12	jue 15-03-12
4	Identificar Riesgos	2 días	vie 16-03-12	lun 19-03-12
5	Investigar sobre motor 3D a utilizar	8 días	mar 20-03-12	jue 29-03-12
6	Investigar sobre modelado 3D	6 días	vie 30-03-12	vie 06-04-12
7	Investigar sobre sistemas multiagentes	5 días	lun 09-04-12	vie 13-04-12
8	Encontrar casos de uso critico	4 días	lun 16-04-12	jue 19-04-12
9	Redactar Informe de avance	30 días	lun 12-03-12	vie 20-04-12
10	Entrega informe de avance	1 día	vie 20-04-12	vie 20-04-12
11	**Elaboración**	**46 días**	**lun 23-04-12**	**lun 25-06-12**
12	Investigar sobre BAP en sistemas multiage	7 días	lun 23-04-12	mar 01-05-12
13	Investigar integración BAP con sistema 3D	7 días	mié 02-05-12	jue 10-05-12
14	Desarrollar diagramas BAP con interfaz 3D	16 días	vie 11-05-12	vie 01-06-12
15	Crear Pruebas de validacipon de interfaz 3D	3 días	lun 04-06-12	mié 06-06-12
16	Elaborar primer prototipo	12 días	jue 07-06-12	vie 22-06-12
17	Redactar informe proyecto 1	44 días	lun 23-04-12	jue 21-06-12
18	Entregar informe proyecto 1	1 día	lun 25-06-12	lun 25-06-12
19	**Construcción**	**69 días**	**lun 30-07-12**	**jue 01-11-12**
20	Investigación de los Agentes existentes	7 días	lun 30-07-12	mar 07-08-12
21	Crear Nuevos Agentes	6 días	mié 08-08-12	mié 15-08-12
22	Crear Modelos 3D	6 días	jue 16-08-12	jue 23-08-12
23	Integración modelos 3D con Motor 3D	6 días	vie 24-08-12	vie 31-08-12
24	Integracion Motor 3D con Agentes	9 días	lun 03-09-12	jue 13-09-12
25	Creación prototipo	15 días	vie 14-09-12	jue 04-10-12
26	Corrección de errores	20 días	vie 05-10-12	jue 01-11-12
27	Realizar Pruebas	18 días	vie 05-10-12	mar 30-10-12
28	**Transición**	**23 días**	**mié 31-10-12**	**vie 30-11-12**
29	Mejoramiento de modelos 3D	5 días	mié 31-10-12	mar 06-11-12
30	Pruebas Finales	9 días	mié 07-11-12	lun 19-11-12
31	Entrega del sistema	1 día	vie 23-11-12	vie 23-11-12
32	Entrega Informe Final	1 día	vie 30-11-12	vie 30-11-12

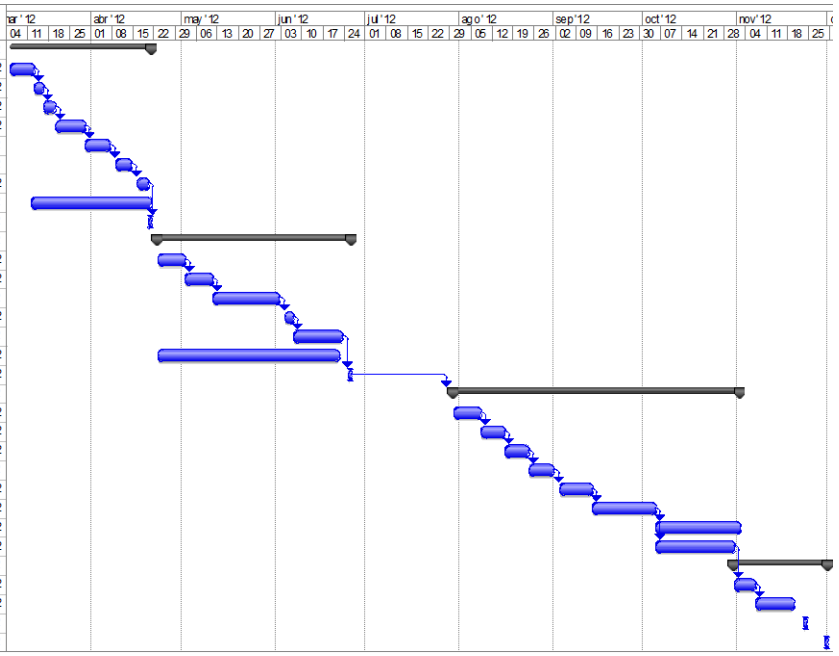


Ilustración 2-2 Plan de Trabajo

2.5 Estudio de Factibilidad

Todo proyecto siempre posee una serie de restricciones, esto se debe a que los recursos siempre son limitados, por lo tanto deben evaluarse siempre para tomar las decisiones más acertadas con el fin de tener un buen termino de proyecto.

2.5.1 Factibilidad Técnica

En este aspecto es necesario comprobar los distintos recursos físicos que se disponen y se necesitan.

Para este efecto se tiene a disposición un laboratorio el cual se puede ingresar todos los días a la semana, el cual cuenta con internet, sillas y mesas, además de un computador. Por lo que la búsqueda de papers e información relevante en internet, se puede realizar en este sitio. Además se posee un computador portátil personal que permite el manejo de objetos 3D que requerirá el proyecto actual.

Con respecto al software, aunque no es un recurso físico como tal, si es muy importante para el desarrollo. Para este apartado fue elegido Jmonkey Engine que es un motor 3D escrito en JAVA que permite manejar las librerías extras en JAVA que requieren los agentes. Para la creación de los modelos se utilizará la herramienta StarUML y para la creación de los distintos modelos 3D se utilizará Blender.

2.5.2 Factibilidad Económica

Debido a que el proyecto está orientado a la investigación y no a un uso empresarial, no existe un retorno de capital.

A continuación se mostrarán los costos asociados tanto al hardware como el software.

Software:

Tabla 2-1 Costos del Software

Herramienta	Costo ($)
StarUML	0
JMonkey Engine	0
JADE	0
Blender	0
Total	0

Hardware:

Tabla 2-2 Costos del Hardware

Herramienta	Costo ($)
Computador	400.000
Total	400.000

En el aspecto del software, debido a que se privilegió herramientas gratuitas no existe un real costo asociado. En el aspecto del Hardware si bien se utiliza actualmente un computador comprado para cumplir de manera óptima las necesidades del proyecto, perfectamente se podría utilizar el entregado por la Universidad para eliminar este único gasto.

2.5.3 Factibilidad Legal

Por el motivo de ahorrar en costos, se privilegió el uso de licencias libres en todos los software utilizados por lo cual no existirán inconvenientes de licencias, ni costos asociados a ella.

2.5.4 Factibilidad Operativa

En el aspecto del uso y creación del software, se utilizará JAVA, Jade y entornos 3D, en el cual ya existen un grado conocimientos en cada uno de estos aspectos, especialmente en el JAVA y Jade, por lo que debería existir un buen desarrollo con ellos. Además de existir múltiples fuentes de aprendizaje para un buen termino de proyecto.

2.6 Análisis de riesgo

Como todo proyecto, este proyecto no está libre de riegos, es por ello que a continuación se presenta la tabla 2-1 donde se visualiza los riesgos más importantes, con su probabilidad de ocurrencia e impacto que produciría, la tabla 2-2 corresponde a planes de mitigación para minimizar los riegos de ocurrencia de estos riesgos.

Tabla 2-3 Tabla de riesgos

Identificador	Descripción del riesgo	Probabilidad	Impacto
1	Incumplimiento de plazos	50%	Catastrófico
2	No disponibilidad de hardware y software	30%	Serio
3	Dificultad al crear los modelos 3D	70%	Serio
4	Integración Agentes y 3D muy compleja	70%	Serio
5	Problemas de funcionamiento del sistema BAP existente	70%	Serio

Tabla 2-4 Tabla de mitigación de riesgos

Medidas para mitigar riesgos	Riesgo asociado
Hacer evaluaciones continuas del estado de proyecto con su planificación.	1
Solicitud de equipos con anticipación y evaluación de alternativas.	2
Disminuir al mínimo aceptable los modelos 3D.	3
Investigar con anticipación integración de los sistemas.	4
Crear un sistema de asignación simple para BAP	5

2.7 Estructura del documento

En el capítulo 1 es introducido el tema.

En el capítulo 2 es visto los objetivos del proyecto, metodología, plan de trabajos, estudios de factibilidad y riesgos.

En el capítulo 3 es definido el tema y el estado del arte.

En el capítulo 4 es visto los conceptos necesarios para la creación de la interfaz 3D.

En el capítulo 5 es visto los sistemas multiagente, definiciones y metodologías.

En el capítulo 6 es mostrado los trabajos anteriores en el que está basado el sistema.

En el capítulo 7 se presenta la metodología multiagente a utilizar, la tecnología para la interfaz y los modelos desarrollados.

En el capítulo 8 se mostrara la implementación del sistema, con sus interfaces, y diagramas.

En el capítulo 9 se presenta los distintos tipos de pruebas que se le realizarán al sistema.

En el capítulo 10 se mencionaran las limitaciones del sistema

En el capítulo 11 se dará algunas pistas a aplicar en trabajos futuros.

En el capítulo 12 corresponde a las conclusiones del sistema.

En el capítulo 13 se presentan las referencias correspondientes.

3 Definición del problema

3.1 Berth Allocation problem

En los últimos años ha crecido la demanda de transportar una gran cantidad de productos entre los diversos países, en la cual en Chile se ha caracterizado por ser una de las principales potencias en esta materia en América Latina. En el año 2010 Movilizo 3.137.285 TEU (Twenty-foot Equivalent Unit equivalentes a 20 pies de largo x 8 pies de ancho x 8,5 pies de altura), otorgándole el segundo lugar en el ranking de transferencia de carga, perdiendo solo contra Brasil que movilizo 7.148.736 TEU**¡Error! No se encuentra el origen de la referencia.**. Los puertos compiten por ser puntos origen-destinos de las rutas empleadas para el transporte. Estos son importantes debido a que son considerados puntos de crecimiento económico. Es por ello que se analizan los principales puertos según su ubicación geográfica, estabilidad política y social y los costes de operaciones para ser elegidos como puntos de operaciones de las principales empresas.

Todo esto demuestra una necesidad cada vez mayor de administrar de manera más óptima el problema de atraque de los barcos en los muelles en donde la velocidad de crecimiento de los puertos es insuficiente.

En la organización del puerto en general se puede encontrar tres subsistemas, la gestión de la línea de atraque, el almacenamiento de los contenedores y el transporte. A medida que van llegando los buques, les es asignado un punto de atraque, donde descargan y/o cargan los contenedores por medio de grúas, luego los contenedores descargados son enviados a un patio de contenedores donde son enviados a sus destinos.

La orientación de este trabajo se concentra fundamentalmente en la primera parte que corresponde a la gestión de la línea de atraque, donde se administra a que barco se le asigna una sección de atraque y se busca obtener que las naves atraquen lo más rápido posible para asegurar una rápida rotación.

3.2 Estado del arte

Existen variados estudios realizados por distintos grupos para resolver la problemática de Berth Allocation Problem, pero principalmente se dividen en los siguientes enfoques:

Caso discreto: En este caso se considera el muelle como un número finito de puntos de atraque, en donde cada sección posee una longitud fija. Este método posee ciertas complicaciones ya que si se consideran segmentos muy pequeños se complica la búsqueda de una solución y si se segmenta muy grande resultaría en una subutilización de la secciones debido a la altamente variable tamaño de los buques.

Caso Continuo: En este caso se utiliza un diagrama espacio-tiempo, donde el eje horizontal representa el tiempo y el eje vertical la longitud del muelle. Se representa cada buque como un rectángulo cuya altura es su longitud y su longitud el tiempo estimado de proceso.

4 Simulador 3D

Para el desarrollo del Simulador 3D es necesaria la comprensión de la creación de modelos 3D y el uso de un motor 3D, que son dos temas distintos, pero muy relacionados. Inicialmente es necesario investigar la creación de los distintos modelos 3D que serán necesarios para la implementación de la interfaz y luego por medio del motor 3D se desarrolla el cómo interactúan entre ellos y con el usuario.

4.1 Modelado 3D

Para lograr Comprender los distintos tipos de modelado, es necesario tener conocimiento de ciertos conceptos **¡Error! No se encuentra el origen de la referencia.**.

4.1.1 Conceptos Matemáticos

- Sistema de Coordenadas: Es un conjunto de valores que permiten definir unívocamente la posición de cualquier punto de un espacio geométrico respecto de un punto denominado origen. En el modelado 3D corresponde hacer la diferencia entre el sistema de coordenadas universal frente al local.
- Vector: Es un segmento de recta dirigido en el espacio. Cada vector posee unas características que son: origen (punto donde actúa el vector), modulo (longitud del vector), dirección (orientación en el espacio de la recta que lo contiene, sentido (se indica mediante una punta de flecha situada en el extremo del vector).
- Segmento: es aquella parte de una línea recta que queda entre dos puntos señalados sobre ella.
- Polígono: es una figura plana y cerrada formada por tres o más segmentos de línea unidos en sus extremos
- Vector Normal: la normal de una superficie en un punto dado es el vector perpendicular a la superficie en ese punto.

4.1.2 Conceptos de Modelaje 3D

Las siguientes estructuras corresponden a las figuras habituales que traen por defecto los distintos softwares de modelaje 3D, en base a estas figuras mezclándolas y modificándolas se forman los modelos más complejos.

Primitivas:

- Plano
- Cubo

- Circulo
- Esfera
- Cilindro
- Cono

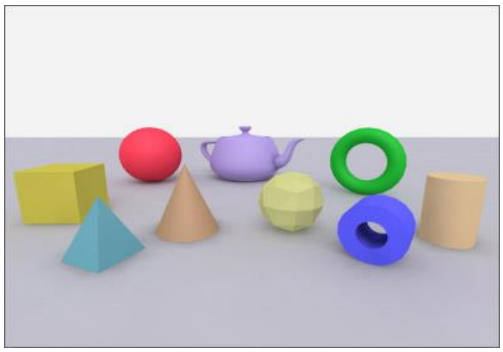


Ilustración 4-1 Ejemplo de Primitivas

Para la identificación, posicionamiento y toda la información necesaria de todos los modelos, luces y cámaras, para su renderización, es creado el archivo llamado escena.

Cada escena puede identificarse por medio del sistema de coordenadas en 3 dimensiones donde es llevada a cabo la renderización. Este habitualmente es llevado a cabo en el sistema de coordenadas universal o "world", pero al operar también con los objetos de la escena, cada uno posee su sistema de coordenadas local.

Al realizarse transformaciones a los objetos u otro tipo de operaciones, se hacen de acuerdo al sistema de de coordenadas seleccionado.

4.1.3 Tipos de modelado

A continuación se presentara los distintos métodos existentes para la creación distintos modelos necesarios **¡Error! No se encuentra el origen de la referencia.**.

4.1.3.1 Modelado Booleano

Trata de combinar las distintas primitivas tridimensionales existentes y utilizar las operaciones lógicas para formar el modelo, también es conocido como Constructive Solid Geometry.

El modelo está compuesto por una raíz que es el modelo resultante, donde sus hojas son las primitivas y los nodos internos los operadores lineales.

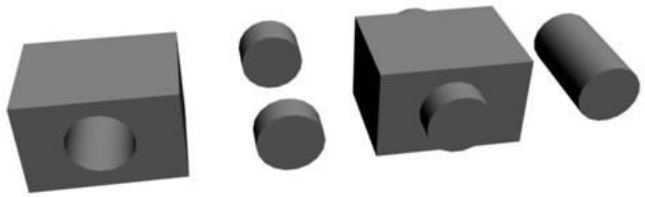


Ilustración 4-2 Ejemplo Modelado Booleano **¡Error! No se encuentra el origen de la referencia.**

4.1.3.2 Modelado de fronteras

Este tipo de modelado utiliza primitivas bidimensionales, por lo que los modelos quedan definidos como estructuras de datos., también es conocido como Boundary representation. Según las estructuras de datos, se puede considerar:

- Poligonal: En esta estructura de datos, la estructura de los datos utilizados almacena información sobre vértices, aristas, caras y relaciones topológicas. Debido a su simplicidad suele ser la representación más utilizada y única en muchos render. Debido a que el hardware está preparado, es un método muy rápido y la conversión de curva a polígono es fácil y rápida. En contra posee un almacenamiento muy costoso, por lo que se deben buscar mejores alternativas para las curvas.
- Superficies curvas: En esta estructura de datos la información guardada incluye puntos de control, puntos de tangente, etc. Un tipo de curva utilizada son las Spline, que son formadas mediante secciones polinómicas que satisfacen ciertas condiciones de continuidad en la frontera de cada intervalo. Este tipo de modelado la generación de estas superficies es mediante redes de parches de curvas, lo que permite suaves cambios en la malla, utilizando solamente puntos de control, por lo que finalmente el costo de almacenaje es muy bajo.

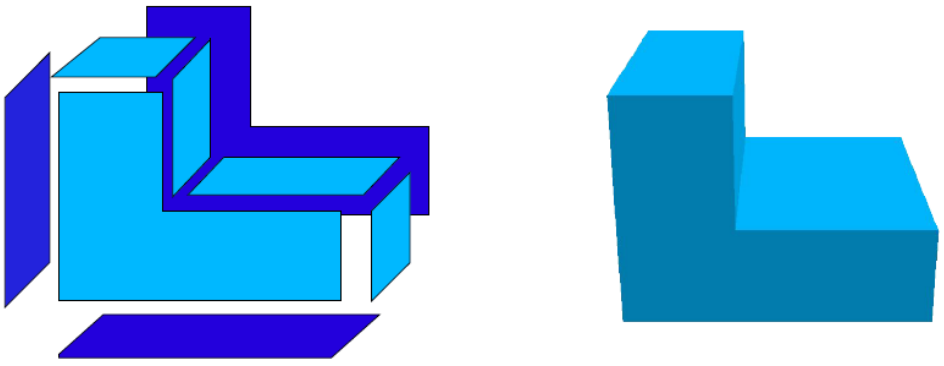


Ilustración 4-3 Ejemplo Modelado por fronteras**¡Error! No se encuentra el origen de la referencia.**

4.1.3.3 Superficies de subdivisión

Es una representación intermedia entre las superficies poligonales y las superficies formadas mediante curvas.

Toda superficie de subdivisión comienza con una superficie poligonal llamada red de control, la cual se subdivide en más polígonos, y así sucesivamente, empleando una aproximación adaptativa que añade únicamente más resolución en las zonas que lo requieren.

Este tipo de superficie no se define mediante una fórmula, lo hace de manera algorítmica y su topología es irregular.

La generación de superficies de subdivisión tiene principalmente dos fases:

- Refinamiento: partiendo de la red de control, se crean nuevos vértices que se reconectan para crear nuevos triángulos.
- Suavizado: calcula la posición de algunos (o todos) los vértices de la nueva malla. Las reglas de suavizado determinan diferentes esquemas y la superficie final, dónde el más conocido es el de Carmull-Clark.

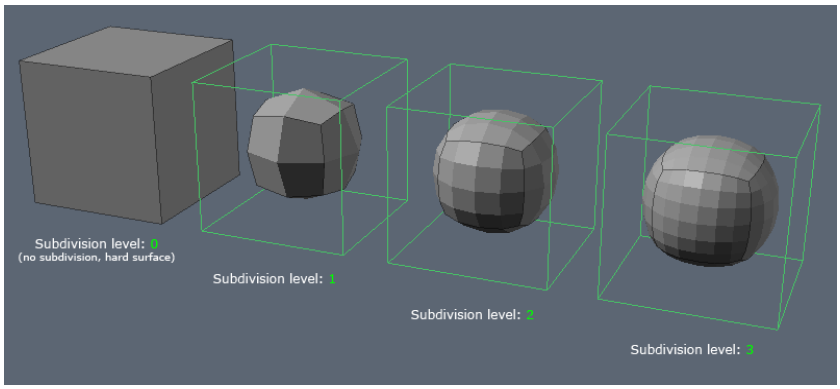


Ilustración 4-4 Ejemplo Modelado de superficies por subdivisión **¡Error! No se encuentra el origen de la referencia.**

4.1.3.4 Superficies implícitas (Meta superficies).

Una superficie implícita es aquella definida mediante funciones que se evalúan en un campo continuó **¡Error! No se encuentra el origen de la referencia.**.

En el caso concreto de meta superficies, cada elemento viene definido, al menos, con su posición, su fuerza y su radio de acción que es el radio que afecta los otros elementos.

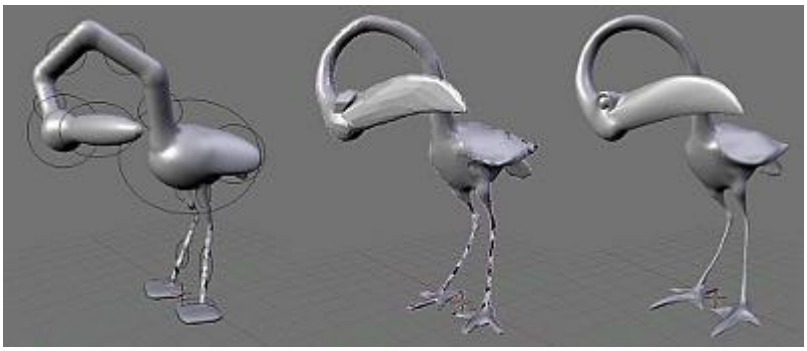


Ilustración 4-5 Ejemplo Modelado de superficies implícitas **¡Error! No se encuentra el origen de la referencia.**

4.1.3.5 Modelado procedural.

Este tipo de modelado es usado principalmente para procesos generados por la naturaleza, por lo que simula el crecimiento y se describe en forma de procedimientos.

- Geometría fractal: Permite la creación de figuras aleatorias parecidas a las de la naturaleza. Consiste en la división de polígonos de formas recursivas y aleatorias de muchas formas irregulares.
- Sistema de partículas: Se generan por medio de esferas y puntos tridimensionales, se le aplican atributos de crecimientos, lo que por medio de estos atributos, permiten dar lugar a las trayectorias de las partículas, que a través del paso del tiempo forman la figura tridimensional. Es usado principalmente para humo, fuego, etc.
- Modelado por simulación física: La forma de modelar materiales naturales cuya forma está en constante cambio es simularla. Es usado para fuego, humo, viento, nubes, etc.

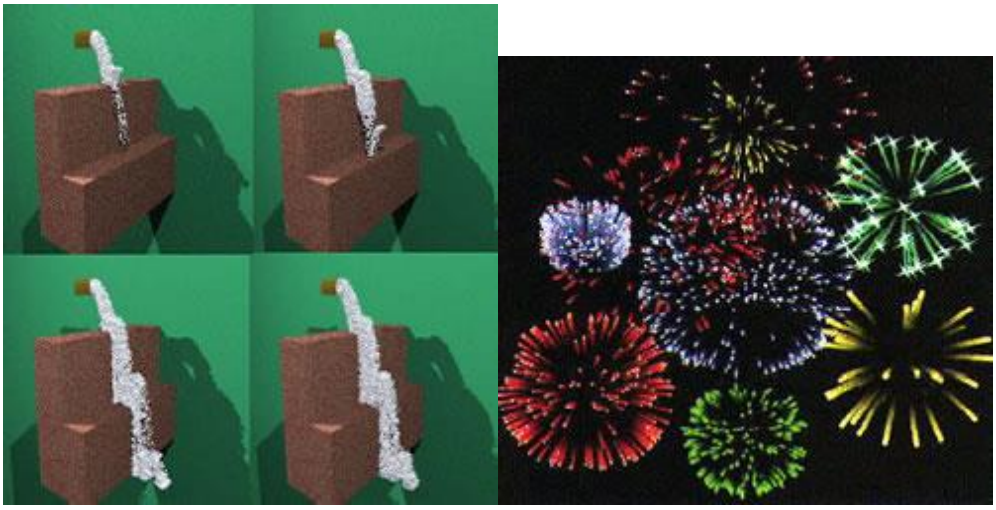


Ilustración 4-6 Ejemplo Modelado procedural **¡Error! No se encuentra el origen de la referencia.**

4.1.3.6 Modelado basado en trazos. "Sketch Modelling".

Este tipo de modelado consiste en formación de la figura por medio de trazos 2D hechos por el usuario, tanto para la generación del objeto, como para las modificaciones del mismo. Esto permite una representación homogénea, ya que el modelado se realiza de manera intuitiva.

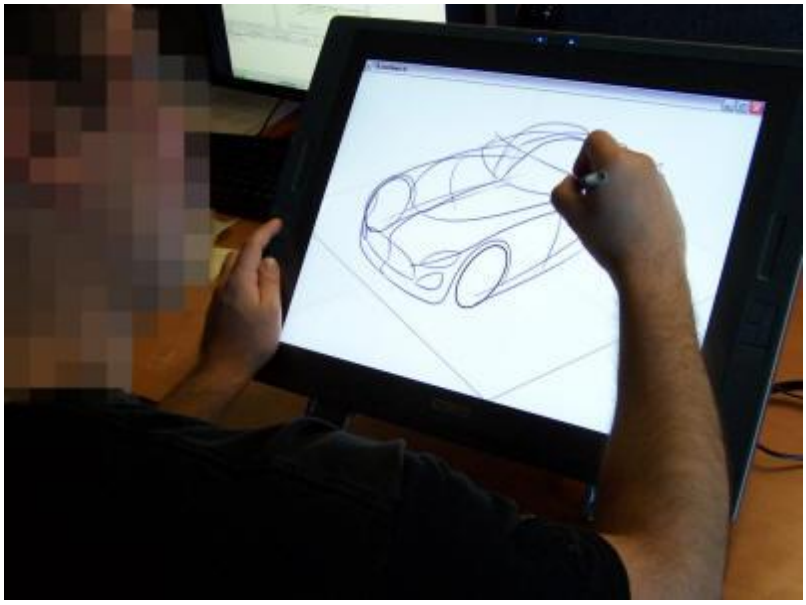


Ilustración 4-7 ILoveSketch Software de modelado a mano **¡Error! No se encuentra el origen de la referencia.**

4.1.4 Operadores.

A continuación se presentan los distintos tipos de operadores que se utilizan para la creación de modelos **¡Error! No se encuentra el origen de la referencia.**.

Los sistemas de modelado booleano se caracterizan por la generación de modelos mediante el uso de operadores booleanos, que permiten (al usuario) realizar una serie de combinaciones para obtener el modelo final.

Se puede definir tres operaciones booleanas básicas:

- Unión: Es la unión de dos o más figuras para formar una nueva figura más compleja.
- Intersección: Es el espacio en común que poseen 2 o más figuras, que forman una nueva figura.
- Diferencia: Al tomar una figura, luego de aplicarle otra figura, el resto es la nueva figura.

En sistemas de modelado por fronteras se puede aplicar una serie de operadores, además de los operadores booleanos citados anteriormente:

- Extrusión (Extrusion o lofting): Consiste en generar una superficie 3D extendiendo una forma 2D a lo largo de un eje.
- Barrido (Swepping): Es una generalización de la extrusión, ya que consiste en generar un objeto 3D extendiendo una forma a lo largo de un camino, si el camino es recto, estamos aplicando el operador extrusión.
- Revolución (Revolve o lathe): Es una variación de las superficies de barrido. Se desplaza una forma 2D alrededor de un eje.

- Skinning: Se genera una superfice curva que cubre secciones cerradas, también llamadas rebanadas, es similar al barrido, sólo que la curva puede cambiar su geometría mientras avanza por el camino.
- Mezclado (Blending): Es un método especial para combinar dos superficies, que no tiene por qué estar en contacto. La nueva superficie curva que se genera se puede controlar mediante puntos de control.
- Truncado (Beveling): Consiste en recortar un vértice situado entre dos superficies adyacentes reemplazándolo por un plano inclinado.
- Redondeado (Rounding): Es una versión del truncado que suaviza los vértices y aristas del objeto.
- Sculpter: Este tipo de operador ha tenido bastante relevancia en los últimos años ya que permite mediante trazos 2D, simulando un pincel o un dedo, modificar la malla 3D como si se estuviese esculpiendo en ella, simulando la arcilla.

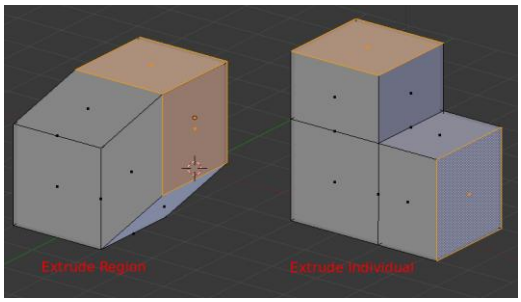


Ilustración 4-8 Ejemplo de extrusión

4.1.5 Herramientas de modelado 3D

Existen diferentes Herramientas de modelado 3D existentes en el mercado, pero se analizará específicamente la siguiente, por ser gratuita.

4.1.5.1 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.

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, y otros más. Incluye además un modo esculpir que funciona como una metáfora de pincel 3D.

También soporta distintos métodos de texturizados para los modelos **¡Error! No se encuentra el origen de la referencia.** y**¡Error! No se encuentra el origen de la referencia.**.

4.2 Motor 3D

Por medio de los motores 3D, son manipulados los distintos modelos 3D, lo que permiten crear un mundo muy detallado por medio de técnicas de programación que permiten mostrar efectos visuales espectaculares que permiten dar una sensación de realismo.

4.2.1 Conceptos de motores 3D

Para la creación de estos mundos, son utilizados los objetos, que utilizan distintas primitivas**¡Error! No se encuentra el origen de la referencia.**:

- Pixeles 3D: también llamados voxels
- Vectores: Indican una dirección
- Polígonos: triángulos, cuadriláteros
- Primitivas de volumen: esferas, conos, cilindros, etc.
- Vértices: Coordenadas 3D (x,y,z)

Las primitivas más utilizadas son los polígonos, que corresponden a vértices que componen un plano.

Los polígonos usados habitualmente son triángulos, debido a su simplicidad al ser representados por solo 3 vértices, además de su facilidad de representar otras figuras como los cuadrados, solamente uniendo 2 triángulos. Además los triángulos pueden compartir vértices, por lo que se pueden formar mallas de polígonos más complejas.

La GPU son procesadores especializados en operaciones sobre triángulos y operaciones de matrices y vectores.

Los vértices poseen la siguiente información:

- Posición 3D (coordenadas x,y,z)
- Un identificador, el cual indica a que triángulo pertenece.
- Coordenadas de textura
- Un vector normal (perpendicular al plano que forma el triángulo)
- Color e información de iluminación.
- Parámetros de Skining.

4.2.2 Técnicas Disponibles

Para una mejor visualización y realismo de los elementos mostrados por pantallas existen diversas técnicas y tecnologías que mejoran la visualización de los entornos 3D.

4.2.2.1 Antialiasing

Conceptualmente funciona renderizando la imagen a una resolución mayor a la deseada por ejemplo: X2 (4 subpixels por pixel), X4 (16 subpixels por pixel), etc.

Se utiliza para eliminar artefactos generado en rasterización de imágenes, esto es claramente identificable por el efecto "serrucho" en las imágenes**¡Error! No se encuentra el origen de la referencia.**.

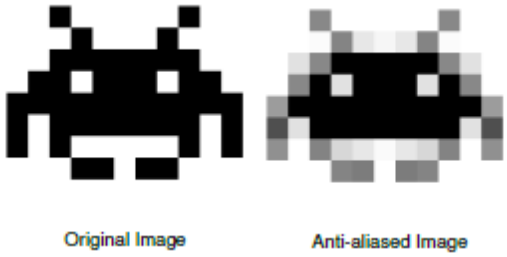


Ilustración 4-9 Ejemplo de Antialiasing **¡Error! No se encuentra el origen de la referencia.**.

4.2.2.2 VSync

VSync, también llamado sincronización vertical, fue requerido principalmente por limitaciones físicas de los monitores CRT relacionados con su tasa de refresco. Esto se debe a que en la mayoría de las ocasiones la cantidad de imágenes generadas por la tarjeta gráfica supera a la tasa de refresco de los monitores, por lo que por medio del VSync, la tarjeta gráfica produce la misma cantidad de imágenes que puede soportar el monitor. De no estar activado pueden producirse imágenes erróneas y movimientos bruscos **¡Error! No se encuentra el origen de la referencia.**.

Ilustración 4-10 Juego 3D sin VSync**¡Error! No se encuentra el origen de la referencia.**

4.2.2.3 Iluminación

Existen diversos tipos de iluminación que se pueden aplicar en el motor para lograr el mayor realismo posible. En la computación gráfica existen 2 tipos de luz: Luz de ambiente y luz directa**¡Error! No se encuentra el origen de la referencia.**.

La luz de ambiente es la luz que está presente en cada punto de la escena 3D y posee la misma intensidad y aparenta venir de todas las direcciones con la misma intensidad. Esto en la naturaleza, es el tipo de luz que se refleja en todos los objetos lo que crea la sensación de generar una luz propia.

La luz directa siempre viene de una fuente de luz el cual su posición es posible encontrar mirando donde la luz cae. Existen 3 subtipos de luz directa:

- Luz direccional: este tipo de luz viene desde una fuente de luz distante, donde la fuente no es importante, debido a su lejanía con el espectador. Es por ello que requiere muy poco cálculo para el renderizado ya que no posee cambios en el ángulo, intensidad o color. Un ejemplo claro de este tipo de luz en la naturaleza es el sol.
- Punto de luz: Este tipo de luz corresponde a un punto en el espacio que emite luz en todas las direcciones, por lo que para calcular la luz en un objeto es necesario saber la posición, color e intensidad. En una superficie cada punto recibe una intensidad distinta debido a que hay un cambio en el ángulo con respecto a la fuente de luz.
- Foco de luz: Es el tipo de luz más complejo debido a que requiere más cálculos. El foco de luz es tipo de luz similar al punto de luz, pero en vez de emitir luz en todas las direcciones, lo hace en un ángulo específico y en una dirección específica.

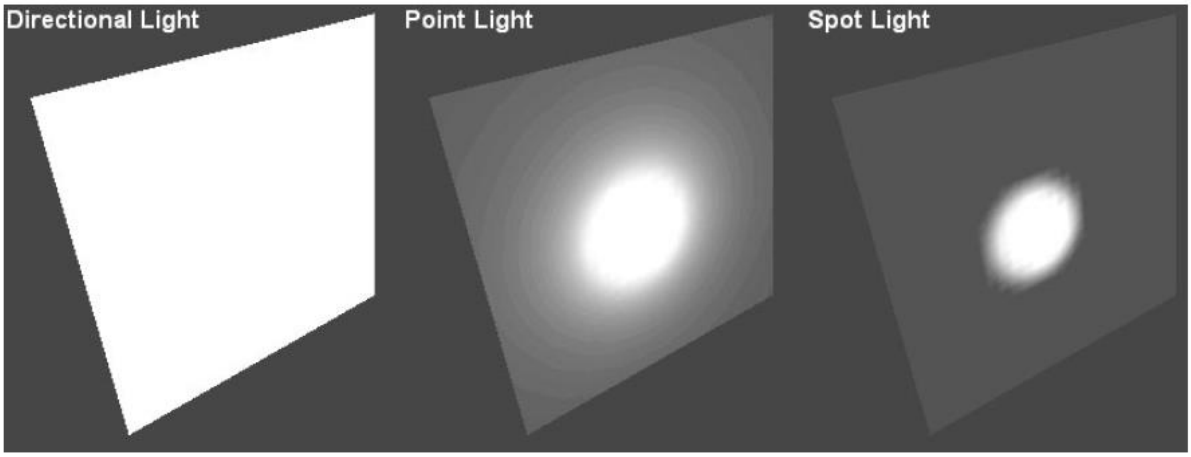


Ilustración 4-11 Tipos de Luz: Luz Direccional, Punto de luz y Foco de luz **¡Error! No se encuentra el origen de la referencia.**

4.2.2.4 Sombreado

Para el mayor realismo de una imagen, se utilizan las sombras para dar una mejor perspectiva de una imagen, además de revelar la posición de la fuente de luz y su intensidad. Debido a que computacionalmente es un proceso compl2ejo, existen múltiples algoritmos para formar sombras. La más utilizada es el "shadow maps". Este utiliza algoritmos de superficie visibles en el Z-buffer, en el cual se guardan valores de profundidad para punto

de la escena con respecto a la fuente de luz **¡Error! No se encuentra el origen de la referencia.**.

Ilustración 4-12 Ejemplo de sombreado **¡Error! No se encuentra el origen de la referencia.**

4.2.3 Flujo de Procesamiento gráfico 3D

A continuación será presentado el flujo de procesamiento gráfico 3D, que aunque varia de un motor a otro, la gran mayoría tiene un flujo como el siguiente.

Tareas de la aplicación y administración de la base datos de la escena se encarga de los movimientos de los objetos y cámaras, y como interactúa con el entorno 3D. Controla propiedades como la física, inercia, detección de factores entre otros factores las cuales actúan con los objetos. Además se encarga de administrar la base de datos de los objetos, donde se evalúa su existencia en la escena 3D y su ubicación **¡Error! No se encuentra el origen de la referencia.**.

- Teselado de superficies de orden superior: Aunque los objetos 3D son creados por principalmente por triángulos, también se pueden usar cuadriláteros. Algunos objetos se crean a partir de curvas, que son representadas por formulas matemáticas de orden superior que pueden resultar muy complejas, estas se caracterizan por tener una variable elevada por una potencia mayor a 1, como por ejemplo, $y = x^2 + 1$. Es por ello que las formulas deben ser desglosados en triángulos para su envió a la GPU.
- Sombreado por vértices, transformación e iluminación: En este paso es calculado las transformaciones que recibirán los objetos o cámaras, que puede ser escalarlos, rotarlos, trasladarlos, animarlos, etc. Debido a un cálculo matemático para obtener la nueva posición de un objeto. También se realizan los cálculos para la iluminación de cada vértice de los triángulos que componen los objetos.

- Organización de los triángulos: Toma los datos procedentes del motor de transformación e iluminación y los convierte matemáticamente de modo que el motor de sombreado por píxel pueda entenderlos.
- Sombreado por píxel y rendering (incluida la creación de texturas): En este paso se determina el color final que poseerá el pixel en base al color del objeto, distintas luces de la escena, usando el motor de transformación e iluminación. Luego se toma en cuenta las texturas que posee y sus propiedades que pueden incluir iluminación, reflejos, propiedades de material, etc. Finalmente el motor de rendering de píxeles es almacenar éstos en la memoria del búfer de tramas.
- Presentación en pantalla: El controlador de la pantalla, lee el búfer de tramas y la envía al dispositivo de salida seleccionado (monitor CRT, LCD, televisor, etc.) donde el usuario visualiza la escena 3D.

4.2.4 API

Las APIs son una capa intermedia que conecta los motores 3D con el hardware a utilizar, permitiéndoles a los motores facilitar la creación de entornos 3D sin importar el hardware en el que este inserto. Además las APIs de más alto nivel, permiten acceder a funciones que las de más bajo nivel no poseen.

4.2.4.1 OPENGL

Opengl es estrictamente definido como "Un interfaz de software para hardware gráfico (A software interface to graphics hardware)" . En esencia es una biblioteca de modelado y gráficas 3D que es portable y muy rápida. Usando Opengl, puedes crear elegantes y hermosas gráficas 3D con una excepcional calidad visual. La gran ventaja de usar opengl es que es más rápido que un "ray tracer" o un motor de software de renderizado. Inicialmente, usa algoritmos cuidadosamente desarrollados y optimizados por Silicon Graphincs, INC. (SGI), un reconocido líder mundial en gráficos de computadora y animación **¡Error! No se encuentra el origen de la referencia.**.

Opengl es una librería de modelado y gráficos 3D. Fue desarrollado por Silicon Graphics, luego paso a convertirse en un estándar, en especial cuando lo adapto Microsoft para su sistema operativo Windows.

Sus características son:

- Es fácilmente portable.
- Es un sistema procedural y no descriptivo, es decir, el programador no describe una escena sino los objetos de la escena y los pasos necesarios para configurar la escena final.
- Actúa en modo inmediato, es decir, los objetos son dibujados conforme van creándose.
- Incluye: Primitivas gráficas: puntos, líneas, polígonos, iluminación, sombreado texturas, animaciones y otros efectos especiales.

- Opengl trata con contextos de visualización o de rendering, asociados a un contexto de dispositivo que, a su vez, se encuentra asociado a una ventana.

4.2.4.2 DirectX

API creada por Microsoft, que permite a los desarrolladores de juegos programar eficientemente bajo Windows, debido a que permite acceder de manera más directa al hardware.

DirectX tiene como origen Direct3D, es una tecnología gráfica basada en COM. Microsoft adquirió esta tecnología en 1996 y la añadió a la API que constituye DirectX. Direct3D contiene una interface en modo inmediato y una estructura jerárquica para el manejo de gráficos. Un objeto COM proporciona acceso a funciones a través de una o más interfaces, por lo que podría considerarse como una especie de clase de C++. Las interfaces asociadas a los objetos COM, así como una gran librería de ejemplos, están definidas dentro de la ayuda suministrada por Microsoft.

Aunque la estructura original de DirectX era muy diferente a la de OpenGL; con el tiempo DirectX se está volviendo más parecido a OpenGL.

Una de sus principales características es ser independiente del lenguaje de programación. No obstante hasta la versión 7.0 sólo era soportada por C++. En la actualidad también es soportada por Visual Basic, Delphi y JAVA. El primero de ellos lo logra por medio de llamadas a funciones presentes en su DLL mientras que en el caso de JAVA se logra por medio de un paquete llamado JVM.

Las dificultades que posee DirectX es no ser un estándar abierto, por lo que Microsoft es el único que puede especificarla **¡Error! No se encuentra el origen de la referencia.**.

4.2.5 Herramientas utilizadas como motores 3D

Un motor 3D es un conjunto de funciones estructuradas para el manejo de objetos en tres dimensiones introducidos en un ambiente y que interactúan entre sí, como por ejemplo el motor 3D Game Studio o Jmonkey Engine.

El objetivo es crear un mundo virtual, donde se maneja el espacio y la física, para producir la sensación de movimiento, gravedad y colisiones.

Existen dos tipos de motores 3D, el comercial y el no comercial, por lo general los comerciales son los más completos y utilizados **¡Error! No se encuentra el origen de la referencia.**, pero en este caso solo se analizará el motor JMonkey Engine por su gratuidad y características que serán descritas a continuación.

4.2.5.1 JMonkey Engine

Es un motor 3D multiplataforma basado en JAVA, lanzado bajo licencia BSD. Entre sus características se encuentran: es de fácil uso, esta 100% escrito en Java, provee clases potentes y fáciles de usar para construir aplicaciones **¡Error! No se encuentra el origen de la referencia.**.

El SDK JMonkey Engine, viene en forma de jMonkey Platform, 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.

Soporta varios tipos de geometrías, como: líneas, puntos, modelos (OBJ, MESH XML, etc.), niveles de detalles como efectos de alto nivel en textura, mapeo de entorno, renderizado e iluminación.

Utiliza una arquitectura basada en SceneGraph.

El SceneGraph permite la organización de los datos de la aplicación 3D en una estructura de árbol. Esto permite un eficiente cálculo, ya que es beneficioso organizar los objetos en jerarquía o árbol, propagando cualquier información compartida hacia la raíz del árbol. Estas es una de las muchas maneras en que los arboles pueden ser usados.

Un scene Graph básico tendría un nodo raíz, con uno o más hijos. Cada nodo hijo a su vez puede tener uno o más hijos, algunos de los cuales pueden ser objetos gráficos que queremos dibujar. Los otros nodos son para propósitos estructurales y pueden volverse complejos **¡Error! No se encuentra el origen de la referencia.**.

Compatibilidad:

- Además de poseer todas las herramientas de JAVA, posee soporte para lenguajes como XML, PHP, Ruby, Scala , C y C++
- Se basa en los archivos de descripción en áreas como materiales, GUI y otros. Mediantes el uso de plugins desarrolladores se pueden fácilmente importar y editar estos formatos.
- Modelos: Ogre Mesh XML, WaveFront OBJ, MTL.
- Texturas: (JPG, PNG, GIF) que son formatos para imágenes.
- Audio: WAV y OGG (OpenAL)
- Video: OGV (OggVorbis)
- jME3 archivos binarios (objetos y escenas): j3o
- jME3 materiales: J3M

Este será el motor utilizado debido a estar escrito íntegramente en JAVA, por lo que se minimizan los riegos al utilizarlo en conjunto con JADE, que será tratado en el siguiente capítulo. Además de soportar los modelos creados por medio de Blender y ser además tanto JMonkey Engine y Blender totalmente gratuitos.

5 Sistemas Multiagente

5.1 Agentes

Según la rae **¡Error! No se encuentra el origen de la referencia.** una de las definiciones de agente es "Persona o cosa que produce un efecto", otras definiciones más cercanas al ámbito informático son: "Un agente es un sistema informático, situado en algún entorno, dentro del cual actúa de forma autónoma y flexible para así cumplir sus objetivos. Un agente recibe de su entorno y a la vez ejecuta acciones para intentar cambiarlo a su gusto"**¡Error! No se encuentra el origen de la referencia.**, pero este es solo una de las múltiples definiciones que se le han dado a los agentes, ya que como tal no existe una definición formal que los describa, es por ello que a lo largo de la literatura se ha intentado obtener la definición más completa y precisa.

5.2 Agente Inteligente

Para que un agente pueda ser considerado un agente inteligente debe poseer ciertos atributos básicos:

- Autonomía: Un agente debe tener una serie de objetivos y debe poseer conocimientos del mundo, de tal forma que partiendo de sus conocimientos sea capaz de aproximarse lo más posible a sus objetivos sin necesidad de que ningún usuario le guíe paso a paso hacia ellos.
- Sociabilidad: Un agente no debe ejecutarse de forma aislada, sino más bien en sistemas complejos (sistemas multiagente) en la que una serie de agentes colaboran entre sí para llevar a cabo una tarea. Se supone que los agentes son capaces de interactuar entre sí, y al mismo tiempo con entidades externas al propio sistema, como es el caso del usuario.
- Reactividad: Un agente debe ser capaz de percibir estímulos externos, tanto para estar de acuerdo a su entorno cambiante, como para saber que está pasando en el mundo. Estos estímulos afectarán las acciones realizadas por el agente para alcanzar sus objetivos.
- Pro-actividad: Debe ser capaz de elegir en cada momento las acciones que debe llevar a cabo para alcanzar sus objetivos. Es decir, una agente no sólo actúa en función de los estímulos que recibe desde el exterior, 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:

- Inteligencia: 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
- Racionalidad: Un agente se considera racional cuando tiene unos conocimientos de su entorno, unos objetivos deseables y unas reglas que determinan cómo alcanzar los objetivos a partir del conocimiento que se tiene del medio. Esta racionalidad le permite tomar decisiones sin intervención humana.
- Coherencia: El conocimiento que un agente tiene de su mundo se almacena en una base de conocimientos propia del agente. Todo este conocimiento debe guardar un alto grado de coherencia para que el comportamiento del mismo sea el esperado.
- Adaptación: El aprendizaje es una de las características más complejas que puede tener un agente. Un agente aprende cuando es capaz de aumentar su base de conocimientos y su base de reglas a partir de las percepciones que recibe del entorno y a partir de sus comportamientos anteriores a la hora de resolver problemas.
- Movilidad: Es una característica opcional que pueden poseer los agentes, y que actualmente está en boga. Un agente móvil es aquel que puede moverse físicamente por los nodos de una red para llevar a cabo sus tareas.

Poseer algunos de estos atributos, además de los correspondientes a la noción débil de agente, se clasifica como noción fuerte del agente.

5.3 Sistema Multiagente

Cuando el problema en sí mismo es imposible de resolver por medio de sólo un agente es necesario crear una sociedad de agentes que por medio de la colaboración permita el desarrollo de una solución. Este tipo de solución permite enfrentar de manera más natural y cercana a la realidad el problema, lo que permite una abstracción que se traduce en menor dificultad para su desarrollo.

Según Weiss **¡Error! No se encuentra el origen de la referencia.** "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"

5.4 Arquitectura de agentes

Debido a los diferentes contextos en los que interactúan los agentes, es necesario que posean distinto grados de inteligencias, que les permitan tener una mejor interacción con el ambiente, en donde se privilegia el uso de recursos, el tiempo o una decisión más compleja. Es por ello que a continuación se analizarán los distintos tipos de arquitecturas con sus ventajas y desventajas.

5.4.1 Arquitecturas Deliberativas

Esta arquitectura también es la llamada la arquitectura basada en lógica.

Es un aproximado tradicional a la construcción de sistemas inteligentes, sugiere que la tarea de la inteligencia puede ser generada en un sistema entregando al sistema una representación simbólica de su ambiente y sus tareas de deseos, y sintácticamente manipular su representación. Este tipo de arquitectura tiene sus problemas, ya que no es sencillo realizar una representación simbólica del ambiente, debido a que puede ser muy complejo y además las ausencias de garantías de que el procedimiento de toma de decisiones termine de manera oportuna.

5.4.2 Arquitecturas Reactivas

Los agentes con este tipo de arquitectura se basan en un ciclo de estimulo- respuesta, por lo que no hay una representación explicita del entorno, de los otros agentes, sus capacidades, etc. Es por ello que las decisiones no tienen en cuenta ni el pasado ni el futuro. Esto les permite funcionar bien en entornos de tiempo real, ya que son computacionalmente económicas. Esto permite la creación de agentes compactos, tolerante a fallos y flexibles. Debido a todo esto es necesario enfatizar el proceso de interacción. Los defensores de la estructura reactiva dicen que la inteligencia no existe en sistemas individuales, sino que es parte implícita de un entorno completo.

5.4.3 Arquitectura BDI

También conocido como la arquitectura de creencias, deseos e intenciones. Esta arquitectura tiene sus raíces en la tradición filosófica del entendimiento del razonamiento práctico, que es el proceso de decidir, momento por momento, que acción realizar para cumplimiento de sus metas **¡Error! No se encuentra el origen de la referencia.**.

El razonamiento practico divide los comportamientos en dos procesos: la decisión de las metas a alcanzar (deliberación) y como se alcanzarán esas metas (razonamiento de medios y fines). Las creencias son los conocimientos del agente de su entorno y de sí mismo. Los deseos son los objetivos o motivos que desea alcanzar el agente .Las intenciones juegan un papel importante en el proceso de razonamiento, ya que es el que determina las acciones del agente, luego por medio de la persistencia se intenta lograr la acción.

Esta arquitectura también es llamada conocimiento procedimental debido a que tiene un conjunto de planes para alcanzar el objetivo o para reaccionar ante situaciones.

5.4.4 Arquitectura Hibrida

Debido a que cada arquitectura tiene sus propios problemas, también se suelen utilizar un arquitectura hibrida, donde los planes y decisiones son realizados por medio de una arquitectura deliberativa y los eventos producidos por el ambiente son manejados por medio de una arquitectura reactiva.

Una solución para esta arquitectura está en dividirlo en dos o más capas, donde las capas pueden tener acceso al conocimiento del y entorno, y alguna de ellas poder realizar acciones sobre él. Debido a estos se dividen en dos tipos:

- Horizontal: Todas las capas tienen conocimiento directo del entorno y pueden actuar sobre él.
- Vertical: Solo una capa recibe directamente conocimiento del entorno, que entrega a otras capas. Luego la misma u otra capa puede realizar la acción.

5.5 Arquitectura Multiagente

En las arquitecturas multiagente, nacen debido a la necesidad de resolver problemas que por medio de un solo agente es imposible o muy difícil, y son necesarios subsistemas que interaccionan entre sí para la distribución de la inteligencia entre los distintos agentes. Este tipo de solución es principalmente ocupada cuando los sistemas físicamente están distribuidos o donde es necesario experiencia heterogénea para su solución. Esto permite descentralizar las decisiones y la información recaiga sobre solo un agente, lo cual permite soluciones más robustas y adaptación en su ambiente.

La Arquitectura FIPA (Foundation for Intelligents Physical Agents) nace de la necesidad de estandarización de la tecnología de agentes, lo que resuelve problemas de interoperabilidad y apertura. En sus especificaciones se definen las características que debe poseer una plataforma multiagente, centrándose en el comportamiento externo, dando libertad en el diseño.

5.6 Infraestructura de agentes

La infraestructura de agentes permite a los agentes socializar y comunicarse entre ellos mediante, definiendo las reglas que les permiten entenderse unos con otros, para conjuntamente lograr una solución.

5.6.1 Ontología

La ontología define las primitivas de la comunicación. Esto quiere decir que son descritas las entidades de los dominios, sus propiedades y relaciones. Es por ello que representa la jerarquización de los conocimientos acerca de las cosas mediante la subcategorización de sus cualidades esenciales.

La gran ventaja de las ontologías no está en el procesamiento, sino en el sentido de compartir, emerger y descubrir los vacíos para mejorar la transferencia del conocimiento tácito. Las ontologías pueden contener información en un lenguaje declarativo específico, pero también podría incluir información no estructurada o no formalizada expresada en un lenguaje natural o código procedimental.

Las ontologías computacionales proveen una representación formal y estructurada del conocimiento del dominio, esto permite tener especificada una definición de cada uno de los términos, para la interoperación de los agentes. Esto es necesario debido a que si bien un agente puede ser muy distinto a otro en sus metas o capacidades, estos se puedan comunicar para lograr avanzar en una tarea en común.

Las ontologías en agentes no pueden ser representadas por UML, cualquier lenguaje basado en objetos debido a la no existencia de las múltiples herencias, propiedades inversas y otras características existentes en RDF, DAM+OIL, OWL **¡Error! No se encuentra el origen de la referencia.**.

5.6.2 Comunicación entre agentes

Para la cooperación entre los agentes, es esencial el poder comunicarse. Es por ello para que la comunicación sea llevada a cabo, los agentes deben compartir tres aspectos:

- Sintaxis: se refiere a la estructura de los símbolos usados para comunicarse y como ellos pueden ser usados en base a sus reglas.
- Semántica: se refiere al significado de los símbolos para su comprensión.
- Pragmática: Representa como los símbolos son interpretados.

Por medio de la combinación entre semántica y pragmática aparece el significado. Para la comunicación es necesario un emisor, que es quien envía el mensaje y un receptor, que es quien recibe el mensaje. En los agentes se basa en el modelo humano del acto del habla para comunicarse**¡Error! No se encuentra el origen de la referencia.**.

5.6.2.1 Acto del habla

La comunicación humana es usada como modelo para la comunicación de los agentes computacionales. La teoría del acto del habla ve el lenguaje natural como acciones, peticiones, sugerencias, compromisos y respuestas.

El Acto del habla posee tres aspectos:

- Locución: la expresión física del orador.
- Ilocución: el significado de lo que el orador quiere decir.
- Perlocución: La acción que resulta de la locución.

La teoría usa el término "performative" para identificar el motivo de la ilocución del mensaje para evitar ambigüedades por medio de verbos que pueden ser por ejemplo: prometer, reportar, convencer, insistir, decir, pedir y demandar.

Por medio de los "performative" se puede por tanto eliminar la ambigüedad del mensaje, pero no clarifica ni su sintaxis o significado del mensaje.

Para la comunicación de agentes actualmente se utilizan dos lenguajes, KQML y FIPA ACL**¡Error! No se encuentra el origen de la referencia.**.

5.6.2.2 KQML

El "Knowledge query and manipulation language (KQML)" es un protocolo para el intercambio de información y conocimiento. La ventaja de KQML es que toda la información para el entender el contenido del mensaje es incluido en el mensaje en sí mismo. La información básica en un mensaje KQML es el siguiente:

- Performative: se refiere al tipo de acto comunicativo del mensaje.

- Content: corresponde al contenido del mensaje.
- Sender: identidad del agente que envía el mensaje.
- Receiver: identidad de el o los agentes que deben recibir el mensaje.
- Language: nombre del lenguaje en el que está escrito el mensaje.
- Ontology: ontología utilizada para dar significado a los símbolos del mensaje.
- Reply-with: Identificador para ser usado por un mensaje en respuesta a este mensaje.
- In-reply-to: Identificador del mensaje que provocó el envío de este mensaje.

El mensaje KQML utiliza la estructura de Lisp para cada uno de sus campos

5.6.2.3 FIPA ACL

FIPA ACL es un lenguaje de comunicación asociado con la arquitectura abierta de FIPA, el cual está basado en los actos de habla. Al igual que KQML, está diseñado para trabajar con cualquier lenguaje y especificación de ontología. FIPA ACL posee una sintaxis similar a la de KQML, pero posee algunas diferencias en las "performatives" predefinidos, semánticas utilizadas y diferente tratamiento de las primitivas**¡Error! No se encuentra el origen de la referencia.**.

Los parámetros de un FIPA ACL mensaje son los siguientes

- Performative: Denota el tipo acto comunicativo del mensaje.
- Sender: La identidad del emisor del mensaje.
- Receiver: La identidad del receptor del mensaje.
- Reply-to: Indica la subsecuencia del mensaje en la conversación, con el nombre del emisor del lenguaje.
- Content: Se refiere al contenido del mensaje.
- Language: Se refiere al lenguaje utilizado en el contenido del mensaje.
- Encoding: Se refiere a la codificación del contenido del mensaje.
- Ontology: Ontología usada para describir el significado de los símbolos del mensaje.
- Protocol: Se refiere al protocolo de interacción utilizado.
- Conversation-id: Introduce un identificador para identificar una conversación en marcha.
- Reply-with: Introduce una expresión para que el agente responda con el identificador el mensaje-
- In-reply-to: Un identificar que denota que el mensaje es una respuesta.
- Reply-by: Denota un tiempo y/o fecha que indica la última vez que se recibió un mensaje.

5.6.3 Protocolos de interacción

En el protocolo de interacción, la comunicación de los agentes se ve desde el punto de vista de los patrones típicos de secuencia de mensaje. Por eso, FIPA ha definido ciertos protocolos.

FIPA para definir estos protocolos, ocupa notación llamada AUML que se basa en diagramas de UML, para dar origen a los diagramas de protocolo. En estos se encuentran:

- Roles de agentes: Se refiere al papel de los agentes en la comunicación.
- Línea de vida: Se refiere al tiempo de participación del agente en la comunicación.
- Hilos de interacción: Se refiere al tiempo utilizado de una tarea como una reacción a un mensaje entrante.
- Mensajes: Es la acción de comunicación realizada de una agente a otro.

A continuación una lista de distintos protocolos de interacción estándar de FIPA.

- Request: Un agente que pide que se realice una acción.
- Request when: Agente que pide que se realice una acción si se cumple la condición.
- Query: Cuando se le pide a un agente que de información.
- Contract net: Un agente pide la realización de una tarea a un conjunto de agentes. Luego estos entregan su propuesta, la propuesta con menor costo (o especificado) es elegido para realizar la tarea.
- Brokering: Un agente hace de intermediario, ofreciendo funcionalidades de otros agentes.
- English Auction: Varios agentes participan en una subasta, donde empieza con un precio bajo, y gana el agente con el mayor precio.
- Propose: El iniciador propones a los agentes una tarea, que pueden realizar o no.

5.7 Plataforma Multiagente

El siguiente entorno de desarrollo es utilizado para propósitos generales, ya que permite acceder a las características de los agentes, según las especificaciones FIPA.

5.7.1 JADE

Es un software framework para desarrollar aplicaciones de agentes en conformidad con las especificaciones de FIPA para sistemas multiagente inteligentes. Su objetivo es simplificar el desarrollo y garantizar el cumplimiento estándar a través de un conjunto integral de servicios del sistema y de los agentes. JADE puede ser considerado un agente middleware que implementa una plataforma de agente y un framework de desarrollo.

JADE está escrito en lenguaje JAVA y hecho por varios JAVA packages, el que cual fue elegido por muchas de sus características como estar basado en la programación orientada a objetos el cual puede funcionar a través de muchos ambientes heterogéneos.

Además JADE proporciona para la facilidad de los desarrollos de agentes un sistema de administración de agentes (AMS), el cual proporciona interesantes características como servicio de páginas amarillas (DF) y que por medio de su interfaz (Ilustración 5.1) visualizar la interacción de los agentes**¡Error! No se encuentra el origen de la referencia.**.

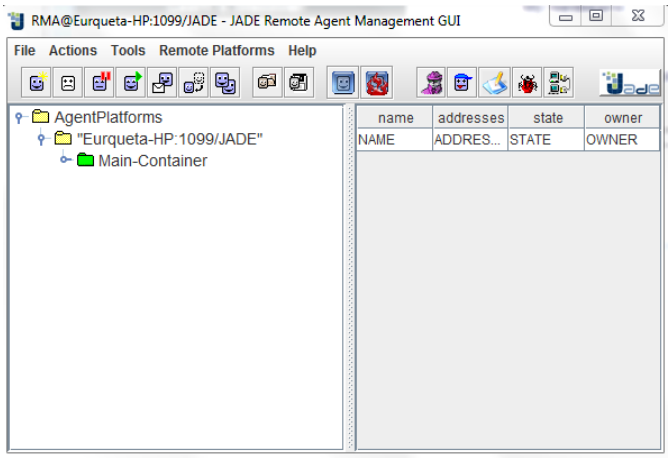


Ilustración 5-1 Interfaz JADE

5.8 Agent Oriented Software Engineering

A continuación se evaluarán las distintas metodologías de desarrollo de sistemas en agentes

5.8.1 Tropos

Tropos está basado en dos ideas principales. Primero, la noción de agente y todo lo relacionado con las nociones mentalistas (por ejemplo metas y planes) son usados en todas las fases del desarrollo de software. Segundo, Topos cubre también las más tempranas fases del análisis de requerimientos, permitiendo así un profundo entendimiento del entorno donde el software debe operar, y el tipo de interacciones que deben ocurrir entre el software y agentes humanos. La metodología es ilustrada con la ayuda de un caso de estudio. El lenguaje tropos para modelado conceptual es formalizado en un metamodelo descrito con un conjunto de diagramas de clase UML **¡Error! No se encuentra el origen de la referencia.**.

5.8.2 SADAAM: Software Agent Development an Agile Methodology

SADAAM es una metodología para desarrollo con agentes que utiliza técnicas ágiles para facilitar el desarrollo e implementación de sistemas multiagente. Se centra en la entrega de código ejecutable más que en la documentación y un detallado diseño, y sistemáticamente guía y ayuda a los desarrolladores a través de las distintas etapas del desarrollo del sistema **¡Error! No se encuentra el origen de la referencia.**.

5.8.3 Prometheus

Pretende ser una metodología práctica, proveyendo todo lo que es necesario para especificar y diseñar sistemas de agentes. Entre sus características se encuentran el ser detallado, lo que provee una guía de cómo realizar cada uno de los pasos del proceso de Prometheus. Además soporta el diseño de agentes basado en metas y planes, también cubre un rango de actividades de especificación de requerimientos a través de un diseño detallado **¡Error! No se encuentra el origen de la referencia.** .

5.8.4 PASSI: Process for Agent Societies Specification and Implementation

Es una metodología de requerimiento a código paso a paso, para el desarrollo e implementación de sociedades multiagente integrando modelos de diseño y conceptos ingeniería de software orientada a objetos e inteligencia artificial usando notación UML. Los modelos y fases de PASSI abarcan representaciones antropomórficas de los requerimientos del sistema, el punto de vista social, la solución de la arquitectura, el reúso de producción de código y la configuración de desarrollo soportando agentes móviles **¡Error! No se encuentra el origen de la referencia.** .

Los modelos de PASSI son los siguientes

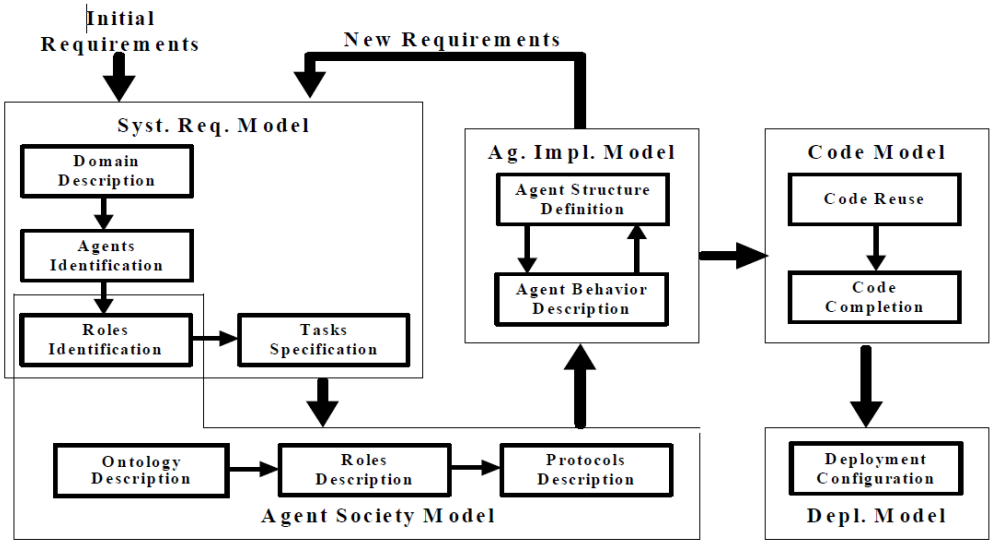


Ilustración 5-2: Metodología PASSI [32]

- Modelo de Sistema de requerimientos: Corresponden a los modelos asociados a la captura de requerimientos y corresponde a cuatro pasos.
    - Descripción de dominio: Es usado para describir los requerimientos a base de diagramas de casos de uso, los cuales resultan en una descripción del sistema en base a diagramas de caso de uso jerarquizados.

    - Identificación de agentes: En esta fase se toman los casos de usos creados en la descripción de dominio y donde cada caso de uso o paquete tiene asociado un agente que se hace responsable.
    - Identificación de roles: Por medio de diagramas de secuencia se analizan los agentes en diferentes escenarios.
    - Especificación de tareas: Por medio de diagramas de actividad se analizan las capacidades de los agentes.

- Modelo de Sociedad de agentes: Corresponde a los modelos donde se analiza las interacciones sociales y dependencias entre los agentes.
    - Identificación de roles: Aun cuando es una etapa que ocurre cercano a la captura de requerimientos, debido a su componente social también corresponde a este modelo.
    - Descripción de la ontología: Posee dos diagramas, la descripción del dominio de la ontología y la descripción de la comunicación de la ontología. Estos representados por medio de esquemas XML que permiten tener un punto de vista desde la ontología la sociedad de agentes.
    - Descripción de roles: Modela el ciclo de vida del agente tomando en cuenta sus roles, las colaboraciones que necesita y las conversaciones en las que está involucrado.
    - Descripción de protocolos: Uso de diagramas de secuencia para especificar la gramática de cada protocolo de comunicación en términos de los "performatives".
- Modelo de implementación de agentes: Un modelo de la solución de arquitectura en términos de clases y métodos.
    - Definición de estructura de agentes: Esta fase está íntimamente ligada a la descripción de los comportamientos del agente, lo que produce un doble nivel de iteración. En esta fase se producen una serie de diagramas de clase que esta divida entre multiagente y agente. Se describen las tareas asociadas, su estructura interna y sus comportamientos.
    - Descripción de los comportamientos de los agentes: Se produce una serie de diagramas dividido en agente y multiagente. Se dibuja el flujo de los eventos por la invocación de métodos y el cambio de mensaje.
- Modelo de código: Un modelo de la solución a nivel de código.
    - Re-uso de código: Se intenta re-usar patrones de agentes y tareas. Esto incluye tanto código como diagramas.
    - Completado de código: Código fuente del sistema objetivo.
- Modelo de despliegue: Un modelo de la distribución de las partes de los sistemas a través del hardware y sus migraciones.
    - Configuración de despliegue: Uso de diagramas de despliegue para describir la posición de los agentes.

Para el desarrollo de la aplicación se utilizará la plataforma JADE que permite el uso de las características de los agentes, en un ambiente JAVA.

6 Trabajos Anteriores

A continuación se describirán "Desarrollo de un sistema multiagente para el Berth Allocation Problem" de René Díaz**¡Error! No se encuentra el origen de la referencia.** donde en base al enfoque discreto, plantea una solución multiagente.

Luego por el lado de las interfaces 3D para sistemas multiagente, se analiza la solución "Visualización 3D de sistema multiagente aplicado al problema de transporte de pasajeros" de Felipe Baranlloni **¡Error! No se encuentra el origen de la referencia.**.

6.1.1 Desarrollo de un sistema multiagente para el Berth Allocation Problem

Se crea una solución basada en el sistema discreto del problema que considera cancelaciones y atrasos de barcos, además del cierre del alguno de los puntos de atraque. Para la creación del algoritmo se utilizo una versión modificada de la heurística de inserción propuesta por Jaw et al. para el problema de multiple dial-a-ride con ventanas de tiempo. Esta será la solución usada como base para el desarrollo de la interfaz del presente proyecto.

Para la creación de la solución se utilizo la siguiente formulación matemática**¡Error! No se encuentra el origen de la referencia.**.

El problema podía fue modelado como un *Multi-Depot Vehicle Routing Problem with Time Windows***¡Error! No se encuentra el origen de la referencia.**. En este modelo los barcos son vistos como *clientes*, y los puntos de atraque como *depósitos* en donde cada vehículo está ubicado. Existen entonces *m* vehículos, uno para cada depósito. Además, cada vehículo parte y termine el tour en su propio depósito. Los barcos son modelados como vértices de un *multigrado*. 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. Los datos de entrada son los mismos que los de DBAP, sumados a los siguientes:

- *bi :* límite superior de la ventana de tiempo de servicio en el barco *i*;
- *vi* : el valor 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:

- $X_{ipk} \in \{0.1\}$ $k \in M$, (i , j) A_k , $X_{ipk} = 1$, sí y solo sí el barco *j* esta 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

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

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

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

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

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

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

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

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

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

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

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

La función objetivo es la minimización de la suma de los tiempos de servicio con un cierto *peso* asignado. Cuando la nave i no está asignada al punto de atraque *k*, el término correspondiente en la función objetiva es 0 puesto que $\sum_{j \in N \cup d(k)} x_{ij}^k = 0$ y $T_{ik} = a_i$, 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 T_k 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) **¡Error! No se encuentra el origen de la referencia.**.

Para el desarrollo del problema en multiagente se utilizo la metodología PASSI sobre la plataforma JADE, creando la siguiente arquitectura:

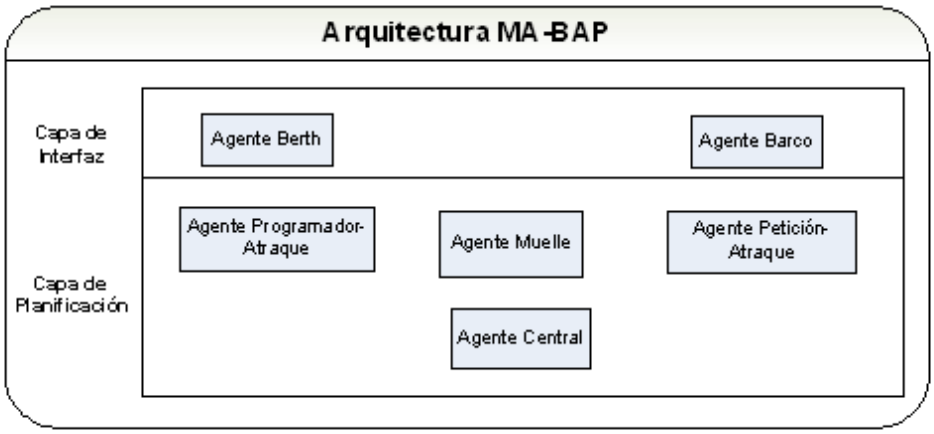


Ilustración 6-1 Arquitectura multiagente BAP por René Díaz **¡Error! No se encuentra el origen de la referencia.**

Donde se puede apreciar a que capa de la arquitectura corresponde cada uno de los agentes. Los agentes de la capa de interfaz corresponden a los que actúan con el usuario donde se puede donde se puede agregar, modificar o eliminar barcos en el sistema y eliminar o modificar sitios de atraque. Cualquier operación realizada en esta capa de la arquitectura gatilla un efecto en la capa inferior llamada capa de planificación.

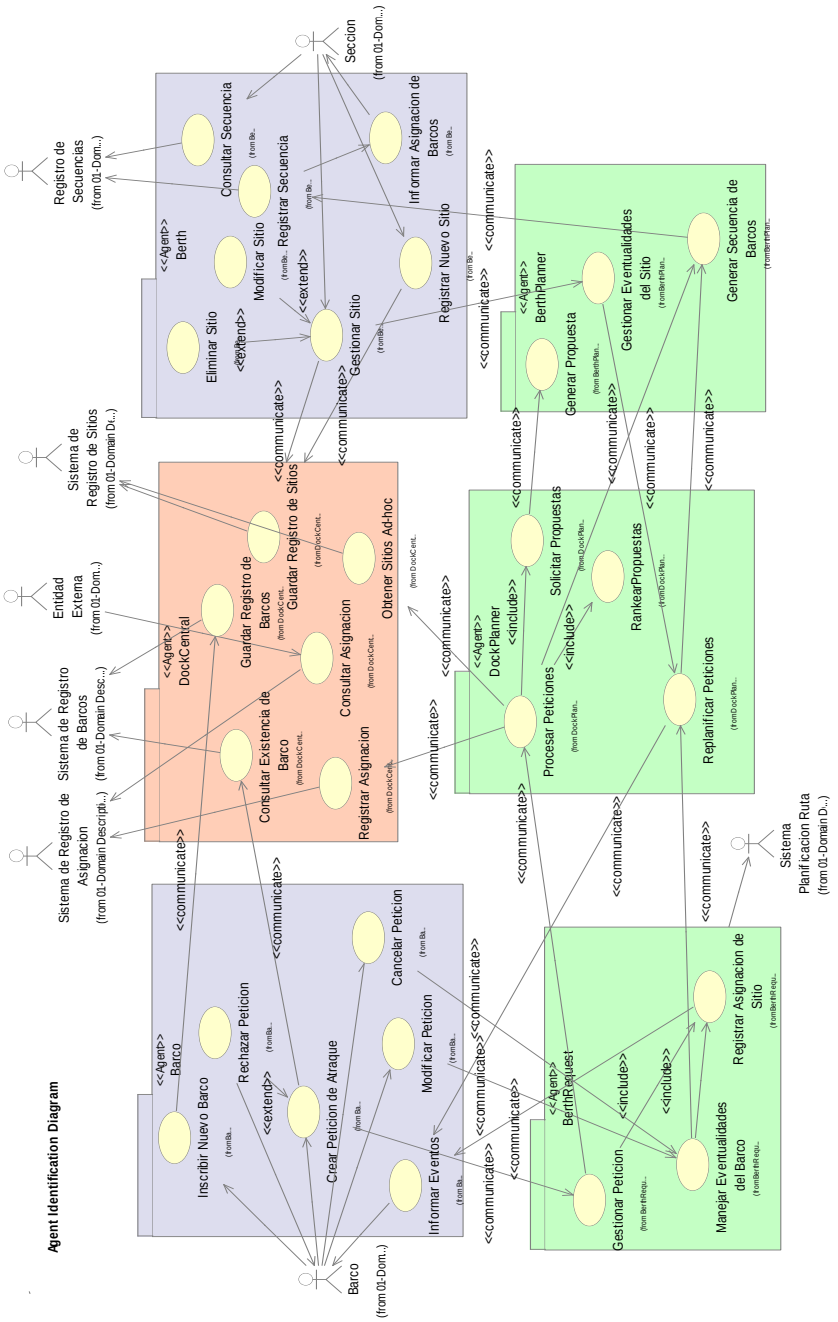
En la capa de planificación se realiza todo lo relacionado con la lógica del problema, donde funciona el algoritmo de inserción creado especialmente para la asignación de los barcos a los puntos de atraque, también incluye el manejo de eventualidades que produce la re-planificación de las asignaciones y además el guardado de los registro de cada una de ellas para su posterior consulta.

Actualmente los agentes tienen las siguientes obligaciones:

- Agente Barco: Se encarga de recibir todas las órdenes de inscripción de nuevo barco, cancelación y modificación que son comunicadas al agente Berth Request, además de recibir diferentes eventos que lo podrían afectar.
- Agente Berth Request: Se encarga de recibir todas las peticiones del agente Barco y generar su petición de asignación al agente Dock Planner.
- Agente Berth: Esta encargado de recibir todas las órdenes relacionadas con eliminar sitios de atraque, modificarlos o crear nuevos sitios, todos estos cambios se informan al agente Dock Planner y para su asignación de barcos, se comunica con el agente Berth Planner.
- Agente Berh Planner: Se encarga de generar la propuesta y generar nuevas propuestas en caso de eventualidad, que comunica al agente Dock Planner.

- Agente Dock Planner: Se encarga de recibir las peticiones del agente Berth Request y las propuestas agente Berth Planner, para realizar las asignaciones y comunicárselas finalmente al agente Dock Central.

-



-

Ilustración 6-2 Identificación de agentes por René Díaz**¡Error! No se encuentra el origen de la referencia.**

- Agente Dock Central: Se encarga de registrar todas las asignaciones, barcos en existencia y sitios de atraques, para ser comunicados a quien lo necesite.

El siguiente diagrama de Identificación de agentes muestra cómo están relacionados los agentes y como se comunican entre ellos:

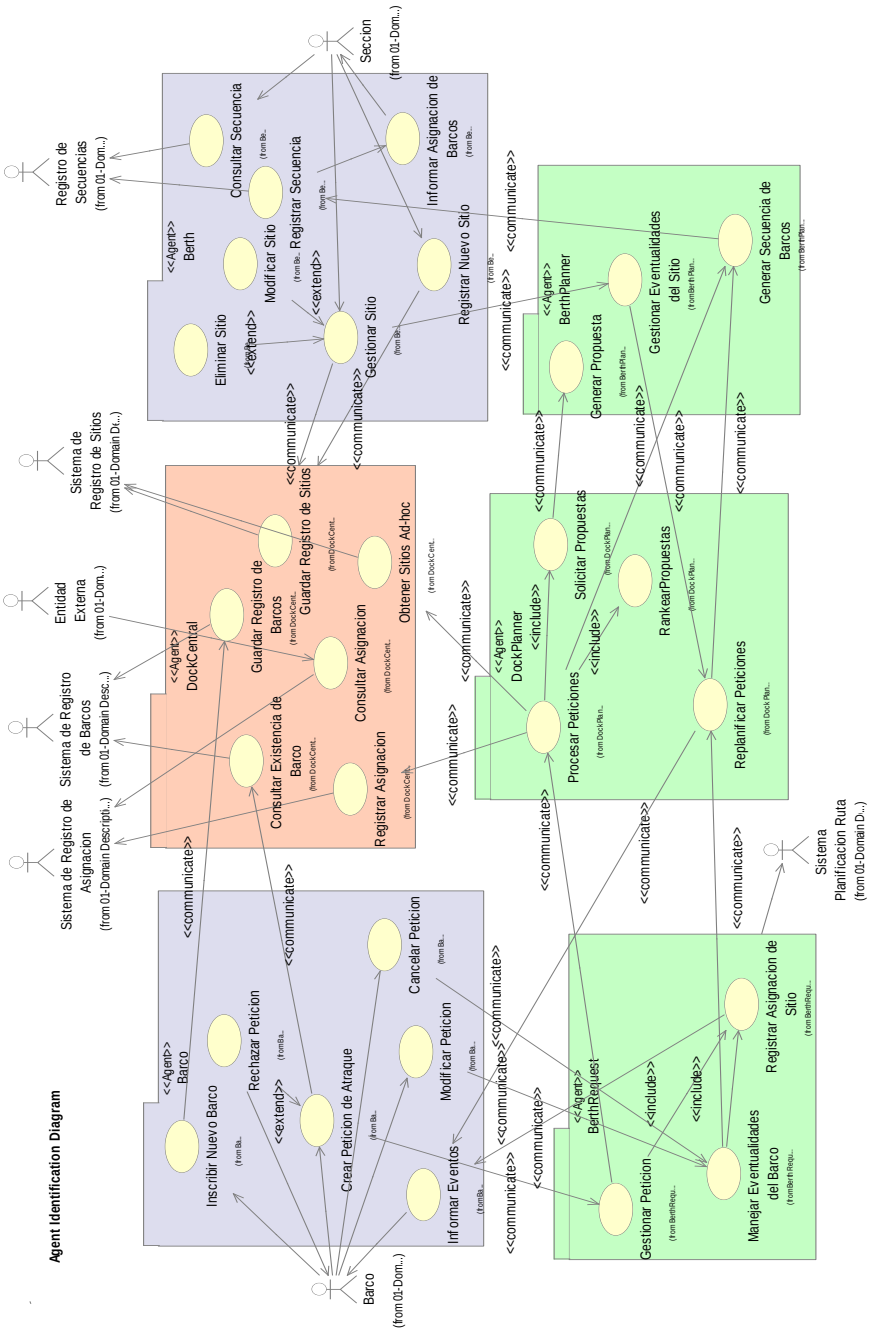


Ilustración 6-3 Identificación de agentes por René Díaz**¡Error! No se encuentra el origen de la referencia.**

El algoritmo de inserción creado para la solución del Berth Allocation Problem fue el siguiente:

Fue basado en la heurística de inserción propuesta por Jaw et al. en **¡Error! No se encuentra el origen de la referencia.** para el problema Dial-a-Ride con ventanas de tiempo, donde se identifico las diferencias con su problema y por tanto se realizaron las modificaciones necesarias. En la solución inicial existían dos eventos, uno de pick-up del cliente y otro de delivery del cliente. En BAP solo existe un evento asociado al barco, que es el comienzo de atención del barco.

La creación de la ventana de tiempo inicial para cada barco tendrá relación con el tiempo de procesamiento asociado a éste. Para el Barco A de tiempos de llegada y partida estamas como ET y LT, y tiempo de procesamiento N, se tendrá como ventana de tiempo máxima [ET,LT] y su tiempo factible inicial como [ET,LT-N].

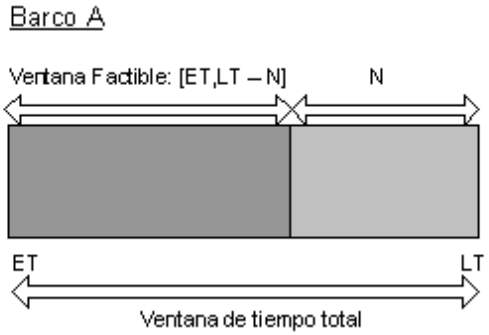


Ilustración 6-4 Ventanas de tiempo por René Díaz**¡Error! No se encuentra el origen de la referencia.**

En el problema original de transporte de pasajeros se buscaba disminuir el tiempo de ocio de los vehículos donde no realizaban ninguna acción en particular debido a que no atendían a ningún cliente en ese determinado momento, en BAP eso se tradujo en disminuir los tiempo de ocio de los berth que no atendían a los barcos en algún determinado momento. El tiempo de ocio o "slack" está determinado entre el tiempo en que el berth este listo para recibir un nuevo barco y la llegada de éste. Por medio de estos tiempos de ocio se permite diferencias las distintas propuestas de solución a cada berth y encontrar mejor solución.

El Algoritmo de inserción funciona exactamente de esta forma:

Al llegar un nuevo evento de atraque **B** al *berth X*, se revisa en la secuencia del *berth* lo siguiente:

- Si la secuencia está vacía, simplemente se inserta el nuevo evento en ella.
- Si la secuencia no está vacía, se busca el evento justo anterior (evento **A**) donde debería insertarse el nuevo evento. Esto comparando el ET de llegada de ambos.
- Una vez que se sabe donde eventualmente debería insertarse, se traslada la ventana factible del evento **A** en un delta dado por el tiempo de procesamiento de **A** más el tiempo que tarde el *berth* en estar listo para recibir un próximo evento. En este procedimiento pueden ocurrir uno de los siguientes casos:

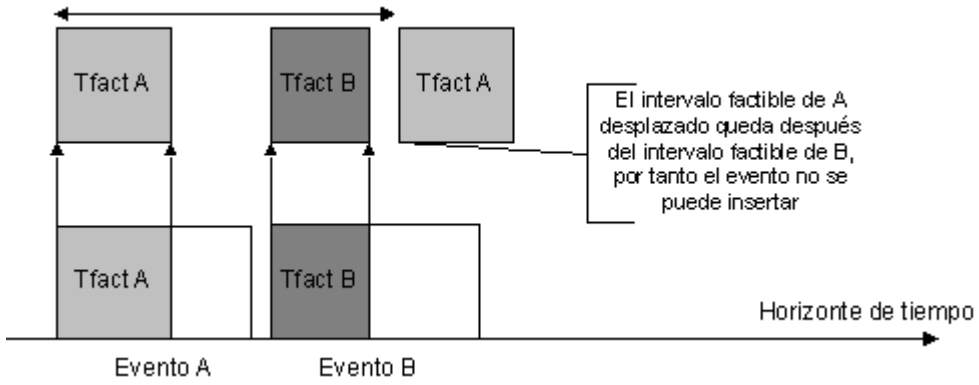


Ilustración 6-5 Caso de imposibilidad de inserción por René Díaz **¡Error! No se encuentra el origen de la referencia.**

En el caso de la Ilustración 6.4, el intervalo factible del evento ya insertado **A** desplazado, no logra intersectar con el intervalo factible de **B**, y queda pasado a éste. Por tanto el evento B no puede insertarse en la secuencia del *Berth X*.

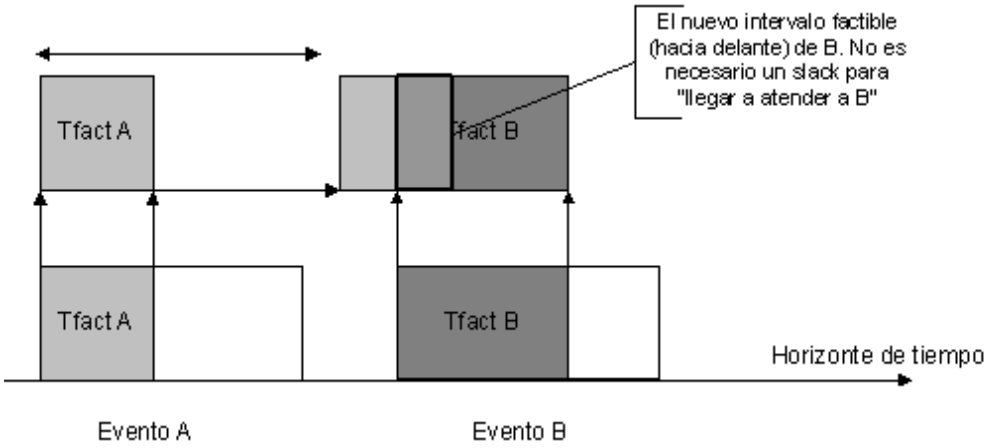


Ilustración 6-6 Caso de posibilidad de inserción por René Díaz **¡Error! No se encuentra el origen de la referencia.**

En la Ilustración 6.5 se puede observar el caso en que la intersección es distinta de vacía, por tanto no es necesaria la inserción de un slack de tiempo para que el evento **B** logre ser atendido en el *berth X*.

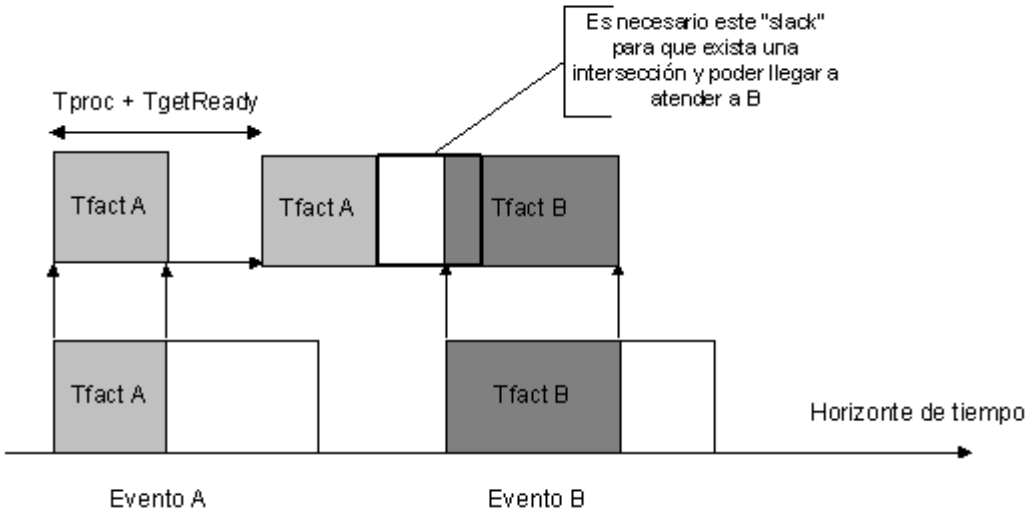


Ilustración 6-7 Caso de posibilidad de inserción mediante tiempo de slack por René Díaz **¡Error! No se encuentra el origen de la referencia.**

En el caso de la Ilustración 5.8, si la intersección resulta ser vacía pero el intervalo factible de **A** desplazado queda al lado izquierdo del intervalo factible de **B** , como muestra la Figura, es posible la inclusión de un tiempo de slack que permita la intersección. Este tiempo de slack finalmente resulta en un tiempo de penalización en esa inserción.

Luego de determinada la factibilidad (hacia delante) y de ser necesario calculado el tiempo de slack, se verifica si el lugar a insertar el nuevo evento es el final de la secuencia o no. Si es el final de la secuencia, simplemente se inserta el nuevo evento en ella, con o sin la penalización que corresponda (tiempo de slack). Si no es el final de la secuencia, y el evento debe insertarse antes que otro, es necesario realizar una validación de factibilidad con el evento posterior **C** (hacia atrás). Para esto se toma el intervalo factible del evento **C** y se desplaza hacia atrás en el tiempo de procesamiento de **B** más el tiempo para que el *berth* esté listo (getReadyTime). Como resultado puede que se de uno de tres casos análogos a los vistos anteriormente.

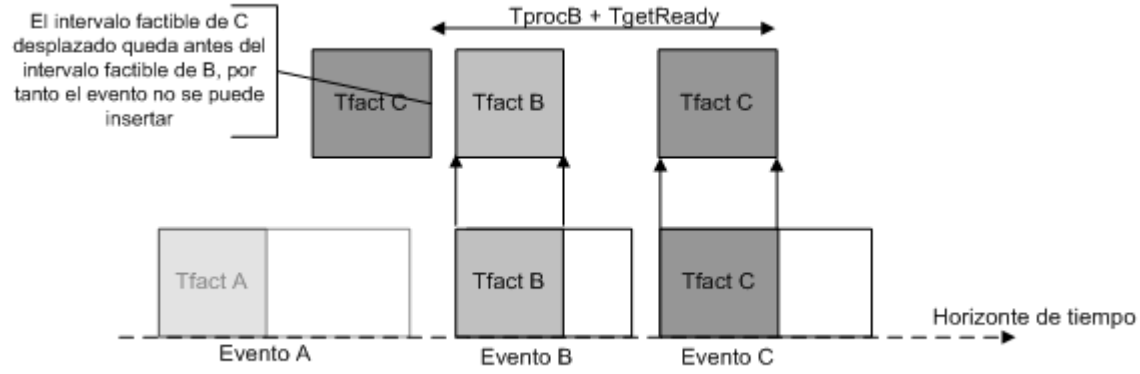


Ilustración 6-8 Caso de imposibilidad de inserción (factibilidad hacia atrás) por René Díaz **¡Error! No se encuentra el origen de la referencia.**

En la Ilustración 6.7 se puede observar que el intervalo factible de **C** desplazado hacia atrás queda al lado izquierdo del intervalo factible de **B**, imposibilitando la inserción del evento **B**.

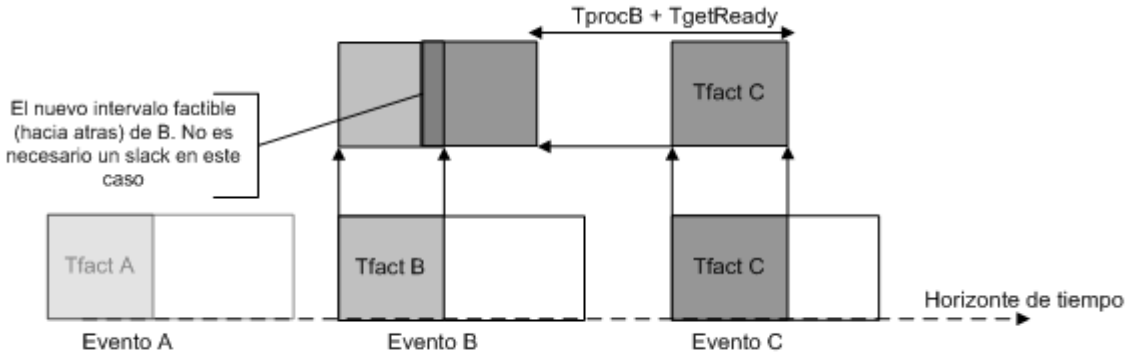


Ilustración 6-9 Caso de posibilidad de inserción (factibilidad hacia atrás) por René Díaz **¡Error! No se encuentra el origen de la referencia.**

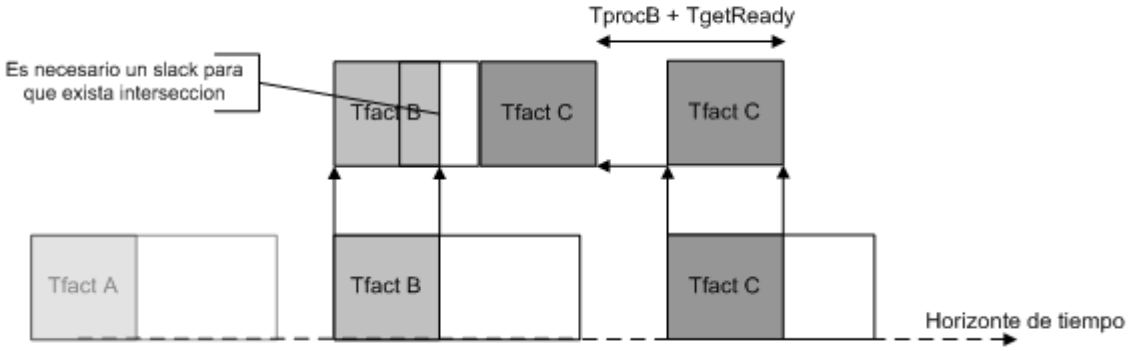


Ilustración 6-10 Caso de posibilidad de inserción con tiempo de slack (factibilidad hacia atrás) por René Díaz **¡Error! No se encuentra el origen de la referencia.**

En la Ilustración 6.8 y 6.9, existe la posibilidad de inserción. En el primer caso, existe una factibilidad "hacia atrás" pues existe una intersección entre el tiempo factible de **C** desplazado y el tiempo factible de **B**. En el caso de la Figura 4.17 es necesario agregar un slack de tiempo para que esta intersección sea distinta de vacía.

En ambos casos anteriores, para verificar definitivamente la posibilidad de inserción, es necesario verificar que el intervalo de tiempo resultado de la intersección realizada en primer lugar "hacia delante", y el intervalo de tiempo resultante del último proceso de verificación de factibilidad "hacia atrás", intersecten. En caso de que éstos se intersecten se puede realizar la inserción del evento, con cualquier penalización que acarree de los procedimientos pasados. En caso de que no intersecten, se puede sumar a la penalización que exista en el momento, un intervalo más de penalización que permita la inserción definitiva del evento. En la Ilustración 6.10 se puede observar finalmente que es necesario agregar un último tiempo de slack para que las ventanas puedan intersectar.

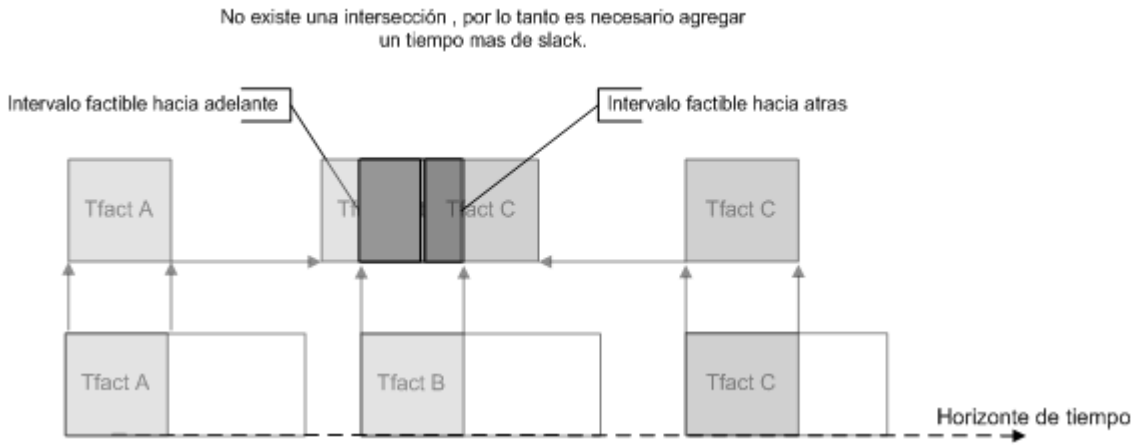


Ilustración 6-11 Proceso de inserción - Intersección "triple" vacía por René Díaz

Finalmente en la Ilustración 6.11 se da el caso de la intersección triple no vacía, por tanto se puede insertar el nuevo evento **B** entre los eventos **A** y **C**, sin agregar un tiempo de slack adicional.

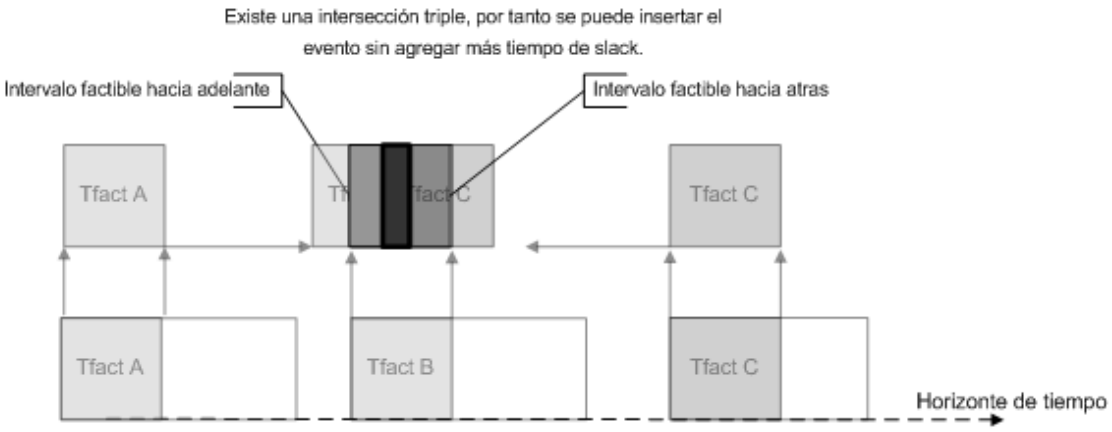


Ilustración 6-12 Proceso de inserción - Intersección "triple" no vacía

Finalmente realizado el algoritmo de inserción la propuesta generada será enviada al agente Dock Planner que las evaluara y aceptara aquella con la mayor minimización de tiempo de espera de los barcos.

6.1.2 Visualización 3D de sistema multiagente aplicado al problema de transporte de pasajeros

En esta solución se realiza una interfaz en tres dimensiones que busca poder ver la representación gráfica de cada uno de los agentes y observar cómo interactúan entre ellos para su comprensión para profesionales como fines académicos.

En ese caso se seleccionó el sistema multiagente “Sistema para el Dynamic Dial-A-Ride Problem (D-DARP)”**¡Error! No se encuentra el origen de la referencia.** que trata sobre como los sistemas multiagente pueden ayudar a regular las rutas, los tiempos de

salida, los vehículos e incluso los operadores para ajustarse a la demanda de los sistemas de transporte inteligentes de pasajeros.

Para el desarrollo de la aplicación se utilizó la metodología PASSI para el desarrollo de los agentes los cuales interactuaban bajo la plataforma de JADE, es por ello que para la interfaz 3D fue utilizado JMonkey Engine.

A continuación se mostrara el diagrama de identificación de agentes más el visualizador 3D utilizado en su solución en la Ilustración 3.2 y en la Ilustración 3.3 la interfaz utilizada.

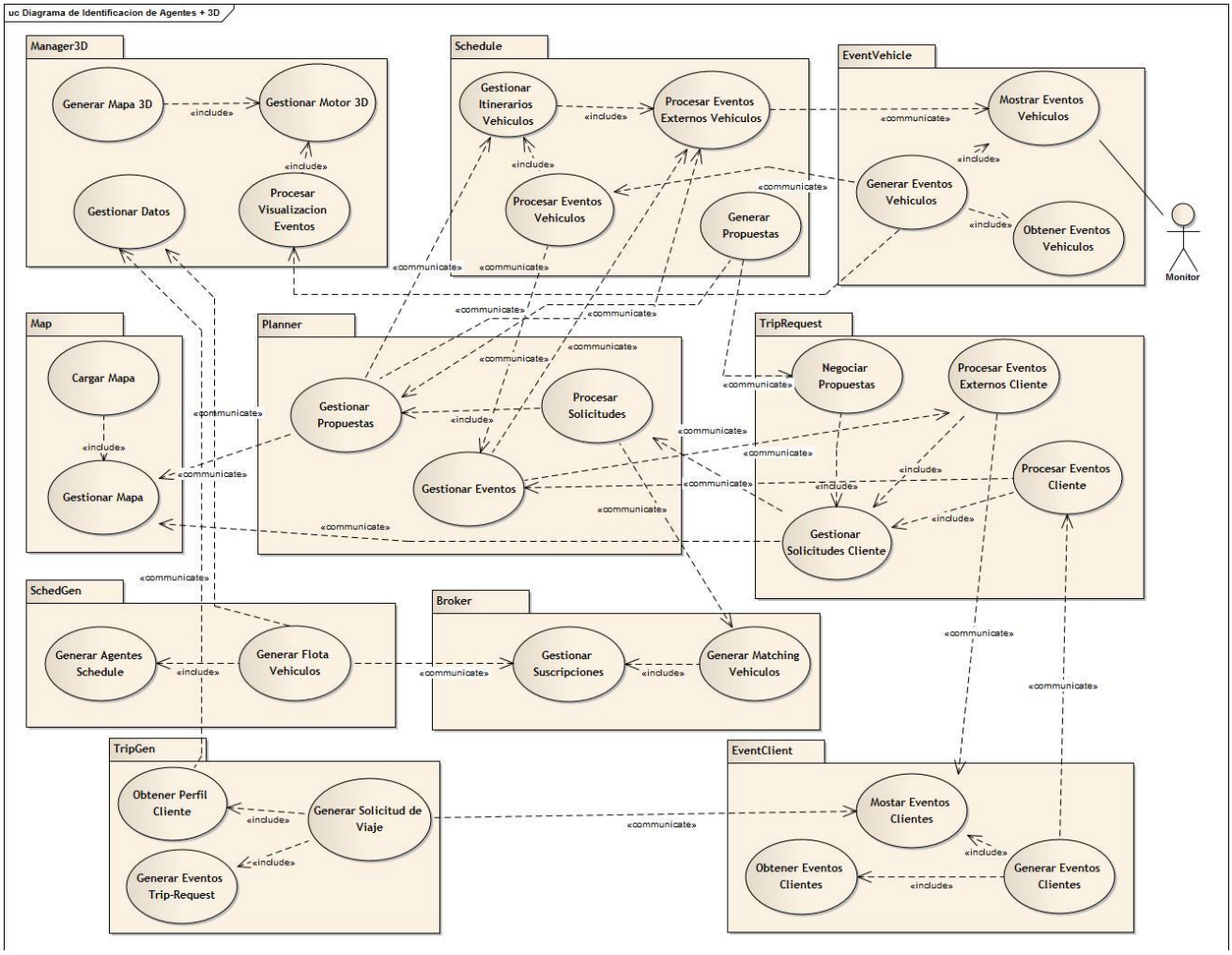


Ilustración 6-13 Diagrama Identificación de Agentes + Manager 3D por Felipe Baranlloni **¡Error! No se encuentra el origen de la referencia.**

La solución fue creada en base al sistema de transporte realizado en "Desarrollo de un sistema multiagente para el Dynamic Dial-a-Ride Problem, utilizando Tecnología de agentes"**¡Error! No se encuentra el origen de la referencia.**, donde luego de analizado el problema fue decidido utilizar un nuevo agente que se encontraría en la capa de interfaz debido a que se comunicaría principalmente con agentes de niveles altos, que es donde se crean las flotas de transporte y los perfiles de clientes.

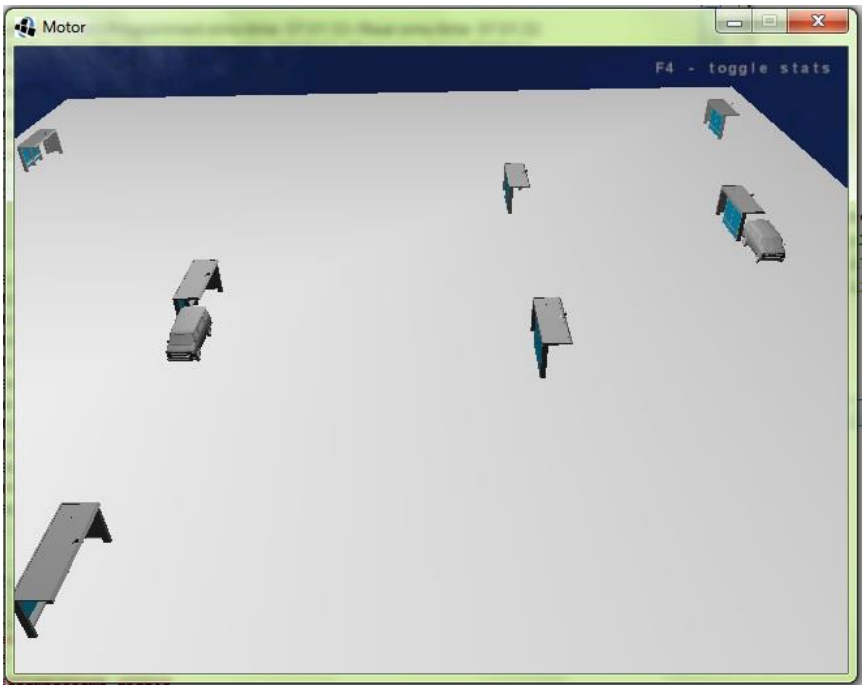


Ilustración 6-14 Interfaz Visualizador 3D por Felipe Baranlloni **¡Error! No se encuentra el origen de la referencia.**

En esta solución no se tomó mayormente en cuenta los otros agentes, debido a la no relevancia del método de resolución del problema.

Esta interfaz permite mostrar los detalles de los buses, detalles de los clientes y opciones de cambio de cámara como visualizar un bus especifico o visualizar todo el entorno.

El Visualizador inicialmente configura el VSync, luego carga las texturas y después por cada paradero que se utilizará carga el modelo del paradero, lo clona cuantas veces sea necesario y los ubica en su posición. Luego carga el modelo base y agrega los nodos. Crea los puntos de luz configurando su posición y color y agregando su nodo. Crea el cielo, carga sus texturas, rotación y posición, donde luego después también agrega el nodo. Finalmente Inicia el ciclo del motor donde por cada posición actualizada del vehículo se actualiza el frame de animación por siempre.

7 Desarrollo de la solución

En este capítulo se describirán las metodologías a utilizar para la solución y tecnologías correspondientes y diagramas usados.

7.1 Metodología Multiagente a utilizar.

Para el desarrollo de la solución se utilizará la metodología PASSI para la creación de los diagramas correspondientes.

En PASSI es utilizada la notación UML que es la más ampliamente aceptada en el ámbito académico e industrial. Con esto permite una rápida adopción de la metodología orientada a agentes y evita el aprendizaje desde cero en un nuevo lenguaje.

7.2 Sistema Multiagente

El sistema multiagente a utilizar estará basado en la solución propuesta por René Díaz**¡Error! No se encuentra el origen de la referencia.** para el problema de Berth Allocation Problem, donde se modificará y agregarán los agentes necesarios para la comunicación y manejo de la interfaz 3D.

7.3 Tecnología Interfaz 3D

Para la visualización de la interfaz en tres dimensiones se utilizará el motor JMonkey Engine que al estar construido completamente en JAVA y entregar un herramienta de desarrollo totalmente compatible la plataforma multiagente JADE, lo convierte en la mejor opción disponible.

Como herramienta de modelado 3D, se utilizará Blender que es una herramienta totalmente gratuita, y suficiente para el correcto diseño de los modelos. Además JMonkey Engine es compatible con esta herramienta de modelado.

7.4 Consideraciones realizadas

En el proceso de prueba del anterior sistema fueron detectados unos problemas importantes en el software. Los diagramas creados para el desarrollo del software están incompletos en lo que a aspectos más específicos se refiere, por lo que es necesario crear esos diagramas para la mejor interoperatibilidad. Además se encontraron los siguientes problemas específicos.

- El sistema no considera ninguno de los actores mostrados en su diagrama de agentes, por lo que se decidió eliminarlos, y crear un solo actor principal que interactuara con el sistema.

- El sistema no maneja eventos como la cancelación de un barco, por lo que es necesario crear los diagramas faltantes y agregar su código correspondiente.
- El sistema no maneja ningún archivo para la creación de los barcos y sitios, solo se encuentra incrustado en el código su único caso, por lo que se creará una interfaz donde se manejarán todos estos aspectos.
- La salida de los datos se imprimen por orden de llegada y no al agente correspondiente, por lo que se arreglará este problema en el código y será guardado en alguna estructura de dato.
- Se decidió que el sistema solo manejará un número fijo de sitios de atraque el cual solo se podrá disminuir debido a la naturaleza propia poco cambiante en la realidad de este aspecto y la difícil creación de modelos que aceptarán este tipo de requerimientos.

7.5 Diagramas de caso de uso

Para explicar las distintas opciones de la interfaz y además ser usado como guía para el resto de los diagramas, fueron creados los siguientes casos de uso.

La interfaz poseerá dos estados, el inicial anterior al inicio de la simulación y el final mientras corre la simulación.

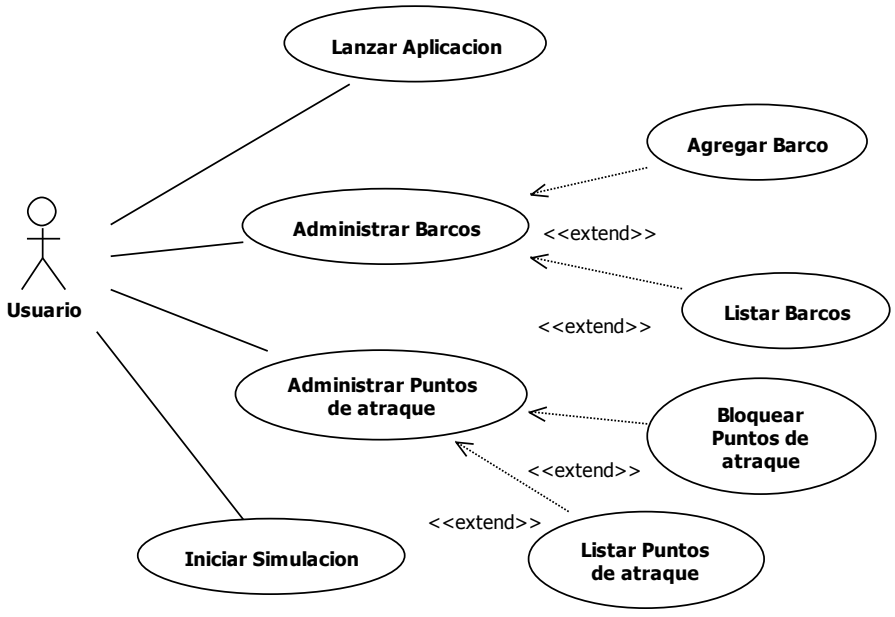


Ilustración 7-1 Diagrama de caso de uso interfaz inicial

En este primer caso de uso inicial se pueden apreciar las distintas opciones que poseerá el usuario antes de correr la simulación, entre ellas se encuentran: agregar barcos al sistema, listar los barcos actualmente agregados al sistema, comprobar los puntos de atraque disponibles y la posibilidad de bloquearlos. Una vez terminado de configurar inicialmente el sistema se puede proceder a iniciar la simulación.

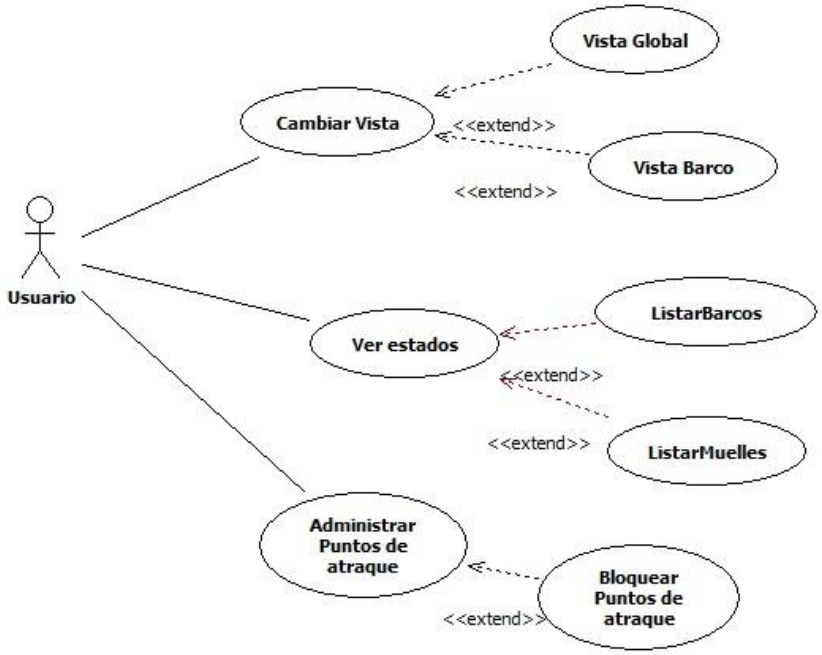


Ilustración 7-2 Caso de uso interfaz final

Una vez iniciado el sistema las opciones cambiaran a las mostradas en el caso de uso de la Ilustración 7.2, en este caso serán: Cambiar la vista a la vista global o a un barco en específico, ver los estados actuales de los barcos que reflejan si ya fueran asignados o están esperando su turno.

7.6 Diagrama de Actividades de la Interfaz 3D

En el siguiente diagrama de actividades se aprecia el flujo de datos habitual de la clase Simulador 3D donde se realizan las tareas iniciales que preparan el motor 3D para su utilización con el usuario en el entorno simulado.

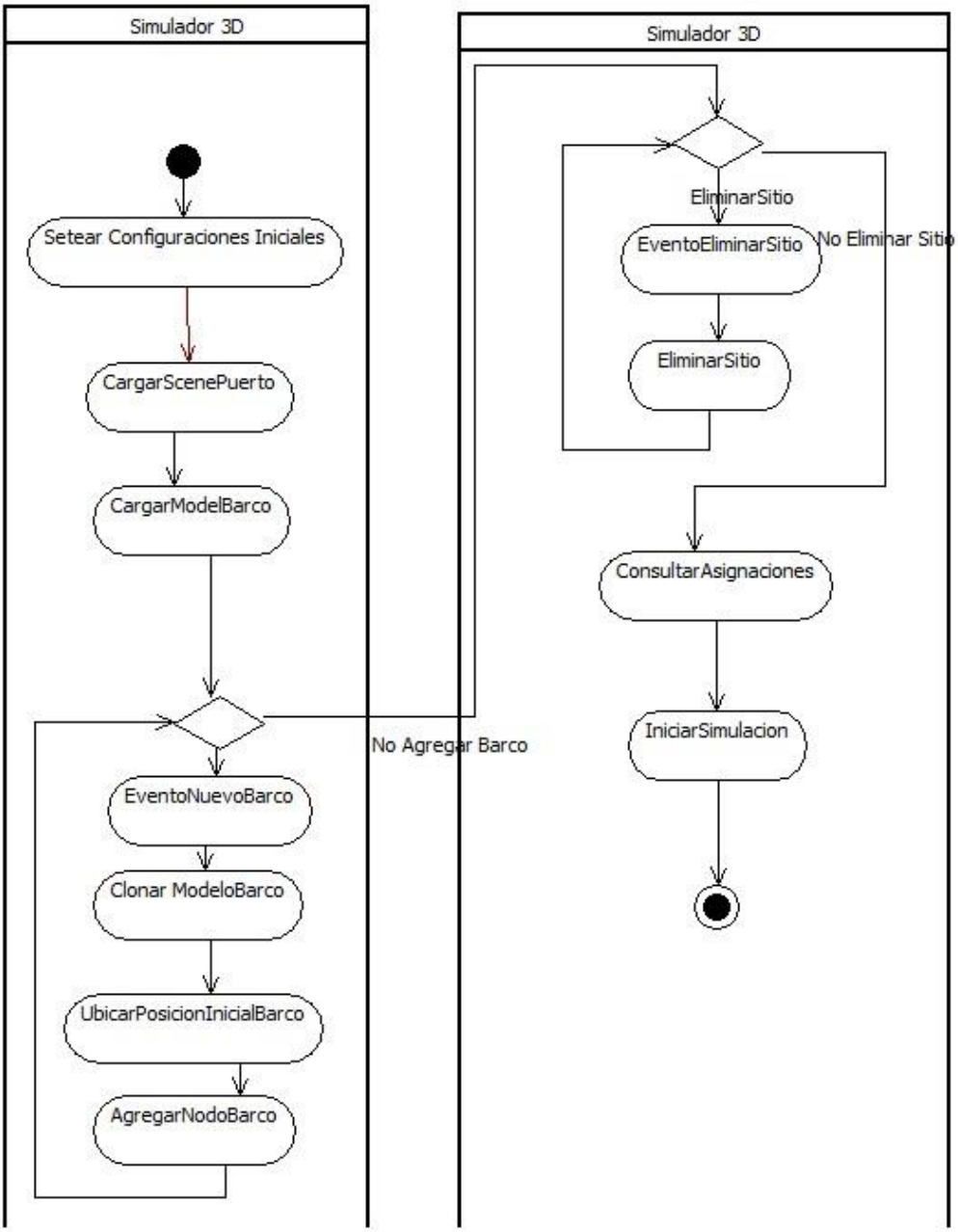


Ilustración 7-3 Diagrama de Actividad

Inicialmente se configuran las opciones básicas relacionadas con la visualización del entorno 3D, como el VSync, resolución a utilizar y el nombre de las ventanas, luego se configura la capa de la GUI con la que interactúa el usuario, configurando sus distintos paneles, botones y cajas. Luego se carga la escena inicial que contiene el modelo del terreno sus texturas, las texturas del skybox, las luces correspondientes y el modelo del muelle con sus texturas. Cada modelo incluido los barcos y muelles ya poseen sus texturas asociadas, por lo que no es necesario tener que cargar sus texturas separadamente. Luego se carga el modelo del barco, que por medio de la interfaz utilizada clonando el modelo la cantidad de veces que sea necesario para luego ubicarlo en su posición y agregarlo al nodo raíz para ser

mostrado en la interfaz, también se pueden configurar la cantidad de sitios iniciales en los que puede atracar los barcos, inicialmente pueden existir 4 sitios para atracar y solo se pueden cerrar por limitación de los modelos del muelle.

Finalmente una vez realizada todas las configuraciones iniciales se consulta la asignación asociada y se inicia la simulación.

7.7 Requerimientos del Sistema

Esta Fase es utilizada para reconocer las distintas tareas que se deben realizar, y los agentes involucrados en el proceso.

7.7.1 Descripción del Dominio

Debido a la existencia de un diagrama de identificación de agentes, se decidió partir por este, por la mayor facilidad de entender las tareas actuales y agregar las nuevas.

7.7.2 Identificación de Agentes

En el siguiente diagrama Ilustración 7.4 se asocian todas las tareas a realizar por el sistema a un agente.

El agente Simulador 3D se encarga de administrar todos los asuntos relacionados con los elementos 3D y la interacción con el usuario, para ello se comunica con el agente Barco y el agente Berth para reflejar las opciones seleccionadas por el usuario con el sistema, estos agentes a su vez se comunican a su vez con los agentes Berth Planner y Berth Request, que por medio del agente Dock Planner se genera la mejor asignación creada para la solución, la cual es informada al agente Dock central, el cual es el agente al que le consulta la Simulador 3D para mostrar los elementos en pantalla.

A continuación un detalle de lo que realiza cada agente por separado.

Agente Simulador 3D: Este agente será el que se encarga de interactuar con el usuario y que manejará el motor 3D del mismo. Esto incluye inicializarlo con todas sus opciones, creación de entornos iniciales y carga de los modelos, además del inicio de la simulación cuando el usuario lo pida. El Agente se encarga de comunicarse con los agente Barco y Berth para informar las tareas de Inscribir Nuevo Barco Nuevo, Cancelar Petición y Eliminar Sitio, que luego consultara al Dock Central por los registros de los barcos, sitios y asignaciones.

Agente Barco: Se encarga la administración de los barcos. En cuanto recibe una comunicación desde el agente Simulador 3D, que pueden ser inscribir nuevo barco o cancelar barco, este se comunica con el agente Berth Request que es el que se encarga de administrar la petición. Además se comunica con el agente Dock Central para guardar el registro de nuevos barcos.

Agente Berth Request: Se encarga de recibir las comunicaciones del agente Barco y genera las peticiones, al agente Dock Planner, para la asignación del barco en un punto de atraque.

Agente Berth: Se encarga de la administración de los sitios como puntos de atraque del muelle. Recibe una comunicación desde la Simulador 3D para la eliminar un sitio el cual esté le comunica al agente Dock Central y al agente Berth Planner para que genere las propuestas con los sitios existentes.

Agente Berth Planner: Este agente se encarga de recibir las comunicaciones desde el agente Berth, para después poder generar las propuestas solicitadas por el agente Dock Planner.

Agente Dock Planner: Se encarga de recibir las peticiones del agente Berth Request y solicitar las propuestas al agente Berth Planner. Luego rankeas las propuestas para seleccionar la mejor y finalmente enviar al agente Dock Central el registro de las asignaciones finales. En el caso de recibir un evento luego de iniciada la simulación, calcula una nueva asignación con las nuevas propuestas y luego las informa al agente Dock Central.

Agente Dock Central: Se encarga de recibir las asignaciones que poseerán los barcos con los sitios de atraques, guardar los nuevos barcos y guardar los sitios disponibles. El agente Simulador 3D le consulta todas asignaciones, barcos y sitios para desplegarlos en pantalla.

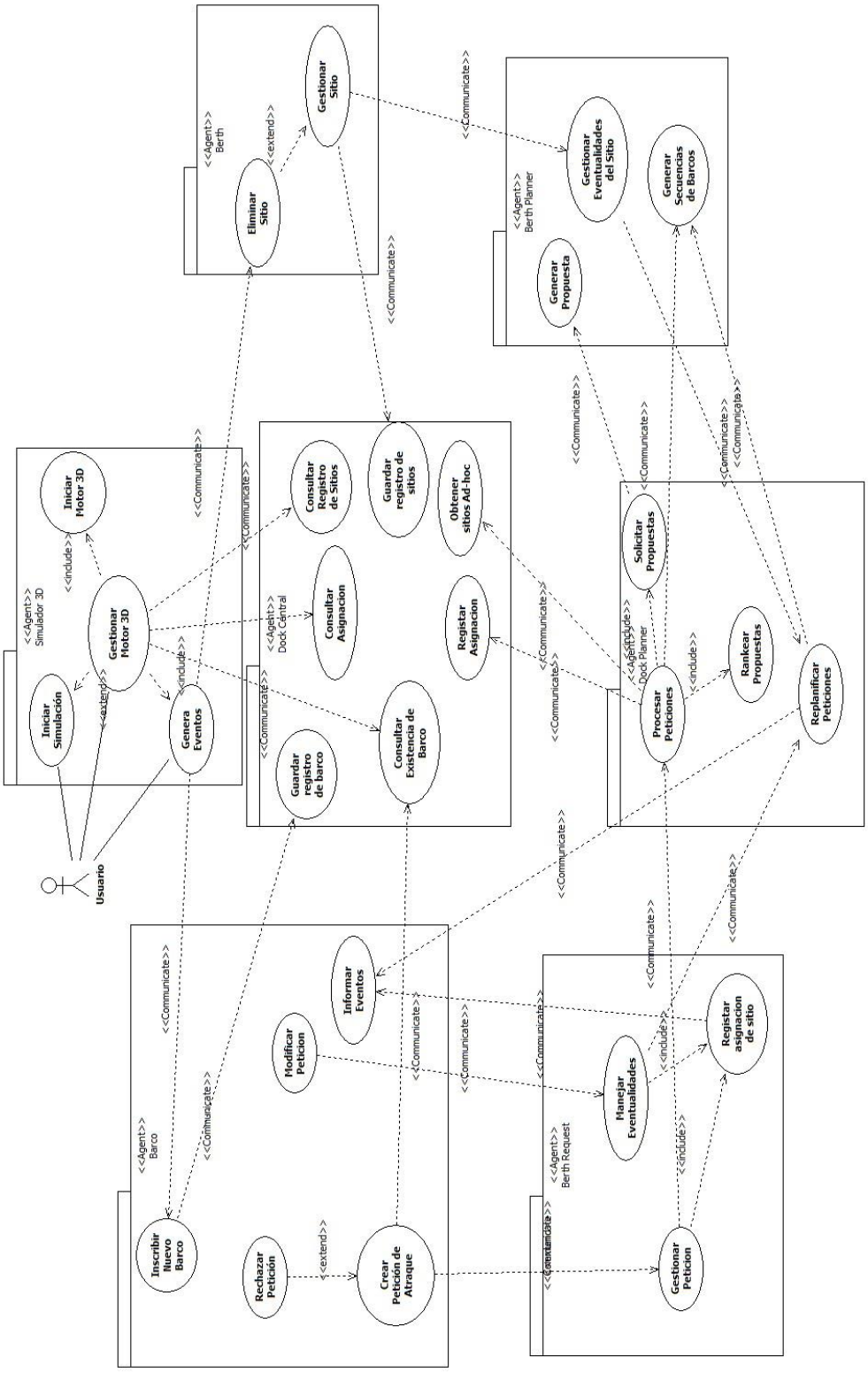


Ilustración 7-4 Identificación de agentes

7.7.3 Identificación de Roles

En esta etapa obtiene una primera aproximación de los distintos roles que poseen los agentes y cómo interactúan entre ellos para realizar un proceso determinado.

7.7.3.1 Ingresar nuevo barco

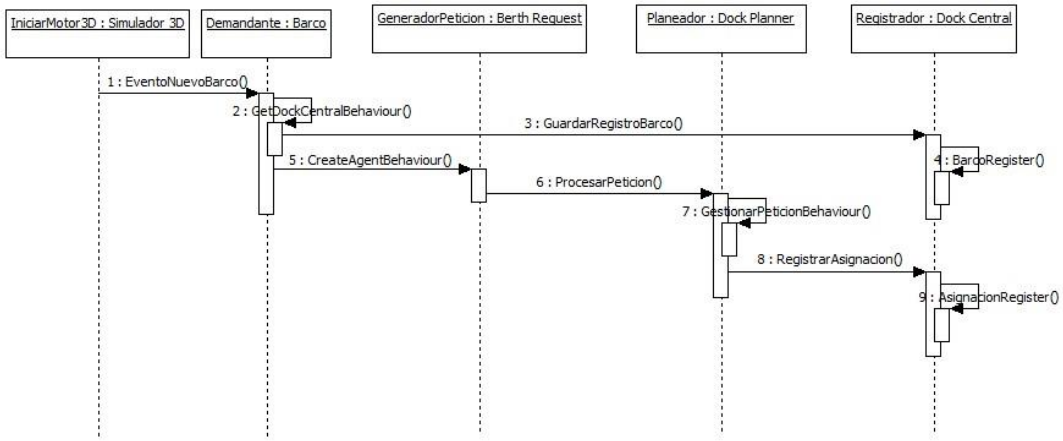


Ilustración 7-5 Ingresar nuevo barco

En el proceso de ingresar un nuevo barco al agente Simulador 3D, se utiliza el behaviour EventoNuevoBarco() para que el agente Barco en el rol demandante, pida el registro del nuevo barco en el Dock Central y además Iniciar la petición por medio del Berth Request para la solicitud de una asignación a algún muelle.

7.7.3.2 Listar barcos

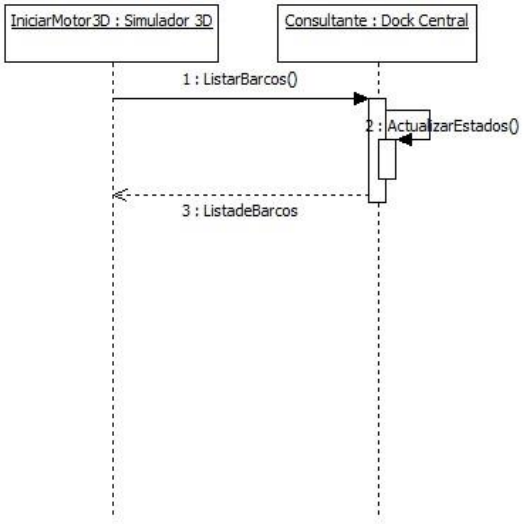


Ilustración 7-6 Listar barcos

Para el proceso de listar barcos solo existen dos agentes involucrados, las cuales son Simulador 3D y el Dock Central. El Simulador 3D pide los datos de los barcos almacenados en el Dock Central para será mostrados al usuario para su mejor comprensión de los barcos

existentes. El Dock Central en este caso asume el rol de responder las consultas que otros agentes podrían tener.

7.7.3.3 Consultar punto de atraque

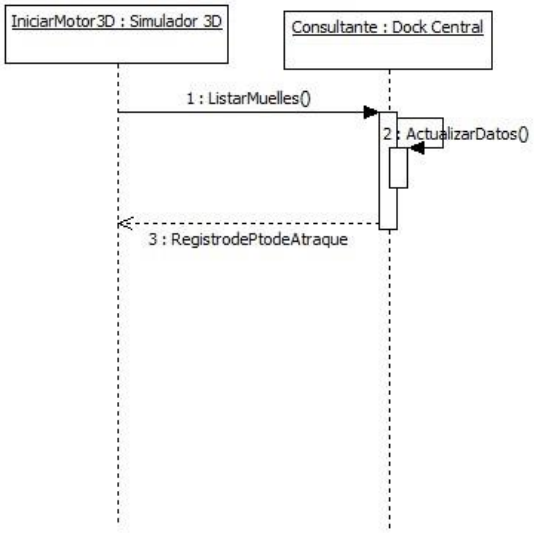


Ilustración 7-7 Consultar punto de atraque

Para el proceso de consultar por los puntos de atraque existentes, el Simulador 3D pide la lista de sitios al Dock Central, y este se la entrega en su rol de Consultante.

7.7.4 Especificación de Tareas

Este tipo de diagrama es utilizado para la comprobar cómo las distintas tareas se comunican entre sí entre los distintos agentes, independiente de los roles que pueda tener este agente en específico.

En este caso debido a que el proyecto está orientado al Simulador 3D del sistema, solo se considera este caso, ya que los otros diagramas orientados a la planificación de las asignaciones corresponden al proyecto en que este está basado.

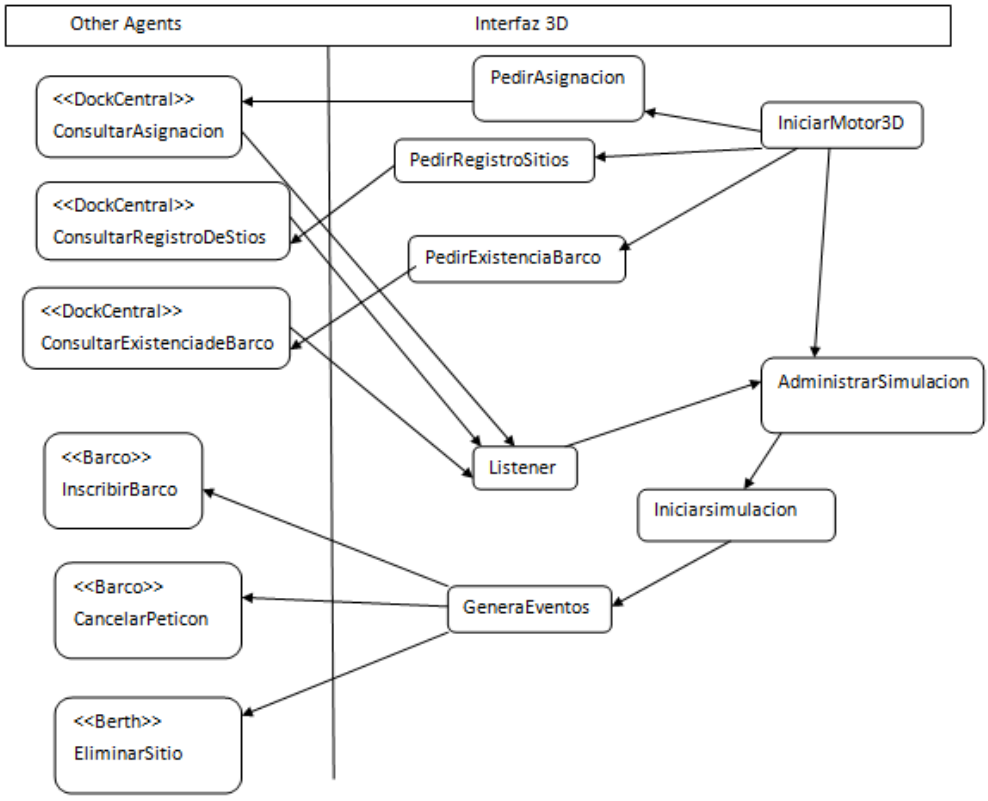


Ilustración 7-8 Especificación de tareas

Como se puede apreciar, el agente Interfaz 3D se relaciona con todos los agentes, excepto a los correspondientes a la capa de planeación.

El agente Interfaz 3D inicia el motor 3D y empieza a pedir los registros de los barcos, sitios y asignaciones realizadas al Dock Central.

Luego de iniciada la simulación desde la interfaz 3D se pueden generar eventos, que pueden ser inscribir barco y cancelar petición en el agente Barco y eliminar un sitio en el agente Berth.

7.8 Sociedad de Agentes

En esta fase se concentra la atención en cómo se comunican los agentes en su conjunto definiendo los distintos roles que poseen e interactúan, además de los conceptos en común que utilizan para comunicarse y los protocolos de comunicación utilizados.

7.8.1 Descripción de roles

En la descripción de roles se aprecia los distintos roles que pueden poseer cada uno de los agentes involucrados en el sistema y como se comunican entre ellos.

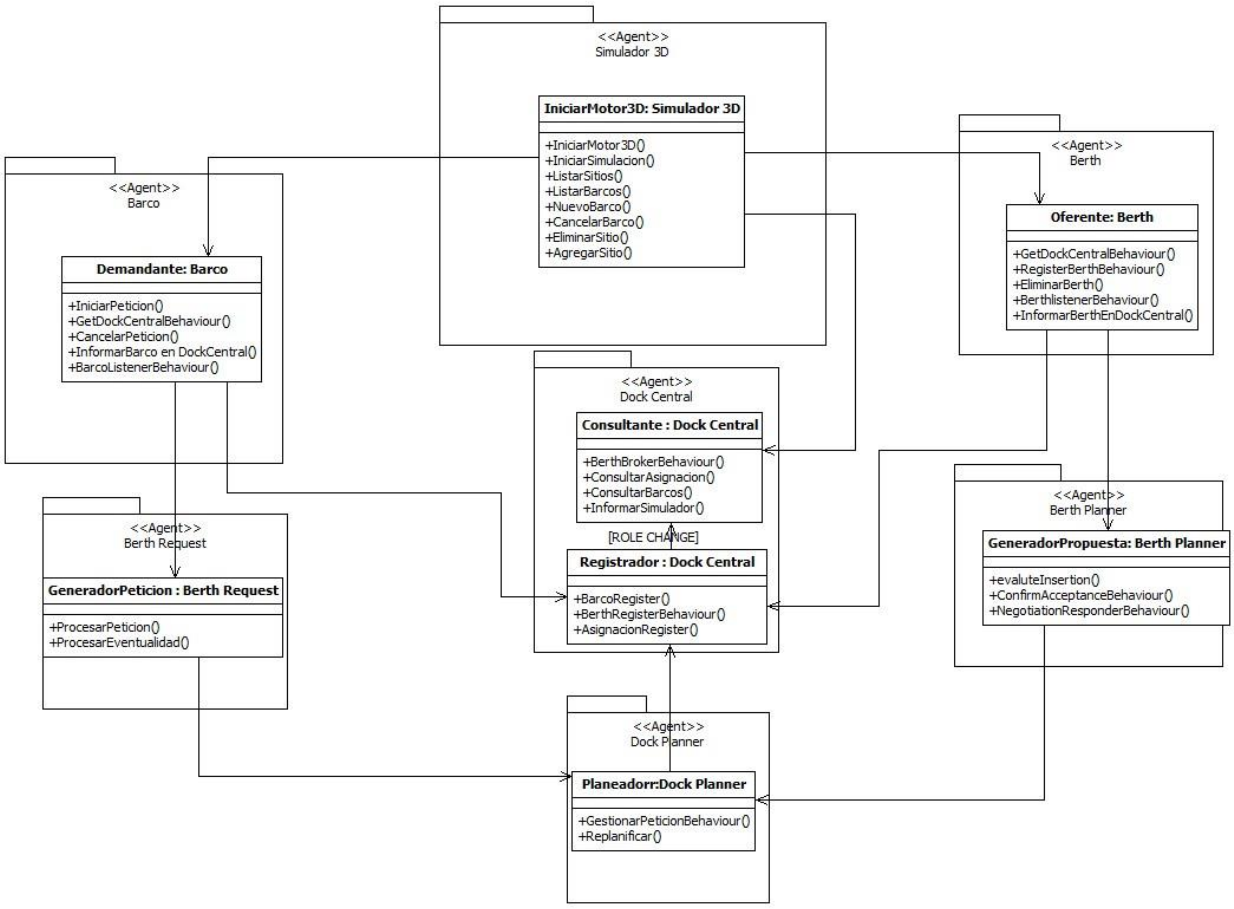


Ilustración 7-9 Descripción de roles

El Simulador 3D posee el rol de IniciarMotor3D que interactúa con los roles de los agentes Barco, Berth y el rol de Consultante del agente Dock Central. En este rol el agente se preocupa de iniciar el motor 3D, ingresar los nuevos barcos y los sitios que estarán disponibles antes de iniciar la simulación.

El agente Barco tiene asociado solo un rol, que es el demandante el cual recibe comunicaciones desde los roles IniciarMotor3D y GeneradorEventos desde el agente Simulador 3D y este se comunica con el rol GeneradorPeticion de Berth Request para el inicio de las peticiones para la planificación de su asignación. Además se comunica con el rol Register del Dock Central para su registro como nuevo barco.

El Agente Berth de igual manera, posee solo un rol correspondiente a oferente que recibe peticiones desde los dos roles del agente Simulador 3D y además se comunica con rol Register del Dock Central para registrar los sitios disponibles y el rol de GeneradorPropuesta del Berth Planner que es el que se encarga de generar propuestas para las peticiones de atraque de los barcos.

El agente Dock Central posee 2 roles correspondientes a Register y Consultante. El primero se encarga de registrar todos los barcos existentes, sitios disponibles y asignaciones realizadas. En su rol de Consultante, recibe las peticiones de otros agentes acerca de consulta de sus registros correspondientes.

El agente Berth Request posee un único rol de GeneradorPeticion. Este único rol recibe una comunicación desde el agente Barco y se comunica con el rol de Planeador del agente Dock Planner para las planificaciones.

El agente Berth Planner posee un único rol de GeneradorPropuesta que recibe una comunicación desde el agente Berth y se comunica con el rol de Planeador del agente Dock Planner para entregar sus propuestas.

El agente Dock Planner posee un único rol de Planeador que recibe las peticiones desde el agente Berth Request y las propuestas Berth Planner para la asignación de barcos con los sitios e informa al rol Register del Dock Central de la asignación realizada.

7.8.2 Descripción de Ontologías

En la descripción de ontologías se definen los conceptos que se utilizan en la comunicación de los agentes involucrados en el sistema. Este diagrama fue propuesto originalmente por René Díaz **¡Error! No se encuentra el origen de la referencia.** y debido a que cumple con todos los requerimientos, solo fue modificado ligeramente.

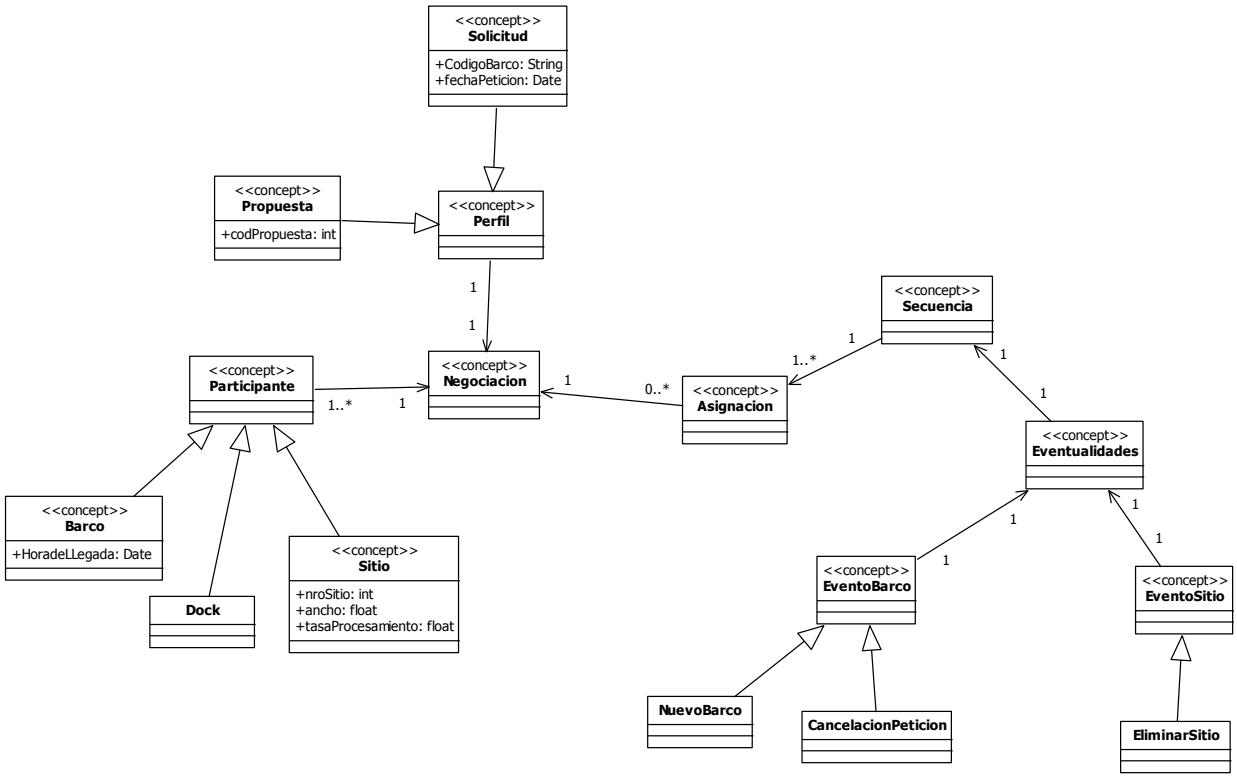


Ilustración 7-10 Descripción de ontologías

En la negociación para la asignación de barcos a sitios de atraque, puede haber más de un participante, las cuales pueden ser: Barco, Dock, Sitio. La negociación maneja inicia con una solicitud, a la cual se le entrega una propuesta para cumplirla. En el caso de un evento

en los barcos o sitios, es necesario hacer una nueva negociación lo que implica una nueva asignación y secuencia.

7.8.3 Descripción de protocolos

En el caso específico de este proyecto donde el énfasis se encuentra en el agente Interfaz 3D el protocolo utilizado es FIPA Request, debido a que solo solicita que el resto de los agentes realicen una acción específica, el cual ellos podrían rechazar por algún motivo.

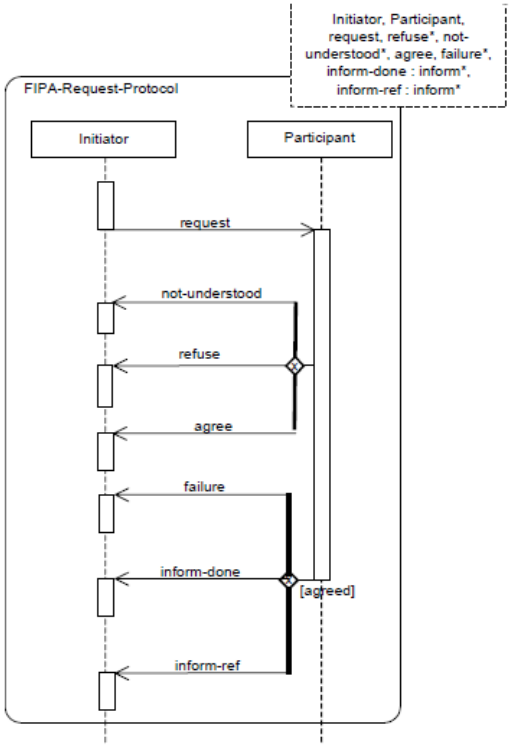


Ilustración 7-11 Fipa-Request

En el caso de los agentes asociados a la resolución del problema de Berth Allocation Problem, el protocolo utilizado es:

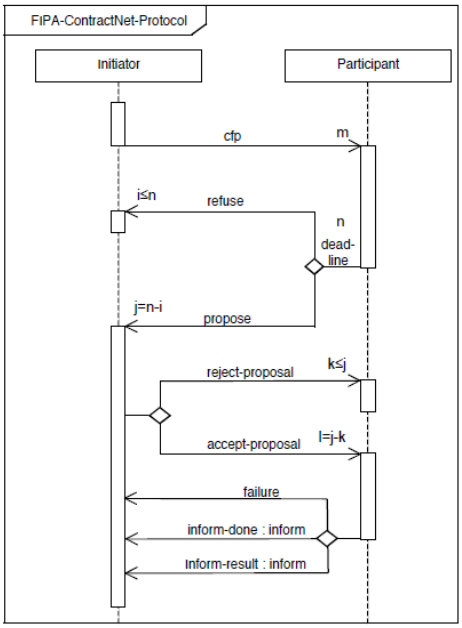


Ilustración 7-12Fipa-Contract Net

Donde el agente iniciador llama a un cfp y los participantes envían propuestas donde el agente iniciador elige una entre ellas, la cual acepta en base a algún criterio y rechaza las demás.

7.9 Implementación de Agentes

En esta fase del modelado, los agentes son descritos en términos de clases y métodos, lo cual lo convierte en una fase muy cercana al código.

7.9.1 Definición de estructura de Agentes

En la definición de estructura de los agentes se encuentra en dos fases, la primera una definición de la estructura del sistema multiagente en su totalidad, y la segunda fase donde se analiza a cada agente por separado.

7.9.1.1 Definición de estructura multiagente

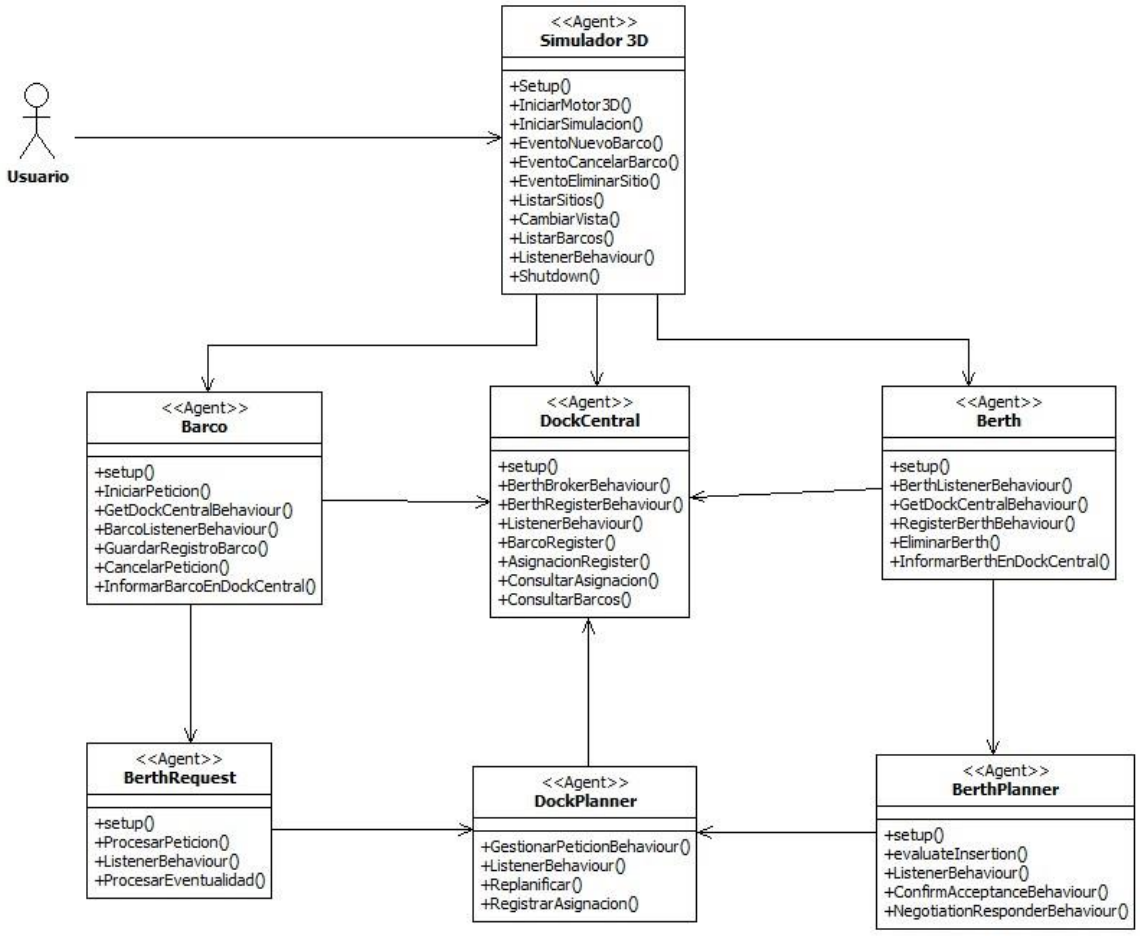


Ilustración 7-13 Definición estructura multiagente

En la definición de estructura multiagente se puede apreciar cada uno de los métodos asociados a cada agente y como se comunican entre ellos.

7.9.1.2 Definición de estructura del agente Simulador 3D

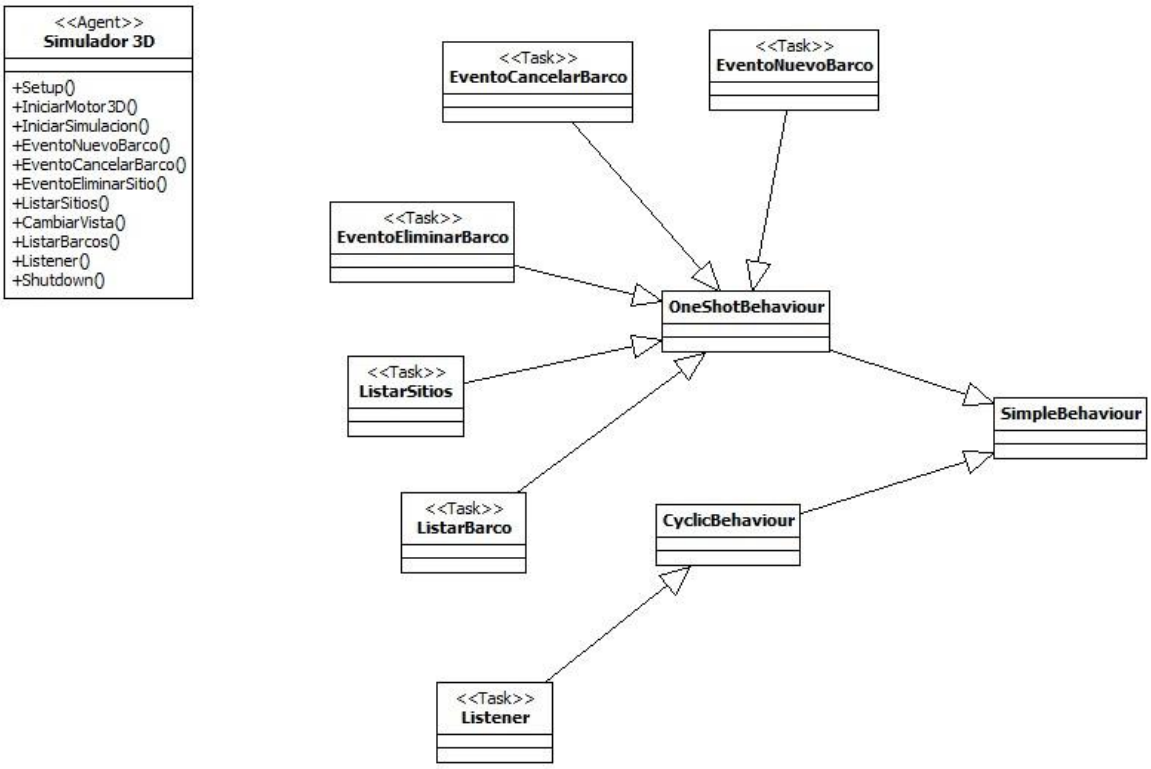


Ilustración 7-14 Definición estructura Simulador 3D

En el agente Simulador 3D se aprecian el tipo de behaviour que utiliza el agente para cada una de sus tareas. Listener es un CyclicBehaviour debido a su necesidad de estar recibiendo todos los mensajes que reciba, luego según el tipo de mensaje invoca otro de los behaviour.

7.9.1.3 Definición de estructura del agente Barco

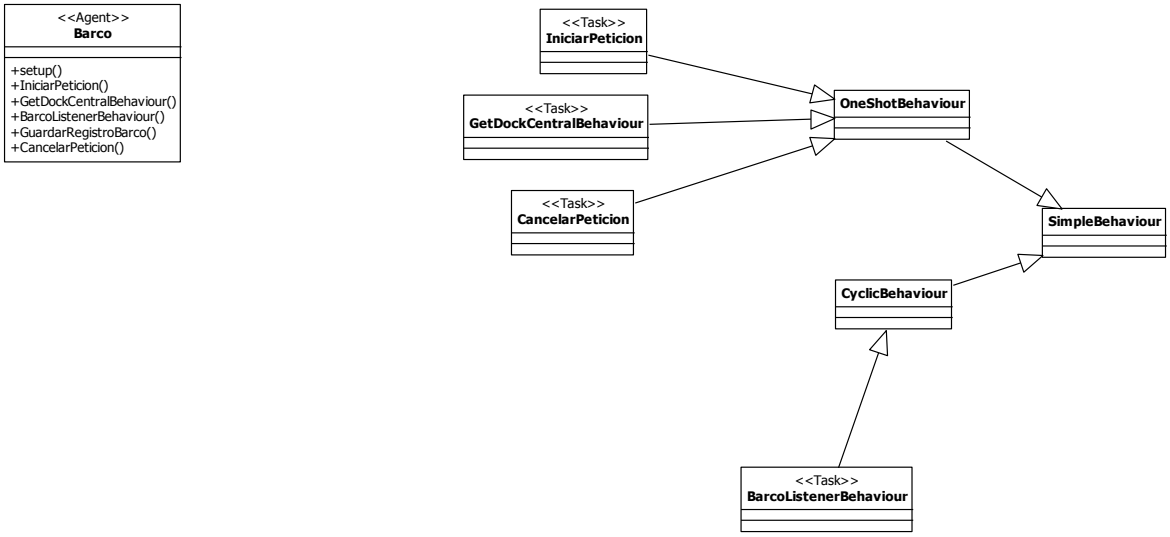


Ilustración 7-15 Definición estructura Barco

En el agente Barco, BarcoListenerBehaviour es un CyclicBehaviour debido a que se encarga de recibir todos los mensajes entrantes al agente y luego invoca al behaviour correspondiente para responder.

7.9.1.4 Definición de estructura del agente Berth

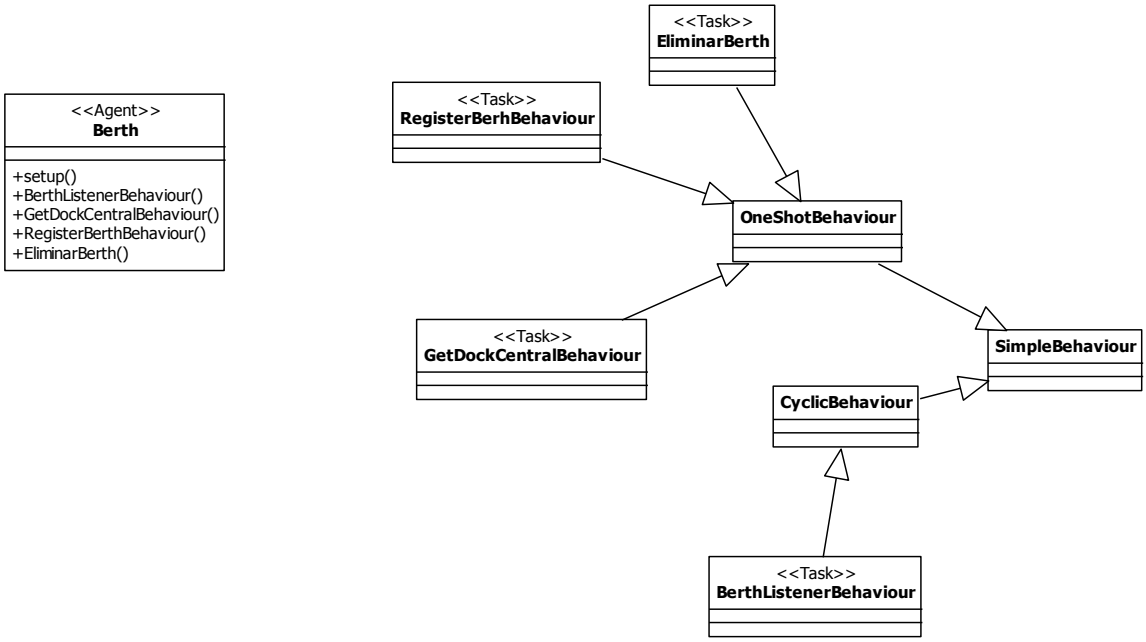


Ilustración 7-16 Definición estructura Berth

En el agente Berth, BerthListenerBehaviour es un CyclicBehaviour debido a que se encarga de recibir todos los mensajes entrantes al agente y luego invoca al behaviour correspondiente para responder.

7.9.1.5 Definición de estructura del agente Berth Planner

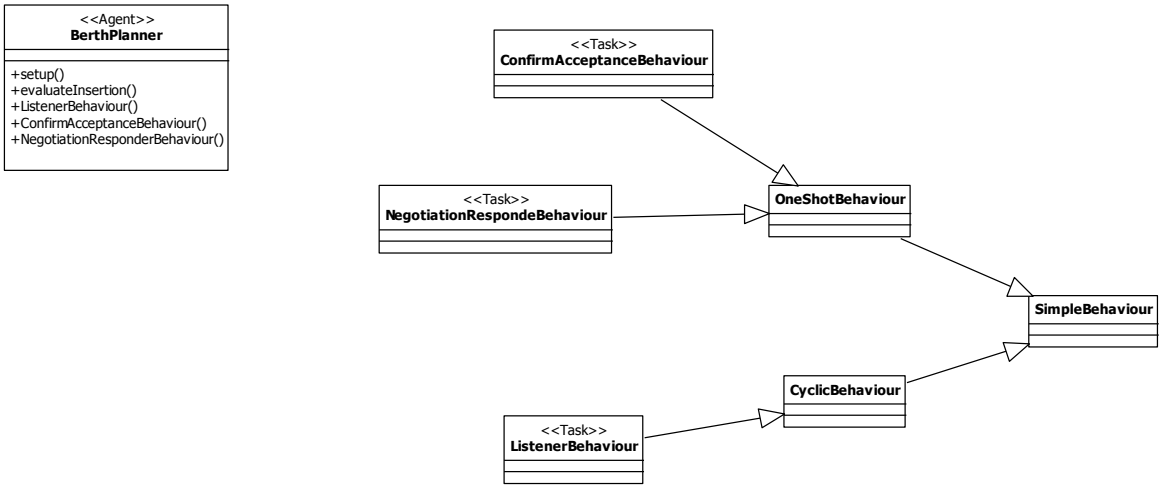


Ilustración 7-17Definición estructura Berth Planner

En el agente Berth Planner, ListenerBehaviour es un CyclicBehaviour debido a que se encarga de recibir todos los mensajes entrantes al agente y luego invoca al behaviour correspondiente para responder.

7.9.1.6 Definición de estructura del agente Berth Request

Ilustración 7-18 Definición estructura Berth Request

En el agente Berth Request, ListenerBehaviour es un CyclicBehaviour debido a que se encarga de recibir todos los mensajes entrantes al agente y luego invoca al behaviour correspondiente para responder.

7.9.1.7 Definición de estructura del agente Dock Planner

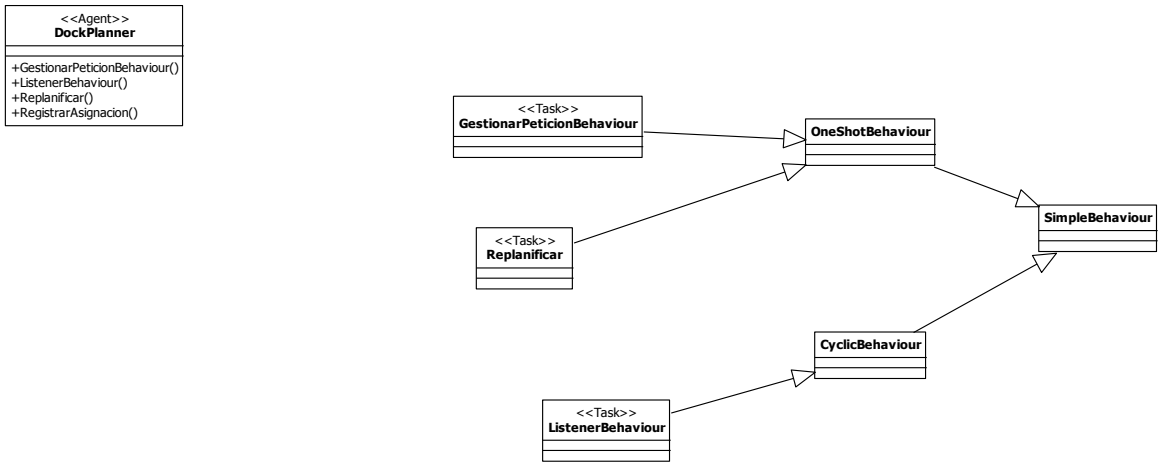


Ilustración 7-19 Definición estructura Dock Planner

En el agente Dock Planner, ListenerBehaviour es un CyclicBehaviour debido a que se encarga de recibir todos los mensajes entrantes al agente y luego invoca al behaviour correspondiente para responder.

7.9.1.8 Definición de estructura del agente Dock Central

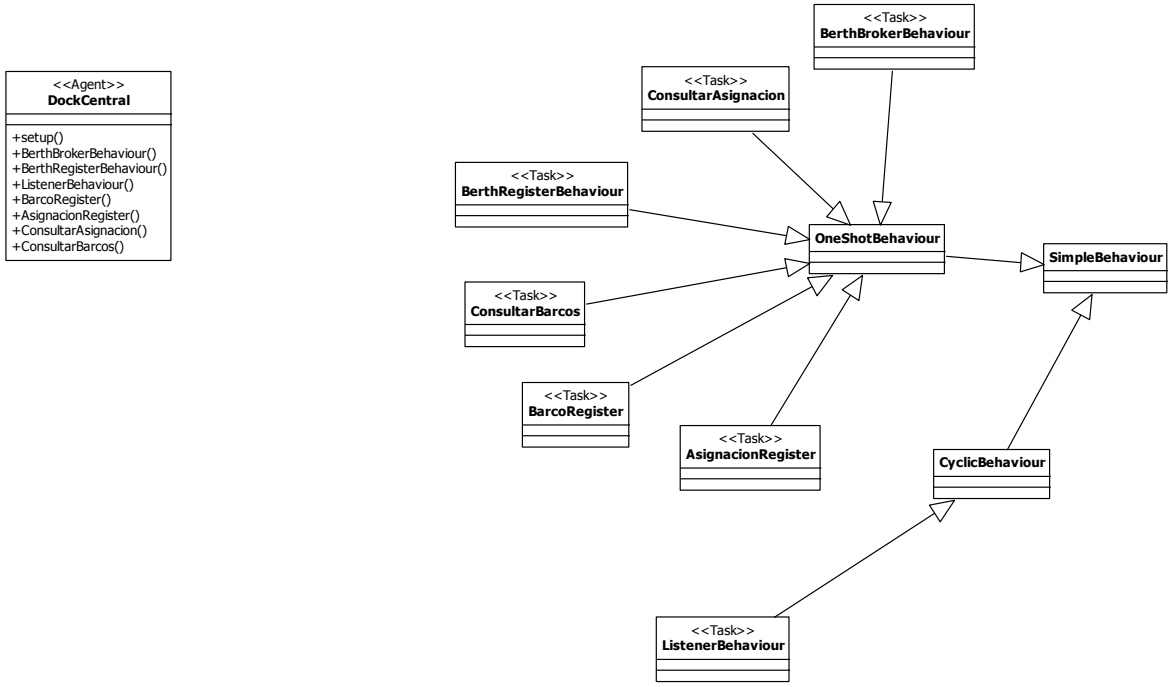


Ilustración 7-20 Definición estructura Dock Central

En el agente Dock Central, ListenerBehaviour es un CyclicBehaviour debido a que se encarga de recibir todos los mensajes entrantes al agente y luego invoca al behaviour correspondiente para responder.

8 Implementación del sistema

8.1 Interfaz del sistema

En base al sistema creado en el capítulo anterior, por medio de PASSI y casos de uso, se presenta el sistema actual en desarrollo con sus principales ventanas y métodos utilizados.

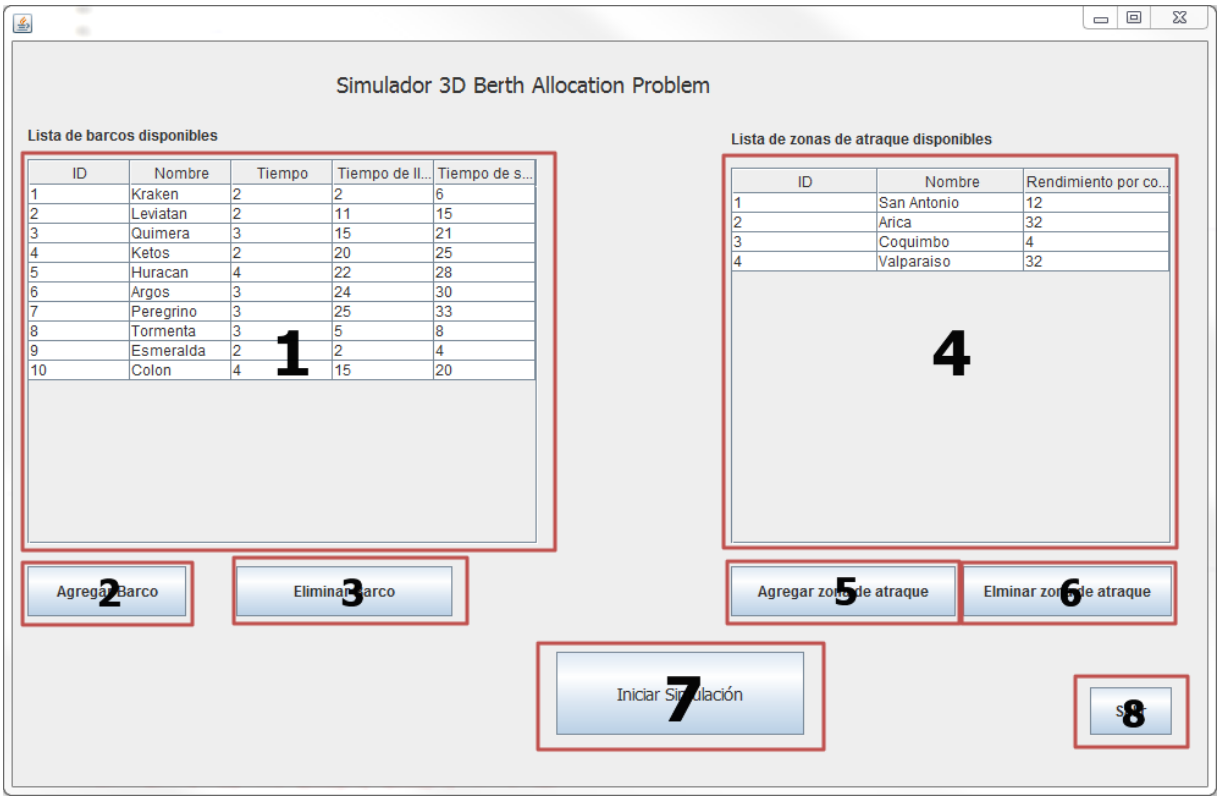


Ilustración 8-1Gui Inicial Simulador 3D

La interfaz mostrada en la Ilustración 8.1, corresponde a la primera ventana que aparece al iniciar la aplicación la cual muestra la lista de barcos disponibles y las zonas de atraque disponibles. Estos datos son leídos de un archivo de texto asociados a los barcos y un archivo de texto asociado a las zonas de atraque, además existen botones para agregar por medio de la misma interfaz barco y zonas de atraque, además de eliminar y la posibilidad de modificar los distintos datos que se entregan.

1. Muestra la información de los barcos ingresados en el sistema con cada uno de sus datos.
2. Botón que abre una ventana(Ilustración 4-2) para el ingreso de la información de un nuevo barco
3. Botón que elimina el barco seleccionado en el cuadro 1
4. Muestra información correspondiente a las zonas de atraque ingresadas en el sistema.
5. Botón que abre una ventana para el ingreso de la información de una nueva zona de atraque.
6. Botón que elimina la zona de atraque seleccionada.
7. Abre la ventana que muestra la simulación realizada con los datos ingresados en el sistema.
8. Finaliza la aplicación.

A continuación son mostradas las ventanas utilizadas para agregar barcos en la Ilustración 8.2 y agregar zona de atraque en la Ilustración 8.3.

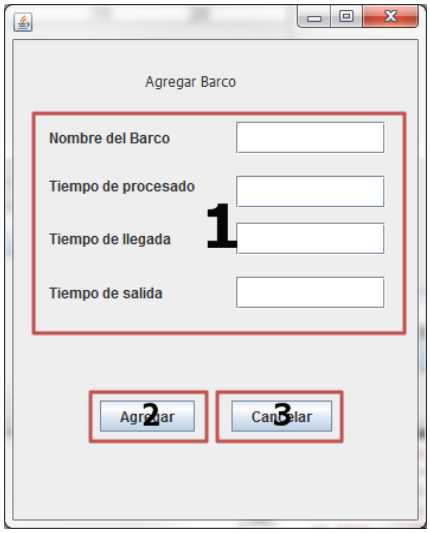


Ilustración 8-2 Agregar Barco GUI

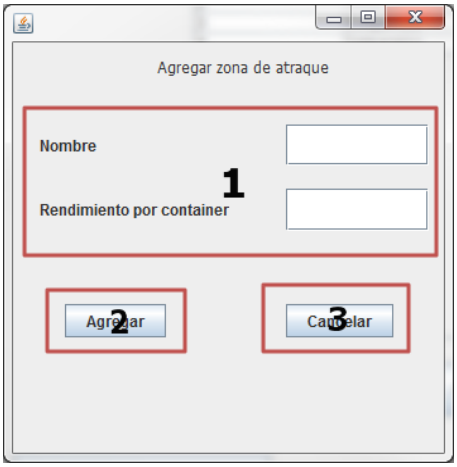


Ilustración 8-3 Agregar zona de atraque GUI

Los datos ingresados por medio de esta interfaz son guardados en el archivo de texto que también puede ser modificado a mano en caso de ser necesario.

Esta interfaz para agregar barcos (Ilustración 8.1) es creada por medio del agente Simulador 3D, el cual se comunica con los agentes Barco los cuales se encargan de comunicarse con el agente Dock Central para ser guardados y luego utilizados durante la simulación.

1. Zona utilizada para el ingreso de datos: Nombre del barco, tiempo de procesado, tiempo de llegada y tiempo de salida.
2. Botón que ingresa el barco con los datos de la zona 1.
3. Botón para cancelar el ingreso de datos.

En el caso de la interfaz para agregar zonas de atraque (Ilustración 8.3) también es creado por el Simulador 3D, el cual se comunica con los agentes Berth los cuales se encargan de comunicarse con el agente Dock Central para ser guardados y luego utilizados durante la simulación.

1. Zona utilizada para el ingreso de datos: Nombre de la zona de atraque, Rendimiento de procesamiento por conteiner.
2. Botón que ingresa la zona de atraque con los datos de la zona 1.
3. Botón para cancelar el ingreso de datos.

Una vez completa la configuración de los actores, por medio del botón “Iniciar Simulación”, el agente Simulador 3D da la orden de inicio de simulación a los demás agentes que iniciaran la negociación en búsqueda de la mejor propuesta de ordenamiento que permita maximizar la llegada de los barcos a las zonas de atraque.

El agente Simulador 3D consultara al agente Dock Central esta propuesta y luego será representada en la siguiente ventana, tal como lo muestra la Ilustración 8.4.

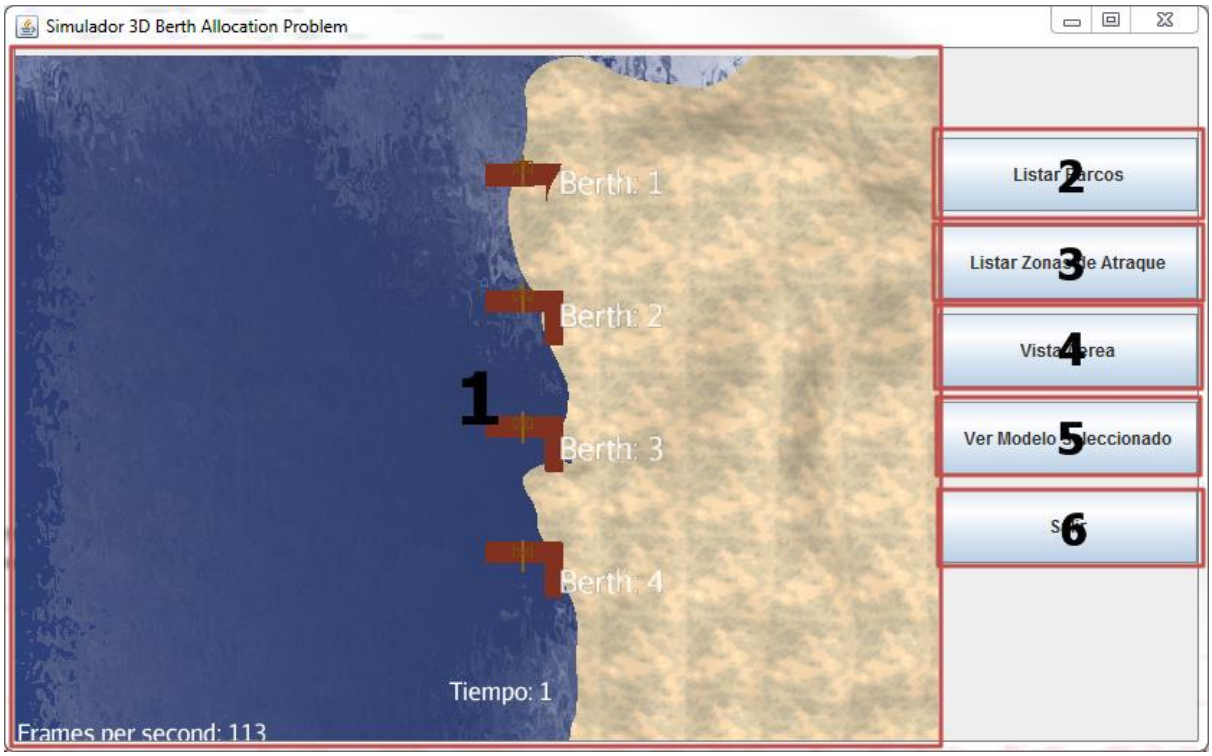


Ilustración 8-4 Prototipo Simulador 3D corriendo

En esta ventana se muestra la llegada de los barcos al puerto según la propuesta entregada por el agente Dock Central al Agente Simulador 3D. Además se da la posibilidad de ver el estado de los barcos en todo momento y de las de las zonas de atraque. También existe la posibilidad de cambiar la vista de la cámara para un mayor acercamiento observación en detalle de algún objeto en específico o la de visualizar el sistema por completo.

También se puede hacer click sobre las zonas atraque o los barcos, para obtener información de ellos.

1. Zona donde se muestra la simulación realizada, incluyendo el tiempo utilizado durante la simulación y los nombres de los distintos actores que aparecen en escena.
2. Botón que muestra una ventana con todos los barcos de la simulación, información y su estado actual.
3. Botón que muestra una ventana con todos los zonas de atraque de la simulación, información.
4. Botón que regresa a la vista aérea la cámara.
5. Botón que cambia la cámara al modelo 3D seleccionado.
6. Finaliza la aplicación.

Dentro de la simulación se puede entrar a la opción de listar barcos, que permite obtener información de los barcos con respecto a su ID asignado, nombre asociado, muelle seleccionado por la simulación, el tiempo de llegado asignado para el barco y el estado actual del barco en el desarrollo de la simulación.

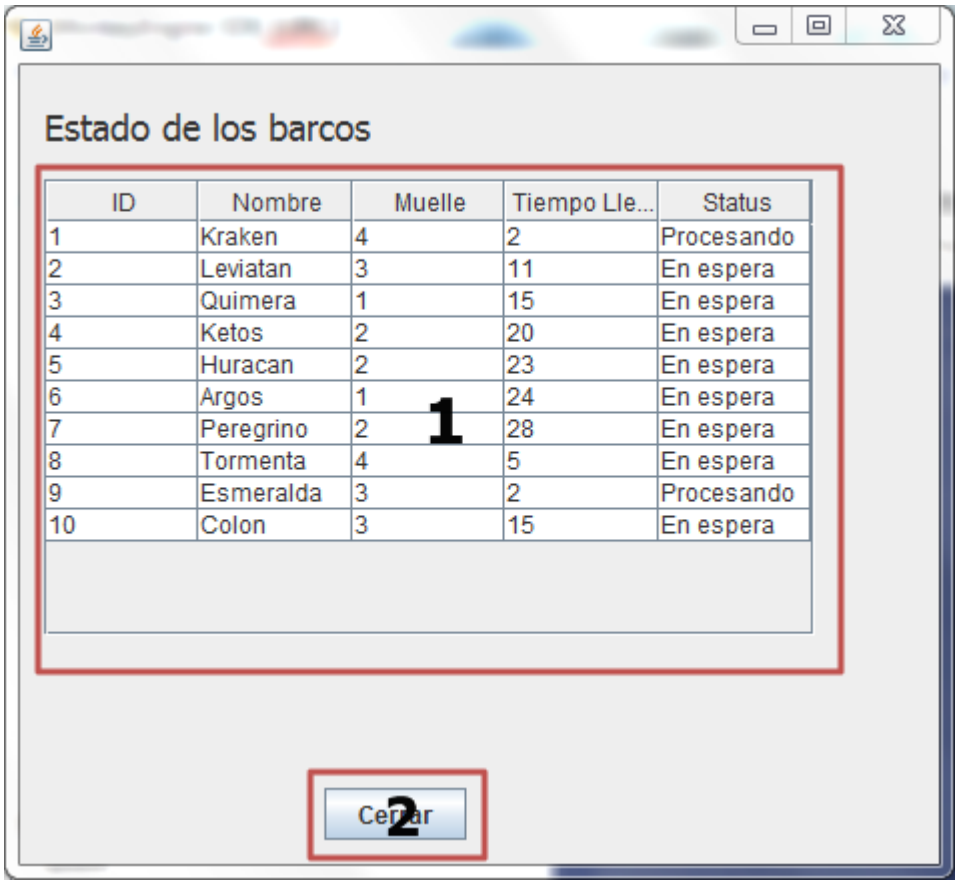


Ilustración 8-5 Lista Barcos

1. Zona donde se muestra el ID asignado, nombre de los barcos, el muelle asignado, el tiempo de llegada al atraque y su estado actual.

2. Botón que cierra la ventana.

Así mismo se puede obtener información relacionada con las zonas de atraque con correspondiente ID, nombre y rendimiento por container.

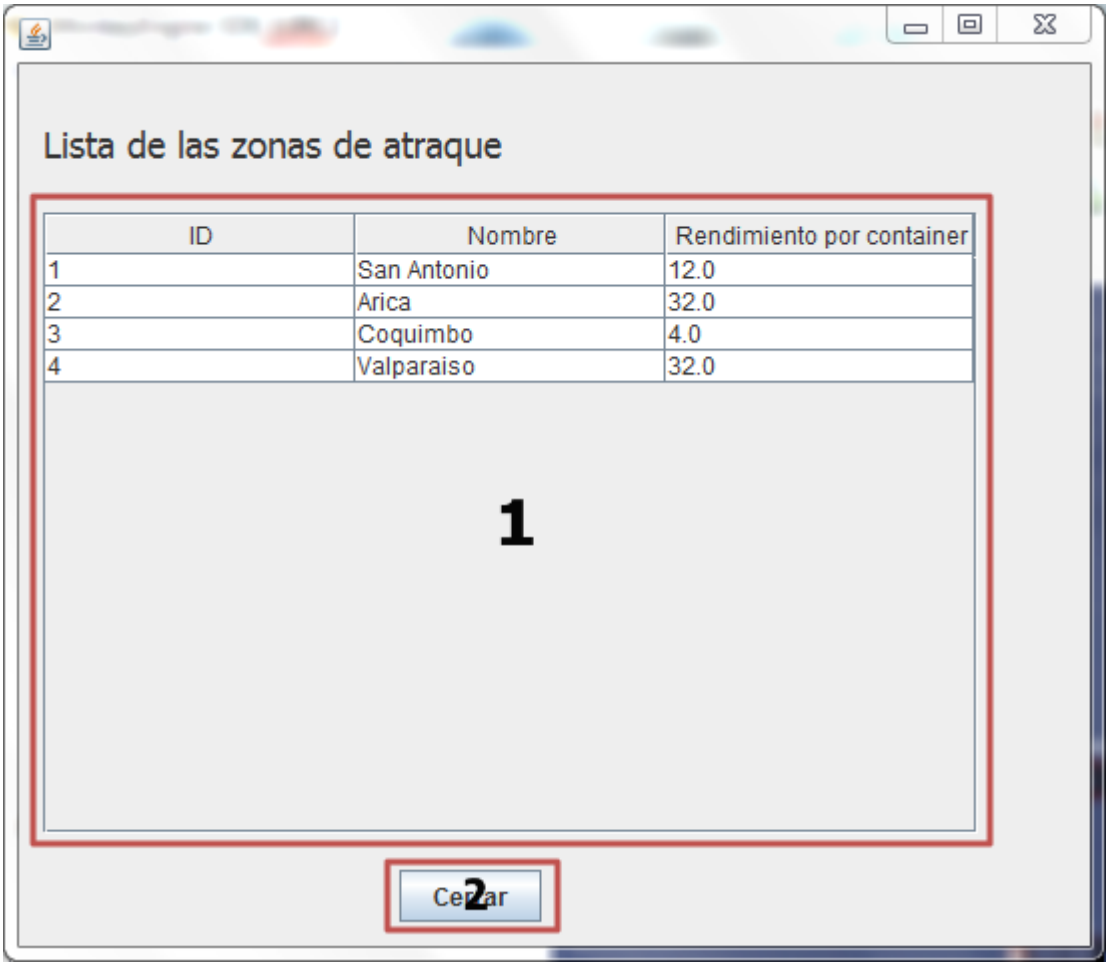


Ilustración 8-6 Lista Zonas de atraque

1. Zona donde se muestra el ID asignado, nombre de los barcos, el muelle asignado, el tiempo de llegada al atraque y su estado actual.
2. Botón que cierra la ventana.

Otra manera de obtener información de los barcos o muelles, es haciendo click sobre ellos. De esta manera aparecen datos de información del barco en pantalla y permite habilitar la opción de hacer un acercamiento para hacer un seguimiento.

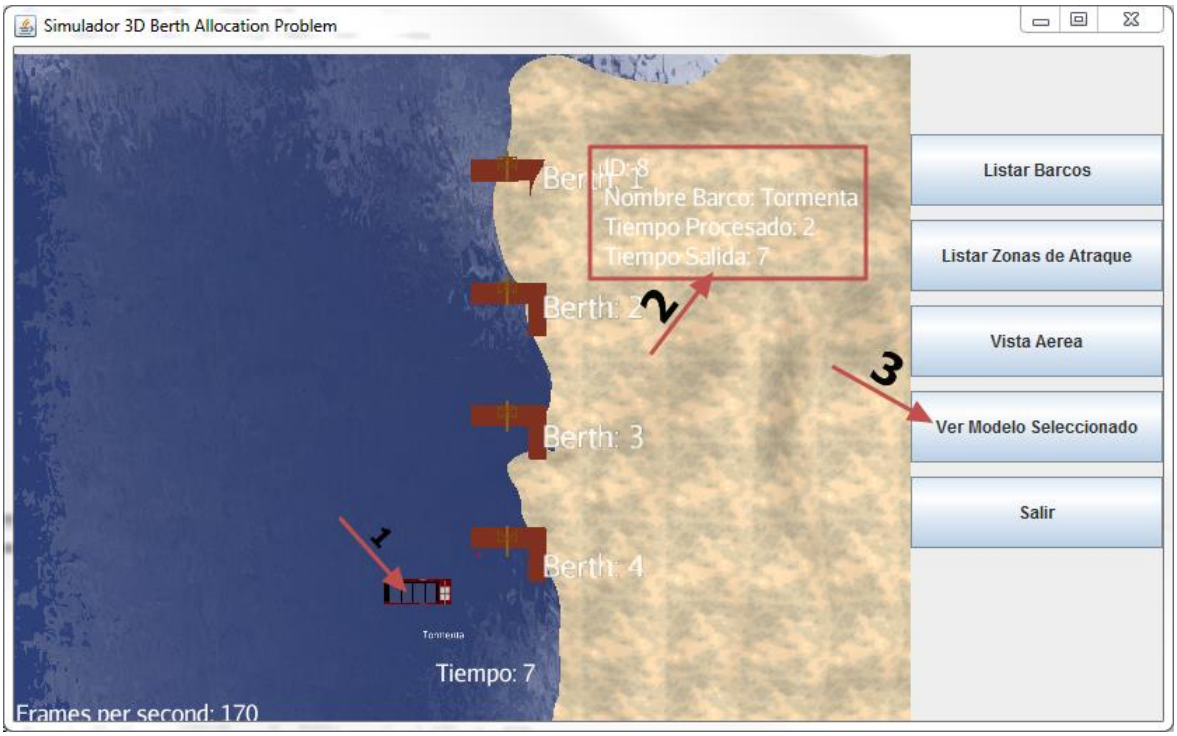


Ilustración 8-7 Información Barco

1. Hacer Click sobre el modelo, se mostrara información del modelo y será seleccionado para un acercamiento.
2. Zona donde se muestra la información del modelo.
3. Botón que permite hacer un acercamiento al modelo seleccionado.

Al usar el botón “Ver Modelo Seleccionado” (3) se realiza un acercamiento como el mostrado en la Ilustración 8-8, este permitirá hacer un seguimiento al modelo y además ver otros datos del barco.

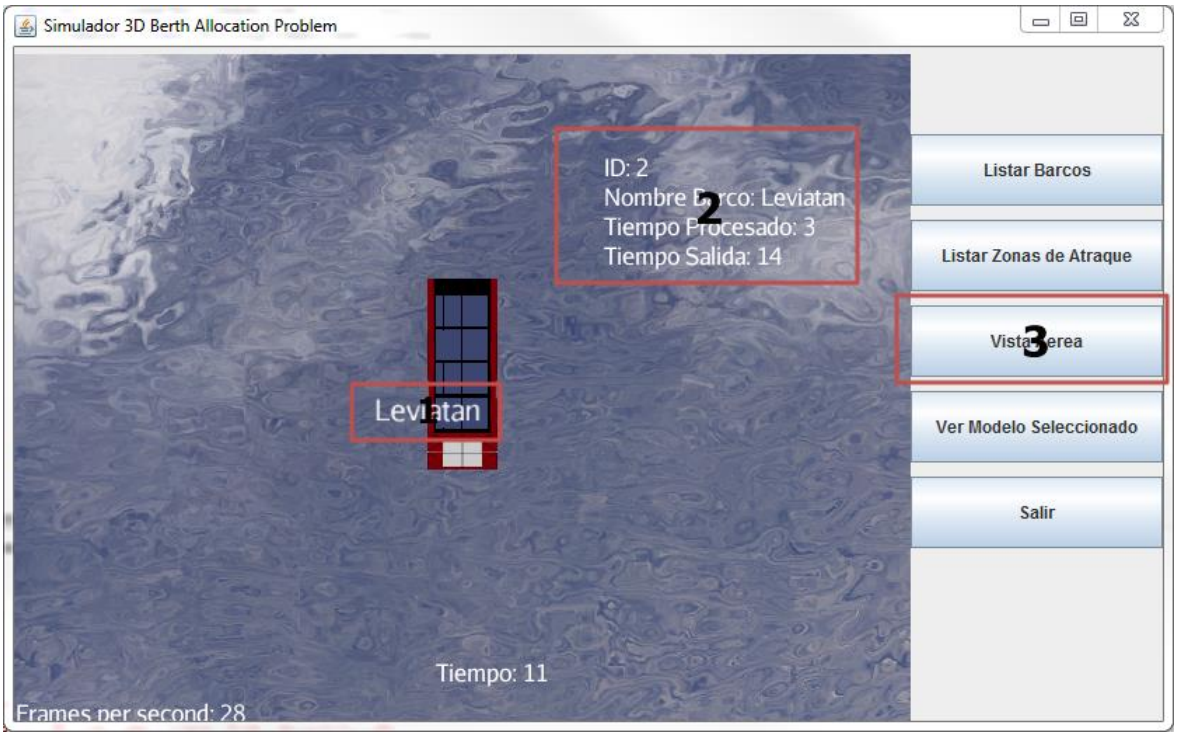


Ilustración 8-8Acercamiento Modelo

1. Nombre del barco sobre el modelo
2. Zona que muestra información del barco acercado
3. Botón que regresa la cámara a su estado inicial.

Para la creación de la ventana de la simulación fue necesario utilizar el canvas de la aplicación de JMonkey Engine para tener acceso a los botones del swing de JAVA que no existen de manera nativa en el motor.

El formato de los archivos de texto que se utilizarán son los siguientes:

Barco.txt:

Como es mostrado en la Ilustración 8.9, el archivo contiene 4 columnas que contienen la información de cada barco correspondiente a una fila. La primera fila corresponde al nombre de la columna que corresponden a nombre, tiempo, tiempo de llegada y tiempo límite en ese orden.

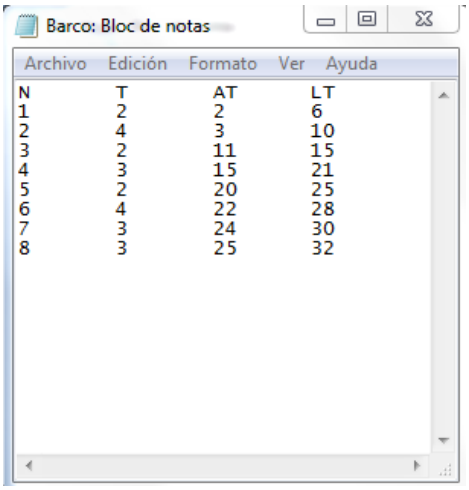


Ilustración 8-9 Ejemplo Barco.txt

Berth.txt:

Como es mostrado en la Ilustración 8.10, el archivo contiene 2 columnas que contienen la información de cada zona de atraque correspondiente a una fila. La primera fila corresponde al nombre de cada columna que corresponden a nombre y rendimiento por container en ese orden.

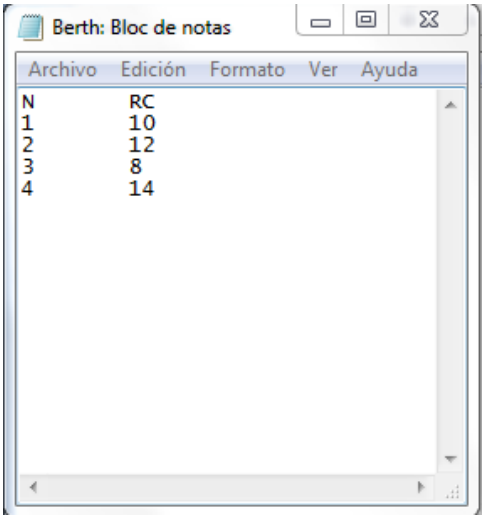


Ilustración 8-10 Ejemplo Berth.txt

8.2 Scene Graph

En Jmonkey Engine se utiliza un Scene Graph para mostrar los modelos 3D en pantalla. En este motor todo lo que se convierta en hijo del root node, es mostrado en pantalla. Gracias a este tipo de diseño, todo lo que se haga sobre un nodo en el grafo, afecta a sus hijos de la misma manera.

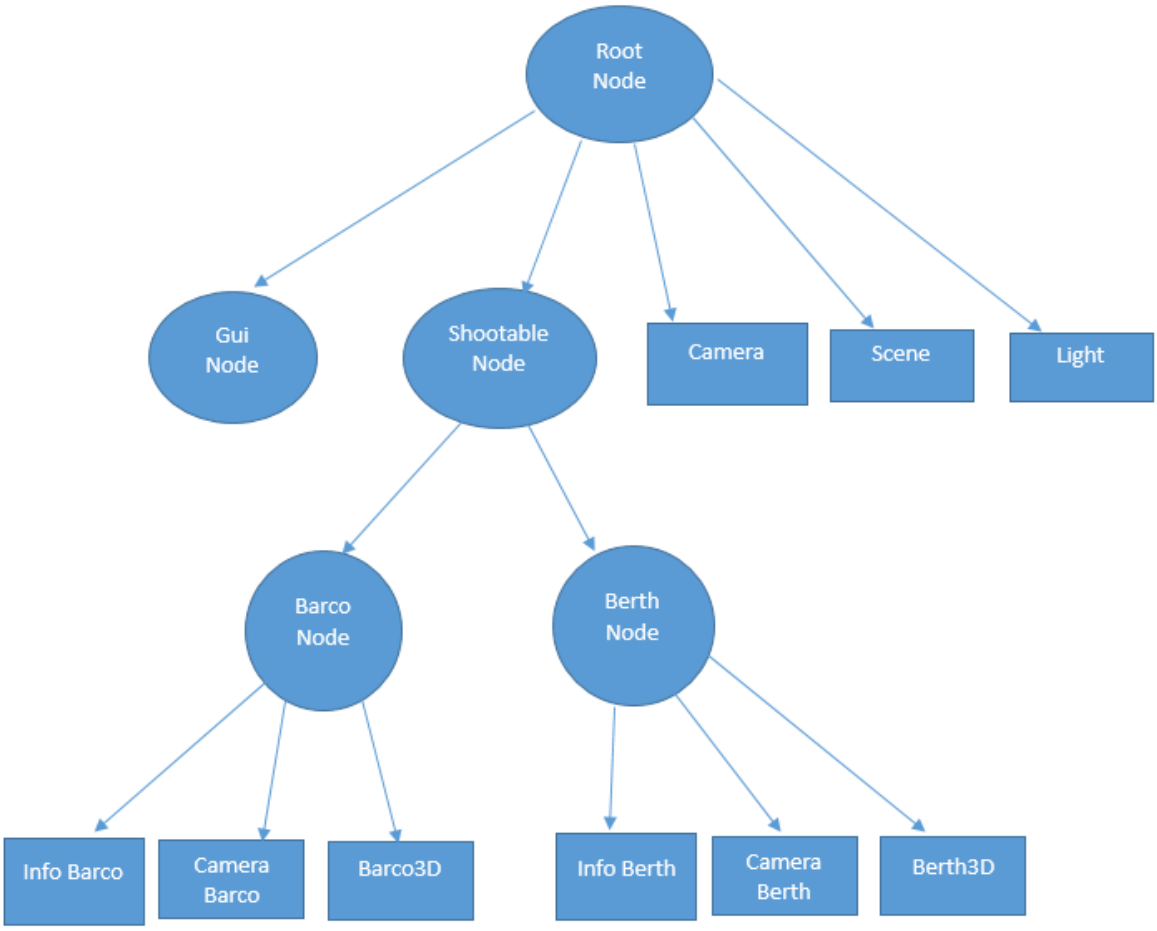


Ilustración 8-11Scene Graph

El grafo creado para la solución del presente trabajo, es el mostrado en la Ilustración 8.11, donde todos los nodos que se convierten en hijos de Root Node son mostrados en pantalla, es por ello que todos los elementos utilizados en el motor están relacionados con este nodo.

-Gui Node: Todos los objetos agregados a este nodo son mostrados en la pantalla en modo 2D, sobre todos los demás objetos, es por eso que este nodo es usado para mostrar información de los distintos barcos, zonas de atraque y el tiempo utilizado en ese momento.

-Camera: Es la cámara principal utilizada en el sistema que permite tener una vista aérea del sistema

-Scene: Se refiere al modelo utilizado en el motor que incluye el terreno usado en el sistema.

- Light: Es la luz ambiental que se utiliza en el sistema, para alumbrar todos los modelos. Esto es importante, debido a la propiedad que permite heredar a todos los hijos, permite que todos los modelos sean iluminados por esta luz.

-Shootable Node: Este nodo tiene la propiedad de ser detectado al ser tocado por el mouse, por lo que sus dos hijos, Barco Node y Berth Node, obtienen la misma capacidad.

- Barco Node: Es el nodo principal que representa cada barco que aparece en pantalla, este contiene el modelo 3D, su cámara asociada y además un texto 3D. Por lo que todos los movimientos de translación de los objetos, es sobre este nodo, el cual permite mover el barco3D, su cámara y texto asociado todos juntos.

-Berth node: Es el nodo principal que representa cada zona de atraque que aparece en pantalla, este contiene el modelo 3D, su cámara asociada y además un texto 3D. Por lo que todos los movimientos de translación de los objetos, es sobre este nodo, el cual permite mover el berth3D, su cámara y texto asociado todos juntos.

8.3 Clases más importantes

A continuación se describirán en detalle, algunas clases que permiten algunas funciones del sistema.

-Motor3D: Esta es la clase más importante, la cual es usada por el agente Simulador3D, el cual se encarga de cargar el AWT usado como GUI para la interfaz del simulador 3D, además de cargar todo el motor 3D de JMonkey Engine. Esto incluye cargar el Root Node, que incluye la scene (terreno), luces, cámara, mar, y los modelos de los barcos y muelles.

-Shootable Node: Este nodo permite que por medio de una función que por medio del uso de un rayo lanzado desde el punto seleccionado de la cámara, trazar una línea recta hasta el primero objeto con el que colisione, del cual se obtiene la información de la figura geométrica seleccionada y a cual nodo está asociado, lo cual permite obtener la información necesaria, para luego ser mostrada en pantalla

-MotionPath: Este código permite trazar un camino de varias etapas para el nodo el barco que permite configurar su duración y velocidad. Gracias a esto es más sencillo ordenarlos para que no existan colisiones en las entradas y salidas de los barcos.

-JMECanvasContext: Debido al uso del AWT como GUI del sistema, es necesario utilizar un canvas que permite incrustar el motor 3D en el AWT. El motor 3D se ejecuta en un hilo distinto al AWT, por lo que los botones al realizar una acción sobre el motor, deben esperar el refresco de la pantalla para realizar los cambios.

-Las clases Barco3D y Berth3D extienden de AbtractAppState por lo que cada objeto creado a partir de estas clases tiene su modelo 3D asociado e información correspondiente, además poseen su propio update por separado del motor principal.

9 Diseño de las pruebas

Las pruebas son necesarias para verificar si el sistema cumple con los objetivos listados inicialmente en el proyecto, lo que además incluye como interactúan cada uno de sus componentes y como se integran al resto del sistema. Las pruebas pueden estar enfocadas a distintos aspectos del sistema como el rendimiento, prestaciones, usabilidad, funcionales, de seguridad, etc. En este caso estarán enfocadas en las funcionalidades del sistema las cuales son las más simple de verificar y además las más importantes.

Las pruebas se realizarán desde el concepto de caja negra, por lo que en base a una acción determinada, se comprobara si se realiza lo deseado, sin entrar en detalle en el proceso que se realizó.

9.1 Planificación de las pruebas.

Existirán dos escenarios que comprobaran el buen funcionamiento del nuevo sistema.

El primer escenario corresponderá al caso más simple el cual consiste en utilizar los 4 puntos de atraque disponibles y 10 barcos que solicitaran una asignación.

El segundo escenario corresponderá a un caso se iniciara con 8 puntos de atraque disponibles y 15 barcos asignados, que permitirá comprobar su correcta ejecución en un mayor escenario.

Ambos escenarios serán revisados para que cumplan con los siguientes requisitos:

-Correcta visualización de las asignaciones realizadas.

-Correcto uso de cámaras.

-Modelos cargados correctamente.

9.2 Especificación de las pruebas

En esta etapa se detallarán los requisitos entregados en la etapa anterior.

Correcta visualización de las asignaciones realizadas:

-Una vez lanzada la interfaz y empezada la simulación se confirmara que lo que se muestra en pantalla, sea lo mismo que se asignó en los agentes.

-Es por ello que se utilizará mantendrá el sistema original de entrega de resultados para su comparación con el sistema 3D

Correcto uso de cámaras:

-Una vez lanzada la interfaz, se probarán los distintos modos de cámara que se ofrecen por lo que la cámara no puede apuntar a lugares no seleccionados en la interfaz.

-No se permitirá un uso libre de la cámara.

Modelos cargados correctamente:

-Se verificará que efectivamente se carguen todos los modelos solicitados por la interfaz.

-Se comprobará que se carguen todas las texturas asociadas a cada modelo.

9.3 Resultados de las pruebas

Una vez finalizada la integración del sistema multiagente con su entorno 3D fue realizado las pruebas que fueron descritas anteriormente.

Se realizo probo el primer escenario donde se utilizaron 4 puntos de atraque y 10 barcos para su asignación.

Asignación

- Se verificó que efectivamente los barcos se aparecían y atracaban según los tiempos entregados por el solver.

Entorno 3D

- Se verificó que efectivamente la cámara solo pudiera ser controlado por los botones del interfaz y no por otros medios.
- Se verificó que efectivamente los modelos 3D se cargaran correctamente, con sus texturas correspondientes

Para el segundo escenario que comprobaba su funcionamiento con una mayor cantidad de puntos de atraques y de barcos.

Asignación

- Se verificó que efectivamente con un número distinto de variables en el entorno el sistema mostraba las asignaciones correctamente.

Entorno 3D

- Se verificó que efectivamente la cámara solo pudiera ser controlado por los botones del interfaz y no por otros medios, utilizando una cantidad de variables distintas al escenario anterior.

  Se verificó que efectivamente los modelos 3D se cargaran correctamente, con sus texturas correspondientes, utilizando una cantidad de variables distintas al escenario anterior.

10 Limitaciones del sistema

Debido a limitaciones del solver y la poca experiencia en entornos 3D, el sistema tiene las siguientes limitaciones.

- No existe la posibilidad de cancelar barcos o zonas de atraque durante la simulación, debido a limitaciones del solver.
- El tiempo utilizado para la simulaciones, es una medida de simulación correspondiente a 1 unidad de simulación es igual a 10 segundos reales.
- La cantidad máxima de muelles a agregar es 9, debido al tamaño del terreno y la no existencia de zoom controlado directamente por el usuario.

- Los muelles a la medida que son agregados, solo se agregan de manera lineal en el entorno 3D.

11 Trabajos futuros

Debido a diversas limitaciones del sistema actual, trabajos futuros se pueden enfocar en realizar las siguientes opciones.

- Debido a que el solver actual no soporta eventos durante la simulación como la cancelación de viajes, trabajos futuros se pueden enfocar en esta característica y luego modificar el sistema 3D para ser visualizado de manera más sencilla.
- Permitir la posibilidad de utilizar tiempos más cercanos a la realidad, y además admitir la posibilidad de acelerar y ralentizar la velocidad del tiempo de simulación.
- Mejorar los modelos 3D que se utilizan para acercar el sistema más a la realidad y a usos reales.

12 Conclusiones

Por medio del proyecto, se ha logrado comprender de mejor manera el problema del Berth Allocation Problem y las soluciones existentes, donde fue analizado especialmente la solución propuesta por René Díaz, debido a que se usó como base para este proyecto. También sirvió como un primer acercamiento a las distintas tecnologías existentes que fueron usadas para el desarrollo de la aplicación y como ellas podían trabajar conjuntamente para lograr de la mejor manera el objetivo.

En la primera parte del proyecto, correspondiente al análisis detallado del anterior sistema, diseño, prototipo y pruebas, se detectaron problemas no esperados inicialmente, pero se entregaron distintas soluciones que deberían permitir minimizar los problemas que se pudiesen encontrar en la etapa siguiente.

En la segunda etapa se realiza la implementación del sistema y las pruebas, los cuales detectan otros problemas relacionados al solver utilizado, por lo que fue necesario eliminar algunas opciones diseñadas originalmente y además de la poca experiencia trabajando con entornos 3D, lo cual no permitió modelos más cercanos a la realidad, a los cuales el usuario pudiera sentirse más identificado.

13Referencias

¡Error! No se encuentra el origen de la referencia.