FACULTAD DE INFORMÁTICA
UNIVERSIDAD NACIONAL DE LA PLATA

ENTORNOS VIRTUALES TRIDIMENSIONALES DE APRENDIZAJE

TESIS DE MAESTRÍA EN TECNOLOGÍA INFORMÁTICA APLICADA EN EDUCACIÓN

Tesista: Ing. Francisco A. Lizarralde

Director: Mg. Luis Mariano Bibbó

Co-Director: Dr. Alejandro Fernández

La Plata - Buenos Aires - Argentina

2016

Dedicatoria

A Raquel, mi novia, mi esposa y mi compañera de toda la vida, por el amor, la comprensión y el apoyo incondicional en todo momento.

A nuestros hijos Ana Lucía y Gonzalo, que me han dado la alegría de verlos crecer y abrirse paso en la vida con vuelo propio.

A mis suegros Nati y Fernando, por todo el cariño y el respaldo que me brindaron desde que los conocí.

A Raúl, por enseñarme a trabajar la piedra y por brindarme el espacio que necesitaba para librar mis propias batallas.

A Gabriel, por ayudarme a recuperar la fe en mí mismo. Y por la magia de su música, que me permitió lograr la fuerza y la serenidad que necesitaba en los tramos finales de este trabajo.

Agradecimientos

A mi director Luis Mariano Bibbó, quien en cada capítulo me fue indicando aquellos aspectos que debía mejorar. Y sobre todo por su apoyo y su consideración, lo cual me comprometió aún más a redoblar mis esfuerzos y no abandonar la tarea emprendida.

A mi co-director Alejandro Fernández, cuyas precisas indicaciones me permitieron orientar el inicio de este trabajo y no perder de vista el objetivo establecido.

Al personal de la Secretaría de Postgrado de la Facultad de Informática de la Universidad Nacional de La Plata, y de manera especial a su Directora Alejandra Pizarro, por su comprensión, asistencia y excelente disposición durante todo el tiempo que me demandó concretar este trabajo.

A la Mg. Marcela González por su atenta lectura y sus oportunos comentarios, que me permitieron aprender más sobre las teorías de aprendizaje y lograr una mejor integración de los aspectos técnicos con los humanísticos.

A todos mis amigos, colegas, profesores y alumnos, que de una u otra forma me ayudaron a transitar este camino.

A todos, muchísimas gracias !!!

El propósito de trabajar sobre el problema no es "llegar a la respuesta correcta" sino tratar de captar el conflicto entre las diferentes maneras de pensar el problema: por ejemplo, entre dos modos intuitivos de pensar o entre un análisis intuitivo y uno formal. Cuando uno reconoce el conflicto, el siguiente paso es elaborarlo hasta sentirse más cómodo...

Seymour Papert

La mejor manera de predecir el futuro es inventándolo.

Alan Kay

Resumen

En el presente trabajo se intenta establecer un marco de evaluación de las capacidades que ofrecen los entornos virtuales 3D, para la enseñanza de temas específicos de la currícula básica de Ingeniería, en particular, aquellos relacionados con la simulación numérica de sistemas dinámicos. Este marco de evaluación es utilizado posteriormente para analizar un conjunto de entornos virtuales 3D, elegidos especialmente por sus características distintivas. Asimismo, se plantea la utilización didáctica de simulaciones numéricas en entornos virtuales 3D, sobre la base de la indagación previa de las principales teorías y modelos de aprendizaje. Finalmente se presenta, como prueba de su factibilidad, la implementación de una simulación numérica desarrollada en uno de los entornos virtuales 3D analizados.

Palabras clave

Entornos Virtuales 3D - Mundos Virtuales - Ambientes Virtuales de Aprendizaje - Simulaciones Numéricas

Índice general

Capítulo 1

Introducción

Se presenta en este capítulo el contexto de la tesis (sección 1.1), su objetivo (sección 1.2), la visión general de la tesis (sección 1.3) y la producción científica derivada de resultados parciales de la misma (sección 1.4).

1.1. Contexto de la Tesis

En los últimos años, los Ambientes Virtuales de Aprendizaje han posibilitado el aprendizaje de diversos temas. Ya sea en el formato de cursos de educación a distancia, o en la modalidad de *blended-learning*, que combina la instrucción electrónicamente mediada con encuentros presenciales, o *e-learning*, donde además, pueden aplicarse mecanismos automáticos de evaluación y/o asistencia personalizada. (Soler, Ferran, Jordi, y Imma, 2012)

A pesar de su diversidad, una de las características que comparte la mayoría de estos entornos es su naturaleza bi-dimensional, basada en la metáfora de escritorio. Sin embargo, la comprensión del ser humano acerca del mundo real, está fuertemente influenciada por la naturaleza tridimensional del espacio que habita.

Ya que es precisamente esta experiencia cotidiana, la que le permite comprender el comportamiento ciertos fenómenos de una manera aparentemente más natural e intuitiva. En este sentido, la representación tridimensional de los entornos virtuales 3D, aporta un elemento fundamental que facilita la comprensión de ciertos contenidos, pues remite a una experiencia similar a la que percibimos en el espacio físico que habitamos.

En la práctica docente en carreras de Ingeniería, es frecuente observar que los estudiantes experimentan cierta dificultad para comprender y modelar sistemas dinámicos, pues perciben a los conceptos subyacentes como un conjunto de fórmulas abstractas, disociadas del sistema físico representado. (Kypuros y Connolly, 2005)

La Ingeniería es un conjunto de conocimientos y técnicas científicas, aplicadas a la resolución de problemas complejos que surgen de la actividad cotidiana de la sociedad. La naturaleza eminentemente práctica de su aprendizaje requiere, además del desarrollo de capacidades analítico-deductivas y del dominio de las matemáticas, la física y otras ciencias; de un manejo eficiente de los recursos para transformar ese conocimiento en una solución aplicable y concreta.

En este sentido, las simulaciones de diversos fenómenos físicos en entornos virtuales 3D se plantean como una promisoria herramienta para vincular dichos fenómenos con los modelos matemáticos que los representan (Daghestani, Ward, Xu, y Al-Nuaim, 2008). Las investigaciones realizadas en estudios de campo, sobre diferentes disciplinas, permiten afirmar que los entornos de simulación brindan a los alumnos un contexto significativo de aprehensión y uso de las herramientas conceptuales (Amaya Franky, 2009).

Sin embargo, la mera incorporación de nuevos recursos tecnológicos no es suficiente para garantizar un *aprendizaje significativo* (Ausubel, 1968), sino

que es necesario establecer previamente un marco pedagógico adecuado. Por este motivo, se han analizado las principales teorías y modelos de aprendizaje, en busca del fundamento teórico que conduzca hacia una adecuada utilización de este nuevo recurso didáctico.

Los entornos virtuales 3D con capacidades colaborativas y de programación proporcionan las herramientas necesarias para la simulación de problemas y el ensayo de posibles soluciones. A la vez que potencian el desarrollo de las capacidades analíticas, así como también, de las competencias necesarias para el trabajo en grupo.

1.2. Objetivos de la Tesis

A continuación, se presentan los objetivos que se pretenden alcanzar en el presente trabajo. Para brindar mayor claridad, estos objetivos se han dividido en dos niveles:

1.2.1. Objetivo General

El objetivo del presente trabajo es analizar las capacidades que ofrecen los entornos virtuales tridimensionales para su utilización dentro del ámbito educativo, como una herramienta que permita reducir la brecha cognitiva presente entre conceptos teóricos y su aplicación práctica, en particular, en la enseñanza de ciertos contenidos que conforman la currícula básica de diversas especialidades de Ingeniería.

1.2.2. Objetivos Específicos

Indagar en las principales Teorías y Modelos de Aprendizaje, en la búsqueda de un marco teórico adecuado para la utilización de los entornos vir-

tuales tridimensionales, en particular, para la incorporación de simulaciones numéricas para el aprendizaje de sistemas dinámicos.

Establecer un marco de evaluación en base a diversos aspectos, considerados relevantes para la elaboración de contenidos y para su utilización como recurso didáctico, en especial, en lo que se refiere a la implementación de técnicas de simulación numérica.

Aplicar este marco de evaluación a un conjunto de ambientes virtuales 3D, elegidos de acuerdo a cierta cualidad representativa, con la finalidad de evaluar comparativamente sus ventajas y desventajas. En este sentido, se ha elegido en primer término a *Second Life*, por ser uno de los entornos más maduros y de mayor difusión. Luego a *OpenSimulator*, pues representa una alternativa de código abierto, lo cual permitiría la instalación de servidores independientes. Y por último a *OpenCobalt*, porque presenta un esquema de comunicaciones en red descentralizado que difiere completamente del que poseen los entornos virtuales 3D anteriormente mencionados.

Finalmente, debido a que el ámbito de aplicación de interés para el presente trabajo se encuentra acotado a la utilización de los entornos virtuales 3D como recurso didáctico para la enseñanza de la Ingeniería, y en particular al estudio del comportamiento de los sistemas dinámicos, se presentará la implementación de una simulación numérica en *OpenCobalt*.

La elección de este ambiente en particular, se debe a que de los entornos virtuales 3D analizados en el presente trabajo, *OpenCobalt* es uno de los menos explorados en la bibliografía, por lo que representa una excelente oportunidad para realizar un aporte y evaluar su desempeño en un caso concreto.

1.3. Visión general de la Tesis

En el capítulo 2 se presenta la evolución de las nuevas tecnologías vinculadas con los procesos de aprendizaje, y en particular, de los entornos virtuales de aprendizaje, tomando como punto de partida, el desarrollo de las primeras interfaces gráficas de usuario interactivas.

A continuación, se presentan las características de las principales plataformas aplicadas al aprendizaje electrónicamente mediado *(e-learning)*, en sus diferentes modalidades. Y finalmente, se analizan las principales características que diferencian a los Entornos Virtuales 3D inmersivos de los Sistemas de Gestión del Aprendizaje, conocidos como LMS por sus siglas en Inglés, *(Learning Management Systems)*.

En el capítulo 3 se discute la forma en que las Nuevas Tecnologías de la Información y las Comunicaciones *(NTICs)* en general, y los Entornos Virtuales en particular, pueden realizar un aporte al mejoramiento del proceso de aprendizaje.

En este sentido, se presentan los fundamentos de las principales Teorías y Modelos de Aprendizaje como una forma de encontrar el marco teórico más adecuado para la incorporación de las simulaciones en entornos virtuales, en un contexto de aprendizaje técnológico de nivel universitario.

En la segunda parte de este capítulo, se establece un marco de evaluación, formado por un conjunto de características consideradas indispensables, para que un entorno virtual 3D presente ciertas ventajas con respecto a otras alternativas más tradicionales de aprendizaje electrónicamente mediado *(e-learning)*.

En el capítulo 4 se detallan los resultados de la aplicación del marco de evaluación establecido en el capítulo 3 y se realiza un análisis detallado de las ventajas y desventajas que presentan cada uno de los entornos evaluados.

Finalmente, en las tablas 4.1 y 4.2, se presenta una síntesis de los resultados obtenidos.

En el capítulo 5 se analizan los aspectos a considerar al implementar simulaciones numéricas en entornos virtuales 3D. En la primera parte, se presentan las características generales de los *Motores de Física* que utilizan *Second Life* y *OpenSimulator*, y posteriormente, se describe el mecanismo de *replicación de código*, que diferencia a *OpenCobalt* de los otros entornos.

En la segunda parte de este capítulo se abordan las cuestiones específicas de la implementación de una simulación numérica en un entorno virtual 3D, y finalmente, se discuten las ventajas y desventajas que presentan, desde un punto de vista pedagógico, los esquemas de programación que posee cada uno de los entornos analizados.

1.4. Producción científica derivada de resultados parciales de la Tesis

Durante el desarrollo de esta tesis se han comunicado resultados parciales a través de diversas publicaciones, los cuales se detallan a continuación:

Revistas Científicas Internacionales

Lizarralde F. - Huapaya, C. (2012). *Análisis de una Plataforma Virtual 3D Descentralizada para el desarrollo de Simulaciones Educativas.* Revista Formación Universitaria. Vol. 5, Nro. 6, ISSN: 0718-5006 – Valparaiso. Chile.

Congresos Internacionales

Lizarralde, F., Huapaya, C. (2012). *Análisis de una Plataforma Virtual para la generación de Simulaciones en un Ambiente Colaborativo.* CLICAP'2012.

Congreso Latinoamericano de Ingeniería y Ciencias Aplicadas. (UNCU). Mendoza - Argentina.

Lizarralde, F., Huapaya, C. (2013). *Simulación Educativa en un Entorno Virtual 3D Colaborativo.* CAIP'2013. XI Congreso Interamericano de Computación Aplicada a la Industria de Procesos. Lima. Perú.

Lizarralde, F., Huapaya, C. (2015). *Enfoque Alternativo de una Simulación Educativa en un Ambiente Virtual 3D Colaborativo.* CLICAP'2015. Congreso Latinoamericano de Ingeniería y Ciencias Aplicadas. (UNCU). ISBN: 978-987-575-119-4 - Mendoza - Argentina.

Congresos Nacionales

Bibbó, L., Lizarralde, F., Huapaya, C. (2011). *Simulación en Entornos Virtuales Tridimensionales de Aprendizaje Colaborativos y Descentralizados.* WICC'2011 - XIII Workshop de Investigadores en Ciencias de la Computación. (UNR). Rosario - Argentina.

Huapaya, C., Lizarralde, F., Arona, G., Vivas, J., Massa, S., Bacino, G., Rico, C., Evans, F. (2012). *Uso de Ambientes Virtuales de Aprendizaje en la Enseñanza de la Ingeniería.* CACIC'2012 – XVIII Congreso Argentino de Ciencias de la Computación (UNS). Bahía Blanca - Argentina.

Capítulo 2

Estado de la Cuestión

2.1. Antecedentes relacionados con el tema

En 1963, Ivan Sutherland, considerado actualmente como uno de los pioneros de los sistemas de Realidad Virtual y Realidad Aumentada, diseñó y presentó en su tesis doctoral *Sketchpad: A Man-machine Graphical Communications System* (Sutherland, 1963), una de las primeras interfaces gráficas de usuario *(GUI)* interactivas. Y por cuyo desarrollo fue galardonado en 1988 con el *ACM Turing Award*, considerado como el Premio Nobel de la Informática. Sin lugar a dudas, *Sketchpad* fue realmente innovador, ya que a principios de los 60's, el procesamiento de datos era realizado generalmente en tandas *(batch processes)*, sin ningún tipo de interacción con el usuario.

Sketchpad fue desarrollado en el Lincoln Laboratory del MIT *(Massachusetts Institute of Technology)*, utilizando una computadora TX-2, diseñada por Wesley Clark (Sutherland, 2012), la cual fue construida para realizar investigaciones sobre los transistores de barrera superficial y su posible empleo en circuitos digitales. Esta computadora, que no tenía un sistema operativo sino apenas un macro assembler, poseía un teclado, una pantalla de nueve

pulgadas en un arreglo de 512x512 puntos, aunque sin generador de caracteres. Y un dispositivo excepcional en ese momento, un light-pen utilizado inicialmente en un proyecto de defensa aérea denominado SAGE.

Figura 2.1: Ivan Sutherland operando el SketchPad.

Sutherland utilizó este dispositivo para realizar algo realmente revolucionario para la época, permitir al usuario dibujar directamente sobre la pantalla. Es decir, por medio del light-pen se podían ingresar directamente las coordenadas del dibujo simplemente tocando la pantalla, y posteriormente con ayuda del teclado se podían ejecutar comandos para aplicar operaciones de rotación y escalado sobre el dibujo.

En su artículo "The Ultimate Display" (Sutherland, 1965), Sutherland afirma: "*La pantalla de la computadora nos brinda la oportunidad de ganar familiaridad con conceptos difíciles de plasmar en el mundo físico. Es un espejo que nos permite ingresar en un país de las maravillas matemático*".

En este sentido, los sistemas informáticos con posibilidad de simular diversos fenómenos, se presentan como una herramienta idónea para mejorar el aprendizaje y amplificar nuestra capacidad de comprensión.

La experiencia que poseemos en relación con el mundo físico que habitamos nos permite comprender muchas de sus propiedades, por lo que nos resulta casi natural predecir gran parte del comportamiento del mismo. Es decir, sabemos que los objetos se caen, que si se encuentran más alejados se ven más pequeños, que debemos ejercer cierta fuerza para vencer a la fricción o para detener un objeto en movimiento. Sin embargo, no solemos tener la misma familiaridad con otra clase de fenómenos como las fuerzas que existen entre partículas cargadas eléctricamente, o con alta inercia, o afectadas por campos de fuerzas no uniformes.(Sutherland, 1965)

En estos casos, la simulación de dichos fenómenos puede facilitar enormemente su comprensión, ya que si bien la práctica virtual no sustituye a la experiencia real, puede ser una alternativa efectiva para plantear situaciones poco frecuentes, así como también, facilitar el aprendizaje en las etapas iniciales (Malbrán y Pérez, 2004).

2.2. Aprendizaje y Nuevas Tecnologías

La sociedad actual nos presenta un escenario fuertemente cambiante en el cual la educación no puede permanecer ajena. Cotidianamente nos enfrentamos con la necesidad de desarrollar nuevas formas de enseñar y de aprender, centradas en el estudiante y en la construcción colaborativa y compartida del aprendizaje.

A partir de esta necesidad, se han instalado en el ámbito educativo diversas tecnologías de la información y de la comunicación, que han dado

lugar a diferentes modalidades de enseñanza, tales como CAI *(Computer Assisted Instruction or Computer Aided Instruction)*, CBI *(Computer Based Instruction)*, DEC *(Digital Educational Collaboration)*, IBT *(Internet Based Training)*, ML *(Multimedia Learning)*, TEL *(Technology-Enhanced Learning)* y WBT *(Web Based Training)*, entre otras. En este sentido, más allá de la modalidad de enseñanza elegida, se suele denominar Ambiente Virtual de Aprendizaje o VLE *(Virtual Learning Environment)*, según sus siglas en inglés, a una combinación de diferentes recursos tecnológicos y de administración de contenidos con una finalidad educativa.

2.2.1. Entornos Virtuales de Aprendizaje

Un Entorno Virtual de Aprendizaje, es un sistema que permite transferir materiales didácticos a los alumnos por medio de la Web, el cual suele estar conformado por cinco áreas principales (Trafford y Shirota, 2011).

- **Información:** Se utiliza para presentar las novedades, noticias y documentos tales como anuncios, reglamentaciones, así como también aquellos eventos tales como fechas de exámenes, presentación de trabajos, etc.

- **Contenido:** Se utiliza para poner una amplia variedad de recursos electrónicos a disposición de los alumnos. Desde archivos de texto, hasta presentaciones, animaciones, archivos de video y de sonido e hiperenlaces a otros recursos disponibles en Internet.

- **Comunicación:** Reúne una amplia variedad de herramientas como listas de correo, foros de discusión, herramientas de mensajería instantánea, wikis, etc.

- **Evaluación:** El proceso de evaluación puede consistir en exámenes, encuestas, asignación de trabajos, etc. Este proceso puede ser en parte automatizado, o realizado por medio de tutores que provean al alumno de la guía y supervisión necesaria.

- **Administración:** Aquí se centralizan las tareas administrativas de la institución, la organización por departamentos, profesores, asignaturas obligatorias y cursos optativos, así como también, las tareas de inscripción, creación de cursos, planificación, auditoría, etc.

La combinación de estas áreas permite implementar además, capacidades de seguimiento de la actividad de los estudiantes, herramientas de comunicación, evaluación y colaboración (Sneha y Nagaraja, 2013). A partir de las cuales, es posible establecer un vínculo entre alumnos y profesores completamente independiente de su ubicación geográfica y de su disponibilidad temporal.

Sin embargo, en aquellos casos en los que el "*aprendizaje electrónicamente mediado*" o "*E-Learning* " no suele ser totalmente conveniente por diversas cuestiones, se suele utilizar un modelo híbrido denominado "*Blended-Learning*", en el cual se combina la distribución de contenidos y tareas en forma remota, con actividades y evaluaciones presenciales.

2.2.2. Plataformas de desarrollo

Si bien existen Entornos Virtuales de Aprendizaje desarrollados a medida para satisfacer específicamente los requerimientos de una determinada institución educativa, la mayoría de los mismos están basados en alguna plataforma de E-Learning conocida, siendo Moodle, BlackBoard *(que antes era WebCT)*, Kenexa, DigitalChalk y eFront LMS, algunas de las más difundidas.

WebCT / Blackboard

WebCT *(Web Course Tools)*, recientemente renombrada como Blackboard, es una de las plataformas más difundidas internacionalmente. Desarrollada en 1997 por Murray Goldberg, en la Universidad de la Columbia Británica, posee como uno de sus factores sobresalientes, la gran flexibilidad de las herramientas para el diseño de los contenidos, lo que hace de este entorno algo muy atractivo, tanto para principiantes como para usuarios experimentados en la creación de cursos en línea.

WebCT / Blackboard cuenta con numerosas herramientas interactivas tales como: tableros de discusión o foros, sistemas de correo electrónico, conversaciones en línea *(chats)*. Además, soporta contenidos en diferentes formatos, destinados principalmente a la implementación de cursos en línea de nivel secundario y terciario, así como también, como complemento de sistemas tradicionales de enseñanza o sistemas híbridos.

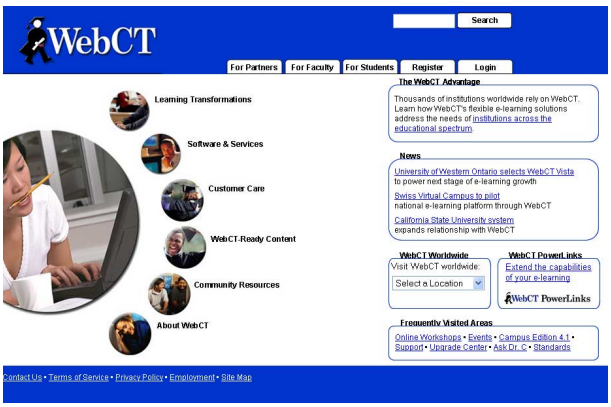

Figura 2.2: WebCT. Captura de pantalla.
(Obtenido de https://veronicaporras.wordpress.com/)

Según Peter Bradford, Blackboard es una buena herramienta curricular, tanto desde el punto de vista de los alumnos como de los docentes y de la institución (Bradford, Porciello, Balkon, y Backus, 2007).

La forma en la que presenta los materiales didácticos promueve el desarrollo organizativo de los estudiantes, así como el mejoramiento de sus habilidades de comunicación. Sin embargo, algunos críticos lo consideran algo restrictivo desde el punto de vista pedagógico, lo que sumado a su alto costo de suscripción, abre el camino a otras alternativas de software libre o de código abierto, como Moodle.

Moodle

En agosto de 2002 se liberó versión 1.0 de Moodle, creada por Martin Dougiamas, por ese entonces ex- administrador de WebCT en la Universidad Tecnológica de Curtin, en Perth, Australia. El nombre de este LCMS *(Learning Content Management System)* proviene del acrónimo de "*Modular Object-Oriented Dynamic Learning Environment*", es decir, Entorno Modular de Aprendizaje Dinámico Orientado a Objetos. En este sentido, la clave de Moodle está en la palabra modular, ya que casi todos sus componentes, menos los foros, pueden agregarse o quitarse, a fin de configurar un ambiente de aprendizaje totalmente adaptado a cada institución.

Desde el punto de vista de la administración de contenidos, Moodle soporta el modelo SCORM *(Sharable Content Object Reference Model)*, que es un conjunto de especificaciones que permite la inter-operatividad, accesibilidad y reusabilidad de contenidos didácticos basados en la Web. Esto permite que dichos contenidos puedan compartirse con cualquier otro Sistema de Gestión del Aprendizaje *(LMS)* compatible con dicha versión de SCORM.

En lo referente a licencias, Moodle es un software libre y gratuito que se distibuye bajo licencia GPL, un detalle no menor, ya que gran parte de su desarrollo depende de la activa participación de la comunidad de usuarios y desarrolladores de Moodle.

Entre sus características técnicas más notables se destacan la integración de una amplia variedad de recursos, desde foros y chats hasta folletería electrónica. Además, se pueden organizar series de preguntas, conjuntos de problemas y ejercicios, notas teóricas y disertaciones, no solamente en formato de texto o Html, sino con imágenes, videos y sonidos.

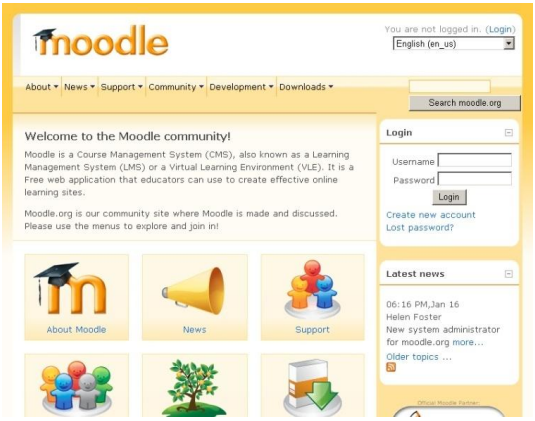

Figura 2.3: Moodle. Captura de pantalla.
(Obtenido de http://www.cloudsurfing.com/site/3653-Moodle/screenshots/)

El diseño de Moodle está basado en una perspectiva pedagógica socioconstructivista. Es decir, su objetivo es proveer un conjunto de herramientas que apoyen un aprendizaje en línea basado en la indagación y el descubrimiento. En este sentido, pretende crear un entorno que posibilite la interacción colaborativa entre los estudiantes, ya sea únicamente a distancia o como complemento de la instrucción en clases presenciales (Kotzer y Elran, 2012).

2.3. Mundos Virtuales

Como se ha visto en el apartado anterior, existen muchas variantes de Entornos Virtuales de Aprendizaje, sin embargo, a pesar de su diversidad, todos estos ambientes comparten una característica en común; una interfaz de usuario bidimensional, basada principalmente en la metáfora de escritorio.

Esta interfaz, aunque muy exitosa por cierto, presenta algunas limitaciones en lo que respecta a la representación de ciertos contenidos y a la interacción con los mismos, ya que en un espacio bidimensional, como un pizarrón o una página web se necesita recurrir a la utilización de recursos tales como símbolos, gráficos o diagramas para poder expresar aquellos conceptos e ideas que exceden las posibilidades de representación de este medio.

John Dionisio (Dionisio, Burns, y Gilbert, 2013) define a los *Mundos Virtuales* como entornos persistentes generados por computadora, en los cuales, varios usuarios pueden acceder simultáneamente de forma remota e interactuar en tiempo real con fines recreativos o laborales.

Estos mundos virtuales constituyen un subconjunto de las aplicaciones de *realidad virtual*, ya que se trata de simulaciones generadas por computadora de objetos tridimensionales que permiten al usuario interactuar de cierta forma "*física*" o aparentemente real.

En este sentido, Richard Gilbert (Gilbert, 2011) enumera cinco características esenciales, que deben poseer los *Mundos Virtuales* para ser considerados actualmente como tales:

- **Tener una interfaz gráfica 3D y sonido espacial integrado.** Este es un requisito básico, ya que si bien pueden tener canales adicionales de comunicación textuales, como pizarras o salas de chat, la interacción principal debe darse dentro de un espacio 3D.

- **Soportar interactividad remota multi-usuario.** Es decir, constituir un espacio virtual de reunión para usuarios físicamente dispersos.

- **Ser persistente.** Los espacios virtuales así como sus contenidos deben permanecer más allá del intervalo de tiempo en el que los usuarios estén conectados.

- **Ser inmersivo.** El nivel de realismo gráfico debe crear en los usuarios una experiencia psicológica de presencia dentro del entorno. Esta sensación de "*residir*", "*habitar*" o "*estar dentro*" del entorno, es un rasgo distintivo de los mundos virtuales.

- **Debe permitir que tanto las actividades como las metas a alcanzar puedan ser definidas por sus usuarios.** Esta es una de las principales diferencias con los juegos multiusuario en red o MMORPG *(Massively Multiplayer Online Role Playing Games)*, los cuales tienen actividades y objetivos predeterminados para los usuarios.

Otro término ampliamente utilizado entre los desarrolladores de ambientes virtuales 3D y videojuegos en red para denominar a los *Mundos Virtuales* es el de *Metaverse*.

Creado por el escritor Neal Stephenson, para describir en una sola palabra el concepto de *Realidad Virtual*, apareció por primera vez en su novela de ciencia ficción *Snow Crash* (Stephenson, 1992), en la cual realiza una breve pero completa descripción de su significado.

"*Dibujando una imagen ligeramente diferente frente a cada ojo se puede producir el efecto de una visión tridimensional. Cambiando la imagen setenta y dos veces por segundo, puede lograrse que la imagen se vea en movimiento. Dibujando la imagen tridimensional en movimiento con una resolución de 2K pixels en cada dimensión, se puede lograr que sea tan nítida como el ojo sea capaz de percibir, y enviando sonido estéreo digital a través de los pequeños auriculares, las imágenes 3D en movimiento pueden tener una banda sonora perfectamente realista. De modo que Hiro no está aquí en realidad. Está en un universo generado por computadora, que su computadora dibuja sobre su visor y le incorpora sonido a través de sus auriculares. En la jerga de los entendidos, este lugar imaginario es conocido como Metaverse.*"

Por otra parte, debido a que los mundos virtuales poseen herramientas de comunicación en tiempo real, y capacidades de interacción y colaboración; es posible pensar que, a pesar de no haber sido diseñados para ser utilizados como Entornos Virtuales de Aprendizaje, los mismos poseen toda la funcionalidad necesaria para desempeñarse satisfactoriamente dentro del ámbito educativo. (Kotsilieris y Dimopoulou, 2013)

2.4. Entornos Virtuales 3D

En los entornos 2D se utilizan principalmente dos tipos de metáforas en concordancia con la finalidad buscada. Por ejemplo, cuando se requiere la representación de contenidos gráficos, geométricos, de esquemas, diagramas, etc. se utiliza la *metáfora de pizarrón*, mientras que cuando se desea organizar en forma visual los contenidos (archivos) y recursos (aplicaciones) existentes, para lograr un fácil acceso a los mismos, se recurre a la *metáfora de escritorio*.

Sin embargo, estas metáforas encuentran ciertas limitaciones cuando se intenta presentar otro tipo de información, por lo cual, se suele recurrir a la utilización de fórmulas o símbolos, lo que requiere del manejo de cierto nivel abstracción para lograr una correcta interpretación. Por ejemplo, un objeto tridimensional puede representarse esquemáticamente por medio de proyecciones o vistas parciales, las cuales requerirán de la interpretación por parte del espectador, precisamente porque dicha representación carece de la tercera dimensión.

En cambio, en los ambientes virtuales 3D, la tercera dimensión incorpora una diferencia fundamental con referencia a la percepción de los contenidos y la interacción con los mismos, pues nuestro cerebro está naturalmente acostumbrado a manejarse en un espacio tridimensional. (Memikoglu, 2014).

Los fenómenos físicos a los que nos enfrentamos cotidianamente suceden en dicho espacio, por lo que una representación visual en un espacio similar nos proporciona una comprensión más inmediata de los mismos, aún cuando se trate de fenómenos que puedan suceder a escala astronómica o molecular (Lang y Kobilnyk, 2009).

Otro aspecto diferencial que aportan los Entornos Virtuales 3D, es una representación "*corpórea*" del usuario, denominada *Avatar*. Chip Morningstar, quien trabajó desde 1984 a 1992 en Lucasfilm Games, como diseñador y programador, es el "*inventor*" del término *Avatar*, con referencia a la representación del usuario.

Precisamente, mientras trabajaba como lider del proyecto "*Habitat*", un entorno gráfico multi-jugador en línea, utilizó este término proveniente del Hinduismo, que simboliza la encarnación de una deidad, para denominar a las representaciones gráficas de los jugadores. Redefiniendo su significado, al menos en el ámbito de la informática. (Morningstar y Farmer, 2008)

Ahora bien, *"¿ Qué puede aportar esta tecnología para mejorar el proceso de aprendizaje, a diferencia de otras tecnologías ?"*.

Según Jay Cross (Cross, O'Driscoll, y Trondsen, 2007), existen cinco sensaciones percibidas por los usuarios, que son exclusivas de los Mundos Virtuales:

- **El sentido de sí mismo.** En los mundos virtuales los usuarios son representados por sus Avatars. A diferencia de los juegos, en los cuales las acciones de los jugadores tienen una libertad restringida, en los mundos virtuales, los Avatars pueden desplazarse, volar o teletransportarse libremente a otros espacios, etc.

A medida que aumenta el tiempo que los usuarios permanecen en el "*mundo virtual*" crece la identificación entre la persona y el Avatar que la representa, por lo que ya no es el Avatar quien se mueve por la pantalla, sino que es el usuario que explora el espacio virtual.

- **Inexistencia de la distancia.** Los Avatars residen en una región virtual aparentemente ilimitada, desde la cual pueden teletransportarse instantáneamente y sin esfuerzo a otras regiones. Por lo tanto, si bien los mundos virtuales se representan en un espacio tridimensional, la noción de distancia entre regiones carece de sentido, como consecuencia de estos desplazamientos instantáneos.

- **El poder de la presencia, el sentido del espacio y la capacidad de crear colaborativamente**. Los Avatars interactúan entre sí a través de las acciones de sus representados. Por lo tanto, se los ve conversar, colaborar, asistir a inauguraciones, presentaciones de libros, a conciertos o reuniones, explorar, construir edificios, etc. Los mundos virtuales alientan la interacción social y la formación de grupos.

- **La omnipresencia de la práctica.** Al poco tiempo de ingresar a algún espacio de *Second Life* uno se da cuenta que no sólo se trata de un mundo virtual social, sino que es además un mundo que fomenta una cultura del aprendizaje colaborativo. En los espacios de práctica o *(Sandboxes)*, se verá como inmediatamente los Avatars de aquellos usuarios más experimentados se acercan a ofrecer ayuda al recién llegado, el cual es fácilmente reconocido por sus movimientos indecisos o por la escasa personalización de su vestimenta. En cada esquina, se observará que la mayoría de las charlas de los Avatars comienza con "*¿ Cómo puedo hacer para ... ?*"

- **Enriquecimiento de la experiencia.** Los mundos virtuales tienen la particularidad de existir en una especie de realidad aumentada, por este motivo las experiencias que transcurren en los mundos virtuales se perciben como realzadas o enriquecidas, sobre todo para aquellas personas que tienen algún tipo de limitación en el mundo real. Es decir, en un mismo espacio virtual uno puede interactuar o discutir un proyecto con colegas de cualquier parte del mundo, pero además, alguien confinado a una silla de ruedas podrá correr o bailar, lo cual puede proporcionar a la persona una forma nueva y atractiva de experimentar la vida. (Gilbert, Murphy, Krueger, Ludwig, y Effron, 2013)

En la actualidad existe una amplia variedad de entornos 3D con capacidades para crear espacios virtuales e interconectar múltiples usuarios de manera simultánea. Y si bien es cierto que en su mayoría, estos entornos están principalmente orientados al desarrollo de actividades sociales, recreativas y comerciales, existe un desarrollo creciente de entornos orientados al área educativa, aunque su número aún es considerablemente menor.

A pesar de la evolución del hardware, que ha posibilitado un incremento notable en la cantidad de operaciones gráficas *(rendering)* de los escenarios virtuales y de la incorporación de nuevas interfaces visuales y de manipulación, el desarrollo de los mundos virtuales aún se encuentra en una etapa experimental. En este sentido, aún hace falta crear los protocolos necesarios para interconectar y transferir información entre diferentes mundos virtuales, algo similar a lo sucedido cuando se establecieron los estándares que permitieron hacer de la World Wide Web, lo que hoy conocemos como Internet.

Capítulo 3

Descripción del problema

3.1. Introducción

Las Nuevas Tecnologías de la Información y de la Comunicación *(NTICs)* son actualmente una realidad ineludible dentro del ámbito educativo. Las búsquedas de información a través de Internet y la utilización de contenidos multimedia se han transformado en algo cotidiano. Los Sistemas de Gestión de Contenidos o LCMS *(Learning Content Management Systems)*, posibilitan que una gran cantidad de estudiantes, sobre todo aquellos que residen en lugares geográficamente alejados de los centros urbanos, pueden acceder a una multiplicidad de cursos en todos los niveles educativos.

Por otra parte, la reciente incorporación de las redes sociales al ámbito educativo presenta nuevos desafíos y el desarrollo de nuevas competencias, pues es necesario que los estudiantes sean capaces de distinguir entre la habilidad necesaria para encontrar información en línea y la capacidad para comprender dicha información.

Según Mark Connolly, es posible utilizar las redes sociales para cultivar y demostrar un aprendizaje profundo. Sin embargo, esto requiere de la

superación de las distracciones permanentes, de la eliminación del exceso de información irrelevante y de vencer la tentación de navegar sin rumbo. (Connolly, 2011)

En este contexto, los entornos virtuales tridimensionales se vislumbran como una interesante herramienta para incorporar en la educación a distancia, adecuándola a los diferentes niveles educativos. Y en el caso específico, de su aplicación en la educación superior, resulta de gran importancia conocer sus características con el propósito de seleccionar el entorno más adecuado en cada caso, teniendo como objetivo, incentivar a los estudiantes a jugar un papel diferente y enriquecedor en el proceso de formación de las experiencias de aprendizaje colaborativo y de la participación en actividades que apoyen el propio aprendizaje y la meta-reflexión (De Freitas, 2008).

Es importante destacar que esto no significa que los Mundos Virtuales vayan a reemplazar a otros recursos didácticos, sino que una utilización reflexiva y complementaria de esta tecnología, dentro de un marco pedagógico adecuado, podría resultar sumamente beneficiosa para el mejoramiento del proceso de enseñanza-aprendizaje.

En el presente capítulo se discutirán dos aspectos de principal relevancia para el presente trabajo. En primer término, en la sección 3.2, se expondrá el marco teórico representado por las diferentes teorías y modelos de aprendizaje junto con sus principales referentes, abarcando desde el modelo conductista hasta el modelo construccionista y destacando los aportes realizados por cada uno de ellos.

Y en segundo término, en la sección 3.6, se discutirán los aspectos tecnológicos que habrán de considerarse para establecer un marco de análisis comparativo, a partir del cual se evaluarán a los entornos seleccionados.

3.2. Marco Teórico

La incorporación exitosa de los entornos virtuales 3D al ámbito educativo requiere, además de ciertos recursos tecnológicos, del conocimiento de las teorías y/o modelos pedagógicos que puedan brindar el marco teórico necesario para la creación de contenidos, materiales didácticos y desarrollo de actividades individuales y/o grupales.

Debido a que a lo largo del presente trabajo, se recurrirá frecuentemente al uso de los términos *pedagogía* y *didáctica*, resulta conveniente comenzar esta sección con la presentación de sus respectivas definiciones.

La palabra pedagogía tiene su raíz etimológica en las palabras griegas **paidos** que quiere decir "*niño*" y **gogós** que significa "*el que conduce*". Aparentemente, el origen de esta palabra se remonta a la Antigua Grecia, ya que los griegos acostumbraban a utilizar la palabra **paidagogós** para referirse al esclavo que conducía a los niños hacia la escuela, o dicho en un sentido más amplio, hacia el conocimiento.

Sin embargo, hoy en día, la pedagogía es considerada una ciencia multidisciplinaria, perteneciente al campo de las ciencias sociales, cuyo objetivo es analizar y comprender el fenómeno de la educación. Y al referirnos a la educación, nos referimos a los procesos sistemáticos de aprendizaje, transmisión de conocmientos, desarrollo de capacidades y de habilidades; no sólo en el caso de los niños sino en todas las etapas de la vida.

En este sentido, la pedagogía orienta las acciones educativas y de formación basándose en principios, métodos, prácticas, técnicas, aportaciones y posturas de pensamiento, presentes en los procesos de enseñanza-aprendizaje.

En cambio, la didáctica es la disciplina científico-pedagógica que tiene por objeto de estudio aquellos procesos y elementos existentes en la enseñanza y el aprendizaje.

Por lo que podría decirse que, es la parte de la pedagogía que se ocupa de las técnicas y métodos de enseñanza, destinados a plasmar en la realidad las pautas de las teorías pedagógicas. Por consiguiente, puede decirse que la *didáctica* es una disciplina que forma parte de aquella dimensión más amplia denominada *pedagogía*.

3.3. Teorías y Modelos de Aprendizaje

La mayoría de las teorías y modelos de aprendizaje están basados en ciertas corrientes psicológicas que explican los mecanismos mediante los cuales se produce el aprendizaje. Por ejemplo, el modelo pedagógico *conductista* se basa en los estudios llevados a cabo principalmente, por el psicólogo estadounidense Burrhus Skinner, sobre la modificación del comportamiento de los individuos o "*condicionamiento*", por medio de acciones del tipo estímulo-respuesta. (Hilgard, 1988)

A su vez, el *cognitivismo* posee su fundamentación en la psicología cognitiva, la cual estudia los mecanismos básicos y profundos presentes en la elaboración del conocimiento, desde la percepción, la memoria y el aprendizaje hasta la formación de conceptos y el razonamiento lógico.

Por último, el *constructivismo* es una corriente a la cual han adherido prestigiosos psicólogos y educadores, y cuya formalización se atribuye generalmente al psicólogo Jean Piaget, quien sugiere que los individuos construyen nuevo conocimiento a través de un proceso de acumulación y asimilación de sus propias experiencias previas. (Gilakjani, Leong, y Ismail, 2013)

Esta última corriente inspiró a Seymour Papert, educador, matemático y creador del lenguaje de programación LOGO, para elaborar una teoría del aprendizaje denominada *construccionismo*, la cual agrega a las ideas desarro-

lladas por la teoría psicológica constructivista, la idea de que el aprendizaje es más efectivo cuando los estudiantes participan de la creación de un producto concreto y significativo para ellos.

3.3.1. Conductismo

El conductismo está basado en los trabajos desarrollados a principios de 1900, por John Watson. En sus investigaciones, Watson buscó aplicar de forma rigurosa el método científico a la psicología, y por esta razón, basó sus estudios únicamente en aquellos eventos considerados observables. Es decir, el conductismo describe científicamente el comportamiento de los individuos en términos de estímulo-respuesta, sin recurrir a eventos fisiológicos internos o a hipotéticos contructos como los pensamientos o creencias.

Por lo tanto, el modelo de aprendizaje conductista consiste en un entrenamiento promovido al rango de principio de aprendizaje, en el que las actividades del tipo estímulo-respuesta, descansan sobre las ideas de "*condicionamiento*" y de "*refuerzo*".

En este caso, el responsable del diseño instruccional, analiza los comportamientos cuyos encadenamientos expresan las competencias que se desea que adquieran los estudiantes. Y posteriormente, elabora preguntas que puedan provocar su manifestación, asociando a las respuestas del estudiante los estímulos de refuerzo, aprobadores o reprobadores, necesarios.

A mediados de 1950, Burrhus Skinner desarrolla una teoría denominada "*conductismo radical*", en la cual se distancia en parte de las ideas de Watson, ya que incluye en su modelo a los eventos privados tales como el pensamiento y los sentimientos. A pesar de que estos eventos no son directamente observables, Skinner afirma que los individuos también son capaces de actuar como observadores de su comportamiento interior.

En lo referente al ámbito educativo, para Skinner un buen programa instruccional es aquel que maximiza el efecto del éxito en base a técnicas de refuerzo del condicionamiento, por lo que recomienda, fraccionar el aprendizaje de los estudiantes en pequeños pasos y brindarles respuestas positivas, no solo a medida que van logrando sus objetivos, sino también en distintos momentos, lo que ayuda a mantener el interés (Skinner, 1986). Este modelo aún suele aplicarse en juegos didácticos en línea y en aplicaciones informáticas educativas muy difundidas, como por ejemplo, la aplicación móvil para aprender idiomas, conocida como **Duolingo** ™.

3.3.2. Cognitivismo

Aproximadamente hacia fines de los años cincuenta y principios de los sesenta, las teorías de aprendizaje comenzaron a alejarse de la corriente conductista abriendo el camino a nuevas teorías de aprendizaje basadas en las teorías y modelos de la psicología cognitiva (Ertmer y Newby, 1993). La principal diferencia consiste en que, mientras en el modelo conductista, el aprendizaje es considerado una mera transmisión de conocimientos hacia un sujeto pasivo, en el modelo cognitivista se considera que el aprendizaje es un proceso activo de construcción por parte del estudiante.

En este sentido, la estructura mental previa del estudiante, formada por sus esquemas y modelos mentales, resulta ser un factor esencial en el proceso de aprendizaje, ya que es precisamente esta estructura que le otorga significado y organización a sus experiencias, la que le permite avanzar más allá de la información recibida.

Entre los diferentes investigadores que aportaron y enriquecieron con sus ideas al modelo cognitivista, y que posteriormente dió nacimiento al modelo constructivista, se encuentran Jerome Bruner, Paul Ausubel y Jean Piaget.

Precisamente Bruner fue quien acuñó el término "*andamio*" (scaffolding) para referirse a la forma en la que aprenden los niños, los cuales necesitan de una estructura de conocimientos ya adquiridos para apoyarse en ella, en el proceso de construir nuevo conocimiento.

En concordancia con esta idea, Bruner propone un modelo de aprendizaje "*en espiral*", en el cual, se van presentando cíclicamente todos los temas de la currícula, aunque tratados cada vez con un nivel de profundidad mayor. En cuanto al diseño del material didáctico, considera que el mismo debe ser capaz de mantener el interés del estudiante, ya que ese interés es primordial en el proceso de aprendizaje.

Como hemos visto, el modelo conductista utiliza la metáfora de los eventos de entrada-salida de un sistema informático para explicar cómo los estímulos se corresponden en las señales de entrada del proceso de aprendizaje y los comportamientos (o respuestas) son percibidos como las señales de salida.

A esta metáfora, el modelo cognitivista le agrega la "*caja negra*", como un elemento interviniente que posee un impacto variable entre los estímulos y las respuestas, como una forma de explicar el mecanismo de procesamiento interno de la información por parte del estudiante (Dabbagh, 2005).

3.3.3. Constructivismo

Finalmente, en el modelo constructivista, los estudiantes ocupan el rol de agentes activos del proceso de aprendizaje creando sus propias representaciones subjetivas de la realidad objetiva. De forma tal, que la nueva información recibida por el estudiante pueda ser enlazada con sus conocimientos previos.

El psicólogo Lev Vygotsky resalta la importancia del aprendizaje guiado y pone especial énfasis en los factores condicionantes propios de la sociedad y la cultura, ya que para él, los procesos de aprendizaje son afectados por la

cultura en la que nacemos y nos desarrollamos, y por la sociedad en la que vivimos.

En su teoría de la "*zona de desarrollo próximo*", afirma que cuando se evalúa el desempeño de los niños al realizar alguna tarea por cuenta propia, por lo general, se observan mejores resultados cuando éstos la realizan en compañía de un adulto. Sin embargo, esto no siempre significa que el adulto le esté enseñando concretamente como realizarla, sino que el proceso de compromiso que se genera con el adulto lo habilita para refinar su pensamiento, así como también mejora su desempeño. (Cakir, 2008)

Por su parte, el psicólogo Jean Piaget señala que el aprendizaje es una reorganización de estructuras cognitivas y es también la consecuencia de los procesos adaptativos al medio, la asimilación de conocimiento y la acomodación de éstos a dichas estructuras, razón por la cual, afirma que cada individuo aprende a su propio ritmo.

Esta concepción dió lugar a una nueva línea de investigación conocida con el nombre de "*didáctica de las ciencias*", cuyas realizaciones renuevan las ideas sobre el proceso de aprendizaje. En este sentido, propone ambientes didácticos que faciliten a la vez el comprender, el aprender y la movilización del saber. Lo cual ha generado nuevas propuestas para mejorar la enseñanza y la divulgación científica.

Sin embargo, para Piaget, la motivación para aprender del estudiante es inherente a él y por lo tanto no es manipulable directamente por el docente. Por lo tanto, la enseñanza debe permitir que el estudiante interactúe con los objetos de su ambiente, transformándolos, encontrándoles sentido y variándolos en sus diversos aspectos. Ya que la experimentación es lo que le permitirá al estudiante hacer inferencias lógicas y desarrollar así, nuevos esquemas y estructuras mentales.

Es decir, en el paradigma constructivista, el estudiante posee un rol activo en su aprendizaje al relacionar el conocimiento asimilado con el nuevo que va descubriendo (Huang, Rauch, y Liaw, 2010).

En este sentido, Piaget sostiene que la adquisición de conocimiento, está fuertemente relacionada con la capacidad del estudiante para explicar lo aprendido, ya que esta capacidad es la demostración de que se ha producido una modificación y/o ampliación de sus estructuras cognitivas previas.

Las teorías de Jean Piaget, sirvieron a Paul Ausubel, como base para el desarrollo de su teoría del "*aprendizaje significativo*", según la cual, los estudiantes van construyendo sus propios esquemas de conocimiento para comprender mejor los conceptos. De esta forma, los nuevos conocimientos se incorporan cuando el estudiante adecúa estos nuevos conocimientos a su estructura cognitiva previa (Adhikari, 2012).

André Giordan (Giordan, 1995) considera que para Ausubel, los nuevos conocimientos sólo se pueden aprender si se reúnen tres condiciones:

- La disponibilidad de conceptos más generales que se van diferenciando progresivamente en el curso del aprendizaje.
- La puesta en marcha de una "*consolidación*" para facilitar el dominio de las lecciones en curso, pues no se puede proponer informaciones nuevas mientras no se dominan las informaciones precedentes, ya que si no se cumple esta condición, el aprendizaje de todos los conocimientos puede verse comprometido.
- La "*conciliación integradora*", consiste en distinguir las semejanzas y las diferencias entre los antiguos conocimientos y los nuevos, en delimitarlos y resolver eventualmente las contradicciones; ya que esto, debería conducir obligatoriamente a reformularlos.

En este sentido, uno de los investigadores en educación que realizó un importante aporte a este concepto del *aprendizaje significativo* fue Joseph Novak, quien logró desarrollar un instrumento didáctico para evaluar en alguna medida, si el estudiante realmente tiene asumidas en sus estructuras cognitivas el nuevo aprendizaje, a través de la creación de los *mapas conceptuales*, pues para poder hacer un mapa conceptual se necesita haber aprendido el concepto, ya que para explicitar las relaciones con otros conceptos es necesario poder manipularlo con significado. (Moreira, 1997)

El aporte teórico de Novak, su teoría de la educación y las técnicas instruccionales surgidas de ellas, como por ejemplo, los mapas conceptuales representan un marco de referencia conceptual y metodológico de gran validez para guiar la práctica docente y mejorar la calidad de la enseñanza, puesto que el aprendizaje no es solo la asimilación de conocimientos, sino que implica su revisión, su modificación y su enriquecimiento mediante nuevas conexiones y relaciones entre ellos.

Bryn Holmes introduce un nuevo enfoque del aprendizaje (Holmes, Tangney, Fitzgibbon, Savage, y Mehan, 2001), al que denomina Constructivismo Comunal, en el que los estudiantes no solo construyen su propio conocimiento (Constructivismo) como resultado de la interacción con su entorno (Constructivismo Social), sino que participan activamente en el proceso de construcción de conocimiento para su comunidad de aprendizaje. Es decir, los principios del Constructivismo Comunal sostienen que los estudiantes investigan y trabajan para crear artefactos de aprendizaje que serán usados y mejorados por futuros estudiantes (Girvan y Savage, 2010).

3.3.4. Construccionismo

A mediados de los años sesenta, Seymour Papert, matemático, especialista en informática e investigador del MIT (Massachusetts Institute of Technology), diseñó el lenguaje de programación LOGO, junto con otros investigadores como Wally Feurzeig, Cynthia Solomon y Daniel Bobrow que colaboraron en su implementación. Este lenguaje, derivado de LISP (LISt Processing) fue específicamente desarrollado para que pudiera ser programado por niños (Dennett, 1993). Por lo tanto, además del lenguaje, crearon un artefacto robótico (ver Fig. 3.1) al que denominaron "*Turtle*" (Tortuga), el cual tenía un doble propósito, propiciar la experimentación, y estimular a los niños a través de la interacción con un objeto real y tangible.

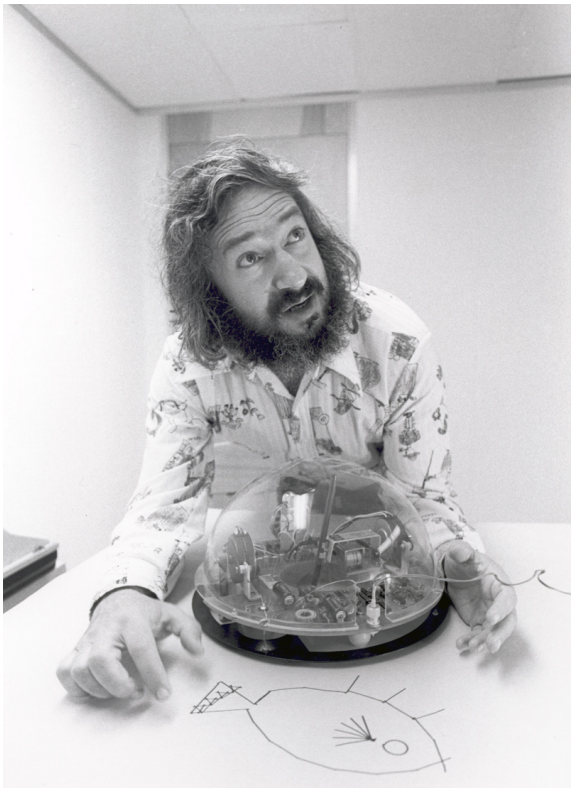

Figura 3.1: Seymour Papert y la tortuga de LOGO.

A medida que fueron aumentando las capacidades de representación visual de las computadoras, la tortuga física de LOGO se transformó en una pequeña imágen (ícono) en la pantalla de la computadora, su nuevo habitat, aunque Papert considera que la interacción con objetos físicos tiene una importancia vital en el proceso de aprendizaje (Papert et al., 1999).

Seymour Papert, quien trabajó junto a Jean Piaget desde 1959 a 1963 en la Universidad de Ginebra, coincide plenamente con él en la importancia que posee el error dentro del proceso de aprendizaje. Para Piaget, un error, es provocado por el desajuste que se genera al intentar aplicar a una realidad una estructura conceptual e interpretativa que no la explica totalmente. En este sentido, el error se transforma en parte del aprendizaje y por lo tanto no debe censurarse, sino que por el contrario, debe utilizarse como un mecanismo para promover la reflexión sobre las causas, y una forma de lograr una comprensión más profunda del tema en estudio.

Si bien Papert, adopta las principales ideas desarrolladas por Piaget acerca del constructivismo, se diferencia del mismo en su aplicación concreta en cuanto a la pedagogía y a la didáctica. Según la visión de Papert, el conocimiento se construye, por lo tanto el docente debe propiciar los espacios para que los estudiantes inicien su proceso de construcción con la realización de actividades creativas.

Según sus propias palabras, la definición más simple de construccionismo evoca la idea de *aprender-haciendo* (Papert y Harel, 1991). Y agrega, que el aprendizaje es más efectivo cuando el estudiante realiza actividades donde manipula materiales y experimenta en la construcción de un producto que le es significativo para él y que puede compartir con los demás.

En este sentido, para Papert, el construccionismo es una teoría de aprendizaje y a la vez una estrategia de educación, es una potente herramienta

de diseño para la transformación, de una educación con actividades pasivas en una educación activa, atractiva, con experiencias educativas ricas que propicien la reflexión (Papert, 1980).

En su teoría construccionista, Seymour Papert establece tres elementos fundamentales:

- Herramientas para pensar
- Resultados compartidos
- Micromundos

Las herramientas para pensar, son operadores tecnológicos utilizados para despertar la curiosidad de los estudiantes y conducirlos a pensar en otros temas. La construcción de artefactos y la experimentación y/o transformación de los mismos, conduce al aprendizaje de otros principios, como por ejemplo, aquellos que posibilitan su funcionamiento.

A medida que los estudiantes logran llevar a cabo sus proyectos a partir del aprendizaje sensorial, de la experimentación y de las ideas y conceptos involucrados, es muy importante que exista una forma de compartir esos resultados. Ya que que esta construcción al ser compartida con los demás refuerza poderosamente el aprendizaje.

Junto con la creación del lenguaje de programación LOGO, Papert creó el concepto de *Micromundo* (Microworld). De hecho, LOGO representa el primer Micromundo creado, ese mundo geométrico en el que habita la famosa "*tortuga*" de LOGO (Papert, 1987).

Los Micromundos son ambientes de exploración, de experimentación, de descubrimiento, de creación individual, y a la vez compartida. Estos ambientes pueden ser reales o simulados en una computadora, pero fundamentalmente son facilitadores del aprendizaje, ya que mediante ellos se pueden

realizar construcciones complejas a partir de algoritmos simples, desarrollando la creatividad y el pensamiento lógico a partir del trabajo en proyectos.

Mitchel Resnick, investigador del MIT, discípulo de Papert y creador de herramientas educativas como *StarLogo* y *Scratch*, sostiene que :
"*Las computadoras pueden ser vistas como un material de construcción universal, ampliando enormemente lo que las personas pueden crear y lo que pueden aprender en el proceso*". (Resnick, 2002)

3.4. Construccionismo y Simulaciones

Una de las razones de la dificultad que experimentan los estudiantes para aprender ciertos conceptos de Física, suele estar asociada a concepciones erróneas basadas en experiencias de la vida cotidiana, como aquellas que remiten a los axiomas Aristotélicos "*Motus simplex terminatur at quietem*" y "*Nullum violentum potest esse perpetuum*", que dicho de otra manera, significa que un cuerpo en movimiento termina por detenerse si no tiene una fuerza aplicada. Este tipo de concepciones erróneas suelen ser muy difíciles de remediar por medio de las estrategias pedagógicas tradicionales.

Por este motivo, Papert impulsa fuertemente la utilización de las TICs como una potente herramienta de transformación, de una educación con actividades pasivas, en una educación activa, atractiva, y con experiencias educativas ricas que propicien la reflexión (Papert, 1980).

Una forma de lograr esto, es proporcionando un contexto de aplicación de los conceptos aprendidos. Es decir, colocando al estudiante en una situación tal, que la comprensión de los conceptos sea más importante que la adquisición y recuperación de información basada principalmente en la memoria. (Juhary, 2006)

En este contexto, las simulaciones actúan como mediadores entre el modelo matemático *(conocimiento abstracto)* y la forma en que el estudiante percibe su funcionamiento *(conocimiento concreto)*. De esta forma, se hace posible analizar situaciones tan complejas como las trayectorias de satélites o el estudio de la mecánica de fluidos, y al mismo tiempo aumentar el interés y la motivación en el estudiante, mejorando claramente su aprendizaje (Zamora y Kaiser, 2009).

Papert demostró con los *"Micromundos de LOGO"* que, aún pudiendo resultar mucho más simples y rústicas las implementaciones realizadas por los estudiantes, éstas les permitían desempeñar un rol más activo y creativo en su proceso de aprendizaje. Por esta razón, resulta mucho más enriquecedor que los entornos virtuales 3D provean a los estudiantes de las herramientas necesarias para recrear, simular o resolver problemas por sus propios medios, que presentarles una simulación ya elaborada, pero que finalmente termine relegándolos a un rol mucho más pasivo. En este sentido, las simulaciones numéricas ofrecen un fascinante campo de aplicación en áreas básicas de la currícula de Ingeniería como el estudio de la Fisica y la Matemática.

En el caso particular del estudio de las *"Leyes del Movimiento de los Cuerpos"*, Papert afirma que la implementación de simulaciones permite una infinita cantidad de variaciones, por lo que incluso sería posible construir mundos virtuales en los que se cumplan leyes físicas completamente diferentes.

Esto le permitiría al estudiante proponer y contrastar diferentes modelos teóricos (Papert, 1980), desde la concepción Aristotélica del movimiento hasta la correspondiente a la Mecánica Clásica de Newton, e incluso experimentar con modelos más complejos como los propuestos por Einstein en su Teoría de la Relatividad (Kortemeyer et al., 2013).

3.5. Aprendizaje con Entornos Virtuales 3D

La teoría construccionista sostiene que el aprendizaje resulta más eficiente cuando el estudiante se involucra en un proyecto o en la construcción de algún material que es significativo para él, ya que esto produce en el estudiante un efecto de mayor compromiso, lo que en definitiva, permite desarrollar nuevas habilidades y enlazar el conocimiento nuevo con los saberes previos.

Bajo esta perspectiva, los entornos virtuales 3D operan como una extraordinaria herramienta para la adquisición de competencias y habilidades en un contexto simulado, permitiendo a los estudiantes aplicar los conocimientos teóricos adquiridos en el aula, en ambientes "*cuasi-reales*" representados en la computadora. Esto permite la generación de un modelo informático al que es posible someter a pruebas exhaustivas con la finalidad de comprobar su exactitud, sin el costo o el riesgo que implicaría comprobar en el mundo real, aquello que se afirma desde el punto de vista teórico. (Quinche y González, 2011).

3.6. Aspectos Tecnológicos

Los entornos virtuales 3D ofrecen un amplio espectro de posibilidades para la interacción social, la colaboración y las innovaciones en los procesos de enseñanza-aprendizaje. En este sentido, su utilización se ha visto facilitada por *aplicaciones Web*, que permiten el intercambio de archivos, reuniones virtuales, conferencias, seminarios y experimentos científicos (De Freitas, 2008).

Según Theodore Kotsilieris, estos ambientes mejoran la interacción y cooperación entre los estudiantes, debido a que los entornos inmersivos resultan particularmente adecuados en el contexto de los modelos de aprendizaje empírico y constructivista (Kotsilieris y Dimopoulou, 2013).

En este sentido, la experiencia de aprendizaje se ve enriquecida, debido a que estos entornos logran establecer canales de comunicación multimodales, es decir, a través de la percepción simultánea de varios sentidos. (De Freitas, 2008)

3.7. Marco de Análisis Comparativo

Establecer un marco para realizar un análisis comparativo entre diferentes entornos 3D requiere de la determinación de un conjunto de características propias de los entornos virtuales, para su evaluación. Sin embargo, la elección de este conjunto de características está fuertemente relacionada con el objetivo que se desea alcanzar a través de la utilización de dichos entornos.

La *Federation of American Scientist* (FAS) ha establecido un conjunto de características para evaluar, desde un punto de vista general, a los entornos virtuales 3D (De Freitas, 2008) y según se desprende del análisis de diversos trabajos de investigación sobre esta temática, existen diversas propuestas, según el criterio establecido por diferentes autores (Blas, Garzotto, y Poggi, 2009), (Tsiatsos, Konstantinidis, y Pomportsis, 2010).

En el presente trabajo se ha considerado que antes de establecer un marco comparativo, es necesario establecer claramente cual será el ámbito de aplicación, así como los objetivos a alcanzar mediante la utilización del mismo.

Puesto que los requerimientos de un entorno virtual 3D, enfocado en un contexto educativo de nivel primario con el objetivo de desarrollar ciertas habilidades sociales, serán completamente diferentes de aquellos necesarios para un contexto de nivel universitario y con un objetivo predominantemente tecnológico. (Escobar Gutiérrez , 2015)

En este sentido, el ámbito de utilización que se ha establecido para el presente trabajo, es un ambiente universitario, de carreras de Ingeniería, con estudiantes que se encuentran promediando su ciclo de estudios, y desde el punto de vista curricular, con el objetivo de simular sistemas dinámicos, con la intención de profundizar el conocimiento tanto del fenómeno en estudio, como de los métodos numéricos involucrados en su resolución.

En este contexto se ha establecido un marco comparativo, a partir del siguiente conjunto de características:

- Fidelidad de la Representación.
- Navegabilidad e Interacción con el Entorno.
- Comunicación entre Usuarios.
- Modelos 3D y contenidos Multimedia.
- Integración de Aplicaciones.
- Entorno de Programación.

3.7.1. Fidelidad de la Representación

La fidelidad de la representación es indispensable para lograr una experiencia *"espacial"* creíble, es decir, para que el usuario tenga la sensación de estar dentro de ese espacio, interactuando con todos los elementos que allí se encuentran. Las tarjetas de video actuales poseen una gran potencia de procesamiento, lo que les permite manejar representaciones gráficas de gran calidad, texturas y efectos de iluminación con gran velocidad y en forma prácticamente autónoma, lo cual resulta imprescindible para lograr una representación dinámica convincente.

A su vez, los objetos del entorno, así como las representaciones de los usuarios (avatars) necesitan comportarse consistentemente. Es decir, el usuario espera que dentro del entorno se cumplan ciertas "*leyes físicas*", como por ejemplo, que la intensidad de las fuentes sonoras presentes, varíe de acuerdo a la posición y la distancia que lo separa de las mismas *(sonido espacial)*.

3.7.2. Navegabilidad e Interacción con el Entorno

Por navegabilidad e interacción con el entorno, nos referimos a los mecanismos utilizados para recorrer los espacios y para modificar ciertos atributos, que afectan tanto al aspecto visual como a su comportamiento, así como también, la creación y manipulación de los objetos contenidos en el mismo.

Esta interacción, generalmente se realiza por medio de dispositivos de señalamiento estándar *(mouse, trackball, etc.)* y combinaciones de teclas, aunque en la medida que su costo disminuya probablemente se irá generalizando la utilización de dispositivos de navegación con acelerómetros incorporados (Wii ™), interpretación de movimientos (Kinect ™), o por medio de interfaces táctiles *(Dataglove)*.

En este sentido, la sensación de presencia en primera persona, exclusiva de los entornos virtuales permite generar nuevos aprendizajes, pues amplía las formas de adquirir conocimiento, en la interacción y transformación de ideas abstractas en representaciones perceptibles a través de objetos virtuales. (Fredes, Hernández, y Díaz, 2012)

3.7.3. Comunicación entre Usuarios

En lo que respecta a la comunicación entre usuarios en los entornos virtuales 3D, existen dos aspectos a considerar, la comunicación oral-escrita y la gestual.

La comunicación oral puede lograrse por medio de sistemas de voz a través de Internet, también conocidos como VoIP *(Voice over Internet Protocol)*; y la comunicación escrita por medio de herramientas de mensajería instantánea o IRC *(Internet Relay Chat)* integradas en la misma plataforma.

Por otra parte, una característica propia de los mundos virtuales es la incorporación de la comunicación gestual, la cual se logra dotando a las representaciones visuales de los usuarios *(avatars)* de un cierto conjunto básico de movimientos o gestos.

3.7.4. Modelos 3D y contenidos Multimedia

Los modelos 3D y los contenidos multimedia son recursos que producen una singular sinergia cuando se incorporan a los mundos virtuales. En el primer caso nos referimos a figuras, vehículos, edificios, etc., y en el segundo caso, a elementos multimediales tales como fotografías, archivos de audio y video. En todos los casos para poder integrarlos como recursos pedagógicos, se requiere que el entorno tenga la capacidad de cargarlos y reproducirlos adecuadamente.

3.7.5. Integración de Aplicaciones 2D

Para que un entorno virtual 3D no sea una aplicación aislada es conveniente que permita integrar aplicaciones externas dentro del mismo o al menos contar con la posibilidad de ejecutarlas aunque sea de forma remota. Esto permitiría a los estudiantes la utilización de navegadores de Internet, planillas de cálculo, editores de texto y otras aplicaciones 2D, sin necesidad de salir del "*espacio virtual*".

3.7.6. Entorno de Programación

Los procesos de aprendizaje y experimentación, como las simulaciones, son más efectivos y motivadores si los mismos pueden ser modificados y compartidos con otros estudiantes (Leask y Younie, 2001). Para lograr este cometido, es necesario contar con un lenguaje dinámico y un entorno de programación integrado que permita realizar modificaciones inmediatas, además de mecanismos de actualización adecuados para que todos los participantes experimenten un comportamiento consistente de la simulación en curso, aún a pesar de estar utilizando equipos de diferente potencia o de estar interconectados por redes con diferente ancho de banda.

3.8. Metodología de Trabajo

En la etapa de investigación sobre Mundos Virtuales correspondiente al presente trabajo, se realizó un relevamiento de los principales exponentes con el objetivo de analizar sus características más destacadas. A partir de la compilación realizada por quien se da a conocer en el *Metaverse* como *Ariane Barnes* (Barnes, 2012), se han incorporado otros entornos virtuales 3D considerados relevantes para este trabajo, obteniéndose como resultado final el listado de Mundos Virtuales 3D que se presenta en el **anexo C**.

A pesar de que los términos *Mundo Virtual 3D* y *Entorno Virtual 3D* se suelen utilizar indistintamente, conviene aclarar en este punto, que dichos términos no son sinónimos. Es decir, un *Mundo Virtual 3D* puede ofrecer facilidades de navegación, comunicación y juegos, pero si carece de facilidades para la creación de contenidos, personalización de espacios y programación del comportamiento de su entorno, no puede ser considerado como un *Entorno Virtual 3D*.

En la lista de *Mundos Virtuales 3D* que se muestra en el **anexo C**, puede observarse que la gran mayoría de los ejemplos citados pertenecen principalmente a dos categorías que podríamos denominar como "*de juegos y entretenimientos*" y "*de interacción social*".

En la primera categoría se observa que, por lo general, la libertad de acciones a tomar se encuentra restringida por lo establecido de antemano por el creador del juego. Mientras que en la segunda, el objetivo principal es brindar un lugar de encuentro para la interacción social entre los usuarios, por lo que la creación de contenidos o las facilidades de programación no suelen considerarse un requisito indispensable.

Teniendo en cuenta estos aspectos, y en concordancia con el alcance y los objetivos propuestos en el presente trabajo, se optó por tomar como objeto de estudio a los Entornos Virtuales 3D, *Second Life*, *OpenSimulator* y *OpenCobalt*, pues como veremos a continuación, cada uno de ellos presenta alguna característica en particular que lo diferencia de los demás.

En este sentido, *Second Life* (Life, 2003) fue elegido por ser uno de los entornos 3D más reconocidos, de mayor vigencia y cantidad de usuarios, contando en la actualidad con más de un millón de usuarios regulares. Este entorno posee, sin lugar a dudas, una clara finalidad comercial y de entretenimiento, aunque también es utilizado por diversas instituciones académicas para la creación de contenidos educativos, y principalmente para tener una presencia institucional en el mundo virtual.

El segundo Entorno Virtual considerado es *OpenSimulator*, un proyecto surgido a partir de la liberación del código fuente del programa cliente de *Second Life*, a comienzos de 2007, lo que lo convirtió en una especie de alternativa de código abierto de *Second Life*.

Esto representa un aspecto clave, ya que posibilita la instalación de servidores *(Grids)*, totalmente independientes de la empresa *Linden Labs*, e incluso abre la posibilidad de vincularlo con otros proyectos educativos de Software Libre, como por ejemplo *Moodle*.

Y por último, se ha elegido a *Open Cobalt*, un entorno muy poco conocido, cuya característica particular es que utiliza un esquema descentralizado de comunicaciones de tipo P2P *(Peer to Peer)*, que lo diferencia del modelo tradicional que utilizan la mayoría de los otros entornos virtuales 3D, los cuales administran todas las comunicaciones e interacciones a través de un servidor central.

Capítulo 4

Análisis de los Entornos Virtuales 3D

En el presente capítulo se examinarán detalladamente, cada una de las características establecidas en el marco de análisis comparativo propuesto. Las conclusiones de este análisis están principalmente basadas en los resultados obtenidos a partir de experiencias realizadas con cada uno de los entornos 3D analizados, además de la información obtenida de diversas fuentes bibliográficas.

4.1. Second Life

Second Life es uno de los entornos virtuales 3D multiusuario de más amplia trayectoria. Desarrollado por Linden Labs, empresa fundada en 1999 por Philip Rosedale *(alias Philip Linden)* y Andrew Meadows *(alias Andrew Linden)*, ha permanecido activo y en contínuo desarrollo desde su lanzamiento oficial el 23 de Agosto de 2003, por lo que representa una de las plataformas virtuales 3D de interacción social más madura. (Escobar Gutiérrez , 2015)

En este sentido, su creciente número de usuarios activos, en comparación con otras plataformas, así como también, la "*presencia virtual*" de prestigiosas instituciones académicas, refleja una posición destacada dentro del ámbito educativo. (Warburton, 2009)

4.1.1. Antecedentes de Second Life

Linden Labs, cuyo nombre proviene de la dirección en la que se hallaban originalmente sus oficinas (Linden Street 333, San Francisco, USA), se originó como una empresa dedicada al desarrollo de interfaces *hápticas*, es decir, hardware para aplicaciones de Realidad Virtual, a la vez que desarrollaban software específico para probar dichos dispositivos.

En el año 2000, *Linden Labs* desarrolló una aplicación que representaba un Mundo Virtual 3D, en el que los usuarios podían socializar en línea. Esta aplicación, denominada *Linden World*, sería la encargada de establecer el nuevo rumbo comercial de la empresa.

Figura 4.1: Linden World.
(Obtenido de http://secondlife.wikia.com)

En palabras de su propio creador, Philip Rosedale:

"*Linden World es fundamentalmente un lugar. Es un mundo orgánico en línea. Está construido por la gente que participa en él, que vive allí. Linden World es realmente un entorno virtual que vive en una red de servidores establecidos en algún lugar de San Francisco. Para acceder a Linden World, los usuarios con banda ancha sólo necesitan conectarse a la red. La tecnología propietaria desarrollada por Linden Labs permite la simulación de imágenes 3D realistas y texturadas en tiempo real. Pero a diferencia de la mayoría de los juegos en línea, tales como* ***EverQuest****, que está basado en un reino de fantasía, Linden World no posee un guión preestablecido*". (Rosedale, 2002)

De esta forma, el creador de *Second Life* marcaba la principal diferencia existente entre un entorno virtual como *Linden World* y los Juegos de Rol Multijugador en Línea *(MMORPG)*, tan populares en ese momento.

Sin embargo, *Linden World*, no parecía ser el nombre adecuado para transmitir la expansividad, la participación y la complejidad que se esperaba que caracterizara a este Mundo Virtual, a medida que fuera creciendo.

Robin Harper *(alias Robin Linden)*, otra de las integrantes del grupo original de *Linden Labs*, recuerda como fue el proceso de elección del nombre definitivo:

"*Uno de mis favoritos era* ***Sansara****, una palabra en Sánscrito, que para el Budismo significa, el eterno ciclo de nacer, morir y volver a nacer, haciendo alusión a un mundo en permanente evolución. Finalmente, analizando otras alternativas derivadas de* ***Terra****,* ***Viva*** *y* ***Life****, se retomó la idea de denominarlo* ***Life2****, y por último se decidió que el nombre* ***Second Life*** *era mucho más interesante y evocativo, ya que remite a la idea de un Mundo que evoluciona y crece tanto como la vida*". (Harper, 2002)

Luego de diez años de evolución, y con cerca de 38 millones de usuarios registrados, en abril de 2013, Philip Rosedale decidió crear una nueva compañía denominada High Fidelity Inc., en la que se encuentra desarrollando junto con Ryan Downe y Fred Heiberger, un nuevo proyecto denominado precisamente *High Fidelity* (HighFidelity, 2013).

El principal objetivo de este proyecto es crear la próxima generación de "*Mundos Virtuales*", poniendo especial énfasis en lograr un comportamiento más real de los *Avatars*, particularmente en lo referido al control de la comunicación gestual.

4.1.2. Análisis de sus Características

Second Life es un entorno virtual 3D basado en un esquema Cliente-Servidor. El servidor consta de diferentes módulos, los cuales se encargan de diversas tareas. El módulo de ingreso (Login Server) es un script CGI *(Common Gateway Interface)* encargado de manejar la verificación de los usuarios, determinando cual es la región en la que reside, o cual fue su última ubicación, con la finalidad de establecer la conexión con su respectivo módulo simulador. Otros módulos son el *Space Server* que se ocupa del manejo de los mensajes a través de las diferentes regiones, y el *Data Server* que interactúa con las bases de datos que almacenan los contenidos de las regiones y los datos de los usuarios.

Fidelidad de la Representación

La Fidelidad de la Representación involucra aspectos tan diversos como el aspecto de los objetos que componen las regiones, la apariencia de los *Avatars*, su comportamiento y el de los objetos fijos y móviles, etc. Gran parte de esta tarea es manejada por un proceso denominado simulador *(Simulator)*.

Cada simulador tiene la responsabilidad de establecer el comportamiento de los objetos que se encuentran dentro de una región, es decir, de un espacio que representa un área de 256 m x 256 m.

Para tener una idea de la magnitud de esta tarea, se calcula que actualmente existen más de 32.000 regiones en *Second Life*. Por lo tanto, si se necesita un simulador por región, y en promedio corren simultáneamente cuatro simuladores en cada servidor, se necesitan aproximadamente 8.000 servidores para mantener todas las regiones en línea.

La conexión con el servidor de *Second Life* se realiza a través de un programa cliente, mediante el cual se visualiza una región determinada. Por lo tanto, si nos desplazamos de una región a otra, es necesario que el programa cliente se conecte con el respectivo simulador de dicha región. Por esta razón, los simuladores de regiones adyacentes, se hallan a su vez comunicados entre sí por medio del protocolo UDP *(User Datagram Protocol)*, para disminuir los tiempos de latencia.

Los simuladores son también los encargados de ejecutar el motor de física *(Physics Engine)*, detectar colisiones, y mantener la información de estado de todos los objetos de la región con la finalidad de enviar dicha información al visualizador, quien se encargará de mostrar la escena desde el punto de vista del usuario.

Las simulaciones físicas de los objetos móviles son manejadas a través de la interfaz de programación de aplicaciones (API) del motor de física Havok *(Havok Physics Engine)*.

Según Dos Santos (Dos Santos, 2012), estas simulaciones se encuentran acotadas al ámbito de la mecánica clásica. Es decir, el comportamiento de los objetos es descripto principalmente en función de la influencia de las fuerzas externas aplicadas y de la interacción con otros objetos de la región.

Sin embargo, este comportamiento tampoco se rige exactamente por las reglas establecidas por la física Newtoniana. Por ejemplo, la masa de los objetos en *Second Life* depende de su tamaño y de su forma, pero no de la densidad del material que lo compone y en el caso particular de los *Avatars*, su masa depende sólo de su altura, por lo que agregar accesorios a un *Avatar* no altera su masa, salvo por los zapatos, ya que éstos si modifican su altura final (Dos Santos, 2009).

En lo que se refiere a la modificación del aspecto del propio *Avatar*, *SecondLife* permite realizar ciertas modificaciones sencillas a su vestimenta o su peinado por medio de un editor incorporado en el visualizador, como se observa en la figura 4.2.

Figura 4.2: Personalización del Avatar en Second Life.
(Obtenido de http://www.secondlife-shirts.com/images/tutorials/)

Aunque si se desea un aspecto más profesional, también se puede comprar con *Linden Dollars (L$)*, ropa, nuevos peinados o accesorios con diseños más sofisticados, en alguna de las tiendas de diseño virtuales, que se dedican a crear estos productos y que forman parte de la economía de *SecondLife*.

Navegabilidad e Interacción con el Entorno

En el esquema Cliente-Servidor que posee *Second Life*, el programa Cliente provisto por *Linden Labs*, denominado *SL Viewer*, es el que le permite al usuario conectarse con el Servidor y navegar por las diferentes regiones del entorno. En 2007, *Linden Labs* liberó gran parte del *código fuente* de este programa cliente, lo que dió lugar a que otros desarrolladores crearan sus propias versiones, como *Firestorm*, *Hippo*, *Imprudence*, *Singularity*, etc.

La tarea principal de los visualizadores es realizar las operaciones gráficas *(rendering)* de la escena. Los visualizadores se comunican con el servidor para obtener información sobre la ubicación y velocidad de los objetos que se encuentran en la región, aunque no son capaces de detectar colisiones entre ellos, ya que esta tarea es responsabilidad del motor de física y del simulador.

Figura 4.3: Mapa de Navegación en Second Life
(Obtenido de http://blog.inf.ed.ac.uk/atate/files/)

Además, los visualizadores proveen a los usuarios cierta información para facilitar la navegación por las diferentes regiones. En la Fig. 4.3 se puede apreciar en primer plano, un mapa que permite al usuario conocer su propia ubicación y la de otros usuarios cercanos, así como el tamaño y características de la región que habita.

Otra función del mapa es la de actuar como una forma rápida de desplazamiento, ya que por medio del mismo, se puede ubicar un lugar de interés y trasladarse inmediatamente al mismo por medio de la opción *Teleport*.

En *Second Life* un usuario puede caminar, correr e incluso volar, aunque una de las formas más elementales de desplazarse por una región, es utilizando las teclas de desplazamiento *arriba, abajo, izquierda* y *derecha*.

Sin embargo, para realizar desplazamientos con cierta precisión, lo usual es pasar al modo *Mouselook*, por medio de la rueda del mouse, en este modo la cámara muestra una vista en primera persona y el *Avatar* se desplazará hacia donde apunte el cursor del mouse.

Comunicación entre Usuarios

Como se ha mencionado, *Second Life* fue concebido fundamentalmente como un lugar de encuentro y participación social, por lo que los mecanismos de comunicación entre usuarios juegan un papel preponderante.

La comunicación a través de VoIP (Voice over Internet Protocol) por ejemplo, es una herramienta clave para la realización de diversos eventos, desde foros y congresos hasta representaciones teatrales (Squires y Benmessaoud, 2008).

Adicionalmente, *Second Life* posee otros mecanismos que complementan a la comunicación oral, con la escrita y la gestual, estos mecanismos son:

- **Text Chat** Es una de las formas de comunicación "*de proximidad*" entre usuarios, ya que lo que un usuario escriba es recibido sólo por aquellos usuarios que se encuentran en un radio de 20 metros *(virtuales)*. Si el usuario elige la opción *Gritar* (Shout) puede ampliar el radio de acción del mensaje a 100 metros *(virtuales)*, pero no se aconseja hacerlo, ya que es considerado un comportamiento grosero e inadecuado.

- **Instant Messaging** Los mensajes instantáneos, a diferencia del mecanismo anterior, son enviados directamente a un usuario en particular, simplemente apuntando a su *Avatar*, si ese usuario se encuentra a la vista. Si no es así, será necesario agregarlo como amigo o hacer una búsqueda por nombre. Los mensajes instantáneos pueden enviarse aún cuando los usuarios no estén conectados al mismo tiempo, en cuyo caso, el mensaje quedará pendiente hasta la próxima vez que se conecten.

- **Voice Chat** Es un mecanismo similar al Text Chat, pero de forma oral. Su utilización es sumamente sencilla y requiere apenas unos mínimos ajustes iniciales, antes de operar.

- **Postcards** Las tarjetas postales permiten enviar una imagen de la región que se esté visitando a cualquier persona por medio del correo electrónico, aún cuando el destinatario no tenga cuenta en *Second Life*.

- **Gestures** La comunicación gestual es un aspecto muy interesante de los mundos virtuales, ya que en el mundo real, este tipo de comunicación es tan o más importante que la comunicación oral o escrita. Para ello se cuenta con un menú, mediante el cual se puede elegir el gesto más apropiado para dar a entender a los demás usuarios cuales son nuestras intenciones. El repertorio de gestos es variado, y si se desea, se pueden crear nuevos gestos y agregarlos al menú.

Modelos 3D y contenidos Multimedia

En *Second Life* es posible generar todo tipo de objetos 3D a partir de figuras geométricas sencillas denominadas *prims*, las cuales pueden ser agrupadas para generar figuras más complejas. Además de los *prims*, existen otras estructuras denominadas *sculpted prims* o simplemente *sculpties*, y los modelos 3D de malla *(Mesh)*, los cuales pueden crearse externamente por medio de programas de diseño 3D, como *Blender* o *Maya*, para ser importados posteriormente.

Sin embargo, debido a que *Second Life* es un emprendimiento comercial, la creación y almacenamiento de contenidos propios de los usuarios, no es gratuito. Por lo tanto, los residentes que así lo deseen, pueden adquirir islas o parcelas de *terreno virtual* en las cuales asentar sus casas, edificios y demás posesiones. Estos terrenos virtuales tienen un costo mensual (en USD), que suele ser proporcional a su superficie y al límite de modelos o *prims* que pueden contener.

Figura 4.4: Recreación de espacios reales en Second Life
(Obtenido de http://www.journaldugeek.com/files/2014/06/)

Como puede apreciarse en la figura 4.4, *Second Life* posee un excelente nivel de detalle en lo que se refiere al aspecto estético de las representaciones visuales de paisajes, ambientes, edificios, modelos 3D y *Avatars*.

Esto permite recrear diversas situaciones y espacios del mundo real, siendo un aspecto muy importante para la generación de lugares de encuentro virtual, lanzamiento de productos comerciales, foros de discusión y actividades sociales, en las que el aspecto visual y participativo suele ser preponderante (Lim, 2009).

Integración de Aplicaciones 2D

A partir de la nueva versión del programa cliente de *Second Life* (Viewer 2.0), está disponible una nueva característica denominada *Shared Media*, mediante la cual es posible visualizar la información de un navegador web o un video de *Youtube*, como se observa en la figura 4.5, simplemente arrastrando y soltando la correspondiente URL (Uniform Resource Locator) sobre la superficie de un *prim*.

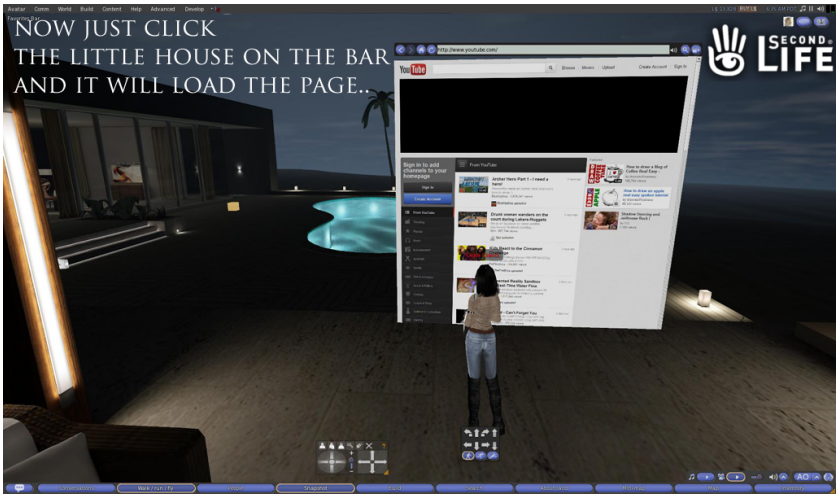

Figura 4.5: Integración de aplicaciones en Second Life
(Obtenido de
https://community.secondlife.com/t5/General-Discussion-Forum/Shared-media-on-a-prim/)

Esto también es posible si se vincula la dirección de un servidor VNC *(Virtual Network Computing)*, mediante el cual se pueden compartir aplicaciones en forma remota.

Entorno de Programación

En cuanto a las facilidades de programación, *Second Life* posee un lenguaje de programación basado en eventos y estados, denominado **LSL** *(Linden Scripting Language)* (Brashears, Meadows, Onderjka, y Soo, 2003). Este lenguaje, que posee una sintaxis similar al lenguaje C, permite responder a una serie de eventos. Los eventos pueden ser generados por el mismo sistema, como los timers, o por otros usuarios, como los eventos de presencia, de chat o email, y también pueden ser producidos otros objetos 3D, por ejemplo, en el caso que se produzca una colisión.

La interfaz de este lenguaje de *scripting* proporciona más de 30 tipos de eventos y más de 400 funciones, a partir de las cuales se escriben pequeños fragmentos de código denominados *scripts*, los cuales son interpretados y ejecutados por el servidor. En este sentido, el modelo de programación consiste en que cada objeto contiene sus propios scripts, que son los encargados de establecer el comportamiento del objeto.

Una de las principales críticas que se ha hecho a este modelo, es que al no existir el concepto de clase, ni de objeto, como se entiende dentro del paradigma de programación orientada objetos; cada elemento es un prototipo que responde reactivamente a ciertos *eventos*. Por lo tanto, sus limitaciones son evidentes, ya que no existen bibliotecas de scripts que permitan su reuso, ni control de versiones, y resulta muy difícil lograr desacoplar a los scripts, de los objetos 3D. (C. Lopes, 2014)

4.1.3. Aspectos destacables de Second Life

Más allá de sus características técnicas, el principal objetivo de *Second Life* es el de formar comunidades de usuarios *(residentes)*, para establecer lazos sociales y comerciales, razón por la cual, las principales compañias de diversos rubros han adquirido espacios virtuales para ofrecer sus productos y servicios. En la figura 4.6, se observa un resumen de la evolución de la economía de *Second Life*, en los últimos 10 años.

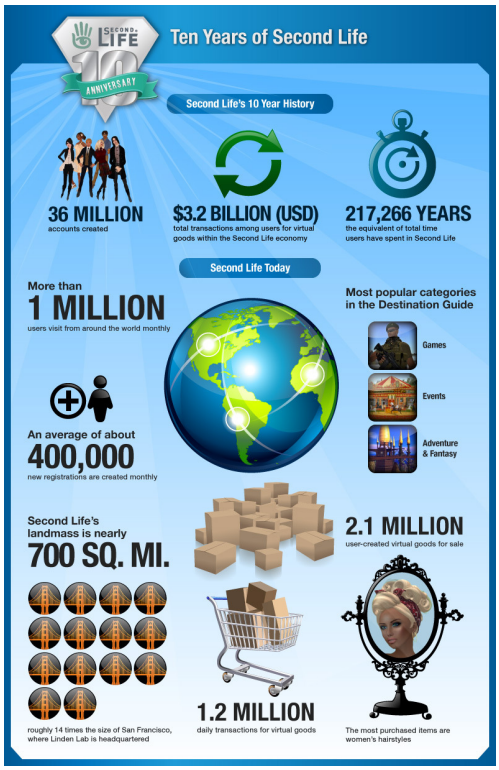

Figura 4.6: Second Life. 10 años en cifras
(Obtenido de http://gwynethllewelyn.net/2013/06/21/infographic-10-years-of-second-life/)

Una economía virtual que posee su propia moneda, el *Linden Dollar* (L$), que posee actualmente una paridad de aproximadamente 270 L$ por cada USD, y mediante el cual no sólo es posible, comprar y vender servicios sino también transformar fácilmente el dinero virtual en dinero real, y viceversa.

4.2. OpenSimulator

Si bien comunmente se hace referencia a *OpenSimulator* en forma abreviada como *OpenSim*, evitaremos hacerlo en el presente trabajo, para no confundirlo con una aplicación desarrollada por el *Centro de Computación Biomédica de la Universidad de Stanford*, para el modelado, análisis y simulación dinámica de estructuras músculo-esqueléticas humanas y animales, denominada precisamente *OpenSim* (Stanford University, 2007).

En realidad, *OpenSimulator* podría considerarse un clon, o mejor dicho, una versión de código abierto de *Second Life*, ya que posee muchas similitudes. Por esta razón, analizaremos sólo aquellos aspectos no tratados con anterioridad, y en los que *OpenSimulator* se diferencie de *Second Life*.

Cuando *Second Life* se inició en 2003, generó un fuerte impacto en esta área, sin embargo, su modelo propietario se transformó en el principal obstáculo para la evolución de una Internet 3D. En este sentido *OpenSimulator*, al ser un proyecto de código abierto, permite a los usuarios y desarrolladores crear y distribuir sus propios entornos inmersivos en una forma análoga a la de los sitios Web. (Oliver et al., 2013)

4.2.1. Antecedentes de OpenSimulator

El objetivo inicial *OpenSimulator* fue probar la factibilidad de construir un servidor de mundos virtuales 3D, al que fuera posible conectarse por medio del visualizador de *Second Life* y realizar ciertas acciones básicas.

Su desarrollo comenzó a partir de una biblioteca de funciones de código abierto llamada *libsl*, proveniente del código fuente del visualizador de *Second Life*, ya que el código fuente del servidor no está disponible, pues se trata de un código privativo propiedad de *Linden Labs*.

Por esta razón, en 2007 Darren Ward *(alias MW)* creó el proyecto *OpenSimulator* para desarrollar un nuevo servidor de mundos virtuales 3D, de código abierto, partiendo de los protocolos establecidos por *Second Life*, aunque sin utilizar ni una sola línea de código del mismo, por lo que esto derivó en una estructura interna completamente diferente.

Con el tiempo, el alcance del proyecto fue aumentando hasta superar con creces los objetivos planteados en sus inicios. En la actualidad *OpenSimulator* se perfila como un framework standard para construir entornos virtuales 3D. (Tarouco, Gorziza, Corrêa, Amaral, y Müller, 2013) Y si bien, aún mantiene la compatibilidad con el cliente de *Second Life*, se está trabajando junto con desarrolladores de otros programas cliente, ya que en el futuro espera soportar otros protocolos y entornos completamente independientes de *Second Life*.

4.2.2. Análisis de sus Características

OpenSimulator es un entorno virtual 3D de código abierto, multi-usuario y multi-plataforma, desarrollado en lenguaje *C# (C-Sharp)*. Por lo tanto, puede ejecutarse tanto en sistemas operativos Windows ™ utilizando el framework *.NET*, como en entornos de tipo UNIX por medio del framework *Mono* (Mono Open Source Project, 2004). Por lo tanto, al ser un proyecto de código abierto, los usuarios no están obligados a depender de un servidor provisto por *Linden Labs*, sino que cualquier persona que lo desee, puede instalar en su computadora su propio servidor *OpenSimulator*, crear su propio Mundo Virtual, y si lo desea, compartirlo a través de Internet.

Fidelidad de la Representación

En lo que se refiere a la estética gráfica y nivel de detalle, no se aprecia a primera vista, una gran diferencia con el aspecto que presenta *Second Life*.

La personalización de los *Avatars* es muy similar, así como también las herramientas que permiten construir objetos a partir de figuras primitivas. Sin embargo, uno de los aspectos menos desarrollados en cuanto a la fidelidad de la representación es el correspondiente al *comportamiento físico* de los objetos móviles dentro del mundo virtual.

Este comportamiento es controlado por un software específico denominado "*Motor de Física*", el cual se encarga de realizar los cálculos necesarios para simular un comportamiento similar al del mundo real. Más adelante, en la sección 5.2, se analizará este aspecto con mayor profundidad.

Navegabilidad e Interacción con el Entorno

La navegación dentro de una región de *OpenSimulator* es, en apariencia, muy similar a la navegación en *Second Life*. Sin embargo, debido a que *OpenSimulator* puede ejecutarse en diferentes modos: *Standalone*, *Grid* y *Hypergrid*, se presentan algunas diferencias.

Una vez instalado el servidor de *OpenSimulator*, el mismo se ejecutará por defecto en el modo *Standalone*, es decir, ejecutando un mundo virtual completamente aislado. Sin embargo, si se ejecuta en modo *Hypergrid*, se puede vincular ese mundo con otros mundo virtuales, y una vez establecidos los enlaces, será posible *teleportarse*, simplemente utilizando el hipervínculo establecido, aunque con la ventaja de conservar nuestros contenidos y objetos 3D, dentro de nuestra propia máquina.

De una forma muy similar al funcionamiento de los enlaces de hipertexto, cuando se produce una *teleportación* de una región a otra, a través del mapa, es probable que también se produzca una migración entre diferentes servidores, algo similar a lo que sucede en la Web cuando se pasa de una página a otra por medio de un hiperenlace.

Comunicación entre Usuarios

Los mecanismos de comunicación oral y textual de *OpenSimulator* son practicamente idénticos a los de *Second Life*, y si bien, los usuarios utilizan principalmente el Chat y los mensajes instantáneos, como se aprecia en la figura 4.7. La comunicación oral, a través de VoIP, junto con los gestos personalizables y sonidos asociados, configuran un medio de comunicación adicional, que reafirma la sensación de presencia (Konstantinidis, Tsiatsos, Demetriadis, y Pomportsis, 2010).

Figura 4.7: Comunicación entre usuarios en OpenSimulator.
(Obtenido de http://opensimulator.org/)

En este sentido, existe una serie de códigos visuales y gestuales propios de los entornos virtuales, que son rápidamente asimilados por los usuarios. Por ejemplo, cuando un *Avatar* alza su mano y un conjunto de partículas une su mano con un objeto, esto significa que ese usuario está construyendo.

En cambio, si el *Avatar* está inmóvil, con sus brazos extendidos, significa que el usuario está cambiando su apariencia física o su vestuario. (Levesque y Lelievre, 2011)

Modelos 3D y contenidos Multimedia

Los mecanismos de edición de objetos en *OpenSimulator* a partir de figuras primitivas *(prims)*, también son muy similares a los existentes en *Second Life*, como se aprecia en la figura 4.8, ya que esta funcionalidad es aportada mayormente por el visualizador, que es precisamente el elemento en común entre ambos entornos.

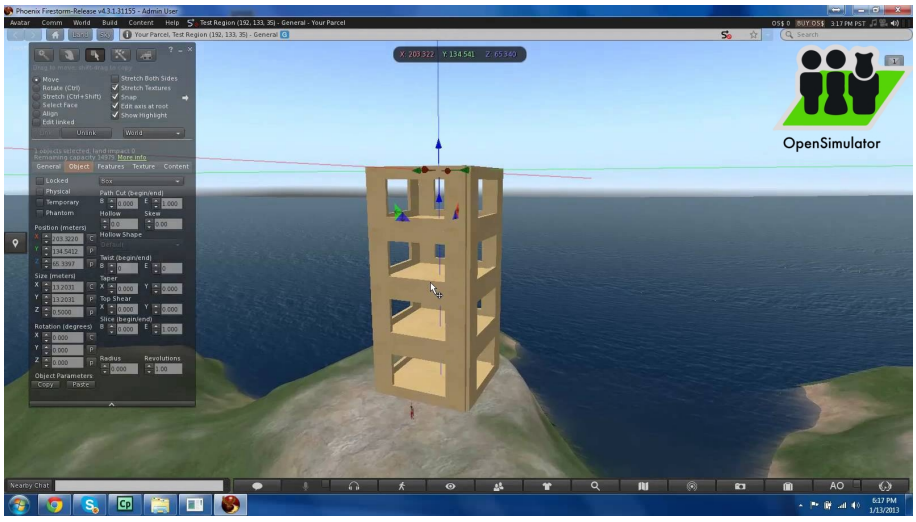

Figura 4.8: Editor de objetos 3D en OpenSimulator.
(Obtenido de http://i.ytimg.com/vi/FWBvSo5XVWA/maxresdefault.jpg)

En cuanto a la importación de modelos, inicialmente solo era posible importar modelos 3D como *sculpties*, sin embargo, posteriormente se incorporó la funcionalidad necesaria para importar modelos de malla *(Mesh)* en formato **.dae** *(Digital Asset Exchange)*, por lo que ahora, es posible importar modelos 3D creados previamente con programas de modelado como *Blender*.

En referencia a los archivos de audio o video, *OpenSimulator* reproduce los formatos más comunes **.wav**, **.mpeg**, etc., e incluso es posible difundir audio y video utilizando *streaming*. Esto se logra por medio de un servidor de *streaming* al que se accede directamente mediante el visualizador, por lo que no se produce un gran impacto en cuanto a la performance del simulador.

Integración de Aplicaciones 2D

En ocasiones, es necesario compartir aplicaciones 2D para realizar un trabajo colaborativo, sin necesidad de salir del entorno 3D, como se aprecia en la figura 4.9.

Figura 4.9: Integración de aplicaciones en OpenSimulator.
(Obtenido de http://knowsense.co.uk/blog/blog/_archives/2010/8/20/4609577.html)

Una de las formas de lograr esto es utilizando una estructura cliente-servidor conocida como **VNC** *(Virtual Network Computing)*. Se trata de un pequeño proceso que se ejecuta en el servidor, que permite los usuarios acceder e interactuar con las aplicaciones 2D disponibles, por medio de sus programas cliente.

Una forma sencilla de integrar aplicaciones 2D en *OpenSimulator*, es utilizando una pequeña aplicación, gratuita y de código abierto, denominada *Guacamole*, que se instala junto con el servidor VNC. Esta aplicación se comunica con el servidor VNC y realiza el rendering de su salida en formato HTML5, lo que permite que los usuarios utilicen aplicaciones de escritorio, dentro del entorno, por medio del visualizador de *OpenSimulator*.

Entorno de Programación

Los scripts en *OpenSimulator* están basados en **LSL** *(Linden Scripting Language)* con la intención de mantener cierto grado de compatibilidad. Sin embargo, no todas las funciones presentes en *Second Life* han sido implementadas, y por otro lado, se han incorporado algunas funciones nuevas.

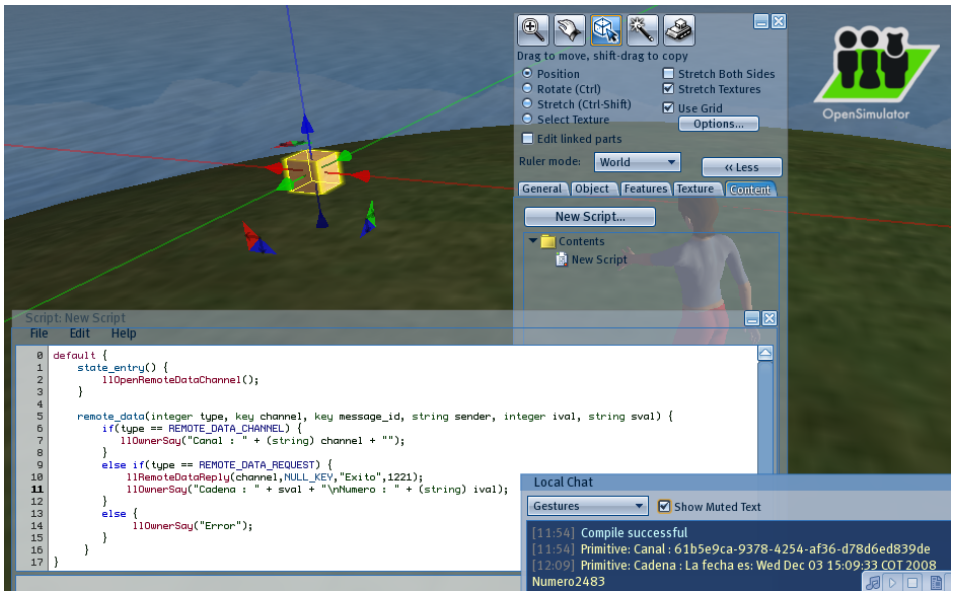

Figura 4.10: Programación de scripts en OpenSimulator.

Estos scripts son convertidos posteriormente a código **.NET**, mediante un mecanismo de compilación en tiempo de ejecución, conocido como **JIT** *(Just In Time)*, que aporta una mejora en la velocidad de procesamiento.

Como se aprecia en la figura 4.10, los objetos 3D pueden tener asociado uno o varios scripts, los cuales se ejecutan en respuesta a diferentes eventos, estableciendo así el comportamiento del objeto. Estos scripts creados por los usuarios son ejecutados en el servidor, mientras que el motor de gráficos 3D que recrea la escena, es ejecutado por el visualizador, de esta forma, cuando el servidor envía el resultado de la ejecución de un script, todos los usuarios cercanos pueden observar inmediatamente los cambios en la escena. (C. V. Lopes, Popov, Kan, y Morla, 2008)

4.2.3. Aspectos destacables de OpenSimulator

El hecho que *OpenSimulator* sea un proyecto de código abierto y que por lo tanto permita que cualquier persona, empresa o institución educativa pueda instalar sus propios servidores no es un datos menor. Pues además de ser una alternativa más económica en la mayoría de los casos, le otorga una completa independencia sobre la propiedad, el almacenamiento y el acceso de los contenidos generados.

Además, es posible configurarlo para diferentes modos de ejecución de forma tal de adaptar su funcionamiento a diferente tipo de requerimientos. Estos modos de operación son: *Standalone*, *Grid* y *Hypergrid*,

- ***Standalone***

  En este modo, la computadora local ejecuta un único proceso servidor, el cual proporciona todos los servicios básicos de un mundo virtual. De esta forma, el usuario puede utilizar un visualizador *(viewer)* para acceder al mismo, y hacer pruebas o generar contenidos dentro de dicho espacio. (Fishwick, 2009)

- ***Grid***

  En este modo, denominado grilla *(Grid)*, el mundo virtual está formado por varias regiones rectangulares contiguas. La grilla posee la información sobre la disposición espacial de las regiones y los límites de cada una, de forma tal, que el tránsito de una región a otra, sea totalmente transparente para el usuario.

  Asimismo, el servidor provee una serie de servicios denominados UGAIM *(User, Grid, Asset, Inventory, Messaging)*, que además de autentificar a los usuarios, administran los contenidos de cada región, conocen la ubicación de cada usuario y proveen los mecanismos de comunicación entre los mismos.

- ***Hypergrid***

  En este modo, un conjunto de simuladores se hallan conectados formando una estructura que se comporta de forma semejante al concepto de *Hipermedia*. En forma simplificada, podría decirse que la *teleportación* desde una región a otra, es un mecanismo equivalente al de los *hiperenlaces* de Internet. Aunque en realidad, la *teleportación* es un proceso mucho más complejo, ya que requiere de un intercambio mucho mayor de mensajes, además de la actualización del estado de la región.

Por cierto que la tarea de instalar un servidor *OpenSimulator* requiere de cierto esfuerzo, ya que es necesario conocer más profundamente su funcionamiento interno. Además, es necesario tener en cuenta que si bien es posible instalar un servidor para experimentar prácticamente sin ningún costo, si se desea que varios usuarios puedan construir mundos virtuales y compartir recursos durante las 24Hs esto requerirá invertir en un potente servidor.

4.3. OpenCobalt

OpenCobalt es un entorno virtual tridimensional, multiusuario, colaborativo y multiplataforma, íntegramente desarrollado en una versión moderna de *Smalltalk* (Goldberg y Robson, 1989), llamada *Squeak* (Ingalls, Kaehler, Maloney, Wallace, y Kay, 1997).

A diferencia de los entornos anteriormente analizados, que poseen una arquitectura centralizada, *Open Cobalt* utiliza un esquema de comunicaciones de tipo **P2P***(Peer to Peer)* para interconectar los mundos virtuales, por lo que no requiere de un servidor central. En este esquema, cada computadora *(nodo)* puede funcionar en modo *Standalone* o puede conectarse con otros usuarios para compartir un mismo espacio virtual.

4.3.1. Antecedentes de OpenCobalt

En el año 2001, el Dr. Alan Kay junto con un equipo integrado por David A. Smith, Andreas Raab y otros colaboradores, iniciaron un proyecto denominado *Croquet* (D. Smith, Kay, Raab, y Reed, 2003), con el objetivo de crear una plataforma de experimentación y generación de espacios virtuales basada en un nuevo paradigma de interfaces de usuario.

Según afirma Alan Kay, "*Croquet fue construido para responder a una sencilla pregunta. Si tuviéramos que crear un nuevo sistema operativo y su interfaz de usuario, conociendo lo que hoy conocemos. ¿Hasta dónde podríamos llegar? ¿Qué tipo de decisiones podríamos tomar hoy, que hubiéramos sido incapaces incluso de considerar hace 20 o 30 años, cuando se crearon inicialmente los sistemas operativos actuales?* ". (Denker, 2005)

En torno a esta premisa, varias universidades, institutos de investigación y empresas privadas aportaron los recursos necesarios para este proyecto.

Este esfuerzo conjunto, permitió concretar en el año 2006 una versión beta del *Croquet SDK* (Software Development Kit), un entorno virtual 3D, implementado íntegramente en *Squeak*, que proveía la infraestructura básica y de comunicaciones necesaria para crear e interconectar mundos virtuales.

Resulta interesante notar que, tanto *OpenCobalt* como los proyectos que lo precedieron (ver figura 4.11), fueron desarrollados básicamente bajo la misma premisa. Proveer los medios necesarios para facilitar la expresión de ideas y promover la experimentación como una forma de mejorar el aprendizaje.

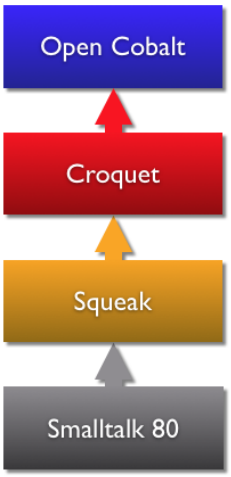

Figura 4.11: Evolución histórica y conceptual de Open Cobalt.
(Obtenido de www.opencobalt.org)

Paralelamente a mediados de 2006, algunos de los integrantes del proyecto *Croquet* crearon la empresa *Qwaq*, posteriormente renombrada como *Teleplace*, con el objetivo de construir aplicaciones 3D colaborativas basadas en dicho entorno, pero con un objetivo comercial. (Teleplace, 2009)

Este emprendimiento produjo cierta interacción positiva en el desarrollo de ambos proyectos (Hut, 2008), aunque lamentablemente gran parte del código desarrollado por *Teleplace*, no pudo ser liberado como código abierto, por cuestiones relacionadas con las licencias.

Así fue como a comienzos de 2008, un equipo de desarrolladores de la Universidad de Duke, entre los que seencontraba Mark McCahill (McCahill y Lombardi, 2004) y John Dougan iniciaron junto con el Dr. Julian Lombardi, la implementación de un nuevo proyecto de código abierto basado en *Croquet*, para la creación de espacios virtuales 3D colaborativos, denominado *OpenCobalt*, liberando en 2010, su versión *Alpha 1.0*. (OpenCobalt, 2010)

4.3.2. Análisis de sus Características

Como se mencionó en la sección 4.3, *OpenCobalt* es un entorno tridimensional descentralizado desarrollado íntegramente en *Squeak* (Guzdial y Rose, 2001), una versión moderna de *Smalltalk*.

Smalltalk (Goldberg y Robson, 1989) es un lenguaje de programación orientada a objetos puro, desarrollado en los laboratorios de investigación de Xerox, por el equipo de investigación liderado por el Dr. Alan Kay, quien fuera galardonado en 2003 con el *ACM Turing Award* por la creación del lenguaje *Smalltalk*, y por establecer las bases de lo que hoy se conoce como *Programación Orientada a Objetos*. (Sproull, 2010)

En este sentido, *OpenCobalt* sigue la línea iniciada con *Smalltalk* y luego continuada por *Croquet*, es decir, convertirse principalmente en un marco de comunicación, no sólo entre computadoras, sino como un medio para crear, expresar y transmitir conocimiento entre seres humanos. (Ingalls, 1981)

Fidelidad de la Representación

A diferencia de los entornos anteriormente analizados, los mundos virtuales en *OpenCobalt* no ocupan regiones rectangulares contiguas, sino que cada mundo virtual representa un espacio completamente independiente, incluso aquellos que se encuentran dentro de una misma computadora.

De este modo, es posible crear y personalizar un mundo virtual desde cero o modificar alguno de los espacios pre-diseñados. Entre las posibilidades de personalización, se pueden modificar las características de la luz ambiente, agregar luces puntuales, eligiendo la ubicación, la orientación y el color de las mismas, modificar las texturas tanto del suelo como de la bóveda celeste *(Skybox)*, agregarle movimiento a la misma e incluso crear irregularidades al terreno mediante el uso de algoritmos fractales.

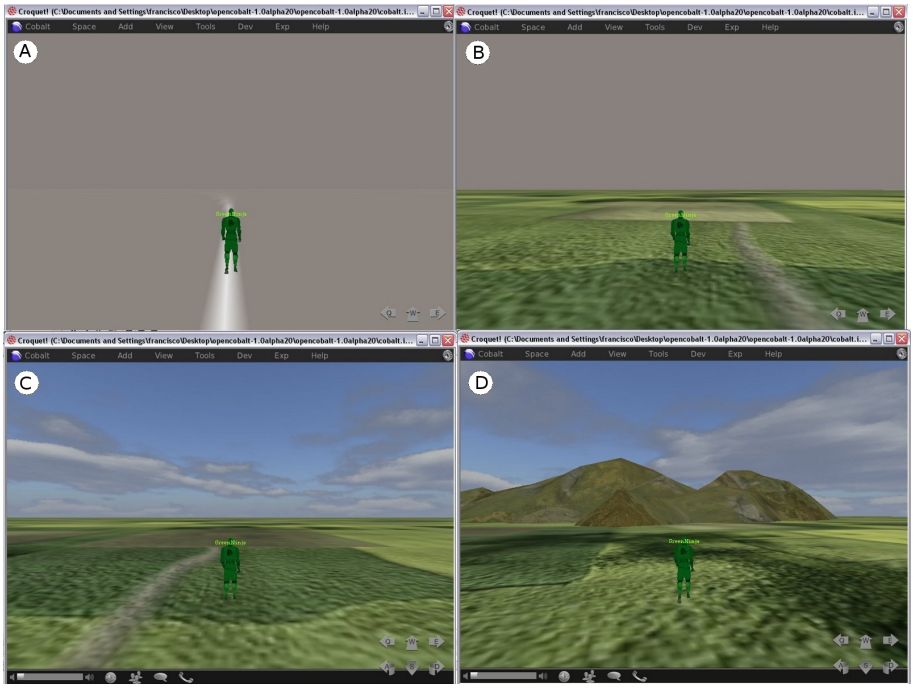

Figura 4.12: Personalización de un espacio en Open Cobalt

En la figura 4.12 se aprecia una secuencia de imágenes que muestran la personalización de un espacio, en diferentes etapas. Inicialmente se observa sólo al avatar dentro de un espacio vacío **(A)**, luego se agregó la textura del suelo **(B)**, posteriormente la textura de la bóveda celeste **(C)**, a continuación se agregaron montañas generadas por medio de algoritmos fractales, y por último se generó una leve neblina para dar mayor realismo a la escena **(D)**.

Si bien *OpenCobalt* cuenta con los elementos básicos para crear mundos virtuales completamente diferentes, de forma muy rápida y sencilla, es evidente que en este aspecto, no posee el grado de sofisticación de otros entornos como *Second Life* o *Blue Mars*, los cuales exhiben un nivel estético y de detalle claramente superior.

Navegabilidad e Interacción con el Entorno

Dentro de un espacio virtual 3D, la navegación se realiza en forma similar a los *juegos en primera persona*, en los que el usuario, representado por su *Avatar*, se desplaza utlizando ciertas teclas para avanzar, retroceder o girar, o haciedo uso de los *botones de navegación*, que se encuentran en la parte inferior derecha de la pantalla (Lombardi y McCahill, 2005).

Además, como los mundos virtuales de *OpenCobalt* no ocupan regiones contiguas, para pasar de un espacio a otro, es necesario utilizar un *portal*. Esto incorpora un concepto interesante, ya que si bien cada espacio posee un tamaño determinado y por lo tanto es posible recorrer una cierta distancia dentro del mismo, cuando se atraviesa un portal, la distancia entre ambos espacios virtuales es prácticamente nula.

En este sentido, un *portal*, no es simplemente una ventana que muestra, en tiempo real, lo que está sucediendo en el otro espacio, sino que al atravesarlo, se ingresa instantáneamente en el otro espacio.

En la figura 4.13, se puede apreciar el punto de vista de un usuario, cuando se encuentra en el espacio en el que se halla el edificio del Cabildo **(A)**, desde donde se puede observar un *portal* que conecta con un mundo virtual submarino. Luego de atravesar dicho portal e ingresar dentro del espacio submarino **(B)**, se puede observar a través del portal, lo que está sucediendo en el espacio de origen.

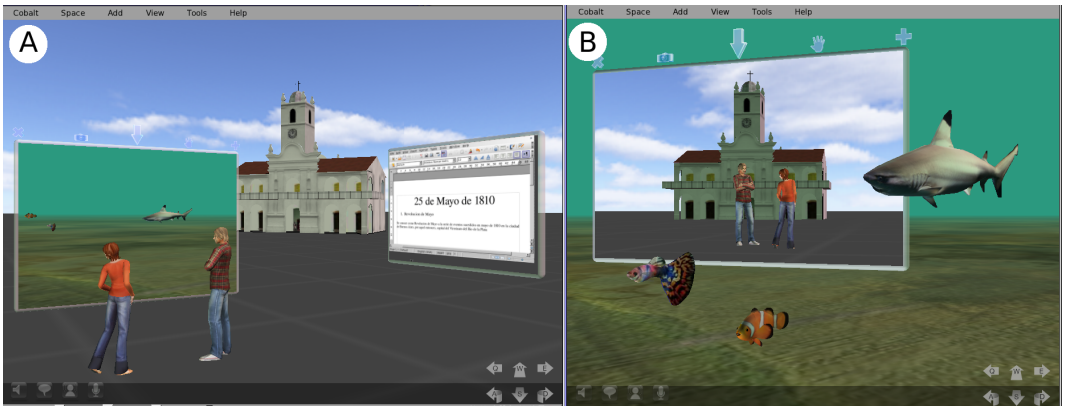

Figura 4.13: Portales entre mundos virtuales en OpenCobalt.

En este sentido, como es posible tener varios portales abiertos simultáneamente, y cualquier usuario puede pasar de un mundo virtual a otro, simplemente atravesando el portal correspondiente; es probable que luego de atravesar varios portales se experimente una cierta sensación de desorientación, ya que en *OpenCobalt* no existen mapas de navegación que le indiquen al usuario donde se encuentra el portal que lo devuelva a su espacio de origen. Por lo tanto, en caso de que no se recuerde cómo realizar el camino inverso, se puede recurrir a la opcion *Space ->Return Home* del menú principal, para regresar instantáneamente al espacio de partida.

Comunicación entre Usuarios

Las herramientas de comunicación son un elemento clave para llevar adelante cualquier actividad de tipo colaborativa, por lo que en este sentido, *OpenCobalt* proporciona dos opciones de comunicación textual.

Una local, para establecer comunicaciones dentro del mismo espacio, y otra externa, por medio del protocolo XMPP *(eXtensible Messaging and Presence Protocol)*, anteriormente conocido como *Jabber*, que es un protocolo de comunicaciones abierto basado en XML *(eXtensible Markup Language)*.

En cambio, para la comunicación oral entre usuarios, utiliza VoIP (Voice over Internet Protocol), aprovechando además, las ventajas del sonido espacial que provee *OpenAL*, una biblioteca de funciones de audio multiplataforma, desarrolladas por *Creative Labs*™ para la ejecución de audio posicional y multicanal en tres dimensiones.

El *sonido 3D* o *sonido espacial* permite escuchar más fuerte a las voces de los usuarios cuyos *Avatars* se encuentren más cercanos y más suave a aquellos más alejados, así como también, diferenciar entre aquellos que se hallen a la derecha o a la izquierda de nuestra posición.

Modelos 3D y contenidos Multimedia

En el año 2005, David Smith y su equipo desarrollaron en *Croquet*, el antecesor de *OpenCobalt*, un framework para crear y editar figuras 3D denominado *Wicket*. Este editor estaba incorporado en un tipo especial de portal, y mediante un conjunto de herramientas de edición, permitía entre otras cosas, la creación de objetos por extrusión. *Wicket* fue concebido como una herramienta de CAD *Computer Aided Design* colaborativa para probar un nuevo concepto de interfaces. Lamentablemente esta herramienta no pasó de la etapa experimental y actualmente no se encuentra disponible en *OpenCobalt*. (D. Smith, Raab, Ohshima, Reed, y Kay, 2005)

Sin embargo, *OpenCobalt* soporta la importación de modelos 3D en diversos formatos estándar, tales como **.ase**, **.kmz**, **.obj** y **.vrml**. Estos modelos pueden crearse previamente con programas externos específicos como *Blender*, *Cheetah3D*, *Trimble SketchUp*, etc., y también pueden incorporarse modelos completos, obtenidos de sitios como *Google 3D Warehouse*. Finalmente, para ajustar ciertos detalles, *OpenCobalt* posee una herramienta incorporada, denominada *3D EditBox*, que permite trasladar, escalar y rotar los modelos.

Estos modelos pueden ser estáticos y complejos, como los monolitos de *Stonehenge* y el helicóptero, que se aprecian en la figura 4.14, ó dinámicos como es el caso de los *Avatars* y otros objetos móviles.

Figura 4.14: Incorporación de modelos 3D en OpenCobalt

En lo que se refiere a la interacción con otros contenidos multimedia, *OpenCobalt* permite importar imágenes en formatos **.bmp**, **.jpg**, **.gif** y **.png**, así como también, reproducir archivos de audio en **.wav** y video compatible con **.mpeg**.

Integración de Aplicaciones 2D

Aplicaciones 2D mono-usuario comunes como navegadores web, procesadores de texto, planillas de cálculo, etc., pueden utilizarse dentro de un entorno multi-usuario colaborativo, por medio del cliente VNC *(Virtual Network Computing)* que *OpenCobalt* posee incorporado.

De esta forma, varios usuarios que se encuentren compartiendo el mismo espacio, podrían escribir sobre el mismo documento utilizando simultáneamente el mismo procesador de textos, o interactuando de forma colaborativa con cualquier otra aplicación, sin necesidad de salir del entorno.

Como ejemplo de integración de aplicaciones 2D, en la figura 4.15 se aprecia una ventana en la que está corriendo, vía VNC, un navegador web *(Firefox)*. En este caso en particular, se observa cómo se accede directamente desde un espacio 3D, al contenido de la página web de la asignatura *Análisis Numérico para Ingeniería*, que se dicta en la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata,

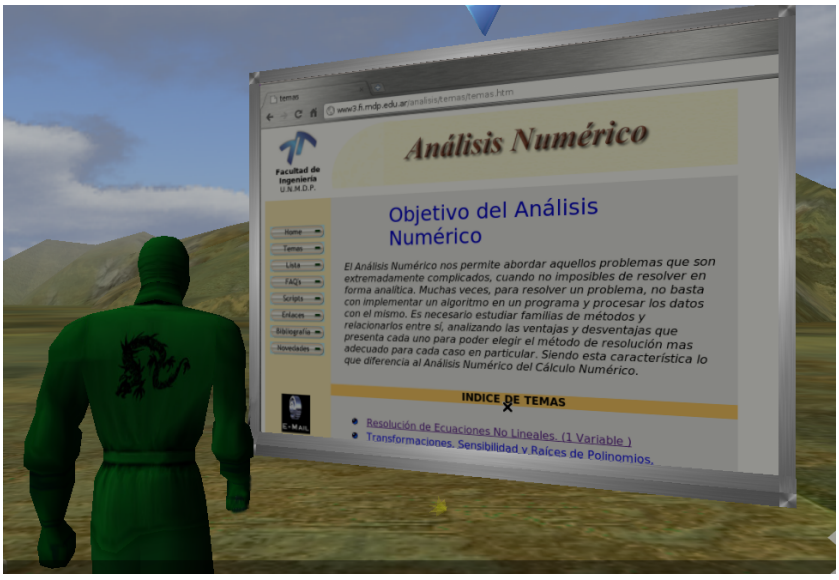

Figura 4.15: Integración de aplicaciones en OpenCobalt.

Precisamente el presente trabajo está íntimamente relacionado con los temas tratados en dicha asignatura, pues los modelos matemáticos que describen a los sistemas dinámicos suelen expresarse por medio de sistemas de ecuaciones diferenciales.

Como se verá con mayor detalle en la sección 5.4.1, el objetivo principal de esta asignatura es el análisis de las ventajas y desventajas de los métodos que permiten resolver estos sistemas, pues precisamente esos métodos son los que en definitiva, permiten simular numéricamente el comportamiento de dichos sistemas dinámicos en un entorno virtual 3D.

Entorno de Programación

Una de las ventajas de que *OpenCobalt* esté íntegramente programado en *Squeak*, es que todas las herramientas de programación y depuración dinámicas propias de un ambiente *Smalttalk*, están disponibles para el usuario de *OpenCobalt*.

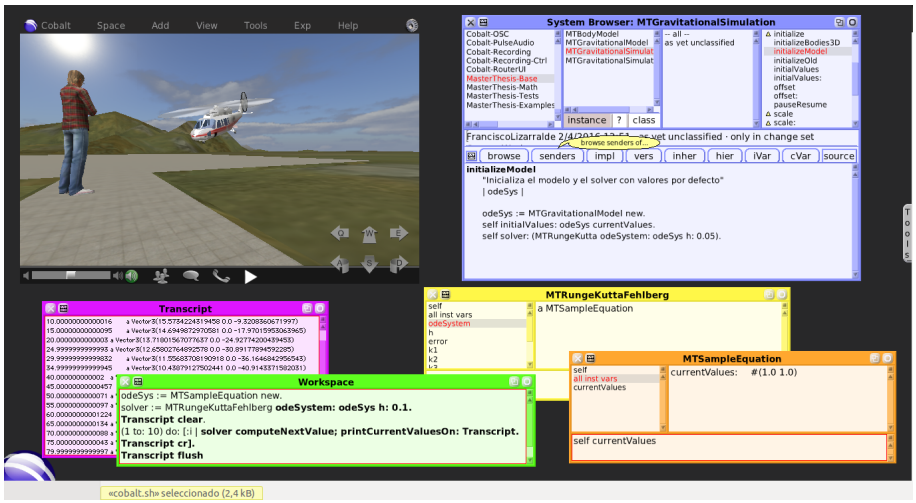

Figura 4.16: Entorno de Programación de OpenCobalt.

En la figura 4.16 se aprecian algunas de estas herramientas, tales como los *Inspectores de Objetos* que permiten observar el estado de los objetos en tiempo real, los *Browsers de Clases* que permiten acceder a todo el código fuente del sistema, los *Espacios de trabajo* en los que se pueden ejecutar pequeños fragmentos de código, etc., siendo todos accesibles en forma inmediata, y en tiempo de ejecución.

Squeak (Black, Ducasse, Nierstrasz, y Pollet, 2007), es un lenguaje de programación orientado a objetos, basado en clases, con capacidades de programación reflexiva y metaprogramación, que se ejecuta sobre una *Máquina Virtual* específica para cada plataforma, lo que hace que el código desarrollado sea totalmente portable entre diferentes sistemas operativos.

El lenguaje de programación *Squeak* mantiene la misma premisa básica de *Smalltalk*, en donde, "*Todo es un objeto y la interacción entre objetos sólo se realiza a través del envío de mensajes*".

Una idea simple, que permite abordar la resolución de problemas desde una perspectiva más conceptual, además de brindar una significativa claridad semántica a la programación.

Finalmente, una ventaja comparativa interesante, es que en *Squeak* no existe el tradicional ciclo de *Edición-Compilación-Ejecución*, de otros lenguajes de programación. Por lo tanto, es posible realizar modificaciones en el código y ver inmediatamente el resultado de los cambios, aún mientras todo el sistema continúa en ejecución.

Programación Colaborativa

Una de las formas más antiguas y tradicionales de almacenar el código programado en *Smalltalk*, fuera del sistema *(image)*, consistía en grabar dicho código en forma de archivos de texto, denominados *(fileouts)*. Por lo tanto, al tratarse de archivos de texto, era muy sencillo copiarlos, para luego cargarlos *(fileIn)* y ejecutarlos en otra máquina ó para distribuirlo a otros programadores.

Esta forma de programación colaborativa un tanto rudimentaria, hacía bastante difícil la tarea de integrar las modificaciones, ya que cuando un proyecto involucra a varios programadores y éstos realizan cambios en el código, es necesario controlar la evolución de las diferentes versiones, por lo que en la actualidad se utilizan herramientas específicas para esta tarea.

OpenCobalt utiliza un sistema concurrente de distribución de versiones, denominado *Monticello*, el cual como se aprecia en la figura 4.17, se encuentra integrado dentro del entorno.

Monticello almacena el código de un proyecto, como un archivo comprimido *(package)* y permite grabarlo tanto en forma local, dentro del directorio **package-cache**, así como también en diferentes repositorios, accesibles via HTTP ó FTP, a través de Internet.

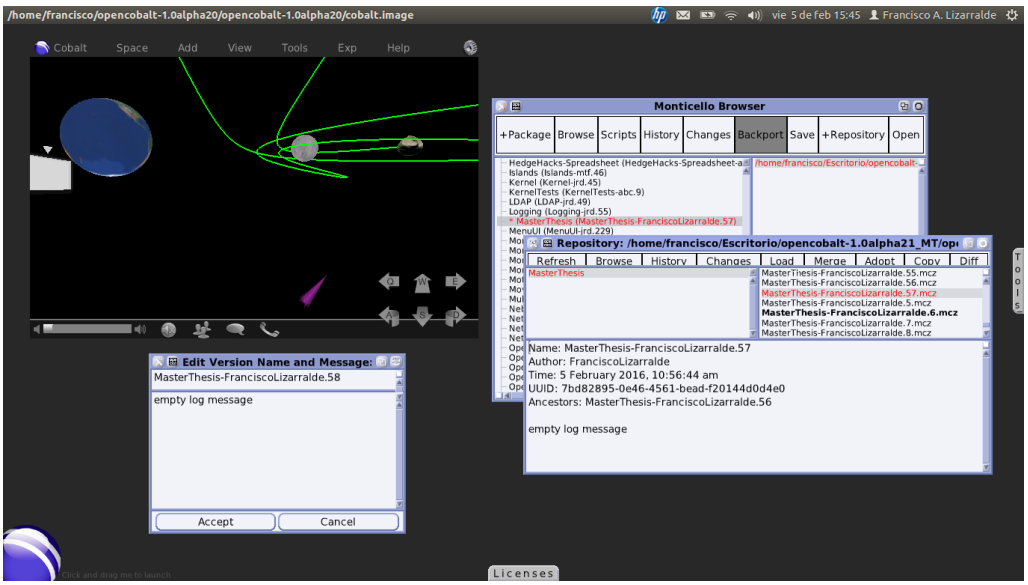

Figura 4.17: Monticello Browser en OpenCobalt.

Esto permite una rápida distribución en grupos de desarrollo, a la vez que cuenta con una funcionalidad similar a CVS *(Concurrent Versioning System)*, en lo que se refiere a control de versiones (Black et al., 2007).

4.3.3. Aspectos destacables de OpenCobalt

Una de los aspectos más destacables de *OpenCobalt*, con respecto a los otros entornos virtuales 3D analizados, es precisamente su esquema descentralizado de comunicaciones, el cual permite que los diferentes mundos virtuales se interconecten directamente entre sí, sin la necesidad de que exista un servidor central.

En este sentido, Rick McGeer, uno de los integrantes del equipo de desarrollo de *Croquet*, afirma que si bien el procesamiento centralizado resuelve los problemas de sincronización e interacción entre los nodos, se puede demostrar que este esquema transforma al servidor en un "*cuello de botella*" de los procesos de cómputo y comunicación, limitando rápidamente su escalabilidad. (McGeer, Raab, Reed, Smith, y Kay, 2006)

Además, desde un punto de vista económico, un esquema descentralizado suena bastante prometedor, ya que permitiría desarrollar mundos virtuales interconectados con un costo relativamente bajo, pues no sería necesario mantener grandes servidores para proveer la interconexión de los espacios de los usuarios y el almacenamiento de contenidos. (Lombardi y Lombardi, 2010)

Ahora bien, como no existe un servidor que administre las comunicaciones entre los nodos, es necesario que exista algún mecanismo de sincronización y computación distribuida para mantener un comportamiento consistente en cada uno de los mundos virtuales interconectados.

Esto se consigue, gracias a una versión modificada del protocolo de comunicación **P2P** *(Peer to Peer)*. Esta versión, desarrollada por David P. Reed, denominada *TeaTime* (Reed, 2005), posibilita la replicación de cómputo a través de múltiples nodos. Este protocolo hace que la replicación de cómputo sea tan sencilla como la replicación de datos, con lo que se obtiene una inmediata propagación de los cambios hacia todos los nodos interconectados. (Wright y Madey, 2009)

Sin embargo, el Dr. Andreas Raab, uno de los desarrolladores principales de *Croquet*, consideró que el proyecto original concebido por David Reed, *TeaTime*, involucraba dos aspectos diferentes. Uno consistía en un protocolo de sincronización de mundos virtuales, y el otro era un protocolo de comunicación **P2P** *(Peer to Peer)*.

Por este motivo, Andreas Raab y David Smith, crearon una nueva versión de dicho esquema, denominada *Hedgehog*, enfocada principalmente en los aspectos de sincronización (Raab, 2010).

Este cambio produjo una mejora sustancial, ya que en el diseño original, para N participantes se requerían N^2 conexiones, lo que hacía totalmente inviable su utilización con un gran número de usuarios. Para solucionar este problema, Andreas Raab y David Smith incorporaron en el sistema, la funcionalidad de un *router*, que en la práctica se implementó como una red de ruteo superpuesta, la cual teóricamente asegura, que un mismo mundo virtual pueda ser compartido por una gran cantidad de usuarios.

En la sección 5.3.1, se explicará con mayor profundidad este esquema, que posee un beneficioso efecto colateral. Debido a que los mensajes a los objetos son transmitidos a los demás nodos conectados al mismo espacio, en el mismo orden en que deben ser ejecutados, la latencia que podría existir en algún nodo debida a un menor ancho de banda o hardware menos potente, no crea problemas de sincronización. (D. A. Smith, Raab, Reed, y Kay, 2006)

Es decir, el comportamiento que se observará en cada nodo será idéntico, algo crucial en el caso de una simulación, ya que a lo sumo, sólo se percibirá una menor velocidad de ejecución en aquellas máquinas con menor potencia.

4.3.4. Resultados del Análisis Comparativo

A lo largo de este capítulo se han descripto y analizado cada uno de los aspectos establecidos como Marco de Análisis Comparativo, en la sección 3.7, destacándose para cada aspecto, las diferencias observadas entre los entornos analizados. Este análisis tiene como objetivo evaluar si los entornos analizados cuentan con los requerimientos necesarios para la implementación de simulaciones, como recurso educativo en la enseñanza de la Ingeniería.

Como resultado del análisis comparativo, puede afirmarse que, si bien los entornos analizados reúnen los requisitos mínimos necesarios para la implementación de simulaciones educativas, la elección del más adecuado dependerá principamente del objetivo buscado y de su ámbito de aplicación.

Para lograr una mayor claridad en cuanto a la exposición de dichos resultados, se ha considerado conveniente condensar parte de la información de las secciones precedentes, en un formato mas compacto, como puede apreciarse en los cuadros 4.1 y 4.2.

En el caso de *Second Life*, su principal ventaja se encuentra en su excelente calidad gráfica, personalización de espacios y *Avatars* y sobre todo en los recursos destinados a la comunicación entre usuarios, conferencias, creación de foros, streaming, etc. Sin embargo, por tratarse de un emprendimiento principalmente comercial, muchos de estos recursos tienen un costo, que puede resultar restrictivo para algunas instituciones o usuarios particulares.

En este sentido, *OpenSimulator* se presenta como una alternativa de código abierto que permite que cualquier institución o individuo pueda montar su propio servidor de Mundos Virtuales y administrarlo sin mayores costos, y algo más importante, almacenando los contenidos desarrollados, en su propia máquina.

Este aspecto es realmente importante, ya que cuando los contenidos se almacenan en servidores de terceros, cualquier cambio ya sea de los aspectos técnicos ó de las políticas de la empresa, pueden conducir a la pérdida de innumerables horas de trabajo, por parte de los usuarios. (Yardley, 2013)

	Fidelidad Representación		**Interacción Entorno**		**Comunicación Usuarios**	
SecondLife	Calidad Gráfica	Excelente	Interacción	Visualizador	Text Chat	Integrado
	Modific. Avatar	Muy Buena	Desplaz. Avatar	Camina-Vuela	Voice Chat	VoIP
	Motor de Física	Havok ™	Mapa Navegación	SI	Gestures	SI
OpenSimulator	Calidad Gráfica	Muy Buena	Interacción	Visualizador	Text Chat	Integrado
	Modific. Avatar	Muy Buena	Desplaz. Avatar	Camina-Vuela	Voice Chat	Mumble
	Motor de Física	Bullet ™	Mapa Navegación	SI	Gestures	SI
OpenCobalt	Calidad Gráfica	Buena	Interacción	Integrada	Text Chat	XMPP
	Modific. Avatar	Elemental	Desplaz. Avatar	Camina	Voice Chat	VoIP
	Motor de Física	No posee	Mapa Navegación	NO	Gestures	NO

Cuadro 4.1: Resumen del Análisis Comparativo (Parte 1)

	Modelos 3D / Multimedia		Integración Aplicaciones		Entorno de Programación	
SecondLife	Formatos 3D	DAE	Acc. Remoto	LLMedia	Cód. Abierto	Solo Cliente
	Audio / Video	WAV / MP4	C. Streaming	SharedMedia	Lenguaje	LSL (Scripting)
	Editor 3D	Excelente	Ap. 2D Integ.	NO	IDE Integrado	LSL Editor
OpenSimulator	Formatos 3D	DAE	Acc. Remoto	ScreenLeap	Cód. Abierto	Acceso Total
	Audio / Video	WAV / MP4	C. Streaming	SharedMedia	Lenguaje	LSL (Scripting)
	Editor 3D	Muy Bueno	Ap. 2D Integ.	NO	IDE Integrado	LSL Editor
OpenCobalt	Formatos 3D	DAE, KMZ	Acc. Remoto	VNC	Cód. Abierto	Acceso Total
	Audio / Video	MP3 / MP4	C. Streaming	NO	Lenguaje	Smalltalk
	Editor 3D	Elemental	Ap. 2D Integ.	SI	IDE Integrado	Squeak

Cuadro 4.2: Resumen del Análisis Comparativo (Parte 2)

Si bien *Second Life* y *OpenSimulator* están programados internamente en lenguajes como `C++` o *C#*, la programación dentro del entorno se realiza escribiendo funciones en LSL *(Linden Scripting Language)*, y asociándolas directamente a los objetos 3D.

Programar el comportamiento de objetos individuales tiende a ser en un principio, relativamente simple, sin embargo, pronto se aprecia la carencia de soporte para la programación de grandes conjuntos de objetos, lo cual impide proveer cierto nivel de coordinación entre los mismos. (C. Lopes, 2014)

En este sentido, *OpenCobalt* permite programar no sólo el comportamiento individual de los objetos 3D, sino también la creación de las clases necesarias para coordinar aquellas tareas que no poseen una representación gráfica concreta, pero que brindan el marco necesario para poder escalar un proyecto o hacer frente a cambios en los requerimientos.

Capítulo 5

Nuevas Tecnologías aplicadas a la Enseñanza de la Ingeniería

En este capítulo se presentará un ejemplo de aplicación de las nuevas tecnologías de la Información y las Comunicaciones (NTICs) en la enseñanza de la Ingeniería, en particular, sobre la utilización de un entorno virtual 3D para la implementación de simulaciones numéricas de sistemas dinámicos. Asimismo se analizarán aquellos aspectos técnicos y matemáticos involucrados en la implementación de simulaciones, así como también, las ventajas pedagógicas de su utilización.

En la sección 5.1 se presentará una introducción a los Motores de Física, describiéndose en detalle, en la siguiente sección, las características principales de aquellos utilizados por los entornos *Second Life* y *OpenSimulator*.

A continuación, en la sección 5.3 se describirá el esquema denominado *"replicación de cómputo"* utilizado por *OpenCobalt*, para obtener un comportamiento idéntico en todos los equipos interconectados, ya que este mecanismo representa una de las principales características que lo diferencian de los entornos centralizados antes mencionados.

Finalmente, a partir de la sección 5.4, se presentará el desarrollo completo de una simulación numérica y su posterior implementación en *OpenCobalt*, como una forma de exponer y evaluar las capacidades de un ambiente muy poco descripto en la bibliografía.

5.1. Simulaciones en Entornos Virtuales 3D

En la enseñanza de la Ingeniería se suele analizar el comportamiento dinámico de diversos sistemas, así como también, se pone énfasis en el estudio de los modelos matemáticos que los representan. Según Kypuros los estudiantes suelen *"perderse"* en dichas representaciones, lo que dificulta el aprendizaje de los conceptos fundamentales (Kypuros y Connolly, 2005). Pareciera ser que, medios tradicionales como el pizarrón o los libros de texto, resultan insuficientes para expresar la naturaleza dinámica de estos sistemas.

En este contexto, los entornos virtuales 3D emergen como un medio adecuado para la experimentación y la representación, tanto dinámica como espacial, de diversos fenómenos físicos, ya que brindan la posibilidad de simular, experimentar y analizar el comportamiento de dichos sistemas dinámicos, bajo diferentes condiciones.

Por otra parte, existen fenómenos que por su escala, ya sea microscópica o astronómica, o que por su costo o peligrosidad, resultan imposibles de reproducir en el ámbito de un laboratorio. En cambio, es posible estudiar el comportamiento de tales fenómenos, simulando numéricamente las leyes físicas que los gobiernan, dotando a los objetos del entorno virtual 3D de propiedades físicas, y estudiar su comportamiento bajo la influencia de campos gravitatorios, eléctricos, magnéticos, etc. (McCahill, Moore, Wendland, y Zampogna, 2006).

La simulación cualquier fenómeno físico o proceso dinámico, como el desplazamiento de un vehículo, la trayectoria de un cohete o la órbita de un cuerpo celeste bajo la influencia de un campo gravitatorio, permite además, el abordaje de diversos temas relacionados, tales como:

- **Modelo Matemático** La dinámica de los fenómenos físicos es representada, por lo general, por un conjunto de ecuaciones diferenciales, las cuales describen su comportamiento a medida que transcurre el tiempo.
- **Resolución Numérica** Para obtener la solución del sistema de ecuaciones diferenciales que describen el comportamiento del fenómeno, se suelen emplear métodos numéricos, ya que la mayoría de los problemas reales no pueden resolverse analíticamente.
- **Representación** Además de calcular la solución numérica que determina la respuesta dinámica del sistema, es necesario representarla visualmente, transfiriendo ese comportamiento a los objetos del espacio virtual 3D.

Por otra parte, desde el punto de vista didáctico, la creación de una simulación por parte de los estudiantes, desde la representación del modelo matemático del fenómeno hasta la elección del algoritmo de resolución numérica más adecuado y su posterior visualización, permite reforzar el aprendizaje mediante el principio denominado *"Learning by doing " (Aprender haciendo).*

En este sentido, estimular a los estudiantes a que intenten crear colaborativamente sus propias simulaciones, provoca un efecto movilizador y generador de un aprendizaje más significativo, ya que los involucra en un rol activo de creadores, de investigadores, en lugar de permanecer relegados al rol pasivo de usuarios de un producto cerrado e inmodificable.

Según Papert *"Aprendemos mejor haciendo... pero aprendemos mejor aún, si combinamos nuestro hacer con hablar y reflexionar acerca de lo que hemos hecho "* (Papert et al., 1999). En este sentido, los entornos virtuales 3D colaborativos parecen poseer todas las herramientas necesarias para llevar a la práctica este principio.

En la próxima sección analizaremos una de estas herramientas, conocidas como *"Motores de Física "*, mediante las cuales, ciertos entornos como *Second Life* y *Open Simulator* controlan el "*comportamiento físico*" de los objetos contenidos en sus mundos virtuales.

5.2. Motores de Física y Entornos Virtuales 3D

Algunos Entornos Virtuales 3D hacen uso de un *Motor de Física* para simular ciertas leyes físicas de movimiento y establecer el comportamiento de los objetos móviles presentes en sus espacios. En este sentido, un *Motor de Física* es un conjunto de algoritmos y funciones especializadas, cuya responsabilidad podría resumirse en la ejecución de las siguientes tareas básicas. (Laurell, 2008)

- Detectar las colisiones entre objetos.
- Resolver la acción a tomar en caso de detectarse una colisión.
- Calcular las fuerzas aplicadas sobre cada objeto.
- Integrar y calcular los valores de velocidad y posición de los objetos en el próximo estado.

Por lo tanto, la función principal de un *Motor de Física* es brindarle al usuario una experiencia, lo más aproximada posible a la que esperaría en la vida real.

Es decir, que más allá de las luces, las formas y las texturas que determinan la apariencia de los objetos en el *Mundo Virtual*, el usuario espera que además, se comporten como si los objetos fueran sólidos, tuvieran masa, etc., ya que sin estas restricciones, nada impediría que los objetos se traspasaran unos a otros en lugar de colisionar entre sí, o que permanecieran flotando inexplicablemente a cierta distancia del suelo.

Manejo de colisiones

En las simulaciones en entornos virtuales 3D se suele buscar una solución intermedia entre velocidad y exactitud. La mayoría de los objetos que forman parte del mundo virtual están basados en formas primarias como cubos, esferas, cilindros y conos, o constituidos por una malla *(Mesh)*. Estas mallas, conformadas por una gran cantidad de polígonos, por lo general, triángulos o cuadrángulos denominados *Quads*, permiten crear objetos más complejos y con mucho mayor realismo y detalle.

Sin embargo, cuanto más complejo es el objeto, mayor será la cantidad de cálculos requeridos para detectar una colisión con otro objeto. Una solución sencilla a este problema consiste en recubrir a un objeto complejo con un objeto primario transparente, pero que posea una forma adecuada como para enmascararlo completamente. Así, el motor de física realizará los cálculos de colisiones sólo tomando en cuenta la forma del objeto simple que envuelve al objeto complejo. A esta simplificación se la denomina *geometría de colisión*.

Optimización del procesamiento

Si bien en el mundo real las Leyes Físicas se aplican todo el tiempo, muchas veces no es necesario replicar este comportamiento en el mundo virtual, ya que esto implicaría una cantidad enorme de cálculos innecesarios.

Además, debido al error inherente a los algoritmos utilizados para el cálculo de la posición de los objetos, éstos pueden presentar un leve movimiento contínuo en lugar de permanecer quietos una vez alcanzada su posición de reposo. Por esta razón es usual desactivarlos, eliminándolos de la lista de objetos móviles del motor de física, una vez que su movimiento se ha reducido por debajo de cierto valor.

Precisión en los cálculos

Uno de los principales límites que enfrentan los motores de física para lograr simulaciones realistas, es la precisión en los cálculos de los valores que representan la posición de un objeto móvil y de las fuerzas que actúan sobre él. Si la precisión no es adecuada, el redondeo acumulado en los cálculos sucesivos termina por producir comportamientos erróneos e inesperados. Estos errores se ven magnificados en el caso de objetos que se mueven libremente pero que están sujetos entre sí a una distancia menor a la precisión que el motor de física es capaz de manejar, lo cual suele producir resultados que no se condicen con un sistema conservativo de la energía. Una forma sencilla de reducir este error es aumentando la precisión, aunque esto conlleva a un mayor costo computacional. Otra forma, consiste en utilizar algoritmos especializados, como los que se describen en la **sección 3** del **apéndice A**.

Velocidad de los objetos y cuadros por segundo

Un aspecto importante en la visualización de una simulación, es la cantidad de cuadros por segundo que pueden generarse para brindar una sensación de continuidad en el movimiento de los objetos. Precisamente, el intervalo de tiempo entre cuadro y cuadro es el que determina la precisión con la que se calculan las posiciones de los objetos en cada instante.

Si la velocidad de los objetos es demasiado elevada con respecto al intervalo de tiempo utilizado en los cálculos, se observa que los objetos parecen saltar casi en forma instantánea de una posición a otra, pero aparentemente sin haber pasado por las posiciones intermedias.

5.2.1. Motores de Física en Second Life

Desde sus comienzos *Second Life* eligió al motor de física *Havok* ^{TM} para simular el comportamiento de los objetos dentro de su entorno virtual. *Havok* es una compañía, recientemente adquirida por *Intel* ^{TM}, que se especializa en el desarrollo de motores de física para videojuegos y películas de animación, logrando un alto desempeño gracias a la utilización de la aceleración provista por la GPU *(Graphics Processing Unit)* de las placas gráficas de última generación.

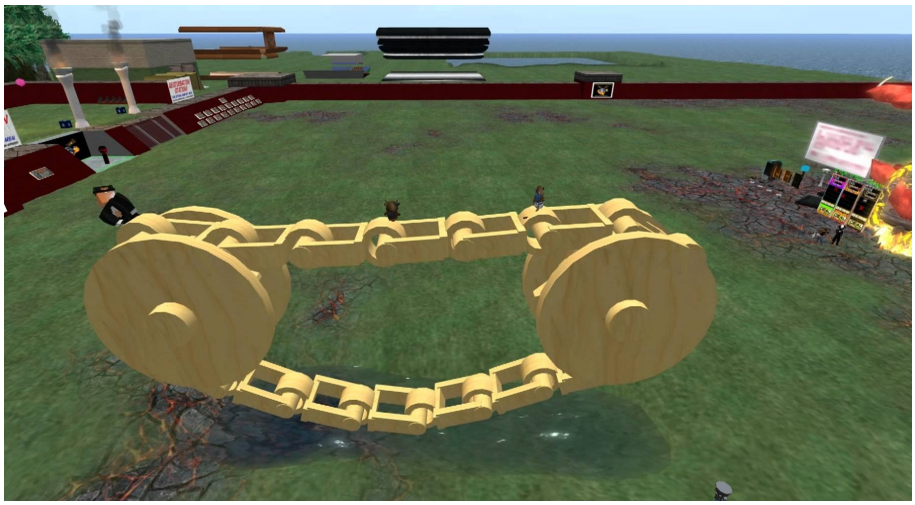

Figura 5.1: Simulación Física con Havok en Second Life.
(Obtenido de http://i.ytimg.com/vi/DSuxu-329T8/maxresdefault.jpg)

En el caso de *Second Life*, la tarea de *Havok* consiste en proveer al simulador las funciones *(Middleware)* necesarias para calcular el comportamiento físico de los objetos contenidos en el entorno.

En este sentido, es importante hacer notar que, en el caso eventual que *Linden Labs* decidiera liberar el código del simulador de *Second Life*, no podría hacer lo mismo con la parte correspondiente al motor de física *Havok*, ya que se trata de un producto de otra compañía.

Si bien actualmente existen otras alternativas, incluso libres o de código abierto, *Second Life* aún posee una vinculación muy estrecha con *Havok*. Recién a partir de la actualización a *Havok 4* en 2008, y a *Havok 7* en 2010, se comenzó a trabajar en una capa de abstracción entre el simulador y el motor de física, que eventualmente permitiría reemplazarlo por otras alternativas.

En lo que se refiere a la detección de colisiones, *Second Life* suele utilizar para los cálculos la misma geometría del objeto. Si bien esto funciona bien para los objetos primitivos *(Prims)*, no resulta lo más conveniente para el caso de objetos más complejos, compuestos por varios *Prims*. Esto se debe a que en *Second Life*, la *geometría de colisiones* no se encuentra separada de la geometría propia del objeto, sino que por el contrario, está determinada por el conjunto de *Prims* que lo conforman.

Por este motivo, si bien la detección de colisiones suele ser más precisa que la de algunos videojuegos, esto conlleva un costo de procesamiento mucho mayor, lo que hace más lentas estas operaciones. Por esta razón, *Havok 4* simplificó el cálculo de colisiones para el caso particular de las esferas, tratándolas matemáticamente como si fueran esferas perfectas, en lugar de los poliedros multifacetados con que se las representa visualmente en el entorno, mejorando notablemente la velocidad de procesamiento.

En cuanto al cálculo de la posición, *Second Life* considera que cualquier objeto que se encuentre apoyado en el suelo y no se mueva una distancia mínima durante dos segundos aproximadamente, se considerará quieto.

Por lo que se deshabilitarán los cálculos del motor de física para el mismo, permaneciendo en dicha posición hasta que sea colisionado por otro objeto activo.

Hiper-Realismo en Second Life

Según su creador Philip Rosedale, una de las principales razones del éxito de *Second Life*, *"es el hecho que ofrece un conjunto de capacidades, que son de muchas formas diferentes y hasta superiores a las del mundo real "*. En este mismo sentido, Dos Santos afirma que *Second Life* no se rige exactamente por las reglas establecidas por la física Newtoniana, sino que elige deliberadamente, no adoptar un comportamiento exactamente igual al del mundo real, alterando dicho comportamiento para transformarlo en *"hiper-real*". (Dos Santos, 2012)

Esta diferencia de comportamiento con el mundo real brinda también la posibilidad de experimentar con objetos que se comporten de forma completamente diferente, o que obedezcan a otro tipo de leyes físicas.

Por ejemplo, si se desactiva el motor de física de *Second Life* y se programa un nuevo comportamiento para cada objeto por medio de scripts en LSL *(Linden Scripting Language)*, podría abrirse un infinito abanico de posibilidades a explorar.

Esto permitiría experimentar, no sólo por diversión, sino dando rienda suelta a la imaginación y preguntarse críticamente sobre la veracidad de las leyes físicas aprendidas en la escuela.

O plantearse:

¿Qué pasaría si la fuerza de gravedad tuviera otra dirección?

¿Y si la masa de los objetos o el transcurrir del tiempo dependiera de su velocidad?

Probablemente, una de las características más interesantes de los entornos virtuales 3D, sea brindarnos esa posibilidad de experimentar en un mundo "*hiper-realista* " o mejor dicho "*surrealista* ". (Dos Santos, 2012)

5.2.2. Motores de Física en OpenSimulator

OpenSimulator generalmente tiende a seguir los lineamientos establecidos por *Second Life*, sin embargo, en el caso de las simulaciones físicas, y por tratarse de un software de código abierto, ha optado por brindar a sus usuarios y desarrolladores, la posibilidad de utilizar diferentes motores de física.

Inicialmente, *OpenSimulator* utilizaba un motor de física extremadamente sencillo que sólo se encargaba de que los objetos permanecieran quietos al hacer contacto con el suelo. Aunque no detectaba las colisiones entre objetos, y por lo tanto, éstos no se comportaban como si fueran sólidos, sino que sencillamente pasaban unos a través de otros.

Figura 5.2: Simulación Física con ODE en OpenSimulator.
(Obtenido de http://opensimulator.org/mantis/view.php?id=2874)

En este sentido, actualmente es posible utilizar *OpenSimulator* con motores de física externos, como por ejemplo, ODE *(Open Dynamics Engine)*.

A partir de Junio de 2014, BPE *(Bullet Physics Engine)* se transformó en el motor de física por defecto de *OpenSimulator*, ya que posee un mejor desempeño que ODE y un mayor nivel de realismo, sobre todo en lo referente al desplazamiento de vehículos.

Según Justin Clark-Casey, uno de los desarrolladores de *OpenSimulator*, "*Bullet es mucho más eficiente que ODE. Para la misma potencia de hardware, es posible manejar muchos más avatars y objetos en movimiento con Bullet que con ODE. Además, en los últimos años, Bullet ha recibido un desarrollo mucho más activo que ODE* "(Korolov, 2014).

En *OpenSimulator*, el principal rol del simulador es brindar al usuario una experiencia verosímil y lo más cercana a la de la vida real. Sin embargo, el simulador sólo conoce la posición de los objetos, junto con referencias a las texturas que los recubren, pero no conoce nada acerca de velocidades, aceleraciones, fuerzas aplicadas o acciones a tomar en caso de colisiones, ya que de estas cuestiones se encarga el *motor de física*, que realiza los cálculos necesarios para que simulador realice las actualizaciones correspondientes.

En el caso de los objetos inmóviles no es necesario realizar ningún tipo de procesamiento especial a menos que un objeto móvil impacte contra ellos. En tal caso, el motor de física lo tratará temporariamente como si fuera un objeto móvil en reposo que es golpeado por otro objeto móvil, determinando las acciones a seguir. Sólo en aquellos casos en los que la situación sea imposible de resolver o signifique una importante sobrecarga para el simulador, éste tomará la decisión de desactivar el comportamiento de los objetos físicos involucrados, para resguardar al sistema de un posible efecto de bloqueo. Un *objeto físico* es aquel que forma parte del conjunto de objetos considerados por el motor de física para calcular sus nuevas posiciones y para determinar las acciones a seguir en caso de colisión.

Uno de los casos más simples de colisión, se presenta cuando dos objetos que se acercan mutuamente alcanzan una posición en la que sus representaciones se solapan, en ese caso, el motor de física debe calcular la nueva posición de ambos objetos teniendo en cuenta las fuerzas que se ejercen mutuamente. Si los objetos son compactos, el choque producirá una separación de los mismos y a lo sumo un cambio de trayectoria.

Sin embargo, si se trata de un objeto complejo formado por gran cantidad de *Prims* ensamblados, pueden presentarse casos muy complicados de resolver, sobre todo si existen huecos en la geometría del objeto.

Otro problema delicado al que debe enfrentarse el motor de física en *OpenSimulator*, es la simulación del efecto de la fuerza de gravedad, o mejor dicho de la aceleración gravitatoria, para presentar un comportamiento similar al del mundo real. En este sentido, cualquier objeto que no posea ninguna fuerza aplicada, descenderá "por acción gravitatoria" hasta chocar contra otro objeto o superficie que se interponga o de lo contrario seguirá descendiendo hasta detenerse en la posición cero en el eje Z. Por lo tanto, para evitar el efecto que se vean objetos suspendidos inexplicablemente en la posición cero del eje z, es que se coloca un plano en dicha posición, representando el suelo.

Un efecto interesante se produce cuando los objetos caen y chocan con el suelo, ya que en este caso, el motor de física deberá resolver la colisión y aplicar sobre el objeto una fuerza en dirección contraria que lo hará despegarse del suelo y elevarse. Sin embargo, deberá considerar además la pérdida de energía que normalmente se produce en la realidad, ya que de no ser así, el objeto continuará rebotando indefinidamente.

La apariencia mayormente plana del entorno *OpenSimulator* en sus primeras versiones se debía principalmente a que el motor de física que utilizaba, sólo era capaz de resolver las colisiones de los objetos contra el suelo.

Si bien esto hacía posible que los Avatars pudieran caminar de un forma aparentemente real, no les permitía subir una escalera formada por *Prims*, sin embargo, esto se solucionó a partir de la utilización de nuevos motores de física como *Bullet* ^{TM}.

5.3. El caso particular de OpenCobalt

A diferencia de *Second Life* y *OpenSimulator* que utilizan un sistema de sincronización centralizado, *OpenCobalt* utiliza un mecanismo mediante el cual, tanto los objetos como el código relacionado con los mismos, son enviados a los diferentes nodos para su procesamiento local, manteniendo la secuencia de ejecución original.

Los diferentes espacios o Mundos Virtuales, denominados *Islands* (Islas) en la terminología de *OpenCobalt*, se hallan interconectados a través del protocolo de comunicaciones P2P *TeaTime* (Reed, 2005). Este protocolo, heredado de *Croquet*, el antecesor de *OpenCobalt*, constituye la base de un sistema diseñado para posibilitar interacciones idénticas, en tiempo real, entre varios usuarios distribuidos en la red (Takada, 2007).

Si bien en el año 2008 se desarrolló un plugin para ODE *(Open Dynamics Engine)* en *Squeak*, con la intención de incorporar un *"Motor de Física"* en *Croquet*, lamentablemente el proyecto fue abandonado.

Aparentemente el principal inconveniente que impide la utilización de un Motor de Física externo es la incompatibilidad con el esquema de replicación de código, que es la base del funcionamiento de *OpenCobalt*, por lo que probablemente la única solución consista en desarrollar un Motor de Física propio, completamente integrado con el entorno, lo cual aún sigue siendo un asunto pendiente.

Sin embargo, a pesar de que *OpenCobalt* no posee un Motor de Física integrado que establezca el comportamiento general de los "objetos físicos" del mundo virtual, posee un completo sistema de desarrollo integrado y un lenguaje de programación orientada a objetos de propósito general, mediante el cual se pueden implementar simulaciones de cualquier tipo de fenómeno.

A continuación, analizaremos en detalle el mecanismo de replicación de cómputo utilizado por *OpenCobalt* para lograr una secuencia de ejecución idéntica, en cada una de las máquinas conectadas a un mismo mundo virtual.

5.3.1. Funcionamiento interno de OpenCobalt

El diseño de *OpenCobalt* garantiza que los cambios que se produzcan en un espacio virtual *(Isla)* en particular, serán idénticos a los de cualquier otra réplica de dicha Isla (D. A. Smith et al., 2006). En este sentido, puede afirmarse que mientras la mayoría de los sistemas utiliza un mecanismo de *replicación de datos*, en *OpenCobalt* se aplica un esquema de *replicación de cómputo.*

Conforme con este mecanismo, los objetos internos de una Isla pueden enviarse mensajes mutuamente y exhibir el mismo comportamiento que cualquier otro objeto, ya que aparentemente poseen los mismos privilegios. La única diferencia es que no pueden enviar mensajes fuera del ámbito de la Isla, ni recibir *mensajes directos* de otros objetos fuera de la Isla.

Sin embargo, en aquellos casos en que sea necesario enviar mensajes desde un objeto externo hacia un objeto dentro de la Isla o viceversa, puede utilizarse un objeto especial, denominado *FarRef* (Referencia Lejana), que existe fuera de la Isla y que actúa como un *proxy* del objeto interno. De esta forma, un mensaje enviado a un objeto *FarRef* es redireccionado al objeto existente en el interior de la Isla, como se muestra en la figura 5.3.

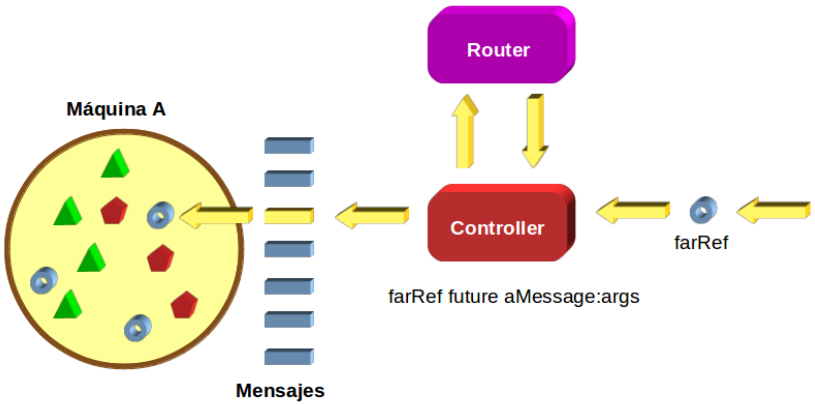

Figura 5.3: Objetos y Mensajes en OpenCobalt.

En *OpenCobalt*, los mensajes poseen cuatro componentes: El destinatario, que es el objeto al cual se le envía el mensaje, el mensaje propiamente dicho, los argumentos del mensaje, si los hubiere, y finalmente una referencia temporal *(time stamp)* que será utilizada posteriormente para ordenar secuencialmente la cola de mensajes a ser ejecutados.

En este sentido, el tiempo juega un rol esencial en el mecanismo que permite sincronizar las acciones en una isla replicada, ya que se debe garantizar que cada mensaje generado internamente sea ejecutado exactamente en el orden y el instante apropiados, y simultáneamente, los mensajes externos deberán también ser intercalados de forma apropiada junto con los mensajes generados internamente.

Router y referencias temporales

La referencia temporal *(time stamp)* que forma parte de los mensajes, es generada en base al tiempo interno de la Isla y es considerada como el instante actual *(now)*, más una demora *(offset)*, que es un número mayor a cero.

De esta manera, todos los mensajes que se encuentran en la cola son *"futuros mensajes"*. Es decir, son mensajes generados como resultado de la ejecución de algún mensaje interno, y como efecto secundario, envían un mensaje a otro objeto a un tiempo preestablecido en el futuro. Mientras que en el caso de los mensajes externos, el tiempo es fijado por un objeto especial, denominado *Router*.

El *Router* tiene dos roles principales. En primera instancia actúa como un reloj para las Islas replicadas, de forma tal de determinar cuando será ejecutado un evento. Estos eventos son la única información que una Isla posee acerca del paso del tiempo, de modo que la Isla simplemente no puede ejecutar ningún mensaje pendiente en su cola de mensajes hasta que reciba un mensaje externo con la respectiva referencia temporal *(time stamp)*. El segundo rol crítico desempeñado por el *Router* es reenviar todos los mensajes recibidos desde un *Controller* en particular, a todas las Islas registradas en ese momento, como puede apreciarse en la figura 5.4.

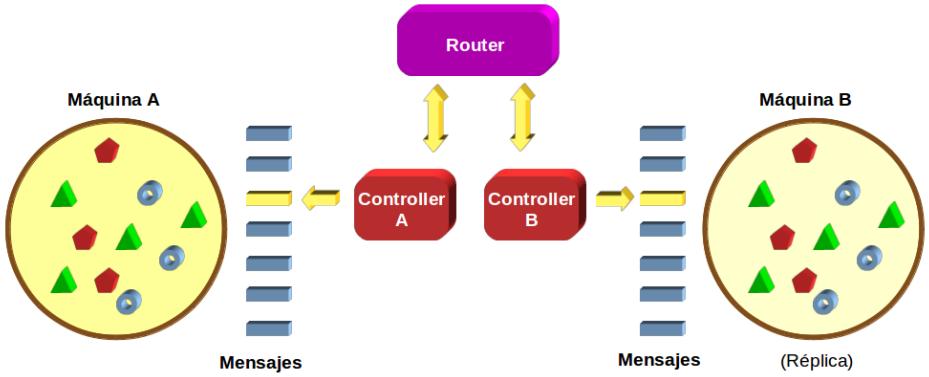

Figura 5.4: Funcionamiento interno de OpenCobalt.

Debido a que las Islas no pueden ejecutar ningún mensaje si no reciben mensajes externos, a veces es necesario generar estos mensajes, simplemente para que el tiempo transcurra.

Por lo tanto, el *Router* también es el encargado de crear estos mensajes denominados *heartbeats* (latidos), que son mensajes que solo contienen la referencia temporal *(time stamp)*, y cuya función es mantener a la Isla en funcionamiento.

El principal objetivo de este esquema, es lograr una interacción consistente, en tiempo real, entre grupos de usuarios conectados en red, que posibilite el desarrollo y distribución de aplicaciones colaborativas e interactivas. Por lo tanto, si una simulación está corriendo dentro de un Mundo Virtual, cada uno de los usuarios conectados a ese nodo, observarán que la *"réplica"* de ese Mundo Virtual, que se ejecuta localmente, presenta un comportamiento totalmente consistente con el de la Isla original.

5.4. Simulación Gravitacional

Existe una amplia variedad de fenómenos interesantes para simular en un Entorno Virtual 3D, y analizar su comportamiento en condiciones diversas, tanto a una escala micróscópica como astronómica, desde el análisis de las órbitas de cometas, planetas y asteroides, hasta la simulación de los lanzamientos de satélites y trayectorias de sondas espaciales. En este sentido, el estudio del movimiento de cuerpos sometidos a fuerzas de atracción gravitacional ofrece un interesante campo de estudio y experimentación.

El famoso problema, conocido como el *"Problema de los Tres Cuerpos"*, que analizaremos en la sección 5.4.1, es un excelente ejemplo de aplicación, ya que desde el punto de vista pedagógico, permite un abordaje múltiple, tanto desde la perspectiva del aprendizaje de los sistemas dinámicos, como desde las cuestiones relacionadas con su resolución numérica.

5.4.1. El Problema de los Tres Cuerpos

El *Problema de los Tres Cuerpos* (Wild, 1980), cuya solución determina la posición y velocidad en cada instante, de tres cuerpos que interactúan gravitacionalmente entre sí, es un ejemplo particularmente interesante para desarrollar una simulación en un ambiente virtual 3D, ya que reune las siguientes características:

- Se trata de un problema de escala astronómica, por lo que no es reproducible en forma práctica en el laboratorio.
- No posee una solución analítica general, por lo que es especialmente adecuado para resolverlo numéricamente.
- Su solución permite una representación natural en un espacio 3D.
- Presenta casos particulares muy interesantes para analizar.

Este problema, sencillo en apariencia pero que encierra una enorme complejidad, ha sido durante años, objeto de estudio de importantes físicos y matemáticos. Afortunadamente hoy en día, gracias a la potencia de cálculo de las computadoras actuales, es posible simular numéricamente el fenómeno y visualizar su comportamiento en un entorno virtual 3D.

5.4.2. Antecedentes Históricos del Problema

La *Ley de Gravitación Universal* de *Newton* enuncia que existe una fuerza gravitatoria de atracción entre dos cuerpos, que es directamente proporcional al producto de sus masas, e inversamente proporcional al cuadrado de la distancia que los separa, y la segunda *Ley de Newton* establece que la fuerza aplicada sobre un cuerpo produce una aceleración directamente proporcional a la masa del mismo.

Por lo tanto, de la aplicación de ambas leyes se concluye que es posible determinar la trayectoria de un objeto en órbita de otros, si se conoce su posición y velocidad en un instante inicial.

El planteo de este problema para el caso de dos cuerpos fue resuelto inicialmente por el propio *Isaac Newton* y por *Leonard Euler* para el caso general, quien lo publicó en su obra *Theoria Motuum Planetarum et Cometarum*, en el año 1744. Sin embargo, cuando *Jean D'Alembert* plantea el mismo problema para tres cuerpos, el mismo permanece sin solución durante bastante tiempo (Cuartero, 2012).

Un caso particular de este problema fue posteriormente resuelto por *Joseph Louis de Lagrange*, quien determinó además, que existían cinco posiciones que podían ser resueltas. Estas posiciones particulares, hoy se conocen como *puntos de Lagrange*, y por sus características especiales, son especialmente importantes en la actualidad para la ubicación de satélites espaciales.

En el año 1776, *Pierre Simon Laplace* presenta una solución general en su tratado de *Mecánica Celeste*, en el cual además de explicar la causa de las anomalías de Saturno y Júpiter, afirma que si se conociera la posición y la velocidad de todas las partículas del Universo en un instante, sería posible calcular todas las posiciones pasadas y futuras de las mismas. Lamentablemente la solución obtenida por *Laplace* no era exacta y por lo tanto el problema continuó sin solución.

En el año 1884, el Rey *Óscar II* de Suecia, organizó un concurso Internacional de Matemáticas, en el que uno de los problemas a resolver era precisamente el problema de los ***n cuerpos***, una generalización del caso de tres. *Henri Poincaré* quien participó de dicho concurso, llegó a la conclusión, en el año 1888 que era imposible hallar una solución estable, ya que dicho problema carece de una solución analítica.

Una de las conclusiones más importantes del trabajo de *Poincaré* es que si bien la solución del problema puede describirse como una serie de potencias, esta serie no es convergente. Por esta razón, cualquier error en la determinación de los datos iniciales hace que el valor resultante sea diferente, y además, que esta desviación continuará incrementándose con el tiempo.

Sin embargo, el hecho que este problema no posea una solución analítica no implica que resulte imposible determinar las trayectorias de los cuerpos, ya que éstas pueden calcularse por medio de aproximaciones numéricas.

De hecho, este problema ha abierto un abanico de planteos similares que conducen a la búsqueda de la solución de sistemas dinámicos muy sensibles a las variaciones de las condiciones iniciales, denominados *Sistemas Caóticos*.

5.4.3. Modelo Matemático

Partiendo de las leyes del movimiento de *Newton*, es posible plantear las ecuaciones diferenciales (5.1) que describen el movimiento de dichos cuerpos al ser afectados únicamente por las fuerzas gravitacionales que se ejercen mutuamente.

$$m_i \cdot \frac{d^2\bar{q}_i}{dt^2} = \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{G \cdot m_i \cdot m_j \cdot (\bar{q}_i - \bar{q}_j)}{\|q_i - q_j\|^3} \tag{5.1}$$

$$\frac{d^2\bar{q}_i}{dt^2} = \sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{G \cdot m_j \cdot (\bar{q}_i - \bar{q}_j)}{\|q_i - q_j\|^3} \tag{5.2}$$

La determinación de la velocidad y la posición en el espacio de cada uno de los cuerpos en cada instante implica la resolución de un conjunto de ecuaciones diferenciales ordinarias *(EDOs)* de orden superior.

Resolución de EDOs de Orden Superior

La metodología utilizada para resolver las ecuaciones diferenciales ordinarias de orden superior, utilizando métodos numéricos, consiste en transformar previamente a cada una de dichas ecuaciones en un sistema de ecuaciones diferenciales ordinarias de primer orden.

En el caso particular del problema planteado, la fuerza que se ejerce sobre cada cuerpo está representada por una ecuación de segundo orden. Por lo tanto, es necesario transformar cada ecuación en un sistema de dos ecuaciones de primer orden, y como el objetivo es calcular la posición y la velocidad de cada cuerpo, en cada instante y para cada una de las dimensiones, obtenemos finalmente para este problema un sistema de 18 ecuaciones diferenciales de primer orden. O sea, seis ecuaciones diferenciales para cada cuerpo.

$$\frac{dx_i}{dt} = vx_i \tag{5.3}$$

$$\frac{dvx_i}{dt} = \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{G \cdot m_j \cdot (x_i - x_j)}{(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2})^3} \tag{5.4}$$

$$\frac{dy_i}{dt} = vy_i \tag{5.5}$$

$$\frac{dvy_i}{dt} = \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{G \cdot m_j \cdot (y_i - y_j)}{(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2})^3} \tag{5.6}$$

$$\frac{dz_i}{dt} = vz_i \tag{5.7}$$

$$\frac{dvz_i}{dt} = \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{G \cdot m_j \cdot (z_i - z_j)}{(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2})^3} \tag{5.8}$$

Una vez que se tiene el sistema de ecuaciones diferenciales que representa el fenómeno, lo que sigue es obtener la solución temporal del mismo, es decir, calcular en cada instante la posición y la velocidad de cada cuerpo, en función de las fuerzas gravitacionales aplicadas. Para realizar esto se utilizan diversos métodos numéricos, ya que este problema, como la mayoría de los problemas reales, no posee una solución analítica general.

5.4.4. Solución Numérica de E.D.O. - P.V.I.

Una gran variedad de problemas reales de Ingeniería, Física, Química, Biología, etc. son modelados por medio de sistemas de Ecuaciones Diferenciales Ordinarias *(E.D.O)*, sin embargo, muchos de estos sistemas no poseen una solución analítica o la misma es muy difícil de calcular, o sólo es válida para ciertos casos particulares. En este tipo de problemas, es necesario recurrir a métodos numéricos para calcular su solución, la cual depende por cierto, de sus valores iniciales *(P.V.I.)*.

Entre los métodos más utilizados para la resolución de Ecuaciones Diferenciales Ordinarias se encuentran los denominados *unipaso*, como Euler y Runge-Kutta, que calculan la solución del próximo paso únicamente a partir de la información del paso anterior, y los *multipaso*, como Adams-Bashfort y Adams-Moulton que requieren de la información de varios pasos previos (Burden y Faires, 2010).

En este sentido, los métodos *unipaso* requieren menos operaciones para el cálculo de la solución, por lo que se utilizan preferentemente en aquellos casos en los que la velocidad de cálculo es una prioridad, mientras que los métodos *multipaso* que calculan una solución más exacta y refinada, se utilizan en aquellos casos en que la exactitud es más importante que la velocidad de cálculo.

En el caso particular de la solución de ciertos problemas específicamente relacionados con el movimiento, como las trayectorias orbitales, el movimiento de partículas, así como también, aquellos relativos a la dinámica molecular, existen algoritmos específicamente desarrollados para estos casos, como los métodos de Euler-Cromer (Cromer, 1981), Störmer-Verlet o Velocity-Verlet, que permiten resolver eficientemente esta clase de problemas en particular.

Debido a que en la simulación implementada en el presente trabajo se han utilizado diferentes métodos numéricos para resolver Sistemas de Ecuaciones Diferenciales Ordinarias *(EDO)*, se ha incluido en el **apéndice A**, un estudio más detallado de sus características, desde su formulación matemática hasta las diferencias que presentan, en cuanto a estabilidad, exactitud y eficiencia de cálculo.

5.5. Simulación en OpenCobalt

Cada uno de los entornos virtuales 3D analizados presenta un conjunto de características que lo distinguen. En una forma resumida podría decirse que *Second Life* posee una fuerte tendencia hacia la interacción social, con un objetivo claramente comercial y de entretenimiento, mientras que *Open Simulator*, al ser practicamente una versión de código abierto de *Second Life*, posibilita la libre instalación de servidores propios, evitando así el costo mensual de una membresía.

Por su parte, *OpenCobalt* no precisa de la instalación de ningún servidor centralizado, ya que cada máquina conectada funciona como cliente-servidor, en un esquema de comunicaciones P2P *(Peer to Peer)*, y como es distribuido bajo una licencia del MIT de código abierto, permite un completo acceso a su código interno, a través de su entorno de programación incorporado.

Esto último, sumado al hecho que existe muy poca información en la bibliografía acerca del desarrollo de aplicaciones en *OpenCobalt*, lo hace particularmente interesante para realizar esta experiencia, y documentar en el presente trabajo ciertas cuestiones técnicas relacionadas con la implementación y su desempeño.

5.5.1. Estructura básica de la Simulación

Como se mencionó en la sección 5.4, el problema elegido para implementar una simulación tiene el doble propósito de servir como ejemplo exploratorio tanto desde el punto de vista del modelo físico-matemático, como desde los aspectos relacionados con su solución numérica y los algoritmos asociados. Por esta razón, resulta imprescindible que ambos aspectos estén convenientemente separados.

Antes de comenzar a describir las clases que componen la simulación resulta importante aclarar que tanto los nombres de las clases como de los métodos asociados se han escrito en inglés por varias razones. En primer lugar, porque permite mantener una semántica clara, con descriptores no demasiado largos, que remiten específicamente a su propósito, en este sentido el inglés es mucho más conciso que el castellano. En segundo lugar, porque el inglés es considerado en informática como una "*lingua franca*", y por último, porque suele ser una práctica usual al programar en *Smalltalk*.

En la figura 5.5, se muestra un diagrama de clases de la simulación. En el mismo se observa, como la clase **OdeSolver** determina el comportamiento genérico de los métodos numéricos que resuelven las ecuaciones diferenciales, y por otra parte, la clase **OdeSystem** que es la que se encarga de representar a los sistemas de ecuaciones diferenciales ordinarias de primer orden.

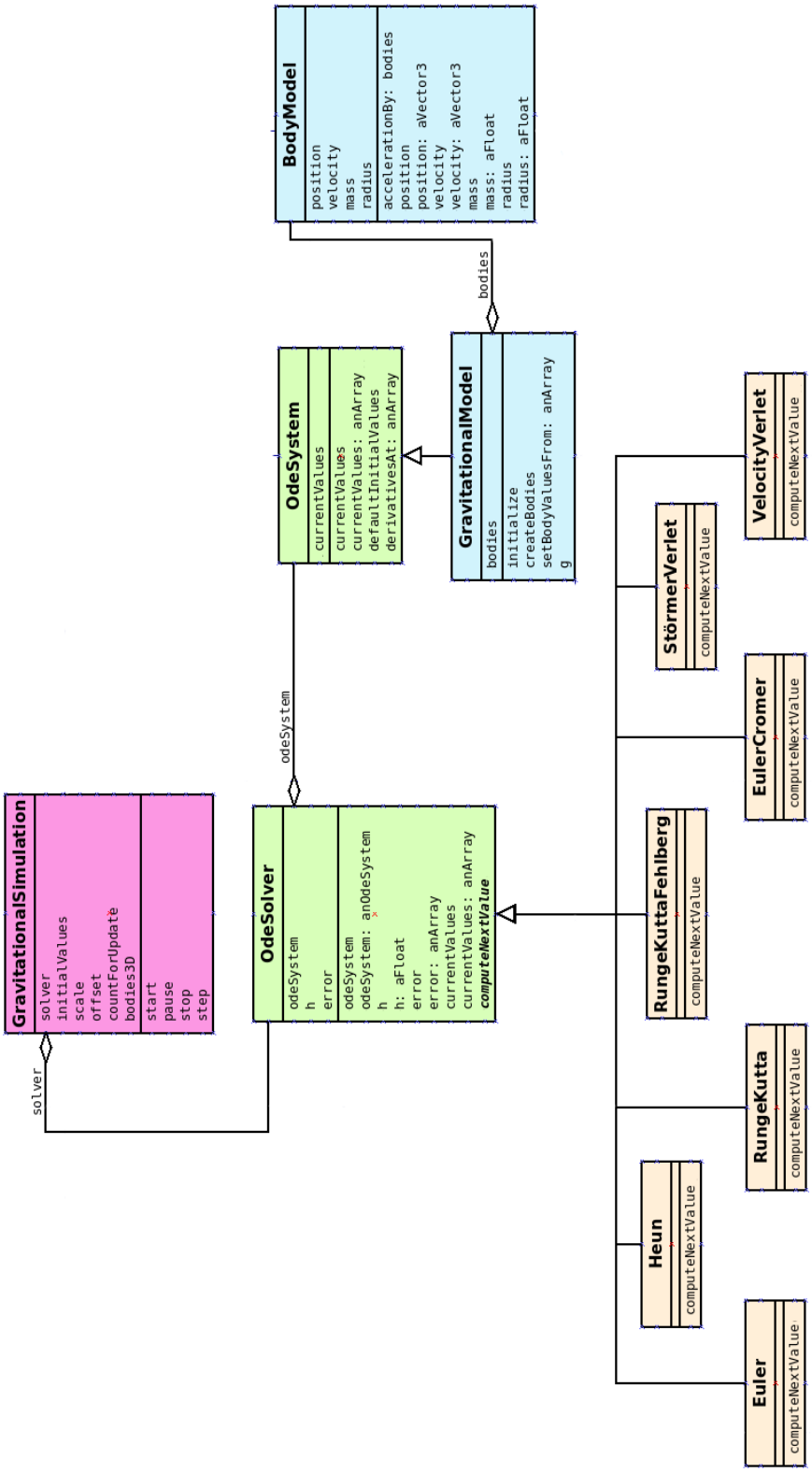

Figura 5.5: Diagrama de Clases de la simulación.

De esta forma, cualquier fenómeno que pueda describirse a partir de un sistema de ecuaciones diferenciales ordinarias podrá representarse por medio de una subclase de **OdeSystem**, y de forma similar, será posible incorporar nuevos algoritmos de resolución, simplemente creando nuevas subclases de **OdeSolver**.

Por lo tanto, mediante las clases derivadas de OdeSystem y de OdeSolver puede describirse el problema e implementar el algoritmo de resolución respectivamente. En este caso en particular, la clase **GravitationalModel** es la encargada de describir el sistema de ecuaciones diferenciales que representa el modelo matemático del fenómeno en estudio.

Esta estructura básica permite una rápida comprensión del funcionamiento de sus partes, lo cual posibilita realizar modificaciones con rapidez, así como también, extender fácilmente su funcionalidad.

Además, como puede observarse, esta estructura es completamente independiente de su representación gráfica, por lo que también puede utilizarse en forma separada para obtener resultados que permitan analizar el comportamiento de diferentes algoritmos de resolución, en cuanto a precisión, estabilidad, cantidad de operaciones involucradas, etc.

5.5.2. Programación en OpenCobalt

Como se mencionó en la sección 4.3.2, *OpenCobalt* está implementado íntegramente en un lenguaje de programación orientada a objetos, denominado *Squeak*, por lo que que todas las herramientas de programación y de depuración, propias de dicho lenguaje, se encuentran accesibles dentro del mismo entorno. De esta forma, es posible programar, y en muchos casos realizar cambios en el código, sin siquiera tener que interrumpir su ejecución, ya que los cambios son incorporados de forma inmediata en el sistema.

Un detalle interesante de programar en un lenguaje como *Smalltalk*, donde "*todo es un objeto*", es esa característica "*antropomórfica*" que se le suele otorgar a los objetos, la cual es muy útil desde el punto de vista didáctico para visualizar claramente las relaciones y mantener una visión conceptual del problema.

Por ejemplo, para resolver numéricamente un sistema de ecuaciones diferenciales ordinarias, es necesario calcular sus derivadas en cada instante. Por lo tanto, las subclases de *OdeSystem*, que representan el modelo matemático, deben implementar el método ***derivativesAt:*** para calcular dichas derivadas.

En el cuadro 5.1 se puede observar como se ha implementado este método en la clase **GravitationalModel**.

```
GravitationalModel >> derivativesAt: anArray
"Calcula las derivadas del sistema."
| d gc i |
self setBodyValuesFrom: anArray.
d := self derivativesArray: (anArray size).
gc := self g.
i := 1.
bodies do: [ :each |
      d at: i put: (each velocity).
      d at: (i+1) put: (gc*(each accelerationBy: bodies)).
      i := i+2].
^d
```

Cuadro 5.1: Cálculo de las derivadas

Como puede observarse, este método no evalúa directamente las fórmulas descriptas en la sección 5.4.3, sino que delega parte de la tarea en los objetos de la clase **BodyModel**, que representa a los cuerpos del sistema.

En el código que se muestra en el cuadro 5.2, se puede observar como cada cuerpo es capaz de calcular el valor de la aceleración, que el campo gravitatorio generado por los cuerpos que lo rodean, produce sobre él.

```
BodyModel >> accelerationBy: bodies
"Calcula la aceleracion provocada por los demas cuerpos."
| d |
^bodies inject: (Vector3 new) into: [:sum :each | (self = each)
    ifTrue: [ sum ]
    ifFalse: [ d := (each position - self position).
    sum + ( ( (each mass)*d ) / ( (d squaredLength) raisedTo: 1.5) ) ] ].
```

Cuadro 5.2: Cálculo de la aceleración

Si bien esta forma de implementar el cálculo produce cierto incremento de procesamiento, con referencia a la cantidad de mensajes enviados, es mucho mayor el beneficio que puede lograrse desde el punto de vista didáctico y conceptual, ya que en lugar de un conjunto de fórmulas abstractas, se pone de manifiesto como cada cuerpo es influenciado por la presencia de los demás.

5.5.3. Construcción del Mundo Virtual

Según el enfoque de la teoría *construccionista* del aprendizaje, desarrollada por Seymour Papert, los seres humanos construyen su conocimiento con particular eficacia cuando participan en la construcción de productos que les son personalmente significativos (Resnick, 1997). En este sentido, cada creación propia realizada por el estudiante resulta mucho más significativa para su aprendizaje que la utilización de un producto terminado mucho más sofisticado, pero que es percibido como una "*caja negra*".

Tomando como referencia este enfoque, en la presente sección se expondrán los pasos a seguir para la creación de un mundo virtual en *OpenCobalt*.

Un espacio virtual que será el encargado de transformar los resultados numéricos de la simulación en una representación tridimensional animada del fenómeno en estudio, pero fundamentalmente que operará como una herramienta de exploración, de experimentación y de aprendizaje.

En primer lugar, para crear un nuevo mundo virtual en *OpenCobalt* es preciso crear una subclase de **CobatlWorld**, en nuestro caso crearemos una clase a la que llamaremos *GravitationalWorld*.

En este punto es necesario destacar que debido a la escasa información que existe en la bibliografía, en referencia a la programación en *OpenCobalt*, la principal fuente de información para la implementación, ha sido el análisis del propio código fuente del entorno.

```
GravitationalWorld >> initialize
"Inicializa el Espacio Virtual"
| space |
space := TSpace new.
space registerGlobal: #mainEntry.
self makeLights.
self makeFloor.
self makeSkyBox.
self makeDestinationPortalWindow: space.
space objectName: self baseName.
space spaceDescription: 'Simulación Gravitacional'.
space spaceCreator: 'Francisco A. Lizarralde'.
space addChild: GravitationalSimulation new.
^space.
```

Cuadro 5.3: Inicialización del Mundo Virtual

En el código que se muestra en el cuadro 5.3 se puede observar la forma básica de inicialización de un espacio virtual. Esta inicializacion consiste en registrar el espacio *(Island)* para que también sea visible por otros usuarios,

para luego establecer ciertos parámetros estéticos, como extensión, luces y texturas, por medio de los métodos *makeLights, makeFloor, makeSkyBox*, etc, y por último la adición de los objetos que contendrá el mundo virtual, por medio del método *addChild*.

En este sentido, se ha creado la clase **GravitationalSimulation** para controlar el comportamiento de la simulación y poder interactuar con la misma para iniciar, pausar o detener su ejecución,

La clase **GravitationalSimulation** se creó como una subclase de **TGroup** que como se detalla en el **Apéndice B**, se suele utilizar principalmente como un contenedor de objetos gráficos **TFrame** o **TMesh**, y en este caso, contiene a las representaciones 3D de los cuerpos celestes.

En la figura 5.6 se observa una vista de la simulación corriendo en pantalla completa y a la izquierda se encuentra un cuadro de diálogo, mediante el cual es posible controlar su ejecución, así como también, modificar los valores iniciales de la simulación, el algoritmo utilizado, etc.

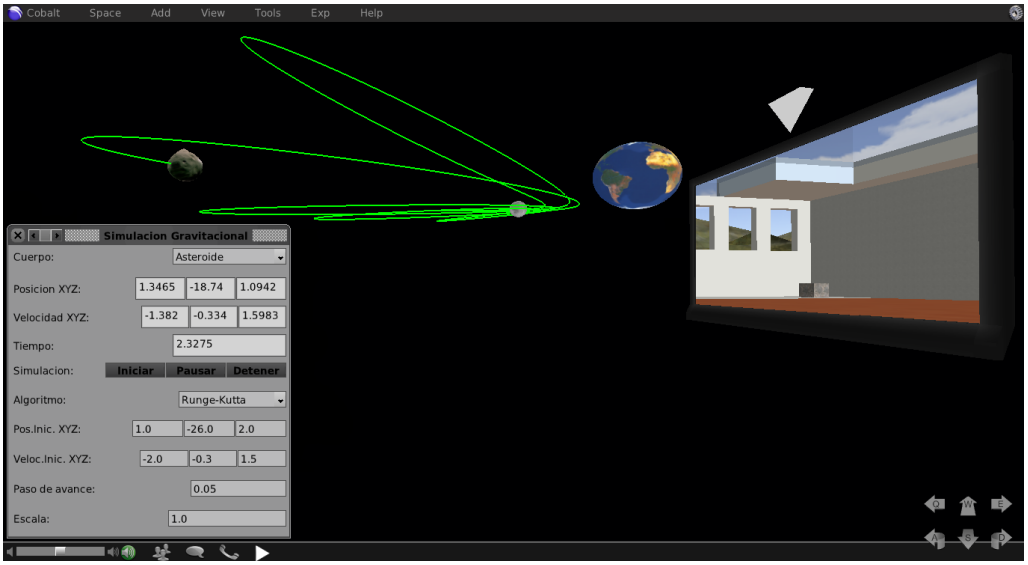

Figura 5.6: Espacio de la Simulación Gravitacional

En este sentido, dos de los cuerpos, que representan la Tierra y la Luna, son objetos de la clase **TSphere**, una subclase de **TMesh**, a los cuales se les ha ajustado su tamaño y se ha aplicado una imagen adecuada como textura, mientras que el tercer cuerpo, correspondiente al asteroide, es en realidad un modelo 3D importado en formato KMZ.

La importación de modelos 3D creados externamente, como en este caso, obtenido del repositorio de Google *3DWarehouse*, es una forma sencilla de incorporar modelos 3D complejos y de excelente calidad estética.

5.5.4. Sincronización y Funcionamiento en Red

Toda simulación dependiente del tiempo necesita una forma expresar su variación. Como se analizó en la sección 5.3.1, cada mundo virtual, denominado *Isla*, en la terminología de *OpenCobalt*, posee su propio punto de vista acerca del transcurso del tiempo. De modo tal, que los mensajes que van llegando, actúan de alguna forma como su reloj interno.

Esto implica que en una determinada *Isla*, el tiempo puede transcurrir a una velocidad diferente de otra *Isla*, con respecto al tiempo real. Sin embargo, debido a que el tiempo permanece atómico mientras se ejecuta cada mensaje, y los mensajes externos son los encargados de hacer avanzar la ejecución, cualquier tipo de latencia en el sistema no debería tener mayor impacto, ya que globalmente se percibe el mismo comportamiento tanto en la *Isla*, como en todas sus réplicas.

Para comprobar esto, se hicieron pruebas con computadoras de diferente potencia de procesamiento, conectadas por WiFi a través de una LAN *(Local Area Network)*, compartiendo el mismo Mundo Virtual, correspondiente a la simulación gravitacional.

Como puede apreciarse en la figura 5.7, la evolución de la simulación en cada equipo es totalmente consistente, y cada usuario observa una vista diferente de la misma, correspondiente a su posición relativa en el espacio.

Figura 5.7: Prueba de funcionamiento en Red

En el equipo de la izquierda, correspondiente al usuario representado por el Avatar verde, se observa como uno de los cuerpos (asteroide) se le aproxima de frente, mientras que en el equipo de la derecha, perteneciente al usuario representado por el Avatar rojo, éste observa como desde su propia perspectiva el asteroide se aleja, desplazándose hacia su izquierda.

La ejecución en cada Mundo Virtual *(Island)* está controlada tanto por mensajes internos como externos. En el caso de procesos que cambian de estado regularmente como una simulación, cada paso de avance se realiza enviando al objeto correspondiente, el mensaje **step**.

En el cuadro 5.4 se muestra el código correspondiente al método *step* que realiza las acciones correspondientes a un paso de simulación.

```
GravitationalSimulation >> step
"Ejecuta un paso de la simulacion"
self isRunning ifFalse: [^self].
self computeNextValue.
self updatePositions.
(self future: 30) step
```

Cuadro 5.4: Ejecución de un paso de la simulación

En la última línea se observa la expresión **(self future: 30) step**, que crea un tipo especial de mensaje, un "*mensaje futuro*", es decir, un mensaje a ser ejecutado en un futuro próximo.

Cuando se genera un "*mensaje futuro*" durante la ejecución del mensaje actual, su ejecución se define en términos de *"el instante actual"*, más un cierto *"desplazamiento temporal"*, que en el código anterior es de 30 milisegundos, aunque podría ser cualquier otro valor mayor a cero.

Según David Smith, uno de los desarrolladores de *OpenCobalt*, si se estableciera un *"desplazamiento temporal"* igual a cero, esto podría producir un ciclo infinito, que detendría la evolución del sistema. En cambio, al utilizar el mensaje ***future:*** *anInteger*, cada nodo *(Isla)* impone una *"recursión de cola temporal"* interna que impide que el sistema *"se congele"*. (D. A. Smith et al., 2006)

De esta forma, como se observó en la sección 5.3.1, los mensajes distribuidos por el *Router* a cada uno de los nodos *(réplicas)*, conservarán la misma secuencia de ejecución, lo cual determinará el mismo comportamiento en cada uno de ellos.

5.6. Consideraciones Finales

En el presente capítulo se desarrolló un análisis completo de una simulación numérica, desde la elección del problema y su formulación matemática, hasta la implementación de los aspectos visuales y de los métodos numéricos necesarios para su resolución.

El contexto que se ha establecido para la creación y experimentación con este tipo de simulaciones, es un ambiente universitario, de carreras de Ingeniería, con estudiantes que se encuentran promediando su ciclo de estudios. Por lo tanto, una de las premisas fundamentales establecidas para su implementación, dado su carácter didáctico, fue mantener un diseño simple, fácil de comprender, modificar y extender, sin necesidad de ser un experto en programación *Smalltalk*.

Es este sentido, la simulación no representa el objetivo sino el medio por el cual se intenta alcanzar el aprendizaje de los temas relacionados con los diferentes elementos que la componen.

Por esta razón, la simulación puede utilizarse de diversas formas. Es decir, puede ejecutarse simplemente para visualizar su comportamiento, o puede tomarse como punto de partida para desarrollar otra más compleja, o pueden realizarse modificaciones para contrastar alguna hipótesis, etc., puesto que el objetivo que se persigue es motivar al estudiante para que construya su propio "*laboratorio colaborativo de experimentación*".

Dennett considera que, es muy importante que los estudiantes puedan "*tomarse su tiempo*" para aprender, ya que desde la perspectiva construccionista que propone Papert, "*El aprendizaje ocurre mejor cuando se puede navegar alrededor del espacio del problema, saboreando las formas, jugando con las partes y piezas*". (Dennett, 1993)

En este caso en particular, el *Problema de los Tres Cuerpos*, es sólo una parte del "*espacio de problema*" que puede abordarse. Las piezas que conforman este espacio de problema pueden ser las leyes que rigen el fenómeno, o su modelo matemático, o los algoritmos numéricos para obtener su solución, o la exactitud de los resultados, o las nociones relacionadas con la geometría espacial involucradas en su representación 3D, etc.

Ahora bien, utilizar una simulación para el abordaje de aspectos tan diversos como los antes mencionados, se facilita enormemente si el lenguaje utilizado es lo suficientemente flexible y cuenta con las herramientas de programación adecuadas. Aquí es donde *OpenCobalt* posee una ventaja comparativa importante con respecto a los demás entornos analizados.

Como se ha mencionado en la sección 4.3.2, *OpenCobalt* se programa utilizando *Squeak*, un lenguaje de programación orientada objetos derivado de *Smalltalk*. A diferencia de *Second Life* y *OpenSimulator* que utilizan un modelo de programación por medio de *scripts*.

En el modelo de programación prototípica basado en *scripts*, utilizado por *Second Life* y *OpenSimulator* no existe el concepto de clase.

Por lo tanto, el comportamiento de cada objeto 3D está determinado únicamente por los *scripts* asociados al mismo. Y por esta razón, intentar desacoplarlos, es cuando menos, dificultoso sino imposible.

Finalmente, en la programación con LSL *(Linden Scripting Language)* los tipos de datos son bastante limitados, ya que por ejemplo, no existen estructuras tales como los diccionarios. Tampoco existe el concepto de reusabilidad, ya que el reuso se realiza copiando el código asociado con un objeto 3D y pegándoselo a otro. Además de que no existen mecanismos para realizar un control de versiones, en el caso que se requiera programar en forma colaborativa. (C. Lopes, 2014)

En cambio, el modelo de programación de *OpenCobalt* es totalmente diferente, ya que utiliza *Squeak*, el mismo lenguaje en que está implementado el propio entorno. En este sentido, tanto *Squeak* como *Smalltalk*, su antecesor, están basados en una poderosa metáfora que podría resumirse en dos frases, "*en Smalltalk todo es un objeto*" y "*los objetos sólo se comunican entre sí, por medio de mensajes*". (Ingalls, 1981)

El objetivo buscado cuando se diseñó *Smalltalk* era construir una herramienta para que los usuarios pudieran describir sus propios modelos del mundo real en forma de modelos de computadora, permitiéndoles experimentar con ellos y de esa forma lograr un aprendizaje más signicativo de conceptos físicos, matemáticos, biológicos, económicos, etc. Es decir, proveer al usuario (estudiante) las herramientas necesarias para modificar en forma rápida y sencilla, el modelo conceptual subyacente, algo muy diferente a la utilización de un programa cerrado.

En este sentido, en *Squeak* es posible modelar propiedades como masa, posición, velocidad, aceleración, etc., e implementar las relaciones existentes entre ellas en forma totalmente transparente, a pesar de que se trate de magnitudes escalares o vectoriales, pues la representación de cada una de ellas contiene internamente toda la información necesaria para realizar adecuadamente las operaciones.

Esto, además de simplificar enormemente los cálculos, permite que el estudiante no se distraiga con ciertos detalles de implementación, sino que se concentre en representar del modelo mediante la propia semántica del problema, y que de esta forma, no pierda de vista el objetivo buscado.

Ciertamente en *Squeak* se fusionaron los conceptos provenientes de varios lenguajes de programación como *Smalltalk*, *LOGO* y *Self* [1]. En él se plasmaron muchas de las ideas de Alan Kay, Dan Ingalls, Adele Goldberg y Seymour Papert, entre otros, cuya máxima aspiración no era simplemente crear un nuevo lenguaje de programación sino algo mucho más trascendente desde el punto de vista pedagógico, la creación de un medio de comunicación y de representación de ideas, mucho más cercano a los seres humanos que a las computadoras.

[1]Self es un lenguaje de programación orientada a objetos basado en prototipos, diseñado por David Ungar y Randall Smith en los laboratorios de Xerox PARC.

Conclusiones

En el presente trabajo se analizaron las características particulares que definen a los entornos virtuales 3D y se presentaron algunas de las principales teorías y modelos de aprendizaje, en busca del abordaje más adecuado para su aplicación en un contexto determinado.

Posteriormente se eligieron tres entornos virtuales 3D para su evaluación, siendo cada uno de ellos una representación particular de su sector. *Second Life* en el área comercial y de entretenimiento, *OpenSimulator* cuyo servidor ha sido desarrollado como software libre, y *OpenCobalt* cuyo modelo descentralizado de comunicaciones *(Peer to Peer)*, y el entorno en sí mismo, están íntegramente desarrollados en *Squeak*.

A partir de dicho análisis se desprenden las siguientes conclusiones, destacándose los resultados obtenidos en la implementación de la simulación numérica desarrollada en *OpenCobalt*, que se detalla en la sección 5.5.

(I) Los entornos virtuales de aprendizaje *(VLE)* tradicionales presentan una interfaz de usuario bidimensional, lo cuál limita en cierta medida la representación de ciertos contenidos. En este sentido, se ha comprobado que los entornos virtuales tridimensionales aportan una vivencia en primera persona, que paradójicamente, "*vuelve más real a lo virtual*", debido principalmente a su naturaleza inmersiva.

(II) A partir de los modelos y teorías de aprendizaje analizados para el presente trabajo, y en relación con el ámbito de aplicación establecido, se concluye que los entornos virtuales 3D son especialmente adecuados para la aplicación de un modelo construccionista, en referencia al desarrollo de proyectos didácticos significativos para los estudiantes.

(III) Con referencia a los resultados obtenidos al aplicar a los entornos 3D seleccionados el marco comparativo establecido en la sección 3.7, se desprende que tanto *Second Life* como *OpenSimulator* poseen un gran desarrollo de sus capacidades gráficas y de comunicación con otros usuarios, lo que los hace particularmente adecuados para el desarrollo de actividades grupales y colaborativas. En cambio, si bien *OpenCobalt* presenta ciertas limitaciones en cuanto a sus capacidades de edición de modelos 3D y comunicación entre participantes, aventaja notablemente a los demás entornos en lo que refiere al lenguaje y herramientas de programación, depuración y control de versiones.

(IV) La implementación de la simulación numérica que se presenta en la sección 5.5, permitió profundizar en aquellos aspectos de *OpenCobalt* escasamente tratados en la bibliografía. Este aporte se realizó a partir del análisis directo del código fuente del propio entorno, aprovechando las facilidades de las herramientas de programación del entorno.

(V) La utilización masiva de los entornos virtuales 3D probablemente requerirá cierto tiempo, junto con la confluencia de diversos factores sociales, tecnológicos y comerciales. Esta conclusión se deriva del hecho que gran parte del sustrato tecnológico en el que están basados fue desarrollándose incrementalmente en nichos tecnológicos, permaneciendo en estado latente, y siendo aún, desconocido para la inmensa mayoría.

Futuras líneas de Trabajo

En términos *histórico-tecnológicos* podría decirse que los entornos virtuales 3D recién están naciendo y esto nos lleva a reflexionar acerca de cómo aprovechar al máximo sus capacidades, ya que en la medida en que estas tecnologías se desarrollen, se asienten y se difundan, aparecerán sin duda, más y mejores formas de utilizarlas.

La experiencia lograda durante el desarrollo del presente trabajo permitió comprobar las excelentes capacidades de programación de *OpenCobalt*. Sin embargo, desde el fallecimiento del Dr. Andreas Raab, uno de sus principales arquitectos, el proyecto entró en un *impasse*, por lo cual aún continúa en la etapa alfa de su desarrollo.

Por este motivo, se ha considerado continuar el trabajo de investigación con la finalidad de profundizar en una de las siguientes áreas de interés.

La utilización de los entornos virtuales 3D vinculados con los sistemas de administración de contenidos de aprendizaje *(LCMS)*, como el proyecto *SLOODLE* (Simulation Linked Object Oriented Dynamic Learning Environment), que vincula a entornos 3D como *Second Life* u *OpenSimulator* con entornos virtuales de aprendizaje como *MOODLE*, se plantea como una interesante plataforma para realizar estudios de campo comparativos, con respecto a la utilización de LCMS tradicionales.

La otra área de interés es el estudio de nuevas interfaces *(HCI)* en entornos de realidad virtual, un interesante campo de investigación sobre todo en lo que respecta a los efectos de su utilización por personas con algún tipo de discapacidad motriz o cognitiva.

Ambos temas son realmente fascinantes y presentan grandes desafíos, pero a la vez me despiertan un gran interés y enormes expectativas.

Apéndices

Apéndice A

Métodos Numéricos para resolver E.D.O. - P.V.I.

A.1. Método de Euler

El método más sencillo para hallar la solución de una Ecuación Diferencial Ordinaria es el método de Euler, también denominado Euler Explícito, el cuál se deriva fácilmente a partir del desarrollo en Serie de Taylor de una función genérica x(t).

$$x(t_0 + \Delta t) = x(t_0) + x'(t_0) \cdot \Delta t + x''(t_0) \cdot \frac{\Delta t^2}{2} + \cdots \tag{A.1}$$

Truncando la serie en el segundo término, se puede generar un esquema iterativo como el siguiente:

$$x_{n+1} = x_n + x'(t_n, x_n) \cdot \Delta t \tag{A.2}$$

Este método posee un error global del orden de Δt, por lo que se lo denomina como un método de *primer orden*. Y como puede apreciarse, es un método muy simple, aunque presenta cierta inestabilidad, sobre todo cuando se lo utiliza para

resolver ecuaciones que poseen algún término que hace variar muy rápidamente su solución. A este tipo de ecuaciones se las suele denominar como *Ecuaciones Diferenciales Ordinarias Rígidas.*

A.2. Métodos de Runge-Kutta

Si bien el método de Euler es muy sencillo y rápido de calcular, el hecho de haber truncado la serie en el segundo término produce un error local del orden de Δt^2. Este error, que puede ser tolerable para pequeños valores de Δt, se va acumulando paso a paso lo que produce un error global mucho mayor, del orden de Δt. Por esta razón es que se han desarrollado diversas variantes con la finalidad de disminuir el error y lograr una mayor estabilidad del método y mejor exactitud en los resultados. A este conjunto de métodos se los suele denominar genéricamente como métodos de Runge-Kutta.

A.2.1. Método de Euler Implícito

El método de Euler Implícito, también denominado Euler hacia atrás *(Backward Euler)*, calcula la próxima posición a partir de la derivada de la función en el instante $t + \Delta t$, a diferencia del método anterior *(Euler Explícito)* que la calcula en el instante t.

$$x_{n+1} = x_n + x'(t_{n+1}, \widetilde{x}_{n+1}) \cdot \Delta t \tag{A.3}$$

Sin embargo, para calcular dicha derivada sería necesario conocer el valor de la posición x_{n+1}, y este valor es desconocido, ya que se trata precisamente de la posición que se pretende calcular, por lo tanto, se adopta un valor aproximado $\widetilde{x}_{n+1}$, el cual se calcula por medio de la expresión A.4.

$$\widetilde{x}_{n+1} = x_n + x'(t_n, x_n) \cdot \Delta t \tag{A.4}$$

Aún cuando este método también posee un error global del orden de Δt, esta simple modificación lo hace mucho más estable que el método de *Euler Explícito*.

A.2.2. Método de Heun

Otra variante es el método de Heun, también conocido como método de *Euler Modificado*. En este caso, la nueva posición se calcula en base al promedio de las derivadas en ambos extremos del intervalo.

$$x_{n+1} = x_n + \frac{x'(t_n, x_n) + x'(t_{n+1}, \widetilde{x}_{n+1})}{2} \cdot \Delta t \tag{A.5}$$

Al igual que en el caso anterior, el valor de la posición, necesario para calcular la derivada en el próximo instante, se aproxima también por medio de la expresión A.4. Pero en este caso, el error global se reduce en un orden de magnitud, por lo que este método es considerado como un método de Runge-Kutta de *segundo orden*.

A.2.3. Método de Runge-Kutta 4to. Orden

Esta forma del método de Runge-Kutta, es un método de *cuarto orden* lo cual significa que el error por paso es del orden de Δt^5, mientras que el error total acumulado tiene el orden Δt^4.

$$k_1 = \Delta t \cdot x'(t_n, x_n) \tag{A.6}$$

$$k_2 = \Delta t \cdot x'(t_n + \frac{\Delta t}{2}, x_n + \frac{k_1}{2}) \tag{A.7}$$

$$k_3 = \Delta t \cdot x'(t_n + \frac{\Delta t}{2}, x_n + \frac{k_2}{2}) \tag{A.8}$$

$$k_4 = \Delta t \cdot x'(t_n + \Delta t, x_n + k_3) \tag{A.9}$$

$$x_{n+1} = x_n + \frac{(k_1 + 2k_2 + 2_3 + k_4)}{6} \tag{A.10}$$

Este método aumenta considerablemente la exactitud de los resultados pero a costa de un mayor esfuerzo computacional, ya que es necesario evaluar cuatro veces la derivada y luego realizar un promedio ponderado en cada paso.

Por esta razón, muchas veces no es posible utilizar este tipo de métodos para simular *On Line* el comportamiento de una gran cantidad de objetos, ya que el tiempo de cálculo queda limitado por la cantidad mínima de cuadros por segundo (fps) necesarios para lograr una animación aceptable. Por esta razón, se han desarrollado métodos de cálculo específicos para resolver ecuaciones diferenciales de movimiento en forma rápida y precisa. A continuación, presentaremos algunos de ellos.

A.3. Métodos Específicos

La simulación de ciertos fenómenos como las trayectorias orbitales, el movimiento de partículas, así como también aquellos relativos a la dinámica molecular requieren de un tratamiento particular, por lo que se han desarrollado métodos específicamente adaptados para resolver las ecuaciones de movimiento. Esta clase de métodos es utilizada principalmente en los Motores de Física presentes en los Entornos Virtuales Tridimensionales, así como también en la mayoría de los videojuegos de última generación, en los que existe una relación de compromiso entre la exactitud, la estabilidad y la velocidad de cálculo requerida.

A.3.1. Método de Euler-Cromer

Cuando se intenta calcular el movimiento oscilatorio de una partícula utilizando el método de Euler Explícito, se observa que la energía del sistema se incrementa en cada iteración, algo que no tiene sentido físico. Sin embargo, una pequeña modificación, que deriva en el método de Euler-Cromer, brinda una solución correcta y estable que conserva la energía del sistema.

$$a_n = \frac{F(x_n, t_n)}{m} \tag{A.11}$$

$$v_{n+1} = x_n + a_n \cdot \Delta t \tag{A.12}$$

$$x_{n+1} = x_n + a_{n+1} \cdot \Delta t \tag{A.13}$$

Sin embargo, la razón por la cual esta simple modificación mejora notablemente la solución no es trivial. Un análisis detallado de las ecuaciones de movimiento oscilatorio permite observar que al utilizar el método de Euler-Cromer se produce un incremento de energía en un ciclo completo que es proporcional a Δt^3, mientras que para Euler Simple dicho incremento es proporcional a Δt, o sea mucho mayor.

A.3.2. Método de Euler-Richardson

El método de Euler-Richardson considera más conveniente obtener la nueva posición a partir del cálculo de la velocidad en la mitad del intervalo de tiempo, que obtenerla a partir de la velocidad al comienzo ó al final del intervalo. Este método resulta particularmente útil para la resolución de problemas en los que las fuerzas aplicadas dependen de la velocidad, como las fuerzas de rozamiento, aunque funciona igualmente bien en aquellos casos en los que no existe tal dependencia.

A partir de la fuerza aplicada, es posible calcular la aceleración a_n por medio de la ecuación A.11, y a partir de ese valor se obtienen los valores de la velocidad

$v_{n+\frac{1}{2}}$, la posición $x_{n+\frac{1}{2}}$ y la aceleración $a_{n+\frac{1}{2}}$, correspondientes al punto medio del intervalo.

$$v_{n+\frac{1}{2}} = v_n + a_n \cdot \frac{\Delta t}{2} \tag{A.14}$$

$$x_{n+\frac{1}{2}} = x_n + v_n \cdot \frac{\Delta t}{2} \tag{A.15}$$

$$a_{n+\frac{1}{2}} = \frac{F(x_{n+\frac{1}{2}}, t_{n+\frac{1}{2}})}{m} \tag{A.16}$$

Luego, a partir de estos valores es posible calcular los valores de velocidad v_n y posición x_n en el próximo instante.

$$v_{n+1} = x_n + a_{n+\frac{1}{2}} \cdot \Delta t \tag{A.17}$$

$$x_{n+1} = x_n + v_{n+\frac{1}{2}} \cdot \Delta t \tag{A.18}$$

A.3.3. Método de Störmer-Verlet

En el método de Euler Explícito, la discretización de la ecuación diferencial se calcula utilizando diferencias en avance, que tienen un error del orden de Δt, en cambio, en el método de Verlet, la aceleración se discretiza por medio de diferencias centradas, según se observa en la ecuación A.19, cuyo error es del orden de Δt^2.

$$a = \frac{d^2x}{dt^2} \approx \frac{\frac{x_{n+1}-x_n}{\Delta t} - \frac{x_n - x_{n-1}}{\Delta t}}{\Delta t} = \frac{x_{n+1} - 2x_n + x_{n-1}}{\Delta t^2} = a_n \tag{A.19}$$

Reordenando la última expresion, obtenemos la ecuación A.20, que nos permite calcular la posición del objeto en el instante siguiente, a partir del valor de la aceleración y de las posiciones anteriores.

$$x_{n+1} = 2x_n - x_{n-1} + a_n \cdot \Delta t^2 \tag{A.20}$$

Sin embargo, como se necesita conocer la posición en dos instantes previos para realizar el cálculo de la posición en el primer instante, se utiliza la ecuación A.21, que se deriva a partir del desarrollo en Serie de Taylor. En cuanto a los valores x_0 y v_0, los mismos son conocidos, ya que son las condiciones iniciales del problema.

$$x_1 = x_0 + v_0 \cdot \Delta t + a_0 \cdot \frac{\Delta t^2}{2} \tag{A.21}$$

Como puede apreciarse en la ecuación A.20, la nueva posición se calcula sin necesidad de calcular la velocidad en forma explícita. Sin embargo, a veces es necesario conocer explícitamente el valor de la velocidad para determinar los valores de ciertas magnitudes físicas, tales como la energía cinética de las partículas. En estos casos, se suele utilizar una variante de este método, denominada Velocity-Verlet.

A.3.4. Método Velocity-Verlet

Esta variante del método de Verlet utiliza un enfoque similar, solo que calcula la velocidad en forma explícita. En este caso, la posición x_{n+1}, se obtiene a partir de los tres primeros términos del desarrollo en Serie de Taylor.

$$x_{n+1} = x_n + v_n \cdot \Delta t + a_n \cdot \frac{\Delta t^2}{2} \tag{A.22}$$

Y luego se calcula el valor de la aceleración a_{n+1} a partir de la siguiente ecuación.

$$a_{n+1} = \frac{F(x_{n+1}, t_{n+1})}{m} \tag{A.23}$$

Y por último se calcula la velocidad v_{n+1} por medio de la siguiente ecuación.

$$v_{i+1} = v_i + \frac{(a_i + a_{i+1})}{2} \cdot \Delta t \tag{A.24}$$

De la ecuación A.23 se desprende que la aceleración no depende de la velocidad, sino que sólo depende de la posición en cada instante, como es el caso por ejemplo, de los fenómenos gravitatorios.

Apéndice B

Clases Principales en OpenCobalt

CobaltWorld

Es una subclase de *BaseWorld* y representa el comportamiento básico de los Mundos Virtuales. A partir de esta clase pueden derivarse Mundos Virtuales con características particulares, como por ejemplo, *GravitationalWorld*.

TSpace

En *OpenCobalt*, cada mundo virtual o espacio colaborativo es representado por la clase *TSpace* que es una subclase de *TFrame*. Además de actuar como contenedor de objetos gráficos tiene otras tareas adicionales tales como, mantener actualizada la información acerca de las luces aplicadas a la escena, y conocer cuáles de los objetos contenidos son opacos y cuáles transparentes para optimizar las tareas de redibujo de la escena.

TFrame

Todas las representaciones visuales de *OpenCobalt* están basadas en una estructura árbol jerárquico de *frames*, para poder realizar las operaciones gráficas *rendering* de la escena. Por lo tanto, todos los objetos gráficos que forman parte

de una escena son instancias de la clase *TFrame* o de alguna de sus subclases.

Su función principal es actuar como un contenedor y un marco para aplicar las transformaciones gráficas necesarias. De esta forma, cuando se necesita actualizar la visualización de la escena, el *"motor de visualización" (rendering engine)*, realiza todas las transformaciones gráficas necesarias sobre cada uno de los objetos contenidos en la jerarquía.

TGroup

Es una clase que hereda de TFrame y que posee la particularidad de ser invisible por defecto, por lo que su principal uso es como contenedor de otras representaciones visuales (frames).

Por ejemplo, en el presente trabajo, la clase **GravitationalSimulation** no posee una representación visual propia, sino que actúa como contenedor de las representaciones visuales de los cuerpos, además de controlar su funcionamiento.

TMesh

Todos los objetos gráficos están constituidos por un conjunto de polígonos, por lo general, triángulos o cuadrángulos. Desde formas tan simples como un cubo hasta otras mucho más complejas y animadas como los *Avatars*, todas constituyen algún tipo de malla de polígonos y por lo tanto son subclases de la clase *TMesh*. Esta clase es la encargada de contener la información necesaria para los procesos gráficos de visualización. Por lo tanto contiene la lista de los vértices de los polígonos y sus caras, así como también cuáles son los materiales y texturas asociados al objeto.

Estas mallas, además de representar objetos primitivos como cubos, esferas, conos y cilindros, también pueden ser creadas a partir de la información contenida en archivos externos, en diversos formatos como (.dae) *Digital Asset Exchange*, creados previamente con programas de modelado 3D como *Blender*, *SketchUp*, *Maya*, *3D Studio Max*, etc.

TMaterial

Los materiales se utilizan para contener la información correspondiente a las propiedades lumínicas de la superficie de los objetos. De esta forma, al modificar el material se pueden representar objetos con apariencia metálica, plástica, opaca, brillante, etc. Esta apariencia se puede complementar con la aplicación de texturas.

TTexture

Las texturas, representadas por la clase *TTexture*, son imágenes o mapas de bits *bitmaps* que pueden utilizarse de dos maneras. La más frecuente es para agregar una textura al material de un objeto, o directamente sobre la superficie del objeto. Aunque también puede utilizarse para agregar imágenes o dibujos en 2D, directamente a la escena. Estas texturas pueden generarse en forma programática o importarse desde un archivo de imágenes en formatos .bmp *(Bitmap)*, .gif *(Graphics Interchange Format)*, .png *(Portable Network Graphics)* y .jpeg *(Joint Photographic Experts Group)*.

Anexos

Anexo C

Relevamiento de Mundos Virtuales 3D

A continuación se presenta un relevamiento de los principales mundos virtuales realizado a partir de la compilación presentada por quien se da a conocer en el *Metaverse*, como *Ariane Barnes* (Barnes, 2012). Se ha complementado la misma con la adición otros entornos virtuales 3D considerados relevantes para el presente trabajo.

Nombre	**3D Chat**
Año	2010
Diseño	Glenn Garritano
Propiedades	Permite la personalización de sus avatars, modificando su cara, cuerpo, facciones, color de piel, etc. Su objetivo principal es proveer servicios de chat en un espacio virtual aunque también posee una versión web llamada My3DChat. Cada mes los usuarios reciben un salario (en 3D Dollars) del cual se deducen los gastos por bienes y servicios virtuales, por lo que no es necesario ningún pago en moneda real para utilizar los servicios básicos. Sitio: http:///www.3dchat.com/

Nombre	**3rd Planet**
Año	2011
Diseño	Terence Mak
Propiedades	Es un sitio turístico que ofrece interacción 3D para visitar lugares turísticos reales, como el Monte Everest, desarrollado en conjunto con la Agencia Nepalés de turismo. Sitio: http://www.3rdplanet.com/
Nombre	**3rd Rock**
Año	2008
Diseño	Terry Ford
Propiedades	Es uno de los primeros mundos virtuales implementados en Opensimulator que posee su propia economía basada en su moneda local, el $Go (Geode). Si bien el acceso es gratuito, la adquisición de parcelas de terreno virtual para construir espacios tiene un costo mensual que depende de la cantidad máxima de prims (formas geométricas elementales) que pueden albergar. Sitio: http://3rdrockgrid.com
Nombre	**ActiveWorlds**
Año	1997
Diseño	ActiveWorlds Inc.
Propiedades	Creación de espacios virtuales. Personalización sencilla de Avatars. Gratuito. Sitio: http://www.activeworlds.com/
Nombre	**Blue Mars**
Año	2009
Diseño	Kaz Hashimoto
Propiedades	Excelentes gráficas. Facilidades para la creación de espacios virtuales para juegos negocios y personalización de avatars. Posee una moneda electrónica para realizar transacciones. Sitio: http://www.bluemars.com/
Nombre	**Club Cooee**
Año	2009
Diseño	Alexander Jorias e Ingo Frick
Propiedades	Solo permite la personalización de avatars, no la creación de contenidos. Pensado como espacio virtual de Chat para adolescentes. Sitio: http://www.clubcooee.com/
Nombre	**FriendsHangout**
Año	2008-2015
Diseño	Jeff White, Miguel Benitez
Propiedades	Es un sistema de Chat 3D o mundo virtual de encuentros sociales y negocios. Permite la creación de espacios virtuales y personalización de Avatars. Posee una moneda electrónica para realizar transacciones. Este proyecto finalizó el 30 de enero de 2015, dando paso a un nuevo proyecto denominado VMA - Virtual Media Architect. Sitio: http://friendshangout.com

Nombre	**Gojiyo**
Año	2010
Diseño	The Godrej Group (India)
Propiedades	Solo permite la personalización de avatars, no la creación de contenidos. Su objetivo principal es la participación social y los juegos. Principalmente orientado al mercado de la India. Sitio: http://gojiyo.com/
Nombre	**IMVU**
Año	2004
Diseño	Will Harvey y Eric Ries
Propiedades	Es un sistema de mensajería instantánea y Chat 3D para encuentros sociales, negocios y juegos. No posee areas de libre movimiento y exploración. Actualmente es uno de los entornos que poseen el mayor catálogo (más de 6 millones) de objetos virtuales a disposición de sus usuarios.Sitio: http://www.imvu.com/
Nombre	**Kaneva**
Año	2007
Diseño	Christopher Klause
Propiedades	Posee una personalización limitada de los Avatars. Los muebles y la ropa pueden adquirirse en tiendas virtuales. La creación de contenidos por parte de los usuarios está limitada a colocar sólo imágenes sobre las paredes. Sitio: http://www.kaneva.com/
Nombre	**NuVO**
Año	2007
Diseño	Tim Pelham y Wyllo Rogers
Propiedades	NuVera OnLine (NuVO) posee una personalización limitada de los Avatars, aunque permite una completa personalización de los ambientes, creación de texturas y objetos 3D. Está dirigido principalmente como sitio de encuentro para adultos (de 18 años en adelante). Sitio: http://www.nuveraonline.com
Nombre	**Onverse**
Año	2007
Diseño	Stephen M. Pierce y Wesley Macdonald
Propiedades	Es un sitio de encuentro social y Chat 3D. Posee una personalización limitada de los Avatars, aunque cuenta con una gran cantidad de objetos 3D y accesorios algunos gratuitos y otros no. Los usuarios pueden trasladarse entre los espacios virtuales utilizando diferentes medios de transporte como automóviles, barcos pirata, cañones de avatars, etc. Sitio: http://onverse.com/

Nombre	**OpenCobalt**
Año	2007-2009
Diseño	Alan Kay, Julian Lombardi, Mark McCahill, Andreas Raab, Dave P. Reed, David A. Smith
Propiedades	Es el sucesor de OpenCroquet, por lo tanto al igual que su antecesor posee un SDK (Software Development Kit) que permite la creación de contenidos, programación de simulaciones, creación de objetos virtuales, comunicación entre usuarios, etc. En ambos casos utiliza un esquema de comunicaciones P2P (Peer to Peer). Sitio: http://opencroquet.org/
Nombre	**OpenCroquet**
Año	2009
Diseño	Alan Kay, Mark McCahill, Andreas Raab, Dave P. Reed, David A. Smith
Propiedades	Es un entorno virtual 3D desarrollado en Squeak. Soporta comunicación, colaboración y compartición de recursos, aunque carece de herramientas propias para la creación de objetos o personalización de avatars, los cuales pueden ser creados por otros programas de contenidos 3D y luego importados en varios formatos. Su entorno de programación es totalmente accesible desde el mismo entorno. Sitio: http://opencroquet.org/
Nombre	**OpenSimulator**
Año	2007
Diseño	Darren Guard
Propiedades	Es una plataforma de código abierto para la creación de mundos virtuales compatible con el programa cliente de Second Life. Está escrito en *C#* y ha sido diseñado para ser fácilmente expandido por medio de la adición de módulos (plugins). Puede operar en diferentes modos: Standalone, Grid o Hypergrid. El modo Standalone es el más sencillo de configurar, el modo Grid permite escalar dentro de una LAN a medida que la cantidad de usuarios crece y el modo Hypergrid,permite conectar varios grids a través de Internet. Sitio: http://opensimulator.org
Nombre	**OpenQwaq**
Año	2011
Diseño	Alan Kay, David A. Smith, Andreas Raab, Dave P. Reed
Propiedades	QpenQwaq es un entorno virtual basado en la tecnología desarrollada para Teleplace (ver más adelante), un entorno virtual, derivado a su vez de OpenCroquet. En su primera versión poseía la misma funcionalidad que Teleplace, con excepción del subsistema de videoconferencia y visualización de videos, por cuestiones de licencia. Posteriormente se superó esta limitación al integrarse en OpenQwaq codecs de audio y video de código abierto. Sitio: https://code.google.com/p/openqwaq/

Nombre	**OpenWonderland**
Año	2010
Diseño	Sun Microsystems
Propiedades	Es un proyecto de código abierto multi-plataforma desarrollado en Java. Posee aplicaciones 2D compartidas, audio inmersivo en integración de aplicaciones de comunicación entre usuarios, chat, telefonía, video, etc. Posee herramientas de personalización limitadas, aunque los objetos 3D pueden crearse con otros programas como Blender e importarse en formatos standard. Sitio: http://openwonderland.org/
Nombre	**OSGrid**
Año	2007
Diseño	Gareth Nelson
Propiedades	OSGrid es uno de los mayores conjuntos de espacios virtuales basados en OpenSimulator. Por lo tanto, posee las mismas características acerca de la personalización de avatars y creación de contenidos. Los usuarios pueden crear o importar texturas, realizar animaciones y programar scripts de una forma muy similar a como lo harían en Second Life. Sitio: http://osgrid.org/
Nombre	**Qwaq**
Año	2011
Diseño	Alan Kay, David A. Smith, Andreas Raab, Dave P. Reed, Howard Stearns, Julian Lombardi
Propiedades	Qwaq fue el nombre de transición del proyecto sucesor de Croquet, antes de que el mismo se dividiera en dos proyectos. Uno comercial, denominado Teleplace (Teleplace Inc.) y otro de código abierto denominado OpenQwak. Sitio: https://code.google.com/p/openqwaq/
Nombre	**Second Life**
Año	2003
Diseño	Philip Rosedale
Propiedades	Es uno de los entornos virtuales con mayor tiempo en actividad, que posee actualmente alrededor de 20 millones de cuentas y más de 700 mil visitantes mensuales. Su finalidad es principalmente social, de entretenimientos y de negocios. Posee su propia economía basada en su propia moneda virtual el L$ (Linden Dollar). Permite la personalización de los Avatars y la creación de contenidos. Si bien el acceso al mismo es gratuito, la adquisición de parcelas para construir espacios virtuales propios tiene un costo mensual. Sitio: http://secondlife.com/

Nombre	**Sloodle**
Año	2006
Diseño	Daniel Livingstone y Jeremy Kemp
Propiedades	Es un proyecto de código abierto que integra los entornos multiusuario de Second Life u OpenSimulator con el sistema de administración de aprendizaje Moodle. El mismo integra las herramientas propias de los mundos virtuales inmersivos con un sistema de administración de aprendizaje ampliamente utilizado en todo el mundo. Sitio: https://www.sloodle.org/
Nombre	**Teleplace**
Año	2009-2011
Diseño	Alan Kay, David A. Smith, Andreas Raab, Dave P. Reed, Howard Stearns, Julian Lombardi
Propiedades	Es un entorno virtual comercial basado en Croquet con la finalidad de proveer de espacios virtuales colaborativos a diversas instituciones como compañías, universidades, organizaciones y agencias del gobierno de los EEUU, así como también de espacios virtuales colaborativos y de entrenamiento para la Aviación, la Marina y el Ejército. En 2011, toda la propiedad intelectual de sus desarrollos fue adquirida por 3D ICC para desarrollar su producto Terf. Sitio: (Ver Terf)
Nombre	**Terf**
Año	2011
Diseño	Julie LeMoine, Ron Teitelbaum, David A. Smith, Andreas Raab
Propiedades	Es producto comercial, sucesor de Teleplace y Openqwaq, que presenta un entorno inmersivo con comunicación de audio y video instantáneo, servicios de Chat y que permite compartir aplicaciones, así como también los elementos del escritorio para realizar tareas en forma colaborativa. Sitio: http://3dicc.com/
Nombre	**There**
Año	2003
Diseño	y Will Harvey y Jeffrey Ventrella
Propiedades	Activo desde 2003 a 2010, reabrió nuevamente en 2012. Con un objetivo predominantemente social, permite la personalización de avatars, creación de contenidos, exturas, ropa, objetos, amoblamiento, vehículos, etc. Posee un arancel de 10 USD mensuales. Las transacciones monetarias en There se realizan a través de su propia moneda virtual, los Therebucks (T$). Los Therebucks pueden adquirirse directamente en There a razón de 1,800 Tbux por cada 1 USD. Sitio: http://www.prod.there.com/

Nombre	**Twinity**
Año	2008
Diseño	Metaversum GmbH
Propiedades	Tiene uno de los más sofisticados sistemas de manejo de texuras. Es posible personalizar los avatars, incluso aplicando texturas fotográficas al rostro. Permite la creación de contenidos, incluyendo la aplicación de páginas web activas en las paredes. También posee su propia moneda (Globals), que puede adquirirse con tarjeta de crédito. Sitio: http://www.twinity.com/en/
Nombre	**vSide**
Año	2006
Diseño	Doppelganger, Inc
Propiedades	Orientado principalmente hacia la interacción social, por lo que se invita a sus residentes a compartir sus perfiles personalizados y sus intereses personales a través de foros y blogs. La personalización de los avatars es limitada, aunque pueden adquirirse ciertos ítems, ropa, etc. Sitio: http://www.vside.com/
Nombre	**Wet FM**
Año	2010
Diseño	Sine Wave Company
Propiedades	Es un entorno virtual con el principal objetivo de crear una comunidad para compartir experiencias musicales, asistir a eventos musicales virtuales en vivo, escuchar música, etc. A pesar de tener una personalización limitada de los avatars, su fuerte consiste en integrar la tecnología de los mundos virtuales 3D junto con los sistemas de streaming de música para acercar a sus residentes con intérpretes y grupos musicales. Sitio: http://www.wet.fm/

Anexo D

Glosario y Abreviaturas

3DVLE - (3D Virtual Learning Environments) Entornos Virtuales Tridimensionales de Aprendizaje.

AR - (Augmented Reality) Realidad Aumentada.

AVATAR - Representación del usuario en el Mundo Virtual.

B-LEARNING - (Blended Learning) Aprendizaje Semi-presencial.

CAI - (Computer Assisted Instruction or Computer Aided Instruction) Instrucción asistida por Computadora.

CBI - (Computer Based Instruction) Instrucción basado en Computadora.

CBT - (Computer Based Training) Entrenamiento basado en Computadora.

CGI - (Common Gateway Interface) Interfaz de Entrada Común.

CLE - (Collaborative Learning Environment) Entorno de Aprendizaje Colaborativo.

CMS - (Content Management System) Sistema de administración de contenidos.

CLIENT - (Cliente) Programa mediante el cuál el usuario navega un espacio tridimensional y se comunica con otros usuarios conectados al mismo servidor.

CROQUET - Entorno Virtual desarrollado en Squeak.

DEC - (Digital Educational Collaboration) Colaboración Educativa Digital.

E-LEARNING - (Electronic Learning) Aprendizaje electrónicamente mediado.

Go$ - (Geode) Moneda virtual de intercambio en 3rd. Rock.

IBT - (Internet Based Training) Entrenamiento basado en Internet.

ICT - (Information and Communication Technologies) Tecnologías de la Información y la Comunicación.

IRC - (Internet Relay Chat) Protocolo de comunicación en tiempo real basado en texto.

L$ - (Linden Dollar) Moneda virtual de intercambio en Second Life.

LCMS - (Learning Content Management System) Sistema de administración de contenidos para el aprendizaje.

LMS - (Learning Management System) Sistema de administración de aprendizaje. (Se lo considera un sinónimo de VLE)

LP - (Learning Platform) Plataforma de Aprendizaje.

ML - (Multimedia Learning) Aprendizaje Multimedio.

METAVERSE - Palabra para denominar al Mundo Virtual.

MIT - Massachusetts Institute of Technology.

MMOFPS - (Massively Multiplayer First Person Shooter) Juegos masivamente multi-jugador de tirador en primera persona.

MMOLE - (Massively Multilearner Online Learning Environments) Entornos de aprendizaje masivamente multi-estudiante en línea.

MMORLG - (Massively Multiplayer Online Real-Life Games) Juegos masivamente multi-jugador basados en la vida real.

MMORPG - (Massively Multiplayer Online Role Playing Games) Juegos de rol masivamente multi-jugador en línea.

MMOW - (Massively Multiplayer Online World) Mundo masivamente multi-jugador en línea.

MOODLE - (Modular Object-Oriented Dynamic Learning Environment) Entorno Modular de Aprendizaje Dinámico Orientado a Objetos.

MUVE - (Multi-User Virtual Environment) Entorno Virtual Multi-Usuario.

NTIC - Nuevas Tecnologías de la Información y de la Comunicación.

OLE - (On Line Education) Educación en Línea.

OPEN COBALT - Entorno Virtual desarrollado en Squeak. Es el sucesor de CROQUET.

P2P - (Peer to Peer) Protocolo de comunicación de Redes de Pares.

RA - Realidad Aumentada.

RV - Realidad Virtual.

SCORM - (Sharable Content Object Reference Model) Modelo de Referencia de Objetos de Contenido Compartible.

SERVER - (Servidor) Programa que comunica a los usuarios (clientes) aceptando sus peticiones y enviando las respuestas correspondientes.

SIE - (Synthetic Inmersive Environment) Entorno Inmersivo Sintético (Artificial).

SMALLTALK - Lenguaje Orientado a Objetos puro desarrollado en los laboratorios de XEROX Parc.

SQUEAK - Versión moderna de Smalltalk

TEL - (Technology Enhanced Learning) Aprendizaje enriquecido por la Tecnología.

TERRAFORMING - Modificar la topología del terreno. Es un término de uso común en la creación de espacios virtuales.

UDP - (User Datagram Protocol) Protocolo de envío de Datagramas a través de la Red.

VE - (Virtual Education) Educación Virtual.

VLE - (Virtual Learning Environments) Entorno Virtual de Aprendizaje.

VoIP - (Voice over Internet Protocol) Transmisión de voz sobre el protocolo de Internet.

VR - (Virtual Reality) Realidad Virtual.

WBT - (Web Based Training) Entrenamiento basado en la Web.

XMPP - (Extensible Messaging and Presence Protocol) Protocolo extensible de Mensajería y Comunicación de Presencia.

Referencias

Adhikari, K. (2012). Ausubel's learning Theory: Implications on Mathematics Teaching. *Parnayan*, *3*, 20–25.

Amaya Franky, G. (2009). Pedagogical potentialities of the simulation, surroundings from the perspective of the located cognition . *Revista TEA*, *13*(1), 97–105.

Ausubel, D. P. (1968). *Educational psychology : A Cognitive View* [Book]. Holt, Rinehart and Winston New York.

Barnes, A. (2012). *Ariane's Life in Metaverse.* http://arianeb.com/. (Accedido: 21 de Noviembre (2014))

Black, A., Ducasse, S., Nierstrasz, O., y Pollet, D. (2007). *Squeak by Example.* Square Bracket Associates. Descargado de `http://squeakbyexample.org/index.html` (with Damien Cassou and Marcus Denker)

Blas, N., Garzotto, F., y Poggi, C. (2009, diciembre). Web Engineering at the Frontier of the Web 2.0: Design Patterns for Online 3D Shared Spaces. *World Wide Web*, *12*(4), 345–379. Descargado de `http://dx.doi.org/10.1007/s11280-009-0065-5` doi: 10.1007/s11280-009-0065-5

Bradford, P., Porciello, M., Balkon, N., y Backus, D. (2007). The Blackboard Learning System . *The Journal of Educational Technology Systems*, *35*, 301–314.

Brashears, A., Meadows, A., Onderjka, C., y Soo, D. (2003). *Linden Scripting*

Language Guide (Inf. Téc.).

Burden, R. L., y Faires, J. D. (2010). *Numerical Analysis* (9nth. ed.). Brooks/Cole Cengage Learning.

Cakir, M. (2008). Constructivist Approaches to Learning in Science and Their Implications for Science Pedagogy: A Literature Review . *International Journal of Environmental & Science Education*, *3*(4), 193–206.

Connolly, M. (2011, October). Benefits and Drawbacks of Social Media in Education. En *Contested Issues in Student Affairs: Diverse Perspectives and Respectful Dialogue* (pp. 135–140). Wisconsin Center for Education Research.

Cromer, A. (1981). Stable solutions using the Euler Approximation. *American Journal of Physics*, *49*(5), 455–459.

Cross, J., O'Driscoll, T., y Trondsen, E. (2007, marzo). Another Life: Virtual Worlds As Tools for Learning. *eLearn*, *2007*(3), 2–. Descargado de `http://doi.acm.org/10.1145/1235511.1235515` doi: 10.1145/1235511.1235515

Cuartero, F. (2012). *El Problema de los Tres Cuerpos.* http://www.hablandodeciencia.com/articulos/2012/03/16/el-problema-de-los-tres-cuerpos/. (Accedido: 16 de Diciembre (2014))

Dabbagh, N. (2005). Pedagogical models for E-Learning: A theory-based design framework. *International Journal of Technology in Teaching and Learning*, *1*(1), 25–44.

Daghestani, L., Ward, R. D., Xu, Z., y Al-Nuaim, H. (2008). The Design, Development and Evaluation of Virtual Reality Learning Environment for Numeracy Concepts Using 3D Virtual Manipulatives . *Fifth International Conference on Computer Graphics, Imaging and Visualization.*

De Freitas, S. (2008, noviembre). Serious Virtual Worlds. A scoping study . *JISC Virtual Worlds & Serious Gaming Resources*, 4–49.

Denker, M. (2005). Squeak and Croquet. En *LinuxTag 2005.* Karlsruhe, Germany. Descargado de `https://hal.inria.fr/inria-00555715`

Dennett, D. C. (1993, November). Review of Papert, The Children's Machine. *New Scientist*, *140*(1898), 45–46. Descargado de `http://cogprints.org/434`

Dionisio, J., Burns, W., y Gilbert, R. (2013, julio). 3D Virtual Worlds and the Metaverse: Current Status and Future Possibilities. *ACM Comput. Surv.*, *45*(3), 34:1–34:38. Descargado de `http://doi.acm.org/10.1145/2480741.2480751` doi: 10.1145/2480741.2480751

Dos Santos, R. (2009, April). Second Life physics: Virtual, Real or Surreal ? . *Journal of Virtual Worlds Research*, *2*(1).

Dos Santos, R. (2012). Second Life as a Platform for Physics Simulations and Microworlds: An Evaluation. En (pp. 173–180). Centre for Research in Science and Mathematics.

Ertmer, P. A., y Newby, T. J. (1993). Behaviorism, cognitivism, constructivism: Comparing critical features from an instructional design perspective. *Performance Improvement Quaterly*, *6*(4), 50–72.

Escobar Gutiérrez , M. L. (2015). *Posibilidades Educativas del Entorno 3D Second Life para Docentes* [Tesis de Maestría]. http://sedici.unlp.edu.ar/handle/10915/49862. Universidad Nacional de La Plata.

Fishwick, P. (2009, Dec). An Introduction to OpenSimulator and Virtual Environment Agent-based M&S Applications. En *Simulation Conference (WSC), Proceedings of the 2009 Winter* (pp. 177–183). doi: 10.1109/WSC.2009.5429324

Fredes, C., Hernández, J., y Díaz, D. (2012). Potencial y Problemas de la Simulación en Ambientes Virtuales para el Aprendizaje. *Formación Universitaria*, *5*(1), pp 45–56.

Gilakjani, A. P., Leong, L.-M., y Ismail, H. N. (2013). Teachers' Use of Technology and Constructivism. *International Journal of Modern Education and Computer Science*, *5*(4), 49–63. DOI: 10.5815/ijmecs.2013.04.07.

Gilbert, R. (2011). The P.R.O.S.E. (Psychological Research on Synthetic Environments) Project: Conducting In-World Psychological Research on 3D Virtual Worlds. *Journal For Virtual Worlds Research*, *4*(1). Descargado de `https://journals.tdl.org/jvwr/index.php/jvwr/article/view/2108`

Gilbert, R., Murphy, N., Krueger, A., Ludwig, A., y Effron, T. (2013). Psychological Benefits of Participating in 3D Virtual Worlds for Individuals with Real World Disabilities. *International Journal of Disability, Development and Education*, *60*(3), 208–224.

Giordan, A. (1995). LOS NUEVOS MODELOS DE APRENDIZAJE: ¿ MAS ALLA DEL CONSTRUCTIVISMO ? *Perspectivas*, *XXV*(1).

Girvan, C., y Savage, T. (2010). Identifying an appropriate pedagogy for virtual worlds: A Communal Constructivism case study. *Computers & Education*, *55*, pp 342–349.

Goldberg, A., y Robson, D. (1989). *SMALLTALK-80: the language* (First ed.). Addison-Wesley. Paperback.

Guzdial, M., y Rose, K. (2001). *Squeak: Open Personal Computing and Multimedia*. Prentice Hall. Paperback.

Harper, R. (2002). *Second Life History.* http://wiki.secondlife.com/wiki/Second_Life_in_2002/News. Linden Labs. (Accedido: 17 de Enero (2015))

HighFidelity. (2013). *High Fidelity Official Site.* https://highfidelity.io/. (Accedido: 1 de Noviembre (2014))

Hilgard, E. R. (1988). Review of B. F. Skinner's The Behavior of Organisms. *Journal of the Experimental Analysis of Behavior*, *50*(2), 283–286. http://doi.org/10.1901/jeab.1988.50-283.

Holmes, B., Tangney, B., Fitzgibbon, A., Savage, T., y Mehan, S. (2001). Communal Constructivism: students constructing learning for as well as with others. En (pp. pp 3114–3119).

Huang, H.-M., Rauch, U., y Liaw, S.-S. (2010, noviembre). Investigating Learners' Attitudes Toward Virtual Reality Learning Environments: Based on a Constructivist Approach. *Comput. Educ.*, *55*(3), 1171–1182. Descargado de `http://dx.doi.org/10.1016/j.compedu.2010.05.014` doi: 10.1016/j.compedu.2010.05.014

Hut, P. (2008, 9). Virtual Laboratories and Virtual Worlds. En *Dynamical Evolution of Dense Stellar Systems* (Vol. 3, pp. 447–456). Descargado de `http://journals.cambridge.org/article_S1743921308016153` doi: 10.1017/S1743921308016153

Ingalls, D. (1981). Design Principles Behind Smalltalk. *Byte*, *6*(8), 286–298. Descargado de `http://www.bibsonomy.org/bibtex/252d8e3c098d59bb98bb4a14a2bc20343/charoy`

Ingalls, D., Kaehler, T., Maloney, J., Wallace, S., y Kay, A. (1997). Back to the future: the story of Squeak, a practical Smalltalk written in itself. En *OOPSLA '97: Proceedings of the 12th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications* (pp. 318–326). New York, NY, USA: ACM Press. doi: 10.1145/263698.263754

Juhary, J. B. (2006, June). Simulation Technologies And Learning Theories

. En *Simulation Conference Proceedings: Challenges and Opportunities for a Complex and Networked World.*

Konstantinidis, A., Tsiatsos, T., Demetriadis, S. N., y Pomportsis, A. S. (2010). Collaborative Learning in OpenSim by Utilizing SLoodle. En T. Atmaca, J. Palicot, A. Nafkha, T. Tsiatsos, M. Marot, y O. Dini (Eds.), *AICT* (pp. 90–95). IEEE Computer Society. Descargado de `http://dblp.uni-trier.de/db/conf/aict/aict2010.html#KonstantinidisTDP10;http://dx.doi.org/10.1109/AICT.2010.75;http://www.bibsonomy.org/bibtex/264834979f088a334f76c3ae36ad0e48c/dblp`

Korolov, M. (2014). *OpenSim's official upgrade is out; supports varregions and Bullet physics.* http://www.hypergridbusiness.com/2014/06/. (Accedido: 23 de Junio (2015))

Kortemeyer, G., Fish, J., Hacker, J., Kienle, J., Kobylarek, A., Sigler, M., ... others (2013). Seeing and Experiencing Relativity–A New Tool for Teaching? *The Physics Teacher*, *51*, 460–461.

Kotsilieris, T., y Dimopoulou, N. (2013). The Evolution of e-Learning in the Context of 3D Virtual Worlds. *The Electronic Journal of e-Learning*, *11* (2), 147–167.

Kotzer, S., y Elran, Y. (2012, September). Learning and teaching with Moodle-based E-learning environments, combining learning skills and content in the fields of Math and Science & Technology . *1st Moodle Research Conference*, 122–131.

Kypuros, J., y Connolly, T. (2005). Collaborative Experimentation and Simulation: A Pathway to Improving Student Conceptualization of the Essentials of System Dynamics and Control Theory. En (pp. pp 1–11).

Lang, A. S. I. D., y Kobilnyk, D. C. (2009, April). Visualizing Atomic

Orbitals Using Second Life. *Journal of Virtual Worlds Research*, *2*(1).

Laurell, B. (2008). The inner workings of real-time physics simulation engines. En *IRCSE'08 IDT workshop on interesting results in computer science and engineering.*

Leask, M., y Younie, S. (2001). Communal constructivist theory: information and communications technology pedagogy and internationalisation of the curriculum. *Journal of Information Technology for Teacher Education*, *10*(1-2), pp 117–134.

Levesque, J., y Lelievre, E. (2011). Creation and communication in virtual worlds Experimentations with OpenSim. En S. Richir y A. Shirai (Eds.), *VRIC 2011 Proceedings .*

Life, S. (2003). *Second Life Official Site.* http:secondlife.com. Linden Labs. (Accedido: 18 de Junio (2014))

Lim, K. Y. T. (2009, April). The six learnings of Second Life: A framework for designing curricular interventions in-world. *Journal of Virtual Worlds Research*, *2*(1), 3–11.

Lombardi, J., y Lombardi, M. M. (2010). Opening the Metaverse. En W. S. Bainbridge (Ed.), *Online Worlds: Convergence of the Real and the Virtual* (pp. 111–122). Springer.

Lombardi, J., y McCahill, M. P. (2005). User Interfaces for Self and Others in Croquet Learning Spaces. En *C5 '05: Proceedings of the Third International Conference on Creating, Connecting and Collaborating through Computing* (pp. 3–10). Washington, DC, USA: IEEE Computer Society. doi: 10.1109/C5.2005.33

Lopes, C. (2014). The Worst Language Ever Designed : The Case for Better Programming Languages for 3D Environments. En *POPL 2014: 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming*

Languages.

Lopes, C. V., Popov, A., Kan, L., y Morla, R. (2008). PRT Simulation in an Immersive Virtual World. En *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops* (pp. 57:1–57:7). ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). Descargado de `http://dl.acm.org/citation.cfm?id=1416222.1416287`

Malbrán, M. d. C., y Pérez, V. R. (2004). Simulacion mediada por ordenadores. Consideraciones en entornos universitarios. En W. de tecnología informática aplicada en educación (Ed.), *X Congreso Argentino de Ciencias de la Computación.* Descargado de `http://hdl.handle.net/10915/22387`

McCahill, M. P., y Lombardi, J. (2004). Design for an Extensible Croquet-Based Framework to Deliver a Persistent, Unified, Massively Multi-User, and Self-Organizing Virtual Environment. *Creating, Connecting and Collaborating through Computing, International Conference on*, *0*, 71–77. doi: 10.1109/C5.2004.1314372

McCahill, M. P., Moore, P., Wendland, L., y Zampogna, A. (2006). Extending Croquet Spaces with Vector Fields, Vehicles, and Virtual Presence. En *C5 '06: Proceedings of the Fourth International Conference on Creating, Connecting and Collaborating through Computing* (pp. 68–72). Washington, DC, USA: IEEE Computer Society. doi: 10.1109/C5.2006.19

McGeer, R., Raab, A., Reed, D. P., Smith, D. A., y Kay, A. C. (2006). Scalability of Collaborative Environments. En *C5 '06: Proceedings of the Fourth International Conference on Creating, Connecting and Co-*

llaborating through Computing (pp. 168–174). Washington, DC, USA: IEEE Computer Society. doi: 10.1109/C5.2006.32

Memikoglu, I. (2014). Utilization of Second Life as a Tool for Spatial Learning in Interior Architecture. *Procedia - Social and Behavioral Sciences*, *116*(0), 1288–1292. Descargado de `http://www.sciencedirect.com/science/article/pii/S1877042814004017` (5th World Conference on Educational Sciences) doi: 10.1016/j.sbspro.2014.01.384

Mono Open Source Project. (2004). *Mono is a software platform designed to allow developers to easily create cross platform applications .* http://www.mono-project.com/. (Accedido: 20 de Febrero (2015))

Moreira, M. (1997). Aprendizaje significativo: un concepto subyacente. *Actas del Encuentro Internacional sobre el Aprendizaje Significativo*, 19–44.

Morningstar, C., y Farmer, F. R. (2008, July). The lessons of Lucasfilm's Habitat. *Journal of Virtual Worlds Research*, *1*(1), 1–21.

Oliver, I., Miller, A., Allison, C., Kennedy, S., Dow, L., Campbell, A., ... McCaffery, J. (2013). Towards the 3d web with open simulator. En *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on* (pp. 900–909).

OpenCobalt. (2010). *Open Cobalt release 1.0-21 Alpha. - Open Cobalt is free software made available under the MIT open source license.* http://www.opencobalt.org/license. (Accedido: 14 de Julio (2014))

Papert, S. (1980). *Mindstorms: Children, Computers and Powerful Ideas.* Basic Books.

Papert, S. (1987). Learning Environments and Tutoring Systems. En R. W. Lawler y M. Yazdani (Eds.), *Artificial Intelligence and Education* (Vol. 1, pp. 79–94). Norwood, NJ, USA: Ablex Publishing Corp.

Papert, S., y Harel, I. (1991). Situating Constructionism. En S. Papert y

I. Harel (Eds.), *Constructionism* (cap. 1). Norwood, NJ: Ablex Publishing Corporation. Descargado de `http://www.papert.org/articles/SituatingConstructionism.html`

Papert, S., et al. (1999). *Logo Philosophy and Implementation.* Logo Computer Systems Incorporated.

Quinche, J. C., y González, F. L. (2011). Entornos Virtuales 3D, Alternativa Pedagógica para el Fomento del Aprendizaje Colaborativo y Gestión del Conocimiento en Uniminuto . *Revista Formación Universitaria*, *4* (2), 45–54.

Raab, A. (2010, 30 de Marzo). *TeaTime Re: (nebraska?).* http://lists.gforge.inria.fr/pipermail/pharo-project/2010-March/024294.html. (Accedido: 18 de Agosto (2014))

Reed, D. P. (2005). Designing Croquet's TeaTime: A Real-time, Temporal Environment for Active Object Cooperation. En *Companion to the 20th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications* (pp. 7–7). New York, NY, USA: ACM. Descargado de `http://doi.acm.org/10.1145/1094855.1094861` doi: 10.1145/1094855.1094861

Resnick, M. (1997). *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds (Complex Adaptive Systems).* The MIT Press. Paperback.

Resnick, M. (2002). Rethinking Learning in the Digital Age . *The Global Information Technology Report: Readiness for the Networked World*, 32–37.

Rosedale, P. (2002). *Second Life History.* http://wiki.secondlife.com/wiki/Second_Life_in_2002/News. Linden Labs. (Accedido: 17 de Enero (2015))

Skinner, B. F. (1986). Programmed Instruction Revisited. *The Phi Delta Kappan*, *68*(2), 103–110.

Smith, D., Kay, A., Raab, A., y Reed, D. (2003). Croquet - a collaboration system architecture. En *Creating, Connecting and Collaborating Through Computing, 2003. C5 2003. Proceedings. First Conference on* (pp. 2–9). doi: 10.1109/C5.2003.1222325

Smith, D., Raab, A., Ohshima, Y., Reed, D. P., y Kay, A. (2005). Filters and Tasks in Croquet. En *C5 '05: Proceedings of the Third International Conference on Creating, Connecting and Collaborating through Computing* (pp. 50–56). Washington, DC, USA: IEEE Computer Society. doi: 10.1109/C5.2005.14

Smith, D. A., Raab, A., Reed, D. P., y Kay, A. C. (2006). Croquet Programming. - A Concise Guide. - Draft 0.14 [Manual de software informático]. ViewPoints Research Institute. - Qwaq Inc.

Sneha, J. M., y Nagaraja, G. S. (2013, June). Virtual Learning Environments-A Survey. *International Journal of Computer Trends and Technology (IJCTT)*, *4*(6), 1705–1709. Descargado de `http://arxiv.org/abs/1402.2404;http://dblp.uni-trier.de/rec/bib/journals/corr/SnehaN14`

Soler, J., Ferran, P., Jordi, P., y Imma, B. (2012). ACME: Plataforma de Aprendizaje Electrónico (e-learning) con Funcionalidades Deseables en el Ámbito de la Ingeniería. *Formación Universitaria*, *5*, 3–16. Descargado de `http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-50062012000300002&nrm=iso`

Sproull, B. (2010). Alan Kay: A Visionary Designer. En I. Piumarta y K. Rose (Eds.), *Points of View* (pp. 1–4). Viewpoints Research Institute, Inc. hardcover.

Squires, J., y Benmessaoud, F. (2008, June). EXPLORING VIRTUAL WORLDS as a METHOD FOR DELIVERING ONLINE EDUCATION. En J. Luca y E. R. Weippl (Eds.), *Proceedings of World Conference on Educational Media and Technology 2008* (pp. 5444–5448). Vienna, Austria: Association for the Advancement of Computing in Education (AACE). Descargado de `http://www.editlib.org/p/29131`

Stanford University. (2007). *OpenSim is a freely available, user extensible software system that lets users develop models of musculoskeletal structures and create dynamic simulations of movement.* . https://simtk.org/home/opensim/. (Accedido: 12 de Enero (2015))

Stephenson, N. (1992). *Snow Crash*. Bantam Spectra Books. hardcover.

Sutherland, I. E. (1963). *SKETCHPAD: A Man-Machine Graphical Communications System* (Tesis Doctoral). Descargado de `http://sandbox.puma.bibliothek.uni-kassel.de/bibtex/2879e28259e251b789642815d52c33d45/alistair`

Sutherland, I. E. (1965). The Ultimate Display. En *Proceedings of the Congress of the International Federation of Information Processing (IFIP)* (Vol. 2, pp. 506–508).

Sutherland, I. E. (2012). The TX-2 Computer and Sketchpad. *Lincoln Laboratory Journal*, *19*(1), 82–84.

Takada, H. (2007). A 3D Collaborative Creation Environment with Tile Programming on Croquet. En *C5 '07: Proceedings of the Fifth International Conference on Creating, Connecting and Collaborating through Computing* (pp. 125–130). Washington, DC, USA: IEEE Computer Society. doi: 10.1109/C5.2007.1

Tarouco, L., Gorziza, B., Corrêa, Y., Amaral, É. M. H., y Müller, T. (2013,

March). Virtual laboratory for teaching Calculus: An immersive experience. En *Global Engineering Education Conference (EDUCON), 2013 IEEE* (pp. 774–781). doi: 10.1109/EduCon.2013.6530195

Teleplace. (2009). *Teleplace - A powerful next-generation solution for virtual teams* (Inf. Téc.).

Trafford, P., y Shirota, Y. (2011). *An Introduction to Virtual Learning Environments* (Inf. Téc.). University of Gakushuin.

Tsiatsos, T., Konstantinidis, A., y Pomportsis, A. S. (2010). Evaluation Framework for Collaborative Educational Virtual Environments. *Educational Technology & Society*, *13*(2), 65–77. Descargado de `http://dblp.uni-trier.de/db/journals/ets/ets13.html#TsiatsosKP10`

Warburton, S. (2009). Second Life in higher education: Assessing the potential for and the barriers to deploying virtual worlds in learning and teaching. *British Journal of Educational Technology*, *40*(3), 414–426. Descargado de `http://www.bibsonomy.org/bibtex/2529243c398ebfaa8ea65f957a2f5b5bd/vwedu`

Wild, W. J. (1980). Euler's Three-body Problem. *American Journal of Physics*, *48*(4), 297–301.

Wright, T. E., y Madey, G. (2009). A Survey of Technologies for Building Collaborative Virtual Environments . *The International Journal of Virtual Reality*, *8*(1), 53–66.

Yardley, J. (2013, September). *Huge problem for all creators in Second Life.* https://joyardley.wordpress.com/2013/09/08/huge-problem-for-all-creators-in-second-life/. (Accedido: 14 de Julio (2014))

Zamora, B., y Kaiser, A. S. (2009). Enseñanza de Temas Avanzados de Mecánica de Fluidos usando Dinámica de Fluidos Computacional. *Revista Formación Universitaria*, *2*(1), 27–36.

Índice de cuadros

Índice de figuras