BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

“Sistema de Mentoría BUAP. Un Ambiente Virtual 3D”

TESIS PROFESIONAL QUE PRESENTA

José González Gaspar

PARA OBTENER EL TÍTULO DE

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

ASESOR:

Dr. Mario Anzures Garcia

COASESOR:

M.C. María Luz Adolfina Sánchez Gálvez

Puebla, Pue. **Enero 2013.**

Esta investigación fue realizada gracias al apoyo del Consejo de Ciencia y Tecnología del Estado de Puebla.

Agradecimientos

A Dios.

Porque de él recibo fuerza, salud, esperanza y sed de superación para ser mejor persona tanto profesional como moralmente, con el fin de contribuir junto a los demás, al mejoramiento de nuestra sociedad.

A mis padres y hermano:

Teodoro González Corril y Estela Gaspar Vázquez.

Miguel Ángel González Gaspar.

Por su amor, disciplina, confianza y apoyo. Gracias por inculcar en mí todas sus experiencias, consejos y valores.

A mis asesores:

Dr. Mario Anzures García y M.C. María Luz Adolfina Sánchez Gálvez.

Por otorgarme el honor de participar bajo su asesoría en este proyecto de tesis. Gracias por su apoyo hacia el tema, por su tiempo, confianza, paciencia, sugerencias y disciplina.

A mi familia:

Por su gran apoyo, sobre todo por conocerme bien y estar siempre conmigo. Nunca olvidaré los consejos y apoyo que recibí de cada uno de ustedes.

A mis amigos, sinodales, catedráticos y demás personas que de diferentes maneras, me brindaron ayuda, sugerencias, críticas y consejos en este proyecto.

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1. Introducción

En la actualidad, la web ha propiciado la creación de un ciberespacio virtual que permite la interacción entre personas de diferentes lugares, creencias, culturas, ideologías, entre otros aspectos, pero con intereses en común. En caso de ser un interés académico se tienen las aplicaciones de aprendizaje colaborativo; cuando se buscan vínculos con los integrantes de una sociedad se encuentran las redes sociales; si desean compartir archivos se encuentran los espacios de trabajo compartido; si se busca comunicarse con otras personas se puede hacer uso del correo electrónico, el chat o las videoconferencias. Otra manera de interactuar es a través de los ambientes virtuales 3D como: *Second Life* y *Microsoft's New Xbox Experience.*

Un aspecto importante del incremento en el uso de los ambientes virtuales es que el usuario puede representarse así mismo a través de *avatars*, que le dan la sensación de ser parte del ambiente utilizado. En *Second Life* los *avatars* interactúan con el ambiente virtual a través de objetos programables creados con *Linden Scripting Language* (LSL), que es un lenguaje *script* que permite programar los objetos como máquinas de estado, permitiendo al *avatar* controlar lo que el objeto debería hacer dependiendo de su estado actual, lo cual proporciona al desarrollador del ambiente virtual la flexibilidad necesaria para que sus objetos hagan lo que ellos deseen.

Este trabajo presenta una plataforma que soporta al Sistema Institucional de Mentoría BUAP, creado por la Coordinación de Acompañamiento al Estudiante (CAE) como una figura complementaria del Sistema Institucional de Tutoría. Esta plataforma es un ambiente virtual 3D que simula las Mentorías, el entorno está formado por edificios y áreas dónde los mentores y estudiantes pueden interactuar y colaborar en beneficio del alumno.

"Sistema de Mentoría BUAP. Un Ambiente Virtual 3D" es la implementación de un espacio virtual 3D exclusivo de profesores y alumnos de la BUAP en un servidor creado con *Opensim*, a través de herramientas de diseño de edificios y objetos 3D; los objetos están programados con LSL, así como la utilización de otros lenguajes como HTML (*HyperText Markup Language)*, PHP (*HyperText Pre-processor*), *MySQL* y *C#*, con el objetivo de generar un ambiente que establezca la comunicación entre usuarios remotos que realizan tareas complejas, tales como la suministración de tutorías por medio de la interacción con *avatars* sin necesidad de estar de manera presencial, la presentación de video tutoriales, la interacción con objetos 3D, entre otras. En el ámbito psicológico, se busca terminar con aquellas actitudes que impiden a los alumnos finalizar un proceso académico o administrativo que les beneficia, como son el tener dudas de un servicio, nerviosismo al solicitar información, miedo a estar equivocado, entre otros. Además de ello, se busca la reducción de tiempo que implica acudir a un centro de información para cerciorarse de los procesos que desarrolla la BUAP.

El proyecto de tesis está organizado de la siguiente manera: En el presente capítulo se muestra la justificación, objetivos generales y específicos, metodología e infraestructura del proyecto. Capítulo 2, Marco Teórico del Ambiente Virtual 3D. Capítulo 3, Análisis del sistema de Mentoría BUAP. Capítulo 4, Diseño y desarrollo del espacio virtual. Capítulo 5, se presenta la interfaz final del Sistema de Mentoría. Por último se presentan las conclusiones y trabajo futuro.

1.1. Justificación

En la actualidad uno de los grandes desafíos de la Educación Superior en América Latina es disminuir los elevados índices de rezago y deserción. La Benemérita Universidad Autónoma de Puebla (*BUAP*), no escapa a este tipo de problemas de baja retención de sus estudiantes, es por ello que ha retomado como estrategia la concentración de servicios y apoyos académicos mediante la creación de una entidad que funcione como eslabón entre las dependencias universitarias, tanto académicas como de apoyo a los estudiantes. Para esto, surge la Coordinación de Acompañamiento al Estudiante, sustentada en el Plan de Desarrollo Institucional y en el Modelo Académico; su objetivo fundamental es "propiciar la mejora de la trayectoria escolar de los estudiantes al favorecer la detección y la atención oportuna de las distintas necesidades y demandas que presentan durante su trayectoria académica, mediante la sistematización de información y acciones de las dependencias involucradas en atención del estudiante" [1].

La Coordinación de Acompañamiento al Estudiante (*CAE*) crea el Sistema Institucional de Mentoría BUAP, figura complementaria del Sistema Institucional de Tutoría, que tiene como fin la atención de estudiantes de cuatrimestres avanzados a estudiantes de cuatrimestres inferiores con el fin de resolver los problemas de rezago y abandono de los estudios; busca mejorar sus resultados académicos, con la finalidad de mejorar la eficiencia terminal de las distintas carreras que ofrece. Asimismo trata de facilitar la integración académica y social en la vida universitaria de los alumnos de reciente ingreso.

Actualmente, en toda institución de formación académica se llevan a cabo procesos en beneficio de los alumnos para su pronta titulación, con el fin de que sean incorporados al mundo laboral y contribuir en el beneficio de nuestra sociedad, sin embargo muchas veces estos objetivos no se cumplen, por la falta de atención de la institución a sus alumnos. La institución siempre está al servicio del alumno, sin embargo los alumnos no siempre llevan a cabo de manera correcta cada proceso solicitado por la unidad académica; los factores que

impiden estos objetivos se centran generalmente en factores físicos (tiempo y costo para ir al lugar donde se realizan los trámites o se brinda información) y psicológicos (miedo a preguntar, temor a equivocarse, nerviosismo, entre otros). Es por ello que con la implementación de mundos virtuales se pretende acabar con los inconvenientes antes mencionados para promover un seguimiento a los alumnos rezagados, brindar información de servicios y apoyar a estudiantes de manera personal, sin la necesidad de la presencia física en el mundo real sino a través de un mundo virtual 3D creado exclusivamente para alumnos y catedráticos de la BUAP.

En ese mismo sentido y considerando la relevancia que han tomado las Tecnologías de la Información y Comunicación (*TIC´s)* en todos los ámbitos; se pretende contar con una plataforma colaborativa virtual que sirva de soporte al Sistema Institucional de Mentoría BUAP.

La creación de mundos virtuales es el siguiente paso para la comunicación entre personas, puesto que en la interacción entre usuarios se aplican los conocimientos basados en la Programación Neurolingüística (PNL), que establece que el ser humano posee tres grandes sistemas para representar la información: visual, auditivo y *kinestésic*o; *Second Life* es una herramienta virtual que permite de alguna manera simular estos sistemas de representación de información a través de un *avatar*, que personifica al usuario en el mundo virtual, lo cual permitirá una mayor interacción entre los usuarios del Sistema de Mentoría, consultando información con la persona responsable sin la necesidad de estar de manera presencial, lo cual genera mayor confianza, mejor comunicación, ahorro de tiempo al no ir físicamente al lugar donde se encuentra la información y mayor interacción con los servicios que la BUAP ofrece a sus alumnos. Esta plataforma es un ambiente virtual 3D para soportar las Mentorías, que contiene edificios y áreas donde los mentores y estudiantes pueden interactuar y colaborar en beneficio del alumno; y por otra parte, un conjunto de servicios para permitir dicha interacción entre los mentores y los alumnos asignados a éste.

El proyecto cuenta con un objetivo general y varios específicos que permiten establecer sus alcances y limitaciones.

1.2. Objetivo General

Desarrollar un entorno virtual 3D para el sistema Institucional de Mentoría BUAP, que proporcione servicios de tutoría a alumnos de la BUAP, por medio de la interacción con objetos del espacio virtual o el trato directo con otros *avatars*.

1.3. Objetivos Específicos.

A continuación se presentan los objetivos específicos del Sistema de Mentoría BUAP:

- Crear cuatro islas que corresponderán a los programas que se actualizarán en la impartición de servicios por medio de un mundo virtual.
- Crear edificios, muebles, entre otros objetos que formarán parte del entorno 3D.
- Importar objetos 3D creados en otras herramientas de diseño al mundo virtual.
- Incrustar y programar objetos que permitan interactuar con páginas web.
- Programar puertas, sillas, sillones, entre otros objetos para establecer el comportamiento del avatar al entrar en contacto con ellos.
- Incrustar y programar pantallas en el mundo virtual que permitan la presentación de videos para las personas que accedan al mundo.
- Crear una página web que permita el registro de un alumno o catedrático al sistema virtual de Mentoría.

1.4. Metodología

El presente trabajo se basó en los siguientes modelos prescriptivos de la ingeniería de software:

- **Incremental:** Se establecieron etapas de desarrollo para obtener el producto final, iniciando con la propuesta, posteriormente el diseño, programación, registro y producto terminado.

- **Prototipos:** Se establecieron modelos de diseño del mundo virtual y fueron mostrados para ser evaluados por el CAE, añadiendo propuestas para su mejora.

- **Cascada:** Los prototipos fueron desarrollados con esta metodología, para su fácil obtención y ser anexados sin inconvenientes al producto final.

1.5. Infraestructura

Para el desarrollo del Sistema de Mentoría BUAP, se solicitó la infraestructura ideal para su soporte y mantenimiento, sus características son las siguientes:

- Procesador Quad-Core Intel® Xeon®Processor E5606 (2.13 GHz, 8MB L3 Cache, 80W, DDR3-800)
- Cache Memory 8MB shared L3 cache
- Memory 4GB (1 x 4GB) PC3-10600E (RDIMM)
- Network Controller HP NC362i Integrated Dual Port Gigabit Server Adapter
- Storage Controller HP B110i SATA RAID Controller
- Hard Drive Up to 4 Large Form Factor Puggable DrivesInternal Storage Up to 4 large form factor Hot Plug drives
- Optical Drive (1) HP Slim 9.5 mm SATA DVD-RW
- Power Supply 500W high efficiency multi-output supply (ships standard)
- Form Factor 1U"
- 500658-B21 HP 4GB 2Rx4 PC3-10600R-9 Kit
- 507614-B21 HP 1TB 6G SAS 7.2K 3.5in DP MDL HDD
- 504568-B21 MS W2008 SBS Prm 5 Dev CAL SPA Lic
- Monitor HP LCD S9133, 18.5, WIDE 5MS, 600:1,1366 X 768, 3/3/3 WEBCAM
- Kit de teclado y mouse
- No Break 500 va
- Tarjeta Gráfica para manejo de gráficos 3D:
- Tarjeta de Video NVIDIA GetForce GTX 550 Ti, 1GB GDDR5, MiniHDMI, Dual DVI, Puerto PCI Express 2.0.

Capítulo 2. Marco Teórico del Ambiente Virtual 3D

Para poder diseñar e implementar el “Sistema de Mentoría BUAP. Un Ambiente Virtual 3D”, se realizaron investigaciones sobre las herramientas que ayudarían a desarrollarlo, estas investigaciones varían desde métodos para desarrollo del *software*, hasta los lenguajes de programación utilizados en el proyecto. Las investigaciones que contribuyeron a la realización del “Sistema de Mentoría BUAP. Un Ambiente Virtual 3D” se describen en el presente capítulo.

2.1. Ingeniería de *Software*

Cuando se desarrolla un proyecto, es importante seguir una metodología que garantice el cumplimiento de los objetivos. Actualmente existe una rama de la computación que estudia los modelos para crear productos de calidad: La ingeniería de *software*.

La Ingeniería del *software* es una disciplina o área de la informática o ciencias de la computación, que ofrece métodos y técnicas para desarrollar y mantener *software* de calidad que resuelven problemas de todo tipo [2].

La ingeniería de *software* es el establecimiento y uso de principios sólidos de la ingeniería para obtener económicamente un *software* confiable y que funcione de modo eficiente en máquinas reales.

El instituto de Ingenieros Eléctricos y Electrónicos, conocido por sus siglas en inglés IEEE (*Institute of Electrical and Electronics Engineers*) establece que: La ingeniería de *software* es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del *software*; es decir, la aplicación de la ingeniería al software.

La ingeniería de *software* es una disciplina que integra al proceso, los métodos y las herramientas para el desarrollo del *software*.

"Más que una disciplina o un cuerpo de conocimiento, la ingeniería es un verbo, una palabra de acción, una manera de abordar el problema". *Scott Whitmire* [3].

La ingeniería de *software* se emplea en el desarrollo de toda aplicación computacional, para poder establecer un producto de calidad acorde a las necesidades del cliente y/o usuario final.

2.1.1. Modelos prescriptivos de proceso

Los modelos prescriptivos de proceso se propusieron originalmente para ordenar el caos del desarrollo de *software*. La historia ha indicado que estos modelos convencionales han traído consigo cierta cantidad de estructuras útiles para el trabajo en la ingeniería de *software*, y han proporcionado un camino a seguir razonablemente efectivo para los equipos de *software* [2].

2.1.1.1. Modelo en cascada

El modelo en cascada o ciclo de vida clásico, es el enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de *software*, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior [4].

Un ejemplo de una metodología de desarrollo en cascada es:

1. Análisis de requisitos.
2. Diseño del Sistema.

3. Diseño del Programa.
4. Codificación.
5. Pruebas.
6. Implantación.
7. Mantenimiento.

En la figura 2.1 se muestra el modelo en cascada o ciclo de vida clásico utilizado para el desarrollo de *software.*

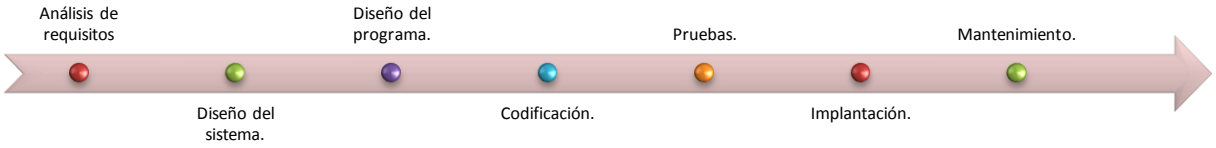


Figura 2.1 Modelo en cascada o ciclo de vida clásico.

- **Análisis de requisitos.** En esta fase se analizan las necesidades de los usuarios finales del *software* para determinar qué objetivos debe cubrir. De esta fase surge una memoria llamada SRD (*Software Requeriments Document*), que contiene la especificación completa de lo que debe hacer el sistema sin detalles internos.

- **Diseño del Sistema.** Descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. Como resultado surge el SDD *(Software Design Document*), que contiene la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras.

- **Diseño del Programa.** Es la fase en donde se realizan los algoritmos necesarios para el cumplimiento de los requerimientos del usuario así como también el análisis para determinar las herramientas a utilizar en la etapa de codificación.

- **Codificación.** Es la fase en donde se implementa el código fuente, haciendo uso de prototipos así como de pruebas y ensayos para corregir errores.

- **Pruebas.** Los elementos ya programados se ensamblan para componer el sistema, se comprueba que funciona correctamente y que cumple con los requisitos antes de ser entregado al usuario final.

- **Implantación o verificación.** Es la fase en donde el usuario final ejecuta el sistema, para ello el o los programadores ya realizaron exhaustivas pruebas para comprobar que el sistema no falle.

- **Mantenimiento.** Es el mantenimiento del *software* ya que al utilizarlo como usuario final puede ser que no cumpla con todas nuestras expectativas [3].

2.1.1.2. Construcción de prototipos

A menudo un cliente define un conjunto de objetivos generales para el *software*, pero no identifica los requisitos de entrada, procesamiento o salida. En otros casos, el responsable del desarrollo del *software* está inseguro de la eficacia de un algoritmo, de la adaptabilidad de un sistema operativo o de la forma que debería tomar la interacción humano-máquina. En éstas y en muchas otras

situaciones, un paradigma de construcción de prototipos puede ofrecer el mejor enfoque [2].

El paradigma de construcción de prototipos (Figura 2.2) se basa generalmente en cinco etapas.

Figura 2.2 Modelo de construcción de prototipos.

- **Comunicación:** El ingeniero de *software* y el cliente encuentran y definen los objetivos globales para el *software*, identifican los requisitos y las áreas del esquema en donde es necesaria más definición.

- **Plan rápido:** Se plantea con rapidez una iteración de construcción de prototipos y se presenta el modelado.

- **Modelado del diseño rápido:** Se centra en una representación de aquellos aspectos del *software* que serán visibles para el cliente o el usuario final (por ejemplo, la configuración de la interfaz con el usuario y el formato de despliegue de salida). El diseño rápido conduce a la construcción de un prototipo.

- **Construcción del prototipo:** Los acuerdos establecidos en fases anteriores, dan pauta a la construcción del prototipo, resaltando la interfaz acordada con el usuario.

- **Desarrollo, entrega y retroalimentación:** Una vez concluido el prototipo, se entrega al cliente, éste lo evalúa (cliente y usuario final) y con la retroalimentación se refinan los requisitos del *software* que se desarrollará.

La iteración ocurre cuando el prototipo se ajusta para satisfacer las necesidades del cliente. Esto permite al mismo tiempo que el desarrollador entienda mejor lo que debe hacer.

De manera ideal, el prototipo debería servir como un mecanismo para identificar los requisitos del *software* [2].

2.1.1.3. Modelo Incremental

El modelo incremental combina elementos del modelo en cascada aplicado en forma iterativa. Como se muestra en la figura 2.3, el modelo incremental aplica secuencias lineales de manera escalonada conforme avanza el tiempo en el calendario. Cada secuencia lineal produce "incrementos de *software*" [2].

El desarrollo incremental es útil sobre todo cuando el personal necesario para una implementación completa no está disponible, es decir, si el desarrollo del sistema implica la utilización de *hardware* complejo, se puede empezar a desarrollar las etapas básicas antes de implementar en el *hardware* final, aún cuando este se encuentre en desarrollo y cuya fecha de entrega es incierta.

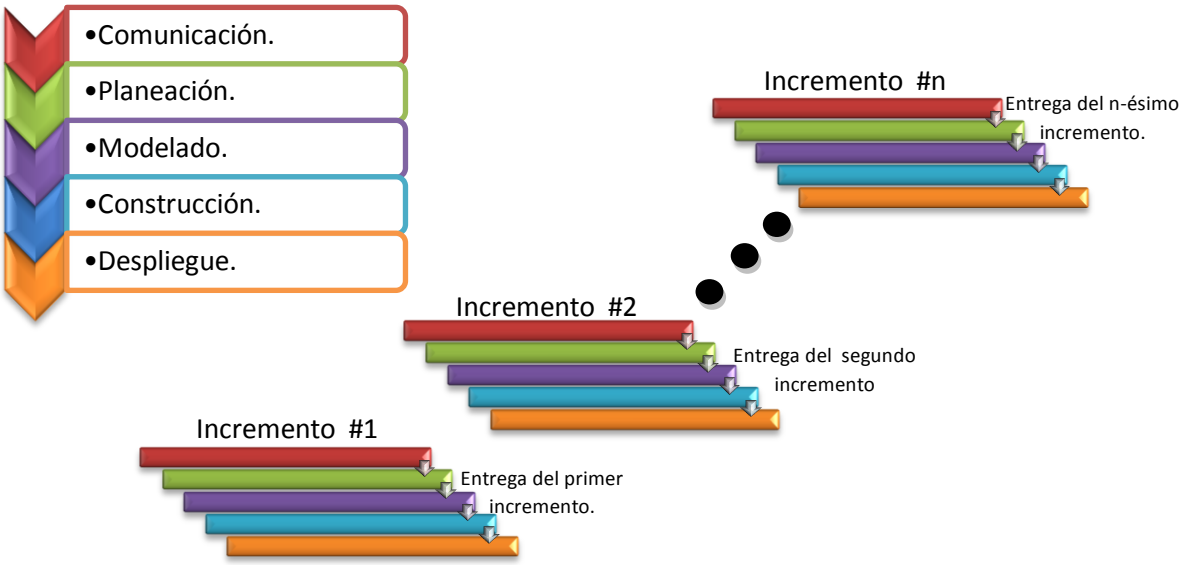


Figura 2.3 Modelo incremental.

- **Comunicación:** Se definen los objetivos globales para el *software* final, se identifican los requisitos y las áreas del esquema en donde es necesaria más definición para la entrega de cada incremento.

- **Planeación:** Se plantea una mecánica de desarrollo para presentar el avance solicitado por el cliente.

- **Modelado:** Se centra en una representación de aquellos aspectos del *software* de acuerdo a la fase a implementar que serán visibles para el cliente o el usuario.

- **Construcción:** Se implementa el avance del proyecto.

- **Despliegue:** Una vez concluida la fase, se entrega el incremento al cliente, éste lo evalúa (cliente y usuario final) y con la retroalimentación se definen los detalles a implementar en la siguiente fase.

2.2. Base de Datos

Una Base de Datos (BD) se define como un fondo común de información almacenada en una computadora para que cualquier persona o programa autorizado pueda acceder a ella, independientemente de su procedencia y del uso que se haga [5].

Un sistema de bases de datos es básicamente un sistema computarizado para llevar registros, un depósito o contenedor de una colección de archivos de datos computarizados [6]. Los usuarios del sistema pueden realizar una variedad de operaciones sobre dichos archivos, por ejemplo:

- Agregar nuevos archivos en la BD.
- Agregar nuevos datos en los archivos existentes.
- Recuperar datos de los archivos existentes.
- Modificar datos en archivos existentes.
- Eliminar archivos existentes de la BD.
- Eliminar datos de los archivos existentes.

2.2.1. Sistema Manejador de Base de Datos

Un Sistema Manejador de Base de Datos (conocido por sus siglas en inglés: *DataBase Management System:* DBMS), es una serie de programas que permiten crear y eliminar BDs y que proporciona al usuario los mecanismos para dotar a las BDs de contenido y acceder a su información. Además dispone de una batería de utilidades para garantizar la disponibilidad y la seguridad de su contenido [5].

Un Sistema Manejador de Base de Datos consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. El objetivo principal es proporcionar una forma de almacenar y recuperar la información de una BD de manera que sea tanto práctica como eficiente [7].

En la figura 2.4 se puede visualizar un diagrama general de lo que define un DBMS.

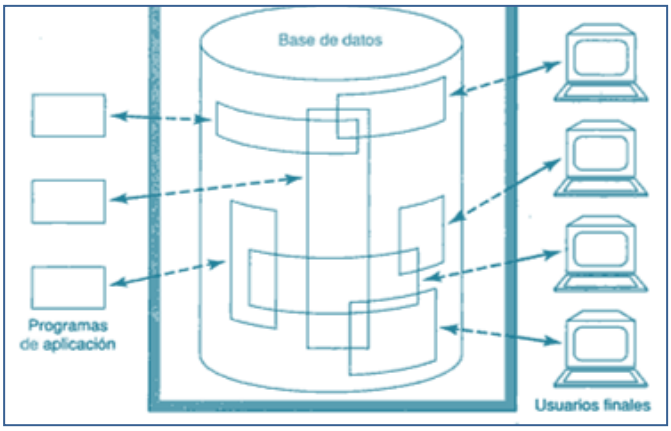


Figura 2.4 DBMS: Sistema Manejador de Base de Datos.

2.2.2. Usuarios de la Base de Datos

Los usuarios constituyen el elemento fundamental del concepto, es decir, el concepto de BD se articula con el propósito de satisfacer las necesidades de información planteadas por los usuarios. En función del uso que hagan del sistema podemos clasificarlos en tres categorías [4]:

- **Usuario Terminal (último o final):** Usuario que usa la BD para acometer sus actividades, de forma directa o a través de programas de aplicación. Los usuarios terminales no necesitan tener conocimiento alguno de cómo está organizado ni implantado el sistema, tan sólo deben conocer cuáles son sus funciones administrativas y la forma en que se maneja el programa de aplicación que les concierne.

- **Programador de aplicaciones:** Se encarga de desarrollar programas de aplicación sobre la BD que proporcionen soporte a las funciones desempeñadas por los usuarios terminales. Necesita recopilar de los usuarios, los requisitos de la aplicación que se van a desarrollar. Interactúa con la BD a través del esquema lógico que concierne a su aplicación.

- **Administrador de la BD (DBA-*DataBase Administrator*):** Es el encargado de gestionar todos los componentes del DBMS para que la BD represente e integre adecuadamente todos los elementos de información precisados por las aplicaciones y los usuarios. Además, debe garantizar la operatividad y la seguridad de la base de datos.

2.3. Programación Neurolingüística

La Programación Neurolingüística (PNL) consiste en el estudio de realidades psicológicas individuales y en la elaboración de medios de observación, de codificación y de acción. En otras palabras, observa la conducta, destaca regularidad e intenta clasificar las observaciones con el propósito de elaborar estrategias adaptadas a las dificultades que se encuentran [8]. La programación neurolingüística se compone de las siguientes palabras:

- **Programación:** Proviene de la ciencia del proceso de la información, bajo la premisa de que la manera en que se almacena, se codifica y se transforma. La experiencia es similar a cómo funciona el *software* en una Computadora Personal (PC: *Personal Computer*); suprimiendo, actualizando o instalando nuestro *software* mental, podemos cambiar la manera de pensar y, como resultado, la manera de actuar.

- **Neuro:** Proviene de neurología, la manera en que procesamos la información que nos llega de los cinco sentidos a través del cerebro y el sistema nervioso.

- **Lingüística:** Posee relación con el uso de los sistemas del lenguaje (no sólo las palabras, sino todos los sistemas de símbolos, incluyendo gestos y posturas) para codificar, organizar y atribuir significados a nuestras

representaciones internas del mundo, y para comunicarnos interna y externamente.

Cuando se unen las palabras, la programación neurolingüística tiene que ver con los procesos que utilizamos para crear una representación interna (nuestra experiencia) del mundo exterior, de la realidad, a través del lenguaje y nuestra neurología [9].

Los seres humanos habitualmente desarrollamos un sentido más que los otros, nos hacemos más sensibles a la vista, el oído o lo kinestésico, es decir, tacto, gusto y olfato [10].

En PNL, los sentidos forman sistemas representacionales, porque con ellos se representa la experiencia interna o externa que se vive, y se clasifican en tres categorías:

- **Visual:** Implica la capacidad de recordar imágenes vistas con anterioridad y la posibilidad de crear nuevas, así como de transformar las ya vistas.

- **Auditivo:** Es la capacidad de recordar palabras y sonidos escuchados con anterioridad y de formar otros nuevos.

- **Kinestésico:** Incluye las sensaciones corporales, táctiles, viscerales y las propioceptivas (la sensación del movimiento de los músculos, por ejemplo), las emociones, sabores y olores.

En PNL los sistemas representacionales tienen mucho mayor significado funcional del que se les atribuye en los modelos clásicos, en los que los sentidos se consideraban como mecanismos pasivos de entrada.

Conviene señalar que usamos todos los sentidos, aunque tengamos un sistema preferente a la hora de procesar la información.

Así como un visual desarrolla mayor sensibilidad a los colores, un auditivo prestará mayor atención a los ruidos, los silencios, las palabras; y un kinestésico a todo lo que se relacione con la experiencia del tacto, el movimiento, la sensación visceral [9].

2.4. Entornos Colaborativos

El trabajo colaborativo se define como aquel proceso intencional de un grupo para alcanzar objetivos específicos, más herramientas diseñadas para dar soporte y facilitar el trabajo [11].

En el marco de una organización, el trabajo en grupo con soporte tecnológico se presenta como un conjunto de estrategias tendientes a maximizar los resultados y minimizar la pérdida de tiempo e información en beneficio de los objetivos organizacionales.

El mayor desafío es lograr la motivación y participación activa del recurso humano. Además deben tenerse en cuenta los aspectos tecnológicos, económicos y las políticas de la organización.

Un entorno colaborativo designa diferentes elementos en el cual todos los participantes del proyecto trabajan, colaboran y se ayudan para la realización de un proyecto; el hecho de pertenecer a un grupo con un objetivo en común permite estrechar lazos en los participantes y les genera sentido de pertenencia. Claro está que el objetivo de un trabajo colaborativo es producir algo, que puede ser un conocimiento o un objeto tangible; pero en ocasiones el hecho de pertenecer a una organización también puede ser el objetivo de los participantes; por lo tanto la motivación puede ser también intrínseca del propio proyecto.

2.5. Mundos Virtuales

Un mundo virtual es un tipo de comunidad en línea que simula un mundo o entorno artificial inspirado o no en la realidad, en el cual los usuarios pueden interactuar entre sí a través de personajes o *avatars* que personifican al usuario en el entorno y usar objetos o bienes virtuales [12].

Para hacer un espacio virtual se requiere un mundo en línea persistente, activo y disponible 24 horas al día y todos los días. Los mundos virtuales son hechos para que los usuarios vivan e interactúen, generalmente en tiempo real. Los personajes o *avatars* son representados por gráficos en 2D o 3D según el mundo virtual [13].

Aparecieron mundos virtuales con fines profesionales de aprendizaje (simuladores de vuelo), de enseñanza, entre otros; pero en la actualidad está siendo llevado por las empresas de ocio electrónico, que ven en esta tecnología una nueva era para videojuegos. Aunque no son limitados en videojuegos, muchos de estos mundos virtuales son conocidos como videojuegos masivos en línea o MMO (*Massively Multiplayer Online Games*) [12].

Existen varias universidades que han empezado a desarrollar proyectos respecto a la utilización de mundos virtuales como nueva herramienta para la proporción de servicios y asesorías, entre ellos destacan los siguientes:

"*REMOTELY INTERACTING WITH A ROBOT ARM IN SECOND LIFE VIA A WEB INTERFACE REPORT*" [14] de la universidad de Ulster en Inglaterra, que implementa un brazo robot, que es manipulado por una serie de parámetros que es enviado por medio de una página web, y que es interpretado por lenguajes C++ y LSL.

En América latina existe el sitio "Argentonia", que provee a empresas, organizaciones e instituciones un amplio espectro de servicios de consultoría y desarrollo 3D utilizando como plataforma el entorno gráfico de *Second Life*, entre

los sitios virtuales del país latino, destaca la creación de la Universidad Argentina de la Empresa *(UADE*), para promover la educación a distancia [15].

En el ámbito de asesorías y tutorías se encuentra "*Virtual Worlds Best Practices in Education Conference*", un presentador de conferencias que se centra en la enseñanza/aprendizaje, el trabajo académico, proyectos, eventos, actividades y herramientas nuevas e innovadoras para la educación virtual. La conferencia incluye múltiples formatos, incluyendo conferencias, talleres, tutoriales en el lugar y visitas fuera de lugar, mesas redondas, paneles de discusión y multimedia [16].

El proyecto *Second USMP (Universidad de San Martín de Porres)* concentra todos los esfuerzos de la USMP dentro del universo virtual *Second Life*, con el objetivo básico de comprender los nuevos espacios donde habitan los nativos digitales y desarrollar habilidades para poder ayudar a los residentes a realizar un buen uso del espacio virtual, aplicando métodos de enseñanza que promuevan el aprendizaje en los alumnos [17].

En México, la universidad del estado de Guerrero creó el CERV (*Centro de Estudios sobre Realidad Virtual*), un campus educativo dentro del mundo virtual de *Second Life* [18]. Además de prestar servicios, el campus ofrece materiales didácticos a los alumnos, organiza exposiciones, clases, conferencias y cuenta con varios foros virtuales en los cuales exponen sus ideas y opiniones, o discuten sobre temas relacionados con las matemáticas y áreas de estudio.

Todos los trabajos mencionados anteriormente constituyen una visión general de los alcances que ha tenido el desarrollo de mundos virtuales, en cuanto al intercambio de información, estos ambientes poseen una ventaja sobre los sistemas de educación desarrollados hoy en día en cuanto a la educación a distancia, que se basa en wikis, foros, chat, entre otros; ya que su interpretación del mundo real, implica los conocimientos proporcionados por la Programación

Neurolingüística (PNL), que indica que la percepción de información se da de forma visual, auditiva y kinestésica.

La importancia del uso de las nuevas tecnologías en el aprendizaje moderno, se ve plasmada en el ahorro de tiempo y costos para los usuarios, ya que pueden recibir asesorías en su casa, disfrutando de todos los beneficios visuales y auditivos de una clase presencial, sin necesidad de estar físicamente en ella, estableciendo interacción síncrona con los integrantes del mundo virtual.

2.5.1. Second Life

Second Life (SL) o Segunda vida, es un *metaverso* lanzado el 23 de junio de 2003, desarrollado por *Linden Lab*, al que se puede acceder gratuitamente en internet. Sus usuarios, conocidos como "residentes", pueden acceder a SL mediante el uso de uno de los múltiples programas de interfaz llamados *viewers* (visores), lo cual les permite interactuar entre ellos mediante un *avatar*. Los residentes pueden así explorar el mundo virtual, interactuar con otros residentes, establecer relaciones sociales, participar en diversas actividades tanto individuales como en grupo, así como crear y comerciar propiedad virtual y servicios entre ellos [19].

Para acceder al programa es requisito imprescindible crear una cuenta, la cual brinda acceso al mundo y al avatar individual. Los *avatars* son caracteres tridimensionales que pueden alterarse físicamente (Figuras 2.5) y personalizarse con datos del usuario final (Figuras 2.6) lo que le da a los usuarios la capacidad de convertirse en el personaje que deseen y "gozar" (como el mismo nombre del programa indica) de una segunda vida.

Figura 2.5 *Avatars* proporcionados por SL.

Figura 2.6 Personalización de los *avatars.*

Su atractivo más importante es la posibilidad de crear objetos e intercambiar diversidad de productos virtuales a través de un mercado abierto que tiene como moneda local el *Linden Dólar* (L$). En el mismo programa se incluye una herramienta de creación en 3D basada en simples figuras geométricas (conocidos como *prims* o primitivas) que permite a los residentes la construcción de objetos virtuales. Estos elementos pueden usarse en combinación con el lenguaje de programación LSL a fin de añadir funcionalidad a los objetos. Objetos más complejos, como *sculpties*, texturas para ropas u objetos, animaciones o gestos pueden ser creados externamente e importados a SL. Los Términos de Servicio de SL (conocidos por su abreviatura inglesa: TOS, *Terms Of Service*) aseguran hasta recientes cambios la retención de los derechos de autor por parte de los creadores de los objetos, mientras que *Linden Lab* proveía simples funciones de gestión de derechos digitales en sus servidores y su acceso [18].

2.5.2. OpenSimulator

OpenSimulator (*Opensim*) es un código abierto multi-plataforma y multi-usuario para generar un servidor de aplicaciones 3D. Puede ser utilizado para crear un entorno virtual 3D al que se puede acceder a través de una variedad de clientes, en múltiples protocolos. *Opensim* permite a los desarrolladores del mundo virtual personalizar sus mundos; está escrito en *C#* , se ejecuta tanto en Windows a través del *Framework* .NET como en Unix con el *Framework* Mono . El código fuente se distribuye bajo una licencia BSD (*Berkeley Software Distribution)*. *Opensim* se puede utilizar para simular entornos virtuales similares a SL [20].

Las características principales del simulador de aplicaciones 3D, se definen a continuación:

- Soporta un entorno multi-usuario 3D en línea.
- Compatible con espacios virtuales 3D de tamaño variable.
- Soporta múltiples clientes y protocolos de acceso.
- Soporta simulación física en tiempo real.
- Soporta clientes que crean contenidos 3D en tiempo real.
- Soporta lenguajes diferentes, incluyendo *LSL, C#* y *Visual Basic .NET.*
- Proporciona capacidad ilimitada para personalizar las aplicaciones virtuales.

2.5.3. Instalación de Opensim

Para poder instalar el servidor 3D de *Opensim*, es necesario descargar la última versión del mismo, que se puede encontrar en: http://opensimulator.org/wiki/Download. Una vez ejecutado, dentro de la carpeta "bin", ejecutar *OpenSim.exe* (Figura 2.7). En la figura 2.8 se muestra la ejecución del programa de instalación.

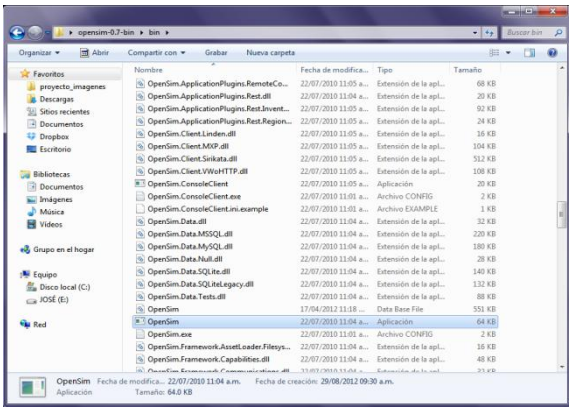


Figura 2.7 Carpeta descargada de *Opensim*

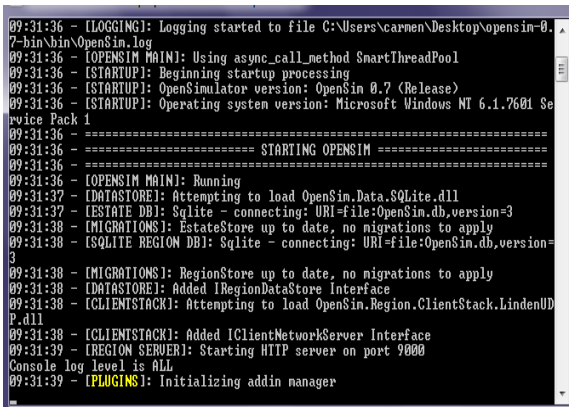


Figura 2.8 Programa de Instalación de *Opensim*.

Posteriormente, el programa de instalación solicitará ingresar los datos que identificarán al mundo virtual (figura 2.9); en la figura 2.10 se observa el ingreso de datos del administrador del mundo virtual.

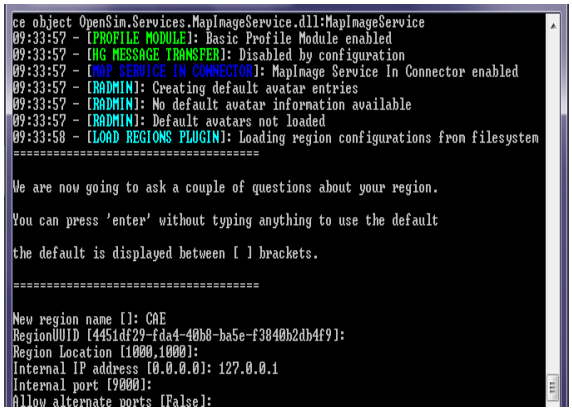


Figura 2.9 Datos del mundo virtual.

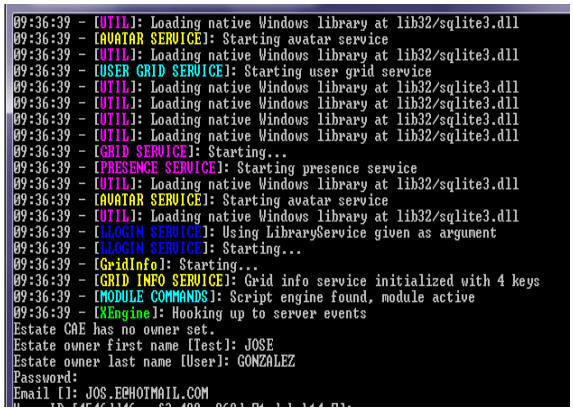


Figura 2.10 Datos del administrador.

El simulador se terminará de instalar al mostrar un estado similar al de la figura 2.11.

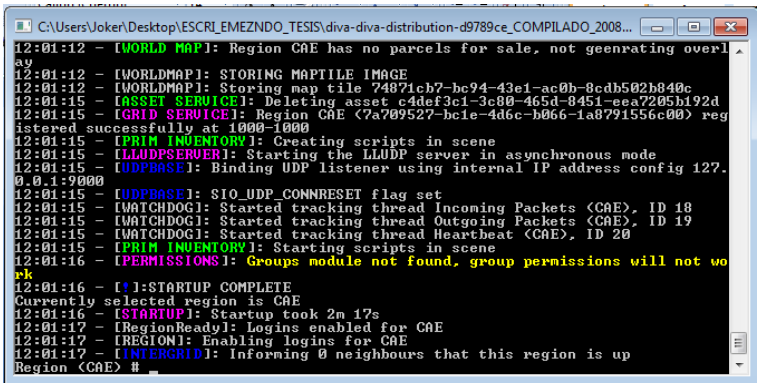


Figura 2.11 Instalación final de *Opensim*.

Con ello, queda instalado un mundo virtual, ahora es necesario un visor 3D para poder conocer e interactuar con el entorno virtual.

2.5.4. Visores

Para poder visualizar el mundo virtual una vez creado en *Opensim*, es necesario contar con visor 3D que pueda mostrar el mundo virtual creado e interactuar con el mismo. El equipo en donde se instalen los visores debe cubrir mínimo los requerimientos especificados en la tabla 2.1 [21].

Windows	Mac OS X	Linux
Conexión a internet: Cable o DSL. Sistema Operativo: XP, Vista o Windows 7. Procesador: 800 MHz Pentium III o Athlon, o	Conexión a internet: Cable o DSL. Sistema Operativo: Mac OS X 10.4.11 o posterior. Procesador: 1.5 GHz Intel based Mac, recomendado	Conexión a internet: Cable o DSL. Sistema Operativo: Se requiere un entorno de 32 bits. Si la distribución es de 64 bits, será recomendable instalar un entorno de

superior. Memoria RAM: 512MB o más. Resolución de pantalla: 1024x768 pixeles o superior. Tarjetas gráficas soportadas: • NVIDIA GeForce 6600 o superior. • ATI Radeon 9500 o superior. • ATI Radeon 8500, 9250 superior. • Intel 945 chipset.	Intel Core 2 Duo a 2GHz. Memoria RAM: 512MB o más. Resolución de pantalla: 1024x768 pixeles o superior. Tarjetas gráficas soportadas: • NVIDIA GeForce 2, GeForce 4. • ATI Radeon 9200 o superior.	compatibilidad con 32 bits. Procesador: 800 MHz Pentium III, Athlon, a 1.5GHz o superior. Memoria RAM: 512MB o más. Resolución de pantalla: 1024x768 pixeles o superior. Tarjetas gráficas soportadas: • NVIDIA GeForce 6600 o superior. • ATI Radeon 8500, 9250 superior.

Tabla 2.1 Requerimientos para instalación de visores en diferentes Sistemas Operativos.

A continuación se presentan los principales visores utilizados en el desarrollo de este proyecto.

2.5.4.1. Hippo Viewer

Creado por *MJM-Labs*, *Hippo Viewer* es un visor de SL modificado que puede ser instalado en los sistemas operativos *Linux, Mac (Macintosh Operating System) y Windows*. Está dirigido a los usuarios de *Opensim*, aunque también permite la conexión a SL. Permite construir hasta una altura de 10,000 metros y

prims con tamaño máximo de 256x256x256 metros (medida dentro del mundo virtual) [22].

Las características principales del visor son:

- Construir a una altura máxima de 10,000mts.
- Creación de *prims* con tamaño máximo de 256x256x256mts.
- Establecer transparencia hasta 100% en la ventana de edición.
- Ajustar el tamaño de perforación de los objetos hasta 0.01% para las formas como cilindro o *torus*.
- Copia de seguridad (exportación e importación de objetos, *scripts* y formas de avatar que se han creado).
- Mini mapa mejorado (mejor zoom, vista panorámica).

Al acceder por primera vez al visor, se presenta un avatar por *default* llamado *Ruth*, el sexo y características físicas es posible cambiarlas con los parámetros proporcionados por el visor, o bien importar ropas y apariencia desarrolladas en otras herramientas de diseño 3D externas. En la figura 2.12 se visualiza *Hippo Viewer* una vez instalado *Opensim* y conociendo por primera vez el mundo virtual.

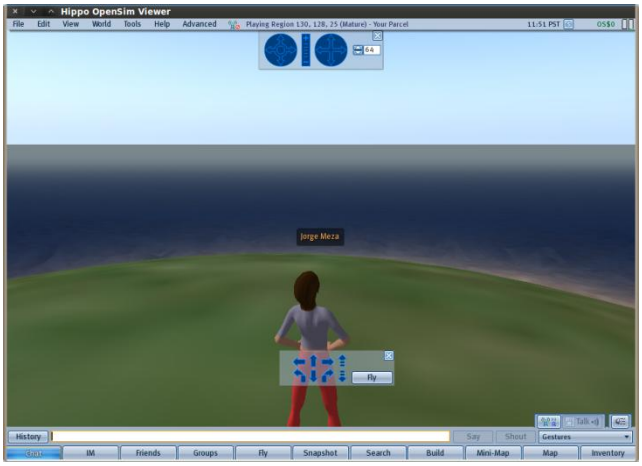


Figura 2.12 *Hippo Viewer.*

2.5.4.2. Second Life Viewer

Creado por *Linden Labs*, *Second Life Viewer* es un visor de *Opensim* y el visor oficial de SL, que puede ser instalado en los sistemas operativos *Linux, Mac OS y Windows* [23].

Las características principales del visor son:

- Compatible para *SL* y *Opensim*.
- Historial de chat.
- Lista de altavoz, para ver una lista de oradores en el canal de voz actual (*avatar* cercano o grupo).
- El botón *Speak* es un simple *on/off* y no soporta dual-modo *push-to-talk* (*PTT*).
- Para teletransportarse rápidamente a un *SLURL* o región, basta pegar el SLURL o escribir el nombre de la región en el campo de *SLURL.*
- Añadir puntos de referencia para su barra de favoritos.
- Historial de teletransportación, atrás y adelante: Haciendo clic y manteniendo los botones atrás y adelante para navegar a través del historial de teletransportación.

En la figura 2.13 se visualiza *Second Life Viewer* en la etapa de ingreso de datos para acceder al mundo virtual.

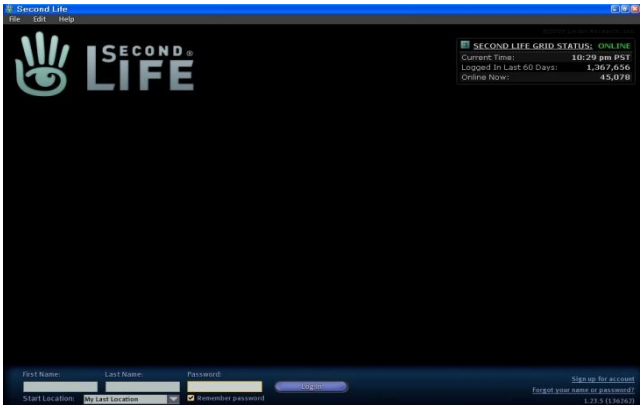


Figura 2.13 *Second Life Viewer.*

2.5.4.3. Imprudence Viewer

Imprudence Viewer, creado por *Imprudence Project Team*, es un visor de código abierto para SL y *Opensim* basado en mundos virtuales. El objetivo de *Imprudence* es la mejora de la interfaz de usuario y la usabilidad del espectador a través de la participación comunitaria. Puede ser instalado en los sistemas operativos: *Linux, Windows y Mac OS* [24].

El objetivo principal de *Imprudence* es simple: mejorar en gran medida la capacidad de uso del visor de SL. En particular, hay tres aspectos de usabilidad que pretende tratar:

- **Accesibilidad**: Mejorar la comodidad y facilidad de uso, especialmente para los nuevos usuarios.
- **Eficiencia***:* Mejorar la velocidad y la facilidad de las tareas comunes y flujos de trabajo.
- **Satisfacción**: Mejorar el efecto emocional del *software* en el usuario.

Para ello, *Imprudence* combina el carácter libre y distribuido de desarrollo de código abierto, promoviendo la mejora de la interfaz y uso del visor. Sus características son:

- Mini mapa mejorado (mejor zoom, vista panorámica, doble clic en el mapa para teletransportarse).
- Construcción de expresiones matemáticas.
- Búsqueda en inventario a través de parámetros como el creador o descripción.
- Inventario de filtro rápido (mostrar sólo los *notecards*, o sólo la ropa, entre otros).
- *Unread Count IM* (muestra cuántos mensajes instantáneos nuevos / no leídos se tienen).

- Cuadros de diálogo de confirmación para evitar accidentes.
- Restauración a la última posición.
- Modo "*Avatar Phantom*".
- Sentarse en cualquier lugar.

En la figura 2.14 se visualiza *Imprudence Viewer* una vez instalado *Opensim* y conociendo por primera vez el mundo virtual.

Figura 2.14 *Imprudence Viewer.*

2.5.5. Diva Distribution

Diva Distribution es una versión especial de *Opensim* (la versión libre del servidor de mundos virtuales compatible con *SL*) que adiciona una interfaz web (Figura 2.15) que permite la administración básica de un servidor desde un navegador, permitiendo acceder a información del mundo virtual [25], entre los datos que proporciona destacan los siguientes:

- Número de usuarios que están actualmente en el mundo virtual.
- Numero de regiones.

- Número total de usuarios.
- La cantidad de usuarios activos los últimos 30 días.
- Permite el registro de nuevos usuarios
- Obtener información del inventario.

Figura 2.15 Interfaz web de *Diva*.

Diva Distribution integra las herramientas para poseer tanto el mundo virtual, como la interfaz web, sin embargo puede ser integrada la parte web a un proyecto que se inició con solo el manejo de *Opensim.*

2.5.6. Programación

Para lograr la interacción de un usuario con el mundo virtual, es necesario programar los objetos con los que se va a interactuar. La interacción se puede realizar solo con el mundo virtual 3D programando los objetos con *LSL*, o interactuando con una página web (*Diva Distribution*).

2.5.6.1. Linden Scripting Language

LSL *(Linden Scripting Language*) es un lenguaje *scripting* que permite programar comportamientos entre objetos y *avatars* en un mundo virtual basado en SL [26].

En el mundo virtual, el comportamiento de los objetos se determina por uno o más *scripts* que se encuentran incrustados en el mismo. Un *script* se compone de variables, definiciones de funciones y uno o más estados designados. Cada estado contiene una descripción de cómo reaccionar a los eventos que ocurren mientras el programa está en ejecución. Los *scripts* pueden cambiar la mayoría de los aspectos del estado del objeto y comunicarse con otros objetos y agentes. Una vez que una secuencia de comandos se añade a un objeto se comienza a ejecutar inmediatamente.

Las instrucciones definidas por LSL son llamadas *primitivas*. Un objeto en SL puede representar por ejemplo una silla o una pared, o posiblemente algo invisible. Varios *scripts* pueden ser colocados dentro de un objeto, donde se pueden ejecutar simultáneamente.

Actualmente hay más de 300 bibliotecas con funciones disponibles. Se puede también definir funciones adicionales. LSL es un lenguaje fuertemente *tipado* que se compila en *bytecode* en tiempo de ejecución antes de la ejecución de una *máquina virtual (VM)* en uno de los servidores de *Linden Lab.*

La estructura de datos de *LSL* incluye números enteros, números de punto flotante, cadenas, llaves *(UUID),* vectores (utilizados para coordenadas 3D y RGB para expresar colores), rotaciones y listas. No hay arreglos. No incorpora el almacenamiento de datos persistentes, como un archivo o base de datos. Por otro lado, los guiones siguen funcionando incluso cuando el usuario no está conectado, y si un objeto se guarda (tener en inventario), y después se incorpora en el mundo virtual, todavía mantiene su estado anterior.

Al programar los objetos, existen ciertas acciones que requieren permisos:

- Tomar dinero de la cuenta del agente.
- Tomar control del agente.
- Iniciar o detener animaciones en agente.
- Conexión/desconexión del agente.
- Cambiar los vínculos.
- El seguimiento de la posición del agente de la cámara y la rotación.
- Controlar la cámara del agente.

Es de vital importancia cuidar los permisos que se colocan en el mundo virtual, para evitar que cualquier usuario pueda eliminar *scripts*, objetos e inclusive el terreno del mundo virtual.

2.5.6.2. Interacción web con el mundo virtual

Para inscribirse y formar parte del mundo virtual, es necesaria la interacción que se establece entre una página web (encargada del registro) y el proyecto realizado (mundo virtual en O*pensim*), esta comunicación permite sincronizar las aplicaciones para el uso adecuado del mundo virtual, ya que los visores solo muestran acceso al ambiente, sin embargo, una interacción web permitirá personalizar los datos que identifican al avatar dentro de la zona virtual. En éste apartado se muestran los lenguajes utilizados para lograr la interacción web con el espacio virtual creado.

2.5.6.2.1. HTML

HTML (*HyperText Markup Language*) o lenguaje de marcado de hipertexto, hace referencia al lenguaje de marcado predominante para la elaboración de páginas web que se utiliza para describir y traducir la estructura y la información en forma de texto, así como para complementar el texto con objetos tales como imágenes [27].

El *HTML* se escribe en forma de etiquetas, rodeadas por corchetes angulares (<,>). *HTML* también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un *script* (por ejemplo *JavaScript*), el cual puede afectar el comportamiento de navegadores web y otros procesadores de *HTML.*

La mayoría de los documentos tienen estructuras comunes (títulos, párrafos, listas, entre otros) que este lenguaje permite definir mediante *tags* (etiquetas). Cualquier cosa que no sea una *tag* es parte del documento mismo.

Este lenguaje no describe la apariencia de un documento sino que ofrece a cada plataforma la información para que le de formato según su capacidad y la de su navegador (tamaño de la pantalla, fuentes que tiene instaladas, entre otras). Por eso es importante diseñar los documentos con un contenido claro y bien estructurado que resulte fácil de leer y entender en cualquier navegador [28].

Normalmente se utiliza un programa editor para dar formato a los documentos, como *Frontpage, Dreamweaver, Amaya* u otro, de modo que no es necesario saber el código interno para crear una página.

2.5.6.2.2. C#

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por *Microsoft* como parte de su plataforma *.NET* [29], que después fue aprobado como un estándar por la ECMA-334 (*European Carton Makers Association)* e *ISO/IEC 23270* (*International Organization for Standardization/International Electrotechnical Commission*) *[29]. C#* es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Su sintaxis básica deriva de *C/C++* y utiliza el modelo de objetos de la plataforma *.NET*, similar al de *Java*, aunque incluye mejoras derivadas de otros lenguajes.

El nombre *C#* fue inspirado por la notación musical, donde *'#'* (sostenido, en inglés *sharp*) indica que la nota (*C* es la nota *do* en inglés) es un semitono más alta, sugiriendo que C# es superior a *C/C++.*

Aunque *C#* forma parte de la plataforma *.NET*, ésta es una *API*, mientras que *C#* es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma. Ya existe un compilador implementado que provee el marco *Mono -DotGNU*, el cual genera programas para distintas plataformas como *Windows, Unix, Android, iOS, Windows Phone, Mac OS y Linux* [30].

2.5.6.2.3. MySQL

MySQL es un sistema de gestión de bases de datos relacionales, *multihilo* y multiusuario de *Sun Microsystems* y este a su vez de *Oracle Corporation* desde abril de 2009, *MySQL* es software libre en un esquema de licenciamiento dual [31].

Para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en *ANSI C*.

Al contrario de proyectos como *Apache*, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, *MySQL* es patrocinado por una empresa privada, que posee el *copyright* de la mayor parte del código.

Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. *MySQL AB* fue fundado por David Axmark, Allan Larsson y Michael Widenius.

Entre los diversos lenguajes de programación que soportan el acceso a las bases de datos *MySQL* se encuentran: *C, C++, C#, Pascal, Delphi, Eiffel, Smalltalk, Lisp, Perl, PHP, Phyton*, entre otros. También existe una interfaz *ODBC*, llamada *MyODBC* que permite a cualquier lenguaje de programación que soporte *ODBC* comunicarse con las bases de datos *MySQL*.

Inicialmente, *MySQL* carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de ello, atrajo a los desarrolladores de páginas web con contenido dinámico, justamente por su simplicidad.

Poco a poco los elementos de los que carecía *MySQL* están siendo incorporados tanto por desarrollos internos, como por desarrolladores de software

libre. Entre las características disponibles en las últimas versiones se puede destacar:

- Disponibilidad en gran cantidad de plataformas y sistemas.
- Posibilidad de selección de mecanismos de almacenamiento que ofrecen diferente velocidad de operación, soporte físico, capacidad, distribución geográfica, transacciones, entre otras.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.

2.5.6.2.4. PHP

PHP *(Hypertext Pre-processor*), es un lenguaje interpretado del lado del servidor que surge dentro de la corriente denominada código abierto (*open source*). Se caracteriza por su potencia, versatilidad, robustez y modularidad. Al igual que ocurre con tecnologías similares, los programas son integrados directamente dentro del código HTML [32].

PHP es un lenguaje de programación interpretado o *framework* para *HTML*, diseñado originalmente para la creación de páginas web dinámicas. Se usa principalmente para la interpretación del lado del servidor (*server-side scripting*) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas *GTK+* o *Qt* [33].

Inicialmente llamado *PHP Tools* o *Personal Home Page Tools*, fue creado originalmente por Rasmus Lerdorf en 1994; sin embargo la implementación principal de PHP es producida ahora por *The PHP Group*. Publicado bajo la *PHP License*, *Free Software Foundation* considera esta licencia como software libre.

Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno.

El gran parecido que posee *PHP* con los lenguajes más comunes de programación estructurada, como *C y Perl,* permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones.

Aunque todo en su diseño está orientado a facilitar la creación de sitios web, es posible crear aplicaciones con una interfaz gráfica para el usuario, utilizando la extensión *PHP-Qt o PHP-GTK*. También puede ser usado desde la línea de comandos, de la misma manera como *Perl* o *Python* pueden hacerlo; a esta versión de *PHP* se la llama *PHP-CLI* (*Command Line Interface*).

Cuando el cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de *PHP.* Éste procesa el *script* solicitado que generará el contenido de manera dinámica (por ejemplo obteniendo información de una base de datos). El resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente. Mediante extensiones es también posible la generación de archivos *PDF*, *Flash*, así como imágenes en diferentes formatos.

PHP permite la conexión a diferentes tipos de servidores de bases de datos tales como *MySQL, PostgreSQL, Oracle, ODBC, DB2, Firebird, SQLite*, entre otros.

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como *Unix, Linux, Mac OS X* y *Microsoft Windows*.

PHP es una alternativa a las tecnologías de *Microsoft ASP* y *ASP.NET* (que utiliza *C#* y *Visual Basic .NET* como lenguajes). Aunque su creación y desarrollo se da en el ámbito de los sistemas libres, bajo la licencia *GNU*, existe además un entorno de desarrollo integrado comercial llamado *Zend Studio*.

Las características que lo distinguen, se describen a continuación:

- Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- Es considerado un lenguaje fácil de aprender, ya que en su desarrollo se simplificaron distintas especificaciones, como es el caso de la definición de las variables *primitivas*.
- El código fuente escrito en *PHP* es invisible al navegador web y al cliente, ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado *HTML* al navegador, esto hace que la programación en *PHP* sea segura y confiable.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con *MySQL y PostgreSQL.*
- Posee una amplia documentación en su sitio web oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones (desde *PHP 5*).
- Si bien *PHP* no obliga a quien lo usa a seguir una determinada metodología a la hora de programar, aun haciéndolo, el programador puede aplicar en su trabajo cualquier técnica de programación o de desarrollo que le permita escribir código ordenado, estructurado y manejable [32].

Capítulo 3. Análisis del Sistema de Mentoría BUAP

La creación del proyecto “Sistema de Mentoría BUAP. Un Ambiente Virtual 3D”, inició con la propuesta de innovar la prestación de servicios hacia los alumnos y catedráticos de la BUAP; el análisis del trabajo realizado se detalla en el presente capítulo, presentando el diagrama de casos de uso, sus especificaciones y la construcción de escenarios.

3.1. Diagrama de casos de uso para utilizar el Sistema de Mentoría BUAP

Para poder ejemplificar el uso del Sistema de Mentoría BUAP, se presenta en la figura 3.1 el diagrama de casos de uso general del proyecto, estableciendo las funciones principales que se llevarán a cabo en el sistema.

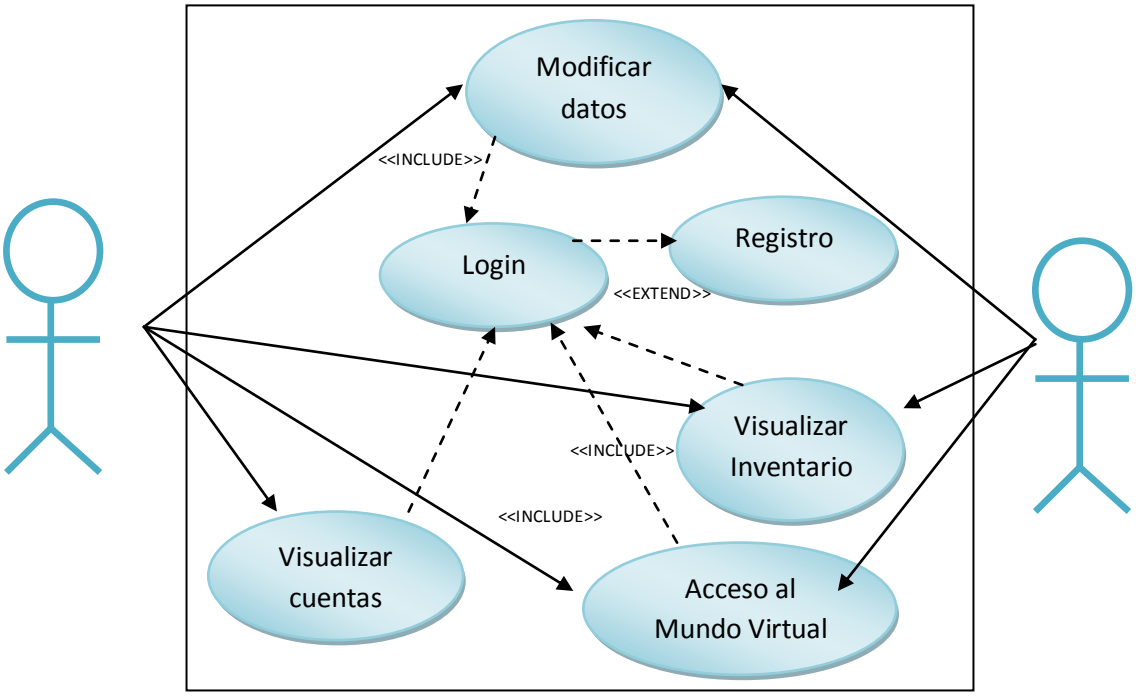


Figura 3.1 Diagrama de casos de uso del Sistema de Mentoría BUAP.

3.2. Especificación de casos de uso

En la tabla 3.1 se encuentra la especificación del caso de uso *Registro* del sistema.

Nombre del caso de uso: Registro.
1. Descripción breve A través de este caso, el sistema permitirá al usuario crear una cuenta para acceder al mundo virtual.
2. Flujo de eventos **2.1 Precondiciones** El cliente tiene que ser alumno o catedrático de la BUAP. **2.2 Flujo Principal** 2.2.1 El sistema muestra una página para el ingreso de datos del alumno/catedrático. 2.2.2 El alumno/catedrático ingresa los datos que lo identifican. 2.2.3 El sistema muestra un mensaje de registro exitoso o no exitoso. 2.2.4 Si la validación es correcta se ingresa en la BD. 2.2.5 El sistema le permite acceder al sistema virtual y página web 2.2.6 Termina el caso de uso. **2.3 Flujos alternos** 2.3.1 Se regresa al flujo 2.2.2 si el sistema detecta errores en el ingreso de datos.

Tabla 3.1 Especificación de casos de uso “Registro”.

En la tabla 3.2 se encuentra la especificación del caso de uso *Login* del sistema.

Nombre del caso de uso: *Login.*
1. Descripción breve Con este caso de uso el sistema evalúa si el usuario ya fue registrado, y si es así darle acceso a su cuenta.
2. Flujo de eventos **2.1 Precondiciones** El cliente ya debe estar registrado. **2.2 Flujo Principal** 2.2.1 El sistema muestra una página para el ingreso de datos del alumno/catedrático. 2.2.2 El alumno/catedrático ingresa los datos que lo identifican. 2.2.3 El sistema muestra un mensaje de *login* exitoso o no exitoso. 2.2.4 Si la validación es correcta se otorga el acceso a su cuenta. 2.2.5 El sistema le permite acceder a la consulta y modificación de su cuenta. 2.2.6 Termina el caso de uso. **2.3 Flujos alternos** 2.3.1 Se regresa al flujo 2.2.2 si el sistema califica como inválidos los datos brindados por el usuario.

Tabla 3.2 Especificación de casos de uso "*Login".*

En la tabla 3.3 se encuentra la especificación del caso de uso *Modificar datos* del sistema.

Nombre del caso de uso: Modificar datos.
1. Descripción breve Con este caso de uso, el sistema permite al usuario modificar su cuenta, alterar campos como nombre, correo, entre otros.
2. Flujo de eventos **2.1 Precondiciones** El cliente ya debe estar inscrito. **2.2 Flujo Principal** 2.2.1 El sistema muestra una página para modificar los datos proporcionados cuando el usuario se registró: “Editar cuenta”. 2.2.2 El alumno/catedrático ingresa los datos que desea modificar. 2.2.3 El sistema muestra un mensaje modificación correcta/incorrecta. 2.2.4 Si la validación es correcta se altera en la BD. 2.2.5 El sistema le permite acceder a la consulta y modificación de su cuenta con nuevos parámetros. 2.2.6 Termina el caso de uso. **2.3 Flujos alternos** 2.3.1 Se regresa al flujo 2.2.2 si el sistema califica como inválidos los datos brindados por el usuario.

Tabla 3.3 Especificación de casos de uso “Modificar datos”.

En la tabla 3.4 se encuentra la especificación del caso de uso *Visualizar inventario* del sistema.

Nombre del caso de uso: Visualizar inventario.
1. Descripción breve Con este caso de uso, el sistema permite al usuario visualizar objetos creados por él, scripts realizados, entre otros.
2. Flujo de eventos **2.1 Precondiciones** El cliente ya debe estar inscrito y logueado. **2.2 Flujo Principal** 2.2.1 El sistema muestra una página para visualizar los objetos que se encuentran a su disposición: “Inventario”. 2.2.2 El sistema permitirá la selección del contenido propiedad del usuario. 2.2.3 El alumno/catedrático podrá eliminar o alterar el orden de muestreo de la información. 2.2.4 El sistema evalúa que el objeto aun exista, para controlar la sincronización con el mundo virtual, indicando exitoso/no exitoso. 2.2.5 El sistema guardará los cambios efectuados en la BD. 2.2.6 Termina el caso de uso. **2.3 Flujos alternos** 2.3.1 Se regresa al flujo 2.2.2 si el sistema detecta que el objeto ya fue alterado o eliminado en el mundo virtual.

Tabla 3.4 Especificación de casos de uso “Visualizar inventario”.

En la tabla 3.5 se encuentra la especificación del caso de uso *Visualizar cuentas* del sistema.

Nombre del caso de uso: Visualizar cuentas.
1. Descripción breve El administrador podrá visualizar las cuentas de usuarios que se encuentran en el mundo virtual.
2. Flujo de eventos **2.1 Precondiciones** El administrador ya debe estar logueado. **2.2 Flujo Principal** 2.2.1 El sistema muestra una página para visualizar las cuentas y regiones creadas por el mismo. 2.2.2 El sistema permitirá la visualización de información actual del mundo virtual. 2.2.3 Termina el caso de uso.

Tabla 3.5 Especificación de casos de uso “Visualizar cuentas”.

En la tabla 3.6 se encuentra la especificación del caso de uso *Acceso al mundo virtual* del sistema.

Nombre del caso de uso: Acceso al mundo virtual.
1. Descripción breve Con este caso de uso, el visor permitirá al usuario entrar al mundo virtual e interactuar con el mismo.
2. Flujo de eventos **2.1 Precondiciones** El cliente ya debe estar registrado y poseer el visor para acceder al mundo virtual. **2.2 Flujo Principal** 2.2.1 El visor muestra la página principal de acceso. 2.2.2 El alumno/catedrático ingresa los datos de registro. 2.2.3 El visor muestra un mensaje de acceso correcto/incorrecto. 2.2.4 Si la validación es correcta se ingresa al mundo virtual. 2.2.5 El sistema le permite moverse en el mundo virtual, interactuar y crear objetos en el mismo. 2.2.6 Termina el caso de uso. **2.3 Flujos alternos** 2.3.1 Se regresa al flujo 2.2.2 si el visor califica como inválidos los datos brindados por el usuario.

Tabla 3.6 Especificación de casos de uso “Acceso al mundo virtual”.

3.3. Escenarios

En la tabla 3.7 se describe el escenario *Registro* del sistema.

Escenario: Registro.
• **Escenarios primarios** ➢ Omar Marín es alumno de la BUAP, y desea tener acceso al mundo virtual de Mentorías BUAP. ➢ Ingresa a la página inicial del Sistema de Mentoría. ➢ Accede a la opción de registro. ➢ Ingresa los datos solicitados por el sistema. ➢ El sistema le muestra un registro exitoso y le brinda su nombre de usuario para acceder al mundo ➢ Termina el caso de uso.
• **Escenarios Secundarios.** ➢ Omar accede a su cuenta vía web y consulta su inventario. ➢ Omar modifica sus datos de registro. ➢ Omar descarga e instala el visor para acceder al mundo virtual. ➢ Omar accede al mundo virtual si ya posee su visor. ➢ Omar cierra sesión.

Tabla 3.7 Escenario del caso de uso “Registro”.

En la tabla 3.8 se describe el escenario *Login* del sistema.

Escenario: *Login.*
• **Escenarios primarios** ➢ Michel es alumno de la BUAP, y se ha registrado en el Sistema de Mentoría BUAP. ➢ Ingresa a la página inicial del Sistema de Mentoría. ➢ Accede a la opción de *login.* ➢ Ingresa los datos solicitados por el sistema. ➢ El sistema le muestra un registro exitoso y le brinda acceso a los datos de su cuenta. ➢ Termina el caso de uso.
• **Escenarios Secundarios.** ➢ Michel accede a su cuenta vía web y consulta su inventario. ➢ Michel modifica sus datos de registro. ➢ Michel descarga e instala el visor para acceder al mundo virtual. ➢ Michel accede al mundo virtual si ya posee su visor. ➢ Michel cierra sesión.

Tabla 3.8 Escenario del caso de uso "*Login".*

En la tabla 3.9 se describe el escenario *Modificar Datos* del sistema.

Escenario: Modificar Datos.
• **Escenarios primarios** ➢ Miguel es alumno de la BUAP, y se ha registrado en el Sistema de Mentoría BUAP, pero considera que su contraseña no es segura. ➢ Ingresa a la página inicial del Sistema de Mentoría. ➢ Accede a la sección de *login.* ➢ Ingresa los datos solicitados por el sistema. ➢ El sistema le muestra un acceso exitoso y le brinda acceso a los datos de su cuenta. Miguel accede a la opción “Editar cuenta”. ➢ Miguel ingresa su contraseña anterior y coloca la nueva. ➢ El sistema lo evalúa e indica correcto. ➢ Termina el caso de uso.
• **Escenarios Secundarios.** ➢ Miguel accede a su cuenta vía web y consulta su inventario. ➢ Miguel vuelve a modificar sus datos de registro. ➢ Miguel descarga e instala el visor para acceder al mundo virtual. ➢ Miguel accede al mundo virtual si ya posee su visor. ➢ Miguel cierra sesión.

Tabla 3.9 Escenario del caso de uso “Modificar datos”.

En la tabla 3.10 se describe el escenario *Visualizar inventario* del sistema.

Escenario: Visualizar inventario.
• **Escenarios primarios** ➢ Karen es alumna de la BUAP, y se ha registrado en el Sistema de Mentoría BUAP, y quiere saber cuántos objetos ha creado. ➢ Ingresa a la página inicial del Sistema de Mentoría. ➢ Accede a la sección de *login.* ➢ Ingresa los datos solicitados por el sistema. ➢ El sistema le muestra un registro exitoso y le brinda acceso a los datos de su cuenta. ➢ Karen verifica su inventario. ➢ Termina el caso de uso.
• **Escenarios Secundarios.** ➢ Karen accede a su cuenta vía web y consulta nuevamente su inventario. ➢ Karen modifica sus datos de registro. ➢ Karen descarga e instala el visor para acceder al mundo virtual. ➢ Karen accede al mundo virtual si ya posee su visor. ➢ Karen cierra sesión.

Tabla 3.10 Escenario del caso de uso “Visualizar inventario”.

En la tabla 3.11 se describe el escenario *Visualizar cuentas* del sistema.

Escenario: Visualizar cuentas.
• **Escenarios primarios** ➢ José es administrador del Sistema de Mentoría BUAP, y desea saber cuántos usuarios hay actualmente en el mundo virtual. ➢ Ingresa a la página inicial del Sistema de Mentoría. ➢ Accede a la sección de *login.* ➢ Ingresa los datos solicitados por el sistema. ➢ El sistema le muestra un registro exitoso y le brinda acceso a los datos de su cuenta de administrador. ➢ José verifica las cuentas de usuario. ➢ Termina el caso de uso.
• **Escenarios Secundarios.** ➢ José accede a su cuenta vía web y consulta nuevamente las cuentas. ➢ José modifica sus datos de registro. ➢ José descarga e instala el visor para acceder al mundo virtual. ➢ José accede al mundo virtual si ya posee su visor. ➢ José cierra sesión.

Tabla 3.11 Escenario del caso de uso “Visualizar cuentas”.

En la tabla 3.12 se describe el escenario *Acceso al mundo virtual* del sistema.

Escenario: Acceso al mundo virtual.
• **Escenarios primarios** ➢ Mariana es alumna de la BUAP y ya se ha registrado en el sistema de Mentorías BUAP. ➢ Mariana ingresa a la página inicial del Sistema de Mentoría y descarga la aplicación *Imprudence Viewer.* ➢ Mariana configura su visor. ➢ Mariana Ingresa los datos solicitados por el visor. ➢ El sistema le muestra un registro exitoso y le brinda acceso al mundo virtual**.** ➢ Termina el caso de uso.
• **Escenarios Secundarios.** ➢ Mariana accede a su cuenta vía web y consulta su inventario. ➢ Mariana modifica sus datos de registro. ➢ Mariana crea objetos en el mundo virtual. ➢ Mariana cierra sesión.

Tabla 3.12 Escenario del caso de uso “Acceso al mundo virtual”.

Capítulo 4. Diseño y desarrollo del Ambiente Virtual 3D para el Sistema de Mentoría BUAP.

El diseño del proyecto de Mentoría BUAP inició con la propuesta de albergar cuatro islas en el mundo virtual que serán utilizadas por los alumnos y catedráticos para poder interactuar y solicitar servicios. En cada una de ellas, se instalaron los inmuebles necesarios para poder ofrecer los trámites e información que brindan los cuatro programas que atenderán el espacio virtual:

- Programa Educando para la Salud.
- Programa de Apoyo Académico a Estudiantes Indígenas (PAAEI).
- Programa de Becas y Apoyo para la Permanencia (BAP).
- Programa de Movilidad Estudiantil.

Para la creación del mundo virtual, se optó por implementar el proyecto con *Opensim*, añadiendo la interfaz web con los módulos implementados por *DIVA Distribution*.

Una vez instalado *Opensim* (Capítulo 2.5.3), es necesario contar con un visor 3D que nos permita interactuar con el mundo virtual, para ello se utilizaron los visores *Imprudence* (diseño del mundo virtual) y *Second Life* (programación de los objetos a interactuar) por su fácil manejo y capacidad de importar archivos 3D creados en otras herramientas de diseño tridimensionales, además de ser los visores que poseen mayor información y documentación de su uso.

4.1. Acceso al Mundo Virtual creado por Opensim

Para acceder al mundo virtual, se optó iniciar con la utilización del visor *Imprudence*, ya que es de fácil uso y consume menos recursos. En la figura 4.1 se puede visualizar el proceso final de la instalación y ejecución del visor *(*descargado en: http://wiki.kokuaviewer.org/wiki/Imprudence:Downloads). Es importante configurar el acceso al mundo virtual creado en *Opensim,* para ello, el visor cuenta con una opción para establecer los datos del mundo a ingresar llamada *GridManager* (Figura 4.2), donde se siguen los siguientes pasos para establecer el acceso al mundo creado.

I. Pulsar "*Add New Grid".*
II. GridName: "Sistema de Mentoría BUAP".
III. Login URI: <*IP* del servidor>:9000.
IV. Login Page: <*IP* del servidor>:9000/wifi
V. Website: <*IP* del servidor>:9000/wifi

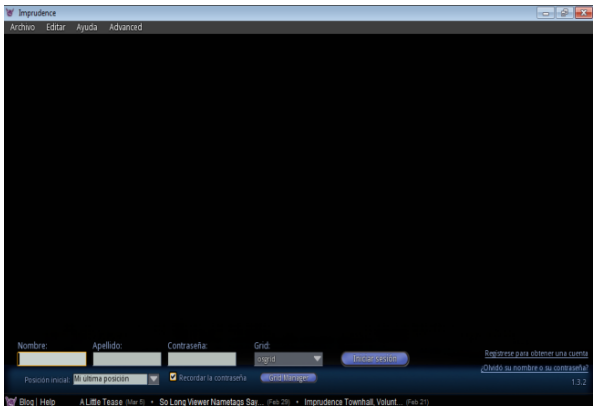


Figura 4.1 Ejecución de *Imprudence Viewer*.

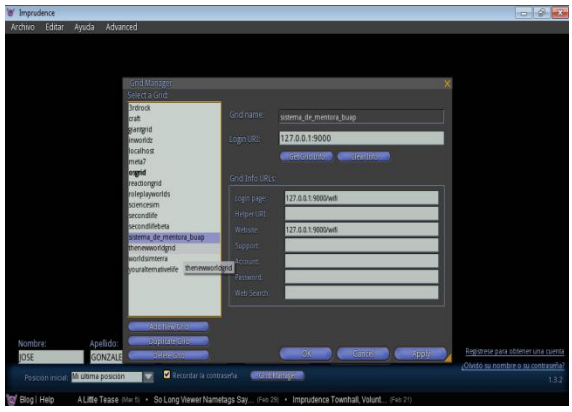
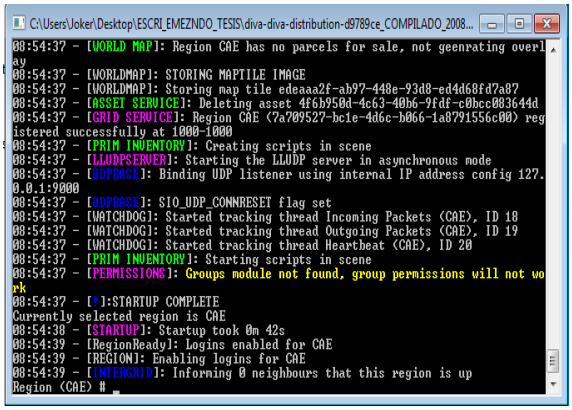


Figura 4.2 Configuración de *GridManager.*

Ahora, se establece el acceso al mundo virtual creado (Capitulo 2.5.3), ejecutando el proyecto implantado en *Opensim* (Figura 4.3) y con el *nombre,*

apellido y *password* configurados en la instalación (Figura 4.4), se accede desde nuestro visor al mundo virtual (Figura 4.5).



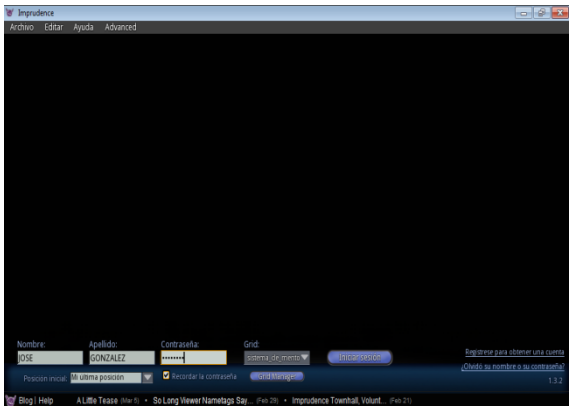


Figura 4.3 Ejecución de *Opensim.*

Figura 4.4 Ingreso de datos configurados en *Opensim.*

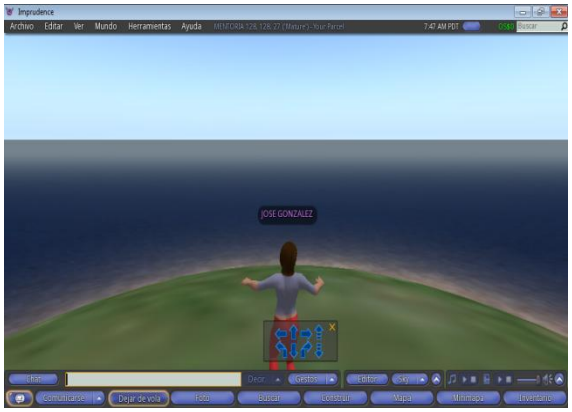


Figura 4.5 Acceso al terreno virtual creado en *Opensim* por medio del visor *Imprudence.*

Opensim nos muestra un cuerpo automático, mismo que puede ser configurable desde el visor *Imprudence* a través del Menú *Editar/Apariencia*; en la figura 4.6 se pude mostrar las opciones editables del avatar dentro del mundo virtual. En la figura 4.7 se puede visualizar algunos de los cambios realizados al avatar *Ruth*, que es el que aparece por *default* a la hora de crear el mundo.

Figura 4.6 Avatar por default: *Ruth.*

Figura 4.7 Avatar configurado.

En la figura 4.8 se muestra el avatar finalizado de editar y la isla o terreno que le fue asignado por *Opensim.*

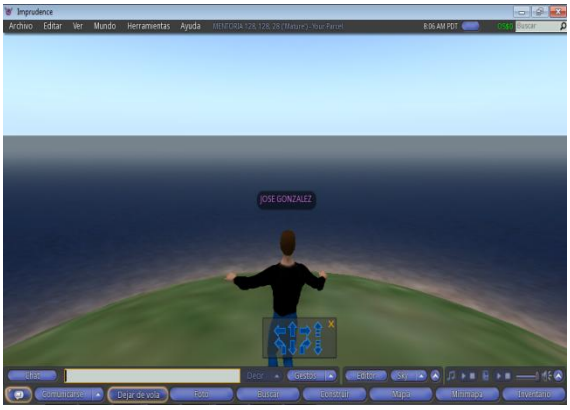


Figura 4.8 Avatar diseñado y mundo virtual.

4.2. Configuración Opensim con MySQL

Opensim, de manera automática respalda todos los cambios que se realizan en el mundo virtual en una base de datos implementada en *SQLite*. *Opensim* ofrece la flexibilidad de establecer otras alternativas para respaldar los datos del mundo virtual, por ello implementa instrucciones para establecer el almacenamiento en *MySQL, SQL* u otro gestor de bases de datos.

El Sistema de Mentoría *BUAP*, es implementado con *MySQL*, por su flexibilidad, sencillez, documentación y por ser de carácter libre.

Para poder realizar el cambio y establecer a *MySQL* como el gestor de base de datos del proyecto, será necesario alterar la conexión que se encuentra en el archivo: *Opensim/bin/config-include/StandaloneCommon.ini.*

Dentro del archivo *StandaloneCommon.ini*, se podrá visualizar la linea que establece la conexiòn con la base de datos, utilizando *SQLite:*

```
; SQLite
Include-Storage = "config-include/storage/SQLiteStandalone.ini";
```

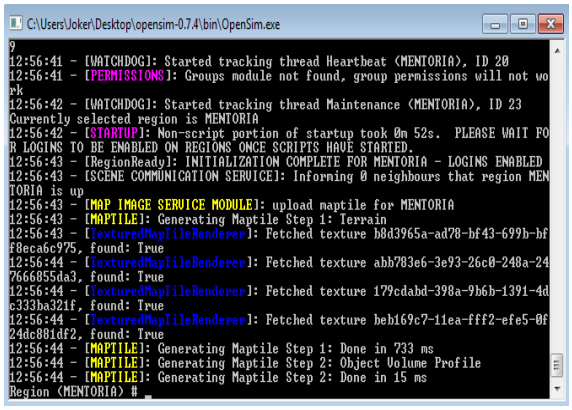
Una vez localizada, se podria borrar o comentar(anteponiendo ";"), y activar la linea que aparece a continuaciòn, donde se establece la conexión con el gestor de base de datos MySQL.

```
; SQLite
;Include-Storage = "config-include/storage/SQLiteStandalone.ini";

;MySql
StorageProvider = "OpenSim.Data.MySQL.dll"
ConnectionString = "Data Source=<IP DEL SERVIDOR>;
Database=opensim** User ID=<USUARIO>**; Password=<PASSWORD>; Old
Guids=true;"

**Es de importancia haber creado primero la base de datos Opensim,
así como el usuario que posee permisos a ella.
```

En el archivo *StandaloneCommon.ini* tambièn es importante cambiar en todo el documento la direcciòn que viene por default: *"127.0.0.1"* con la IP(Internal Protocol) que será utilizada para el servidor. Después de ello, se vuelve a ejecutar el proyecto (figura 4.9). Con un total de 27 tablas (figura 4.10), *Opensim* crea el contenido de la base de datos *opensim* necesaria para respaldar la informaciòn vital del mundo virtual 3D.



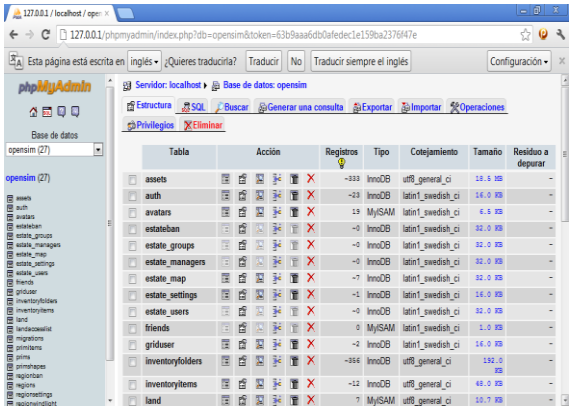


Figura 4.9 Creando BD MySQL en *Opensim.*

Figura 4.10 Contenido *MySQL* generado por *Opensim.*

4.3. Configuración del terreno

El Sistema de Mentoría BUAP, albergará cuatro instituciones que servirán de apoyo a los alumnos, para ello se propuso la creación de cuatro islas en el mundo virtual. En la figura 4.11 se puede observar que el terreno que nos proporciona *Opensim* es reducido y se necesita suficiente espacio para albergar cuatro instituciones.

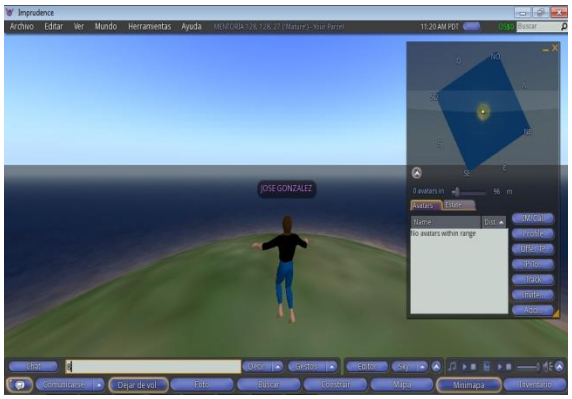


Figura 4.11 Isla proporcionada por *Opensim.*

Para la creación de las cuatro islas es necesario modificar el archivo *Regions.ini*, ubicado en la ruta *Opensim/bin/Regions*. El contenido se muestra a continuación:

```
[MENTORIA]
RegionUUID = 3ed95bbb-cb10-4b76-9be9-b3e84a4efdc8
Location = 1000,1000
InternalAddress = <IP del servidor>
InternalPort = 9000
AllowAlternatePorts = False
ExternalHostName = "<IP del servidor>"
```

Para poder crear otras tres islas, se añaden los datos de las mismas. En el archivo *Regions.ini* se anexan las siguientes líneas:

```
[MENTORIA 2]
RegionUUID = 5c1d5c55-c7ec-4cc5-9afe-f9737abf59eb
Location = 1000,1001
InternalAddress = <IP del servidor>
InternalPort = 9001
AllowAlternatePorts = False
ExternalHostName = "<IP del servidor>"

[MENTORIA 3]
RegionUUID = 4a6e648c-8b50-406c-87a1-57f7eb443bf6
Location = 1001,1000
InternalAddress = <IP del servidor>
InternalPort = 9002
AllowAlternatePorts = False
ExternalHostName = "<IP del servidor>"

[MENTORIA 4]
RegionUUID = 0a2f2d4c-89cb-4b6b-bbe9-3497daf58b12
Location = 1001,1001
InternalAddress = <IP del servidor>
InternalPort = 9003
AllowAlternatePorts = False
ExternalHostName = "<IP del servidor>"
```

Los Atributos o parámetros de las líneas agregadas, se mencionan en la tabla 4.1 [34]:

Atributo	Descripción
[NOMBRE]	Nombre de la región.
RegionUUID	El identificador único de la región.
Location	La ubicación (x, y) de la región en el mundo virtual.
InternalAddress	Se puede establecer a una IP específica.
InternalPort	Puerto IP para todas las conexiones de cliente entrantes.
AllowAlternatePorts	No se utiliza. Dejar siempre False.
ExternalHostname	Dirección IP externa del *router* o *FQDN.*

Tabla 4.1 Parámetros utilizados para poder agregar regiones a un proyecto.

Las nuevas regiones se nombran MENTORIA (2, 3 y 4), que poseen un *UUID* diferente. Tienen una ubicación diferente (por ejemplo: 1000,1001) y un *InternalPort* diferente (por ejemplo: 9001). Aparte de eso, los detalles son los mismos.

Para crear un nuevo *UUID* se puede modificar uno ya existente a mano (por ejemplo, cambiando un carácter por lo que el nuevo está en el rango [0-9] [A-F]).

Hay algunos datos opcionales que pueden cambiar las propiedades de las regiones, estos se muestran en la tabla 4.2:

Atributo	Descripción
Maxagents	El número máximo de agentes que pueden estar en la en la región en cualquier momento dado. El valor por defecto es 100.

MaxPrims	El número máximo de *prims* de la región. El límite máximo que puede ser mostrado es 45000.
PhysicalPrimMax	Las dimensiones máximas de un *prim*. Este es un número único que se aplica a las coordenadas X, Y y Z. Esto afectará el cambio de tamaño de *prims* existentes. El valor predeterminado es 10.
NonphysicalPrimMax	Las dimensiones máximas para un prim no físico. Este es un número único que se aplica a las coordenadas X, Y y Z. Esto afectará el cambio de tamaño de *prims* existentes. El valor predeterminado es 256.
ClampPrimSize	Si es *true,* cuando un usuario intente crear un objeto que tenga alguna dimensión mayor que la NonphysicalPrimMax, entonces esa dimensión se reduce a la establecida por NonphysicalPrimMax. El valor predeterminado es false.

Tabla 4.2 Propiedades de las regiones.

El resultado de agregar las líneas propuestas, es la creación de tres islas adicionales, cuatro con la principal, las cuales albergarán las instituciones del Sistema de Mentoría. En la figura 4.12 se puede visualizar el cambio realizado, el *mini mapa*, muestra las cuatro islas creadas, así como la ubicación del *avatar* entre las mismas.

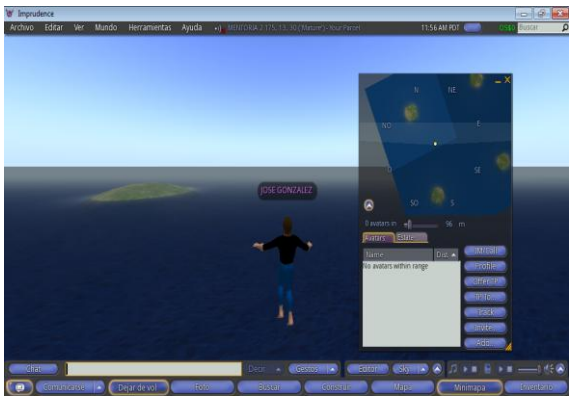
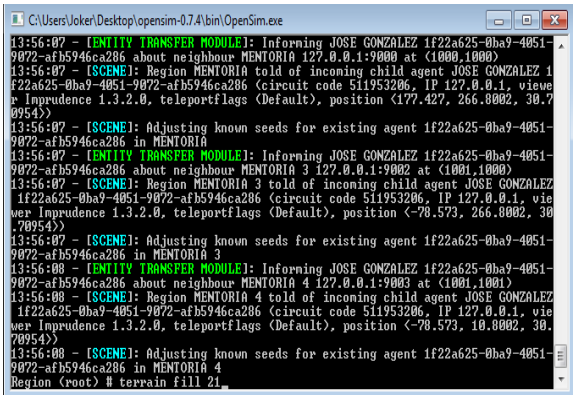


Figura 4.12 Cuatros islas creadas en *Opensim.*

Para poder establecer instituciones en las islas, es necesario contar con un terreno amplio; para tener mayor control del espacio virtual y con el objetivo de ampliar el espacio a construir, se utiliza la instrucción: *terrain fill 21*.

La instrucción *terrain* hace el suelo plano y su parámetro *21*, establece la altura del terreno, que es al nivel del mar, se pude aumentar o reducir, de acuerdo a las necesidades. El código se incrusta en el período de ejecución de *Opensim* (Figura 4.13) y los cambios que genera, se pueden visualizar en la figura 4.14:



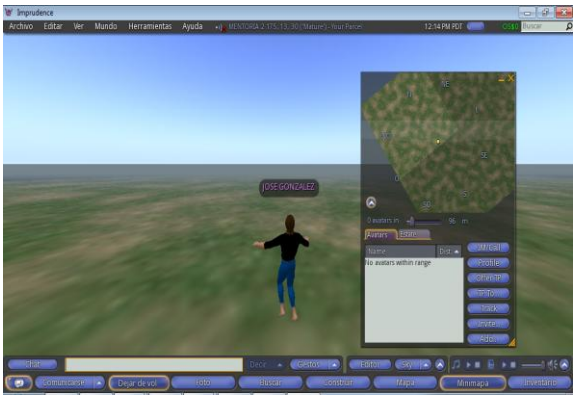


Figura 4.13 Ejecución *Terrain fill 21.*

Figura 4.14 Aplanamiento del terreno.

El resultado es un terreno plano dividido en cuatro regiones. Para poder establecer nuevamente cuatro islas, se empleó la instrucción *construir,* que se encuentra en el Menú *Ver/Construir* (figura 4.15) del visor; utilizando la opción *terreno/bajar* y estableciendo la opción *fuerza* al máximo, se empezó a reducir de nivel el terreno para personalizar la isla de cada institución (figura 4.16).

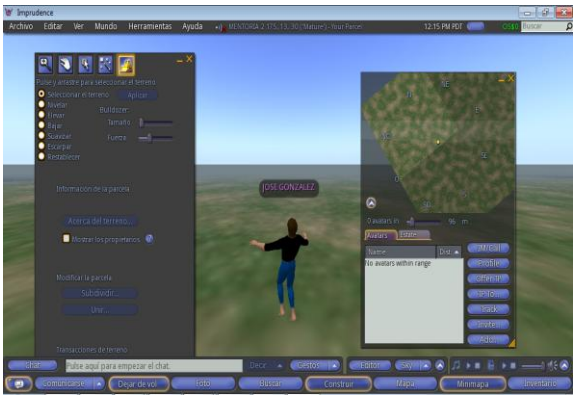


Figura 4.15 Herramienta *Construir*,

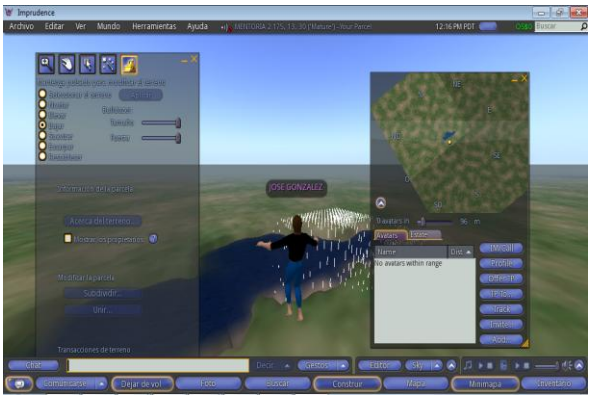


Figura 4.16 Uso de la **h**erramienta *bajar* terreno.

El resultado final, se puede visualizar en la Figura 4.17, donde se presentan las cuatro islas que formarán parte del Sistema de Mentoría BUAP.

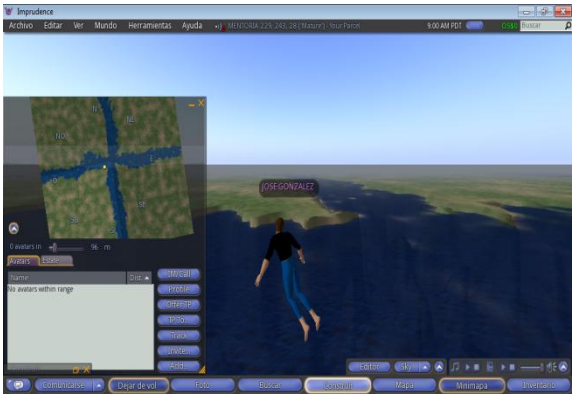


Figura 4.17 Modificación del terreno en cuatro islas.

4.4. Creación y uso de edificios

La creación del terreno da pauta a la construcción de los edificios y elementos que formarán parte del proyecto, la creación puede realizarse de manera manual, usando *prims* y herramientas del visor para poder implementarlos, o bien, importar edificios ya realizados al proyecto.

4.4.1. Creación de edificios utilizando prims

Un *prim* es un objeto básico 3D creado en el ambiente virtual, son llamados así porque son formas básicas que sirven para la creación de objetos complicados, cuando varios de ellos conforman un cuerpo complejo se les denomina *sculpted prim* o *sculptie.*

La creación del mundo virtual inició con el desarrollo de un edificio utilizando *prims*. El visor pone a nuestra disposición varias figuras iniciales 3D como lo son: cubo, prisma, pirámide, tetraedro, cilindro, semicilindro, cono, semicono, esfera, semiesfera, toroide o torus, tubo, cono truncado, árbol e hierba (Figura 4.18). En la figura 4.19 se pude visualizar la inserción del *prim* inicial del proyecto.

Figura 4.18 *Prims* para construcción 3D.

Figura 4.19 *Prim* Cubo.

Al ingresar un nuevo objeto, el visor proporciona los elementos necesarios para lograr la edición del mismo, los *prims* podrán cambiar de posición, girar, estirar, editar por cara, enlazar, desenlazar con otros objetos, entre otras opciones (figura 4.20). Los *prims*, automáticamente detectan el usuario del mundo que los creó, los permisos principales se pueden visualizar en la ficha *General*, como se muestra en la figura 4.21.

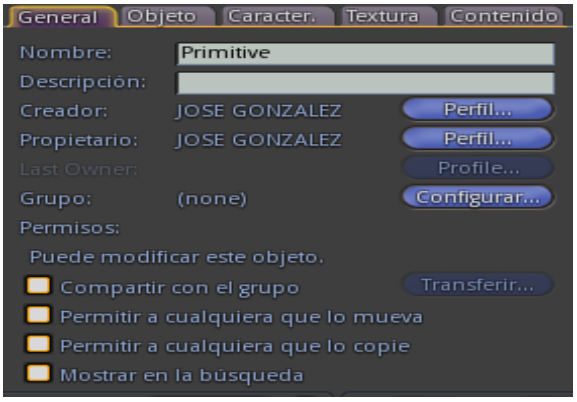
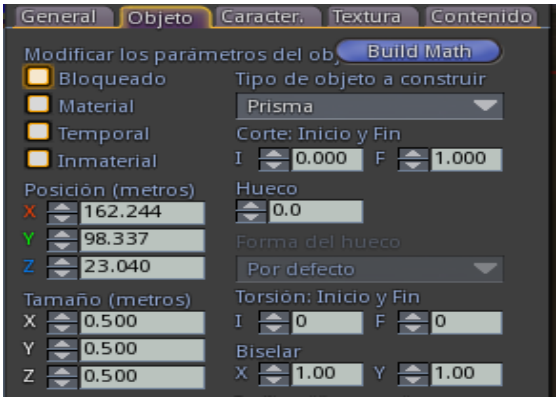


Figura 4.20 Herramientas para modificación *de prims.* **Figura 4.21** Ficha *General*.

En la ficha de edición *Objeto*, se puede modificar la forma física del *prim* con parámetros complejos tales como: posición exacta, cortes, torsión, hueco, estado material, entre otras opciones (figura 4.22). En la ficha *Caracter*, se pueden alterar las propiedades de flexibilidad y luz que puede ejercer el ambiente físico en el objeto (figura 4.23).



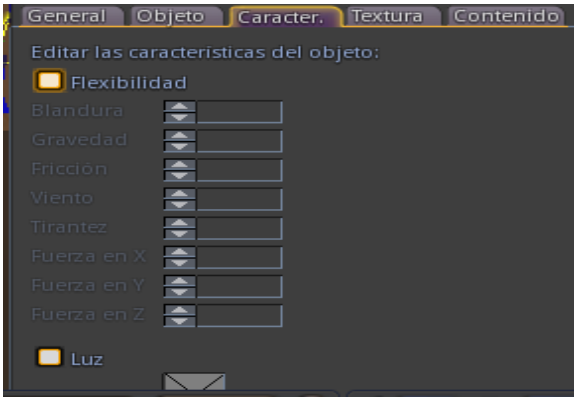


Figura 4.22 Ficha *Objeto.*

Figura 4.23 Ficha *Caracter.*

La edición de los objetos o *prims* creados incluyen propiedades de textura, color, transparencia, resplandor, entre otros parámetros que se pueden visualizar en ficha *Textura* (4.24). Por último, es posible controlar el comportamiento del *prim*; para ello, los objetos se programan con *scripts* realizados en lenguaje LSL, que pueden ser creados e incrustados en la ficha Contenido (figura 4.25).

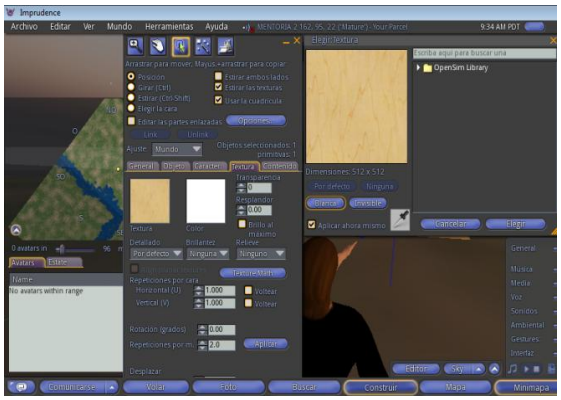


Figura 4.24 Ficha *Textura.*

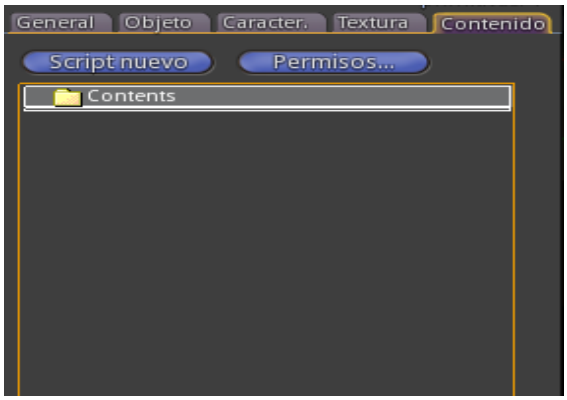


Figura 4.25 Ficha *Contenido.*

El uso de estas herramientas principales contribuyó a la construcción del primer edificio con el uso exclusivo de *prims*. La construcción de los elementos del primer edificio, inició con la creación de paredes (figura 4.26), y edición del mismo con ayuda de parámetros como texturas, color, entre otros (figura 4.27).

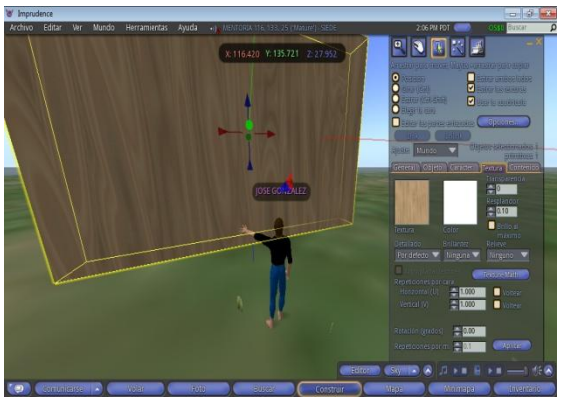


Figura 4.26 Creación de paredes.

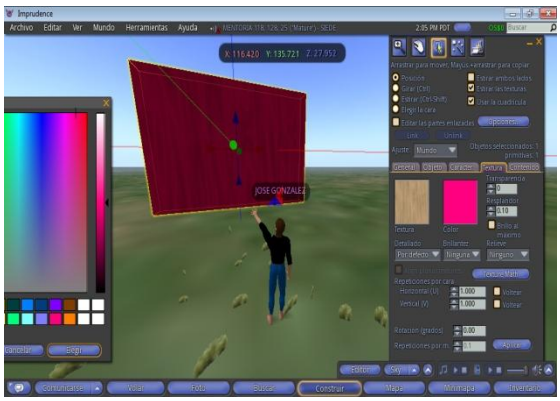


Figura 4.27 Edición de texturas y color en *prims*.

La creación del edificio incluye alteración de los *prims* básicos para poder crear detalles del edificio, tal como es el caso de ventanas, alterando la propiedad de *hueco* del *prim* (Figura 4.28) y agregando objetos con texturas transparentes para simular cristales (Figura 4.29).

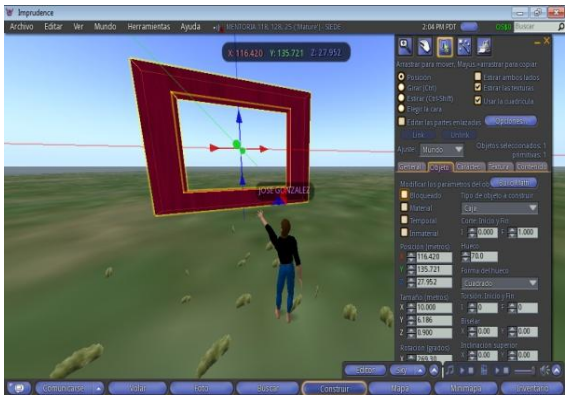


Figura 4.28 Propiedad *Hueco* de un *prim*.

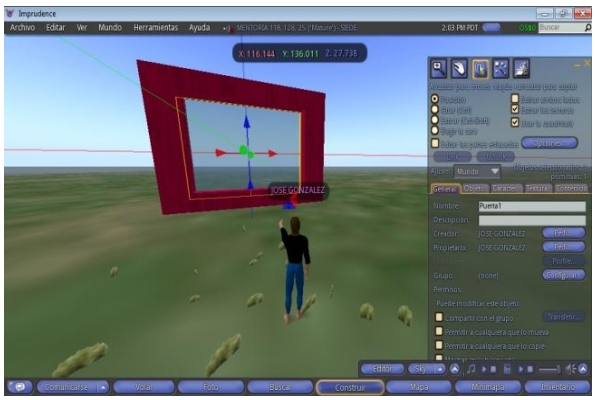


Figura 4.29 Trasparencia de objetos.

El primer edificio incluye cuatro secciones, que son dos salas de recepción, una zona de cubículos para atender asuntos personales y un auditorio, en las figuras 4.30 y 4.31 se puede observar la creación de la recepción del primer edificio.

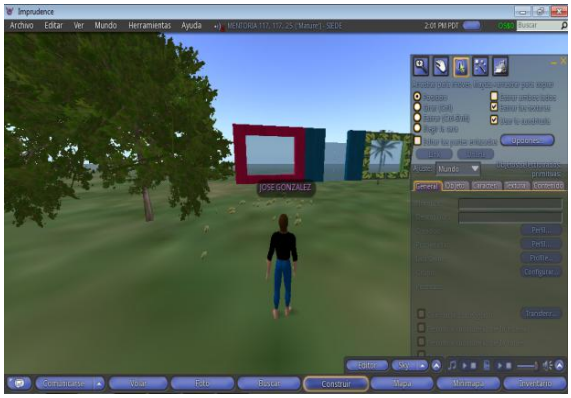


Figura 4.30 Parte frontal de la sala de recepción

Figura 4.31 Creación de la sala de recepción.

Después de la creación del espacio de sala de recepción, se acordó la creación del auditorio en la parte trasera del edificio, para ello se creó un espacio al doble del tamaño de la sala de recepción como se puede visualizar en la figura 4.32. En la figura 4.33 se puede observar la inclusión de piso y techo en el edificio.

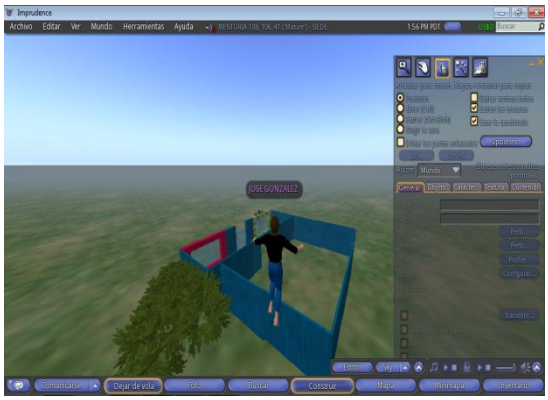


Figura 4.32 Creación del auditorio.

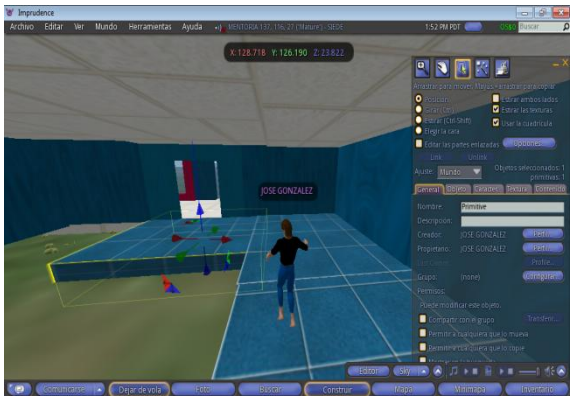


Figura 4.33 Creación de piso y techo del edificio.

En el auditorio se anexan detalles de ambiente que ayudarán a establecer un espacio adecuado para la reunión de 42 *avatars* a la vez (sentados), para ello se diseña una pantalla (figura 4.34) que pueda ser visualizada desde cualquier punto del espacio asignado al auditorio. En la figura 4.35 se puede observar la creación de diferentes niveles del piso para mayor alcance visual de la pantalla principal.

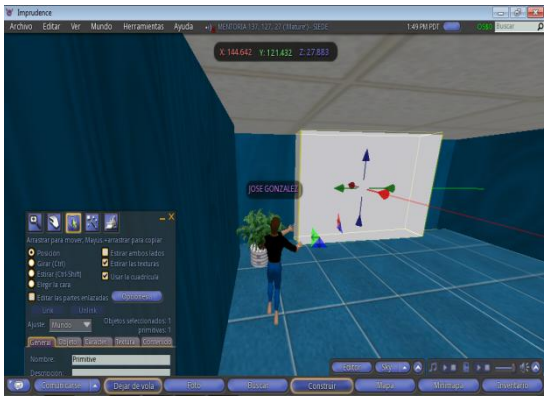


Figura 4.34 Pantalla principal del auditorio.

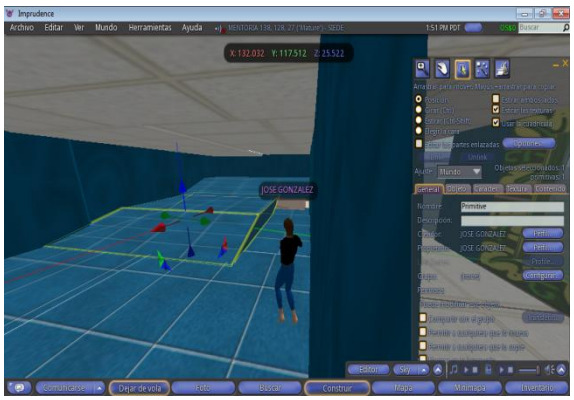


Figura 4.35 Niveles de piso en el auditorio.

Para lograr la interacción entre un asesor y sus asesorados, se incluyó la creación de un presídium para presentar o exponer el contenido mostrado en la pantalla principal (figura 4.36); la creación de los muebles es creada solo con *prims*, como se puede visualizar en la figura 4.37.

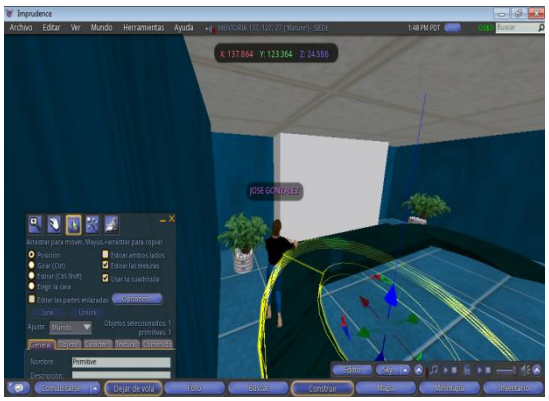


Figura 4.36 Creación de presídium.

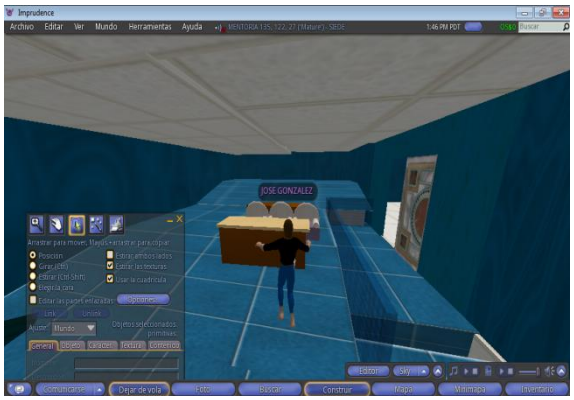


Figura 4.37 Creación de muebles de auditorio.

La creación total del auditorio, se puede visualizar en las figuras 4.38 y 4.39.

Figura 4.38 Vista asesor desde el auditorio.

Figura 4.39 Vista asesorado desde el auditorio.

La primera sala de recepción (Figura 4.40) implementa sillones, computadoras y puertas, todas realizadas con *prims*, los resultados se pueden mostrar en la figura 4.41.

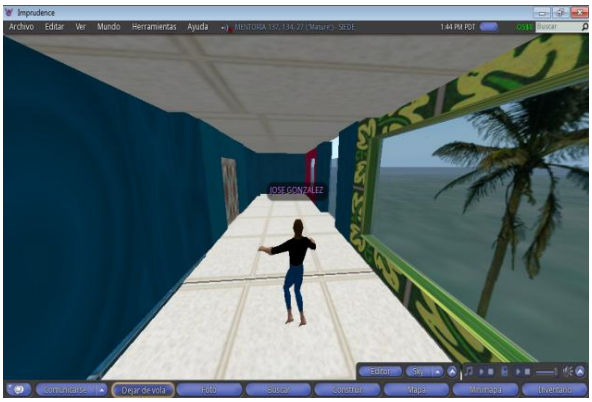


Figura 4.40 Recepción sin amueblar.

Figura 4.41 Recepción amueblada con *prims.*

Con ello se culmina la creación del auditorio y la primera recepción; el resultado se puede visualizar en la figura 4.42. Para la creación de la segunda recepción y la zona de cubículos de asesorías, se diseña un segundo piso para implementar estas dos secciones (figura 4.43).

Figura 4.42 Primer piso del CAE.

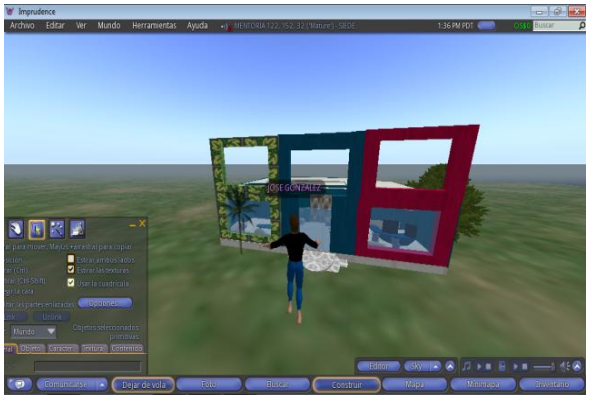


Figura 4.43 Diseño de segundo piso del CAE.

El acceso al segundo piso se logra con la creación de una escalera (Figura 4.44). En el segundo piso se diseñan cinco cubículos conde se podrán establecer asesorías, como se muestra en la figura 4.45.

Figura 4.44 Acceso al segundo piso.

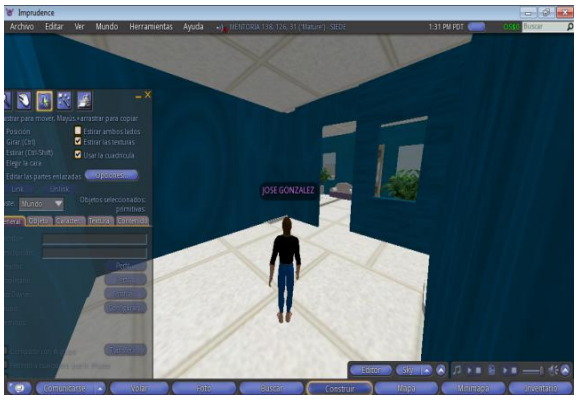


Figura 4.45 Diseño de cubículos.

El diseño final del edificio concluye con la creación y amueblamiento de los cubículos (Figuras 4.46 y 4.47), y la segunda sala de recepción (Figuras 4.48 y 4.49), el edificio final se puede visualizar en la figura 4.50.

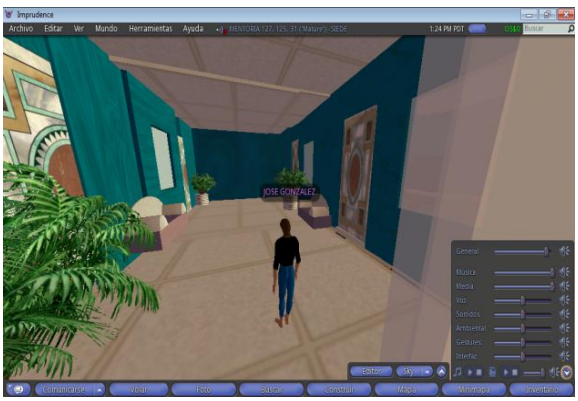


Figura 4.46 Área de cubículos.

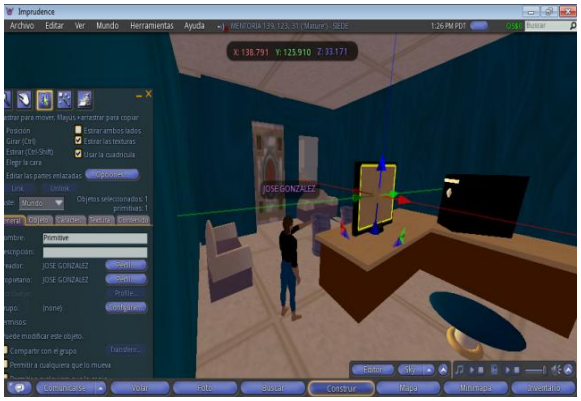


Figura 4.47 Amueblamiento de cubículos.

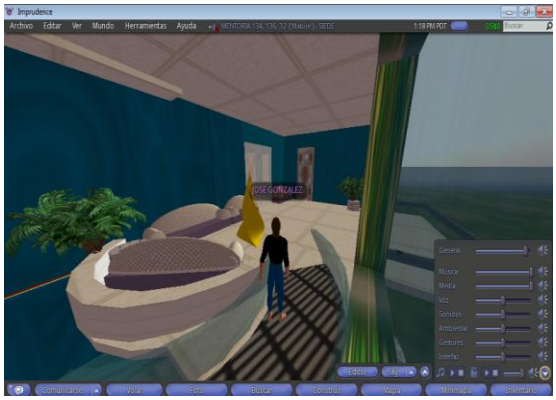


Figura 4.48 Segunda recepción.

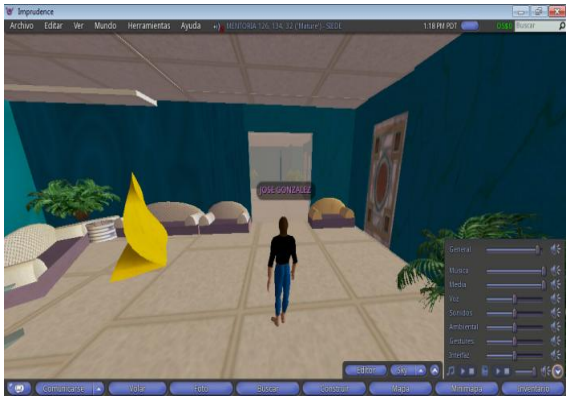


Figura 4.49 Amueblamiento de recepción.

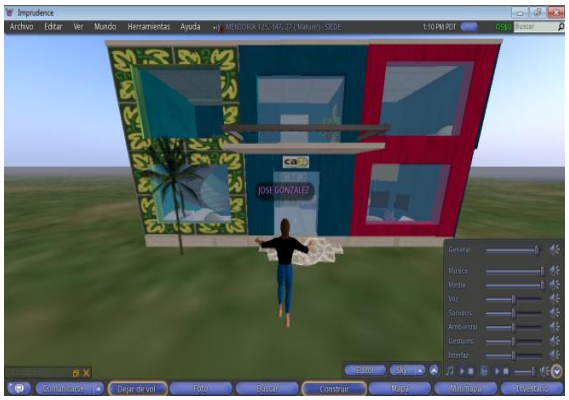


Figura 4.50 Diseño final del edificio CAE con *prims*.

4.4.2. Importación de edificios y objetos.

Imprudence Viewer implementa herramientas básicas para la construcción de objetos 3D; sin embargo, debido a que su objetivo principal es establecer la interacción entre en el mundo virtual y el usuario, estas herramientas lo colocan en desventaja con otras aplicaciones especializadas en diseño 3D, tales como *Blender, Autocad, Maya*, entre otros.

Imprudence Viewer permite importar objetos complejos desarrollados en otras herramientas de diseño 3D, se pueden importar imágenes, sonidos, videos, objetos y *sculpties*, a través de las opciones del menú *Archivo* en sus opciones *Import Objetc e Import Objetc + Upload(texture),* estas opciones se pueden visualizar en la figura 4.51.

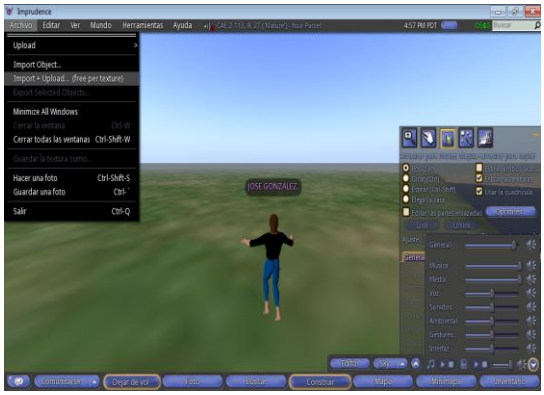


Figura 4.51 Opciones de importación de objetos.

Los edificios u objetos creados en otras aplicaciones de diseño se importan en formato *.xml.* Existen construcciones que se obtienen con licencia libre. Para diseñar el resto del mundo virtual, se utilizaron objetos y edificios importados a *Imprudence Viewer*. En las figuras 4.52 y 4.53 se puede observar la importacion de objetos básicos y *sculpties* respectivamente.

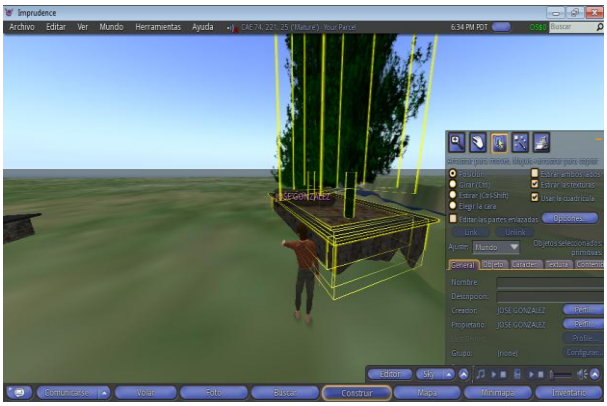


Figura 4.52 Importación de objeto: *Árbol*.

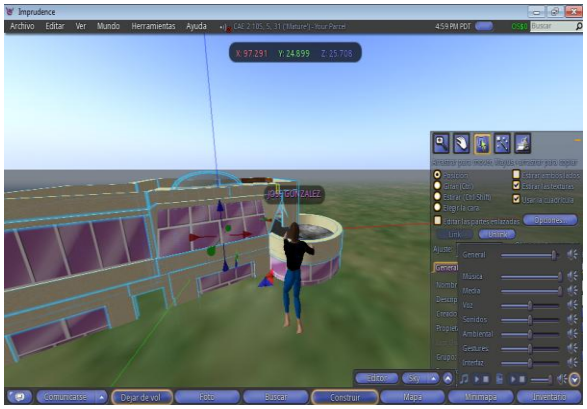


Figura 4.53 Importación de *sculpties.*

4.4.3. Programa de Movilidad Estudiantil

El programa de Movilidad Estudiantil proporciona información y trámites necesarios para que los estudiantes inscritos en la Benemérita Universidad Autónoma de Puebla realicen estancias en otras universidades, ya sea fuera o dentro del país, con el objetivo de establecer relaciones entre las unidades académicas, además de compartir experiencias que ayuden al crecimiento profesional de los alumnos para el beneficio de la sociedad.

En el mundo virtual podrá acceder cualquier alumno o catedrático de la BUAP utilizando su matrícula o identificador de usuario, el edificio especializado para este programa se basa en dos construcciones, la primera es la utilización del edificio principal que es realizado solo con *prims (*visualizado en el capitulo 4.4.1); el segundo, es una implementación de cuatro espacios de trabajo utilizados para

poder establecer interacción con los alumnos, ya sea con conferencias, videos e información visual, estas instalaciones se obtienen de manera externa. Los resultados de construcción se pueden visualizar en las figuras 4.54 y 4.55.

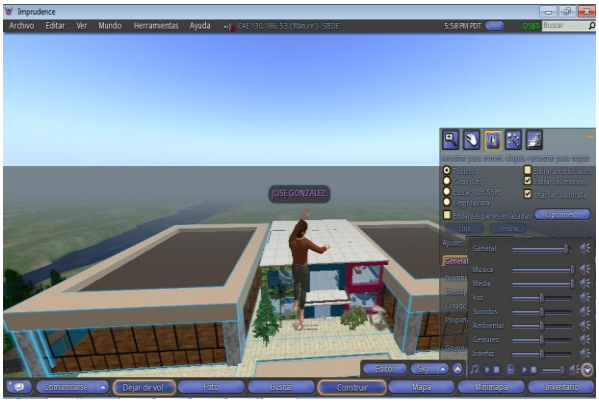


Figura 4.54 Anexión de CUATRO edificios de apoyo.

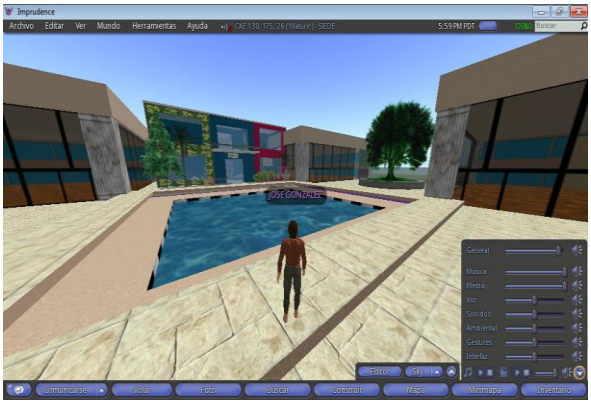


Figura 4.55 Movilidad Estudiantil: Vista frontal.

4.4.4. Programa Educando para la Salud

El programa Educando para la salud es un esfuerzo para sembrar en los estudiantes el interés por la adopción de hábitos y conductas saludables como parte de su desarrollo, creando una cultura de prevención y de compromiso social, a través del auto-cuidado a la salud [35]. Sus objetivos son los siguientes:

- Promover una actitud responsable con el medio ambiente.
- Promover la práctica del deporte.
- Promover hábitos sobre higiene personal y colectiva.
- Informar sobre los daños que las adicciones más comunes ocasionan a la salud.
- Concientizar sobre la necesidad de una conducta sexual responsable.
- Informar sobre las bondades de una alimentación sana y adecuada.
- Fomentar la cultura por la legalidad y prevención del delito.

Para fomentar a cumplir estos objetivos, en el mundo virtual se crea una instalación especial para este programa donde será colocada información acerca de los beneficios que posee la salud para un buen desarrollo profesional. Este programa consta de una isla especial, donde se incrusta un edificio de dos plantas, la construcción de manera general se puede visualizar en la figuras 4.56 y 4.57.

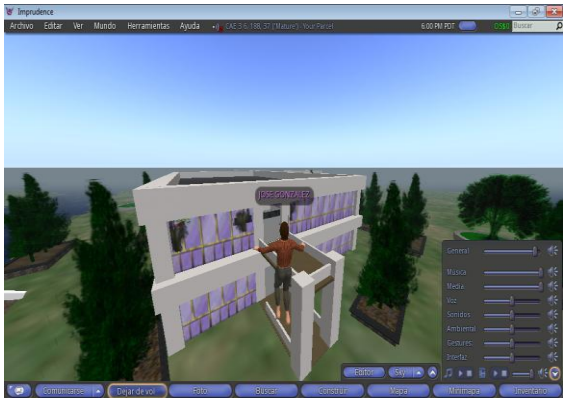


Figura 4.56 Programa de Salud: Vista frontal.

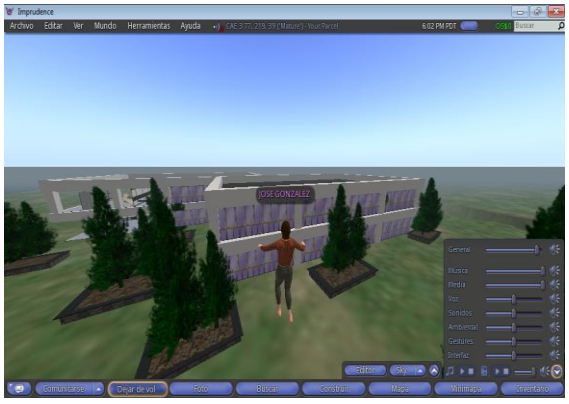


Figura 4.57 Programa de Salud: Vista posterior.

En la parte interior del edificio se pueden visualizar dos pisos (Figura 4.58) en donde existen salas de proporción de información (Figura 4.59), salas de discusión (Figura 4.60), cubículos donde se brinda información personal (Figura 4.61), entre otros aspectos, para poder establecer los objetivos del programa.

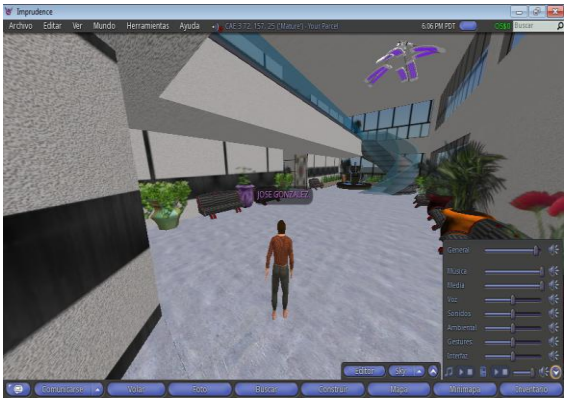


Figura 4.58 Programa de Salud: Dos pisos.

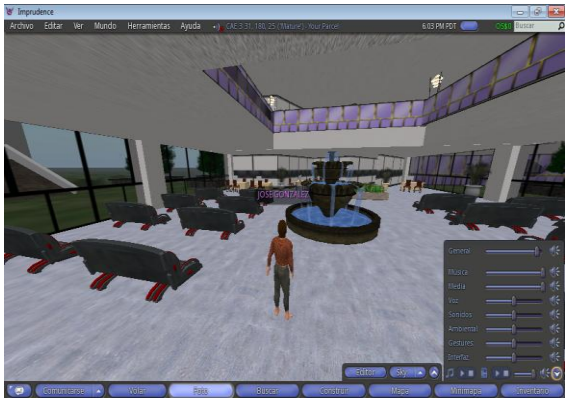


Figura 4.59 Programa de Salud: Sala de información.

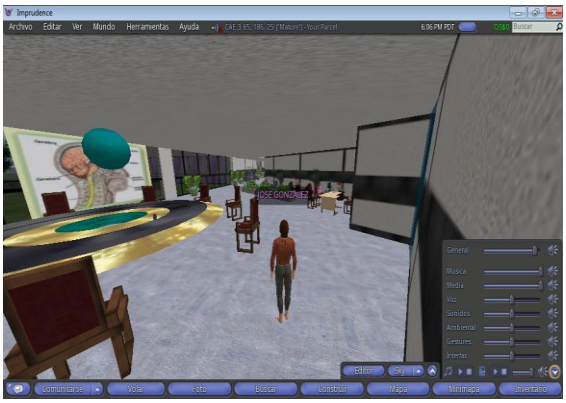


Figura 4.60 Programa de Salud: Sala de discusión.

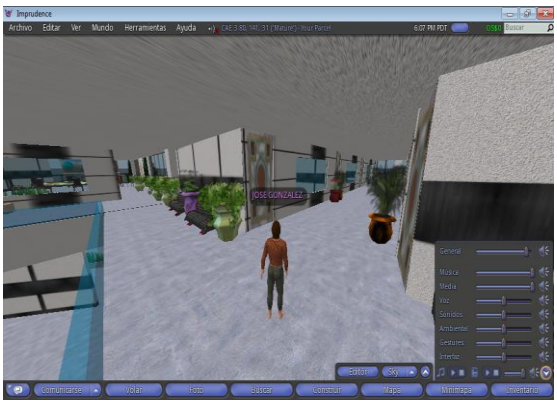


Figura 4.61 Programa de Salud: Cubículos.

4.4.5. Programa de Apoyo Académico a Estudiantes Indígenas de la BUAP.

El “Programa de Apoyo Académico a Estudiantes Indígenas de la BUAP” conocido por sus siglas: *PAAEI,* se creó con el fin de proporcionar a los estudiantes universitarios originarios de comunidades indígenas, los apoyos necesarios para que logren una formación académica integral que los conduzca a concluir sus estudios profesionales, con igualdad de oportunidades y con pleno respeto a su identidad cultural [36].

Al concentrar programas y acciones de apoyo, se busca fortalecer la atención de todas y cada una de las necesidades que presentan los estudiantes inscritos en el programa, identificando y ofreciendo una gama de herramientas (cursos, talleres), que complementen su instrucción y que les permitan la identificación de aptitudes que no solo les sirvan en su trayectoria profesional, sino que puedan aplicarlas también en el ámbito personal, familiar o incluso como servicio en su propia comunidad, aspecto muy relevante para la mayor parte de ellos.

Para cumplir estos objetivos, en el mundo virtual se crea una instalación especial para este programa donde será colocada información acerca de los apoyos que tendrán los beneficiados, con el fin de evitar nerviosismo, temor, entre

otros factores psicológicos que llegan a afectar este programa. Este programa consta de una isla especial, donde se incrusta un edificio de dos plantas, la construcción de manera general se puede visualizar en la figuras 4.62 y 4.63.

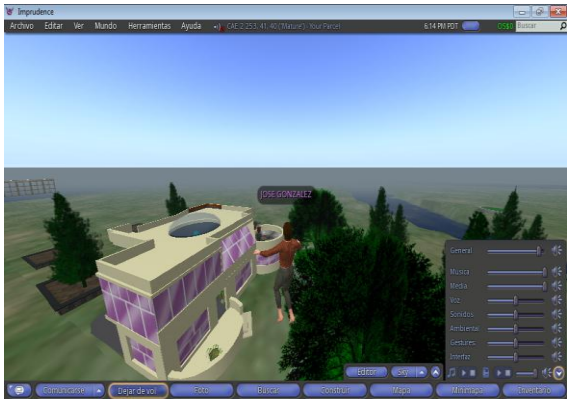


Figura 4.62 PAAEI: Vista frontal.

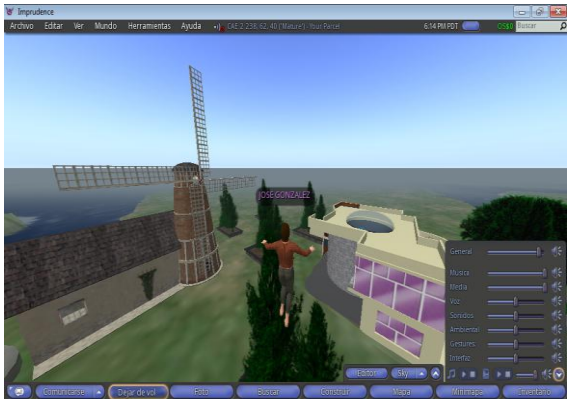


Figura 4.63 PAAEI: Vista posterior.

En la parte interior del edificio se pueden visualizar dos pisos (Figura 4.64) en donde existen salas de proporción de información (Figura 4.65 y Figura 4.66), salas de discusión (Figura 4.67), entre otros aspectos visuales para poder cumplir los objetivos del programa.

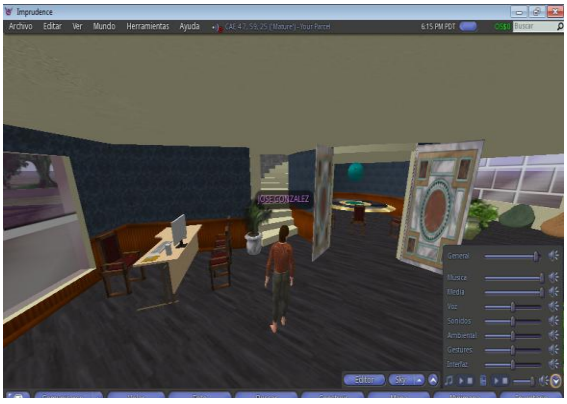


Figura 4.64 PAAEI: Acceso a segundo piso.

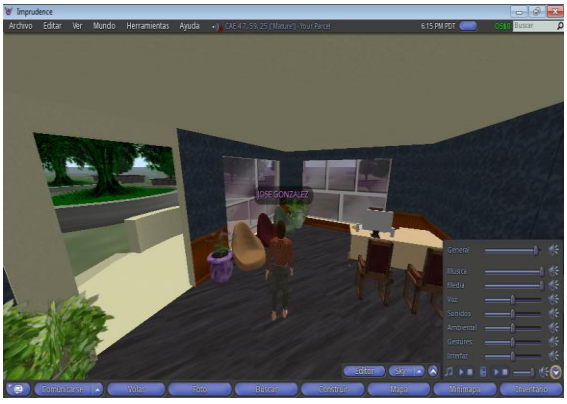


Figura 4.65 PAAEI: Recepción.

Figura 4.66 PAAEI: Sala de información.

Figura 4.67 PAAEI: Sala de discusión.

4.4.6. Programa de Becas y Apoyo para la Permanencia

El programa de Becas y Apoyo para la Permanencia, mejor conocido por sus siglas: BAP, es un sistema creado por la Coordinación de Acompañamiento al Estudiante con el fin de hacer cercanos los apoyos económicos que existen actualmente hacia los alumnos de la BUAP. Posee como objetivo, facilitar la información, trámites o procesos para obtención de becas a los alumnos que poseen los requisitos para ser beneficiarios y así lograr un mayor aprovechamiento escolar.

Para poder lograr estas expectativas, se crea en la cuarta isla del mundo virtual el edificio del Programa BAP, cuenta con un total de cuatro pisos, la vista general se presenta en las figuras 4.68 y 4.69.

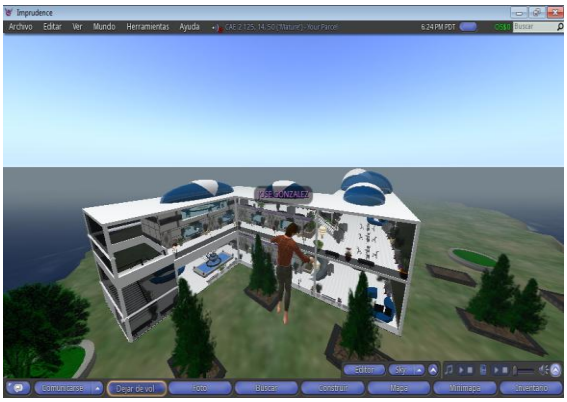


Figura 4.68 BAP: Vista frontal.

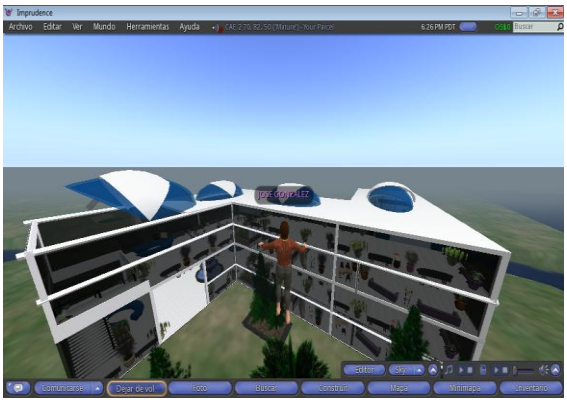


Figura 4.69 BAP: Vista posterior.

El interior del edificio se puede visualizar en la figura 4.70, donde se encontrarán además salas de reunión o discusión (Figura 4.71), salas de espera o recreación (Figura 4.72) y cubículos para atención personalizada (Figura 4.73).

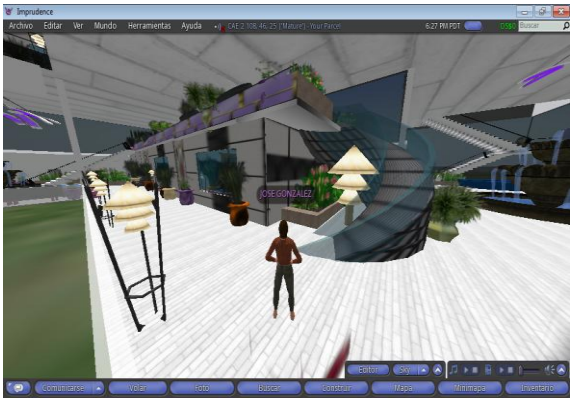


Figura 4.70 BAP: Interior del edificio.

Figura 4.71 BAP: Sala de reunión.

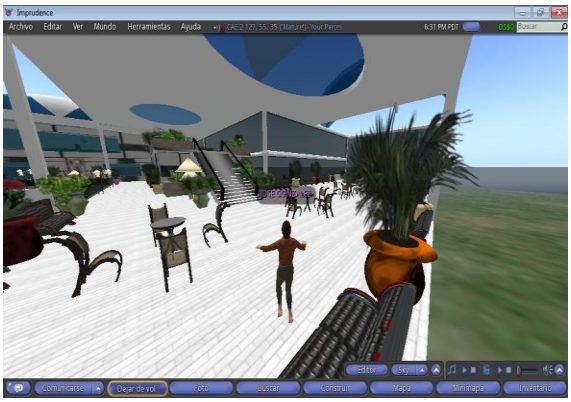


Figura 4.72 BAP: Sala de espera o recreación.

Figura 4.73 BAP: Cubículos.

4.5. Diseño Final

El diseño final es la creación de cuatro islas que albergan los programas de apoyo a los estudiantes de la BUAP, la vista general se presenta en el *mini mapa* de la Figura 4.74. Con ello queda concluido el diseño del mundo virtual, y se procede a establecer la interacción con el mundo virtual creado.

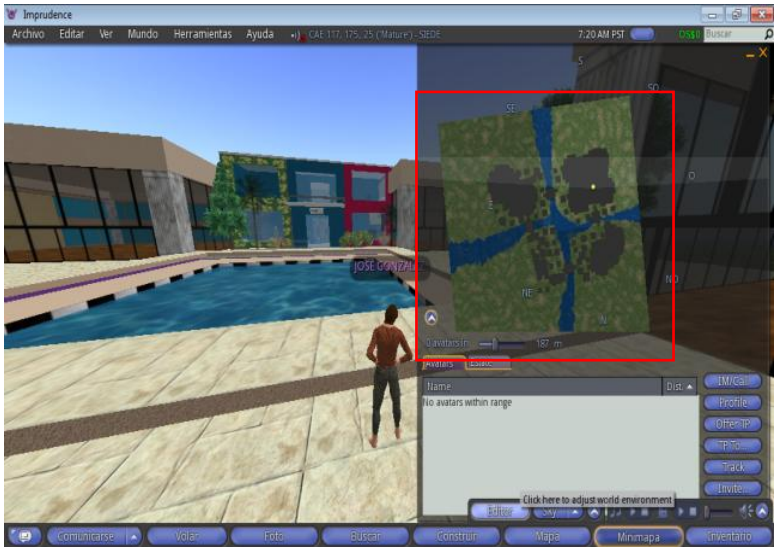


Figura 4.74 Sistema de Mentoría BUAP: Diseño Final.

4.6. Interacción con el mundo virtual

Para poder establecer la interacción con el mundo virtual de Mentoría, es necesario programar los objetos a interactuar, para que estos puedan realizar las acciones que ayuden a los alumnos a visualizar información de los programas entre otras acciones.

La interacción se basa en establecer las acciones que realizarán los objetos cuando los usuarios lleguen a tocarlos o estar cerca de ellos, estas acciones se pueden programar incrustando código LSL en los objetos a interactuar.

Para facilitar el aprendizaje de LSL, existe un programa gratuito llamado *LSL Editor*, que simula el comportamiento de este lenguaje, este puede ser descargado en: http://www.lsleditor.org/Download.aspx. La interfaz del programa puede visualizarse en la figura 4.75, y por defecto posee un script básico: cada vez que se toque un objeto, este mandará el mensaje "*Hello Avatar*", su ejecución se puede visualizar en la figura 4.76.

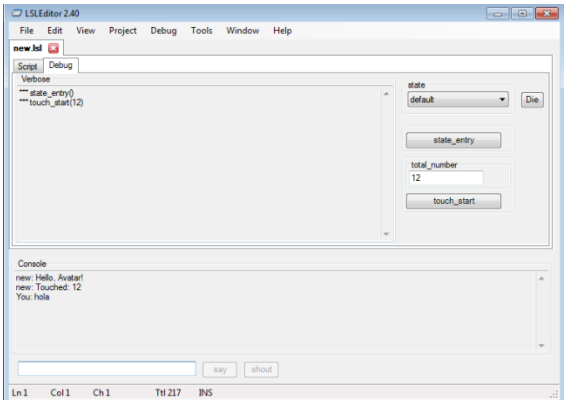
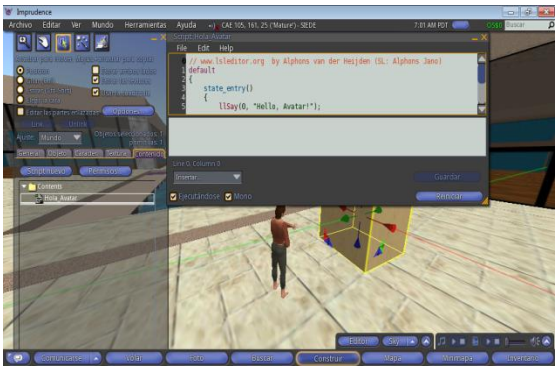


Figura 4.75 *LSL Editor* con ejemplo "Hello Avatar." **Figura 4.76** Ejecución del script en *LSL E ditor.*

La simulación se basa en dos estados: *State_entry,* identifica la acción que realizará el objeto a la hora de ser ejecutado. *Touch_start,* detecta que objeto fue el que lo tocó (identificador, que puede ser un avatar), además de mostrar el mensaje "*Hello Avatar".*

Este mismo código puede ser colocado en un objeto del mundo virtual para poder visualizar de manera clara su funcionamiento. El código es insertado en: *propiedades del objeto/contenido/script nuevo* (Figura 4.77). El resultado se visualiza en la figura 4.78.



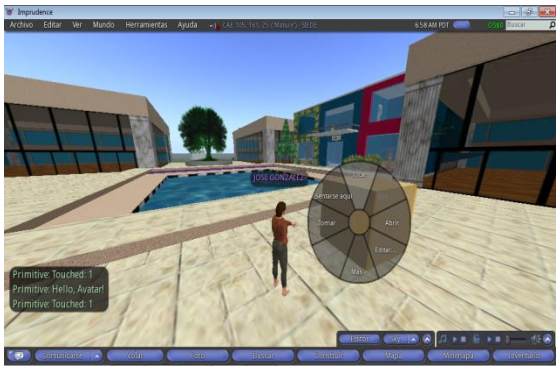


Figura 4.77 Incrustación de scripts. **Figura 4.78** Ejecución de script en mundo virtual.

4.6.1. Scripts de sentarse

En el mundo virtual fueron colocados muebles, para simular un entorno más acorde a la realidad.

Para hacer el entorno amigable al usuario se implementó un *script* que permite al usuario sentarse en los objetos destinados a este movimiento natural del ser humano. La codificación se puede visualizar a continuación.

```
default
{
    state_entry ()
    {


     llSitTarget(<0.50, -0.5, -0.7>,
     llEuler2Rot(<0,90,0> * DEG_TO_RAD));
     llSetSitText ("Sit Awhile");
    }
}
```

El *script* se basa generalmente de dos instrucciones:

- **llSitTarget:** Es método implementado por LSL, que permite al avatar tomar la posición de una persona sentándose, sus parámetros controlan la altura, posición izquierda-derecha, y la profundidad que tendrá la posición del avatar con respecto al objeto.
- **llEuler2Rot:** Función que controla la rotación de la posición del avatar sobre el objeto.

Es importante mencionar que los parámetros varían de acuerdo a la forma del objeto y la manera de concebirlo (programador), por lo tanto todo objeto donde se implemente este *script,* será totalmente personalizable, posiblemente con diferentes parámetros a los demás objetos que simulen este comportamiento humano en el mundo virtual.

Los resultados se pueden visualizar en las figuras 4.79 y 4.80, donde se implementa el *script* en un sillón que forma parte del mundo virtual creado.

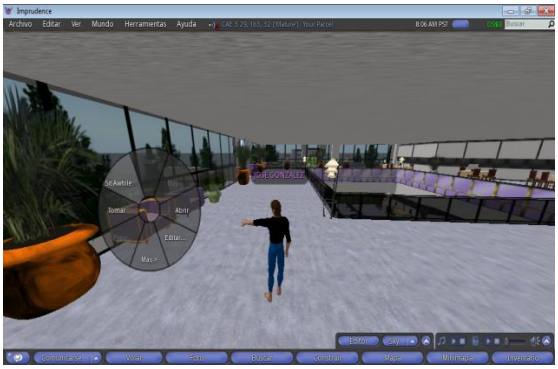


Figura 4.79 Solicitud de "*Sit Awhile"* (sentarse).

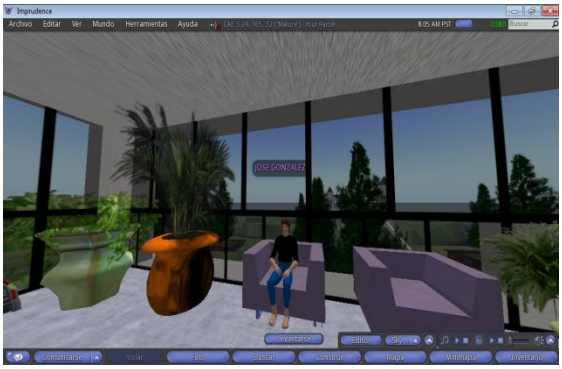


Figura 4.80 Efecto causado por el *script "Sit Awhile".*

4.6.2. Scripts de video y sitios web

Para la incrustación de contenido web en el mundo virtual, se implementa un script con la instrucción *llSetPrimMediaParams.*

```
default
{
    state_entry()
    {
        llSetPrimMediaParams(0,
            [PRIM_MEDIA_AUTO_PLAY,TRUE,
             PRIM_MEDIA_CURRENT_URL,"http://youtu.be/T-sxSd1uwoU",
             PRIM_MEDIA_HOME_URL,"http://youtu.be/T-sxSd1uwoU",
             PRIM_MEDIA_HEIGHT_PIXELS,512,
             PRIM_MEDIA_WIDTH_PIXELS,512]);
    }
}
```

- **llSetPrimMediaParams**: Indica que se colocará un contenido multimedia, el parámetro "0", indica la parte o cara del objeto donde se colocará el contenido.
- **PRIM_MEDIA_AUTO_PLAY:** Reproduce inmediatamente el contenido.

- **PRIM_MEDIA_CURRENT_URL:** Indica la página o dirección de video a visualizar en el objeto.
- **PRIM_MEDIA_HOME_URL:** Indica la página de inicio del *script.*
- **PRIM_MEDIA_HEIGHT_PIXELS:** Altura en pixeles del contenido a mostrar.
- **PRIM_MEDIA_WIDTH_PIXELS:** Ancho en pixeles del contenido a mostrar.

Los resultados se pueden visualizar en las figuras 4.81 y 4.82, donde se reproduce una página web y un video extraído de www.youtube.com respectivamente.

Figura 4.81 *Script* mostrando una página web.

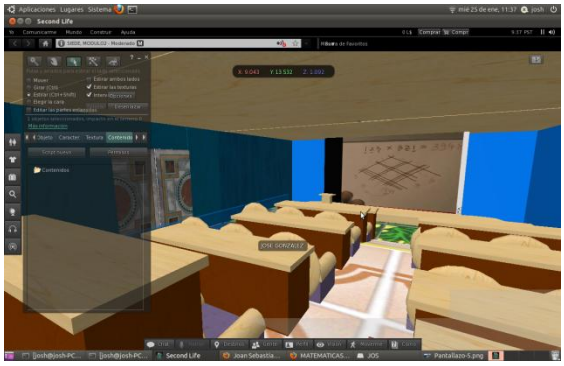


Figura 4.82 *Script* mostrando un video.

4.6.3. Scripts de puertas

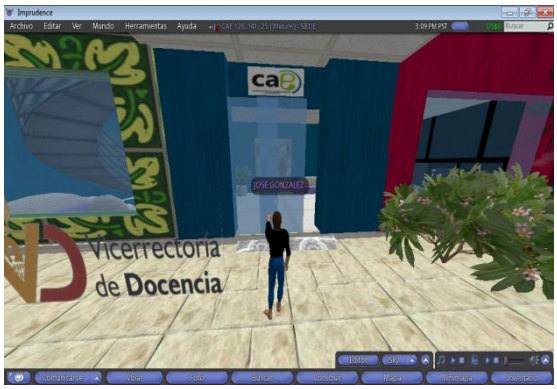
En el mundo virtual se implementaron puertas para permitir el acceso a los residentes. Se desarrollaron dos tipos de puertas, la primera de ellas permite deslizarse hacia los lados, es decir, paralelamente a la pared, en el siguiente código se rescata de manera general el comportamiento que tendría el objeto:

```
default
{
    state_entry()
    {
        pos = llGetPos();
        rot = llGetRot();
        opos = pos + dir*(<0,amt,offset>*rot);
        open = FALSE;
    }
    on_rez(integer n)
    {
        llResetScript();
    }

     touch_start(integer total_number)
    {
        if( open )
        {
            close();
        }
        else
        {
            slide(opos);
            llSetTimerEvent(delay);
            open = TRUE;
        }
    }

    timer()
    {
        close();
    }
}
```

El *script* desarrolla tiempo de espera para abrir la puerta y tiempo que tardará en cerrarse; además de que puede funcionar al tacto del *avatar* (ya sea dando *"click"* sobre el objeto o a la hora de entrar en colisión con él). El resultado se visualiza en las figuras 4.83 y 4.84.



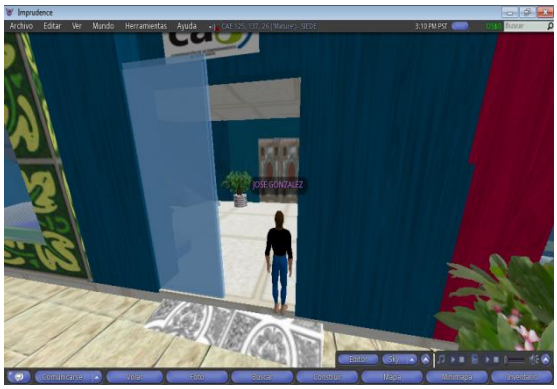
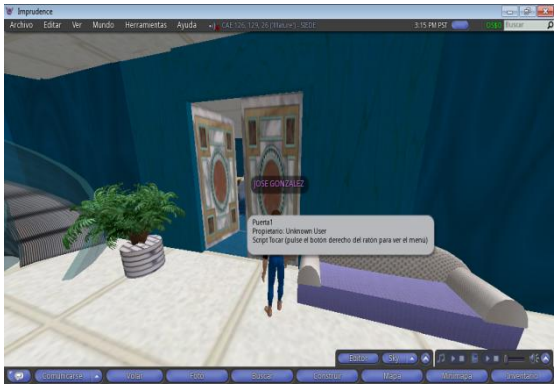


Figura 4.83 Abriendo la puerta deslizable. **Figura 4.84** Entrando a través de la puerta deslizable.

Las puertas son solo objetos básicos (ambas puertas son solo un par de cubos), la simulación que hace posible el “abrir o cerrar”, es implementado gracias a instrucciones de rotación y desplazamiento del objeto, en las figuras 4.85 y 4.86, se puede visualizar una puerta que se abre hacia dentro de la habitación.



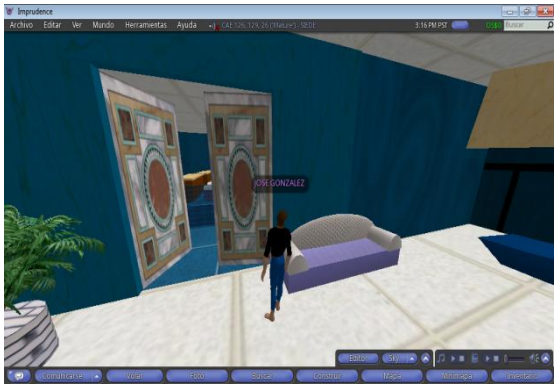


Figura 4.85 Contacto para abrir una puerta. **Figura 4.86** Efecto de puerta que se abre hacia el interior.

La incrustación de los scripts básicos, permiten la interacción del avatar con el mundo virtual. Sin embargo es necesario que nuevos usuarios accedan al mundo, para ello se implementa una página de registro al entorno virtual.

4.7. Registro

Opensim, simula el mundo virtual, sin embargo no posee una interfaz que pueda establecer el registro de nuevos usuarios para acceder al mundo.

Diva Distribution es una versión especial de *Opensim* que permite la administración básica de un servidor desde un navegador, permitiendo acceder a información del mundo virtual de usuarios ya existentes o la creación de nuevos residentes en el espacio.

Diva Distribution, puede ser anexada al proyecto actual, descargando la última versión de esta distribución libre en la página: http://github.com/diva/d2/downloads.

Una vez teniendo el archivo, se descomprime y solo basta con realizar dos acciones:

- ✓ Copiar la carpeta *WifiPages* (incluida en la distribución de *diva* descargada) en la raíz del proyecto de *Opensim* creado para el mundo virtual.
- ✓ Copiar a la carpeta *Opensim/bin* del proyecto realizado los siguientes archivos:

  - Diva.Wifi.dll
  - Diva.Wifi.ScriptEngine.dll
  - Diva.OpenSimServices.dll
  - Diva.Utils.dll
  - Diva.Modules.dll

Además de ello, es necesario modificar el archivo *bin/config-include/ StandaloneCommon.ini* anexando al inicio del mismo la instrucción que indica el puerto a utilizar por la aplicación web:

```
[Network]
    port = 9000
```

Se indica la utilización de *Diva Distribution* anexando las siguientes líneas [24]:

```
[Modules]
WifiModule = true


[WifiService]
    GridName = "grid name"
    LoginURL = "http://<IP del servidor>"
    WebAddress = "http://IP del servidor"

    AdminFirst = "Wifi"
    AdminLast = "Admin"
    AdminEmail = "you@example.com"

    AccountConfirmationRequired = false

    AvatarAccount_Female = "Female Avatar"
    AvatarAccount_Male = "Male Avatar"
    AvatarAccount_Neutral = "Neutral Avatar"
    SmtpHost = "mail.example.com"
    SmtpPort = "587"
    SmtpUsername = "your_account_in_this_mail_server"
    SmtpPassword = "your_password_in_this_mail_server"
```

Se ejecuta el proyecto de O*pensim,* su funcionamiento es normal, solo con la excepción de que se hace accesible la dirección http://<Ip del servidor>:9000/wifi, mostrando una interfaz similar al de la figura 4.87:

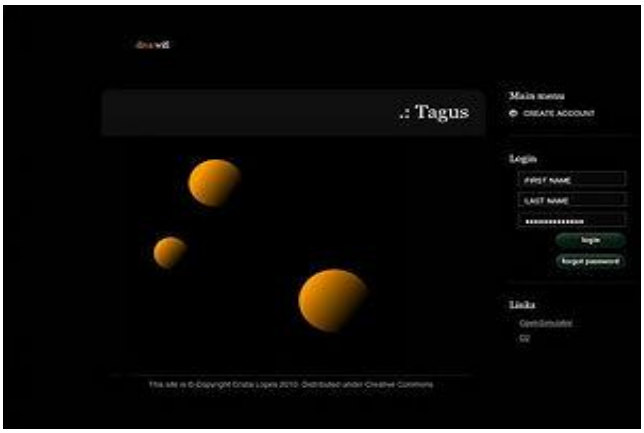


Figura 4.87 Interfaz de *Diva.*

La interfaz de *Diva* permite: El registro de nuevos usuarios al mundo virtual, el acceso de los usuarios a los objetos que le pertenecen en el mundo virtual, modificación de su cuenta, entre otras opciones.

4.8. Respaldo del proyecto de Mentoría

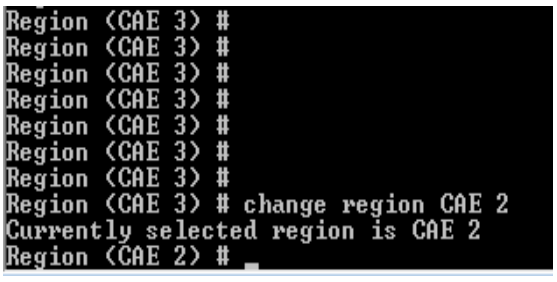
El mundo virtual creado en *Opensim,* puede ser respaldado a través de la instrucción:

save oar "region".oar

El proyecto consta de cuatro regiones: MENTORIA, MENTORIA 2, MENTORIA 3 y MENTORIA 4; es por ello que se realiza un respaldo de cada isla o región. Cuando se ejecuta la instrucción *save,* se guarda solo la región predeterminada (la primera isla creada). Para hacer un respaldo de las cuatro regiones es de vital importancia cambiar de región para poder respaldar cada una de ellas a través de la instrucción:

```
change region "Region"
```

En la figura 4.88 se visualiza el cambio de región para poder respaldarla y en la figura 4.89 se ejecuta la instrucción *save,* los archivos respaldados se guardan automáticamente en la carpeta */bin.*



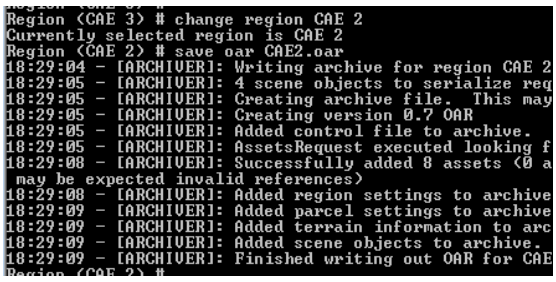


Figura 4.88 Cambio de región en el proyecto. **Figura 4.89** Respaldando región actual.

El procedimiento inverso (importar una región) se logra con la instrucción:

```
load oar "region".oar
```

4.9. Registro al Sistema de Mentoría

El diseño del Sistema de Mentoría BUAP es diseñado exclusivamente para alumnos y catedráticos de la BUAP, es por ello, que para poder establecer el registro de alumnos se exportó el proyecto creado en *Opensim* a la distribución de *Diva.*

Para poder establecer el registro validando a los usuarios pertenecientes a la BUAP, se descargó la versión de *Opensim* en su versión *source* desde: http://opensimulator.org/wiki/Download. *Diva* puede ser descargada en su versión *source* desde: *https://github.com/diva/diva-distribution.*

Es importante mencionar que la modificación del código de *Opensim*, así como de *Diva,* se llevó a cabo con el uso de *Visual Studio 2010*, utilizando la versión *source* de o*pensim 0.7.*3, esto debido a que en versiones anteriores a estas aplicaciones existen conflictos por falta de librerías aun no implementadas en las mismas.

Una vez descargada la versión *source* de diva (*diva-diva-distribution-d9789ce,* es la que se utilizó en este proyecto), es necesario copiar la carpeta "bin" del proyecto actual (*Opensim*) y ejecutar el archivo por lotes: *runprebild.bat.* (Figura 4.90); el resultado es la creación de un proyecto que puede ser modificado con Visual Studio en lenguaje C# (Figura 4.91).

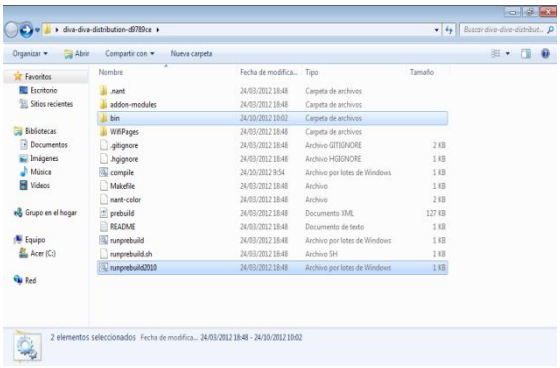


Figura 4.90 Compilación de DIVA (*source*).

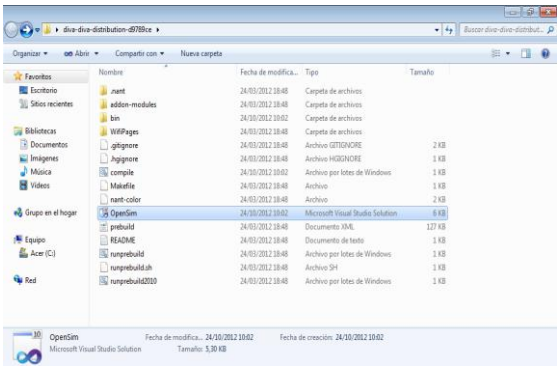


Figura 4.91 Creación del proyecto DIVA.

Diva Distribution es un conjunto de 10 proyectos, estos se visualizan en la figura 4.92 en la parte superior derecha. Para poder realizar lo necesario para validar la matricula de un alumno, es necesario ingresar la variable "matricula" como parámetro de conexión. Las modificaciones se realizan en el proyecto o módulo *Diva.Wifi/WebApp/Services.NewAccount.cs* (Figura 4.93).

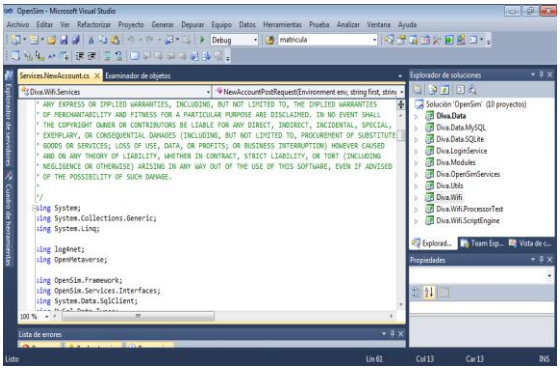


Figura 4.92 Proyectos de *Diva Distribution*.

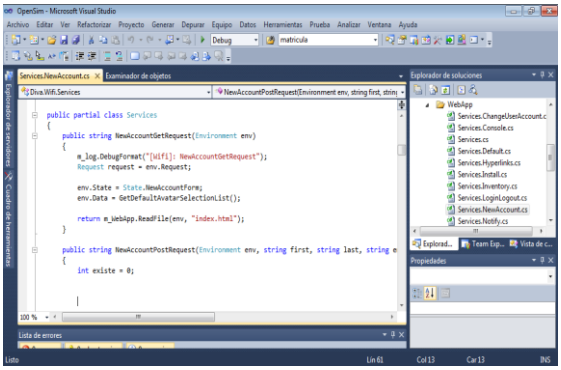


Figura 4.93 *Services.NewAccount.cs.*

Dentro del proyecto *Diva.Wifi.NewAccount* se agregó el campo *"matricula*" en la función *NewAccountPostRequest,* que se encargará de recuperar los campos que sean otorgados por la página web.

```
public string NewAccountPostRequest(Environment env, string first, string last,
string email, string password, string password2, string avatarType, string
matricula)
```

Se verificará si el dato enviado *"matricula"* así como los demás campos no sean vacios.

```
if ((password != string.Empty) && (password == password2) && (first != string.Empty)
&& (last != string.Empty) && (matricula != string.Empty))
```

Una vez verificada se envía la instrucción que prepara la nueva cuenta en el sistema, pero sin ingresarla a la BD:

```
account = new UserAccount(UUID.Zero, first, last, email, matricula);
                    account.ServiceURLs = urls;
                    account.UserTitle = "Local User";
```

Para validar si los usuarios pertenecen a la institución, se realiza una validación por medio de *MySQL*, a una BD externa donde se podrá validar que la matricula sea real.

```
int existe = 0;
first = matricula;
string path2 = "Server=<Ip del servidor>;Database=<nombre de la base>;Uid=root;Pwd=
;Port=3306";

MySqlConnection cn2 = new MySqlConnection(path2);
cn2.Open();
MySqlCommand cmd2 = new MySqlCommand();
cmd2.Connection = cn2;

cmd2.CommandText = "select count(*) from acadeicos where
Id_Usuario='"+matricula+"'";
existe = Convert.ToInt32( cmd2.ExecuteScalar());
m_log.DebugFormat("[Wifi] Existe = {0}",existe);
```

En caso que la matricula exista, se guardará en la Base de Datos de *Opensim* creando un nuevo usuario, en caso contrario, nos devolverá a la página principal de registro.

```
if (existe > 0)
    m_UserAccountService.StoreUserAccount(account);
else
    {
    string notification2 = _("La Matricula no existe en la Base de Datos", env);
    NotifyWithoutButton(env, notification2);
    return m_WebApp.ReadFile(env, "index.html");
    }
```

```
cn2.Close();
```

Estas acciones se realizan en el proyecto *Diva.Wifi/Apps/NewAccount*, posteriormente se compila el proyecto *Diva.Wifi.* Al realizar esta acción marcará errores que indicarán que la matricula nunca ha sido declarada; la declaración se realiza en el proyecto de *Opensim*, es por ello que la modificación del proyecto de *Diva* queda finalizado, y ahora se ejecutará el proyecto *source* de *Opensim.*

Los pasos para poder visualizar el proyecto de o*pensim* en su versión *source*, son similares a los de *Diva Distribution*, se compone de 92 proyectos, como se visualiza en la figura 4.94.

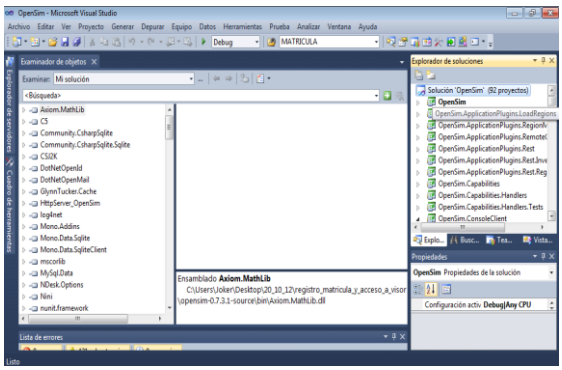


Figura 4.94 Proyecto *source* de *Opensim*.

Es necesario modificar el proyecto de *Opensim*, para exportar los módulos (.dll) al proyecto de Diva, a continuación se muestran los métodos principales modificados en el proyecto.

En el proyecto *OpenSim.services.Interfaces/IUserAccountService.cs* es necesario validar el nuevo campo “matricula” como dato que caracterice la creación de un nuevo usuario, en los métodos *UserAccount* se agregó el campo matricula, así como su declaración.

```
public string Matricula;


public UserAccount(…, string matricula)
{Matricula = matricula;

}
```

Es necesario modificar el archivo *Opensim.Services.UserAccount,* se anexará un método que sirva para validar exclusivamente la matricula entre los registros que ya se encuentren en la BD de *Opensim*.

```
public UserAccount GetUserAccount(UUID scopeID, string firstName,
                string lastName, string Matricula)
        {

            UserAccountData[] d;

            if (scopeID != UUID.Zero)
            {
                 d= m_Database.Get(
                        new string[] { "ScopeID", "Matricula" },
                        new string[] { scopeID.ToString(), Matricula });

                if (d.Length < 1)
                {
                    d = m_Database.Get(
                            new string[] { "ScopeID", "Matricula" },
                            new string[] { UUID.Zero.ToString(), Matricula });

                }
            }
            else
            {
              d = m_Database.Get(
                        new string[] { "Matricula" },
                        new string[] { Matricula });
            }

            if ((d.Length < 1))
                return null;

            return MakeUserAccount(d[0]);
        }
```

Se anexa el campo “matricula” en la instrucción que guarda la información en la BD.

```
public bool StoreUserAccount(UserAccount data)
        {

            UserAccountData d = new UserAccountData();

            d.FirstName = data.FirstName;
            d.LastName = data.LastName;
            d.Matricula = data.Matricula;
     ...}
```

En la función *HandleShowAccount* es importante rescatar el valor del campo recibido “matricula”, ya que ahora forma parte de la validación:

```
protected void HandleShowAccount(string module, string[] cmdparams)
        {
            if (cmdparams.Length != 4)
            {
    MainConsole.Instance.Output("Usage: show account <first-name> <last-name>");
                return;
            }

            string firstName = cmdparams[2];
            string lastName = cmdparams[3];
            string Matricula = cmdparams[6];
UserAccount ua = GetUserAccount(UUID.Zero, firstName, lastName, Matricula);
       …}
```

La función *CreateUser* es de vital importancia modificarla para el agregado de la matricula pues realiza la creación del usuario tanto físicamente como en la BD.

```
public UserAccount CreateUser(UUID scopeID, UUID principalID, string firstName,
string lastName, string password, string email, string matricula)
{
UserAccount account = GetUserAccount(UUID.Zero, firstName, lastName, matricula);
if (null == account)
            {
account = new UserAccount(UUID.Zero, principalID, firstName, lastName, email,
matricula);
…}}
```

OpenSim.Data/IUserAccountData es donde se lleva a cabo la declaración de la variable “Matricula”, que es solicitada por el proyecto de Diva.

```
public class UserAccountData
    {
        public UUID PrincipalID;
        public UUID ScopeID;
        public string FirstName;
        public string LastName;
        public string Matricula;
        public Dictionary<string, string> Data;
    }
```

Por último, en el proyecto *Diva.Wifi/Handlers/wIFILoginHandler.cs,* es importante validar los datos que llegarán de la página web; por ello se anexa la importación de un dato adicional: la matricula.

```
  byte[] LoginAgent(string resource, IOSHttpRequest httpRequest, Dictionary<string,
object> request)
        {
            string first = String.Empty;
            string last = String.Empty;
            string password = String.Empty;
            string matricula = String.Empty;

            if (!request.ContainsKey("firstname") ||
!request.ContainsKey("lastname") || !request.ContainsKey("password") ||
!request.ContainsKey("Matricula"))
                return WebAppUtils.FailureResult();

            first = request["firstname"].ToString();
            last = request["lastname"].ToString();
            password = request["password"].ToString();
            matricula = request["Matricula"].ToString();
...}
```

Posteriormente a la modificación de los módulos que implicarán el ingreso de un nuevo campo “matricula”, se ejecutan y se sustituyen en el proyecto de diva los módulos afectados:

- Opensim.Services.Interfaces.
- Opensim.Services.UserAccountServices.
- Opensim.Interfaces.UserAccount.
- Opensim.Data.

Se ejecuta nuevamente el proyecto final de *Diva Distribution* en su versión *source.* Además de ello, se anexa en la tabla “useraccounts” (de la BD *Opensim*) el campo *Matricula* como un valor de cadena (*string*).

De forma interna el proyecto permite el ingreso de un nuevo parámetro de identificación: Matrícula.

4.10. Wifi Pages

La inclusión de la matricula en el proyecto originó el cambio de la interfaz web proporcionada por *diva distribution.* En la figura 4.95 se puede visualizar la página de inicio del proyecto de Mentorias y en la figura 4.96 el ingreso del campo matrícula para permitir el registro de usuarios asociados a la BUAP. La interfaz se logró con el uso de lenguajes HTML y PHP.

Figura 4.95 Página principal de registro.

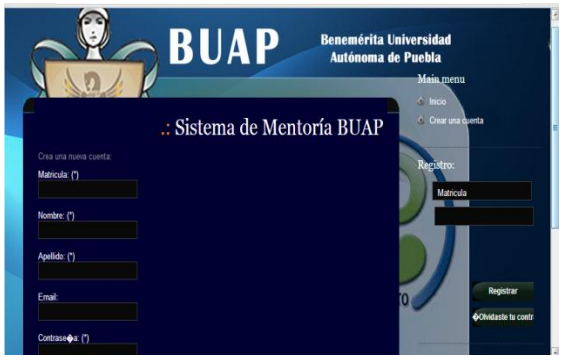


Figura 4.96 Registro de un nuevo usuario.

Capítulo 5. Interfaz final del Sistema de Mentoría

El proyecto desarrollado puede ser consultado en la dirección http://148.228.56.11/Mentoria.html. La figura 5.1, describe la página de registro.

Figura 5.1 Descripción de la página de registro.

1. **Inicio:** Muestra la página principal del proyecto.
2. **Crear una cuenta:** Opción donde los alumnos/catedráticos realizan su inscripción.
3. **Información del mundo virtual:** Indica cuantos usuarios en ese momento se encuentran en el espacio virtual creado.
4. **Login:** Permite a los usuarios registrados acceder a su cuenta.
5. **Registrar:** Botón que solicita datos de acceso.
6. **Zona de descarga:** Para descargar las herramientas necesarias para acceder al mundo virtual.

En la figura 5.2 se puede visualizar el registro de un nuevo usuario; la figura 5.3 confirma el registro.

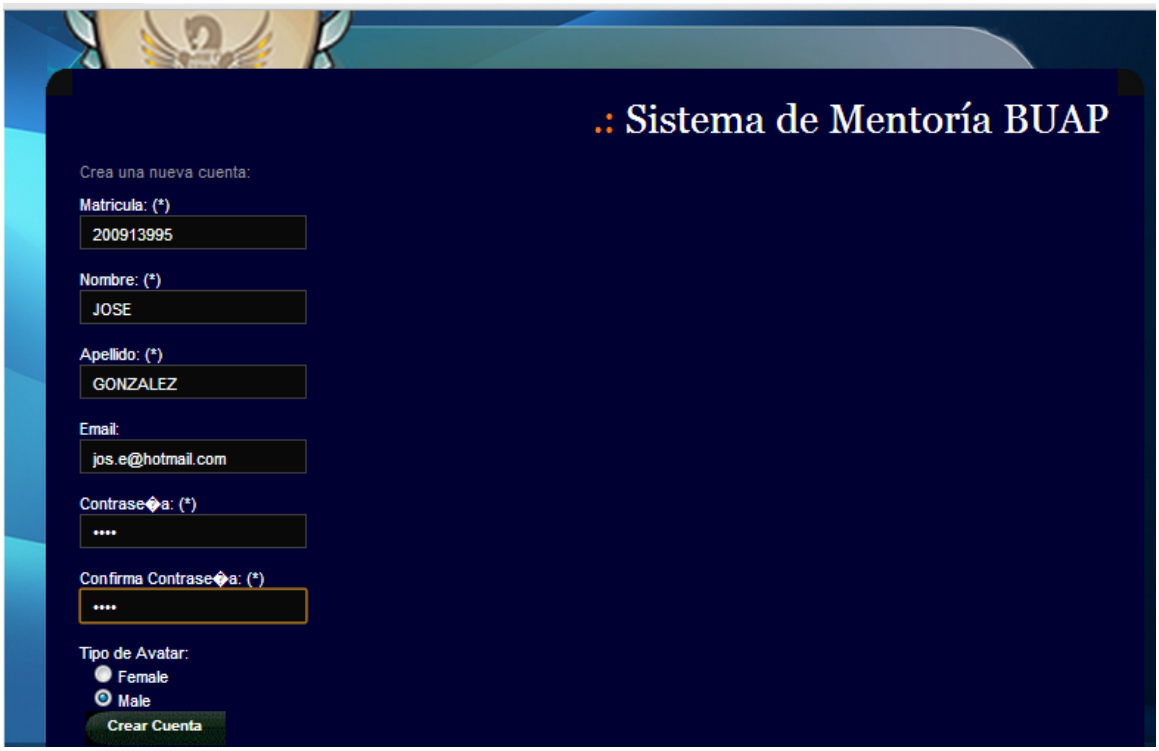


Figura 5.2 Registro de un nuevo usuario.

Figura 5.3 Confirmación de registro.

Una vez registrado, el usuario puede seguir interactuando con la página del mundo virtual, en la figura 5.4 se hace un *login*, y en la figura 5.5, se muestra el resultado de la acción.

Figura 5.4 Acceso de un usuario después del registro.

Figura 5.5 Acceso a la cuenta de un usuario del mundo virtual.

Dentro de la autentificación de la página web, destaca la edición de la cuenta y el uso del inventario del mundo virtual, estas interfaces se visualizan en las figuras 5.6 y 5.7 respectivamente.

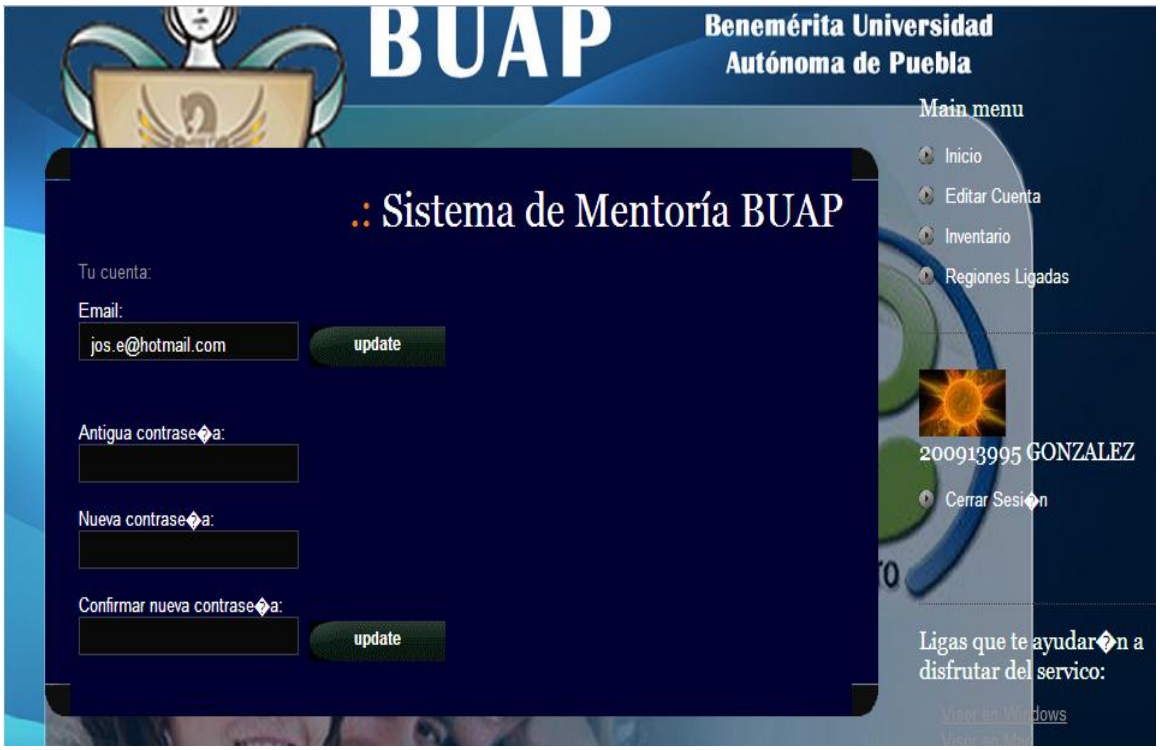


Figura 5.6 Configuración de la cuenta de un usuario creado.

Figura 5.7 Visualización de inventario vía web.

Con el mismo Nombre y Apellido brindados por el sistema en la etapa de registro, es posible establecer la comunicación con el visor, una vez configurado (Capítulo 4.1). En la figura 5.8 se muestra el *acceso a*l visor y en la figura 5.9 el acceso al mundo virtual.

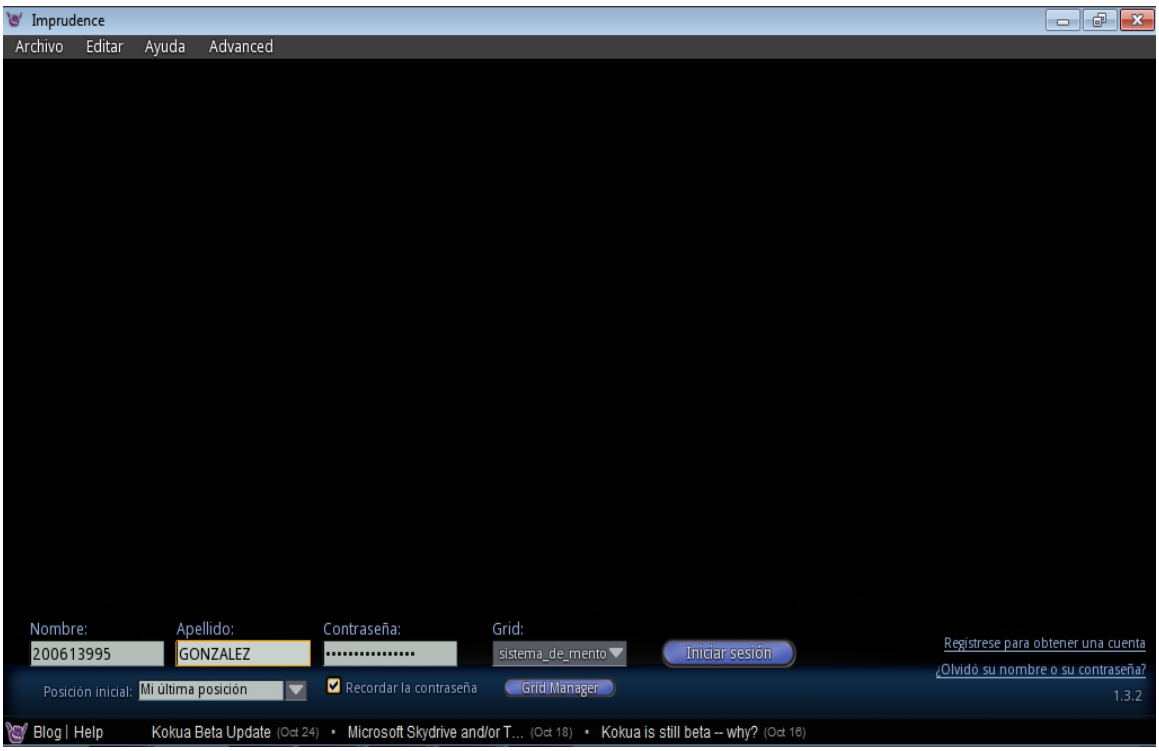


Figura 5.8 Acceso de un nuevo usuario por medio del visor Imprudence.

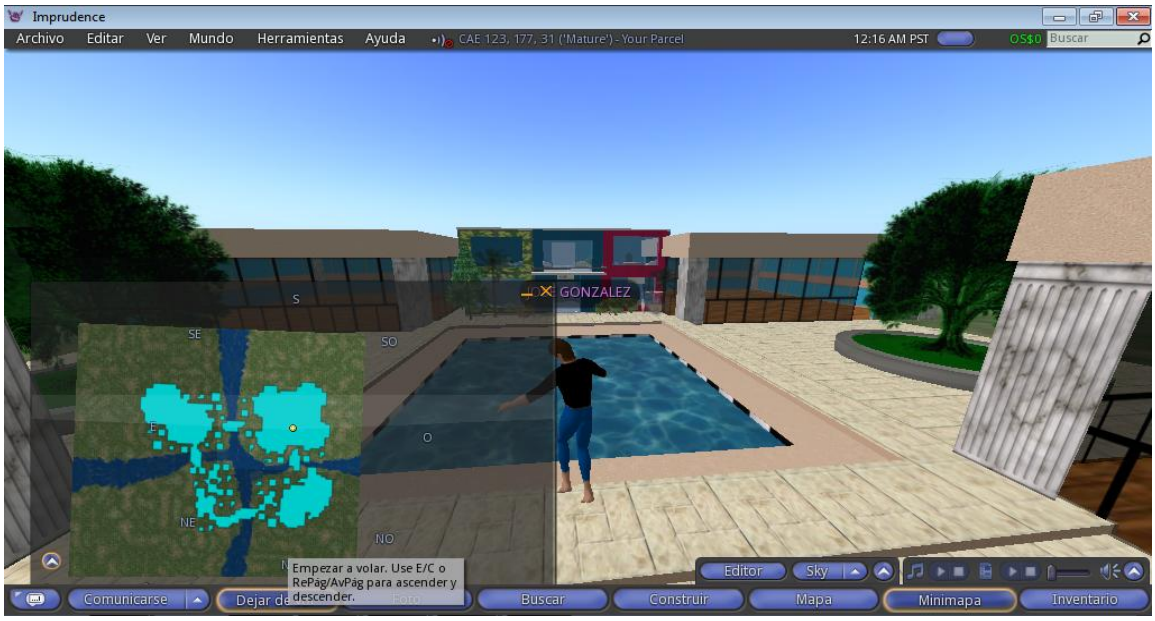


Figura 5.9 Ingreso al mundo virtual.

En la figura 5.10 se presenta el diseño final del Sistema de Mentoría BUAP. Un ambiente virtual 3D.

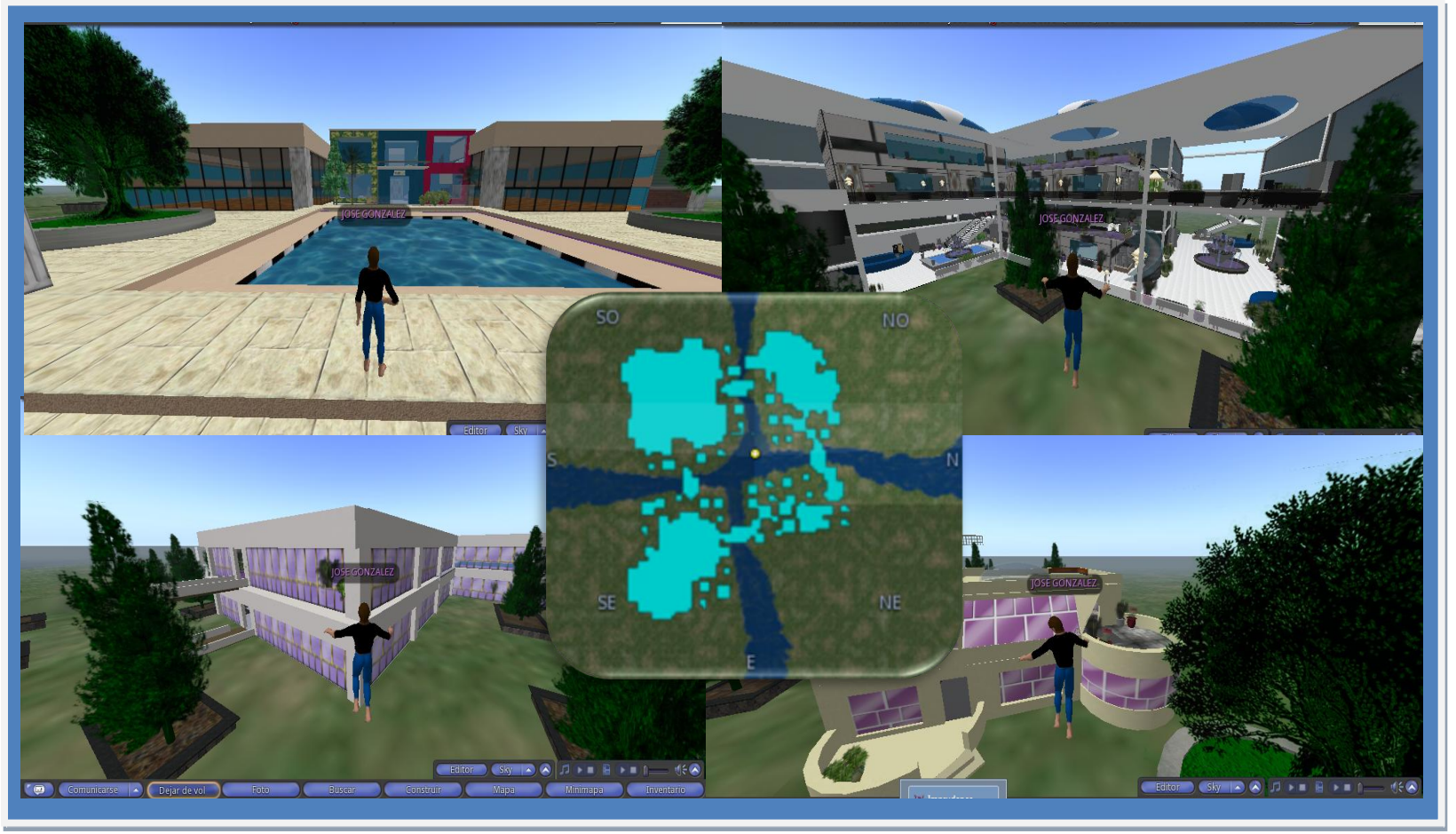


Figura 5.10 Sistema de Mentoría BUAP. Un ambiente virtual 3D.

6. Conclusiones y trabajo futuro.

Se ha presentado el proyecto “Sistema de Mentoría BUAP. Un Ambiente Virtual 3D”, que permite a estudiantes de la BUAP vincularse con los responsables que laboran en las diferentes áreas con que cuenta la Coordinación de Acompañamiento al Estudiante. Éste ambiente virtual 3D de mentorías permite al estudiante crear y personalizar un *avatar* a su gusto, lo cual le ayudará a identificarse como parte del ambiente virtual; facilitando así la tarea de las diferentes coordinaciones para llevar a cabo trámites, información, entre otros servicios hacia los alumnos de manera síncrona, sin la presencia física en el mundo real.

También se desarrolló el área física de las coordinación del CAE en el ambiente virtual; el mundo virtual fue diseñado de acuerdo a las tareas que realizan cada una de las coordinaciones que se encuentran a cargo del CAE; por ejemplo, el área que tiene a su cargo el programa de salud contiene escenarios en blanco, consultorios, entre otros aspectos médicos; lo cual no sólo permitirá identificar a la coordinación, sino también al programa. Todo esto contribuirá a que la atención y/u orientación del estudiante tanto en cuestiones académicas y/o administrativas sea sencilla, ágil y personalizada, sin que el estudiante tenga que trasladarse a Ciudad Universitaria y a cualquier hora del día; lo cual permitirá ahorrar tiempo y dinero. Con esto el estudiante se sentirá acompañado en todo momento y sus dudas y temores se disiparán para hacer su estancia en la BUAP lo más agradable posible, pero sobre todo para que toda su atención se centre en sus labores académicas que finalmente es en dónde se basa su formación profesional e integral.

Finalmente, el contar con un ambiente 3D pondrá a la BUAP a la vanguardia en la utilización y desarrollo de plataformas de apoyo al estudiante, que es el eje central de toda institución educativa.

Como trabajo futuro se plantean hacer enfoque en los siguientes puntos:

- Implementar *scripts* avanzados que ayuden a establecer una interacción más cercana a la realidad.
- Diseño de nuevos espacios de trabajo.
- Modificar el diseño de la página web, ya que el proporcionado por *Diva distribution*, carece de eliminación de usuario.
- Fomentar el uso del mundo virtual.

7. Bibliografía

[1]. **Vicerrectoría de Docencia BUAP,** Coordinación de Acompañamiento al Estudiante. [En línea] Septiembre de 2010. [Citado el: 2012 de Octubre de 30.] http://cmas.siu.buap.mx/portal_pprd/work/sites/Consejo_de_Docencia/resources/PDFContent/213/coordinacion_acompanamiento_estudiante.pdf.

[2]. **Quispe-Otazu Rodolfo,** ¿Qué es la Ingeniería de Software?. [En línea] 13 de Mayo de 2007. [Citado el: 16 de Septiembre de 2012.] http://www.rodolfoquispe.org/blog/que-es-la-ingenieria-de-software.php.

[3]. **Pressman Roger S.,** Ingeniería del Software, Un enfoque práctico. España : Mac Graw Hill, 2005, Sexta Edición.

[4]. **Pressman Roger S.,** Ingeniería del Software: Un enfoque práctico. España : Mac Graw Hill, 1993, Tercera edición.

[5]. **Date C. J.,** Introducción a los SISTEMAS DE BASES DE DATOS. México : Pearson Education, 2001, Séptima edición.

[6]. **Capote Olga Pons,** Introducción a las bases de datos: El modelo relacional. España : Thomson, 2009, Primera edición.

[7]. **Silberschatz Abraham, Korth Henry F., Sudarshan S.,** Fundamentos de Bases de Datos. España : Mc Graw Hill, 2002, Cuarta edición.

[8]. **Cudicio Catherine,** La PNL: Las claves para una mejor comunicación. Barcelona : Ediciones Gestión, 2006.

[9]. **Bavister Steve, Vickers Amanda,** Prolongación Neurolingüistica, las claves para una comunicación más afectiva. Barcelona : AMAT, 2005.

[10]. **Bertolotto Vallés Gustavo,** Programación Neurolingüistica, desarrollo personal. México : Diana, 1996.

[11]. **Cruz-Lara, Samuel.** *Mundos virtuales: Una infraestructura global para facilitar las interacciones sociales multilingües y el aprendizaje de idiomas.* 2012.

[12]. **Bourgeais Laurent, Laude Mathieu.** *REMOTELY INTERACTING WITH A ROBOT ARM IN SECOND LIFE VIA A WEB INTERFACE REPORT.* University of Ulster.

[13]. **Carmona, Edgar Altamirano.** *CERV: Un campus virtual para educación a distancia.* 2007.

[14]. **Vargas, Julian.** Mundo Virtual. [En línea] 06 de Septiembre de 2012. [Citado el: 16 de Septiembre de 2012.] http://mundovirtualdofus.blogspot.mx/.

[15]. **Universidad Autónoma de México, Facultad de Ingeniería, DECDFI.** Instalación y configuración de Second Life. 2011. [Citado el: 15 de Octubre de 2012.] http://campusvirtual.mineria.unam.mx/geometria_analitica/file.php/1/Contenido/Instalacion_y_configuracion_de_Second_Life_para_Open_Sim_DECDF.pdf.

[16]. **Potvin Pablo, Watts Julián, Krasopani Ferran, Perry Erik.** Second Life Argentonia. [En línea] 28 de Abril de 2008. [Citado el: 10 de Octubre de 2012.] http://www.argentonia.com.ar/blog/.

[17]. **Feenan Kevin.** Virtual Worlds - Best Practices in Education. [En línea] 15 de Marzo de 2012. [Citado el: 15 de Octubre de 2012.] http://www.vwbpe.org/.

[18]. **Castro Marianela Fuentes.** Trabajo colaborativo. [En línea] 27 de Agosto de 2010. [Citado el: 16 de Septiembre de 2012.] http://tsdinamicasgrupales.blogspot.mx/2010/08/trabajo-colaborativo.html.

[19]. **Adamski Gamolan.** Ayuda para Second Life. *Imprudence Viewer.* [En línea] 28 de Mayo de 2010. [Citado el: 18 de Octubre de 2012.] http://ayuda-secondlife.blogspot.mx/2010/05/imprudence-viewer.html.

[20]. **3.0 Creative Commons Attribution-Share Alike.** Hippo OpenSim Viewer. [En línea] 26 de Octubre de 2012. [Citado el: 28 de Octubre de 2012.] http://wiki.secondlife.com/wiki/Downloads#Hippo_OpenSim_Viewer.

[21]. **2.5 Attribution-Share Alike.** OpenSimulator. [En línea] 17 de Septiembre de 2012. [Citado el: 12 de Octubre de 2012.] http://opensimulator.org/wiki/Main_Page.

[22]. **1"i"aRincón.** Second Life. [En línea] 21 de Septiembre de 2012. [Citado el: 11 de Octubre de 2012.] http://1iarincon.blogspot.mx/2012/09/second-life.html.

[23]. **© Linden Research Inc,** Second Life. [En línea] [Citado el: 17 de Octubre de 2012.] http://secondlife.com/support/downloads/.

[24]. **(USMP) Universidad de San Martin de Porres,** [En línea] 2011. [Citado el: 16 de Octubre de 2011.] http://www.usmpvirtual.edu.pe/secondlife/index.php.

[25]. **Attribution-Share Alike 2.5,** OpenSimulator. *Diva Distribution.* [En línea] 27 de Septiembre de 2012. [Citado el: 20 de Octubre de 2012.] http://opensimulator.org/wiki/Wifi.

[26]. **Copyright © 2009 Linden Research Inc. Licensed under Creative Commons Attribution-Share Alike 3.0.,** Getting started with LSL. *Getting started with LSL.* [En línea] 06 de Octubre de 2012. [Citado el: 2012 de Octubre de 20.] http://wiki.secondlife.com/wiki/Help:Getting_started_with_LSL#What_is_LSL.3F.

[27]. **Martinez Enrique,** HTML. [En línea] 2010. [Citado el: 22 de Octubre de 2012.] http://www.tutorialesenvideo.net/html/.

[28]. **Informática Argentina,** [En línea] [Citado el: 22 de Octubre de 2012.] http://www.ri5.com.ar/ayuda03.php.

[29]. **Cerezo López Yolanda, Olga Peñalba Rodriguez, Rafel Caballero Roldán.** *Iniciación a la programación en C#, un enfoque práctico.* Madrid : Delta publicaciones, 2007.

[30]. **EcuRed,** [En línea] [Citado el: 2012 de Octubre de 30.] http://www.ecured.cu/index.php/Lenguaje_de_Programaci%C3%B3n_C_Sharp.

[31]. **3.0 Licencia Creative Commons Atribución Compartir Igual y Fundación Wikimedia Inc,** Wikipedia, la enciclopedia libre. *C#.* [En línea] 24 de Octubre de 2012. [Citado el: 2012 de Octubre de 30.] http://es.wikipedia.org/wiki/C_Sharp.

[32]. **Licencia Creative Commons Atribución Compartir Igual 3.0 Fundación Wikimedia, Inc.** MySQL. [En línea] 09 de Octubre de 2012. [Citado el: 30 de Octubre de 2012.] http://es.wikipedia.org/wiki/MySQL.

[33]. **Cobo Angel, Gómez Patricia, Pérez Daniel, Rocha Rocio.** *PHP Y MySQL, Tecnologías para el desarrollo de aplicaicones web.* España : Ediciones Diaz de Santos, 2005.

[34]. **3.0, Licencia Creative Commons Atribución Compartir Igual y Fundación Wikimedia, Inc.** PHP. [En línea] 28 de Octubre de 2012. [Citado el: 30 de Octubre de 2012.] http://es.wikipedia.org/wiki/PHP.

[35]. **2.5 Attribution-Share Alike.** OpenSimulator 0.7 and later. *Configuring Regions.* [En línea] 20 de Abril de 2012. [Citado el: 30 de Octubre de 2012.] http://opensimulator.org/wiki/Configuring_Regions.

[36]. **Benemérita Universidad Autónoma de Puebla.** Programa: Educando para la SALUD, [En línea] 2011. [Citado el: 30 de Octubre de 2012.] http://cmas.siu.buap.mx/portal_pprd/work/sites/cae/resources/PDFContent/56/Educando%20para%20la%20Salud.pdf.

[37]. **Programa de Apoyo Académico a Estudiantes Indígenas (PAAEI)**, [En línea] [Citado el: 30 de Octubre de 2012.]

http://www.buap.mx/portal_pprd/wb/EDUCATIVA/programa_de_apoyo_academico_a_estudiantes_indigena.