CARTA DE AUTORIZACIÓN DE LOS AUTORES

(Licencia de uso)

Bogotá, D.C., 29 de febrero de 2012

Señores

Biblioteca Alfonso Borrero Cabal S.J.
Pontificia Universidad Javeriana
Ciudad

El suscrito:

Darwin René Rodríguez Correa, con C.C. No. 80735319, en mí calidad de autor exclusivo de la obra titulada:

CAPA DE INTEGRACIÓN PARA OGRE 3D EN PS3._____

(por favor señale con una "x" las opciones que apliquen)

Tesis doctoral [ ] Trabajo de grado [X] Premio o distinción: **Si** [ ] **No** [X]

cual:_____ _____

presentado y aprobado en el año 2014 , por medio del presente escrito autorizo a la Pontificia Universidad Javeriana para que, en desarrollo de la presente licencia de uso parcial, pueda ejercer sobre mi (nuestra) obra las atribuciones que se indican a continuación, teniendo en cuenta que en cualquier caso, la finalidad perseguida será facilitar, difundir y promover el aprendizaje, la enseñanza y la investigación.

En consecuencia, las atribuciones de usos temporales y parciales que por virtud de la presente licencia se autorizan a la Pontificia Universidad Javeriana, a los usuarios de la Biblioteca Alfonso Borrero Cabal S.J., así como a los usuarios de las redes, bases de datos y demás sitios web con los que la Universidad tenga perfeccionado un convenio, son:

AUTORIZO	SI	NO
1. La conservación de los ejemplares necesarios en la sala de tesis y trabajos de grado de la Biblioteca.	X	
2. La consulta física o electrónica según corresponda	X	
3. La reproducción por cualquier formato conocido o por conocer	X	
4. La comunicación pública por cualquier procedimiento o medio físico o electrónico, así como su puesta a disposición en Internet	X	
5. La inclusión en bases de datos y en sitios web sean éstos onerosos o gratuitos, existiendo con ellos previo convenio perfeccionado con la Pontificia Universidad Javeriana para efectos de satisfacer los fines previstos. En este evento, tales sitios y sus usuarios tendrán las mismas facultades que las aquí concedidas con las mismas limitaciones y condiciones	X	
6. La inclusión en la Biblioteca Digital PUJ (Sólo para la totalidad de las Tesis Doctorales y de Maestría y para aquellos trabajos		X

De acuerdo con la naturaleza del uso concedido, la presente licencia parcial se otorga a título gratuito por el máximo tiempo legal colombiano, con el propósito de que en dicho lapso mi obra sea explotada en las condiciones aquí estipuladas y para los fines indicados, respetando siempre la titularidad de los derechos patrimoniales y morales correspondientes, de acuerdo con los usos honrados, de manera proporcional y justificada a la finalidad perseguida, sin ánimo de lucro ni de comercialización.

De manera complementaria, garantizo en mi calidad de estudiante y por ende autor exclusivo, que la Tesis o Trabajo de Grado en cuestión, es producto de mi plena autoría, de mi esfuerzo personal intelectual, como consecuencia de mi creación original particular y, por tanto, soy el único titular de la misma. Además, aseguro que no contiene citas, ni transcripciones de otras obras protegidas, por fuera delos límites autorizados por la ley, según los usos honrados, y en proporción a los fines previstos; ni tampoco contempla declaraciones difamatorias contra terceros; respetando el derecho a la imagen, intimidad, buen nombre y demás derechos constitucionales. Adicionalmente, manifiesto que no se incluyeron expresiones contrarias al orden público ni a las buenas costumbres. En consecuencia, la responsabilidad directa en la elaboración, presentación, investigación y, en general, contenidos de la Tesis o Trabajo de Grado es de mí competencia exclusiva,

eximiendo de toda responsabilidad a la Pontifica Universidad Javeriana por tales aspectos. Sin perjuicio de los usos y atribuciones otorgadas en virtud de este documento, continuaré conservando los correspondientes derechos patrimoniales sin modificación o restricción alguna, puesto que de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación de los derechos patrimoniales derivados del régimen del Derecho de Autor.

De conformidad con lo establecido en el artículo 30 de la Ley 23 de 1982 y el artículo 11 de la Decisión Andina 351 de 1993, "Los derechos morales sobre el trabajo son propiedad de los autores", los cuales son irrenunciables, imprescriptibles, inembargables e inalienables. En consecuencia, la Pontificia Universidad Javeriana está en la obligación de RESPETARLOS Y HACERLOS RESPETAR, para lo cual tomará las medidas correspondientes para garantizar su observancia.

NOTA: <u>Información Confidencial:</u>

Esta Tesis o Trabajo de Grado contiene información privilegiada, estratégica, secreta, confidencial y demás similar, o hace parte de una investigación que se adelanta y cuyos resultados finales no se han publicado:

Sí [ ] No [X]

En caso afirmativo expresamente indicaré, en carta adjunta, tal situación con el fin de que se mantenga la restricción de acceso.

NOMBRE COMPLETO	No. Del documento de identidad	Firma
Darwin René Rodríguez Correa	80.735.319	

FACULTAD: Ingeniería

PROGRAMA ACADÉMICO: Ingeniería de sistemas

DESCRIPCIÓN DE LA TESIS DOCTORAL O DEL TRABAJO DE GRADO

FORMULARIO

TÍTULO COMPLETO DE LA TESIS DOCTORAL O TRABAJO DE GRADO			
Capa de integración para OGRE 3D en PS3			
SUBTÍTULO, SI LO TIENE			
AUTOR O AUTORES			
Apellidos Completos	**Nombres Completos**		
Rodríguez Correa	Darwin René		
DIRECTOR (ES) TESIS DOCTORAL O DEL TRABAJO DE GRADO			
Apellidos Completos	**Nombres Completos**		
Flórez Valencia	Leonardo		
FACULTAD			
Facultad de Ingeniería			
PROGRAMA ACADÉMICO			
Tipo de programa (seleccione con "x")			
Pregrado: **X**	Especialización:	Maestría:	Doctorado:
Nombre del programa académico			
Ingeniería de sistemas			
Nombres y apellidos del director del programa académico			
Germán Alberto Chavarro Flórez			
TRABAJO PARA OPTAR AL TÍTULO DE:			
Ingeniero de sistemas			
PREMIO O DISTINCIÓN *(En caso de ser LAUREADAS o tener una mención especial)*:			

CIUDAD	AÑO DE PRESENTACIÓN DE LA TESIS O DEL TRABAJO DE GRADO	NÚMERO DE PÁGINAS
Bogotá	2014	83 Páginas

TIPO DE ILUSTRACIONES (seleccione con "x")						
Dibujos	Pinturas	Tablas, gráficos y diagramas	Planos	Mapas	Fotografías	Partituras
		X				

SOFTWARE REQUERIDO O ESPECIALIZADO PARA LA LECTURA DEL DOCUMENTO

Nota: En caso de que el software (programa especializado requerido) no se encuentre licenciado por la Universidad a través de la Biblioteca (previa consulta al estudiante), el texto de la Tesis o Trabajo de Grado quedará solamente en formato PDF.

MATERIAL ACOMPAÑANTE					
TIPO	**DURACIÓN (minutos)**	**CANTIDAD**	**FORMATO**		
Vídeo			CD	DVD	Otro ¿Cuál?
Audio					
Multimedia					
Producción					
Otro ¿Cuál?					

DESCRIPTORES O PALABRAS CLAVE EN ESPAÑOL E INGLÉS Son los términos que definen los temas que identifican el contenido. (*En caso de duda para designar estos descriptores, se recomienda consultar con la Sección de Desarrollo de Colecciones de la Biblioteca* Alfonso Borrero Cabal S.J *en el correo biblioteca@javeriana.edu.co, donde se les orientará).*	
ESPAÑOL	**INGLÉS**
Consolas de videojuegos	Game consoles
Computación gráfica	Computer graphics
PlayStation 3	PlayStation 3
OGRE3D	OGRE3D
Motor de renderizado 3D	3D rendering engine

RESUMEN DEL CONTENIDO EN ESPAÑOL E INGLÉS
(Máximo 250 palabras - 1530 caracteres)

Este trabajo de grado busca determinar, de ser posible, el uso de la consola de videojuegos PlayStation 3 en aplicaciones de computación gráfica diferentes a la creación de videojuegos, mediante el uso de un motor de renderizado 3D usado típicamente en arquitecturas de hardware convencionales. Tras la investigación de la arquitectura del procesador CellBE creado especialmente para la consola y de las diferencias entre el modelo de programación convencional y el de programación distribuida, se identificaron varias dificultades para implementar una aplicación que permita hacer un uso eficiente del procesador. Aunque la tecnología de la PS3 aún es muy poderosa, las restricciones para su uso han generado un estancamiento en la creación de sistemas operativos y herramientas de desarrollo que permitan crear aplicaciones eficientes para la PS3.

This work seeks to determine, if possible, the use of the PlayStation 3 console in computer graphics applications different than videogames, using a 3D rendering engine typically used in conventional hardware architectures. After research about CellBE processor architecture created especially for the PS3 console and the differences between the conventional programming model and distributed programming, we identified several difficulties in implementing an application that makes efficient use of the processor. Although the PS3 technology is still very powerful, use restrictions have led to inactivity in the development of operating systems and development tools for building efficient applications for the PS3.

<CIS1330TK02>

CAPA DE INTEGRACIÓN PARA OGRE 3D EN PS3

DARWIN RODRIGUEZ CORREA

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERIA DE SISTEMAS
BOGOTÁ, D.C.
2013

<CIS1330TK02>

CAPA DE INTEGRACIÓN PARA OGRE 3D EN PS3

Autor:
DARWIN RODRIGUEZ CORREA

MEMORIA DE TRABAJO DE GRADO REALIZADO PARA CUMPLIR UNO DE LOS REQUISITOS PARA OPTAR AL TÍTULO DE INGENIERO DE SISTEMAS

Director:
Leonardo Flórez Valencia

Jurados del Trabajo de Grado:
Andrea Del Pilar Rueda
Christian Benavides

Página web del Trabajo de Grado:
http://pegasus.javeriana.edu.co/~CIS1330TK02/

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
CARRERA DE INGENIERIA DE SISTEMAS
BOGOTÁ, D.C.
2013

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA DE SISTEMAS

Rector Magnífico
Joaquín Emilio Sánchez García S.J.
Decano Académico Facultad de Ingeniería
Ingeniero Jorge Luis Sánchez Téllez
Decano del Medio Universitario Facultad de Ingeniería
Antonio José Sarmiento Nova S.J.
Director de la Carrera de Ingeniería de Sistemas
Ingeniero Germán Alberto Chavarro Flórez
Director Departamento de Ingeniería de Sistemas
Ingeniero Rafael Andrés González Rivera

Artículo 23 de la resolución no. 1 de junio de 1946

"La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado. Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque no contengan ataques o polémicas puramente personales. Antes bien, que se vean en ellos el anhelo de buscar la verdad y la Justicia"

AGRADECIMIENTOS

Agradezco a la Universidad por brindarme la oportunidad no solamente de laborar sino también de estudiar y crecer como persona. El proceso ha sido largo y difícil pero muy satisfactorio y constructivo. Durante los últimos años la Universidad no solo ha sido mi lugar de trabajo sino mi fuente de conocimiento, una gran entidad que me ha abierto las puertas en todos los sentidos, permitiéndome finalizar mi carrera y mejorando mi calidad de Vida.

Agradezco a todas las personas que han estado involucradas en este proceso, a todos los ingenieros pertenecientes al Departamento de Ingeniería de Sistemas, quienes me han dado siempre su apoyo y han estado dispuestos a enseñarme y corregirme cuando lo he necesitado. En particular agradezco al Ingeniero Leonardo Flórez por acceder a dirigir este trabajo de grado y todo el apoyo y enseñanzas impartidas durante el proceso.

De igual manera agradezco al personal administrativo que me ha colaborado con los trámites respectivos, en particular a Maida Urrego, secretaria de la Carrera de Ingeniería de sistemas, pues sin sus consejos, apoyo e impulso no habría logrado terminar este pregrado.

Finalmente agradezco a mi madre quien ha sido desde siempre la gestora de este largo proceso, quien siempre ha creído en mis capacidades y ha hecho todo lo posible por ayudarme en la consecución de mis metas, este trabajo de grado está dedicado a ella.

Tabla de contenido

Tabla de ilustraciones

Tabla de tablas

ABSTRACT

This work seeks to determine, if possible, the use of the PlayStation 3 console in computer graphics applications different than videogames, using a 3D rendering engine typically used in conventional hardware architectures. After research about CellBE processor architecture created especially for the PS3 console and the differences between the conventional programming model and distributed programming, we identified several difficulties in implementing an application that makes efficient use of the processor. Although the PS3 technology is still very powerful, use restrictions have led to inactivity in the development of operating systems and development tools for building efficient applications for the PS3.

RESUMEN

Este trabajo de grado busca determinar, de ser posible, el uso de la consola de videojuegos PlayStation 3 en aplicaciones de computación gráfica diferentes a la creación de videojuegos, mediante el uso de un motor de renderizado 3D usado típicamente en arquitecturas de hardware convencionales. Tras la investigación de la arquitectura del procesador CellBE creado especialmente para la consola y de las diferencias entre el modelo de programación convencional y el de programación distribuida, se identificaron varias dificultades para implementar una aplicación que permita hacer un uso eficiente del procesador. Aunque la tecnología de la PS3 aún es muy poderosa, las restricciones para su uso han generado un estancamiento en la creación de sistemas operativos y herramientas de desarrollo que permitan crear aplicaciones eficientes para la PS3.

INTRODUCCIÓN

La PlayStation 3 ha revolucionado el mercado de consolas de videojuegos durante los últimos 7 años; su mayor potencial está centrado en el procesador CellBE que fue creado por Sony en conjunto con IBM y Toshiba, con la idea de superar la capacidad de cómputo de los procesadores típicos de la época.

El CellBE cuenta con una arquitectura diferenciadora que le permite procesar en paralelo las tareas optimizando el tiempo de CPU y agilizando los cálculos aritméticos. Cuenta con 7 SPE y un PPE, aunque en la PS3 tan solo 6 SPE son accesibles, también cuenta con una memoria interna de 256MB; una configuración atractiva para los desarrolladores de aplicaciones de computación gráfica, por el desempeño que ofrece, aun con su restricción de memoria.

La PS3 se ha empleado en investigación, por su gran poder de procesamiento, en aplicaciones diferentes al diseño de videojuegos, permitiendo el uso de esta tecnología en áreas diferentes a la computación gráfica obteniendo muy buenos resultados.

Siendo la PS3 una computadora diseñada específicamente para el uso de videojuegos, utilizar su configuración de hardware para crear programas que usen un motor de renderizado 3D, supone una combinación adecuada para la creación de aplicaciones de computación gráfica de alto desempeño.

Este trabajo de grado procura hacer un aporte, permitiendo el uso de un motor de renderizado 3D en la consola PS3 conocido como OGRE 3D, con el objetivo de permitir la ejecución de aplicaciones de computación gráfica que requieran el poder de dicha aplicación y pretendiendo aprovechar el desempeño que ofrece el procesador CellBE, así mismo, permitir el uso del control DualShock 3 propio de la PS3, para facilitar la interacción del usuario con este tipo de aplicaciones.

I - DESCRIPCION GENERAL DEL TRABAJO DE GRADO

1. Oportunidad, Problemática, Antecedentes

1.1. Descripción del contexto

Las técnicas de Computación Gráfica se aplican para generar y visualizar imágenes que ayudan a expertos en diferentes áreas en la realización de sus labores[1]. Los investigadores en el área de la Computación Gráfica enfrentan grandes retos en cuanto a desempeño y eficiencia. Aun cuando se logren generar algoritmos eficientes, en el trabajo de visualización se requiere de una alta capacidad de procesamiento de la información[2].

El modelo Medical Interactiva visualization Tool (MIsT)[3] y el simulador VRLS[4], desarrollados en el grupo de investigación Takina, hacen uso del motor gráfico OGRE 3D [5] y la librería ITK[6], para implementar algoritmos que mejoran la visualización de imágenes médicas, en el caso de MIsT, y para generar un entorno virtual de aprendizaje de procedimientos quirúrgicos, en el caso de VRLS.

Las características de hardware de la consola de videojuegos PlayStation3 (PS3)[7] ofrece beneficios que podrían potencializar las funcionalidades de MIsT y VRLS entre otras aplicaciones creadas con OGRE 3D. Sin embargo, estas aplicaciones han sido probadas en computadores de escritorio regulares, por lo que se plantea diseñar una capa que permita desplegar OGRE3D en la consola.

1.2. Justificación

OGRE 3D es un motor de renderizado 3D utilizado en diversas aplicaciones de computación gráfica, principalmente en el diseño de videojuegos. Recientemente en el grupo de investigación Takina, se han desarrollado aplicaciones que utilizan OGRE 3D para realizar procesamiento de imágenes médicas, tarea que requiere una alta capacidad de procesamiento y el uso de periféricos diferentes a teclado y mouse.

La consola cuenta con un procesador CellBE[8] que ofrece un enfoque arquitectónico innovador, pues tiene un diseño radicalmente diferente a los ofrecidos por otros procesadores multinúcleo. Esta

arquitectura tiene un gran potencial para mejorar la velocidad, el rendimiento y la eficiencia energética de muchas aplicaciones de alto desempeño, incluyendo aplicaciones gráficas [2]. Además cuenta con una tarjeta de video NVIDIA, que puede ser utilizada para el cálculo numérico, junto con el procesador CellBE[9], logrando optimizar la visualización de imágenes pesadas.

Teniendo en cuenta que tanto MIsT como VRLS usan OGRE 3D para la visualización pesada de imágenes médicas[3][4], implementar este modelo en una consola de juegos con alta tecnología para el procesamiento de imágenes, resulta conveniente y se ajusta a la solución propuesta en estos proyectos.

1.3. Impacto Esperado

El impacto esperado era transportar OGRE 3D a la consola PlayStation 3, aprovechando el uso de su procesador para mejorar el desempeño de las aplicaciones allí ejecutadas, así como facilitar la interacción de los usuarios mediante el uso del *gamepad* propio de esta consola. Se esperaba que a largo plazo esta capa se utilizara en el desarrollo de videojuegos y otras aplicaciones de Computación gráfica directamente sobre la consola.

2. Descripción del Proyecto

2.1. Visión global

Este trabajo de grado pretende, de ser posible, transportar el motor gráfico OGRE 3D y la librería ITK a la consola de video juegos PlayStation 3 para que aplicaciones desarrolladas con estas herramientas, como el modelo MIsT y el simulador VRLS puedan ser ejecutados aprovechando los beneficios que ofrece la configuración de hardware de la consola.

A continuación la pregunta que da lugar a ésta propuesta de trabajo de grado:

¿Cómo implementar el modelo Medical Interactive visualization Tool (MIsT), desarrollado en el grupo de investigación Takina, usando la tecnología de la PS3?

2.2. Objetivo general

Implementar una capa de interpretación que permita hacer uso del modelo Medical Interactive visualization Tool (MIsT), en la consola de videojuegos PlayStation3.

2.3. Fases Metodológicas o conjuntos de objetivos específicos

Las fases metodológicas que se encuentran a continuación muestran los objetivos específicos que abarcan y el número correspondiente a cada uno.

2.3.1. Investigación teórica y de tecnología

- Caracterizar el estado del arte acerca del desarrollo en la consola PS3 con OGRE 3D.

2.3.2. Exploración y Planeamiento

- Diseñar una capa de interpretación de OGRE 3D en la consola PS3.

2.3.3. Producción y Muerte

- Implementar la capa diseñada anteriormente.
- Validar la capa desarrollada, ejecutando el prototipo desarrollado con MIsT.

2.4. Método que se propuso para satisfacer cada fase metodológica

2.4.1. Investigación teórica y de tecnología:

Durante la primera Fase se realizó una revisión de la bibliografía que permitió conocer la estructura del procesador CellBE, los diferentes sistemas operativos diseñados para éste procesador y los procedimientos para instalarlos en la consola. Esta revisión de fuentes también permitió definir las herramientas necesarias para generar un ambiente de desarrollo adecuado en la PS3.

Posteriormente se indagó acerca de otros intentos previos por ejecutar OGRE 3D en la PS3 permitiendo estudiar su arquitectura, qué prerrequisitos requiere para su funcionamiento y las dificultades para ejecutar aplicaciones realizadas con OGRE 3D en la consola.
Como resultado se enriqueció el estado del arte que se caracterizó desde la construcción del anteproyecto (ver Sección II Marco Teórico).

2.4.2. Exploración:

Para las siguientes fases se emplearon algunas herramientas de Extreme Programming, una la metodología de desarrollo ágil que permite comprobación constante con el cliente (en este caso el Director de trabajo de grado), sobre los avances realizados y los cambios pertinentes sobre la marcha del tiempo [10][11].

Durante la Exploración se definieron los requerimientos, mientras se preparaba la consola para realizar la instalación del Sistema Operativo [12] [13] (ver Sección 2.1. Levantamiento de requerimientos).

2.4.3. Planeamiento:

En esta fase se priorizaron los requerimientos asignando a cada uno un puntaje que representa la importancia de cumplimiento. Esta priorización se realizó en conjunto con el director de trabajo de grado (ver Sección 3.1 Priorización de Requerimientos). De la misma manera y teniendo en cuenta las restricciones y los requerimientos priorizados, se determinó el alcance del proyecto (ver Sección 3.2 Alcance). Posteriormente se realizó el diseño arquitectónico [12] [13] que satisfacía el alcance determinado en el numeral anterior, utilizando el modelo 4+1Vistas de Philippe Kruchten[14] (ver Sección 3.3 Diseño Arquitectónico).

2.4.4. Producción:

En esta fase se realizaron pruebas en la consola ejecutando ejemplos sencillos con OpenGL y OGRE 3D (ver Sección 4.1 Pruebas)

De Igual manera, siguiendo el diseño, se generó un prototipo que permite hacer uso del control Dual Shock de la PS3 en una aplicación desarrollada con OGRE 3D (ver Sección 4.2 Prototipo PS3Controller).

2.4.5. Muerte:

En esta fase se hace entrega de la documentación del proyecto que describe los artefactos generados en las fases anteriores [12] [13].

II - MARCO TEÓRICO

1. Marco Contextual

1.1. Procesador CellBE

La consola de videojuegos PlayStation 3 cuenta con un procesador CellBE, este es una CPU de 8 núcleos asimétricos. Se compone de un elemento de procesamiento de energía (PPE), y 7 elementos de procesamiento sinérgicos (SPE). Cada uno de estos elementos corre a 3.2GHz y están conectados mediante un Bus de interconexión de 4 elementos (EIB), capaz de un rendimiento máximo de 204.8GB/s. Cada elemento de procesamiento en el bus tiene su propio controlador de acceso directo a memoria (DMA). Otros elementos en el bus son el controlador de memoria (RAM de 256MB XDR), y dos controladores de E/S flexibles [15].

A continuación se describen los componentes más importantes dentro del procesador:

- **PPE:** es un procesador doble hilo de 64 bits con una unidad VMX de 32 registros. Cuenta con una caché L1 de 32 Kb para instrucciones, una caché L1 de 32 Kb para datos y una caché L2 de 512 Kb. El PPE cuenta con un procesador multihilo llamado PPU que provee dos unidades de ejecución a la capa de software.

  A pesar de su gran velocidad, el propósito general del PPE es servir como controlador y supervisor de los cores en el procesador, es decir de los SPE.

- **SPE:** Son procesadores vectoriales (SIMD) independientes que corren a 3.2Ghz cada uno. Al ser vectoriales (SIMD), permiten ejecutar la misma instrucción sobre un conjunto de varios datos en el mismo ciclo de reloj.

  Cada SPE cuenta con un SPU, un MFC y un LS. Los SPE no pueden acceder a la memoria principal directamente por lo que tanto el código como los datos deben ser almacenados en su LS; toda información que necesite el SPE que esté almacenada en memoria principal, debe ser cargada explícitamente por software al LS respectivo mediante una operación DMA.

- **EIB:** puede transmitir 96 bytes por ciclo, para un ancho de banda de 204,8 Gb/s, lo que permite tener más de 100 solicitudes DMA pendientes. El EIB consta de cuatro anillos unidireccionales, dos en cada dirección.

El PPE se diferencia de los SPE por su forma de acceder a la memoria principal y por su capacidad de procesamiento. Mientras el PPE es más rápido en asignación de tareas, los SPE son más veloces procesándolas. EL PPE accede a la memoria principal de manera directa, haciendo uso de sus cachés L1 y L2. En la siguiente figura se muestra la arquitectura del procesador Cell:

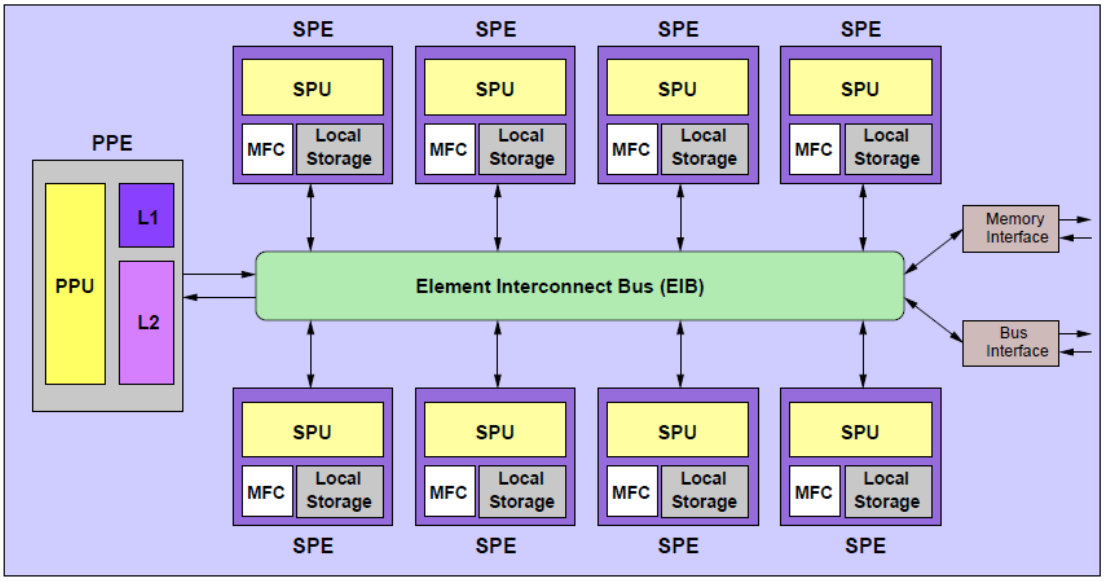

Ilustración 1: Arquitectura del Procesador CellBE[2]**.**

1.2. Trabajos Importantes en el área

1.2.1. Unleashing the Power of the PlayStation 3 to Boost Graphics Programming[2]

Este trabajo aprovecha la arquitectura del procesador CellBE para manejar eficientemente problemas de rendimiento. Se centra en las diferencias entre programar para un procesador regular y programar utilizando los cores del procesador CellBE, dando algunos ejemplos de programas sencillos en C++ haciendo uso del PPE y de los SPE.

Algunos de los ejemplos que incluye:

- **Eliminación de saltos dinámicos**

  La arquitectura SPE no incluye predicción de saltos dinámicos. Por lo tanto, como los saltos dinámicos son relativamente caros para el procesador CellBE, a veces es importante

eliminar las condiciones de rendimiento crítico en el código. El secreto para eliminar los saltos es aprovechar la instrucción *select bits*, es decir, una sentencia *if-then-else* puede ser eliminada mediante el procesamiento de ambas cláusulas y el uso de los bits de selección para elegir el resultado como una función condicional.

- **Programación del PPE**

  Los SPE y el PPE están diseñados para ejecutar diferentes tipos de código. El núcleo SPE utiliza un tipo de instrucción y registro establecido en el núcleo PPE, por lo que los programas escritos para el PPE y el SPE deben ser procesados por diferentes compiladores.

  Un conjunto de extensiones de lenguaje C están disponibles para la programación del PPE. Estas extensiones incluyen otros tipos de datos vectoriales y un amplio conjunto de comandos escalares y vectoriales, llamados *intrinsics*.

- **Programación de los SPE**

  Un conjunto de extensiones del lenguaje C, también llamados *intrinsics*, están disponibles para la programación de los SPE. Estas extensiones proporcionan un control explícito de las instrucciones SPE. Hay 204 instrucciones del SPU llamadas Instruction Set Architecture (ISA), y se agrupan en 11 clases diferentes, según su funcionalidad.

  Para el desarrollo de código sobre el procesador Cell, se recomienda seguir estos pasos:

  - Ejecutar el código principal en el PPE.
  - Paralelizar el código para los SPE; mantener el acceso a memoria secuencial y no utilizar vectorización todavía.
  - Vectorizar el código de los SPE.
  - Preparar correctamente los datos a procesar en los SPE; alineación de los datos.
  - Incluir *overlapping* en las transferencias a los DMA.
  - Evitar saltos dinámicos en las rutas críticas.

  Otras sugerencias de programación que pueden mejorar el desempeño:

  1. Asignar tanto trabajo como sea posible a los SPE, el PPE comanda la ejecución en los SPE así que es realmente el controlador de los procesos.
  2. Particionar el trabajo y diseñar una estrategia de asignación que minimice las operaciones atómicas y la sincronización. Esto ayuda al paralelismo utilizando los SPE para particionar las tareas.

3. Acomodar las posibles diferencias entre tipos de datos. Ayuda a determinar qué datos se pueden procesar en el PPE y cuáles en los SPE.
4. Explotar el *multi-threading* que ofrece el PPE.
5. Diseñar accesos eficientes a las estructuras de datos. Así se logra un acceso eficiente a los SPE.
6. Inicializar los DMA de los SPE. Es mejor que los SPE extraigan los datos del PPE utilizando el MFC para iniciar la DMA. Esto se hace para evitar cuellos de botella en el PPE, y porque el número de ciclos para iniciar una transferencia a los SPE es más pequeño que el número de ciclos para iniciar la transferencia a partir del PPE.
7. Mantener los datos en el procesador tanto como sea posible, así se reduce el acceso a la memoria principal y se utilizan los LS tanto como el programa lo permita.
8. Evitar instrucciones escalares externas en los SPE.
9. Evitar loops. Se deben retirar todos los circuitos en los que el número de iteraciones es constante.
10. Evitar multiplicación de enteros. Se debe evitar la ejecución de multiplicaciones de enteros de 32 bits, ya que el SPE sólo contiene un multiplicador de 16x16 bits.
11. Evaluar el procesamiento vs. el uso de resultados pre-calculados. Una estrategia común que muchos programadores utilizan para mejorar el rendimiento de la aplicación es hacer llamadas a tablas de valores pre calculados, pero estos no son eficaces en la programación de los SPE, ya que consumen espacio valioso en las LS. Si es posible, es preferible calcular los valores de nuevo en lugar de la leer de la memoria.
12. Diseño de almacenamiento local limitado. El almacenamiento local en los SPE está limitado a 256 KB, para instrucciones de programa, pilas de funciones, estructuras de datos locales y buffers DMA.

    En las secciones siguientes, se explica cómo preparar la consola para poder compilar y ejecutar código. Inicialmente es necesario instalar un sistema operativo diferente al que trae por defecto la PS3, en este caso Linux y se hace uso del SDK de IBM para el CellBE.

1.2.2. A Rough Guide to Scientific Computing On the PlayStation 3[8]

Este trabajo describe la arquitectura de la consola PS3 y muestra el procedimiento necesarios para acceder al procesador CellBE y una guía de instalación del SDK[16] que permite programar sobre la consola. Posteriormente explica algunas técnicas de

programación adecuadas para la arquitectura del procesador que se describen a grandes rasgos a continuación:

- **Short Vector SIMD-ization:**

El poder del procesador está en los SPE, estos son procesadores vectoriales en sí mismos, con capacidad de procesar múltiples datos en un mismo ciclo de reloj. El objetivo de vectorizar es eliminar las operaciones escalares, es decir eliminar las operaciones de inserción y extracción de elementos de un vector para operarlo, por esta razón no es óptimo usar código escalar estándar cuando se programan operaciones para un SPE.

Lo ideal en este caso es ajustar el código gradualmente haciendo uso de los *intrinsics* que provee el SDK. Para tal efecto se propone una guía de pasos a seguir:

1. Iniciar escribiendo el código de manera escalar estándar en C.
2. Gradualmente introducir los *intrinsics*, se puede mezclar código escalar estándar y vectorización.
3. Intentar eliminar las operaciones escalares reemplazando extracciones e inserciones por operaciones de vectorización (shifts, rotations, shuffles).
4. Abrir los bucles, este paso requiere mayor experiencia y esfuerzo, pero siempre mejora el desempeño de las implementaciones.

- **Intra-Chip Communication:**

El procesador CellBE proporciona tres métodos básicos de comunicación:

1. DMA transfers: los medios más importantes de comunicación en el procesador facilitando tanto sincronización como transferencia de grandes volúmenes de datos. Estos mecanismos permiten envío de mensajes tanto entre SPE como entre PPE y SPE, sin embargo tiene prioridad por los mensajes de los SPE.
2. Mailboxes: permiten envío de mensajes tanto entre SPE como entre PPE y SPE, mediante una cola FIFO. Cada SPE tiene 4 entradas para recepción y dos salidas para envío de mailbox.
3. Signals: no se describen en este trabajo.

Este trabajo también describe las distribuciones de Linux disponibles para ejecutar en la consola: Yellow Dog 5, Fedora Core 5 y Gentoo PowerPC 64 edition, así como los compiladores recomendados: GCC y XLC.

Es de resaltar la recopilación de diferentes proyectos que se han realizado sobre la consola a nivel de programación, que realiza este trabajo y que pueden ser de gran ayuda a la hora de abordar cualquier proyecto de desarrollo, a continuación se listan:

- **CorePy:** es una librería desarrollada en la Universidad de Indiana, que permite crear código para el procesador CellBE en lenguaje Pyton.
- **Octopiler:** es un compilador creado por IBM que automatiza el paralelismo (SIMD'ization), también permite generar el código tanto para el PPE como para los SPE, entre otras características que permiten optimizar el código para mejorar el performance de la implementación.
- **RapidMind:** provee una plataforma de desarrollo que ofrece diversas extensiones en C y C++ para el desarrollo de aplicaciones que corren en GPU y procesadores Cell.
- **PeakStream:** es una plataforma de desarrollo de aplicaciones diseñada para maximizar el poder de punto flotante de los procesadores multi-núcleo, tales como la familia de procesadores x86 y GPU. Aunque el procesador CellBE no se admite, el modelo es bastante aplicable a la programación de este tipo de procesador.
- **Lenguaje Sequoia:** es un proyecto de investigación realizado en la Universidad de Stanford que incluye un compilador para el procesador Cell.
- **Mercury Multicore Framework:** es un API para programación sobre el procesador CellBE que explota el paralelismo.

1.2.3. Early Experiences with Algorithm Optimizations on Clusters of PlayStation 3 [17]

Este trabajo documenta el experimento que llevó a cabo la Fuerza Aérea de los Estados Unidos utilizando un cluster de PS3 como un programa de modernización de los sistemas de computación de alto rendimiento.

Un primer caso de estudio fue utilizar el Cluster de PS3 para realizar investigación en arquitecturas informáticas neuromórficas. Un segundo caso de estudio empleó 3 PS3 para procesar video HD de 1080p en escala de grises. Finalmente presenta un tercer caso de estudio sobre procesamiento en tiempo real de un algoritmo de retroproyección para un radar SAR.

1.2.4. Limitations of the PlayStation 3 for High Performance Cluster Computing[18]

Este trabajo describe de manera detallada la arquitectura del CellBE mencionando el rol que cumple cada uno de los elementos que lo componen y menciona algunas técnicas de programación para obtener el mejor rendimiento del mismo.

Finalmente tras un par de experimentos, que enfrentan el desempeño teórico contra el desempeño real del procesador, lista una serie de limitaciones que tiene la consola:

- **Tasa de acceso a Memoria Principal:** las unidades de ejecución pueden generar resultados de coma flotante a una velocidad que es mucho mayor que la velocidad a la que la memoria puede alimentar datos a las unidades de ejecución. La memoria principal del procesador CellBE es capaz de conseguir una velocidad de transferencia máxima 25.6 GB/s, mientras que cada SPE tiene un rendimiento máximo del 25,6 Gflop/s en precisión simple. Para un valor de 4 bytes de largo, de un solo punto flotante de precisión, un solo SPE tiene que realizar cuatro operaciones de punto flotante con el fin de ocultar la comunicación. Si 6 SPE operan simultáneamente 24 operaciones se deben realizar en cada valor de precisión simple con el fin de ocultar la comunicación.
- **Velocidad de conexión a red:** la PlayStation 3 está equipada con una tarjeta de red Gigabit Ethernet. La capacidad de esta interconexión esta fuera de balance si se compara con el rendimiento teórico máximo de cada nodo. Sólo las aplicaciones computacionalmente intensivas pueden beneficiarse de la conexión de múltiples PS3 juntas para formar un clúster.
- **Tamaño de la Memoria Principal:** la PlayStation 3 está equipada con sólo 256MB de memoria principal. Esto representa una seria limitación cuando se combina con la lentitud de la interconexión de red (para el caso de clusters).
- **Rendimiento de precisión doble:** el máximo rendimiento en precisión doble aritmética de punto flotante, es un factor de 14 por debajo del máximo rendimiento de precisión simple. Los cálculos que exigen la máxima precisión verán un máximo rendimiento de sólo 14,4 Gflop/s (casi 11 Gflop/s en la PS3).
- **Paradigma de programación:** Escribir código eficiente y rápido para el procesador CellBE es una tarea difícil ya que requiere un profundo conocimiento de la arquitectura del procesador, del entorno de desarrollo, y un poco de experiencia.

La conclusión es que el atractivo de la PlayStation 3 para cálculos científicos dependerá de las características de la aplicación. Es necesario tener en cuenta las limitaciones mencionadas anteriormente para optimizar el código y así obtener el mejor resultado posible.

1.3. Experiencias previas

1.3.1. MIsT

Medical Interactiva Visualization Tool[3], es un modelo arquitectónico que permite hacer uso de un motor de videojuegos para la visualización de imágenes médicas pesadas mediante la implementación de una capa de comunicación entre OGRE 3D, ITK y VTK.

VTK es un kit de herramientas de visualización de distribución libre, que consiste en una serie de librerías para C++, Pyton y Java. Es compatible con una amplia variedad de algoritmos de visualización incluyendo: escalar, vector, tensor, textura, métodos volumétricos y técnicas de modelado avanzadas [19].

ITK es un toolkit de distribución libre, implementada en C++ y disponible para cualquier plataforma. ITK realiza el proceso de identificación y clasificación de datos que se encuentran en una imagen digital. MIsT hace uso de ITK para realizar la lectura de las imágenes que posteriormente se visualizan con OGRE 3D[6].

OGRE 3D se utiliza para visualizar las imágenes que se cargan previamente con ITK mediante un sistema de partículas o mallas. La arquitectura de MIsT permite que una imagen sea cargada con ITK y se guarde la información correspondiente a cada pixel, esta información luego es utilizada por OGRE 3D para su representación en pantalla. Otra parte del modelo permite generar mallas a partir de la imagen cargada con ITK y posteriormente visualizarla con mallas de OGRE 3D, sin embargo la cantidad de información que posee una imagen médica es mayor a la que OGRE 3D puede soportar, aun así es posible reducir el nivel de detalle en la visualización de la imagen a visualizar mediante un filtro de decimación que reduce el número de caras a mostrar. Se partirá de esta limitación de memoria que tiene OGRE 3D y que representa un factor importante a tener en cuenta, para diseñar la capa que interpretará el motor en la PS3.

1.3.2. VRLS

Virtual Reality Laparoscopic Simulator, es un simulador de laparoscopias desarrollado con OGRE 3D e ITK que permite el uso del dispositivo conocido como *PhantomOmni*. Esta

aplicación se ha desarrollado con el objetivo de apoyar el entrenamiento de los residentes de cirugía en laparoscopia digestiva[4].

VRLS utiliza ITK para cargar mallas que modelan los objetos, una vez cargadas las mallas se generan objetos tipo *entity* de OGRE 3D para simular los órganos y los instrumentos quirúrgicos que componen el simulador.

La interacción con el simulador se realiza gracias a un dispositivo llamado *PhantomOmni*, mediante el uso de las librerías *OpenHaptics* y *VRPN*, este dispositivo permite simular físicamente las herramientas que se utilizan para realizar la laparoscopia y permite tener retroalimentación física por parte de la aplicación, por ejemplo en el caso de colisiones con los órganos. Otros dispositivos de interacción con el simulador son el teclado y el mouse, estos son utilizados gracias la herramienta *OIS*, sin embargo estos dispositivos no permiten retroalimentación al usuario como el *PhantomOmni.*

1.3.3. OGRE 3D

Algunos intentos por hacer uso de OGRE 3D en la consola PS3 se evidencian en el foro de soporte de dicho motor. Tras la investigación realizada se concluye que no se ha adelantado una estudio profundo sobre el tema, simplemente intentos de programadores por hacer un *port* del motor hacia la consola.

La única implementación exitosa encontrada fue desarrollada por el estudio inglés de videojuegos *HandCircus*. Es un juego llamado *OKABU Meet the héroes*[20]. Como parte de la investigación para este trabajo de grado, se realizaron comunicaciones con dicho grupo con el objetivo de reunir información acerca de la implementación por ellos realizada puesto que no existe documentación de la misma.

HandCircus realizó dos implementaciones con aproximaciones diferentes. Inicialmente utilizaron *PSGL*, una versión de OpenGL especial para la consola, para realizar el renderizado de las imágenes, sin embargo era demasiado lenta dado que no utilizaba los SPU ejecutándose todo desde la PPU. Para la segunda aproximación generaron un hibrido entre *PhyreEngine* y OGRE 3D con el fin provechar las ventajas de los SPU y otras optimización que trae el *PhyreEngine* [21].

1.3.4. ITK

The insight Segmentation and Registration Toolkit es una plataforma para el desarrollo de aplicaciones de segmentación y registro de imágenes. Fue desarrollada como un conjunto de algoritmos open-source para analizar las imágenes del "Visible Human Project[3][6].

ITK procesa en una pipeline. Una imagen de entrada pasa a través de uno o varios filtros que realizan operaciones sobre la misma produciendo un nueva imagen [3].

En la investigación realizada acerca de trabajos referentes a ITK, no se encontraron implementaciones que se ejecuten en la consola PS3.

2. Marco Institucional

El trabajo se realizará en el grupo de investigación Takina, perteneciente al Departamento de Ingeniería de Sistemas de la Pontificia Universidad Javeriana.

III – DESARROLLO DEL TRABAJO

1. INVESTIGACIÓN INICIAL Y DE TECNOLOGÍA

Inicialmente se realizó una investigación acerca del trabajo con la consola y OGRE 3D, el cual permitió caracterizar el estado del arte descrito en el Marco Teórico (ver Sección II Marco Teórico). Posteriormente se realizó una selección de herramientas y referencias a utilizar para el desarrollo del proyecto.

1.1. Selección de herramientas

1.1.1. Sistema operativo

Actualmente la instalación de sistemas operativos diferentes al *GameOS*, propio de la PS3 está restringida por Sony que decidió deshabilitar la función *OtherOs* de sus consolas PS3 a partir del *Official Firmware OFW 3.21* con el fin de evitar que se pudiesen desarrollar *Custom Firmwares CFM* y de esta manera frenar la piratería de videojuegos. En versiones anteriores se permitía la instalación de Sistemas Operativos Linux y estaba disponible el kit de desarrollo de la consola.

Con la prohibición por parte de Sony para el uso de sistemas operativos Linux, muchas de las distribuciones dejaron de soportarse y actualizarse, en muchos casos los servidores de repositorios están fuera de línea y las organizaciones que las desarrollaban decidieron no crear nuevas versiones de sus distribuciones para la consola.

Sin embargo aún es posible realizar la instalación de sistemas operativos Linux en consolas que cuenten con un *Official Firmware OFW* inferior a 3.21, es decir que nunca se hayan actualizado, pues aún conservan la opción *OtherOS* o consolas con *Custom Firmware* inferior a 3.55. Para las consolas con *OFW superior a 3.21* es necesario realizar un proceso de *Downgrade* para habilitar la opción OtherOS.

- **Linux Fedora**

Fedora es una distribución de Linux que en la actualidad cuenta con 19 versiones, las versiones comprendidas entre la F5 y la F10 soportan la arquitectura del procesador CellBE, sin embargo este sistema operativo no ha sido construido específicamente para la consola PS3, por este motivo es necesario configurar algunas características de forma manual para el correcto funcionamiento de Fedora en la consola [22].

En principio cualquier distribución de Linux PPC podría correr sobre la PS3 pues el sistema operativo corre sobre el PPE que es compatible con los procesadores PowerPC (Macintosh). Para acceder a los SPE y habilitar otros dispositivos, es necesario utilizar el Kernel de Linux. El Kernel de las distribuciones de Fedora F5-F10 soporta el procesador CellBE, sin embargo es necesario recompilar el Kernel para mejorar el desempeño del procesador[8].

- **Linux Yellow Dog**

Yellow Dog es un sistema operativo Linux baso en Fedora, diseñado por la compañía Fixstars específicamente para soportar los procesadores de arquitectura CellBE. Cuenta con 7 distribuciones aunque la última disponible para descarga es la 6.2, esta versión de Yellow Dog incluye el SDK de IBM versión 3.1[23].

Por ser un sistema operativo diseñado específicamente para la consola PS3 fue inicialmente elegido para el desarrollo de este trabajo de grado, pues no solamente simplifica el proceso de instalación sino también cuenta con las herramientas(Kit de desarrollo glibc 2.5 [24], IBM CellBESDK v 3.1) necesarias para el desarrollo de la capa a implementar.

- **Linux Red Ribbon**

Es una distribución de Linux para el PS3 basada en Debian, cuyo objetivo es facilitar el proceso de instalación para usuarios poco experimentados[25]. Red Ribbon 7 es la última versión disponible para la descarga e instalación en la PS3.

1.1.2. Compilador

- **GCC**

Es un compilador integrado del proyecto GNU para C, C++, Objective C y Fortran; es capaz de recibir un programa fuente en cualquiera de estos lenguajes y generar un programa ejecutable binario en el lenguaje de la máquina donde ha de correr.
La sigla GCC significa "GNU Compiler Collection". Originalmente significaba "GNU C Compiler"; todavía se usa GCC para designar una compilación en C. G++ refiere a una compilación en C++[26].

1.1.3. Cell-SDK

Es un kit de desarrollo creado por IBM específicamente para el procesador CellBE, cuenta con dos compiladores basados en GCC que permiten compilar código creado en C y C++.

PPU-GCC y SPU-GCC permiten compilar y generar ejecutables de aplicaciones que corren en el PPU y en los SPU respectivamente.

El Cell-SDK también cuenta con una serie de librerías que permiten hacer uso de los componentes del procesador, este grupo de herramientas es llamado *Toolchain.* El SDK proporciona todas las librerías estándar de C y C++, permitiendo que se ejecute cualquier tipo de aplicación codificada en estos lenguajes de programación[27].

El Cell-SDK fue diseñado únicamente para distribuciones de Fedora y viene empaquetado en archivos tipo *RPM*, lo que implica que en sistemas operativos que utilicen otro sistema de paquetes, se dificulta la instalación del SDK, por ejemplo Debian que utiliza archivos tipo DEB.

Actualmente IBM no permite el uso libre del Cell-SDK, el acceso al kit de desarrollo está restringido a desarrolladores registrados y debe pagarse por su uso. Sin embargo como el Cell-SDK venia incluido en distribuciones de Fedora, aún se puede acceder a estas desde los discos de instalación.

Algunos grupos de desarrolladores informales han trabajado para generar una versión del Cell-SDK que pueda instalarse en distribuciones Debian[28] aunque aún se encuentran en desarrollo.

1.2. Tarjeta de Video

La PS3 cuenta con una tarjeta de video diseñada por Nvidea especialmente para trabajar con el procesador CellBE, está basada en la GeForce 7800 GTX y fue llamada RSX “Reality Synthetizer”[29]. El trabajo de desarrollo de aplicaciones para la PS3 está centrado en el procesador CellBE pues no hay acceso a la RSX desde Linux [30].

2. EXPLORACIÓN

Durante esta fase se construyeron los requerimientos en conjunto con el director de trabajo de grado y se inició el trabajo de preparación de la consola.

2.1. Levantamiento de Requerimientos

La siguiente tabla contiene los requerimientos definidos para el proyecto, las tareas asociadas a cada uno y el tiempo estimado para realizarlas:

Id	Requerimientos	Tarea	Horas requeridas	Horas empleadas
RQ01	El sistema debe ejecutarse en la consola PS3	Flashear consola	3	48
		Configurar Other OS	2	6
		Instalar Linux	2	50
		Recompilar kernel de Linux	2	No se realizó
RQ02	El sistema debe desarrollarse en lenguaje C++	Instalar Compilador	1	1
		Instalar CMAKE	1	1
		Instalar el SDK del procesador CELL	2	Más de 50. No se consiguió
RQ03	El sistema debe ejecutar aplicaciones que usen OpenGL	Instalar Open GL	1	1
		Construir una aplicación con OpenGL	2	1
RQ04	El sistema debe ejecutar aplicaciones que usen RSXGL	Instalar RSXGL	1	Más de 10. No se consiguió
		Construir una aplicación con RSXGL	1	No se realizó
RQ05	El sistema debe ejecutar aplicaciones que usen OGRE 3D	Instalar prerrequisitos	2	3
		Compilar OGRE 3D	2	8
		Instalar OGRE 3D	2	1
		Instalar IDE	2	No se realizó
		Construir una aplicación con OGRE 3D	2	5

		Compilar una aplicación con OGRE 3D	1	5
		Ejecutar una aplicación con OGRE 3D	1	1
		Construir una aplicación con OGRE para el PPU	2	No se realizó
		Construir una aplicación OGRE 3D para los SPU	2	No se realizó
RQ06	El sistema debe ejecutar aplicaciones usen ITK	Instalar ITK	4	48
		Construir una aplicación con ITK	4	No se realizó
RQ07	El sistema debe permitir ejecutar el modelo MIsT	Ejecutar el Prototipo 1 de MIsT	4	1
		Ejecutar el Prototipo 2 de MIsT	4	1
		Ejecutar el Prototipo 3 de MIsT	4	1
RQ08	El sistema debe permitir ejecutar VRLS	Ejecutar el simulador VRLS	4	2
		Ejecutar el simulador VRLS usando el control PS3	4	48
RQ09	El sistema debe permitir al usuario interactuar mediante diferentes dispositivos	El sistema debe permitir al usuario interactuar por medio del Pad de juegos	1	50

		El sistema debe permitir al usuario interactuar por medio del teclado y el mouse	1	1
RQ10	El sistema debe ejecutar aplicaciones que usen PhyreEngine	Instalar PhyreEngine	2	No se realizó.
		Construir una aplicación con PhyreEngine	2	No se realizó

Tabla 1: Requerimientos.

2.2. Preparación de la consola

2.2.1. Firmware

La consola PS3 cuenta con un sistema operativo diseñado específicamente para el manejo de juegos, es conocido como *GameOS*. En las primeras versiones de la consola se contaba con la opción *OtherOS*, que permitía la instalación de sistemas operativos diferentes al que trae por defecto la consola, sin embargo a partir de la versión 3.21 Sony decide eliminar la opción *OtherOS* como medida de seguridad.

Para realizar la instalación de Linux en la consola es necesario que ésta cuente con una versión de *Firmware* inferior a 3.21; para las consolas con versiones superiores, es necesario realizar un *Downgrade* para regresar a la versión con soporte de *OtherOS*. El procedimiento se realiza a nivel de software y basta con copiar el archivo que contiene el *Firmware* deseado en una memoria USB y actualizarlo directamente en la consola (ver Anexo guía de Downgrade).

Para las consolas cuyo *Firmware* de fábrica *OFW* es superior a 3.55 el proceso de *Downgrade* no es posible y es necesario realizar un procedimiento sobre el hardware denominado *Flashing*. El procedimiento de flasheado requiere que la consola sea destapada para trabajar sobre sus circuitos; tiene el riesgo de inutilizar la consola, por lo que debe ser realizado por un experto.

2.2.2. OtherOS

Una vez la consola cuenta con la versión de *Firmware* 3.55, se debe configurar la opción *OtherOs*, que permite la instalación de Linux como sistema operativo alternativo [31] (Ver Anexo Guía de instalación OtherOs).

2.2.3. Instalación Linux

- Se realizaron varios intentos por instalar Linux en la consola. Inicialmente se instaló Yellow Dog 6.2 sin embargo, una vez finalizada la instalación, la PS3 mostraba un mensaje que indicaba que el *GameOs* se había borrado y debía ser restaurado. Se procedió a instalar de nuevo el *CFW 3.55* y realizar todo el proceso de montaje del *OtherOS.*

- Posteriormente se instaló la distribución Red Ribbon 7 de Linux. La instalación se realizó sin inconvenientes y en esta ocasión, al reiniciar la PS3 se pudo iniciar el sistema operativo. Una vez instalado se procedió a preparar las herramientas a utilizar (ver Anexo Guía de instalación Red Ribbon).

  En la siguiente tabla se resume el avance en la instalación de las herramientas:

Herramienta	Problemas
Compilador GCC	Instalado sin problemas
CMake	Instalado sin problemas
OpenGL	Instalado sin problemas
OGRE 3D3D	Instala sin problemas. La ejecución es lenta pues solo usa el PPU para renderizar.
OIS	Instalado sin problemas. Reconoce el control de la PS3 pero no captura las acciones sobre los botones.
Cell-SDK	La versión oficial del SDK existe solo para Fedora. Una versión creada de manera no oficial, se logra instalar, pero no permite el correcto uso de los compiladores PPU-GCC y

	SPU-GCC
RSXGL	El proceso de instalación es complejo pues requiere la instalación de varias dependencias. No se lograron ejecutar los ejemplos por problemas con el SDK y con la instalación de las dependencias.
PhyreEngine	No se logró tener acceso al PhyreEngine para Linux. La versión para Windows necesita VisualStudio 2008 SP1 que no pudo ser instalado.

Tabla 2: Herramientas necesarias para el desarrollo.

Una vez compilado e instalado OGRE 3D, se procedió a compilar y ejecutar una pequeña aplicación (ver Sección 4.1.2 Pruebas OGRE 3D), notando que la ejecución se realizaba de manera extremadamente lenta, debido a que el proceso de renderizado de OGRE3D utiliza OpenGL[32] y este corre únicamente en la PPU, desaprovechando los 6 SPU del procesador.

Tras una corta investigación se descubrió que actualmente se trabaja en la construcción de RSXGL[33][34], una versión de OpenGL para el PS3 que utiliza los SPU. Esta librería se instaló en la consola pero no pudo ser probada debido a que no se contaba con el Cell-SDK.

- Posteriormente se decidió cambiar la distribución de Linux para solucionar el problema del SDK, por este motivo se realizaron pruebas de instalación con Fedora Core 9,17 y una máquina virtual con Fedora Core 7, todas las versiones presentaron inconvenientes y se describen a continuación:

  La instalación de Fedora Core 17 no tuvo éxito pues nunca se logró lanzar el asistente de instalación en modo texto o gráfico, así que se pasó a la distribución de Fedora Core 9, esta última sí permitió realizar instalación del sistema aunque ésta, al igual que Yellow Dog, eliminaba el *GameOS* imposibilitando el arranque de la consola.

En cuanto a la máquina virtual con Fedora Core 7, se encontraron dos problemas: el primero es que no se pueden ejecutar programas compilados con el Cell-SDK; el segundo es que aunque el sistema operativo reconoce el control de juegos de la PS3, no permite hacer uso de sus botones. Este segundo fallo se descubrió ejecutando pruebas con OIS y VRPN.

- Posteriormente se realizaron pruebas de instalación con Fedora Core 9 y Yellow Dog 6.2, utilizando un disco duro externo para evitar que se eliminara el *GameOS* y de esta manera iniciar el arranque sin inconvenientes. Los dos sistemas operativos se instalaron de manera adecuada, sin embargo tanto Fedora Core 9 como Yellow Dog 6.2 ya no cuentan con repositorios, por este motivo no es posible preparar un entorno de desarrollo adecuado en ninguno de los dos sistemas operativos.

Es importante aclarar que en la actualidad para tener acceso a las herramientas de desarrollo de la PS3 es necesario ser un desarrollador registrado de Sony[35]. Al inicio de esta fase, se realizó la inscripción diligenciando el formulario de solicitud disponible en la página de desarrollo de Sony [36], sin recibir respuesta por parte de Sony.

3. PLANEAMIENTO

3.1. Priorización de requerimientos

Una vez definidos los requerimientos se procedió a realizar la priorización de los mismos, dándoles un puntaje de 1 a 9, siendo 1 el valor de menor relevancia y 9 el valor para los requerimientos más importantes.

Id	Requerimientos	Puntaje
RQ01	El sistema debe ejecutarse en la consola PS3	9
RQ04	El sistema debe ejecutar aplicaciones que usen OGRE 3D	9
RQ02	El sistema debe desarrollarse en lenguaje C++	8
RQ03	El sistema debe ejecutar aplicaciones que usen OpenGL	8
RQ06	El sistema debe ejecutar aplicaciones usen ITK	8
RQ07	El sistema debe permitir ejecutar el modelo MIsT	7
RQ09	El sistema debe permitir al usuario interactuar mediante diferentes dispositivos	7
RQ08	El sistema debe permitir ejecutar VRLS	6
RQ05	El sistema debe ejecutar aplicaciones que usen RSXGL	3
RQ10	El sistema debe ejecutar aplicaciones que usen PhyreEngine	2

Tabla 3: Requerimientos priorizados.

3.2. Alcance

Teniendo en cuenta las limitaciones encontradas en la fase de Exploración del Cell-SDK y a la priorización de los requerimientos, se decidió que el alcance debería cubrir las siguientes actividades:

- Se realizarán pruebas de rendimiento de OpenGL en la PS3 y en un PC de escritorio y se compararán los resultados.
- Se ejecutarán los prototipos del modelo MIsT y el simulador VRLS y en la PS3 y en un PC de escritorio y se compararán los resultados.
- Se diseñará e implementará un componente que permita utilizar el Control Dual Shock de la PS3, en aplicaciones hechas con OGRE 3D utilizando OIS.
- Se diseñará un componente que verifique la presencia de las dependencias en la consola (OGRE 3D, ITK, OIS, Cell-SDK).
- Se Diseñará un componente que permita hacer uso de todos los elementos del procesador (PPU y SPU) para optimizar el desempeño de las aplicaciones creadas

con OGRE 3D e ITK, aunque el componente no se implementa pues no se cuenta con el Cell-SDK.

3.3. Diseño

Una vez definido el alcance y teniendo los requerimientos priorizados, se inició el diseño arquitectónico utilizando el modelo 4+1 vistas de Phillippe Kruchten [14]. A continuación se muestran los diagramas que componen la arquitectura propuesta:

3.3.1. Vista Lógica

La vista lógica muestra la funcionalidad que ofrecerá el sistema, descompuesta en clases[14]; el diagrama de clases generado se muestra a continuación:

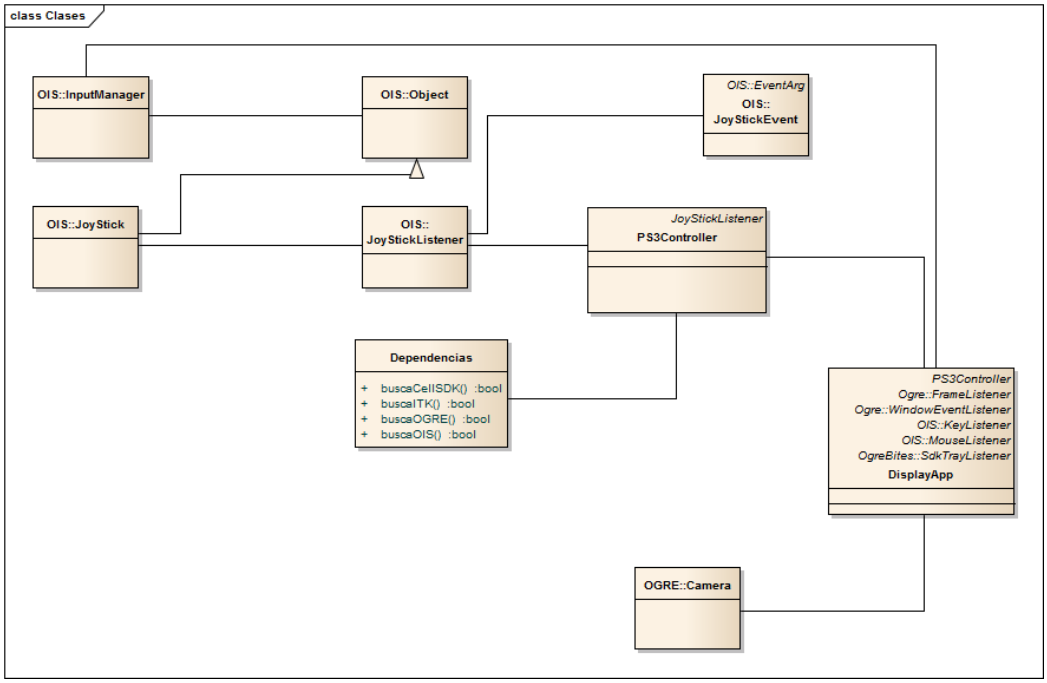

Ilustración 2: Diagrama de Clases

OIS Input Manager: es un objeto de la librería OIS necesario para controlar los dispositivos de entrada como teclado, mouse y joysticks.

OIS JoyStick: es un objeto de la librería OIS que permite representar un *JoyStick* y sus componentes. Esta clase permite hacer uso de los *axis* y botones del control y asignarle un *Listener* para capturar la actividad del control.

OIS JoyStick Listener: es un objeto de la librería OIS que permite capturar los eventos del joystick.

OIS JoyStick Event: es un objeto de la librería OIS que permite capturar el estado del JoyStick en un momento de tiempo determinado.

OGRE Camera: es un objeto de OGRE 3D que permite crear una cámara y hacer movimientos sobre ésta.

Display App: Es una aplicación que utiliza OGRE 3D para renderizar objetos, cuenta con una cámara cuyos atributos de posición y dirección pueden ser modificados a partir de los dispositivos de entrada.

PS3Controller: Esta clase utiliza un JoyStick Listener para capturar eventos del JoyStick, en este caso el control DualShock 3 de la PS3 y asignarle movimientos de la cámara u otros métodos para mover los objetos declarados en la Display App, a sus botones y axis.

3.3.2. Vista de Desarrollo

La vista de desarrollo muestra como está dividido el sistema en componentes de software y las dependencias que hay entre esos componentes [14]; el diagrama de componentes generado se muestra a continuación:

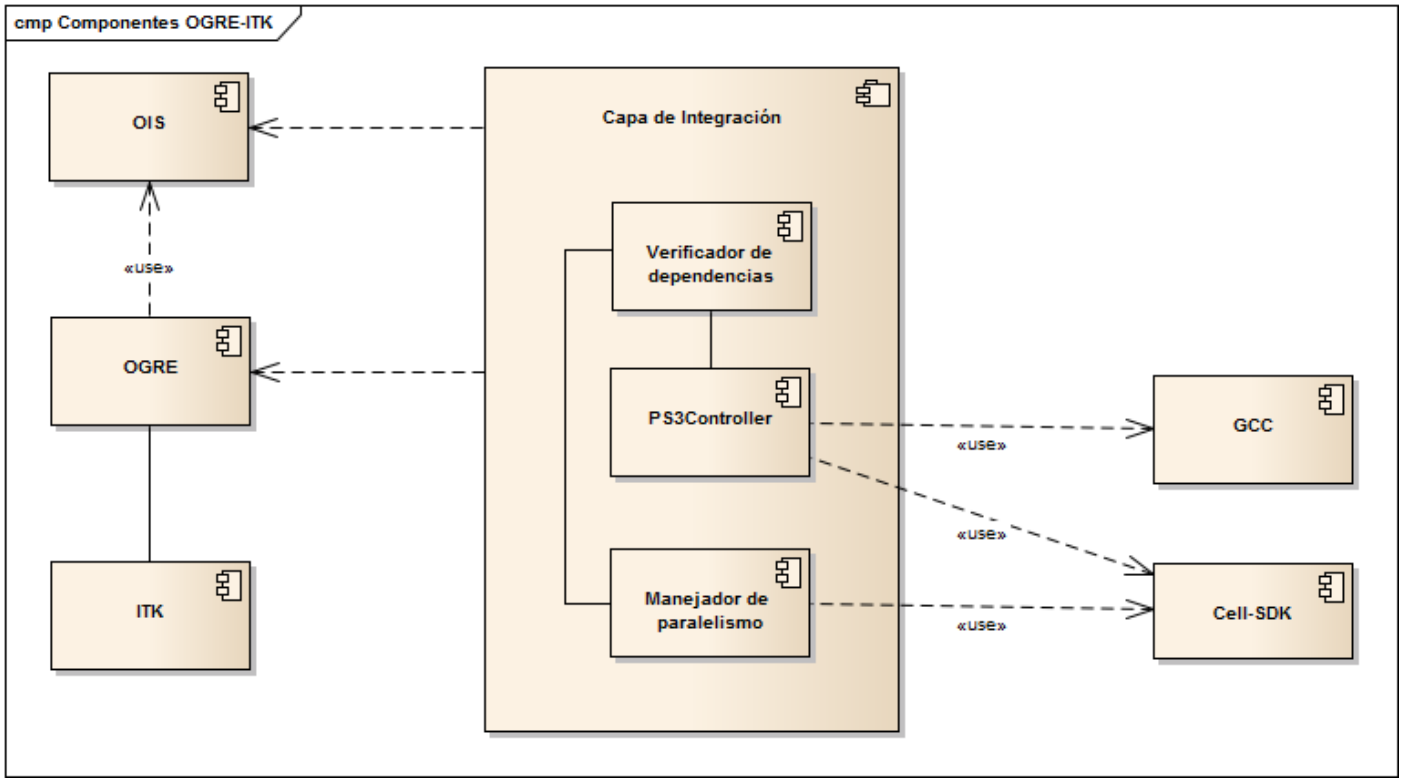

Ilustración 3: Diagrama de Componentes.

OIS: es un componente externo que permite hacer uso de los periféricos de entrada como mouse, teclado y joystick.

OGRE: este componente externo representa a OGRE 3D.

ITK: este componente externo representa a ITK.

GCC: este componente representa el compilador GCC que genera ejecutables usando únicamente la PPU.

Cell-SDK: este componente representa los compiladores que permiten generar ejecutables usando PPU y SPU.

Capa de Integración

Verificador de dependencias: este componente permite comprobar que los demás componentes de software (OGRE, OIS, CellSDK, ITK) estén instalados en el sistema.

PS3Controller: este componente permite hacer uso del control DualShock 3 de la PS3 en aplicación es que usan OGRE 3D.

Manejador de Paralelismo: este componente permite distribuir los objetos de OGRE 3D e ITK en los diferentes *cores* del procesador para optimizar el rendimiento de las aplicaciones.

3.3.3. Vista Física

Esta vista describe cómo se ubicaran físicamente todos los componentes del sistema [14]; el diagrama de despliegue generado se muestra a continuación:

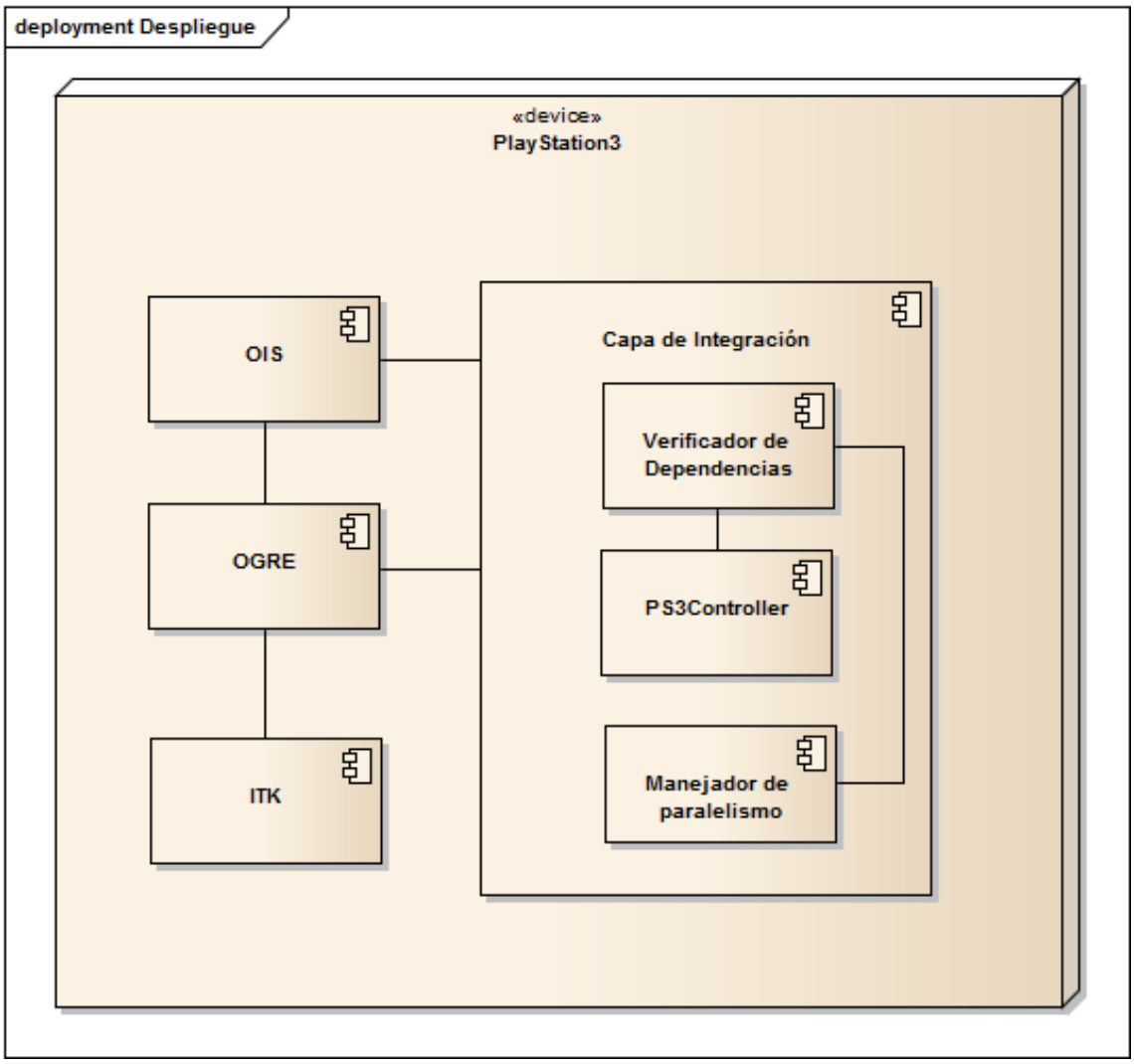

Ilustración 4: Diagrama de despliegue

3.3.4. Vista de Procesos

En esta vista se muestran los procesos que hay en el sistema y la forma en la que se comunican estos procesos [14]; el diagrama de actividad generado se muestra a continuación:

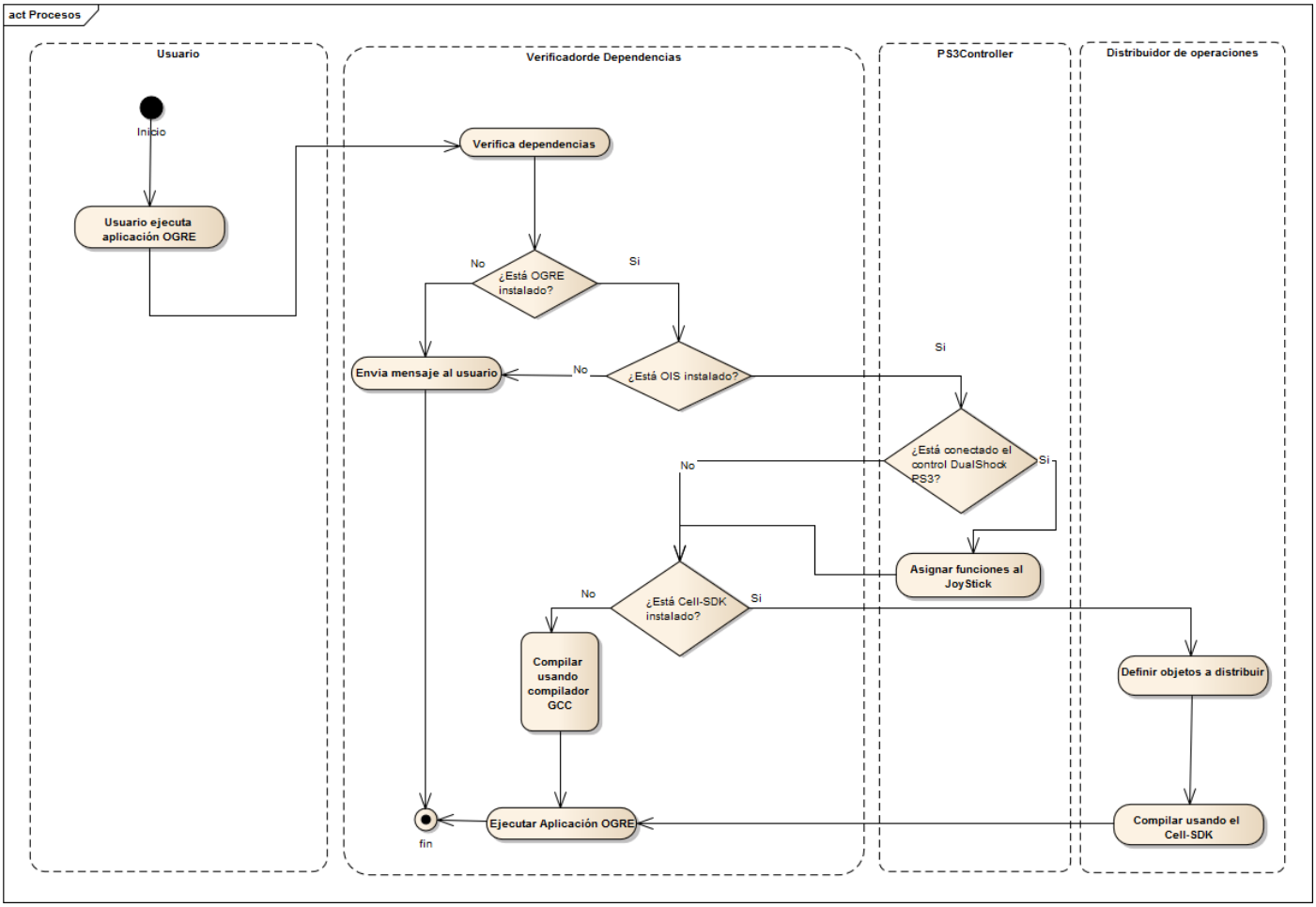

Ilustración 5: Diagrama de actividad.

3.3.5. Vista de Escenarios

Esta vista va a ser representada por los casos de uso software y va a tener la función de unir y relacionar las otras 4 vistas[14]; el diagrama de casos de uso generado se muestra a continuación:

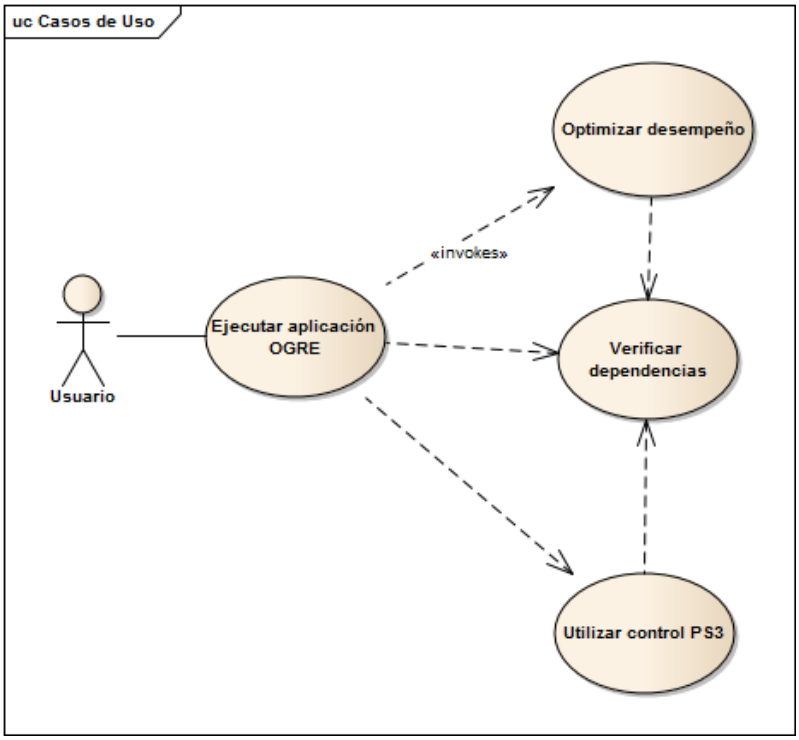

Ilustración 6: Casos de Uso.

4. PRODUCCIÓN

4.1. Pruebas

Como se ha mencionado anteriormente, las aplicaciones desarrolladas solo se han podido ejecutar en la PPU, motivo por el cual el desempeño de estas es inferior al que presentarían de ejecutarse de manera distribuida por todo el procesador, e inferior al que tienen al ejecutarse en un PC de escritorio.

Para realizar las pruebas descritas a continuación, se decidió realizar una comparación de desempeño entre la PS3 y un PC con una configuración de hardware y software que se describe en la siguiente tabla, al igual que la configuración de la PS3:

Procesador	Tarjeta de Video	Disco Duro	Sistema Operativo
Intel Core 2 Duo 2.6ghz	Intel G45/G43 Express	250 GB	Ubuntu 13.10, basado en Debian
IBM CellBE 3.2ghz	RSX basada en la NVIDIA 7800 GT	120 GB, con partición 24GB para Linux	Red Ribbon 7, basado en Debian

Tabla 4: Configuración PC de pruebas.

4.1.1. OpenGL

Con el fin de determinar si era posible utilizar este prerrequisito en la consola, se realizaron pruebas compilando y ejecutando código realizado con OpenGL, pues OGRE 3D utiliza esta API para generar los gráficos 3D

Los resultados de la prueba fueron positivos, los programas desarrollados con OpenGL se ejecutaron de manera adecuada, sin embargo, se notó que el desempeño es limitado pues la ejecución se realiza en la PPU y no se utilizan los SPU. El primer programa es una pequeña aplicación que genera un *SolidTeaPot* y un *WiredTeaPot* y los hace rotar en el eje, mientras que el segundo cuenta con una mayor complejidad, genera y anima más objetos simultáneamente, simulando un sistema planetario. Los dos programas (ver Anexo Pruebas con OpenGL), fueron ejecutados tanto en la PS3 como en el PC de pruebas anteriormente descrito, los resultados se resumen en la siguiente tabla:

Dispositivo	FPS	
	Aplicación 1	Aplicación 2
PS3	16.52 fps, con una ventana de tamaño 900x900 px.	8.47 fps, con una ventana de tamaño 1024x1024 px.
PC	59.82 fps, con una ventana de tamaño 900x900 px.	59.70 fps, con una ventana de tamaño 1024x1024 px.

Tabla 5: Pruebas con OpenGL

Es evidente la reducción de rendimiento en la PS3 que equivale aproximadamente al 20% del rendimiento en el PC de pruebas.

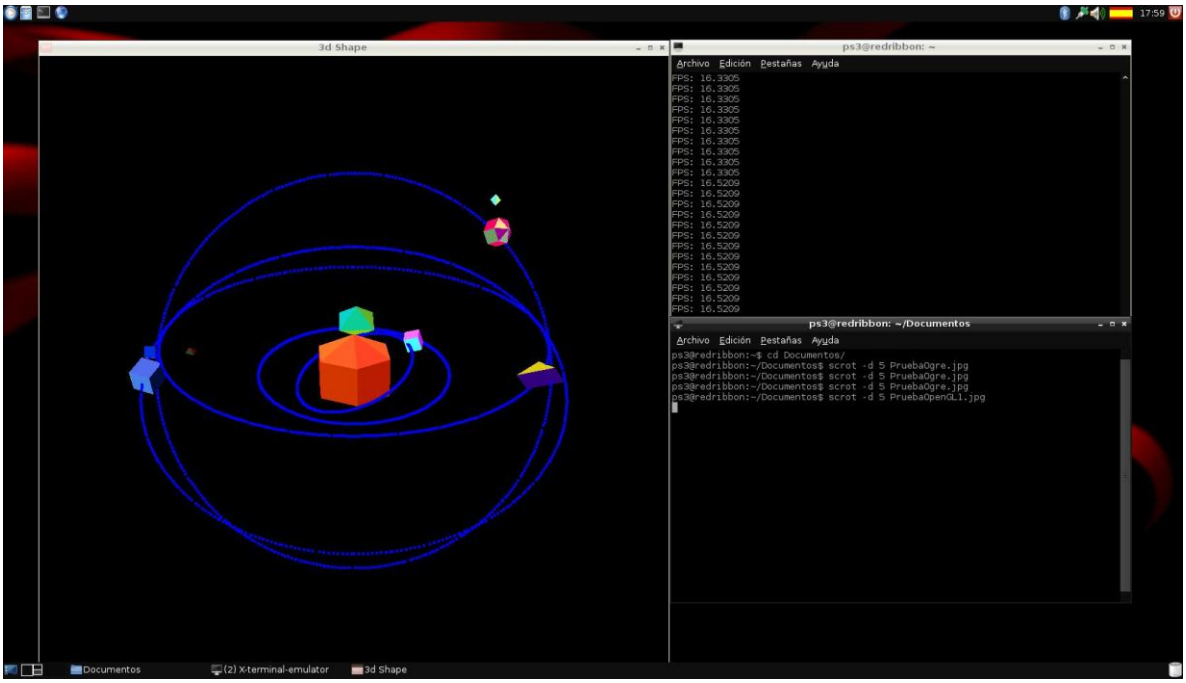

Ilustración 7: Primera Prueba con OpenGL.

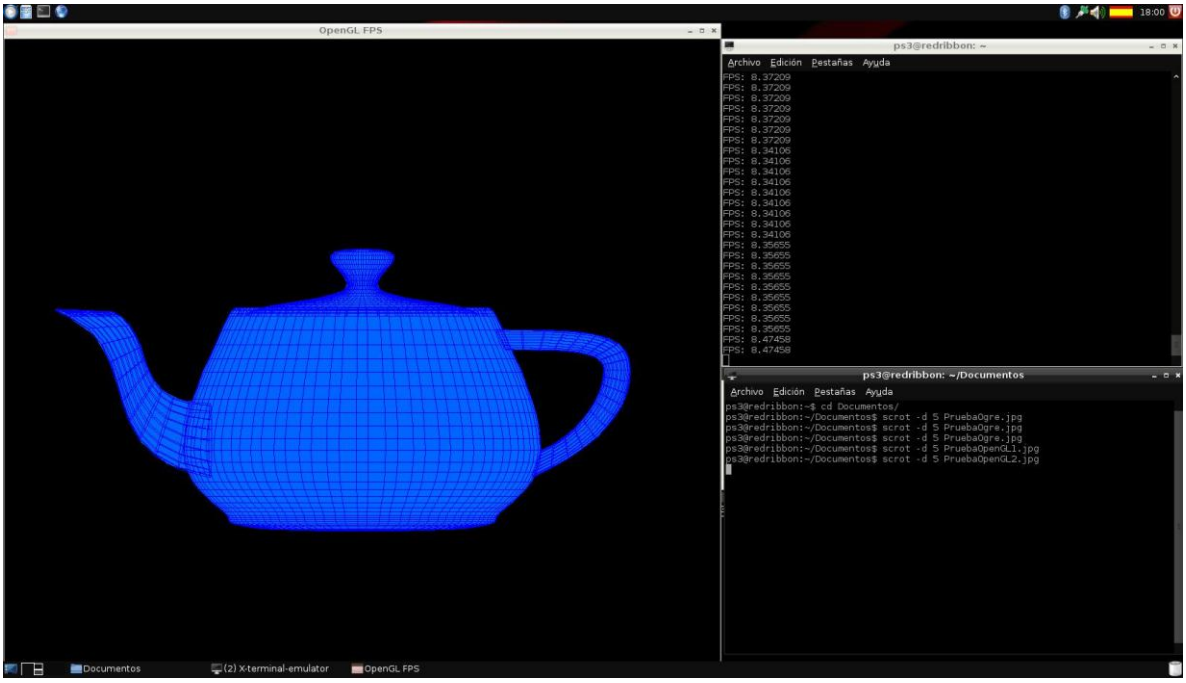

Ilustración 8: Segunda prueba con OpenGL.

4.1.1.1. GLXGEARS

Para determinar el desempeño de la PS3 en la ejecución de OpenGL se realizó una comparación ejecutando *glxgears*[37], un programa de OpenGL que muestra la cantidad de frames por segundo que pueden ser renderizados. Este programa (ver Anexo Pruebas con OpenGL), se ejecutó tanto en la PS3 como en el PC anteriormente descrito, los resultados se resumen en la siguiente tabla:

Dispositivo	FPS
PS3	72,49 fps en promedio, con una ventana de 300x300 px, 8.27 fps en promedio en ventana maximizada, 1920x1080 px
PC	60fps sin importar el tamaño en pantalla

Tabla 6: Pruebas con OpenGL glxgears.

Para este caso, cuando se ejecuta en la PS3 con una ventana de 300x300 pixeles la aplicación tiene un rendimiento mayor que en el PC de pruebas, sin embargo se puede evidenciar la reducción de rendimiento en la PS3 al aumentar el tamaño de la ventana. Con la ventana de 300x300 pixeles la PS3 tiene un desempeño equivalente al 121% del generado por el PC de pruebas, pero al aumentar el tamaño de la ventana al máximo, se reduce este desempeño al 13.82% en relación con el PC.

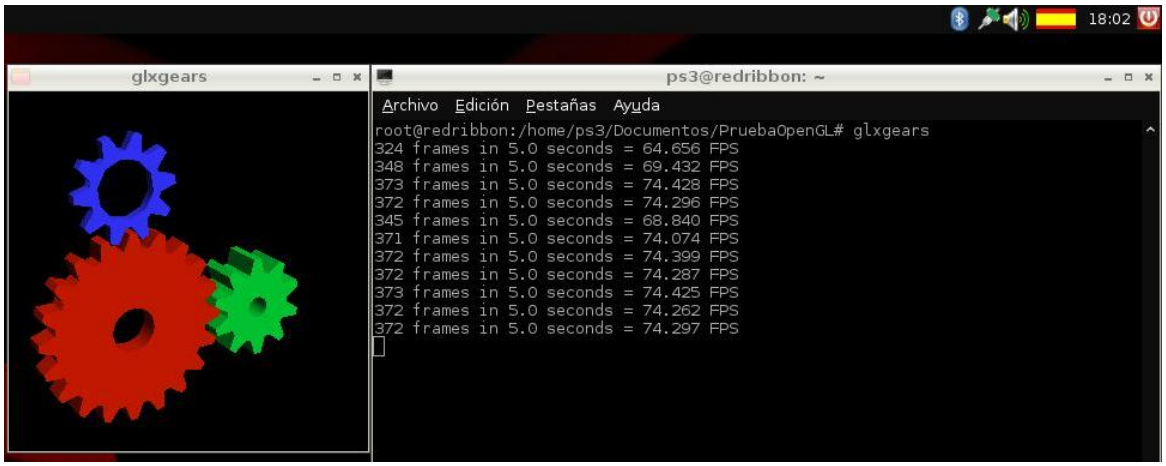

Ilustración 9: Primera prueba con glxgears.

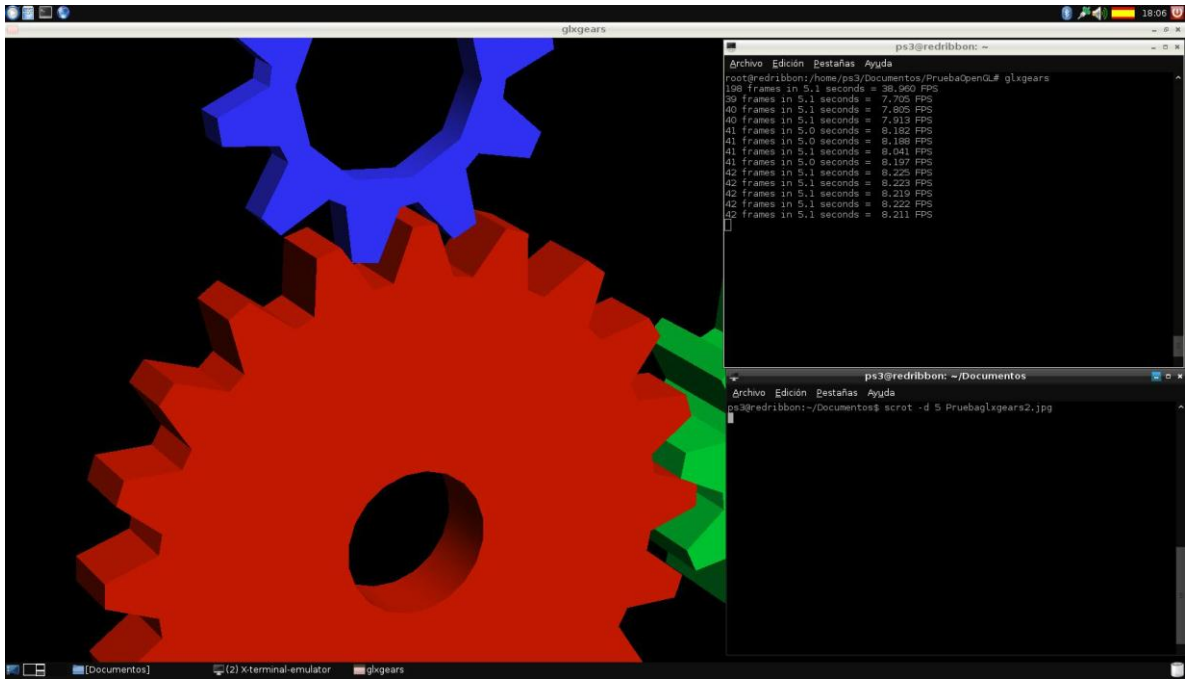

Ilustración 10: Segunda prueba con glxgears.

4.1.2. OGRE3D

Posterior a las pruebas realizadas con OpenGL, se procedió a compilar y ejecutar un ejemplo básico realizado con OGRE 3D[38]. En este ejemplo se generan dos objetos *Entity* que usan una malla para generar una cabeza de ogro. A estos objetos se les puede cambiar sus atributos de ubicación, tamaño, *pitch, yaw* y *roll* dentro del código y se puede mover la cámara durante la ejecución utilizando el teclado, el mouse y el control DualShock 3.

Esta pequeña aplicación fue probada en la PS3 sin problemas de compilación o ejecución, pero es evidente que el rendimiento durante la ejecución es demasiado bajo y el tiempo de respuesta en el uso del teclado, mouse y control, que para este caso representan la interfaz hombre-máquina, es demasiado alto. Los resultados de la comparación con pruebas realizadas en el PC son:

Dispositivo	(t) Compilación	(t) Ejecución	FPS	Detalles
PS3	112 segundos	3 segundos	0.8	Algunos polígonos de las mallas se sobrelapan. El tiempo de respuesta de teclado y mouse es muy alto.
PC	3 segundos	1 segundo	59.9	

Tabla 7: Pruebas con OGRE 3D.

Una vez en OGRE 3D al aumentar la complejidad en la construcción y el renderizado de los objetos, el rendimiento en la PS3 se reduce aún más; para esta prueba equivale al 1.34% del rendimiento mostrado por el PC de pruebas.

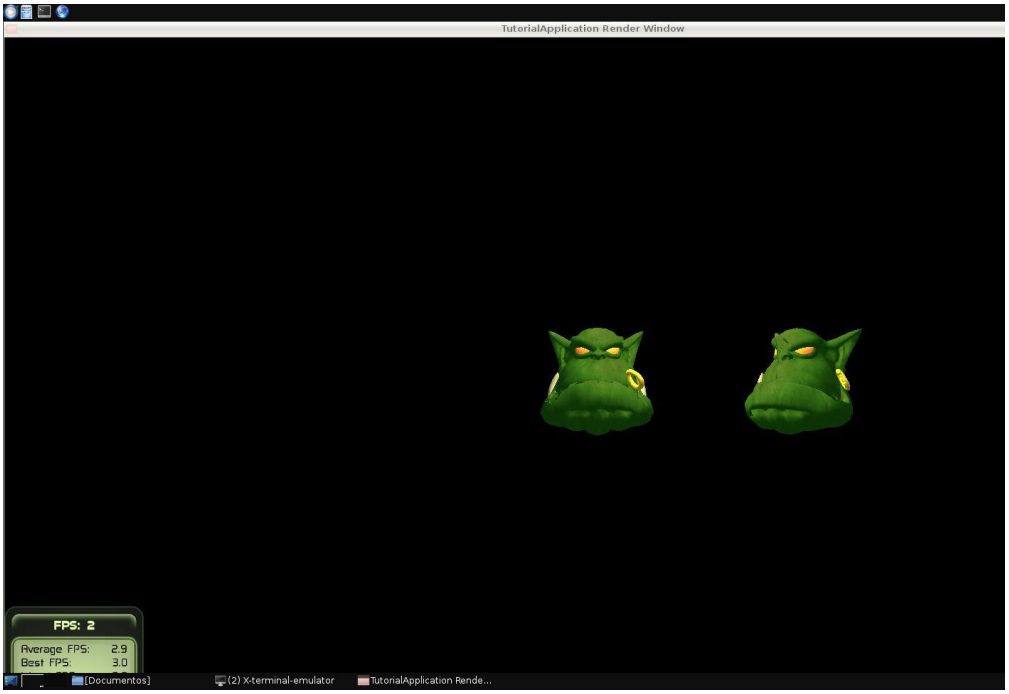

Ilustración 11: Prueba con OGRE 3D.

4.1.2.1. VRLS

A medida que se agregan más objetos o la complejidad en la construcción de estos aumenta, se nota la disminución de desempeño en compilación y ejecución. Este es el caso del simulador *VRLS*, que fue probado en la consola:

Dispositivo	(t) Compilación	(t) Ejecución	FPS promedio	Detalles
PS3	4 min, 45 seg	45 segundos	0,74	El tiempo de compilación y ejecución es mayor. No se cargan los colores de los objetos. No se pueden activar/desactivar las diferentes vistas. No hay respuesta del teclado.
PC	35 segundos	6 segundos	21,66	Moviendo las cámaras e instrumentos, el rango está entre 19 y 24 FPS.

Tabla 8: Pruebas con VRLS.

Las pruebas con el simulador VRLS son quizás las más críticas, pues ya no solo se ejecutan instrucciones de OGRE 3D y OpenGL sino que se suma el trabajo de ITK en la carga de los órganos que se visualizan posteriormente con OGRE 3D. En este caso no solo se reduce el rendimiento y la calidad del renderizado, también, el tiempo de compilación y ejecución se incrementan críticamente (este tiempo se midió desde el momento en que se ejecuta el comando ./MyApp, hasta el momento en que la aplicación inicia). Para esta prueba la cantidad de *fps* en la PS3 equivale al 3,42% de *fps* en el PC de pruebas y el retardo en compilación y ejecución es aproximadamente 8 veces mayor.

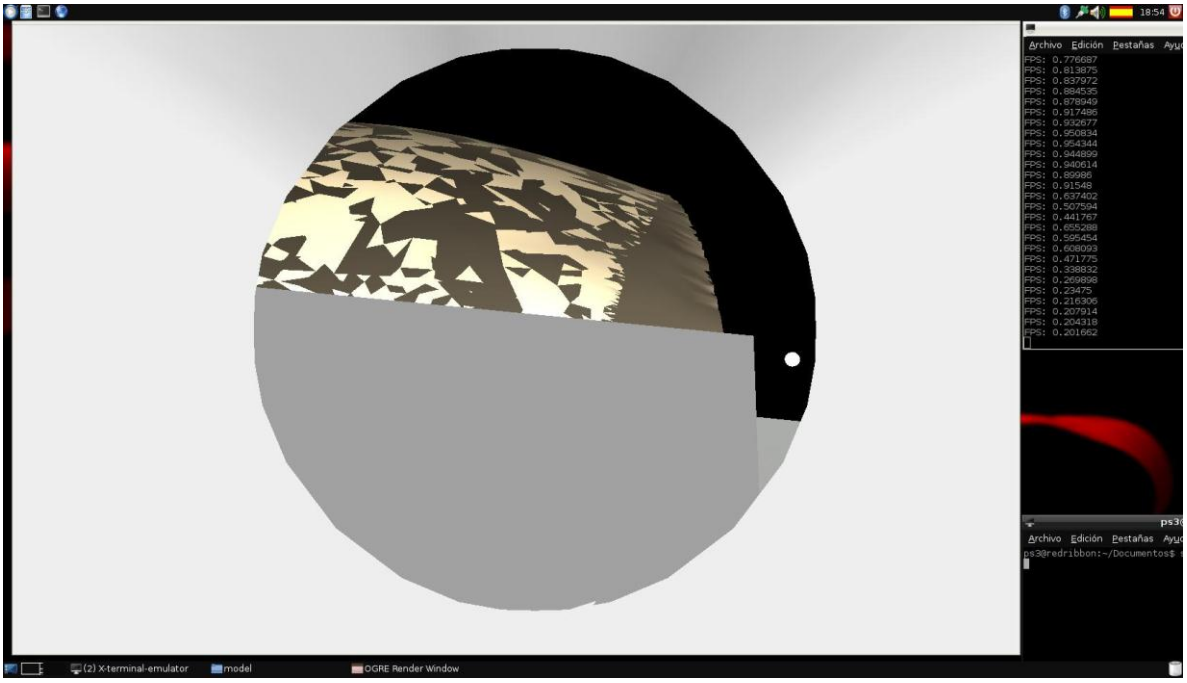

Ilustración 12: Prueba con VRLS.

4.1.2.2. MIsT

MIsT cuenta con tres prototipos que fueron ejecutados en el PC de pruebas y en la PS3. El primero permite la visualización con OGRE 3D de una imagen medica cargada con ITK. El segundo prototipo crea un filtro de decimación para permitir reducir el número de caras que conforman una malla. El tercer prototipo permite visualizar una malla médica cargada con ITK usando OGRE 3D [3].

Una vez ejecutados los prototipos se obtuvo la información de rendimiento mediante el registro de la cantidad de *frames* por segundo que toma en cada máquina el renderizado, los resultados se muestran a continuación:

Dispositivo	FPS			Detalles
	Prototipo 1	Prototipo 2	Prototipo 3	
PS3	3.55 fps	3.52 fps	0.87 fps	Los prototipos se ejecutan sin perdidas de calidad en las imágenes, aunque el renderizado y el refresco de pantalla es demorado
PC	59.81 fps	59.82 fps	18.13 fps	

Tabla 9: Pruebas con MIsT.

Para esta prueba la cantidad de *fps* en la PS3 equivale al 5.94%, 5.88% y 4.80% de *fps* en comparación con la ejecución de los 3 prototipos en el PC de pruebas respectivamente.

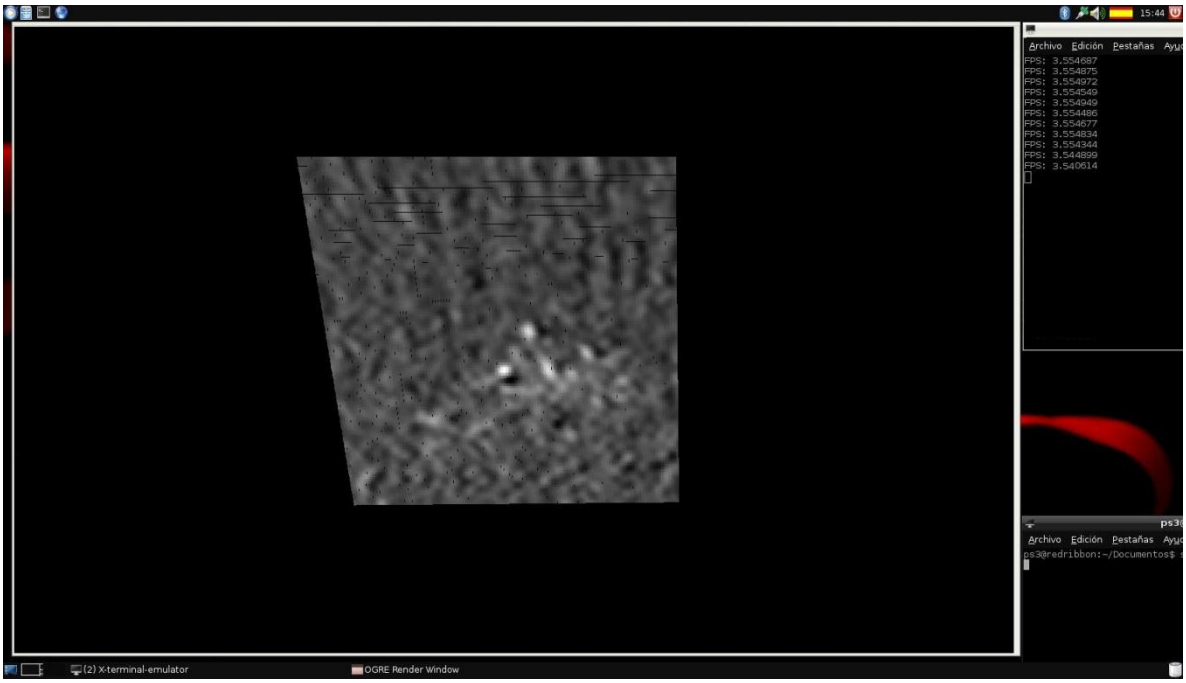

Ilustración 13: Prueba MIsT 1.

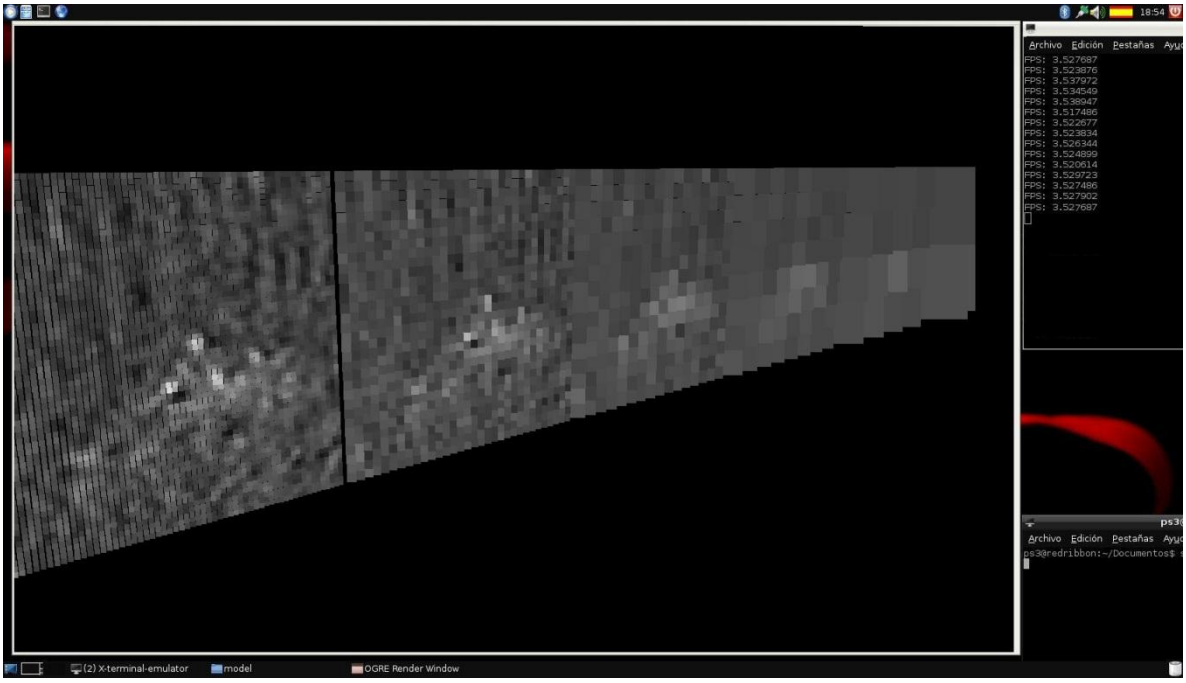

Ilustración 14: Prueba MIsT 2.

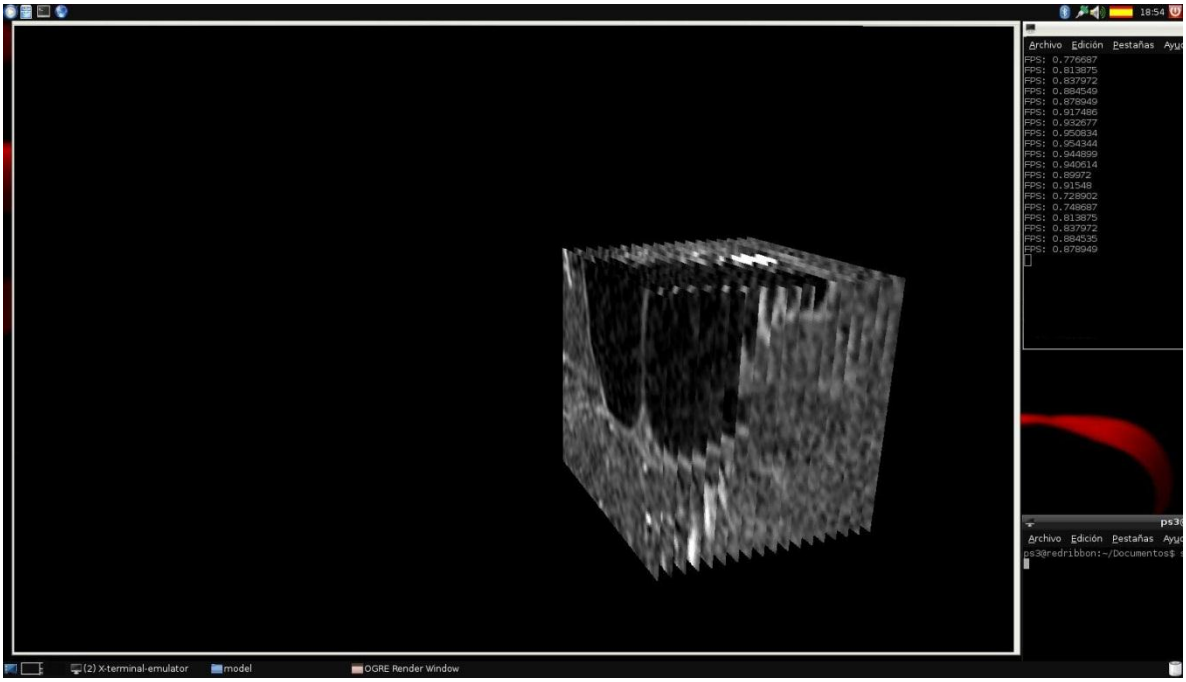

Ilustración 15: Prueba MIsT 3.

4.2. Prototipo PS3Controller

Para tener una interacción adecuada con la consola se hacía necesario el uso del control de juegos DualShock 3 de la PS3 en reemplazo de teclado y mouse como interfaz hombre-máquina. OGRE 3D utiliza OIS para implementar dicha interacción, es por esto que tiene ya definidos métodos para que el usuario pueda interactuar utilizando el teclado y el mouse. Estos métodos ya incluidos en OGRE 3D permiten el movimiento de las cámaras usando las teclas de movimiento del teclado y la posición del mouse dentro de la ventana de ejecución de las aplicaciones OGRE 3D.

OIS también permite el uso de JoySticks, sin embargo como la configuración de estos difiere según el fabricante, OGRE no incluye una implementación que permita hacer movimiento de cámaras u objetos con el control de la PS3. El proceso de configuración del control en la PS3 no es directo y depende de varias dependencias (ver Anexo Guía de configuración conexión control PS3 Dualshock).

Se desarrolló una clase que permite hacer uso del control DualShock 3 en una aplicación de OGRE 3D, asignando la ejecución de métodos a sus botones y análogos. Esta clase puede ser utilizada en cualquier aplicación que use OGRE 3D ya sea en un PC o directamente en la consola. Para ver su uso (ver Anexo PS3Controller).

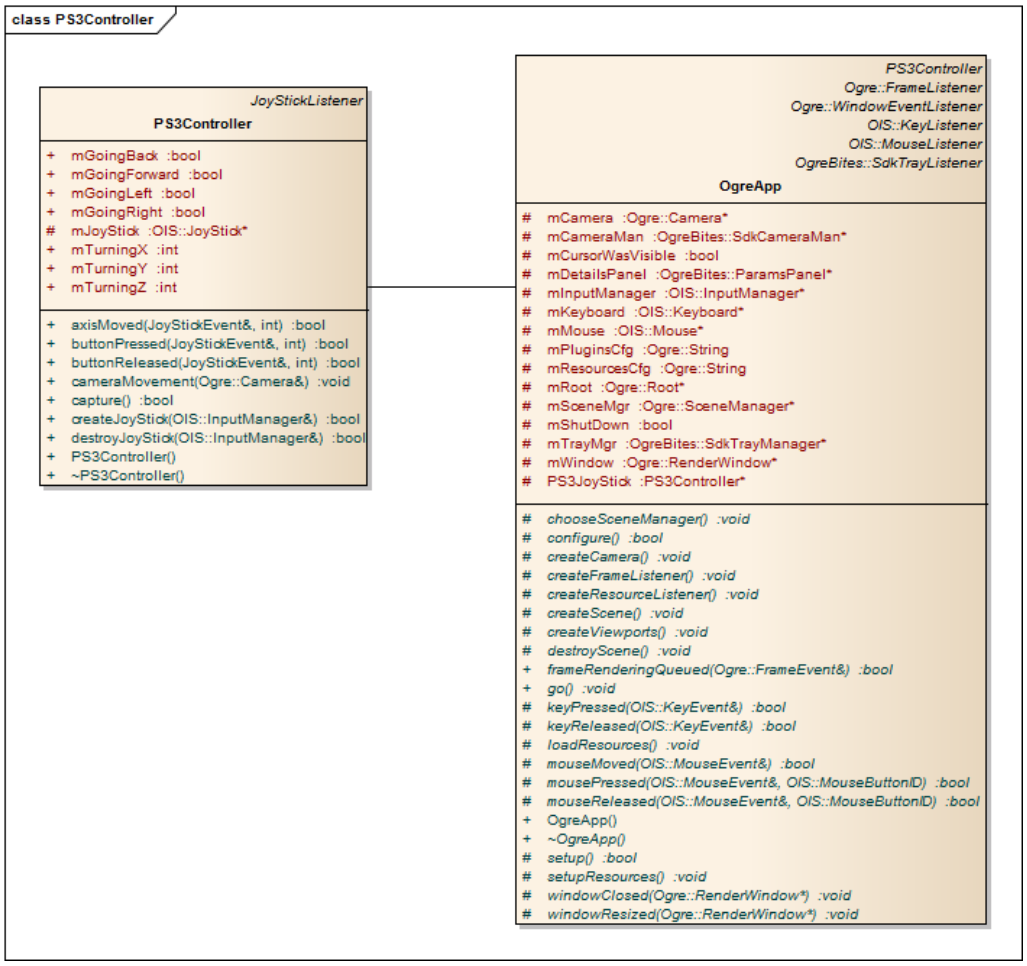

Ilustración 16: PS3Controller

4.3. Verificador de Dependencias

En el diseño se pensó en crear una clase que se encargara de verificar la existencia de dependencias en el sistema, sin embargo este procedimiento se puede realizar haciendo uso de una instrucción de CMake, las cual verifica que los paquetes estén instalados antes de la compilación.

Las instrucción *find_package(PACKAGE REQUIRED)*[39] permite buscar en el sistema la existencia de alguna librería determinada; en caso de no encontrarla, generará un mensaje de error para el usuario indicando que paquete no está presente. Esta instrucción se incluye en el

archivo *CMakeList.txt* y se utiliza por cada dependencia a buscar. En la siguiente imagen se muestran las líneas que permiten buscar las dependencias necesarias para este proyecto:

```
54
55  FIND_PACKAGE(ITK REQUIRED)
56  IF(NOT ITK_FOUND)
57    MESSAGE(FATAL_ERROR "ITK not found!")
58  ENDIF(NOT ITK_FOUND)
59  INCLUDE(${ITK_USE_FILE})
60
61  FIND_PACKAGE(OGRE)
62  IF(NOT OGRE_FOUND)
63    MESSAGE(FATAL_ERROR "OGRE not found!")
64  ENDIF(NOT OGRE_FOUND)
65  INCLUDE_DIRECTORIES(${OGRE_INCLUDE_DIRS})
66
67
68  find_package(OIS REQUIRED)
69  if(NOT OIS_FOUND)
70      message(SEND_ERROR "Failed to find OIS.")
71  endif()
```

Ilustración 17: Control de dependencias con CMake.

IV – RESULTADOS Y REFLEXIÓN SOBRE LOS MISMOS

1. Cumplimiento de objetivos

1.1. Objetivo general

Objetivo	Conclusión
Implementar una capa de interpretación que permita hacer uso del modelo Medical Interactive visualization Tool (MIsT), en la consola de videojuegos PlayStation3.	El Objetivo se cumplió dado que fue posible ejecutar aplicaciones que usan OGRE 3D en la PS3 incluyendo los prototipos de MiST utilizando el control DualShock 3 propio de la consola como interfaz hombre-máquina.

Tabla 10: Resultados Objetivo general

1.2. Objetivos específicos

Objetivo	Conclusión
Caracterizar el estado del arte acerca del desarrollo en la consola PS3 con OGRE 3D	El objetivo se logró generando un documento en el que se describe el estado del arte correspondiente.
Diseñar una capa de interpretación de OGRE 3D en la consola PS3.	El Objetivo se cumplió dado que se realizó el diseño arquitectónico de la capa que permite hacer uso de OGRE 3D en la consola, generando una serie de 5 diagramas basados en el modelo 4+1 vistas de Kruchten.
Implementar la capa diseñada anteriormente.	El objetivo se cumplió aunque no se llegó a implementar la totalidad del diseño propuesto por dificultades con el entorno de desarrollo, la capa fue implementada y probada. La parte que no se logró implementar se diseñó con el fin de optimizar el desempeño de OGRE3D en la consola aunque esto no hacia parte de los objetivos.
Validar la capa desarrollada, ejecutando el prototipo desarrollado con MIsT.	El objetivo se cumplió, pues se realizaron múltiples pruebas no solo con MiST sino con otras aplicaciones que usan OGRE 3D y se generó la documentación respectiva.

Tabla 11: Resultados Objetivos específicos

V – CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS

1. Conclusiones

Crear aplicaciones que funcionen de manera distribuida en el procesador CellBE es una tarea que requiere de bastante práctica y habilidad, aunque es difícil, es la mejor manera de sacar provecho del potencial que ofrece la arquitectura de éste procesador. Es indispensable contar con las herramientas adecuadas para realizar éste tipo de programación, de lo contrario, programar para el CellBE no es posible y se terminaría cayendo en un modelo de programación convencional, generando aplicaciones que tienen mejor desempeño en un PC de escritorio.

Una de las herramientas esenciales para este proceso es el sistema operativo, que desafortunadamente por las restricciones que ha impuesto Sony para prevenir la piratería, ya no son fáciles de instalar, actualizar y mantener. De ser posible, generar un sistema operativo para la PlayStation 3 que utilice la arquitectura del CellBE y que ofrezca un entorno de desarrollo adecuado, permitiría que ésta tecnología se pueda seguir explotando, pues en comparación con otros procesadores del mercado, el potencial que aún puede ofrecer la PS3 es grande.

En este trabajo es evidente el desaprovechamiento del procesador, pues en todas las pruebas tan solo se utilizó la PPU, mientras los 6 SPU están absolutamente en desuso. No tiene mucho sentido hacer uso de una herramienta tan poderosa como OGRE 3D si se desaprovecha todo el poder de procesamiento del CellBE, pues ejecutar OGRE 3D usando tan solo la PPU equivale a ejecutarlo en un PC de hace 10 años.

Con la llegada de la PS4, seguramente, todo el interés se volcará sobre esta nueva máquina y posiblemente el desarrollo sobre la PS3 disminuirá cada vez más al punto de desaparecer, sin embargo mientras la PS4 toma acogida y se genera investigación y conocimiento sobre ésta nueva consola, la PS3 puede ser aun explotada. Es entonces un buen momento para rescatar éste dispositivo y procurar crear un entorno de desarrollo que permita utilizarlo para el beneficio de las aplicaciones de computación gráfica que puedan sacar provecho de su gran potencial.

2. Recomendaciones

2.1. Para la Carrera

Los grupos de investigación deben ser no solo apoyados sino reforzados, todo el conocimiento generado en éstos es de vital importancia para el avance de la carrera. Es triste ver que espacios y equipos estén en desuso mientras se podría estar generando conocimiento y generando proyectos interesantes que impliquen el trabajo con tales dispositivos.

Sería oportuno usar la tecnología en su momento de auge y no hacia el final de su vida útil, de esta manera poder utilizar los trabajos de otros investigadores así como poder aportar al conocimiento colectivo.

Podría incentivarse a estudiantes de otras carreras a participar en los grupos de investigación de la carrera, pues la interdisciplinariedad permite generar nuevos proyectos o encontrar nuevas aplicaciones y usos para la tecnología con la que cuenta la carrera.

3. Trabajos Futuros

3.1. Sistema Operativo y SDK

Es necesario contar con un sistema operativo que esté diseñado especialmente para la PS3, aunque ya en el pasado se crearon algunos sistemas Linux especiales para el uso del CellBE, estos ya no cuentan con repositorios activos y se hace difícil la instalación de herramientas necesarias para poder desarrollar aplicaciones que usen todos los componentes del procesador. Crear o revivir uno de estos sistemas operativos sería de gran utilidad para el desarrollo de aplicaciones que puedan ser implementadas en la PS3.

De igual manera es de gran importancia contar con el SDK del procesador, pues es la colección de herramientas que lo componen, la que permite desarrollar aplicaciones que trabajan en paralelo en los SPE así como realizar la orquestación desde el PPE. Sería entonces importante buscar la forma de acceder a esté SDK y generar una versión multiplataforma que sea de fácil instalación y uso.

3.2. PS3Dev

Sony provee herramientas de desarrollo para personas o entidades que estén interesadas en crear aplicaciones y/o videojuegos en la PS3. Sería de importante acceder a estas herramientas para

desarrollar proyectos futuros en la consola. Por ejemplo, para optimizar el desempeño de OGRE 3D, teniendo en cuenta que el diseño arquitectónico se ha adelantado.

3.3. Control

Se puede extender la funcionalidad del control DualShock 3, cuyo uso no es exclusivo de la consola, en todo tipo de aplicaciones que requieran una interfaz hombre – máquina intuitiva y sencilla. Sería interesante investigar la forma de programar la vibración de los análogos para generar una retroalimentación al usuario; por ejemplo en el caso del simulador VRLS, se podría implementar alguna función que genere vibración en los análogos dependiendo de una condición determinada, como una colisión con un órgano o con otra herramienta.

VI – REFERENCIAS Y BIBLIOGRAFIA

1. Referencias

[1] R. S. Gallagher, *Computer Visualization: Graphics Techniques for Scientific and Engineering Analysis*, 1st ed. Florida: CRC Press, 1994.

[2] A. Maximo, G. Cox, C. Bentes, and R. Farias, "Unleashing the Power of the Playstation 3 to Boost Graphics Programming," in *2009 Tutorials of the XXII Brazilian Symposium on Computer Graphics and Image Processing*, 2009, pp. 45–58.

[3] N. Mejía, "MIsT (Medical Interactive visualization Tool)," Pontificia Universidad Javeriana, 2011.

[4] P. E. Q. Garcia, "VRLS: Virtual Reality Laparoscopic Simulator," 2012. [Online]. Available: http://pegasus.javeriana.edu.co/~CIS1310TK02/documents/Memoria_TG.pdf. [Accessed: 05-Jul-2013].

[5] Ogre3D, "OGRE – Open Source 3D Graphics Engine." [Online]. Available: http://www.ogre3d.org/.

[6] "ITK - Segmentation & Registration Toolkit." [Online]. Available: http://www.itk.org/.

[7] Sony, "PlayStation 3." [Online]. Available: http://es.playstation.com/ps3/. [Accessed: 05-Mar-2013].

[8] P. Luszczek, "A Rough Guide to Scientific Computing On the PlayStation 3," 2007.

[9] J. Kurzak, A. Buttari, P. Luszczek, and J. Dongarra, "The PlayStation 3 for High Performance Scientific Computing." 08-Feb-2008.

[10] K. Beck, *Extreme Programming Explained: Embrace Change*. Addison-Wesley P, 1999, p. 224.

[11] D. Berrueta, "Programación Extrema y Software Libre," 2006.

[12] "Programacion-Extrema." [Online]. Available: http://programacion-extrema.wikispaces.com/. [Accessed: 05-Apr-2013].

[13] Univalle, "Programación Extrema." [Online]. Available: http://eisc.univalle.edu.co/materias/WWW/material/lecturas/xp.pdf. [Accessed: 06-Apr-2013].

[14] P. Kruchten(Rational Software Corp.), "Architectural Blueprints—The '4+1' View Model of Software Architecture," *IEEE Softw. 12*, 1995.

[15] "PS3 Hardware: Explained - PS3Hax Network - Playstation 3 Hacks and Mods." [Online]. Available: http://www.ps3hax.net/showthread.php?t=5623. [Accessed: 15-Apr-2013].

[16] "Cell Programming IBM - PS3 Development Wiki." [Online]. Available: http://www.ps3devwiki.com/wiki/Cell_Programming_IBM. [Accessed: 10-May-2013].

[17] R. Linderman, "Early Experiences with Algorithm Optimizations on Clusters of Playstation 3," in *2008 DoD HPCMP Users Group Conference*, 2008, pp. 468–471.

[18] A. Buttari, J. Dongarra, and J. Kurzak, "Limitations of the PlayStation 3 for High Performance Cluster Computing." 03-Jul-2007.

[19] W. J. Schroeder, K. M. Martin, and W. E. Lorensen, "The Design and Implementation Of An Object-Oriented Toolkit For 3D Graphics And Visualization," vol. 1.

[20] "OKABU – HandCircus." [Online]. Available: http://www.handcircus.com/okabu/.

[21] "Ogre Forums • View topic - OKABU - PS3 (PSN) Downloadable title, using Ogre." [Online]. Available: http://www.ogre3d.org/forums/viewtopic.php?f=11&t=67101.

[22] "PlayStation - FedoraProject." [Online]. Available: http://fedoraproject.org/wiki/PlayStation.

[23] Fixstars, "Yellow Dog Linux 6.2," 2010. [Online]. Available: http://www.yellowdoglinux.com/products/ydl/. [Accessed: 01-Aug-2013].

[24] "The GNU library (glibc)." [Online]. Available: http://www.gnu.org/software/libc/. [Accessed: 01-Aug-2013].

[25] RedRibbon, "Linux Red Ribbon." .

[26] "El Compilador GCC." [Online]. Available: http://iie.fing.edu.uy/~vagonbar/gcc-make/gcc.htm. [Accessed: 06-Aug-2013].

[27] M. Scarpino, *Programming the Cell Processor: For Games, Graphics, and Computation*, 1st ed. Prentice Hall, 2008.

[28] PS3Dev, "PS3Dev." [Online]. Available: https://github.com/ps3dev. [Accessed: 11-Jun-2013].

[29] "RSX | NVIDIA." [Online]. Available: http://www.nvidia.es/object/IO_21296.html.

[30] "Anexo:Linux en PlayStation 3 - Wikipedia, la enciclopedia libre." [Online]. Available: http://es.wikipedia.org/wiki/Anexo:Linux_en_PlayStation_3.

[31] Sony, "Install OtherOs." .

[32] "Documentation Architecture." [Online]. Available: http://www.ogre3d.org/tikiwiki/tiki-index.php?page=Documentation+Architecture. [Accessed: 18-Oct-2013].

[33] "[Phoronix] RSXGL: OpenGL 3.1 Support For The PlayStation 3." [Online]. Available: http://www.phoronix.com/scan.php?page=news_item&px=MTE5MTg. [Accessed: 18-Oct-2013].

[34] A. Betts, "RSXGL/Github," 2013. [Online]. Available: https://github.com/gzorin/RSXGL. [Accessed: 18-Oct-2013].

[35] "SCE DevNet." [Online]. Available: http://scedev.net/ps3/.

[36] "PlayStation® Developer & Publisher Support - Become A Registered Developer." [Online]. Available: http://us.playstation.com/develop/.

[37] "Glxgears is not a Benchmark - cchtml.com." [Online]. Available: http://wiki.cchtml.com/index.php/Glxgears_is_not_a_Benchmark.

[38] "Basic Tutorials." [Online]. Available: http://www.ogre3d.org/tikiwiki/tiki-index.php?page=Basic Tutorials. [Accessed: 23-Sep-2013].

[39] "CMake:How To Find Libraries - KitwarePublic." [Online]. Available: http://www.cmake.org/Wiki/CMake:How_To_Find_Libraries.

[40] "Elements Interconnect Bus (Element Interconnect Bus EIB) | ubi-corp." [Online]. Available: http://www.ubi-corp.net/content/elements-interconnect-bus-element-interconnect-bus-eib.

[41] Psdevwiki.com, "OtherOS++." [Online]. Available: http://www.psdevwiki.com/ps3/OtherOS++. [Accessed: 02-Sep-2013].

[42] M. Gschwind, H. P. Hofstee, and M. Hopkins, "SYNERGISTIC PROCESSING IN CELLMULTICORE ARCHITECTURE," pp. 10–24, 2006.

[43] Homebrew.org, "MinVerCheck." [Online]. Available: http://www.homebrew-connection.org/files/PS3/PS3DowngradeChecker/min firmware downgrade/MinVerChk.rar. [Accessed: 06-Jun-2013].

[44] Elotrolado.net, "OtherOS++ de Graf_chokolo." [Online]. Available: http://www.elotrolado.net/hilo_otheros-de-graf-chokolo-y-otros-loaders-de-gnu-linux-09-02-2013_1589945. [Accessed: 02-Sep-2013].

[45] "Debian Linux Install GNU GCC Compiler and Development Environment." [Online]. Available: http://www.cyberciti.biz/faq/debian-linux-install-gnu-gcc-compiler/. [Accessed: 23-Sep-2013].

[46] "CMake: Quick Start Guide - MashWiki." [Online]. Available: https://secure.mash-project.eu/wiki/index.php/CMake:_Quick_Start_Guide. [Accessed: 23-Sep-2013].

[47] "OpenGL (Glut) with Linux Mint and Ubuntu 12.04. « Igor Barbosa." [Online]. Available: http://igorbarbosa.com/articles/opengl-glut-with-linux-mint-and-ubuntu-12-04/. [Accessed: 23-Sep-2013].

[48] "Habillitar el comando add-apt-repository en Debian - Taringa!" [Online]. Available: http://www.taringa.net/posts/linux/10764193/Habillitar-el-comando-add-apt-repository-en-Debian.html. [Accessed: 23-Sep-2013].

[49] "Installing the Ogre SDK." [Online]. Available: http://www.ogre3d.org/tikiwiki/Installing+the+Ogre+SDK?tikiversion=Linux. [Accessed: 23-Sep-2013].

[50] "OGRE 3D - Prerequisites." [Online]. Available: http://www.ogre3d.org/tikiwiki/Prerequisites?tikiversion=Linux. [Accessed: 23-Sep-2013].

[51] "Source « OGRE – Open Source 3D Graphics Engine." [Online]. Available: http://www.ogre3d.org/download/source. [Accessed: 23-Sep-2013].

[52] P. Quiñones, "Vrls: manual de instalación," pp. 1–10, 2013.

[53] "Find out more about Object Oriented Input System | SourceForge.net." [Online]. Available: http://sourceforge.net/projects/wgois/postdownload?source=dlp. [Accessed: 23-Sep-2013].

[54] I. Ruhnke, "jstest-gtk." [Online]. Available: https://github.com/Grumbel/jstest-gtk. [Accessed: 21-Oct-2013].

[55] "Sixaxis - Community Ubuntu Documentation." [Online]. Available: https://help.ubuntu.com/community/Sixaxis.

[56] "Welcome to QtSixA!" [Online]. Available: http://qtsixa.sourceforge.net/.

[57] "sixpair.c." [Online]. Available: https://help.ubuntu.com/community/Sixaxis?action=AttachFile&do=get&target=sixpair.c. [Accessed: 05-Oct-2013].

VII – ANEXOS

1. Glosario

API: *Application Programming Interface* o interfaz de programación de aplicaciones, es el conjunto de funciones y métodos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.}

CellBE: *CellBEBroadband Engine Processor* o procesador de banda ancha Cell, desarrollado en conjunto por Sony, Toshiba e IBM y hace parte del hardware de la consola PS3

Core: Núcleo de procesamiento independiente, cada procesador puede contar con uno o más cores o núcleos. El Procesador CellBE cuenta con 8 cores aunque en la PS3 solo 7 son operativos, 6 se puede utilizar en programación y el séptimo se encarga de la ejecución del sistema operativo.

CFW: *Custom Firmware* se refiere a versiones de *firmware* creadas por usuarios, es decir no oficiales.

DMA: *Direct Memory Access* o memoria de acceso directo, permite a cierto tipo de componentes de una computadora acceder a la memoria del sistema para leer o escribir independientemente de la unidad central de procesamiento principal.

EIB: *Element Interconnect Bus* o bus de interconexión de elementos, es un bus de comunicación interno del procesador Cell, cuya función es interconectar los diversos elementos del sistema en chip[40].

ENTITY: En OGRE, es una instancia de una malla que permite asignar diferentes materiales a cada entidad aunque estas compartan la misma malla.

FIXSTARS: Empresa creadora de Yellow Dog Linux.

GAMEPAD: Dispositivo de entrada usado para interactuar con un videojuego ya sea para consola o PC.

GPU: *Graphics Processing Unit* o unidad de procesamiento gráfico, es un coprocesador dedicado al procesamiento de gráficos u operaciones de coma flotante, para aligerar la carga de

trabajo del procesador central en aplicaciones como los videojuegos y o aplicaciones 3D interactivas.

Intrinsics: extensiones o librerías en leguaje C que hacen permiten hacer uso de los SPE del procesador CellBE, estos intrinsics mapean las instrucciones a lenguaje ensamblador, esto evita que el programador tenga que ir a bajo nivel [8].

LS: *Local Store memory* o memoria de almacenamiento local, en ella se guardan tanto datos como código para los SPE.

MFC: *Memory Flow Controller* o controlador de flujo de memoria, es el responsable de la transferencia de datos entre el procesador y los dispositivos E/S.

MULTITHREADING: Ejecución en múltiples hilos.

OFW: *Official Firmware* se refiere a versiones de *firmware* oficiales, es decir creadas por Sony.

OGRE: *Object-Oriented Graphics Rendering Engine* es un motor de 3D elaborado en C++.

OIS: Object-oriented Input System Library es una librería que permite usar los dispositivos de entrada como teclado, mouse o Pad de juego, para construir interfaces de usuario.

OVERLAPPING: Solapamiento.

OTHEROS: Era una característica con que contaban las consolas PS3 hasta el *firmware* OFM 3.20. Esta característica permitía la instalación de otros sistemas operativos como Linux.

OTHEROS++: Es una versión del antiguo *OtherOS* creada por usuarios de la PS3 para permitir la instalación de Linux en consolas con *firmware* CFW 3.55 [41].

PETITBOOT: Gestor de arranque gráfico que permite seleccionar imágenes del kernel de Linux al arrancar.

PITCH: Rotación en el eje x.

PORT: es el proceso de adaptación de software de manera que un programa ejecutable se puede crear para un entorno informático que es diferente de aquel para el cual fue diseñado originalmente.

PPU: *Physics Proccessing Unit* o unidad de procesamiento físico es un microprocesador dedicado diseñado para manejar los cálculos de física; es el motor de física de los videojuegos.

ROLL: Rotación en el eje z.

SAR: *Synthetic Aperture Radar* o radar de apertura sintética, es un tipo de sistema radar. Consiste en procesar mediante algoritmos la información capturada por la antena del radar.

SDK: *Software development kit* o kit de desarrollo, es un conjunto de herramientas de desarrollo de software; contiene las librerías requeridas para programar sobre el procesador.

SIMD'ization: es un método para el manejo de las restricciones en arquitecturas (SIMD) mediante la reorganización los datos.

SPU: *Synergistic Processing Unit* o unidad de procesamiento sinérgico es un motor de procesamiento de datos en paralelo destinado a proporcionar paralelismo en todos los niveles de abstracción[42].

VMX: *Vector Multimedia Extension unit* o unidad vectorial multimedia.

XMB: *Cross Menu Bar*, es el menú con que cuenta la PS3.

YAW: Rotación en el eje y.

YDL: *Yellow Dog Linux.*

2. Post-Mortem

2.1. Metodología propuesta vs. Metodología realmente utilizada.

Durante el desarrollo del trabajo de grado se vio la necesidad de ajustar la metodología que se había propuesto, debido a los inconvenientes encontrados durante la preparación del entorno de desarrollo. Inicialmente se pensó que el mayor esfuerzo estaría en la parte de programación de una capa que permitiera ejecutar OGRE en la consola, sin embargo realmente se debió dedicar demasiado tiempo y esfuerzo a la preparación de la consola para llegar en algún punto a realizar el desarrollo propuesto.

Este cambio, también generó modificaciones en las actividades planeadas inicialmente y afectó la consecución de los objetivos.

2.2. Actividades propuestas vs. Actividades realizadas.

El siguiente cuadro muestra las actividades propuestas para el desarrollo del trabajo de grado y su realización:

Id	Requerimientos	Tarea	Horas requeridas	Horas empleadas
RQ01	El sistema debe ejecutarse en la consola PS3	Flashear consola	3	48
		Configurar Other OS	2	6
		Instalar Linux	2	50
		Recompilar kernel de Linux	2	No se realizó
RQ02	El sistema debe desarrollarse en lenguaje C++	Instalar Compilador	1	1
		Instalar CMAKE	1	1
		Instalar el SDK del procesador CELL	2	Más de 50. No se consiguió
RQ03	El sistema debe ejecutar aplicaciones que usen OpenGL	Instalar Open GL	1	1
		Construir una aplicación con OpenGL	2	1
RQ04	El sistema debe ejecutar aplicaciones que usen RSXGL	Instalar RSXGL	1	Más de 10. No se consiguió
		Construir una aplicación con RSXGL	1	No se realizó
RQ05	El sistema debe ejecutar aplicaciones que usen OGRE 3D	Instalar prerrequisitos	2	3
		Compilar OGRE 3D	2	8
		Instalar OGRE 3D	2	1
		Instalar IDE	2	No se realizó

		Construir una aplicación con OGRE 3D	2	5
		Compilar una aplicación con OGRE 3D	1	5
		Ejecutar una aplicación con OGRE 3D	1	1
		Construir una aplicación con OGRE para el PPU	2	No se realizó
		Construir una aplicación OGRE 3D para los SPU	2	No se realizó
RQ06	El sistema debe ejecutar aplicaciones usen ITK	Instalar ITK	4	48
		Construir una aplicación con ITK	4	No se realizó
RQ07	El sistema debe permitir ejecutar el modelo MIsT	Ejecutar el Prototipo 1 de MIsT	4	1
		Ejecutar el Prototipo 2 de MIsT	4	1
		Ejecutar el Prototipo 3 de MIsT	4	1
RQ08	El sistema debe permitir ejecutar VRLS	Ejecutar el simulador VRLS	4	2
		Ejecutar el simulador VRLS usando el control PS3	4	48
RQ09	El sistema debe permitir al usuario interactuar mediante diferentes dispositivos	El sistema debe permitir al usuario interactuar por medio del Pad de juegos	1	50

		El sistema debe permitir al usuario interactuar por medio del teclado y el mouse	1	1
RQ10	El sistema debe ejecutar aplicaciones que usen PhyreEngine	Instalar Phyre Engine	2	No se realizó.
		Construir una aplicación con PhyreEngine	2	No se realizó

Tabla 12: Requerimientos y actividades

2.3. Efectividad en la estimación de tiempos del proyecto.

Las actividades fueron diseñadas pensando en un desarrollo común sin mayores inconvenientes lo cual generó un desfase de tiempos pues la mayor cantidad de tiempo y esfuerzo se invirtió en investigar e intentar configurar la máquina para conseguir un entorno de desarrollo adecuado y no en el desarrollo como tal. Finalmente no se consiguió la instalación de las herramientas necesarias para el desarrollo propuesto por lo que fue necesario ajustar el cronograma para cumplir los objetivos.

2.4. Costo estimado vs. Costo real del proyecto

Los costos presupuestados no superaron los reales, aunque surgieron algunos gastos que fueron contemplados como varios, la siguiente tabla describe los costos del proyecto:

Ítem	Valor Total	Detalle
Papelería	$ 100.000	Resma, fotocopias, tinta
Transporte	$ 960.000	2 Diarios por 6 días a la semana durante 16 semanas
Varios (Servicios Públicos)	$ 500.000	6 días a la semana durante 16 semanas
Alimentación	$ 1.500.000	5 días a la semana durante 18 semanas
PC	$ 1.500.000	5 días a la semana durante 18 semanas
Software	$ 5.000.000	Microsoft Word, Enterprise Architect

Material Consulta	$ 200.000	Recursos de biblioteca; Programming the CellBE Processor: For Games, Graphics, and Computation
Trabajo Estudiante	$ 2.688.000	384 horas: 4 horas diarias, 6 días a la semana, 16 semanas. $7000/hora
Total	$ 12.448.000	

Tabla 13: Costos del proyecto

2.5. Efectividad en la estimación y mitigación de los riesgos del proyecto.

La siguiente tabla muestra los riesgos que fueron contemplados para el desarrollo del proyecto y la clasificación según su probabilidad y su impacto dentro del trabajo de grado:

Impacto / Probabilidad	Bajo	Medio	Alto
Bajo	Aceptable	Aceptable	Tolerable
Medio	Aceptable	Tolerable	Considerable
Alto	Tolerable	Considerable	Catastrófico

Tabla 14: Clasificación de los riesgos.

Id	Riesgo	Probabilidad	Impacto	Clasificación Final del Riego	Mitigación
1	Mala estimación del tiempo asignado a las actividades	Alto	Alto	Catastrófico	Se realizará una verificación de los tiempos asignados a cada una de las actividades con el Director de Tesis
2	Retraso de las fechas definidas en el cronograma	Medio	Medio	Tolerable	Se mantendrá un control frecuente sobre el calendario vs las actividades programadas. De ser necesario se trabajará en

					domingo o festivo.
3	Falta de disponibilidad del PS3	Bajo	Bajo	Aceptable	Se establecerá en que horarios la consola se encuentra disponible para trabajar con ella.
4	Baja calidad en las actividades desarrolladas	Bajo	Alto	Tolerable	Se realizaran revisiones constantes junto al Director de Tesis para asegurar la calidad en las actividades desarrolladas
5	Dificultades en el diseño de la arquitectura	Medio	Alto	Considerable	Se consultará con el Director de Tesis o con otro profesor experto en arquitectura.
6	Dificultades en la implementación	Medio	Medio	Tolerable	Se consultará con el Director de Tesis o con otro profesor experto.
7	Dificultad para instalar S.O en el PS3	Alto	Alto	Catastrófico	Se realizará un *downgrade* del *fimware* de la consola, para lograr instalar el S.O. necesario. Se acudirá a las referencias bibliográficas para encontrar un procedimiento adecuado.
8	Disponibilidad baja por parte del Director de Tesis	Medio	Alto	Considerable	Se confirmará con anticipación las reuniones semanales. Se realizará comunicación vía correo electrónico.
9	Perdida de información y documentos	Baja	Alto	Tolerable	Almacenamiento y actualización en un repositorio en la nube.
10	Carga académica muy elevada	Baja	Medio	Tolerable	Se tendrá disponibilidad para trabajar horas extras los fines de semana y festivos.

11	El prototipo MIsT no se pueda ejecutar en la PS3	Medio	Alto	Considerable	Se documentaran las pruebas y se determinará el motivo por el cual el prototipo no se puede ejecutar en la consola.

Tabla 15: Riesgos estimados

Algunos de los riesgos contemplados se presentaron durante el desarrollo del proyecto, aunque algunos no lograron ser mitigados el impacto para el desarrollo de la investigación no es negativo.

La siguiente tabla muestra los riesgos presentados y las acciones tomadas:

Id	Riesgo	Mitigación
1	Mala estimación del tiempo asignado a las actividades	El cronograma se ajustó permitiendo invertir más esfuerzo en las actividades de mayor dificultad.
2	Retraso de las fechas definidas en el cronograma	El cronograma se ajustó permitiendo invertir más esfuerzo en las actividades de mayor dificultad.
5	Dificultades en el diseño de la arquitectura	Se consultó con el Director de trabajo de grado y se investigó acerca de la arquitectura tanto de OGRE como de OIS.
6	Dificultades en la implementación	Se consultó con el Director de Tesis, quien ayudó en la solución de dudas.
7	Dificultad para instalar S.O en el PS3	Mediante investigación se logró hacer *downgrade* de la consola y la instalación de diferentes sistemas operativos
8	Disponibilidad baja por parte del Director de	Se confirmará con anticipación las reuniones semanales. Se realizará

	Tesis	comunicación vía correo electrónico.
11	El prototipo MIsT no se pueda ejecutar en la PS3	Se documentaron las pruebas y se determinó que aunque el prototipo se puede ejecutar en la consola, su rendimiento no es aceptable.

Tabla 16: Riesgos presentados.

3. Guía de Downgrade

Inicialmente se debe verificar la versión de *Firmware* actual de la consola:

- Encender la consola y ubicar el menú Ajustes> Ajustes de sistema> Información de sistema.

EL siguiente paso es verificar la versión de Firmware de fábrica:

- Descargar MinVerCheck [43].
- En una memoria USB con formato FAT de 32bits, crear un directorio llamado PS3, dentro, crear un directorio llamado UPDATE y copiar el archivo PS3UPDAT.PUP.
- Encender la consola e insertar la memoria USB.
- Navegar por el menú en el siguiente orden, Ajustes> Actualización del sistema> Actualizar desde dispositivo de almacenamiento.
- El sistema instalará el *Firmware* versión 3.xx si la versión de fábrica es 3.55 o inferior, de lo contrario el sistema indicará en pantalla cual es la mínima versión a la que se puede hacer Downgrade que en este caso es la versión de fábrica.

4. Guía de instalación de OtherOS

En una memoria USB con formato FAT de 32bits, crear un directorio llamado PS3, dentro, copiar los archivos “rr_otheros_installer” y “rr_otheros_loader”. Insertar la USB en la consola, una vez encendida, navegar por el menú Juego> Packages> e instalar rr_otheros_installer, esta instalación lanzará un menú en modo texto con las siguientes opciones:

1. Exit to GameOS
2. Restore to factory mode
3. Install PetitBoot
4. Free reserved space
5. Reserve 8 GB
   16 GB
   24 GB
   32 GB

Inicialmente se debe elegir la opción 3. Install PetitBoot, al finalizar regresará al menú inicial.

Luego se debe elegir una de las opciones del numeral 5, dependiendo el tamaño que se quiere reservar para la instalación de Linux (para este trabajo de grado se utilizó la opción de 24 GB). El programa pedirá confirmación para formatear y particionar el disco duro y reiniciara la consola. Una vez la consola se reinicie, aparecerá un erro indicando que es necesario poner de nuevo el *firmware*, se debe ignorar este mensaje presionando el botón PS del control.

Al terminar iniciará el menú XMB, se debe navegar por el menú Juego> Packages> e ejecutar rr_otheros_loader. Este programa lanza el menú PetitBoot en modo texto con diferentes opciones de arranque, desde este punto ya es posible realizar la instalación de Linux [44].

5. Guía de instalación Red Ribbon

Desde el menú PetitBoot y teniendo el DVD con Linux Red Ribbon insertado en la unidad BlueRay de la PS3, se elige la primer opción del menú y se espera a que se lance el Asistente de instalación.
Se deben seguir los pasos típicos de instalación del sistema operativo. Para este caso particular se indican los siguientes datos:

Instalación de paquetes:

Compilador GCC [45].
Cmake [46]: Desde una terminal ejecutar los siguientes comandos:

- sudo apt-get install cmake cmake-gui cmake-curses-gui

OpenGL [47].

OGRE 3D:
Habilitar el comando app-apt-repository [48].
Agregar repositorio [49].

Prerrequisitos [50]: desde una terminal ejecutar los siguientes comandos,

- sudo apt-get install build-essential automake libtool
- sudo apt-get install libfreetype6-dev libfreeimage-dev libzzip-dev libxrandr-dev libxaw7-dev freeglut3-dev libgl1-mesa-dev libglu1-mesa-dev sudo apt-get install nvidia-cg-toolkit libois-dev libboost-thread-dev

OGRE 3D [51]: Se descarga y posteriormente se descomprime, compila e instala [52],

- tar xjf ogre_src_v1-8-1.tar.bz2
- cd ogre_src_v1-8-1.
- mkdir build
- cd build
- cmake ..
- make && make install.

Plugin OIS [53]:

- Descargar, Descomprimir y dar permisos.
- En una terminal ejecutar los comandos:
    - ./bootstrap
    - ./configure
    - make && make install.
- Copiar los archivos desde /usr/local/include/OIS/ a /usr/local/include/OGRE/

6. Guía de configuración conexión control PS3 Dualshock

El control de la PS3 tiene dos modos de conexión a la consola, mediante cable USB y conexión Bluetooth. Cada uno de estos tipos de conexión requiere una configuración para que el sistema operativo reconozca el gamepad.

La conexión mediante cable USB es reconocida de manera directa y se puede verificar consultando la ruta "/dev/input", siendo el dispositivo nombrado como jsx (x es un número que identifica el control y por lo general es cero (0), siendo en este caso js0 el identificador del gamepad).

Para el caso de la conexión Bluetooth, es necesario realizar la instalación y configuración de varias herramientas descritas a continuación:

HERRAMIENTA	DESCRIPCION	DEPENDENCIAS
X11	Sistema de ventanas para linux	
JSTEST-GTK[54]	Permite hacer *tests* a los joysticks o gamepads conectados a la consola	Scons sigc++ gtkmm exapt
Sixaxis[55]	Manejador de gamepads	Libusb Sixpair
QtSixa[56]	Permite que el sistema operativo reconozca el gamepad	Libusb Libbluetooth3 Libdbus-1 Libdbus-glib-1 Libjack Python-qt4-dev bluez

Tabla 17: Herramientas Gamepad Bluetooth

- Instalar prerrequisitos:
    - sudo apt-get install libusb-dev libusb-0.1-4
    - Conectar el control con el cable USB.
    - Descargar sixpair.c [57].
    - Compilar sixpair (gcc -o sixpair sixpair.c -lusb).
    - Correr sixpair (sudo ./sixpair).
    - Sixpair mostrará la siguiente información

        Current Bluetooth master: xx:xx:xx:xx:xx:xx

        Setting master bd_addr to xx:xx:xx:xx:xx:xx.

- Monitorear dispositivos Bluetooth:
    - sudo /etc/init.d/bluetooth stop
    - sudo hidd --server --nocheck –n
    - Presionar el botón PS del control, se debe ver la siguiente información:

        hidd[pid]: Bluetooth HID daemon

        hidd[pid]: New HID device 00:19:C1:xx:xx:xx (Sony Computer Entertainment Wireless Controller)

    - Presionar Ctrl + c para detener el proceso e iniciar el servicio Bluetooth.

        sudo /etc/init.d/bluetooth start.

Una vez configurados los dos tipos de conexión del control, se debe verificar que OIS reconozca y permita usar todos los componentes del *gamepad* (botones y análogos). Para esto se puede ejecutar una de las aplicaciones que incluye OIS en sus demos llamada ConsoleApp que muestra los periféricos conectados al sistema (mouse, teclado, gamepads):

>: cd /ruta a /ObjectOriented_InputSystem/demos/
>: ./ConsoleApp.

7. Pruebas con OpenGL

Para compilar los programas:

>:/ cd /ruta a/ PruebaOpenGL/

>:/ g++ - o main main.cpp –lGL –lGLU –lglut

>:/ g++ - o main2 main2.cpp –lGL –lGLU –lglut

>:/ ./main

>:/ ./main2

Para ejecutar glxgears:

>:/ sudo apt-get mesa-utils

>:/glxgears

8. Prueba con OGRE 3D

Para compilar:

>:/ cd /ruta a/ PruebaOGRE3D/build

>:/ ccmake ..

Configurar y generar.

>:/ make

>:/ cd /dist/bin

>:/ ./OgreApp

9. PS3Controller

Esta clase debe ser referenciada desde las aplicaciones que usen OGRE 3D, haciendo un *include* en la sección de librerías

```
20  #include <OgreCamera.h>
21  #include <OgreEntity.h>
22  #include <OgreLogManager.h>
23  #include <OgreRoot.h>
24  #include <OgreViewport.h>
25  #include <OgreSceneManager.h>
26  #include <OgreRenderWindow.h>
27  #include <OgreConfigFile.h>
28
29  #include <OISEvents.h>
30  #include <OISInputManager.h>
31  #include <OISKeyboard.h>
32  #include <OISMouse.h>
33
34  #include "PS3Controller.h"
```

Hacer que la clase herede de PS3Controller

```
56
57  class BaseApplication : public Ogre::FrameListener, PS3Controller,
```

Se debe declarar un objeto PS3Controller

```
108     //OIS Input devices
109     OIS::InputManager* mInputManager;
110     OIS::Mouse*    mMouse;
111     OIS::Keyboard* mKeyboard;
112     //Joystick
113     PS3Controller* PS3JoyStick;
114 };
```

E inicializarlo en el constructor

```
BaseApplication::BaseApplication(void)
    :
    PS3JoyStick(),
    mRoot(0),
    mCamera(0),
    mSceneMgr(0),
    mWindow(0),
    mResourcesCfg(Ogre::StringUtil::BLANK),
    mPluginsCfg(Ogre::StringUtil::BLANK),
    mTrayMgr(0),
```

Invocar el método *createJoyStick* en la función que crea el *FrameListener*

```
93  void BaseApplication::createFrameListener(void)
94  {
95      Ogre::LogManager::getSingletonPtr()->logMessage("*** Initializing OIS *
96      OIS::ParamList pl;
97      size_t windowHnd = 0;
98      std::ostringstream windowHndStr;
99
100     mWindow->getCustomAttribute("WINDOW", &windowHnd);
101     windowHndStr << windowHnd;
102     pl.insert(std::make_pair(std::string("WINDOW"), windowHndStr.str()));
103
104     mInputManager = OIS::InputManager::createInputSystem( pl );
105
106     PS3JoyStick->createJoyStick(*mInputManager);
```

Invocar el método *capture* en la función *frameRenderingQueued*

```
258 bool BaseApplication::frameRenderingQueued(const Ogre::FrameEvent& evt)
259 {
260     if(mWindow->isClosed())
261         return false;
262
263     if(mShutDown)
264         return false;
265
266     //Need to capture/update each device
267     mKeyboard->capture();
268     mMouse->capture();
269     PS3JoyStick->capture();
270
271     mTrayMgr->frameRenderingQueued(evt);
```

Invocar el método *cameraMovement* en la función *frameRenderingQueued*

```
283                 mDetailsPanel->setParamValue(6, Ogre::StringConverter::
284                 mDetailsPanel->setParamValue(7, Ogre::StringConverter::
285             }
286         }
287
288     PS3JoyStick->cameraMovement(*mCamera);
289
290     return true;
291 }
```