UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

DESARROLLO DE UN JUEGO BASADO EN TETRIS BAJO UN AMBIENTE TRIDIMENSIONAL CONTROLADO POR UN ACELERÓMETRO

TRABAJO ESPECIAL DE GRADO

Presentado ante la

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

Como parte de los requisitos para optar al título de

INGENIERO EN INFORMÁTICA

REALIZADO POR ALBERTO JOSÉ GÜERERE MONSALVE

PROFESOR GUIA ING. JOSÉ ENRIQUE REYES ESCALONA

FECHA 12 de Marzo de 2010

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

DESARROLLO DE UN JUEGO BASADO EN TETRIS BAJO UN AMBIENTE TRIDIMENSIONAL CONTROLADO POR UN ACELERÓMETRO

Este Jurado; una vez realizado el examen del presente trabajo ha evaluado su contenido con el resultado: Diecisiete puntos

J U R A D O E X A M I N A D O R

Firma:
Nombre: Ricardo Casanova

Firma:
Nombre: Jose Reyes

Firma:
Nombre:

REALIZADO POR ALBERTO JOSÉ GÜERERE MONSALVE

PROFESOR GUIA ING. JOSÉ ENRIQUE REYES ESCALONA

FECHA 12 de Marzo de 2010

ÍNDICE GENERAL

DEDICATORIA i

AGRADECIMIENTOS ii

ÍNDICE GENERAL iii

LISTA DE ILUSTRACIONES v

LISTA DE TABLAS vii

SINÓPSIS viii

INTRODUCCIÓN 1

CAPÍTULO I 2

I.1. Planteamiento del Problema 2

I2 .Objetivo General 3

I2.1.Objetivos Específicos 3

I.3. Alcance y Limitaciones. 3

I.3.1. Alcance 3

I.3.2. Limitaciones 3

I.4. Justificación 4

CAPÍTULO II 5

II.1. MARCO TEÓRICO 5

II.1.1. Tetris 5

II.1.2 Interacción Hombre-Computador 6

II.1.3. Control Inalámbrico del Nintendo Wii 7

II.1.4. Nunchuk 8

II.1.5. Acelerómetro 9

II.1.6. Bluetooth 12

II.1.7. WiimoteLib 12

II.1.8. Computación Grafica 13

II.1.9. Realidad Virtual 14

II.1.10. OGRE 3D 16

II.1.11. Mogre 17
II.1.12. Autodesk Maya 17
II.1.13. Shaders 17
II.1.14. Sombreado de Phong 18
CAPITULO III 19
III. Marco metodológico 19
CAPÍTULO IV 24
IV.1. Desarrollo 24
IV.1.1 Iteración 1 24
IV.1.2 Iteración 2 31
IV.1.3 Iteración 3 38
IV.1.4 Iteración 4 44
CAPÍTULO V 54
V.1. RESULTADOS 54
CONCLUSIONES 57
RECOMENDACIONES 59
BIBLIOGRAFÍA 60
APÉNDICE A: DIAGRAMA DE CLASES DE LA ITERACIÓN 2 63
APÉNDICE B: DIAGRAMA DE CLASES DE LA ITERACIÓN 4 64
APÉNDICE C: DETALLE DEL CONVERTIDOR DE COORDENADAS DE LA MATRIZ EN MEMORIA A COORDENADAS DEL MOTOR GRÁFICO 65
APÉNDICE D: UBICACIÓN INICIAL DE LOS NODOS AUXILIARES DE LAS PIEZAS FORMADAS66
APÉNDICE E: DETALLE DE IMPLEMENTACIÓN DEL CICLO Y LAS REGLAS DEL JUEGO 67
APÉNDICE F: USO DE LA LIBRERÍA WIIMOTELIB 70
APÉNDICE G: TRIGONOMETRÍA APLICADA PARA OBTENER EL CUADRANTE DONDE SE ENCUENTRA LA CÁMARA 72
APÉNDICE H: FOTOGRAFÍAS DEL USO DEL WIIMOTE PARA LA ROTACIÓN DE LAS PIEZAS . 74
APÉNDICE I: FOTOGRAFÍAS DEL USO DEL NUNCHUK PARA EL MOVIMIENTO DE LA CÁMARA 76

LISTA DE ILUSTRACIONES

Ilustración 1: Captura de pantalla del juego de Tetris de Pázhitnov 5
Ilustración 2: Tetrominós, piezas del juego Tetris 6
Ilustración 3: Persona interactuando con un computador utilizando equipos de realidad virtual 7
Ilustración 4: Wiimote 8
Ilustración 5: Nunchuk conectado a un Wiimote sostenidos por un usuario 9
Ilustración 6: Partes de un acelerómetro piezoeléctrico 10
Ilustración 7: Módulo del acelerómetro usado en el Wiimote (AnalogDevicesADXL330) .. 11
Ilustración 8: Ejemplo de gráficos 3D generados por computador 13
Ilustración 9: Usuario con casco y guantes de realidad virtual 15
Ilustración 10: Escena renderizada por el motor gráfico Ogre 3D 16
Ilustración 11: Escena 3D renderizada con shaders aplicados 17
Ilustración 12 : Ejemplo de sombreado de Phong 18
Ilustración 13: Diagrama de la metodología basada en prototipos evolutivos 21
Ilustración 14: Diagrama de la adaptación de la metodología basada en prototipos evolutivos propuesta 23
Ilustración 15: Modelo del cubo base 24
Ilustración 16: Borde del área de juego 25
Ilustración 17: Borde con las dimensiones del área de juego 27
Ilustración 18. Pieza T con coordenadas de los nodos auxiliares relativas al nodo base. ... 28
Ilustración 19: Ventana de renderización y ventana de verificación de la matriz en memoria 33
Ilustración 20: Rotación de la pieza en forma de Z 34
Ilustración 21: Ventana de verificación de la matriz en memoria 36
Ilustración 22: Modelo del piso 3D cargado en la herramienta Autodesk Maya 39
Ilustración 23: Nomenclatura de los ejes con respecto al Wiimote 45
Ilustración 24: Esfera del movimiento de la cámara 50

Ilustración 25: Diagrama de clases de la iteración 2 63
Ilustración 26: Diagrama de clases de la iteración 4 64
Ilustración 27: Rotación de la pieza sobre el eje Z paso 1 74
Ilustración 28: Rotación de la pieza sobre el eje Z paso 2 74
Ilustración 29: Rotación de la pieza sobre el eje X paso 1 75
Ilustración 30: Rotación de la pieza sobre el eje X paso 2 75
Ilustración 31: Movimiento de la cámara usando el Nunchuk paso 1 76
Ilustración 32: Movimiento de la cámara usando el Nunchuk paso 2 76
Ilustración 33: Movimiento de la cámara usando el Nunchuk paso 3 77

LISTA DE TABLAS

Tabla 1: Tipos de dispositivos Bluetooth por alcance 12
Tabla 2: Métodos de Mogre utilizados en la iteración 1 27
Tabla 3: Descripción de los manejadores de eventos de teclado utilizados 29
Tabla 4: Asignación de las teclas en la iteración 3 40
Tabla 5: Acciones asignadas a los botones del Nunchuk 46
Tabla 6: Acciones asignadas a los botones del Wiimote 47
Tabla 7: Eventos y métodos de la librería WiimoteLib utilizados 48
Tabla 8: Estados de la cámara con respecto al cuadrante donde se encuentre 50

SINÓPSIS

Este Trabajo Especial de Grado titulado "Desarrollo de un Juego basado en Tetris bajo un Ambiente Tridimensional Controlado por un Acelerómetro" consiste en la conceptualización, diseño y construcción de una versión del juego de video Tetris en un ambiente tridimensional que permite ser controlado con movimientos de la mano haciendo uso del acelerómetro integrado al control inalámbrico de la consola de juegos de video Nintendo Wii.

El desarrollo de este trabajo se realizó utilizando una adaptación de la metodología basada en prototipos evolutivos en el transcurso de cuatro iteraciones durante las cuales la construcción del prototipo fue dividida con respecto al nivel de complejidad de los algoritmos que fueron diseñados e implementados, añadiendo funcionalidades a medida que fue necesario.

Se decidió implementar la aplicación en torno a Ogre 3D, un motor gráfico tridimensional de alto nivel, a través de la interfaz Mogre que permite desarrollar en el lenguaje de programación C#. Se diseñaron y construyeron los modelos tridimensionales utilizando la herramienta Autodesk Maya.

La interacción humano-computador lograda con el control inalámbrico de la consola Nintendo Wii demuestra que es posible aumentar el desempeño del usuario ante una tarea complicada con tecnología que le ofrezca una experiencia intuitiva y gratificante, permitiendo desarrollar aplicaciones que utilicen conceptos existentes con características innovadoras y de vanguardia manteniendo el grado de entretenimiento logrado en implementaciones pasadas.

INTRODUCCIÓN

El control inalámbrico de la consola de juegos de video Nintendo Wii ha recibido mucha atención debido a sus características únicas en contraste con los controles de juegos de video típicos. Ha encontrado gran acogida en los hogares gracias a su capacidad de detección de movimiento.

El Tetris es un juego de video muy popular en todo el mundo. Sus sencillas reglas permiten aprender a lograr el objetivo del juego rápidamente y da al usuario una satisfacción prácticamente inmediata gracias a que su manejo es intuitivo. Dado sus exitosas ventas a lo largo de más de veinticinco años, se han desarrollado versiones de Tetris que hacen uso de tecnologías actuales como los ambientes tridimensionales.

Así como el Tetris en tres dimensiones, se han desarrollado muchos productos de software de entretenimiento con características que hoy en día son la norma, pero que no logran ofrecer al usuario una experiencia gratificante debido a que su implementación exige grandes niveles de concentración. Esta situación es más evidente en los juegos de video cuyo objetivo es resolver rompecabezas porque las habilidades necesarias de deducción, reconocimiento de patrones y de razonamiento inductivo se ven comprometidas al interactuar con un computador de una manera que implique distracción.

En torno a las circunstancias descritas anteriormente, se planteó hacer una implementación de Tetris bajo un ambiente tridimensional controlado por un dispositivo que permita expresar movimiento en tres dimensiones de manera intuitiva haciendo uso de un acelerómetro, siendo el candidato ideal el control inalámbrico de la consola de juegos de video Nintendo Wii.

CAPÍTULO I

I.1. Planteamiento del Problema

Los dispositivos de entrada de un computador doméstico representan un gran reto a la hora de diseñar la interfaz humano-computador de un desarrollo de software de entretenimiento, ya que el hardware del que puede disponer un usuario no siempre se escapa de los usuales teclado y ratón; limitando así la creatividad del desarrollador. Incluso los "dispositivos para juegos", o también conocidos como Joystick y Gamepad, disponibles en el mercado actualmente diseñados para ser utilizados con un PC no ofrecen el grado de libertad necesario para crear un juego innovador.

Tomemos como ejemplo el caso del famoso juego Tetris. Originalmente diseñado para ambientes bidimensionales, este juego logró comprobar que unas sencillas reglas, combinadas con una interfaz hombre-computador ideal producen un grado de entretenimiento sin igual. Por supuesto, en su momento de apogeo (finales la década de 1980 y principios de 1990), los computadores y otros dispositivos que podían ejecutar diferentes versiones del juego tenían limitaciones que hoy en día no son obstáculo para, por ejemplo, reproducirlo en un ambiente tridimensional. Y, aunque a mediados de la década de 1990 hubo intentos de hacerlo, estas versiones no fueron satisfactorias por no ofrecerle al usuario la naturalidad necesaria para poder lograr un grado de entretenimiento si quiera parecido al juego original en dos dimensiones.

Un posible acercamiento al desarrollo de una versión de Tetris que logre darle al usuario la naturalidad necesaria para jugar en un ambiente tridimensional sería combinar un motor para gráficos tridimensionales de última generación con un dispositivo ideal para hacer una interfaz humano-computador: el control inalámbrico de la consola para juegos Nintendo Wii. Éste control incluye un acelerómetro, dispositivo capaz de dar una referencia a la gravedad de la Tierra permitiendo determinar grados de rotación del control en tres ejes, de forma inalámbrica a través del protocolo Bluetooth. Además, es un dispositivo de costo relativamente bajo (unos $40) y que puede ser encontrado en más de setenta millones de hogares hoy en día.

I2 .Objetivo General

Desarrollar un juego tipo Tetris en un ambiente tridimensional utilizando como dispositivo controlador principal el acelerómetro integrado al control inalámbrico de la consola de juegos Nintendo Wii.

I2.1.Objetivos Específicos

- Conceptualización, diseño y creación de los modelos tridimensionales del juego.
- Conceptualización, diseño y creación del escenario de juego.
- Implementación de la lógica y reglas del juego.
- Diseño e implementación de la aplicación en torno al motor de gráficos Object-Oriented Graphics Rendering Engine (OGRE 3D).
- Implementación de la interpretación necesaria de los datos obtenidos del acelerómetro incorporado al control inalámbrico de Nintendo Wii para utilizarlo como sensor de orientación de las piezas del juego, empleando alguna de las librerías disponibles para tal fin.
- Integración de la lógica y reglas del juego con el ambiente tridimensional y el control inalámbrico.

I.3. Alcance y Limitaciones.

I.3.1. Alcance

- Se desarrollará un ambiente tridimensional para el área de juego
- Se implementarán las reglas básicas del juego de Tetris
- Cada pieza del rompecabezas tendrá un color característico
- La comunicación con el control será inalámbrica a través del protocolo Bluetooth

I.3.2. Limitaciones

- El juego será controlado por un (1) control inalámbrico de Nintendo Wii
- Las piezas se rotarán en ángulos rectos
- La calidad de los gráficos estará limitada a probar el concepto del juego

- Sólo se permitirá un jugador a la vez
- No se guardará la partida en curso si el jugador desea abandonar la aplicación

I.4. Justificación

El presente trabajo propone la creación de un juego tipo Tetris que use el mando de Wii para detectar una referencia a la gravedad con base en la cual rotar las piezas del juego y lograr el objetivo de crear planos utilizando los bloques de las piezas, eliminando los planos completos y ganando puntos cada vez que esto se logre.

Este trabajo deberá explorar nuevas tecnologías disponibles al consumo masivo que permitan una mejor interacción humano-computador, utilizando en este caso un control inalámbrico que logra detectar inclinaciones que se le apliquen, a través de una referencia a la fuerza de gravedad de la Tierra.

Dicha exploración permitirá que otras investigaciones se lleven a cabo al demostrar que es posible tomar una idea antigua y ya implementada con la tecnología disponible en su momento, y convertirla en un concepto innovador, dándole vida de nuevo utilizando tecnología de punta.

Queda claro que estas investigaciones le permitirán al consumidor obtener productos innovadores que mejoren la calidad de la interacción humano-computador, permitiendo así una mayor usabilidad (o diversión y alcance de edades más amplio) generando más demanda por productos de este tipo, en definitiva dándole a los desarrolladores la capacidad de ofrecer mejores productos.

CAPÍTULO II

II.1. MARCO TEÓRICO

II.1.1. Tetris

El Tetris es uno de los juegos de video de rompecabezas más populares de la historia. Ha sido llevado a todas las plataformas conocidas por el hombre, computadoras y consolas de video.

Fue creado en el año 1985 por Alexey Pázhitnov quién lo programó para un computador soviético llamado Elektronika 60, mientras que fueron Dmitri Pavlovsky y Vadim Gerasimov quienes lo llevaron al PC (ver ilustración 1). [Gerasimov, Sin Fecha]

Tiene unas reglas muy sencillas. El jugador deberá colocar una pieza que cae verticalmente en el fondo del espacio de juego de manera tal que logre completar líneas horizontales con los cuadros que forman las piezas. Las piezas se van apilando verticalmente, y cuando el jugador llega a completar una línea horizontal esa línea se elimina, los cuadros que están encima de ella caen hacia abajo y el jugador acumula puntos. El juego termina cuando las piezas apiladas llegan al tope superior del espacio de juego.

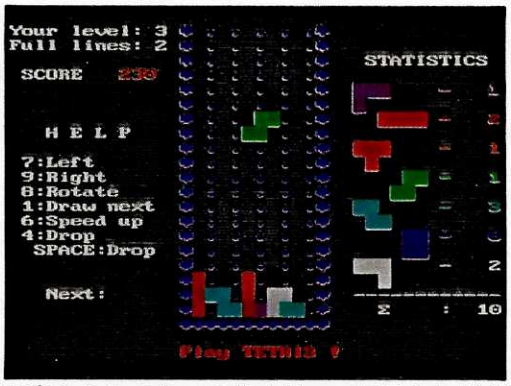

Ilustración 1: Captura de pantalla del juego de Tetris de Pázhitnov
[Fuente: http://titov-en.livejournal.com/24530.html]

Las piezas del Tetris son tetrominós, es decir, formas geométricas compuestas por cuatro cuadrados conectados ortogonalmente y son un tipo particular de polinomios como los dominós o los pentominós. [Golomb, 1996]

Los siete tetrominós que pueden formarse son (ver ilustración 2):

- Pieza I

- Pieza T
- Pieza cuadrado
- Pieza L
- Pieza L invertida
- Pieza Z invertida
- Pieza Z

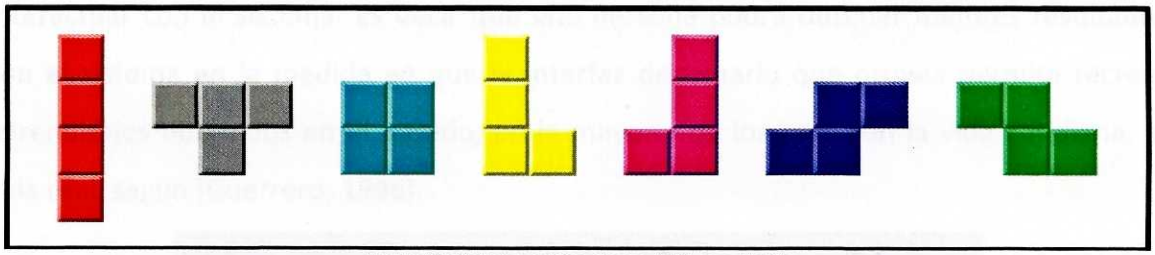

Ilustración 2: Tetrominós, piezas del juego Tetris
[Fuente: http://www.cs.uic.edu/~troy/spring06/cs340/mps/mp3.html]

II.1.2 Interacción Hombre-Computador

La investigación sobre la interacción hombre-computador fue comenzada por Donald Norman, psicólogo cognoscitivo que trabajó sobre el concepto de la usabilidad. Comprende el estudio de la interacción entre las personas y los computadores y es una materia multidisciplinaria que relaciona las ciencias de la computación, psicología, ergonomía, diseño y muchas otras áreas (ver ilustración 3).

La interacción hombre-computador se realiza a través de una interfaz de usuario formada por hardware y software; y es utilizada para la manipulación de periféricos de computadores y otras máquinas como automóviles hasta plantas nucleares.

El desempeño humano para el uso de computadores y sistemas de información ha sido un área de investigación y desarrollo que se ha expandido mucho en las últimas décadas. Para ofrecer a un usuario humano un medio adecuado a través del cual pueda interactuar con un computador deben tenerse en cuenta las limitaciones de la capacidad humana y las restricciones de la tecnología existente.

Debido a que la mayoría de las personas no hacen uso del raciocinio inductivo o deductivo cuando se enfrentan a una situación desconocida, es común encontrar

interfaces de usuario que son difíciles de usar, confusas y hasta frustrantes a pesar de que los desarrolladores de la interfaz invierten mucho tiempo en su diseño.

Al contrario, comúnmente las personas aplican un conjunto de heurísticas basadas en procedimientos, reglas y estrategias obtenidas de experiencias pasadas. Dichas heurísticas tienden a ser específicas de un dominio particular, y por tal razón la interfaz de usuario desarrollada debe permitir a la persona hacer uso de esas heurísticas para interactuar con el sistema. Es decir que una persona podrá obtener mejores resultados con el sistema en la medida en que la interfaz de usuario que provea permita recrear aprendizajes obtenidos en el pasado, en la mayoría de los casos en la vida cotidiana, la vida real, según [Guerrero, 1996].

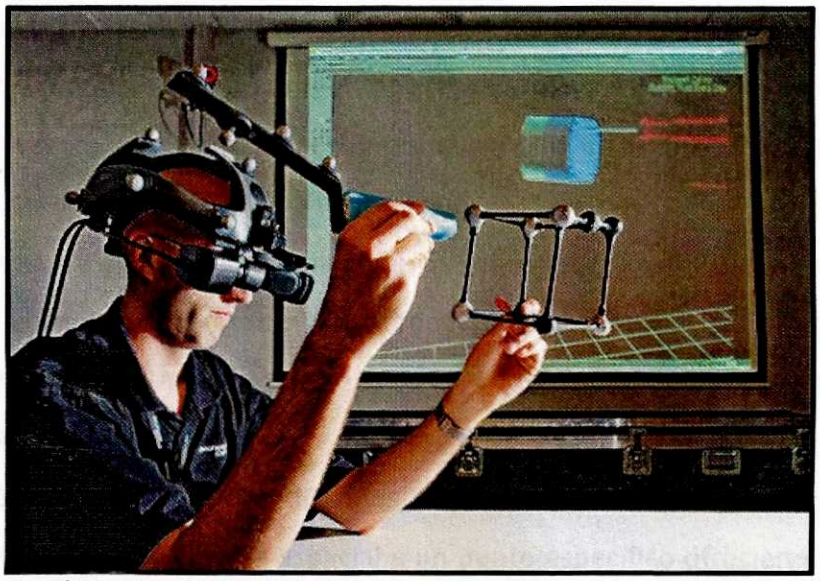

Ilustración 3: Persona interactuando con un computador utilizando equipos de realidad virtual [Fuente: http://iowa.com/2009/02/simulation-lab-at-rockwell-collins/]

II.1.3. Control Inalámbrico del Nintendo Wii

El control inalámbrico de la consola de juegos Nintendo Wii, también conocido como Wiimote, es un dispositivo que incluye botones, diodos luminosos indicadores, un motor de vibración, una pequeña corneta, una cámara sensible a la luz infrarroja, un sistema de comunicación inalámbrico basado en tecnología Bluetooth y un acelerómetro (ver ilustración 4).

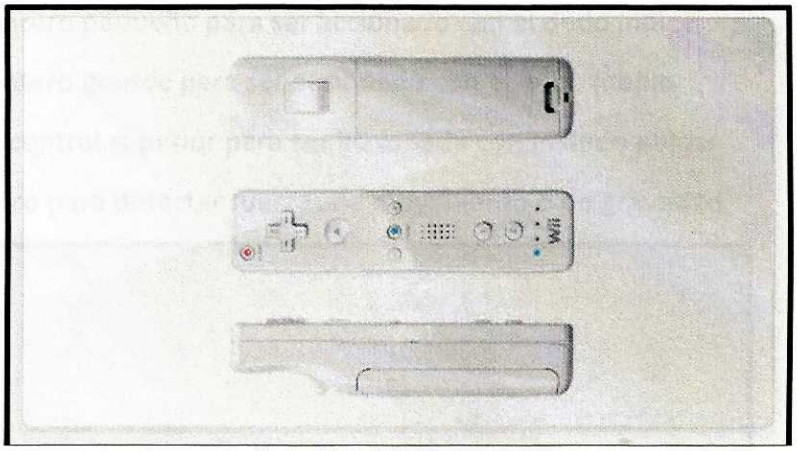

Ilustración 4: Wiimote
[Fuente: http://www.nintendo.com/wii/what/controllers#remote]

Este control es utilizado para interactuar con la consola de juegos de una manera que ha revolucionado la calidad de la usabilidad y el entretenimiento del usuario, ya que permite una interacción hombre-computador mucho más intuitiva que otros dispositivos. Los adelantos tecnológicos más importantes que pone en la mano del usuario y a la disponibilidad del desarrollador de software, son:

- El sistema de comunicación inalámbrico sobre tecnología Bluetooth que permite utilizar el control sin un cable que lo conecte a la consola de juegos para transmitir los datos recolectados. Así, el usuario puede mover libremente el control de manera intuitiva.
- Una cámara que detecta emisiones de luz infrarroja que permite, entre otras cosas, tener una referencia espacial a un punto específico ofreciendo la capacidad de calcular la posición relativa del control respecto a ese punto específico conocido.
- Un acelerómetro que permite detectar fuerzas que se le apliquen al control. Se puede detectar movimiento a través de este sensor, además de la inherente fuerza de la gravedad de la Tierra.

II.1.4. Nunchuk

El nunchuk es un control adicional para el Nintendo Wii que se conecta como una extensión al Wiimote con un cable y tiene forma ovalada para tomarlo con la mano opuesta a la que se sostiene el Wiimote (ver ilustración 5). Tiene los siguientes sensores:

- Botón delantero pequeño para ser accionado con el dedo índice
- Botón delantero grande para ser accionado con el dedo medio
- Palanca de control superior para ser accionada con el dedo pulgar
- Acelerómetro para detectar fuerzas de movimiento o de gravedad

Ilustración 5: Nunchuk conectado a un Wiimote sostenidos por un usuario
[Fuente: http://www.tief-im-see.de/tag/wii]

II.1.5. Acelerómetro

Un acelerómetro es un dispositivo electro-mecánico que mide fuerzas de aceleración (ver ilustración 6). Las fuerzas que detecta pueden ser de tipo estáticas, como la gravedad de la Tierra, o pueden ser dinámicas al causar movimiento o vibraciones al sensor.

Al medir el tamaño de la fuerza estática producida por la gravedad se puede obtener el ángulo de inclinación del acelerómetro con respecto a la Tierra. Al determinar el tamaño de la fuerza dinámica se puede determinar en qué dirección se está moviendo el acelerómetro.

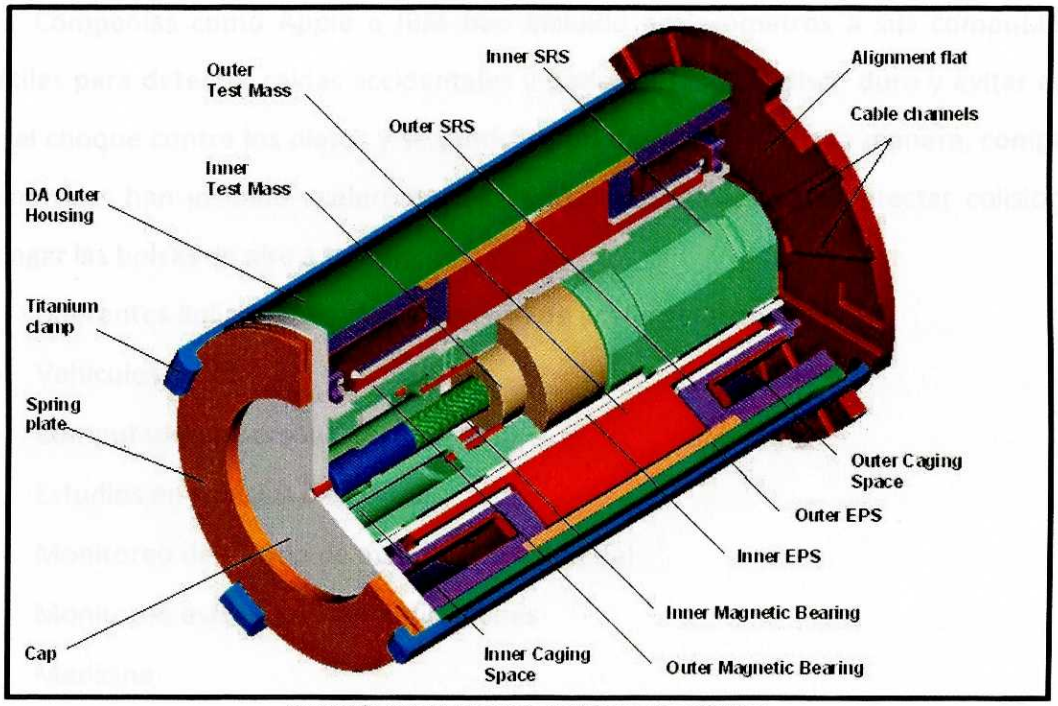

Ilustración 6: Partes de un acelerómetro piezoeléctrico
[Fuente: http://zone.ni.com/devzone/cda/tut/p/id/6031]

En la actualidad, la mayoría de los acelerómetros están construidos utilizando la tecnología de los Sistemas Microelectromecánicos (Microelectromechanical Systems, MEMS; por sus siglas en inglés) por lo cual estos dispositivos pueden alcanzar a tener tamaños microscópicos de entre un micrómetro y un milímetro.

Algunos de los tipos de acelerómetros dependiendo de la tecnología con la que están construidos son:

- De sensor piezoeléctrico
- Térmicos
- Servo electromecánico
- De galga extensiométrica
- De resonancia
- De inducción magnética
- Ópticos
- De láser

El tipo de acelerómetro a utilizar depende de la aplicación.

Compañías como Apple e IBM han incluido acelerómetros a sus computadores portátiles para detectar caídas accidentales y poder detener el disco duro y evitar que el cabezal choque contra los platos y se produzca un daño. De la misma manera, compañías automotrices han incluido acelerómetros en los automóviles para detectar colisiones y desplegar las bolsas de aire a tiempo.

Diferentes aplicaciones que hacen uso de acelerómetros son:

- Vehículos
- Computadores personales
- Estudios en ciencias biológicas
- Monitoreo de estado de maquinaria industrial
- Monitoreo estructural de edificaciones
- Medicina
- Sistemas de navegación
- Sistemas de guiado inercial para cohetes

Productos electrónicos de consumo; teléfonos celulares, reproductores de música portátiles, consolas de juegos de video, entre otros (ver ilustración 7).

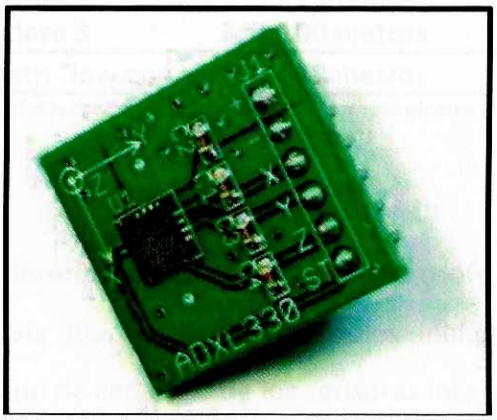

Ilustración 7: Módulo del acelerómetro usado en el Wiimote (AnalogDevicesADXL330)
[Fuente: http://www.seeedstudio.com/depot/wiimote-3axis-accelerometer-module-adxl330-p-107.html]

II.1.6. Bluetooth

La tecnología Bluetooth permite establecer una comunicación inalámbrica de corto alcance entre dos dispositivos. Es un protocolo abierto que ofrece altos niveles de seguridad para la transmisión de datos eliminando la necesidad de conexiones cableadas entre dispositivos que desean intercambiar información.

La "Especificación Bluetooth" establece un conjunto de estándares y una estructura uniforme para que una gran cantidad de dispositivos diferentes se comuniquen entre sí. Todos los dispositivos con tecnología Bluetooth pueden comunicarse formando redes ad-hoc de corto alcance llamadas piconet. En cada piconet pueden comunicarse simultáneamente hasta siete dispositivos que operan sobre el espacio radioeléctrico comprendido entre 2.4 y 2.485 Ghz. La banda ISM 2.4 Ghz está disponible en la mayoría de los países y no necesita licenciamiento para su uso.

Los dispositivos están divididos dependiendo del alcance bajo la siguiente clasificación (ver tabla 1):

CLASE	RANGO DE ALCANCE
Radio clase 1	De 1 a 3 metros
Radio clase 2	10 metros
Radio clase 3	100 metros
Bluetooth "low energy"	200 metros

Tabla 1: Tipos de dispositivos Bluetooth por alcance
[Fuente: Elaboración Propia]

II.1.7. WiimoteLib

WiimoteLib es una librería para la plataforma Microsoft .NET con la cual se puede establecer una conexión vía Bluetooth con controles inalámbricos de Nintendo Wii (Wiimote) y obtener el estado de cada uno de los sensores integrados al control y también de las extensiones conectadas al control como el Nunchuk.

La librería WiimoteLib fue creada por Brian Peek, Microsoft C# MVP; en el año 2008 y en el curso del tiempo ha liberado numerosas actualizaciones para incorporar funcionalidad a la librería. [.NET Managed Library for the Nintendo Wii Remote, 2008] La última versión es la 1.8b1 que añade soporte (en pruebas) para la última extensión para el Wiimote producida por Nintendo, el Wii MotionPlus.

II.1.8. Computación Grafica

La computación gráfica se refiere a la generación de imágenes con el uso de un computador y a la representación y manipulación de los datos de la imagen por parte del computador. La investigación y el desarrollo de la computación gráfica ha tenido un gran impacto en distintos tipos de medios y ha ocasionado una revolución en la producción de animaciones y en la industria de los juegos de video, ya que permite que la interacción humano-computador sea más fácil de lograr mostrando diferentes tipos de datos de una manera más comprensible por un usuario.

Las imágenes generadas por un computador pueden ser categorizadas de la siguiente manera:

- Bidimensionales (2D)
- Tridimensionales (3D) (ver ilustración 8)
- Gráficos animados

Se han desarrollado muchas herramientas para la visualización de datos representados en imágenes. A medida que la tecnología ha avanzado, los gráficos tridimensionales se han vuelto más y más populares, a pesar que los bidimensionales se siguen usando.

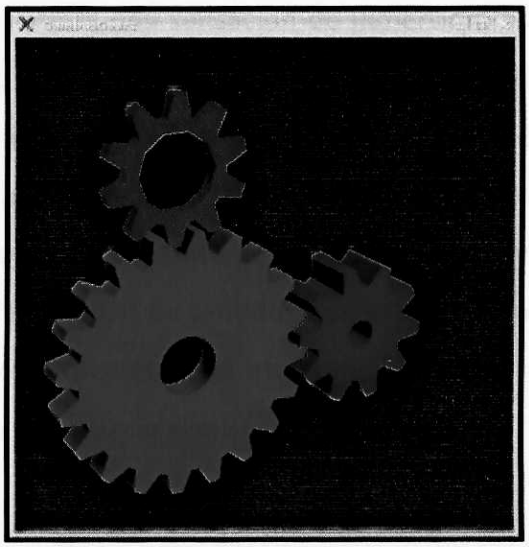

Ilustración 8: Ejemplo de gráficos 3D generados por computador
[Fuente: http://gambasdoc.org/help/comp/gb.opengl]

La computación gráfica se ha derivado en un campo de estudio de las ciencias de la computación y se han desarrollado campos especializados como la visualización científica, que hace énfasis en la visualización tridimensional de fenómenos meteorológicos, médicos, biológicos, etcétera donde puede apreciarse representaciones realísticas de volúmenes, superficies, fuentes de iluminación y demás, a veces con un componente dinámico como el tiempo caso en el que la imagen es animada.

II.1.9. Realidad Virtual

La realidad virtual comprende el uso de tecnología computacional moderna para simular un entorno o mundo tridimensional que el usuario puede manipular y explorar con la sensación de que está en ese mundo realmente. Grandes cantidades de dispositivos y programas han sido desarrollados por científicos, teóricos e ingenieros para alcanzar este objetivo (ver ilustración 9).

Por lo general, una experiencia de realidad virtual completa debe incluir:

- Imágenes tridimensionales que parecen de tamaño real desde el punto de vista del usuario.
- Ajustes sobre las imágenes presentadas al usuario para reflejar cambios en la perspectiva con respecto a movimientos de los ojos y de la cabeza del usuario.

El usuario experimentará una sensación de inmersión, es decir, una sensación de estar dentro del mundo virtual y ser parte de él; y la calidad de dicha inmersión dependerá de la profundidad de la información y la amplitud de la información que el sistema da al usuario. La profundidad de la información se refiere a la cantidad y cantidad de datos que el usuario recibe cuando interactúa con el entorno virtual; y la amplitud de la información se refiere a la cantidad de sentidos que son excitados simultáneamente en el usuario bajo cierta situación, por ejemplo mostrando imágenes, reproduciendo audio y dando respuestas al tacto del usuario al mismo tiempo.

Ilustración 9: Usuario con casco y guantes de realidad virtual
[Fuente: http://briansrapier.wordpress.com/2008/07/15/make-virtual-a-reality/]

El usuario debe ser capaz de cambiar de perspectiva sin inconvenientes, y la latencia desde que el usuario realiza una acción y el sistema hace la reacción correspondiente deberá ser mínima (menor a 50 milisegundos).

Los dispositivos con los que el usuario interactúa con el sistema (de entrada) juegan un papel importante en la realidad virtual. Algunos de los dispositivos de entrada más populares son:

- Palancas de mando
- Forceballs / tracking balls
- Controles inalámbricos
- Guantes con sensores
- Reconocimiento de voz
- Trajes de detección de movimiento
- Cintas de correr

La industria de los juegos de video ha contribuido con el desarrollo de avances gráficos, de sonido y de control que pueden ser incorporados a diseños de sistemas de realidad virtual. En particular, el control inalámbrico de la consola de juegos Nintendo Wii ha marcado un hito en cuanto a la aceptación del público al uso de dispositivos intuitivos para la detección de movimiento por lo cual este control podría estar formando parte de

una nueva ola de avances tecnológicos importantes para la realidad virtual. [Strickland, Sin Fecha]

II.1.10. OGRE 3D

Object-Oriented Graphics Rendering Engine (OGRE 3D) es un motor gráfico de última generación distribuido a través de una licencia de software libre tipo LGNU y escrito en el lenguaje C++. Este motor se utiliza en un proyecto de software como una librería de enlace, y libera al desarrollador de utilizar directamente las librerías que provee el sistema operativo para utilizar DirectX u OpenGL, para así poder dedicarse al desarrollo de la aplicación real en vez de ocuparse de programar directamente sobre estas plataformas, tarea inherentemente ineficiente. De esta manera un proyecto desarrollado sobre el motor OGRE 3D puede ser finalizado mucho más rápido que usando las librerías nativas del sistema operativo.

Igualmente asegura un manejo eficiente de DirectX u OpenGL, manejo automático del estado de renderización, soporte para shaders hechos en Cg, HLSL y GLSL, soporte para gran variedad de formatos de texturas, renderizado sobre texturas, animaciones de esqueletos, técnicas de renderizado de sombras, entre otras características importantes (ver ilustración 10).

Ilustración 10: Escena renderizada por el motor gráfico Ogre 3D
[Fuente: http://www.ogre3d.org/gallery/]

Es un proyecto de software libre empezado en 2001 por su fundador, Steve Streeting y mantenido hoy en día por un pequeño grupo de ingenieros y utilizado por una comunidad muy activa. Ha sido usado como motor gráfico de varios juegos de video comerciales, como es el caso del título Pacific Storm.

II.1.11. Mogre

Mogre es una interfaz de acceso a las funciones del motor gráfico OGRE 3D para poder utilizarlo en aplicaciones escritas en la plataforma .NET 2.0 de Microsoft incluyendo el lenguaje C#.

II.1.12. Autodesk Maya

Autodesk Maya es una suite de software para la producción de gráficos tridimensionales, efectos especiales y animación. Fue galardonada con un premio Oscar por realizar un profundo impacto sobre la industria cinematográfica gracias a su gran capacidad de ampliación y personalización. Su última versión es la 2010.

II.1.13. Shaders

En el ámbito de la computación gráfica, un shader es un conjunto de instrucciones interpretables por un procesador gráfico con las cuales es posible realizar modificaciones directamente a su canal de renderizado con el objeto de obtener efectos gráficos customizables sobre la escena producida con un alto nivel de flexibilidad y adaptación (ver ilustración 11).

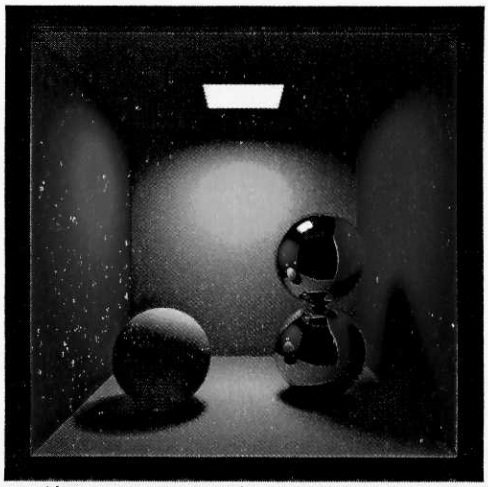

Ilustración 11: Escena 3D renderizada con shaders aplicados
[Fuente: http://graphics.cs.yale.edu/hongzhi/zrt.html]

II.1.14. Sombreado de Phong

Es una técnica de sombreado para imágenes 3D generadas por computador en la que el brillo de las superficies se divide en tres componentes: luz de ambiente, reflexión difusa y destellos especulares (ver ilustración 12). Su efecto en una renderización en tiempo real se consigue a través del uso de shaders, logrando que las superficies que lo requieran tengan una apariencia de alto reflejo de la luz.

Ilustración 12: Ejemplo de sombreado de Phong
[Fuente: http://en.wikipedia.org/wiki/Phong_shading]

CAPITULO III

III. Marco metodológico

Una metodología define el conjunto de pasos, actividades y fases sobre las que debe desarrollarse un proyecto. En el ámbito de Ingeniería del Software, se proponen diferentes tipos de metodologías cuyo fin común es establecer una ruta para lograr un producto de software de calidad. Cada proyecto de software debe establecer un modelo a seguir, que no necesariamente es el mismo para todos los casos ya que cada proyecto tiene necesidades diferentes, será producido por grupos de personas de diferentes tamaños y tendrá recursos diferentes. La elección de la metodología a seguir para un proyecto específico es de vital importancia, pues es uno de los factores que puede determinar el éxito o el fracaso del mismo.

Para este TEG la metodología escogida fue la de desarrollo evolutivo o también llamada metodología basada en prototipos evolutivos. Esta metodología divide el desarrollo del proyecto en diferentes iteraciones que a partir de un prototipo inicial producen cada una un prototipo funcional. En la iteración inicial se produce el prototipo inicial al cual se le irán añadiendo nuevas funcionalidades en las iteraciones siguientes y se irán mejorando las funcionalidades ya implementadas a conformidad del cliente, logrando al final un sistema completo y funcional.

Para el uso de la metodología basada en prototipos evolutivos debe partirse del hecho de que no todos los requerimientos del sistema son conocidos o bien comprendidos al inicio del desarrollo, y sólo se irán implementando aquellos que si se conozcan y se comprendan a plenitud por parte del (los) desarrollador(es). Además dado que es una metodología basada en iteraciones, el proyecto puede ser dividido por partes, cada una de complejidad manejable por el equipo de desarrollo. El proyecto evoluciona cada vez que una versión nueva del prototipo funcional se muestra al usuario y este retroalimenta al proyecto con las observaciones pertinentes, que luego serán añadidas a las propias del equipo de desarrollo para ser tomadas en cuenta en la próxima iteración.

La documentación consultada propone que la metodología basada en prototipos evolutivos consista de las siguientes actividades en cada iteración (ver ilustración 13):

1. Recolección y refinamiento de requisitos

   La primera fase contempla la recolección de los requisitos a cumplir en la iteración actual así como su respectivo análisis y el de sus factores de riesgo que pueden producir a su vez otros requisitos. Los requisitos especificados deben ser medibles y exactos, y son producto de la interacción del equipo de desarrollo con el cliente.

2. Diseño rápido

   A partir de los requisitos el equipo de desarrollo realiza un concepto de las respectivas soluciones a implementar para cada uno, donde se describe los elementos a desarrollar durante la construcción y se discute la estrategia de implementación.

3. Construcción del Prototipo

   En la fase de construcción el equipo de desarrollo utiliza todas las herramientas, conocimiento y experticia necesaria para ejecutar el desarrollo e implementar las soluciones a los requerimientos del cliente. El equipo de desarrollo debe procurar que las implementaciones realizadas sean de la mejor calidad posible porque deben lograr un prototipo funcional.

4. Evaluación del prototipo por el cliente

   Luego de finalizadas las fases de diseño y construcción el prototipo conseguido es mostrado al cliente, quién ofrecerá una retroalimentación al equipo de desarrollo a través de observaciones que surjan durante el período de prueba que tendrá con el prototipo conseguido en la iteración.

5. Refinamiento del prototipo

   Una vez que el prototipo es evaluado por el cliente, el equipo de desarrollo añade sus propias observaciones al respecto y decide del conjunto completo de observaciones cuáles serán tomadas en cuenta en la próxima

iteración, con el objetivo de producir las mejoras necesarias. Esta fase retroalimenta el proyecto y se lleva a cabo una nueva iteración.

6. Producto de ingeniería

   Al finalizar todas las iteraciones necesarias en las que el prototipo funcional se vino refinando y completando el mismo es considerado como completo y procede a denominarse Producto de Ingeniería, y es la solución conseguida por el equipo de desarrollo a los requerimientos del cliente.

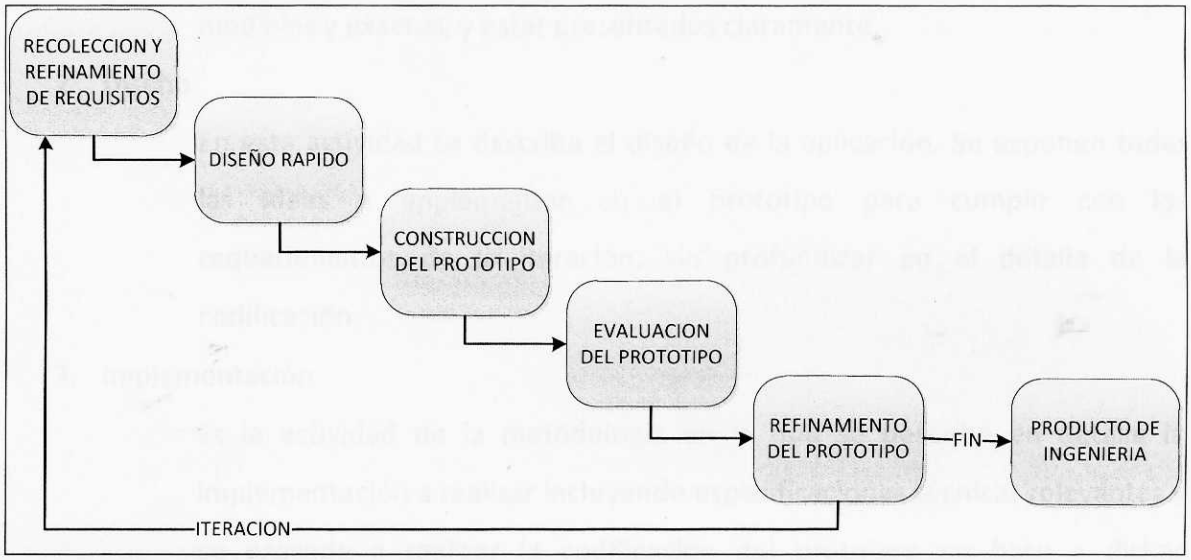

Ilustración 13: Diagrama de la metodología basada en prototipos evolutivos
[Fuente: Elaboración Propia]

Ésta metodología ofrece al desarrollador gran flexibilidad y una amplia capacidad de adaptación y dado que se trata de un trabajo de investigación, el proyecto propuesto en este TEG prescinde de un cliente y en su lugar el equipo de desarrollo puede asumir ese rol y hacer los aportes correspondientes al proyecto. Por tales razones y debido a las características del proyecto, las actividades de "Evaluación del prototipo por el cliente" y "Refinamiento del prototipo" pueden ser resumidas en una sola actividad de Pruebas, en la cual se comprobará el funcionamiento correcto de la implementación realizada en cada iteración y en caso de ser necesario se propondrán las mejoras a realizar que serán recibidas como requisitos por la siguiente iteración.

De esta manera el conjunto de actividades a seguir para el desarrollo del proyecto propuesto en este TEG es el siguiente (ver ilustración 14):

1. Recolección de Requerimientos

   La primera fase contempla la especificación de los requerimientos a cumplir en la iteración actual así como su respectivo análisis y el de sus factores de riesgo que pueden producir a su vez otros requerimientos. Deben ser medibles y exactos, y estar presentados claramente.

2. Diseño

   En esta actividad se describe el diseño de la aplicación. Se exponen todas las ideas a implementar en el prototipo para cumplir con los requerimientos de la iteración, sin profundizar en el detalle de la codificación.

3. Implementación

   Es la actividad de la metodología en la que se describe en detalle la implementación a realizar incluyendo especificaciones técnicas relevantes y se procede a realizar la codificación del prototipo en base a dichas especificaciones. Toda implementación debe ser completa y correcta, es decir, debe satisfacer a plenitud el requerimiento que soluciona y hacerlo de manera efectiva.

4. Pruebas

   La etapa final de cada iteración es donde se desglosan el conjunto de pruebas que se realizan al prototipo de manera tal de confirmar el cumplimiento satisfactorio de las soluciones planteadas para la iteración en el diseño y la implementación, y de exponer los pormenores que deben mejorarse para la siguiente iteración en caso de que existan. También, de ser necesario, se pueden proponer variaciones sobre el diseño o la implementación para obtener una solución más eficiente en la siguiente iteración.

5. Prototipo funcional

Cada iteración tendrá como producto una versión funcional del prototipo y a partir del mismo se debe iniciar de nuevo el proceso definido por la metodología hasta conseguir un prototipo funcional que satisfaga completamente el objetivo general y los objetivos específicos del proyecto.

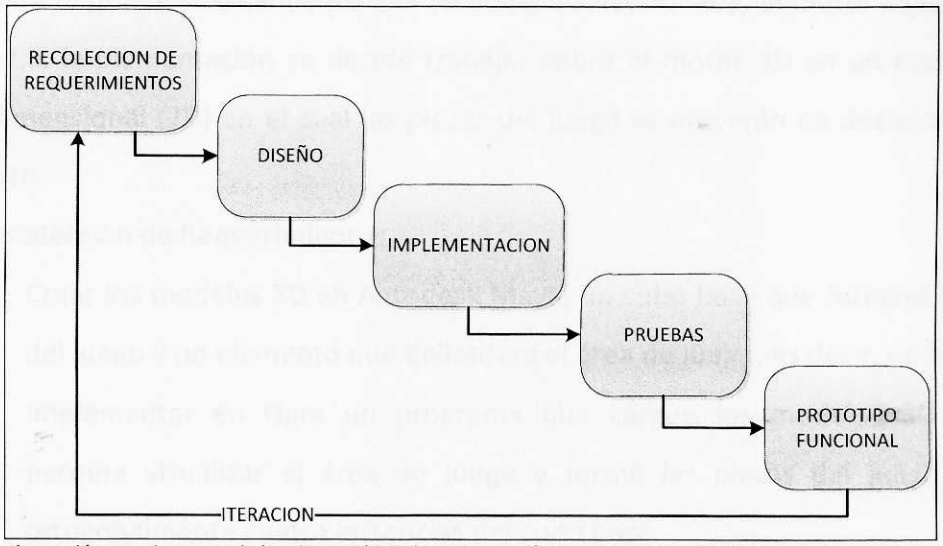

Ilustración 14: Diagrama de la adaptación de la metodología basada en prototipos evolutivos propuesta [Fuente: Elaboración propia]

Dado el grado de dificultad del proyecto, se decidió dividirlo en iteraciones de la siguiente manera:

- Iteración 1: Prueba de la tecnología seleccionada para el desarrollo
- Iteración 2: Juego de Tetris con reglas en dos dimensiones controlado con el teclado
- Iteración 3: Juego de Tetris con reglas en tres dimensiones controlado con el teclado
- Iteración 4: Juego de Tetris con reglas en tres dimensiones controlado con el control inalámbrico de Nintendo Wii.

CAPÍTULO IV

IV.1. Desarrollo

IV.1.1 Iteración 1

La primera iteración del proyecto, consiste en realizar una prueba de concepto sencilla para el primer acercamiento a la tecnología seleccionada, el motor Ogre 3D. Para simplificar la implementación se decide trabajar sobre el motor 3D en un escenario de juego bidimensional (2D) en el cual las piezas del juego se moverán en dos coordenadas únicamente.

1. Recolección de Requerimientos
   - Crear los modelos 3D en Autodesk Maya, un cubo base que formará las piezas del juego y un elemento que delimitará el área de juego, es decir, un borde.
   - Implementar en Ogre un programa que cargue los modelos 3D creados, permita visualizar el área de juego y forme las piezas del juego uniendo ortogonalmente cuatro instancias del cubo base.
   - Implementar el control de las piezas de juego con teclado.
2. Diseño
   a. Modelos 3D
      - El cubo base tiene unas medidas específicas y un bisel para que al ser cargado en el escenario 3D de un efecto de sombra sobre los demás cubos adyacentes que forman la pieza. Se le aplica un material que produce un efecto agradable a la vista (ver ilustración 15).

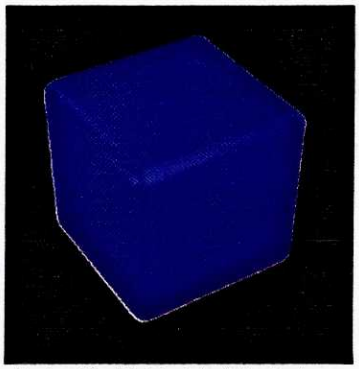

Ilustración 15: Modelo del cubo base
[Fuente: Elaboración Propia]

- El borde del área de juego se diseña de manera que delimite el área de juego y sea de las mismas medidas que las implementaciones de Tetris tradicionales.

  Está compuesto por tres cubos alargados que le dan forma de "U". Se propone pintarlo de un color que no interfiera en la vista pero tenga cierto contraste sobre el fondo negro (ver ilustración 16).

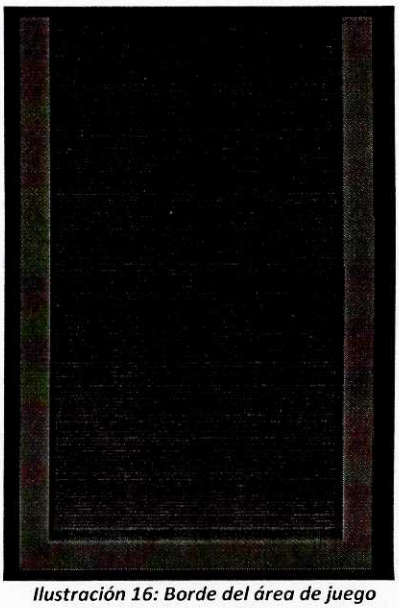

Ilustración 16: Borde del área de juego
[Fuente: Elaboración Propia]

b. Ogre.

- Para diseñar la implementación en Ogre se utiliza el programa base de uno de los tutoriales de Mogre en el lenguaje de programación C#. Está compuesto por diferentes métodos que al ser invocados con una secuencia particular, inicializan el motor 3D, cargan los elementos a mostrar (modelos tridimensionales y demás recursos), realizan la configuración del ambiente inicial, entra en un ciclo de renderización hasta que el mismo es interrumpido, con lo cual se desencadena un conjunto de llamadas para liberar la memoria reservada y finalizar la ejecución del motor.
- Las piezas se forman a partir de instancias del cubo base colocadas en cierta disposición y pintadas del color correspondiente a la pieza en cuestión. Cada pieza tiene un nodo (o cubo) base y una cierta cantidad de nodos auxiliares.

Con este diseño la pieza podrá rotar sobre el nodo base y los demás nodos se reubicarán a la posición correcta. Por esta razón, las posiciones de los nodos auxiliares son calculadas en base a la posición del nodo base.

- En los algoritmos de movimiento de las piezas se trabajará en base a unidades imaginarias sobre la matriz imaginaria que forma el área de juego. Por esta razón se debe hacer un convertidor de coordenadas imaginarias en la matriz a coordenadas reales del motor 3D en la escena visual. Dado que el cubo base tendrá unas medidas específicas las coordenadas en el motor 3D pueden ser fácilmente obtenidas con una fórmula.

c. Movimiento por teclado.

- Se propone utilizar la librería MouseKeyboardLibrary para atender eventos de teclado definiendo el uso de las teclas (flecha izquierda, flecha derecha y flecha abajo) con la llamada al movimiento correspondiente de la pieza que esté jugando en el momento.

3. Implementación

a. Modelos 3D. Elaborados en el software Autodesk Maya 2008.

- El cubo base mide dos por dos (2x2) centímetros. Tiene aplicado un bisel de tres segmentos de 0,15 centímetros de separación con un ángulo de suavizado de 30 grados. Tiene aplicado también un material del tipo Phong que el motor 3D renderiza usando un shader que produce un efecto reflectivo que da belleza al escenario de juego con el fin de ofrecer una interfaz agradable al usuario.
- El borde del juego mide doce (12) casillas de alto por diez (10) de ancho. Estas medidas fueron tomadas a partir de la información obtenida durante la exploración de las diferentes versiones de Tetris bidimensional siendo doce por diez casillas (12x10) la medida del área de juego más común (ver ilustración 17). Está pintado de color gris.

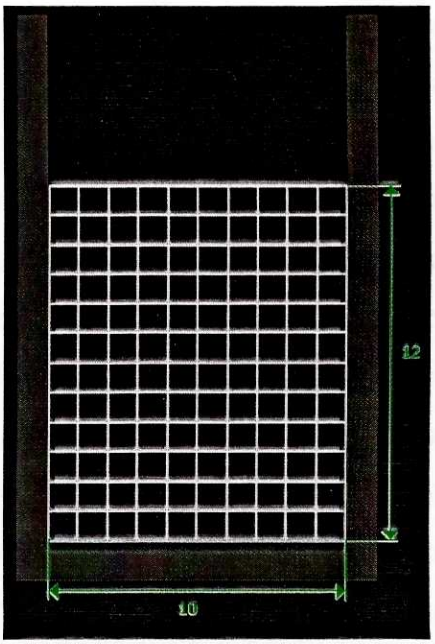

Ilustración 17: Borde con las dimensiones del área de juego
[Fuente: Elaboración Propia]

b. Ogre. El desarrollo se realizó en C# utilizando Mogre.

- Se utilizó como base el programa de ejemplo del tutorial "Mogre Basic Tutorial" disponible en: [Mogre Basic Tutorial 5, Sin Fecha]

  Los métodos invocados son los siguientes (ver tabla 2):

METODO	DESCRIPCION
CreateRoot	Crea el objeto Root necesario para la ejecución de Ogre
DefineResources	Lee el archivo de configuración de recursos y carga la ubicación de los recursos en Ogre
SetupRenderSystem	Configura el sistema de renderizado a utilizar, DirectX o OpenGL
CreateRenderWindow	Crea una ventana de Ogre donde el sistema de renderizado seleccionado puede mostrar sus resultados
InitializeResourceGroups	Carga e inicializa los recursos definidos en los archivos de configuración
CreateScene	Instancia el manejador de la escena, la vista y la cámara a utilizar, y carga los modelos 3D que se llaman una vez y los ubica en la posición deseada
StartRenderLoop	Llama a los métodos encargados de dibujar la escena utilizando el motor gráfico para actualizarla continuamente y refresca la ventana de visualización

Tabla 2: Métodos de Mogre utilizados en la iteración 1
[Fuente: Elaboración propia]

- La clase PiezaAbstracta tiene un conjunto de variables protegidas (ver apéndice A) visibles por las clases que hereden de ella que indican cuántos nodos auxiliares tiene la pieza y cuántas referencias a instancias de SceneNode de Ogre (que son las referencias a cada uno de los cubos que la forman) tiene la pieza. En el constructor de la clase de la pieza particular, ejemplo PiezaT, se especifica que tiene una cantidad particular de nodos auxiliares, se instancian los nodos, se guardan referencias a ellos en un arreglo de instancias de SceneNode y por último se procede a inicializar la propiedad Position de cada nodo auxiliar de la pieza de manera tal de disponerlos correctamente relativos a la posición del nodo base o principal. Como puede observarse en la ilustración 18, las coordenadas lógicas de cada cubo son relativas a las coordenadas del nodo base. (Ver el apéndice D)

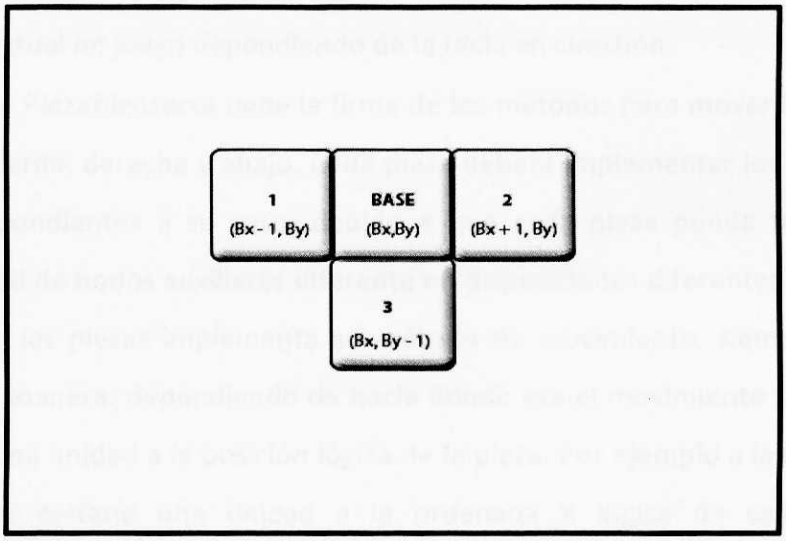

Ilustración 18. Pieza T con coordenadas de los nodos auxiliares relativas al nodo base. [Fuente: Elaboración Propia]

- Se implementó un convertidor de coordenadas de la matriz imaginaria (en memoria) a coordenadas de la escena en el motor 3D. Sabiendo las coordenadas especificadas en la propiedad Position del nodo base, es un simple problema de diferencias de distancias. (Ver el apéndice C)
- En este momento, al ejecutar la aplicación se muestra la pieza formada correctamente y el borde del área de juego en su posición correcta. La

cámara está ubicada en el centro del escenario (coordenadas 0 en el eje X y 0 en el eje Z) subida 50 unidades en el eje Y, mirando el origen del escenario (punto (0,0,0)). De esta manera se logra visualizar un área de juego plana conformada por elementos tridimensionales.

c. Movimiento de las piezas con teclado

- Se utilizó la librería MouseKeyboardLibrary. Esta librería ofrece diferentes manejadores de eventos para capturar entrada por teclado. Se utilizaron los siguientes manejadores de eventos (ver tabla 3):

EVENTO	DESCRIPCIÓN
KeyboardHook.KeyDown	Una tecla ha sido presionada
KeyboardHook.KeyUp	Una tecla estaba presionada y ha sido soltada
KeyboardHook.KeyPress	Una tecla está siendo presionada continuamente

Tabla 3: Descripción de los manejadores de eventos de teclado utilizados
[Fuente: Elaboración Propia]

En cada uno de ellos se hace la llamada al método correspondiente de la pieza actual en juego dependiendo de la tecla en cuestión.

- La clase PiezaAbstracta tiene la firma de los métodos para mover la pieza a la izquierda, derecha y abajo. Cada pieza deberá implementar los métodos correspondientes a su caso, debido a que cada pieza puede tener una cantidad de nodos auxiliares diferente en disposiciones diferentes. Así cada una de las piezas implementa su método de movimiento, siempre de la misma manera: dependiendo de hacia donde sea el movimiento se resta o suma una unidad a la posición lógica de la pieza. Por ejemplo a la izquierda significa restarle una unidad a la ordenada X lógica de cada nodo, simplemente se modifican las posiciones gráficas de los nodos de esta manera y se obtiene la nueva posición.

4. Pruebas

- Los modelos tridimensionales creados con Maya son adecuados visualmente para el cabal cumplimiento del juego de Tetris. Se cargan y se muestran correctamente en el motor de gráficos 3D Ogre, y el efecto de sombra del bisel de los cubos ofrece el resultado esperado. El color y la técnica de shader

utilizada en los cubos también ofrecen un efecto agradable a la vista. En cuanto al borde, no es muy elaborado visualmente pero cumple perfectamente la función de demarcar el área de juego y en un color que no distrae al jugador.

- La aplicación se carga sin errores de ejecución, y realiza todas las operaciones de inicialización y carga de los elementos necesarios. La escena se forma correctamente con la pieza construida correctamente y el ciclo de renderización se ejecuta de manera satisfactoria. No existe aún un ciclo de juego que cambie la pieza al llegar al fondo del área de juego. El convertidor de coordenadas imaginarias a reales funciona bien porque la pieza formada se muestra correctamente y el algoritmo lo utiliza para construir la pieza.
- El movimiento por teclado funciona como era de esperar, al pulsar alguna de las teclas soportadas en esta iteración (flecha izquierda, flecha derecha, flecha abajo) se ejecuta la operación correspondiente en la pieza. De momento la pieza puede moverse infinitamente en cualquier dirección.

5. Prototipo funcional

- El prototipo de esta iteración permite concluir que la selección de la tecnología para crear los modelos 3D y para crear el software del juego es correcta y satisface las necesidades del proyecto. Se logró crear y utilizar el cubo base para formar las piezas en tiempo de ejecución; así como también se pudo utilizar un método sencillo para introducir comandos al juego como es el teclado del computador. Dado que la captura de comandos de entrada y la ejecución de las operaciones correspondientes están separadas será muy fácil cambiar el dispositivo de entrada en el momento que sea necesario, sin necesidad de modificar la lógica ya implementada para el movimiento de las piezas.

IV.1.2 Iteración 2

Una vez realizada la prueba de concepto tecnológico para el desarrollo del proyecto en la primera iteración, para esta segunda se añadirán todas las funcionalidades necesarias para ejecutar un juego completo de Tetris en dos dimensiones.

1. Recolección de Requerimientos
   a. Restringir el movimiento de las piezas al área de juego demarcada por el borde.
   b. Añadir la funcionalidad de rotar la pieza en juego en sentido horario.
   c. Implementar el ciclo del juego en el programa y las reglas del juego de Tetris bidimensional.
   d. Integrar las reglas del juego con el motor de renderización 3D y crear un depurador para verificar la correcta representación del juego en la escena 3D.
   e. Añadir la funcionalidad de pulsar una tecla para hacer bajar la pieza en juego rápidamente mientras dicha tecla esté pulsada.
2. Diseño
   a. Restringir el movimiento de las piezas
      - La pieza en juego podrá moverse solamente dentro del área demarcada por el borde del juego.
   b. Habilitar rotación de las piezas
      - La pieza en juego podrá ser rotada pulsando la tecla flecha arriba del teclado. Esto implica modificar las coordenadas de posición de los nodos auxiliares, haciendo la pieza rotar sobre el eje que representa el nodo base.
   c. Incorporar el ciclo de juego
      - El ciclo de juego consiste en hacer bajar un nivel la pieza en juego cada cierta cantidad de tiempo; y cuando la pieza llegue al fondo del área de juego o colisione con otra pieza ya colocada entonces escoger aleatoriamente otra pieza que será otorgada al jugador para que siga intentando acumular puntos. Este ciclo se detiene cuando la nueva pieza no pueda bajar del nivel inicial en el tope del área de juego, lo que significa que el jugador ha perdido la partida y la misma ha terminado.

- Las reglas del juego Tetris en un ambiente bidimensional están documentadas en las muchas otras implementaciones de dicho juego. Consiste en colocar la pieza en juego que va cayendo verticalmente desde el tope del área de juego de manera tal de conseguir llenar líneas horizontales completas en el área de juego que serán eliminadas al ser completadas y las piezas que están en los niveles superiores del juego caerán un nivel. Cuando la pieza en juego llega al fondo del área de juego o colisiona con otra pieza ya colocada, el jugador obtendrá una pieza nueva que podrá controlar, cuya forma será seleccionada aleatoriamente por el sistema. Al eliminar una línea horizontal completa el jugador ganará cien (100) puntos. Si por alguna circunstancia una pieza no puede bajar del nivel inicial entonces el jugador perderá la partida. A medida que el jugador va ganando puntos la velocidad de caída de las piezas irá aumentando para ofrecer una experiencia que exija cada vez más destreza del jugador.

d. Integrar las reglas con el motor gráfico

- Para la verificación del cumplimiento de las reglas del juego, se diseñó un conjunto de algoritmos que trabajan con una matriz en memoria que representa el estado del área de juego en cada momento. Para verificar el correcto funcionamiento de los algoritmos de verificación, se diseñó una ventana para el sistema operativo Windows que muestra el contenido de la matriz en memoria (ver ilustración 19). De esta manera se pueden separar las pruebas del funcionamiento de las reglas de las pruebas correspondientes a la correcta representación del juego en la escena del motor gráfico.

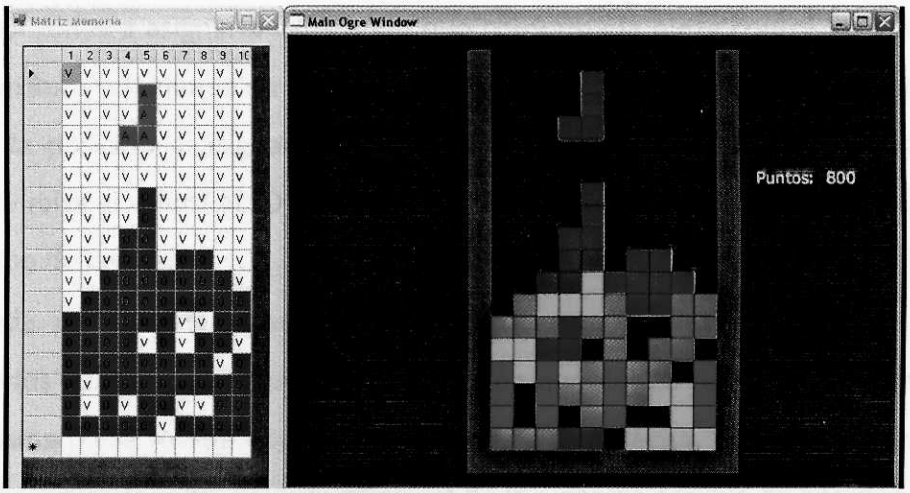

Ilustración 19: Ventana de renderización y ventana de verificación de la matriz en memoria [Fuente: Elaboración Propia]

e. Habilitar tecla para bajar rápidamente la pieza en juego

- Todas las implementaciones del juego de Tetris incluyen algún método para hacer bajar de nivel la pieza en juego actual más rápidamente con el objetivo de ofrecerle al jugador un mecanismo para acelerar el juego. Esta aceleración implica reducir el tiempo de cada ciclo de juego mientras el jugador lo solicite. Para producir éste efecto se asignó a la Flecha Abajo del teclado del computador como la tecla que lo invoca, e inmediatamente después que el jugador presione la tecla la velocidad de la caída de la pieza será aumentada cierta cantidad utilizándose ésta velocidad mientras el jugador mantenga la tecla pulsada. Cuando el jugador suelte la tecla el ciclo de juego reanudará el tiempo de cada iteración al que se estaba utilizando antes que el jugador pulsara la tecla.

3. Implementación

a. Restricción del movimiento de las piezas

- Para evitar que las piezas salieran fuera del borde del área de juego se implementó un algoritmo de chequeo en métodos correspondientes que se ejecutan antes de llamar a los métodos de movimiento a la izquierda y derecha de cada pieza. Si el método de chequeo devuelve el valor VERDADERO la llamada al método de movimiento correspondiente es realizada, caso contrario se omite la llamada. Dicho algoritmo compara las

posiciones a asignar a cada nodo que conforma la pieza contra los índices máximos y mínimos de la matriz de estado del área de juego en memoria. Si la posición a asignar no está dentro del rango permitido el método devuelve el valor FALSO y en caso contrario devuelve VERDADERO.

b. Rotación de las piezas

- Cada pieza implementa un algoritmo de rotación que re posiciona los nodos auxiliares de la pieza a la ubicación que deben tener para formar la misma pieza rotada noventa (90) grados a la derecha, en sentido horario (ver ilustración 20). El nodo principal de la pieza no cambia de posición por lo que se dice que la pieza "pivota" sobre su nodo principal. También se implementó un método de chequeo en cada pieza para verificar que la pieza puede ser rotada en su ubicación actual de manera que ningún nodo se salga del área de juego; de la misma manera que se implementó con los métodos de movimiento a los lados. Por tal razón el chequeo y el reposicionamiento de los nodos auxiliares es llevado a cabo primero en la matriz de estado del área de juego en memoria y luego es representado visualmente por el motor gráfico Ogre3D. La tecla escogida para la invocación de este movimiento fue la flecha arriba del teclado, ya que muchas de las diferentes implementaciones del Tetris bidimensional tienen esa asignación por defecto.

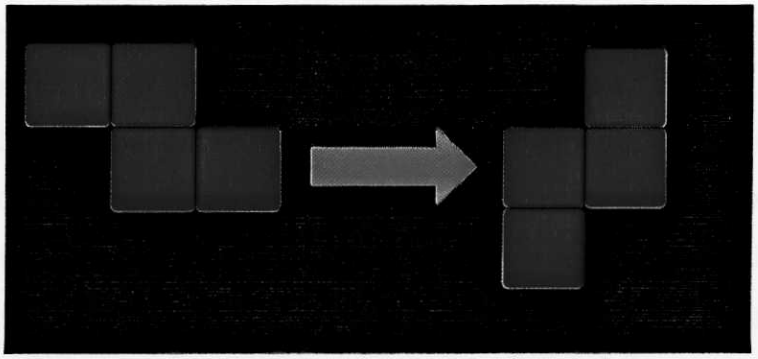

Ilustración 20: Rotación de la pieza en forma de Z
[Fuente: Elaboración Propia]

c. Ciclo y reglas del juego

- Al iniciar un ciclo, alguna de las piezas se forma y un algoritmo chequea cada cierto tiempo si la pieza puede bajar un nivel. Cada pieza implementa un algoritmo diferente porque depende de su forma. Si la pieza puede bajar baja y si no puede bajar se forma una nueva pieza seleccionada aleatoriamente. (Ver el apéndice C)
- Cuando la pieza en juego no puede bajar otro nivel sea porque colisiona con el fondo del área de juego o con otra pieza ya colocada, un algoritmo chequea todos los niveles de abajo hacia arriba para detectar si 35algún nivel está completamente ocupado por cubos. Si está completamente ocupado, ese nivel se marca como vacío en memoria y todas las posiciones ocupadas de los niveles superiores bajan un nivel. Desde el punto de vista gráfico, los cubos que ocupan ese nivel son eliminados y los que están en los niveles superiores bajan un nivel. (Ver el apéndice C)

d. Integración de las reglas del juego con el motor gráfico

- Para la implementación de la integración de las reglas del juego existió la necesidad de desarrollar una herramienta de depuración para visualizar el estado de la matriz en memoria "pMatrizMemoria" de la clase "SingletonMatrizMemoria". Esta herramienta es una ventana para el sistema operativo Windows que muestra en una tabla el contenido de la matriz (ver ilustración 21), y se actualiza en cada ciclo de renderización del motor gráfico mostrando cuando ocurre algún cambio en el contenido de la misma. De esta manera es posible representar fielmente el contenido de la matriz de una manera bastante automática y libre de errores, cada casilla de la tabla corresponde a una coordenada de la matriz. Dependiendo del estado de cada casilla (el valor que tenga asignado) se pinta de un color el fondo de la casilla, blanco para las casillas vacías, azul para las casillas ocupadas y rojo para las casillas correspondientes a la pieza en juego

actual. También se escribe sobre la casilla el valor V para vacío, O para ocupado y A para la pieza actual.

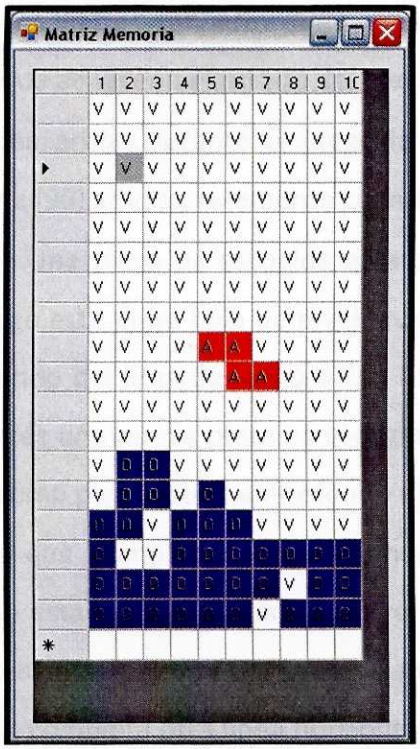

Ilustración 21: Ventana de verificación de la matriz en memoria
[Fuente: Elaboración Propia]

e. Tecla para bajar rápido la pieza

- Si durante el juego la tecla flecha abajo del teclado es pulsada, una variable del programa principal le indica al ciclo de juego que el jugador solicita bajar la pieza rápidamente. Esto implica que el tiempo del ciclo de juego actual sea almacenado en una variable y es reemplazado por el valor anterior menos cien (100) milisegundos. Luego, cuando el jugador suelta la tecla el tiempo del ciclo de juego es restablecido al valor almacenado anteriormente y la variable del programa principal vuelve a su valor por defecto. Si mientras el jugador mantiene pulsada la tecla ocurre un cambio de pieza, el tiempo de juego es restablecido al original.

4. Pruebas

- El algoritmo de chequeo de movimiento impide que la acción pueda realizarse más allá de los límites definidos que coinciden con el tamaño de la matriz de memoria que almacena el estado del juego.
- Al pulsar la tecla flecha arriba durante el juego la pieza actual rota en sentido horario noventa (90) grados, solamente cuando puede hacerlo.
- Cuando el juego inicia una pieza es escogida aleatoriamente y asignada como pieza actual, luego esta pieza baja de nivel en el área de juego hasta que colisiona con el fondo del área de juego o con otra pieza ya colocada. Inmediatamente después una nueva pieza es escogida aleatoriamente por el sistema y asignada como pieza actual empezando el ciclo nuevamente.
- Al colocar una pieza que completa una línea horizontal en algún(os) nivel(es) los cubos que están colocados en ese nivel desaparecen y todos los cubos que están en los niveles superiores bajan un nivel. Si al bajar alguno de los niveles se completa otra línea horizontal entonces se procede a realizar de nuevo la operación hasta que no quedan más líneas horizontales de cubos que eliminar. Por cada línea eliminada el puntaje aumenta y a medida que se van eliminando más líneas la dificultad del juego va aumentando, ofreciendo una experiencia de juego que exige más destreza del jugador.
- La representación gráfica del juego es completamente fiel a las operaciones realizadas en memoria con lo cual las reglas del juego son reflejadas visualmente.
- Si se pulsa la tecla flecha abajo del teclado la pieza en juego bajará de nivel más rápido de lo normal ofreciendo acelerar el juego por un período de tiempo limitado mientras dicha tecla se mantenga pulsada. Al soltarla el juego reanuda su velocidad normal dependiendo del nivel en el que se esté jugando.

5. Prototipo funcional

- El producto de la segunda iteración del proyecto es un juego de Tetris bidimensional completo. Incluye todo lo que cabe esperar de dicho juego, no sólo el cabal cumplimiento de sus reglas básicas sino detalles añadidos como lo es la tecla de hacer bajar rápido la pieza actual en juego. Los algoritmos de chequeo y movimiento están diseñados e implementados de manera tal que añadir una nueva dimensión a la lógica del juego sea una labor de menor dificultad.

IV.1.3 Iteración 3

Habiendo completado todos los requisitos del juego de Tetris 2D se procede a implementar una versión que le añada la tercera dimensión, rediseñando la capa lógica del programa y realizando los cambios independientemente del mecanismo de control del juego. Por esta razón se seguirá utilizando el teclado como único medio de entrada de comandos al programa y se modificarán las reglas del juego para abarcar tres dimensiones.

1. Recolección de Requerimientos
    - Crear un nuevo modelo 3D para demarcar el área de juego, es decir, el borde.
    - Modificar la formación de las piezas para que sean apropiadas al juego en tres dimensiones y asignarles nuevos colores distinguibles
    - Implementar un mecanismo para mover las piezas en tres dimensiones con el teclado.
    - Modificar las reglas del juego de Tetris bidimensional para abarcar tres dimensiones.
    - Integrar las reglas del juego con el motor de renderización 3D y la matriz del estado de juego en memoria.

2. Diseño

a. Creación del nuevo modelo 3D del borde

- Con respecto al borde del juego, debió rediseñarse el modelo para conseguir una figura que demarque el área de juego pero a la vez no sea un obstáculo visual para el jugador. El modelo diseñado fue un piso en forma de cuadrado que demarca el área de juego (ver ilustración 22), con lo cual el jugador puede visualizar imaginariamente el área de juego teniendo como referencia una forma muy sencilla, fácil de interpretar y que no obstaculiza la visibilidad del juego.

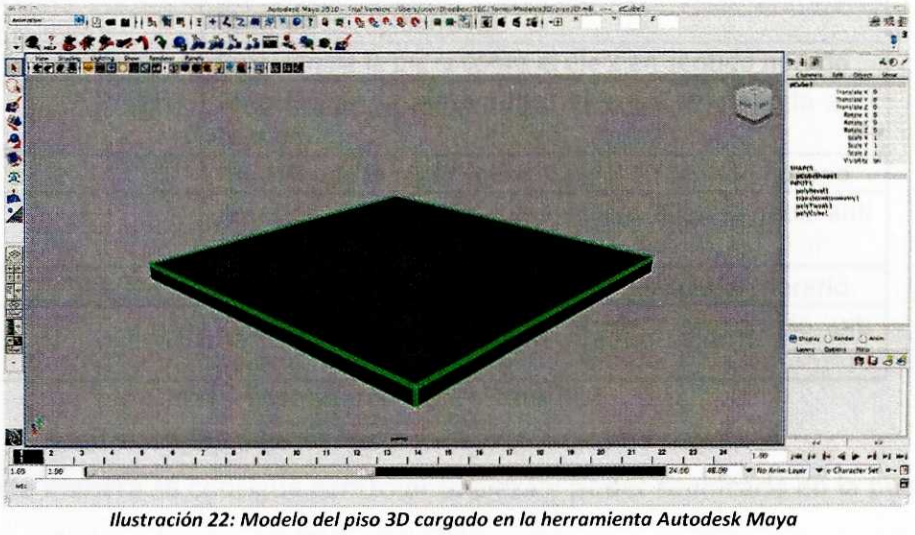

Ilustración 22: Modelo del piso 3D cargado en la herramienta Autodesk Maya
[Fuente: Elaboración Propia]

b. Modificación de la formación y de los colores de las piezas

- Las piezas de esta versión se seleccionaron de manera que no existan piezas repetidas; por ejemplo las piezas en forma de Z de la versión 2D existían en versiones a la izquierda y a la derecha, pero al agregar la tercera dimensión al juego cualquiera de las dos puede ser rotada de manera tal de ser exactamente igual que la otra y por tal razón sólo una es suficiente como pieza para el juego tridimensional. Esto mismo ocurre con la pieza en forma de L.
- Los colores utilizados en las piezas han de ser modificados para que cada pieza sea distinguible claramente.

- La ubicación inicial de los nodos de la pieza en forma de cuadrado fue modificada para añadirle un grupo de nodos en el eje de la profundidad (Z) de manera que no sea un cuadrado sino un cubo formado por 8 nodos.

c. Reasignación de teclas para el movimiento de las piezas.

- Para que el juego pueda ser utilizado con el teclado fue necesario reasignar las teclas a un nuevo mapeo que permitiera probar las nuevas reglas. Se definió de la siguiente manera (ver tabla 4):

TECLA	FUNCIÓN
Flecha arriba	Movimiento sobre el plano, eje Z positivo
Flecha abajo	Movimiento sobre el plano, eje Z negativo
Flecha izquierda	Movimiento sobre el plano, eje X negativo
Flecha derecha	Movimiento sobre el plano, eje Z positivo
Barra espaciadora	Bajar pieza rápidamente
Letra I	Rotación de la pieza sobre el eje Z en sentido horario
Letra J	Rotación de la pieza sobre el eje Z en sentido anti horario
Letra O	Rotación de la pieza sobre el eje X en sentido horario
Letra K	Rotación de la pieza sobre el eje X en sentido anti horario
Letra P	Rotación de la pieza sobre el eje Y en sentido horario
Letra L	Rotación de la pieza sobre el eje Y en sentido anti horario

Tabla 4: Asignación de las teclas en la iteración 3
[Fuente: Elaboración Propia]

- De esta manera se consigue controlar la pieza y poder reubicarla en la posición deseada fácilmente.

d. Modificación de las reglas de juego para incorporar la tercera dimensión

- Para proveer al jugador de un juego de Tetris tridimensional con reglas equivalentes a la versión bidimensional se diseñaron un conjunto de reglas que establecen un mecanismo de juego con el mismo objetivo del juego original: llenar niveles del juego con las piezas para eliminar ese nivel y hacer bajar los niveles superiores. Al añadir la tercera dimensión, cada "nivel" de juego es un plano que debe ser rellenado con los cubos que conforman las piezas para que esos cubos puedan ser eliminados y hacer bajar los niveles superiores de cubos. A la matriz del estado de juego en memoria se le incorporó una tercera

dimensión, y los algoritmos de chequeo, borrar línea y de bajar niveles superiores también tuvieron que ser modificados para tomar en cuenta los cambios.

e. Integración de las nuevas reglas con el motor gráfico

- Dado que se añadió una dimensión a la lógica del juego, a la matriz del estado de juego en memoria se le incorporó una tercera dimensión, y los algoritmos de chequeo, borrar línea y de bajar niveles superiores también tuvieron que ser modificados para tomar en cuenta los cambios. La visualización que representa la escena del juego en el motor gráfico es una fiel representación de la información que está almacenada en la matriz en memoria sobre la cual se realizan los cálculos necesarios descritos anteriormente para llevar a cabo las reglas del juego.

3. Implementación

a. Nuevo borde del área de juego

- El nuevo piso del área de juego se diseñó con el software Autodesk Maya utilizando un cubo de medidas veinte (20) cm de ancho por veinte (20) cm de largo por un (1) cm de alto, al cual le fue aplicado un bisel de tres segmentos y un material del tipo Phong de color blanco.

b. Piezas de juego

- Las clases de las piezas escogidas para esta iteración se copiaron al directorio correspondiente de los archivos de recursos del proyecto. Las piezas que no fueron seleccionadas se desecharon en esta versión.
- Algunos de los archivos de materiales asociados a los modelos fueron modificados de manera tal que al ser renderizados por el motor gráfico tuvieran un color más agradable y fácil de distinguir.

c. Mecanismo para mover las piezas con el teclado

- Se tuvo que reasignar algunas teclas a otras funcionalidades del juego y asignar unas nuevas a nuevas funcionalidades tal como se propuso en la etapa de diseño.

d. Reglas del juego Tetris tridimensional

- El funcionamiento básico de las reglas para la versión en dos dimensiones se mantiene ya que el ciclo de juego es exactamente igual. Las modificaciones necesarias incluyeron añadir una dimensión adicional a las matrices (ver apéndice B) "pMatrizMemoria" y "pMatrizNodosOcupados", así como modificar los algoritmos de los métodos "bajarLineasSuperiores" y "borrarLineaCompleta" de la clase "SingletonMatrizMemoria" que fueron renombrados a "bajarPlanosSuperiores" y "borrarPlanoCompleto" para reflejar su nuevo comportamiento que toma en cuenta la tercera dimensión para realizar la acción necesaria en cada caso. La lógica de estos algoritmos se mantiene igual a la lógica implementada en la iteración anterior. Todas las demás reglas implementadas también se mantienen.

e. Integración de las nuevas reglas con el motor gráfico

- La herramienta de depuración implementada en la iteración anterior tuvo que ser desechada porque no hay manera de representar la matriz de tres dimensiones con alguno de los controles que provee la herramienta Visual Studio por defecto para aplicaciones para el sistema operativo Windows.
- Los algoritmos para la rotación de las piezas en tres dimensiones tuvieron que ser implementados de nuevo dado que las piezas pueden rotar sobre tres ejes en vez de uno solo, manteniendo la lógica de la reubicación de los nodos que conforman la pieza rotando siempre sobre el nodo principal de la misma.

4. Pruebas

- Las piezas escogidas para esta iteración representan correctamente el comportamiento deseado para el juego de Tetris. Así mismo, el modelo tridimensional que representa el piso del área de juego logra mostrar correctamente el tamaño y los límites de la misma. El color blanco escogido para el

piso resalta de manera que siempre se distingue cuando es posible colocar una pieza en él, al contrario de lo que ocurre con las piezas ya colocadas que al ser cada una de un color diferente no permiten distinguir en qué nivel está cada cubo que la conforma y resulta muy difícil para el jugador planear la jugada y completar planos con las piezas ya colocadas por encima del piso.

- La pieza en juego puede moverse adecuadamente con el teclado, pero no resulta una experiencia de juego agradable ya que, a pesar de que las teclas escogidas para la rotación de las piezas en cada eje son las más apropiadas para un teclado QWERTY, no son intuitivas y requieren que el jugador haga uso del razonamiento para presionar la tecla haciéndolo perder la concentración necesaria para obtener buenos resultados. Este resultado en las pruebas fue el esperado y es la causa del fracaso comercial de las otras implementaciones del Tetris tridimensional existentes. Además, como el punto de vista del jugador no puede moverse, cuando alguna pieza limita la profundidad del campo visual es muy difícil para el jugador lograr llenar planos completos obstruyendo el cumplimiento del objetivo del juego.
- El comportamiento de las reglas de juego modificadas para añadir la tercera dimensión a la lógica es correcto y satisfactorio en todos los casos.

5. Prototipo funcional

- El prototipo logrado en ésta iteración comprueba el motivo de éste trabajo especial de grado: una implementación del juego de Tetris tridimensional controlada con los periféricos de entrada tradicionales de un computador personal no ofrece usabilidad intuitiva para que el jugador tenga satisfacción y considere el juego divertido.
- A pesar de que el juego cumple las reglas satisfactoriamente, tiene graves deficiencias en cuanto a la percepción del jugador.

IV.1.4 Iteración 4

Como se ha visto, las implementaciones del juego de Tetris tridimensional han fracasado debido a la inefectividad de los periféricos usuales de un computador de escritorio para ofrecer al jugador unos instrumentos intuitivos y de gran usabilidad. Por lo tanto y dado que el presente trabajo de grado propone utilizar un dispositivo más apropiado para la tarea como el control inalámbrico de la consola de juegos Nintendo Wii para ofrecer una experiencia de juego más satisfactoria, en ésta iteración se incluyó la funcionalidad para utilizar este control, además del uso de otras técnicas para mejorar la curva de aprendizaje y el perfeccionamiento de la destreza y habilidad del jugador con el fin de dominar el juego.

1. Recolección de Requerimientos
   a. Agregar la funcionalidad de comunicación con el control inalámbrico de Nintendo Wii (Wiimote)
   b. Interpretar las medidas de aceleración del Wiimote
   c. Implementar un mecanismo para controlar el punto de vista del jugador (cámara)
   d. Integrar los comandos leídos del Wiimote con el motor gráfico y la lógica del juego
   e. Crear visualizador para predecir la posición final de la pieza en juego
   f. Implementar algoritmo para colorear cada nivel de piezas colocadas en el área de juego en un color particular
2. Diseño rápido
   a. Comunicación con el Wiimote
      - Para establecer la comunicación del computador con el control inalámbrico de Nintendo Wii se hizo uso de la librería WiimoteLib. Ésta librería provee al desarrollador el Application Programming Interface (API) necesario para obtener los valores capturados por los diferentes sensores que tiene el control, resolviendo todo el trabajo asociado a la conexión y desglose de los

datos enviados por el control a través de la interfaz Bluetooth del computador.

b. Interpretación de las medidas de aceleración del Wiimote

- El objeto de la librería WiimoteLib que representa el control inalámbrico (Wiimote) tiene un conjunto de propiedades con valores entre los que se encuentra la lectura en el momento de las fuerzas a las que está siendo sometido el acelerómetro integrado al control.
- La fuerza gravitacional de la Tierra es representada en la lectura que hace el sensor, con lo cual el juego puede tener una referencia hacia la misma y por consecuencia una referencia a la manera en que esté inclinado el control. Como el acelerómetro del Wiimote es de tres ejes (ver ilustración 23), cuando ocurre una transición de la fuerza de un eje a otro el juego puede determinar que el control ha sido inclinado y que en ese momento el vector que define la fuerza de la gravedad de la Tierra es paralelo al eje correspondiente luego de la transición.
- Cuando dicha transición sea detectada, el programa principal del proyecto deberá invocar al método de rotación sobre el eje correspondiente de la pieza en juego actual. Para evitar que el jugador rote la pieza sin intención, se requiere que el jugador hale del botón de gatillo del Wiimote al momento de inclinar el control para que el método de rotación correspondiente sea ejecutado.

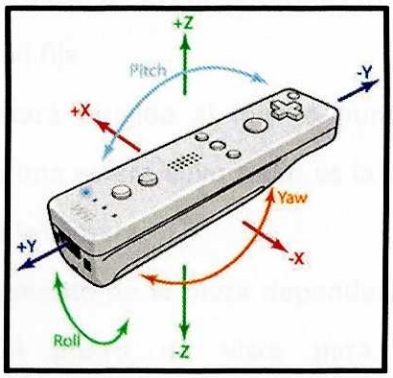

Ilustración 23: Nomenclatura de los ejes con respecto al Wiimote
[Fuente: http://hacknmod.com/hack/easily-connect-a-wiimote-to-your-pc/es/]

c. Mecanismo para controlar la cámara

- Para controlar el punto de vista del jugador, o la cámara como es llamada por la documentación del motor gráfico, se decidió utilizar la palanca integrada en la extensión "Nunchuk" del control inalámbrico Wiimote. Los controles del Nunchuk han de ser asignados de la siguiente manera (ver tabla 5):

COMANDO	ACCIÓN
Palanca arriba	Desplazamiento de la cámara hacia arriba
Palanca abajo	Desplazamiento de la cámara hacia abajo
Palanca derecha	Desplazamiento de la cámara hacia la derecha
Palanca izquierda	Desplazamiento de la cámara hacia la izquierda
Botón Home	Bajar la pieza rápidamente
Botón A	Rotación de la pieza sobre el eje Y en sentido horario

Tabla 5: Acciones asignadas a los botones del Nunchuk
[Fuente: Elaboración Propia]

- La posición de la cámara en la escena tridimensional es definida por un vector de tres dimensiones. Cuando el jugador haga uso de la palanca del Nunchuk, la librería WiimoteLib detectará a qué posición está siendo empujada y en función de esto la cámara subirá, bajará, se moverá a la izquierda o derecha; haciendo un recálculo del vector que define su posición en base a la posición anterior. Como la palanca del Nunchuk es progresiva, es decir, tiene valores intermedios entre los valores extremos, a medida que la palanca sea empujada más o menos respecto a su posición central la cámara se moverá rápida o lentamente. Así mismo, para acercar o alejar la cámara se hace uso de los dos botones frontales del Nunchuk, en este caso a una velocidad fija.
- La cámara siempre estará dirigida al mismo punto de manera que su movimiento será sobre una esfera cuyo radio es la distancia del punto fijo de mira a la posición de la cámara.
- Los comandos de movimiento de la pieza dependerán del cuadrante en el que se encuentre el punto de vista para poder mantener el

comportamiento de los comandos independientemente del punto de vista en el momento en que sean ejecutados.

d. Integrar los comandos del Wiimote con el resto del proyecto

- Para hacer la integración de los comandos leídos desde el Wiimote, se diseñó una interfaz donde se enlazan los comandos con acciones de la lógica y el motor gráfico. Se propone hacer uso de los botones del Wiimote de la siguiente manera (ver tabla 6):

COMANDO	ACCIÓN
Botón arriba	Desplazamiento eje Z positivo
Botón abajo	Desplazamiento eje Z negativo
Botón derecha	Desplazamiento eje X positivo
Botón izquierda	Desplazamiento eje X negativo
Botón Home	Bajar la pieza rápidamente
Botón A	Rotación de la pieza sobre el eje Y en sentido horario
Botón B	Habilita la rotación la pieza inclinando el control

Tabla 6: Acciones asignadas a los botones del Wiimote
[Fuente: Elaboración Propia]

- Como la rotación de la pieza sobre el eje Y no puede ser detectada a partir de la fuerza de la gravedad de la Tierra, se asignó el botón A del control para realizar dicha rotación en sentido horario únicamente.

e. Visualizador de la posición final de la pieza en juego

- Cuando el jugador está haciendo su partida no es fácil predecir donde caerá la pieza si decidiera dejarla caer en algún momento a su posición final dado que bajo el diseño realizado para el escenario del juego no hay puntos de referencia claros sobre los cual realizar medidas rápidas de aproximación. Por tal motivo, se decidió diseñar un método para que sea intuitivo para el jugador saber la posición final de la pieza cuando termine de bajar todos los niveles. Luego de varias propuestas fue escogido una proyección que representará la pieza actual en juego apoyada sobre el nivel más bajo que pueda descender con un color transparente para no entorpecer el campo

visual del jugador. Cada vez que ocurra un cambio de posición o rotación en la pieza, el fantasma será recalculado para mostrar su nueva posición final.

f. Algoritmo para asignar un color a cada nivel

- A medida que el jugador va apilando piezas en el área de juego se hace más complicado llevar cuenta de cuáles cubos están en el mismo nivel para poder predecir las jugadas subsecuentes y lograr el objetivo del juego. Entonces se diseñó un proceso mediante el cual los cubos que conforman la pieza son pintados de un color distintivo dependiendo del nivel en el que se encuentren cuando la pieza llega a su posición final.

3. Implementación

a. Comunicación con el Wiimote

- La librería utilizada para establecer la comunicación con el control inalámbrico fue WiimoteLib en su versión 1.7. Establece un conjunto de eventos, objetos y propiedades que permiten manipular la interacción con el Wiimote (ver tabla7). (Ver el apéndice D)
- Cabe destacar que para que la librería pueda conectar el control, el mismo debe ser añadido previamente como dispositivo vinculado al gestor de comunicaciones Bluetooth instalado en el computador en el que desee ser utilizado (Bluesoleil, Widcomm, o el stack de Microsoft).

EVENTO / METODO	DESCRIPCIÓN
WiimoteChanged	Reporta cambios en los sensores del control
WiimoteExtensionChanged	Detecta cuando se conecta o desconecta una extensión al Wiimote
Connect	Establece la comunicación con el Wiimote
Disconnect	Termina la comunicación con el Wiimote

Tabla 7: Eventos y métodos de la librería WiimoteLib utilizados
[Fuente: Elaboración Propia]

b. Interpretación de la fuerza de aceleración

- El acelerómetro del Wiimote muestra a través de la librería utilizada tres valores que representan la fuerza de aceleración a la que está sometida cada uno de sus tres ejes. La fuerza de la gravedad de la Tierra es detectada por el sensor, mostrando el valor -1 en la componente del eje Y del control

y el valor 0 en las otras dos componentes. El rango de valores posibles para cada eje es de -1 a 1 con precisión de centésimas.

- Para detectar que efectivamente hay un cambio o transición de la referencia a la gravedad de la Tierra, el valor de la componente en el eje sobre la que se está calculando la transición debe pasar un umbral de +0.5 (cuarenta y cinco grados positivo) de rotación o -0.5 (cuarenta y cinco grados negativo) de rotación sobre el eje perpendicular al que se está detectando (por ejemplo, si se está detectando la transición al eje X del sensor, el control debe rotar unos 45 grados sobre su eje Z).
- Una vez detectado que el valor ha rebasado el umbral para detectar la transición, el signo del valor determina la dirección de la rotación que puede ser en sentido horario o anti horario con lo que se determina el algoritmo de rotación particular a invocar.
- Como fue discutido en el diseño de la solución al problema, el jugador debe presionar el gatillo (botón B) del Wiimote para que las transiciones de la referencia a la fuerza de la gravedad ejecuten los comandos de rotación de la pieza.

c. Movimiento de la cámara con el Nunchuk

- La librería WiimoteLib también provee acceso a los valores reportados por los sensores de las extensiones que se conecten al Wiimote. En el caso de la extensión Nunchuk, los valores de la palanca y el estado de los botones pueden ser obtenidos fácilmente. Para la palanca se puede obtener el desplazamiento a la que esté siendo sometida en un rango de valores de -1 a 1 en los ejes X y Y, dependiendo del signo se determina la dirección a la que esté siendo empujada la palanca. Así, para detectar que el jugador desea mover la cámara el valor reportado por cada eje deberá rebasar el umbral de ±0.1 y el valor absoluto del valor reportado determina a través de la fórmula "(Valor Detectado * 50) / 0.5" la velocidad de giro de la cámara. Esos valores fueron conseguidos mediante prueba y error,

ofreciendo una justa velocidad máxima de rotación de la cámara y un control progresivo de la misma bastante preciso. La medida de la velocidad en el eje correspondiente llega al programa principal y mediante un vector auxiliar de traslación en cada ciclo de renderización del motor gráfico se modifica la posición de la cámara. Cuando el valor en cada eje vuelve a ser menor a ±0.1 el vector auxiliar de traslación se evalúa en 0 para el eje correspondiente, anulando el movimiento de la cámara en ese eje durante los ciclos de renderización deseados.

- Dado que al mover la cámara la asignación de los controles de movimiento y rotación de las piezas deben ser reasignados para mantener el comportamiento deseado, se elaboró un algoritmo que es ejecutado cada vez que se mueve la cámara que determina sobre qué cuadrante del área de juego está localizado el punto de vista del jugador (Ver tabla 8) (Ver el apéndice E). La cámara siempre mira al punto (0,0,0) y por lo tanto se mueve sobre una esfera como se puede apreciar en la ilustración 24.

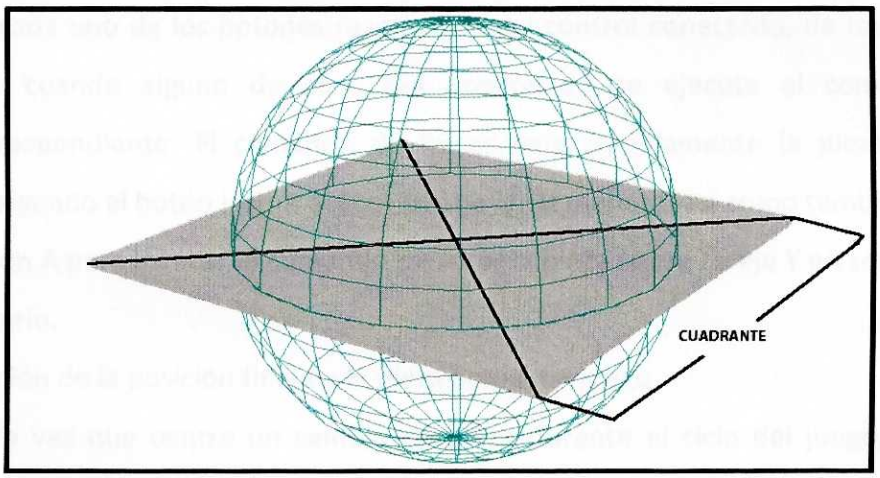

Ilustración 24: Esfera del movimiento de la cámara
[Fuente: Elaboración Propia]

α MINIMO	α MAXIMO	ESTADO DE LA CAMARA
315	45	1
45	135	2
135	225	3
225	315	4

Tabla 8: Estados de la cámara con respecto al cuadrante donde se encuentre
[Fuente: Elaboración Propia]

d. Integración Wiimote-Ogre

- Para separar la lógica del juego, el motor gráfico y los comandos del control inalámbrico se realizó un diseño en capas. La comunicación con el Wiimote fue implementada en una clase llamada "InterfazWiimote" que contiene todos los cálculos correspondientes a la interpretación de los valores reportados por los diferentes sensores del control, además de los métodos encargados de realizar la conexión mediante la librería WiimoteLib.
- El programa principal de la aplicación tiene una instancia de la clase "InterfazWiimote", y el constructor clase "InterfazWiimote" recibe como parámetro la instancia del programa principal para poder invocar los métodos de movimiento y rotación de la pieza actual en juego a partir de los comandos recibidos por el control.
- La reasignación de los comandos de movimiento de la pieza sobre el nivel en el que se encuentre (X,Z) a la tecla en forma de cruz del Wiimote, fue realizada con una simple consulta a la librería WiimoteLib acerca del estado de cada uno de los botones respectivos del control conectado, de manera que cuando alguno de ellos sea presionado se ejecute el comando correspondiente. El comando de hacer bajar rápidamente la pieza fue reasignado al botón Home del control de igual manera, así como también el botón A para invocar el comando de rotar la pieza sobre su eje Y en sentido horario.

e. Proyección de la posición final de la pieza actual en juego

- Cada vez que ocurre un cambio de pieza durante el ciclo del juego, una nueva instancia de la clase "PiezaProyeccion" es creada, y su constructor recibe como parámetro la forma de la pieza seleccionada por el programa principal de la aplicación para que el fantasma pueda ser mostrado correctamente, y también se coloca en la misma posición que la pieza actual en juego. Luego se ejecuta el método "colocarAlFondo" que hace llamadas al método "moverAbajo" la cantidad de veces que pueda mientras

"chequearAbajo" devuelva el valor VERDADERO, indicando que la pieza puede bajar de nivel.

- Siempre que la pieza actual en juego sea movida o rotada la proyección es reubicada, dispuesta en la misma posición que la pieza actual y luego colocada al fondo.

f. Asignar color para cada nivel de piezas colocadas

- En el momento que una pieza llega a su posición final, se ejecutan un par de métodos que modifican la propiedad Material de los nodos que la conforman según una tabla que define el color que deben tener los nodos en cada nivel.
- De la misma manera se modifica la propiedad Material cuando un nivel del área de juego es completado y los nodos de los niveles superiores bajan, representando el cambio pintando los nodos del color correspondiente al nivel donde fueron reubicados.

4. Pruebas

- El Wiimote logra comunicarse satisfactoriamente con el computador donde corre la aplicación a través del protocolo Bluetooth. Los eventos configurados son capturados de manera correcta y los métodos asociados a manejar estos eventos son invocados como se esperaba. La librería WiimoteLib ofreció un entorno muy limpio, claro y fácil para realizar la comunicación con el control.
- Efectivamente el diseño y la implementación logrados para interpretar los valores reportados por el sensor de aceleración del Wiimote son adecuados para una versión más amigable del juego de Tetris 3D. Uniendo el control de la pieza actual en juego mediante el control inalámbrico y su acelerómetro, con el control del punto de vista del jugador (cámara) con la extensión Nunchuk; el conjunto ofrece unas características de usabilidad y jugabilidad que cumplen el objetivo de dar al jugador una experiencia mucho más satisfactoria y placentera que las versiones del juego de Tetris tridimensional manejado con los periféricos usuales de un computador personal (ratón y teclado).

- Como ocurre en el punto anterior, el movimiento de la cámara con el Nunchuk es el esperado; y el hecho de poder variar la velocidad del movimiento es una experiencia intuitiva para el jugador. Como la cámara siempre mira hacia el mismo punto, es muy fácil ubicarla en el mejor sitio para la jugada.
- La comunicación entre las capas de interfaz con el Wiimote y la lógica del juego se realiza satisfactoriamente, recibiendo los comandos del control y ejecutando los métodos correspondientes implementados en las clases que representan las piezas del juego. Como las capas de la lógica y el motor gráfico estaban separadas desde las iteraciones anteriores, no existió necesidad de realizar cambios en éste ámbito.
- La proyección que predice la posición final de la pieza actual en juego logra su objetivo satisfactoriamente. Al ser transparente no obstruye la visibilidad del jugador para planear su jugada siguiente; ofreciendo un nivel superior de jugabilidad a la aplicación.
- Al igual que la proyección, el procedimiento de colorear cada nivel de nodos correspondientes a piezas ya colocadas de un color particular, aumenta la facilidad para el jugador de predecir sus jugadas siguientes y lograr el objetivo del juego.

5. Prototipo funcional

- En el prototipo funcional conseguido en la última iteración del proyecto logran vencerse la mayoría de los inconvenientes asociados a una implementación del juego de Tetris en tres dimensiones. El dispositivo de control escogido es bastante exitoso a la hora de ofrecer una interfaz humano-computador de mayor calidad para el usuario, en el contexto de las características intuitivas de su funcionamiento. Así mismo ocurre con los procedimientos añadidos para mejorar la capacidad del jugador de dominar el juego como el fantasma de predicción de posición final de la pieza y el color de cada nivel de los cubos (o nodos) que conforman las piezas ya colocadas.

CAPÍTULO V

V.1. RESULTADOS

Los resultados de este Trabajo Especial de Grado deben medirse con respecto a los objetivos establecidos durante el planteamiento del problema. Por lo tanto se realizará un análisis para cada objetivo argumentando los resultados obtenidos en base a las soluciones diseñadas e implementadas.

Conceptualización, diseño y creación de los modelos tridimensionales del juego.

Gracias a que se tuvo una etapa de conceptualización del juego bastante profunda pudo determinarse con alto grado de acertamiento los modelos tridimensionales que tuvieron que ser diseñados y creados. A través de un rápido entrenamiento en el uso de la herramienta Autodesk Maya fue posible lograr plasmar el diseño ideado en modelos tridimensionales que pudieron ser utilizados en el motor gráfico. El modelo tridimensional del delimitador del área de juego es bastante simple y aun así logra transmitir la información que necesita el jugador de manera de poder planificar su jugada con éxito. El modelo tridimensional del cubo base permite un gran grado de flexibilidad a la hora de diseñar las piezas del juego y también puede ser reutilizado como nodo individual, es decir, como mínima expresión de las piezas colocadas en el área de juego que necesitan ser desmembradas al borrar planos completos.

Conceptualización, diseño y creación del escenario de juego.

Durante la etapa de conceptualización se hizo énfasis en la simplicidad y usabilidad que debían reflejarse en el escenario de juego. Es un escenario verdaderamente sencillo y por tal razón el concepto pudo materializarse a plenitud en la escena del motor gráfico Ogre 3D así como también ofrece al jugador una experiencia directa y sin elementos de embellecimiento que pueden afectar su concentración.

Implementación de la lógica y reglas del juego

La implementación de las reglas del juego fue la tarea más importante de este Trabajo Especial de Grado ya que para obtener un prototipo funcional y completo del juego se tuvo que dedicar mucho tiempo al análisis y optimización de los algoritmos que

llevan a cabo las reglas y los algoritmos de autorización para la ejecución de acciones del jugador. El conjunto de reglas implementadas para la última iteración están completas para considerar al juego una implementación de Tetris tridimensional, y gracias a que las reglas fueron evolucionando durante tres iteraciones el grado de complejidad de las mismas fue progresivo ayudando a que pudieran probarse también progresiva y completamente.

Diseño e implementación de la aplicación en torno al motor de gráficos OGRE (Object-Oriented Graphics Rendering Engine)

El motor gráfico OGRE 3D ofrece gran flexibilidad tanto desde el punto de vista de codificación en el lenguaje C++ como desde el punto de vista de representación de escenas. Requiere un nivel intermedio de entrenamiento para su uso y por tal razón se usó una interfaz que permite utilizarlo en el lenguaje de programación C# (Mogre), bajando el nivel de complejidad de la codificación y permitiendo aprovechar el tiempo para desarrollar los algoritmos de las reglas del juego. La aplicación fue implementada a partir de un ejemplo disponible en la página web de Mogre siguiendo los estándares propuestos por sus creadores y de esta manera simplificar al máximo su uso. Desde el primer prototipo el motor gráfico funciona completamente y de ahí en adelante el resto de la aplicación fue diseñada e implementada en torno al mismo.

Implementación de la interpretación necesaria de los datos obtenidos del acelerómetro incorporado al control inalámbrico de Nintendo Wii para utilizarlo como sensor de orientación de las piezas del juego, empleando alguna de las librerías disponibles para tal fin

Fue utilizada la librería WiimoteLib que permite tener acceso a los valores reportados por el acelerómetro del Wiimote de una manera muy sencilla. La interpretación de esos valores fue implementada en un conjunto de algoritmos que describen una lógica donde se determina la acción solicitada por el jugador a través de cálculos matemáticos sencillos.

Integración de la lógica y reglas del juego con el ambiente tridimensional y el control inalámbrico

El proyecto descrito en este Trabajo Especial de Grado se divide en tres grandes partes: las reglas del juego, el ambiente tridimensional y el control inalámbrico. Las primeras dos partes fueron evolucionando durante el curso de la segunda y tercera iteración y en cada una se llevaron a cabo actividades de integración de la lógica y reglas del juego con el ambiente tridimensional de manera que los prototipos funcionales de esas iteraciones estuvieran completos. Durante la cuarta iteración se integró la tercera parte, el uso del Wiimote como dispositivo de control del juego después de haber resuelto los algoritmos de interpretación de las acciones solicitadas por el jugador.

Las soluciones elaboradas durante el transcurso del proyecto logran satisfacer todos los objetivos planteados en este Trabajo Especial de Grado.

CONCLUSIONES

Se obtuvieron el siguiente conjunto de conclusiones durante el desarrollo del proyecto:

- El Tetris es un juego que ofrece una experiencia muy satisfactoria y gratificante ya que a pesar de que demanda al jugador un alto nivel de concentración tiene un conjunto de reglas sencillas y fáciles de aprender. Por estas razones, el Tetris es un juego de por sí exitoso y por lo tanto permite que sea una opción ideal para probar nuevas tecnologías para su implementación dado que esa tecnología utilizada para su desarrollo puede determinar el éxito o el fracaso de esa implementación.
- La metodología basada en prototipos evolutivos es adaptable a cualquier tipo de proyecto y su flexibilidad ofrece grandes ventajas al desarrollador para dividir el trabajo en secciones de acuerdo a la complejidad que implique su desarrollo. De esta manera este proyecto pudo crecer progresivamente, resolviendo problemas particulares sin profundidad y amplitud muy grandes en ningún caso.
- El motor gráfico Ogre 3D permite desarrollar aplicaciones que tengan un alto nivel de complejidad en cuanto a gráficos tridimensionales generados en tiempo real, utilizando técnicas avanzadas como el shading, sombras e iluminación con muy buen rendimiento sobre hardware de capacidad baja-media.
- El control inalámbrico de Nintendo Wii (Wiimote) permite desarrollar aplicaciones con las que un usuario puede obtener muy buena calidad de interacción gracias al acelerómetro integrado que da la capacidad de interpretar la acción que el usuario desee realizar en un momento determinado a partir del movimiento al que sea sometido. Como es un dispositivo disponible ampliamente y de bajo costo su uso no impone obstáculos para el despliegue de una aplicación.
- Una aplicación que ha sido desplazada por nuevas tecnologías puede volver a ser gratificante para el usuario si se implementa de nuevo con tecnología actual y se hace un cambio de paradigma sobre el concepto que permita utilizarlo de nuevo sin perder usabilidad en la transición.

- Es importante dar al usuario todas las ayudas posibles para que logre llevar a cabo con éxito la tarea propuesta. Por ejemplo, la proyección de predicción de la posición final de la pieza, es una característica que le sirve al jugador para planificar su jugada sin exigirle un análisis mental muy profundo brindando una experiencia agradable.

RECOMENDACIONES

- Debe tenerse mucho cuidado con el uso de tecnologías que impliquen disminuir el grado de usabilidad de una aplicación y compensarlas con cambios en el concepto u otras tecnologías que disminuyan el grado de dificultad añadido.
- Para otras implementaciones del juego deberían agregarse elementos que ayuden a aumentar la inmersión del jugador como sonidos y efectos visuales.
- El acelerómetro del Wiimote es un sensor bastante preciso con respecto al pulso de la mano del jugador, por lo que las vibraciones del pulso hacen fluctuar al valor que representa la referencia a la gravedad de la Tierra entre los valores mayores y menores al umbral de rotación cuando el control está inclinado justamente sobre el valor de transición de estado. Ya que la pieza cambia de estado de rotación cada vez que esto ocurre, se observa un comportamiento extraño al respecto. Debe suavizarse la transición de un estado de rotación al siguiente y al anterior para que la pieza no quede en un estado de rotación indeterminado.
- El uso de la interfaz Mogre para poder desarrollar el proyecto en el lenguaje C# implica una reducción del rendimiento del motor gráfico Ogre 3D, por lo que en aplicaciones en las que el rendimiento sea un factor crítico deben desarrollarse en C++ que es el lenguaje nativo de Ogre 3D.
- Los algoritmos de recorrido de matrices utilizados en proyectos como el que se presenta deben ser lo más eficientes posibles para no desperdiciar ciclos de reloj del computador, aumentando el rendimiento general de la aplicación.

BIBLIOGRAFÍA

Libros

Junker, G. (2006). *Pro OGRE 3D Programming*. Berkeley: Apress.

[Guerrero, 1996] Guerrero, J. (1996). *Realidad Virtual, La última frontera*. España: Abeto Editorial.

Varios Autores. (1999). *Enciclopedia de los Juegos* (1ª ed.). España: Editorial Paidotribo.

Landa, N. (2006). *DirectX: Programación de gráficos 3D* (1ª ed.). Argentina: Gradi S.A.

Wright, C. (2003). *Superutilidades para C#* (1ª ed.). España: McGraw-Hill.

Archer, T. (2001).*A fondo C#* (1ª ed.). España: McGraw-Hill.

Derakhshani, D. (2009). *Introducing Maya 2009*. Estados Unidos: Wiley Publishing.

Sommerville, I. (2005). *Ingeniería del Software* (7ª ed.). España: Pearson Educación.

Sears, A. y Jacko, J. (2008). *The Human-Computer Interaction Handbook* (2ª ed.). Estados Unidos: Lawrence Erlbaum Associates.

Pressman, R. (2001). *Software engineering* (5ª ed.). Estados Unidos: McGraw-Hill.

Zaphiris, P. y Ang, C. (2009). *Human Computer Interaction*. Estados Unidos: Information Science Reference.

Tucker, A. (2004). *Computer Science Handbook* (2ª ed.). Estados Unidos: Chapman & Hall/CRC.

[Golomb, 1996]. Golomb, S. (1994). *Polyominoes* (2ª ed.). Estados Unidos: Princeton University Press.

Eco, U. (1982). *Cómo se hace una tesis* (2ª ed.). Argentina: Editorial Celtia.

Day, R. (1994). *Cómo escribir y publicar trabajos científicos* (4ª ed.). Estados Unidos: The Oryx Press.

Trabajos Especiales de Grado

Álvarez, L. y Valor, A. (2008). *Diseño y construcción de un ambiente de interacción orientado al desarrollo de un escritorio experimental 3D y a la adaptación de los dispositivos de detección de movimientos que permitan su control*. Tesis de Pre-Grado, Escuela de Ingeniería Informática, Universidad Católica Andrés Bello.

Documentos electrónicos

The Windows Team (2002). *Página de definición de gravedad*, [en línea]. Michigan: University Corporation of Atmospheric Research. Disponible en: http://www.windows.ucar.edu/tour/link=/glossary/gravity_defn.sp.html&edu=high [Diciembre, 2008].

Tetris, a history, [en línea]. Champaign, Illinois, Estados Unidos. Disponible en: http://atarihq.com/tsr/special/tetrishist.html [Diciembre, 2008].

[Gerasimov, Sin Fecha] Gerasimov, V. *Tetris Story*, [en línea]. Sidney: Massachusetts Institute of Technology. Disponible en: http://vadim.oversigma.com/Tetris.htm [Enero, 2009].

Dimension Engineering. *A beginner's guide to accelerometers*, [en línea]. Akron: Dimension Engineering, Akron, Ohio, Estados Unidos. Disponible en: http://www.dimensionengineering.com/accelerometers.htm [Diciembre, 2008].

Sagem Défense Sécurité (2009). *Avionics Systems & Navigation: Navigation*, [en línea]. Francia: Sagem Défense Sécurité. Disponible en: http://www.sagem-ds.com/eng/site.php?spage=02010301 [Abril, 2009].

MEMS Inertial Sensors FAQ (Sin Fecha), [en línea]. Disponible en: http://www.analog.com/en/technical-library/faqs/design-center/faqs/CU_faq_MEMs/resources/fca.html [Julio, 2009].

Bluetooth Basics (Sin Fecha), [en línea]. Estados Unidos: Bluetooth Special Interest Group. Disponible en: http://www.bluetooth.com/Bluetooth/Technology/Basics.htm [Abril, 2009].

[Strickland, Sin Fecha] Strickland, J. (2007) *How Virtual Reality Works*, [en línea] Estados Unidos: HowStuffWorks.com. Disponible en: http://electronics.howstuffworks.com/gadgets/other-gadgets/virtual-reality.htm# [Julio, 2009].

About Ogre. (Sin Fecha), [en línea]. Disponible en: http://www.ogre3d.org/about [Agosto, 2009].

[Mogre Basic Tutorial 5, Sin Fecha] Culvier, C. (Sin Fecha) *Mogre Basic Tutorial 5*, [en línea]. Ogre Wiki. Disponible en: http://www.ogre3d.org/wiki/index.php/Mogre_Basic_Tutorial_5 [Agosto, 2009]

[.NET Managed Library for the Nintendo Wii Remote, 2008]Peek, B. (2008) WiimoteLib - .NET Managed Library for the Nintendo Wii Remote, [en línea]. Brian Peek's Blog. Disponible en: http://www.brianpeek.com/blog/archive/2008/06/07/wiimotelib.aspx

APÉNDICE C: DETALLE DEL CONVERTIDOR DE COORDENADAS DE LA MATRIZ EN MEMORIA A COORDENADAS DEL MOTOR GRÁFICO

Suponiendo que el nodo base tiene coordenadas reales (50,50,0) entonces un nodo auxiliar cuya posición es calculada en base a las coordenadas imaginarias sobre la matriz del nodo base con la fórmula (Bx-1, By) las coordenadas reales del nodo serán (50 – (Arista del cubo), 50,0). Se le resta el tamaño de la arista del cubo porque en el motor 3D el punto central del cubo está en el centro del mismo, entonces para colocar dos cubos de manera adyacente hay que separar los centros de cada uno la distancia equivalente al tamaño completo de un cubo, o en otras palabras la mitad del tamaño de dos cubos. El código que hace la conversión en la segunda iteración es el siguiente:

```
publicclassPosicionMemoria
    {
publicint x;
publicint y;


public PosicionMemoria()
        {
            x = 0;
            y = 0;
        }
    }

publicstaticclassMapeadorOgreMemoria
    {
publicstaticPosicionMemoria calculaPosicionMemoria(int xGrafico, int
yGrafico)
        {
PosicionMemoria laPosicionMemoria = newPosicionMemoria();
            laPosicionMemoria.x +=
xGrafico/Constantes.multiplicadorPosiciones + 4;
            laPosicionMemoria.y +=
yGrafico/Constantes.multiplicadorPosiciones + 9;

return laPosicionMemoria;
        }
    }
```

APÉNDICE D: UBICACIÓN INICIAL DE LOS NODOS AUXILIARES DE LAS PIEZAS FORMADAS

Dado que el espacio de juego es de 12x10 casillas, se debe trabajar en base a una matriz imaginaria de 12x10 posiciones. Las coordenadas de cada casilla de la matriz tienen una correspondencia en el escenario 3D que maneja el motor gráfico. Para encontrar dicha equivalencia se implementó un convertidor de coordenadas gráficas a coordenadas lógicas y viceversa.

Para resolver las posiciones de los nodos auxiliares a partir de la posición del nodo base en la matriz imaginaria, entonces si las coordenadas del nodo base son (Bx, By) las coordenadas del nodo auxiliar 1 son (Bx – 1, By) dado que este nodo debe estar trasladado una posición a la izquierda (en el eje X) Bx – 1, pero quedarse en la misma posición del eje Y del nodo base By. Así mismo, para los demás nodos auxiliares se realizan las traslaciones correspondientes en cada ordenada sea en el eje X o en el eje Y para formar la pieza correspondiente. Los nodos auxiliares están numerados en sentido horario empezando por el cuadrante superior izquierdo en la posición inicial de la pieza formada. Luego cuando la pieza sea rotada el orden cambiará pero ya en ese momento estarán almacenadas las referencias a cada nodo particular.

Un ejemplo de código para la pieza en forma de cuadrado de la iteración dos es el siguiente:

```
protectedoverridevoid asignarPosicionInicial()
        {
            nodoPivote.Position =
vectorPosicionRelativa.posicionRelativa(0, 0, -9);
//Es -9 en la Z porque se trabaja con el 0,0,0 en el centro de la grilla
//y luego el vector la multiplica por la constante del multiplicador de
//posicion definida en la clase Constantes

            nodosAuxiliares[0].Position = nodoPivote.Position +
vectorPosicionRelativa.posicionRelativa(1, 0, 0);
            nodosAuxiliares[1].Position = nodoPivote.Position +
vectorPosicionRelativa.posicionRelativa(1, 0, 1);
            nodosAuxiliares[2].Position = nodoPivote.Position +
vectorPosicionRelativa.posicionRelativa(0, 0, 1);

            actualizarMemoria();

            estadoRotacion = 1;
        }
```

APÉNDICE E: DETALLE DE IMPLEMENTACIÓN DEL CICLO Y LAS REGLAS DEL JUEGO

- Ciclo del juego: En el primer nivel del juego, cada 800 milisegundos se ejecuta un método que corre un algoritmo de verificación para chequear si la pieza en juego puede bajar un nivel. Implementado de la misma manera que los métodos de chequeo de movimiento a los lados, devuelve el valor VERDADERO si la pieza puede bajar un nivel o el valor FALSO en caso contrario. Si la pieza puede bajar se procede a invocar el método de movimiento hacia debajo de la pieza en juego. Y si la pieza no puede bajar, entonces el objeto que representa la pieza en juego es desechado para proceder a seleccionar aleatoriamente alguna de las piezas posibles incluyendo: pieza en L, pieza en Z a la izquierda, pieza en Z a la derecha, pieza en T, pieza en I y pieza en cuadrado, ETC. La pieza escogida aleatoriamente es instanciada y su referencia es almacenada en el objeto de tipo PiezaAbstracta del programa principal. Este objeto representa la "pieza en juego" como concepto.
- Reglas del juego: Al finalizar cada ciclo de juego el jugador obtiene una nueva pieza para controlar. Cuando la pieza en juego no puede bajar un nivel, se marca en la matriz de memoria las posiciones ocupadas por la pieza. Esta matriz "pMatrizMemoria" se referencia en una clase de tipo Singleton "SingletonMatrizMemoria", y almacena valores del tipo del enumerador "estadoPosicionMemoria". Este enumerador está definido globalmente al namespace Tetris2D que incluye todo el proyecto. Los valores posibles de dicho enumerador son "vacío, ocupado y actual". La pieza que no puede bajar un nivel marca las posiciones que ocupa en las matrices cuando el programa principal invoca el método marcarPosicionesOcupadas() de la pieza en cuestión. También se lleva una matriz llamada "pMatrizNodosOcupados" que almacena referencias a objetos SceneNode de Ogre, definida también en la clase "SingletonMatrizMemoria". Cuando el método marcarPosicionesOcupadas correspondiente se ejecuta, se asigna el valor OCUPADO del enumerador "estadoPosicionMemoria" a las posiciones correspondientes de los nodos que conforman la pieza en cuestión en la matriz "pMatrizMemoria". También se almacenan

en la matriz "pMatrizNodosOcupados" las referencias a los objetos SceneNode que conforman la pieza, que son los nodos mismos a los que la pieza guarda referencia localmente. Se almacenan allí para luego poder borrarlos según sea necesario durante el juego.

Luego la bandera "cambioDePieza" del programa principal cambia al valor VERDADERO para indicar al ciclo de juego la acción a realizar en la próxima iteración según el tiempo de juego actual. De esta manera, se invoca al método borrarLineaCompleta() de la clase "SingletonMatrizMemoria". Este método itera sobre la matriz "pMatrizMemoria" de abajo hacia arriba y de izquierda a derecha. Al recorrer la matriz el algoritmo compara si en una línea horizontal todas las posiciones de la matriz tienen el valor OCUPADO; si esta condición se cumple entonces se procede a eliminar gráficamente los nodos que ocupan esa línea utilizando las referencias almacenadas en la matriz "pMatrizNodosOcupados", haciendo una llamada por cada uno a un método que realiza esa función en el motor gráfico Ogre. También se asigna el valor VACIO a la matriz "pMatrizMemoria" en las posiciones correspondientes a la línea a eliminar. Luego se procede a realizar la llamada al método bajarLineasSuperiores definido también en la clase "SingletonMatrizMemoria"; recibe como parámetro la línea en la matriz desde la cual deben bajar todas las líneas superiores.

Este método itera sobre la matriz "pMatrizMemoria" de abajo hacia arriba y de izquierda a derecha preguntando en cada posición si la misma está marcada como OCUPADA. Si esto es así entonces se asigna dicha posición como VACIA, se marca la misma ordenada X del nodo en la línea inferior como OCUPADA. Luego la posición de la instancia de SceneNode ubicada en esa posición de la matriz "pMatrizNodosOcupados" se cambia para que se refleje el cambio visualmente en el motor de gráficos Ogre calculando las nuevas coordenadas del cubo asociado al nodo utilizando el convertidor implementado en la primera iteración. Por último la posición en la matriz "pMatrizNodosOcupados" del SceneNode en cuestión se cambia a la línea inferior en la misma ordenada X para que el cambio corresponda con la matriz "pMatrizMemoria" y se mantenga la integridad de la información almacenada en memoria y el proceso sea

consistente para las demás invocaciones del proceso completo de eliminar y bajar líneas del juego.

Cuando el programa principal detecta que el proceso se ha ejecutado a través del valor de vuelta de la llamada al método borrarLineaCompleta(), los puntos acumulados por el jugador aumentan en 100 unidades; y cada 1000 unidades de puntos acumulados el tiempo del ciclo de jugo disminuye en 100 milisegundos.

Para finalizar el proceso, la variable "piezaActual" del programa principal obtiene una nueva instancia de alguna pieza seleccionada aleatoriamente; la bandera "cambioDePieza" vuelve al valor FALSO y el ciclo de juego es ejecutado normalmente con esta nueva pieza y el juego continúa.

APÉNDICE F: USO DE LA LIBRERÍA WIIMOTELIB

La librería WiimoteLib requiere instanciar una clase que define como "Wiimote" y asociar a este objeto métodos para el manejo de los eventos necesarios, como son por ejemplo el evento "WiimoteChanged" que reporta cambios en los valores de estado de los botones del control y el evento "WiimoteExtensionChanged" que detecta si se conecta o desconecta alguno de los periféricos compatibles con el Wiimote como el Nunchuk. También tiene métodos para hacer vibrar el control o establecer el estado de las luces indicadoras del control.

Para iniciar la comunicación es necesaria una llamada al método "Connect" de la instancia de la clase Wiimote; e indicar el tipo de reporte que necesita la aplicación que esté usando la librería, siendo en este caso el que involucra los botones del control, el acelerómetro integrado y los valores de la extensión conectada (Nunchuk). Al terminar de correr la aplicación debe hacerse una llamada al método "Disconnect" para liberar los recursos reservados por el enlace Bluetooth con el control.

Un ejemplo en código es el siguiente:

```
classInterfazWiimote
    {
Wiimote wm = newWiimote();
        publicbool conectar()
        {
try
            {
                wm.Connect();
if ((wm.WiimoteState.Extension) && (wm.WiimoteState.ExtensionType ==
ExtensionType.Nunchuk))
                {
                    wm.SetReportType(InputReport.ExtensionAccel, true);
                    wm.SetLEDs(true, false, false, true);
                }
else
                {
                    wm.SetReportType(InputReport.ButtonsAccel, true);
                    wm.SetLEDs(false, true, true, false);
                }
returntrue;
            }
catch (WiimoteNotFoundException e)
            {
```

```
//LogManager.Singleton.LogMessage("-------WiimoteNotFound------------>" +
e.Message);
returnfalse;
            }
        }


publicvoid desconectar()
        {
            wm.Disconnect();
}
    }
```

APÉNDICE G: TRIGONOMETRÍA APLICADA PARA OBTENER EL CUADRANTE DONDE SE ENCUENTRA LA CÁMARA

Se utilizó un artefacto trigonométrico de la siguiente manera:

Como la cámara se mueve sobre una esfera, la proyección de esta última sobre el plano (X,Z) en el que se encuentra el piso del área de juego describe una circunferencia. El radio de la circunferencia es conocido, así como también el punto sobre el que se proyecta la cámara en esa circunferencia. Por lo tanto un triángulo puede ser descrito por el radio de la circunferencia medido desde el punto (0,0) del plano trazado sobre Z=0, la distancia sobre el eje X de la cámara al punto (0,0) del plano y la distancia de la proyección (X,Z) de la posición de la cámara al punto (0,0) del plano. Con esos tres valores se calcula el ángulo α descrito entre Z=0 y el punto de la proyección de la posición de la cámara, con lo que se determinan cuatro rangos de la medida de α en un momento determinado, definiendo los cuadrantes del área de juego con lo que se puede establecer un estado de la cámara que determina el comportamiento de la asignación de los comandos de control de la pieza en juego.

Un ejemplo de código es el siguiente:

```
void actualizaEstadoCamaraControles()
        {
float catetoAdyacente;
float catetoOpuesto;
float hipotenusa;
float coseno;
float angulo;

catetoAdyacente = cam.Position.z;
            catetoOpuesto = cam.Position.x;
            hipotenusa = Mogre.Math.Sqrt(Mogre.Math.Sqr(cam.Position.z) +
Mogre.Math.Sqr(cam.Position.x));
            coseno = catetoAdyacente / hipotenusa;
if (catetoOpuesto >= 0)
            {
                angulo = Mogre.Math.ACos(coseno).ValueDegrees;
            }
```

```
else
            {
                angulo =
Mogre.Math.Abs(Mogre.Math.ACos(coseno).ValueDegrees - 180) + 180;
}

//LogManager.Singleton.LogMessage("Angulo: " + angulo);

if ((angulo >= 315) || (angulo < 45))
            {
                estadoCamaraControles = 1;
            }
elseif ((angulo >= 45) && (angulo < 135))
            {
                estadoCamaraControles = 2;
            }
elseif ((angulo >= 135) && (angulo < 225))
            {
                estadoCamaraControles = 3;
            }
elseif ((angulo >= 225) && (angulo < 315))
            {
                estadoCamaraControles = 4;
            }

//LogManager.Singleton.LogMessage("Estado Camara Controles: " +
estadoCamaraControles);
        }
```

APÉNDICE H: FOTOGRAFÍAS DEL USO DEL WIIMOTE PARA LA ROTACIÓN DE LAS PIEZAS

Rotación sobre el eje Z de la pieza en forma de Z:

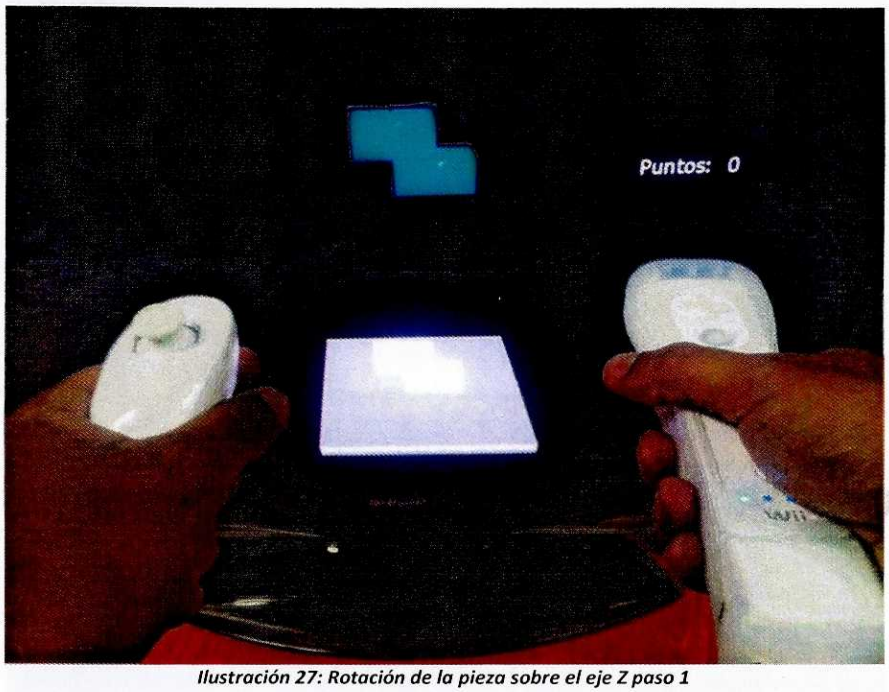

Ilustración 27: Rotación de la pieza sobre el eje Z paso 1
[Fuente: Elaboración Propia]

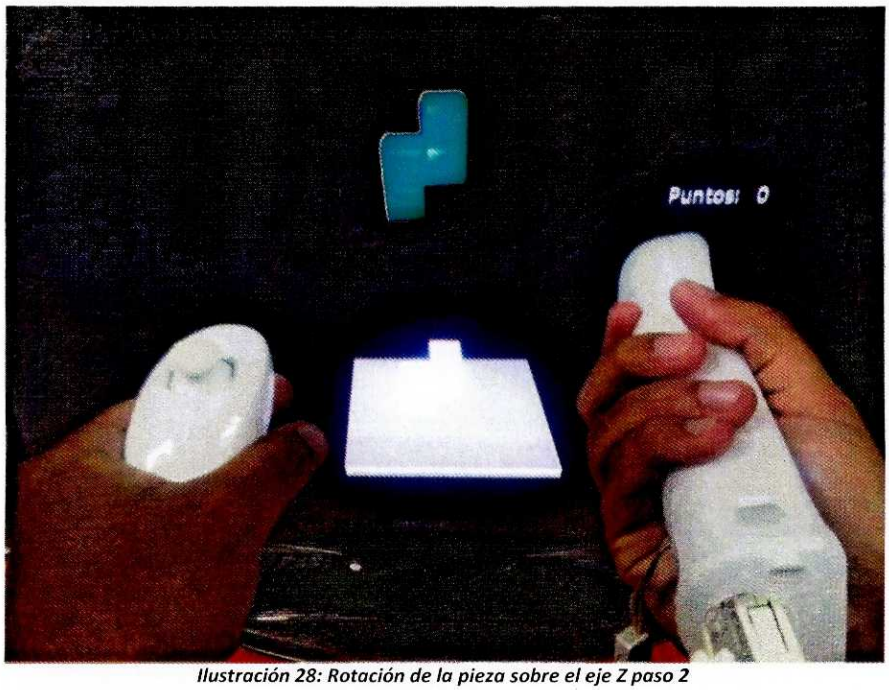

Ilustración 28: Rotación de la pieza sobre el eje Z paso 2
[Fuente: Elaboración Propia]

Rotación sobre el eje X de la pieza en forma de L:

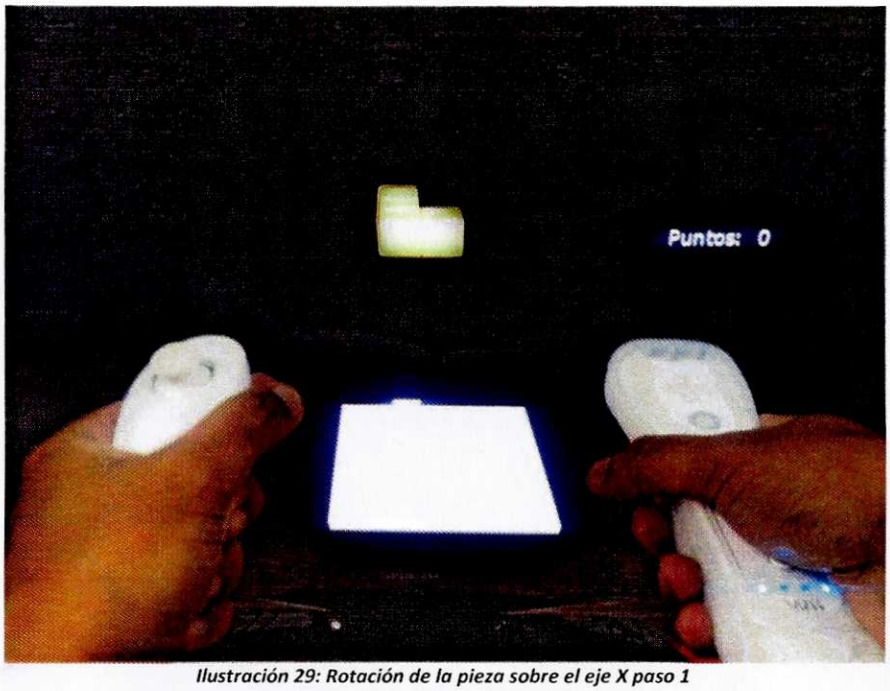

Ilustración 29: Rotación de la pieza sobre el eje X paso 1
[Fuente: Elaboración Propia]

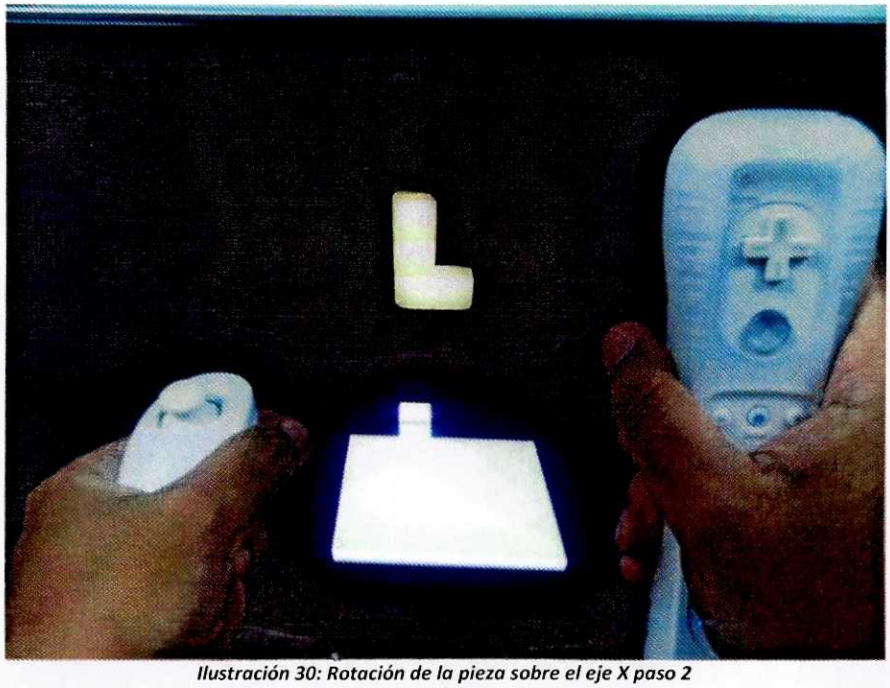

Ilustración 30: Rotación de la pieza sobre el eje X paso 2
[Fuente: Elaboración Propia]

APÉNDICE I: FOTOGRAFÍAS DEL USO DEL NUNCHUK PARA EL MOVIMIENTO DE LA CÁMARA

Movimiento horizontal de la cámara:

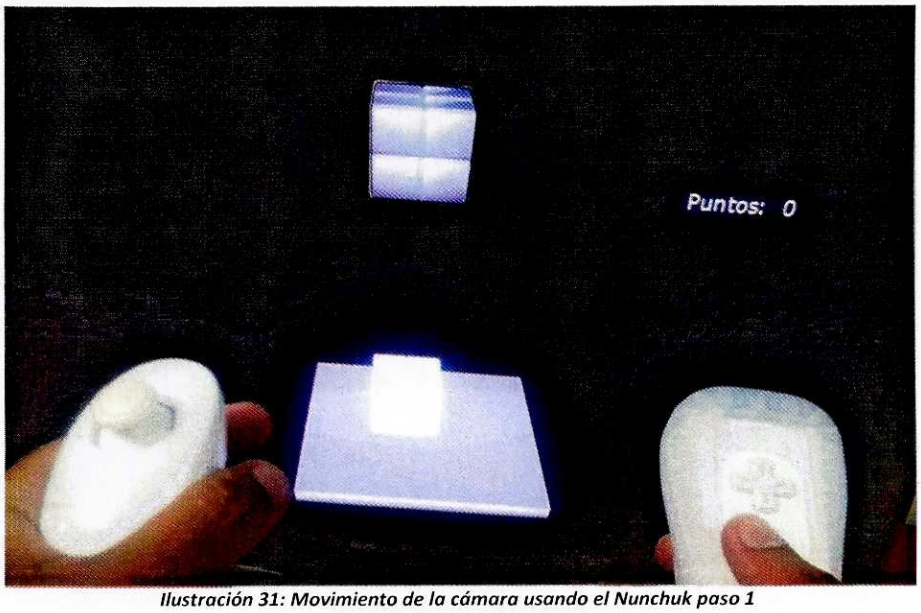

Ilustración 31: Movimiento de la cámara usando el Nunchuk paso 1
[Fuente: Elaboración Propia]

Ilustración 32: Movimiento de la cámara usando el Nunchuk paso 2
[Fuente: Elaboración Propia]

Movimiento vertical de la cámara:

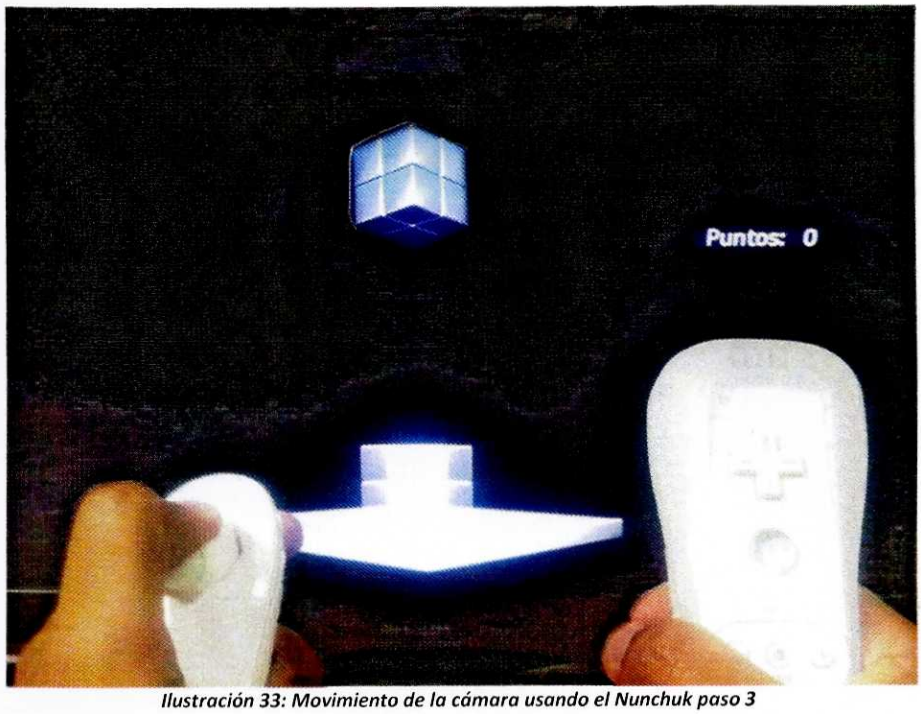

Ilustración 33: Movimiento de la cámara usando el Nunchuk paso 3
[Fuente: Elaboración Propia]