INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

E S C O M

Trabajo de titulación

“Juego de pelota Virtual en Red (Pok-Ta-Pok)”

Que para cumplir con la opción de titulación curricular en la carrera de:

“Ingeniería en Sistemas Computacionales”

Presentan

José Guadalupe Medina Martínez
Jesús Adrián Montaño Ordaz

Directores o asesores

M. en C. Martha Rosa Cordero López
M. en C. Marco Antonio Dorantes González

ESCOM

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

SUBDIRECCIÓN ACADEMICA

Registro de titulación: ISCCR0106-2008-0045/2015 Fecha: 9 de Septiembre de 2015

Documento técnico

“Juego de pelota Virtual en Red (Pok-Ta-Pok)”

Presentan

José Guadalupe Medina Martínez[1]
Jesús Adrián Montaño Ordaz[2]

Directores o asesores:

M. en C. Martha Rosa Cordero López.
M. en C. Marco Antonio Dorantes González.

Resumen:

El presente documento describe el desarrollo de un videojuego que recrea las partidas del Juego de Pelota al estilo de los mayas en tercera dimensión, haciendo uso de algoritmos de inteligencia artificial y física de colisiones para lograr el mayor realismo posible en red local.

ADVERTENCIA

"Este documento contiene información desarrollada por la Escuela Superior de Cómputo del Instituto Politécnico Nacional, a partir de datos y documentos con derecho de propiedad y por lo tanto, su uso quedará restringido a las aplicaciones que explícitamente se convengan."

La aplicación no convenida exime a la escuela su responsabilidad técnica y da lugar a las consecuencias legales que para tal efecto se determinen.

Información adicional sobre este reporte técnico podrá obtenerse en:

La Subdirección Académica de la Escuela Superior de Cómputo del Instituto Politécnico Nacional, situada en Av. Juan de Dios Bátiz s/n Teléfono: 57296000, extensión 52000.

AGRADECIMIENTOS

Luperrock! || Ra(0_o)

A Google
Por haber estado en todo momento a mi lado, cuando más lo necesitaba
Por ése motor de búsqueda tan eficiente
Por la cantidad de energía que sus servidores gastaron en mis continuas búsquedas fallidas
Por permitirme crecer por mí mismo y aprender por cuenta propia

A Wikipedia
Por sus definiciones claras y concisas (aún con fuentes poco claras)
Por explicar con peras y manzanas términos difíciles de entender
Por todo el ancho de banda proporcionado para descarga de imágenes e información, nunca te lo podré pagar, gracias!

A MSDN y XNA Creators Club
Por todo el conocimiento proporcionado entorno a la API de C# y de XNA 3.0
Por los ejemplos siempre disponibles (salvo el contenido Premium) para aprender nuevas técnicas en XNA.

A todos los miembros del Creators Club y los geeks y gamers del mundo cuyos posts, tutoriales y wikis aclararon mi mente en los momentos más turbios.

Y a todas aquellas páginas que me faltó mencionar, pero saben que en algún momento mi IP pasó por sus servidores, gracias a todos ustedes!!

Con agradecimiento dedicamos también la presente
a nuestros maestros y asesores por el apoyo
brindado durante estos años de estudio y
como un reconocimiento de gratitud
al haber finalizado la carrera

Índice

INTRODUCCIÓN

Los continuos avances tecnológicos impulsados en gran medida por la presencia de las computadoras en la vida diaria de las personas ha permitido el surgimiento de industrias multimillonarias dedicadas al entretenimiento y/o comunicación social, encontramos entre éstas : la música, industria cinematográfica, de redes sociales, tecnología de información y videojuegos, es precisamente ésta última una de las más importantes, pues su crecimiento a últimos años ha comenzado a desplazar a la industria del cine; según estudios[1] , la industria de videojuegos representó un valor de 27.000 millones de dólares mundialmente en el año 2005, desplazando incluso a la industria cinematográfica.

México, por su parte, ha tenido escasa (casi nula) participación en ésta importante rama del entretenimiento, lo que origina el interés en el desarrollo de éste tipo de tecnologías, continuando una línea de desarrollo de videojuegos en ESCOM y buscando solidificar los esfuerzos aislados que permitan fortalecer la industria del videojuego en México.

Además, se ha detectado un reducido conocimiento acerca del pasado prehispánico de México a nivel nacional e internacional, es debido a lo anterior que se ha propuesto el desarrollo del presente trabajo, el proyecto “Juego de Pelota Virtual en Red” (Pok-Ta-Pok).

Dicho proyecto permitirá recrear partidas del Juego de Pelota al estilo maya en 3 dimensiones usando algunas de las técnicas más comunes en el desarrollo de videojuegos actual, las cuales incluyen: motores de física, algoritmos de inteligencia artificial, efectos y programación sobre el procesador gráfico, modelado en tres dimensiones y manejo de red.

Debido a la cantidad de áreas del conocimiento que se requieren en el desarrollo de éste tipo de sistemas, el presente proyecto permitirá ampliar y profundizar los conocimientos en la implementación de estos sistemas a nivel profesional, algunos de ellos aprendidos en nuestra estancia en ESCOM y otros tantos producto de investigación más específica en libros y recursos digitales.

OBJETIVO

Desarrollar una aplicación funcional, llamativa y entretenida que permita recrear las partidas del “Juego de Pelota” en una interfaz tridimensional con soporte en red local.

CAPITULO 1 – ANTECEDENTES

1.1 Introducción.

La cultura maya y el juego de pelota

La civilización maya habitó una vasta región ubicada geográficamente en el territorio del sur-sureste de México, específicamente en los cinco estados de: Campeche, Chiapas (lugar donde se ubica la ciudad principal), Quintana Roo, Tabasco, Yucatán, además de territorios de los actuales Belice, Guatemala, Honduras y El Salvador.

Se desarrolló entre los siglos III y XV, con una historia de aproximadamente 3.000 años.

Fig. 1- Mapa de localización de la civilización Maya.

Fueron una de las más brillantes y poderosas culturas conocidas de Mesoamérica. Dominaban un lenguaje escrito, eran hábiles arquitectos, arriesgados comerciantes y talentosos artistas. No constituían un estado unificado, sino que se organizaban en varias ciudades-estado independientes entre sí. Eran relativamente pacíficos y vivían organizados por tribus en ciudades y pueblos que se confederaban sin un soberano común que ejerciera el poder.

La tierra era propiedad común, distribuida por el cacique de la tribu. Adoraban divinidades astrales y siderales, a las que ofrecían sacrificios animales y humanos. El arte maya ofrece: monumentales edificios de piedra, imponentes pirámides, templos y palacios recubiertos de elaborados relieves, pinturas murales, esculturas y ricas cerámicas. Vivieron en una sociedad agrícola y poseían un sistema religioso desarrollado que veneraba al cosmos.

RASGOS Y APARIENCIA DE LOS MAYAS

La belleza era importante para los mayas. Entre los aspectos más importantes se encuentran:

- Presionaban las caras de sus hijos jóvenes para que tuvieran algo de atractivo.
- Insertaban objetos colgantes en su cuerpo, otorgaba una apariencia sabia.
- La mayoría eran de tez oscura y usualmente tenían cabello largo y ojos negros.
- Portaban muy poca vestimenta, andaban descalzos o con poca cobertura en los pies.
- Las plumas y otras pieles de animales eran usadas como vestidos y como joyas.
- Los penachos eran usados para ceremonias y eventos especiales.

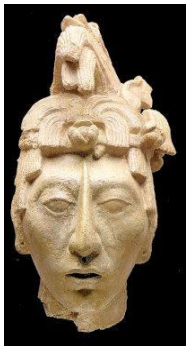


Fig. 2- Rostro Maya.

PRINCIPALES CIUDADES

En las ciudades mayas, los palacios, los templos y las casas principales eran construidos alrededor de las plazas donde se realizaban las ceremonias públicas, como ahora se pueden apreciar en la Plaza Mayor de Tikal. Cerca estaban las residencias de los príncipes y sacerdotes y, más alejadas, según categorías sociales, estaban las viviendas del resto de los habitantes. Las casas de los pobres eran construidas de bajareque (caña y barro) en las afueras de la ciudad.

Actualmente existen cerca de 116 ciudades, entre las más importantes se encuentran:

- Tikal , Uaxactún y Piedras Negras en El Petén; Las de Copán, en Honduras;
- Chichén Itzá, Palenque, Yaxchilan y Mayapán en México.

Consistían en palacios y templos en forma de pirámide escalonada hecha de piedra y estuco (mezcla de cal y arena fina). Las estelas se erigían cada veinte años y en ellas se recogían los acontecimientos importantes ocurridos en ese lapso.

EL CALENDARIO MAYA

Fue el centro de su vida y su mayor logro cultural. Su precisión deriva del hecho de que se basa en una cuenta continua e ininterrumpida de los días (llamados *Kin* en maya) a partir de un día cero inicial, ubicado el 13 de agosto de 3114 a.C.

El conocimiento ancestral del calendario guiaba la existencia de los mayas a partir del momento de su nacimiento y poco escapaba a la influencia calendárica.
Los mayas manejaban 2 diferentes calendarios:

- Calendario de 260 días - Tzolkin
  - El más usado por los pueblos del mundo maya, para regir los tiempos de su quehacer agrícola, su ceremonial religioso y sus costumbres familiares.
- Calendario de 365 días – Haab

- Se basa en el recorrido anual de la Tierra alrededor del Sol en 365 días. Los mayas dividieron el año de 365 días en 18 "meses" llamados *Winal* de 20 días cada uno y 5 días sobrantes que se les denominaba *Wayeb*.

EL CICLO DE 18,980 DÍAS - LA RUEDA CALENDÁRICA

La combinación de los calendarios de 260 y 365 días crea un ciclo mayor de 18,980 días (el mínimo común múltiplo de 260 y 365), a esta combinación se le ha llamado la Rueda Calendárica.

Fig. 3. – Tzolkin: Calendario Maya

EL JUEGO DE PELOTA

El juego de pelota (o en el dialecto original, "Pok-Ta-Pok", llamado así por el ruido que hacia la pelota al rebotar en las paredes y en los jugadores) aparte de ser el deporte representativo de la cultura maya y de toda Mesoamérica, también tenía muchísimo carácter religioso y espiritual.

Características y reglas del juego

El argumento del juego consiste en que los integrantes del equipo luminoso golpearán la pelota con sus caderas, codos o rodillas buscando efectuar jugadas que no pueda responder el equipo contrario, y con ello lograr el triunfo de la luz y el nacimiento del Sol, mientras que el otro equipo buscará el predominio de la oscuridad. Generalmente, el partido concluía cuando alguno de los equipos marcaba el primer gol. En algunos Juegos Rituales, el líder del equipo perdedor, era decapitado y su cráneo se usaba para hacer el centro de una nueva pelota.

Jugadores

El número de jugadores (Pitziil) varía entre 2 y 5 en cada equipo, usaban protección en la cabeza, llamada Pix'om, Caderas (Tz´um) hechas de cuero de venado o jaguar, y en las rodillas y codos, Kipachq'ab'.

Fig.4 – Imagen representativa de los jugadores del Juego de Pelota

La pelota

Era hecha con una mezcla de látex de Hule (Hevea Brazilienzis) y de Guamol (Calonyction aculeatum), su tamaño variaba entre 22 y 25 cm. de diámetro, con un peso de 3 a 6 libras.

La cancha

La Cancha tenía forma de I o doble T, cuyo tamaño variaba, pero en promedio era de 30 m. de largo y 8 m. de ancho. Se anotaba un gol al tocar el marcador, con la pelota, la cual no se podía tocar con las manos o los pies, solo con las partes ya descritas de protección.

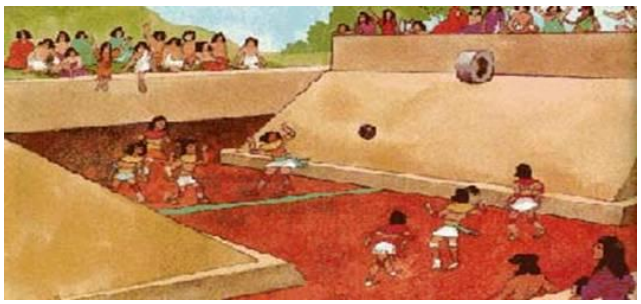


Fig.5 – Representación pictórica moderna del Juego de Pelota

Tenía paredes inclinadas a lo largo de la cancha, con marcadores planos al centro, y la pelota era siempre grande. Si un jugador tocaba la pelota con el pie, el otro equipo se anotaba un gol y mantenía el control de la pelota, había árbitros que se encargaban de hacer cumplir las reglas.

1.2 – Estado Del Arte

Hay algunos proyectos previos que se han desarrollado usando la temática del México Prehispánico y del juego de pelota, entre los que se encuentran los siguientes:

1.- Aplicación

Juego de pelota En Mesoamérica, CD- ROM interactivo [1].

2.- Trabajos Terminales En ESCOM

TT0018: JACK: Video Juego con realidad virtual multidimensional en red [2].
TT0155: Sistema de videojuego de estrategia evolutivo [3].
TT0768: Aztlán, un mundo Persistente en línea [4].

3.-Otros

Arquitectura **BDI** de un Agente Jugador de Fútbol [5]

Hace uso de una base de conocimientos y agentes BDI (Beliefs, desires, intentions).

En la siguiente tabla se muestra una comparación de dichos trabajos previos.

SOFTWARE	**CARACTERÍSTICAS**	**PRECIO EN EL MERCADO**
Juego De Pelota En Mesoamérica, CD-ROM interactivo.	• Traducido a 3 idiomas: inglés francés y alemán • Textos, locución, videos, imágenes, animaciones, aplicaciones interactivas y juegos (en flash)	$207, sujeto a disponibilidad
TT0018 JACK: Video Juego con realidad virtual multidimensional en red	• Uso de realidad virtual • Limitaciones gráficas • Soporte para uso en red	Acceso Gratuito
TT0155: Sistema de videojuego de estrategia evolutivo	• Algoritmos evolutivos	Acceso Gratuito
TT0768: AztlánRPG, un mundo Persistente en línea.	• Persistencia • Gráficos tridimensionales • Uso de elementos fantásticos • Soporte para varios jugadores en red	Acceso Gratuito

1.3 – Problemática

El juego de pelota, muy popular entre los pueblos prehispánicos como los mayas u olmecas, fue parcialmente exterminado con la llegada de los españoles, y por lo tanto la sociedad sabe muy poco acerca de dicho juego.

Las demostraciones que se realizan en la actualidad con jugadores humanos suelen llevarse a cabo en diversos lugares de la ribera Maya, tales como XCaret o Chichen Itzá que forman parte de eventos especiales y en algunos casos con precios poco accesibles por la sociedad.

Existen algunos videos en la red [6] o como parte de documentales que muestran la forma de jugar el "Juego de Pelota" pero su calidad suele ser deficiente o bien, no permiten interactividad.

1.4 – Solución Propuesta

Se propone realizar una aplicación que permita de manera interactiva jugar el "Juego De Pelota", a través de una interfaz en 3 dimensiones, haciendo uso de tecnologías actuales en el desarrollo de videojuegos, modelado y texturización, con soporte de red local para dos jugadores.

El usuario podrá crear su perfil y modificarlo según sea posible, contará con un equipo formado por varios jugadores, dicho equipo será controlado parcialmente por el jugador quien buscará ganar el partido según las reglas básicas del juego, de la misma forma, el jugador contrario (un humano conectado en red controlando su propio equipo) buscará la victoria por su parte.

1.5 – Justificación

El desarrollo de la aplicación propuesta permitirá difundir, de manera entretenida y sencilla, el Juego de Pelota Maya.

Originalidad del trabajo.

En la actualidad, pocas aplicaciones se orientan en dar un panorama acerca de la cultura antigua mexicana, para mejorar ésta situación nosotros hemos propuesto una herramienta llamativa para dar a conocer de manera interactiva el juego de pelota.

Vinculación con los usuarios potenciales.

Debido a la temática del juego de pelota, será de interés para:

- Quien desee conocer acerca de la cultura prehispánica, y en particular, el juego de pelota.
- Personas aficionadas a deportes como el futbol
- Recurso educativo para que niños o personas en edad escolar aprendan de manera más fácil el Juego de Pelota.

Mejora a lo ya existente.

Actualmente las recreaciones que se hacen del juego de pelota son principalmente de carácter real, es decir, es necesario unirse a un equipo, entrenar y adquirir condición para jugar el juego de pelota, lo cual implicar mayor consumo de tiempo y recursos para adentrarse en el tema, en el campo de la informática, se cuenta con suposiciones breves como imágenes o animaciones que representan una visión básica del juego. Este sistema hará uso de tecnologías actuales,

1.6 – Objetivos

1.6.1 – Objetivo General

Desarrollar una aplicación funcional, llamativa y entretenida que permita recrear las partidas del “Juego de Pelota” en una interfaz tridimensional con soporte en red local.

1.6.2 – Objetivos Específicos

- Obtener información histórica acerca del juego de pelota, reglas, modos de juego, medidas de campos, civilizaciones participantes, simbología.
- Diseñar y modelar personajes en torno a la temática, así como diseñar escenarios para llevar a cabo los “partidos” del juego de pelota.
- Analizar las diferentes herramientas usadas en el desarrollo de éste tipo de aplicaciones para implementarlo.
- Desarrollar e implementar el juego de Pelota Virtual.

CAPITULO 2 – ANÁLISIS

El objetivo de éste capítulo es identificar los elementos necesarios para el desarrollo del sistema, su viabilidad y posibles complicaciones que podamos encontrar a lo largo del proceso.

2.1 –Descripción De Elementos Del Sistema

Previo al análisis que se requiere, se definen algunos elementos necesarios para la creación del sistema que permiten aclarar los términos usados.

Perfil

Cada usuario también llamado "Jugador" podrá tener un perfil, con la finalidad de personalizar su experiencia al usar el sistema, dicho perfil contendrá un nombre de usuario, el equipo que conforma su perfil y el "logo" del equipo (almacenado en archivo XML).

Equipo

Conjunto de personajes que interactúan entre sí identificados por pertenecer al mismo perfil.

XML

Metalenguaje de etiquetas, estándar para el intercambio de información estructurada entre diferentes plataformas

Preferencias

Conjunto de valores que definen la forma en que se ejecutará el sistema, tales como preferencias de volumen y resolución (almacenadas en archivo XML).

Partida

Juego con valores determinados (tiempo y jugadores máximos, preferencias), definidos previamente a su creación para poder iniciar el juego, en los juegos en red, el concepto de partida está ligado con el de servidor, ya que por lo general un servidor crea la partida, mientras que los clientes se conectan a ella.

Lobby

Interfaz que permite observar la(s) partida(s) en la red y poder conectarse a una de ellas.

2.2 – Análisis De Requerimientos

En ésta sección se pretende identificar los requerimientos tanto funcionales como no funcionales para poder evaluar las acciones que el sistema debe llevar a cabo para su funcionamiento.

2.2.1 - Requerimientos Funcionales

RF 1 – Crear Perfil
- RF1.1 – Ingresar datos de perfil
- RF1.2 – Leer índice de usuarios
- RF1.3 – Comprobar disponibilidad de nombre
- RF1.4 – Almacenar datos de perfil en archivo XML
- RF1.5 – Agregar perfil a índice de perfiles

RF 2 – Modificar Perfil
- RF2.1 – Elegir perfil
- RF2.2 – Ingresar nuevos datos de perfil
- RF2.3 – Almacenar nuevos datos de perfil en archivo XML

RF 3 – Borrar Perfil
- RF3.1 – Confirmar eliminación de perfil
- RF3.2 – Eliminar datos de usuario en archivo XML
- RF3.3 – Eliminar perfil del índice de perfiles

RF4 - Configurar Equipo
- RF4.1 – Elegir Personajes
- RF4.2 – Elegir logo de equipo
- RF4.3 – Guardar configuración en perfil

RF5 – Crear Partida
- RF5.1 – Configurar parámetros de Partida
- RF5.2 – Crear sesión de red local (network session)
- RF5.3 – Manejar Lobby en espera de otro jugador
- RF5.4 – Atender evento de conexión de otro jugador
- RF5.5 – Crear objetos para cada jugador (local y remoto)
- RF5.5 – Esperar a que ambos jugadores estén listos
- RF5.6 – Iniciar juego.

RF6 – Unirse a una partida
- RF6.1 – Buscar sesiones creadas en el lobby (red local)
- RF6.2 – Conectar (unirse) a una sesión

RF7 - Configurar Preferencias
- RF7.1 – Fijar resolución de pantalla
- RF7.2 – Fijar volumen de música
- RF7.3 – Fijar volumen de efectos de sonido
- RF7.4 – Almacenar nueva configuración de video en archivo XML

RF8 – Inicializar Juego
- RF8.1 –Cargar texturas
- RF8.2 – Cargar modelos
- RF8.3 – Cargar audio

RF9 – Jugar
- RF9.1 – Manejar interfaces de entrada como mouse y teclado
- RF9.2 – Cargar modelos
- RF9.3 – Asignar texturas a modelos
- RF9.4 – Animar modelos
- RF9.5 – Enviar paquetes con estado del juego
- RF9.6 – Recibir paquetes del otro jugador
- RF9.7 – Actualizar estado del juego
- RF9.8 – Realizar análisis de la física necesaria
- RF9.9 – Realizar acción respecto al estado del juego.

RF10 – Cargar Preferencias
- RF10.1 – Leer Preferencias de archivo XML
- RF10.2 – Asignar Preferencias al juego

RF11 - Cargar Perfil
- RF11.1 – Dado un perfil leído, asignar dicho perfil al perfil general del juego.

RF12 – Leer Perfiles
- RF12.1 – Obtener los datos del perfil para posteriormente ser usado por otro casos de uso

2.2.2–Requerimientos No Funcionales

Estabilidad
- Almacenar estadísticas del jugador:
  - Juegos ganados, perdidos y totales.
  - Nivel
  - Score
- Manejar interrupciones de conexión o salida abrupta de jugadores

Rendimiento
- Reducir la cantidad de polígonos en pantalla en los modelos usados
- Mantener una tasa de cuadros por segundo que permita hacer agradable el uso del sistema.
- Optimizar los paquetes enviados para no saturar la red.
- Agilizar la búsqueda de acciones al momento de elegir.

Documentación
- Contar con manuales de usuario que permitan comprender el uso del sistema.
- Contar con manuales técnicos que especifiquen la instalación del sistema.

Extensibilidad
- Permitir la presencia de varios jugadores en el lobby

Portabilidad
- Poder usar el sistema en computadoras que cumplan con los requisitos mínimos.

Otros
- Basarse en la cultura Maya para el uso de edificios, personajes, música y demás elementos conceptuales del sistema
- Manejar interfaces gráficas claras y entendibles en español para el máximo aprovechamientos del sistema.

2.3 – Análisis Técnico

La creación de un sistema como el propuesto supone el empleo de diversos elementos que confirman el entorno general del mismo, dichos elementos incluyen los siguientes, se muestran las opciones consideradas y elegidas, salvo que se indique que aún están por definir:

- Modelado
  - Blender
  - 3D Studio Max
- Texturización
  - Gimp
  - Photoshop
- Animación
  - Blender
  - 3D Studio Max
  - XNA (Propia y a través de herramientas auxiliares, dlls)
- Interfaces de entradas y salidas
  - XNA
- Lógica del juego. (Por definir)
  - XNA (Lógica básica)
  - Motores de inferencia
  - Árboles de decisión
- Motores de Física (Por definir)
  - XNA (colisiones básicas)
  - PhysX
  - Havok
  - Oops
  - Bullet
- Conexiones de red.
  - XNA - Envío y recepción de paquetes en una red local

2.3.1 – Motores De Videojuegos Y Visualización

Permiten integrar diversos elementos como modelos 3D, pantallas de usuario, audio, soporte de red, colisiones, interfaces de entrada/salida, en un solo programa (generalmente ejecutable) logrando con ello un sistema funcional.

2.3.1.1 - Xna

Sus siglas significan : XNA's Not Acronymed, es una API desarrollada por Microsoft para el desarrollo de videojuegos para las plataformas Xbox 360, Zune y Windows
Es un Marco de Trabajo (Framework), basado en .NET Framework 2.0 corre sobre el CLR (Common Languaje Routine), provee un manejo optimizado para la ejecución de videojuegos.

Para el desarrollo sobre este Marco de Trabajo (Framework) se usa el Microsoft GameStudio el cual usa como único lenguaje C# pero debido a la naturaleza de la filosofía de desarrollo .NET que mediante el CLR permite que todos ellos compilen a código nativo usando un mismo código intermedio (MSIL). Este Marco de Trabajo(Framework) sirve para hacer conexiones con el DirectX 9.0c en tiempo de ejecución justo después de la compilación JIT.

A continuación se muestra un diagrama con las capas que conforman XNA

Fig.6 – Capas que conforman XNA

Proceso De Creación De Gráficos 3d

El proceso de creación de gráficos 3D por computadora puede ser dividido en estas tres fases básicas:

- Modelado - Dar forma a objetos individuales que luego serán usados en la escena.
- Composición de la escena - distribución de objetos, luces, cámaras y otras entidades
- Render - Generar la imagen 2D o animación a partir de la escena creada.

Éstos 2 últimos elementos son realizados directamente por XNA, mientras que el modelado requiere de otras herramientas.

2.3.2 – Modelado

2.3.2.1 - 3d Studio Max

Software de modelado 3D, desarrollado por *Autodesk Media & Entertainment* como sucesor de 3D Studio para DOS, pero para la plataforma Win32. Fue creado por *Yost Group* y publicado por *Autodesk 3D Studio Max*.

Es uno de los programas de animación 3D mas usados. Tiene gran capacidad de modelado y una potente arquitectura de plugins. Usado ampliamente por desarrolladores de videojuegos, películas, efectos especiales y presentaciones arquitectónicas.

Incluye *shaders*, simulación dinámica, sistema de partículas, radiosidad, iluminación global, una interfaz intuitiva y personalizable, su propio lenguaje de Script (MaxScript) y un motor físico creado originalmente por Havok Reactor.

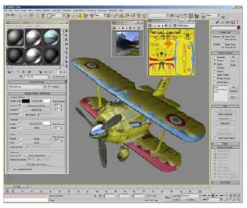


Fig. 7 - Captura del programa 3D Studio Max

2.3.2.2 - Blender

Programa de modelado gratuito, actualmente compatible con diversas versiones de Windows, Mac OS X, Linux, Solaris, FreeBSD e IRIX.

Entre sus características se encuentran:

- Multiplataforma, libre, gratuito, formatos gráficos como TGA, JPG, Iris, SGI, o TIFF. .
- Primitivas geométricas, curvas, mallas poligonales, vacíos, NURBS.
- Detección de colisiones, recreaciones dinámicas y lógica.
- Lenguaje Python para automatizar o controlar varias tareas.
- Simulaciones dinámicas para softbodies, partículas y fluidos.

2.3.3 - Texturización

2.3.3.1 - Gimp (Gnu Image Manipulation Program.)

Es un programa de tratamiento de imágenes libre. Sirve para procesar gráficos y fotografías digitales. Los usos típicos incluyen la creación de gráficos y logos, el cambio de tamaño, recorte y modificación de fotografías digitales, la modificación de colores, la combinación de imágenes usando un paradigma de capas, la eliminación o alteración de elementos no deseados en imágenes o la conversión entre distintos formatos de imágenes.

Gimp tiene múltiples características, algunas de ellas

- Trabaja con capas (layers).
- Soporte de plug-ins.
- Reconoce múltiples formatos de imágenes (PNG, JPEG, GIF, TIFF, MNG, BMP, PSD).
- Puede usarse tanto en sistemas Unix como en los Windows de Microsoft.

2.3.3.2 - Photoshop

Aplicación que trabaja sobre un *"lienzo"*, está destinado para la edición, retoque fotográfico y pintura a base de imágenes de mapa de bits o conocidos en Photoshop como gráficos rasterizados, elaborado por la compañía de software Adobe Systems inicialmente para computadores Apple posteriormente también para plataformas PC con Windows.

Se ha convertido, casi desde sus comienzos, en el estándar *de facto* en retoque fotográfico, también se usa extensivamente en disciplinas del campo del diseño y fotografía, como diseño web, composición de imágenes bitmap, estilismo digital, fotocomposición, edición.
Soporta muchos tipos de archivos de imágenes, como BMP, JPG, PNG, GIF, entre otros, pero tiene ciertos formatos de imagen propios como: P**SD** (Photoshop Document).

2.4 - Análisis De Riesgo

Su objetivo es identificar los riesgos: su probabilidad de ocurrir (%) e impacto en el proyecto.

IMPACTO	VALOR	DESCRIPCIÓN
Catastrófico	4	Provocaría fracaso de la misión.
Critico	3	Degradaría el rendimiento del sistema hasta un punto donde el éxito de la misión es cuestionable.
Marginal	2	Provocaría la degradación de la misión secundaria.
Despreciable	1	Provocaría inconvenientes o impactos no operativos.

TIPOS DE RIESGO	ID	EFECTO
RIESGOS DEL PROYECTO	P	Amenazan el plan del proyecto, si se presentan, es probable que la planeación se atrase y los costos aumenten
RIESGOS TECNICOS	T	Amenazan la calidad y la planeación temporal, si se presentan, la implementación puede ser difícil o imposible
RIESGOS DEL NEGOCIO	N	Amenazan la viabilidad del software a construir. Ponen en peligro al proyecto o al producto.

2.4.1 - Clasificación De Riesgos

RIESGOS DE PROYECTO	ID	%	Impacto
Falta de presupuesto	P1	5%	1
Retraso en el diseño	P2	10%	2
Retraso en la codificación	P3	20%	2
Retraso en creación de prototipos	P4	20%	2
Cambios en el equipo de desarrollo	P5	5%	4
Falta de coordinación en equipo de desarrollo	P6	5%	3
Arte insuficiente	P7	10%	1
Manuales, tutoriales y documentación escasa sobre herramientas	P8	5%	2
Falta de equipo (hardware) para desarrollo	P9	5%	4
Información histórica escasa o difusa	P10	10%	2
Cambio en los requerimientos	P11	10%	3
Mala delimitacion de alcances	P12	10%	3

RIESGOS TECNICOS	ID	%	Impacto
Herramientas incompatibles	T1	20	2
Versiones atrasadas o descontinuadas	T2	10	1
Desconocimiento de las herramientas usadas	T3	25	2
Retraso en el Modelado	T4	15	1

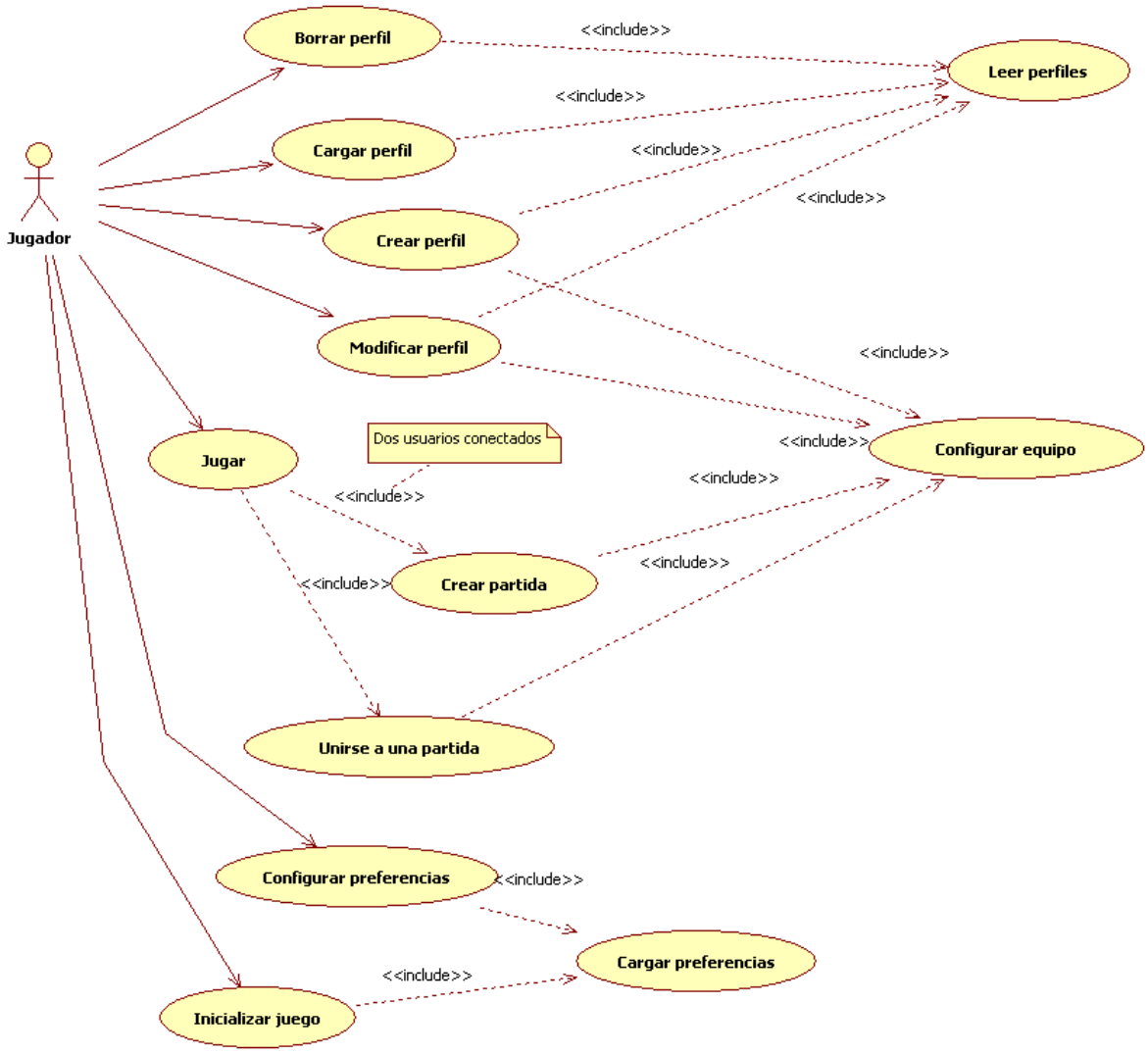
Fallas en la implementación de física adecuada	T5	40	4
Fallas en la implementación de comunicación en red local	T6	25	4
Fallas en la implementación de inteligencia artificial	T7	40	4
El sistema exige demasiados recursos	T8	15	3
Problemas de compatibilidad de prototipos	T9	15	3
Creación de modelos y animaciones inexactos	T10	15	2
Errores de diseño	T11	15	3

CAPITULO 3 – DISEÑO

Haciendo uso del lenguaje UML se han generado los diagramas de: Casos de uso, paquetes, clases, secuencia y estados, los cuales nos permites describir con mayor detalle el sistema.

3.1 – Diagrama De Casos De Uso General :: Pok-Ta-Pok

Muestra una vista general de las funciones que se podrán realizar en el sistema, así como la participación del actor dentro del mismo.



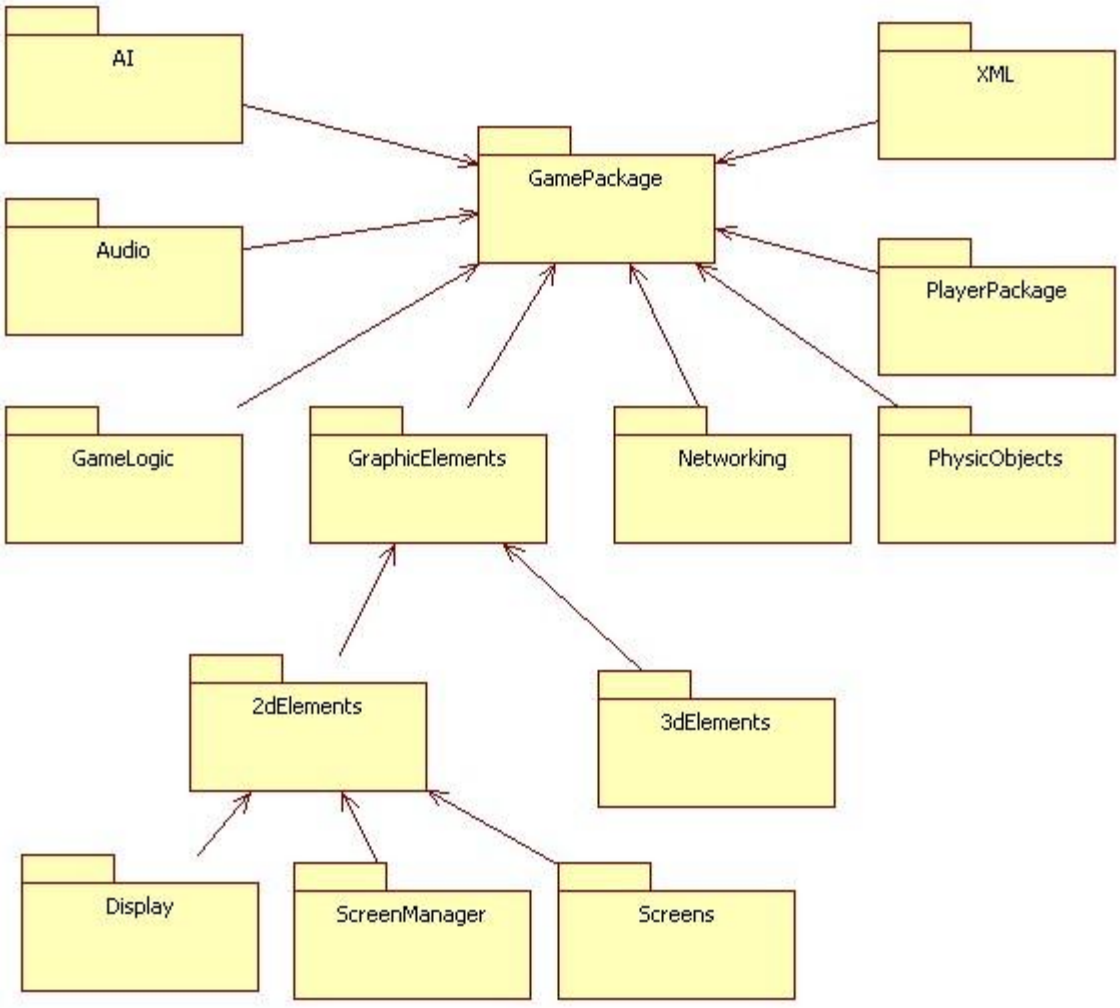
3.1.1 – Descripción De Diagrama De Casos De Uso General

Caso de Uso	**Configurar equipo**
Actores	Jugador
Tipo	Inclusión
Descripción	Permite al jugador seleccionar los personajes que conforman su equipo, así como cambiarlos en caso de ser necesario
Caso de Uso	**Crear Perfil**
Actores	Jugador
Tipo	Base
Descripción	Solicita el nombre del perfil que habrá de crearse, para posteriormente poder referirse a él, la creación de un perfil permite customizar o personalizar las preferencias del jugador en el juego, además de llevar un registro de sus scores, el perfil será almacenado en un archivo XML.
Caso de Uso	**Borrar Perfil**
Actores	Jugador
Tipo	Base
Descripción	Elimina uno de los perfiles creados previamente, lo cual borra sus preferencias y puntajes previos
Caso de Uso	**Modificar Perfil**
Actores	Jugador
Tipo	Base
Descripción	Cambia los valores previamente establecidos de un perfil determinado, sus preferencias y configuraciones.
Caso de Uso	**Cargar Perfil**
Actores	Jugador
Tipo	Base
Descripción	Carga un perfil de los ya existentes según la decisión del usuario, los perfiles son almacenados en archivos XML, se contará con un índice de perfiles para poder accesar a ellos de manera más eficaz
Caso de Uso	**Crear Partida**
Actores	Jugador
Tipo	Inclusión
Descripción	Crea una partida con la máquina como "host", lo cual nos permite actuar como servidor, para ello, se espera una conexión desde un cliente quien tendrá su propio equipo en una máquina diferente y con quien se comunicará a través de la red con el equipo propio
Caso de Uso	**Unirse a una partida**

Actores	Jugador
Tipo	Inclusión
Descripción	Crea una partida con la máquina como "cliente", para ello, deberá conectarse a otra máquina en la red, que fungirá como servidor
Caso de Uso	**Jugar**
Actores	Jugador
Tipo	Base
Descripción	Una vez que se ha elegido el perfil y que se tiene una partida definida (ya sea creada o unido a una en la red) el jugador puede jugar e interactuar con su equipo contra el equipo del rival
Caso de Uso	**Configurar Preferencias**
Actores	Jugador
Tipo	Base
Descripción	Define y fija los valores deseados de video (como resolución o modo de pantalla) con los que se desplegará el juego; y audio (volumen de música y efectos de sonido que serán reproducidos durante el juego), los valores fijados serán almacenados en un archivo XML
Caso de Uso	**Cargar Preferencias**
Actores	Jugador
Tipo	Inclusión
Descripción	Carga las preferencias previamente almacenadas en un archivo XML, dichas preferencias se refieren al audio y video a desplegar en el sistema
Caso de Uso	**Inicializar Juego**
Actores	Jugador
Tipo	Base
Descripción	Carga los elementos a ser mostrados durante la ejecución de la aplicación, por ejemplo, modelos, audio, texturas, sprites
Caso de Uso	**Leer Perfiles**
Actores	Jugador
Tipo	Inclusión
Descripción	Accede a la información relacionada con los perfiles del juego, para poder ser usada por algún otro caso de uso.

3.2 – Diagrama De Paquetes :: Pok-Ta-Pok

Se decidió generar un diagrama de paquetes debido a la cantidad de clases usadas, esto nos permite tener mayor control sobre las clases y reducir la complejidad de los diagramas. Los diagramas de clases correspondientes a cada paquete son mostrados en el documento Anexo.



3.3 - Metodología

Debido a la naturaleza del proyecto, el modelo seleccionado para el desarrollo del mismo fue el Modelo en espiral propuesto por Boehm, éste es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos. Proporciona el potencial para el desarrollo rápido de versiones incrementales de Software.

Se basa en un enfoque heurístico hacia el desarrollo de sistemas y de software a gran escala, como el software evoluciona a medida que progresa el plan, el desarrollador y el cliente comprenden y reaccionan mejor ante riesgos en cada uno de los niveles evolutivos. Utiliza la construcción de prototipos como mecanismo de reducción de riesgos, permite aplicar el enfoque de construcción de prototipo en cualquier etapa de evolución del proyecto.

Para cada ciclo habrá cuatro actividades:

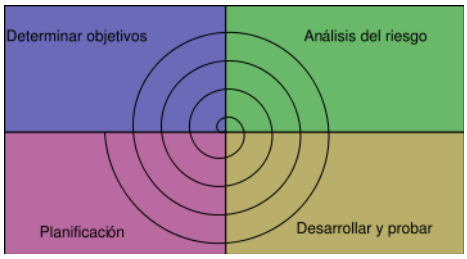


Fig.8 – Estructura básica del modelo en espiral de Boehm

3.3.1 - Primer Ciclo

3.3.1.1 - Determinar O Fijar Objetivos

Analizar las herramientas necesarias para el desarrollo del proyecto, tales como: modelado, manejo de audio, así como su integración con el framework para desarrollo del juego.

- **Productos a obtener.**
  1. Especificación sobre las herramientas usadas para el desarrollo en lo respectivo a modelado, texturizado, y framework para el desarrollo del sistema.
  2. Tech Demo 1.0 implementando audio, texturizado, animación, cámara, iluminación básica y manejo de controles usando las herramientas seleccionadas.
- **Restricciones**
  1. Limitarse a una integración básica de los componentes mencionados, para verificar su correcta integración en la herramienta seleccionada.
- **Riesgos**
  1. Posibles problemas de compatibilidad entre herramientas
  2. Errores en la exportación e importación de elementos.
  3. Cambio de versiones a lo largo del proceso de desarrollo.

4. Obtención de licencias para los productos seleccionados o bien, búsqueda de licencias estudiantiles para el desarrollo del sistema.

3.3.1.2 - Análisis Del Riesgo

Posibles problemas de compatibilidad entre herramientas

Aún cuando las herramientas seleccionadas manejan formatos estándar, no todos ellos se integran de manera óptima, en ocasiones es necesario crear importadores o exportadores propios, conseguir plugins o aditamentos respectivos a cada versión.

Errores en la exportación e importación de elementos.

Modelado

Los 2 programas considerados para realizar modelado (Blender 2.47) y 3DS Max 8 exportan sus modelos (incluyendo huesos, efectos y animaciones) a varios formatos, para el desarrollo del sistema, se optó por utilizar XNA el cual maneja de manera nativa 2 formatos para modelos en 3 dimensiones .fbx y .x, nos centraremos en el uso de formatos .fbx, el cual es un formato soportado por ambos programas.

Sonido

Para el uso de sonido, XNA tiene algunas herramientas propias (XACT – Audio Creation Tool) para generar "Proyectos de audio" que podrán ser utilizadas posteriormente en XNA, soporta los formatos de sonido wav, aif, aiff, siendo el formato wav el más común, por lo que en el Tech Demo 1.0 se utilizará dicho formato para los sonidos utilizados.

Texturizado

Un problema común es encontrar texturas cuyos tamaños no son potencia de 2 y por lo tanto, ciertas tarjetas de video podrían tener problemas al momento de renderizarlas, de la misma forma, si la imagen es demasiado grande, podría no ser soportada por ciertas tarjetas de video, para ello, en el Tech Demo 1.0 se están utilizando texturas de los siguientes tamaños: 256x256, 512x256 y 128x128.

Interfaces

Un punto importante del sistema son las interfaces, las cuales deben ser llamativas, y fáciles de entender por cualquier usuario; existen varias opciones para el manejo de interfaces gráficas en XNA, entre ellas encontramos : utilizar animaciones hechas en Flash, importarlas y generar interacción entre ellas; crear un conjunto de imágenes y hacerlas funcionar como botones utilizando comparaciones de posición de mouse respecto a un espacio cuadrado que lo limita, y una opción más es utilizar Microsoft Expression Blend, una nueva herramienta de Microsoft que nos permite generar interfaces visualmente más llamativas con total integración de código en C#.

El principal riesgo de utilizar Flash es que la herramienta que permite utilizarlo en C# (Fluix) no se integra correctamente en la versión 2.0 de Microsoft XNA GameStudio, ya que la

modificación del Content Pipeline no nos ha permitido agregar las librerías requeridas. Buscar una forma de solucionarlo podría requerir más tiempo del dispuesto para creación de interfaces.

Utilizar imágenes para la creación de la interfaz de usuario sería una forma sencilla de implementar pero es más complicado crear interfaces agradables o con formas diferentes a un cuadrado, ya que la detección de posición se limita a una figura rectangular.

Hacer uso de la Microsoft Expression Blend permite obtener resultados con gran calidad visual, que además permite la programación directa en C#, para ello, se hace uso de un formato creado por Microsoft, sin embargo, el software no es libre y aún cuando existe una versión académica gratuita, no se ha encontrado la opción de descargarlo para nosotros como alumnos de ESCOM, además, como no conocida por el equipo de desarrollo, aprender a usarla podría requerir más tiempo del esperado.

Como resultado de la evaluación previa, se optó (temporalmente) de crear una interfaz básica (no implementada del todo, aún) haciendo uso de imágenes con ligeros efectos alpha para ser usados como interfaz.

3.3.1.3 - Desarrollar, Verificar Y Validar (Probar)

Se instalaron las herramientas consideradas para evaluarse, tales como:

- XNA Game Studio 2.0
- Blender 2.47
- 3ds Max 8 (Trial Version)
- PhotoShop CS3 (Trial Version)
- Microsoft Expression Blend 2

Se comprobó la compatibilidad de dichas herramientas.

- La exportación tanto de Blender como de 3DS Max en formato fbx no representó mayores problemas, salvo que el escalamiento en cada una de ellas es diferente, esto genera algunos objetos demasiado grandes y otros de menor tamaño.
- PhotoShop permite crear una gran variedad de tamaños con diferentes configuraciones en diversos formatos, por lo tanto, se hicieron pruebas con texturas incluso de 1920x1024, lo cual generó Warnings respecto a su compatibilidad con ciertos sistemas, pero la imagen se mostró satisfactoriamente.
- El audio se puede reproducir haciendo uso de formato .wav repitiéndose continuamente.

El Tech Demo 1.0 permite mostrar los resultados de la implementación, en la cual se manejan: mouse, texturas, modelos, y controles básicos.

3.3.1.4 - Planificar

- Buscar la mejor forma de mostrar interfaces, haciendo un demo básico del uso de Microsoft Expression Blend en XNA.
- Realizar pruebas de conexión básica entre 2 máquinas a través de un Lobby sencillo.

- Implementar un sistema de colisiones básico entre objetos.
- Mejorar el manejo de controles de usuario, haciendo uso de teclado y mouse.

CAPITULO 4 – AVANCES

Dentro de los avances en el desarrollo del proyecto se encuentran:

4.1 - Obtención De Información

Se ha encontrado información basta acerca de los mayas y el juego de pelota en particular, que nos permiten definir los modelos básicos, texturas y temática a utilizar en el juego, parte de dicha información se encuentra en el presente artículo y comprende la introducción y antecedentes.

4.2 - Análisis Y Diseño Del Sistema

El análisis y diseño del sistema se ha planteado de manera general para las partes básicas del desarrollo del mismo, lo que incluye diagramas de casos de uso, de clases y de secuencia, además ante la complejidad del diagrama de clases, se optó por realizar uno de paquetes que muestra de manera más sencilla (y ordenada) la relación entre las clases que conforman el juego.

4.3 - Selección De Herramientas

Después de un análisis se determinaron las herramientas a usar para la parte de modelado, animación, texturizado y desarrollo del sistema en general.

Dichas herramientas son:

- Blender 2.48 y 3D Studio Max 8 para el modelado y animación
- Gimp y Photoshop CS3 (Trial Version) para la creación de texturas
- XNA como framework de desarrollo, está basado en C# por lo que se optó por usar C# como lenguaje de programación, en particular usando la versión C# Express 2005 que nos permite trabajar de manera gratuita con XNA (que a su vez es gratuito). Incluye soporte de red, colisiones, lógica de juego, controles de usuario (mouse, teclado).
- XACT para el manejo de audio (archivos en formato wav, aiff), forma parte del XNA Game Studio.

4.4 - Diseño De Arte Del Juego

Como resultado del análisis y obtención de los datos referentes a la cultura maya, se comenzó con el diseño de personajes, logos y simbología relacionada con el Juego de Pelota, lo cual nos permitirá crear la atmósfera lo más realista posible de la cultura Maya.

En el anexo se muestra parte del arte relacionada con el Juego de Pelota que fue creada.

4.5 - Página Web

Los diseños fueron subidos a la página del proyecto, en la cual se muestran para acceso de los visitantes, quienes pueden realizar los comentarios pertinentes para la retroalimentación del equipo, así como la posible modificación de aspectos erróneos.

La página web del proyecto es: pok-ta-pok.blogspot.com

4.6 - Análisis De Física

Se han realizado pruebas de algunos motores de física, aunque no se ha determinado uno en particular para usarlo en el proyecto.

4.6.1 - Oops Framework For XNA

Entorno de trabajo (conjunto de clases) que se encargan de extender la funcionalidad de las colisiones incluidas en XNA. Maneja, entre otras cosas: colisiones entre objetos, objetos compuestos, colisión con Heightmaps (mapas de nivel), cuerpos sólidos.

Resultados de prueba:

- Implementación nativa en C# lo que simplifica su uso
- Conteo de frames por segundo aceptable (180-200fps), aunque disminuye con la adición de sombras (40-60 fps).

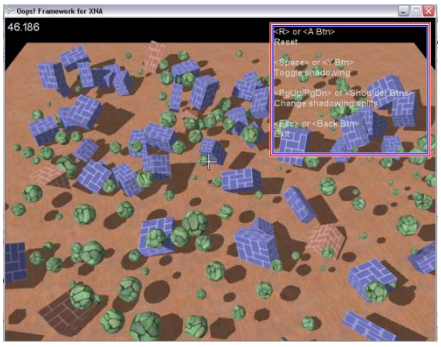


Fig.9 –Captura de pantalla de Oops Framework

4.6.2 - Bepu Physics

Motor de física compatible con XNA
Resultados de prueba:

- Amplia capacidad de física: fuerzas, colisiones, restricciones, aceleración, campos gravitacionales.
- Conteo de frames medio (60fps), sin aplicar texturas ni sombras.

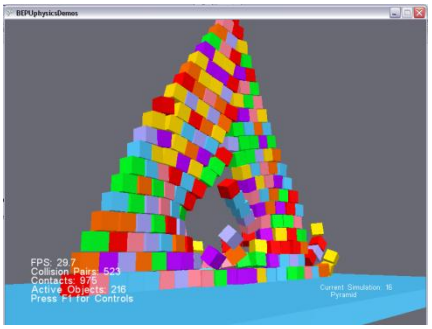


Fig.10 –Captura de pantalla de BEPU Physics

4.6.3 - Bullet Physics

- Manejo de cuerpos blandos (suaves) lo que otorga más realismo y precisión a la física
- Cuerpos compuestos
- Aceleración, gravedad, fricción

Resultados de prueba:

- La precisión y realismo de las simulaciones es destacable
- El conteo de frames por segundo es bastante pobre, tomando en cuenta que no se manejan texturas ni sombras, debido en gran parte a la precisión usada en la simulación

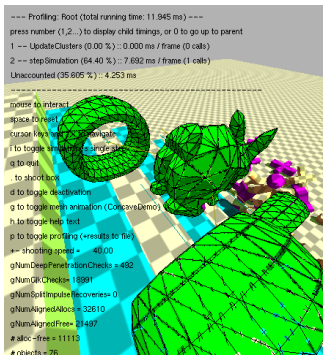


Fig.11 – Captura de pantalla de Bullet Physics

4.7 - Inteligencia Artificial

Se hicieron pruebas básicas de un agente lógico que reaccionaba en un entorno buscando minerales, el agente posee una base de conocimientos en Prolog que le permiten realizar inferencias utilizando una interfaz para comunicación entre C# y Prolog.

Su base de conocimientos se puede modificar haciendo aserciones o retractaciones según sea necesario, para mantener actualizada la información.

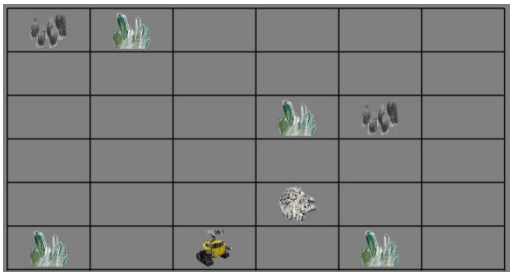


Fig.12 – Captura de pantalla del robot recolector de minerales en un entorno desconocido

Resultados de la prueba:

- La base de conocimientos del robot se mantiene actualizada correctamente
- La comunicación entre C# y Prolog es adecuada

- El tiempo de respuesta en la toma de decisión para la siguiente acción del robot es tardada en algunos momentos (principalmente al encontrar obstáculos o buscar rutas más cortas)

4.8 - Techdemo 1.0

Es el resultado de la aplicación de texturas, modelado, pruebas de audio, ocultamiento de caras, obtención de datos de un archivo XML, contador de frames por segundo (para verificar rendimiento del programa), iluminación por default.

Así como control de interfaces de usuario (mouse en particular), que nos permiten observar la complejidad de la creación del proyecto utilizando XNA.

A continuación se muestra una captura de pantalla del Tech Demo 1.0, como su nombre lo indica, sólo es un demo para mostrar las tecnologías usadas y el resultado que se puede lograr.

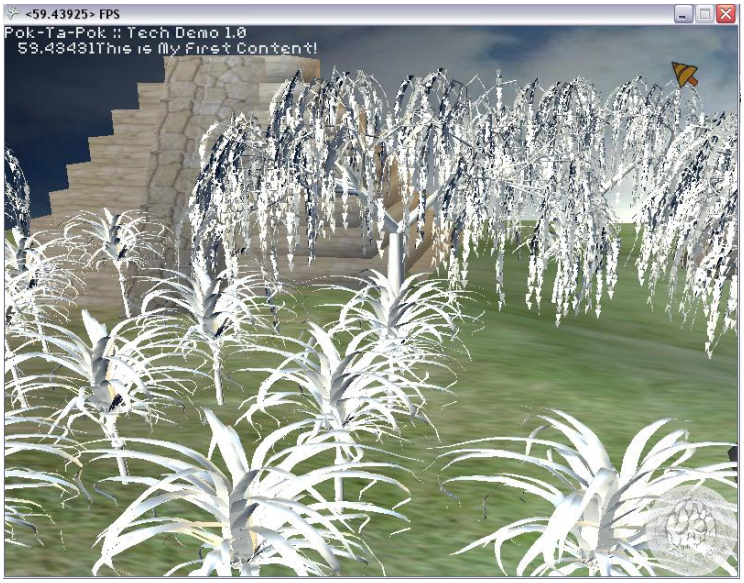


Fig.13 – Captura de pantalla del Tech Demo 1.0

En la imagen anterior se puede notar el resultado previamente descrito, se procede a explicarlo a detalle:

- Se tienen 24 árboles sencillos sin textura (como los que están más cercanos a la cámara)
- 4 Sauces (árboles un poco más detallados) igualmente sin texturas.
- Al fondo se puede ver un bosquejo de una pirámide Maya, con una textura simple
- El cielo está hecho por una esfera, llamada “Sky Sphere” con los vectores normales invertidos, lo que permite verla “desde adentro”.

- El terreno tiene una textura en efecto “tiling” que nos permite replicarla a lo largo y ancho de la superficie, que además cuenta con un ligero nivel de ruido, lo que la convierte en una superficie ligeramente irregular.
- En la esquina superior derecha se puede observar una pequeña flecha color café que representa al cursor y responde a los movimientos del mouse.
- El titulo de la ventana “<59.43925> FPS” refiere al conteo de frames por segundo a los que está corriendo el demo, permite obtener una estimación del rendimiento del sistema
- El texto que lee “This is my first Content!” es obtenido a través de un archivo XML, que permitirá manipular la información dentro del juego, (Perfiles y Preferencias).
- Se tiene un archivo de audio formato .wav reproduciéndose indefinidamente, mientras no se presione la tecla “Escape”, permitirá tener música ambiental a lo largo de la ejecución del demo, creando una atmósfera más envolvente en torno a la temática Prehispánica.

CAPITULO 5 – DESARROLLO DEL SISTEMA

En éste capítulo se aborda la implementación de la solución planteada para llevar a cabo el sistema. Para ello se evalúan los objetivos básicos del juego y la manera de implementarlos haciendo uso de XNA Game Studio 3.0, con programación en C# parte de la Suite de desarrollo Microsoft Visual Studio 2008, la integración de los diferentes elementos multimedia que conforman el mismo, así como el tratamiento de la física e inteligencia artificial.

Los elementos que conforman la presente aplicación se dividen en los siguientes aspectos:

- Manejo de Gameplay
- Interfaz gráfica de usuario (GUI)
- Manejo de Perfiles
- Modelado de escenarios y personajes
- Física (colisiones y fuerzas)
- Inteligencia Artificial
- Conexión de Red
- Técnicas avanzadas de graficación (Shaders y efectos, programación directa sobre el GPU)
- Elementos de Audio
- Manejo de entradas de usuario (Teclado y control de Xbox 360)

Los detalles de implementación de cada uno de ellos serán abordados en las siguientes páginas.

5.1 - Manejo De Gameplay

El Gameplay se refiere a la experiencia del jugador y su interacción con el sistema de juego. Su uso más adecuado también incluye a "lo que hace el jugador", también se relaciona con el término "game mechanics" que se refiere a las reglas en el juego que buscan lograr una experiencia agradable de juego, tomando en cuenta los estados por los que pasa el juego, la fluidez del mismo, y para algunos más complejos, la trama y secuencias que forman la historia del videojuego.
El gameplay está igualmente relacionado con el tipo de juego, entre los que encontramos los siguientes:

- First Person Shooter (FPS)
  - Características
    - Cámara en primera persona (Simulando visión del personaje)
    - Manejo de armas y disparos
    - Desarrollados en 3 Dimensiones
    - Alta inmersión de juego
    - Temáticas comunes : Guerra, Invasión extraterrestre
  - Ejemplos
    - Halo Series
    - Unreal Tournament Series
    - Quake Series
    - Doom Series

Fig. 14 - Captura de pantalla de Doom 3

- Third Person Shooter (TPS)
  - Características
    - Similar a los FPS, pero con cámaras en tercera persona
    - Mayor libertad y ángulo de visión
  - Ejemplos

- Gears of War Series
- Metroid Prime
- Max Payne
- Grand Theft Auto

Fig. 15 - Captura de pantalla de Gears Of War

- Racing (carreras)
  - Características
    - Basados en un sistema de manejo (autos, motos, camiones)
    - Uso de elementos adicionales : volante, pedales, palanca
  - Ejemplos
    - Need for Speed Series

Fig. 16 - Captura de pantalla de Need For Speed

- Otros tipos
  - Puzzle
    - Prueban el ingenio del jugador a través de acertijos
    - Suelen ser en 2 dimensiones
    - Ejemplos : Tetris
  - Pelea
    - Enfrentamiento de 2 o más jugadores cara a cara.
    - Suelen ser en 2 dimensiones
    - Ejemplos : King Of Fighters, Street Fighter
  - Deportes
    - Simulación de algún deporte en particular
    - Manejo de física en balón, pelota y animación de personajes
    - Ejemplos : FIFA Series

Fig. 17 - Captura de pantalla de ISS 64

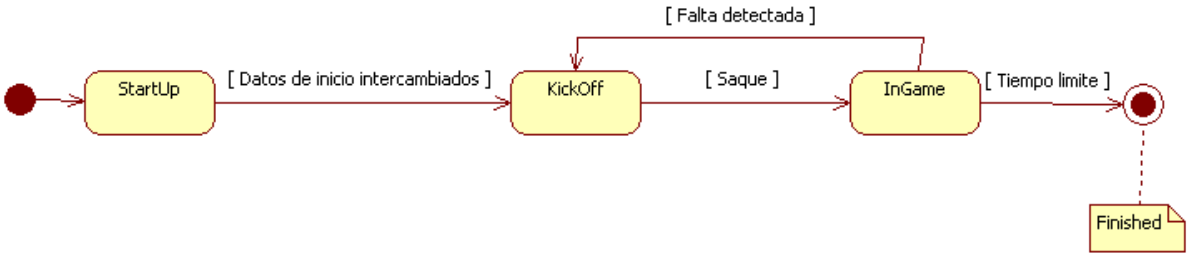
Debido a la clasificación presentada anteriormente, el juego se clasifica en el género de Deportes.
Los juegos de deportes tienen estados bien definidos para el gameplay, usando ése principio, se crearon estados según los escenarios que se presentan a lo largo del desarrollo del juego, los cuales se enuncian a continuación:

5.1.1 - Estados De Juego

Los estados de juego determinados para desarrollar el GamePlay adecuadamente son los siguientes:

- StartUp
- KickOff
- InGame
- Finished

La interacción entre los mencionados estados se muestra en el siguiente diagrama de estados.



5.1.1.1 - Estado: Startup

Estado al principio del juego, después de realizada la conexión, se intercambian datos, tales como: nombre, imagen y modelos del equipo del jugador contrario, con ello se obtiene información del contrincante y se despliega en la pantalla de ambos jugadores.
Se incluye una bandera (flag) que indica si el equipo remoto ha recibido o no los datos enviados.

Al haber recibido los datos, se pasa al estado KickOff, en donde el host tiene el saque inicial en un principio.

5.1.1.2 - Estado: Kickoff

Una vez que se ha confirmado la recepción por ambas partes, se procede a cambiar a un estado de KickOff, o saque inicial, en el cual se espera a que el jugador correspondiente libere la pelota, lanzándola hacia algún punto deseado, después de lo cual se procede al estado InGame.

5.1.1.3 - Estado: Ingame

Cuando la pelota ha sido lanzada por el jugador, el estado de juego cambia a InGame, en el cual se desarrollan los movimientos de los diferentes equipos en su intento de ganar la partida.

Para ello, se deben considerar las siguientes reglas:

Son causantes de puntos a favor y saque para el equipo contrario además de reseteo del estado de juego a KickOff:

- Si un jugador lanza la pelota y cae en su misma área de juego (su mitad del escenario) o bien, es arrojada fuera de los límites de la cancha del juego de pelota.
- Si una pelota, proveniente del equipo contrario bota más de una ocasión en el piso sin ser golpeada por alguno de los miembros del equipo controlado
- Si el jugador tiene la posibilidad de golpear la pelota (estar en una casilla en la que se encuentre la pelota) y no lo hace.

El ciclo se repite mientras el tiempo de juego no llegue a su fin, evento que cambia el estado a finalizado.

5.1.1.4 - Estado: Finalizado

Al llegar el tiempo límite, el juego finaliza mostrando la pantalla indicada con los resultados finales de ambos equipos.
En éste estado se actualizan los valores de los jugadores (score, dependiendo si gana o pierde el juego, será el valor; y además podrá aumentar su velocidad) así como el puntaje del perfil en general, juegos ganados y perdidos, los empatados sólo aumentan el total de juegos jugados.

Fig. 18 – Mensaje de partida finalizada

5.2 - Interfaz Gráfica De Usuario (GUI)

Con la finalidad de lograr una experiencia agradable en el uso del sistema, se buscó implementar un manejo de interfaces sencillas e intuitivas que permitieran aprovechar los recursos y proporcionaran acceso a los casos de uso presentados en la documentación.

Pantallas De Menú

- Manejables a través de:
  - Control de Xbox 360
    - Stick direccional : Navegación de opción
    - Botones A,B : Selección de opción, cancelar opción, respectivamente
  - Teclado
    - Flechas arriba/abajo : Análogo al Stick direccional (Véase control Xbox 360)
    - Enter/Scape : Análogo a botones A,B (Véase control Xbox 360)
- Permiten acceso a las diferentes opciones de configuración y creación de partidas
- Efectos de transiciones entre una y otra (fade-out)
- Manejo de un fondo vistoso para mejorar apariencia

Entre las pantallas de menú encontramos las siguientes:

5.2.1- Menú Principal

Muestra las principales opciones dentro del Juego:

1) **Jugar**
2) **Perfil**
3) **Preferencias**
4) **Salir** * Para todos los casos, regresa a la opción anterior

Fig. 19 –Menú Principal

1) *Jugar*

a. *Crear Sesión*
b. *Encontrar Sesiones*
c. *Salir*

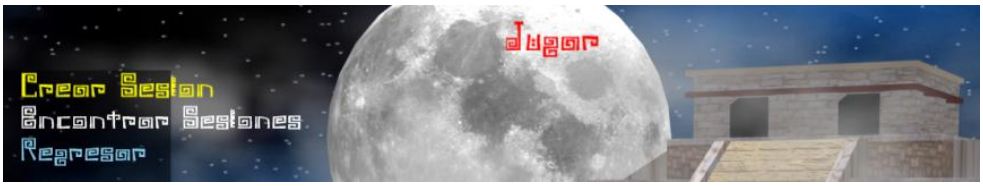


Fig. 20 – Menú Jugar

a. *Crear Sesión*

Inicia la conexión de red local para dar de alta un Host(servidor) que alojará la partida creada, dicha partida se creará en la máquina local y esperará la conexión de un cliente con quien habrá de desarrollar la partida en una pantalla de "Lobby".

Si la red no está disponible o hay algún problema (firewall, etc.) el sistema lo notificará haciendo necesario revisar la conexión de red local

Fig. 21 – Lobby

b. *Encontrar sesiones*

Buscará a modo de cliente, sesiones creadas en la red local con la cual pueda conectarse, de ésta forma, si la conexión es satisfactoria, se entra al mismo Lobby en el que el host se encuentra, de otra forma, se notifica la inexistencia del Host.

Fig. 22 –Búsqueda de sesiones

2) Perfil

Permite accesar a la administración de perfiles a través de una interfaz especial que se mostrará posteriormente (Véase "Pantallas de Configuración")

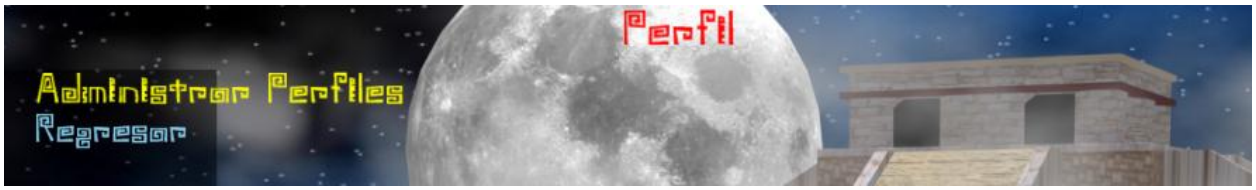


Fig. 23 – Perfil/Administración de Perfiles

3) Preferencias

Permite accesar a la administración de preferencias a través de una interfaz especial que se mostrará posteriormente (Véase "Pantallas de Configuración")

1. **Pantallas de Configuración**
    - Manejables a través de:
        - Mouse
            - Implementación de elementos típicos de formulario
                - Combo Box

- Botones
- Labels
- SpinBoxes

- Basados en la dll Tom Shane Neoforce Controls

Se cuenta con dos pantallas de configuración

a) Perfil

a. Inicialmente, verifica la existencia de un perfil abierto, de no ser así, pide iniciar sesión con un perfil existente de Games Live de Microsoft, ya sea en modo Online u Offline, una vez elegido, se muestra la pantalla en cuestión con los datos propios del perfil, los cuales podrá modificar a su gusto, tomando en cuenta:
   i. No se puede exceder de 4 integrantes ni tener menos de 4 para poder jugar.
   ii. Hay valores fijos de los integrantes que no se pueden modificar (nombre y atributos o modelo).
   iii. Los datos serán almacenados en el XML del Perfil sólo si se guardan los cambios.

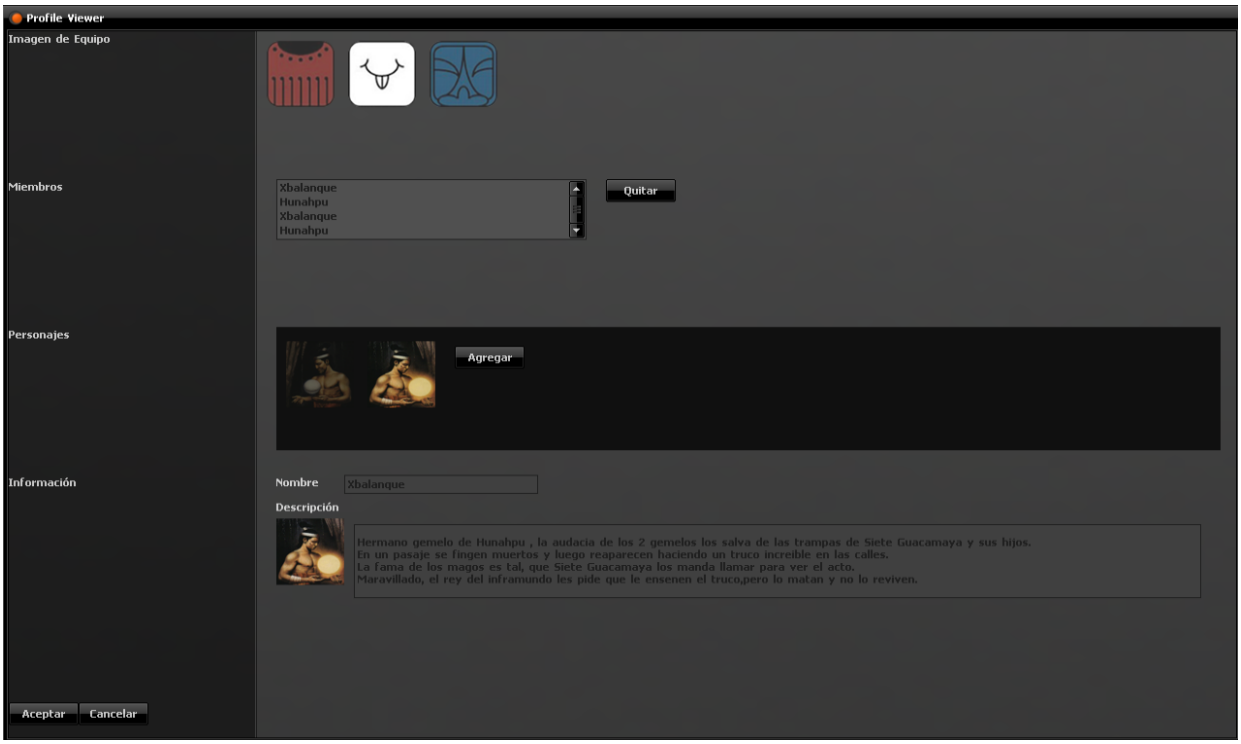


Fig. 24 – Pantalla Configuración de Equipo

b) Preferencias

a. Permite customizar el volumen tanto de efectos de audio (choques con objetos), como el volumen de la música de fondo, permitiendo obtener una mejor experiencia de juego

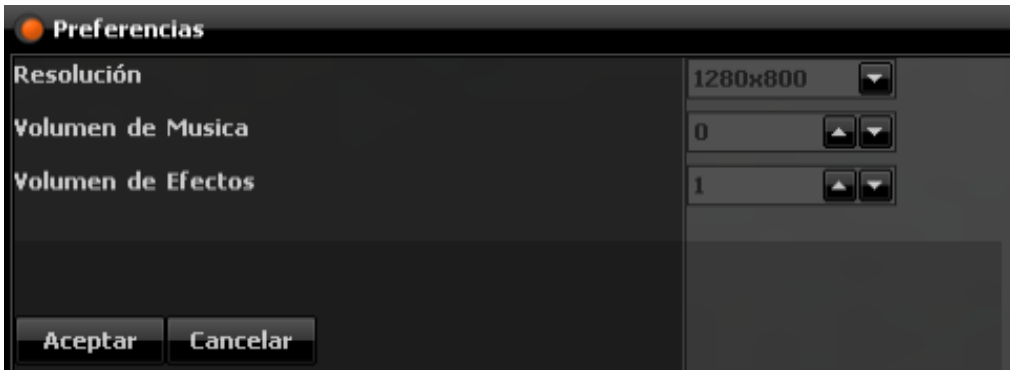


Fig. 25 – Pantalla Preferencias

2. Pantallas en Tiempo de Juego

- Pause:
  - Controlable con teclado y/o control de Xbox 360 (al estilo de pantallas de menú)
  - Permite pausar el juego sin detener la partida en curso, ni el tiempo transcurrido
  - Permite salir o finalizar la sesión.
- Información de los jugadores en la sesión
  - Muestra entrada/salida de un jugador a la sesión
  - Despliega la imagen del equipo (local y remoto)
  - Despliega el nombre del contrincante (local y remoto)
  - Muestra tiempo/estado transcurrido del juego y puntaje de los equipos

Fig. 26 – Pausa en tiempo de juego

5.3 - Manejo De Perfiles

5.3.1 - Almacenamiento De La Información

XML (EXtensible Markup Languaje)

El lenguaje de etiquetas XML por sus siglas en ingles, creado por la W3C, que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas, se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable, y que además es una tecnología sencilla que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

XSD (XML Schema Definition)

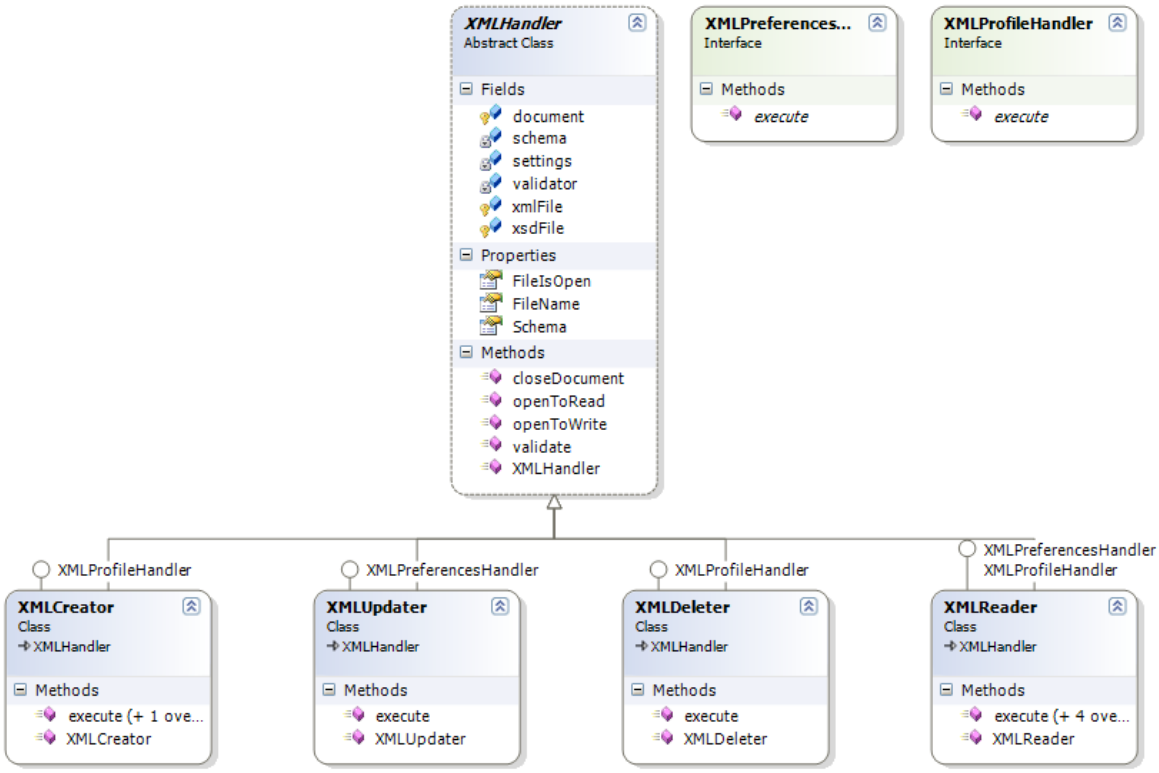
XML Schema es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, escrito en XML, basado en la gramática y pensado para proporcionar una mayor potencia expresiva que las DTD, menos capaces al describir los documentos a nivel formal.

Los documentos esquema (usualmente con extensión .xsd de XML Schema Definition (XSD)) se concibieron como una alternativa a las DTD, más complejas, intentando superar sus puntos débiles y buscar nuevas capacidades a la hora de definir estructuras para documentos XML. La principal aportación de XML Schema es el gran número de tipos de datos que incorpora. De esta manera, XML Schema aumenta las posibilidades y funcionalidades de aplicaciones de procesado de datos, incluyendo tipos de datos complejos como fechas, números y strings.

Se utilizaron XML en conjunto con XSD, para el almacenamiento de la información que es utilizada en el juego. XML como un lenguaje para el almacenamiento de la información concerniente al juego, como son preferencias, perfiles almacenados, información acerca de cada modelo e imagen que se utiliza en la aplicación, esto se complemento con documentos .XSD para validar la coherencia de los archivos y así evitar problemas de corrupción de archivos.

XML Y XSD

En cuanto al uso que se le dio a XML y XSD, fue para el almacenamiento de información y validación de esta, se diseño toda una jerarquia de clases que nos permite realizar todo el manejo de los archivos .xml, tales como la creación, actualización, borrado y lectura de los mismos. A continuación se muestran las clases creadas:



Las interfaces XMLPreferenceHandler y XMLProfileHandler fueron creadas para definir qué cosas y sobre qué tipo de información podían trabajar las clases que heredan de XMLHandler, por ejemplo, no tiene sentido que se puedan borrar las preferencias del juego.

Método execute(Profile p) perteneciente a la clase XMLReader, la cual se carga de cargar , validar e interpretar los archivos xml, en este caso se realiza la lectura de un perfil previamente guardado en el disco.

```
public bool execute(Profile profile)
{
	if (!validate())
	{
	profile.defaultValues();
	return false;
	}
	XmlDocument document = new XmlDocument();
	document.PreserveWhitespace = true;
	document.Load(xmlFile);
	XmlNode prof = document.SelectSingleNode("profile");
	profile.Name = prof.Attributes["name"].Value;
	XmlNode games = prof["games"];
	profile.TotalGames = uint.Parse(games.Attributes["total"].Value);
	profile.WonGames = uint.Parse(games.Attributes["won"].Value);
	profile.LostGames = uint.Parse(games.Attributes["lost"].Value);
	XmlNode team = prof["team"];
	XmlNodeList members = team.SelectNodes("member");
	if (Int32.Parse(team.Attributes["members"].Value) != members.Count)
```

```
        {
                profile.defaultValues();
                return false;
        }
        Team tempTeam = new Team();
        for (int i = 0; i < members.Count;i++ )
        {
                string name= members.Item(i).Attributes["name"].Value;
                int indice = Program.game.AvailableChars.IndexOf(
                new Character(members.Item(i).Attributes["name"].Value));
                if (indice < 0)
                {
                        indice = 0;
                        name = Program.game.AvailableChars[0].Name;
                }
                tempTeam.Members.Add(new Character(name,
                float.Parse(members.Item(i).Attributes["speed"].Value),
                float.Parse(members.Item(i).Attributes["strength"].Value),
                float.Parse(members.Item(i).Attributes["score"].Value)));

                tempTeam.Indices.Add(indice);
        }
        profile.Team = tempTeam;
        profile.PictureName = null;
        foreach (Picture p in Program.game.AvailableTeamPics)
        {
                if (prof["picture"].Attributes["name"].Value == p.Name)
                {
                        profile.PictureName = p.Name;
                        profile.PictureIndex                                                    =
                        Program.game.AvailableTeamPics.IndexOf(p);
                }
        }
        if (profile.PictureName == null)
        {
                profile.PictureName = Program.game.AvailableTeamPics[0].Name;
                profile.PictureIndex = 0;
        }
return true;
}
```

Otro método importante es la validación del archivo xml el cual está definido en la superclase XMLHandler

```
public bool validate()
{
        validator = null;
        try
        {
                settings = new XmlReaderSettings();
                settings.ValidationType = ValidationType.Schema;
                schema = new XmlSchemaSet();
                schema.Add("", xsdFile);
                settings.Schemas = schema;
                validator = XmlReader.Create(xmlFile, settings);
```

```
            while (validator.Read()) { }
            return true;
        }

        catch (XmlException)
        {
            return false;
        }
        catch (XmlSchemaException )
        {
            return false;
        }
        catch (ArgumentNullException )
        {
            return false;
        }
        catch (FileNotFoundException ex)
        {
            throw ex;
        }
        catch (FileLoadException ex)
        {
            throw ex;
        }
        finally
        {
            if(validator != null)
            validator.Close();
            settings = null;
            schema = null;
        }
}
```

A continuación se presenta la estructura del archivo profile.xml

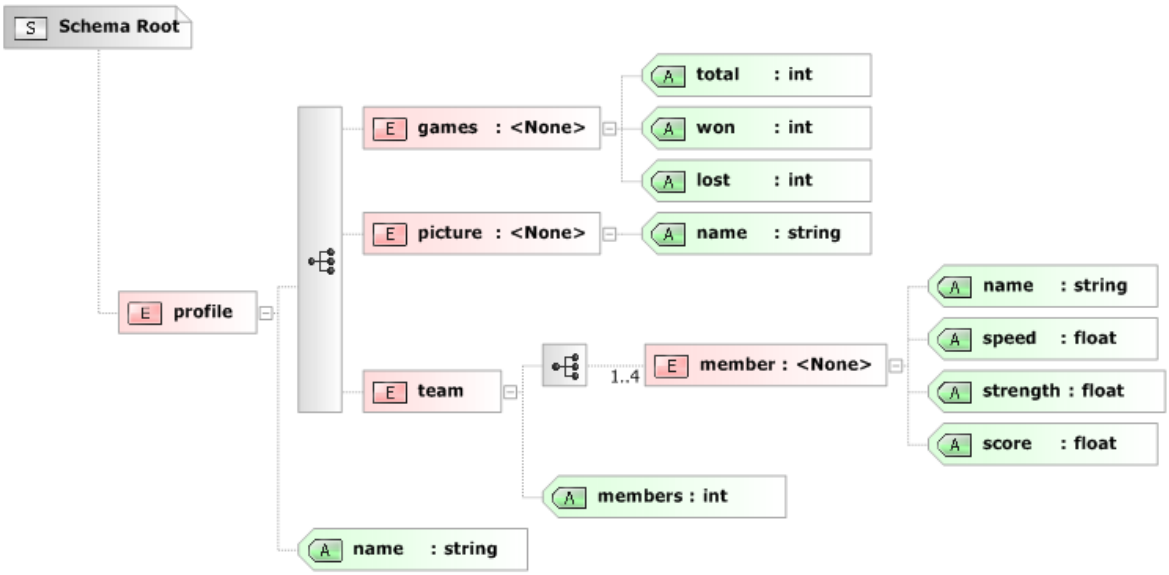


Fig. 27 – Esquema del XML “profile.xml”

Estructura del archivo carácter.xml que define a cada personaje del juego:

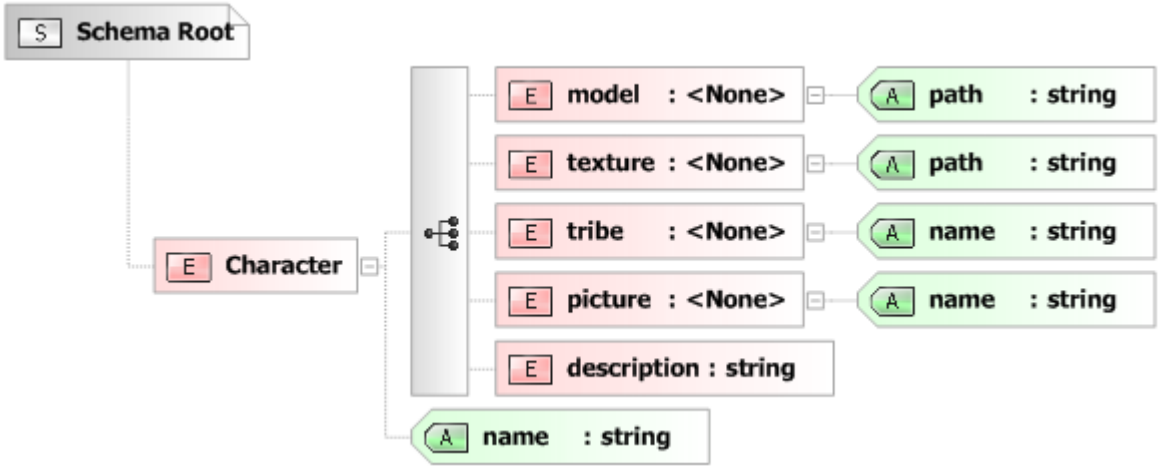


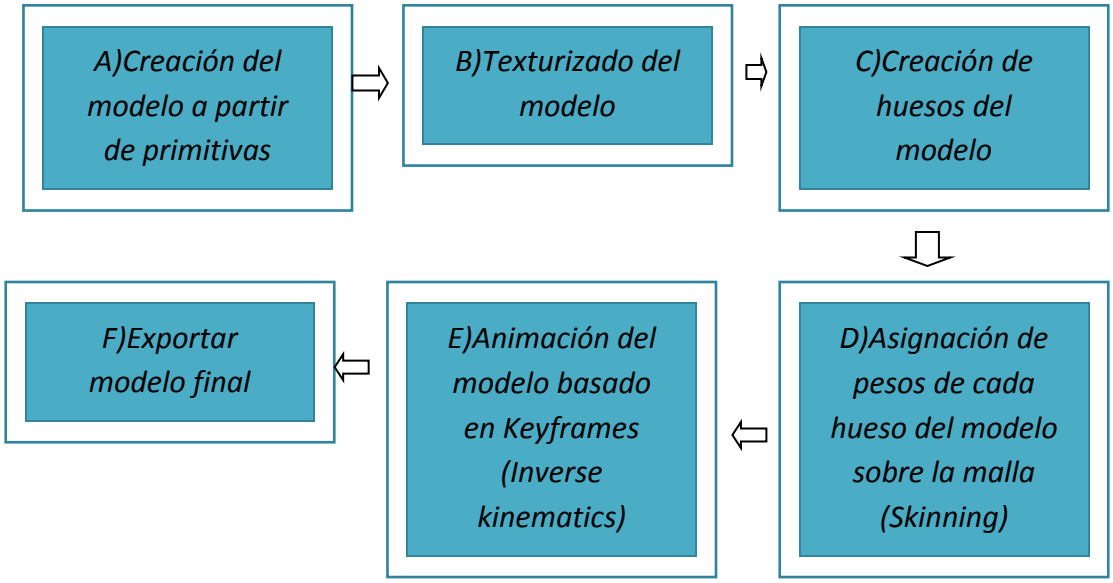
Fig. 28 – Esquema de XML “character.xml”

Todas las estructuras mostradas son la representación del archivo xsd, que se diseño para la validación de los mismos archivos xml.

5.4 - Modelado De Escenarios Y Personajes

Para el modelado de escenarios y personajes, se utilizó 3d Studio Max 2009 en su versión trial, ya que presenta exportadores adecuados para el uso de modelos en XNA 3.0

Para el proceso de modelado de personajes, se siguió el siguiente conjunto de acciones



A) Creación del modelo a partir de primitivas.

3ds Max 2009 permite crear modelos bastante complejos a partir de primitivas tales como cubos, esferas, tubos, toroides, etc.

Para ello conviene subdividir la primitiva y generar diversas partes del cuerpo que al final se habrán de unir en un solo modelo o malla.

Además, con la finalidad de agilizar el proceso de creación, 3ds Max 2009 permite usar técnicas como modelado en espejo, extrusión, añadir mallas, optimizar mallas, o bien un efecto de suavizado para evitar bordes muy bruscos.

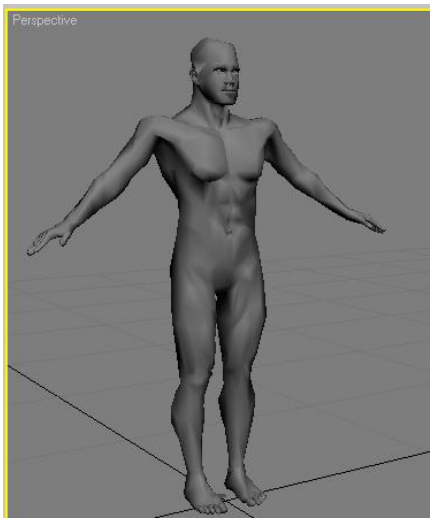


Fig. 29 – Modelo “lowpoly” sin texturas

B) Texturizado del modelo

Es posible añadir texturas, para buscar un modelo más realista, dichas texturas son creadas en programas de edición de imágenes como Photoshop o GIMP, y abarcan los formatos más comunes de imagen, entre los que tenemos: jpg, bmp, gif, png, etc.

Por razones de optimización, conviene tener texturas ligeras pero realistas y cuyas proporciones (largo y ancho) sean potencia de 2.

Fig. 30 – Modelo “lowpoly” con texturas básicas

C)Creación de huesos del modelo

3ds Max cuenta con módulos especializados en la creación de bípedos, los cuales se pueden usar para adaptarlo al modelo creado. Haciendo coincidir las respectivas partes del cuerpo

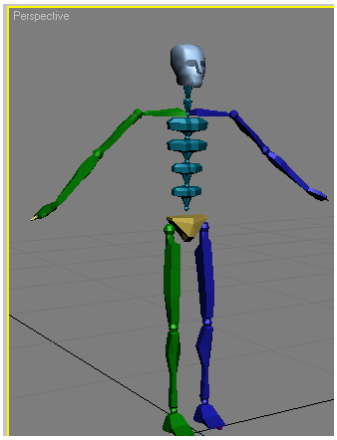


Fig. 31 – Bípedo usado para manejo de huesos

D)Asignación de pesos de cada hueso sobre la malla

Una vez creado el sistema de huesos y la malla, se deben relacionar entre sí, en un proceso llamad Skinning, en el cual es posible asignar el peso o influencia que tiene un hueso sobre alguna parte especifica del modelo creado.

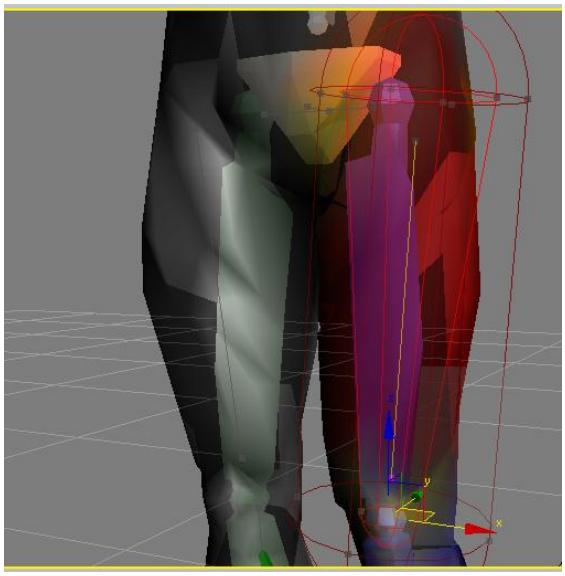


Fig. 32 – Asignación de pesos de hueso sobre malla

E) Animación del modelo basado en Keyframes (Inverse kinematics)

Ésta técnica consiste en poder animar un cuerpo basado en sus poses, por lo que el sistema se encarga de calcular las rotaciones necesarios de los huesos y de la piel para lograr el efecto deseado, además de que genera los frames intermedios que se encuentran entre cada keyframe o fotograma clave.

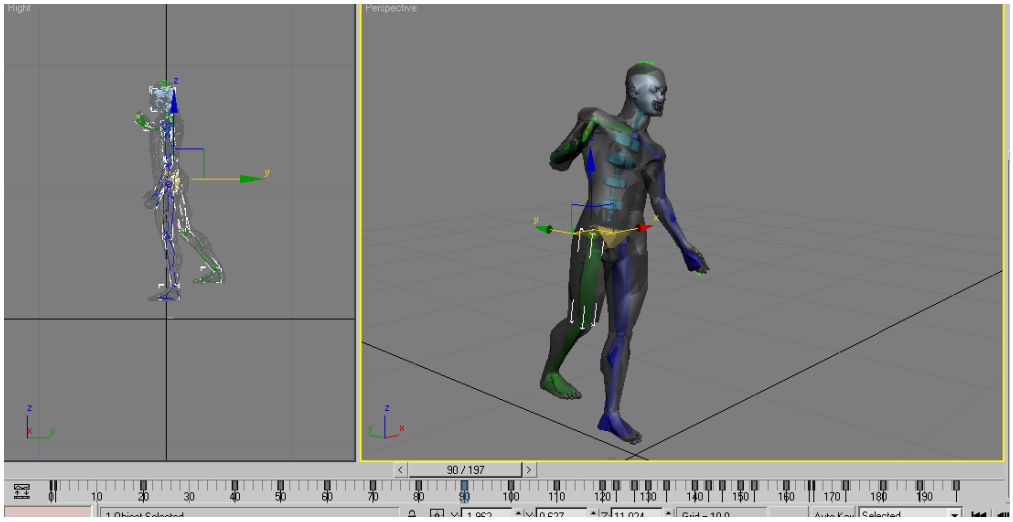


Fig. 33 – Animación y generación de poses

F) Exportar el modelo final

La exportación del modelo final se realiza a través de un plugin de 3ds Max que permite exportar también sus animaciones como una gran animación si importar cuantas sub-animaciones tenga dentro.

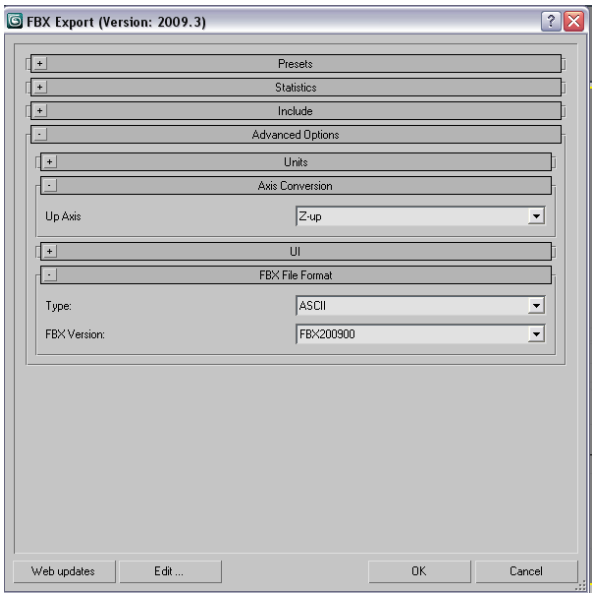


Fig. 33 – Pantalla del exportador a FBX

A través de un archivo XML, la animación se descompone en partes para que XNA pueda cambiarlas por la animación correspondiente al botón presionado, a continuación una muestra del XML que define el movimiento de los jugadores.

```
<?xml version="1.0" encoding="utf-8" ?>
<Animations>
 <Animation>
  <Name>Take 001</Name>
  <Framerate>30</Framerate>
  <SplitTask>
   <Name>Idle</Name>
   <StartFrame>1</StartFrame>
   <EndFrame>40</EndFrame>
  </SplitTask>
  <SplitTask>
   <Name>Run</Name>
   <StartFrame>41</StartFrame>
   <EndFrame>80</EndFrame>
  </SplitTask>
 </Animation>
</Animations>
```

Para la creación de los demás elementos del escenario se utilizó una técnica similar, pero evitando skinning, inverse kinematics, y creación de animación

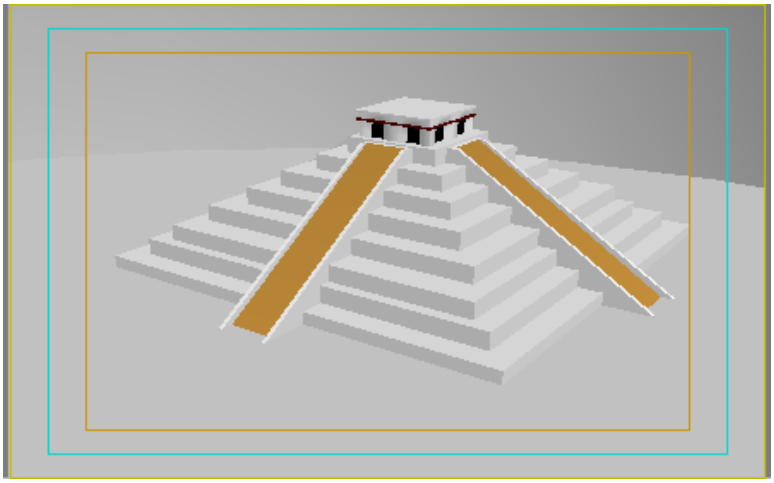


Fig. 34 – Pirámide creada, parte del escenario

5.5 - Física

Después del análisis de motores de física, se había escogido BEPU Physics, el cual está escrito en C#, sin embargo se decidió cambiar de motor de física, debido, a que presentaba problemas en la exportación de modelos con animación (huesos, animación en la línea de tiempo), y además de que aun se encuentra en fase de desarrollo, por lo cual en cada versión nueva del motor, había cambios sustanciales, agregando la poca o casi nula información de apoyo acerca del motor de física en cuestión. Por lo que se decidió utilizar otro motor de física que se adaptara a nuestras necesidades; el motor escogido fue el que estaba en segundo lugar de la lista de candidatos; este motor es JigLibX Library.

Fig. 35 – Logo JigLibx, motor de física

JigLibX Physics Library, es una versión especialmente escrita y desarrollada en C# del motor original Physic Engine JigLib.

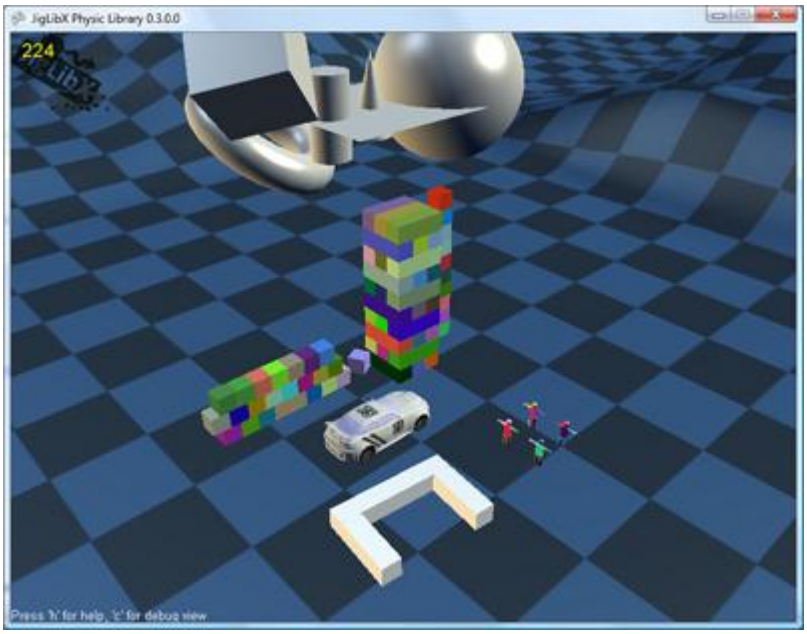


Fig. 36 – Ejemplo del uso de JiglibX

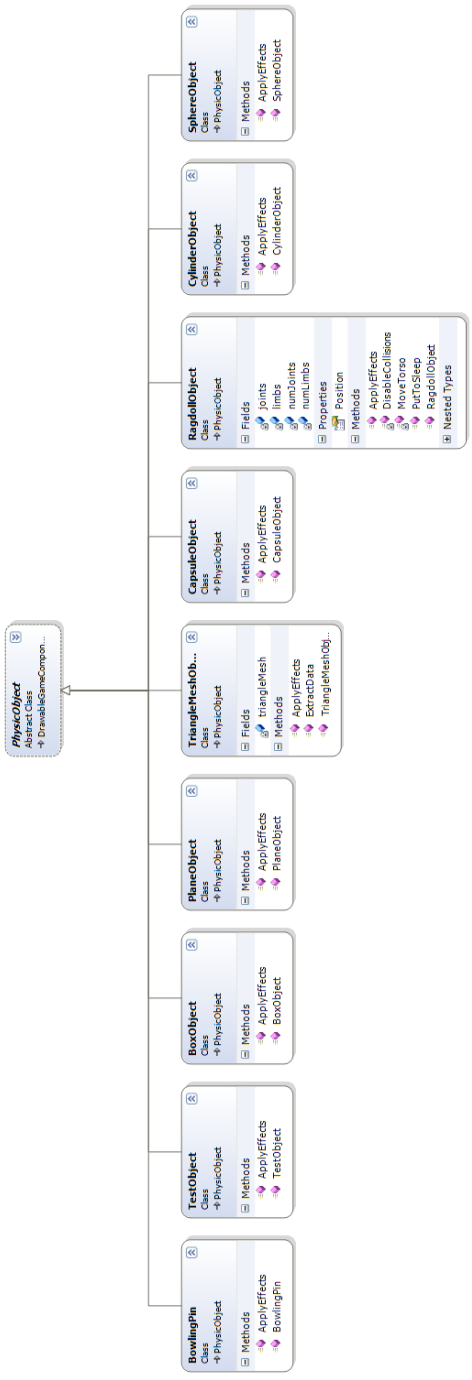
Dentro del diseño del modulo de la física, se tomo la decisión de no importar el modelo del personaje como tal al motor de física, esto es debido al alto costo computacional que se tiene al verificar la colisión de miles, si no es que millones de triángulos que conforman la malla del modelo. Cabe hacer notar que la mayoría de los desarrolladores de videojuegos toman esta decisión por cuestión de performance, y de no sobrecargar el CPU con cálculos de este tipo; por lo cual se optó por un esquema diferente para detectar la colisión de los personajes.

Dicha técnica consiste en establecer una primitiva (en este caso una caja) sobre cada parte importante del modelo; es decir acomodada de tal manera que concuerde con la posición del hueso que define esa parte del modelo; con el fin de simplificar el cálculo de las colisiones. Esta es una técnica muy usada en juegos del estilo FPS (First Person Shooter), como por ejemplo Half Life 1 y 2 de la empresa Valve.

En cuanto al escenario se estableció una fuerza de gravedad y se importaron al motor de física los muros que conforman el escenario, el piso o terreno sobre el que se desarrolla el juego, sin olvidar la pelota de juego, a la que el motor se encarga de calcular en todo momento su interacción con los demás elementos que se encuentran en el PhysicSpace del motor.

5.5.1 - Implementación Y Desarrollo De La Física

Para la implementación de la física se utilizó toda una jerarquía de clases, las cuales hacen posible la interacción de los modelos en el mundo controlado por la física.



Como se puede ver en el diagrama de clases del paquete Physics Objects, toda entidad que interactúa en el mundo controlado por el motor de física hereda de Physic Object, por ende tienen un Physics Body el cual es el que contiene atributos físicos del elemento, tales como masa, velocidad, posición y demás atributos, además de contener un Collision Skin, que es la malla del modelo y que permite evaluar si han ocurrido colisiones entre la malla y algún otra instancia.

Al momento de ocurrir una colisión se puede lanzar un evento y determinar cuáles con las entidades que han colisionado, para así determinar las reacciones necesarias; como por ejemplo saber si la pelota ha chocado con el terreno y poder determinar el número de ocasiones que esto ha ocurrido.

Método PhysicsSkin_callbackFn, el cual es llamado cuando se presenta una colisión entre dos entidades del juego, en este caso se trata de saber si la pelota ha colisionado con el terreno, y tener un control del número de rebotes que esta última ha tenido con el terreno.

```
bool PhysicsSkin_callbackFn(CollisionSkin skin0, CollisionSkin skin1)
//Evento lanzado cuando ha ocurrido una colision
{
      if (skin1.Equals(plane.PhysicsSkin))// se compara para saber si se ha
colisionado con el plano o terreno
      {
            posCol = skin0.NewPosition;
            bounceCounter++;                // se incrementa el contador de
      rebotes
            ((AudioHandler)Program.game.Services.GetService(typeof(AudioHandle
      r))).playClipEffect("fx");//efecto de audio para simular el rebote
      }
      if (bounceCounter > 1 && currentGameState!=GameStates.KickOff)//Si ha
rebotado en 2 ocasiones, se regresa al estado inicial de saque
      {
            currentGameState = GameStates.KickOff;
            bounceCounter = 0;
      }
      if(skin1.Equals(triObj.PhysicsSkin))
      {
            ((AudioHandler)Program.game.Services.GetService(typeof(AudioHandle
      r))).playClipEffect("fx");
      }
      return true;
}
```

En cuanto al movimiento de la pelota, el motor de fisica determina su moviemiento, dacuerdo a un tiro parabolico, sin embargo esto se explica con más detalle en la parte de la inteligencia artificial, como parte del agente inteligente.

5.6 - Inteligencia Artificial

Para la inteligencia artificial se optó por programar un agente inteligente, que trata de simular el comportamiento humano.

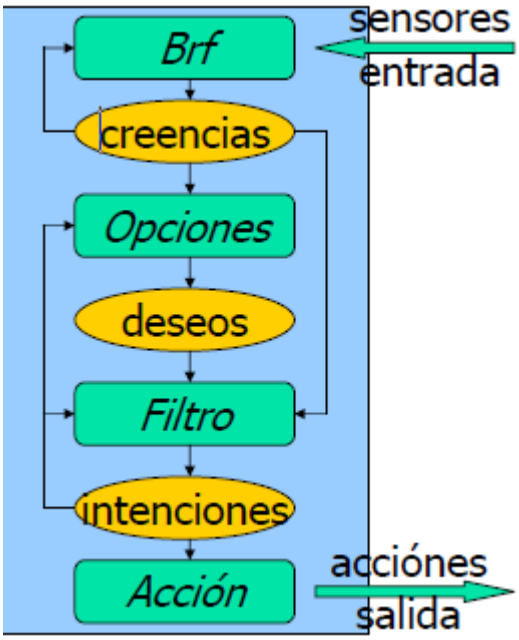


Fig. 37 – Esquema básico del modelo de un agente inteligente

En la figura mostrada arriba, se muestra los estados a grandes rasgos por los que pasa un agente BDI, que es que se modeló en la aplicación, para que esta controle a los otros jugadores de la persona que se encuentra interactuando con la aplicación en la PC, en la que se encuentra en la partida, cabe hacer notar que las posiciones de los jugadores de la otra persona se enviaran a través de la red y esta solo se actualizara en el tablero que tienen los agentes para representar su entorno.

5.6.1 - Implementación De IA

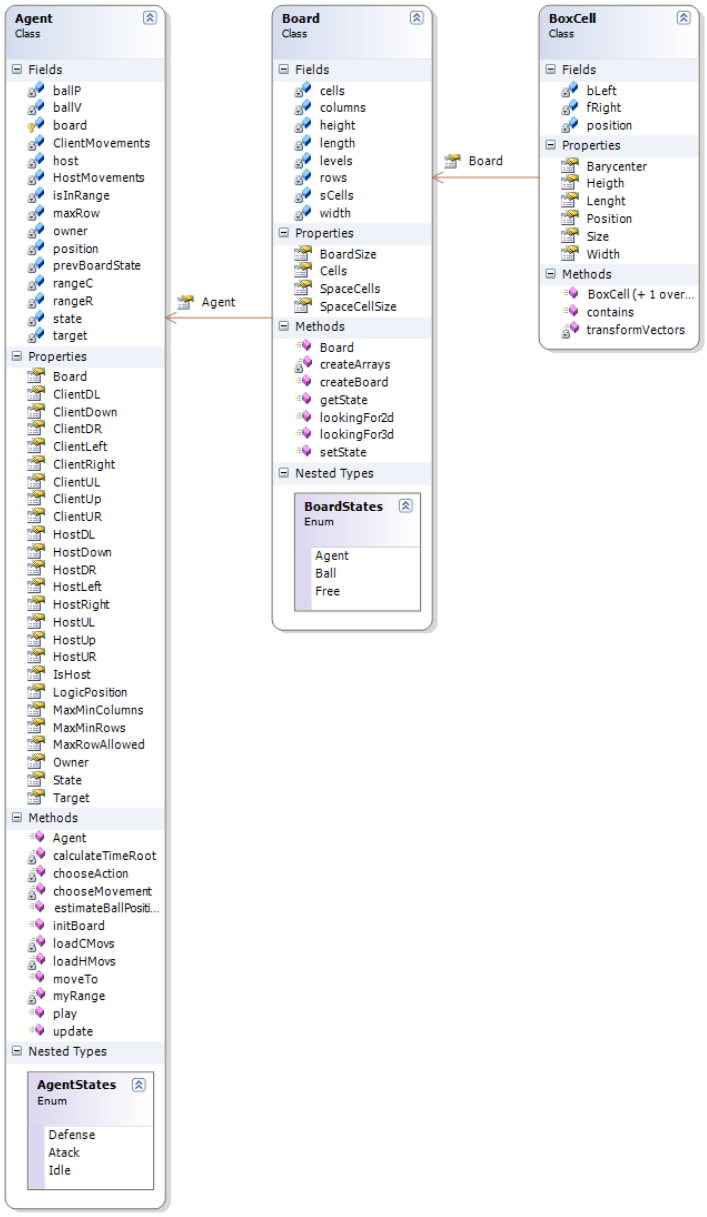
En cuanto a la inteligencia artificial se modelo un agente BDI, el cual tiene Creencias, Deseos e Intenciones.

Las creencias del agente, son el mundo que lo rodea.

Los deseos son sus metas u objetivos.

Las intenciones son las reacciones dependiendo de sus estado; es decir; la decisión que se toma.

Se muestran las clases creadas para el agente:



5.6.2 - Transformación Del Espacio En Un Tablero Discreto.

En este caso, el plano o terreno fue divido en casillas, de tal forma que se tenga un tablero de 2 dimensiones al estilo de un juego de ajedrez, llamado “cells”, y a su vez el espacio del escenario se dividió en un arreglo de 3 dimensiones de cajas, llamadas “spaceCells”, todo esto se realizo para hacer discreto el espacio y sea más fácil el movimiento y toma de decisiones de los agentes.

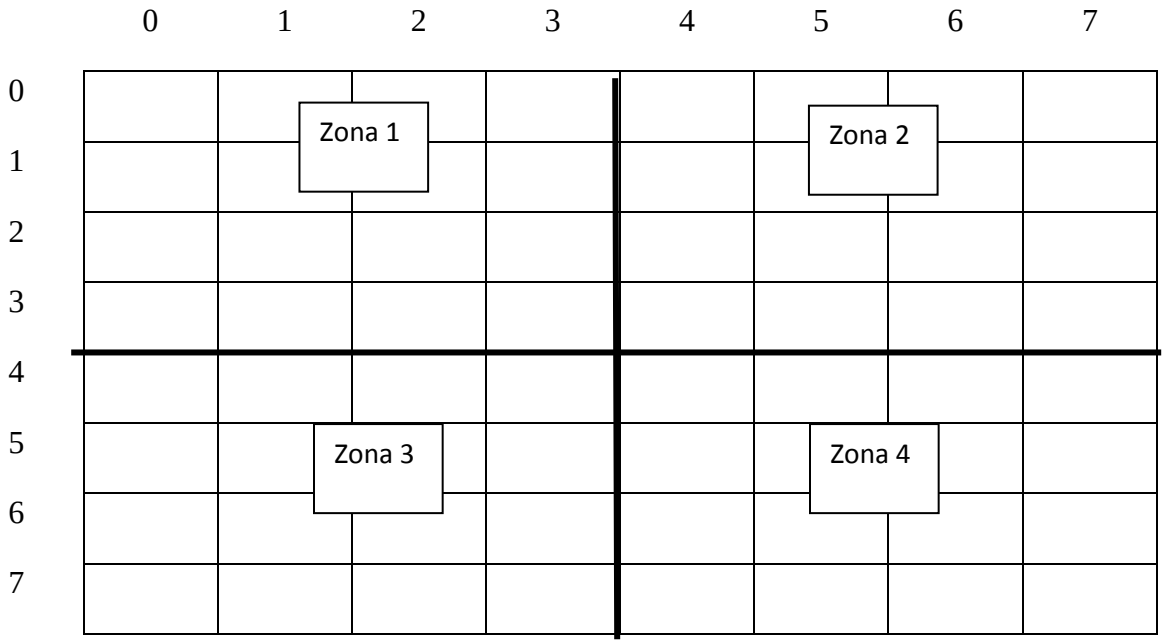


Fig. 38 – División del tablero en zonas

Además para evitar colisiones entre los personajes, se definieron espacios de movimiento para cada agente, de tal forma que solo se puedan mover en ese espacio, algo parecido a lo que es el volleyball. Esta división del tablero en zonas nos permite definir que agente se va a hacer cargo de golpear la pelota, después de estimar donde caerá la pelota sobre el terreno.

Movimiento del agente.

Para el movimiento del agente se definieron 4 tipos básicos y 4 compuestos, en este caso :

Los movimientos básicos son {Arriba, Abajo, Derecha, Izquierda}

Movimientos compuestos {Arriba-Izquierda, Arriba-Derecha, Abajo-Izquierda, Abajo-Derecha}, estos movimientos son en diagonal.

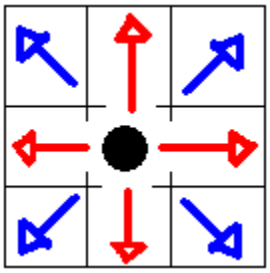


Fig. 39 – Movimientos compuestos y básicos del agente.

5.6.3 - Estimación Del Lugar De Caída De La Pelota.

Para estimar la posición de la pelota se utiliza las ecuaciones del tiro parabólico, tomando en cuenta que la pelota tiene un vector de velocidad y otro de posición en un determinado instante de tiempo, los cuales son modificados por el motor de física.

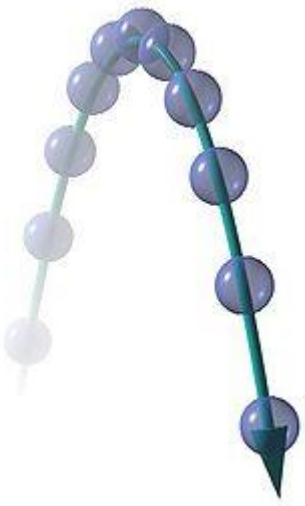


Fig. 40 – El tiro parabólico se desarrolla en 3 dimensiones

Estos vectores de velocidad y de posición instantánea nos sirven para estimar donde puede caer la pelota, un proceso clave para el funcionamiento del agente, ya que de ello depende la dirección que tiene que escoger, para tratar de llegar de la forma más rápida al objetivo y así poder determinar qué tipo de acción de golpeo va a efectuar, auxiliado de los spaceCells mencionados anteriormente.

A continuación se muestran las ecuaciones para estimar la posición de la pelota:

$$Vector\ de\ velocidad\ instantanea$$
$$V = V_0 = (Vx, Vy, Vz)$$

$$Vector\ de\ posición\ instantanea$$
$$P = (Px, Py, Pz)$$

$$y = V_{0y}t - \frac{gt^2}{2} + P_y$$

$$\frac{gt^2}{2} - V_{0y}t - P_y = 0$$

Lo que nos da una ecuación cuadratica, para determinar el tiempo total de vuelo de la pelota

$$t = \frac{\mathrm{V_{0y}} \pm \sqrt{{V_{0y}}^2 + 2\mathrm{gP_y}}}{\mathrm{g}}$$

Una vez obtenido el tiempo, se procede a calcular las componentes, en donde se pretende que debe caer la pelota.

$$x = V_{0x}t + P_x$$

$$z = V_{0z}t + P_z$$

Una vez hecho esto se tiene el vector de la posición futura de la pelota.

5.7 - Conexión De Red

Para implementar el manejo de la red local, en el modo de juego, se utiliza la clase networksession, que permite conocer el estado de la sesión actual en todo momento dentro de la ejecución del juego, ante cualquier problemática, ocurre la notificación y la sesión finaliza.

Además, hay conocimiento de todos los jugadores, en adelante conocidos como "Gamers" que participan en la sesión local, en éste caso uno, quien puede enviar datos, tipo "broadcast" a todos los jugadores conectados en la sesión, incluyéndose a sí mismo.

Al enviar datos, se hace uso de la clase llamada PacketWriter, la cual genera un paquete con el contenido a enviar, dependiendo del tipo de envío, serán las opciones del mismo, ya sea en orden y confiable, o no. De manera análoga, al recibir paquetes, se hace uso de la clase PacketReader, en donde se almacenan los datos obtenidos y la persona que envía dichos datos.

Sin embargo la continua llamada de algunos métodos del Packetwriter y PacketReader, dificultan el manejo del código, por lo que se optó por crear una clase serializable que se pueda mandar directamente al packetWriter y así ser enviada por completo, igualmente, al recibir información, la clase es deserializada y se analizan los datos obtenidos.

Se cuenta con dos clases serializables que permiten manejar la información en los diferentes estados del juego.

5.7.1 - Clase Serializable : StartUpPacket

Estado de Juego : StartUp

En el estado de juego StartUp, ocurre un intercambio de información tal como: la imagen del equipo y nombre del jugador contrarios, así como el id de los jugadores del equipo de ésa manera es posible saber contra quién se está jugando, y los modelos que usa.

Además un valor booleano que indica si quien envía ya ha recibido respuesta por parte del otro jugador, de ésta forma se puede saber cuando ambos han recibido los respectivos datos.

5.7.2 - Clase Serializable : InGamePacket

Estado de Juego : InGame

El mayor intercambio ocurre en éste estado de juego, por lo que se busca reducir al mínimo posible la cantidad de datos a enviar entre jugadores, debido a ello se optó por mandar, a grandes rasgos:

- Posición de los jugadores y su estado
- Posición de la pelota
- Score del equipo
- Tiempo transcurrido
- Booleano (ante alguna falta)

Con ésta información, es posible llevar a cabo el Sistema de Juego Planteado, de manera satisfactoria y segura en el intercambio de datos.

A continuación se muestran algunos fragmentos de código para ejemplificar lo explicado

Recepción de datos de la red

```
//Recibir datos
while (gamer.IsDataAvailable) //mientras haya datos por recibir
       {
           gamer.ReceiveData(packetReader,out sender); //Recibir datos

           if (!sender.IsLocal)
           {
              receivedPacket=StartUpPacketSerializer.Deserialize(packetReader);
              recvPacketType = receivedPacket.GetType();
           }
```

Envío de datos a través de la red

```
//envío de información
foreach (LocalNetworkGamer gamer in networkSession.LocalGamers)
{
    ReceiveNetworkData(gamer, gameTime);
    switch (currentGameState)
    {
       case GameStates.InGame:
       UpdateToSend(); //Actualizar datos antes de enviar, para ser enviados
       inGamePacketSerializer.Serialize(packetWriter, InGamePacket);//
       Serializar clase
        gamer.SendData(packetWriter, SendDataOptions.None); //Envío de datos
       …
   }
`
```

5.8 - Técnicas Avanzadas De Graficación

Con la finalidad de aumentar el realismo y la experiencia de juego, se optó por agregar algunos efectos adicionales al escenario que lo permitieran, con un consumo de pocos recursos.

Entre las técnicas usadas se tiene:

- Scattering/Skydome
- Billboarding
- Efectos de luz y niebla
- Efectos básicos de modelos
- Tiling

Las tres técnicas se basan y están programadas usando shaders en High Level Shading Language (HLSL) con Shader Model 3.0, que permite programar directamente sobre el GPU, liberando de carga al CPU para que éste se encargue de otras tareas como: tráfico de red, colisiones, audio, entradas de usuario, actualizaciones, lógica de juego, etc. A continuación se explica brevemente el concepto de shaders.

5.8.1 - Shaders

Un Shader, es un conjunto de instrucciones gráficas destinadas para el acelerador gráfico, estas instrucciones dan el aspecto final de un objeto. Los Shaders determinan materiales, efectos, color, luz, sombra y etc.

Para la escritura de esas instrucciones, los programadores hacen uso de unos lenguajes de programación diseñados específicamente para ello. Cada uno de estos lenguajes de programación necesita enlazarse con un API concreto, bien sea DirectX o bien OpenGL. Sin embargo, nVidia también diseñó un lenguaje que podía usarse indistintamente con DirectX u OpenGL. Entre los diferentes lenguajes para programación de GPU's tenemos:

- HLSL usado con DirectX y propiedad de Microsoft. Empezando con el DirectX 8.0 con Shader Model 1.0 y hasta el DirectX 10.1 con el Shader Model 4.1.
- GLSL usado con OpenGL y libre.
- CG propiedad de la empresa Nvidia.

El Shader Model 4.1, es el más avanzado entre los nombrados, y puede llevar gráficas sumamente reales como en los juegos y programas, usualmente de programación 3D. Uno de los juegos actuales que más utiliza el DirectX 10 con Shader Model 4.0 es Crysis de Crytek, que en

todos sus aspectos se le puede notar un gran avance con respecto a correrlo en Shader Model 3.0 y Shader Model 4.0.

Usualmente mientras más avanzado sea el Shader, la cantidad de objetos, texturas, efectos ambientales (Sol, Nubes 3D, Humos, Fuegos Realistas, Aguas, Iluminación) serán mayores con formas, colores y texturas más realistas.

La programación de los shaders puede ser auxiliada por programas de edición tales como: FX Composer, desarrollado por nVidia y que permite la programación de diversos tipos de shaders, a través de ejemplos básicos iníciales, customizables por el usuario, tanto para OpenGL como para DirectX.

Fig. 41 – Uso de shaders en juegos actuales

5.8.2 - Técnicas Usadas En El Juego

5.8.2.1 - Scattering/ Skydome

Combinación de efectos que consiste en generar un cielo más realista, con transición entre el día y la noche, con sus respectivos sol y luna(y estrellas) y que a su paso modifica la iluminación de los diferentes elementos en la escena, incluye un pequeño efecto de nubes, el cielo se mueve a la par de la cámara, lo que permite dar un efecto de profundidad realista.

Fig. 42 – Efecto de Skydome

5.8.2.2 - Billboarding

Técnica basada en los ejemplos de GPU Programming Gems para lograr grandes cantidades de follaje renderizadas por el GPU, consiste en imágenes simples (planos de una sola vista) que giran siempre hacia la cámara, por lo que en diversos puntos de vista aparentan ser pasto en 3 dimensiones.

Fig. 43 – Billboarding en el pasto

5.8.2.3 - Efectos De Luz Y Niebla

Al transcurrir el día y la noche, la luz recibida por cada elemento es diferente creando una escena más dinámica y llamativa, la niebla se comienza a presentar a una determinada distancia de los objetos, para simular profundidad.

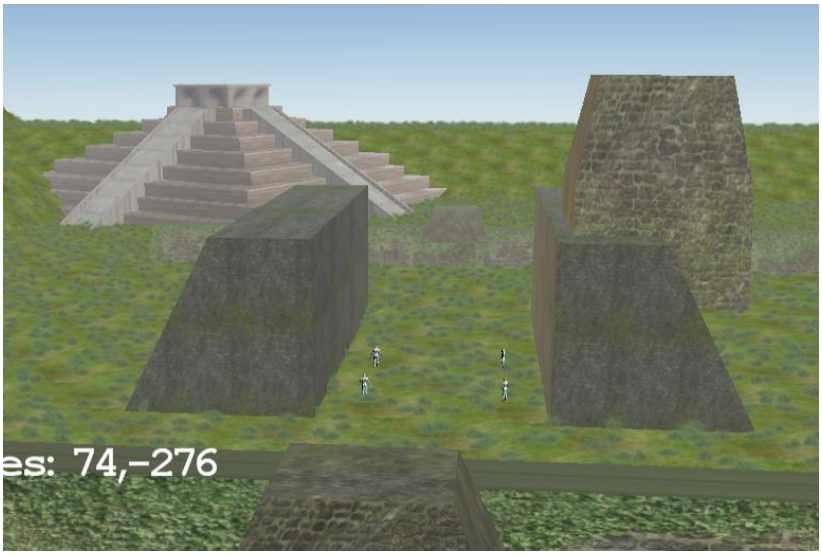


Fig. 44 – Ligero efecto de neblina (fog) en elementos lejanos a la cámara

5.8.2.4 - Efectos Básicos De Modelos

Los modelos creados cuentan con efectos básicos de iluminación y texturizado, suficientes para ser desplegados en XNA.

Fig. 45 – Iluminación básica sobre el personaje

5.8.2.5 - Tiling

Efecto simple aplicado a una textura con la que se genera la repetición de la misma a lo largo, ancho y alto de una superficie, disminuye el efecto de barrido en modelos muy grandes y compresión en aquellos muy pequeños.

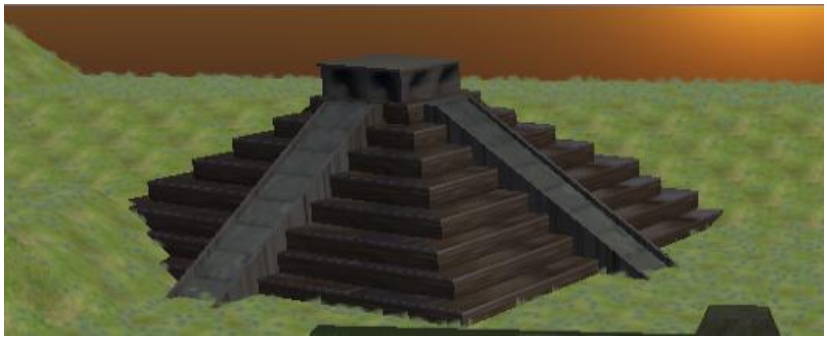


Fig. 45 – Efecto Tiling o repetición de texturas

5.9 - Elementos De Audio

Considerando que el juego es una mezcla de elementos multimedia, la inclusión de efectos de audio, temas de fondo y clips continuos permite crear una atmósfera más creíble y envolvente.

Se cuenta con un banco de sonidos desde el cual es posible mandar reproducir algún tema en particular en un determinado momento, por ejemplo, en el evento de la colisión del balón con algún muro o con el suelo.

Además de un tema de fondo, reproduciéndose en modo loop (“Reproducción continua”) con algunos efectos de audio, como ondas de viento.

Para la edición de banco de sonidos se usa una herramienta propia de XNA llamada XACT – Audio Creation Tool, la cual soporta exclusivamente formatos wav y al ser parte de de XNA Game Studio 3.0 su integración es muy natural.

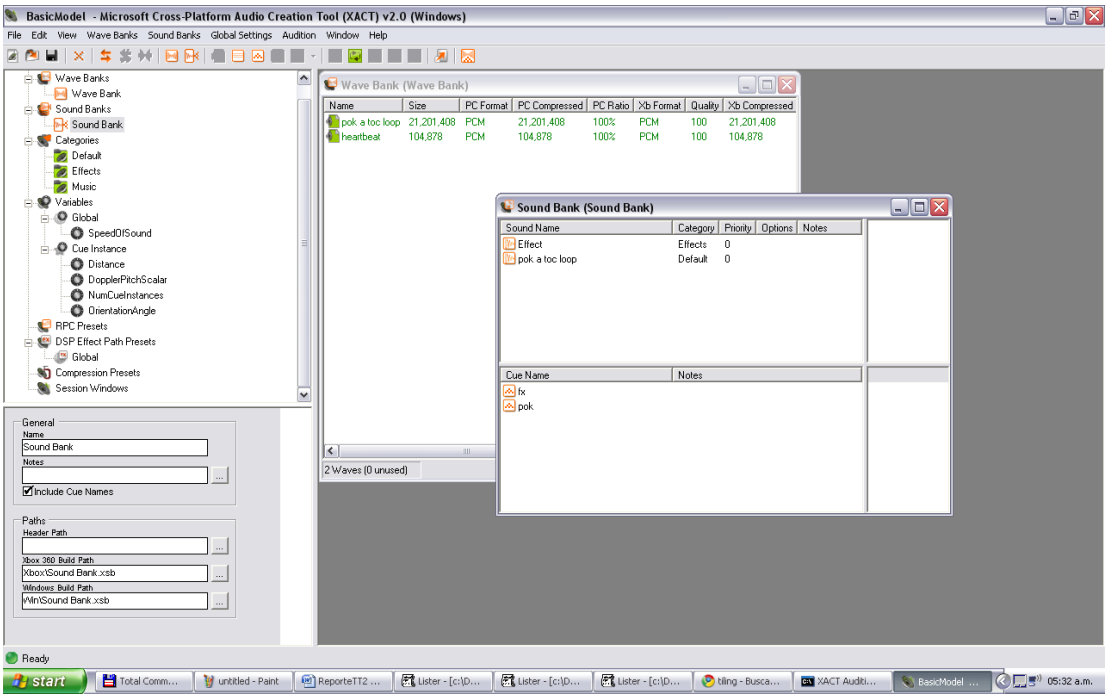


Fig. 46 –XACT, herramienta para manejo de audio en XNA

A continuación se agrega un fragmento de código del uso de AudioHandler, componente usado para el manejo de audio.

```
public class AudioHandler : Microsoft.Xna.Framework.GameComponent
    {
        AudioEngine myAudioEngine;
        AudioCategory myMusicCategory;
        AudioCategory myEffectsCategory;
        Cue myCue, effectCue;
        SoundBank mySoundBank;
        WaveBank myWaveBank;
        Preference currentPreference;

        public AudioHandler(Game game,Preference pref)
            : base(game)
        {
            currentPreference = pref;
        }
        public override void Initialize()
        {
           myAudioEngine = new AudioEngine("Content\\Audio\\BasicModel.xgs");
           mySoundBank = new SoundBank(myAudioEngine, "Content\\Audio\\Sound Bank.xsb");
           myWaveBank = new WaveBank(myAudioEngine, "Content\\Audio\\Wave Bank.xwb");
           myMusicCategory = myAudioEngine.GetCategory("Default");
           myEffectsCategory = myAudioEngine.GetCategory("Effects");
           myMusicCategory.SetVolume((float)currentPreference.MusicVolume);
           myEffectsCategory.SetVolume((float)currentPreference.EffectsVolume);
           base.Initialize();
        }

        public void playClipSong(String clipName)
        {
           myCue = mySoundBank.GetCue(clipName);
           myCue.Play();
        }

       public void stopClipSong(String songName)
       {
                  myCue = mySoundBank.GetCue(songName);
                  if (myCue != null)
                    if (myCue.IsPlaying)
                        myCue.Stop(AudioStopOptions.Immediate);
       }
```

5.10 - Manejo De Entradas De Usuario

La **interfaz de usuario** es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo, normalmente suelen ser fáciles de entender y fáciles de accionar.

Dentro de las Interfaces de Usuario se puede distinguir básicamente dos tipos:

> A) Una interfaz de hardware, a nivel de los dispositivos utilizados para ingresar, procesar y entregar los datos: teclado, ratón y pantalla visualizadora, gamepads o controles;
> B) Una interfaz de software, destinada a entregar información acerca de los procesos y herramientas de control, a través de lo que el usuario observa habitualmente en la pantalla.

Sus principales funciones son los siguientes:

- Puesta en marcha y apagado
- Control de las funciones manipulables del equipo
- Manipulación de archivos y directorios
- Herramientas de desarrollo de aplicaciones
- Comunicación con otros sistemas
- Información de estado
- Configuración de la propia interfaz y entorno
- Intercambio de datos entre aplicaciones
- Control de acceso
- Sistema de ayuda interactivo.

Las tendencias tecnológicas actuales buscan la mayor comodidad al usuario, se ha logrado esto a través del uso del control de una de las consolas más populares del momento, la Xbox 360, la cual permite interacción sencilla con XNA a través de la constante actualización de los estados, permitiendo acceder a cualquiera de sus botones, sticks, gatillos en el momento deseado.

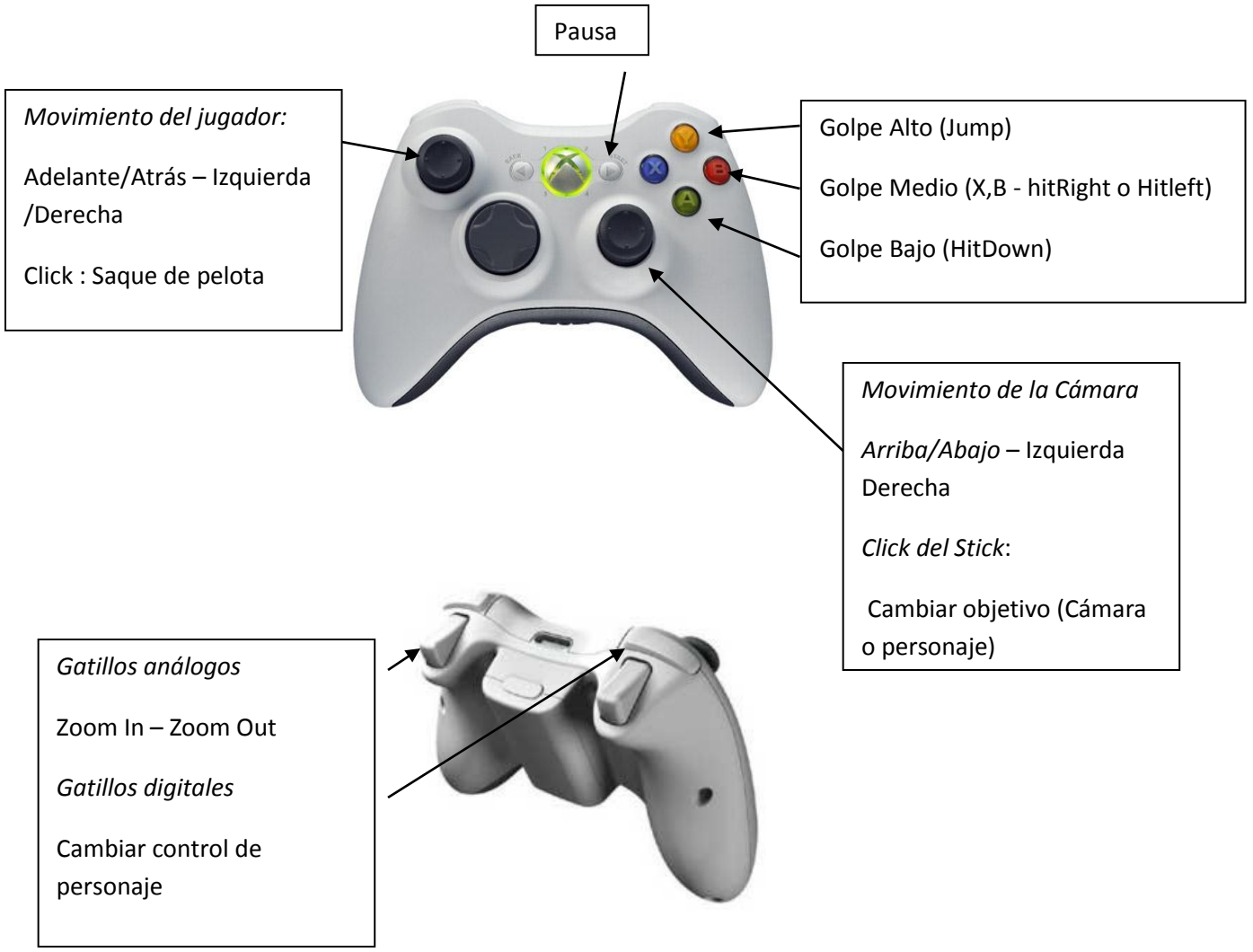


Fig. 47 – Vista del control de XBOX 360 y su configuración

A su vez, es posible combinarlo con interacción del teclado, como el manejo de flechas y teclas en particular, tales como Esc o Enter.

De ésta forma es posible lograr una experiencia más enriquecedora de fácil interacción con el sistema.

Se muestra un fragmento de código correspondiente a la entrada de usuario, en el uso del control de la Xbox 360

En el siguiente fragmento de código se muestra el manejo del estado del control de Xbox 360, obteniendo primero su estado(actual y previo), es decir, el estado de todos los botones presionados en el gamepad en ése momento, ya sea presionado o no; además de la posición de los sticks análogos y la cruz direccional.

```
bool HandlePlayerInput(InputState input, PlayerIndex
playerIndex)
{
  if (!basicControllerFunctions(true, input, playerIndex))
       return false;
     #region inGameControls
     if (gamePadState.Buttons.A == ButtonState.Pressed)
       currentModelAnimation = (int)Player.ModelStates.HitDown;
     if(gamePadState.Buttons.X == ButtonState.Pressed)
       currentModelAnimation = (int)Player.ModelStates.HitLeft;
     if (gamePadState.Buttons.Y == ButtonState.Pressed)
       currentModelAnimation = (int)Player.ModelStates.Jump;
     if (gamePadState.Buttons.B == ButtonState.Pressed)
       currentModelAnimation = (int)Player.ModelStates.HitRight;
     #endregion
  return true;
}
```

CAPITULO 6 – CONCLUSIONES

Este trabajo representa la inversión del conocimiento que adquirió en la Escuela Superior de Cómputo.

A lo largo de su desarrollo se ha aprendido a apreciar también otra cuestión más sutil pero no menos importante que está detrás: nuestra cultura.

Lo que se pretendió fue dar un paso en el rescate de nuestra cultura, mostrando que se pueden iniciar proyectos con base en ella. Es decir, tomando algunos elementos de las culturas del México antiguo, en concreto de la cultura Maya, se creó Pok-Ta-Pok.

Al desarrollar este proyecto se estuvo muy consciente de que la parte importante es la técnica, sin embargo conforme se realizó la investigación sobre cuestiones antropológicas se logró apreciar cuánto ignoramos sobre nuestro pasado.

Esperamos que este esfuerzo sea un detonante en la promoción de nuestro país, el cual puede tener incluso un impacto a nivel mundial.

La cultura es lo que une a un país y le identifica ante el mundo.

CAPITULO 7 – TRABAJO A FUTURO

7.1 – Optimizar tráfico de la red

Se pretende mejorar el envío y recepción de paquetes a través de la red, para reducir al mínimo el retardo que se pueda tener en el juego.

7.2 – Mejora de la Inteligencia Artificial

En esta propuesta de mejora se tiene como objetivo tener un algoritmo para los agentes que se están corriendo en el juego y que tienen a su cargo la I.A. Dicho algoritmo pretende mejorar la toma de decisiones, así como la ruta más corta a su objetivo, en este caso la pelota.

7.3 – Aumento del detalle en los modelos y uso de "Normal Mapping"

En este punto se tiene como objetivo, realzar el nivel de realismo de los modelos del juego, para ello haciendo uso de una técnica de texturizado llamada *normal mapping* que consiste en lo siguiente:

Es la aplicación de una técnica 3D que permite dar una iluminación y relieve mucho más detallado a la superficie de un objeto. Es una evolución del bump mapping y se aplica en videojuegos a los que les dota un mayor realismo, así como en películas de animación para agilizar los cálculos y reducir por tanto el número de polígonos con los que en un principio contaban los objetos.

CAPITULO 8 - REFERENCIAS

[1] Juego de pelota en Mesoamérica, CD- ROM interactivo, Fundación Cultural Armella Spitalier.
[2] JACK: Innovador Video Juego con realidad virtual multidimensional en red, TT 0018, 1998, M. en C. Adrián Alcántar Torres
y M. en C. Marco A, Marván Cabrera
[3] Sistema de videojuego de estrategia evolutivo, TT 0155, 2000, M. en C. Jesús Manuel Olivares Ceja.
[4] AztlánRPG, Un mundo persistente en línea, TT 0768, 2004, M. en C. Marco Antonio Dorantes González, M. en C. Martha Rosa Cordero López
[5] *http://www.gsi.dit.upm.es/~gfer/ssii/trabajos2006/ag4-MFdez-ANavas.pdf*
Desarrollado por el Grupo de Sistemas Inteligentes de la Escuela Técnica Superior de Ingenieros de Telecomunicación[**ETSIT-UPM**], España.
http://www.gsi.dit.upm.es/
[6] Videos disponibles vía youtube.com

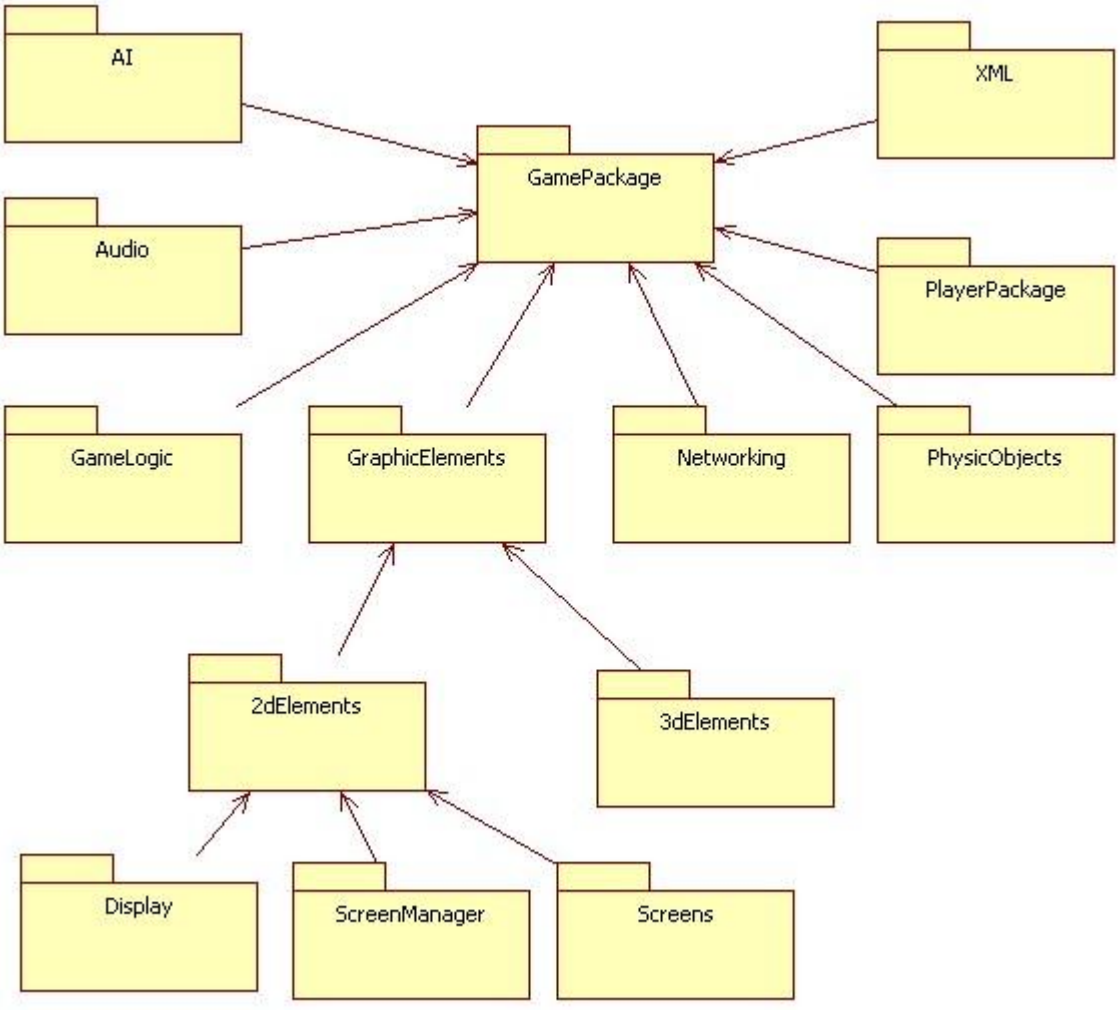
Revista arqueología Mexicana, números:

Chichén Itzá (E27), Guía Visual Edición Especial, abril de 2008, número 27
El juego de pelota (no.44), Volumen VIII. (julio-agosto, 2000)
Los mayas: rutas arqueológicas - Yucatán y Quintana Roo, Edición especial # 21, Mayo de 2006
La pintura maya, (no. 93), Septiembre – Octubre de 2008
Los dioses mayas, (no. 88) Noviembre - Diciembre de 2007.

Mysteries of the Maya, Collector's Edition, National Geographic, 2008

CAPITULO 9 - ANEXO

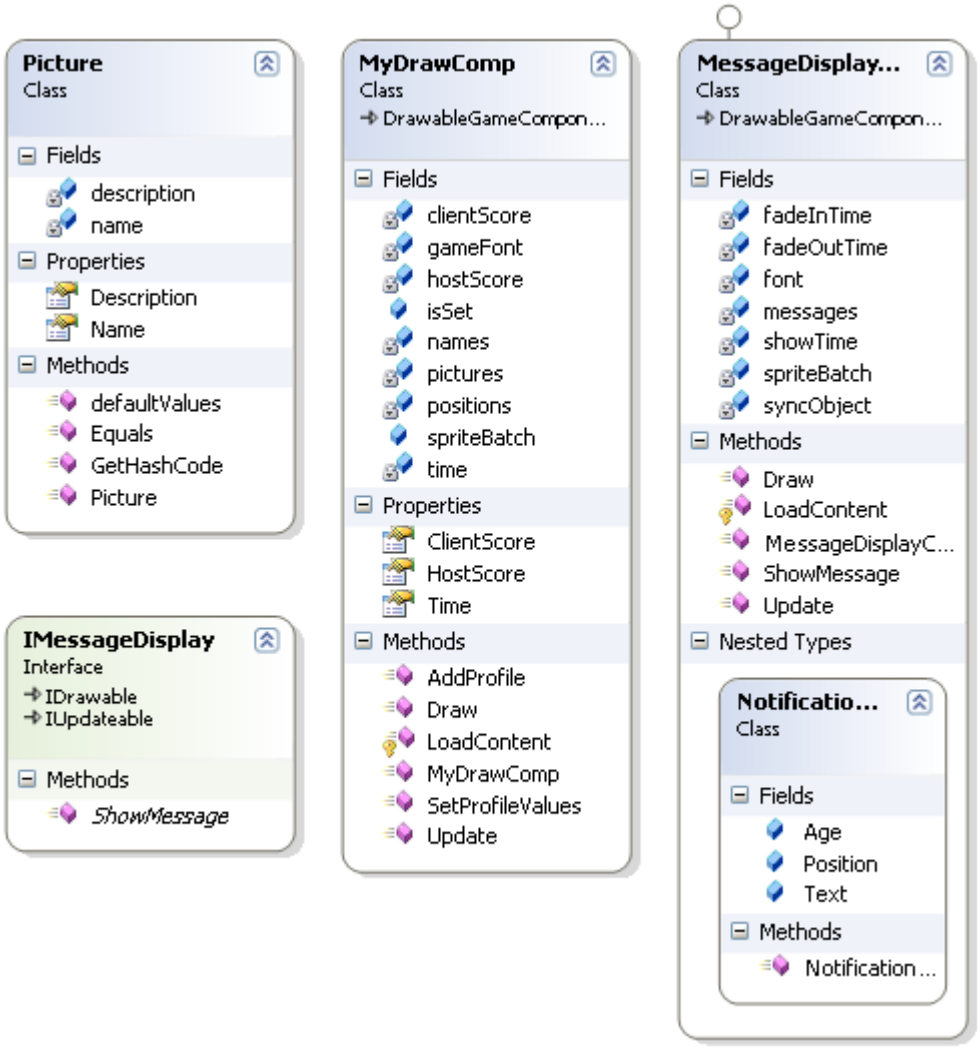
9.1 – Diagrama De Paquetes



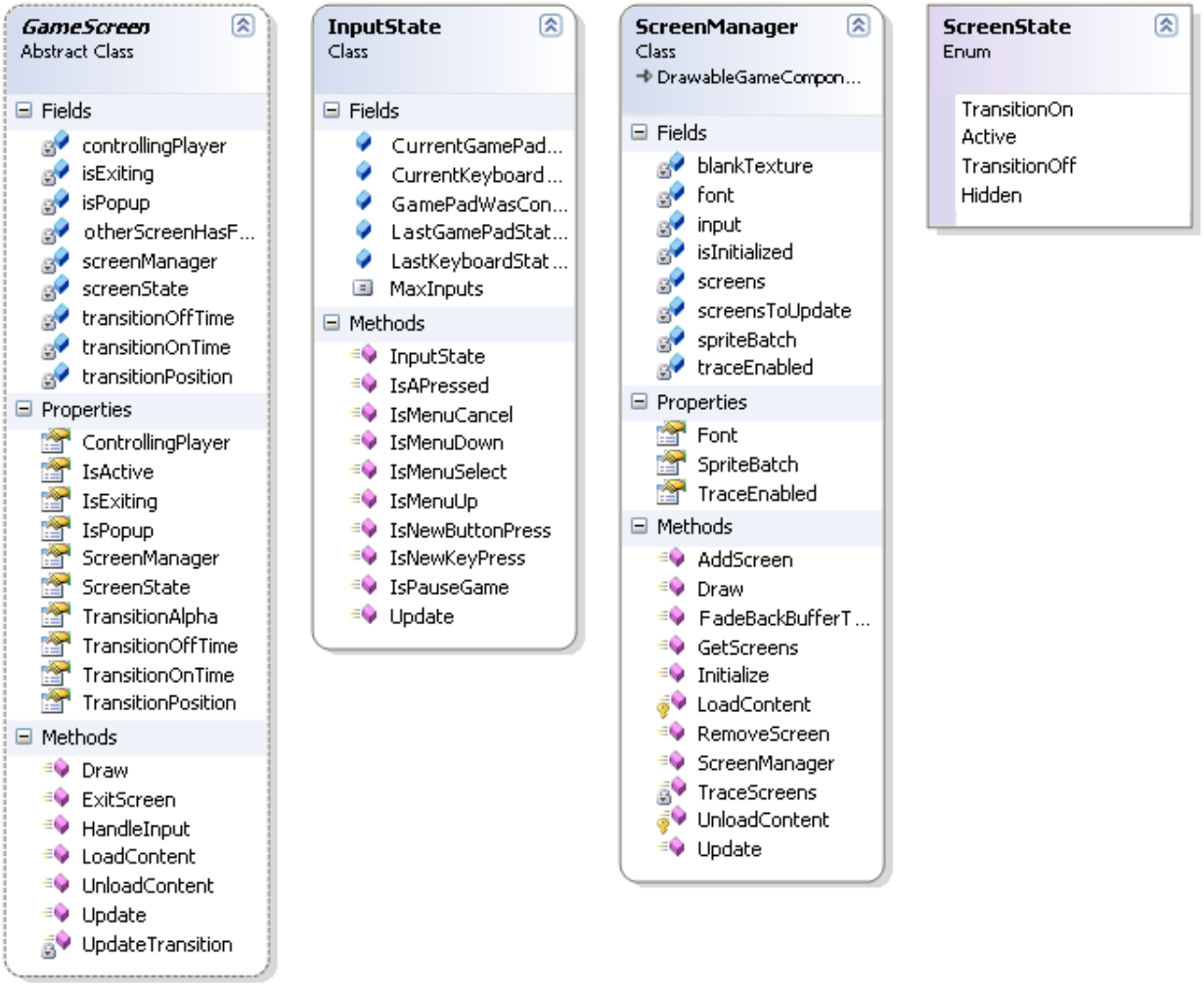
c

9.2 - Diagramas De Clases

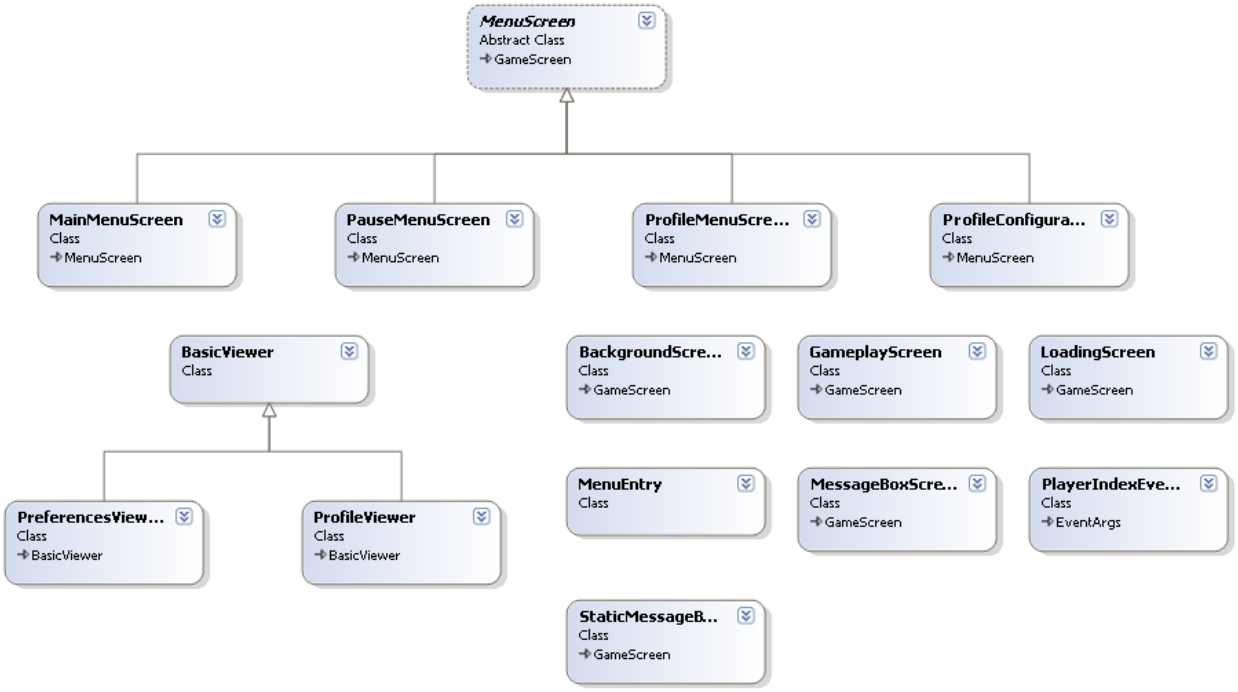
9.2.1 – Display



9.2.2 - ScreenManager

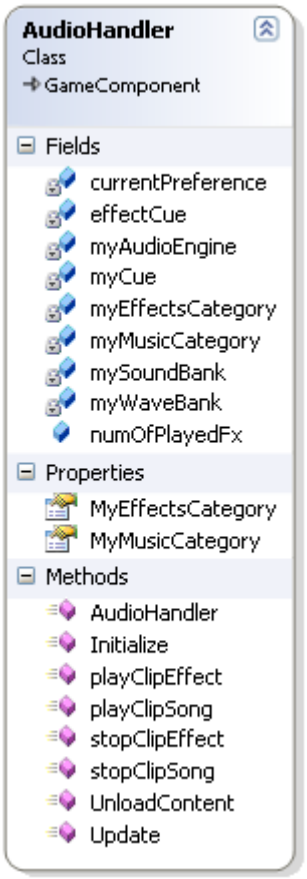


9.2.3 – Screens

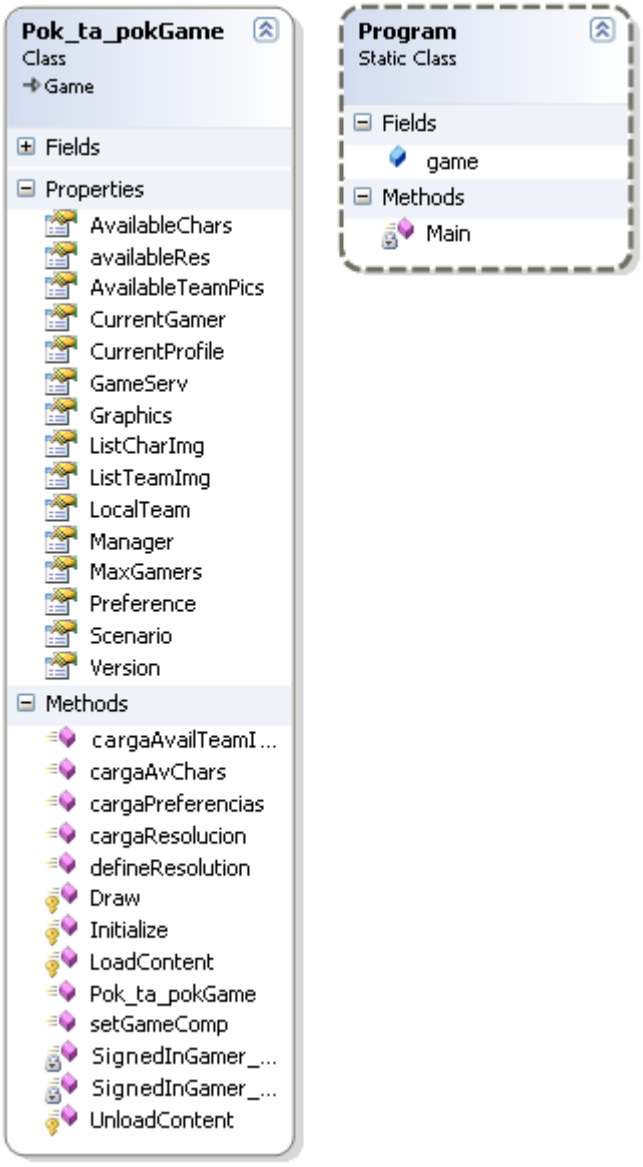


9.2.4 – 3d Elements

9.2.5 – Audio

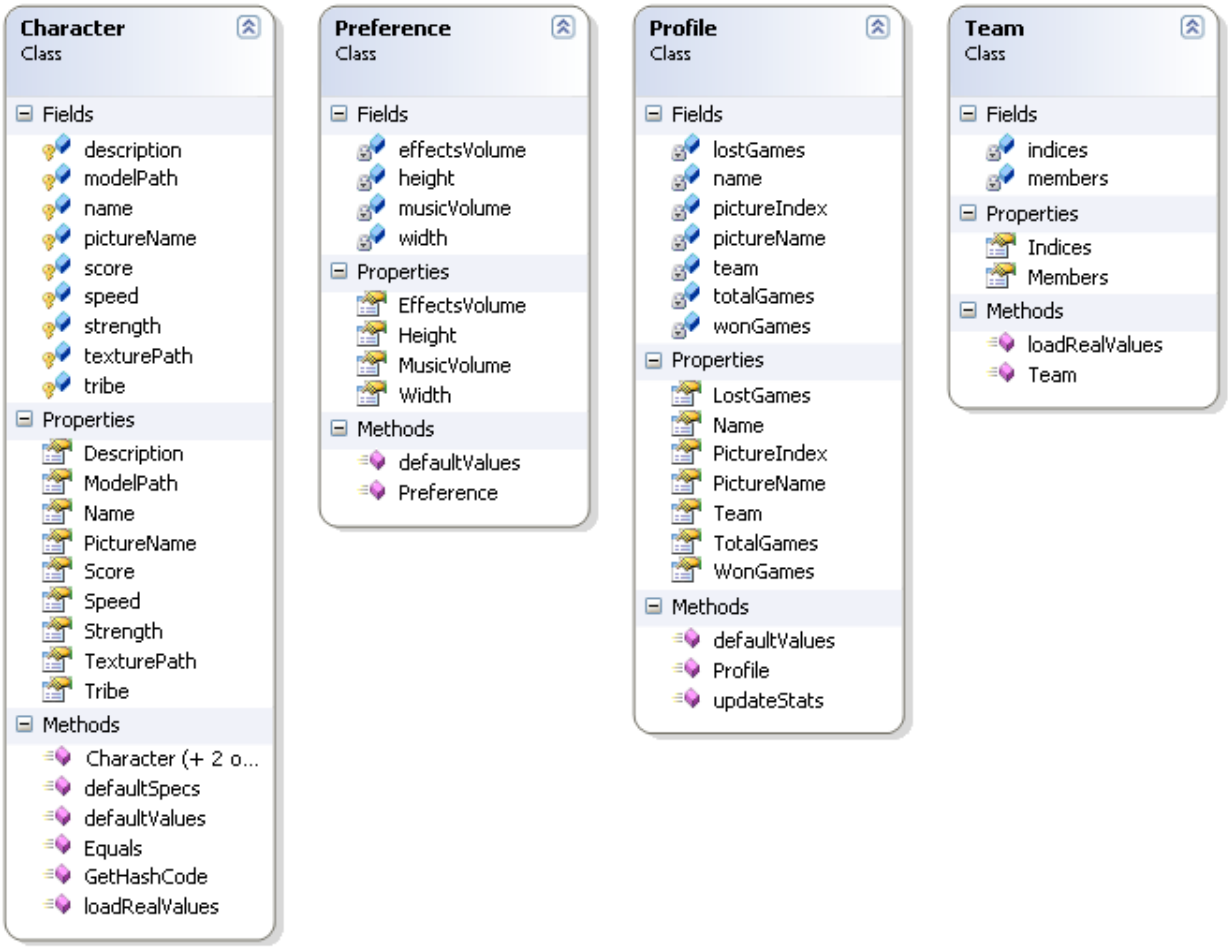

AudioHandler
Class
GameComponent
Fields
currentPreference
effectCue
myAudioEngine
myCue
myEffectsCategory
myMusicCategory
mySoundBank
myWaveBank
numOfPlayedFx
Properties
MyEffectsCategory
MyMusicCategory
Methods
AudioHandler
Initialize
playClipEffect
playClipSong
stopClipEffect
stopClipSong
UnloadContent
Update

9.2.6 – Game

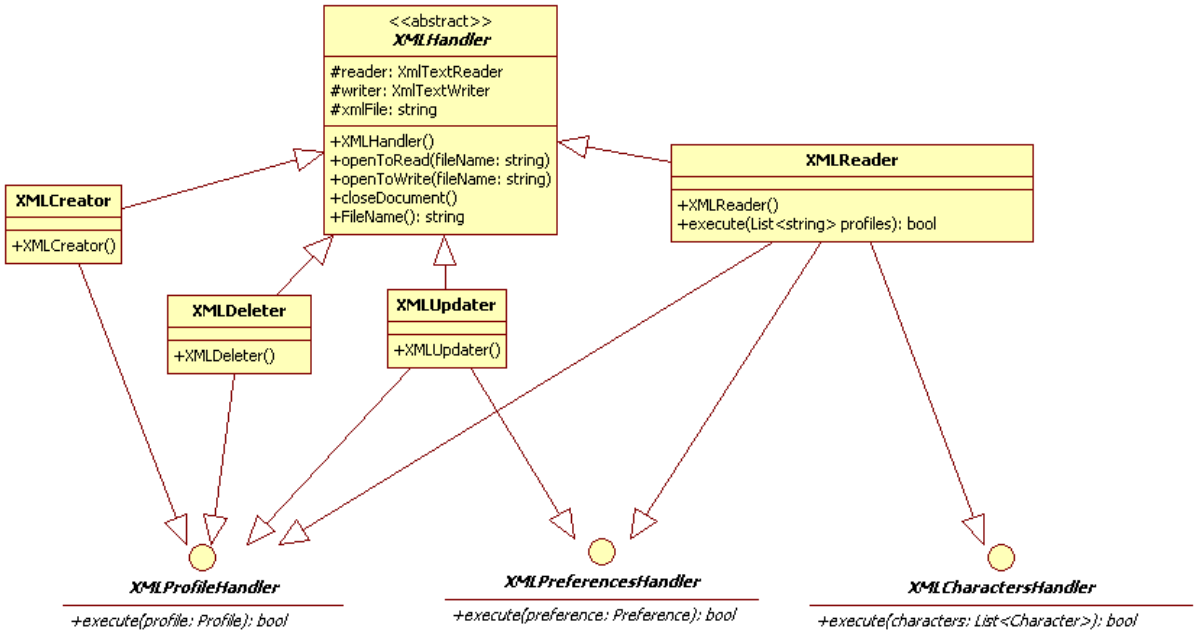

Pok_ta_pokGame
Class
Game
Fields
Properties
AvailableChars
availableRes
AvailableTeamPics
CurrentGamer
CurrentProfile
GameServ
Graphics
ListCharImg
ListTeamImg
LocalTeam
Manager
MaxGamers
Preference
Scenario
Version
Methods
cargaAvailTeamI...
cargaAvChars
cargaPreferencias
cargaResolucion
defineResolution
Draw
Initialize
LoadContent
Pok_ta_pokGame
setGameComp
SignedInGamer_...
SignedInGamer_...
UnloadContent
Program
Static Class
Fields
game
Methods
Main

9.2.7 – Networking

9.2.8 – Player

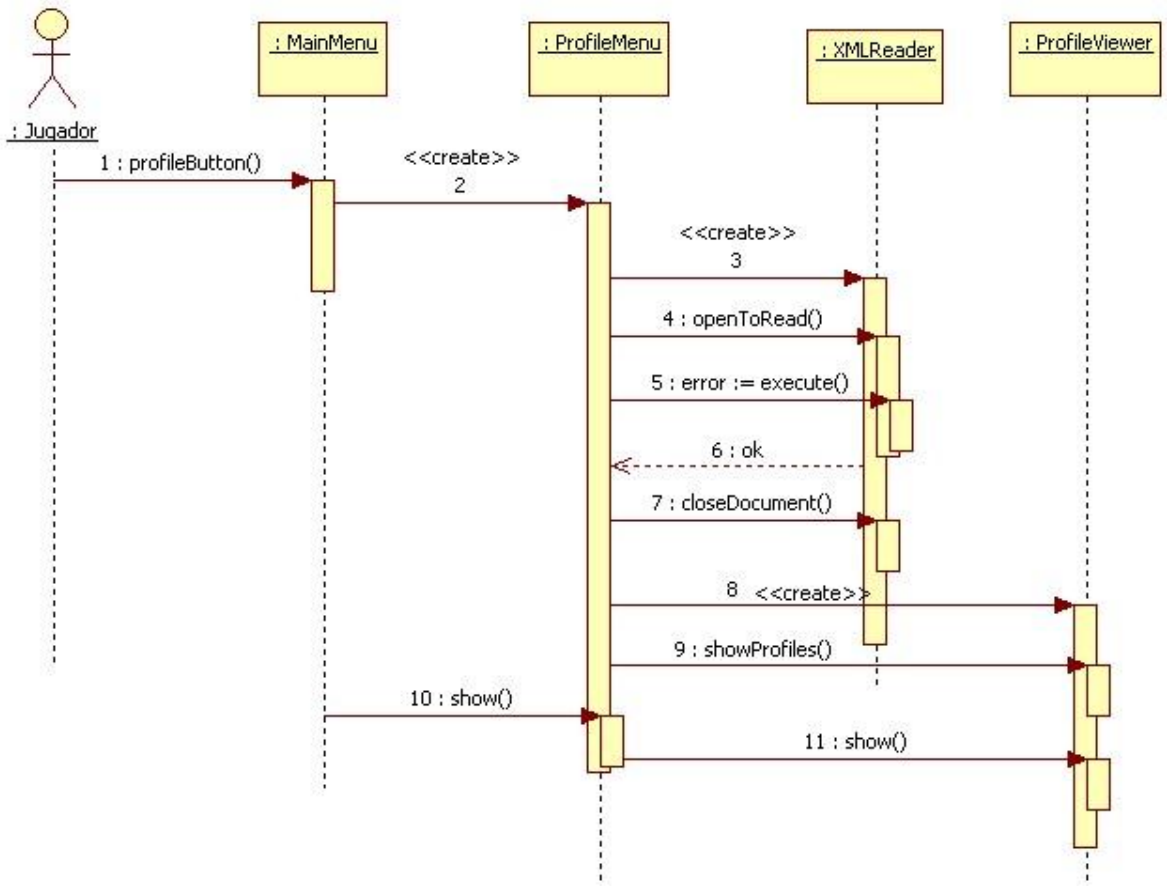

Character
Class
Fields
description
modelPath
name
pictureName
score
speed
strength
texturePath
tribe
Properties
Description
ModelPath
Name
PictureName
Score
Speed
Strength
TexturePath
Tribe
Methods
Character (+ 2 o...
defaultSpecs
defaultValues
Equals
GetHashCode
loadRealValues
Preference
Class
Fields
effectsVolume
height
musicVolume
width
Properties
EffectsVolume
Height
MusicVolume
Width
Methods
defaultValues
Preference
Profile
Class
Fields
lostGames
name
pictureIndex
pictureName
team
totalGames
wonGames
Properties
LostGames
Name
PictureIndex
PictureName
Team
TotalGames
WonGames
Methods
defaultValues
Profile
updateStats
Team
Class
Fields
indices
members
Properties
Indices
Members
Methods
loadRealValues
Team

9.2.9 – Xml

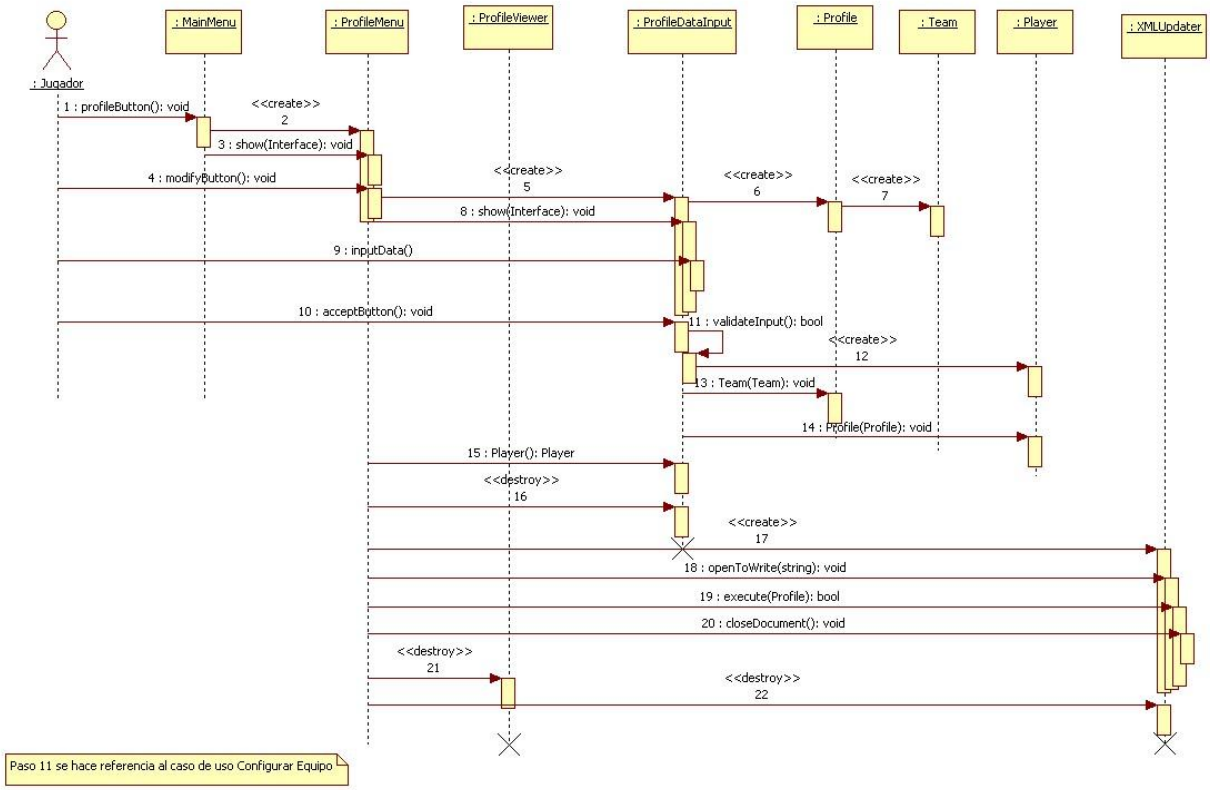

<<abstract>>
XMLHandler
#reader: XmlTextReader
#writer: XmlTextWriter
#xmlFile: string
+XMLHandler()
+openToRead(fileName: string)
+openToWrite(fileName: string)
+closeDocument()
+FileName(): string
XMLCreator
+XMLCreator()
XMLReader
+XMLReader()
+execute(List<string> profiles): bool
XMLDeleter
+XMLDeleter()
XMLUpdater
+XMLUpdater()
XMLProfileHandler
+execute(profile: Profile): bool
XMLPreferencesHandler
+execute(preference: Preference): bool
XMLCharactersHandler
+execute(characters: List<Character>): bool

9.3 – Diagramas De Secuencia

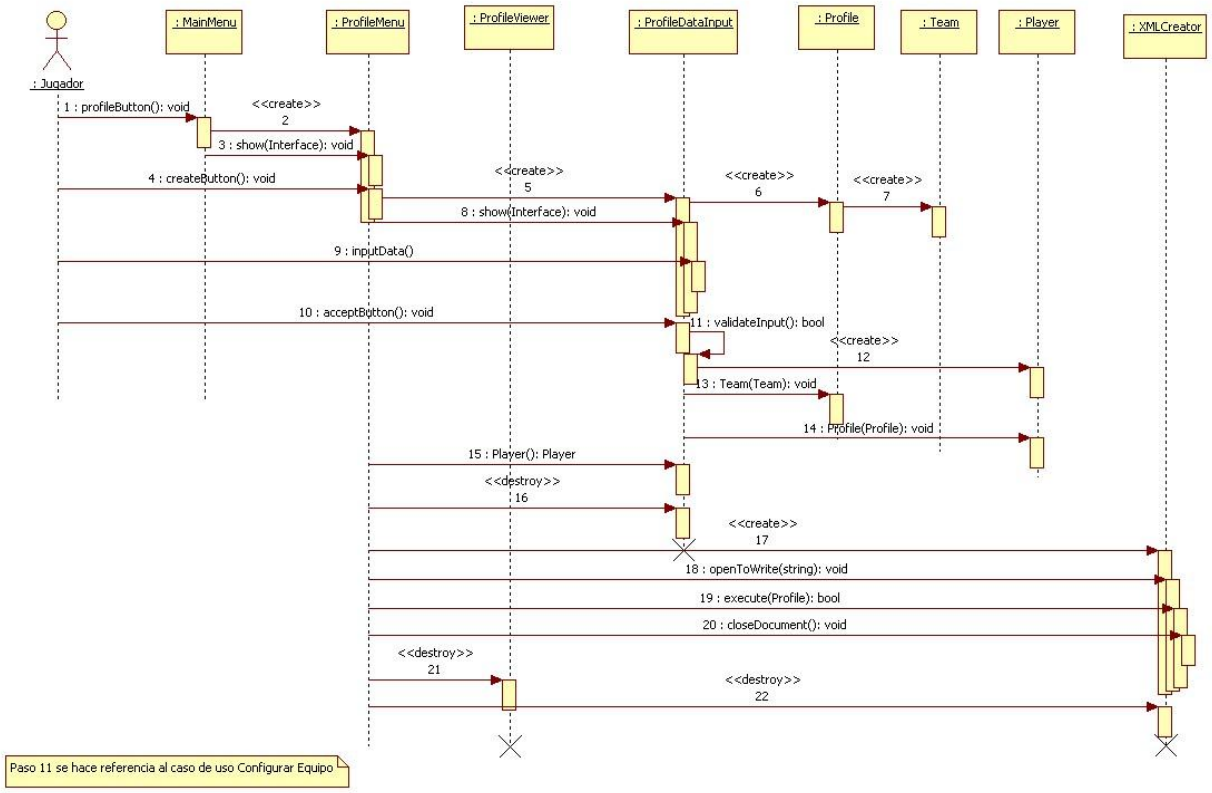
9.3.1 - Leer Perfil



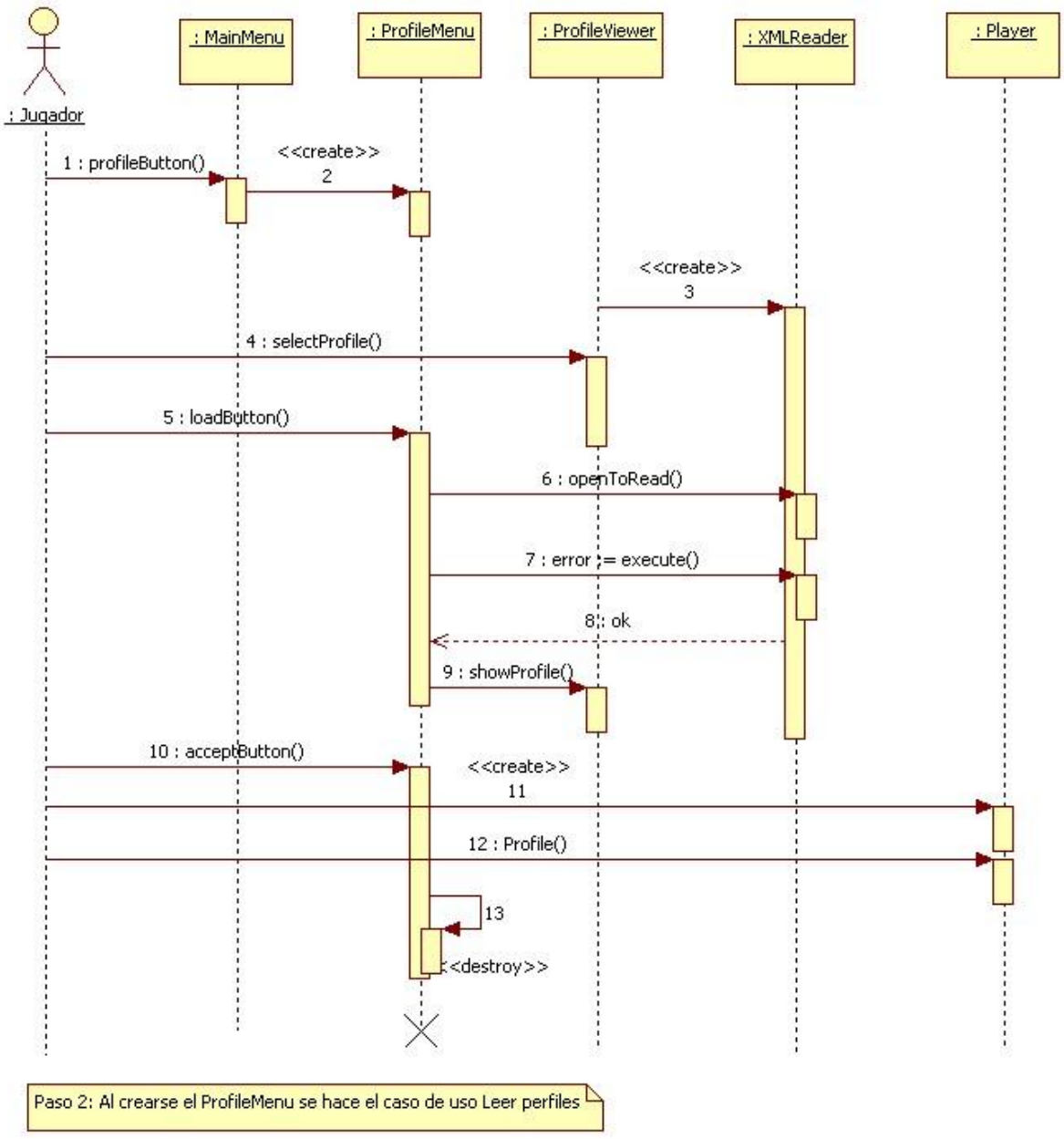
9.3.2 - Modificar Perfil



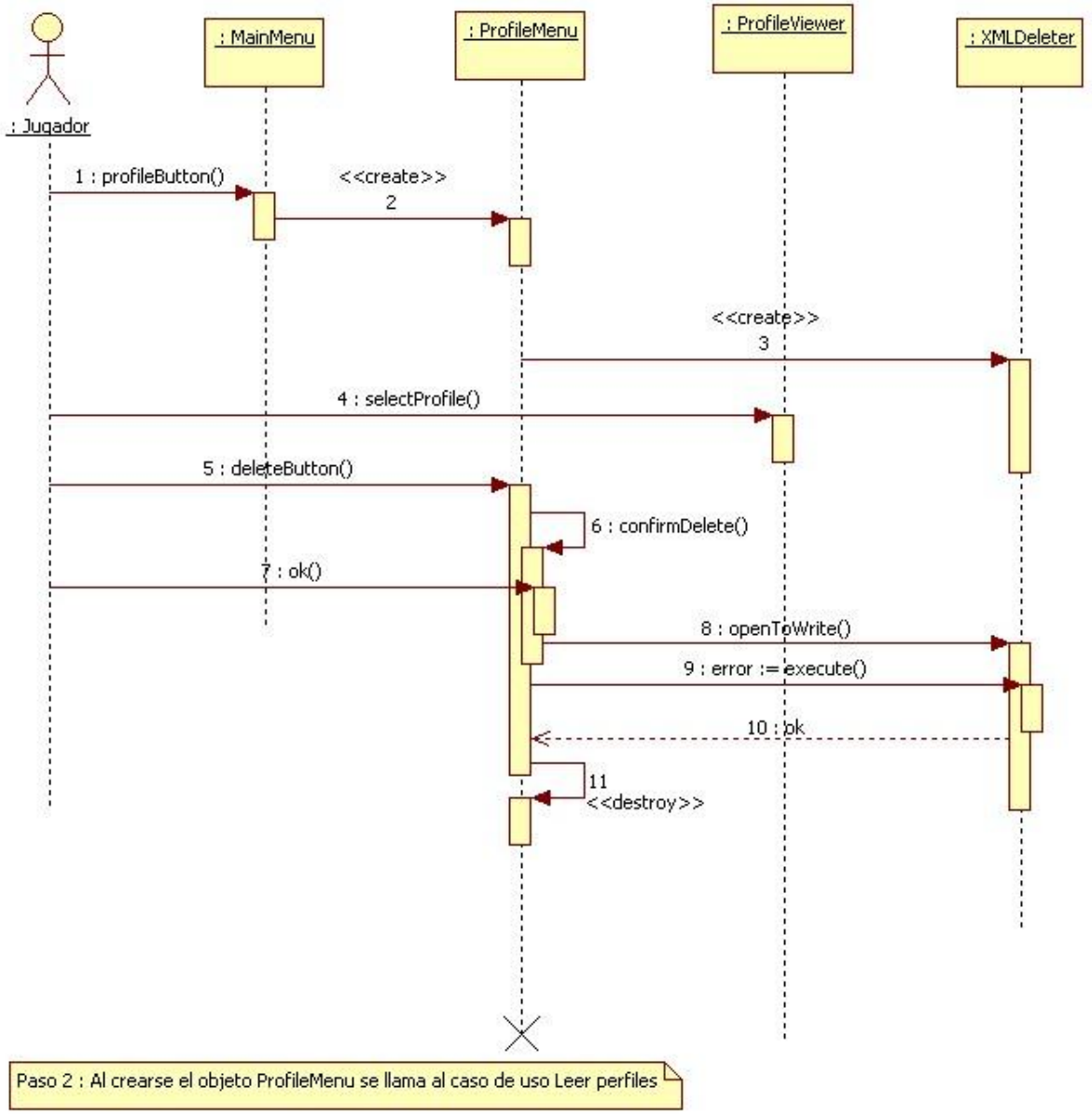
9.3.3 Crear Perfil



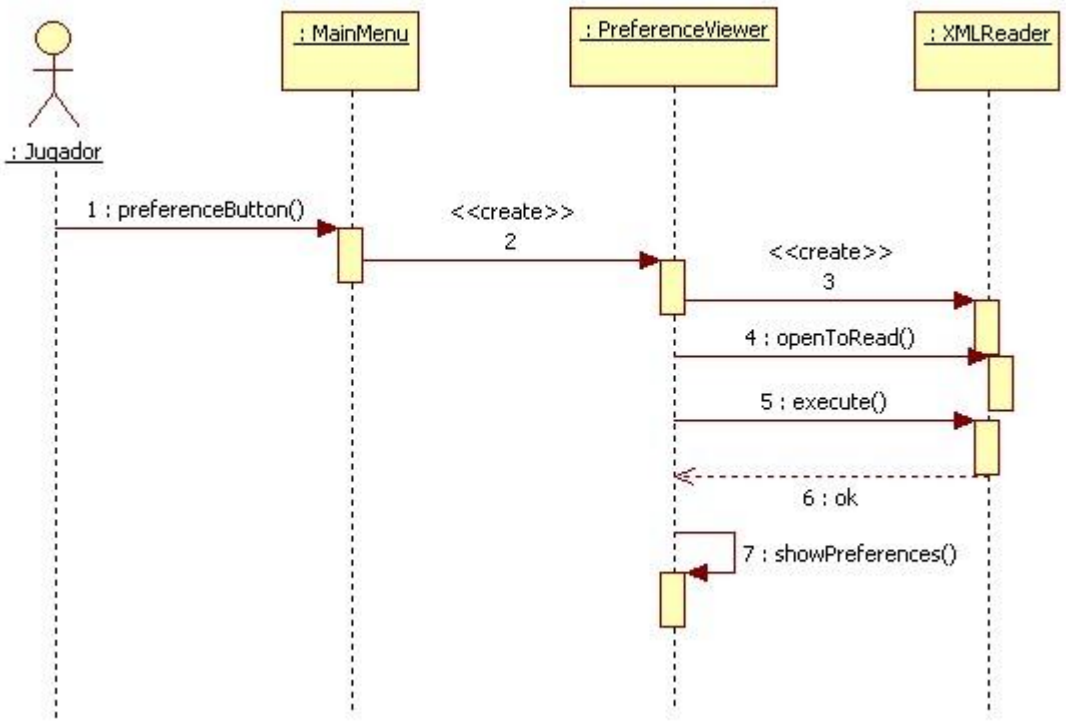
9.3.4 - Cargar Perfil



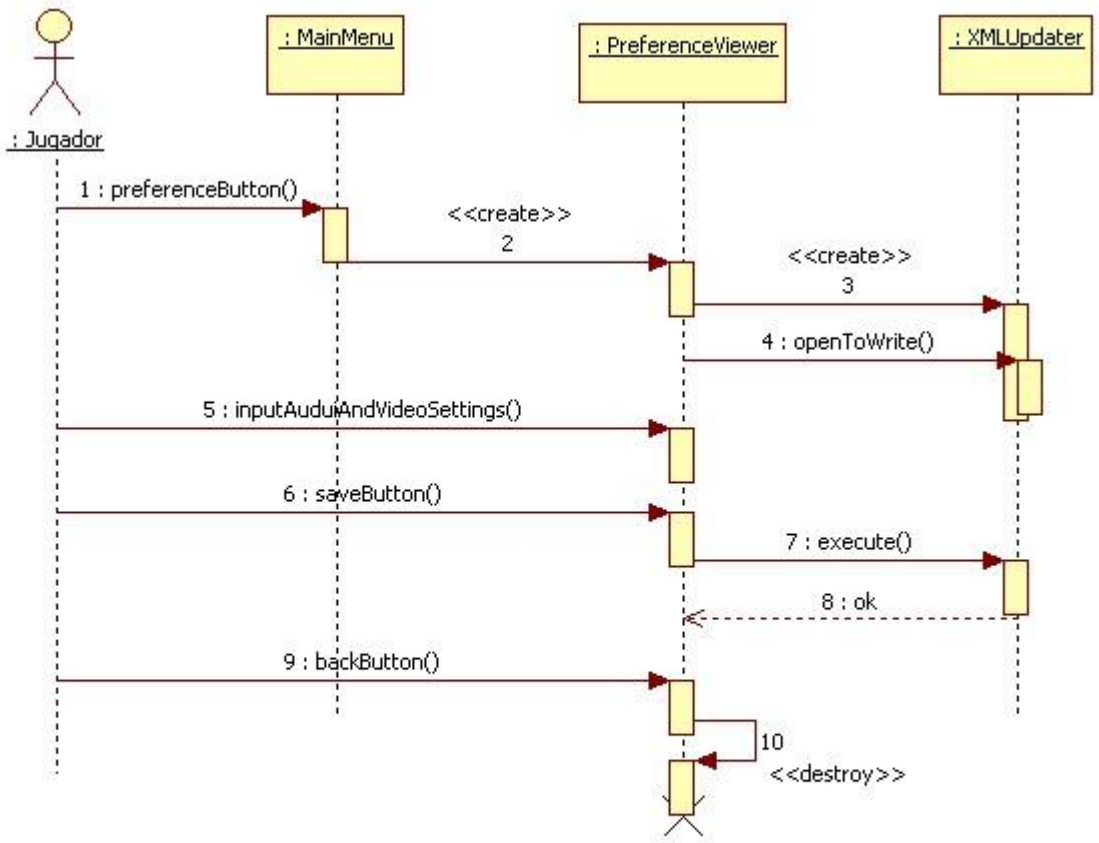
9.3.5 - Borrar Perfil



9.3.6 - Cargar Preferencias

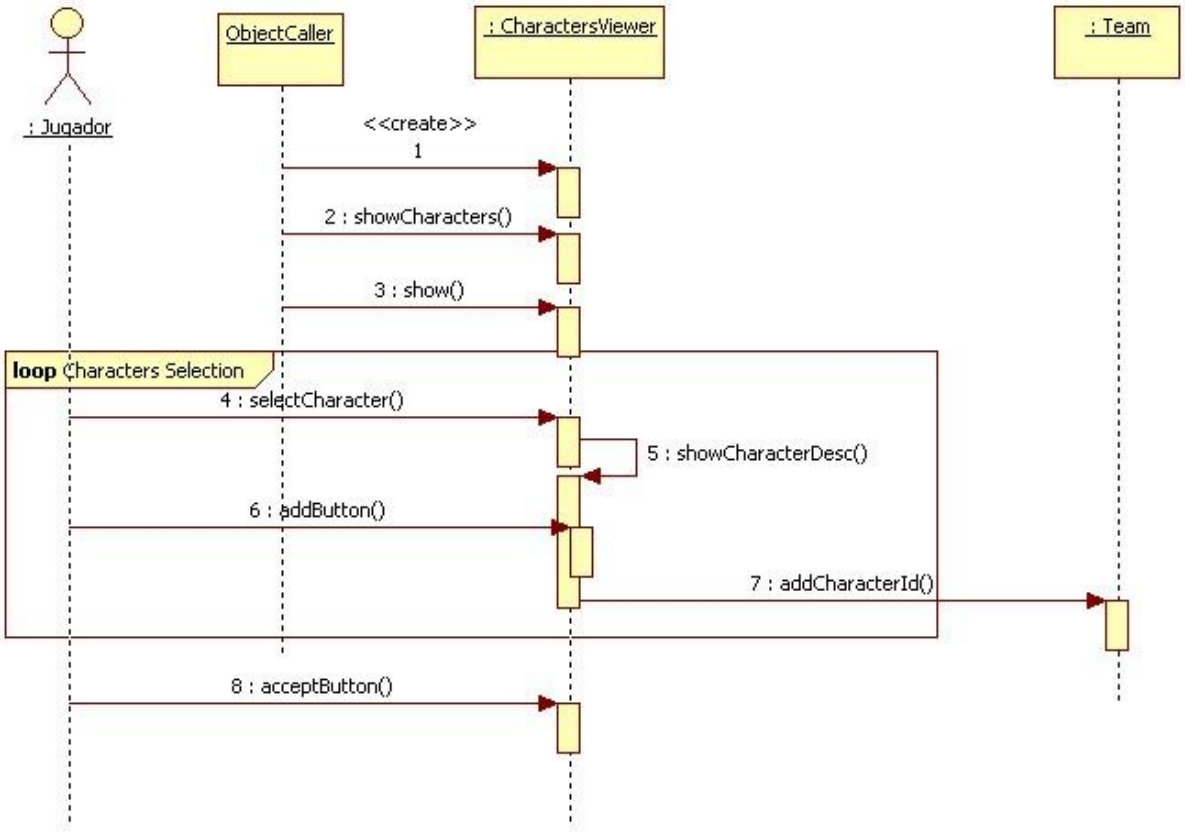


9.3.7 – Configurar Preferencias



Paso: Al crearse el PreferenceViewer se hace referencia al caso de uso Cargar preferencias

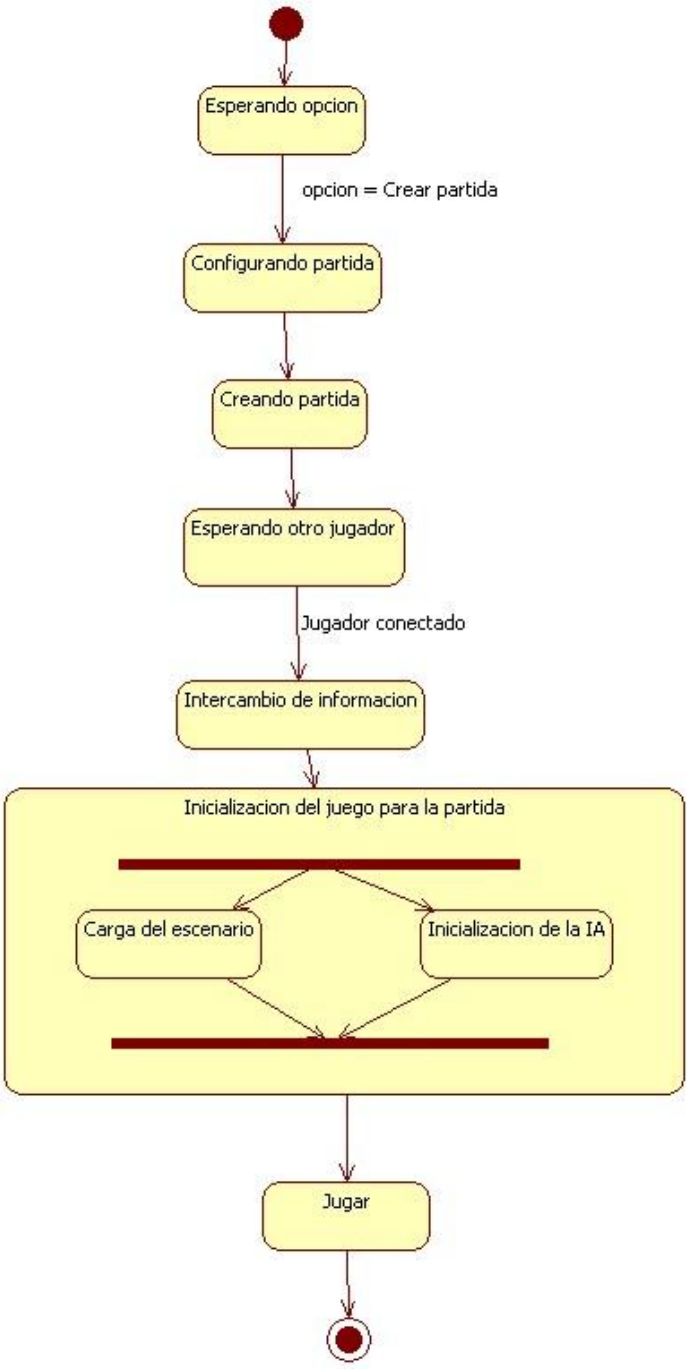
9.3.8 – Configurar Equipo



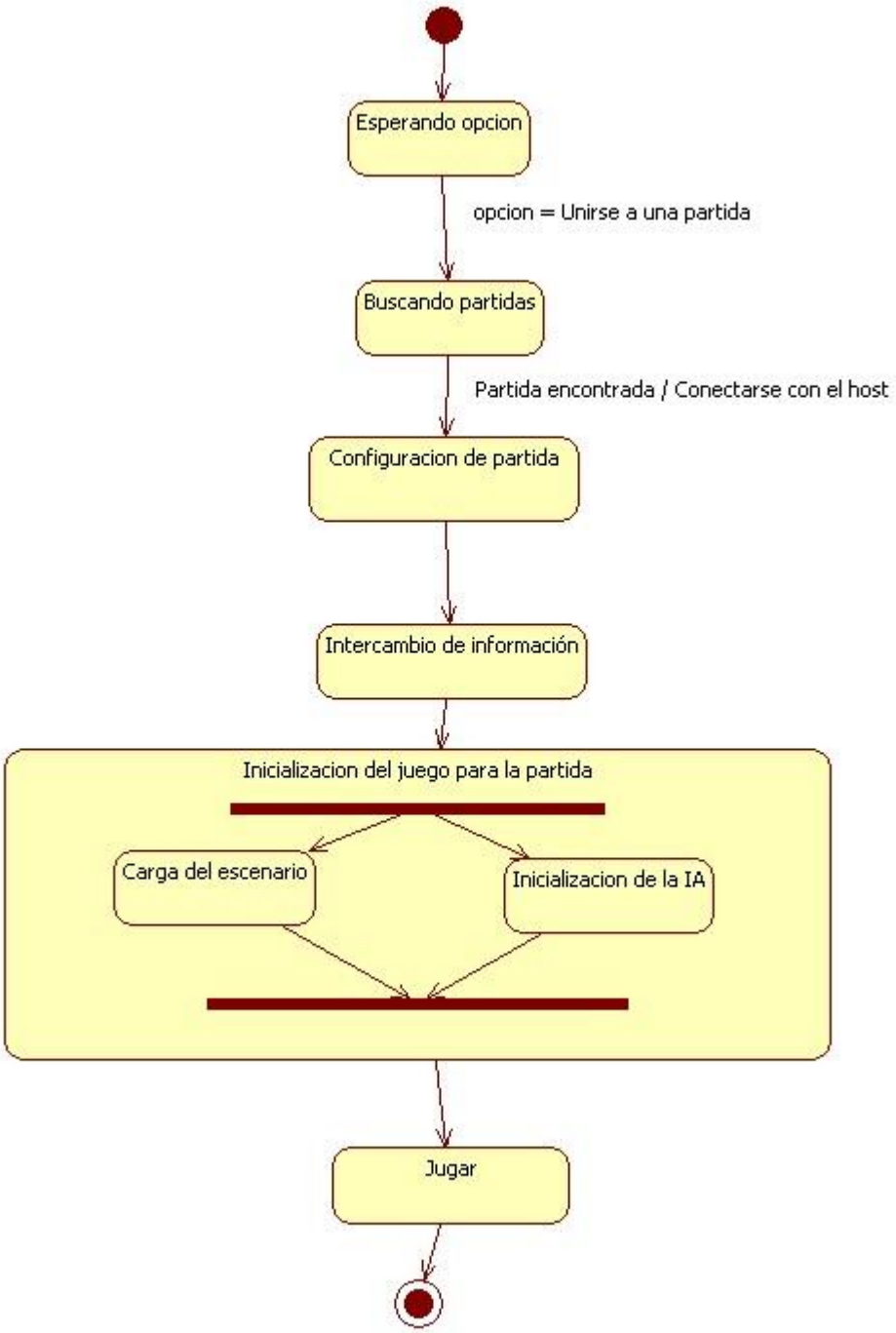
9.4 – Diagramas De Estados

Nos permiten representar los estados básicos por los cuales pasa el sistema en un determinado momento

9.4.1 – Crear Partida


Esperando opcion
opcion = Crear partida
Configurando partida
Creando partida
Esperando otro jugador
Jugador conectado
Intercambio de informacion
Inicializacion del juego para la partida
Carga del escenario
Inicializacion de la IA
Jugar

9.4.2 – Unirse A Una Partida



9.5 - Diseño De Arte Del Juego

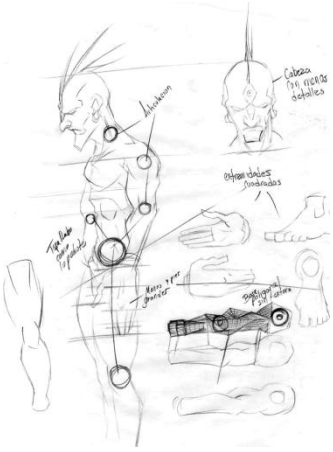
Diseño llamado "Logo No.2", idea para "Splash Screen" o pantalla de Presentación del juego.

"Personaje Principal" perteneciente a la "Tribu de la tormenta", con algún detalle sobre el sello (o logo) que lo identifica como miembro de la tribu

Logo Circular, posible logo general del juego, que permita identificarlo fácilmente a simple vista

Bosquejo de juez (que en realidad podría fungir como árbitro de los partidos).

Versión de “Screen Saver” con marcas de agua del texto “Pok-ta-Pok”



Detalle de personaje especificando la ubicación de huesos (utilizados para animarlo), así como forma de manos, pies y cara.

Bosquejo de la animación del personaje

9.6 – Cronogramas De Actividades

Nombre del alumno: Medina Martínez José Guadalupe

Actividad	AGO	SEP	OCT	NOV	DIC	ENE	FEB	MAR	ABR	MAY
Análisis y diseño del sistema										
Investigación y recopilación de información sobre el juego de pelota										
Diseño y modelado de personajes y escenarios.										
Evaluación de TT I.										
Generación del código.										
Creación de servidor para administración de partidas en red.										
Integración de la aplicación										
Reingeniería										
Generación del Manual de Usuario										
Generación de Reporte Técnico.										
Evaluación de TT II.										

Nombre del alumno: Montaño Ordaz Jesús Adrián

Actividad	AGO	SEP	OCT	NOV	DIC	ENE	FEB	MAR	ABR	MAY
Análisis y diseño del sistema										
Síntesis de la información obtenida										
Análisis de la física necesaria										
Evaluación de TT I.										
Análisis de la inteligencia artificial										
Generación del código.										
Pruebas de comunicación con servidor										
Integración de la aplicación										
Reingeniería										
Generación de página Web										
Generación de Reporte Técnico.										
Evaluación de TT II.										

CAPITULO 10 - GLOSARIO

Billboard – Plano de 2 dimensiones ubicado en un mundo tridimensional, que siempre esta orientado a la cámara

Gameplay – Termino designado para referirse a la presencia de acción o movimiento dentro del juego como resultado de acciones del usuario.

GUI – Graphic User Interface

Input – Termino designado para referirse a la entrada de comandos al sistema a través de algún dispositivo externo

Mesh – Malla creada a partir de la unión de puntos para la generación de un modelo tridimensional

Shader – Tecnica de programación consistente en el uso de directivas de procesador especificas de la tarjeta grafica