DISEÑO Y DESARROLLO DE UN DEMO DE UN JUEGO MULTIMEDIA PARA EL AUTOAPRENDIZAJE DEL PROYECTO EDUCATIVO INSTITUCIONAL DE LA UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA

AUTORES
YESID ALBERTO CARO, YESIKA LABORDE FORERO

DIRECTOR
MSC. ROMÁN EDUARDO SARMIENTO PORRAS

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA
FACULTAD DE INGENIERÍA DE SISTEMAS
GESTIÓN DEL CONOCIMIENTO E INGENIERÍA DE SOFTWARE

INTRODUCCIÓN

Según el PEI de la Universidad Autónoma de Bucaramanga en su resolución numero 276, se plantea como la guía de referencia para orientar la labor de la institución, en los diferentes campos de actuación: dirección, quehacer universitario y soporte institucional. Dada la importancia del PEI en la UNAB, se hace necesario que toda la comunidad universitaria: estudiantes, docentes y personal administrativo, tenga conocimiento de él y lo ponga en práctica.

Pero la realidad es otra, según la investigación realizada a lo largo de este proyecto de grado y el Informe de Evaluación Externa del Consejo Nacional de Acreditación elaborado en el 2002, el Proyecto Educativo Institucional (PEI) en su trayectoria ha tenido un duro desarrollo debido a que la comunidad académica ha mostrado poco interés en el aprendizaje y desarrollo del mismo, por lo cual se ha visto en la obligación de encontrar nuevas formas de hacer conocer el PEI y que a su vez la comunidad académica se muestre interesada. Se han planeado diversos métodos de enseñanza pero, sin ningún resultado satisfactorio.

1. ANTECEDENTES

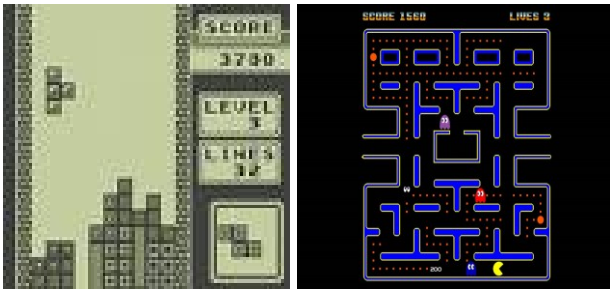
Para llevar a cabo el estudio de video juegos, es de suma importancia como primer paso investigar y profundizar acerca de la evolución que en este tema se ha tenido a través del paso de los años. Para comenzar hablar del mundo de los videojuegos, es necesario realizar una breve reseña histórica:

El desarrollo de los videojuegos en multimedia se ha venido trabajando desde hace tiempo. Esta historia se remonta a la época en que el creador del radar pensó que se dejaba al usuario excluido en los juegos que para ese entonces se comercializaban; entonces, para generar una mayor interacción desarrolló la primera consola basándose en los fundamentos del radar, este primer juego fue llamado “telebolito”

Este juego causó curiosidad en la comunidad de programadores, quienes observaron otra forma de usar sus habilidades. Algunos de los interesados empezaron a desarrollar nuevos juegos y nuevas formas para la creación de otros ambientes.

Lo anterior llevó a la creación de escenarios y variables para poner a prueba las habilidades de los usuarios incrementalmente. De ahí nació el Arcanoid, Tetris, Granarix, para mayor información ver figura 1. Después con la aparición de las pantallas a color, dio otro salto el desarrollo de juegos apareciendo el famoso Pacman, etc.

Figura 1. Primeros juegos diseñados.



Fuente. Quest3D. Historia de los videojuegos. [En línea]. 2000. [Citado el 17 de agosto del 2006]. Disponible en Internet en: <http://www.quest3d.com/index.php?id=17>

Figura 2. Vista de los juegos hoy en día.

Fuente. IGN ENTERTAINMENT GAMES. Desarrollo de juegos. [En línea]. 1996. [Citado el 26 de Agosto del 2006]. Disponible en Internet en: <http://www.ign.com/>

A medida que se introducían los video juegos en el mundo y se incrementaban las habilidades de las personas que los jugaban, se pensó en utilizar este medio como una forma de enseñar o educar a las personas, un ejemplo de estos son los Simuladores de Vuelo que utilizan las Fuerzas Aéreas para enseñar a sus pilotos y Simuladores de Combate para los soldados que los prepara para la guerra. Pero estas no son las únicas aplicaciones; también hay juegos de aprendizaje para hablar con los niños menores de edad y los juegos para la instrucción de personas que toman un nuevo empleo y necesitan aprender las normas y reglas de la institución.[1]

Esta clase de simuladores o juegos utilizados para la enseñanza de reglas y habilidades se llama juegos instruccionales. Esta clase de juegos se vienen desarrollando desde la aparición del computador, ya que ha revolucionado la enseñanza. Gracias a la aparición de estos juegos se introducen diferentes recursos de gran utilidad en la educación. Entre estos se encuentran los programas de ejercicios - práctica y tutórales, estos brindan la oportunidad de aprender de una manera variada. Los juegos instruccionales son actividades basadas en la computadora con imágenes y animaciones que producen una motivación. Entre los juegos instruccionales más usados se encuentra: ¿"Where in the World is Carmen San Diego?" Las simulaciones por computadora ayudan al estudiante a entender situaciones del mundo real.

[1] CANOS, José; LETELIER, Patricio Y PENADES, Mª Carmen. Metodologías Ágiles en el Desarrollo de Software, DSIC - Universidad Politécnica de Valencia. Alicante, España. Noviembre 2003. Editado Por: Patricio Letelier Torres Emilio A. Sánchez López, Grupo ISSI. 59 P

2. MARCO CONCEPTUAL

2.1 PROYECTO EDUCATIVO INSTITUCIONAL (PEI)[2]

EL PEI es la orientación de toda labor académica de la institución para su formulación y mejoramiento en la continua construcción de programas y profesionales con la participación de toda la comunidad académica para su mejoramiento. Es de suma importancia realizar un estudio profundo del PEI para desarrollar el juego adecuado que permita a la comunidad universitaria apropiarse de estos conocimientos.

La comunidad académica tiene como puntos básicos dar a conocer, promulgar, promover el estudio y aplicación del PEI en todas las formas tanto docencia como los diferentes planes de estudio. Pero el PEI ha tenido un desarrollo lento pero conciso.

El desarrollo del PEI comenzó a partir de 1991 - 1993 bajo el consentimiento de la comunidad académica, con el fin de actualizar sus formas de educar a los estudiantes, basándose en los pilares con los que se han formado las universidades que tienen como fin formar y educar pensando en el futuro del mundo, que es el cultivo de la ciencia subordinada a el desarrollo humano.[3]

Para llegar al nivel de desarrollo humano junto al nivel científico deseado sea ha venido trabajando desde 1994 con dos responsabilidades: la investigación y la accesoria curricular. La primera fue hecha desde el punto de investigación: Acción y Participación.

Hoy en día después de la primera publicación del PEI en 1999, se ha vuelto objeto de investigación y discusión por parte de la comunidad académica en los diversos seminarios y encuentros de facultades, los cuales han ayudado a ir elaborando y actualizando el PEI para un mejor desarrollo de las instituciones.

Hasta ahora se han utilizado juegos educativos didácticos y otros métodos para promover el estudio del PEI, pero no ha sido fructífero pues sigue el mismo desinterés por parte de la comunidad académica. Por este motivo se plantea otra forma para dar a conocer el PEI, un videojuego de auto aprendizaje. Lo cual es una de las formas mas novedosas que se están utilizando hoy en día.

El paso a seguir en esta investigación es conocer los aspectos y detalles del PEI que serán relevantes para el desarrollo del juego, como los son el reglamento docente y estudiantil.[4]

2.2 REGLAMENTO DOCENTE Y ESTUDIANTIL DE LA UNAB[5]

Para comprender el PEI de una manera más fácil, se hace necesario conocer los reglamentos que establece la Universidad Autónoma de Bucaramanga a nivel de Docencia y Estudiantil que a continuación se encontrarán.

2.2.1 Derechos del Profesor

- El ejercicio de todos los derechos y garantías consagrados en la Constitución Política de Colombia.

---

[2] Universidad Autónoma de Bucaramanga. Proyecto Educativo Institucional. 2005

[3] IGN ENTERTAINMENT GAMES. Desarrollo de juegos. [En línea]. 1996. [Citado el 26 de Agosto del 2006]. Disponible en Internet en: <http://www.ign.com/>

[4] EXEBlog. Categoría de juegos. [En línea]. 2006. Valencia, España. [Citado el 22 de Agosto del 2006]. Disponible en Internet en: <http://www.exelweiss.com/blog/categoria/juegos-educativos/>

[5] Universidad Autónoma de Bucaramanga. Reglamento docente y estudiantil. 2005

- El pleno ejercicio de la libertad de enseñanza, aprendizaje, investigación y cátedra de acuerdo con la Ley Fundamental, los Estatutos y los Reglamentos de la Universidad.

- Participar en la elaboración de propuestas en materia académica, acordes con las políticas institucionales.

- Participar en los planes de capacitación y mejoramiento pedagógicos, científicos y técnicos, de acuerdo con los planes y políticas de la Institución y del campo de formación del profesor.

- Elegir y ser elegido para las representaciones profesorales ante el Consejo Académico, los Consejos de Facultad, y otros en los que la Universidad establezca representación del profesorado.

- Recibir tratamiento respetuoso por parte de todos los miembros de la comunidad académica.

- Participar y usufructuar de la propiedad intelectual y derechos de autor, en su propiedad intelectual, conforme a las prescripciones legales y a los reglamentos de la Universidad.

- Disponer de los medios necesarios para la realización de la actividad académica, en condiciones de calidad, eficiencia, y seguridad industrial.

- Recibir oportunamente la retribución en dinero y el reconocimiento que le correspondan conforme a su categoría dentro del escalafón, y a las disposiciones legales vigentes que rijan la modalidad contractual de su vinculación.

- Solicitar la promoción en el escalafón docente de acuerdo con lo establecido en el presente Reglamento.

- Disfrutar de las licencias y permisos solicitados con causa justificada.

- Gozar de las actividades de bienestar universitario ofrecidas por la Universidad. m) Gozar de los estímulos consagrados en éste Reglamento y de los que adicionalmente pueda reconocer la Universidad.

- Conocer y hacer parte del proceso de evaluación de su desempeño: ser notificado oportunamente del resultado, y utilizar los recursos de reposición y apelación que sean del caso.[6]

2.2.2 Deberes de los Profesores

- Conocer y cumplir las obligaciones que se derivan de la Constitución Política de Colombia, de las leyes de la República, de los Estatutos y reglamentos de la Universidad.

- Observar las normas inherentes a la ética profesional y a su condición de docente.

- Cumplir su compromiso con la Misión y el Proyecto Educativo de la Universidad.

- Desempeñar con responsabilidad y eficiencia las funciones inherentes a su ejercicio docente y a las actividades que le sean confiadas.

- Participar en el proceso de evaluación integral de aprendizaje del estudiante, e informar oportunamente sus resultados a los alumnos y a la Facultad correspondiente.

- Cumplir el horario convenido en el contrato celebrado con la Universidad.

- Elaborar, presentar y actualizar oportunamente, los programas de las asignaturas a su cargo, y desarrollarlos de acuerdo con los lineamientos definidos por las Facultades y la Universidad.

---

[6] CONITEC DATASYSTEMS. Motor A6. [En línea]. [Citado el 20 de marzo del 2007]. Disponible en Internet en: <http://www.conitec.net/german/gstudio/3dgs2.htm.>

- Dar tratamiento respetuoso a todos los miembros de la comunidad académica.
- Preservar las instalaciones, equipos y elementos de apoyo académico de la Universidad y, responder por los daños y pérdidas de los bienes confiados a su guarda o administración.
- Coordinar la actividad académica con los profesores de la misma asignatura atendiendo criterios de gestión curricular que permitan articular sus contenidos con las líneas, niveles y soportes pertinentes.
- Participar en los grupos de trabajo que le sean asignados en desarrollo de los programas y planes institucionales.
- Contribuir, con elevado ejercicio de sus responsabilidades académicas, al buen uso, a la guarda, engrandecimiento del nombre y del patrimonio cultural, científico, técnico, social y físico de la Universidad.
- Fomentar la educación para la conservación de los recursos naturales y del ambiente.
- Participar en las actividades de perfeccionamiento docente y de capacitación profesional.
- Respetar los derechos de producción intelectual, la propiedad industrial y derechos de autor que correspondan a la Universidad, o a terceros, de acuerdo con la Ley y las normas institucionales.
- Ejercer la actividad académica con dinamismo intelectual, respetando diferencias de credos e ideologías de los educandos.
- Participar en el proceso de evaluación de su desempeño académico.
- Participar activamente en la construcción y actualización de su portafolio docente.[7]

2.2.3 Derechos del estudiante

- Conocer el Proyecto Educativo Institucional y las implicaciones que de él se deriven.
- Recibir la educación integral propuesta en el currículo institucional y en los planes de estudio de cada programa.
- Conocer oportunamente la guía de cátedra, el resultado de su evaluación académica y de su proceso formativo.
- Exponer y discutir en libertad las ideas, teorías y conocimientos, reconociendo el pluralismo ideológico, la diversidad de los saberes y la particularidad de las formas culturales.
- Presentar y obtener respuesta oportuna a solicitudes dirigidas a la autoridad competente.
- Utilizar los espacios formativos y demás materiales e implementos educativos que la UNAB ofrece para el desarrollo de sus competencias.
- Participar responsablemente en la evaluación del desempeño docente, los procesos y los programas académicos.
- Elegir y ser elegido como representante de los estudiantes en los organismos de dirección de la UNAB en los cuales tengan participación: Junta Directiva, Consejo Académico, Consejo de Escuela, Comité Curricular y Comité Estudiantil de Facultad, en conformidad con las directivas expedidas por la Rectoría.

[7] MARQUEZ RAMOS, Ángel. La sociedad ante los avances tecnológicos, marzo del 2006, Universidad Interamericana de Puerto Rico, Recinto de Aguadilla Departamento de Ciencias y Tecnología. Puerto rico, 10P

- Ejercer control de gestión y de resultados sobre sus representantes.
- Recibir estímulos como reconocimiento a su desempeño académico, deportivo, cultural y liderazgo estudiantil, y obtener de la UNAB su certificación.
- Ser informado y participar en las convocatorias para becas, pasantías, programas de intercambio que instituciones de orden nacional e internacional ofrezcan a la UNAB.
- Representar a la Universidad o a otros organismos reconocidos en eventos académicos, científicos, artísticos, culturales o deportivos y obtener el correspondiente permiso académico.
- Recibir copia del reglamento estudiantil al inicio de su carrera.

2.2.4 Deberes del estudiante.

- Cumplir los estatutos, reglamentos y demás normas de la UNAB.
- Asistir puntualmente a clase y demás actividades académicas programadas en el respectivo currículo.
- Asumir los compromisos derivados de su matrícula.
- Informarse permanentemente de los resultados de su proceso académico.
- Usar debidamente el carné que lo acredita como estudiante UNAB y abstenerse de utilizar el nombre de la Universidad sin autorización o en forma indebida.
- Participar en los eventos académicos, culturales, recreativos y deportivos que programe la UNAB.
- Mantener el nivel académico exigido por la UNAB.
- Participar en los procesos de evaluación institucional y de docentes.
- Cuidar y mantener en buen estado las instalaciones de la UNAB, los bienes de uso de la comunidad universitaria y responder por los daños que ocasione.
- Poner en conocimiento de la autoridad institucional competente la conducta de cualquier miembro de la UNAB que por acción, omisión o extralimitación atente contra la integridad de la comunidad universitaria o el normal desarrollo del proceso formativo.
- Abstenerse de actuar dentro del campus universitario bajo el efecto de sustancias que alteren su comportamiento y pongan en riesgo el prestigio, la seguridad, la tranquilidad y la salubridad de la comunidad universitaria.

2.3 ACREDITACIÓN

El término acreditación y sus factores acuñan en gran parte los orígenes de este proyecto. Es por lo anterior, que su descripción es fundamental para fines prácticos de este documento.

La acreditación es el proceso a través del cual una agencia o asociación legalmente responsable otorga reconocimiento público a una escuela, instituto, colegio, universidad o programa especializado, que reúne ciertos estándares educativos y calificaciones previamente establecidas. La acreditación es determinada por medio de una evaluación inicial, seguida de otras periódicas. El propósito del proceso de acreditación es

proporcionar una evaluación profesional aceptable de la calidad de las instituciones y programas educativos y estimular a su mejoramiento constante[8].

En esta parte del documento se enumeran los factores de acreditación que intervienen en el desarrollo del proyecto de grado.[9]

- Factor proyecto institucional

Es la orientación de toda labor académica de la institución que permite una formulación y un mejoramiento en la construcción de programas y profesionales con la participación de toda la comunidad académica[10].

- Factor estudiantes y profesores

Los estudiantes de la Facultad de Sistemas de la UNAB son todas aquellas personas que se presentaron en un programa de admisión, donde se evalúa y analiza y se acepta su perfil como el que se exige en este programa.

Los docentes son personas especializadas en las diferentes ramas de la carrera que enseñan y apoyan los programas educativos de esta facultad.

Profundizar en video juegos es fundamental en la investigación de este proyecto, se debe ser conciente de lo que es un video juego, la clase de juegos, cómo se hace, cómo se debe hacer, en que herramienta se puede hacer, entre otros.

2.4 VIDEOS JUEGOS

2.4.1 ¿Qué es un juego? Un juego es una actividad recreativa que involucra a uno o más jugadores. Este puede ser definido por:

- Un objetivo que los jugadores tratan de alcanzar.
- Un conjunto de reglas que dicen lo que los jugadores pueden o no pueden hacer.

La función principal de un juego es la de entretener y divertir, pero puede también representar un papel educativo.

Los juegos no son más que el reflejo de nuestras actividades diarias, esto se puede apreciar en los economistas que tienen que desempeñar diariamente todos estos roles y situaciones.

2.4.2 Clase de juegos. Los juegos se clasificar de la siguiente manera:

- Juegos de habilidad o destreza
- Juegos de estrategia
- Juegos de azar
- Juegos de aventura
- Juegos de acción

---

[8] Centro nacional de acreditación. [En línea]. [Citado el 1 de Abril del 2006]. Disponible en Internet en: <http://www.cna.gov.co/>

[9] FOWLER, M; BECK, K and BRANT, J. Refactoring: Improving the Design of Existing Code. Laboratorio de Sistemas de Información. Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia .Addison - Wesley. 1999. 17 P

[10] Ibit 8

- Juegos educativos
- Juego de palabras[11]

A medida que se ha avanzando en este tema han aparecido otro tipo de juegos, se han sistematizado los que ya existían y con la combinación de estos juegos han aparecido otras clasificaciones. A continuación en las figuras 3, 4, 5, 6 y 7 se mostrarán la evolución en las interfaces de estos juegos:[12]

- Juegos de estrategia

Figura 3. Juegos de estrategia (command and conquer (EA), warcraft (Blizzard)).

Fuente. Blizzard Entertainment. Juego command and conquer (EA), warcraft Blizzard. [En línea]. [Citado el 17 de agosto del 2006]. Disponible en Internet en: <www.blizzard.com>

- Juegos Simuladores

Figura 4. Simuladores (Flight Simulator (Microsoft)).

Fuente. Flight Simulator (Microsoft). [En línea]. [Citado el 23 de agosto del 2006]. Disponible en Internet en: <www.microsoftgames.com>

[11] Introducción a la teoría de juegos. Tipo de juegos. [En línea]. [Citado el 20 de Agosto del 2006]. Disponible en Internet en: <http://www.eumed.net/cursecon/juegos/>

[12] Categoría de juegos. [En línea]. [Citado el 1 de Abril del 2006]. Disponible en Internet en: <http://www.eumed.net/cursecon/libreria/bg-micro/5.htm/>

- Primera persona

Figura 5. Primera persona (DOOM (Atari), Battlefield 2(EA)).

Fuente. DOOM (Atari), Battlefield 2 (EA). [En línea]. 2002. [Citado el 21 de agosto del 2006]. Disponible en Internet en: <www.ea.battlafied.com>.

- RPG

Figura 6. RPG (Role Playing Game) (Dungeon Siege (Microsoft), Fable (Microsoft)).

Fuente. Flight Simulator (Microsoft). [En línea]. [Citado el 23 de agosto del 2006]. Disponible en Internet en: <www.microsoftgames.com>

- Juego de Deportes

Figura 7. Deportes (FIFA (EA)).

Fuente. MADDEN NFL 08. Deportes. FIFA (EA). [En línea]. [Citado el 21 de agosto del 2006]. Disponible en Internet en: <www.easports.com>

Hoy en día han aparecido nuevos tipos de video juegos gracias al Internet y a las telecomunicaciones, como es el caso del MMORPG (Massively Multiplayer Online Role Playing Game**),** que aunque es muy reciente ha tenido una acogida por toda la comunidad de jugadores de videojuegos y comunidades educativas. Debido a que ya no se enfrenta a una maquina con los mismos patrones, si no que se enfrenta a otras personas en diferentes partes del mundo, esto hace que los juegos se superen a otro nivel, diferentes habilidades y destrezas; y lo más curioso es que este tipo de juegos ha llamado a la universidad mas prestigiosa a entrenar desarrolladores de video juegos llamada DIDGIPEN la cual esta tratando de implementar esta nueva forma para educar mas rápido sus estudiantes.[13]

A continuación se podrá observar en la figura 8 el tipo de interfaces y ambientación en la cual se desarrollan los videojuegos MMORPG:

Figura 8**.** Nuevo tipo de juego MMORPG.

Fuente. World Warcraft. Nuevo tipo de juego MMORPG. [En línea]. [Citado el 18 de Agosto del 2006]. Disponible en Internet en: <www.worldofwarcraft.com>

2.4.3 Las situaciones de juego. Existe una situación de juego cuando dos o más individuos buscan relacionarse. Evidentemente tal situación puede tomar las formas más diversas y para avanzar en la reflexión es necesario ser más precisos, especialmente en lo referido al marco en el cual los individuos interactúan con

[13] Quest3D. Historia de los videojuegos. [En línea]. 2000. [Citado el 17 de agosto del 2006]. Disponible en Internet en: <http://www.quest3d.com/index.php?id=17>

las reglas del juego, la información disponible por los jugadores y sus tipos de comportamiento, que puede ser más o menos cooperativo.

- Juegos y cooperación. Todo juego supone reglas y evidentemente su aceptación por los participantes, la situación postulada y no verdaderamente explicada es lo que impone una restricción a priori a la elección hecha por los jugadores. Dicho de otra manera, todo juego supone un consenso mínimo de los participantes.

- Juegos e información. La información permite a los jugadores tomar decisiones importantes dentro del juego, esta información consiste en comunicar el estado del jugador y de los otros participantes.

- Sobre la importancia de los turnos. Es importante para la persona que esta jugando cualquier tipo de juego tener definido y claro cuando es que le corresponde jugar. Esto le permite crear estrategias, tomar decisiones y actuar correctamente para obtener su premio u objetivo dentro del juego.

- Acciones y estrategias. Todo modelo de juego necesita que se precise el dominio de elección de cada uno de los participantes, es decir, del conjunto de acciones a su disposición, pues la solución de un juego puede cambiar radicalmente según el tipo de acción.

En tanto conozcan las acciones que se les "permite", lo mismo que las reglas del juego y su turno, los jugadores pueden establecer planes de acción, denominados estrategias, en las cuales se consideran todas las situaciones posibles. Evidentemente, si el juego tiene un solo golpe, con decisiones simultáneas, las acciones y las estrategias se confunden.[14]

Por fuera de tal caso, las estrategias son condicionales, en tanto deben considerar todas las acciones posibles en diversas oportunidades.

Estas son las características básicas hoy en día de los juegos de video. Pero los juegos educativos tienen otras cualidades y categorías las cuales son muy semejantes pero con algunas adiciones. Pero para poder saber cuales son las esas diferencias primero hay que saber que son los juegos educativos

2.4.4 Juegos Educativos. Los juegos que proporcionan información relevante con una metodología de enseñanza de modo entretenido y divertido al mismo tiempo, se denominan juegos educativos. Los juegos educativos tienen diferentes características, las cuales se mencionan a continuación.

- Aventura y Juegos de Rol. El jugador desempeña un determinado rol o papel protagónico donde se presenta cierta información y recursos disponibles para superar obstáculos y cumplir su objetivo.

- Juegos de Negocios. Este tipo de juegos son una herramienta de apoyo en el proceso de aprendizaje de los negocios, incorporando un ambiente competitivo e incentivando a la toma de decisiones económicas.

- Juegos de Mesa. Allí intervienen el razonamiento y la estrategia de cada jugador para lograr superar cada actividad del juego.

- Juegos de Combate. Esta modalidad de juegos es muy popular, debido al alto nivel de competitividad y violencia que motivan al jugador a defender su territorio.

- Juegos de Lógica y Rompecabezas. Interviene el racionamiento y destreza del jugador para resolver un problema.

- Juegos de Palabras. Se basan en la interpretación y análisis de palabras para resolver dificultades.[15]

---

[14] Blizzard Entertainment. Juego command and conquer (EA), warcraft Blizzard. [En línea]. [Citado el 17 de agosto del 2006]. Disponible en Internet en: <www.blizzard.com>

[15] World Warcraft. Nuevo tipo de juego MMORPG. [En línea]. [Citado el 18 de Agosto del 2006]. Disponible en Internet en: <www.worldofwarcraft.com>

2.4.4.1 Factores Generales. En Juegos existen tres factores básicos que intervienen en un juego: la introducción, el cuerpo y la conclusión. Entre ellos surgen características generales que deben considerarse, tales como:

- Metas. Es la finalidad del juego, lo que el jugador desea superar, y contrasta con la finalidad de los objetivos educativos del juego.

- Reglas. Definen las acciones que pueden desempeñarse en el juego, simulando la realidad y generando un aspecto interesante en el juego.

- Competencia. Es una de las características más importantes a la hora de efectuar el juego. Se enfocan en buscar la rivalidad entre oponentes, es decir, entre el jugador o jugadores y la máquina.

- Desafío. Son juegos orientados a cumplir ciertas etapas establecidas con una variedad de obstáculos con alto nivel de dificultad y donde prueban al máximo la lógica del jugador.

- Fantasía. Juegos que generan gran motivación debido a su que su contenido va mas allá de la realidad, con escenarios creados con gran imaginación e innovación.

Otros factores son la seguridad y la hospitalidad, que juegan un papel definitivo en cada juego, debido a que muestran un ambiente adecuado al jugador para lograr superar las metas del juego.

2.4.4.2 Factores en la introducción de un Juego

- Metas, son la finalidad del juego.

- Reglas, definen la naturaleza de cada jugador.

- Jugadores, elemento esencial, con variedad de roles.

- Equipo, define elementos para jugar, como teclado, mouse, joystick.

- Direcciones, orientación del juego.

- Premios, reconocimiento por ganar.

- Penas, acciones incorrectas del jugador

- Opciones, definen las características del juego, preferencias

Esta clase de juegos se pueden desarrollar en un motor llamado A6, el cual esta siendo utilizada hoy en día por varios diseñadores de juegos, este motor permite manipular con mayor facilidad graficas, acciones y demás ventajas que mas adelante serán explicadas.

El demo del juego PEI se desarrollará en tercera dimensión, a continuación están las especificaciones que conlleva esta decisión.

2.5 GRÁFICOS EN 3D

El término gráficos 3D por computadora o por ordenador (3D computer graphics) se refiere a trabajos de arte gráfico que fueron creados con ayuda de computadoras y programas especiales 3D. En general, el término puede referirse también al proceso de crear dichos gráficos, o el campo de estudio de técnicas y tecnología relacionadas con los gráficos 3D.

Un gráfico 3D difiere de uno 2D principalmente por la forma en que ha sido generado. Este tipo de gráficos se origina mediante un proceso de cálculos matemáticos sobre entidades geométricas tridimensionales producidas en un ordenador, y cuyo propósito es conseguir una proyección visual en dos dimensiones para ser mostrada en una pantalla o impresa en papel.

En general, el arte de los gráficos 3D es similar a la escultura o la fotografía, mientras que el arte de los gráficos 2D es análogo a la pintura. En los programas de gráficos por computadora esta distinción es a veces difusa: algunas aplicaciones 2D utilizan técnicas 3D para alcanzar ciertos efectos como iluminación, mientras que algunas aplicaciones 3D primarias hacen uso de técnicas 2D.

2.5.1 Tecnología OpenGL y Direct3D. APIs (Application Programming Interface - Interfaz de Programación de Aplicaciones) muy populares para la generación de imágenes 3D en tiempo real. Muchas tarjetas gráficas modernas proveen de cierto grado de aceleración por hardware basado en estas APIs, frecuentemente habilitando el despliegue de complejos gráficos tridimensionales en tiempo real. Sin embargo, no es necesario emplear alguna de estas interfaces para la generación de imágenes 3D.

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

- Modelado
- Composición de la escena
- Rénder (creación de la imagen final)[16]

2.5.3 Modelado. La etapa de modelado consiste en ir dando forma a objetos individuales que luego serán usados en la escena. Existen diversas técnicas de modelado; Constructive Solid Geometry, modelado con NURBS y modelado poligonal son algunos ejemplos. Los procesos de modelado pueden incluir la edición de la superficie del objeto o las propiedades del material (por ejemplo, color, luminosidad, difusión, especularidad, características de reflexión, transparencia u opacidad, o el índice de refracción), agregar texturas, mapas de relieve (bump-maps) y otras características.

El proceso de modelado puede incluir algunas actividades relacionadas con la preparación del modelo 3D para su posterior animación. A los objetos se les puede asignar un esqueleto, una estructura central con la capacidad de afectar la forma y movimientos de ese objeto. Esto ayuda al proceso de animación, en el cual el movimiento del esqueleto automáticamente afectara las porciones correspondientes del modelo.

El modelado puede ser realizado por programas dedicados (como Lightwave 3D, Rhinoceros 3D o Moray), un componente de una aplicación (Shaper, Lofter en 3D Studio) o por un lenguaje de descripción de escenas (como en POV-Ray). El modelado es sólo una parte del proceso de creación de escenas.

2.5.4 Composición de la escena. Esta etapa involucra la distribución de objetos, luces, cámaras y otras entidades en una escena que será utilizada para producir una imagen estática o una animación. Si se utiliza para Animación, esta fase, en general, hace uso de una técnica llamada "Keyframing", que facilita la creación de movimientos complicados en la escena. Con la ayuda de la técnica de keyframing, en lugar de tener que corregir la posición de un objeto, su rotación o tamaño en cada cuadro de la animación, solo se necesita marcar algunos cuadros claves (keyframes). Los cuadros entre keyframes son generados automáticamente, lo que se conoce como 'Interpolación'.

[16] EBERLY, David H. 3D game engine design: a practical approach to real-time computer graphics. 2nd ed.Amsterdam; Boston: Elsevier Morgan Kaufmann, c2007. 187 P

La iluminación es un aspecto importante de la composición de la escena. Como en la realidad, la iluminación es un factor importante que contribuye al resultado estético y a la calidad visual del trabajo terminado. Por eso, puede ser un arte difícil de dominar. Los efectos de iluminación pueden contribuir en gran medida al humor y la respuesta emocional generada por la escena, algo que es bien conocido por fotógrafos y técnicos de iluminación teatral.

2.5.5 Tesselation y mallas. El proceso de transformar la representación de objetos, como el punto medio de coordenadas de una esfera y un punto en su circunferencia, en una representación poligonal de una esfera, se conoce como tesselation. Este paso es usado en el rénder basado en polígonos, donde los objetos son descompuestos de representaciones abstractas primitivas como esferas, conos, etc, en las denominadas mallas, que son redes de triángulos interconectados.

Las mallas de triángulos son populares ya que está probado que son fáciles de 'renderizar' usando Scanline rendering.

Las representaciones poligonales no son utilizadas en todas las técnicas de rénder, y en estos casos, el paso de tesselation no es incluido en la transición de representación abstracta y la escena 'renderizada'.

2.5.6 Renderizado. Se llama rénder al proceso final de generar la imagen 2D o animación a partir de la escena creada. Esto puede ser comparado a tomar una foto o en el caso de la animación, a filmar una escena de la vida real. Generalmente se buscan imágenes de calidad fotorrealista, y para este fin se han desarrollado muchos métodos especiales. Las técnicas van desde las más sencillas, como el rénder de alambre (wireframe rendering), pasando por el rénder basado en polígonos, hasta las técnicas más modernas como el Scanline Rendering, el Raytracing, la radiosidad o el Mapeado de fotones.

El software de rénder puede simular efectos cinematográficos como el lens flare, la profundidad de campo, o el motion blur (desenfoque de movimiento). Estos artefactos son, en realidad, un producto de las imperfecciones mecánicas de la fotografía física, pero como el ojo humano está acostumbrado a su presencia, la simulación de dichos efectos aporta un elemento de realismo a la escena. Se han desarrollado técnicas con el propósito de simular otros efectos de origen natural, como la interacción de la luz con la atmósfera o el humo. Ejemplos de estas técnicas incluyen los sistemas de partículas que pueden simular lluvia, humo o fuego, el muestreo volumétrico para simular niebla, polvo y otros efectos atmosféricos, y las cáusticas para simular el efecto de la luz al atravesar superficies refractantes.

El proceso de rénder necesita una gran capacidad de cálculo, pues requiere simular gran cantidad de procesos físicos complejos. La capacidad de cálculo se ha incrementado rápidamente a través de los años, permitiendo un grado superior de realismo en los rénders. Estudios de cine que producen animaciones generadas por ordenador hacen uso, en general, de lo que se conoce como render farm (granja de rénder) para acelerar la producción de fotogramas.

2.5.7 Modelos de reflexión y sombreado. Los gráficos 3D por ordenador modernos cuentan con un modelo de reflexión llamado Phong reflection model, que no debe ser confundido con Phong shading, que es algo completamente diferente.

Este modelo de reflexión y las técnicas de sombreado que permite, se aplican solo a rénders basados en polígonos. Por ejemplo, raytracing y radiosity no lo utilizan.
Técnicas de rénder de reflexión populares son:[17]

- Flat shading. Una técnica que sombrea cada polígono de un objeto basado en la normal del polígono y la posición e intensidad de una fuente de luz.

[17] AMBLER, Scott. Ambysoft. Metodología AUP. [En línea]. 2005. [Citado el 10 de enero del 2007]. Disponible en Internet en: <http://www.ambysoft.com/unifiedprocess/agileUP.html>

- Gouraud shading. Inventado por H. Gouraud en 1971, es una rápida técnica de sombreado de vértices usada para simular superficies suavemente sombreadas.

- Texture mapping. Es una técnica para simular un gran nivel de detalle superficial, aplicando imágenes (texturas) sobre los polígonos.

- Phong shading. Inventado por Wu Tong Phong, es utilizado para simular brillos especulares y superficies sombreadas suaves.

- Bump mapping. Creado por Jim Blinn, es una técnica de perturbación utilizada para simular superficies rugosas.

2.5.8 APIs de Gráficos 3D. Los gráficos 3D se han convertido en algo muy popular, particularmente en juegos de computadora, al punto que se han creado APIs especializadas para facilitar los procesos en todas las etapas de la generación de gráficos por computadora. Estas APIs han demostrado ser vitales para los desarrolladores de hardware para gráficos por computadora, ya que proveen un camino al programador para acceder al hardware de manera abstracta, aprovechando las ventajas de tal o cual placa de video.

Las siguientes APIs para gráficos por computadora son particularmente populares:

- OpenGL

- Direct3D (subconjunto de DirectX para producir gráficos interactivos en 3D)

- RenderMan

2.5.9 Software de gráficos 3D. A pesar de haber muchos paquetes de modelado y animación 3D, los cuatro que se han ganado la mayor popularidad son:

- Maya (Alias Wavefront). Es el software de modelado más popular en la industria. Tras la adquisición de la empresa fabricante, ALIAS, por parte de AUTODESK, la versión octava de Maya fue publicada

- 3D Studio Max (Discreet). Fue originalmente escrito por Kinetix (una división de Autodesk) como el sucesor de 3D Studio para DOS. Más tarde Kinetix se fusionaría con la última adquisición de Autodesk, Discreet Logic. La versión más reciente en Octubre de 2006 era la 9.0. Es el líder en el desarrollo 3D de la industria del videojuego.

- Lightwave 3D (Newtek). Fue originalmente desarrollado a principios de la década de los 90. Más tarde evolucionó en un avanzado paquete gráfico y animación 3D. Actualmente disponible para Windows, Mac OS y Mac OS X. La versión a principios del 2006 era la 8.5. El programa consiste en dos componentes: el modelador y el editor de escena. Es utilizado en multitud de productoras de efectos visuales como Digital Domain.

- Softimage XSI (Avid). El contrincante más grande de maya. En 1987, Softimage Inc, una compañía situada en Montreal, escribió Softimage|3D, que se convirtió rápidamente en el programa de 3D más popular de ese período. En 1994, Microsoft compró Softimage Inc. y comenzaron a reescribir SoftImage|3D para Windows NT. El resultado se llamó Softimage|XSI. En 1998 Microsoft vendió Softimage a Avid. La versión a mediados del 2003 era la 3.5.

Junto a estos paquetes mayores, hay otros que no se han ganado tal aceptación general, pero que no son simples juguetes. Algunos son:

- Caligari trueSpace. Es una aplicación 3D integrada, con una interfase muy intuitiva. Una característica distintiva de esta aplicación es que todas las fases de creación de gráficos 3D son realizadas dentro de un único programa. No es tan avanzado como los paquetes líderes, pero provee características como simulación de fenómenos físicos (viento, gravedad, colisiones entre cuerpos).

- Cinema4d. Es un motor de rénder rápido, cálculo de radiosidad.

- formZ. Ofrece manipulación topológica de las geometrías.

- Rhinoceros 3D. Un potente modelador bajo NURBS.

- POV-Ray. Un avanzado software gratuito de Raytracing. Usa su propio lenguaje de descripción de escena, con características como macros, bucles y declaraciones condicionales. Es completamente gratuito aunque no fue lanzado bajo GPL. No incluye modelador.

- Moray. Modelador para POV-Ray.

- Blender (NaN). Programa de modelado y animación libre, con características como soporte para programación bajo Python con una amplia gamma de script en constante desarrollo, posee un engine robusto para la programación de juegos, un Motor de render propio y una comunidad de usuarios totalmente abierta y dispuesta a colaborar.

- RealSoft3D. Modelador 3D para Linux y Windows. Incluye rénder.

- Universe por Electric Image. Paquete de modelado y animación con uno de los motores de rénder más rápidos que existen.

2.6 MOTORES

En este capítulo se muestran las características de algunos de los mejores y mas usados motores para el desarrollo de juegos.

2.6.1 Game Engine. Hace referencia a una serie de rutinas de programación que permiten el diseño, la creación y la representación del juego. El motor es el corazón del sistema de desarrollo esto genera la imagen de 3D y controla el comportamiento del mundo virtual.

La analogía con el motor de un automóvil es ilustrativa: el motor debajo del capot no es visible pero le da la funcionalidad al automóvil que es la de transportar. La misma analogía permite explicar algunos de los aspectos que generalmente maneja un motor de juego: las texturas y los modelos 3D serían la carrocería, pintura e interiores.

Del mismo modo en que carrocería, pintura y exteriores no andan sin un motor, el arte y los guiones del juego no funcionan sin un motor de juego. Un ejemplo de motor de juego seria el motor grafico del juego Doom.[18]

- Assets (Activos). Son los modelos, animaciones, sonidos, IA, físicas. Son los elementos que forman el juego en sí, el código hace funcionar los assets.

- Application Programming Interface (Interfaz de Programación de Aplicaciones). Es un sistema de rutinas, de protocolos y de herramientas para desarrollar programas de aplicación. Un buen API hace más fácil desarrollar un programa proporcionando todos los bloques del desarrollo del programa. El programador pone los bloques juntos.

Entre estos los más importantes son el DirectX (de Microsoft) y el OpenGL (que trabaja con la mayoría de los sistemas operativos).

- Render (Renderización). Es la parte del código que pone en pantalla los ambientes y objetos.

---

[18] COAD, P; LEFEBVRE, E and DE LUCA, J. Java Modeling In Color With UML: Enterprise Components and Process. Prentice Hall Londres, Inglaterra, 1999. 215

- Objetos 3D. Los objetos se almacenan por puntos en un mundo 3D, llamados vértices. Los vértices van formando polígonos; cuanto más polígonos posea un objeto, más complicado se hace, lleva más tiempo de procesamiento pero es más detallado. El juego no necesita saber cuantos objetos hay en memoria o como el Render va a mostrarlos, solo le interesa que el render los despliegue de la forma correcta, y que el modelo este en el cuadro correcto de la animación.

- Higher-order surfaces (superficies de alto orden). Renderizado matemáticamente, usado en las tarjetas gráficas más recientes y poderosas. También llamado: Patches (parches).

- Patches (Parches). Son perfectos para describir geometrías, sobre todo cuando se trata de curvas, pues la expresan mediante fórmulas matemáticas logrando colocar puntos en el mundo del juego

- Three-point polygon (polígonos de 3 puntos-Triángulos). El más usado por las tarjetas aceleradoras 3D.

- Culling. Codificado que logra que los objetos que no se ven en determinado cuadro de la animación por causa de objetos que los obstaculizan (como una pared) no tomen tiempo de renderizado. Así se reduce la cantidad de trabajo del motor. El Culling es más fácil de implementar en juegos en donde la visión es controlada como los RTS en comparación con lo FPS. Un método de Culling puede ser por "Árboles BSP"

- BSP Tree Hierarchy (BSP Árbol de Jerarquía). Es un método para determinar qué superficies de un mundo, y qué objetos, están realmente en la escena en momento dado, dada su localización en el mundo. Esto se utiliza a menudo para los objetos del desecho, y también para entresacarlos para reducir el proceso del AI (Inteligencia Artificial) y de la animación. Retesselation: técnica usada por la característica de TruForm de ATI que consiste en tomar un modelo basado en triángulos y transformarlo en uno de High-Order Surfaces para alisarlo y de nuevo pasarlo a un modelo de Triángulos.

- Iluminación (lighting). Distintos APIs proveen diferentes tipos de iluminación

- Vertex Lighting. Se determinan cuantos polígonos cruzan el vértice, se toma el total de todas las orientaciones de los polígonos (Normal) y se asigna la normal al vértice. Para cada vértice, un polígono dado reflejará la iluminación en una forma levemente distinta. La ventaja es que le hardware le toma menos tiempo de procesar pero este tipo de iluminación no produce sombras.

- Flat Shading Lighting (Iluminación de Sombreado Plano). Consiste en que cada polígono represente un valor leve que se pase al polígono completo que genere una imagen plana del mismo, a esta imagen también se le asigna un color determinado.

- Vertex Shading (Sombreado de Vértice, Gouraud shading). Solicita al motor de renderizado un color para cada vértice, luego por medio de interpolación se renderiza cada píxel por la distancia en relación con su respectivo vértice.

- Phong Shading. Es similar al Gouraud Shading, trabajan con la textura, solo que el Phong Shading usa a los píxeles en lugar de lo vértices. El Phong Shading toma más tiempo de procesamiento que el Vertex Shading pero su resultados son mucho mejores en cuestión de suavizado de texturas.

- Light Map Generation (Generación del mapa de luz). Se usa una segunda capa de textura (mapa de luz) que dará el efecto de iluminación a los modelos, es un efecto excelente pero debe tomarse antes del renderizado pero si se tienen Luces Dinámicas (o sea luces que se mueven, encienden o apagan sin intervención de programa) se debe estar regenerando los mapas en cada Frame de animación lo que toma mucha cantidad de memoria (pero son de render rápido).

- Textura. Es esencial para que las escenas 3D se vean reales, en si las texturas son imágenes que se rompen en los distintos polígonos del modelo, muchas imágenes tomarán mucho espacio en la memoria por eso se debe usar técnicas de compresión:
    - Mapeo MIP. Consiste en preprocesar las texturas creando múltiples copias del mismo cada una la mitad del anterior, esto porque si la textura solo es pegada al polígono cada textura es a cada píxel y tomara más tiempo de render; así cada Texel (elemento de Textura) toma menos espacio.

- Texturas Múltiples. Requiere múltiples renderizados por lo que para obtener buen resultado se necesita una tarjeta con Acelerador de Gráficos, provee mejor calidad que el simple mapeo. Se puede colocar una imagen sobre otra (más transparente) para dar el sentido de movimiento pulso o hasta sombra.
- Bump Mapping. Técnica vieja de texturas que tratan de mostrar como la luz se refleja en el objeto. Solo hasta hace poco se vuelto a retomar.
- Antialiasing. El anti-aliasing revisa los polígonos y difuminará las bordes y vértices, para que los bordes no se vean como dentados. Esta técnica se puede hacer de dos maneras. La primera se realiza de modo individual, entremezclando polígonos para sobreponerlos unos delante de otros.

La segunda manera se hace por medio de tomar todo el marco y quitarle los bordes dentados, pero esto requiere de mucha memoria.

- Vertex and Pixel Shaders (Vértices y Sombreo de Pixeles). Con este método se pueden extraer y utilizar directamente las características y facilidades de la tarjeta de video, sin tener que utilizar mucho la API. Pero no es utilizable en todas las tarjetas.

- Stencil Shadowing (Plantilla de Sombreado). La idea es renderizar una vista de un modelo desde la perspectiva de la fuente de luz y después utilizar esto para crear o para generar un polígono con la forma de esta textura sobre las superficies afectadas por el modelo. Así se obtiene una iluminación que parece real. Pero es costosa, porque usted está creando texturas “en vuelo”, y hace múltiple render de la misma escena.

- El manejo del cache de textura es imprescindible para que el juego se desarrolle rápido (y para cualquier motor), ya que si se presenta un constante swapping de las texturas en la tarjeta el juego se vera lento y tedioso, algunos APIs descargan cada textura cuando esto pasa, pero eso haría que en cada cuadro se refresquen las texturas dando más lentitud. Todo se trata de cargar la menor cantidad de veces una misma textura, pero eso también depende del API que se utilice. Otra técnica es la compresión de texturas, comprimir texturas es como comprimir MP3, los algoritmos de compresión logran una relación 4:1 que no es mucho pero ayuda.

- LOD (Nivel de Detalle). El sistema de nivel de detalle esta relacionada con la complejidad geométrica de los modelos. Algunos sistemas necesitan que se hagan múltiples versiones del modelo, para que dependiendo de cuan cerca se este del modelo así será su cantidad de polígonos. Otros sistemas ajustan dinámicamente esta característica pero en este caso da más carga al CPU

- Depth Testing (prueba de profundidad). Con esto se empieza a eliminar los píxeles ocluidos y se pone en práctica el concepto de sobre dibujado. La prueba de profundidad es una técnica utilizada para determinar que objetos están delante de otros en la misma localización del píxel.

- Sobre Dibujado. Es la cantidad de veces que se ha dibujado un píxel en un frame. Se basa en la cantidad de elementos existentes en la tercera dimensión (profundidad).

Scripting Systems (Sistemas de Guionaje)

- Pre-scripted Cinematics. Usada normalmente en una situación que necesita la explicación en una manera controlada. Para presentar las escenas de la historia, ahora se utilizada el cortar-escenas que presenta la historia en video digital y luego por medio de transiciones se pasa a las graficas reales del juego.

- El Guionaje le permite al diseñador tomar mando de la escena y manipularla, como colocar objetos o eventos que el jugador no controla. En muy complicado, se necesita de una mente muy metódica y lógica, la mayoría de estos scripts se basan en lenguaje C.

- Visual Scripting Systems. Como lo dice su nombre, permite manejar el script en un ambiente grafico en lugar de un código escrito, se maneja un carácter real en un ambiente del juego real.

- Sonido. Creative Labs ahora ha proporcionado sus extensiones manejadores de sonido EAX para DirectX, y la nueva iniciativa de OpenAL (biblioteca audio abierta). OpenAL, como suena, es un API para los sistemas de los sonidos de la misma manera que OpenGL es un API

- Para el procesado de sonido es muy similar al procesado de los modelos, muchas veces un software los procesa antes de pasar al hardware respectivo, por ejemplo DirectSound hace al sonido para la Tarjeta de sonido lo que Direct3D hace al modelado antes de llegar al la Tarjeta 3D. Esto es llamado "premezcla" en el software

- Music Tracks in Games (pistas de audio). Hay dos formas de manejar el sonido. Uno es por medio de archivos. wav (o similares), lo cual emite un muy buen sonido, pero se requiere de mucha memoria. Por otro lado se pude utilizar archivos midi, esto reduce la necesidad de memoria, pero los sonidos no son tan buenos.

- Inteligencia Artificial. Es la característica más importante que se le atribuye a un motor al lado de la representación de modelos o Render. AI provee de estimulo al juego, es critico en la parte de la Forma de juego (game play).

La inteligencia artificial ha determinado que un juego puede volverse muy difícil de usar y entender, como solución se recomienda seguir los siguientes pasos en el diseño y desarrollo, primero se debe definir la línea base del comportamiento de los NPC (Personajes que no son el Jugador), segundo debe definirse que hace el NPC (patrulla, guarda, etc.), luego se delimita su "visión de mundo", qué es lo es el NPC puede ver del mundo del juego; se debe tomar en cuenta que el personaje no estará solo en medio del mundo del juego, sino que también interactuara con él, después vienen las rutinas de Toma de Decisión: si el NPC está patrullando, y hay un sonido, ¿debe tomarle importancia o no?, ¿investiga su origen o no?, etc.

Es un sistema de reglas para las acciones que responden (o inician) y que el jugador debe responder, esto a un concepto más general de inteligencia artificial.

- Emergent game play (Forma de Juego Emergente). Consiste en programar al AI con un conjunto de reglas que le permitan al programa adherir situaciones que el programador no previera

2.6.2 Unreal Engine. Es un motor para juegos de PC y consolas creados por la compañía Epic Games. Implementado inicialmente en el shooter en primera persona llamado Unreal en 1998, siendo la base de muchos juegos desde entonces. También se ha utilizado en otros géneros como el rol y juegos de perspectiva en tercera persona. Esta escrito en C++, creando varias versiones que engloban las plataformas PC (Microsoft, windows, GNU/Linux), Apple Macintosh (Mac Os, Mac Os X) y la mayoría de consolas (Dreamcast, Xbox, Xbox 360, Playstation 2, Playstation 3, Wii). Unreal Engine también ofrece varias herramientas adicionales de gran ayuda para diseñadores y artistas.[19]

- Versión y año de lanzamiento. Unreal Engine (1998), Unreal Engine 2.0 (2002), Unreal Engine 2.5 (2003 - 2004), Unreal Engine 3.0 (2007).

- Unreal Engine 3. El Unreal Engine de la tercera generación fue diseñado específicamente para DirectX 9/10 PC y las consolas de última generación (actualmente el Xbox 360 y Playstation 3), aunque han surgieron noticias que Ubisoft estaba trabajando con Epic para proporcionar una

- Versión con requerimientos de hardware más bajos y aplicable en el Wii. Finalmente se desechó la idea debido a las bajas prestaciones gráficas de la consola y según ellos a la falta de alta definición que prestaba la consola.

Su nuevo motor reescrito apoya muchas técnicas avanzadas incluyendo HDR, iluminación por pixel, y sombras dinámicas. También existen herramientas complementarias al igual que las anteriores versiones del motor. Se

[19] MotorUnrealengine3. [En línea]. 1997. [Citado el 25 de marzo del 2007]. Disponible en Internet en: <http://www.3dpoder.com/foro3dpoder/showthread.php?t=25830>

sustituye a Karma por PhysX de Ageia, y FaceFX se incluye además para generar animaciones faciales. Epic utilizó esta versión del motor para los Gears of War y la está utilizando para el Unreal Tournament 3.

Debido a su política de licencias, está generando el apoyo de importantes marcas como Sony, Electronic Arts y Square Enix. Gracias a su versatilidad se aplica en sectores no relacionados con los videojuegos como simulación de construcciones, simuladores de conducción, previsualisación de películas y generación de terrenos utilizados por la NASA

2.6.3 Motor A6. Como se anunció anteriormente el motor para desarrollar el demo del juego PEI es el A6, en esta parte del capitulo de motores se encontrará la información necesitaría de este motor.

Un Motor de Juego es básicamente una librería de funciones de software relacionadas con los juegos. El motor A6 contiene funciones para gráficos en 2D y 3D, detección de colisiones, sonido, multijugador, física, e interfaz de usuario, junto a un lenguaje de scripting que ofrece fácil acceso a esas funciones.

El renderizador es el núcleo de un motor de juego, el cual pinta los objetos 3D entidades en la pantalla. El motor A6 usa diferentes algoritmos de renderizado para cinco tipos de entidades: modelos, sprites, terreno, niveles BSP, y partículas. Dependiendo de qué método es usado para acceder al hardware 3D, existen renderizadores de DirectX y OpenGl. Un renderizador de OpenGl utiliza el hardware 3D a través de la librería gráfica OpenGL, la cual está disponible para la mayoría de las tarjetas 3D. Un renderizador de DirectX utiliza la librería DirectX de Microsoft, la cual está integrada en Windows. En tarjetas 3d antiguas, OpenGL normalmente renderiza un poco más rápidamente, mientras que en las tarjetas modernas, DirectX ofrece mejores características y rendimiento. El motor A6 usa un renderizador de DirectX 9.0.

Un sistema de culling renderiza sólo las partes de un nivel de juego que no están cubiertas por muros u otros objetos. Los sistemas de culling usuales son basados en un árbol BSP o en Portales. El sistema de árbol BSP es el más rápido y el más efectivo, especialmente para niveles interiores, pero tiene la desventaja de que el árbol BSP debe ser precalculado por el editor de niveles. Los renderizadores que no soportan culling mayoritariamente usan sistemas Octree para organizar la escena.

La mayoría de los motores 3D comerciales, incluyendo a A6, usan un sistema de culling basado en el árbol BSP. Con un sistema de culling basado en el árbol BSP, la velocidad de renderizado de interiores es independiente del tamaño del nivel y el número de objetos, lo cual permite a los juegos ejecutarse a una velocidad decente aún en computadoras antiguas. Un sistema LOD también aumenta la velocidad en los niveles exteriores. Automáticamente cambia a formas más 'simples' de los objetos cuando están lejos de la cámara, reduciendo de estos modos el número de polígonos dibujados por cuadro de animación.

El Mapeado de Sombras también llamado Lightmapping es un sistema para crear luces y sombras realistas sin penalizar la velocidad de ejecución. Un compilador de mapeado de sombras permite colocar un número ilimitado de fuentes de luz estáticas en el nivel, y luego precalcula el flujo de luz y las sombras estáticas para cada superficie. La mayoría de los motores 3D comerciales hoy en día usan mapeado de sombras.

Un Sistema de Partículas es un generador de efectos que crea un enorme número de pequeñas partículas para efectos especiales como humo, fuego, o explosiones. Los efectos de partículas bien hechos lucen mejor que una animación prerenderizada, y por tanto los efectos de partículas son usados en todos los juegos y consolas nuevas. Para crear efectos de partículas realistas, el generador de partículas debe ser capaz de mover miles de partículas sin reducir la velocidad de ejecución. Los sistemas de partículas simples sólo permiten asignar a las partículas algunas propiedades como duración, gravedad, o color; los sistemas más sofisticados, como el de A6, también permiten programar funciones de movimiento individuales para cada partícula, y contiene un generador de rayos para crear haces de luz o estelas de movimiento.

Los Shaders añaden una nueva dimensión al renderizado de los gráficos. Permiten que la funcionalidad de transformación, iluminación, y renderizado sean modificadas durante la ejecución a nivel de vértex o píxel. Un shader es un pequeño script que se ejecuta en el hardware gráfico para cada píxel o vértex que es renderizado en pantalla. Esto le da al usuario un nuevo nivel de flexibilidad dinámica sobre la manera en que los píxeles son renderizados. Los shaders de Vértex y píxel pueden ser usados para crear ondas realistas en el agua, reenderizar al estilo 'toon', cubrir modelos con pelaje, o controlar el flujo de la lava de un volcán.

Como se ha dicho anteriormente el motor A6 utiliza para el proceso de renderizado el rederizador DirectX, por lo cual fue importante el estudio, analisis y apropiación de estos conceptos.

2.6.3.1 Directx. Es importante conocer acerca de Directx, porque es una herramienta que permite una mejor la creación y diseño de graficas en tercera dimensión.

Microsoft DirectX es una colección de APIs para las tareas de dirección relacionadas con multimedia, especialmente programación del juego, encendido Microsoft plataformas. Una porción de él, Direct3D, compite contra OpenGL y contra SDL. Es ampliamente utilizado en el desarrollo de juegos de computadora para Microsoft Windows, Microsoft Xbox y Microsoft Xbox 360.

DirectX también se utiliza entre otras industrias de la producción del software, lo más notablemente posible entre el sector de la ingeniería debido a su capacidad de rendir rápidamente los gráficos de alta calidad 3D usando el más último hardware de los gráficos.

El DirectX tiempo de pasada y kit del desarrollo del software están disponibles gratuitamente, pero es un software propietario. El tiempo de pasada de DirectX fue redistribuido originalmente para desarrolladores de juegos, pero mas adelante fue incluido en Microsoft Windows. Los desarrolladores de juegos todavía incluyen a menudo una versión actualizada de DirectX que incite la instalación automáticamente después de la instalación del juego para asegurar funcionalidad apropiada del programa. La ultima versión lanzada de DirectX, DirectX 10, es exclusiva de Microsoft Windows Vista.[20]

Los varios componentes de DirectX están bajo la forma de COM “objetos obedientes”. Los componentes que abarcan DirectX son:

- Gráficos de DirectX, abarcan dos APIs (DirectX 8.0 hacia adelante):
  - DirectDraw: para segundos gráficos de dibujo (gráficos de la trama)
  - Direct3D (D3D): para dibujar gráficos 3D

- DirectInput. Son teclado, ratón, palanca de mando u otros reguladores del juego.

- DirectPlay. Para comunicación networked de juegos.

- DirectSound. Para el aparato de lectura y grabación del sonido de forma de onda.
  - DirectSound3D (DS3D). Para el aparato de lectura de sonidos 3D.

- DirectMusic. Aparato de lectura de las bandas de sonido.

- DirectSetup. Para la instalación de los componentes de DirectX.

- Medios de DirectX. Abarcando DirectAnimation, DirectShow, Aceleración del vídeo de DirectX, El modo conservado Direct3D y DirectX transforman para la animación, multimedias aparato de lectura y usos que fluyen, 3D, e interactividad respectivamente.

- Objetos de los medios de DirectX. Apoyo para los objetos que fluyen tales como codificadores, el decodificador y efectos.

Microsoft desarrolló XNA, que es un marco de diseñó para asistir al desarrollo de juegos haciéndolo más fácil integrar a DirectX, un lenguaje de alto nivel de Shader (HLSL) y otras herramientas en un paquete.

Microsoft durante el 2002 lanzó una versión de DirectX compatible con Microsoft Marco de .NET, que permitió que los programadores aprovecharan al máximo de las características de .NET (tales como el uso del C# y Visual Basic lenguajes de programación); simultáneamente con el desarrollo de DirectX se conoció este API

[20] Microsoft. DirectX. [En línea]. [Citado el 25 de Marzo del 2007]. Disponible en Internet en: <www.microsoft.com/directx.htm>

como “DirectX manejado“(o MDX para el cortocircuito) y el funcionamiento demanda ser el 98% del software nativo de DirectX. Las ideas del diseño posibles en DirectX manejado se pueden considerar en el más nuevo marco XNA.

En el 2005 – 2006 Microsoft lanzó una versión de DirectX que se diseña para el marco de .NET 2.0. En más viejas versiones, DirectX estuvo partido de diversos módulos; esto ha cambiado con la versión de .NET 2.0, pues ahora es un solo archivo y es mucho más fácil utilizar. Sin embargo, la versión de .NET 2.0 de DirectX no es una versión concluida; sigue siendo un beta. Durante el GDC Microsoft 2006 presentó el marco de XNA.

2.7 METODOLOGÍAS TRADICIONALES

En este proyecto se aplicará como metodología la “XP”, que pertenece al grupo de metodologías ágiles, lo que hace necesario realizar un estudio para compararlas con las metodologías tradicionales que comúnmente son usadas.

El concepto de Ingeniería del Software llegó después de producirse la crisis del software. Su desarrollo era excesivamente artesanal y no permitía planificar y estimar el esfuerzo de una manera razonable. Los proyectos eran muy ambiciosos y la ausencia de metodologías en muchas ocasiones acababa en un caos. Por este motivo, se importaron metodologías de otros campos donde también existían procesos de Ingeniería. Se trataba de procedimentar y documentar todo el proceso, para minimizar el riesgo de cada proyecto y controlar su evolución. Estas metodologías importadas son las que conocemos como metodologías tradicionales.

Las metodologías tradicionales abordan estos problemas proponiendo comenzar con una fase de análisis, en la que se tomen todas las decisiones, previa al comienzo del desarrollo. Finalizada esta fase de análisis será el momento de comenzar el desarrollo que debe finalizar con una etapa de prueba que asegure la calidad antes de implantar el sistema en producción. Para llevar a cabo todo el proceso proponen una serie de documentos que se deben realizar a lo largo del proyecto. Esta documentación que en su mayoría debe realizarse en la primera fase, nos permitirá tener un plan de proyecto y entender todas las decisiones que se aplican en cada momento del mismo.[21]

Las mayoría de las metodologías tradicionales definen un proceso secuencial donde cada proceso se alimenta del anterior y en el que el software esta disponible al final de todo el proceso. Son los llamados procesos en cascada véase figura 9.

La adopción de estas metodologías fue una reacción ante la crisis del Software. Esta aproximación era una respuesta de naturaleza defensiva ante los problemas que se habían detectado en el desarrollo de software.

Estos problemas se encontraban en la incorrecta estimación y en la compleja ejecución técnica del proyecto. Estas metodologías nos proponen solucionarlos definiendo correctamente el alcance de los proyectos y resolviendo las dificultades técnicas antes de comenzar la ejecución de un proyecto. Para definir el alcance nos proponen una detallada especificación de los requisitos, para eliminar en lo posible su ambigüedad. Para eliminar las dificultades técnicas que podamos encontrarnos en el entorno tecnológico volátil en el que se suelen desarrollar los proyectos, proponen una primera fase donde se detalle la solución técnica que debe ejecutarse, tratando de despejar cualquier duda tecnológica antes de comenzar.

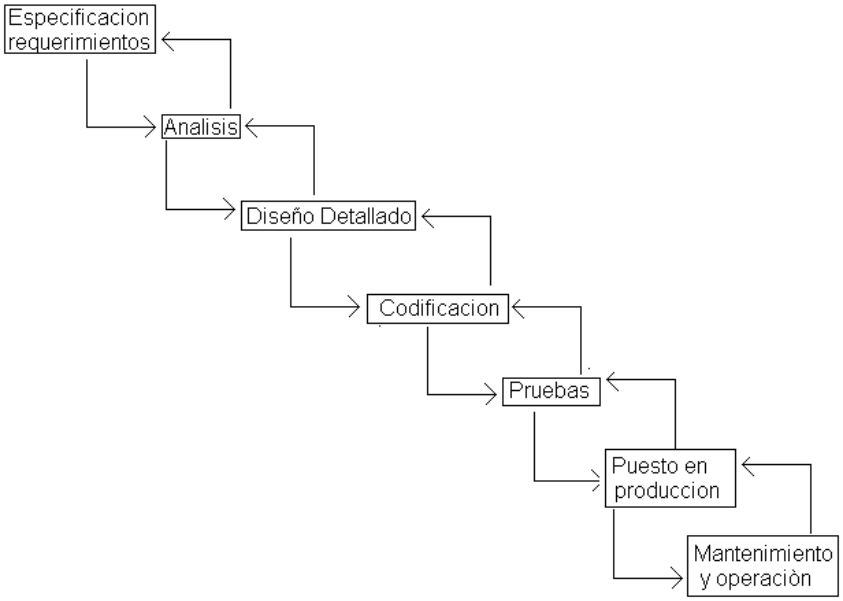
Estos problemas se hicieron evidentes debido a la crisis del software donde la estimación de esfuerzo era claramente insuficiente para los ambiciosos alcances que se definían y que a al final era generalmente la razón del fracaso del proyecto.

La estimación de un proyecto de desarrollo tiene tres variables principalmente: el tiempo, el coste y los requisitos. Podríamos incluir la calidad como la cuarta, pero dado lo complicado que es de especificarla no suele utilizarse en la estimación. En una oferta de servicios profesionales las variables de tiempo y de coste se

[21] CANTOR, Murray. Software Leadership: A Guide to Successful Software Development (Paperback). Addison-Wesley, NY, 2002, 234 P

especifican sin ningún tipo de ambigüedad, mientras que el alcance es muy difícil de detallar y su interpretación en muchos casos es subjetiva. Este es uno de los principales problemas que intentan resolver las metodologías tradicionales, intentando eliminar toda la ambigüedad al definir el alcance. Cuanto mayor sea el proyecto, mayor será el esfuerzo en definir el alcance de una forma clara y precisa. Se trata de conseguir delimitar lo que está contratado de lo que no lo está.

Figura 9. Procesos en Cascada 1.



Fuente. Murray Cantor. Software Leadership: A Guide to Successful Software Development (Paperback).. Addison-Wesley, NY, 2002, 234 P

Una definición que deberá en muchos casos entrar en detalle para ser realmente útil. Una vez definida esa delgada línea, servirá de referencia en todo el proyecto para defenderse de nuevos requisitos.

Definir el alcance del proyecto no resuelve complemente el problema de estimar el esfuerzo que será necesario para llevar a cabo los objetivos definidos. Si se consigue cerrar un alcance pero no se es capaz de estimar el esfuerzo necesario para llevarlo a cabo no se habrá conseguido reducir el riesgo que se trata de minimizar. La predicción del esfuerzo necesario sólo es posible basarla en la experiencia pasada acometiendo proyectos de similar complejidad y analizando el alcance de las tareas que se debe realizar. Para llevar a cabo una estimación es necesario profundizar en esa especificación inicial para abordar detalles de tipo técnico que permitan razonablemente conocer el esfuerzo relacionado con cada tarea a realizar. No vale con conocer los requisitos, se tendrá que conocer con que herramientas se cuenta, cuando se tiene disponibles otros sistemas con los que se tenga que integrar, etc. Este análisis permitirá tener un plan detallado que dirigirá el proyecto y un detalle sobre como se abordará técnicamente.

Antes de iniciar a desarrollar el proyecto se deben tener la mayoría de los detalles cerrados. Las metodologías tradicionales justifican este planteamiento puesto que con toda esta información se podrá encontrar la mejor solución técnica para el problema que se este planteando; ya que se podrá planificar el uso de recursos de una forma más óptima. Se tendrá toda la información lista para ser lo más eficientes y puntuales en la ejecución del proyecto, ya teniendo todos lo detalles de lo que se quiere realizar

La planificación nos dará un mapa exacto de cuál será nuestro viaje. Se tendrán todos lo detalles del camino que hay que recorrer. Siguiendo la metodología no hay que arrancar nuestro vehículo hasta que analizar la ruta, elegido el vehículo que mejor se adapte al recorrido e incluso programado nuestro vehículo para que adapte la conducción a las curvas que tendremos que tomar. Pero lo que no se sabe es cómo se comportaría en otros recorridos. No se tendrá la posibilidad de cambiar el recorrido a mitad de viaje porque eso obligaría a reprogramar el vehículo e incluso a cambiarlo, pero si el recorrido no cambia será probablemente que sea de los más rápidos.

2.7.1 ¿Cuáles son los problemas? La experiencia ha demostrado que las metodologías tradicionales no ofrecen una buena solución para proyectos donde el entorno es volátil y donde los requisitos no se conocen con exactitud, porque no están pensadas para trabajar con incertidumbre. No están preparadas para el cambio.

Aplicar metodologías tradicionales obliga a forzar al cliente a que tome todas las decisiones al principio. El verdadero problema no será el forzar a detallar este alcance con él. Detallarlo siempre ayudará a conocer mejor los requisitos y a que el cliente entienda mejor sus necesidades. El problema será que estos detalles hacen tomar decisiones que luego serán muy costosas de cambiar.

El cliente deberá ser capaz de describir y entender a un gran nivel de detalle para poder acordar un alcance del proyecto con él. Este alcance dará lugar a muchas decisiones técnicas que serán muy costosas cambiar y de las cuales en alguna ocasión el cliente no será consciente de las implicaciones que tienen. El cliente estará en una situación que puede ser muy incomoda para el.

Decidir el alcance de un proyecto que puede durar más de un año, en una primera fase que puede durar unos meses y donde no se avanzará nada en las necesidades a corto plazo que tiene.

La situación es más incómoda cuanto mayor grado de incertidumbre tenga sobre el alcance del proyecto. Esto también estará relacionado con el tamaño del proyecto. La incertidumbre será mayor cuanto mayor sea éste.

Ante el problema de tener que detallar todo prematuramente, el cliente suele adoptar una posición defensiva que es contraria a sus intereses. El cliente tratará que el sistema sea lo suficientemente flexible para poder cubrir las necesidades que pudieran surgir o cambiar. Por este motivo el sistema implementará ciertas funcionalidades que no se llegaran a utilizar. El cliente no es consciente de que cuando toma estas decisiones está introduciendo una complejidad en el proyecto que reducirá el alcance de otras funcionalidades y hará que el sistema sea más complejo de gestionar y de extender. Limitará el alcance de su proyecto final porque para el mismo esfuerzo de desarrollo tender menos funcionalidades de valor para él. Quizás una negociación hábil pueda conseguir que la empresa de servicios se vea obligada a realizar mayor esfuerzo del que estimó, pero sin duda esto también será un problema para él.

Para tomar estas decisiones el cliente tiene como única realimentación la documentación técnica que se va generando en la primera fase de diseño. Dado que esto es claramente insuficiente en muchos casos se ayuda al cliente con prototipos para llevar a cabo estas decisiones. Pero esto tampoco será realmente útil, puesto que el usuario final no suelen ser las personas del cliente responsables de definir el alcance. Muchos de las necesidades del software se encontrarán cuando el software ponga en producción y comience a ser utilizado por sus usuarios finales.

Las decisiones que vaya tomando el cliente serán irrevocables o muy costosas. Pero ¿por qué son estas decisiones tan costosas de cambiar? ¿Por qué ciertas decisiones no pueden ser tomadas conforme avance el proyecto? ¿Cuál es el objetivo del proyecto? ¿Acabar en tiempo y hacerlo técnicamente perfecto, o maximizar el valor que aporta al cliente? ¿En que caso es más probable que el cliente siga contando con la empresa de servicios: cuándo el proyecto ha tenido éxito para el cliente o cuando se ha realizado técnicamente de forma perfecta? Quizás haya que ponderar la satisfacción del cliente tanto como la excelencia técnica. Después de estudiar las metodologías tradicionales el punto a seguir es el estudio de las principales características de las metodologías ágiles.

2.8 METODOLOGÍAS ÁGILES

Las siguientes son las ventajas que ofrecen las metodologías ágiles para el desarrollo de Software:

- Al individuo y las interacciones del equipo para desarrollo sobre el proceso y las Herramientas. La gente es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que este configure su propio entorno de desarrollo en base a sus necesidades.

- Desarrollar software que funciona para conseguir más que una buena documentación. La regla a seguir es no producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante. Estos documentos deben ser cortos y centrarse en lo fundamental.

- La colaboración con el cliente es más que la negociación de un contrato. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su Éxito.

- Responder a los cambios más que seguir estrictamente un plan. La habilidad de responder a los cambios que puedan surgir a los largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

Los valores anteriores inspiran los doce principios del manifiesto ágil. Son características que diferencian un proceso ágil de uno tradicional. Los dos primeros principios son generales y resumen gran parte del espíritu ágil. El resto tienen que ver con el proceso a seguir y con el equipo de desarrollo, en cuanto metas a seguir y organización del mismo. Los principios son:

- La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de Software que le aporte un valor.

- Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.

- Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.

- La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.

- Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.

- El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.

- El software que funciona es la medida principal de progreso.

- Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.

- La atención continua a la calidad técnica y al buen diseño mejora la agilidad.

- La simplicidad es esencial.

- Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.

- En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento

2.8.1 Metodologías ágiles vs Metodologías tradicionales. A continuación se presentará un cuadro comparativo que evidencia las ventajas y desventajas entre estas metodologías para el desarrollo de proyectos.

Tabla 1. Diferencias entre metodologías ágiles y no ágiles.

METODOS AGILES	METODOLOGIA TRADICIONALES
Basadas en heurísticas provenientes de practicas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas de internamente (por el equipo).	Impuesta Externamente
Procesos menos controlados	Proceso mucho mas controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Mas artefactos
Pocos roles	Mas roles
Menos énfasis en la arquitectura	La arquitectura de software es esencial y se expresa mediante modelos

Fuente. CANOS, José; LETELIER, Patricio Y PENADES, Mª Carmen. Metodologías Ágiles en el Desarrollo de Software, DSIC -Universidad Politécnica de Valencia.

2.8.2 Algunas metodologías ágiles. Aunque los creadores e impulsores de las metodologías ágiles más populares han suscrito el manifiesto ágil y coinciden con los principios enunciados anteriormente, cada metodología tiene características propias y hace hincapié en algunos aspectos más específicos. A continuación se resumen otras metodologías ágiles. La mayoría de ellas ya estaban siendo utilizadas con éxito en proyectos reales pero les faltaba una mayor difusión y reconocimiento.

- SCRUM. Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.

- Crystal Methodologies. Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño

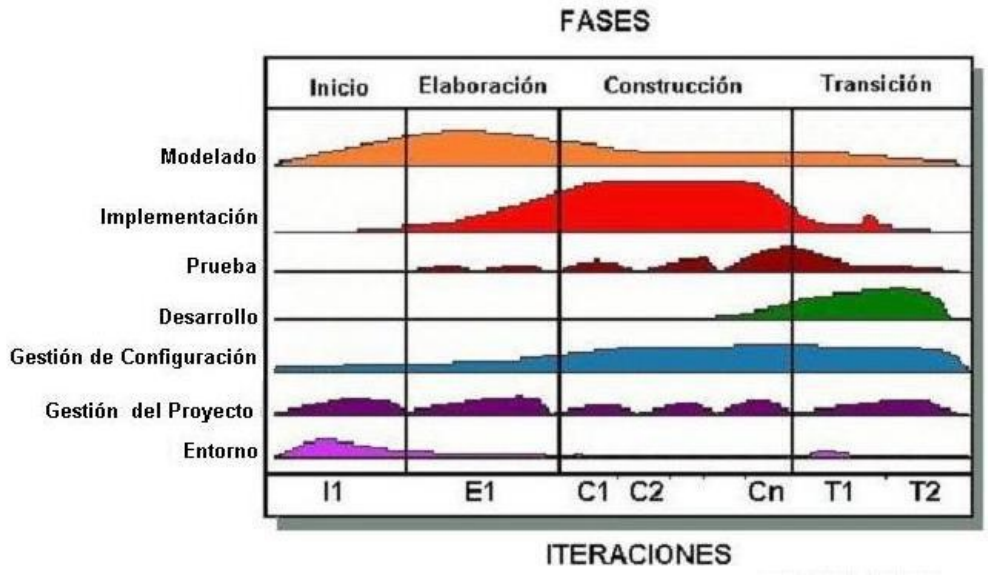
del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros).

- Dynamic Systems Development Method (DSDM). Define el marco para desarrollar un proceso de producción de software. Nace en 1994 con el objetivo de crear una metodología RAD unificada. Sus principales características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación. Las tres últimas son iterativas, además de existir realimentación a todas las fases

- Adaptive Software Development (ASD). Su impulsor es Jim Highsmith. Sus principales características son: iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. El ciclo de vida que propone tiene tres fases esenciales: especulación, colaboración y aprendizaje. En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda desarrollan las características y finalmente en la tercera se revisa su calidad, y se entrega al cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo
  - Feature -Driven Development (FDD). Define un proceso iterativo que consta de 5 pasos. Las iteraciones son cortas (hasta 2 semanas). Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software. Sus impulsores son Jeff De Luca y Peter Coad.
  - Lean Development (LD). Definida por Bob Charette.s a partir de su experiencia en proyectos con la industria japonesa del automóvil en los años 80 y utilizada en numerosos proyectos de telecomunicaciones en Europa[22]

2.9 METODOLOGÍA ÁGIL UP

Figura 10. Ciclo de vida AUP.



Fuentes. AMBLER, Scott. Ambysoft. Metodología AUP. [En línea]. 2005. [Citado el 10 de enero del 2007]. Disponible en Internet en: <http://www.ambysoft.com/unifiedprocess/agileUP.html>

[22] FOWLER, M; BECK, K and BRANT, J. Refactoring: Improving the Design of Existing Code. Laboratorio de Sistemas de Información. Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia .Addison - Wesley. 1999. 17 P

Ágil UP (AUP) es una versión simplificada del Proceso Racional Unificado (RUP). Se describe como una metodología de desarrollo de aplicaciones software simple y fácil de entender, la cual usa técnicas ágiles y conceptos básicos del RUP. AUP captura su naturaleza sucesiva en estas cuatro fases:

A continuación la figura 10 complementa la explicación de los ciclos de vida.

- Inicio. El objetivo de esta fase es de identificar el alcance inicial del proyecto, una arquitectura potencial para su sistema y obtener los fundamentos iniciales y la aceptación del equipo de trabajo del proyecto.

- Elaboración. El objetivo es aprobar la arquitectura del sistema.

- Construcción. El objetivo de esta fase es de construir el software en una base regular, incremental que encuentra las necesidades de mayor prioridad para el equipo de trabajo y los usuarios finales.

- Transición. El objetivo es de validar y desplegar el sistema en su ambiente de producción.[23]

Las disciplinas en el AUP son realizadas de una manera iterativa, definiendo las actividades que los miembros de equipo de desarrollo realizan para construir, validar, y entregar el software que sigue con los requerimientos dados por los usuarios finales. Las disciplinas son:

- Modelado. El objetivo de esta disciplina es de entender la organización del negocio, el dominio de problema orientado por el proyecto e identificar una solución viable del problema.

- Implementación. El objetivo de esta disciplina es de transformar el o los modelos en código ejecutable y realizar pruebas a nivel básico.

- Prueba. El objetivo de esta disciplina es realizar una evaluación objetiva para asegurar la calidad. Esto incluye descubrimiento de defectos, validez que el sistema trabaja para lo que fue diseñado y verificación de los requerimientos.

- Desarrollo. El objetivo de esta disciplina es planificar para la entrega del sistema y ejecutar el plan de hacer el sistema útil a usuarios finales.

- Gestión de Configuración. El objetivo de esta disciplina es de manejar el acceso a los mecanismos de proyecto. Esto incluye no sólo versiones de mecanismos de rastreo sobre el tiempo sino también el control y la dirección de los cambios que se le realicen.

- Gestión del Proyecto. El objetivo de esta disciplina es de dirigir las actividades que ocurren sobre el proyecto. Esto incluye riesgos directivos, dirección a la gente (asignar tareas, rastrear el progreso, etc.), y coordinación gente y sistemas fuera del alcance del proyecto para asegurar que se esta entregado a tiempo y dentro del presupuesto.

- Entorno. El objetivo de esta disciplina es de apoyar el resto del esfuerzo por asegurando que el proceso apropiado, la dirección (normas y directrices), e instrumentos (el hardware, el software, etc.) están disponibles para el equipo.

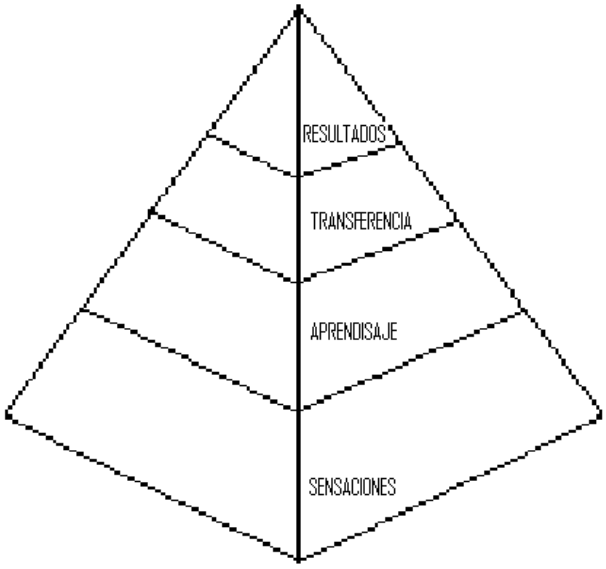
El método que permitirá evaluar las sensaciones que causa el Demo de Juego PEI en los usuarios, es el primer nivel de Kirkpatrick, a continuación se encuentra la explicación de cada uno de los cuatro niveles de este método de evaluación.

[23] CANOS, José; LETELIER, Patricio Y PENADES, Mª Carmen. Metodologías Ágiles en el Desarrollo de Software, DSIC - Universidad Politécnica de Valencia. Alicante, España. Noviembre 2003. Editado Por: Patricio Letelier Torres Emilio A. Sánchez López, Grupo ISSI. 59 P

2.10 LOS CUATRO NIVELES DE EVALUACION DE KIRKPATRICK

La utilización del modelo implica cuatro niveles de desarrollado de Donald Kirkpatrick. Según este modelo, la evaluación siempre deberá comenzar con el nivel uno, y luego, dependiendo del tiempo y el presupuesto, deberían moverse secuencialmente por los niveles dos, tres y cuatro. La información de cada nivel anterior sirve como base para la evaluación del siguiente nivel. Así, cada nivel superior representa una medida más exacta de la eficacia del programa de adiestramiento, al mismo tiempo requiere un análisis más riguroso que lleva mucho tiempo. La figura 11 dará un preámbulo de lo que tratan los cuatro niveles de Kirkpatrick.

Figura 11. En el modelo de cuatro niveles de Kirkpatrick.



Fuente. WINFREY, E.C. Kirkpatrick's Four Levels of Evaluation. In B. Hoffman (Ed.), Encyclopedia of Educational Technology. [En línea]. 1999. [Citado el 20 de Agosto del 2006]. Disponible en Internet en: <http://coe.sdsu.edu/eet/Articles/k4levels/start.htm.>

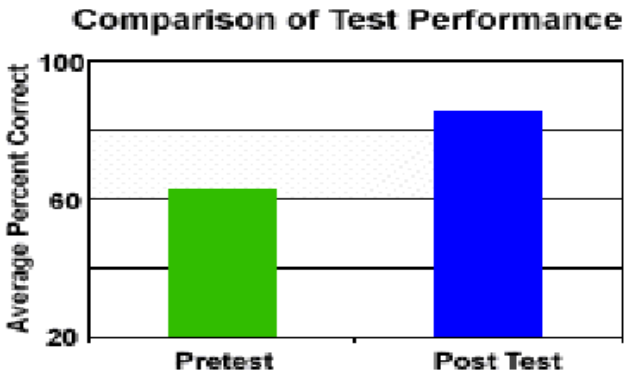
2.10.1 Nivele 1 Evaluación – Reacciones. La evaluación que se realiza en este nivel básicamente debe supervisar el adiestramiento de los participantes a la reacción al programa. ¿Esto intenta contestar que las preguntas de las percepciones de los participantes?, ¿A ellos les gusta esto? ¿El material era relevante a su trabajo?, A menudo llaman este tipo de evaluación un "smilesheet".

Según Kirkpatrick, cada programa al menos debería ser evaluado en este nivel para asegurar un mejoramiento al programa de adiestramiento. Además, las reacciones de los participantes tienen consecuencias importantes para aprender. Aunque una reacción positiva no garantice el estudio, una reacción negativa casi seguramente reduce su posibilidad.

Se va llevar a cabo una prueba piloto para evaluar el demo del Juego Educativo PEI basándose en este primer nivel de la teoría de Kirkpatrick que permita establecer el nivel de satisfacción de los usuarios. Esta información se puede consultar en capitulo 4.6

2.10.2 Nivele 2 Evaluación – Estudio. En este nivel la evaluación va más allá de la satisfacción del principiante e intenta evaluar a los estudiantes de grado avanzado en habilidades, conocimiento, o actitud. La medida en este nivel es más difícil y laboriosa que el nivel uno. Los métodos se extienden de pruebas formales a pruebas informales, para combinar la evaluación y la autovaloración. De ser posible, los participantes toman la prueba o la evaluación antes el se entrenados (pre - prueba) y después del entrenamiento (post - prueba) que determinar el aumento de aprendizaje que ha ocurrido. Un ejemplo de la estadística que nos arroja como resultado este nivel es la figura 12

Figura 12. Para evaluar el aumento de aprendizaje que ha ocurrido debido a un programa de adiestramiento.

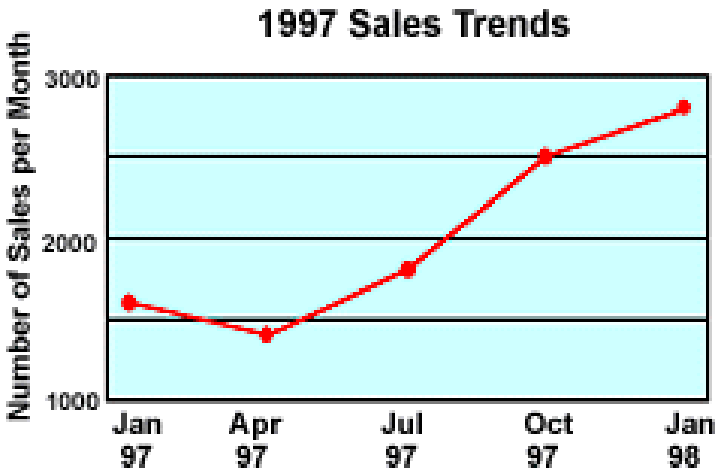


Fuente., Ibíd.

2.10.3 Nivele 3 Evaluación – Transferencia**.** Este nivel mide el cambio en el comportamiento de los principiantes debido al programa de adiestramiento. La evaluación en este nivel intenta contestar la pregunta: ¿son habilidades, conocimiento recientemente adquiridas, o es la actitud que se usa en el ambiente diario del principiante? Para muchos entrenadores este nivel representa la evaluación más verdadera de la eficacia de un programa. Sin embargo, midiendo este nivel es imposible predecir cuando el cambio de comportamiento ocurrió, y así requiere decisiones importantes en términos de ¿cuándo evaluar?, ¿qué a menudo evaluar?, ¿y cómo evaluar?

2.10.4 Nivele 4 Evaluación – Resultados. Este nivel mide el éxito del programa en términos de condiciones que los gerentes y ejecutivos pueden entender (la producción aumentada) como lo muestra la figura 13, se mejora la calidad, reduce costos, la frecuencia reducida de accidentes, aumentó ventas, y ganancias aún más altas o rendimiento de la inversión. De una perspectiva de negocio y de organización, esta es la razón de un programa de adiestramiento, que nivela cuatro resultados típicamente no dirigidos. La determinación causa condiciones financieras difíciles de medir, y estas son complicadas de unir directamente con la educación también llamado el entrenamiento.

Figura 13. Nivele cuatro: intenta evaluar la educación (el entrenamiento) en términos de resultados de negocio.



Fuente., Ibíd.

2.10.2 Métodos para Evaluación a Largo Plazo.

- Enviar revisiones que se post-entrenan.
- La Oferta la educación (el entrenamiento) en curso, ordenada y el entrenamiento por el período del tiempo.
- La continuación de Conducta necesita la evaluación.
- El métrico de Comprobación (por ejemplo, el trozo, adaptar, errores, etc.) para medir si los participantes alcanzaran objetivos que se entrenan.
- Entrevistan a aprendices y sus gerentes, o sus grupos de cliente (por ejemplo, pacientes, otro personal departamental)

3. FORMULACIÓN Y DESARROLLO DE LA INFORMACIÓN

La Facultad de Ingeniería de Sistemas ha planteado como proyecto de grado un Demo de un juego multimedia educativo desarrollado sobre el motor A6, diseñado bajo criterios de la teoría instruccional. Este proyecto pretende crear espacios amenos para el aprendizaje del PEI, ya que de acuerdo a estudios realizados, una forma eficiente y eficaz de aprendizaje para este tipo de temáticas, es través de los juegos debido al factor motivador que estos ejercen sobre la persona que debe aprender y apropiar dichos conocimientos.

En el transcurso del segundo semestre del 2006 y lo que va del 2007 se han realizados varios avances que a continuación serán enunciados.

Para la creación del Demo fue necesario un guión que básicamente consiste en la historia en la cual se va a desarrollar el juego plasmada en papel, a partir de este guión se realizó la ingeniería de software necesaria para modelar las ideas en el Demo.[24]

3.1 DEMO DEL JUEGO DEL PEI

Inicialmente se había planteado para este proyecto el diseño y desarrollo de un juego, pero tomando en cuenta aspectos que antes habían sido ignorados como: definiciones de juegos, conceptos de desarrollo como motores o lenguajes de programación de juegos y procesos a seguir para el diseño, se cree que el mejor término a utilizar es *Demo.*

Este proyecto sólo esta planteado para contribuir a la acreditación de la facultad de Ingeniería de Sistemas por consecuente sólo será aplicado a esta facultad.

También se presentaron los siguientes conflictos en el proceso de diseño en flash:

- Saturación del servidor a causa del tamaño de la aplicación
- No se permite actualizaciones y manipulaciones en el código para futuras mejoras
- No se permite implementar el modo de juego multi-jugador
- El acción script no permite conexiones a base de datos complejas

Después de una investigación sobre flash y algunos motores para juegos se concluyó que flash no es un motor apropiado para el desarrollo de un juego multijugador, debido a que no es un motor de juego sino una

[24] SCHWABER, K; BEEDLE, M. Agile Software Development with SCRUM. Prentice Hall. Stockholm, Sweden. 2001158 P

herramienta para animación. Por este motivo, muchas de las acciones que se requieren para la elaboración del Demo no se pueden hacer o representan gran complejidad. Que a medida que pasa el tiempo podrían representar un conflicto en la actualización del mismo.

Otro factor influyente fue la no reutilización de código para figuras y objetos (lo que normalmente se llama clonación) lo cual representa más líneas de código y más espacio del navegador donde se va a cargar el Demo, lo que genera un Demo que poca gente podría utilizar por su latencia.

Otro motivo por el cual flash no se acopla al desarrollo de este Demo, es porque no maneja base de datos complejas debido que flash no fue creado con propósitos de construcciones de diseño de juegos si no de animación y desarrollo de Web.

Otro inconveniente de flash es la tasa de transferencia de los datos, puesto que es alta y la cantidad a trasmitir es aproximada a 1mg por segundo para que el Demo se pueda desarrollar en los menores niveles de calidad, teniendo en cuenta que es una considerable de imágenes y código.

A consecuencia de los factores anteriormente mencionados se decidió que flash no es la herramienta indicada para el desarrollo de un Demo con los parámetros requeridos, entonces se procedió a la búsqueda de un motor acorde a las especificaciones del Demo; y se llegó a la conclusión de que A6 es la mejor opción, debido que es un motor dedicado a la clase de juegos que utilizan vistas, búsquedas y acertijos.

En el Proceso de la elaboración del demo del juego PEI se ha determinado que por motivos de complejidad y tiempo que los actores serán dos: profesores–administrativos y estudiantes–egresados. También el acceso será limitado solo al edificio L y casas E, F, G, H, I, J, K, los otros edificios como la Biblioteca, el edificio administrativo, edificio D y demás estructuras sólo quedaran diseñadas pero sin acceso.

Teniendo en cuenta el objetivo real de este Demo “enseñar el PEI de una forma divertida y atractiva” y después de analizarlo se decidió que se realizará tipo aventura por roles, tomando como referencia el juego de ¿where is in USA carmen san diego?, que trata de buscar pistas, resolviendo acertijos hasta encontrar a Carmen San Diego.

3.2 CASOS DE USO Y SUS ESPECIFICACIONES

Entre el análisis y estudio previo que se realizó para el desarrollo del Demo del juego PEI, sirviendo como base fundamental, se encuentra la ingeniería de software basada en la metodología ágil AUP antes mencionada y estudiada, donde se elaboraron casos de uso y sus respectivas plantillas, ver anexos A y B.

3.3 MANUAL PARA EL DISEÑO INSTRUCCIONAL DE JUEGOS MULTIMEDIA

La teoría que se escogió para el diseño del Demo del juego PEI fue la teoría instruccional, de la cual se ha elaborado un manual que permita a futuros creadores de videojuegos multimedia guiarse de una forma fácil de entender y aplicar, ver documento anexo.

3.4 ENCUESTA DE SATISFACION Y SENSACIONES

Para medir el nivel de satisfacción y conocer las sensaciones que el Demo del juego PEI produjo en los jugadores se ha planteado como referencia el primer nivel de Kirkpatrick explicado en el capitulo anterior. El método que se ha escogido es la encuesta. Ver anexo C.

4. CONCLUSIONES

Se definió que el producto final será un Demo del Juego Multimedia Para El Autoaprendizaje Del Proyecto Educativo Institucional De La Universidad Autónoma De Bucaramanga.

La opción más optima para el diseño y desarrollo del Demo PEI es el motor A6.

La metodología que más de acopla para el desarrollo de este proyecto de grado es AUP porque no se requiere de contacto con el usuario final y sus etapas se realizan simultáneamente. Permitiendo cambios cuando sean requeridos.

El demo del juego PEI es limitado a la Facultad de Ingeniería de Sistemas y algunas aulas del edificio de Ingenierías y sus alrededores.

Se tomó cómo referencia el juego ¿where is in USA carmen san diego?, porque presenta afinidad con las metas que se quieren lograr con este proyecto.

El tipo de juego aventura y roles es la mejor forma de enseñar, es el tipo de juego más apetecido por las personas.

El interés de los jugadores radica en que el videojuego presente una historia interesante que motive a jugar.

5. TRABAJOS FUTUROS

Crear la segunda versión, habilitando el resto de edificios como la biblioteca, el administrativo, CSU, hasta el campus del bosque. Básicamente hacerlo mas real con mas actores que interactúen con los jugadores como por ejemplo “audiovisuales”, mejorar el nivel de detalles y gráficos. Arreglar y actualizar los campus.

Esta continuación pondría proponerse como proyecto de grado de la Facultad de Ingeniería de Sistemas de la UNAB.

BIBLIOGRAFÍA

AMBLER, Scott. Ambysoft. Metodología AUP. [En línea]. 2005. [Citado el 10 de enero del 2007]. Disponible en Internet en: <http://www.ambysoft.com/unifiedprocess/agileUP.html>

Blizzard Entertainment. Juego command and conquer (EA), warcraft Blizzard. [En línea]. [Citado el 17 de agosto del 2006]. Disponible en Internet en: <www.blizzard.com>

CANOS, José; LETELIER, Patricio Y PENADES, Mª Carmen. Metodologías Ágiles en el Desarrollo de Software, DSIC -Universidad Politécnica de Valencia. Alicante, España. Noviembre 2003. Editado Por: Patricio Letelier Torres Emilio A. Sánchez López, Grupo ISSI. 59 P

CANTOR, Murray. Software Leadership: A Guide to Successful Software Development (Paperback). Addison-Wesley, NY, 2002, 234 P
Categoría de juegos. [En línea]. [Citado el 1 de Abril del 2006]. Disponible en Internet en: <http://www.eumed.net/cursecon/libreria/bg-micro/5.htm/>

COAD, P; LEFEBVRE, E and DE LUCA, J. Java Modeling In Color With UML: Enterprise Components and Process. Prentice Hall Londres, Inglaterra, 1999. 215 P

CONITEC DATASYSTEMS. Motor A6. [En línea]. [Citado el 20 de marzo del 2007]. Disponible en Internet en: <http://www.conitec.net/german/gstudio/3dgs2.htm.>

DOOM (Atari), Battlefield 2 (EA): [En línea]. 2002. [Citado el 21 de agosto del 2006]. Disponible en Internet en: <www.ea.battlafied.com>.

EBERLY, David H. 3D game engine design: a practical approach to real-time computer graphics. 2nd ed.Amsterdam; Boston: Elsevier Morgan Kaufmann, c2007. 187 P

ESCUDERO Sofía, Macromedia flash MX, McGraw Hill/ Iberoamericana de España, 1 ed. 2002. 320 P

EXEBlog. Categoría de juegos. [En línea]. 2006. Valencia, España. [Citado el 22 de Agosto del 2006]. Disponible en Internet en: <http://www.exelweiss.com/blog/categoria/juegos-educativos/>

Flight Simulator (Microsoft). [En línea]. [Citado el 23 de agosto del 2006]. Disponible en Internet en: <www.microsoftgames.com>

FOWLER, M; BECK, K and BRANT, J. Refactoring: Improving the Design of Existing Code. Laboratorio de Sistemas de Información. Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia .Addison - Wesley. 1999. 17 P

IGN ENTERTAINMENT GAMES. Desarrollo de juegos. [En línea]. 1996. [Citado el 26 de Agosto del 2006]. Disponible en Internet en: <http://www.ign.com/>

Introducción a la teoría de juegos. Tipo de juegos. [En línea]. [Citado el 20 de Agosto del 2006]. Disponible en Internet en: <http://www.eumed.net/cursecon/juegos/>

MADDEN NFL 08. Deportes. FIFA (EA). [En línea]. [Citado el 21 de agosto del 2006]. Disponible en Internet en: <www.easports.com>

MARQUEZ RAMOS, Ángel. La sociedad ante los avances tecnológicos, marzo del 2006, Universidad Interamericana de Puerto Rico, Recinto de Aguadilla. Departamento de Ciencias y Tecnología. Puerto rico, 10 P

Microsoft. DirectX. [En línea]. [Citado el 25 de Marzo del 2007]. Disponible en Internet en: <www.microsoft.com/directx.htm>

MotorUnrealengine3. [En línea]. 1997. [Citado el 25 de marzo del 2007]. Disponible en Internet en: <http://www.3dpoder.com/foro3dpoder/showthread.php?t=25830>

Quest3D. Historia de los videojuegos. [En línea]. 2000. [Citado el 17 de agosto del 2006]. Disponible en Internet en: <http://www.quest3d.com/index.php?id=17>

SCHWABER, K; BEEDLE, M. Agile Software Development with SCRUM. Prentice Hall. Stockholm, Sweden. 2001158 P

UNIVERSIDAD AUTONOMA DE BUCARAMANGA. Plan educativo institucional. 2005.

UNIVERSIDAD AUTONOMA DE BUCARAMANGA. Reglamento docente y estudiantil. 2005

WINFREY, E.C. Kirkpatrick's Four Levels of Evaluation. In B. Hoffman (Ed.), Encyclopedia of Educational Technology. [En línea]. 1999. [Citado el 20 de Agosto del 2006]. Disponible en Internet en: <http://coe.sdsu.edu/eet/Articles/k4levels/start.htm.>

World Warcraft. Nuevo tipo de juego MMORPG. [En línea]. [Citado el 18 de Agosto del 2006]. Disponible en Internet en: <www.worldofwarcraft.com>