UNIVERSIDAD MILITAR
NUEVA GRANADA

CREACIÓN DE PROTOTIPO DE VIDEOJUEGO A TRAVÉS DEL MODELO DE DESARROLLO SCRUM Y LA ESTRUCTURA NARRATIVA SIMPLE DE DAVID SIEGEL

CLAUDIA JANNETH BUITRAGO BUITRAGO
GONZALO DÁNIEL FERREIRA

TESIS

DIRECTOR
CHRISITAN QUINTERO
INGENIERO EN MULTIMEDIA

UNIVERSIDAD MILITAR NUEVA GRANADA
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA EN MULTIMEDIA
BOGOTA
2012

CREACIÓN DE PROTOTIPO DE VIDEOJUEGO A TRAVÉS DEL MODELO DE DESARROLLO SCRUM Y LA ESTRUCTURA NARRATIVA SIMPLE DE DAVID SIEGEL

GONZALO DÁNIEL FERREIRA

CLAUDIA JANNETH BUITRAGO BUITRAGO

Trabajo de grado para optar el título de Ingeniero(a) en Multimedia

Director
CHRISTIAN QUINTERO
Ingeniero en Multimedia

UNIVERSIDAD MILITAR NUEVA GRANADA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA EN MULTIMEDIA
BOGOTÁ, D.C.

2012

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

Bogotá D.C. Enero de 2012

-DEDICATORIA-

A mi madre, Glenda Ferreira Prada, por su eterna paciencia y apoyo en la búsqueda de mis sueños y proyectos de vida. A mi hermana Juliana Dániel por ser más que todo amiga y compañera y a mi Padre, Gonzalo Augusto Dániel que brindó los medios para llevar este proceso de aprendizaje a término.

Gonzalo Dániel Ferreira

- DEDICATORIA-

“Cuando hagas algo, hazlo bien, con dedicación y empeño, y lo mas importante: con el corazón.” Anónimo

A mis padres Yolanda Buitrago y Luis Buitrago, por su apoyo constante y por ser símbolo de superación y entrega. A mis hermanos y abuelita, por ser fuente diaria de inspiración y apoyo incondicional. A mis amigos por motivarme con su energía, y demás personas que directa o indirectamente hicieron posible la culminación de esta etapa.

Claudia Buitrago

- AGRADECIMIENTOS -

Los Autores expresan sus agradecimientos a:
Christian Quintero, Ingeniero en Multimedia, profesor de tiempo completo de la Universidad Militar Nueva Granada por su asesoría en el desarrollo de este proyecto.

Luis Vallejo, asesor de tecnología en el Tecnoparque del SENA, por su asistencia en la creación e implementación de animaciones creadas con tecnología de captura de movimiento dentro del videojuego.

Rodrigo Castro, por su asistencia en la afiliación de este proyecto como parte del SENA y permitir el uso de las instalaciones de captura de movimiento.

Tatiana Rodríguez, por proveer las voces que caracterizaron a la protagonista del videojuego.

Juan Camilo Martínez, Comunicador Social, por la asistencia en la creación de videos y secuencias animadas.

Andrea Buitrago, por permitir tomar como referencia su brazo para la protagonista.

El Tecnoparque del SENA por brindar sus instalaciones y equipos para la codificación de animaciones por captura de movimiento.

Comunidad del Foro de UDK, sin ellos este trabajo no hubiera sido posible.

Y a las demás personas que hicieron posible llevar a buen término este trabajo de grado.

Resumen

En el desarrollo de un videojuego se encuentran diversos enfoques que resultan ser fundamentales al momento de crear un factor diferencial con respecto a lo ya encontrado en el mercado. Históricamente se ha visto una prelación a mecánicas de juego y aspectos técnicos por encima de una estructura narrativa, generando problemas estructurales en éste, teniendo como consecuencia una falta de claridad en las motivaciones del jugador o un desenlace donde no se logra proveer la satisfacción esperada. Este fenómeno se debe a la modificación de la estructura narrativa alrededor de eventos que demuestran fortalezas técnicas o de mecánicas de juego. El Ingeniero en Multimedia posee la capacidad de integrar estos dos enfoques de manera objetiva en la creación de un videojuego, planteando y desarrollando un producto que es la muestra de dicha asociación, creando una pauta para futuros desarrolladores.

Por medio de una estructura narrativa y una metodología de desarrollo definidas se creó un prototipo de videojuego dónde la prioridad fue implementar su historia, para luego analizar la relevancia que posee este enfoque por medio de pruebas a diferentes jugadores.

Como resultado se obtuvo una retroalimentación positiva del prototipo de videojuego, juzgándolo como medio de entretenimiento, capacidad de brindar una experiencia narrativa y exponente del género al que pertenece.

CONTENIDO

pág.

5. PRUEBAS Y RESULTADOS EXTERNOS

LISTA DE TABLAS

pág.

LISTA DE FIGURAS

pág.

LISTA DE ANEXOS

pág.

INTRODUCCIÓN

En este trabajo se muestra el planteamiento, creación y resultados al implementar una metodología de desarrollo en la creación de un prototipo de videojuego, dónde se da prelación a una estructura narrativa. Existen diversos factores que influyen en el resultado final de un juego, los cuáles se pueden reducir a tres ramas: técnicas (calidad de gráficos, sonidos, música), mecánicas de juego y estructura narrativa. De acuerdo a la evolución tecnológica de los motores para desarrollo de videojuegos, sus características técnicas equivalentes entre sí y una nueva tendencia de distribución gratuita, la capacidad tecnológica pasa a segundo plano y entra a protagonizar la rama narrativa, convirtiéndose en el factor diferencial más relevante entre los videojuegos actuales. Con esto en mente, se decidió tomar un enfoque narrativo ya establecido en la industria del entretenimiento, siendo ésta la perspectiva más ligada a la composición creativa de un juego e independiente en esencia de las limitaciones técnicas del mismo. A continuación se sometieron a valoración diferentes motores gráficos de videojuegos y sus ambientes de desarrollo, valorando factores como costos, calidad visual y auditiva, requerimientos técnicos e intuitividad de la interfaz y su respectiva documentación. Se eligió una metodología de desarrollo que permite una retroalimentación constante en función del avance del prototipo de videojuego como tal uniéndola con una estructura lineal, cumpliendo con los múltiples requerimientos exigidos por el argumento planteado en la narrativa y las necesidades funcionales del mismo, como animaciones ayudadas por captura de movimiento y grabaciones de voz, entre otros. Todos estos fueron implementados en un nivel enfocado a un público mayor de 18 años. Por último se realizaron pruebas al videojuego y a la metodología implementada para así analizar los resultados y brindar conclusiones relevantes a los objetivos planteados.

1. OBJETIVOS

1.1 OBJETIVO GENERAL

Integrar mediante un prototipo de videojuego la estructura de narrativa simple propuesta por David Siegel usando la metodología de desarrollo de software SCRUM.

1.2. OBJETIVOS ESPECÍFICOS

- Desarrollar un prototipo de videojuego en perspectiva de primera persona.

- Unificar la metodología de desarrollo SCRUM con la estructura de narrativa de David Siegel en la creación de un prototipo de videojuego.

- Someter el prototipo de videojuego y su proceso de desarrollo a una etapa de evaluación para obtener una retroalimentación del proyecto de grado y así analizar cuál es el impacto que tiene la narrativa en el género desarrollado.

2. MARCO TEÓRICO

2.1. ANTECEDENTES

En el ámbito de los juegos de video, la industria se ha dirigido en diversas direcciones en lo relacionado al género de los FPS (First Person Shooters o juegos de disparar con perspectiva de primera persona), los cuáles vienen siendo vigentes desde 1992. Todos han trabajado diversas temáticas, desde la segunda Guerra Mundial (Wolfenstein 3D [1] , Medal of Honor [2] , Call of Duty[3], etc...), pasando por las guerras entre pandillas (Kingpin: Life of a Crime [4]) hasta invasiones por seres de otras dimensiones (Half Life [5]). Todos estos exploran nuevas formas de narrativa, con el objetivo de generar una experiencia más relevante para el jugador. Históricamente podemos hablar de los primeros juegos de video como experiencias centradas en las mecánicas de juego, ya que las limitaciones de los sistemas de la época de los ochenta y noventa encontraban la forma de expresar su información en imágenes básicas.

A medida que se desarrollan nuevas tecnologías que son capaces de almacenar y mostrar mayores cantidades de información, las exigencias del “por qué”, el “cómo” y el “dónde” del juego se hacen cada vez más necesarias, lo que obliga a los diseñadores a argumentar los contextos y motivaciones de los usuarios para jugar y completar cada juego. Es a esta altura donde se crean diferentes enfoques o géneros los cuales se pueden dividir de una forma muy general, entre acción, estrategia y aventura. Tradicionalmente cada género ha dependido de una mayor o menor proporción de contextos e historias que los fundamenten, y con estos, técnicas narrativas para hacerlos explícitos de la mejor manera posible. Es así como hemos visto a través del tiempo nuevos géneros donde se experimentan nuevas formas de contar historias, y se interactúa con un medio susceptible a las decisiones y acciones del jugador entregando así una nueva experiencia en

entretenimiento donde el jugador deja de ser un observador externo y pasa a sentirse parte del juego.

Tenemos los casos específicos de juegos que han logrado demostrar su gran impacto a nivel histórico por el planteamiento revolucionario de su narrativa. Avanzando de forma cronológica encontramos el juego “Maniac Mansion” (Lucasfilm Games, 1987) [6] el cual brinda la posibilidad de obtener distintos finales dependiendo de las decisiones tomadas por el jugador en el transcurso del juego, lo que trasciende mas allá al esquema lineal usualmente encontrado hasta su época. Luego será iterado en los muy populares juegos de aventura desarrollados por Lucasarts (Day of the Tentacle[7], Full Throttle [8], entre otros).

Adicionalmente, la gran mayoría de juegos han seguido un esquema lineal en el desarrollo de su argumento, en gran parte debido a la proporción de la experiencia de usuario contra contenido. Es por esto que en el desarrollo y diseño de los videojuegos se ha dado prioridad a la narrativa dentro de un marco lineal, lo que ha llevado a teorizar estructuras de narración que logren la máxima inmersión del jugador. Como exponentes de estas tendencias tenemos clásicos como Half Life [5] donde la historia se desenvuelve por medio de eventos previamente programados que generan la impresión de eventos al azar, volviendo al jugador parte de la historia.

Actualmente, en el campo de desarrollo narrativo se encuentra Mass Effect y Mass Effect 2 [9], estos son juegos que combinan una narrativa que se desenvuelve en relación a las decisiones del jugador, inclusive guardando el progreso de la primera parte a la segunda, utilizando efectos avanzados de modelado de personajes, foto realismo en modelos y animaciones, actores de voz lo suficientemente convincentes para generar empatía y afectar profundamente las decisiones del jugador, todo siguiendo un riguroso esquema de diseño y narrativa.

A nivel de desarrollo en Colombia, existen diversas empresas desarrolladoras de videojuegos, cada una de ellas con distintos enfoques, tanto en plataforma de desarrollo, como en estilo de videojuegos desarrollados y tecnologías implementadas. En el desarrollo local se debe hablar de Immersion Games[10], compañía local desarrolladora de un juego en perspectiva de primera persona completo (CellFactor) [11]. Gracias a los avances tecnológicos se puede apreciar el desarrollo nacional de juegos en plataformas móviles como SocCars de Xoc Games [12] y The Clover´s Quest, de Gametron Studios LTDA [13].

En el programa de Ingeniería en Multimedia se han realizado varios prototipos de videojuegos como proyectos de grado, como lo son:

- Desarrollo de un prototipo de Videojuego basado en el cuento “Dalia Zair” del escritor colombiano Jairo Aníbal Niño. [14]

- Análisis y diseño para el desarrollo de un prototipo video juego 3D, sobre conflictos entre tribus indígenas en Colombia [15]

- Prototipo de videojuego El Mito de la Torre. [16]

- Prototipo de un videojuego de aventura, fantasía y acción para jóvenes basado en un mito de la cultura Chibcha tomando como base la Deidad Chibcha Bachué [17]

La mayoría de ellos están enfocados al desarrollo de prototipos de juegos educativos.

2.2. ESTRUCTURA NARRATIVA

La forma en la que se estructura la narrativa en un juego de video, se ve beneficiada por las ventajas que éste medio digital brinda, principalmente la de adaptarse a las decisiones y acciones hechas por el jugador. Éstas pueden ser programadas para que el juego muestre escenarios o eventos consecuentes a las acciones o decisiones del usuario. La profundidad entonces en el que un juego puede cambiar dependiendo de lo que haga un jugador posee varios niveles de influencia y de relevancia en la experiencia. Un nivel de profundidad se caracteriza por la toma de una decisión, y como consecuencia se varíe la experiencia sutilmente (el tipo de arma con el que se decide eliminar a un enemigo) o profundamente (decidir si el personaje principal vive o muere).

En la narrativa se habla de linealidad cuando sin importar las decisiones del jugador los eventos e historia dentro de éste no son afectados, lo que genera una serie de ventajas y desventajas que deben ser apreciadas. Como ventajas de una estructura lineal tenemos una planeación más sencilla ya que no existen variables dependientes de las decisiones tomadas por el jugador, y una cantidad de trabajo reducido al momento del desarrollo del juego ya que se crea contenido ligado directamente a una sola línea narrativa. Como desventajas de un juego lineal, está el hecho de que la narrativa no se adaptará de forma dinámica a las decisiones del jugador, generando una reducción en la percepción del jugador como actor relevante en el argumento.[18]

Podemos encontrar varias clases de estructuras narrativas, Pearce identifica seis clases diferentes que se pueden encontrar en un juego como lo son:

- Experiencial: Es la narrativa emergente que se desarrolla fuera del conflicto inherente del juego, como una experiencia del jugador mismo.

- Adaptativa: Es la narrativa emergente de las acciones del jugador, todo lo que haga el jugador influirá en el transcurso del juego.
- Incremental: Está compuesta por capas de información, interpretación y un esquema contextual que gira alrededor del juego y mejora otros agentes narrativos.
- Descriptiva: Hace el recuento de la descripción de los eventos del juego en tres partes, y la cultura que emerge de esta.
- Macro-Historia: Es una narrativa específica “incrustada” que crea un contexto o un esquema del conflicto del juego.
- Sistema de la Historia: Es una historia basada en un sistema o kit de partes de una narrativa genérica que le permite al jugador crear su propio contenido de la narrativa; los sistemas de las historias pueden ser independientes o por una combinación con las Macro-Historias

La estructura que se implementó es una herramienta realizada por el guionista David Siegel [68] para asegurarse que en la historia el conflicto sea legítimo y sus resoluciones demuestren una congruencia. Esta estructura es un intento de plantear cómo las películas de acción y los videojuegos son construidos.
Siegel distingue nueve actos diferentes [69], los cuales pueden ser encontrados en diversas películas y videojuegos, estos actos serán descritos junto con un ejemplo para tener una mayor claridad de cada acto en esta estructura:

- Acto 0: Se crea un antecedente relevante al protagonista y/o a la historia, creando un tipo de conflicto sin resolución. Ejemplo: En la película Batman del director Christopher Nolan, los padres del héroe son asesinados por su enemigo el Joker. [19]
- Acto1: Se establece el contexto con una imagen, la cual impone el tono de comunicación de la producción. Ejemplo: En Kill Bill 2 del director Quentin

Tarantino, se marca el contexto en el momento en que su protagonista recibe un disparo, el cual marca el inicio de la historia. [20]

-Acto 2: Se crea un problema, o algo inesperado ocurre. Ejemplo: En el Aro del director Gore Verbinski, la película se desencadena luego de encontrar el cuerpo desfigurado de una mujer, mostrando que algo misterioso está pasando. [21]

- Acto 3: Descripción del héroe y de su oposición, usualmente relatando cómo son en un contexto normal, haciendo claras sus cualidades extraordinarias. Ejemplo: En Indiana Jones: En busca del Arca Perdida del director StevenSpielberg, al principio se muestra la vida cotidiana del protagonista como profesor de arqueología.[22]

- Acto 4: Se establece un compromiso a resolver el problema, generalmente haciendo explícita la relevancia que éste tiene con el personaje. Ejemplo: El personaje Richard Kimble quiere probar que él no mató su esposa en El Fugitivo del director AndrewDavis. [23]

- Acto 5: El héroe no posee lo necesario para resolver el problema, o no tiene claro cómo resolverlo. Ejemplo: En Indiana Jones: La última cruzada del director StevenSpielberg, el protagonista necesita un mapa para llegar a su objetivo, pero este se encuentra en un lugar no accequible. [24]

- Acto 6: Se revela la clave para resolver el problema, generalmente brindada por el antagonista. Ejemplo: En la película Fight Club del director DavidFincher, el antagonista Tyler Durden revela al protagonista que él encarna a otra persona sin ser consciente de ello. [25]

- Acto 7: Se estructura una nueva meta y un plan para lograrla. Ésta debe ser planteada en el transcurso de la acción y no puede ser planeada. Ejemplo: En Terminator 2 del director JamesCameron, Sarah Connor se entera de la existencia del creador de la inteligencia artificial Skynet y cambia su rumbo para acabar con este individuo dejando atrás su objetivo anterior. [26]

- Acto 8: El antagonista es derrotado y el problema es resuelto. Se equilibra el contexto y el héroe obtiene el crédito. Ejemplo: En el Señor de los Anillos: El

Retorno del Rey del director PeterJackson, Frodo destruye el anillo derrotando al villano y convirtiéndose en el héroe equilibrando el contexto. [27]

Esta estructura se puede observar en varias producciones cinematográficas, y gracias a la simplicidad de esta, se hace posible utilizarla en videojuegos.

Hay ciertas diferencias cuando se aplica a películas y cuando se aplica a juegos. En el Acto 0 y 1 se suelen mostrar las cinemáticas del juego y el héroe es introducido antes de que algo malo ocurra. Puede que sea un problema de esta estructura forzar al héroe a ir por el objetivo equivocado, ya que el jugador siempre quiere ir por el objetivo acertado.[1] Como en cualquier estructura narrativa, es cuestión de diseñar el ritmo en que se progresa el juego para obligar al jugador a tomar una dirección errada. Un buen ejemplo de un juego que fuerce al héroe ir por la meta equivocada es Deus Ex donde sus aliados y sus mentores demuestran ser el enemigo real. [28]

2.3. ARGUMENTACIÓN DEL GÉNERO: SUPERVIVENCIA EN UN CONTEXTO DE HORROR

Entendiendo la magnitud de una tarea como la creación de un prototipo de juego de video, las motivaciones para el grupo de trabajo deben estar claras, ya que éstas definen las cualidades y estilo del producto a crear [29]. Aparte de desarrollar un juego como proyecto de grado y que éste figure en la hoja de vida, existe la razón experiencial de haber creado un juego de video. De igual forma, se pensó cual sería el juego que se desearía jugar y porqué. Es por esto que se eligió

[1]CARLQUIST, Jonas. Playing the story - Computer Games as a Narrative Genre. En : Human It. Umeå Area. Vol. 6, no.13 (2002); pg 22-23.

el género de supervivencia en un contexto de horror, por ser aquel que entrega intensas emociones de juego, a la vez que su credibilidad e inmersión dependen de gran manera en su narrativa [30]. Es importante argumentar la polémica alrededor del género de acción y subgénero de horror, la cual critica fuertemente el explícito contenido de violencia que poseen los juegos que pertenecen a este género. Esta argumentación se hará por medio de las siguientes teorías:

- Teoría de la catarsis de Feshbach y Singer**:**
Los individuos pueden descargar impulsos de agresividad acumulados si se dejan absorber por eventos violentos. En este caso el juego de video servirá como intermediario, ya que se agrede un avatar virtual y no una persona real, permitiendo liberar sentimientos de hostilidad acumulados en el día de una forma inofensiva.[31]

-Teoría del aprendizaje por observación de Bandura y Walters:
El público puede modelar su conducta hacia descripciones agresivas hechas por los medios de comunicación, imitando a personajes que utilizan la agresividad para alcanzar poder, fama, gloria, etc. [32]

- Teoría de los efectos del estímulo de Berkowtz:
La exposición a los estímulos agresivos incrementa la susceptibilidad de una persona para la excitación fisiológica y emocional, lo que a su vez hará aumentar su conducta agresiva. Un estímulo agresivo no provocará siempre una reacción agresiva, ni es probable que provoque un mismo grado de agresividad en todo el público, dependerá de la frustración del individuo, de la justificación de la actitud violenta, etc. [33]

- Teoría del refuerzo de Klapper:
Las imágenes de violencia en televisión o descripciones agresivas de noticias en prensa, refuerzan pactos establecidos de conducta violenta que los receptores de los medios tengan consigo ante los medios de comunicación.[33]

Estas perspectivas proveen una apreciación de la manera como el jugador puede interpretar los contenidos expuestos en el juego, y cómo es importante entender el impacto que puede provocar un exceso en contenido violento.

El género de acción, específicamente el de supervivencia en un contexto de horror (Survival Horror) [34] es uno que rápidamente creció en popularidad gracias a juegos como Silent Hill [35] y Resident Evil [36]. Sus fortalezas están en crear un contexto de tensión a la vez que el jugador debe planear cuidadosamente el manejo de recursos (balas, llaves, fuentes de “vida”, etc..) a la vez que se mantiene constante la presencia de elementos que intimidan (contexto y enemigos), ya que están diseñados minuciosamente para que el jugador nunca se sienta seguro[37]. Es entonces en este género donde la importancia del argumento y la forma de presentárselo al jugador demuestran su importancia, ya que de estos depende la credibilidad del contexto que se está presentando y por ende, la inmersión del jugador dentro del ambiente de terror.

La dependencia del género de terror a la narrativa de su historia hacen de éste un género idóneo para el desarrollo de este trabajo.

2.4. MODELO DE DESARROLLO

La creación de un juego de video es una tarea que tiene una tendencia al fracaso y a hacer replanteamientos en el transcurso del mismo; está expuesta a una

cantidad de variables las cuales influyen en muchos niveles y cada una de estas lo compone de manera crítica.

En cuanto al proceso de desarrollo se deben tener en cuenta tres fases: pre-producción, producción y post-producción. En un esquema simple de desarrollo en la primera fase se gesta el concepto y se diseña, en la segunda se implementa dicho concepto y en la última fase se realizan las pruebas y se despliega el producto final. [38]

Al momento de plantear la metodología de desarrollo, nos topamos con distintos modelos ya planteados tradicionalmente en el desarrollo de software. Encontramos el modelo de cascada, el cual se enfoca en crear puntos de revisión del producto con entregas calendarizadas [39], un videojuego desarrollado bajo esta metodología es Brutal Legend [72].En el modelo incremental se desarrollan versiones parciales del software al cual se le agregan mejoras y detalles que lo acercan cada vez más al software final [40].El modelo evolutivo es una modificación del modelo incremental dónde se hacen iteraciones de forma secuencial (una tras otra) en vez de desarrollarlas en paralelo [41].El modelo en espiral trabaja en ciclos, en donde se analiza en cada uno los riesgos y beneficios de proceder al siguiente [42].

Finalmente tenemos SCRUM el cual se basa en entregas de resultados quincenales o mensuales, lo que demuestra un desarrollo que se adapta de forma dinámica a los problemas y requerimientos de creación de un videojuego, además vuelve al proyecto mucho más adaptable a las expectativas del cliente. Todo esto se manifiesta en productividad y calidad del producto, alineamiento entre el cliente y el equipo de desarrollo a la vez que éste último, se mantiene motivado [43]. Esta metodología cíclica, tiene como cabeza del proyecto a un Dueño del Producto (Product Owner), encargado de realizar una lista de funciones a realizar (Product

Backlog). Luego se realiza mensual o semanalmente una reunión rápida de planeación (Sprint Planning Meeting) con el equipo de trabajo (Scrum Team) y el Director, en este encuentro se examinan las tareas realizadas y faltantes, y se proponen metas a cumplir para la próxima reunión. Posteriormente el equipo de trabajo tendrá una reunión diaria (Daily Scrum Meeting) dónde siempre se responderán tres preguntas: Qué hiciste ayer?, Qué vas a hacer hoy?, Qué ayuda necesitas?, para luego tener una retroalimentación del Director y observadores externos en un retrospectiva rápida (Sprint Retrospective), se realizan modificaciones y se entrega un prototipo. [44]

Esta metodología se evalúa por medio de teorías establecidas, una de ellas es la creada por el grupo Construx, reconocidos como evaluadores de metodologías de desarrollo de software. Estos plantean un método en el que se juzgan cinco parámetros o condiciones que la metodología debe cumplir para así determinar su eficacia, como lo son la accesibilidad del usuario o un representante al producto, la cantidad de talento técnico en el grupo de trabajo, la estabilidad para los requerimientos del sistema, la distribución geográfica del equipo de trabajo y la infraestructura de soporte.[45]

Left 4 Dead 2 [73] es uno de los videojuegos que han sido desarrollados con la metodología SCRUM.

A continuación se presenta un cuadro comparativo de los diferentes modelos de desarrollo de software mostrando así las ventajas y desventajas de cada uno:

MODELO	VENTAJAS	DESVENTAJAS
CASCADA	- La planificación es sencilla - La calidad del producto resultante es alta Permite trabajar con personal poco cualificado	- Modelo lineal - Se necesita tener todos los requisitos al principio - No hay posibilidad de corregir errores a tiempo - Aumento en los costos de desarrollo
ESPIRAL	- No necesita una definición completa de los requisitos para empezar a funcionar - Es más fácil validar los requisitos - El riesgo en general es menor - El riesgo de generar retrasos es menor	- Es difícil evaluar los riesgos - Necesita de la participación continua por parte del cliente - Cuando se subcontrata hay que producir previamente una especificación completa de lo que se necesita y esto lleva tiempo
INCREMENTAL	- Con un paradigma incremental se reduce el tiempo de desarrollo inicial - También provee un impacto ventajoso frente al cliente, que es la entrega temprana de partes operativas del software - Permite entregar al cliente un producto más rápido en comparación del modelo de cascada	- Requiere de mucha planeación, tanto administrativa como técnica - Requiere de metas claras para conocer el estado del proyecto
SCRUM	-La creación iterativa permite evidenciar el progreso del producto en diferentes etapas de desarrollo. -La producción es adaptativa a los percances que puedan suceder, gracias a la implementación de controles diarios y de entrega de iteración. -Es ideal para proyectos pequeños ya que su fortaleza reside en la capacidad de trabajo de cada pequeño equipo de trabajo.	-Exige un gran conocimiento técnico por parte de los desarrolladores, ya que cada equipo debe ser auto suficiente y auto regulado, lo que obliga a que cada uno de estos posea competencias en todas las áreas de desarrollo. -No se fundamenta en la documentación creada en etapas iniciales como en otras metodologías de desarrollo. -No genera todas las evidencias de documentación de otras metodologías

[70][71]

Tabla 1. Comparación modelos de desarrollo de software

3. METODOLOGÍA DE DESARROLLO

En el presente capítulo se muestra la estructura metodológica en la creación de un prototipo de Videojuego y los diversos elementos que lo componen.

3.1 SELECCIÓN TIPO DE JUEGO

Una vez se entra en el contexto de los videojuegos, se pueden apreciar distintas temáticas y géneros a los cuales se pueden recurrir. En una amplia lista podemos encontrar 7 tipos fundamentales de juegos: Multijugador Masivo de Rol en Línea (MMORPG), Rol, Simulación, de disparo con perspectiva en primera persona (FPS), Acción/Aventura, Estrategia y Acertijo [46]. Entre estos se implementó el de supervivencia en un contexto de horror con una perspectiva en primera persona, con el fin de restringir el proyecto en limitantes técnicas de modelado y animación de personajes, dejando como resultado procesos de diseño exclusivamente dedicados a enemigos y a la elaboración de la perspectiva en primera persona de la mano y arma del jugador.

3.2 SELECCIÓN DE METODOLOGÍA

Con el videojuego como proyecto y su género identificado, se adaptó el modelo de desarrollo SCRUM, el cual es reconocido por su eficiencia de trabajo en equipo, cualidades únicas en ambientes de desarrollo complejos y adaptabilidad a errores y requisitos cambiantes o no definidos. Desafortunadamente las evidencias detrás de los procesos de desarrollo implementadas en juegos de video ya realizados no es clara ni fácil de acceder debido a que hay pocas ocasiones en la que un desarrollador comparte la forma en la que resolvió sus problemas de producción, o

sencillamente no adapta una metodología ya existente si no que gira alrededor de una creada internamente por el equipo o compañía [74]. La gran fortaleza tras esta metodología de desarrollo es el enfoque ágil con el que busca resolver las tareas y acontecimientos en el transcurso de desarrollo del proyecto. A diferencia del método de cascada, el cual plantea la resolución completa de cada uno de los elementos del proyecto para visualizar un producto final una vez finalizado del ciclo de producción, SCRUM pone énfasis en la creación preliminar de un producto o iteración donde se busca evidenciar sus falencias para luego la asignación de trabajo dependiendo de las cualidades esenciales que se buscan en el producto. *(véase Figura 1)*

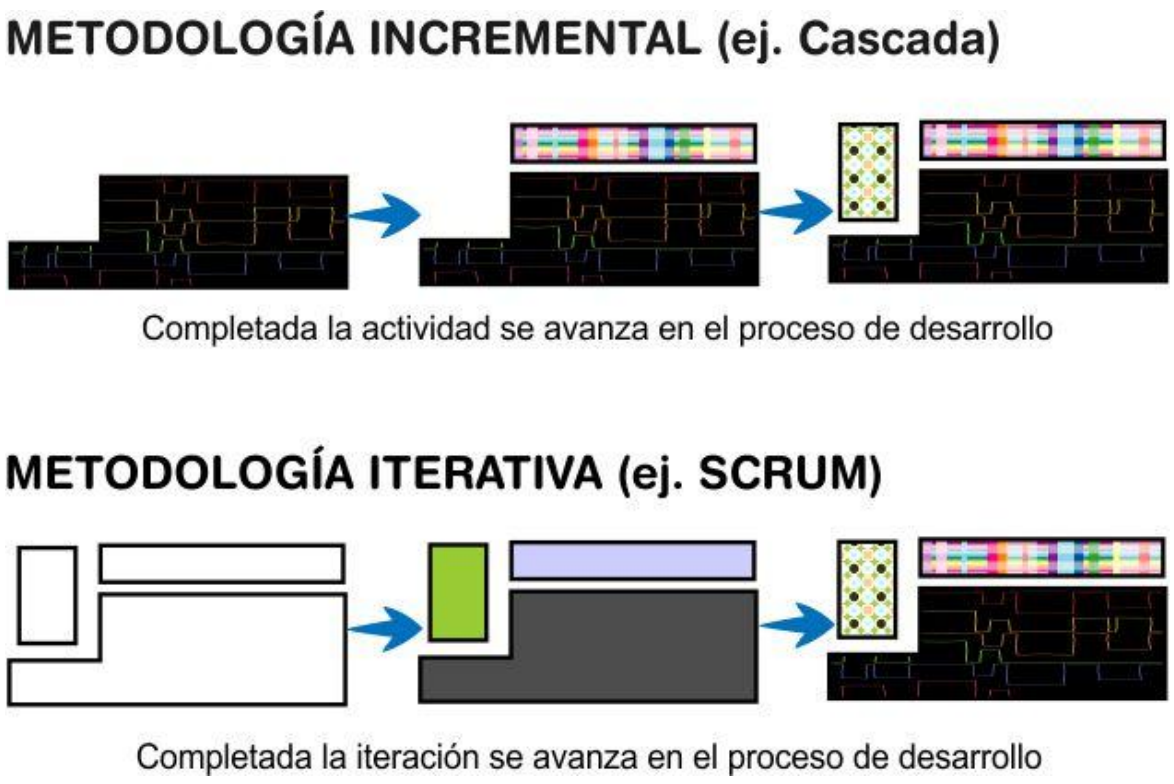


Figura 1. Metodología Incremental Vs Metodología Iterativa

Dado el contexto y documentación existente, las ventajas y desventajas planteadas por cada modelo de desarrollo evidenciadas en el marco teórico, SCRUM fue aquella que ofreció las mejores herramientas de trabajo para un grupo pequeño y completamente nuevo en el ámbito de desarrollo de videojuegos, al permitir adaptar la creación del videojuego a la capacidad de trabajo y aprendizaje de cada integrante, así como a la limitante de tiempo existente.

3.3 ARGUMENTO

3.3.1 Justificación

Para el desarrollo del prototipo de videojuego, se decidió crear una historia que ocurriera en Bogotá Colombia, con el fin de mostrar el trabajo de estudiantes de esta ciudad y país. De igual forma se optó por desarrollarse en el barrio La Candelaria, al ser el uno de los lugares más distintivos en cuanto a tradición y estética visual. Una vez planteado el ambiente y el género de la historia, se elige como creador de atmósfera de horror a muertos vivientes o “Zombies”, los cuales son creados por una maldición que recae en aquellos que pretenden abusar nocivamente de los recursos naturales del país.

Se decidió desarrollar el argumento en el idioma inglés, siendo éste requisito fundamental para la distribución de juegos independientes por agentes virtuales reconocidos como Steam [47] u Onlive [48]. De igual forma esta característica permite al prototipo de videojuego ser presentado a nivel internacional como una muestra de cualidades a nivel personal de los desarrolladores e institucional de la Universidad Militar Nueva Granada.

Se escogió el muerto viviente o “Zombie” fundamentalmente por factores de tiempo, ya que éste es un personaje con comportamientos elementales de diseñar y programar. Adicionalmente, posee la facultad de generar en cualquier persona repulsión y aversión al intentar parecerse a un ser humano sin serlo, activando instintos básicos de preservación rechazando cualquier intento de representación de su especie [49], encajando en el perfil planteado por las restricciones de diseño.

El argumento tiene como personaje principal a Claudia Colomar, la cual es creada con el objetivo de expresar diversas clases de emociones ligadas a su género, por las cuales se busca manipular los sentimientos del jugador, tales como la fragilidad implícita que tiene una mujer que se enfrenta a un ambiente y condiciones desconocidas, la adaptabilidad y sentido de supervivencia a situaciones extremas, la capacidad de manipulación para lograr sus metas, entre otras. Siendo esto indispensable para la narrativa del juego, permitiendo que el jugador sea envuelto por la historia.

Históricamente en el género de los juegos de disparar en perspectiva de primera persona, se ha mantenido un balance entre la capacidad de "aguante" o vida de un jugador, y el daño que éste recibe por parte de los enemigos que se encuentran. Éste “aguante” se justifica a nivel narrativo con botiquines que curan al jugador inmediatamente como lo visto en el juego Half Life [5] ó con un escudo gradual que se regenera si el jugador no es golpeado en un lapso de tiempo específico como en el juego Mass Effect 2 [9]. Para el caso de este juego, se decidió otorgar una cantidad fija de “vida” por una cantidad de tiempo al jugador, y para argumentar este hecho, se le otorgaron poderes especiales a la protagonista, todo con el fin de hacer congruente una estructura de juego con la técnica narrativa.

3.3.2 Contexto de la Historia

El argumento utilizado requiere de un fundamento contextual completo con los que sus elementos narrativos tendrán la solidez y credibilidad necesarias para lograr generar la percepción deseada en el jugador. Esto se hace con el objetivo de crear la impresión de que una historia como la que el jugador está experimentando sea posible, enriqueciendo la experiencia narrativa del videojuego.

Con el fin de estructurar el universo del videojuego, se redactó un documento que relata los elementos principales que componen la ficción del contexto, como guía para los desarrolladores más no como elemento fundamental de la narrativa en la experiencia de juego (*véase anexo A*). Para la aprobación del argumento, se recurrió al reconocido escritor y docente de la UMNG Fernando Soto Aparicio, el cual concluyó que el argumento tenía lo necesario para ser implementado en un videojuego.

3.3.3 Relación guión - Estructura de nueve actos de David Siegel

Como elemento fundamental en el desarrollo del proyecto, se redactó un guión (*véase anexo B*) y se estructuró su relación directa con la estructura narrativa de 9 pasos de David Siegel.

- Acto 0: *Se crea un antecedente relevante al protagonista y/o a la historia, creando un tipo de conflicto sin resolución.*
La descripción completa del personaje se realizará al final del videojuego ya que es la esencia del guión. El inicio del juego no tendrá cinemáticas, se pretende dejar al jugador lo más perdido posible en el lugar que empieza.

- Acto1: *Se establece el contexto con una imagen, la cual impone el tono de comunicación de la producción.*

El contexto será dado en la habitación donde ocurren los hechos iniciales, introduciendo al jugador en un ambiente desconocido donde hay personajes en este caso Zombies que tendrá que matar para sobrevivir, luego una serie de objetos lo ayudará a descubrir su objetivo.

-Acto 2: *Se crea un problema, o algo inesperado ocurre.*

El problema principal será dado cuando el jugador empieza a ver e interactuar con los Zombies y posteriormente se encontrará con el radio que le brindará cierta información para desarrollar la trama del juego.

- Acto 3: *Descripción del héroe y de su oposición, usualmente relatando cómo son en un contexto normal, haciendo claras sus cualidades extraordinarias.*

A medida que se avanza en el juego, se irán mostrando indicios de cómo resolver el problema principal por medio de diálogos y secuencias.

- Acto 4: *Se establece un compromiso a resolver el problema, generalmente haciendo explícita la relevancia que éste tiene con el personaje.*

Se asocia con la necesidad de salir con vida del lugar y averiguar porque el jugador, específicamente, se encontró bajo esa situación.

- Acto 5: *El héroe no posee lo necesario para resolver el problema, o no tiene claro cómo resolverlo.*

Para escapar del lugar, el jugador debe tomar posesión de un objeto el cual no posee.

- Acto 6: *Se revela la clave para resolver el problema, generalmente brindada por el antagonista.*

Esta clave será brindada no por un antagonista sino por el bueno que quedará eventualmente encerrado debido a las acciones del jugador.

- Acto 7: *Se estructura una nueva meta y un plan para lograrla. Ésta debe ser planteada en el transcurso de la acción y no puede ser planeada.*

Se motiva al jugador a escapar por sus propios medios sin la asistencia del personaje que lo ha ayudado en el transcurso del juego.

- Acto 8: *El antagonista es derrotado y el problema es resuelto. Se equilibra el contexto y el héroe obtiene el crédito.*

El antagonista es derrotado, pero se revela que los roles han sido inversos desde un comienzo, con el personaje principal como villana de la historia. Se equilibra el contexto con ella escapando triunfante del lugar.

3.4 SELECCIÓN DEL MOTOR GRÁFICO

En la actualidad existe una cantidad considerable de motores gráficos a los que se recurre para la creación de juegos de video. Es por esto que se valoraron las características de aquellos más utilizados en el mercado.(*Véase Tabla 2*)

Motor / Características	Unity 3	UDK	Source Engine	Id Tech 4	CryEngine 3
Calidad Gráfica Alta	X	X	X	X	X
Calidad de Audio Alta	X	X	X	X	X
Gratuito	/	X			
Requerimientos Técnicos Bajos	*	*	X		
Documentación	X	/	/	/	/
Soporte	X	X	/	/	/
Curva de Aprendizaje en Programación	X	X	X		

* Requerimientos Técnicos se reducen si se implementa en Dispositivos Móviles / Con limitantes

[50][51][52][53][54]

Tabla 2. Características de motores gráficos.

Una vez identificadas las cualidades de los diferentes motores gráficos, se opta por utilizar el Unreal Development Kit (UDK)[51], tomando como prioridad de sus características la adquisición y uso gratuito sin limitantes de ningún tipo.

3.5 DISEÑO DEL NIVEL

3.5.1 Diagramación

Con la estructura narrativa establecida, se diseñó un esquema visual a nivel de boceto, dónde se diagramó el nivel dentro del motor gráfico, con la ruta que debe tomar el jugador y marcas visuales para localización de elementos clave, teniendo como fundamento mantener el tono y ambiente de un contexto de horror desarrollado en el barrio La Candelaria.

En el primer piso del primer edificio, se creó un esquema fundamentado rigurosamente en el guión, los Zombies representados por puntos verdes, el inicio del personaje por un punto rosado, y la ruta por una secuencia de flechas.(*véase Figura 2*)

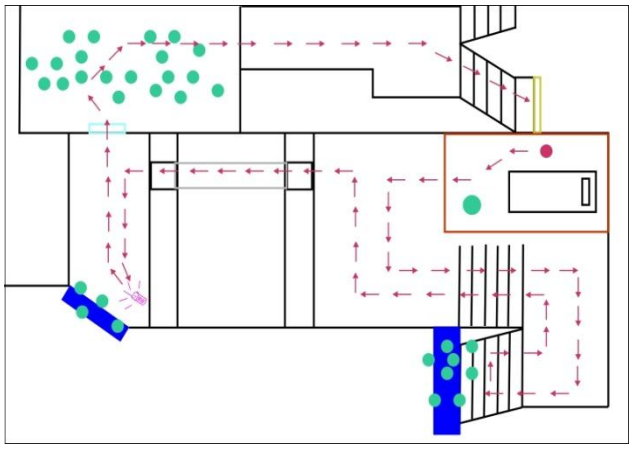


Figura 2. Primer Piso Edificio 1

El segundo piso del primer edificio, posee una descripción general de las diferentes habitaciones encontradas, tales como cafetería, laboratorio, etc., con el fin de tener una guía al momento de poblar el espacio y generar un tono congruente en éstas.(*véase Figura 3*)

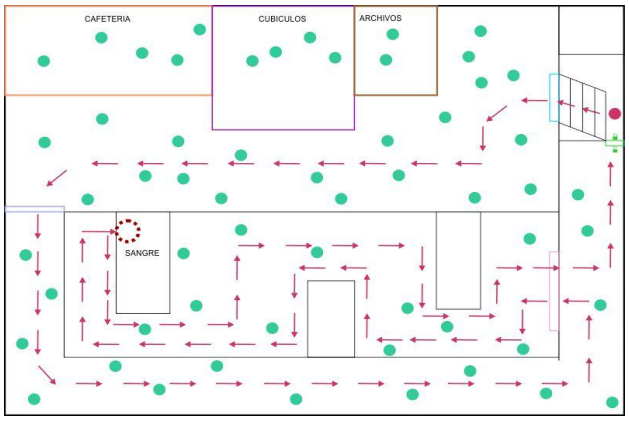


Figura 3. Segundo Piso Edificio 1

El exterior fue diagramado según lo redactado en el guión, posicionando obstáculos y bloqueos, elementos clave como carros de policía y puntos donde hay fuego.(*véase Figura 4*)

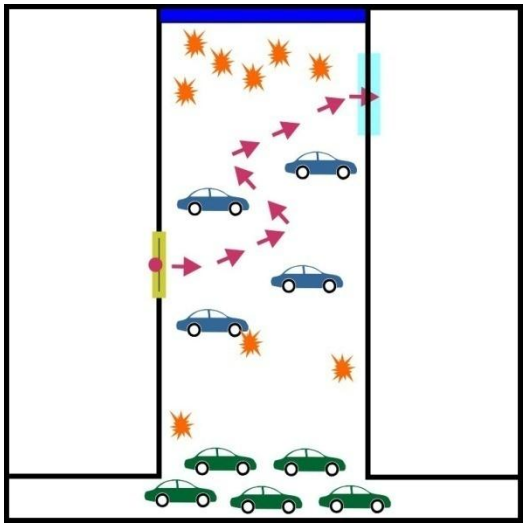


Figura 4. Exterior

El diagrama del edificio final muestra el recorrido del jugador a su objetivo final, con ayudas visuales para elementos relevantes en la narrativa como botones y la entrada al acueducto.(*véase Figura 5*)

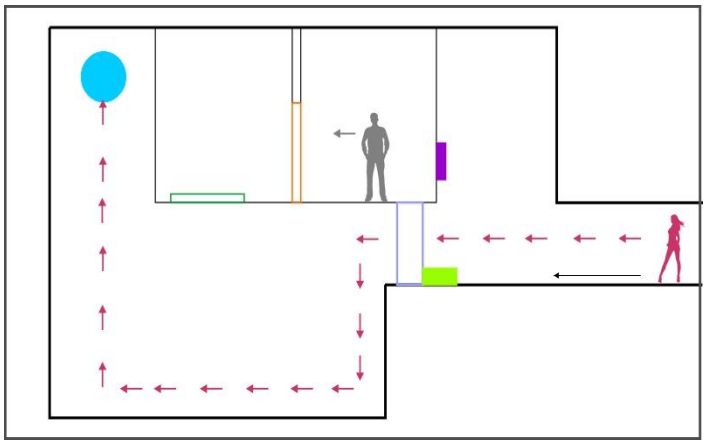


Figura 5. Edificio 2

3.5.2 Referencias Visuales

Con el fin de obtener elementos para crear el tono artístico adecuado dentro del juego, se tomaron diferentes fotos del barrio La Candelaria desde perspectivas y locaciones diferentes. (*véase Figura 6*)

Figura 6. Barrio La Candelaria

Para más imágenes de referencia (*véase Anexo C*)

3.6 CREACIÓN DE MATERIALES

Una vez diseñado el nivel en BSP's[2], se les asignó un material que adopta las propiedades de color, reflexión de luz, emisión de colores como fuente lumínica y mapa de normales de una superficie. Para esto, el UDK utiliza el Editor de Materiales (Material Editor), conectando los canales RGBA de un archivo de imagen TGA a los diferentes atributos del material.

En los BSP´s, el material puede ser escalado y rotado en cualquier dirección, lo que brinda un gran control en la forma como se pueden graficar las texturas sobre una superficie con tal que éstas estén teseladas.

En los modelos (Meshes) la creación de materiales es igual, y la forma como éste se muestra es dependiente del mapeado de UV´s del modelo.

Con los materiales sobre los BSP´s se agregó un nuevo tipo de material llamado decal, el cual tiene transparencia permitiendo agregar irregularidad y detalle.

3.7 DISEÑO DE PERSONAJES

Como únicos personajes que fueron mostrados de forma física en el juego se tienen los Zombies y la protagonista. Éstos fueron diseñados acorde a las restricciones técnicas del motor gráfico y de animación, ya que como elemento esencial del proyecto se decidió realizar animaciones por captura de movimiento.

[2]Partición binaria del espacio (BSP) es un método para subdividir recursivamente un espacio en elementos convexos empleando hiperplanos

3.7.1 Zombie

Para el diseño del Zombie se realizó el Arte Conceptual luego de una previa investigación en juegos, películas, etc. Inicialmente se diseñó un Zombie con 4 brazos y una estructura desproporcionada, el cual fue descartado por restricciones de animación ya que las extremidades adicionales deberían ser animadas independientemente de los resultados de captura. (*Véase Figura 7*).

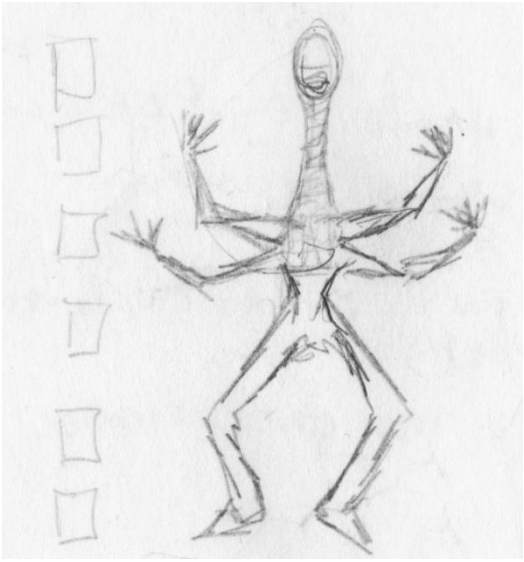


Figura 7.Boceto 1 Zombie

De igual forma se diseñó un Zombie con una posición de cabeza inusual, igualmente descartado al no adaptarse de forma consecuente a la narrativa. (*Véase Figura 8*)

Figura 8. Boceto 2 Zombie

Finalmente se optó por implementar una estructura de humanoide básica, y utilizar un enfoque en detalles faciales a nivel morfológico depictado por escultura indígena, siendo este enfoque el más congruente con el argumento. (*Véase Figura 9*).

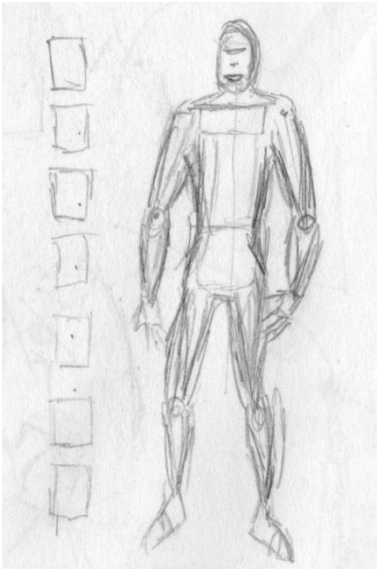


Figura 9. Boceto 3 Zombie

Las fuentes de morfología en expresiones artísticas indígenas se consultaron en varios libros de cultura precolombina ubicados en la Biblioteca Luis Ángel Arango. Se obtuvo como resultado claves morfológicas para ser representadas en el Zombie.(*Véase Figura 10*)

Figura 10. Diseño Precolombino [55]

El modelo final posee características de humanoide, desproporcionado en sus medidas corporales y faciales. Igualmente se constituyeron sus elementos faciales similar a lo encontrado a las imágenes fuente consultadas, con un conteo de 3200 polígonos. (*Véase Figura 11*)

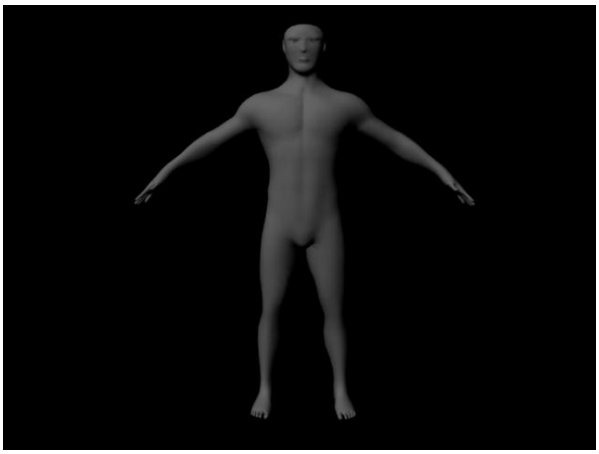


Figura 11. Modelo 3D Zombie

3.7.2 Protagonista

Al ser un juego con perspectiva de primera persona, el diseño del personaje principal se limitó a crear una mano femenina y un candelabro tal como fue diseñado en la estructura narrativa. (*Véase Figura 12*)

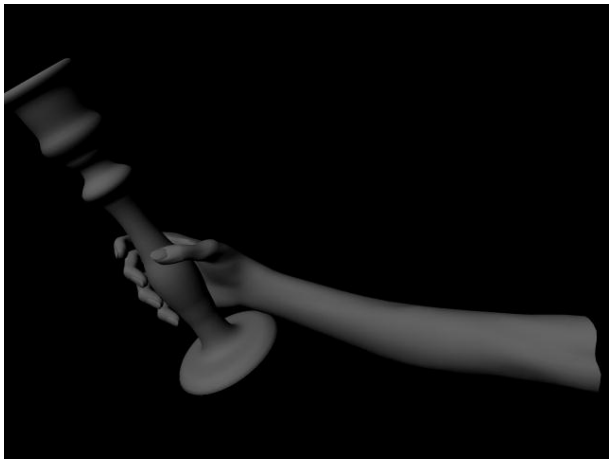


Figura 12. Modelo 3D Mano y Arma

3.8 MODELADO

Se realizaron modelos 3D con un conteo máximo de 2000 caras por cada uno, para luego crear sus respectivos mapas de UV´s y ser exportados en formato .PSA o .FBX.Inicialmente, se obtuvieron buenos resultados exportando algunos de los modelos en formato .PSA pero surgieron problemas de compatibilidad entre los plugins y las programas de modelado, por lo tanto se escogió como formato final el .FBX, brindando igualmente buenos resultados sin problemas de compatibilidad.
En esta etapa se utilizaron el software Autodesk Maya[56] y el plugin ActorX[57].

3.9 MAPEADO Y TEXTURIZADO DE PERSONAJES

En el mapeo de UV´s se protegieron las zonas del modelo dónde no se deseaban ver cortes de texturas, luego se crearon las texturas de cada modelo y se editaron algunas de uso gratuito, provenientes de los sitios web CG Textures [58], Image After [59] y Mayang's Free Textures [60].

Teniendo listas las texturas, se le aplicaron a cada modelo teniendo en cuenta una estética visual agradable y que fuera acorde a los requisitos de la narrativa; una vez texturizado el modelo, se exportó el mapa Difuso en formato .TGA a un tamaño de 1024 x 1024 pixeles debido a que el UDK sólo admite texturas en potencia de dos en sus ejes X y Y y tiene una restricción de formatos. Basándose en el mapa Difuso se generaron los mapas de Normales, Especular y Poder Difuso para lograr un mejor acabado del modelo sin necesidad de tener un polígono de muchas caras. Finalmente se toma el canal Alfa del Difuso y se agrega el mapa de Poder Difuso, se realiza el mismo procedimiento en el de Normales pero esta vez teniendo el especular como Alfa.

Las herramientas empleadas en esta fase fueron Adobe Photoshop CS5 [61], y Autodesk Mudbox[62]

3.10 DISEÑO DE INTERFAZ

Para la creación del menú principal, se recurrió al soporte de Scaleform[63] por parte del UDK. Éste permite la creación de contenidos desde archivos .SWF implementándolos en tiempo real en la creación de interfaces, menús y videos entre otros. De esta forma se diseñó un menú con pantalla inicial que permite iniciar el juego, cambiar la resolución de pantalla y salir de la aplicación.

Todos los comandos son relacionados al ambiente Kismet[3] en UDK por medio de eventos de Actionscript 2[64].

3.11 ANIMACIÓN

3.11.1 Convencional

La animación convencional por fotogramas fue utilizada para la animación del brazo y el arma de la protagonista, con sus respectivos ciclos de ataque y estado de reposo. Las restricciones por parte del formato .FBX para separar el modelo con esqueleto ligado a otro modelo que no hace parte de dicha jerarquía, obligó a utilizar los formatos .PSK para el modelo base y .PSA para sus animaciones, brindados por el plugin ActorX de Maya 2011.

3.11.2 Captura de Movimiento

Las animaciones por captura de movimiento fueron implementadas para la caracterización de los Zombies dentro del juego, donde fueron creados 17 ciclos como caminados y ataques.

Como primera medida, se modeló el personaje con la cantidad de polígonos adecuada para ser implementado en un juego de video y se mapearon sus UV's. A continuación se creó una estructura de esqueleto (Joints), dónde cada uno de sus elementos tiene el mismo nombre que el encontrado en una estructura base hecha para la captura de movimiento. Esta estructura es ligada al modelo, con pesos en sus nodos asignados para que la

[3]Sistema de visualización de código, que permite a artistas y diseñadores crear contenidos programados de forma rápida e intuitiva.

deformación sea acorde a la esperada en un humanoide de las características diseñadas.

Con el modelo terminado, se procede a grabar las animaciones por captura de movimiento, asistido por el programa Autodesk MotionBuilder[65]. Para esto se utilizaron las instalaciones brindadas por el Tecnoparque del Sena, con un actor que utiliza un traje que graba los movimientos que éste realiza. Luego de que éste interpreta los movimientos deseados, se obtiene información de animación que puede ser ligada a un modelo con el mismo esqueleto (rig) determinado (*Véase Figura 13*)

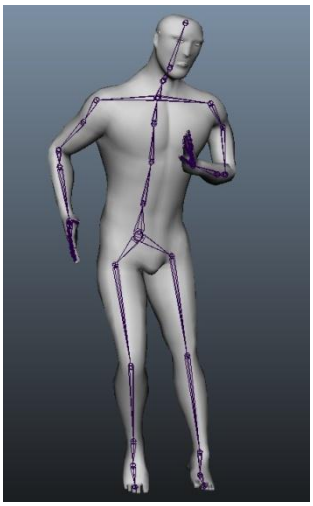


Figura 13. Modelo del Zombie, con estructura de huesos

Luego se edita y se convierte en un ciclo, el cual se puede animar de forma perpetua sin crear la impresión de cortes. Ésta animación es exportada a UDK, el cual reconoce el modelo base como un Skeletal Mesh[4], y las animaciones como AnimNodes[5]. Éstos AnimNodes son relacionados directamente con un árbol de jerarquías de animación de personaje, donde por defecto están creadas las transiciones entre animaciones y la fusión

[4]Modelo con huesos al cual se le puede ligar información de animación, contrario al Static Mesh.

[5]Ciclo de animación

entre varias de ellas, habilitando la referencia por código de ésta a cualquier actor, permitiendo que éste tome la apariencia y movimientos creados. (*véase Anexo D*)

3.12 PROGRAMACIÓN

3.12.1 Diseño Orientado a Objetos

El proceso de desarrollo bajo una metodología iterativa llevó a la creación de artefactos congruentes a las necesidades de producto en su respectiva etapa, ligadas a las necesidades del cliente y a la evolución del producto. En las *Figuras 14 y15* se presentan diagramas que unifican las necesidades del usuario final con el procedimiento de la metodología de desarrollo, con el fin de obtener un producto satisfactorio siguiendo los inconvenientes especificados en el desarrollo.

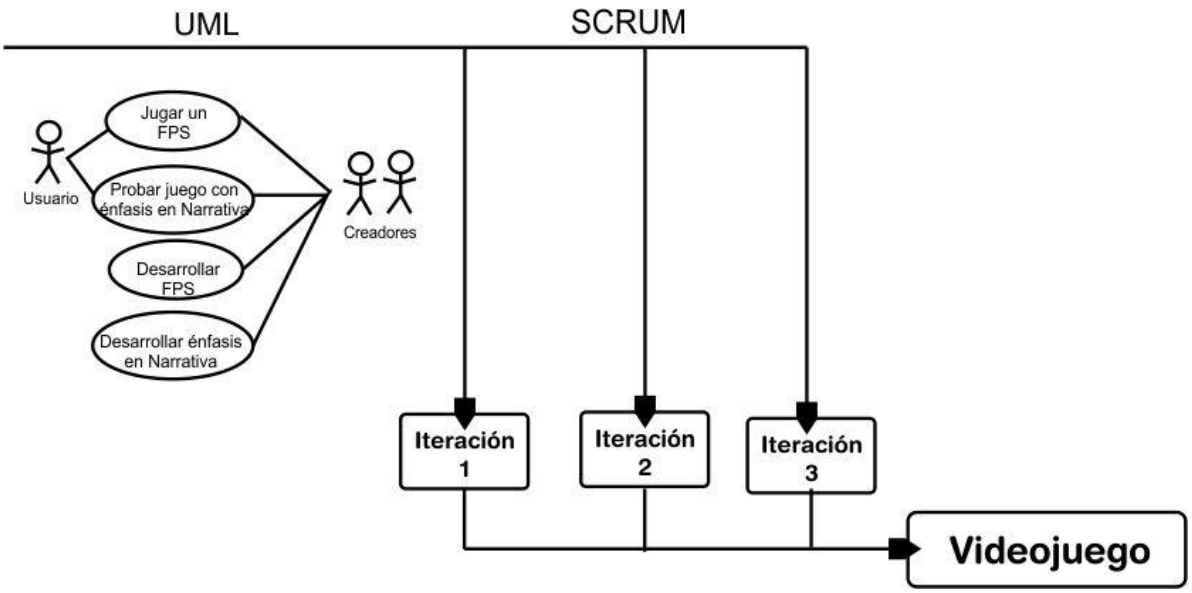


Figura 14. Integración casos de uso - videojuego con SCRUM

Figura 15. Contenido de una iteración

Los artefactos[6] desarrollados ante cada iteración variaron dependiendo de las necesidades del producto y las exigidas por la metodología de desarrollo. Como ejemplo se puede evidenciar un artefacto dentro de cronograma completamente planeado, como fue el de la modificación de la interfaz con el fin de cambiar la forma como se podía ver la vida del jugador, hasta artefactos inesperados que la metodología de desarrollo permitió adherir y adicionar, como fue el añadir una función al código principal de disparo del arma para retrasar el impacto y coordinarlo con la animación de ataque.

La relación entre las estructuras de clases y la herramienta gráfica se puede observar en las *Figuras 16 y 17*, las cuales contienen respectivamente un diseño preliminar de funcionalidad y un diagrama detallado de jerarquía de clases.

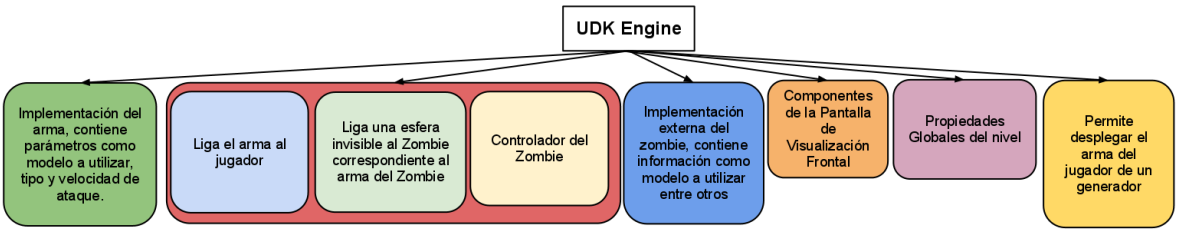


Figura 16. DOO con funciones

[6] Término empleado para referirse a un resultado del trabajo, ya sea un texto, diagrama, código, etc.

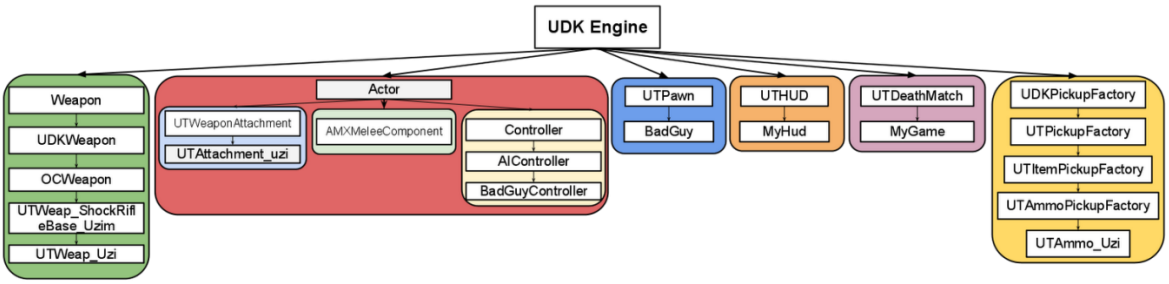


Figura 17. DOO con jerarquía de clases

3.12.2 Inteligencia Artificial

Para la Inteligencia Artificial del Zombie se realizaron tres clases por Zombie con el fin de controlar su comportamiento. Las *Figuras 18, 19 y 20* muestran el diagrama de clases correspondiente, para su posterior implementación dentro del UDK.

Figura 18. Diagrama para clase Badguy.uc

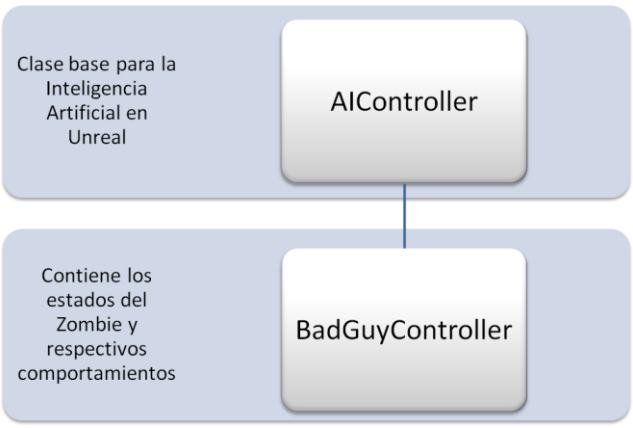


Figura 19. Diagrama para clase BadguyController.uc

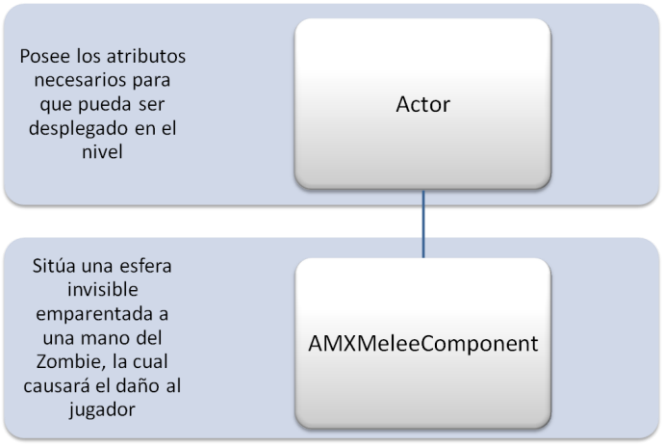


Figura 20. Diagrama para clase AMXMeleeComponent.uc

Al tener estas tres clases, se pasa al el nivel realizado en el UDK Editor y se agrega el actor de la clase BadGuy, luego se traza el trayecto que debe recorrer por medio de nodos de camino (Path Nodes) y finalmente se le asignan al Zombie.

Para referencia total del código fuente (*véase Anexos E, F y G*)

3.12.3 Arma

En cuanto al arma del jugador se crearon seis clases y se implementó el código “OcWeapon.uc”[66] de los foros del UDK. Las *Figuras 21, 22 y 23* muestran el diagrama de clases correspondiente, para su posterior implementación dentro del UDK.

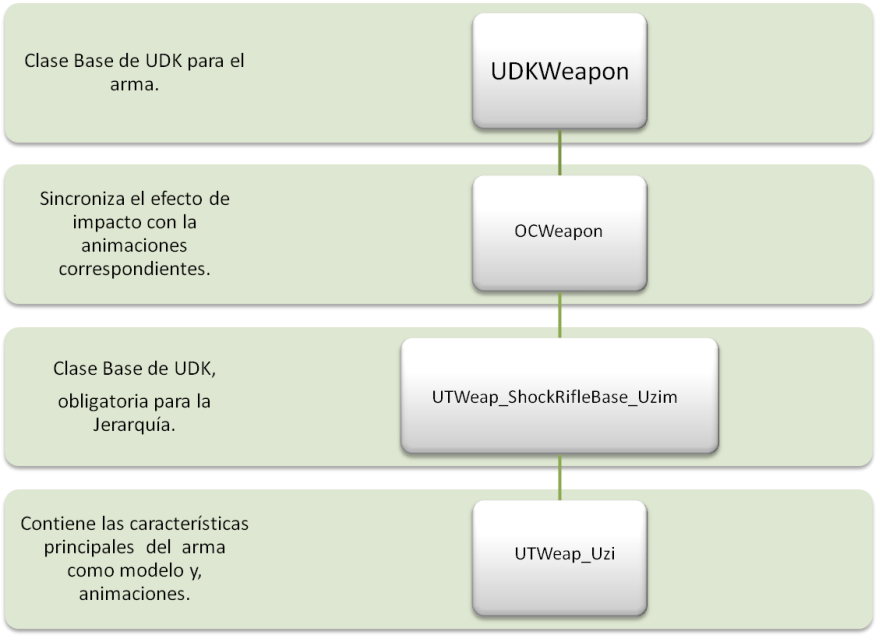


Figura 21. Diagrama de clases 1 para el arma

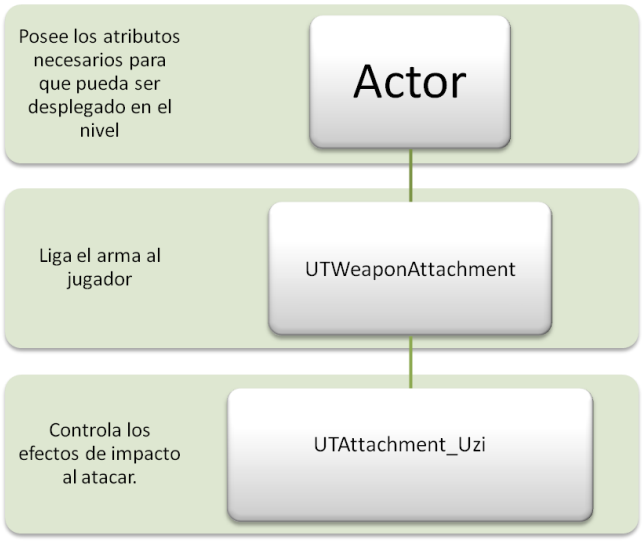


Figura 22. Diagrama de clases 2 para el arma

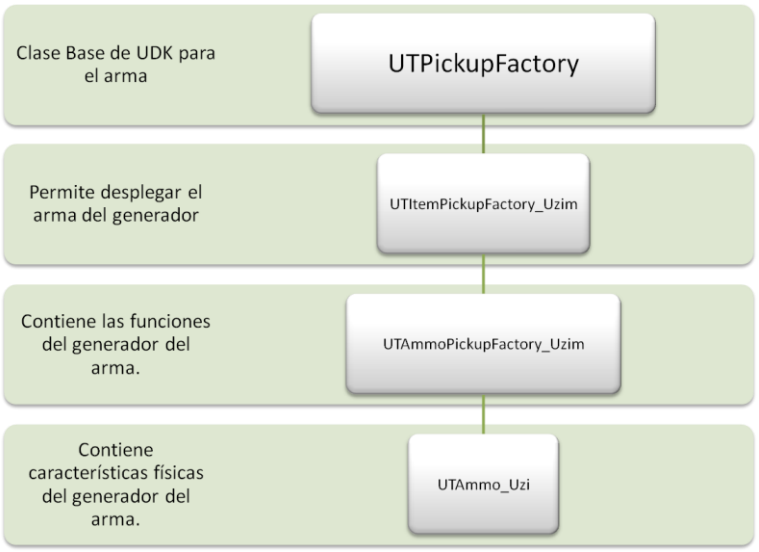


Figura 23. Diagrama de clases 3 para el arma

Para referencia total del código fuente (*véase Anexos H, I, J, K, L, M, N*)

3.12.4 Pantalla de Visualización Frontal

Por defecto, el UDK trae una interfaz diseñada para la creación de mapas de competencia multijugador (Deathmatch). Debido a que el juego a desarrollar posee un enfoque completamente diferente, se decidió crear una interfaz sencilla donde de forma intuitiva se conozca el único dato relevante: la vida del jugador.

Con este fin, se implementó un ojo completamente verde, el cual se torna rojo a medida que la vida del jugador decrece. Las Figuras *24 y 25* muestran el diagrama de clases correspondiente, para su posterior implementación dentro del UDK.

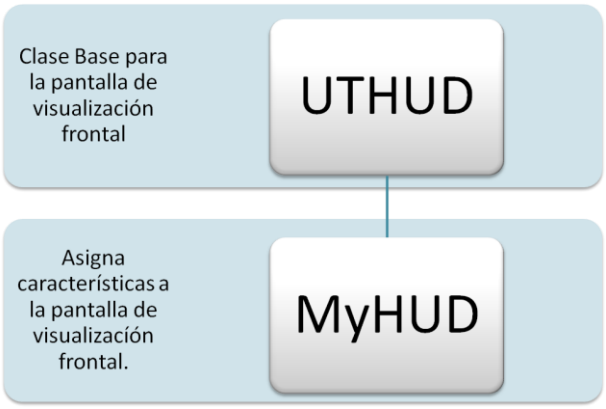


Figura 24. Diagrama para la clase MyHUD.uc

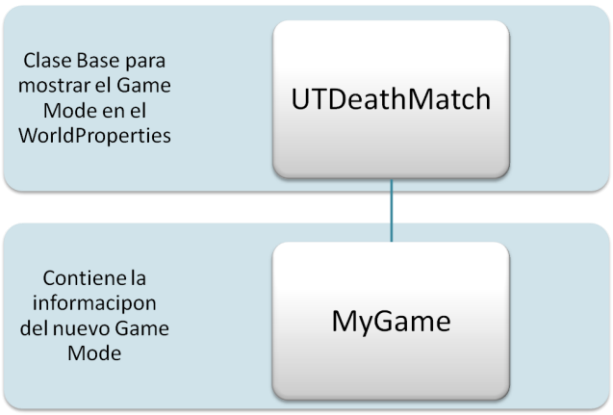


Figura 25. Diagrama para la clase MyGame.uc

Para referencia total del código fuente (*véase Anexos O y P*)

3.13 VOCES Y AUDIO

El diseño sonoro del juego requirió la creación de sonidos de todo tipo. Para las voces se debieron grabar los diálogos con una actriz de voz que interpretase el rol de la protagonista y su contraparte masculina, siendo estos codificados en formato .WAV. Cada segmento fue luego cortado, editado y convertido en un Sound Cue[7] dentro del UDK, donde se le asignan atributos como reverberación, repetición y volumen. Dentro del UDK se crean volúmenes de reverberación donde se establecen prioridades de sonido dependiendo de la zona, creando ambientes sonoros como cuevas o exteriores.

3.14 VIDEO

El video se creó en formato .AVI, el cual muestra una secuencia de imágenes y el diálogo de un relator con la secuencia final del juego. UDK recibe únicamente

[7] Formato de audio dentro del UDK para su secuencialización en eventos a través de Kismet.

videos codificados bajo el formato .BIK, el cual puede ser llamado y secuenciado dentro de Kismet o el editor de materiales para ser mostrado como secuencia ó material sobre un BSP. El audio debe ser extraído y separado como .WAV. En este caso, el video fue asociado con un BSP, para luego programar un eventoque permite mostrar la imagen y tocar el sonido a la vez.

3.15 EVENTOS

Para la creación de eventos se utilizó el ambiente de desarrollo Kismet. En este se pueden visualizar actores, elementos y comportamientos para programar secuencias y lograr los eventos deseados. Por ejemplo, para lograr que en el comienzo del nivel el protagonista diga “hola”, se debe crear un agente que espera a que el jugador interactúe con él llamado Trigger. Una vez el jugador entra en contacto, se ejecuta una secuencia de órdenes predeterminadas. En este caso, se esperan 4 segundos, se toca el sonido y se le asigna como objetivo al jugador. (*Véase figura 26*).

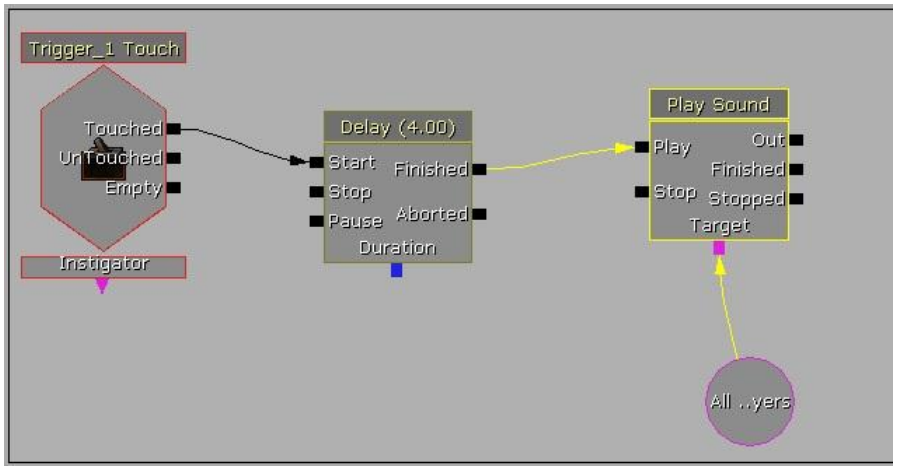


Figura 26. Secuencia en Kismet toque de trigger, espera de 4 segundos y toque de audio.

De esta forma se programaron triggers que activan secuencias como diálogos y animaciones que abren puertas o prenden luces mediante Matinée[8].

3.16 INTEGRACIÓN SCRUM-VIDEOJUEGO

El proceso de desarrollo tuvo como fundamento los diversos ciclos y niveles de implementación que la metodología exige. Como desarrolladores se dio lugar a una reunión diaria donde se realizaron controles sobre las actividades realizadas el día anterior y las planeadas a realizar el mismo. Así mismo se planearon reuniones cada quince días con el tutor de proyecto, el cual hacía seguimiento al proceso de desarrollo con sus debidas correcciones y sugerencias. En el nivel más amplio, se produjeron iteraciones del producto en su estado actual de desarrollo, programadas para hacerse cada dos meses. En la *Figura 27* se muestra gráficamente el desarrollo de procesos como complemento.

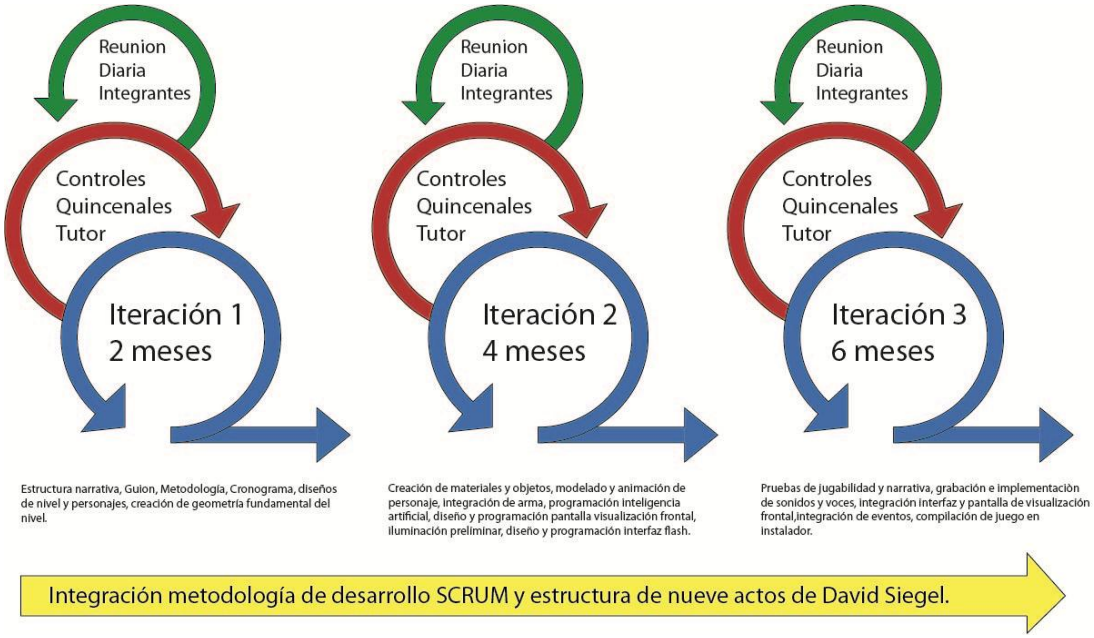


Figura 27. Diagrama de Procesos

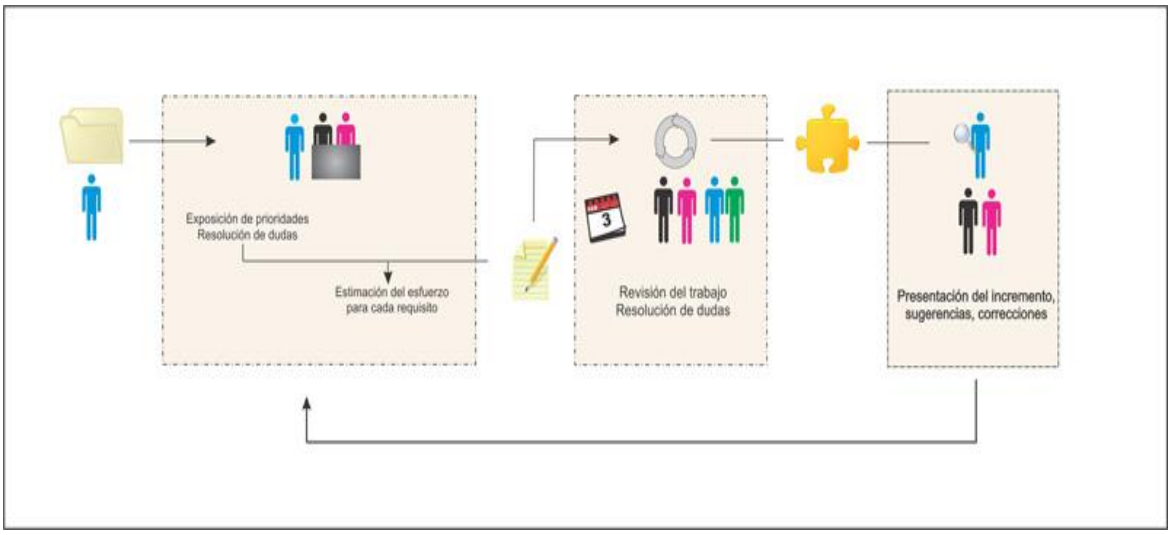
[8] Herramienta con controles de edición que permite la construcción de cinemáticos y secuencias de video.

En las reuniones diarias se hizo retroalimentación del trabajo realizado, la resolución de problemas y reajuste del cronograma dependiendo de los requerimientos inmediatos del proceso de desarrollo.

Las reuniones quincenales tenían como objetivo hacer el control por parte del Director de proyecto, para así observar y controlar el progreso, resolución de dudas e inquietudes y delimitar los requerimientos de la próxima iteración del producto.

La iteración 1 corresponde a la fase de diseño y planeación, bocetos y esquemas preliminares del mapa del videojuego. En la iteración 2 se ejecutó el proceso de producción del videojuego con tareas como modelado, edición de imagen, animación, programación y pruebas internas; en la última iteración se agregaron elementos de producción como voces o eventos, y se realizaron pruebas generales al juego y a la narrativa para realizar una evaluación y así extraer información relevante para el proceso final de análisis.

La metodología SCRUM se relacionó con el prototipo de videojuego mediante el siguiente proceso de desarrollo:



[67]

Dónde:

	Director del Trabajo de Grado: Ing. Christian Quintero
	Equipo SCRUM Gonzalo Dániel Claudia Buitrago
	Interesados: Asesoran y Observan
	Pila Del Producto: Requisitos del Proyecto
	Planificación y Creación de Lista con las funciones a realizar semanalmente
	Lista con funciones a realizar diariamente por el Equipo SCRUM
	Reunión Diaria entre el Equipo SCRUM, se actualiza la lista de funciones diarias.
	Incremento
	Revisión y Retroalimentación

Tabla 3. Estructura flujo de trabajo SCRUM

Al iniciar el Proyecto de Grado, se estructuró una lista con las funciones a realizar. Semanalmente se programó una reunión entre el equipo SCRUM y el Director del Proyecto dónde se examinaron las actividades realizadas y las faltantes, haciendo una retroalimentación y acordando objetivos para el próximo encuentro.

A la vez, se acordó tener diariamente una reunión ya sea presencial o virtual entre el equipo SCRUM, contestando las preguntas base que la metodología propone: Qué hiciste ayer?, Qué vas a hacer hoy?, Qué ayuda necesitas?, para así tener claras las funciones que se iban completando en cada fase del proyecto.

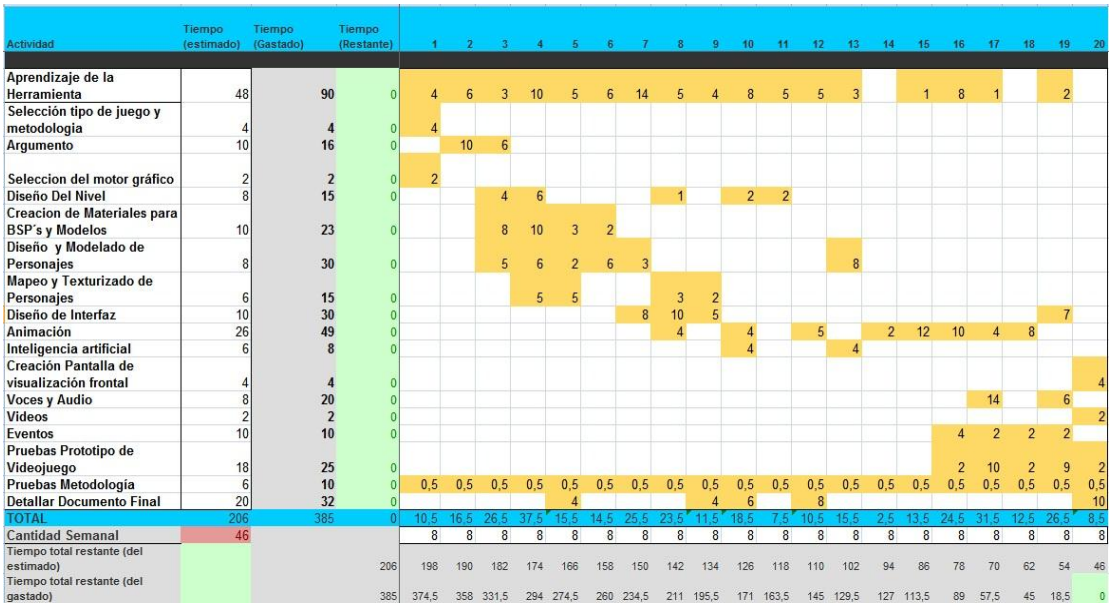
La presentación de cada prototipo se realizó cada dos meses, debido a diversos factores que retrasaron la ejecución de los mismos. El Director e Interesados brindaron las respectivas críticas y sugerencias, las cuales permitieron tener un producto final de gran calidad.

3.16.1 Seguimiento de Procesos

Basándonos en el esquema de desarrollo de iteraciones *(Véase Figura 24),* se hizo un control sobre las diferentes tareas realizadas en el transcurso del desarrollo del proyecto, con el fin de establecer las cargas de trabajo de cada integrante y sus responsabilidades.

Siguiendo el cronograma planteado originalmente para la implementación de la metodología de desarrollo, se creó un cronograma donde se estableció un proceso de desarrollo que tomaría 5 meses. Desafortunadamente, debido a la intensidad del proceso de aprendizaje en todas las áreas de desarrollo, se debió recurrir a una extensión de cronograma a 8 meses.

La distribución de tareas con sus respectivas cargas semanales fueron diagramadas utilizando Simple Scrum Agile Project Managementbrindado por Google Docs. Los cronogramas de cantidad de trabajo semanal por persona fueron hechos a medida que se avanzó en el desarrollo. (*Véase Figuras 28, 29, 30 y 31*)



Actividad	Tiempo (estimado)	Tiempo (Gastado)	Tiempo (Restante)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Aprendizaje de la Herramienta	48	90	0	4	6	3	10	5	6	14	5	4	8	5	5	3		1	8	1		2	
Selección tipo de juego y metodología	4	4	0	4																			
Argumento	10	16	0		10	6																	
Seleccion del motor gráfico	2	2	0	2																			
Diseño Del Nivel	8	15	0			4	6				1		2	2									
Creacion de Materiales para BSP´s y Modelos	10	23	0			8	10	3	2														
Diseño y Modelado de Personajes	8	30	0			5	6	2	6	3						8							
Mapeo y Texturizado de Personajes	6	15	0				5	5			3	2											
Diseño de Interfaz	10	30	0							8	10	5										7	
Animación	26	49	0								4		4		5		2	12	10	4	8		
Inteligencia artificial	6	8	0										4			4							
Creación Pantalla de visualización frontal	4	4	0																				4
Voces y Audio	8	20	0																	14		6	
Videos	2	2	0																				2
Eventos	10	10	0																4	2	2	2	
Pruebas Prototipo de Videojuego	18	25	0																2	10	2	9	2
Pruebas Metodología	6	10	0	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5
Detallar Documento Final	20	32	0					4				4	6		8								10
TOTAL	206	385	0	10,5	16,5	26,5	37,5	15,5	14,5	25,5	23,5	11,5	18,5	7,5	10,5	15,5	2,5	13,5	24,5	31,5	12,5	26,5	8,5
Cantidad Semanal	46			8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
Tiempo total restante (del estimado)			206	198	190	182	174	166	158	150	142	134	126	118	110	102	94	86	78	70	62	54	46
Tiempo total restante (del gastado)			385	374,5	358	331,5	294	274,5	260	234,5	211	195,5	171	163,5	145	129,5	127	113,5	89	57,5	45	18,5	0

Figura 28. Carga Horaria Semanal, Gonzalo Dániel, Enero-Mayo

Actividad	Tiempo (estimado)	Tiempo (Gastado)	Tiempo (Restante)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Aprendizaje de la Herramienta	48	90	0	4	6	3	10	5	6	14	5	4	8	5	5	3		1	8	1		2	
Selección tipo de juego y metodologia	4	4	0	4																			
Argumento	10	16	0		10	6																	
Seleccion del motor gráfico	2	2	0	2																			
Diseño Del Nivel	8	15	0			4	6				1		2	2									
Creacion de Materiales para BSP´s y Modelos	10	23	0			8	10	3	2														
Diseño y Modelado de Personajes	8	30	0			5	6	2	6	3						8							
Mapeo y Texturizado de Personajes	6	15	0				5	5			3	2											
Diseño de Interfaz	10	30	0							8	10	5										7	
Animación	26	49	0								4		4		5		2	12	10	4	8		
Inteligencia artificial	6	8	0										4			4							
Creación Pantalla de visualización frontal	4	4	0																				4
Voces y Audio	8	20	0																	14		6	
Videos	2	2	0																				2
Eventos	10	10	0																4	2	2	2	
Pruebas Prototipo de Videojuego	18	25	0																2	10	2	9	2
Pruebas Metodología	6	10	0	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5
Detallar Documento Final	20	32	0				4				4	6			8								10
TOTAL	206	385	0	10,5	16,5	26,5	37,5	15,5	14,5	25,5	23,5	11,5	18,5	7,5	10,5	15,5	2,5	13,5	24,5	31,5	12,5	26,5	8,5
Cantidad Semanal	46			8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
Tiempo total restante (del estimado)			206	198	190	182	174	166	158	150	142	134	126	118	110	102	94	86	78	70	62	54	46
Tiempo total restante (del gastado)			385	374,5	358	331,5	294	274,5	260	234,5	211	195,5	171	163,5	145	129,5	127	113,5	89	57,5	45	18,5	0	

Figura 29. Carga Horaria Semanal, Claudia Buitrago, Enero-Mayo

Actividad	Tiempo (estimado)	Tiempo (Gastado)	Tiempo (Restante)	1	2	3	4	5	6	7	8	9	10	11	12
Aprendizaje de la Herramienta	48	75	0	4	6	3	10	5	6	14	5	4	8	5	5
Creacion de Materiales para BSP´s y Modelos	10	23	0			8	10	3	2						
Mapeo y Texturizado de Personajes	6	15	0				5	5			3	2			
Diseño de Interfaz	10	23	0							8	10	5			
Animación	26	32	0	14	10	8									
Inteligencia artificial	6	12	0	7	2	3									
Creación Pantalla de visualización frontal	4	4	0	2	2										
Voces y Audio	8	31	0				10	12	9						
Videos	2	2	0									2			
Eventos	10	21	0	6		3				8			4		
Pruebas Prototipo de Videojuego	18	71	0	7	12	10	6	7	6	3	14	2	1	2	1
Pruebas Metodología	6	6	0	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Detallar Documento Final	20	22	0					4				4	6		8
TOTAL	174	337	0	40.5	32.5	35.5	41.5	32.5	23.5	33.5	32.5	15.5	13.5	7.5	6.5
Cantidad Semanal	78			8	8	8	8	8	8	8	8	8	8	8	8
Tiempo total restante (del estimado)			174	166	158	150	142	134	126	118	110	102	94	86	78
Tiempo total restante (del gastado)			337	296.5	264	228.5	187	150.5	127	93.5	61	41.5	22	14.5	0

Figura 30. Carga Horaria Semanal, Gonzalo Dániel, Junio-Agosto

Actividad	Tiempo (estimado)	Tiempo (Gastado)	Tiempo (Restante)	1	2	3	4	5	6	7	8	9	10	11	12
Aprendizaje de la Herramienta	40	55	0	9	8	7	5	5	6	6		6	1	2	
Creacion de Materiales para BSP´s y Modelos	35	53	0	8	10	8	9	4	4	5	3	2			
Mapeo y Texturizado de Personajes	20	35	0	7	5	7	5	5	6						
Diseño de Interfaz	2	2	0	1		1									
Animación	12	17	0	4	3	6	4								
Inteligencia artificial	20	41	0	7	9	3	8		4	6	4				
Creación Pantalla de visualización frontal	8	17	0	4	4	5	1	1	2						
Voces y Audio	6	16	0		6	4	2	4							
Videos	2	2	0									2			
Eventos	2	3	0			3									
Pruebas Prototipo de Videojuego	15	71	0	7	12	10	6	7	6	3	14	2	1	2	1
Pruebas Metodología	6	6	0	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5
Detallar Documento Final	20	22	0					4				4	6		8
TOTAL	188	340	0	47,5	57,5	54,5	40,5	30,5	28,5	20,5	21,5	16,5	8,5	4,5	9,5
Cantidad Semanal	134			5	4	5	4	5	4	5	4	5	4	4	5
Tiempo total restante (del estimado)			188	183	179	174	170	165	161	156	152	147	143	139	134
Tiempo total restante (del gastado)			340	292,5	235	180,5	140	109,5	81	60,5	39	22,5	14	9,5	0

Figura 31. Carga Horaria Semanal, Claudia Buitrago, Junio-Agosto

Como fue evidenciado en los cronogramas, la combinación de seguir un plan académico en la Universidad y el proceso de aprendizaje en la utilización de la herramienta generaron un gasto inesperado de tiempo al cual se debió adaptar el proyecto, requiriendo una extensión de 3 meses (*Véase Tabla 4*).

Iteración	Tiempo Estimado	Tiempo Gastado
1	7	10
2	6	9
3	7	13
Total	20	32

Tabla 4. Tiempos de desarrollo estimados y gastados en semanas para cada iteración.

3.17 INTEGRACIÓN SCRUM-NARRATIVA DE DAVID SIEGEL

Como objetivo fundamental de desarrollo en el proyecto, se tiene el desarrollo de una estructura narrativa congruente con una estructura de desarrollo. Para esto fue esencial determinar las necesidades establecidas por la narrativa en cada iteración del producto. Como primera medida se creó el guión restringiéndolo a las limitantes impuestas por la estructura narrativa. Por medio de SCRUM se delimitó un tiempo para este proceso, el cual fue originalmente de 10 horas por persona, en el cual se redactó un documento donde se describen los sucesos y diálogos que ocurrirían en el transcurso del mismo.

Como etapa siguiente se diagramó el nivel de forma que se pudiesen ver los diferentes elementos y sucesos de forma espacial, primero de forma borrador y luego en mapas de vectores, proceso al que le fueron asignadas 8 horas de trabajo por persona.

Una vez creados el guión y el diseño del nivel, se entró de lleno en la creación de contenidos dentro del UDK, avanzando en diferentes iteraciones dependientes de los documentos ya creados. Debido al tiempo requerido para importar nuevos contenidos al motor, se estructuraron dos líneas de trabajo simultáneo, donde se introducía el nuevo contenido por un integrante a medida que el otro seguía creando las piezas necesarias para poblar el nivel.

Los 9 pasos en la estructura narrativa fueron segmentados en los procesos de desarrollo dependiendo de sus requerimientos técnicos, obteniendo como resultado una implementación casi exclusiva al tercer ciclo de desarrollo, debido al alto grado de dependencia del guión sobre el diálogo. La implementación del Acto 1 (tono visual del contexto) fue la única completada en la segunda iteración del juego, al ser completamente dependiente de la comunicación visual, área que fue desarrollada en el segundo ciclo de creación. Todas las otras etapas en la estructura narrativa fueron obtenidas en la tercera iteración, ya que cualquier expresividad por parte de la protagonista fue hecha por medio de la actriz de voz,

cualidad integrada en la tercera fase de producción como se puede observar en la *Tabla 5*.

Acto	Descripción	Iteración de Desarrollo
0	Antecedentes Personaje	2
1	Tono del contexto	1
2	Revelación problema	3
3	Descripción héroe	3
4	Compromiso Resolución Problema	3
5	No capacidad de resolución	3
6	Clave de resolución	3
7	Nueva Meta	3
8	Antagonista derrotado	3

Tabla 5. Integración narrativa de David Siegel con SCRUM

4. RESULTADOS INTERNOS

4.1 METODOLOGÍA DE DESARROLLO

Como evaluación a la metodología de desarrollo utilizada para la creación del juego de video se recurrió a la teoría implementada por la compañía consultora de software Construx. Ésta fundamenta las pruebas a una metodología de desarrollo en cinco parámetros fundamentales:

-La accesibilidad del usuario o un representante al producto:

En este caso, se evaluó la accesibilidad al producto del tutor de grado, siendo ésta no la más óptima, ya que existieron grandes limitantes de nivel técnico que evitaron una evaluación directa del mismo, limitándolo a presentaciones y evaluaciones remotas.

-La cantidad de talento técnico en el grupo de trabajo:

La extensión de evaluación de este parámetro se debe hacer a nivel técnico y artístico. Aunque al comenzar el proyecto existían inmensas limitantes técnicas, la asistencia brindada por el tutor de proyecto, asesores del SENA y fundamentalmente documentación encontrada en internet, brindaron los conocimientos necesarios para ejecutar el mismo. Desafortunadamente en el proceso de desarrollo se debió invertir una gran cantidad de recursos a procesos de aprendizaje, reduciendo la cantidad de recursos efectivos invertidos dentro del juego.

-La estabilidad para los requerimientos del sistema:

Gracias a que se utilizó una herramienta de desarrollo como el UDK, se logró obtener un producto estable en todo momento, con una falencia en

tamaños de archivo pero no de estabilidad en momento de ejecución del juego.

-La distribución geográfica del equipo de trabajo:

SCRUM como metodología de desarrollo para un grupo de trabajo localizado en la misma ciudad es de gran utilidad debido a lo práctico que es programar y realizar reuniones. De igual forma las nuevas tecnologías de información permiten acortar distancias y realizar trabajos prácticamente ignorando este parámetro.

-La infraestructura de soporte:

Como infraestructura de soporte se tiene la posibilidad de regresión entre versiones y corrección de problemas volviendo a iteraciones anteriores. SCRUM demostró ser eficaz desarrollando un juego de video de las características diseñadas, gracias a la flexibilidad del UDK en permitir el retroceso y corrección entre iteraciones del videojuego.

4.2 ESTRUCTURA NARRATIVA

La implementación de una estructura narrativa por medio de un videojuego logra resultados inmediatos gracias a la fidelidad sensorial que brindan las herramientas actuales, especialmente el UDK. Desafortunadamente la fidelidad de la experiencia está directamente relacionada con la calidad de los recursos que fueron desarrollados e implementados. Un proyecto de grado desarrollado por dos personas en un lapso de 6 meses no es suficiente para lograr la fidelidad y calidad vistas en el mercado actual. Aún así, los resultados fueron satisfactorios ya que se logró implementar la estructura narrativa en su totalidad, con todos los elementos planteados en el guión y documentos de diseño.

4.3 METÁFORA VISUAL

En el área gráfica se buscaba recrear ambientes que hicieran alusión al contexto de horror y posición geográfica en Bogotá Colombia, lo cual fue logrado satisfactoriamente gracias al poder que brinda una herramienta como el UDK. Las animaciones desarrolladas por captura de movimiento lograron un resultado improbable de lograr por medios convencionales, y los sonidos brindaron una contextualización satisfactoria. Los resultados graduales a medida que se avanzaba en el proceso de producción fueron evidentes como se muestra a continuación:

Iteración 1: Creación de nivel con BSP´s, etapa a la cual fue asignada un mes comenzando desde la primera semana de Febrero.(*Véase Figura 32*)

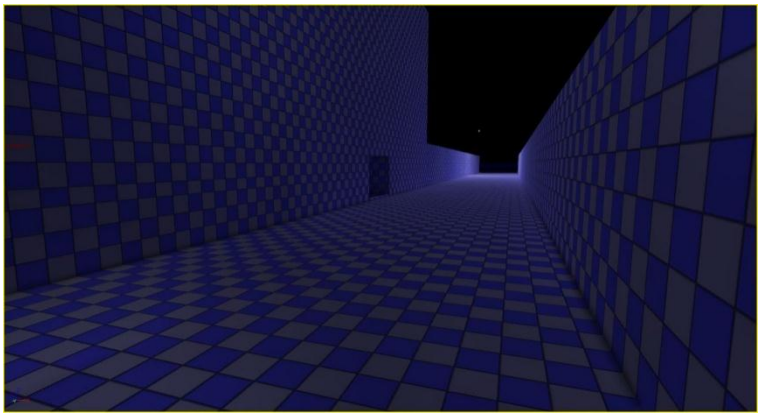


Figura 32. Iteración 1

Iteración 2: Asignación de materiales a los BSP´s manteniendo el tono de comunicación exigido por la narrativa. Para este proceso fue asignado un mes de desarrollo, tiempo que demostró no ser suficiente para los requerimientos del nivel, extendiéndose por medio mes más. (*Véase Figura 33*)

Figura 33. Iteración 2

Iteración 3: Poblado de nivel con modelos congruentes a la definición de tono y sensación de cada ambiente declarados en los documentos de diseño. Para esta etapa ya se contaba con un gran número de modelos creados gracias a el desarrollo en paralelo del proyecto, donde se modelaban elementos a medida que se poblaba el nivel. (*Véase Figura 34*)

Figura 34. Iteración 3

Las restricciones introducidas desde la concepción del guión planteaban la creación de un ambiente congruente al estereotipo clásico post cataclismo de horror. Con este fin, se diseñó un ambiente tenue con una predominancia al uso constante de sombras y una paleta de colores tierra.

4.4 MECÁNICAS DE JUEGO

La implementación de mecánicas de juego donde se utiliza un candelabro como arma y Zombies que atacan al jugador fueron logradas plenamente, y la manera como se estructuró la narrativa y los eventos dentro del juego fueron logrados de igual manera.

5. PRUEBAS Y RESULTADOS EXTERNOS

5.1 PRUEBAS MEDIANTE SONDEO

Para la evaluación del juego, se juzgaron dos parámetros fundamentales: la estructura narrativa y las mecánicas de juego. Para esto se citó a diez personas con diferentes perfiles y experiencia previa con juegos de video teniendo claras las restricciones de edad y limitantes de acuerdo al género escogido, y se les pidió que intentaran terminar el juego, a los cuales se les citó el día lunes 21 de Noviembre en las instalaciones del Tecnoparque SENA nodo Bogotá (Calle 57 No 8-69) (*Véase Figuras 35, 36 y Anexo Q*) para realizar las respectivas pruebas y cuestionarios. Para ver el registro de jugadores (*Véase Anexo* R) Los equipos utilizados para la prueba contaban con las siguientes características: Procesador 3.0 ghz Intel Dual Core, 4 GB de memoria RAM, tarjeta de video Nvidia 8400gs 256 mb, y sistema operativo de 32 bits.

Vale anotar que la muestra no posee un tamaño estadísticamente relevante, debido a que el interés de la investigación es exclusivamente de percepción inmediata, proveniente de diversos perfiles de jugador. Al ser éste un prototipo de videojuego sus primeras iteraciones deben ser evaluadas por grupos pequeños de individuos, los cuales reportan las falencias más evidentes y relevantes del mismo, asemejándose al proceso de pruebas internas realizadas por cualquier desarrollador de juegos de video antes de entregar un juego completo. [76]

Figura 35. Pruebas del prototipo en el Tecnoparque del Sena

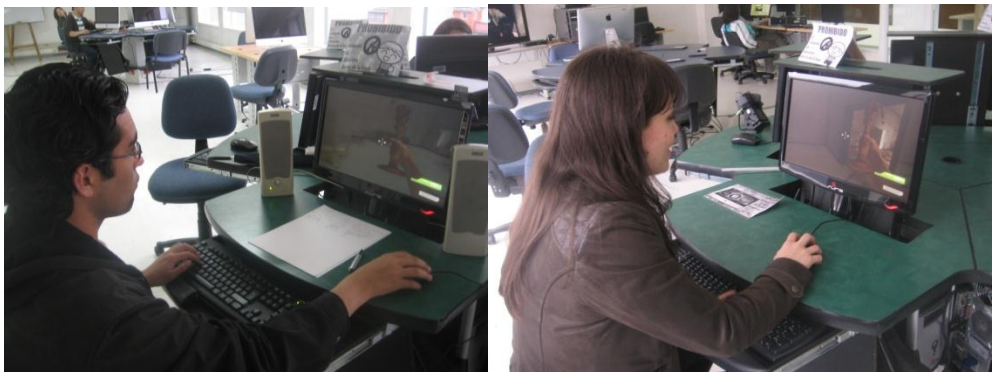


Figura 36. Jugadores a mitad de prueba

La realización del respectivo cuestionario tuvo como fundamento evaluar las cualidades narrativas y cualitativas del juego de video, al igual que obtener un resultado general de penetración narrativa en el jugador final y así apreciar de manera cuantitativa el impacto de la implementación de una estructura narrativa con prelación a los modelos de desarrollo y de mecánicas de juego.

A pesar de que no se logró encontrar una referencia puntual que haga alusión a la evaluación de la narrativa en un videojuego, se tomó como referencia el artículo "Virtual Experience Test: a Virtual Environment Evaluation Questionnaire"[75] publicado en la IEEE, ya que se acoplaba a nuestras necesidades para evaluación y análisis del prototipo.

5.2 ESTRUCTURA NARRATIVA

La evaluación de la estructura narrativa tuvo como principal enfoque evaluar la valoración emocional al momento de experimentar la historia del juego. Para esto se plantearon las siguientes preguntas:

NUM. PREGUNTA	CONTENIDO
1	Experimenté un alto grado de interacción con los agentes virtuales dentro del juego.
2	Cuando experimenté una reacción emocional sentí que ésta era apropiada dados los eventos que ocurrían en el juego en ese momento.
3	Encontré que la historia en el ambiente virtual hizo un buen trabajo para explicar las tareas que debía desarrollar.
4	Sentí una diversidad de emociones mientras jugaba.
5	Encontré que un alto nivel de interacción con los agentes virtuales era necesario para completar mis tareas dentro del juego.
6	Sentí que los agentes virtuales controlados por el computador (inteligencia artificial) fueron bien utilizados en el juego.
7	Sentí que el ambiente en el juego usó múltiples técnicas para provocar emociones.

Tabla 6. Cuestionario para evaluación de narrativa

5.3 MECÁNICAS DE JUEGO

Las mecánicas de juego incluyen el flujo del prototipo y la percepción de jugabilidad por parte de los encuestados. Para esto se plantearon las siguientes preguntas:

NUM. PREGUNTA	CONTENIDO
1	Encontré que el hardware relacionado con la capacidad gráfica del equipo fue de gran calidad
2	Encontré que el contenido visual del ambiente del juego era de gran calidad.
3	Tuve una reacción emocional al realizar tareas dentro del juego.
4	Encontré que el contenido en el ambiente del juego fue útil en informarme de mi tarea actual.
5	Pienso que el ambiente del juego me dejó claro lo que debía y no debía hacer.
6	Encontré que el hardware relacionado con el audio era de gran calidad.
7	Encontré que el contenido del audio del ambiente del juego era de gran calidad.
8	Pienso que las tareas que podía hacer en el juego eran interesantes.
9	Sentí que el ambiente me permitió completar mi tarea de diferentes maneras.
10	Sentí que podía reutilizar continuamente las técnicas aprendidas de tareas anteriores en tareas posteriores.

Tabla 7. Cuestionario para evaluación de mecánicas de juego

5.3.1 Resultados

Con base en las pruebas realizadas se hizo un análisis cualitativo de los aspectos relevantes a la narrativa y mecánicas de juego, con el fin de evidenciar las falencias, debilidades en estructura narrativa y modelos de desarrollo y conclusiones para futuros proyectos de la misma índole.

5.3.1.1 Narrativa y Mecánicas de Juego

La evaluación a la narrativa arrojó resultados satisfactorios a través de todas las preguntas planteadas (*Véase Figura37*), con un 80% de decisión con un puntaje de 4 o superior, en escala de 1 a 5. Cabe notar una inconformidad con los métodos instruccionales

dentro del juego, ya que al parecer los objetivos no eran del todo claros y la secuencialidad de tareas se rompía en algún punto del recorrido. Para observar todos los resultados de la encuesta de narrativa *(Véase Anexo S).*

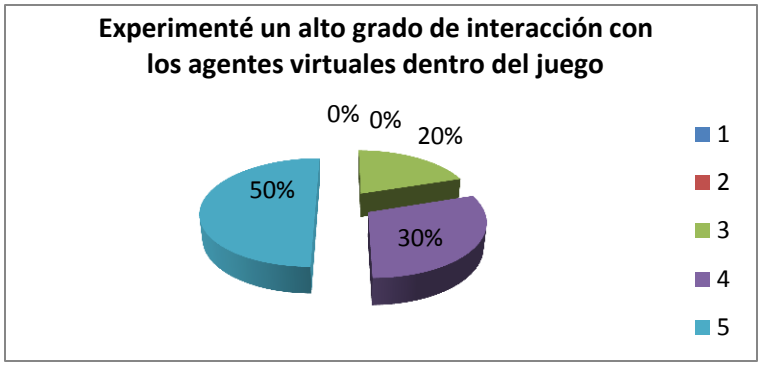


Figura 37. Resultado Pregunta 1 Narrativa

Los resultados en mecánicas de juego *(Véase Figura 38)* fueron congruentes con aquellos arrojados por la evaluación narrativa, dando indicios que se debió profundizar en la señalización y explicación de tareas dentro del juego. Los aspectos técnicos evaluados fueron igualmente satisfactorios, con una mención especial a la apreciación que se tuvo con la interacción con los agentes del juego, la calidad técnica y el interesen el argumento. Para observar todos los resultados de la encuesta de mecánicas de juego *(Véase Anexo T).*

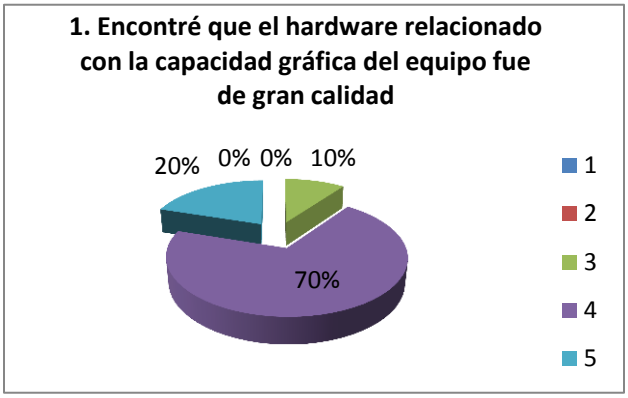


Figura 38. Resultado Pregunta 1 Mecánicas de Juego

5.3.1.2 Metodología de Desarrollo

La metodología de desarrollo SCRUM es óptima para la realización de un prototipo de videojuego, ya que se adapta a los percances que van surgiendo durante el desarrollo del mismo, sin atrasar significativamente las demás actividades.

Entre los aspectos a destacar de SCRUM encontramos:

- Gracias a las reuniones diarias entre el equipo, los problemas son detectados a tiempo permitiendo que se resuelvan rápidamente y no creen conflictos de mayor escala.
- Es flexible a cambios no previstos en el cronograma
- Permite entregar un buen producto en corto tiempo
- La productividad y motivación individual aumenta al ver resultados diarios
- Es idónea para equipos de trabajo pequeños

Las desventajas de la metodología aplicada al desarrollo de un prototipo de videojuego son:

- Si una actividad no se define bien en el cronograma, generará grandes pérdidas de tiempo y recursos.
- Los resultados en función de los parámetros a evaluar (en este caso narrativa) no son obtenidos de forma gradual y se deben tener una cantidad determinada de iteraciones o requisitos implementados para poder ser apreciados.
- En su estructura es imposible incluir el proceso de aprendizaje por parte de los integrantes.
- Al ser un equipo de trabajo pequeño para un proyecto tan grande, es necesario contar con un capital humano con grandes aptitudes y entrega, de lo contrario la metodología puede llegar a fracasar

CONCLUSIONES

- Se integró satisfactoriamente la estructura de narrativa simple propuesta por David Siegel con la metodología de desarrollo de software SCRUM.

- La creación de un videojuego en perspectiva de primera persona es práctica para un desarrollo semejante al de un proyecto de grado con las limitantes de tiempo y capital humano cómo las encontradas en este caso, gracias a una notable reducción en la cantidad de trabajo.

- El género en perspectiva de primera persona brinda una experiencia más personal al no tener un personaje visible realizando las acciones que impone el jugador en el ambiente virtual.

- El desarrollo de un videojuego utilizando una sincronía entre el proceso de desarrollo SCRUM y la estructura narrativa de David Siegel, arrojó resultados aceptables a nivel de retención y apreciación del argumento por parte del jugador.

- Existen claras desventajas al no tener una prelación del diseño del nivel y tareas a cumplir sobre la estructura narrativa, como fue el caso del juego desarrollado, fue repetitivo el caso dónde el flujo de juego fue detenido completamente por diálogos y desarrollo narrativo creando confusión e incluso frustración por parte del jugador.

- El entendimiento y retención del argumento por parte del jugador es un factor fundamental en su motivación para completar el juego, a pesar de su dificultad y presentación cruda y/o grotesca. Una narrativa elaborada de

forma correcta se convierte entonces en un factor esencial en el género de supervivencia en un contexto de horror.

- La metodología de desarrollo SCRUM es apropiada para el desarrollo de un videojuego ya que evidencia de forma inmediata y se adapta oportunamente a los inconvenientes originados por éstos.

- SCRUM resulta ineficiente cuando se implementa para un proyecto conformado por dos integrantes en un lapso de tiempo limitado, ya que la capacidad de resolución de tareas planteadas por la metodología se fundamenta en la versatilidad del grupo de trabajo. Cuando un miembro del equipo presenta un inconveniente, el cronograma del otro integrante se verá afectado y por ende el tiempo de resolución del proyecto.

- La creación de un juego de video exige un profundo conocimiento en el género, ya que sin éste no es posible conocer las limitantes de desarrollo a múltiples niveles, ni los requerimientos exigidos por los clientes en la actualidad.

- Las animaciones por captura de movimiento exigen un conocimiento técnico previo, pero su implementación brinda resultados de enorme calidad y su uso es absolutamente recomendado.

- Las competencias brindadas por la Universidad Militar a un Ingeniero en Multimedia son muy generales y necesitan profundizarse para brindar las cualidades necesarias para ser una parte relevante en el proceso de desarrollo de un juego de video actual.

TRABAJOS FUTUROS

- El prototipo de videojuego fue diseñado para que estudiantes de Ingeniería en Multimedia puedan realizar niveles anteriores o posteriores del mismo, basándose en el contexto desarrollado.

- La implementación del videojuego actual está desarrollada fundamentalmente en una estética de perspectiva en primera persona, estableciendo los parámetros para una elaboración con una perspectiva en tercera persona e inclusive como juego de plataformas longitudinal (Side Scroller).

GLOSARIO

HUMANOIDE: refiere a cualquier ser cuya estructura corporal se asemeja a la de un humano.

MAPA DE NORMALES: textura que simula el desplazamiento de los pixeles de acuerdo a la posición de la cámara.

MAPA DE PODER DIFUSO: textura en escala de grises que define el degradado entre brillo y sombra en el mapa de normales.

MAPA DIFUSO: color básico de una superficie, esta propiedad es la que más afecta la apariencia del modelo.

MAPA ESPECULAR: textura en escala de grises que define el poder de reflejo en las diversas áreas del material.

TESELACIÓN: edición de síntesis de textura sin obtener aberturas ni sobre posiciones.

BIBLIOGRAFÍA

[1] Wolfenstein 3D,Id Software, 1992.

[2] Medal Of Honor,DreamWorks Interactive, 1999.

[3] Call of Duty,Infinity Ward, 2003.

[4] Kingpin: Life of a Crime, Xatrix Entertainment, 1999.

[5] Half Life,Valve Software, 1998.

[6] Maniac Mansion,Lucasfilm Games, 1987.

[7] Day of The Tentacle,Lucas Arts, 1993.

[8] Full Throttle,Lucas Arts, 1995.

[9] Mass Effect 2,BioWare, 2010.

[10] IGN Staff. Immersion Games [en línea] Estados Unidos [Consultado 12 de Octubre de 2010]. Disponible en línea: http://games.ign.com/objects/755/755277.html

[11] Immersion Games & Timeline Interactive & Artificial Studios, 2007.

[12] Xor Games Staff. Bogotá, Colombia, 2008. [Consultado 12 de Octubre de 2010]. Disponible en Internet: http://www.xorgames.net/

[13] GametronStudios Staff. The Clover's Quest.Bucaramanga, 2010. [Consultado 12 de Octubre de 2010]. Disponible en Internet: http://www.gametronstudios.com/newsite/?p=128

[14] DOMINGUEZ MARIN, Lorena. Desarrollo de un prototipo de Videojuego basado en el cuento "Dalia Zair" del escritor colombiano Jairo Aníbal Niño, Bogotá DC. 2008. Trabajo de Grado (Ingeniera en Multimedia). Universidad Militar Nueva Granada. Ingeniería en Multimedia.

[15] MORA AVILAR, Michael y MUÑOZ ALVAREZ, Maritza. Análisis y diseño para el desarrollo de un prototipo video juego 3D, sobre conflictos entre tribus indígenas en Colombia, Bogotá DC. 2008. Trabajo de Grado (Ingenieros en Multimedia). Universidad Militar Nueva Granada. Ingeniería en Multimedia.

[16] ORTEGON RODRIGUEZ, Diego. Prototipo de videojuego El Mito de la Torre, Diego Rodriguez Ortegon. Bogotá DC. 2009. Trabajo de Grado (Ingeniero en Multimedia). Universidad Militar Nueva Granada. Ingeniería en Multimedia.

[17] BELLO RODRIGUEZ, Angela Vanesa y GOMEZ, Nestor Fabián. Prototipo de un videojuego de aventura, fantasía y acción para jóvenes basado en un mito de la cultura Chibcha tomando como base la Deidad Chibcha Bachué, Bogotá DC. 2010. Trabajo de Grado (Ingenieros en Multimedia). Universidad Militar Nueva Granada. Ingeniería en Multimedia.

[18] PEDERSEN, Rogert. Game Design Foundations. Editorial Wordware Game and Graphics Library. Segunda Edición (2009). pg: 193-211

[19] Imdb Staff, Plot Summary forBatmanBegins [en línea]. 2010 [Consultado 26 de Octubre de 2010]. Disponible en Internet:

http://www.imdb.com/title/tt0372784/plotsummary

[20] Imdb Staff, Plot Summary for Kill Bill Vol.2[en línea]. 2010 [Consultado 26 de Octubre de 2010]. Disponible en Internet: http://www.imdb.com/title/tt0378194/plotsummary

[21] Imdb Staff, Plot Summary for The Ring[en línea]. 2010 [Consultado 26 de Octubre de 2010]. Disponible en Internet: http://www.imdb.com/title/tt0298130/plotsummary

[22] Imdb Staff, Plot Summary for Indiana Jones: Raiders of the Lost Ark [en línea]. 2010 [Consultado 26 de Octubre de 2010]. Disponible en Internet: http://www.imdb.com/title/tt0082971/plotsummary

[23] Imdb Staff, Plot Summary for The Fugitive [en línea]. 2010 [Consultado 26 de Octubre de 2010]. Disponible en Internet: http://www.imdb.com/title/tt0106977/plotsummary

[24] Imdb Staff, Plot Summary for Indiana Jones and the Last Crusade [en línea]. 2010 [Consultado 26 de Octubre de 2010]. Disponible en Internet: http://www.imdb.com/title/tt0097576/plotsummary

[25] Imdb Staff, Plot Summary for Fight Club [en línea]. 2010 [Consultado 26 de Octubre de 2010]. Disponible en Internet: http://www.imdb.com/title/tt0137523/plotsummary

[26] Imdb Staff, Plot Summary for Terminator 2: Judgement Day[en línea]. 2010 [Consultado 26 de Octubre de 2010]. Disponible en Internet: http://www.imdb.com/title/tt0103064/plotsummary

[27] Imdb Staff, Plot Summary for The Lord of the Rings: The Return of the King[en línea]. 2010 [Consultado 26 de Octubre de 2010]. Disponible en Internet: http://www.imdb.com/title/tt0167260/plotsummary

[28] CARLQUIST, Jonas. Playing the story - Computer Games as a Narrative Genre. En : Human It. Umeå Area. Vol. 6, no.13 (2002); pg 7-53.

[29] Vale.Motivation and Amateur Game Development [en línea]. 2008. [Consultado 12 de Septiembre de 2010]. Disponible en Internet: http://www.rpgrevolution.com/tutorial/motivation-and-amateur-game-development_86.html

[30] PERRON, Bernard. Sign of a Threat: The Effects of Warning Systems in Survival Horror Games [en línea]. Canadá, 2004. [Consultado 8 de Octubre de 2010]. Disponible en Internet: www.cosignconference.org/downloads/papers/perron_cosign_2004.pdf

[31] BRYANT, Jennings y ZILLMAN, Dolf. Los efectos de los medios de comunicación: investigaciones y teorías. Editorial Paidos. Barcelona, 1996; pg. 228 - 229.

[32] ÁLVARO ESTRAMIANA, José Luis. Psicología social: perspectivas teóricas y metodológicas. Editorial Siglo Veintiuno de España Editores, S.A. Madrid, 1995; pg. 47 - 51

[33] BUXARRAIS ESTRADA, Maria Rosa. Los medios de comunicación y la educación en valores [en línea]. España, 1996. [Consultado 12 de Septiembre de 2010] Disponible en Internet: http://www.robertexto.com/archivo15/medios_educacion.htm

[34] GARCÍA CARRILLO, Antonio Alberto. Survival Horror [en línea]. España, 2007. [Consultado 8 de Octubre de 2010]. Disponible en Internet: http://www.lcc.uma.es/~afdez/ACTAS_MATVI_2008.pdf

[35] Silent Hill, Team Silent Konami, 1999

[36]Resident Evil, Capcom, 1996.

[37]Travis Fahs. 2009. IGN Presents the History of Survival Horror [en línea]. Estados Unidos, 2009.

[38] HIRST, Tony. The Process of Game Creation & the Game Design Document [en línea]. 2008. [Consultado 8 de Octubre de 2010]. Disponible en Internet: http://digitalworlds.wordpress.com/2008/04/10/the-process-of-game-creation-the-game-design-document/

[39] SOMMERVILLE, Ian. Ingeniería de Software. Editorial Pearson. Segunda Edición (2005); pg.: 61-63 "50"

[40] Laboratorio de Computación Universidad de los Andes, Colombia. [Consultado 18 de Octubre de 2010]. Disponible en línea: http://sistemas.ing.ula.ve/pr2/unidad_1/unidad_1_parte2.html

[41] SOMMERVILLE, Ian. Ingeniería de Software. Editorial Pearson. Segunda Edición (2005); pg.: 63-64

[42] SOMMERVILLE, Ian. Ingeniería de Software. Editorial Pearson. Segunda Edición (2005); pg.: 68 -69

[43] Proyectos Ágiles Staff. Beneficios de SCRUM [en línea]. [Consultado 8 de Octubre de 2010]. Disponible en Internet: http://www.proyectosagiles.org/beneficios-de-scrum

[44] Chuidian Staff. SCRUM [en línea]. [Consultado 15 de Julio de 2011]. Disponible en Internet: http://www.chuidiang.com/ood/metodologia/scrum.php

[45] STUART, Jenny. 10 Keys to Successful Scrum Adoption [en línea]. 2008 [Consultado 10 de Agosto de 2011] Disponible en Internet: http://i.i.com.com/cnwk.1d/html/itp/ConstruxSoftware_ScrumAdoption.pdf

[46] BARAB, Sasha A., Games Genres [en línea]. 2010 [Consultado 26 de Octubre de 2010]. Disponible en Internet: http://simworkshops.stanford.edu/05_1007/presentations/game_genres_brutlag.pdf

[47] Valve Staff. Steam, La Plataforma de Juegos Online Definitiva [en línea]. 2011 [Consultado 5 de Febrero de 2011]. Disponible en Internet: http://store.steampowered.com/about/

[48] Onlive Staff. Onlive [en línea]. 2011 [Consultado 5 de Febrero de 2011]. Disponible en Internet: http://www.onlive.com/#3

[49] ANGLISS, Sarah. Uncanny Valley [en línea]. 2009. [Consultado 27 de Julio de 2011]. Disponible en Internet: http://www.sarahangliss.com/spotted/uncannyvids

[50] Unity Technologies. Unity 3 [en línea]. 2011 [Consultado 28 de Julio de 2011]. Disponible en Internet: http://unity3d.com/unity/

[51] Epic Entertainment, Documentation – Epic UDK UDK [en línea]. 2010 [Consultado 26 de Octubre de 2010]. Disponible en Internet: http://www.udk.com/documentation

[52] Valve Software. Source Engine [en línea]. 2007 [Consultado 28 de Julio de 2011]. Disponible en Internet: http://source.valvesoftware.com/

[53] Id Software. Id Tech [en línea]. 2011 [Consultado 28 de Julio de 2011]. Disponible en Internet: http://www.idsoftware.com/

[54] CryTek. CryEngine 3 [en línea]. 2011 [Consultado 28 de Julio de 2011]. Disponible en Internet: http://mycryengine.com/index.php?conid=1

[55] SONDEREGUER, César. Manual de Iconografía Precolombina y su Análisis Morfológico. Editorial Geka, 1990

[56] Autodesk Staff. Autodesk Maya [en línea]. 2011 [Consultado 28 de Julio de 2011]. Disponible en Internet: http://usa.autodesk.com/maya/

[57] Epic Games Staff. Mesh Export Tools [en línea]. 2010[Consultado 28 de Julio de 2011]. Disponible en Internet: http://udn.epicgames.com/Two/ActorX.html

[58] CG Textures Staff. CG Textures [en línea]. [Consultado de Marzo a Julio de 2011]. Disponible en Internet: http://www.cgtextures.com/

[59] Image After Staff. Image After [en línea]. [Consultado de Marzo a Julio de 2011]. Disponible en Internet: http://www.imageafter.com/textures.php

[60]Mayang's Free Texture Staff. Mayang's Free Texture [en línea]. [Consultado de Marzo a Julio de 2011]. Disponible en Internet: http://mayang.com/textures/

[61] Faton Studio Staff. PixPlant [en línea]. 2011 [Consultado 16 de Diciembre de 2011] Disponible en Internet: http://www.pixplant.com/

[62] Autodesk Staff. Autodesk Mudbox [en línea]. 2011 [Consultado 28 de Julio de 2011]. Disponible en Internet: http://usa.autodesk.com/adsk/servlet/pc/index?id=13565063&siteID=123112

[63] Epic Games Stuff. Scaleform GFx Integration Overview [en línea]. 2010 [Consultado 12 de Marzo de 2011] Disponible en Internet: http://udn.epicgames.com/Three/Scaleform.html

[64] Adobe Stuff. Flash Cs5 Professional ActionScript 2.0 [en línea]. 2010 [Consultado 15 de Marzo de 2011] Disponible en Internet: http://help.adobe.com/es_ES/AS2LCR/Flash_10.0/help.html?content=00001925.html

[65] Autodesk Staff. Autodesk Motion Builder [en línea]. 2011 [Consultado 28 de Julio de 2011]. Disponible en Internet: http://usa.autodesk.com/adsk/servlet/pc/index?id=13581855&siteID=123112

[66] AL-HASAN, Hamad. Delay Fire [en línea]. 2010 [Consultado 5 de Julio de 2011] Disponible en Internet: http://forums.epicgames.com/showthread.php?t=741845&highlight=weapon+fire+delay

[67] PALACIO, Juan. El modelo SCRUM [en línea]. 2011 [Consultado 2 de Marzo de 2011] Disponible en Internet: http://www.navegapolis.net/files/s/NST-010_01.pdf

[68] FINE, Marshall. Siegel/McGehee and the 'Uncertainty" of Independent film [en línea]. 2009 [Consultado Enero 10 de 2011] Disponible en Internet: http://hollywoodandfine.com/interviews/?p=547

[69] SIEGEL, Dave. "The Nine Act Story Structure." GDC 1996. San Francisco: Miller Freeman, 1996

[70] CARDOZO, Luis. Tabla comparativa: de algunas metodologías de diseño de software. [en línea]. 2010. [Consultado 14 de Noviembre de 2011] Disponible en Internet: http://bicovemcali.blogspot.com/2010/12/tabla-comparativa-de-algunas-de-las.html

[71] Club de Desarrolladores Stuff. Metodología SCRUM [en línea]. 2008. [Consultado 14 de Noviembre de 2011] Disponible en Internet: http://www.clubdesarrolladores.com/articulos/mostrar/63-metodologia-scrum/2

[72] ESMURDOC, Caroline. Postmortem: Double Fine's Brutal Legend [en línea]. 2010. [Consultado 14 de Noviembre de 2011] Disponible en Internet: http://www.gamasutra.com/view/feature/4308/postmortem_double_fines_brutal_.php?print=1

[73] NUTT, Christian. Q&A: Valve's Swift On Left 4 Dead 2's Production, AI Boost [en línea]. 2009. [Consultado 14 de Noviembre de 2011] Disponible en Internet: http://www.gamasutra.com/view/news/25701/QA_Valves_Swift_On_Left_4_Dead_2s_Production_AI_Boost.php

[74] MENSAH, Kwasi. Your Time Estimate Are Broken [en línea]. 2011. [Consultado 14 de Noviembre de 2011] Disponible en Internet: http://www.gamasutra.com/blogs/KwasiMensah/20110816/8205/Your_Time_Estimates_Are_Broken.php

[75] CHERTOFF, Dustin. GOLDIEZ, Biran. LAVIOLA, Joseph. Virtual Experience Test: A Virtual Environment Evaluation Questionnaire. [en línea]. 2010. [Consultado 15 de Noviembre de 2011] Disponible en Internet: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5444804

[76] CROSBY, Tim. How video game testers work. [en línea]. 2008. [Consultado 16 de Noviembre de 2011] Disponible en Internet: http://electronics.howstuffworks.com/video-game-tester1.htm

Anexo A
Contexto del videojuego "Anathema"

“En el transcurso de la historia de nuestro planeta han existido un gran número de civilizaciones y en Colombia encontramos una de éstas, su nombre ya olvidado para no dejar ninguna evidencia hoy en día, con su residencia en una enorme ciudad enterrada bajo tierra. Es esta civilización la que logró en su cumbre llegar a la armonía máxima con su entorno, fundiendo la forma física de cada individuo que la componía en el espíritu del planeta mismo, desapareciendo completamente de nuestras dimensiones existenciales. Pero, antes de hacerlo, deciden asignar a un individuo de otra tribu menor el legado del conocimiento de sus avances y la tradición de que sólo puede existir una persona a la vez con este conocimiento, la cual debe pasarlo a una generación siguiente a la espera de que algún día el resto del mundo esté preparado para hacer la misma transición, y mientras tanto, protegerla. Así llegamos a los tiempos actuales, dónde la depredación del ser humano sobre la tierra genera estragos ambientales acabando con el balance mismo del planeta. Claudia Colomar es la poseedora del conocimiento perdido de esta civilización, y domina gran parte de sus secretos. Ella sabe dónde se encuentra la gran ciudad perdida, y aun más, conoce las terribles consecuencias si el ser humano llegase a entrar en ella sin estar preparado.

La compañía multinacional Sun Petrol ha sido la encargada de extraer el petróleo y recursos minerales del suelo colombiano, y en una de sus excavaciones hace sondeos bajo tierra lo que revela la existencia de la gran ciudad. El problema, es que no existe hoy en día ningún método que permita al hombre excavar tan profundo, lo que intriga a los extranjeros. Ellos deciden no revelar a nadie más la existencia de este lugar, a la vez que buscan antecedentes antropológicos de una posible existencia de la civilización perdida. Así es como contactan a Claudia, la cual entra en pánico al enterarse de la posibilidad de que el hombre, por primera vez, pueda desencadenar los mecanismos de defensa de la ciudad. Ella, recurre a los escritos de su maestro, los cuales explican detalladamente los procedimientos que se desencadenan si este caso se llegase a dar. Claudia nota un fragmento que la cautiva inmediatamente: “En caso de que el hombre que ponga un pie en el templo no esté preparado para trascender con el mundo, una terrible maldición será liberada, donde todo conocimiento de la existencia de éste será extirpada del planeta. El protector se convertirá en vehículo para el templo, y todo el poder del espacio y el tiempo le será otorgado, pero advertencia: si la maldición se libera y el protector posee el conocimiento de la existencia del templo, su falla será rotunda y su conciencia será extirpada para que el templo domine sus

acciones” Claudia ve en este pasaje una oportunidad que nunca antes se le había presentado a nadie. Ella accede a colaborar con los exploradores de Sun Petrol y les muestra cómo ingresar al templo. Ellos envían un enorme grupo de exploración completamente armado a la zona. Allí, ella activa los mecanismos para comenzar la maldición a la vez que se inyecta ella misma una solución de Benzodiacepina que la deja inconsciente y le induce una amnesia retrógrada la que borra de su memoria, de forma temporal, los transcursos pasados en su vida. Al soltarse la maldición el templo comienza a emitir sonidos horroríficos que espantan al equipo explorador. Ellos evacúan la zona con Claudia aún inconsciente, y la trasladan a Bogotá a su sede en el barrio La Candelaria para hacer un seguimiento a su estado. La maldición llega a Bogotá con aquellos que entraron al templo, y el caos comienza cuando se transforman en horribles criaturas que matan a golpes todo lo que se les topa. Aquí es cuando Claudia despierta, sin recuerdo alguno de quién es o por qué está allí, pero con fuerza, velocidad y otros atributos que la hacen ser algo fuera de lo normal... ”

Anexo B

Guión del prototipo de Videojuego "Anathema"

Claudia despierta en un cuarto oscuro.

CLAUDIA: Hola?

De repente, visualiza un Zombie que se acerca al jugador de forma amenazante obligando al jugador a huir de la habitación. Inmediatamente a la salida, se ve un viejo candelabro el cual toma y utiliza para golpear al Zombie. Ella procede entonces a huir del lugar, sigue el pasillo que se encuentra al lado del cuarto y se topa con unas escaleras, al bajar se topa con un bloqueo hecho para evitar el paso de un gran número de zombies. La única forma de salir del área es utilizando un ducto de ventilación cerca al techo. Del otro lado se encuentra lo que pareció haber sido un lugar donde un grupo de personas se refugió, para luego desaparecer. Allí se encuentra con un radio emitiendo sonidos de estática. Ella lo toma e intenta comunicarse por él.

CLAUDIA: Hola? Hay alguien ahí?

CLAUDIA: Hola?

RADIO: Quién está ahí?

CLAUDIA: Dios mío! Por favor, ayúdeme!

RADIO: Dígame su nombre.

CLAUDIA: Yo... yo no me acuerdo... Me acabo de levantar de una cama en este horrible lugar, y no me acuerdo de nada!

RADIO: Es usted parte del grupo de SP?

CLAUDIA: Qué? No se... no me acuerdo!

En este momento, la comunicación se corta

CLAUDIA: Están acá, por favor ayúdeme!

RADIO: *Estática*

De repente, se abre una puerta y el jugador se topa con una situación en la que es obligado a defenderse, y tras una secuencia de acción, el radio deja de emitir sonidos. La vía se abre a una serie de habitaciones repletas de Zombies hasta que de repente, el radio suena:.

RADIO: Hola? Sigue usted ahí?

CLAUDIA: Si! Dónde ha estado? Pensé que me había abandonado!

RADIO: No estoy autorizado a ayudar a nadie.

CLAUDIA: Por qué? Qué está pasando? Por favor venga y ayúdeme!

RADIO: Lo siento, no puedo. Estoy encerrado en un cuarto de seguridad en un edificio al frente del suyo y la única entrada está repleta de Zombies. La única persona que puede abrirla debe estar en el edificio, y si no ha regresado aún significa que ya es uno de ellos.

CLAUDIA: Por qué está solo? Es que no viene más ayuda?

RADIO: No, van a purgar la zona entera.

CLAUDIA: Qué?

RADIO: Mire, no hay tiempo para perder la cabeza, usted va a tener que ayudarme.

CLAUDIA: Yo? Está loco?

RADIO: Oiga, estoy completamente encerrado, y usted ha sido capaz de mantenerse con vida hasta ahora. Necesitamos una muestra de sangre guardada en el tercer piso de su edificio.

CLAUDIA: *Emite sonidos de respiros acelerados*

RADIO: Oiga! Compóngase! Tiene que salir ya de ese lugar! Puedo abrir la puerta que da a la calle por usted, pero solo si vuelve a la clínica de SP en el tercer piso y busca por cualquier muestra de sangre del Doctor Ubaque.

CLAUDIA: No voy a ayudarle, voy a encontrar una forma de escapar yo sola.

RADIO: No hay ninguna otra manera, el edificio donde usted se encuentra está completamente sellado, y el barrio entero está rodeado por la policía con órdenes de disparar a todo lo que se mueva.

CLAUDIA: Todavía no sé porque tengo que ayudarle, si todo está rodeado no hay forma que salgamos de aquí.

RADIO: Hay una forma, pero vamos a necesitar la sangre del Doctor Ubaque para lograrlo.

CLAUDIA: Muy bien... voy para arriba, pero sepa que no confío en usted.

El jugador ahora debe dirigirse al tercer piso, a través de múltiples grupos de Zombies. Una vez allí encuentra la sangre del Doctor Ubaque.

CLAUDIA: Ya tengo la sangre.

RADIO: Pero, qué es usted! Esas cosas no tienen forma de pararla!

CLAUDIA: Son frágiles, pero cómo sabe eso?

RADIO: Puedo verla por las cámaras de seguridad.

En el cuarto, se puede ver una cámara de seguridad

CLAUDIA: Bueno, ya tengo lo que necesita, ahora abra la puerta para que pueda salir de acá.

RADIO: Muy bien, pero tiene que tener cuidado. La calle está vigilada por la Policía, y si la llegan a ver, le van a disparar sin pensarlo. debe cruzarla escondida.

CLAUDIA: Muy bien, voy para allá.

El jugador se dirige a la entrada principal del edificio que se encuentra abierta . En la calle todo es caos, con llamas. A un extremo de la calle policía, al otro, la vía bloqueada.

CLAUDIA: Estoy en la calle.

RADIO: Quédese escondida y vaya al edificio del frente.

Una vez adentro, el jugador deberá enfrentarse a grupos de zombies hasta llegar a una puerta con una cámara de seguridad, y lo que parece ser un aparato electrónico para abrirla sin energía.

RADIO: Muy bien! Ahora deme la sangre por este hueco que voy a abrir.

CLAUDIA: Qué? No va a venir acá? No nos vamos a ver?

RADIO: No, no podemos. La puerta al lado suyo está bloqueada, pero puedo abrirla con un interruptor de este lado, lo único que necesito es la muestra de sangre que usted tiene.

Un dispositivo similar a un cajón se abre.

RADIO: Páseme la sangre.

CLAUDIA: Espero que tenga razón, solo quiero irme de acá!

El jugador interactúa con el cajón. Se cierra con la sangre en el.

RADIO: Esto es perfecto, muchas gracias. Ahora, solo espere a la purga señorita.

CLAUDIA: Qué?!

RADIO: Lo siento, pero no puedo dejarla escapar. Vine acá precisamente a contener amenazas como usted.

CLAUDIA: Pero no soy una amenaza! Estuve ayudándole todo el tiempo!

RADIO: Y al hacerlo me mostró que clase de monstruo es. No puedo arriesgarme, usted se queda acá. Gracias, me ayudó mucho.

CLAUDIA: No! Espere, no me deje acá! Abra esta puerta se lo pido!

RADIO: *Estática*

De repente, se iluminan unas luces verdes y comienzan a sonar voces grabadas con anterioridad.

SISTEMA: Acceso al sistema antiguo de alcantarillado es restringido. únicamente a personal no autorizado. Bienvenido Doctor Ubaque.

El sistema electrónico junto a la puerta se activa, listo para ser utilizado. El jugador tiene ahora la oportunidad de utilizarlo, el cual abrirá la puerta de forma automática.

SISTEMA: Acceso especial aceptado. Bienvenida Claudia Colomar.

CLAUDIA: Claudia? yo? Qué....?

La puerta se abre a una nueva habitación, y lo que parece ser una tapa de alcantarilla en el piso que lleva al alcantarillado antiguo. Al parecer, el cuarto de control donde se encontraba el sujeto del radio no ha abierto aún. El cuarto se encuentra lleno de manuscritos antiguos. Al acercarse a detallarlos, se activa una secuencia donde Claudia se acuerda cómo ella misma planeó el contexto

en el que se encuentra, liberando la maldición de los zombies y configurando la puerta que acaba de atravesar para que solamente ella pueda abrirla.

RADIO: Usted! Usted es Claudia Colomar!... Usted planeo todo esto!

CLAUDIA: Creo... Creo que si...

RADIO: Por qué?, por qué comenzó todo esto! Se suponía que tenía que protegerlo! Proteger el secreto!

CLAUDIA: Y dejar todo este poder de lado? No, tenía que hacerlo mío, y ahora que lo es, las cosas van a cambiar como yo quiero de una forma u otra.

Las luces del cuarto cambian a rojo y la voz automática suena.

SISTEMA: Paso denegado Doctor Ubaque. Por favor, comuníquese con el administrador del sistema para obtener acceso.

RADIO: Qué?! Usted me encerró?! Usted me encerró!

CLAUDIA: Este mundo ahora es mío.

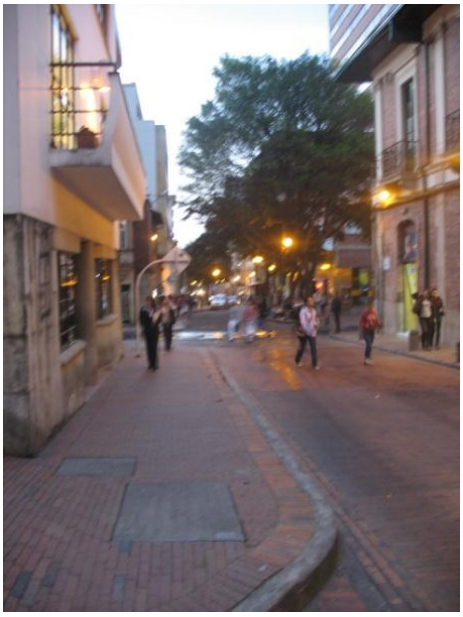
El jugador sólo puede interactuar con un dispositivo que se encuentra cerca a la salida por el alcantarillado. Una vez dentro, la voz automática suena de nuevo.

SISTEMA: Alerta. Reacción de ácido nafténico detectada en niveles estructurales superiores, evacuar las instalaciones inmediatamente. El sistema continua sonando hasta que el jugador llega al final del túnel, aquí la pantalla se oscurece y solo se oye la voz de Claudia.

CLAUDIA: *Risas*

Anexo C

Fotos La Candelaria



Anexo D

Fotos, Captura de Movimiento en Tecnoparque SENA

Anexo E
Clase Badguy

```
//Definición clase Badguy derivada de la clase UTPawn para asignar las características al zombie

//La clase UTPawn representa las características del jugador dentro del juego
//placeable permite que la clase creada en el UnrealEd se pueda colocar dentro del nivel del juego o en kismet
class BadguyUn extends UTPawn placeable;

//Declaración de variables

//Contiene el modelo del zombie con rig
var SkeletalMesh defaultMesh;

// Contiene las animaciones conectadas por nodos, creadas en el AnimTree Editor
var AnimTree defaultAnimTree;

//Contiene el Ragdoll Physics (animación muerte Zombie)
var PhysicsAsset defaultPhysicsAsset;

//Contiene las animaciones en secuencias
var array<AnimSet> defaultAnimSet;

//Para convocar los objetos de la clase BadguyController
var BadguyControllerUn MyController;

//Para llamar el AnimSet creado en el Content Browser del UDK Editor
var Name AnimSetName;

//Para determinar cuándo debe atacar el zombie
var bool AttAcking;

//Para asignarle un arma a un join del zombie, en este caso una esfera invisible para que sea MeleeWeapon
var AMXMeleeComponentUn myHandCollisionL;
var AMXMeleeComponentUn  myHandCollisionR;

//Para localizar la ubicación y rotación de los join o sockets del zombie
var vector SocketLocL;
var rotator SocketRotL;
var vector SocketLocR;
var rotator SocketRotR;

//Arreglo para controlar los PathNodes ( camino que recorre el zombie), esta variable se coloca dentro del editor gracias a la declaración var ()
var () array<NavigationPoint> MyNavigationPoints;

//simulated indica que la función es muy importante para el cliente
//La función PostBeginPlay() es llamada automáticamente por el motor al inicio del juego (Game Start)
```

```
//Llama a escena a todos los actores que se crearon
simulated function PostBeginPlay()
{
	//Constructor para llamar la función en la clase padre
	super.PostBeginPlay();

	//Condicional para agregarle el controlador al zombie de la clase BadGuy
	//Si esta null
	if (MyController == none)
	{
		//self es una constante y permite referenciar al actor de la clase que se llama
		//Se llama el objeto en la clase
		MyController = Spawn(class'BadguyControllerUn', self);

		//Se agrega el controlador al zombie
		MyController.SetPawn(self);
	}

	//GetSocketWorldLocationAndRotation es una función que devuelve la ubicación y rotación
del parámetro que se solicita, en este caso el del join del SkeletalMesh para agregar el arma
	//Si no encuentra el socket devuelve false
	//Parámetros de la función:	Nombre del socket que se va a buscar, Ubicación, Rotación
	Mesh.GetSocketWorldLocationAndRotation('WeaponPoint', SocketLocL, SocketRotL);
	Mesh.GetSocketWorldLocationAndRotation('WeaponPoint2', SocketLocR, SocketRotR);

	//Devuelve los parámetros encontrados a la clase que contiene el arma (Melee Weapon)
	myHandCollisionL = Spawn(class'AMXMeleeComponentUn',self,,SocketLocL,SocketRotL);
	myHandCollisionR =
Spawn(class'AMXMeleeComponentUn',self,,SocketLocR,SocketRotR);

	//Se agrega un parent a dicha ubicación
	Attach(myHandCollisionL);
	Attach(myHandCollisionR);

}

//Función que controla momento de ataque del zombie
function SetAttacking(bool atacar)
{
	//Cuando el bool devuelve true ataca
	AttAcking = atacar;

}

//Función que controla momento de muerte del zombie
function SetPhysicsAsset(bool mor)
{
	mor=true;
}

//simulated indica que la función es muy importante para el cliente
```

```
//El evento Tick  es llamado por el motor cada cierto cantidad de frames en que el juego se actualiza
simulated event Tick(float DeltaTime)
{
	//La variable local se activa sólamente cuando el evento se está ejecutando
	//UTPawn hace referencia a una clase predeterminada de Unreal
	local UTPawn gv;

	//Constructor para llamar la variable que contra cada cuanto se llamará el evento Tick
	super.Tick(DeltaTime);

	//Se agrega al mesh la ubiciación del socket para que la recorra constantemente, por variación de la posición del zombie
	Mesh.GetSocketWorldLocationAndRotation('WeaponPoint', SocketLocL, SocketRotL);
	Mesh.GetSocketWorldLocationAndRotation('WeaponPoint2', SocketLocR, SocketRotR);

	//La Locación del socket se agrega a la variable que contiene el arma en un parent, por variación de la posición del zombie
	myHandCollisionL.SetLocation(SocketLocL);
	myHandCollisionR.SetLocation(SocketLocR);

	//Para colisión del arma del jugador (actor principal) con el zombie y quitar vida del zombie
	//foreach es una iteración que permite que se trabajen con muchos actores a la misma vez
	//VisibleCollidingActors es una clase que devuelve todos los actores que colisionan con el zombie en un cierto rango para
	//parámentros: clase a la cual pertenece, actor o zombie y radio de la locación
	//Al haber colisión:
	foreach VisibleCollidingActors(class'UTPawn', gv, 100)
	{
		//Si el mesh fue asignado correctamente y ataca
		if(AttAcking && gv != none)
		{
			//Se el mesh tiene como nombre 'MyPawn_0' asignado en la clase UTPawn y tiene mas de 1 punto de vida
			if(gv.Name == 'MyPawn_0' && gv.Health > 0)
			{
				//Se quita vida
				gv.Health -= 50;

				//Variable tipo bool que indica si hubo o no daño
				gv.IsInPain();
			}
		}
	}
}

//simulated indica que la función es muy importante para el cliente
//La función SetCharacterClassFromInfo establecer propiedades básicas del UTPawn basado en los datos del zombie
simulated function SetCharacterClassFromInfo(class<UTFamilyInfo> Info)
{
```

```
        //Se agrega la información del SkeletalMesh o modelo con rig del zombie al mesh
        Mesh.SetSkeletalMesh(defaultMesh);

        //Se agrega la información de las secuencias de animaciones al mesh
        Mesh.AnimSets=defaultAnimSet;

        //Se agrega la información de las animaciones del zombie ya conectadas por nodos
        Mesh.SetAnimTreeTemplate(defaultAnimTree);

        //Se agrega la información de las animacion de muerte del zombie por Ragdoll Physics
        Mesh.SetPhysicsAsset(defaultPhysicsAsset);

}

//Propiedades generales del zombie, siempre deben ir al final de la clase para evitar errores
defaultproperties
{
        //Controla la vida del zombie
        Health=400

        //Se asigna el AnimSet creado en el content browser por medio del nombre del mismo
        AnimSetName="AnimsetZombie"'

        //El zombie empieza en modo pasivo, no ataca apenas aparece
        AttAcking=false

        //Se asigna el SkeletalMesh o modelo con rig creado y luego cargado en el Content
Browser, por medio de la ruta brindada por el UDK Editor
        defaultMesh=SkeletalMesh'Zombie.SkeletalMesh.Zombie'

        //Se asignan las animaciones del zombie ya conectadas por nodos creadas en el Content
Browser, por medio de la ruta brindada por el UDK Editor
        defaultAnimTree=AnimTree'Zombie.AnimTree.Arbol'

        //Se asignan las secuencias de animaciones del zombie creadas y luego cargadas en el
Content Browser, por medio de la ruta brindada por el UDK Editor
        defaultAnimSet(0)=AnimSet'Zombie.AnimSet.AnimsetZombie'

        //Se asigna la información de las animacion de muerte del zombie por Ragdoll Physics
        defaultPhysicsAsset=PhysicsAsset'Zombie.Physics.Zombie1_Physics'

        //Begin es el valor por defecto para iniciar el código, permite pausar o detener el código en
cualquier punto
        //Se declara  un objeto con propiedades de SkeletalMesh (variables preestablecidas de la
clase Engine.SkeletalMeshComponent de UDK)
        Begin Object Name=WPawnSkeletalMeshComponent

                //Variable tipo bool, al encontrarse en false el zombie estará visible en el viewport
                bOwnerNoSee=false

                //Variable tipo bool, al estar en true le crea una sombra al zombie
```

```
                CastShadow=true

                //Variable tipo bool, al estar en true no es necesario especificar información del
método de sombreado del personaje, es un complemento del CastShadow
                bAllowApproximateOcclusion=true
                //Variable tipo bool, complemento del CastShadow
                bUsePrecomputedShadows=false
                //Variable tipo bool, al estar en true asigna lightmaps a todas las static lights para
que influyan sobre el zombie, así no estén configuradas de tal forma
                bForceDirectLightMap=true
                //Arreglar escala del zombie
                Scale=0.8
                //Se asigna el SkeletalMesh o modelo con rig creado y luego cargado en el Content
Browser, por medio de la ruta brindada por el UDK Editor para poder visualizarlas en el workspace
                SkeletalMesh=SkeletalMesh'Zombie.SkeletalMesh.Zombie'
                //Se asignan las secuencias de animaciones del zombie creadas y luego cargadas
en el Content Browser, por medio de la ruta brindada por el UDK Editor para poder visualizarlas en
el workspace
                AnimSets(0)=AnimSet'Zombie.AnimSet.AnimsetZombie'
                //Se asignan las animaciones del zombie ya conectadas por nodos creadas en el
Content Browser, por medio de la ruta brindada por el UDK Editor para poder visualizarlas en el
workspace
                AnimTreeTemplate=AnimTree'Zombie.AnimTree.Arbol'

                //Se asigna la información de las animacion de muerte del zombie por Ragdoll
Physics
                PhysicsAsset=PhysicsAsset'Zombie.Physics.Zombie1_Physics'

        //End termina el código del objeto declarado
        End Object
        //Se le asigna a un mesh el objeto creado para poder encontrar los joins y aplicarlo al nivel
        mesh = WPawnSkeletalMeshComponent
        //Variable tipo bool para agregar colisión al actor
        bCollideActors=true
        //Variable tipo bool para activar evento tick, es necesario que esté en false
        bStatic=false
        //Variable tipo bool, para agregar movimiento al actor
        bMovable=True
        //Para evadir cornisas o repisas
        bAvoidLedges=true
        bStopAtLedges=true

        //Para determinar si el zombie se dirige hacia una cornisa, si la cornisa es mas corta que el
valor el actor no la atraviesa
        //Valor por default 4.0f

        LedgeCheckThreshold=0.5f
        GroundSpeed=200.0
}
```

Anexo F
Clase BadguyController

```
//Definición clase BadguyController derivada de la clase AIController

//La clase AIController es la clase base para la Inteligencia Artificial en Unreal, los controladores que se crean
//son actores no físicos que se pueden conectar a un Pawn para poder modificar su comportamiento
class BadguyControllerUn extends AIController;

//Se crea un actor de acuerdo a la clase creada Badguy que contiene las características del zombie basadas en la clase UTPawn
var BadguyUn MyBadguyPawn;

//Se crea un Pawn que hace referencia al jugador para asignarle características
var Pawn thePlayer;

//Arreglo para controlar los PathNodes ( camino que recorre el zombie), esta variable se coloca dentro del editor gracias a la declaración var ()
var () array<NavigationPoint> MyNavigationPoints;

//Número de nodo actual en el que se encuentra el zombie
var int actual_node;

//Número de nodo siguiente del path que tiene que recorrer el zombie
var int last_node;

//Determina la distancia en la cual reconocerá al jugador
var float perceptionDistance;

//Determina la distancia en la cual el zombie empieza a atacar
var float attackDistance;

//Indica la distancia entre el zombie y el jugador
var float distanceToPlayer;

//Para llamar el AnimSet creado en el Content Browser del UDK Editor
var Name AnimSetName;

//Para determinar cuando debe atacar el zombie
var bool AttAcking;

//Para determinar si debe o no seguir el camino trazado en el UDK Editor
var bool followingPath;

//Para perseguir al jugador cuando escuche ruidos
var bool noiseHeard;

//Tiempo que debe quedarse quieto el zombie cuando no ataca
var Float IdleInterval;
```

```
//Para crear un actor y agregarle estados y path
function SetPawn(BadguyUn NewPawn)
{
	//Se crea el actor de tipo BadGuy es decir el zombie
   MyBadguyPawn = NewPawn;

	//Agrega al zombie como el actor en la función Posses
	Possess(MyBadguyPawn, false);

	//Se agrega el arreglo de PathNodes al zombie
	MyNavigationPoints = MyBadguyPawn.MyNavigationPoints;
}

// La función Possess inicializa el zombie en la información del juego, proviene de la clase
PlayerController, parámetros por default
function Possess(Pawn aPawn, bool bVehicleTransition)
{
	//Si el zombie está a punto de morir
   if (aPawn.bDeleteMe)
	{
		//Se borra el script para ayudar a depurar procesos
		ScriptTrace();

		//Llama el estado muerto para aplicar atributos
		GotoState('Dead');
	}
	else
	{
		//Gira el zombie hacia el jugador
		Super.Possess(aPawn, bVehicleTransition);

		//Se agregan físicas de movimiento por default
		Pawn.SetMovementPhysics();
		if (Pawn.Physics == PHYS_Walking)
		{
			Pawn.SetPhysics(PHYS_Falling);
		}
   }
}

//Descripción estado de muerte
state Dead
{
	//Begin es el label por default que da el punto de partida para el estado
	Begin:
	//Se activa el PhysicsAsset
	MyBadguyPawn.SetPhysicsAsset(true);
}

//Descripción estado de reposo
state Idle
```

```
{
        //Reconocer al jugador
        event SeePlayer(Pawn SeenPlayer)
        {
                //el actor se liga al evento
           thePlayer = SeenPlayer;

                //Cálculo de distancia entre el jugador y el zombie
                //VSize devuelve el tamaño del vector que se crea al restar la ubicación entre el
jugador y el zombie
distanceToPlayer = VSize(thePlayer.Location - Pawn.Location);

                //Si la distancia entre el jugador y el zombie es menor que la percepciónde
profundidad
                if (distanceToPlayer < perceptionDistance)
      {
                        //Llama al estado perseguir al jugador
                        GotoState('Chaseplayer');
      }
   }

        //Begin es el label por default que da el punto de partida para el estado
        Begin:


                //El zombie queda en reposo y no ataca
                MyBadguyPawn.SetAttacking(true);



                //Si no localiza el jugador luego del intervalo se dirige al último path
                followingPath = true;

                //Para recorrido del path
                actual_node = last_node;

                //Llama al estado seguir el camino que se trazó en el UDK Editor
                GotoState('FollowPath');

}

//Descripción estado perseguir al jugador
state Chaseplayer
{
        //Begin es el label por default que da el punto de partida para el estado
        Begin:

        //Mientras corre el zombie no ataca
        MyBadguyPawn.SetAttacking(false);

        //Mientras q el jugador siga vivo
```

```
while (Pawn != none && thePlayer.Health > 0)
{


		//Si el jugador se puede alcanzar
		if (ActorReachable(thePlayer))
		{
			//Se calcula la distancia que hay entre el jugador y el zombie

			//VSize devuelve el tamaño del vector que se crea al restar la ubicación
entre el jugador y el zombie
			distanceToPlayer = VSize(thePlayer.Location - Pawn.Location);

			//Si la distancia calculada es menor que la distancia de ataque (50
unidades)
			if (distanceToPlayer < attackDistance)
			{
				//El zombie ataca al jugador llamando el estado atacar
				GotoState('Attack');
				break;
			}
			else
			{
				//La función MoveToward permite que el zombie se acerque un
poco más al objetivo, en este caso 20  unidades más
				MoveToward(thePlayer, thePlayer, 20.0f);

				//Si el zombie alcanzó el objetivo
				//La función ReachedDestination comprueba si lo alcanzó o no
				if(Pawn.ReachedDestination(thePlayer))
				{
					//El zombie ataca al jugador llamando el estado atacar
					GotoState('Attack');
					break;
				}
			}
		}

		//Si lo percibe pero no lo suficiente para atacarlo
		else
		{
			//Se busca la forma de llegar a él por medio de la función FindPathToward
			//Parámetros:    actor,bool opcional , tamaño máximo del camino a recorrer
			MoveTarget = FindPathToward(thePlayer,,perceptionDistance +
(perceptionDistance/2));

			//Si es posible alcanzarlo
			if (MoveTarget != none)
			{
				//Se calcula la distancia que hay entre el jugador y el zombie
```

```
                                        //VSize devuelve el tamaño del vector que se crea al restar la
ubicación entre el jugador y el zombie
                                        distanceToPlayer = VSize(MoveTarget.Location - Pawn.Location);

                                        //Si la distancia entre el jugador y el zombie es menor de 100
unidades
                                        if (distanceToPlayer < 100)

                                                //La función MoveToward permite que el zombie se
acerque un poco más al objetivo, en este caso 20 unidades más
                                                MoveToward(MoveTarget, thePlayer, 20.0f);
                                        else

                                                //Recorre el camino trazado hacia el objetivo 15 unidades
más
                                                MoveToward(MoveTarget, MoveTarget, 15.0f);

                                }
                                else
                                {
                                        //Si no se percibe queda en reposo llamando el estado reposo
                                        GotoState('Idle');
                                        break;
                                }
                        }

                Sleep(0);
        }
}

//Descripción estado atacar
state Attack
{
        //Begin es el label por default que da el punto de partida para el estado
        Begin:

        //Mientras el zombie está lo suficientemente cerca ataca al jugador
        MyBadguyPawn.SetAttacking(true);

        //Si el jugador está vivo
        while(true && thePlayer.Health > 0)
        {
                //Se calcula la distancia que hay entre el jugador y el zombie
                //VSize devuelve el tamaño del vector que se crea al restar la ubicación entre el
jugador y el zombie
                distanceToPlayer = VSize(thePlayer.Location - Pawn.Location);

                //Si la distancia entre ambos actores es mayor q el doble del rango permitido para
atacar
if (distanceToPlayer > attackDistance * 2)
      {
```

```
                        //El zombie no ataca
                        MyBadguyPawn.SetAttacking(false);
                        //Se llama al estado perseguir al jugador
            GotoState('Chaseplayer');
                        break;
        }
            Sleep(0);
            }
            //Si está muerto el jugador el zombie para de atacar
            MyBadguyPawn.SetAttacking(false);
}

//auto state permite que apenas se cargue el zombie la primera acción que realice sea seguir el
path trazado en el UDK Editor
auto state FollowPath
{
            //Reconocer al jugador
            event SeePlayer(Pawn SeenPlayer)
            {
                        //el actor se liga al evento
               thePlayer = SeenPlayer;

                        //Cálculo de distancia entre el jugador y el zombie
                        //VSize devuelve el tamaño del vector que se crea al restar la ubicación entre el
jugador y el zombie
distanceToPlayer = VSize(thePlayer.Location - Pawn.Location);

                        //Si la distancia entre el jugador y el zombie es menor que la percepciónde
profundidad
                        if (distanceToPlayer < perceptionDistance)
        {
                                    //Se puede localizar al jugador por algún sonido que haga
                                    noiseHeard = true;
                                    //Se sigue el path
                                    followingPath = true;
                                    //Llama al estado perseguir al jugador
                                    GotoState('Chaseplayer');
        }
    }
            //Begin es el label por default que da el punto de partida para el estado
            Begin:
            //Mientras que se recorre el camino
            while(followingPath)
            {
                        //Se reconoce la ubicación de los Path Nodes
                        MoveTarget = MyNavigationPoints[actual_node];
                        //Si el zombie llegó al nodo inicial
                        if(Pawn.ReachedDestination(MoveTarget))
                        {
                                    //WorldInfo.Game.Broadcast(self, "Nodo localizado");
                                    //Se recorre el arreglo de nodos
```

```
                        actual_node++;
                        if (actual_node >= MyNavigationPoints.Length)
                        {
                                actual_node = 0;
                        }
                        last_node = actual_node;
                        MoveTarget = MyNavigationPoints[actual_node];
                }
                //Para luego moverse a los siguientes nodos trazados
                if (ActorReachable(MoveTarget))
                {
                        MoveToward(MoveTarget, MoveTarget);
                }

                //Si no llegó al primero nodo, busca la forma de llegar a él por medio de la función
FindPathToward
                else
                {
                        MoveTarget = FindPathToward(MyNavigationPoints[actual_node]);

                        //Si lo encontró se dirige hacia él
                        if (MoveTarget != none)
                        {
                                MoveToward(MoveTarget, MoveTarget);
                        }
                }
                Sleep(0);
        }
}
//Propiedades generales del controlador, siempre deben ir al final de la clase para evitar errores
defaultproperties
{
        //Distancia de ataque 50 unidades
   attackDistance = 20
        //Rango de percepción 1000 unidades
   perceptionDistance = 1000
        //Se asigna el AnimSet creado en el content browser por medio del nombre del mismo
        AnimSetName ="AnimsetZombie""
        //Se inicializa en 0 el nodo actual para evitar errores
        actual_node = 0
        //Se inicializa en 0 el nodo siguiente para evitar errores
        last_node = 0
        //Se activa la opción recorrer el Path o camino trazado
        followingPath = true

        //Número de frames que queda en reposo el zombie cuando no ve al jugador
        IdleInterval = 2.0f


}
```

Anexo G
Clase AmXMeleeComponent

```
//Definición clase AMXMeleeComponent derivada de la clase Actor

//La clase AMXMeleeComponent es la encargada de situar una esfera invisible enparentada a una mano del zombie la cual causará el daño al jugador
//placeable permite que la clase creada en el UnrealEd se pueda colocar dentro del nivel del juego o en kismet
class AMXMeleeComponentUn extends Actor placeable;


//El evento touch permite comprobar si hay colisión, la función Tick es su complemento para que se actualize de acuerdo a la posición del zombie
event Touch(Actor Other, PrimitiveComponent OtherComp, vector HitLocation, vector HitNormal)
{
	//La variable local se activa sólamente cuando el evento se está ejecutando
	//UTPawn hace referencia a una clase predeterminada de Unreal
	local UTPawn HitPawn;

	//Para mirar si hay colisión con el componente
	HitPawn = UTPawn(Other);

	//Si hay colisión
	if(HitPawn != None)
	{
		//Si en la clase PlayerController el controlador no está nulo
		if( PlayerController(HitPawn.Controller) != None )
		{
			//Se aplica cierto daño al jugador
			//Parámetros: Cantidad de daño, Controlador, Lugar donde ocurrió la colisión, Fuerza causada por la colisión, Tipo de daño
			HitPawn.TakeDamage(2 , None, HitPawn.Location, vect(0,0,0) , class'UTDmgType_Lava');
			}

	}
}

//Propiedades generales de la esfera, siempre deben ir al final de la clase para evitar errores
DefaultProperties
{
	//Basados en la clase DrawSphereComponent se crea una esfera
	Begin Object Class=DrawSphereComponent Name=DrawSphere0

		//De color azul la malla y con alfa el material
		SphereColor=(B=255,G=0,R=0,A=255)

		//De radio 8
	SphereRadius=10.000000
```

```
                //Con capacidad de colisión
                CollideActors=true

    End Object

        //Se agrega colisión a la esfera con todos los componentes
    CollisionComponent=DrawSphere0

        //Se añade a los componentes del juego la esfera
    Components.Add(DrawSphere0)

        //Se agrega colisión con todos los actores
        bCollideActors=true

        //Variable tipo bool para activar evento tick, es necesario que esté en false
    bStatic=false

        //Se puede desplazar por el nivel
    bMovable=true

        //Se activa la función de parent para unir con la mano del zombie y así se desplace a la par
del mismo
bEdShouldSnap=True
}
```

Anexo H
Clase UTWeap_Uzi

```
class UTWeap_Uzi extends UTWeap_ShockRifleBase_Uzim;

// AI properties (for shock combos)
var UTProj_ShockBall ComboTarget;
var bool bRegisterTarget;
var bool bWaitForCombo;
var vector ComboStart;

var bool bWasACombo;
var int CurrentPath;
//-----------------------------------------------------------------
// AI InterFface
function float GetAIRating()
{
	local UTBot B;

	B = UTBot(Instigator.Controller);
	if ( (B == None) || (B.Enemy == None) || Pawn(B.Focus) == None )
		return AIRating;

	if ( bWaitForCombo )
		return 1.5;
	if ( !B.ProficientWithWeapon() )
		return AIRating;
	if ( B.Stopped() )
	{
		if ( !B.LineOfSightTo(B.Enemy) && (VSize(B.Enemy.Location - Instigator.Location)
< 5000) )
			return (AIRating + 0.5);
		return (AIRating + 0.3);
	}
	else if ( VSize(B.Enemy.Location - Instigator.Location) > 1600 )
		return (AIRating + 0.1);
	else if ( B.Enemy.Location.Z > B.Location.Z + 200 )
		return (AIRating + 0.15);

	return AIRating;
}


/**
* Overriden to use GetPhysicalFireStartLoc() instead of Instigator.GetWeaponStartTraceLocation()
* @returns position of trace start for instantfire()
*/
simulated function vector InstantFireStartTrace()
{
	return GetPhysicalFireStartLoc();
```

```
}

function SetComboTarget(UTProj_ShockBall S)
{
	if ( !bRegisterTarget || (UTBot(Instigator.Controller) == None) || (Instigator.Controller.Enemy
== None) )
		return;

	bRegisterTarget = false;
	bWaitForCombo = true;
	ComboStart = Instigator.Location;
	ComboTarget = S;
	ComboTarget.Monitor(UTBot(Instigator.Controller).Enemy);
}

function float RangedAttackTime()
{
	local UTBot B;

	B = UTBot(Instigator.Controller);
	if ( (B == None) || (B.Enemy == None) )
		return 0;

	if ( B.CanComboMoving() )
		return 0;

	return FMin(2,0.3 + VSize(B.Enemy.Location -
Instigator.Location)/class'UTProj_ShockBall'.default.Speed);
}

function float SuggestAttackStyle()
{
	return -0.4;
}

simulated function StartFire(byte FireModeNum)
{
	if ( bWaitForCombo && (UTBot(Instigator.Controller) != None) )
	{
		if ( (ComboTarget == None) || ComboTarget.bDeleteMe )
			bWaitForCombo = false;
		else
			return;
	}
	Super.StartFire(FireModeNum);
}

function DoCombo()
{
	if ( bWaitForCombo )
	{
```

```
                bWaitForCombo = false;
                if ( (Instigator != None) && (Instigator.Weapon == self) )
                {
                        StartFire(0);
                }
        }
}

function ClearCombo()
{
        ComboTarget = None;
        bWaitForCombo = false;
}

/* BestMode()
choose between regular or alt-fire
*/
function byte BestMode()
{
        local float EnemyDist;
        local UTBot B;

        bWaitForCombo = false;
        B = UTBot(Instigator.Controller);
        if ( (B == None) || (B.Enemy == None) )
                return 0;

        if (B.IsShootingObjective())
                return 0;

        if ( !B.LineOfSightTo(B.Enemy) )
        {
                if ( (ComboTarget != None) && !ComboTarget.bDeleteMe && B.CanCombo() )
                {
                        bWaitForCombo = true;
                        return 0;
                }
                ComboTarget = None;
                if ( B.CanCombo() && B.ProficientWithWeapon() )
                {
                        bRegisterTarget = true;
                        return 1;
                }
                return 0;
        }

        EnemyDist = VSize(B.Enemy.Location - Instigator.Location);

        if ( (EnemyDist > 4*class'UTProj_ShockBall'.default.Speed) || (EnemyDist < 150) )
        {
                ComboTarget = None;
```

```
		return 0;
	}

	if ( (ComboTarget != None) && !ComboTarget.bDeleteMe && B.CanCombo() )
	{
		bWaitForCombo = true;
		return 0;
	}

	ComboTarget = None;

	if ( (EnemyDist > 2500) && (FRand() < 0.5) )
		return 0;

	if ( B.CanCombo() && B.ProficientWithWeapon() )
	{
		bRegisterTarget = true;
		return 1;
	}
/*
	// consider using altfire to block incoming enemy fire
	if (EnemyDist < 1000.0 && B.Enemy.Weapon != None && B.Enemy.Weapon.Class != Class
&& B.ProficientWithWeapon())
	{
		return (FRand() < 0.3) ? 0 : 1;
	}
	else
	{
		return (FRand() < 0.7) ? 0 : 1;
	}*/
}

// for bot combos
/*
simulated function Projectile ProjectileFire()
{
	local Projectile p;

	p = Super.ProjectileFire();
	if (UTProj_ShockBall(p) != None)
	{
		SetComboTarget(UTProj_ShockBall(P));
	}
	return p;
}*/

simulated function rotator GetAdjustedAim( vector StartFireLoc )
{
	local rotator ComboAim;

	// if ready to combo, aim at shockball
```

```
	if (UTBot(Instigator.Controller) != None && CurrentFireMode == 0 && ComboTarget !=
None && !ComboTarget.bDeleteMe)
	{
		// use bot yaw aim, so bots with lower skill/low rotation rate may miss
		ComboAim = rotator(ComboTarget.Location - StartFireLoc);
		ComboAim.Yaw = Instigator.Rotation.Yaw;
		return ComboAim;
	}

	return Super.GetAdjustedAim(StartFireLoc);
}

simulated state WeaponFiring
{
	/**
	* Called when the weapon is done firing, handles what to do next.
	*/
	simulated event RefireCheckTimer()
	{
		if ( bWaitForCombo && (UTBot(Instigator.Controller) != None) )
		{
			if ( (ComboTarget == None) || ComboTarget.bDeleteMe )
				bWaitForCombo = false;
			else
			{
				StopFire(CurrentFireMode);
				GotoState('Active');
				return;
			}
		}

		Super.RefireCheckTimer();
	}
}

simulated function ImpactInfo CalcWeaponFire(vector StartTrace, vector EndTrace, optional out
array<ImpactInfo> ImpactList, optional vector Extent)
{
	local ImpactInfo II;
	II = Super.CalcWeaponFire(StartTrace, EndTrace, ImpactList, Extent);
	bWasACombo = (II.HitActor != None && UTProj_ShockBall(II.HitActor) != none );
	return ii;
}

function SetFlashLocation( vector HitLocation )
{
	local byte NewFireMode;
	if( Instigator != None )
	{
		if (bWasACombo)
		{
```

```
                NewFireMode = 3;
            }
            else
            {
                NewFireMode = CurrentFireMode;
            }
            Instigator.SetFlashLocation( Self, NewFireMode , HitLocation );
        }
}

defaultproperties
{
        // Weapon SkeletalMesh
        Begin Object class=AnimNodeSequence Name=MeshSequenceA
        End Object

        // Weapon SkeletalMesh
        Begin Object Name=FirstPersonMesh
            SkeletalMesh=SkeletalMesh'Arma_Final.Arma.ArmaFinal'
            Scale = 0.4
            AnimSets(0)=AnimSet'Arma_Final.AnimacionesArma_Final'
            Animations=MeshSequenceA
            Rotation=(Yaw=-16384)
            FOV=60.0
        End Object

        //AttachmentClass=class'UTGameContent.UTAttachment_ShockRifle'
        AttachmentClass=class'MyFolder.UTAttachment_Uzi'

        Begin Object Name=PickupMesh
            SkeletalMesh=SkeletalMesh'Arma_Final.Candelabro_SK'
            Scale = 0.5
        End Object

        InstantHitMomentum(0)=+60000.0

        WeaponFireTypes(0)=EWFT_InstantHit
        WeaponFireTypes(1)=EWFT_None
        //WeaponFireTypes(1)=EWFT_Projectile
        //WeaponProjectiles(1)=class'UTProj_ShockBall'

        InstantHitDamage(0)=90
        FireInterval(0)=+0.77
        FireInterval(1)=+0.6
        InstantHitDamageTypes(0)=class'UTDmgType_ShockPrimary'
        InstantHitDamageTypes(1)=None

        WeaponFireSnd[0]=SoundCue'Arma_Final.Ataque_SC'
        //WeaponFireSnd[1]=SoundCue'A_Weapon_ShockRifle.Cue.A_Weapon_SR_AltFireCue'

        WeaponEquipSnd=SoundCue'Arma_Final.Pickup_SC'
```

```
	WeaponPutDownSnd=SoundCue'A_Weapon_ShockRifle.Cue.A_Weapon_SR_LowerCue'
	//PickupSound=SoundCue'A_Pickups.Weapons.Cue.A_Pickup_Weapons_Shock_Cue'

	MaxDesireability=0.65
	AIRating=0.65
	CurrentRating=0.65
	bInstantHit=false
	bSplashJump=true
	bRecommendSplashDamage=true
	bSniping=true
	ShouldFireOnRelease(0)=0
	ShouldFireOnRelease(1)=1
	ShotCost(0)=1
	ShotCost(1)=1
	FireOffset=(X=20,Y=5)
	PlayerViewOffset=(X=17,Y=10.0,Z=-8.0)
	AmmoCount=100000
	LockerAmmoCount=100000
	MaxAmmoCount=100000		//FireCameraAnim(1)=none

	//WeaponFireAnim(1)=WeaponIdle

	MuzzleFlashSocket=MF
	//MuzzleFlashPSCTemplate=WP_ShockRifle.Particles.P_ShockRifle_MF_Alt
	//MuzzleFlashAltPSCTemplate=WP_ShockRifle.Particles.P_ShockRifle_MF_Alt
	//MuzzleFlashColor=(R=200,G=120,B=255,A=255)
	//MuzzleFlashDuration=0.33
	//MuzzleFlashLightClass=class'UTGame.UTShockMuzzleFlashLight'
	CrossHairCoordinates=(U=256,V=0,UL=64,VL=64)
	LockerRotation=(Pitch=32768,Roll=16384)

	IconCoordinates=(U=728,V=382,UL=162,VL=45)

	WeaponColor=(R=160,G=0,B=255,A=255)

	InventoryGroup=4
	GroupWeight=0.5

	IconX=400
	IconY=129
	IconWidth=22
	IconHeight=48

	Begin Object Class=ForceFeedbackWaveform Name=ForceFeedbackWaveformShooting1

	Samples(0)=(LeftAmplitude=90,RightAmplitude=40,LeftFunction=WF_Constant,RightFuncti
on=WF_LinearDecreasing,Duration=0.1200)
	End Object
	WeaponFireWaveForm=ForceFeedbackWaveformShooting1

}
```

Anexo I
Clase UTWeap_ShockRifleBase_Uzim

```
/**
 * Copyright 1998-2011 Epic Games, Inc. All Rights Reserved.
 */
class UTWeap_ShockRifleBase_Uzim extends OCWeapon;

defaultproperties
{
	InventoryGroup=4
}
```

Anexo J
Clase OcWeapon

```
/**
 * ********************************************************************************************************************
 * OCWeapon. The shared Weapon class implementation.
 *
 * By Hamad Al-Hasan (aka seenooh) for The Outcasts project (http://www.the-outcasts.tk/).
 *
 * ********************************************************************************************************************
 * Version: 1.0
 *
 * Description: This class will be subclassed by all our weapon implementations. It will have
 *              all the common functional features between all weapons.
 *
 * Last Modified by: Hamad Al-Hasan
 *          Last Modify Date: Aug, 16th 2010
 *
 */
class OCWeapon extends UTWeapon;

/** This is an array of two elements (fire/alt fire) containing how much (in seconds) we can delay fire
after the fire animation starts */
var array<float> DelayFireTime;
/*
 * The WeaponFiring state is overriden to provide the delay fire functionality. The uncommented
functions are stock implementations
 *
 * Workflow:
 *
 *      1- Once we fire, WeaponFiring state is activated, and BeginState is immediately executed.
 *      2- If we don't have delay, we'll fire immediately and refire according to FireInterval.
 *      3- If we have delay, we'll play the firing animation first. Then activate the delay timer,
 *         which will delay the firing logic according to DelayFireTime[FireNumMode].
 *      4- Once we fire, the refire logic will start.
 */
simulated state WeaponFiring
{
        simulated event bool IsFiring()
        {
                return true;
        }
        /**
         * Timer event, call is set up in Weapon::TimeWeaponFiring().
         * The weapon is given a chance to evaluate if another shot should be fired.
         * This event defines the weapon's rate of fire.
         */
        simulated function RefireCheckTimer()
        {
                // if switching to another weapon, abort firing and put down right away
                if( bWeaponPutDown )
```

```
                {
                        `LogInv("Weapon put down requested during fire, put it down now");
                        PutDownWeapon();
                        return;
                }
                // If weapon should keep on firing, then do not leave state and fire again.
                if( ShouldRefire() )
                {
                        //Hamad: If we have delay, refire according to our delay logic
                        if (DelayFireTime[CurrentFireMode] > 0)
                        {
                                ClearTimer('DelayFire');
                                PlayFireEffects( CurrentFireMode );
                        }
                        else
                                FireAmmunition(); //Else, follow the default implementation

                        return;
                }
                // Otherwise we're done firing
                HandleFinishedFiring();
        }

        simulated event BeginState( Name PreviousStateName )
        {
                `LogInv("PreviousStateName:" @ PreviousStateName);
                //If we don't have delays, resume with default implementation
                if (DelayFireTime[CurrentFireMode] <= 0)
                {
                        FireAmmunition();
                        TimeWeaponFiring( CurrentFireMode );
                }
                else
                        PlayFireEffects( CurrentFireMode ); //Otherwise, use ours
        }

        simulated event EndState( Name NextStateName )
        {
                `LogInv("NextStateName:" @ NextStateName);
                // Set weapon as not firing
                ClearFlashCount();
                ClearFlashLocation();
                ClearTimer('RefireCheckTimer');

                NotifyWeaponFinishedFiring( CurrentFireMode );
        }
}
simulated function TimeWeaponFiring( byte FireModeNum )
{
        // if weapon is not firing, then start timer. Firing state is responsible to stopping the timer.
        if( !IsTimerActive('RefireCheckTimer') )
```

```
	{
		SetTimer( GetFireInterval(FireModeNum) , true, nameof(RefireCheckTimer) );
	}
}

//Hamad: This is the delay timer function. It'll fire the default implementation and clear itself.
simulated function DelayFire()
{
	FireAmmunition();
	TimeWeaponFiring( CurrentFireMode );
	ClearTimer('DelayFire');
}

//Hamad: Play the animation and activate the delay timer if we are in delay mode
simulated function PlayFireEffects( byte FireModeNum, optional vector HitLocation )
{

	if (DelayFireTime[FireModeNum] > 0) // Do we have delay?
	{
		/* Is the timer active already? if yes, don't do anything. This is a tricky part.
		 * We injected this function at the BeginState, which is originally fired by
Pawn.WeaponFired()
		 * I wanted to remove this effect without touching the pawn's base code, so I could
do it
		 * by checking if the timer is active or not (caused by our BeginState). If it does,
then
		 * the second call by Pawn.WeaponFired() will have no effect, this avoid
duplication.
		 */
		if( !IsTimerActive('DelayFire') )
		{
			SetTimer( DelayFireTime[FireModeNum] , true, 'DelayFire' );
			super.PlayFireEffects(FireModeNum, HitLocation);
		}

	}
	else
	{
		//No delay. Resume default implementation.
		super.PlayFireEffects(FireModeNum, HitLocation);
	}

}

DefaultProperties
{

	DelayFireTime(0) = 0.2;   //By default, no delay for the main fire.
//	DelayFireTime(1) = 0;   //By default, no delay for the alt fire.

}
```

Anexo K
Clase UTAttachment_Uzi

```
/**
* Copyright 1998-2011 Epic Games, Inc. All Rights Reserved.
*/
class UTAttachment_Uzi extends UTWeaponAttachment;

//var ParticleSystem BeamTemplate;
var class<UDKExplosionLight> ImpactLightClass;

var int CurrentPath;

simulated function bool AllowImpactEffects(Actor HitActor, vector HitLocation, vector HitNormal)
{
	return (HitActor != None && UTProj_ShockBall(HitActor) == None &&
Super.AllowImpactEffects(HitActor, HitLocation, HitNormal));
}

defaultproperties
{
	// Weapon SkeletalMesh
	Begin Object Name=SkeletalMeshComponent0
		SkeletalMesh=SkeletalMesh'Arma_Final.Arma.ArmaFinal'
		Scale = 2
	End Object

	//DefaultImpactEffect=(ParticleTemplate=ParticleSystem'VH_Scorpion.Effects.PS_Wheel_
Rocks', Sound=SoundCue'A_Weapon_ShockRifle.Cue.A_Weapon_SR_AltFireImpactCue')
	//ImpactEffects(0)=(MaterialType=Water,
ParticleTemplate=ParticleSystem'WP_LinkGun.Effects.P_WP_Linkgun_Beam_Impact_HIT',
Sound=SoundCue'A_Weapon_Link.Cue.A_Weapon_Link_FireCue')
	//BulletWhip=SoundCue'A_Weapon_ShockRifle.Cue.A_Weapon_SR_WhipCue'

	//WeaponClass=class'UTWeap_Uzi'

}
```

Anexo L
Clase UTAmmo_Uzi

```
/**
* Copyright 1998-2011 Epic Games, Inc. All Rights Reserved.
 */
class UTAmmo_Uzi extends UTAmmoPickupFactory;

defaultproperties
{
	AmmoAmount=3
	TargetWeapon=class'UTWeap_ShockRifleBase'
	//PickupSound=SoundCue'A_Pickups.Ammo.Cue.A_Pickup_Ammo_Shock_Cue'
	MaxDesireability=0.28

	Begin Object Name=AmmoMeshComp
		StaticMesh=StaticMesh'mio.caja_fbx'
		Translation=(X=0.0,Y=0.0,Z=-15.0)

	End Object

	Begin Object Name=CollisionCylinder
		CollisionHeight=14.4
	End Object
}
```

Anexo M
Clase UTAmmoPickupFactory_Uzim

```
/**
 * Copyright 1998-2011 Epic Games, Inc. All Rights Reserved.
 */
//=============================================================================
// Ammo.
//=============================================================================

class UTAmmoPickupFactory_Uzim extends UTItemPickupFactory_Uzim
	ClassGroup(Pickups,Ammo)
	abstract;

/** The amount of ammo to give */
var int AmmoAmount;

/** The class of the weapon this ammo is for. */
var class<UTWeapon> TargetWeapon;

function SpawnCopyFor( Pawn Recipient )
{
	if ( UTInventoryManager(Recipient.InvManager) != none )
	{
		UTInventoryManager(Recipient.InvManager).AddAmmoToWeapon(AmmoAmount,
TargetWeapon);
	}

	Recipient.PlaySound(PickupSound);
	Recipient.MakeNoise(0.2);

	if (PlayerController(Recipient.Controller) != None)
	{

	PlayerController(Recipient.Controller).ReceiveLocalizedMessage(MessageClass,,,,Class);
	}
}

simulated static function UpdateHUD(UTHUD H)
{
	local Weapon CurrentWeapon;

	Super.UpdateHUD(H);

	if ( H.PawnOwner != None )
	{
		CurrentWeapon = H.PawnOwner.Weapon;
		if ( CurrentWeapon == None )
```

```
                return;
        }

        if ( Default.TargetWeapon == CurrentWeapon.Class )
                H.LastAmmoPickupTime = H.LastPickupTime;
}

auto state Pickup
{
        /* ValidTouch()
         Validate touch (if valid return true to let other pick me up and trigger event).
        */
        function bool ValidTouch( Pawn Other )
        {
                if ( !Super.ValidTouch(Other) )
                {
                        return false;
                }

                if ( UTInventoryManager(Other.InvManager) != none)
                  return UTInventoryManager(Other.InvManager).NeedsAmmo(TargetWeapon);

                return true;
        }

        /* DetourWeight()
        value of this path to take a quick detour (usually 0, used when on route to distant objective,
but want to grab inventory for example)
        */
        function float DetourWeight(Pawn P,float PathWeight)
        {
                local UTWeapon W;

                W = UTWeapon(P.FindInventoryType(TargetWeapon));
                if ( W != None )
                {
                        return W.DesireAmmo(true) * MaxDesireability / PathWeight;
                }
                return 0;
        }
}

function float BotDesireability(Pawn P, Controller C)
{
        local UTWeapon W;
        local UTBot Bot;
        local float Result;

        Bot = UTBot(C);
        if (Bot != None && !Bot.bHuntPlayer)
        {
```

```
                W = UTWeapon(P.FindInventoryType(TargetWeapon));
                if ( W != None )
                {
                        Result = W.DesireAmmo(false) * MaxDesireability;
                        // increase desireability for the bot's favorite weapon
                        if (ClassIsChildOf(TargetWeapon, Bot.FavoriteWeapon))
                        {
                                Result *= 1.5;
                        }
                }
        }
        return Result;
}

defaultproperties
{
        //RespawnSound=SoundCue'A_Pickups.Ammo.Cue.A_Pickup_Ammo_Respawn_Cue'

        MaxDesireability=+00000.200000

        Begin Object Name=CollisionCylinder
                CollisionRadius=24.0
                CollisionHeight=9.6
        End Object

        Begin Object Class=StaticMeshComponent Name=AmmoMeshComp
           CastShadow=FALSE
                bCastDynamicShadow=FALSE
                bAcceptsLights=TRUE
                bForceDirectLightMap=TRUE

        LightingChannels=(BSP=TRUE,Dynamic=FALSE,Static=TRUE,CompositeDynamic=TRUE)
                LightEnvironment=PickupLightEnvironment
                CollideActors=false
                BlockActors = false
                BlockZeroExtent=false
                BlockNonZeroExtent=false
                BlockRigidBody=false
                Scale=1.8
                MaxDrawDistance=4000
        End Object
        PickupMesh=AmmoMeshComp
        Components.Add(AmmoMeshComp)
}
```

Anexo N
Clase UTItemPickupFactory_Uzim

```
/**
 * Copyright 1998-2011 Epic Games, Inc. All Rights Reserved.
 */

class UTItemPickupFactory_Uzim extends UTPickupFactory
	abstract;

var		SoundCue			PickupSound;
var		localized string	PickupMessage;			// Human readable description when picked up.
var		float				RespawnTime;

/** Human readable string describing the use of this item (for UI) */
var		localized string	UseHintMessage;

simulated function InitializePickup()
{
	InitPickupMeshEffects();
}
static function string GetLocalString(
	optional int Switch,
	optional PlayerReplicationInfo RelatedPRI_1,
	optional PlayerReplicationInfo RelatedPRI_2
	)
{
	return Default.PickupMessage;
}

/**
 * Give the benefit of this pickup to the recipient
 */
function SpawnCopyFor( Pawn Recipient )
{
	Recipient.PlaySound( PickupSound );
	Recipient.MakeNoise(0.2);

	if ( PlayerController(Recipient.Controller) != None )
	{

	PlayerController(Recipient.Controller).ReceiveLocalizedMessage(MessageClass,,,,class);
	}
}
defaultproperties
{
 //RespawnTime=10.000000
	 MessageClass=class'UTPickupMessage'
	 InventoryType=class'UTGame.UTPickupInventory'
}
```

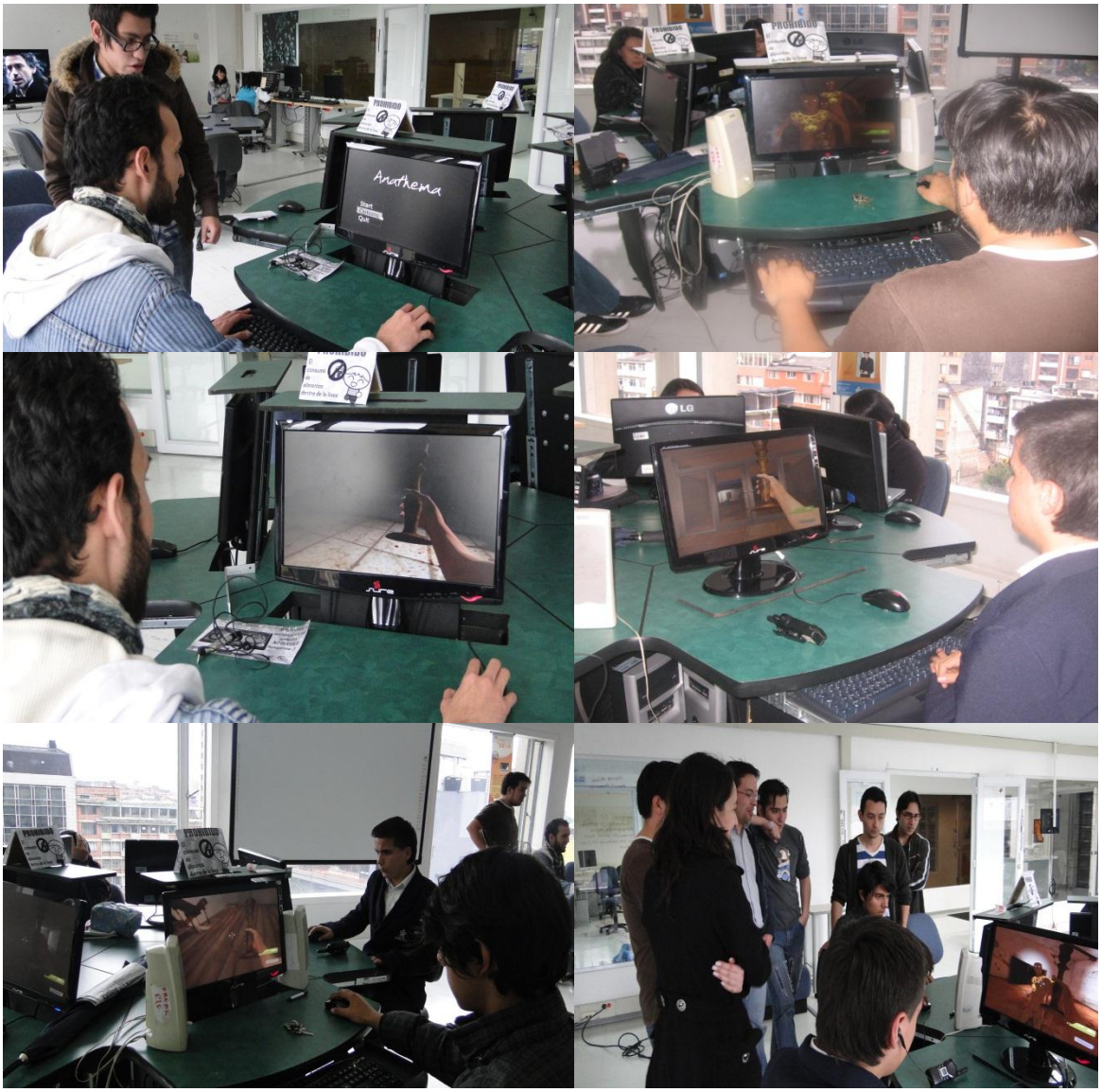
Anexo O
Clase MyHud

```
//Definición clase MyHUD derivada de la clase UTHUD para asignar las características a la interfaz de usuario
class MyHUD extends UTHUD;
//La función DrawBar se encarga de posicionar y mostrar la vida regenerativa del personaje
//Value es la vida del jugador
//MaxValue es la vida maxima del jugador
//X, Y dan la posición donde se mostrará la vida
function DrawBar(float Value, float MaxValue,int X, int Y)
{       //Declaración numero de imagenes a mostrar
  local int NbCases,i;
        //Número de imágenes activas a dibujar
  NbCases = 20 * Value / MaxValue;
        //Número de imágenes ya dibujadas
  i=0;
        //Mostrar Imagenes activas (sin daño verde)
  while(i < NbCases && i < 20)
  {             //Se posiciona en 20,20(esquina superior izquierda)
                Canvas.SetPos(X,Y);
                //Imagen a escala 0.5
                Canvas.DrawTexture (Texture2D'mio.ojopng', 0.5f);
                //Se agrega verde y alfa de 60
                Canvas.SetDrawColor(0,255,0,60);
                //Contador
                i++;
        }
  //Mostrar Imagenes no activas (con daño rojas)
        while(i < 20)
  {             //Se posiciona en 20,20(esquina superior izquierda)
    Canvas.SetPos(X,Y);
                //Imagen a escala 0.5
        Canvas.DrawTexture (Texture2D'mio.ojopng2', 0.5f);
                //Se agrega rojo y alfa de 15
                Canvas.SetDrawColor(255,0,0,15);
                //Contador
                i++;
        }
}
//La función DrawGameHud es llamada cada vez que se refresca la imagen el HUD
function DrawGameHud()
{       //Si el judador no esta muerto y esta quieto

  if ( !PlayerOwner.IsDead() && !UTPlayerOwner.IsInState('Spectating'))
  {             //Se llama la función DrawBar previamente declarada
    DrawBar(PlayerOwner.Pawn.Health, PlayerOwner.Pawn.HealthMax,20,20);
        }
}
defaultproperties
{
}
```

Anexo P
Clase MyGame

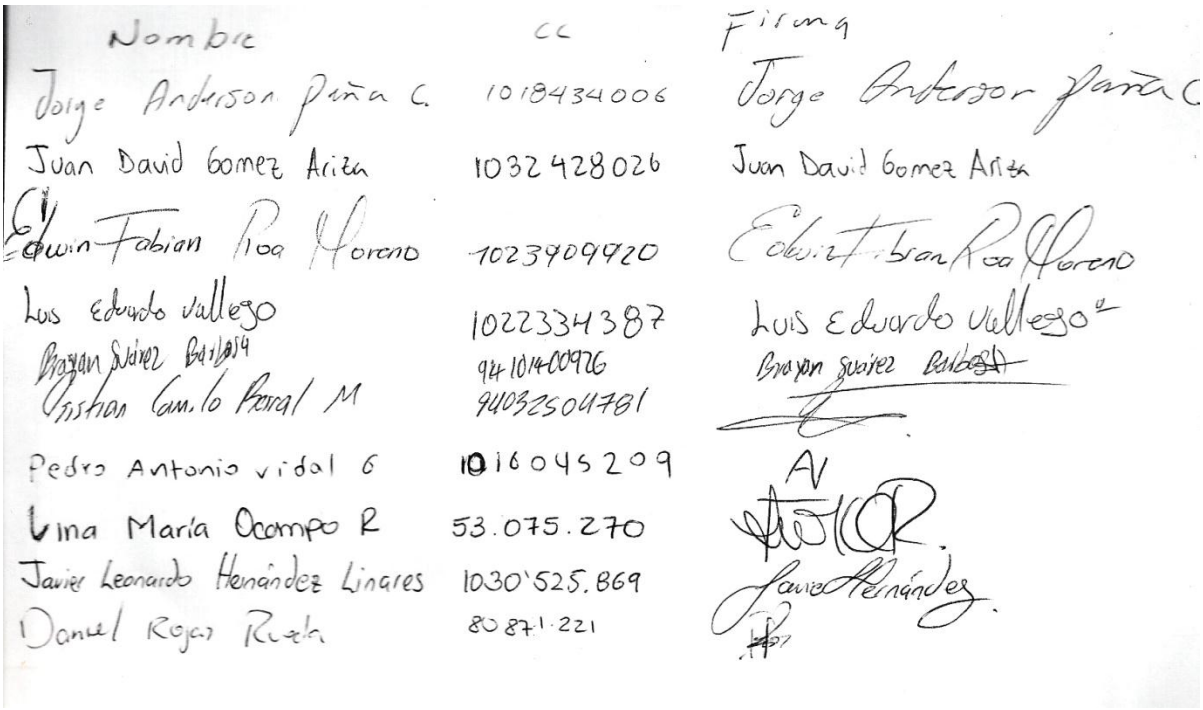
```
//Definición clase MyGame derivada de la clase UTDeathmatch
class MyGame extends UTDeathmatch
config(MyGame);

defaultproperties
{
	//Esta clase será vista en el WorldProperties del UDK para visualizar la inerfaz creada y no la interfaz por defecto
HUDType=class'MyFolder.MyHUD'
	bUseClassicHUD=true
}
```

Anexo Q
Registro fotográfico pruebas prototipo



AnexoR

Registro Jugadores para prueba del prototipo

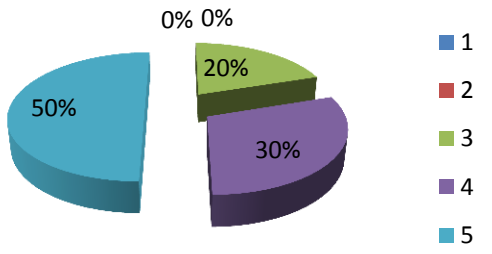


Nombre	cc	Firma
Jorge Anderson Peña C.	1018434006	Jorge Anderson Peña C
Juan David Gomez Arita	1032428026	Juan David Gomez Arita
Edwin Fabian Roa Moreno	1023909920	Edwin Fabian Roa Moreno
Luis Eduardo Vallejo	1022334387	Luis Eduardo Vallejo
Brayan Suarez Bal	94101400976	Brayan Suarez
Cristian Camilo Perez M	940325041781	
Pedro Antonio vidal G	1016045209	
Lina María Ocampo R	53.075.270	
Javier Leonardo Hernández Linares	1030'525.869	Javier Hernández
Daniel Rojas Rueda	80871.221	

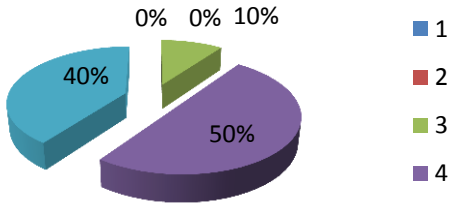
Anexo S

Resultados completos de evaluación narrativa

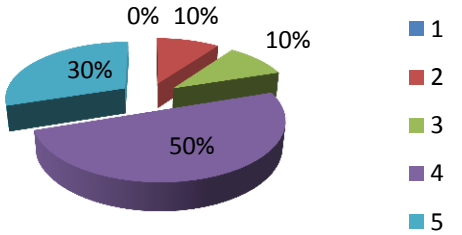
1. Experimenté un alto grado de interacción con los agentes virtuales dentro del juego



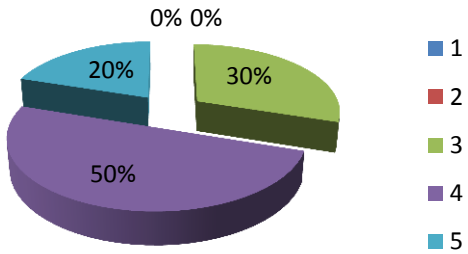
2. Cuando experimenté una reacción emocional sentí que ésta era apropiada dados los eventos que ocurrían en el juego en ese momento



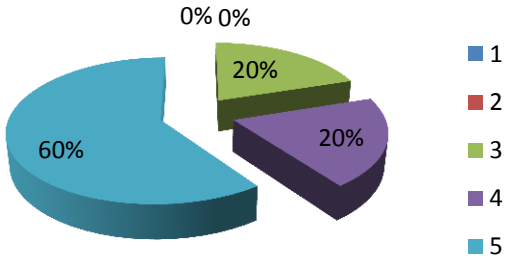
3. Encontré que la historia en el ambiente virtual hizo un buen trabajo para explicar las tareas que debía desarrollar



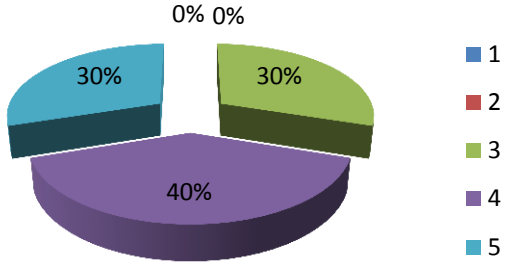
4 - Sentí una diversidad de emociones mientras jugaba



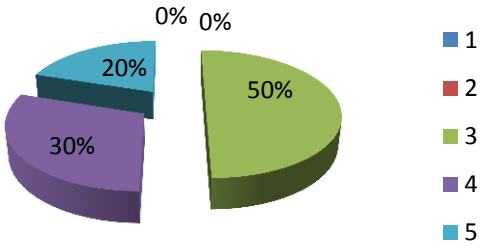
5 - Encontré que un alto nivel de interacción con los agentes virtuales era necesario para completar mis tareas dentro del juego



6 - Sentí que los agentes virtuales controlados por el computador (inteligencia artificial) fueron bien utilizados en el juego



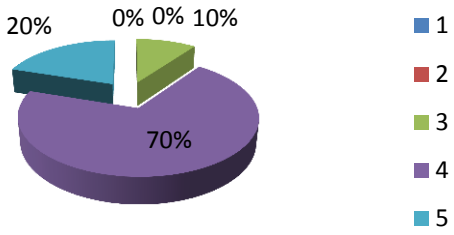
7 - Sentí que el ambiente en el juego usó múltiples técnicas para provocar emociones



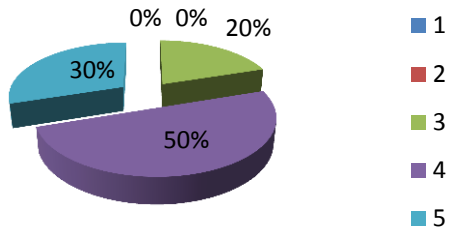
Anexo T

Resultados completos de evaluación a mecánicas de juego

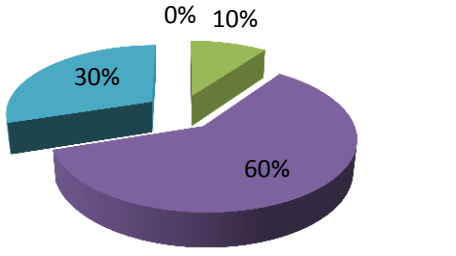
1. Encontré que el hardware relacionado con la capacidad gráfica del equipo fue de gran calidad



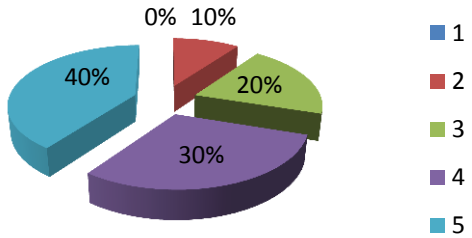
2.Encontré que el contenido visual del ambiente del juego era de gran calidad



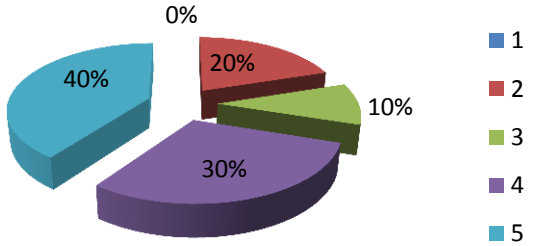
3.Tuve una reacción emocional al realizar tareas dentro del juego

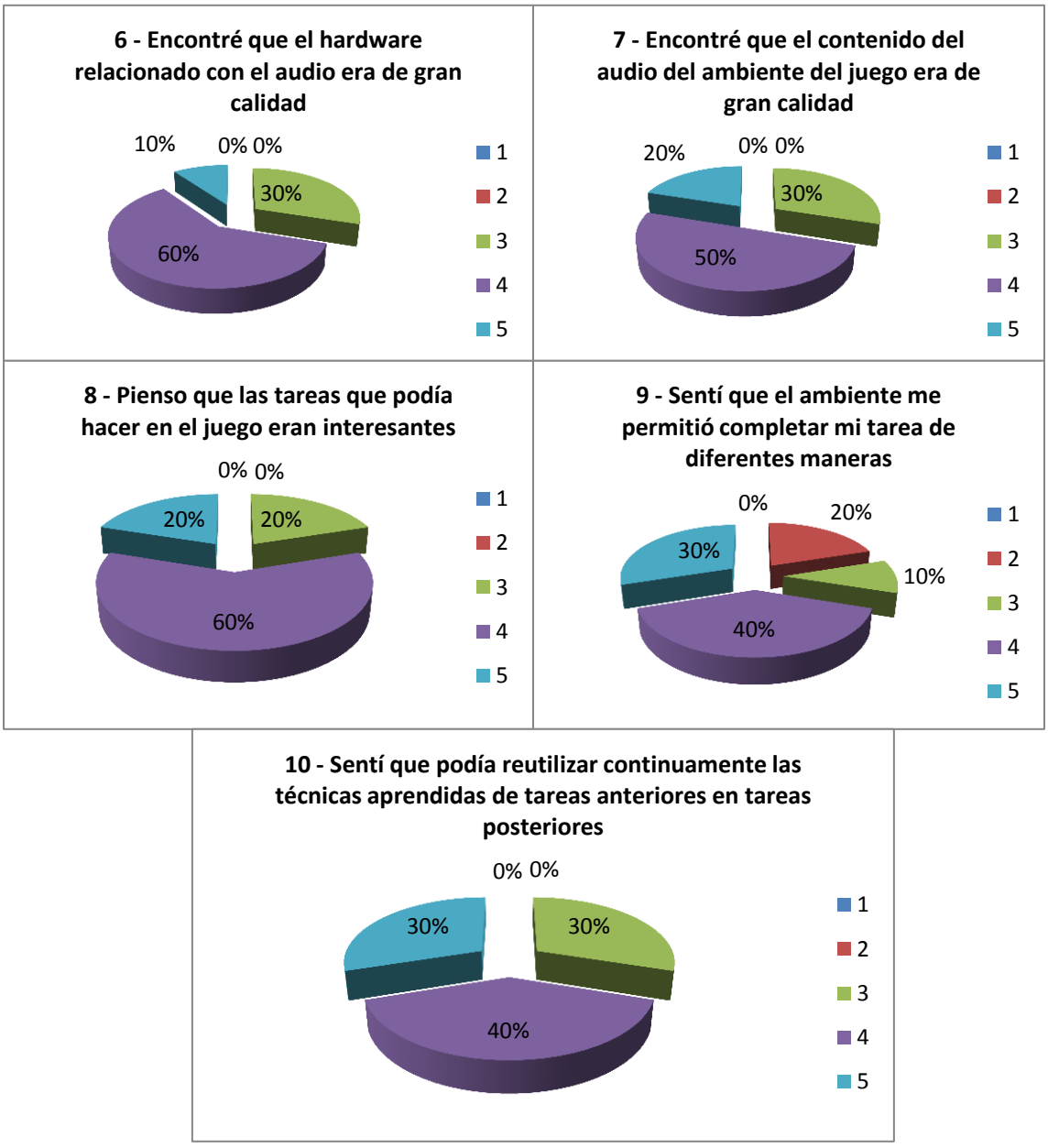


4.Encontré que el contenido en el ambiente del juego fue útil en informarme de mi tarea actual



5. Pienso que el ambiente del juego me dejó claro lo que debía y no debía hacer




6 - Encontré que el hardware relacionado con el audio era de gran calidad
10%
0% 0%
30%
60%
1
2
3
4
5
7 - Encontré que el contenido del audio del ambiente del juego era de gran calidad
20%
0% 0%
30%
50%
1
2
3
4
5
8 - Pienso que las tareas que podía hacer en el juego eran interesantes
0% 0%
20%
20%
60%
1
2
3
4
5
9 - Sentí que el ambiente me permitió completar mi tarea de diferentes maneras
0%
20%
30%
10%
40%
1
2
3
4
5
10 - Sentí que podía reutilizar continuamente las técnicas aprendidas de tareas anteriores en tareas posteriores
0% 0%
30%
30%
40%
1
2
3
4
5