Universidad de las Ciencias Informáticas

Facultad 5

Título: Simulación de pecera virtual como herramienta para las terapias de relajación

Trabajo de Diploma para optar por el título de
Ingeniero Informático

Autor(es): Alexander Navarro Hernández

Tutor(es): MsC. Omar Correa Madrigal

Co-tutor: Ing. Roberto Elías Pérez Ozete

Consultante: Ronniel Arencibia Tejeda

Asesor: Dr. Julio Zamarreño Hernández

Junio, 2014

"Nosotros no solo somos responsables de lo que hacemos, sino también de lo que no hacemos"

Jean Baptiste Poquelin "Molière"

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al centro Vertex de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Alexander Navarro Hernández

_____

Omar Correa Madrigal

_____

Roberto Elías Pérez Ozete

_____

DATOS DE CONTACTO

Generales del Tutor.

Nombre y Apellidos: Omar Correa Madrigal.

Especialidad: Máster en Informática Aplicada.

Síntesis del tutor:

- Actual director del centro Vertex de la Universidad de las Ciencias Informáticas.
- Categoría docente: Instructor.
- 9 años de experiencia.

Correo Electrónico: ocorrea@uci.cu.

Generales del Co-tutor.

Nombre y Apellidos: Roberto Elías Pérez Ozete.

Especialidad: Ingeniero en Ciencias Informáticas

Síntesis del co-tutor:

- Trabajador del centro Vertex de la Universidad de las Ciencias Informáticas.
- Categoría docente: Recién graduado.
- 1 año de experiencia.

Correo Electrónico: reperez@uci.cu.

AGRADECIMIENTOS

Primeramente agradecer a mi madre por luchar cada segundo de su vida para que yo sea alguien sin esperar otra cosa en el mundo que mi amor en correspondencia.

A mi familia por todo el apoyo que me han dado, en especial a mi abuelo Rolando (EPD) quien fue mi superhéroe y siempre lo llevaré en el corazón.

A mi tutor y cotutor quienes confiaron y confían en mí como profesional y persona, gracias por su paciencia.

A todos mis amigos que sin ellos no sería quien soy en estos momentos.

DEDICATORIA

A mi madre que se gradúa junto conmigo, a quien le debo todo en la vida, sirva este trabajo para demostrar que no le fallé.

RESUMEN

La investigación que se presenta se orientó en desarrollar una aplicación que simule un entorno de pecera tridimensional y adaptable en general a las terapias de relajación por su sencilla configuración. Producto de las características del sistema y las variaciones en su realización se decidió utilizar como metodología de desarrollo de software *Xtreme Programming* (XP) y como herramienta principal el motor de videojuegos *Unity 3D*, la cual satisface las necesidades que permiten satisfacer los objetivos.

Como resultado se obtuvo un sistema de simulación adaptable a las etapas del proceso de relajación terapéutica, contando con las funcionalidades requeridas para dicho proceso, las cuales se pueden configurar permitiendo trazar estrategias viables en su aplicación práctica. Se demostró la importancia del análisis de las técnicas de *Inteligencia Artificial* para imitar procesos y fenómenos de la naturaleza.

Finalmente se realizó el despliegue de prueba de la aplicación en una institución médica con el objetivo de evaluar los posibles resultados y la efectividad del producto informático en un ambiente práctico obteniéndose resultados satisfactorios y la aceptación de la propuesta por parte de los usuarios finales.

PALABRAS CLAVE

Xtreme Programming, Unity 3D, Inteligencia Artificial, terapias de relajación, entorno de pecera.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En la actualidad, el progreso y desarrollo de las grandes ciudades y las tecnologías, han dado lugar a que las personas se vean expuestas a constantes estados de tensión, vacío y aburrimiento, influyendo negativamente en el comportamiento de estas. Dicha influencia se manifiesta de diversas maneras como cambios de conducta, baja productividad, agresividad, estados crónicos de tensión muscular, estrés, trastornos psicosomáticos: hipertensión arterial, obesidad, úlcera, asma y otros malestares.

Con el paso del tiempo y con el objetivo de evitar que el cansancio y el estrés influyan negativamente en las personas, se han estudiado y desarrollado diversos métodos y técnicas. Las mismas van desde simples ejercicios de respiración, hasta la visita a centros especializados de rehabilitación, donde se aplican técnicas y métodos más complejos. El progreso de la sociedad y la industrialización a gran escala existente en el mundo, han traído consigo la disminución de las áreas naturales, las que tradicionalmente se emplean para la práctica de ejercicios y deportes. A raíz de dicho fenómeno se han desarrollado nuevas herramientas a fin de que las personas no se vean obligadas a dejar su puesto laboral o la comodidad del hogar en aras de disminuir el estrés acumulado con el quehacer cotidiano.

En dicho contexto han surgido diversidad de tratamientos, entre estos los de relajación, que consisten en un profundo descanso físico, psíquico y afectivo, lo cual es inducido por los profesionales mediante métodos elaborados en la asistencia de los pacientes, como lo son los software de relajación.

A nivel mundial se pueden encontrar varios productos de software orientados a dicha actividad los cuales son utilizados como herramientas en las terapias, estos en su mayoría son protectores de pantalla, videojuegos o videos tridimensionales.

El gobierno cubano actualmente aboga por la informatización de la sociedad y los procesos que en ella influyen, prestando especial atención a sectores como: la educación, los servicios, el sector empresarial y la salud entre otros. A fin de lograr el objetivo anteriormente planteado, son llevados

a cabo numerosos proyectos entre los que se encuentra la creación de la Universidad de las Ciencias Informáticas (UCI) en el año 2002.

La UCI ha estado presente desde sus inicios en los mayores logros informáticos del país. Debido a que en Cuba existen pocas empresas que se dediquen al desarrollo de software, la UCI ocupa un lugar importante en la producción de software a nivel nacional.

La UCI está compuesta por varias facultades, dentro de las que existen centros productivos integrados por varios proyectos. Tal es el caso del Centro de Entornos Interactivos 3D VERTEX el cual pertenece a la facultad 5. Uno de los proyectos que conforman dicho centro es “Rehabilitación de Funciones Motoras y Cognitivas” que dentro de las investigaciones que desarrolla, aparecen las relacionadas con las terapias de relajación. Teniendo en cuenta la experiencia obtenida con respecto a las terapias de relajación utilizando entornos de pecera tales como Dream Acuario se han obtenido resultados satisfactorios y se demuestra que tales entornos son capaces de relajar a diversas personas pero son incapaces de ajustarse a la terapia a medida que la misma esté en transcurso ya que poseen configuraciones ubicadas en una misma ventana lo que dificulta el proceso terapéutico interrumpiéndolo si se modifican sus elementos, dichas modificaciones se realizan de forma manual e internamente lo que dificulta su integración con otras herramientas empleadas. Estos detalles destacan la necesidad de simular entornos capaces de adaptarse al proceso terapéutico, por lo que este trabajo de diploma tiene como **problema a resolver:** ¿Cómo simular de manera virtual un entorno de pecera para las terapias de relajación y que sea adaptable a las características de la misma?

Se presenta como **objetivo general** del trabajo de diploma: Desarrollar una simulación virtual de un entorno de pecera adaptable a las características de la terapia para la relajación de pacientes.

El **objeto de estudio** se enmarca en los Sistemas de Realidad Virtual para tratamientos psicológicos.

Se define como **campo de acción,** los Sistemas de Realidad Virtual para las terapias de relajación.

Tareas Investigativas

- Estudio de la bibliografía existente relacionada con el desarrollo de entornos virtuales para terapias de relajación.
- Caracterizar los elementos fundamentales que conforman un entorno virtual de pecera para las terapias de relajación.
- Definir la solución técnica.
- Detección de los principales componentes ingenieriles para la fase de análisis y diseño.
- Implementación de la solución.
- Estudio de usuarios.

Posibles resultados:

Un sistema informático adaptable a diferentes condiciones de terapias de relajación basado en entornos de pecera.

Métodos teóricos

Histórico - Lógico: se empleó para la fundamentación y sistematización de los aspectos teóricos contemplados en el desarrollo de la investigación acerca de terapias de relajación, las metodologías a utilizar, los planes de estudios y demás elementos relacionados con el contenido del trabajo.

Analítico - Sintético: durante el proceso de investigación permitió descubrir los elementos esenciales, concepciones y conceptos en torno al objeto de investigación.

Entrevista o Encuesta: este método permitió establecer una comunicación con los usuarios escogidos para la realización de las pruebas de relajación de la simulación.

Métodos empíricos

Observación: se empleó como método referencial al observar los distintos tipos de software de relajación, que sirvieron como objeto de análisis y comparación para establecer las características y elementos fundamentales que debía cumplir la propuesta que plantea el autor.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

El presente capítulo recoge la base teórica y metodológica sobre el cual se ha sustentado la investigación, define los conceptos básicos y además se realiza un estudio de las tendencias actuales en la utilización de herramientas y tecnologías para el desarrollo de entornos virtuales para terapias de relajación, y se referencian las utilizadas para realizar la solución.

1.1. Relajación. Conceptos fundamentales

Estrés es el estado de gran tensión nerviosa, generalmente causado por un exceso de trabajo, que suele provocar diversos trastornos físicos y mentales. Situación en la que un organismo o alguno de sus órganos sufren presiones del medio o exigencias superiores a lo habitual, por lo que puede llegar a enfermar (Hierrezuelo, 2004). Una forma de atacar este padecimiento se centra en las técnicas de relajación.

El término **relajación** se entiende como el estado breve o largo de actividad metabólica, nerviosa y consciente reducida, que se puede medir y definir en el plano subjetivo, fisiológico y motor (Complementaria, 2000). También se puede definir a la relajación como la disminución o la desaparición de la tensión **(estrés)**. De tal modo y en diversos casos, elementos o situaciones el término relajación es permisible, siempre y cuando desaparezca la tensión. Esto puede ser entendido tanto a nivel físico como corporal, emocional, psicológico, político, etc.

Uno de los usos más aplicados del término relajación es el relacionado con la entrada en un estadío de comodidad el cual es medible en diversos planos tales como:

Subjetivo: su caracterización es una experiencia consciente de los estados afectivos, por ejemplo: la sensación de placer versus displacer, bienestar versus malestar, alegría versus tristeza. (Complementaria, 2000)

Fisiológico: se caracteriza por cambios viscerales, somáticos y corticales como cambios en el ritmo cardiaco, disminución de la tensión muscular, cambios de los ritmos electroencefalográficos, temperatura interna, etc. (Complementaria, 2000)

Motor: se caracteriza por acciones externas observables de la persona, por ejemplo: inactividad versus hiperactividad, relajación versus tensión de las expresiones corporales y faciales. (Complementaria, 2000)

Una de las formas que más recursos brinda para la medición de la relajación son las variables asociadas al plano fisiológico y dentro de este, uno de los elementos más importantes son las pulsaciones **(ritmo cardiaco)**. Según experiencias en centros donde se aplican terapias de relajación tales como el Centro Memorial Dr. Martin Luther King Jr. y el centro de rehabilitación nacional Julio Díaz existen dos formas de medir los niveles de relajación basándose en el elemento anteriormente mencionado. En caso de que la terapia fuese dirigida a un solo paciente (Caso 1), si el número de pulsaciones finales fuese menor o igual que el número de pulsaciones iniciales, el nivel de relajación es considerado **aceptable,** en caso contrario el nivel de relajación es considerado **no aceptable.** En caso de que la terapia sea dirigida a un grupo de pacientes (Caso 2), si el porcentaje de las personas en las cuales disminuye o se mantiene el número de pulsaciones finales con respecto a las iniciales es mayor que el porcentaje de las personas en las que el número de pulsaciones finales son altas con respecto a las iniciales entonces el nivel de relajación es **aceptable** en caso contrario es considerado **no aceptable.** Las siguientes fórmulas muestran lo anteriormente dicho:

Caso 1: un paciente:

$$NR = \begin{cases} si\ NPF \leq NPI \rightarrow aceptable \\ caso\ contrario \rightarrow no\ aceptable \end{cases}$$

Donde NPI es el Número de Pulsaciones Iniciales, NPF es el número de Pulsaciones Finales y NR se define como Nivel de Relajación.

Caso 2: varios pacientes:

$$NR = \begin{cases} si(PPD + PPM) > PPA \ \rightarrow \ aceptable \\ si\ (PPD\ + PPM) \ \leq \ PPA \ \rightarrow \ no\ aceptable \end{cases}$$

Donde PPA es el Promedio de Pulsaciones que Aumentan, PPD es el Promedio de Pulsaciones que Disminuyen, PPM es el Promedio de Pulsaciones que se Mantienen y NR se define como Nivel de Relajación.

1.1.1 Tipos de terapias de relajación

Existen varias clases de terapias de relajación y cada una utiliza una gran variedad de técnicas, sin embargo la mayoría de estas comparten ciertas características relacionadas. En muchos de los métodos de este tipo, la persona opta por recostarse o adoptar una postura que permita la rápida relajación de los músculos, ubicándose en un lugar tranquilo y con los ojos cerrados. Los siguientes pasos difieren un tanto, dependiendo de la técnica a utilizar. En el **entrenamiento autogénico**, **respuesta a la relajación** y en diferentes formas de meditación, la persona enfoca su mente hacia las sensaciones internas, como la respiración. Las variantes de imágenes dirigidas, emplean una visualización deliberada de escenarios o acciones, como caminar por una playa tranquila. Las técnicas de relajación progresiva en cambio, implican una disminución gradual de la tensión en los músculos. A continuación se exponen dos de los diversos tipos de terapias de relajación:

Relajación Autoalusiva: se centra en unificar todos los elementos que conforman la experiencia presente de forma simultánea, todos los niveles de la realidad presente se unifican en el ejercicio de relajación. (Barrios, 2013)

La relajación autoalusiva es una relajación en busca de la unidad, es una técnica inclusiva que pretende tomemos conciencia de todo aquello que sucede a la vez integrándolo en una realidad mayor y generando así una nueva realidad que podrá a su vez ser integrada de nuevo, y así sucesivamente.

Relajación Paradójica: pretende la comprensión de uno mismo utilizando como principal herramienta la observación. La observación es un proceso, un cambio en la parte que genera una

reestructuración de la totalidad y ese cambio total permite una nueva exploración de una nueva realidad potencial. (Barrios, 2013)

1.2 Software de relajación.

La palabra *software* proveniente del idioma inglés y su uso se ha masificado gracias al desarrollo de las TIC, y ha sido aceptada por la Real Academia de la Lengua Española (RAE). Según ésta, el *software* es un conjunto de programas, instrucciones y reglas informáticas que permiten ejecutar distintas tareas en una computadora. Es considerado que el software es el equipamiento lógico e intangible de un ordenador. Para generalizar, su concepto abarca todo el ámbito de trabajo con ordenadores, desde los sistemas operativos, hasta los procesadores de texto más básicos como el Bloc de Notas (Notepad). (Española, 2014)

Después del análisis realizado el autor define el concepto de **software de relajación** como las aplicaciones informáticas que constituyen herramientas para facilitar el trabajo en los procesos psicoterapéuticos dedicados a lograr altos niveles de relajación.

1.2.1 Tipos de software de relajación

Existen varios tipos de software de relajación que abarcan desde los protectores de pantalla hasta videojuegos, ejemplo de ello son los siguientes:

Refrescadores de Pantalla

Existe una inmensa cantidad de estos hasta el momento, simulando diversas locaciones que van desde entornos sencillos como los de pecera hasta algunos mucho más complejos como los que simulan locaciones enteras; playas, entornos selváticos, saltos de agua e incluyendo ambientes fantásticos que simulen lugares donde coexistan la paz y la tranquilidad. Algunos de ellos son:

Dream Aquarium: es un acuario virtual y protector de pantalla que simula un acuario de agua dulce con un alto nivel de realismo. Muestra el comportamiento natural y el movimiento de sus habitantes. Estas son algunas de sus características: Aletas articuladas, movimiento de los ojos, branquias y boca, rayos de luz, ondulaciones de la tierra, movimiento suave de las plantas por los cuales los

peces pueden nadar, sombras suaves, secuencias de burbuja configurables, alimentación automática. Añade más de un centenar de peces y cambia la configuración sin tener que salir del acuario. (Kapler, 2013)

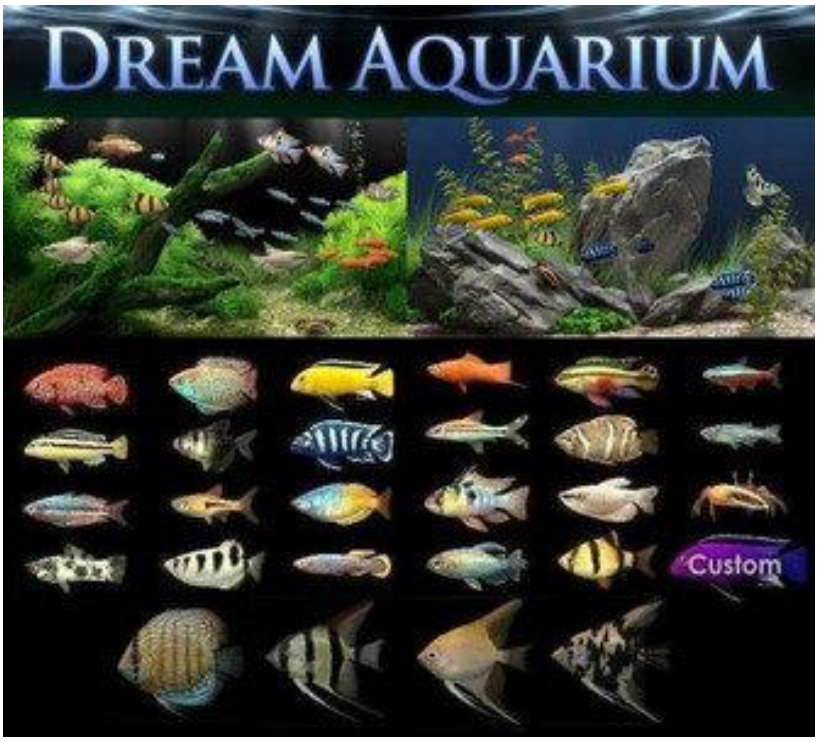

Figura 1 Dream Aquarium

Fantastic Ocean 3D: consiste en un tranquilo vuelo sobre el océano, cuidadosamente recreado en escenas 3D. La cámara está dispuesta como si se estuviera planeando a muchos metros de las olas. El sol poniente se refleja en el océano con cientos de gamas de colores diferentes. Posee una interfaz amena, la cual cambia cada cierto período tiempo. (Team, 2013)

Figura 2 Fantastic Ocean 3D

Algunos tipos de software solo son compendios o bancos de sonido que combinan estos para lograr un efecto tranquilizador y pausado, logrando así el estado de relajación. Como ejemplo existe:

Pzizz: es capaz de generar varias de estas combinaciones. Su biblioteca de secuencias musicales y sonidos se divide en tres categorías: energética, meditación y sueño. Se pueden crear secuencias propias modificando el volumen, la longitud y otros parámetros. Así, por ejemplo, es posible crear una pieza relajante para una siesta de veinte minutos. Las instrucciones vocales están en inglés, pero su tono es suficiente para inducir el estado deseado. Es capaz de exportar la pieza a varios formatos compatibles con reproductores de música. (Ashenden, et al., 2013).

Figura 3 Pzizz

Halotea: La generación de los temas de sonido es única en la aplicación, sobre la base de repeticiones al azar con parámetros aleatorios de la reproducción de los sonidos individuales de alta calidad. Esto permite la creación de un sonido realista y único. Todos los *presets* están divididos en grupos, de modo que es fácil de encontrar presets requeridos, así como añadir fácilmente con el tema de sonido elegido. (ibaiondo, 2014)

Figura 4 Halotea

El mundo de los videojuegos no ha estado ajeno a los procesos terapéuticos, algunos son solo sencillos juegos flash en los cuales se interactúa solo para acceder a alguna biblioteca de efectos sonoros o visuales que ayudan en el proceso de la relajación, entre ellos:

Subtle Energy: brinda información acerca de los chakras y lo conocido como energías sutiles. (Meloleo, 2014)

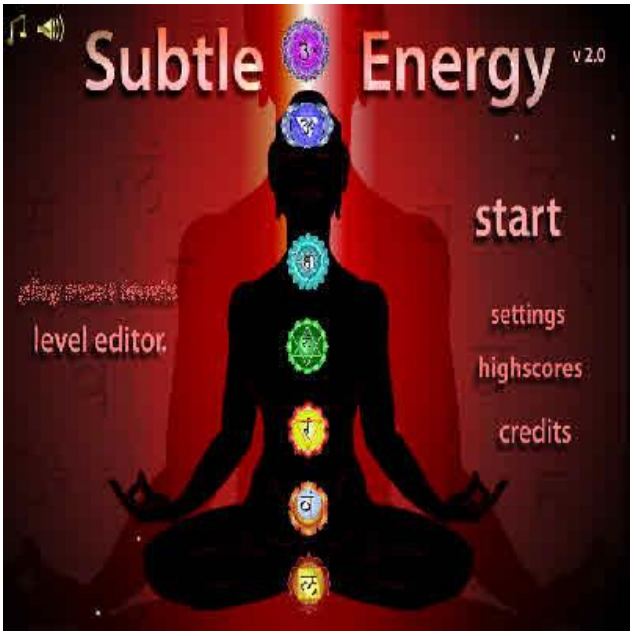

Figura 5 Subtle Energy

Symphony: juego desarrollado en plataforma flash, en el cual la persona puede crear sus propias melodías y escucharlas, así como escuchar algunos sonidos antiestrés que contiene. (Meloleo, 2014)

Figura 6 Symphony

Waterfalls: es un juego desarrollado en plataforma flash en el cual se conduce la Aurora Boreal hacia la dirección de los planetas para llenarlos de maná y disfrutar de algunos efectos conocidos como mantras. (Meloleo, 2014)

Figura 7 Waterfalls

Proteus: no es un juego, más bien es un software audiovisual. Un programa de relajación interactiva, que sumerge al usuario en una isla de colores y píxeles gordos sin objetivo concreto. En *Proteus* la persona se encuentra ubicada en una isla desierta. Esta isla se genera automáticamente en cada nueva partida y no existen dos islas iguales. Tampoco se puede hacer pausa ni grabar la partida. La variedad de elementos y sonidos extraños hacen que se despierte la curiosidad de moverse por el entorno. Necesariamente no tiene que terminarse el juego para pasar un momento agradable solo con caminar por la isla un tiempo determinado cada día permite que se llegue a eliminar el estrés. (Key & Kanaga, 2012)

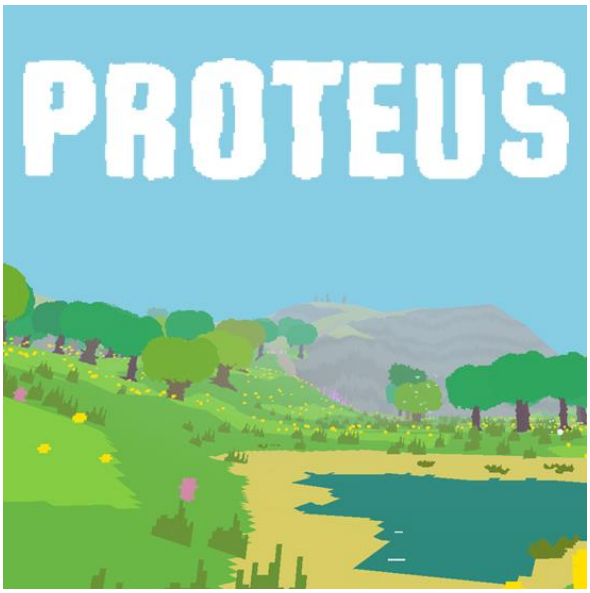

Figura 8 Proteus

1.3. Simulación

Puede definirse a la simulación como la experimentación con un modelo que imita ciertos aspectos de la realidad. Esto permite trabajar en condiciones similares a las reales, pero con variables controladas y en un entorno que se asemeja al real pero que está creado o acondicionado artificialmente. (Cabrera, 2012)

El objetivo es que la simulación permita comprobar el comportamiento de una persona, de un objeto o de un sistema en ciertos contextos que, si bien no son idénticos a los reales, ofrecen el mayor parecido posible. Así, es posible corregir fallos antes de que la experiencia, efectivamente, se concrete en el plano de lo real.

La simulación se puede considerar también como una herramienta de análisis que permite sacar conclusiones sin la necesidad de trabajar directamente con el sistema real que se está simulando. Esta es especialmente útil cuando no se dispone de dicho sistema real o resulta demasiado arriesgado realizar experimentos con él.

En general, se podría considerar que muchos juegos de ordenadores son simuladores ya que recrean parte de la realidad. Sin embargo lo que marca la diferencia es su intención, en el caso de los juegos es el ocio y en la simulación es su análisis.

En muchas áreas de la ingeniería se utilizan los simuladores como una herramienta de trabajo más. Por ejemplo, en el diseño de nuevos fármacos se suelen utilizar modelos moleculares que sirven para simular por medio de computadoras la interacción de compuestos químicos. Los ingenieros de automóviles también utilizan modelos computarizados para analizar el impacto de los choques en la seguridad de los viajeros.

1.3.1. Tipos de simulación existentes

Simulación, según la Real Academia española, se conceptualiza como la representación de algo, fingiendo o imitando lo que no es. Según el Handbook of Simulation (1998), es una imitación de las operaciones de un sistema o proceso real a lo largo del tiempo, esto último en sistemas más complejos. Involucra la generación de una historia artificial del comportamiento del sistema y a partir de la historia se efectúan inferencias relativas a las características operacionales del sistema real que representa. También permite describir y analizar el sistema real que representa. (Cabrera, 2012)

La simulación puede llegar a convertirse en la metodología ideal para resolver muchos problemas reales, ayudando en la toma de decisiones, apoyando en el diseño de sistemas mucho antes de que estos sean construidos e incluso probando políticas de funcionamiento antes de que estas sean implantadas. Aunque esta no resuelve los problemas por sí misma, sí ayuda a identificar los problemas relevantes y a evaluar cuantitativamente las soluciones alternativas. Dentro de los tipos de simulaciones podemos encontrar:

Simulación Predictiva: en ésta, se interesan por los resultados absolutos finales, no por las comparaciones. Determinan promedios e intervalos de confianza de una corrida de simulación con valores específicos en las variables de decisión (varias corridas, mejores resultados).

Este tipo de simulación se puede utilizar para realizar pronósticos, por lo que es necesario contar con datos históricos de entrada confiables, se utiliza en procesos de decisiones que se repiten. Ejemplo: predecir el número de pacientes que necesitan trasplante de riñón). (Cabrera, 2012)

Simulación Comparativa: En la simulación comparativa *se* determina cuando una opción es mejor que otra. Se debe especificar detalladamente qué significado tiene la palabra "mejor", para definir cuáles serán los datos de salida a comparar. Mejor significa disminuir el tiempo de espera o es un compromiso entre tiempo de servicio, largo de cola y costo por servidor. Se puede usar para tomar decisiones casuales o repetitivas, utilizar datos de entrada y salidas confiables.

Si los objetivos no son claros, se proveerá de un rango variado de resultados, que le permitan al usuario definir a posteriori la importancia relativa de cada uno de ellos. Si los resultados o los datos de salida son claros se puede usar técnicas de hipótesis estadística de los resultados. (Cabrera, 2012)

Simulación Investigativa: La simulación investigativa indica factores que afectan el flujo de entidades en el sistema pero no requiere de respuestas precisas, por lo que la calidad de los datos de entrada no es crítico. (Cabrera, 2012)

Simulación visual interactiva: La técnica de simulación visual interactiva es adecuada para apoyar la toma de decisiones. (Cabrera, 2012)

Simulación de caja negra: Pensando al modelo como parte de un proceso de toma de decisiones, es conveniente, a veces, considerar el modelo como una **caja negra**, de donde salen flechas con datos, derivados directamente de los objetivos (y que difieren de un problema a otro) y a donde ingresan flechas con datos relacionados estrechamente con las hipótesis de trabajo del modelo. (Cabrera, 2012)

La solución planteada se encuentra enmarcada dentro de las simulaciones del tipo visual interactiva. Las mismas toman como elemento primordial la observación y la interacción con el entorno, rasgos que son fundamentales en dicha solución.

1.3.2 Características fundamentales de la simulación de entornos 3D

Son entornos que pueden asemejarse a entornos reales o pueden ser total o parcialmente ficticios en los cuales intervienen diferentes factores ambientales. En los últimos años, los avances en hardware y software de ordenador han sentado las bases para el diseño de entornos virtuales (EV) con más calidad y fidelidad. Estas mejoras en la tecnología de los mismos han revivido el interés en el uso de los escenarios virtuales para brindar conocimiento. Hay muchas ventajas en la utilización de los mundos virtuales, por ejemplo; ofrece un ambiente de formación flexible y rentable que se puede volver a configurar rápida y fácilmente para proporcionar la formación específica para la misión. También ofrece a los instructores la oportunidad de exponer a los estudiantes a situaciones que de otro modo serían imposibles (es decir, potencialmente mortal) para recrear en escenarios de formación de la vida real. Además, ofrece una oportunidad única para que los entrenadores puedan evaluar a sus alumnos, ya sea en tiempo real, mediante la congelación de la formación en los puntos críticos, o mediante la reproducción de todo el escenario de la formación sobre la terminación. (Lathan, 2002)

Los mundos virtuales tienen su origen en la simulación militar y en concreto en los simuladores de vuelo, donde el principal problema consiste en extraer de la base de datos visual (presumiblemente grande) el mundo visible en cada instante en función de la posición del observador o cámara virtual, en el escenario simulado. Tiempo después, con la comercialización de esta tecnología para uso civil surgió el concepto de lo que se conoce como Realidad Virtual que todavía hoy perdura y gana terreno en muchos ámbitos de la sociedad, por ejemplo los gráficos 3D en entornos inmersivos que usan artefactos como guantes, cascos, etc. en busca de mayores grados de interacción con el ambiente virtual.

La utilización de entornos virtuales comienza a desarrollarse en la enseñanza a distancia. Los entornos virtuales han sido utilizados como espacios de intercambio de información, asociados a una serie de recursos telemáticos ideados para el estudio a distancia y para facilitar la comunicación entre el profesor y su grupo de estudiantes.

El salto hacia entornos tridimensionales ha supuesto un gran avance y ha permitido el conocimiento de mundos pasados, presentes o incluso futuros; permite caminar por el interior de edificios que dejaron de existir hace siglos, interactuar con objetos de maneras imposibles en el mundo real, o estudiar objetos demasiado frágiles o de difícil acceso; ayudando a su vez a la conservación del mismo. Permite ensayar técnicas de restauración sobre modelos sintéticos, así como explorar diferentes teorías sobre su construcción, todo ello teniendo en cuenta no dañar el original.

La infografía 3D ha supuesto un importante perfeccionamiento en la reproducción de la realidad. La simulación es un proceso que nos permite estudiar un sistema físico sustituyéndolo por otro más fácilmente observable o medible. Mediante la simulación pueden abarcarse diversos intereses: un astronauta puede aprender a pilotar una lanzadera espacial, un gobierno puede prever la evolución de la población en su país, un meteorólogo puede predecir tormentas y un arqueólogo puede reconstruir una ciudad perdida a partir de sus restos. Estos enfoques requieren diferentes niveles de complejidad en la simulación: así, el astronauta necesitará sentir la dinámica del sistema para poder percibir aceleraciones, mientras que el arqueólogo puede conformarse con unos gráficos 3D que muestren su ciudad con un elevado nivel de realismo. (Lozano & Calderón, 2004)

Los entornos virtuales tridimensionales poseen características básicas. Poseen un **espacio compartido** donde participan muchos usuarios simultáneamente; cuentan con una **interfaz gráfica** donde se encuentran ambientes tridimensionales inmersivos; los caracterizan la **inmediatez**, la **interactividad** y la **persistencia**, permitiendo que la interacción tenga lugar en tiempo real y que los usuarios puedan crear, modificar y poseer contenidos virtuales, además de que el entorno virtual persiste aún en ausencia de sus creadores. También contribuyen a la **formación de comunidades sociales** donde el conocimiento se comparte.

Entorno adaptable: Se considera que es un entorno adaptable aquel que posee características configurables las cuales permiten el manejo del mismo en diferentes situaciones a través de la práctica.

Efecto de cáustica: Desde el punto de vista de la física, la cáustica se define como el efecto visible de la refracción o reflexión de los rayos de luz por un sistema óptico (Española, 2014). Para mayor entendimiento la siguiente imagen muestra tal efecto:

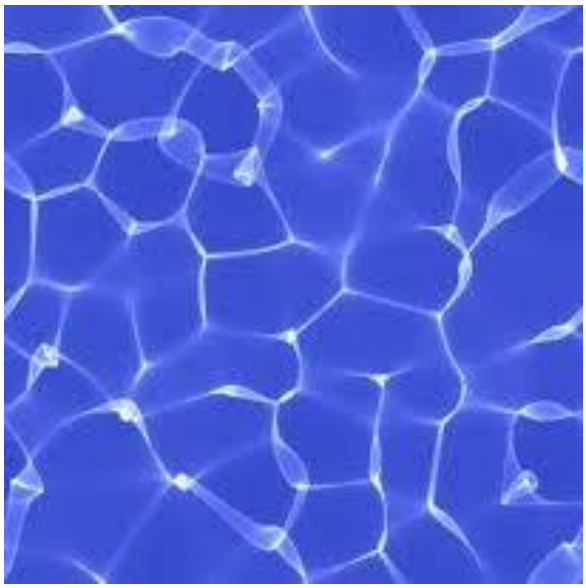

Figura 9 Efecto de cáustica en el agua

1.4 Metodología de desarrollo de software

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte tienen aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán.

Las propuestas existentes han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros muchos. Una posible mejora es incluir en los procesos de desarrollo más actividades, más artefactos y más restricciones, basándose en los puntos débiles detectados. Sin embargo, el resultado final sería un proceso de desarrollo más complejo que puede incluso limitar la propia habilidad del equipo para llevar a cabo el proyecto. Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el

producto software. Esta es la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

Las metodologías ágiles están revolucionando la manera de producir software, y a la vez generando un amplio debate entre sus seguidores y quienes por escepticismo o convencimiento no las ven como alternativa para las metodologías tradicionales. (Penadés, 2009)

Antes de resumir algunas de las metodologías ágiles más utilizadas, a continuación se enumeran las principales diferencias respecto de las metodologías tradicionales (“no ágiles”). La Tabla 1 (Tellez, 2012) recoge esquemáticamente estas diferencias que no se refieren sólo al proceso en sí, sino también al contexto de equipo y organización que es más favorable a cada uno de estas filosofías de procesos de desarrollo de software.

Tabla 1 Diferencias entre metodologías ágiles y no ágiles

Metodologías Agiles	**Metodologías Tradicionales**
Basadas en Heurísticas provenientes de prácticas de producción de código. Especialmente preparados para cambios durante el proyecto.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo. Cierta resistencia a los cambios.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas y normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.

Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos roles.	Más roles.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

A continuación se resumen varias de estas metodologías ágiles:

SCRUM: Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días. El resultado de cada *sprint* es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto. Éstas son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. (Tellez, 2012)

Crystal Methodologies: Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo (de ellas depende el éxito del proyecto) y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn. El desarrollo de software se considera un juego operativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros). (Tellez, 2012)

Extreme Programming (XP): Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en

equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. (Tellez, 2012)

La Tabla 2 (Téllez, 2012) compara las distintas aproximaciones ágiles en base a tres parámetros: vista del sistema como algo cambiante, tener en cuenta la colaboración entre los miembros del equipo y características más específicas de la propia metodología como son simplicidad, excelencia técnica, resultados, adaptabilidad, etc. También incorpora como referencia no ágil el Capability Madurity Model (CMM).

Tabla 2 Posicionamiento de “agilidad” (Los valores más altos representan una mayor agilidad)

	CMM	**Crystal**	**SCRUM**	**XP**
Sistema algo cambiante	1	4	5	5
Colaboración	2	5	5	5
Características Metodología (CM)				
Resultados	2	5	5	5
Simplicidad	1	4	5	5
Adaptabilidad	2	5	4	3
Excelencia técnica	4	3	3	4
Prácticas de colaboración	2	5	4	5
Media CM	2.2	4.4	4.2	4.4
Media Total	1.7	4.5	4.7	4.8

No existe una metodología universal para hacer frente con éxito a cualquier proyecto de desarrollo de software. Toda metodología debe ser adaptada al contexto del proyecto (recursos técnicos y humanos, tiempo de desarrollo, tipo de sistema, etc. (Tellez, 2012)

Se propone utilizar la metodología XP porque la misma simplifica los procesos a través de la reducción de la irreversibilidad. Además ha probado ser de utilidad en proyectos pequeños y con requerimientos altamente cambiantes. También XP reduce el tiempo entre una idea, su criterio de validación y su implementación.

1.5 Herramientas utilizadas

La selección de las herramientas es un paso importante en el desarrollo de software y debe hacerse teniendo en cuenta las ventajas que proporciona cada una de ellas.

1.5.1 Gráficos 2D

Adobe Photoshop

El software Adobe Photoshop CS6 permite la edición de imágenes de gran calidad, nuevas opciones creativas para lograr un rendimiento rápido, tales como nuevos filtros, optimizaciones en sus funcionalidades y mejoras en sus herramientas. Retoca con las nuevas funciones basadas en el contenido y crea diseños y películas excelentes mediante los nuevos flujos de trabajo y herramientas rediseñadas. Este software es utilizado en la creación de texturas, mapas y elementos relacionados con el diseño de los componentes de la simulación.

1.5.2 Gráficos 3D

Autodesk 3ds Max

Autodesk 3ds Max y Autodesk 3ds Max Design proporcionan potentes herramientas integradas de modelado, animación y renderización en 3D que permiten a los artistas y los diseñadores dedicar más energía a la creatividad en lugar de a las dificultades técnicas. Aunque ambos productos comparten la tecnología principal, uno ofrece herramientas especializadas a los desarrolladores de juegos, creadores de efectos visuales, diseñadores de gráficos de movimiento y otros profesionales

de la creatividad que trabajan en el diseño de medios, mientras que el otro está concebido específicamente para los arquitectos, diseñadores, ingenieros y especialistas en visualización. Ha sido seleccionado para ocuparse de la parte de la creación de componentes necesarios para la simulación.

1.5.3 Motor de simulación

Un **motor gráfico** es un conjunto de programas que enlazados forman una rutina, por ello a veces solo dicen que es una rutina de programas, que permiten la creación de un vídeo juego. En un principio un programador no requería de grandes herramientas para realizar un vídeo juego, como ocurre con el famoso juego de tenis que se jugaba en una pantalla monocromática o en un televisor. Sin embargo al aumentar el poder de los procesadores gráficos se fueron haciendo necesarias nuevas herramientas no solo para crear los vídeo juegos si no para poder correrlos.

Es aquí donde empiezan los motores gráficos. El término parece que nace de la comparación con un motor de automóvil, se dice que la carrocería es todo lo que vemos en el vídeo juego mientras que el motor es lo que hace funcionar a todo. Hace casi unos veinte años (en la década de los 90) el mercado de los vídeojuegos entra en el mundo de tres dimensiones (el famoso 3D) y con ellos nacen los motores de vídeo juegos o motores gráficos. Al parecer fue la compañía Origin System la que generó el primero de ellos para el juego Ultima Underworld (al motor se le conoce como el motor de Ultima Underworld y se hizo costumbre darles nombre en base al primer juego que produjeron), luego está el mítico Doom.

Ambos lograban los efectos en 3D utilizando figuras “planas” que se colocaban virtualmente un poco lejos unas de otras dando la sensación de profundidad. En el caso del primero se utilizaban texturas que daban la sensación de profundidad mientras que en el segundo se usaban planos superpuestos para lograr tal efecto. Voxel fue un emulador de vuelos logrado con un motor más complejo que los anteriores. Se han citado los primeros tres para ejemplificar brevemente. Hoy día la mayoría de los motores utilizan una plataforma o *interface* que se basa en rutinas preestablecidas, las más populares son Direct3D para Windows, Glide API y OpenGL tanto para Linux, Windows y Mac. Tanto Glide como OpenGL son de uso libre y Direct3D es propiedad de

Microsoft. Estas plataformas permiten el renderizado de las figuras. El término viene de *render* en inglés y se puede decir que es "interpretar" o "representar", aunque no tiene en español un equivalente representativo.

Lo que se hace con este renderizado es hacer que el computador forme una figura plana (2D) a partir de una "real" (3D), con esto se ahorra información y la calidad de la imagen es muy alta. Estos también son denominados motores de renderizado y utilizan vectores de posición formando pequeños triángulos que dan la forma. Estos triángulos son calculados por medio de complejas funciones matemáticas que la computadora realiza. Al renderizar se “suaviza” la forma y la textura parece plana y no formada por los triángulos.

Esto añadido a los motores gráficos generan efectos realmente espectaculares como los que utilizaron para crear la película “Toy Story” la primera generada totalmente con motores gráficos de última generación, como se puede apreciar, no solo en juegos se aplican los motores gráficos. Esto conlleva a un problema que enfrentan todos los desarrolladores de motores gráficos. Toma mucho tiempo crearlos para luego generar el juego, entonces ocurre que cuando el producto (el juego) está terminado pues ya está obsoleto, de allí que exista una competencia tan feroz en este mercado. Para solventar esto muchos desarrolladores “liberan” los códigos de programación (los famosos códigos fuentes) que permiten generar los motores gráficos para que otros los mejoren y así puedan competir más lealmente. (Anónimo, 2014)

Como se plantea anteriormente los motores gráficos no solo se usan en la creación de videojuegos, tienen muchas más aplicaciones en el campo de la realidad virtual, una de ellas es la de las simulaciones, por lo que se escogió el motor gráfico Unity3D teniendo en cuenta las características mostradas a continuación (Tabla 3).

Tabla 3. Motor gráfico Unity3D

Motores gráficos	**Unity3D**
	Android

Plataformas	iOs Mac Windows Web XBOX LA Wii PS Network
Licencias	*Nombre, precio(en dólares)* Indie, Gratuita Unity Pro, $1500 Android licencia básica, $400 iOs licencia básica, $400 Android Pro, $1500 iOs Pro, $1500
Scripting	C# Javascript Boo
Extensiones o plugins	C# C++
Importación de Assets	Modelos 3D animados (formatos nativos).

	Blender 3DStudio Max Maya Cinema 4D Texturas PNG JPG TGA BMP Sonidos OGG MP3 WAV Videos AVI MPG MP4 OGG

Después de realizada esta comparación se decide usar Unity3D como motor gráfico porque este hace el proceso de producción de juego simple, dándole un set de pasos lógicos para construir cualquier panorama concebible de juego. Establece el uso del concepto *Game Object (GO)*, donde se puede agrupar los elementos del juego en objetos de fácil manejo, que está hecho de muchos componentes.

Haciendo objetos individuales dentro del juego e iniciando funcionabilidad en ellos con cada componente que se sume, se puede expandir el juego en una manera progresiva lógica. Los componentes a su vez tienen variables, por las cuales los mismos serán controlados.

Dentro de sus componentes encontramos:

Assets

Son los bloques constructivos de todos los elementos que Unity posee en sus proyectos. Se guardan en forma de archivos de imagen, modelos del 3D y archivos de sonido, Unity se refiere a los archivos que se usarán para crear su juego como activos.

Game Objects

Cuando un activo es usado en una escena de juego, se convierte en un "Game Object". Todo Game Objects contiene al menos un componente con el que comenzar, es decir, el componente *Transform*. Transformación simple la cual le dice al motor de Unity la posición, rotación, y la escala de un objeto.

Components

Los componentes vienen en formas diversas. Pueden ser para crear comportamiento, definiendo apariencia, e influenciando otros aspectos de la función de un objeto en el juego. Los componentes comunes de producción de juego vienen construidos dentro del Unity, desde el Rigidbody, hasta elementos más simples, como luces, las cámaras, los emisores de partículas, y más.

Scripts

El Scripting es una parte esencial de Unity ya que define el comportamiento del juego. El Scripting es la forma en la que el usuario define el comportamiento del juego (o las normas) en Unity. El lenguaje de programación recomendado para Unity es JavaScript, aunque C# o Boo pueden ser igualmente usados. En Mac, es llamado como Unitron, y en PC, Uniscite.

Prefabs

Almacena los objetos como activos para ser reusado en partes diferentes del juego, y luego creados o copiados en cualquier momento.

Conclusiones del capítulo

En este capítulo se le dio cumplimiento a la primera tarea investigativa trazada en el presente trabajo, la cual sirve de soporte teórico para la investigación. Se abordaron conceptos como: simulación, relajación, Entornos Virtuales, software de relajación, realidad virtual, terapias de relajación, así como una descripción del motor de juegos a utilizar. Además fueron seleccionadas las herramientas y metodología con que se desarrollará la solución del problema, permitiendo las primeras, la agilidad en el diseño y programación de la aplicación.

CAPÍTULO 2: DESCRIPCIÓN DEL SISTEMA

Introducción

El siguiente capítulo está enfocado en la presentación detallada de la solución al problema de investigación. Se realiza una descripción acerca de la estructura y funcionamiento del prototipo elaborado.

2.1 Propuesta de Solución

Teniendo como objetivo la simulación de un entorno de pecera que sea adaptable a las terapias de relajación, se ha dividido el desarrollo de esta en dos partes:

La primera parte consiste en el desarrollo de una simulación donde los elementos que interactúan son los básicos requeridos para el correcto funcionamiento del entorno. Este consta de al menos dos peces y a lo sumo de dieciséis de estos logrando así la interactividad de más de uno de estos con sus similares, es un entorno estático lo cual quiere decir que todos sus elementos siempre se mantienen en la misma posición. Se opta por la mejor configuración de los elementos físicos tales como los efectos visuales, modelos tridimensionales y física del ambiente para permitir la interacción de los peces con el entorno: En esta primera parte también se definen las funcionalidades para la configuración de la simulación tales como agregar o eliminar peces, realizar acercamientos de cámara a estos e incluso alimentarlos.

En una segunda parte se define un archivo de configuración que reúne las funcionalidades anteriormente mencionadas. Dicha configuración se hace efectiva a través de una interfaz de usuario que se construye en una ventana diferente y que escribe en un fichero las respectivas configuraciones hechas por éste.

En la figura 10 se muestra un diagrama que explica cómo quedaría organizado el sistema:

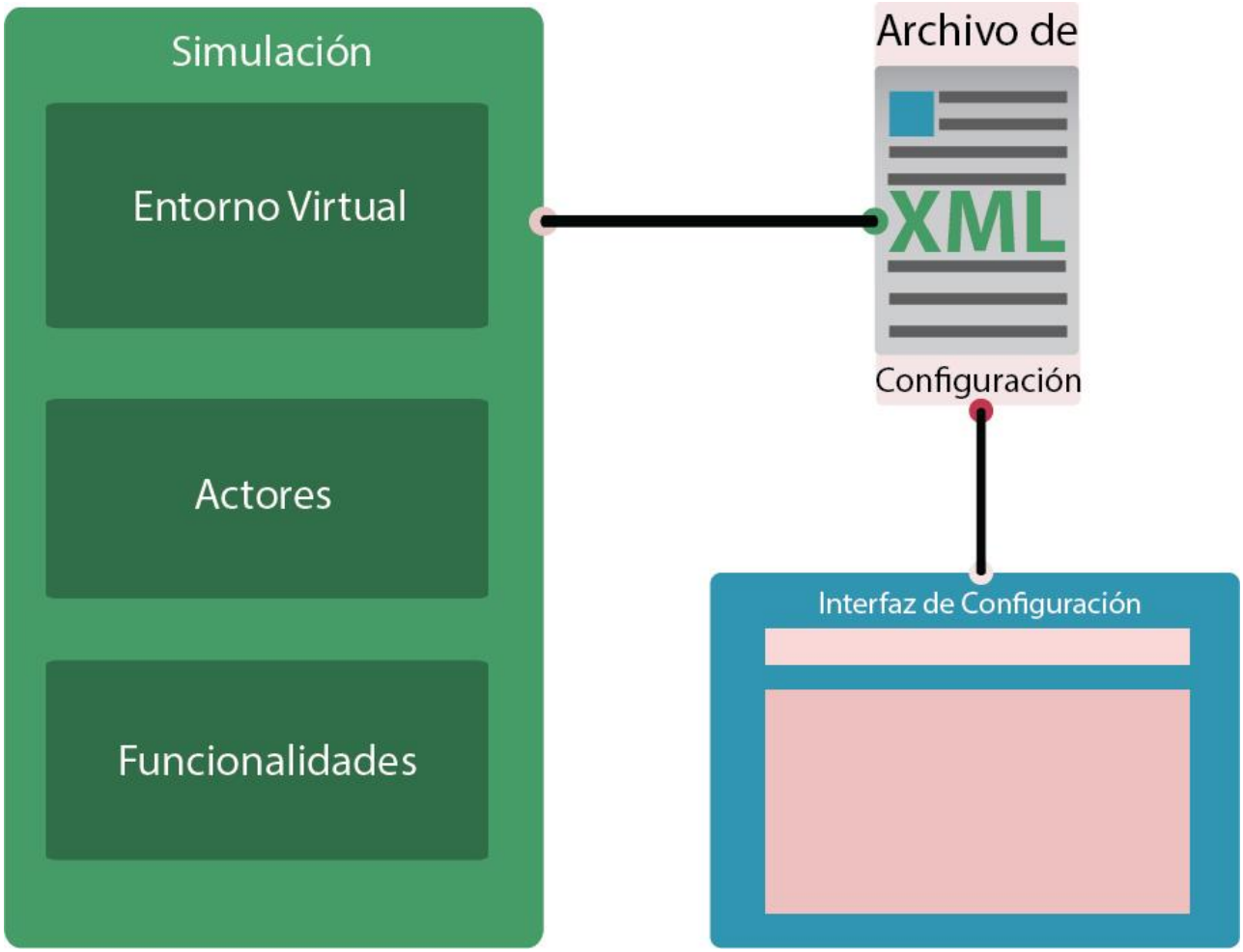

Figura 10 Propuesta de solución

Básicamente, como todo objeto, los peces poseen direcciones, las cuales están asociadas a las componentes X, Y y Z de las coordenadas en el espacio tridimensional. La modificación de dichas direcciones es un elemento importante en la validación de la evasión de obstáculos así como también en la implementación de otras funcionalidades ya que sirven para desviar al pez en caso de encontrar objetos u otros peces en su camino. A continuación se presenta una figura que muestra las direcciones y sus relaciones con las coordenadas:

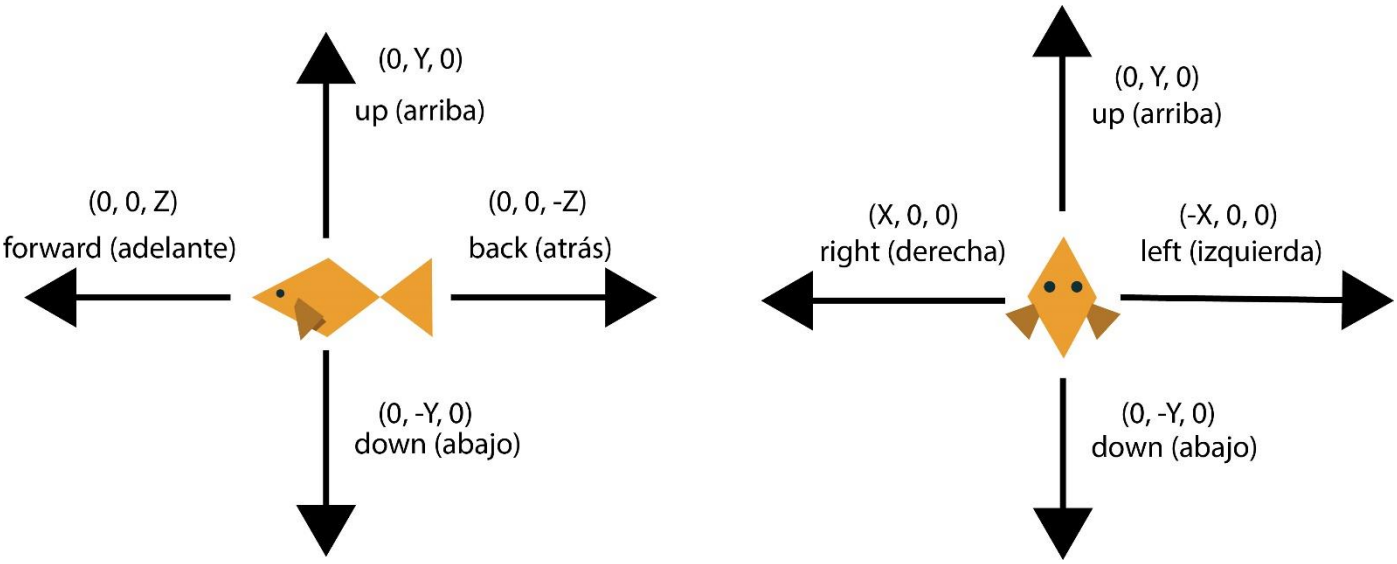

Figura 11 Dirección del pez y su relación con el espacio 3D

El movimiento de los peces se realiza teniendo en cuenta el espacio en el cual se van a desplazar. Los objetivos se definen de forma aleatoria en dicho espacio. Una vez alcanzado un objetivo este se vuelve a recalcular de forma aleatoria y dentro de un volumen definido, generando otro punto el cual será el nuevo objetivo tal y como se representa en el gráfico mostrado a continuación:

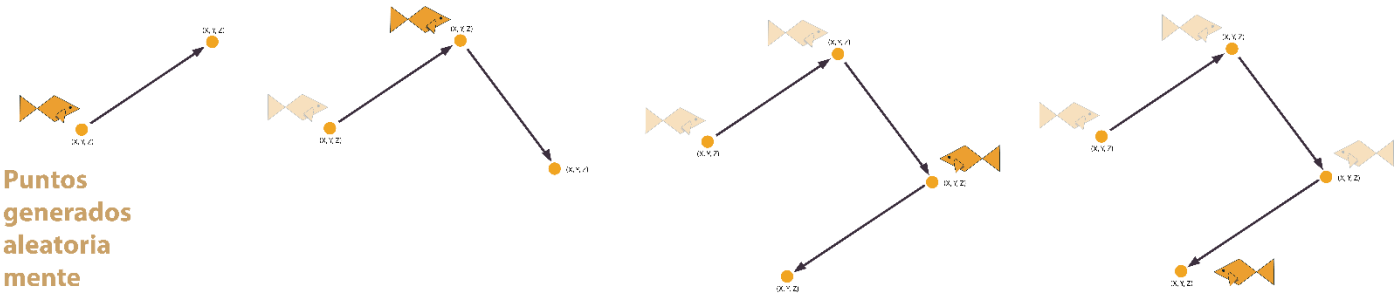

Figura 12 Generación aleatoria de objetivos

Uno de los aspectos fundamentales que debe estar presente durante el desarrollo de la simulación es la evasión de obstáculos. Ante la presencia de éstos, se recalcula el punto donde debe ir el actor modificando la componente Z (forward) a un valor negativo con el objetivo de que el nuevo punto no se ubique detrás del objeto. Dicha evasión está representada fundamentalmente en dos tipos, el primero trata sobre la evasión de objetos inmóviles y se ve representada en el gráfico siguiente:

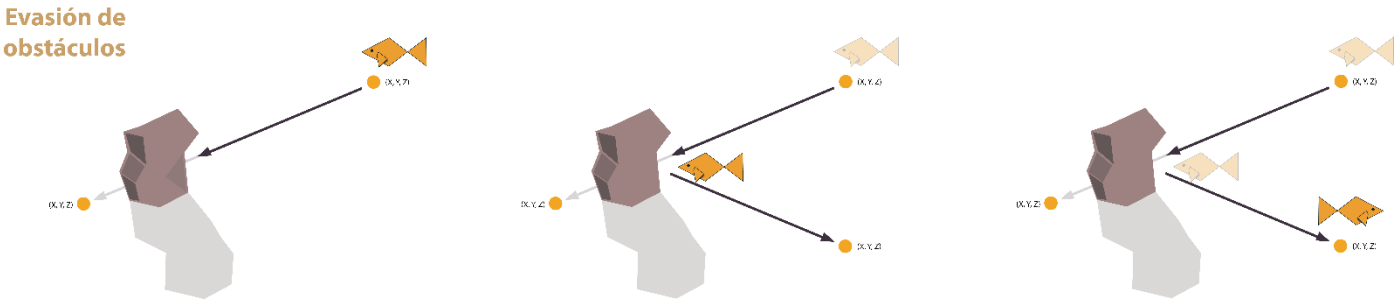

Figura 13 Evasión de obstáculos inmóviles.

El otro tipo de evasión se concentra en los obstáculos móviles, en este caso otros peces. Ante la presencia de otros actores, el punto hacia donde se dirige el actor se recalcula y un nuevo punto es generado, modificando el valor de la componente *Z por -Z y* variando los valores de X y Y, logrando con esto que el pez se mueva en sentido contrario del objetivo y que las posiciones de los actores se solapen lo menos posible.

La imagen mostrada a continuación ejemplifica dicho proceso:

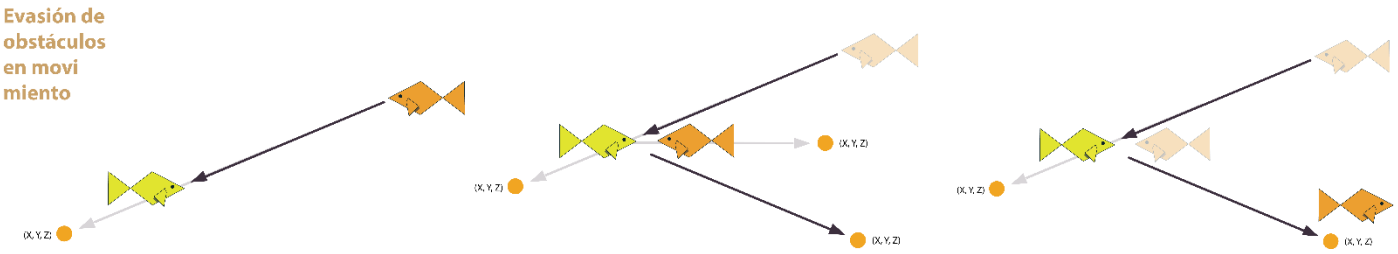

Figura 14 Evasión de obstáculos móviles

El alimento es uno de los efectos que también debe estar presente en la simulación, dado que aporta realismo a la misma y logra un flujo natural que no debe estar ausente en dicho ambiente. Unos segundos después que la comida es arrojada en la escena, uno de los actores tomado de forma aleatoria, el cual se dirige en ese momento a un punto, desvía su curso para buscarla, cuando la alcanza, ésta desaparece, simulando el efecto de que el actor comió. Luego de esto continua la generación de puntos aleatorios. La siguiente figura muestra cómo se comporta el proceso de alimentación de los peces:

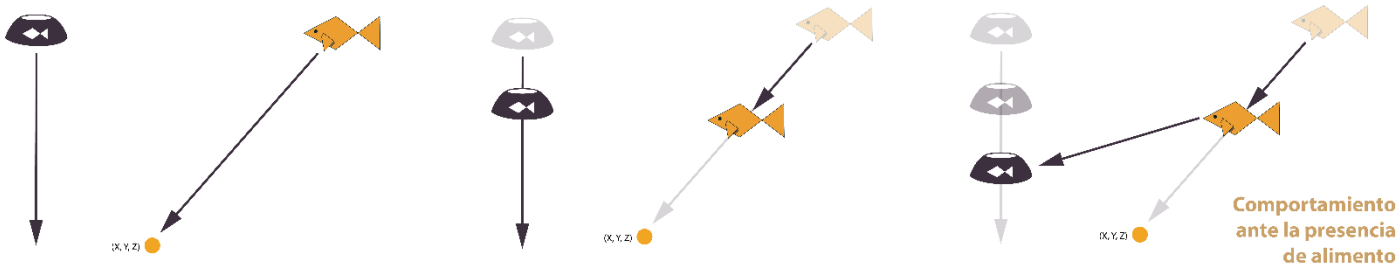

Figura 15 Comportamiento del actor ante la presencia de alimento

Otro de los elementos importantes es el enfoque y seguimiento de los actores de forma individual. Dicho elemento es logrado a partir de lo que se conoce como Enfoque y paneo de cámara, el cual se encarga de seleccionar de forma aleatoria, en una primera etapa, el actor el cual será seguido por la cámara para luego realizar el enfoque y el paneo en éste. El siguiente gráfico explica de forma sencilla el proceso:

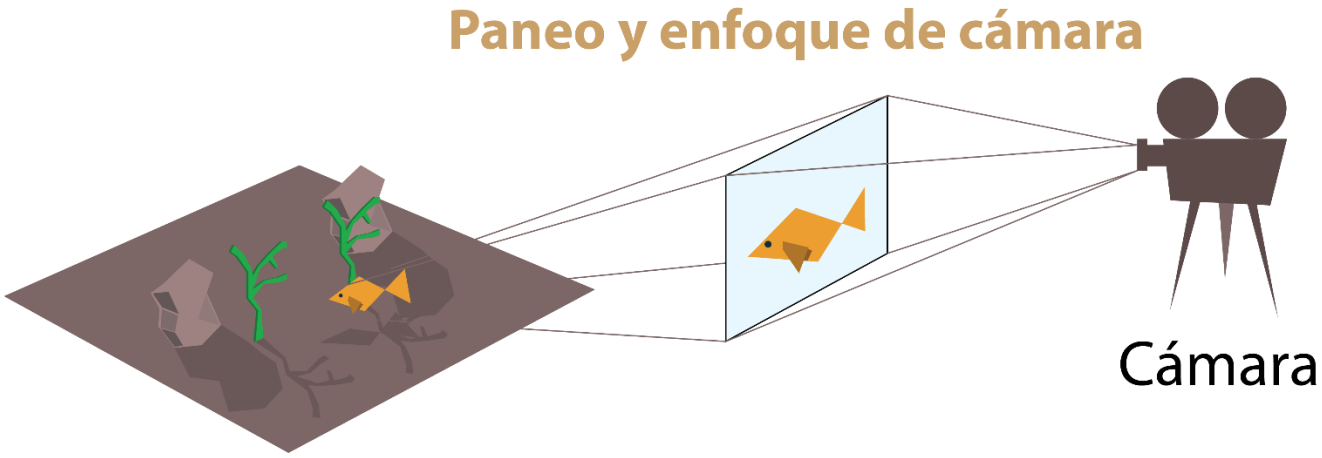

Figura 6 Enfoque y paneo de cámara

La simulación es controlada a través de una interfaz de configuración la cual genera un archivo de tipo XML el cual contiene los valores que toman las funcionalidades. Dentro de dicho archivo la información está organizada a través de etiquetas y queda estructurada de la siguiente forma:

Primeramente se etiquetan los datos del archivo tales como tipo de codificación y versión de éste tal como se muestra a continuación:

```
<?xml version="1.0" encoding="utf-8"?>
```

Figura 16 Encabezado del archivo de configuración

Donde *?xml* indica el tipo de archivo que es, *version* indica la versión del mismo y *encoding* indica el estándar de codificación que utiliza este tipo de archivo.

Después del encabezado son especificados los esquemas utilizados por este tipo de archivo así como el cuerpo de éste donde se localizan las etiquetas de las funcionalidades tal como se muestra a continuación:

```
<UserData
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        Cuerpo del archivo
</UserData>
```

Figura 17 Esquemas estándares utilizados por el archivo y cuerpo del mismo.

En el caso de las funcionalidades su estructura se ubica en el cuerpo del archivo y queda organizada de la siguiente forma:

```
<Agregar>True/False</Agregar>
<Eliminar>True/False</Eliminar>
<Paneo>True/False</Paneo>
<Alimentar>True/False</Alimentar>
<tipoPez>0/1/2</tipoPez>
```

Figura 18 Estructura de las funcionalidades en el archivo de configuración.

La metodología a utilizar en el desarrollo del sistema es Extreme Programming (XP), la cual es ideal por su simplicidad, su comunicación, la realimentación y la reutilización del código generado

por los desarrolladores. Es la metodología por excelencia para proyectos pequeños y de pocos integrantes haciendo énfasis en la colaboración y la comunicación entre estos. Se caracteriza por generar pocos artefactos y es flexible al cambio ya que la arquitectura de los sistemas hechos con ésta se encuentran en constantes cambios, definiéndose y mejorándose a lo largo del proceso de desarrollo.

2.2 Requerimientos no funcionales

Los requerimientos no funcionales son las características que hacen al producto atractivo, usable, rápido o confiable. Aunque no son parte de la razón fundamental del producto, son necesarios para hacer que el software funcione de la manera deseada. También se deben tener en cuenta a la hora de diseñar e implementar el sistema, ya que tienen repercusión directa en la calidad de la herramienta. (Pressman, 2008)

Usabilidad: El sistema debe ser usado por los especialistas en las terapias de relajación.

Rendimiento: La aplicación debe lograr valores en la frecuencia de visualización superiores o iguales a 30 cuadros por segundo (FPS por sus siglas en inglés).

Hardware: Es necesario para el funcionamiento del sistema mínimo, un microprocesador Intel Pentium IV a 2,6 GHz, 1 GB de RAM, 32 bits de profundidad de color y 128 MB de memoria de video integrado.

Software: El sistema es compatible con Unity3D sobre Windows. Deben tener instalado las bibliotecas del lenguaje C++ (VisualC++ Redistributable 2005 o superior) y DirectX 9.0 o superior.

2.3 Fase de exploración

La metodología XP se enfoca principalmente en la satisfacción del cliente, ya que trata de dar al usuario final el software que él necesita y en el momento que lo requiere. También permite aprovechar al máximo las ventajas del trabajo en grupo, porque crea un elevado nivel de colaboración y comunicación, lo que hace que tanto los jefes de proyecto, los clientes y desarrolladores, sean parte del equipo y protagonistas en el desarrollo del software.

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

Para el desarrollo de la herramienta que se desea implementar se utilizarán 6 semanas, debido a que a pesar de no ser un sistema de gran extensión, el equipo de desarrollo está poco familiarizado con las herramientas que se utilizarán.

2.4 Historias de Usuario (HU)

<table>
<tr><th colspan="3">Historia de Usuario</th></tr>
<tr><td>Número: 1</td><td colspan="2">Nombre: Desplazamiento del pez</td></tr>
<tr><td colspan="3">Usuario: Autor</td></tr>
<tr><td colspan="2">Modificación de Historia Número:</td><td>Iteración Asignada: 1</td></tr>
<tr><td colspan="2">Prioridad en Negocio: Alta
(Alta / Media / Baja)</td><td>Puntos Estimados:</td></tr>
<tr><td colspan="2">Riesgo en Desarrollo: Alto
(Alto / Medio / Bajo)</td><td>Puntos reales:</td></tr>
<tr><td colspan="3">Descripción:
El sistema debe permitir al actor desplazarse de forma totalmente aleatoria incluyendo la evasión de obstáculos que se encuentre en su trayectoria</td></tr>
</table>

Observaciones: Permite el desplazamiento de forma aleatoria del actor, incluyendo la evasión de obstáculos	
Historia de Usuario	
Número: 2	Nombre: Crear entorno de simulación
Usuario: Autor	
Modificación de Historia Número:	Iteración Asignada: 1
Prioridad en Negocio: Alta (Alta / Media / Baja)	Puntos Estimados:
Riesgo en Desarrollo: Alto (Alto / Medio / Bajo)	Puntos Reales:
Descripción: Se crea el entorno en que se desarrolla la simulación, así como los actores que interactúan con él, logrando la calidad gráfica necesaria, así como validando el sistema de colisiones a través de *scripts* y los efectos propios de dicho sistema y de los actores tales como el efecto de cáustica y la iluminación.	
Observaciones: Es el mundo virtual en el cual se desarrolla la simulación	

Historia de Usuario	
Número: 3	Nombre: Agregar pez
Usuario: Autor	
Modificación de Historia Número:	Iteración Asignada: 1

Prioridad en Negocio: Alta (Alta / Media / Baja)	Puntos Estimados:
Riesgo en Desarrollo: Bajo (Alto / Medio / Bajo)	Puntos Reales:
Descripción: El sistema permite que el usuario seleccione el tipo de pez con el cual desarrollar la simulación y lo puede agregar a través del teclado con una tecla definida o pulsando el botón Agregar en la interfaz de configuración.	
Observaciones: Agrega un nuevo pez a la escena	

Historia de Usuario		
Número: 4	Nombre: Eliminar pez	
Usuario: Autor		
Modificación de Historia Número:		Iteración Asignada:1
Prioridad en Negocio: Alta (Alta / Media / Baja)		Puntos Estimados:
Riesgo en Desarrollo: Bajo (Alto / Medio / Bajo)		Puntos Reales:
Descripción: El sistema permite al usuario seleccionar el tipo de pez que se elimina de la simulación, puede eliminarlo a través del teclado o pulsando el botón Eliminar de la interfaz de configuración		

Observaciones: Agrega un pez

Historia de Usuario		
Número: 5	Nombre: Alimentar Pez	
Usuario: Autor		
Modificación de Historia Número:		Iteración Asignada: 2
Prioridad en Negocio: Alta (Alta / Media / Baja)		Puntos Estimados:
Riesgo en Desarrollo: Medio (Alto / Medio / Bajo)		Puntos Reales:
Descripción: El sistema permite arrojar comida en la simulación al presionarse una tecla definida o pulsarse un botón en la interfaz de configuración.		
Observaciones: Alimenta al pez		

Historia de Usuario		
Número: 6	Nombre: Realizar paneo de cámara	
Usuario: Autor		
Modificación de Historia Número:		Iteración Asignada: 2
Prioridad en Negocio: Alta		Puntos Estimados:

(Alta / Media / Baja)	
Riesgo en Desarrollo: Medio (Alto / Medio / Bajo)	Puntos Reales:
Descripción: Al presionarse una tecla definida, el sistema permite la ejecución de la funcionalidad de realizar paneo de cámara.	
Observaciones: Realiza un paneo de cámara	

Historia de Usuario	
Número: 7	Nombre: Generar archivo de configuración
Usuario: Autor	
Modificación de Historia Número:	Iteración Asignada: 3
Prioridad en Negocio: Alta (Alta / Media / Baja)	Puntos Estimados:
Riesgo en Desarrollo: Medio (Alto / Medio / Bajo)	Puntos Reales:
Descripción: Se genera un archivo de configuración de tipo XML que sirve para almacenar los parámetros con los cuales se maneja la simulación del entorno	
Observaciones: Genera el archivo configuración de tipo XML	

Historia de Usuario	
Número: 8	Nombre: Desarrollo de la interfaz de configuración
Usuario: Autor	
Modificación de Historia Número:	Iteración Asignada: 3
Prioridad en Negocio: Alta (Alta / Media / Baja)	Puntos Estimados:
Riesgo en Desarrollo: Medio (Alto / Medio / Bajo)	Puntos Reales:
Descripción: Interfaz de configuración que permitirá al usuario realizar ajustes antes y durante el desarrollo de la simulación.	
Observaciones: Se desarrolla la interfaz de configuración	

2.4.1 Diseño de Casos de Pruebas

Se realizan los diseños de los casos de pruebas teniendo en cuenta las características del software y las funcionalidades a comprobar. Las pruebas demuestran que la implementación realizada es correcta y válida. Se define como fin de la etapa de desarrollo cuando no son encontrados errores en el sistema.

2.5 Pruebas de aceptación

Las pruebas de aceptación son consideradas como las pruebas de caja negra, estas tienen una importancia crítica para el éxito de lo que es considerado una iteración. Este tipo de pruebas son creadas y definidas a partir de las historias de usuario propuestas por el cliente. Tienen como objetivo asegurar que las funcionalidades cumplen con lo que se espera de ellas. Marcan el camino

a seguir indicándole al equipo de desarrollo las funcionalidades más relevantes. Permiten a los programadores conocer que es lo que resta por hacer. Además de brindar la posibilidad de que las pruebas se asemejen al ambiente de producción.

En el capítulo 3, epígrafe 3.2 del presente documento se relacionan las pruebas realizadas al sistema, así como sus resultados y análisis.

2.6 Planificación y entrega

Estableciendo niveles de prioridad para cada historia de usuario, se realiza una estimación del esfuerzo necesario para cada uno de ellos. Se realiza un cronograma en conjunto con el cliente a partir de los acuerdos anteriormente planteados y se entrega como resultado de esta etapa, un Plan de Entregas.

2.6.1 Plan de entregas

El plan de entregas se planifica para no más de 3 semanas. Se elabora con el objetivo de que los programadores obtengan una estimación de dichas historias en cuanto al nivel de detalle, o sea, para fijar el período de tiempo que se puede tardar en la implementación de cada una. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura.

Los elementos que se toman en cuenta para la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable. (Penadés, 2009)

Tabla 4.Plan de Entregas

Entregas	Historias de usuario	Número

Entrega 1	Desplazamiento del pez	1
	Crear entorno de simulación	2
Entrega 2	Funcionalidad de agregar un pez	3
	Funcionalidad de eliminar un pez	4
	Funcionalidad de alimentar al pez	5
	Funcionalidad de paneo de cámara	6
Entrega 3	Generar archivo de configuración	7
	Desarrollo de la interfaz de configuración	8

Se realizan versiones atendiendo al Plan de Entregas, basándose en un rango de fechas estimada concluyendo con la terminación del producto.

Tabla 5. Planificación de entregas

Entrega	**Iteración** **1ra Semana de Abril**	**Iteración 2** **2ra Semana de Mayo**	**Iteración 3** **3ra Semana de Mayo**
Entrega 1	Versión 1	Versión 2	Versión 3
Entrega 2		Versión 1	Versión 2

Entrega 3			Versión 1

Tabla 6. Estimación del esfuerzo por historia de usuario

Iteración	Historias de usuario	Tiempo estimado	Tiempo real
1	Desplazamiento del pez	3	3
	Crear entorno de simulación	3	3
2	Funcionalidad de agregar un pez	2	2
	Funcionalidad de eliminar un pez	2	2
	Funcionalidad de alimentar al pez	2	2
	Funcionalidad de paneo de cámara	1	1
3	Generar archivo de configuración	2	2
	Desarrollo de la interfaz de configuración	3	3

2.7 Diseño del sistema

El diseño crea una estructura para la organización de la lógica del sistema y permite que sea escalable con cambios en un solo lugar. Los diseños se crean con el mayor grado de sencillez posible, dividiéndolos en varias partes que pudieran tornarse complejas.

En la metodología XP no es necesaria la descripción del sistema a través de diagramas de clase utilizando notación UML, sino que se guía por técnicas como las tarjetas CRC (Contenido, Responsabilidad y Colaboración).

También pueden ser utilizados otros recursos como los diagramas para obtener una mejor visión y comunicación entre el equipo de trabajo, siempre y cuando genere información importante.

2.7.1 Arquitectura del sistema

El sistema se desarrolla sobre el motor gráfico Unity 3D y posee una arquitectura en 3 capas, las cuales se muestran a continuación.

Capa de presentación: Es la capa que interactúa con el usuario y está conformada por la vista de la simulación en el entorno virtual y la interfaz de configuración.

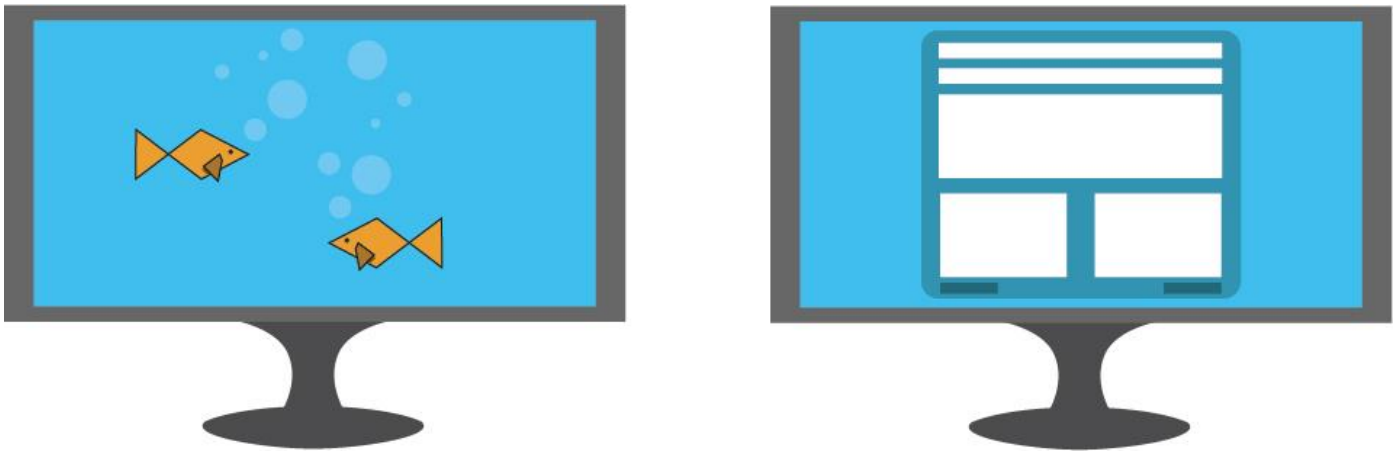

Figura 19 Capa de presentación

Capa de configuración: En esta capa se encuentran todos los scripts que hacen posible toda la configuración de la simulación, incluyendo los scripts base para el entorno virtual.

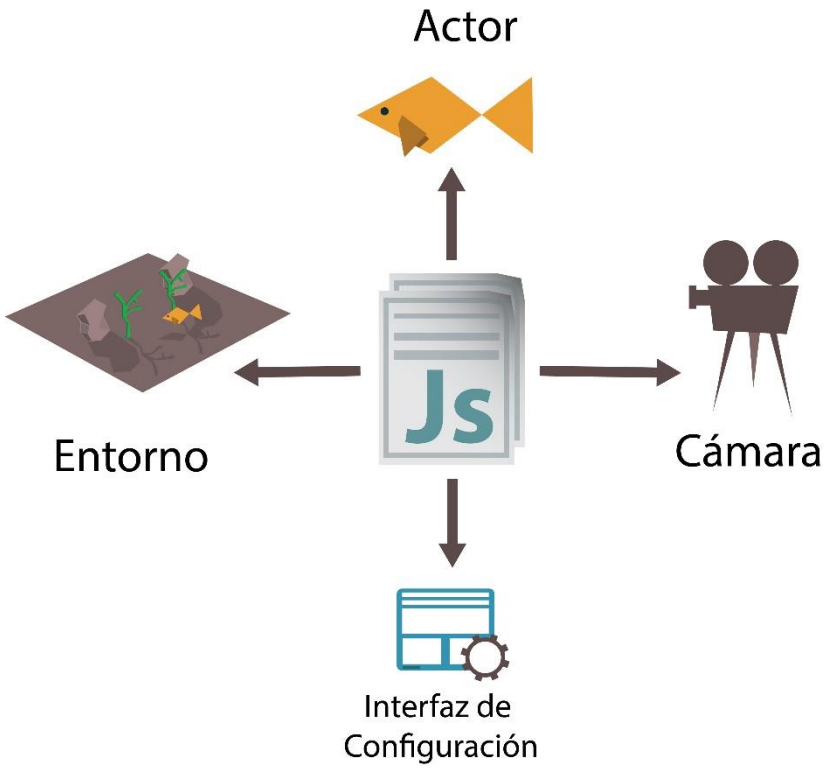

Figura 20 Capa de configuración

Capa de soporte: Es la capa que contiene las bibliotecas para la física, los gráficos, el sonido y el almacenamiento de los gráficos suministrados por el motor gráfico de Unity 3D.

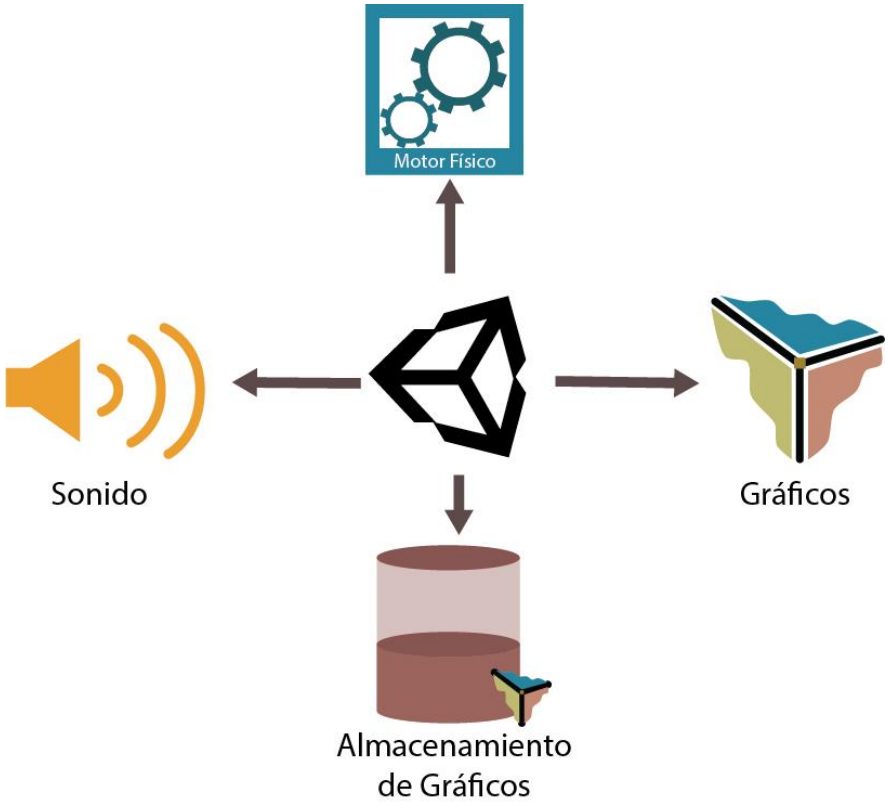

Figura 9 Capa de soporte

2.7.2 Tarjetas CRC del sistema

“fishMovement”	
Descripción: Pertenece a los scripts básicos del sistema, permite el movimiento del pez en la simulación de forma aleatoria, dentro del mismo se valida la evasión de obstáculos y varios efectos que aportan mayor realismo a la simulación.	
Patrol()	Es la función principal en la cual se ejecutan las instrucciones que permiten al pez interactuar de forma básica con el entorno recorriéndolo a través de puntos generados de forma aleatoriamente y evadiendo obstáculos.

MoveTowards()	Función que permite el movimiento de traslación y rotación del pez en la simulación.
TargetReached()	Permite recalcular el punto aleatorio hacia donde el pez debe moverse, una vez que este haya alcanzado el punto anterior.
EvadeAnotherFish()	Se encarga de la toma de decisiones cuando un pez se encuentra en la trayectoria de otro, evitando el solapamiento en los modelos y validando así las colisiones entre los actores.
AvoidObjects()	Valida las colisiones entre los peces y los objetos estáticos en la escena, evitando el solapamiento de los modelos y permitiendo un nivel de simulación más realista.
SlowSpeed()	Funcionalidad que regula la velocidad del pez cuando está próximo a alcanzar el objetivo, reduciendo esta para hacer el movimiento más suave y natural.
Responsabilidades: Start()	

“feedFish”	
Descripción: Script que permite la funcionalidad de alimentar a los peces, constituye un elemento importante que contribuye al nivel de realismo que se desea alcanzar en la simulación.	
feedFish()	Implementa la funcionalidad de alimentar a los peces. Dicha funcionalidad es realizada de dos formas, la primera, de manera automática, cada cierto tiempo se lanza comida a la pecera, dicho tiempo está controlado por una variable. La segunda es a decisión

	del usuario, por medio de una tecla predefinida se podrá arrojar después de 5 segundos.
Responsabilidades: Update()	

“AddFish”	
Descripción: Script que permite la funcionalidad de agregar peces a la escena.	
AddFish()	Funcionalidad que permite la adición de uno o más peces en la escena ésta consiste en presionar una tecla predefinida o pulsar un botón en la interfaz de configuración y la funcionalidad agrega peces teniendo en cuenta el tipo de pez que se selecciona en dicha interfaz.
Responsabilidades: Update()	

“RemoveFish”	
Descripción: Script que permite la funcionalidad de eliminar peces a la escena.	
RemoveFish()	Funcionalidad que permite la eliminación de uno o más peces en la escena ésta consiste en presionar una tecla predefinida o pulsar un botón en la interfaz de configuración y la funcionalidad elimina peces teniendo en cuenta el tipo de pez que se selecciona en dicha interfaz.
Responsabilidades: Update()	

“PanCam”

Descripción: Script que permite la funcionalidad de realizar paneos de cámara por la escena.	
PanCam()	Funcionalidad que permite la realización de paneos de cámara en la escena de la simulación. Dichos paneos se realizan a disposición del usuario con un tiempo de duración finito que no excede los 40 segundos.
Responsabilidades: Update()	

"MoveLight"	
Descripción: Script que permite la animación del efecto de caustica en la escena	
Update()	Funcionalidad que permite la animación de la cáustica generada en la escena, dicha animación es un simple movimiento rotatorio.
Responsabilidades: Update()	

"RotatePlanes"	
Descripción: Script que permite la rotación de planos inclinados en la escena.	
Update()	Funcionalidad que permite la rotación de planos inclinados en la escena, con el objetivo de lograr el efecto de rayos de luz atravesando el agua.
Responsabilidades: Update()	

"Config"

Descripción: Funcionalidad que permite la escritura en el archivo de configuración.	
Start()	Inicializa las variables que necesitan tomar valores antes de que se ejecute la función Update y comience la simulación.
Update()	Examina y carga los datos de la pre-configuración en el archivo y escribe en éste los datos anteriormente mencionados.
OnGUI()	Genera los elementos con los que se interactúa en la interfaz de configuración, en este caso, su responsabilidad es sobre la generación de botones y *toolbars*.
CreateXML()	Se encarga de la creación del archivo de configuración en formato XML, en la ruta que se le entre por datos
SerializeObject()	Se concentra fundamentalmente en llevar los datos de configuración a un formato entendible por el script de entrada de datos, codificando estos en UTF8
Responsabilidades: Start(), Update(), OnGUI()	

“loadConfig”	
Descripción: Script que permite al sistema leer los datos proporcionados por el archivo de configuración.	
Awake()	Crea el objeto que contiene los datos almacenados en el archivo de configuración.
Start()	Inicializa las variables que necesitan tomar valores antes de que se ejecute la función *Update* y comience la simulación.

Update()	Se encarga de analizar por cada etiqueta los valores que se han extraído del archivo de configuración y ejecuta cada funcionalidad acorde a dichos valores.
LoadXML()	Su responsabilidad abarca la lectura del archivo de configuración para su posterior transferencia hasta el objeto de almacenamiento.
DeserializeObject()	Realiza la conversión del texto en el archivo de configuración, decodificándolo y llevando sus datos a texto plano.
Responsabilidades: Update()	

Conclusiones del capitulo

Con la culminación de este capítulo, y apoyándose en lo anteriormente expuesto, se ha llegado a la conclusión de que se han definido las bases para el desarrollo del sistema. Gracias al conjunto de bibliotecas y características aportadas por Unity3D.

CAPÍTULO 3: PRUEBAS

Introducción

Dentro de las bases para el desarrollo de software utilizando la metodología Extreme Programming (XP) se encuentra la fase de pruebas. En esta metodología, las pruebas son uno de los pilares fundamentales, y es también un proceso el cual no se puede pasar por alto. Estas permiten elevar sistemáticamente la calidad del software, disminuyendo la cantidad de errores detectados, así como también el tiempo entre el momento de la aparición de estos y su detección. (Beck, 1999)

En este capítulo se muestran los resultados obtenidos de las pruebas de aceptación las cuales fueron diseñadas anteriormente, aquí se muestran sus resultados. Estas pruebas son guiadas en su realización por las historias de usuarios y validan que el sistema posea el rendimiento, la calidad y las características que el cliente espera.

3.1 Pruebas de software

Las pruebas conforman un proceso en el cual el software es ejecutado bajo ciertas condiciones y usando requerimientos específicos. Los resultados de estas ejecuciones son registrados y analizados póstumamente con el objetivo de evaluar algún aspecto específico de dicho software.

Según Pressman en el libro “Ingeniería de software, un enfoque práctico” (Pressman, 2002), definiendo los objetivos de las pruebas, hace referencia a Glem Myers, quien establece algunos de los atributos que definen los objetivos de las estas:

La prueba es un proceso de ejecución de un programa con la intención de descubrir un error.

Un buen caso de prueba es aquel que tiene una alta posibilidad de mostrar un error no descubierto hasta entonces.

Una prueba tiene éxito si descubre un error no detectado anteriormente.

3.2 Pruebas de aceptación

Apoyándose en las pruebas de aceptación se determina si el software cumple con las necesidades del cliente, obteniéndose la conformidad de éste. Los resultados de las mismas se muestran a continuación.

Caso de prueba de Aceptación	
Numero: 1.	**Historia de Usuario: 1.**
Nombre: Desplazamiento del pez	
Descripción: El caso de prueba permite comprobar el desplazamiento de los peces en el entorno.	
Condiciones de ejecución: La simulación debe estar ejecutándose.	
Entradas/Pasos de ejecución: El usuario ejecuta el programa y configura uno o más peces que se encuentren en la simulación.	
Resultado esperado: El movimiento de los peces es aceptable, pero luego de aumentar la cantidad de estos a 5 se identificó la existencia de un solapamiento parcial. Resulta importante analizar si el comportamiento individual de la inteligencia de los peces es suficiente para resolver este problema. Se procede a calibrar la distancia y se identifica que existe menos solapamiento pero se muestran efectos visuales de latencia en la toma de decisiones del pez que afecta la fluidez de la simulación.	
Evaluación de prueba: Satisfactorio.	

Caso de prueba de Aceptación	
Numero: 2**.**	**Historia de Usuario:** 2**.**

Nombre: Crear entorno de simulación
Descripción: El caso de prueba permite verificar que el entorno cumpla con los niveles de calidad definidos.
Condiciones de ejecución: La simulación debe está detenida.
Entradas/Pasos de ejecución: El usuario ejecuta el programa y automáticamente puede visualizar el entorno con o sin peces en el mismo.
Resultado esperado: El entorno fue creado teniendo en cuenta asegurando una calidad gráfica acorde con el objetivo del sistema, así como el conjunto de efectos físicos y visuales para que los peces interactúen con el mismo.
Evaluación de prueba: Satisfactorio.

Caso de prueba de Aceptación	
Numero: 3**.**	**Historia de Usuario:** 3**.**
Nombre: Agregar pez	
Descripción: El caso de prueba permite agregar un pez a la simulación, presionando una tecla definida anteriormente, o a través de la interfaz de configuración.	
Condiciones de ejecución: La simulación debe estar ejecutándose.	
Entradas/Pasos de ejecución: Cuando la simulación está ejecutándose, presionar la tecla R para agregar un pez de forma aleatoria. Configurar el tipo de pez en la interfaz de configuración.	
Resultado esperado: Los peces se agregan de forma satisfactoria, incorporándose a la simulación y manteniendo la aleatoriedad en sus movimientos.	

Evaluación de prueba: Satisfactorio.

Caso de prueba de Aceptación	
Numero: 4**.**	**Historia de Usuario:** 4**.**
Nombre: Eliminar pez	
Descripción: El caso de prueba permite eliminar un pez a la simulación, presionando una tecla definida anteriormente, o a través de la interfaz de configuración.	
Condiciones de ejecución: La simulación debe estar ejecutándose.	
Entradas/Pasos de ejecución: El usuario ejecuta el programa y configura uno o más de un pez en la escena.	
Resultado esperado: Los peces se eliminan de forma satisfactoria, retirándose hacia un punto fuera de la pantalla de la simulación y luego destruyéndose automáticamente.	
Evaluación de prueba: Satisfactorio.	

Caso de prueba de Aceptación	
Numero: 5**.**	**Historia de Usuario:** 5**.**
Nombre: Alimentar pez	
Descripción: El caso de prueba permite alimentar los peces de forma manual, presionando una tecla definida anteriormente, o de forma automática, cada cierto tiempo.	
Condiciones de ejecución: La simulación debe estar ejecutándose.	

Entradas/Pasos de ejecución: El usuario presiona la tecla F para agregar comida en la simulación, o de forma automática se agregará comida cada cierto tiempo.
Resultado esperado: La comida se introduce correctamente de forma manual y de forma automática.
Evaluación de prueba: Satisfactorio.

3.3 Validación del sistema

La validación es un proceso importante en el desarrollo de software porque permite conocer cuán efectiva es la aplicación que se desarrolla. Esta actividad se llevó a cabo en el Centro Memorial Dr. Martin Luther King Jr. a un grupo de 16 ancianos con edades comprendidas entre los 70 y 90 años. El proceso terapéutico consistió en un primer momento de una charla donde se le explicó al grupo cómo se iba a proceder. Colocándolos en posiciones cercanas al plano de proyección donde, teniendo en cuenta el tamaño de éste, pudieran observar el contenido de la simulación siguiendo una distribución tal y como muestra la siguiente figura:

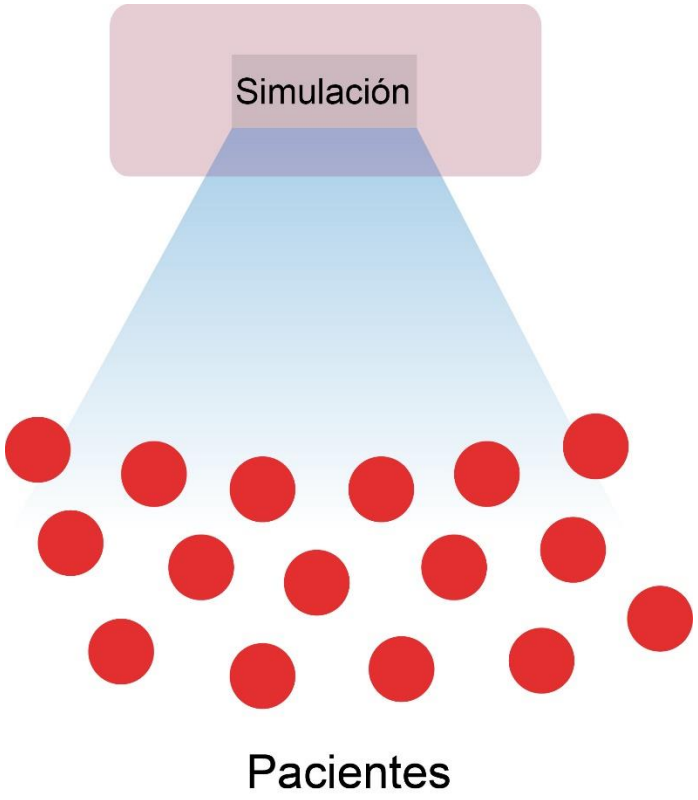

Figura 21 Distribución de los pacientes frente a la simulación

Antes de proceder con el ejercicio se realizó la recogida inicial de datos de los pacientes registrándose el nombre de éstos y el número de pulsaciones iniciales, con el objetivo de medir cuán relajados estaban al final de la sesión. Durante el desarrollo de ésta última el especialista orientó un grupo de ejercicios a los pacientes mientras éstos observaban la simulación que además se acompañó de contenido sonoro para enriquecer el proceso terapéutico. También se acondicionó el lugar para que la iluminación no fuese intensa y se procuró estar completamente en silencio para evitar interrumpir la actividad.

La terapia duro aproximadamente 15 minutos tomando entre cada ejercicio realizado 2 minutos, observándose durante los primeros minutos síntomas de relajación por parte de los pacientes los cuales disfrutaron de la simulación.

Concluyendo la sesión se volvió a recoger la relación de pulsaciones a los pacientes las cuales fueron analizadas y estableciendo las operaciones descritas en el capítulo 1 epígrafe 1.1, arrojaron los siguientes resultados:

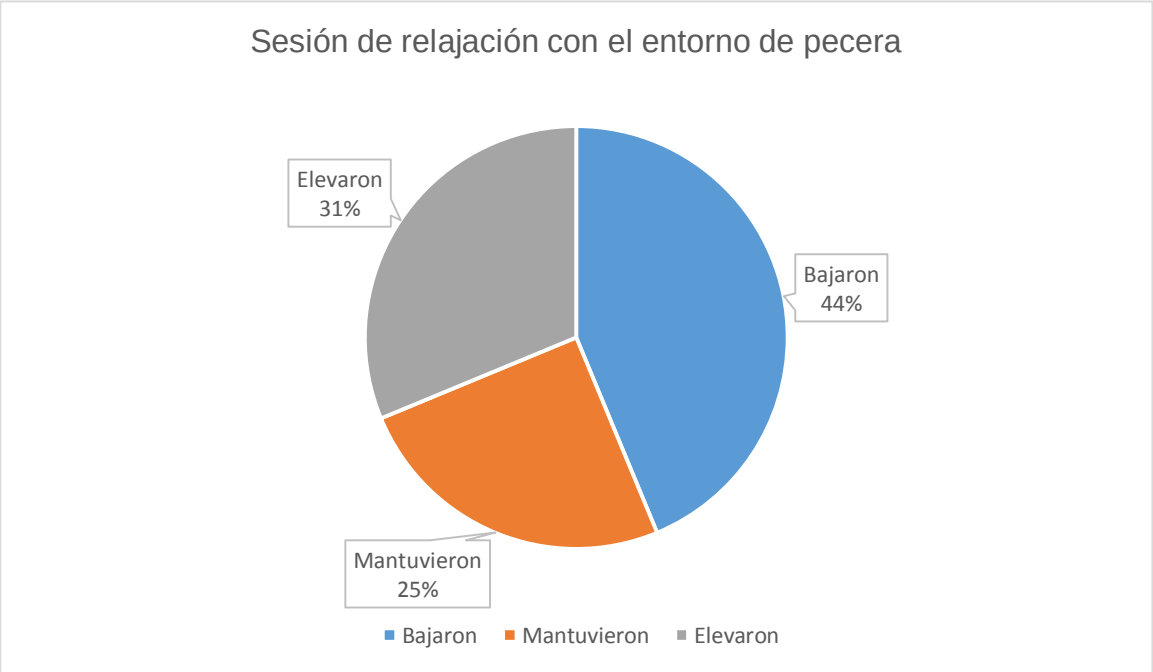

Figura 22 Resultados obtenidos durante la validación

Por lo cual se puede concluir que la solución obtuvo aproximadamente un 70% de éxito teniendo en cuenta que estos resultados pueden variar dependiendo de la forma en que se realice la terapia por parte del especialista.

En un último momento se escucharon y registraron las opiniones personales de los pacientes respecto a la efectividad de la solución llegando a la conclusión de que ésta obtuvo un elevado nivel de aceptación por parte de los mismos los cuales exhortaron al autor a seguir elevando la calidad de ésta con el objetivo de obtener mejores resultados en futuras sesiones.

Conclusiones del capítulo

En el capítulo se realizaron las pruebas de aceptación a todas las historias de usuarios antes definidas e implementadas, las cuales obtuvieron resultados satisfactorios. Apoyándose en la

planificación realizada, la calidad alcanzada y los resultados de las pruebas llevadas a cabo queda demostrado que el sistema está terminado y cumple con los objetivos planteados además de satisfacer las necesidades de los usuarios finales.

CONCLUSIONES

Después de terminada la investigación y el desarrollo de la solución se puede concluir:

- El estudio de las tendencias actuales sobre el uso de tecnologías para el desarrollo de entornos virtuales para terapias de relajación, posibilitó la selección de las mismas con el fin de construir la solución.
- El estudio realizado demuestra que Unity 3D es viable como herramienta para la creación de simulaciones de entornos naturales por el soporte que brinda para la implementación multilenguaje y su flexibilidad en la integración con otras herramientas y *plugins*.
- El desarrollo de una aplicación informática que simule un entorno de pecera, con parámetros configurables permite alcanzar altos niveles de relajación en la ejecución de las terapias.
- La aplicación de las pruebas de aceptación permitió comprobar que la solución cumple con las funcionalidades requeridas y la validación de un demo de la solución permitió comprobar el grado de efectividad de la misma.

RECOMENDACIONES

Teniendo en cuenta el tiempo de desarrollo, se proponen las siguientes recomendaciones:

- Agregar soporte para estereoscopía y visión tridimensional en todas sus variantes.
- Incorporar la negociación entre agente como un mecanismo para lograr un mejor comportamiento de los peces en el entorno.
- Aumentar la diversidad en cuanto a entorno, actores y movimientos de la cámara.

BIBLIOGRAFÍA

Alonso, Jesús Alonso. 2009. Videojuegos 3D. [PDF] Cataluña : Universidad Abierta de Cataluña, 2009.

Barrios, Francisco. 2013. La Relajación. La Relajación. [En línea] La Relajación, 2013. http://www.larelajacion.com/larelajacion/43autoalusiva.php.

Buckland, Mat. 2005. Programming Game AI by Example. [Help] s.l. : Wordware Publishing, 2005. 1556220782.

Cabrera, MsC Pascual Felipe Pérez. 2012. Slideshare. Slideshare. [En línea] 12 de Diciembre de 2012. [Citado el: 22 de Marzo de 2014.] http://www.slideshare.net/uzita1/simulacin-15675046.

Complementaria, Programa Nacional de Medicina. 2000. Manual de Relajación. Lima : Centro de Documentación Carlos Enrique Paz Roldán, 2000. 9972-785-10-6.

Correa, Omar y Gomez, Elizabeth. 2010. VirtualRelax, Sistema de Realidad Virtual para la Terapia de Relajación. [PDF] La Habana : Universidad de las Ciencias Informáticas, 2010.

Day, Robert A. 2005. Cómo escribir y publicar trabajos científicos. Washington DC : The Oryx Press, 2005. 92 75 31598 1.

Gobierno de España, Ministerio de Educación, Política Social y Deporte. 2013. Educación. Educación. [En línea] Gobierno de España, 3 de Octubre de 2013. [Citado el: 5 de Febrero de 2014.] http://ares.cnice.mec.es/informes/19/contenidos/1.htm#_Toc170824153.

Hierrezuelo, Hydeé Asela. 2004. Relajación 21. [Documento de Texto] La Habana : s.n., 2004.

ibaiondo. 2014. iPadforos. iPadforos. [En línea] 1 de Junio de 2014. [Citado el: 10 de Noviembre de 2013.] http://www.ipadforos.com/software/halotea-102-software-creador-sonidos-naturaleza-para-relajarse-t32761.html.

Illián, Isabel, Hurtado, Ana y Martín, Julio César. 2006. Protocolo de Relajación. Murcia : Servicio Murciano de Salud, 2006. MU-554-2004.

Kniberg, Henrik. 2007. Scrum y XP desde las trincheras. [PDF] s.l. : C4Media, 2007.

Meloleo. 2014. Meloleo. *Meloleo.* [En línea] Meloleo, 5 de Abril de 2014. [Citado el: 14 de Mayo de 2014.] http://www.meloleo.com/10-juegos-flash-para-relajarse.

Morales, Gerardo Abraham, y otros. 2010. *Procesos de Desarrollo para Videojuegos.* [PDF] Juarez : Instituto de Ingeniería y Tecnología. Universidad Autónoma de Ciudad Juárez, 2010.

NYU Langone Medical Center. 2013. NYU Langone. *NYU Langone.* [En línea] NYU Langone, 12 de Agosto de 2013. [Citado el: 22 de Marzo de 2014.] http://www.med.nyu.edu/content?ChunkIID=126576.

Penadés, Patricio Letelier y Carmen, Maria. 2009. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Valencia : Universidad Politécnica de Valencia, 2009.

Pressman, Roger S. 2008. *Ingeniería de Software, un enfoque práctico.* s.l. : McGraw-Hill, 2008. 970-10-5473-3.

Relajación. **López, Rosa. 1996.** 12, La Habana : Revista Cubana de Medicina General Integral, 1996, Vol. 4.

Riva, Giuseppe. 2004. *Immersive Virtual Telepresence.* [PDF] Amsterdam : IOS Press, 2004.

Sánchez, Orlando. 2013. *Módulo para la creación de videojuegos para jugadores virtuales de tipo batalla utilizando Unity 3D.* [PDF] La Habana : Universidad de las Ciencas, Universidad de las Ciencias Informáticas, 2013.

SourceForge. 2004. Aplicación de eXtreme Programming en ONess. [En línea] SourceForge, 2004. http://oness.sourceforge.net/proyecto/html/index.html.

Tejedor, Heliodoro. 2008. *Intligencia Artificial.* [PDF] Cataluña : Universidad Abierta de Cataluña, 2008.

Tellez, Gabriel. 20012. *XP Practices: A Successful Tool for Increasing and Transferring Practical Knowledge in Short-Life Software Development Projects.* 20012.

Terzopoulos, Demetri, Tu, Xiaoyuan y Grzeszczuk, Radek. 2000. *Artificial Fishes: Autonomous Locomotion, Perception, Behavior, and Learning in a Simulated Physical World.* [PDF] Toronto : Department of Computer Sciences, 2000.

Unity 3D. 2011. Unity 3D. Physics. *Unity 3D. Physics.* [En línea] Unity Technologies, 2011. http://unity3d.com/unity/features/physics.

—. 2012. Unity 3D. Scripting. [En línea] 2012. http://unity3d.com/unity/features/scripting.

—. 2012. Unity 3D. Terrain. *Unity 3D. Terrain.* [En línea] 2012. http://unity3d.com/unity/features/terrains.

Vicente, Mayte. 2008. *Protocolo de Relajación.* [PDF] s.l. : Centro de Salud de Castro Urdiales, 2008.

ANEXOS

Figura 23 Vista general del entorno de pecera

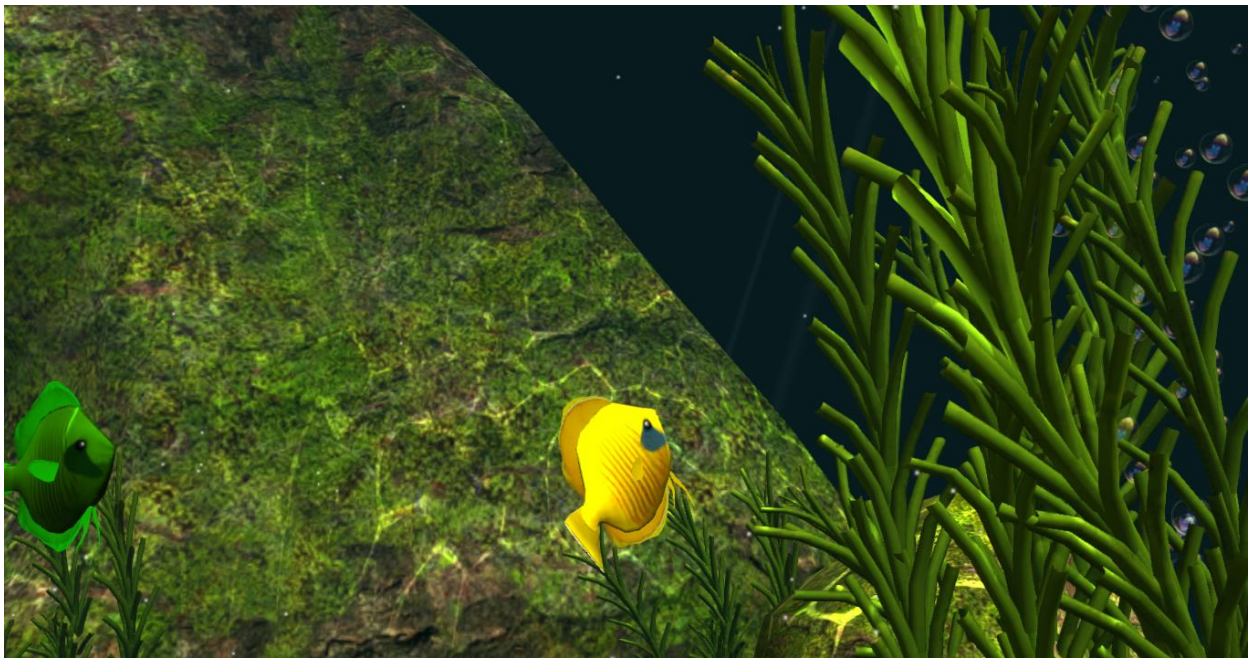

Figura 24 Vista de acercamiento (Paneo de cámara) del entorno de pecera

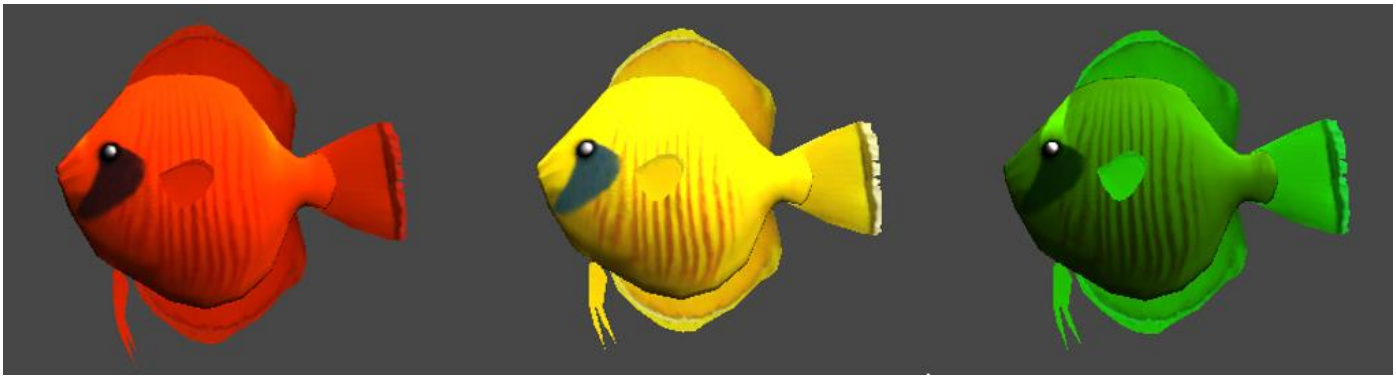

Figura 25 Colores de los peces

Figura 26 Vista de la interfaz de configuración

GLOSARIO

2D: En geometría y análisis matemático, un objeto o ente es tridimensional si tiene dos dimensiones. Cada uno de sus puntos puede ser localizado especificando dos números dentro de un cierto rango.

3D: En geometría y análisis matemático, un objeto o ente es tridimensional si tiene tres dimensiones. Es decir cada uno de sus puntos puede ser localizado especificando tres números dentro de un cierto rango.

Realidad Virtual: Es un ciencia basada en el empleo de ordenadores y otros dispositivos, cuyo fin es producir una apariencia de realidad que permita al usuario tener la sensación de estar presente en ella.

Render: Es el proceso de generar una imagen desde un modelo. Este término técnico es utilizado por los animadores o productores audiovisuales y en Software de diseño en 3D.

Osciloscopio: Instrumento de medición eléctrica para la representación gráfica de señales eléctricas que pueden variar en el tiempo.

Plugin: Es un complemento que se relaciona con otra aplicación para aportarle una nueva función.

Bit: Es la unidad básica y mínima que puede transmitirse en un ordenador.

MB: También conocido como megaocteto se refiere a una medida de cantidad en la informática.

GB: Se trata de una unidad de medida y su equivalencia es de 1024 MB.

GHz: Unidad de medida para las frecuencias del reloj interno de un procesador electrónico.

Python: Python es un lenguaje de programación creado por Guido van Rossum en el año 1991. En la actualidad se desarrolla como un proyecto de Código abierto, administrado por la Python Software Foundation.

C++: Es un lenguaje de programación orientado a objetos derivado del C. Creado en 1983 por Bjarne Stroustrup.

C#: Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft.

JavaScript: es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Boo: Es un lenguaje de programación orientado a objetos. Se considera como un dialecto de Pyton por su similitud con el mismo, aunque en su creación se tuvieron en cuenta también elementos de Perl y Rugby.

Script: Los script son casi siempre interpretados. El uso habitual de los scripts es realizar diversas tareas como combinar componentes, interactuar con el sistema operativo o con el usuario.

.NET: Arquitectura tecnológica, para la creación y distribución del software como un servicio.

Mac OS: (Macintosh Operating System) es el nombre del sistema operativo creado por Apple para su línea de computadoras Macintosh.

GNU\Linux: Linux es un núcleo libre de sistema operativo basado en Unix.

RAM: (Random Access Memory) es donde el computador guarda los datos que está utilizando en el momento presente.

Intel Pentium IV: Es un microprocesador de séptima generación basado en la arquitectura x86 y fabricado por Intel.

DirectX 3D: Parte de DirectX (conjunto de bibliotecas para multimedia), propiedad de Microsoft. Consiste en una API para la programación de gráficos 3D.

Open GL: Es una librería escrita en C que permite la manipulación de gráficos 3D a todos los niveles.

Assets: (Bloques de construcción), son utilizados por el editor de entornos de *Unity 3D* para estructurar y organizar los elementos que se utilizan en los videojuegos.

Rigid Body: Componente que brinda el editor de entornos de *Unity 3D* para añadir las propiedades físicas que permiten que los objetos se comporten como cuerpos rígidos**.**