Universidad de las Ciencias Informáticas

Centro de Entornos Interactivos 3D, Vertex

Facultad 4

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Componente para la personalización de personajes 3D en videojuegos desarrollados para Unity

Autor: Eric Mañalich Cossio

Tutores:

MSc. Andy Hernández Paez

Ing. Juan Carlos Miranda Corrales

Ing. Carlos Alberto Gavilla Cruz

La Habana, Mayo 2019

"Año 61 de la Revolución"

Frase

Programar sin una arquitectura o diseño en mente es como explorar una gruta sólo con una linterna: no sabes dónde estás, dónde has estado ni hacia dónde vas.

Danny Thorpe

Declaración de autoría

Declaramos ser autores de la presente tesis titulada: “Componente para la personalización de personajes 3D en videojuegos desarrollados para Unity”, y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Eric Mañalich Cossio

_____

Firma del Autor

MSc. Andy Hernández Paez

_____

Firma del Tutor

Ing. Juan Carlos Miranda Corrales

_____

Firma del Tutor

Ing. Carlos Alberto Gavilla Cruz

_____

Firma del Tutor

Datos de contactos

Nombre y Apellidos: MSc. Andy Hernández Paez

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero en Ciencias Informáticas

e-mail: andyhp@uci.cu

Ingeniero en Ciencias Informáticas (2012), graduado en la Universidad de las Ciencias Informáticas (UCI), siete años de experiencia en las áreas del conocimiento de Ingeniería, Gestión y Calidad de Software en proyectos informáticos, en procesos de desarrollo web y de videojuegos. Profesor Asistente de la disciplina de Ingeniería y Gestión de Software. Máster en Ciencias (2017), graduado de la IV Edición de la Maestría en Calidad de Software de la UCI. Institución actual: Centro de Entornos Interactivos 3D, Vertex, Facultad 4, UCI.

Nombre y Apellidos: Ing. Juan Carlos Miranda Corrales

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero en Ciencias Informáticas

e-mail: jcmiranda@uci.cu

Especialista A en Ciencias Informáticas, graduado en 2015, 2 años de experiencia en el desarrollo de videojuegos.

Nombre y Apellidos: Ing. Carlos Alberto Gavilla Cruz

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero en Ciencias Informáticas

e-mail: cagavilla@uci.cu

Especialista A en Ciencias Informáticas, graduado en el 2015 con 2 años de experiencia en el desarrollo de videojuegos.

Dedicatoria

Este trabajo se lo dedico a todas las personas que han tenido un rol importante en mi vida y me han impulsado a ser quien soy:

A mi familia, en especial a mis padres que siempre me han apoyado y me han impulsado a ser mejor persona en el ámbito personal y profesional.

A mis amigos, tanto con los que he compartido durante 5 años en la universidad, como los que he arrastrado de enseñanzas anteriores que me han seguido apoyando hasta el día de hoy.

De Eric M. C.

Agradecimientos

A toda mi familia por quererme, consentirme y estar al tanto de mis necesidades.

A mis padres que han sido mi modelo a seguir y guiarme durante toda mi vida.

A mis compañeros de aula con los que he compartido durante estos 5 años las alegrías y tristezas de la universidad.

A mis tutores por aconsejarme durante todo el proceso de la tesis.

A los profesores que me formaron durante estos 5 años.

De Eric M. C.

Resumen

Los juegos en Cuba todavía no alcanzan su madurez, debido a la falta de personal calificado en este sector. Actualmente pocos juegos en Cuba permiten editar los personajes, un ejemplo es “La Chivichana”. Este videojuego fue implementado por el Centro de Entornos Interactivos 3D, Vertex de la Universidad de las Ciencias Informáticas, en colaboración con el Instituto Cubano de la Industria del Arte y la Industria Cinematográfica. Sin embargo, las funciones de edición de los personajes en este videojuego de referencia eran elementales y no podían ser reutilizadas en otros desarrollos. La presente investigación tiene como objetivo desarrollar un componente para el motor de juego Unity, que permita la personalización de los personajes 3D de videojuegos implementados en el centro Vertex. La propuesta de solución permite utilizar todas las características básicas, de la edición de personajes de tipo humanoide, con buen nivel de detalle y puede ser utilizado en la mayoría de los géneros de juegos 3D. Al ser desarrollada en Unity puede ser exportada a cualquier plataforma. El proceso de desarrollo de la propuesta de solución se guía utilizando las buenas prácticas propuestas por la metodología Programación Extrema. Las características y diseño de la solución se describieron mediante las historias de usuario, requisitos no funcionales, tarjetas CRC y tareas de programación. Además, se realizaron pruebas de unidad, regresión y aceptación con el objetivo de diseñar casos de pruebas consistentes para comprobar el funcionamiento adecuado de las operaciones del componente.

Palabras clave: componente, personajes, personalización, Unity, videojuegos.

Índice contenido

Índice de figuras

Índice de tablas

Introducción

Los videojuegos son juegos electrónicos en los que interactúan una o más personas utilizando controladores (mandos), con un dispositivo que muestra imágenes de video (Baer, 1972). Desde hace unos años, el sector del videojuego está viviendo su época dorada. Los videojuegos se han convertido en una industria en continuo crecimiento que suma cada día más seguidores y demanda profesionales calificados en diferentes áreas (CICE, 2018).

Los videojuegos de los últimos 10 años han tenido un gran impacto económico y social. El primer ejemplo es el *Grand Theft auto V* que en tres días alcanzó la cifra de 800 millones de dólares, actualmente es uno de los videojuegos más vendidos de la historia. Otro ejemplo es el FIFA, videojuego de fútbol que en sus comienzos no tenía rival. Su mayor atractivo es que cada año añaden más modos de juegos, mejoran los gráficos y la jugabilidad (CICE, 2018).

Hay que tener en cuenta varios puntos a la hora de desarrollar un videojuego para que sea popular. Es importante saber lo que quiere la gente (juego de coches, fútbol, aventura, etc.). Ser original y no copiar contenidos de otro juego. El videojuego debe tener buena jugabilidad, buenos gráficos y permitir que los usuarios se identifiquen con los personajes. Tener en cuenta el precio para que sea distribuido rápidamente entre los jugadores. Los juegos online son los más populares, pues permiten interactuar con más personas y jugar con amigos (CICE, 2018).

Para muchos jugadores es una experiencia gratificante invertir horas en la creación de un personaje. Los jugadores tienden a construir versiones idealizadas de su propia figura, prefiriendo crear personajes que son más altos, musculosos o voluptuosos en comparación a la percepción del propio cuerpo, esto ocurre sobre todo cuando el jugador tiene una percepción negativa de sí mismo, ocurriendo una suerte de efecto compensatorio. Los jugadores que siguen esta dinámica reportan que se sienten más inmersos en la experiencia de juego, disfrutan más de las sesiones de juego, y están más motivados a cambiar sus conductas virtuales en contraste a quienes usan un avatar predeterminado, en pocas palabras, la personalización del avatar parecería tener un impacto importante en la experiencia de juego. La teoría de la autopercepción plantea que los seres humanos utilizan la percepción del cuerpo como referencia para inferir el estado emocional y la actitud hacia

un evento particular. Cuando un diseñador permite que los usuarios den forma a un grupo de polígonos de un avatar, indirectamente les están dando poder para cambiar la predisposición, las conductas y las actitudes de los usuarios que interactúen con el personaje (Yee, 2015).

La personalización del avatar puede ser tan sencilla como asignarle a un personaje específico algún objeto u atributo, o tan compleja como cambiar la forma física del mismo en tiempo real. Hoy en día este componente forma parte de muchos géneros de videojuegos, teniendo en común la edición de las características físicas del cuerpo y los rasgos faciales. También cuentan con la posibilidad de escoger el género (M/F), los peinados, así como el color de la piel, el pelo y los ojos.

El centro Vertex en la actualidad posee una línea de producción orientada al desarrollo de videojuegos. Ciertamente esta industria en Cuba es muy joven, aun así, se han desarrollado productos que gozan de la aprobación y el gusto de la población cubana. Uno de los objetivos trazados por el centro es lograr la aceptación de los videojuegos creados, por lo cual, el desarrollo de este componente y su adición a los videojuegos que están por desarrollarse les daría un valor agregado a los mismos poniéndolos a la par en este aspecto con los juegos realizados por otras compañías. El videojuego La Chivichana es el único que posee esta característica, pero no se tiene en cuenta su componente de personalización en otros videojuegos, debido a:

- Antigüedad de la versión de Unity con la que se desarrolló el videojuego.
- El componente de personalización no se implementó para que fuera genérico.
- El código de la programación está obsoleto.

Actualmente en el desarrollo de videojuegos del centro Vertex, para la personalización de personajes 3D implementados sobre el motor de juego Unity, existen varias limitantes, entre las que se encuentran:

- No se tienen identificadas las características comunes que pueden ser personalizadas de los personajes en los videojuegos, por lo cual no se conocen los elementos que podrían variar de estos personajes.
- Las propiedades asociadas a los personajes son implementadas específicamente desde el motor de juego sin poder reutilizarse en otros desarrollos, lo cual provoca duplicación de esfuerzo y tiempo en la programación de estas propiedades.

Teniendo en cuenta la situación problemática antes descrita, se identifica como **problema de la investigación:** ¿Cómo contribuir a la personalización de los personajes 3D implementados sobre el motor de juego Unity, de videojuegos desarrollados en el centro Vertex?

A partir del problema planteado en la investigación, se identifica como **objeto de estudio**: el proceso de desarrollo de videojuegos sobre el motor de juego Unity.

Para dar solución al problema de investigación planteado, se define como **objetivo general:** Desarrollar un componente para el motor de juego Unity, que permita la personalización de los personajes 3D de videojuegos implementados en el centro Vertex.

Dentro del objeto de estudio de la investigación se precisa como **campo de acción**: el proceso de personalización de los personajes 3D desarrollados sobre el motor de juego Unity.

Para el cumplimiento de la investigación serán desarrolladas las siguientes tareas:

1. Elaboración del marco teórico de la investigación a través del estudio del estado del arte para una mejor comprensión de la investigación.
2. Identificación de los elementos personalizables en videojuegos para lograr un componente que sea genérico.
3. Selección de la metodología, herramientas de software a utilizar y lenguajes de programación para el desarrollo del componente.
4. Generación de los artefactos que corresponden a las disciplinas definidas por la metodología seleccionada.
5. Implementación del componente para la personalización de personajes 3D en videojuegos desarrollados sobre el motor de juego Unity.
6. Validación del componente desarrollado a partir de las técnicas y métodos científicos definidos para valorar la propuesta de solución.

Como métodos de investigación científica se emplearon los siguientes:

1. **Análisis-síntesis:** se utiliza para descomponer el problema, que es la necesidad del centro Vertex de un editor de personajes en el motor de juegos Unity 3D, en partes concretas, analizar la información obtenida mediante la metodología seleccionada y concretar con la integración de los artefactos ingenieriles.

2. **Histórico-lógico:** se utiliza para precisar las tendencias que se han presentado acerca de los editores de personajes en los juegos actuales, en cuanto a las distintas opciones de edición, la forma en que es realizada y como el usuario interactúa con las herramientas de la interfaz. Permite analizar el estado del arte existente sobre los editores de personajes actuales sobre el motor de juegos Unity 3D.
3. **Modelación:** se emplea para modelar toda la estructura del componente de edición de personajes en cada iteración de la metodología asociada. Los diagramas y modelos posibilitan una mejor comprensión de las funcionalidades del editor de personajes, permitiendo entender las relaciones entre componentes.

Métodos empíricos:

Observación: se utiliza como método de referencia al observar distintos juegos con edición de personajes que sirvieron de análisis. Esto permitió definir las características fundamentales del componente de edición de personajes.

Estructura capitular:

- **Capítulo 1:** Marco teórico referencial sobre el desarrollo de videojuegos y funciones de edición de personajes. En este capítulo se hace un estudio de las tecnologías a utilizar, así como de los sistemas homólogos de edición de personajes.
- **Capítulo 2:** Características y diseño del componente para la personalización de personajes 3D en videojuegos desarrollados para Unity. En este capítulo se extraen los requisitos funcionales, en forma de historias de usuario, que sirven de guía durante todo el proyecto. También se calcula el tiempo de implementación de cada requisito para crear un plan de entrega por iteraciones. Por último, se definen las responsabilidades de cada clase, así como la interacción entre ellas, teniendo en cuenta la arquitectura y los patrones de diseño escogidos.
- **Capítulo 3:** Implementación y prueba del componente para la personalización de personajes 3D en videojuegos desarrollados para Unity. En este capítulo se definen las tareas de ingeniería o programación para cada historia de usuario y se realizan pruebas al software para extraer y corregir las no conformidades.

CAPÍTULO 1: MARCO TEÓRICO REFERENCIAL SOBRE EL DESARROLLO DE VIDEOJUEGOS Y FUNCIONES DE EDICIÓN DE PERSONAJES

1.1. Introducción

En este capítulo se hace un acercamiento a los videojuegos, así como a los motores de videojuegos desde el punto de vista conceptual. Se aborda sobre las herramientas y tecnologías a utilizar estableciendo una comparación entre ellas para escoger la que más se ajusta a las necesidades del proyecto. También se hace una investigación sobre distintas aplicaciones relacionadas con la edición de personajes. Se muestran las características de las distintas metodologías de desarrollo de software y se explica la selección de *Extreme Programming* (XP).

1.2. Videojuegos

Se considera juego a todas las actividades realizadas con fines recreativos utilizando una prueba física o mental bajo determinadas reglas. El videojuego es un juego electrónico en el que una o más personas interactúan utilizando dispositivos electrónicos (Julián Pérez Porto, s.f.) (Baer, 1972).

1.2.1. Géneros de videojuegos

Los videojuegos se empezaron a comercializar en la década de 1970 utilizando máquinas tragamonedas en centros recreativos. Más tarde se crearon las consolas, permitiendo que los usuarios disfrutaran los videojuegos desde la comodidad de sus hogares. En la década de 1980 la industria de los videojuegos estaba dominada por Atari, Nintendo y SEGA (es una empresa japonesa desarrolladora de software y hardware en el campo del entretenimiento 'videojuegos'); además, empezaron a aparecer ordenadores personales con capacidades gráficas similar a las consolas. Hoy en día existen miles de títulos de videojuegos respaldados por cientos de empresas donde existe una elevada competencia para obtener usuarios (Retro informática, 2018) (Stack, 2005).

Los videojuegos actuales con más fama recrean complejos entornos y situaciones virtuales donde el jugador controla a uno o varios personajes para cumplir varios objetivos dentro de unas reglas determinadas.

Entre los requisitos que más usuarios atrae están: los gráficos (que tan real se ve el juego), la trama (historia de fondo), las mecánicas (reglas que determinan lo que se puede hacer en el juego), la libertad (darle control al usuario sobre el juego, permitirle ir a donde desee, interactuar o editar libremente con los objetos del juego) (La Vanguardia, 2012).

En este trabajo se escogió crear un componente de juego que permita editar los personajes, para que los usuarios puedan sentirse identificados y se sienten más inmersos en la experiencia de juego. Este componente está enfocado principalmente en juegos de rol o “role-playing game” (RPG), pero también puede ser usado en cualquier género de juego 3D, que necesite editar personajes humanoides 3D.

1.2.2. Procedimiento de edición de personajes en videojuegos

Los avatares son la representación virtual del jugador con el cual se va a interactuar durante cientos de horas. Por esto, se hace necesario permitir su edición para que el jugador se sienta identificado y disfrute su experiencia de juego (Tejeiro R., 2014) (Yee, 2015).

Se estudiaron diez (10) juegos con edición de personajes para sacar elementos comunes.

Los videojuegos estudiados son:

- **Arche Age:**

Ilustración 1: Arche Age (Fuente: (XL games, 2013)).

- **Archlord 2:**

Ilustración 2: Archlord 2 (Fuente: (Webzen, 2014)).

- **Bless Online:**

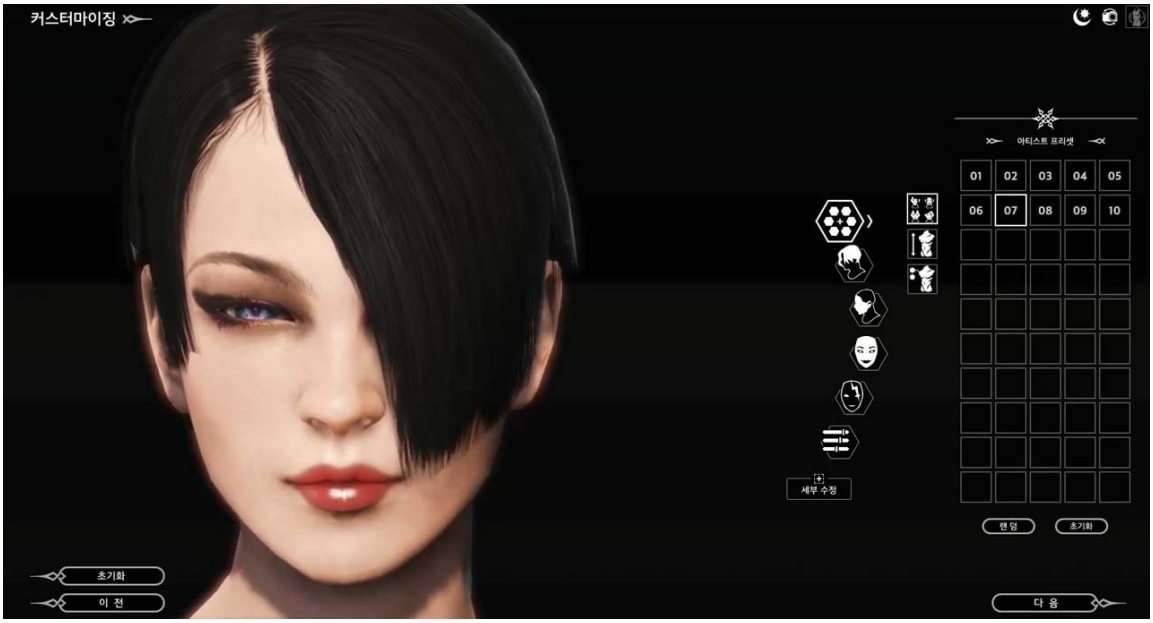


Ilustración 3: Bless Online (Fuente: (Neowiz, Neowiz Bless Studio, 2018)).

- **Destiny:**

Ilustración 4: Destiny (Fuente: (Bungie Studios, Vicarious Visions, High Moon Studios, Radical Entertainment, 2014))

- **Dragon Age 3 Inquisition:**

Ilustración 5: Dragon Age 3 Inquisition (Fuente: (BioWare, 2014)).

- **Dragons Prophet:**

Ilustración 6: Dragons Prophet (Fuente: (Runewaker Entertainment, Runewaker, 2013)).

- **EVE:**

Ilustración 7: EVE (Fuente: (CCP Games, 2016)).

- **Kingdom Under Fire II:**

Ilustración 8: Kingdom Under Fire II (Fuente: (Blueside, Phantagram, 2015)).

- **Scarlet Blade:**

Ilustración 9: Scarlet Blade (Fuente: (Liveplex, 2012)).

- **WildStar:**

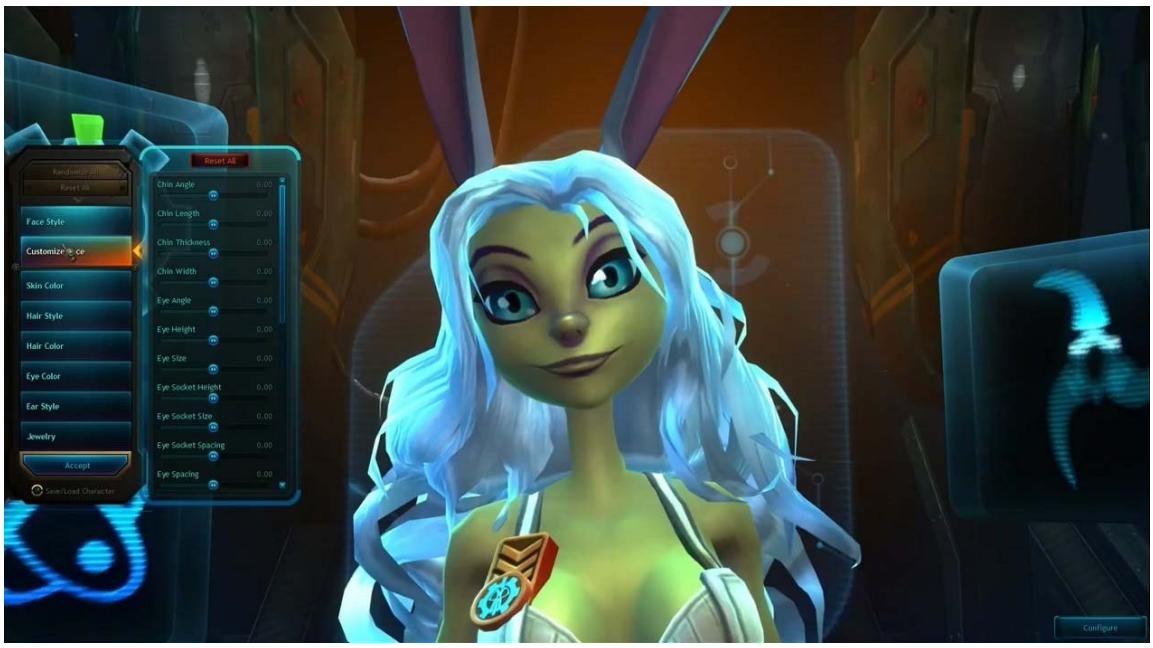


Ilustración 10: WildStar (Fuente: (Carbine Studios, 2014)).

En todos se puede seleccionar el sexo, el modelo base, el peinado, el color de pelo, la piel, la forma y el color de los ojos, la forma de la nariz, la forma de la boca y la forma de las orejas. Siempre se hace énfasis en el rostro y en segundo lugar en el torso, generalmente se enfocan en la posición de los hombros, el tamaño del pecho y el ancho de la cintura.

1.3. Herramientas para la personalización de personajes

Actualmente existen programas que permiten a los desarrolladores crear personajes realistas para posteriormente insertarlos en el juego. Uno de estos programas es el '*MakeHuman*', el cual es libre pero no se acopla directamente al juego, es necesario importar sus personajes en la etapa de desarrollo del juego. Unity ya posee un componente propietario, '*Ubrin Character Costumization*', con el cual es necesario pagar licencia, por tanto se hace necesario implementar un componente compatible con esta herramienta para lograr la independencia tecnológica, que pueda ser utilizado para los personajes, de tipo humanoides, de los juegos desarrollados en el centro Vertex (Unity asset store, 2016).

Ilustración 11: Ubrin Character Costumization (Fuente: (Unity 3D share wordpress, 2014)).

1.4. Metodología de desarrollo de la propuesta de solución

"Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en subfases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo" Avison y Fitzgerald (1995).

Tabla 1: Comparación de metodologías (Fuente: (J, 2005)).

Metodologías ágiles	Metodologías tradicionales
Se basan en heurísticas provenientes de prácticas de producción de código.	Se basan en normas provenientes de estándares seguidos por el entorno de desarrollo.
Preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente por el equipo.	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso muy controlado, numerosas normas.

Contrato flexible e incluso inexistente.	Contrato prefijado.
El cliente es parte del desarrollo.	Cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10).	Grupos grandes.
Pocos artefactos.	Más artefactos.
Menor énfasis en la arquitectura del software.	La arquitectura del software es esencial.

Teniendo en cuenta la comparación antes expuesta se decide seleccionar un enfoque ágil, debido a que, el equipo está compuesto por un desarrollador y el cliente (Vertex) forma parte del equipo de desarrollo. Además, se cuenta con la experiencia pertinente para el desarrollo de aplicaciones sobre la infraestructura tecnológica necesaria y se conoce claramente el dominio del problema a resolver.

Entre las metodologías ágiles más utilizadas para el desarrollo de pequeños productos de software se encuentra: Programación extrema (extreme programming, XP), Crystal Methodologies y SCRUM.

Tabla 2: Síntesis metodologías agiles de desarrollo con sus características (Fuente: (Lina María Montoya Suarez, 2017)).

Metodología	Características
Programación extrema (extreme programming, XP).	Metodología basada en prueba y error. Cliente bien definido. Los requisitos pueden y van a cambiar. Grupo pequeño y muy integrado (máximo 12 personas). Equipo con formación elevada y capacidad de aprender. Fundamentada en valores. Expresada en 12 prácticas. Implementa compatibilidad y usabilidad con otras metodologías.

	Está orientada hacia quien produce y usa el software. Reduce el costo del cambio en todas las etapas del ciclo de vida del sistema. Programación organizada. Reduce la taza de errores.
Crystal Methodologies.	Entregas frecuentes, en base a un ciclo de vida iterativo e incremental. Cuantas más personas estén implicadas, más grande debe ser la metodología. Entorno técnico con pruebas automatizadas, gestión de la configuración e integración continua, Si el proyecto tiene mucha densidad, un error no detectado puede ser crítico. El aumento de tamaño o densidad añade un costo considerable al proyecto. La forma más eficaz de comunicación es la interactiva (cara a cara). Tamaño de un equipo (número de componentes). Equipo: • Claro es para equipos de hasta 8 personas o menos. • Amarillo para equipos entre 10 a 20 personas. • Naranja para equipos entre 20 a 50 personas. • Roja para equipos entre 50 a 100 personas. • Azul para equipos entre 100 a 200 personas. Comunicación entre los componentes. Distintas políticas a seguir. Mejora reflexiva.
SCRUM.	Permite la gestión de diferentes equipos industriales.

	Aplicación de las metodologías ágiles en los procesos de calidad, entregando resultados óptimos. Su adopción en prácticas orientadas a planes como CMMI (Integración de modelos de madurez de capacidades o *Capability Maturity Model Integration* es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software) o PSP/TSP (*Personal Software Process* o Proceso Personal de Software, es un conjunto de prácticas disciplinadas para la gestión del tiempo y mejora de la productividad personal de los programadores o ingenieros de software, en tareas de desarrollo y mantenimiento de sistemas, mediante el seguimiento del desempeño predicho frente al desempeño real)/(*Team Software Process* proporciona un marco de trabajo de procesos definidos que está diseñado para ayudarle a equipos de gerentes e ingenieros a organizar y producir proyectos de software de gran escala). La productividad y competitividad de las empresas de software al implementar este tipo de metodología. El aporte a la industria a través de las buenas prácticas. El aporte a la gestión de proyectos a través del proceso. Aporte efectivo dentro de la gerencia de proyectos de software. La competitividad en el desarrollo de productos y proyectos industriales a través de la implementación de esta metodología ágil.

Teniendo en cuenta las características de XP, la misma es más factible porque el tiempo de desarrollo es corto, de aproximadamente 4 meses; el cliente está bien definido: Vertex; los requisitos para editar el personaje pueden variar en dependencia de las necesidades del videojuego; y es un equipo pequeño, compuesto por una persona.

1.4.1. Extreme Programming (XP)

Es una metodología ágil que promueve el aprendizaje de los desarrolladores. Se basa en la comunicación fluida entre el cliente y el equipo de desarrollo. Propone simplicidad en las

soluciones implementadas y se enfrenta a cambios continuos en los requisitos de la aplicación. XP es una metodología ligera basada en Historias de Usuario (Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales).

Características fundamentales de XP:

- No se planifica o diseña para un futuro distante, estas actividades se realizan durante el proceso de desarrollo.
- Se basa en prueba y error.
- Pocos artefactos.
- Pocos roles.
- Se aplica en grupos pequeños.
- Pruebas continuas.
- Código compartido con el equipo.

El ciclo de vida ideal de la metodología XP consta de seis fases, en este proyecto se usará la versión reducida que consta de cuatro partes:

- **Planificación:** En esta fase se recogen los requisitos de los clientes, se establecen las prioridades para los requisitos escogidos, se realiza la estimación del esfuerzo necesario para cada tarea, se elaboran las Historias de Usuario (HU), el plan de iteraciones y el plan de entrega.
- **Diseño:** En esta fase se describe la arquitectura utilizada, se crea el diagrama de paquetes, se realiza la descripción de las tarjetas de Contenido, Responsabilidad y Colaboración (CRC), se describen los patrones de diseño GRASP (patrones generales de software para asignación de responsabilidades) y GOF (‘*Gang Of Four*’) utilizados.
- **Desarrollo:** En esta fase se desglosan las operaciones en las que se pueden implementar cada HU mediante las tareas de ingeniería o programación.
- **Prueba:** Se realizan pruebas unitarias, aceptación y regresión.

1.5. Herramientas y tecnologías

Para diseñar y desarrollar un software es necesario utilizar diversas herramientas informáticas de las cuales se hace referencia a continuación:

1.5.1. Herramienta de modelado

Visual Paradigm es una herramienta CASE (Ingeniería de Software Asistida por Computación). La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (Visual Paradigm, 2019).

Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta privada disponible en varias ediciones, cada una destinada a satisfacer diferentes necesidades: *Enterprise, Professional, Community, Standard, Modeler y Personal*. Existe una alternativa libre y gratuita de este software, la versión *Visual Paradigm* UML 6.4 *Community Edition (Community Edition*). Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos (Pressman, 2010).

Se escogió *Visual Paradigm Suite Windows* 5.0sp1, debido a que es la versión disponible en la universidad, esta usa la versión UML 6.4.

1.5.2. Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML) fue creado para forjar un lenguaje de modelado visual común y semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento. UML tiene aplicaciones más allá del desarrollo de software, por ejemplo, en el flujo de procesos en la fabricación (Temas especiales de computación, 2000) (Lucidchart, 2018).

Es comparable a los planos usados en otros campos y consiste en diferentes tipos de diagramas. En general, los diagramas UML describen los límites, la estructura y el comportamiento del sistema y los objetos que contiene (Lucidchart, 2018).

UML no es un lenguaje de programación, pero existen herramientas que se pueden usar para generar código en diversos lenguajes usando los diagramas UML. UML guarda una relación directa (Lucidchart, 2018).

Se escogió el lenguaje UML por ser el utilizado por el *Visual Paradigm*.

1.5.3. Herramienta de diseño

La interfaz fue desarrollada en Unity utilizando elementos 2D creados en Adobe Photoshop CC 2018 19.1.5.61161.

Adobe Photoshop Es un programa de edición de imágenes comercial y multiplataforma desarrollado por Adobe, sirve para editar y retocar imágenes de todo tipo. Es una de las mejores y más completas herramientas en su categoría, siendo utilizada por profesionales dentro del ámbito de la imagen como fotógrafos, diseñadores web o diseñadores gráficos, entre otros (Guzmán, 2017).

Su primera versión fue lanzada en 1990 para Mac, y la de Windows no vio la luz hasta 1992. Desde entonces se ha convertido en la opción número uno para profesionales del diseño y aficionados al retoque fotográfico para editar imágenes desde la PC. Junto a otros conocidos programas de diseño gráfico, web y edición de vídeos, como Illustrator, InDesign, Auditon, After Effects y Premier, forma parte del paquete de aplicaciones Adobe Creative Cloud (Guzmán, 2017).

Es compatible con los archivos de imágenes más utilizados y utiliza una edición basada en capas, lo que le permite alterar al mínimo detalle cualquier matiz de una fotografía (Guzmán, 2017).

1.5.4. Motor de juego

El motor de videojuegos es una serie de rutinas de programación que permite diseñar y crear un mundo virtual complejo. Se encarga de renderizar los gráficos 2D y 3D, los efectos físicos, sonidos, animación, programación y administración de los recursos de la PC (Gamer Dic, 2013) (Observatorio del gabinete de tele-educación, 2018) (Game Career Guide, 2008).

Existen diversos motores de juego, tales como: Blender Game Engine, Unity 3D, Unreal Engine y Godot (Game Career Guide, 2008).

1.5.5. Entorno de desarrollo

1.5.5.1. Unity 3D

Es un motor gráfico 3D para PC y Mac que viene empaquetado como una herramienta para crear juegos, aplicaciones interactivas, visualizaciones y animaciones en 3D y tiempo real (Unity 3D, 2018).

El componente se va a desarrollar en Unity 3D 2017.2.0f3, debido a que se escogió por el centro Vertex, producto a que es una potente herramienta que permite crear juegos multiplataformas. Está enfocada en bloques de construcción (*assets*) que permite desarrollar componentes integrables a la interfaz de usuario. Esto facilita la reutilización de código propiciando la optimización del tiempo de desarrollo. Además, posee una amplia documentación (Fenrir, 2018).

1.5.5.2. Microsoft Visual Studio

Es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc., a lo cual hay que sumarle las nuevas capacidades online bajo Windows Azure en forma del editor Monaco (Microsoft, 2017).

Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así, se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y consolas, entre otros (Microsoft, 2017).

Se escogió Microsoft Visual Studio 2017 por ser una potente herramienta de desarrollo que se integra a Unity 3D 2017.2.0f3.

1.5.6. Lenguaje de programación

Un *script* debe estar vinculado con un *GameObject* (nombre de los objetos en Unity) en la escena para poder ser invocado por Unity. Los *scripts* se escriben en un lenguaje especial que Unity puede entender. Este lenguaje permite implementar las funcionalidades en el motor de videojuego. El lenguaje que se usa en Unity se denomina C#. Todos los lenguajes que Unity emplea son lenguajes de *scripting* orientados a objetos. Al igual que cualquier lenguaje, los lenguajes de *scripting* tienen sintaxis, o partes de diálogo, y las partes

principales se denominan variables, funciones y clases (La revista informática .com, 2015) (Unity 3D, 2018).

Se utiliza la última versión estable de C# 7.2 que se lanzó en 2015.

1.6. Conclusiones parciales

A raíz del estudio del estado del arte de los juegos y aplicaciones que permiten la edición de personajes, se puede concluir que:

- Es necesario crear un componente de personalización de personajes genérico que permita editar el sexo, la tonalidad de la piel, el pelo, los ojos, los rasgos físicos y faciales.
- La metodología XP permite un desarrollo ágil del proyecto, facilitando el trabajo del desarrollador.
- El entorno de desarrollo Unity 3D, como motor de videojuegos, y Microsoft Visual Studio, como IDE de programación, permiten la implementación de la solución teniendo en cuenta que son potentes herramientas de desarrollo que se integran.

CAPÍTULO 2: CARACTERÍSTICAS Y DISEÑO DEL COMPONENTE PARA LA PERSONALIZACIÓN DE PERSONAJES 3D EN VIDEOJUEGOS DESARROLLADOS PARA UNITY

2.1. Introducción

En este capítulo se realizan las especificaciones necesarias para la propuesta de solución. Se brinda una descripción del proceso de creación del componente. Se abordan las fases de planificación y diseño de la metodología XP.

En la fase de planificación se extraen los requisitos funcionales en forma de HU, a los cuales se le determinan la importancia para el negocio y el riesgo de implementación. A cada HU se le diseñan prototipos de interfaces y se estima el tiempo de duración para su implantación. Se le asigna a cada HU una iteración y se calcula la duración de cada iteración para crear un plan de entregas.

En la fase de diseño se escoge y describe la arquitectura a utilizar mediante el uso del diagrama de paquete. Se evidencian las funcionalidades de cada clase y sus relaciones mediante el uso de tarjetas CRC. También se especifica el uso de los patrones GRASP en cada clase y se escoge el patrón GoF más adecuado para efectuar la programación de la aplicación.

2.2. Propuesta de solución

Con la presente investigación se propone realizar un componente de edición de personajes, que pueda ser incorporado en los juegos desarrollados por el centro Vertex para la edición de personajes humanoides. El componente deberá estar hecho en Unity 3D para ser compatible con los juegos desarrollados en Vertex.

El componente permitirá armar personajes humanoides, seleccionando el sexo, el modelo de cuerpo, los peinados, el tono de la piel, el color del pelo, la textura de los ojos. Después de seleccionar las características base, el componente deberá permitir moldear los rasgos faciales y la constitución del cuerpo. Además, deberá permitir acercar la cámara y girar el personaje para facilitar su edición. Finalmente deberá guardar los cambios para poder ser usado en el juego.

Para moldear los rasgos faciales y físicos se escogió utilizar “BlendShape”, la cual es una herramienta que ofrece el software Blender para editar los vértices (los objetos 3D están formado por la unión de varios vértices) de los modelos 3D (AWS, 2019) (Blender, 2018).

Las “BlendShape” se colocan en el modelo cuando está terminado y después pueden ser modificadas en Unity. Para no tener problemas de incompatibilidad con las “BlendShape” se suprimieron las tildes en los nombres, en todos los modelos se deben de utilizar con exactamente los mismos nombres escogidos en este trabajo y siempre deben de tener el mismo orden. Se deben crear dos “BlendShape” por cada punto a editar, una con el sufijo Min (que indica el valor mínimo) y otra con el sufijo Max (que indica el valor máximo), haciendo un total de 48 “BlendShape” por modelo. A continuación, se muestra la estructura de las “BlendShape” escogida:

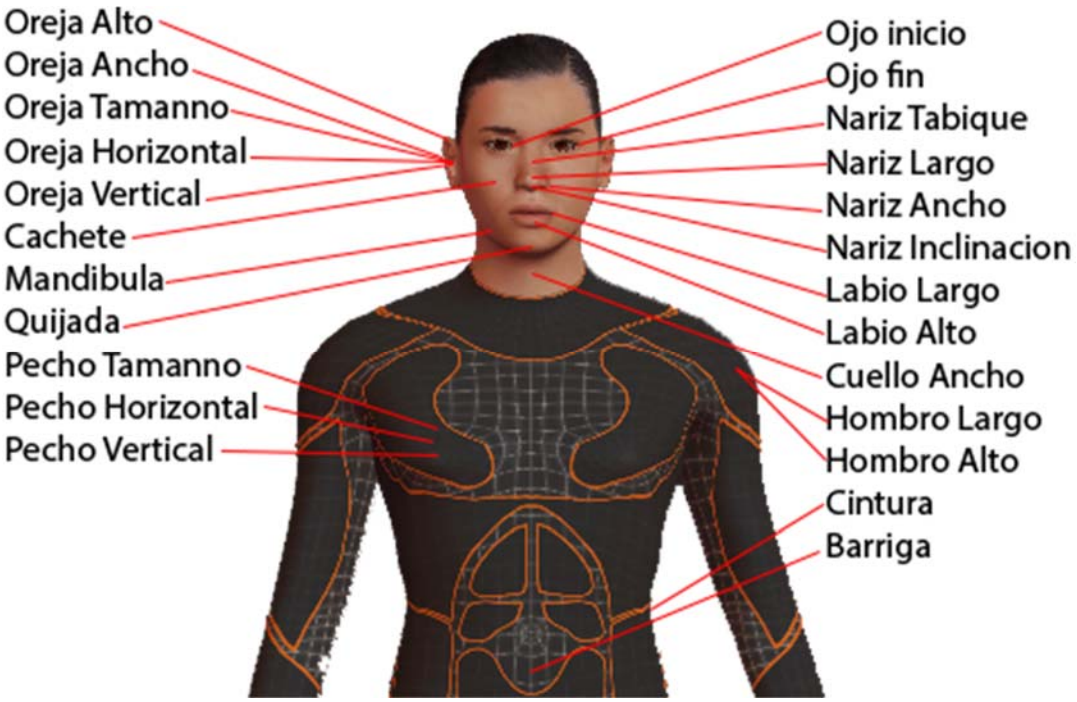


Ilustración 12: Zonas editables del avatar (Fuente: Elaboración propia).

Es necesario aclarar que el componente no incluye los modelos 3D de los avatares o las texturas, es necesario cargar los modelos de los diseñadores 3D con las características descritas anteriormente. Para cargar los modelos deben ser arrastrados hasta “*Cuerpo Models H/M*” dentro de la clase “*CharacterCustomization*”, utilizando la interfaz de Unity. De igual manera se agregan los modelos de pelo (en “*PeloModels*”), el color de piel (“*PielColor*”), el color del pelo (“*PeloColor*”) y el color de los ojos (“*OjoColor*”).

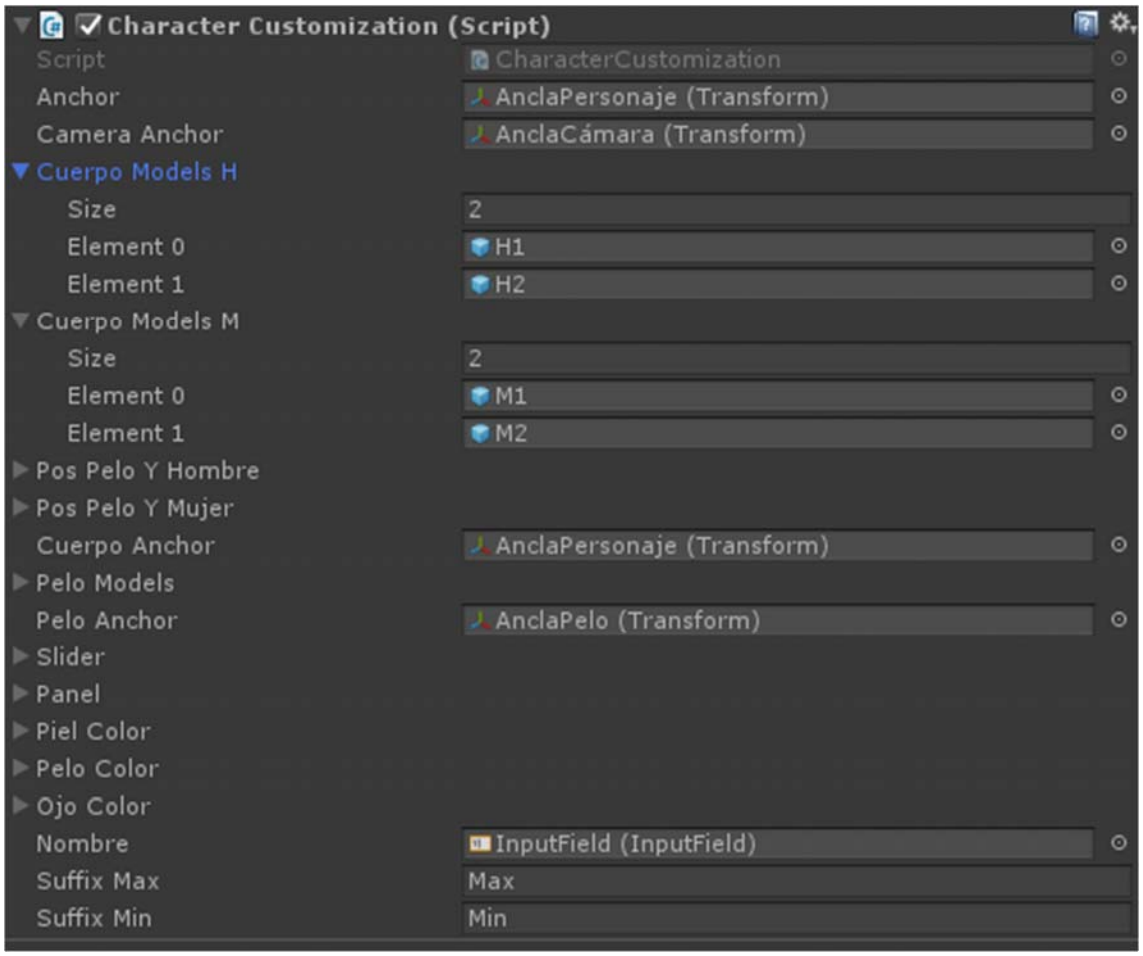


Ilustración 13: Agregar modelos de avatares (Fuente: Elaboración propia).

Como la interfaz gráfica del componente depende del juego donde sea utilizado, las texturas de los paneles, botones y "s*lider*" (barras deslizantes) deben ser agregadas, por un diseñador, en correspondencia a la ambientación del juego.

Para agregar más características, faciales o físicas, editables se le deben agregar las "*BlendShape*" correspondientes a los modelos 3D. Para editarlas en el componente se debe clonar una de las "*sliders*" y modificar el nombre y en el campo "*BlendShapeSlider*", se debe colocar el mismo nombre de la "*BlendShape*" agregada al avatar.

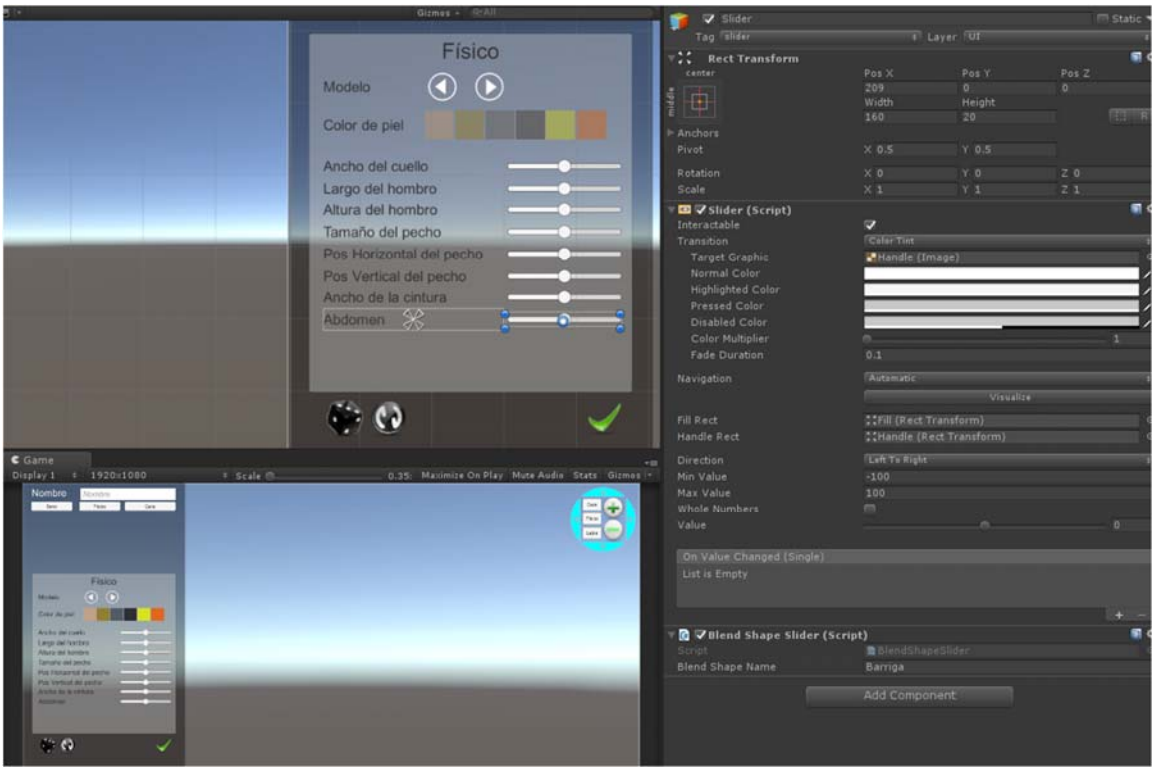


Ilustración 14: Agregar BlendShape (Fuente: Elaboración propia).

Para utilizar el componente se debe cargar la escena en donde está implementado. Al terminar la personalización y seleccionar acertar, se guarda la configuración en la clase estática: “Personaje” y se carga la siguiente escena del juego.

2.3. Fase de Planificación

En esta fase se realizan entrevistas con los clientes para definir las HU, determinar los requisitos funcionales y los programadores definen el alcance del proyecto. También es la etapa donde el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas del proyecto (Patricio Letelier, 2015).

El nivel de prioridad para el negocio de los requisitos obtenidos, en la entrevista con el cliente, se dividen en:

- **Alto:** Funcionalidades de las cuales el sistema depende completamente para su funcionamiento.
- **Medio:** Funcionalidades que no son primordiales pero otras funciones dependen de estas para poder ejecutarse.

- **Bajo:** Funcionalidades de las cuales no depende ninguna otra función y la aplicación puede ejecutarse correctamente sin ella.

En la siguiente gráfica se muestra la prioridad para el negocio asignada a los requisitos funcionales:

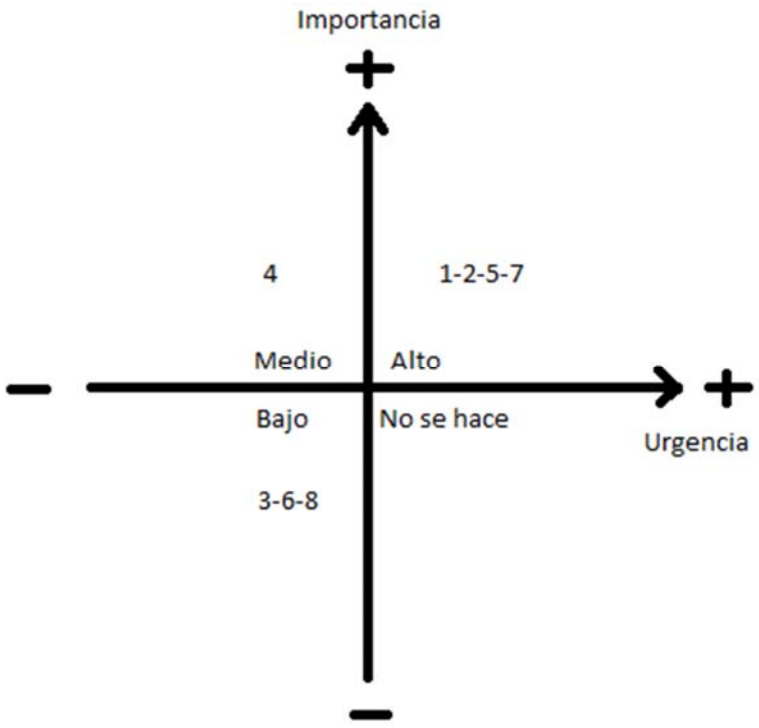


Ilustración 15: Prioridad de los requisitos para el negocio (Fuente: (Weigers, 2003)).

El riesgo en desarrollo es el nivel de desajuste que pueden presentar los requisitos a la hora de ser implementados. Los niveles son:

- **Alto:** Funcionalidades que dependen de herramientas externas que no sea completamente estable o que dependa de otras funcionalidades con igual nivel de riesgo. Las HU 4 y 5 dependen de que hayan sido colocadas correctamente las "blendshape" en el modelo del personaje. Las HU 6, 7 y 8 dependen de las dos anteriores.
- **Medio:** Funcionalidades que dependen de otras funcionalidades, pero no dependen de herramientas externas. La HU 3 depende de la HU 1.
- **Bajo:** Funcionalidades que solo dependen de su código asignado. Las HU 1 y 2 no dependen de ninguna funcionalidad para trabajar.

2.3.1. Historias de usuario

Las HU permiten especificar los requisitos del software, describen las características con las que el producto debe contar (requisitos funcionales), debe permitir a los programadores su implementación en el menor tiempo posible. Las HU deben contar con una serie de pruebas de unidad (acoplamiento con otros componentes ya implementados del software) para asegurar el correcto funcionamiento de los componentes. Al terminar cada HU se les debe mostrar a los clientes para que la acepten o rechacen.

En esta fase se identificaron ocho (8) historias de usuario que responden a las funcionalidades solicitadas por el cliente (Sintya Milena, 2016).

Tabla 3: HU Mostrar panel de configuración (Fuente: Elaboración propia).

HISTORIA DE USUARIO	
Número: 1.	**Usuario:** Jugador.
Nombre: Mostrar panel de configuración.	
Prioridad en negocio: Alto.	**Riesgo en desarrollo:** Bajo.
Puntos estimados: 1.	**Iteración asignada:** 1.
Programador responsable: Eric Mañalich Cossio.	
Descripción: Permite a los jugadores modificar las características de los personajes.	
Observaciones: Las texturas del fondo, el panel, los botones y los "*sliders*" de la interfaz serán agregadas por los desarrolladores del juego donde sea utilizado el componente.	
Prototipo de interfaz gráfica:	

Tabla 4: HU Configurar género del personaje (Fuente: Elaboración propia).

HISTORIA DE USUARIO	
Número: 2.	**Usuario:** Jugador.
Nombre: Seleccionar género del personaje.	
Prioridad en negocio: Alto.	**Riesgo en desarrollo:** Bajo.
Puntos estimados: 1.	**Iteración asignada:** 1.
Programador responsable: Eric Mañalich Cossio.	
Descripción: Permite a los jugadores escoger el género del modelo deseado.	
Observaciones: A partir del modelo escogido se efectuarán todos los cambios.	
Prototipo de interfaz gráfica:	

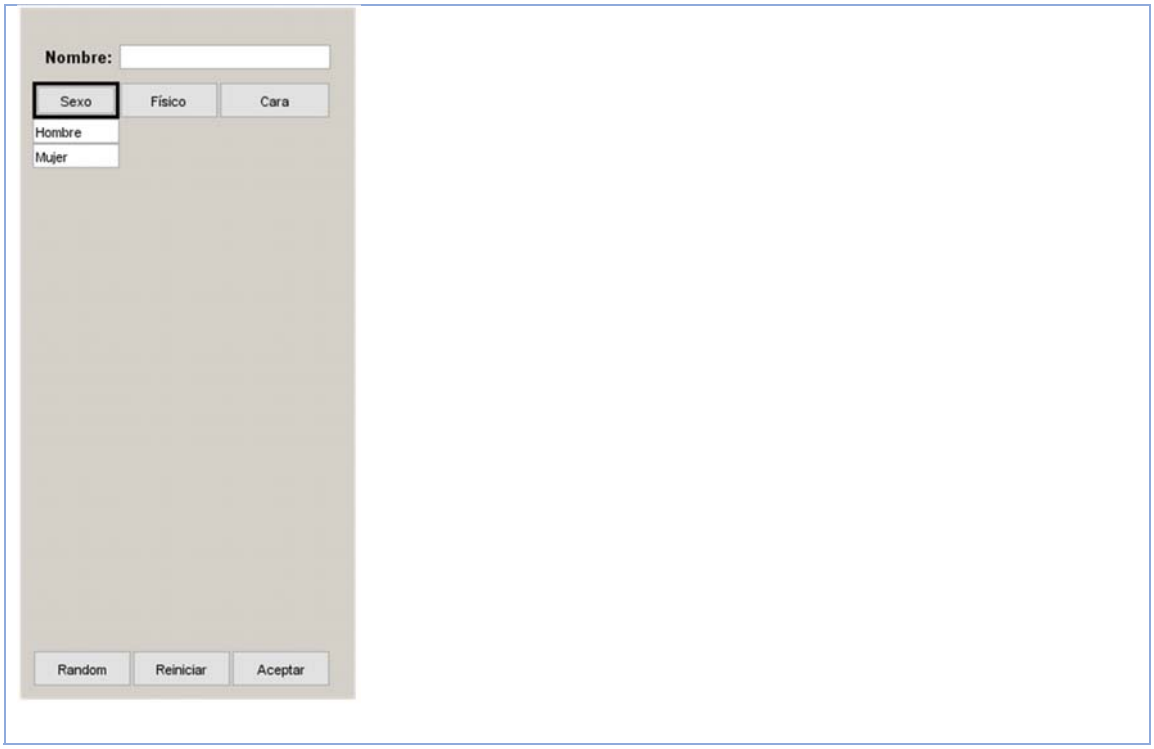


Tabla 5: HU Mover proyección del personaje (Fuente: Elaboración propia).

HISTORIA DE USUARIO	
Número: 3.	**Usuario:** Jugador.
Nombre: Mover proyección del personaje.	
Prioridad en negocio: Bajo.	**Riesgo en desarrollo:** Medio.
Puntos estimados: 1.	**Iteración asignada:** 1.
Programador responsable: Eric Mañalich Cossio.	
Descripción: Permite a los jugadores mover la cámara para ver mejor al personaje.	
Observaciones: Se debe configurar la cámara automáticamente para enfocar en la región donde se está editando al personaje. Rotar el avatar se debe efectuar mediante el movimiento del ratón, de forma horizontal, al hacer clic en la pantalla.	
Prototipo de interfaz gráfica:	

Tabla 6: HU Configurar aspectos físicos del personaje (Fuente: Elaboración propia).

HISTORIA DE USUARIO	
Número: 4.	**Usuario:** Jugador.
Nombre: Configurar aspectos físicos del personaje.	
Prioridad en negocio: Medio.	**Riesgo en desarrollo:** Alto.
Puntos estimados: 3.	**Iteración asignada:** 1.
Programador responsable: Eric Mañalich Cossio.	
Descripción: Permite a los jugadores seleccionar el modelo base y editar características del cuerpo de los personajes como el color de la piel; el ancho, al alto y el largo de los hombros; el tamaño de los pechos; el ancho del cuello; el ancho de la cintura y el tamaño del abdomen**.**	
Observaciones: No importa que el modelo tenga ropa o esté desnudo**.**	
Prototipo de interfaz gráfica:	

Tabla 7: HU Configurar aspectos faciales del personaje (Fuente: Elaboración propia).

HISTORIA DE USUARIO	
Número: 5.	**Usuario:** Jugador.
Nombre: Configurar aspectos faciales del personaje.	
Prioridad en negocio: Alto.	**Riesgo en desarrollo:** Alto.
Puntos estimados: 3.	**Iteración asignada:** 2.
Programador responsable: Eric Mañalich Cossio.	
Descripción: Permite a los jugadores editar características faciales como la nariz, la boca, los ojos y su color, la mandíbula, la quijada, los cachetes, las orejas, el pelo y su color.	
Observaciones: El color del pelo debe ser el mismo en el peinado y las cejas.	
Prototipo de interfaz gráfica:	

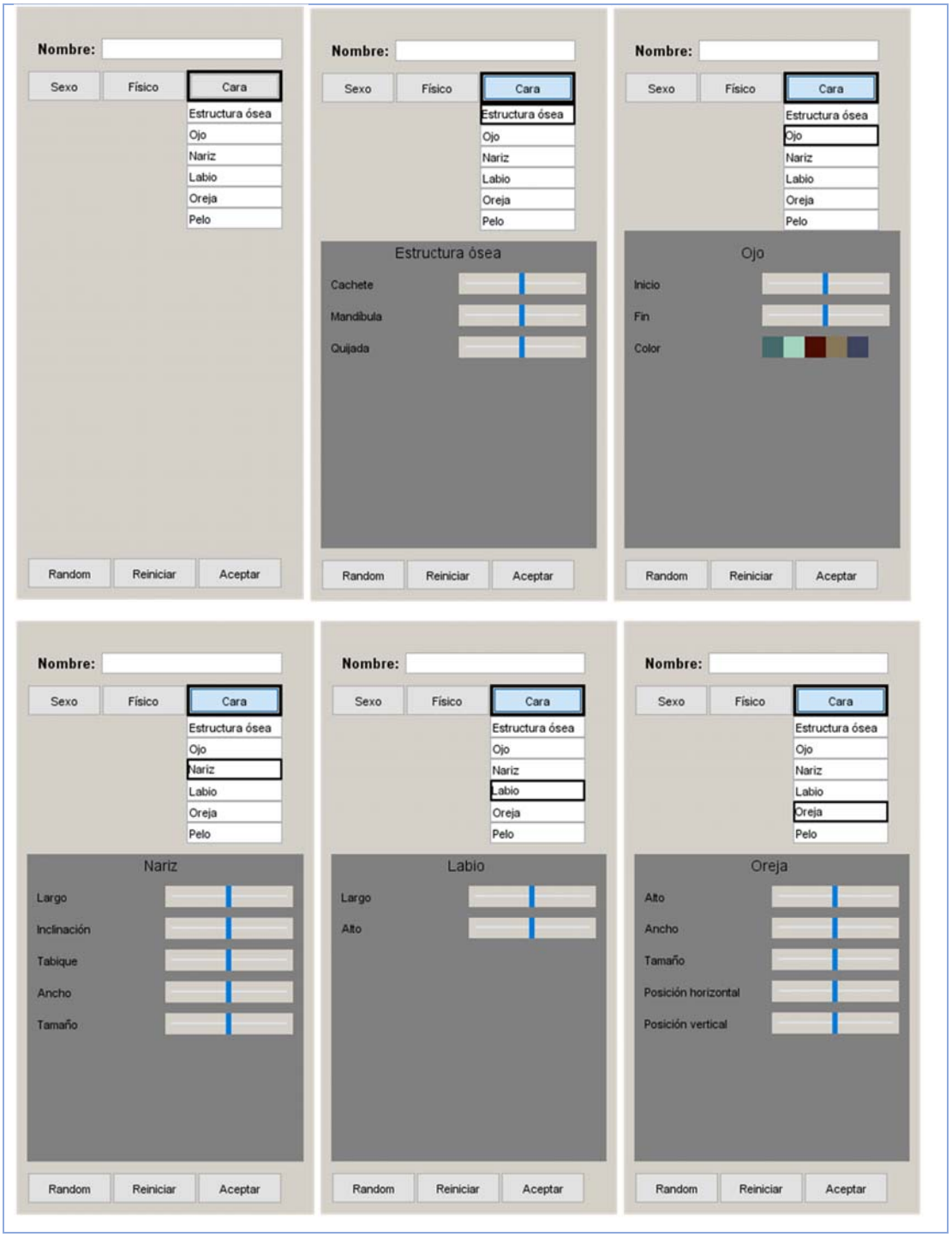

Nombre:
Sexo
Físico
Cara
Estructura ósea
Ojo
Nariz
Labio
Oreja
Pelo
Random
Reiniciar
Aceptar
Estructura ósea
Cachete
Mandíbula
Quijada
Ojo
Inicio
Fin
Color
Nariz
Largo
Inclinación
Tabique
Ancho
Tamaño
Labio
Largo
Alto
Oreja
Alto
Ancho
Tamaño
Posición horizontal
Posición vertical

Tabla 8: HU Configurar aspectos del personaje aleatoriamente (Fuente: Elaboración propia).

HISTORIA DE USUARIO	
Número: 6.	**Usuario:** Jugador.
Nombre: Configurar aspectos del personaje aleatoriamente.	
Prioridad en negocio: Bajo.	**Riesgo en desarrollo:** Alto.
Puntos estimados: 2.	**Iteración asignada:** 2.
Programador responsable: Eric Mañalich Cossio.	
Descripción: Permite al usuario generar una configuración aleatoria del personaje.	
Observaciones: El sexo no está incluido en la selección aleatoria.	
Prototipo de interfaz gráfica:	

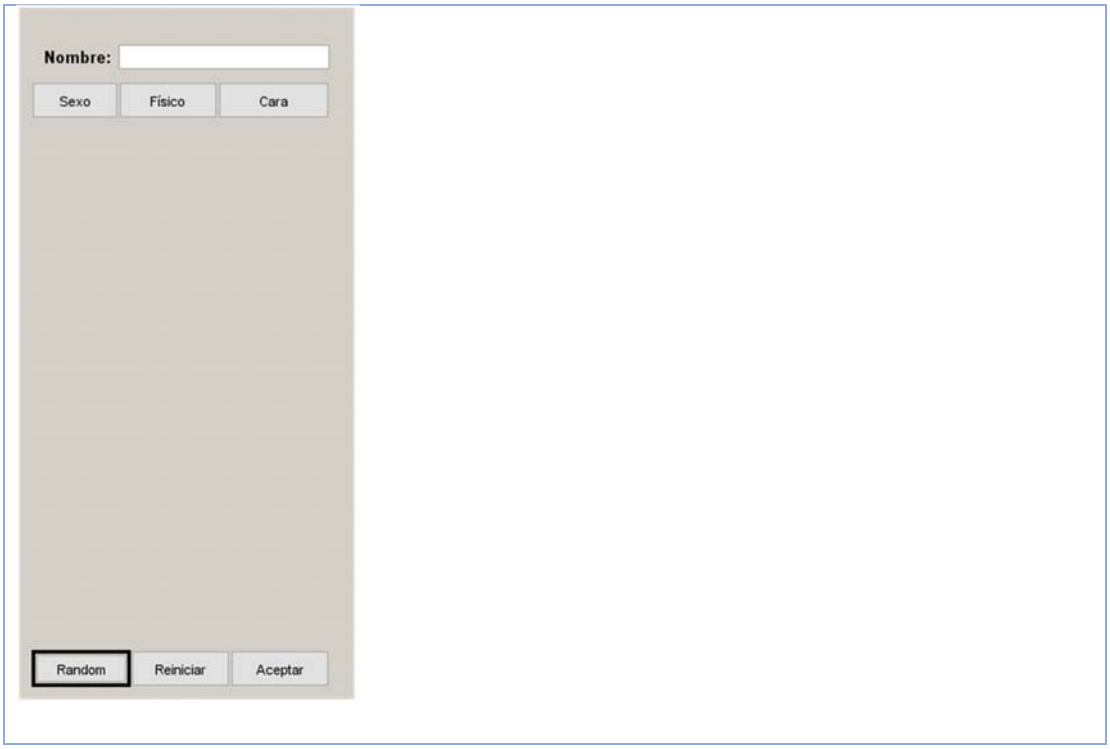


Tabla 9: HU Salvar la configuración del personaje (Fuente: Elaboración propia).

HISTORIA DE USUARIO	
Número: 7.	**Usuario:** Jugador.
Nombre: Salvar la configuración del personaje.	
Prioridad en negocio: Alto.	**Riesgo en desarrollo:** Alto.
Puntos estimados: 2.	**Iteración asignada:** 2.
Programador responsable: Eric Mañalich Cossio.	
Descripción: Salva la configuración del personaje y finaliza el componente de edición pasando al escenario del juego.	
Observaciones: El escenario del juego depende del juego en donde se utilice el componente.	
Prototipo de interfaz gráfica:	

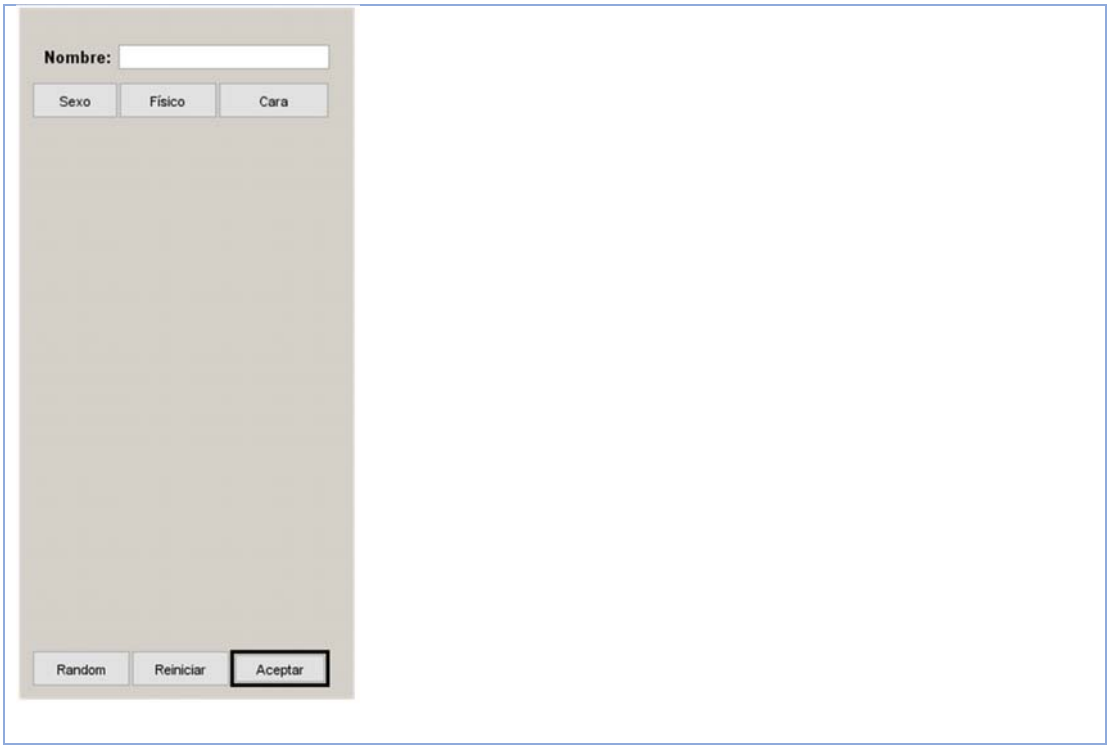
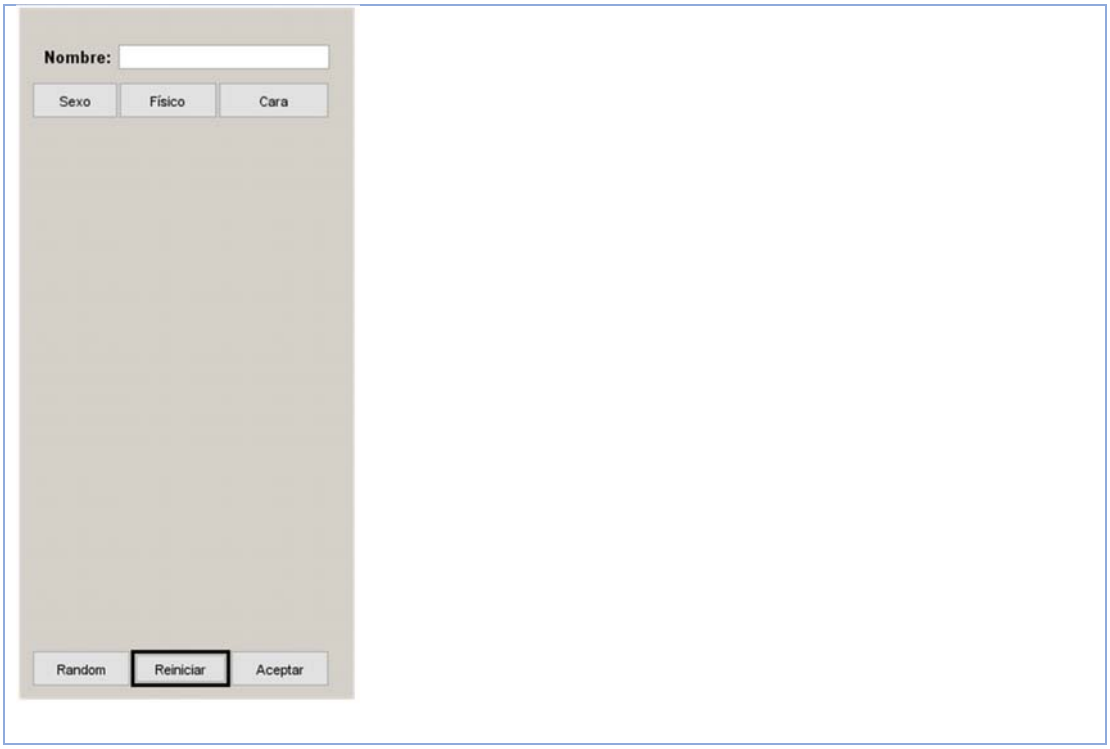


Tabla 10: HU Salvar la configuración del personaje (Fuente: Elaboración propia).

HISTORIA DE USUARIO	
Número: 8.	**Usuario:** Jugador.
Nombre: Reiniciar la configuración del personaje.	
Prioridad en negocio: Bajo.	**Riesgo en desarrollo:** Alto.
Puntos estimados: 1.	**Iteración asignada:** 2.
Programador responsable: Eric Mañalich Cossio.	
Descripción: Permite al usuario reiniciar la configuración del personaje.	
Observaciones: Solo se reinician los *"slider"*.	
Prototipo de interfaz gráfica:	



2.3.2. Estimación de esfuerzo por historias de usuario

Se realiza la estimación de esfuerzo por cada HU para lograr un correcto desarrollo de la aplicación.

Tabla 11: Estimación de esfuerzo por historias de usuario (Fuente: Elaboración propia).

Historias de usuario		Puntos estimados (semanas)
1.	Mostrar panel de configuración.	1.
2.	Seleccionar género del personaje.	1.
3.	Mover proyección del personaje.	1.
4.	Configurar aspectos físicos del personaje.	3.
5.	Configurar aspectos faciales del personaje.	3.
6.	Configurar aspectos del personaje aleatoriamente.	2.
7.	Salvar la configuración del personaje.	2.

8.	Reiniciar la configuración del personaje.	1.
Total:		14.

2.3.3. Requisitos no funcionales (RNF)

Son las restricciones sobre las HU identificadas:

RNF1: Requisitos de usabilidad.

- **RNF1.1:** El componente será usado por programadores del centro Vertex con conocimientos en Unity 3D y C#.

RNF2: Requisitos de restricciones de diseño e implementación.

- **RNF2.1:** El componente será implementado en el motor de juegos Unity 3D 2017.2.0f3.
- **RNF2.2:** El componente será implementado en el IDE de desarrollo Visual Studio 2017.
- **RNF2.3:** El componente será implementado usando el lenguaje de programación C#.
- **RNF2.4:** El componente será modelado con la herramienta CASE: *Visual Paradigm Suite Windows* 5.0sp1.

RNF3: Requisitos de software.

- **RNF3.1:** Depende del juego en donde sea utilizado el componente.

RNF4: Requisitos de hardware.

- **RNF4.1:** Depende del juego en donde sea utilizado el componente.

RNF5: Requisitos de apariencia o interfaz externa.

- **RNF5.1:** Los botones y 'slider' deben ser intuitivos, su diseño depende del juego en donde sea usado.

RNF6: Requisitos de rendimiento.

- **RNF6.1:** La aplicación tendrá un tiempo de respuesta, en todas sus funciones, de menos de 30ms. Esto varia en dependencia de la calidad gráfica de los modelos que sean utilizados.

2.3.4. Desarrollo del plan de iteraciones

En esta fase se confecciona un plan de iteraciones, en donde se especifica la prioridad con las que se implementan las HU según la iteración correspondiente.

La siguiente tabla refleja la duración (en semanas) y el orden en que se deberán implementar las iteraciones:

Tabla 12: Plan de duración de las iteraciones (Fuente: Elaboración propia).

Iteración	HU a implementar		Duración de la iteración
1.	1.	Mostrar panel de configuración.	6.
	2.	Seleccionar género del personaje.	
	3.	Mover proyección del personaje.	
	4.	Configurar aspectos físicos del personaje.	
2.	5.	Configurar aspectos faciales del personaje.	8.
	6.	Configurar aspectos del personaje aleatoriamente.	
	7.	Salvar la configuración del personaje.	
	8.	Reiniciar la configuración del personaje.	
Total:			14.

2.3.5. Plan de entregas

Al final de cada iteración se realizará una entrega de una versión de la aplicación.

Tabla 13: Plan de entrega de las iteraciones (Fuente: Elaboración propia).

Iteración	Fecha de Entrega
1.	01/12/2018-23/01/2019.
2.	24/01/2019-22/03/2019.

2.4. Fase de Diseño

En esta fase se crean estructuras que permiten organizar la lógica de la aplicación. Un diseño dividido en partes facilita el proceso de programación permitiendo que la aplicación sea escalable.

2.4.1. Arquitectura de la aplicación

La aplicación se desarrolla sobre el motor de videojuegos Unity 3D en su versión 2017 que posee una arquitectura en tres capas:

- **Capa de presentación:** Se encuentra “ButtonManager” que se encarga de capturar y mandar a ejecutar las funcionalidades de los botones. La clase “BlendShapeSlider” que se encarga de capturar y mandar a modificar la información de las “blendshape”. También se encuentra “MuestraRotate” que permite rotar el personaje utilizando el ratón.
- **Capa de negocio:** Se encuentra “CharacterCustomization” que es la clase que se encarga de implementar casi todas las funcionalidades. La clase “Singleton” que implementa el patrón Singleton para crear una de “CharacterCustomization”. La clase “Personaje” que guarda los cambios en el avatar. La clase “BlendShape” que son objetos para guardar la información de las “blendshape”. La clase “Cámara” que implementa los controles de la cámara; y la clase “ReiniciarTransformaciones” que reinicia la posición, rotación y escala de los objetos instanciados en la escena.
- **Capa de soporte:** Se encuentran las librerías “System.Collection” y “System.Colletion.Generic” que contienen funcionalidades del lenguaje C#. La librería “System.UnityEngine” que contiene las funciones de Unity. También se encuentra “System.Linq” que permite ejecutar funcionalidades de forma simultánea para optimizar el tiempo de respuesta de la aplicación.

La siguiente figura muestra la relación entre cada una de las capas:

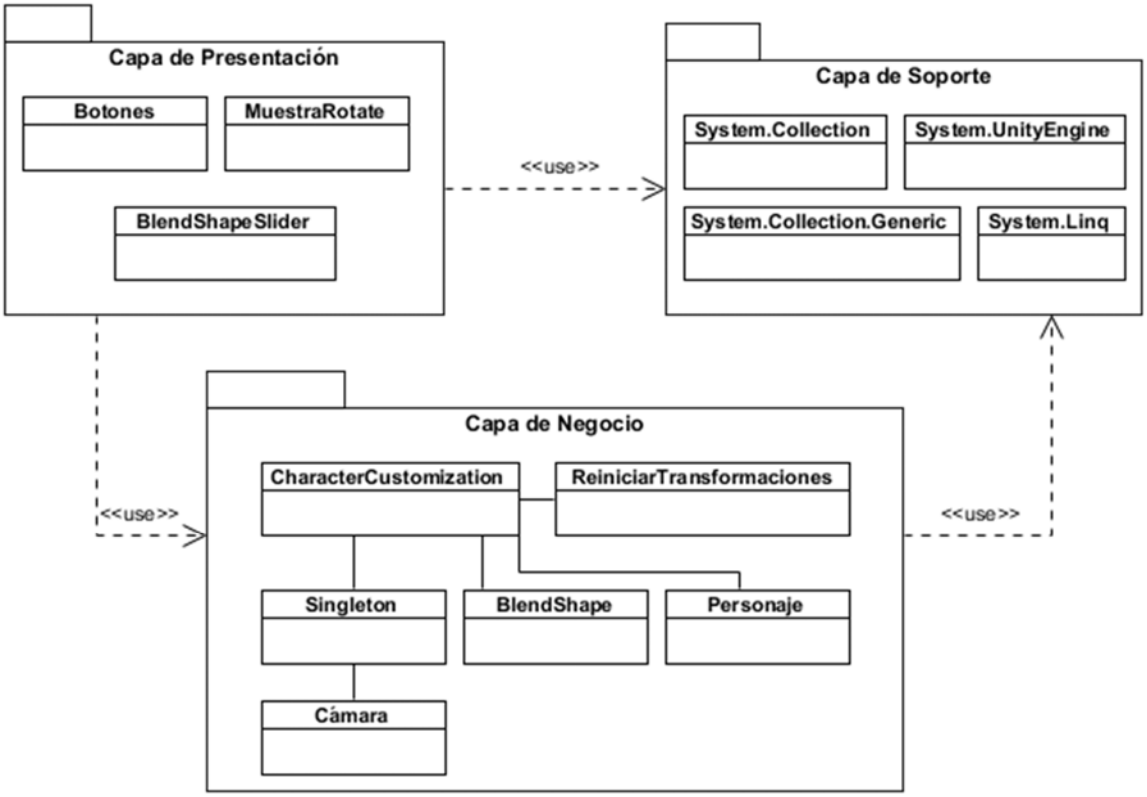


Ilustración 16: Diagrama de paquete (Fuente: Elaboración propia).

2.4.2. Tarjetas CRC

La metodología XP se utiliza tarjetas CRC para describir las relaciones entre clases.

Tabla 14: Tarjeta CRC CharacterCustomization (Fuente: Elaboración propia).

Tarjeta CRC	
Clase: CharacterCustomization.	
Responsabilidad	**Colaboración**
CuerpoModelUp(): Permite cambiar el modelo de cuerpo base del avatar. **PeloModelUp():** Permite cambiar el modelo de pelo del avatar.	Singleton. Blendshapes. ReiniciarTransformaciones. Personaje.

PielColor(int color): Permite cambiar el color de la piel del avatar. **PeloColor(int color):** Permite cambiar el color de pelo del avatar. **OjoColor(int color):** Permite cambiar el color de los ojos del avatar. **ApplyModification(AppearanceDetails detail, int id):** Aplica las modificaciones anteriores. **Randomize():** Selecciona configuraciones aleatorias. **Resett():** Reinicia las configuraciones. **ChangeBlendshapeValue(string blendshapeName, float value):** Cambia el valor de los parámetros que editan las mayas del modelo (blendshape). **Initialized():** Inicializa la aplicación. **ParseBlendShapesToDictionary():** Es un parse que escanea, captura y guarda todas las 'blendshape' y sus valores correspondientes.	

Tabla 15: Tarjeta CRC Singleton (Fuente: Elaboración propia).

Tarjeta CRC	
Clase: Singleton.	
Responsabilidad	**Colaboración**

Awake(): Crea una instancia de la clase CharacterCustomization y de la clase Cámara.	CharacterCustomization. BlendShapeSlider. Botones. Cámara.

Tabla 16: Tarjeta CRC Blendshapes (Fuente: Elaboración propia).

Tarjeta CRC	
Clase: Blendshapes.	
Responsabilidad	**Colaboración**
Blendshapes(int positiveIndex, int negativeIndex): Constructor.	CharacterCustomization.

Tabla 17: Tarjeta CRC BlendShapeSlider (Fuente: Elaboración propia).

Tarjeta CRC	
Clase: BlendShapeSlider.	
Responsabilidad	**Colaboración**
Start(): Captura el valor de los 'Slider' y se lo envía a CharacterCustomization a través del patrón Singleton.	Singleton.

Tabla 18: Tarjeta CRC Botones (Fuente: Elaboración propia).

Tarjeta CRC	
Clase: Botones.	
Responsabilidad	**Colaboración**
CuerpoModelUp(): Llama a la función CuerpoModelUp() de la clase	Singleton.

CharacterCustomization a través del patrón Singleton.	

Tabla 19: Tarjeta CRC "Cámara" (Fuente: Elaboración propia).

Tarjeta CRC	
Clase: Cámara.	
Responsabilidad	**Colaboración**
Enfocar(float x, float y, float z): Enfoca la cámara a las coordenadas seleccionadas. **Camera(string newName):** Cambia el foco de la cámara hacia la cara, el cuerpo o el personaje completo. **moverCamera(string accion):** Mueve la cámara en dependencia de la acción seleccionada.	Singleton.

Tabla 20: Tarjeta CRC ReiniciarTransformaciones (Fuente: Elaboración propia).

Tarjeta CRC	
Clase: ReiniciarTransformaciones.	
Responsabilidad	**Colaboración**
ResetTransform(this Transform): Reinicia las transformaciones de posición, rotación y escala de los objetos.	CharacterCustomization.

Tabla 21: Tarjeta CRC MuestraRotate (Fuente: Elaboración propia).

Tarjeta CRC
Clase: MuestraRotate.

Responsabilidad	Colaboración
Update(): Captura el movimiento del ratón, al hacer clic, para rotar al personaje.	

Tabla 22: Personaje (Fuente: Elaboración propia).

Tarjeta CRC	
Clase: Personaje.	
Responsabilidad	**Colaboración**
Set(): Salva la configuración del avatar. **Get():** Devuelve la configuración del avatar.	CharacterCustomization.

2.4.3. Patrones de diseño

Los Patrones son soluciones comunes a problemas de diseño de software orientado a objetos y que además poseen ciertas características de efectividad para resolver ese problema. Son reusables ya que pueden ser aplicados en otros diseños o problemas (Palacios, 2013).

En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Teniendo presente los siguientes elementos de un patrón: su nombre, el problema (¿cuándo aplicar un patrón?), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios) (Palacios, 2013).

Los Patrones de Asignación de Responsabilidades (GRASP) son un sistema orientado a objetos, los cuales envían mensajes a otros objetos para que lleven a cabo las operaciones requeridas. Los diagramas de interacción describen gráficamente estas operaciones, a partir de los objetos en interacción, que se responsabilizan de una actividad determinada (Palacios, 2013).

Patrones GRASP:

- **Experto:** Clase que tiene la responsabilidad de ejecutar una acción determinada y cuenta con el acceso a los datos necesarios.
- **Creador:** Se encarga de crear instancias de una clase o de un objeto.
- **Alta Cohesión:** Cuando cada clase realiza pocas funciones.
- **Bajo Acoplamiento:** Cuando una clase no depende de muchas clases.
- **Controlador:** Clase que se encarga de manejar los objetos del sistema.

Los patrones “Experto”, “Creador” y “Controlador” se aplican en la clase CharacterCustomization, la cual contiene todos los modelos, las texturas y las “blendshape” necesarias para crear instancias de los avatares y editarlos. Esta clase es la encargada de gestionar todos los eventos del sistema.

Los patrones “Alta Cohesión” y “Bajo Acoplamiento” se aplican a las clases Singleton, Botones que se encarga de llamar a las funciones asociadas a los botones de la clase CharacterCustomization; BlendShapeSlider que se encarga de capturar la información de los “Slider” y enviársela a la clase CharacterCustomization; Blendshapes que crea objetos que contienen la información de los “Slider”; ReiniciarTransformaciones que reinicia las transformaciones de los objetos 3D; MuestraRotate que permite rotar, con el ratón, el personaje; Cámara que se encarga de gestionar los controles relacionados a la cámara; Personaje que guarda la configuración del avatar. Además, esto garantizan que cada una de estas clases se encarguen solamente de su área funcional a partir de disminuir las relaciones entre clases, contribuyendo de esta manera a la cohesión y el acoplamiento.

Patrón GoF:

Los patrones GoF se descubren como una forma indispensable de enfrentarse a la programación. Indican resoluciones técnicas basadas en programación orientada a objetos, ya que favorecen la reutilización de código y la utilización de clases reutilizables (Palacios, 2013).

Se utilizó el patrón Singleton que crea una instancia de la clase CharacterCustomization y la clase Cámara para proporcionarles un punto de acceso global.

2.5. Conclusiones parciales

En este capítulo se abordaron las fases de planificación y diseño de la metodología XP, permitiendo obtener los siguientes resultados:

- La extracción de los ocho (8) RF, en forma de HU, sirvieron de guía durante todo el proyecto.
- Los prototipos de interfaces de usuario dan una visualización de los elementos a editar en los avatares y su ubicación, agrupándolos en dos grupos fundamentales: los rasgos físicos y faciales.
- El plan de entrega diseñado en dos (2) iteraciones le brinda al cliente dos (2) oportunidades para revisar el producto sistemáticamente.
- La arquitectura en tres capas propicia buena organización a la hora de implementar el componente y facilita su integración con juegos desarrollados en Unity.
- Las nueve (9) tarjetas CRC posibilitan comprender la interacción entre clases y los métodos necesarios a implementar.
- El patrón Singleton permite crear una instancia segura de la clase principal (CharacterCustomization) y de la clase (Cámara) proporcionándoles un acceso global.

CAPÍTULO 3: DESARROLLO Y PRUEBA DEL COMPONENTE PARA LA PERSONALIZACIÓN DE PERSONAJES 3D EN VIDEOJUEGOS DESARROLLADOS PARA UNITY

3.1. Introducción

En este capítulo se abordará la fase de desarrollo, en donde se abordará de las tareas de ingeniería y programación necesarias para la implementación de las HU. También se hace referencia a la fase de prueba, en donde se confeccionan planes de pruebas para extraer y corregir las no conformidades.

3.2. Fase de Desarrollo

En esta fase se desglosan las operaciones en las que se puede implementar cada HU mediante tareas de ingeniería o programación. Es en donde se genera el código fuente de la aplicación (Joskowicz, 2008).

Tabla 23:TP Crear un panel de configuración (Fuente: Elaboración propia).

TAREA DE PROGRAMACIÓN	
Número de tarea: 1.	**Historia de usuario:** 1. Mostrar panel de configuración.
Nombre de la tarea: Crear un panel de configuración.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1.
Fecha de inicio: 01/12/2018.	**Fecha de fin:** 07/12/2018.
Programador responsable: Eric Mañalich Cossio.	
Descripción: Implementar, en el gestor de interfaces de usuario de Unity, todas las interfaces de las HU.	

Tabla 24: TP Seleccionar género del personaje (Fuente: Elaboración propia).

TAREA DE PROGRAMACIÓN	
Número de tarea: 2.	**Historia de usuario:** 2. Seleccionar género del personaje.
Nombre de la tarea: Seleccionar género del personaje.	

Tipo de tarea: Desarrollo.	**Puntos estimados:** 1.
Fecha de inicio: 08/12/2018.	**Fecha de fin:** 15/12/2018.
Programador responsable: Eric Mañalich Cossio.	
Descripción: Crear dos arreglos de objetos, uno para cada género, y una variable bandera para poder realizar las opciones de configuración en dependencia del sexo.	

Tabla 25: TP Controles de la cámara (Fuente: Elaboración propia).

TAREA DE PROGRAMACIÓN	
Número de tarea: 3.	**Historia de usuario:** 3. Mover proyección del personaje.
Nombre de la tarea: Controles de la cámara**.**	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1.
Fecha de inicio: 16/12/2018.	**Fecha de fin:** 23/12/2018.
Programador responsable: Eric Mañalich Cossio.	
Descripción: Implementar las funciones de acercar, alejar, rotar y mover la cámara**.**	

Tabla 26: TP Seleccionar el modelo base (Fuente: Elaboración propia).

TAREA DE PROGRAMACIÓN	
Número de tarea: 4.	**Historia de usuario:** 4. Configurar aspectos físicos del personaje.
Nombre de la tarea: Seleccionar el modelo base.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.5.
Fecha de inicio: 01/01/2019.	**Fecha de fin:** 03/01/2019.
Programador responsable: Eric Mañalich Cossio.	

Descripción: Implementar la funcionalidad de seleccionar un modelo de un arreglo en dependencia del sexo escogido.

Tabla 27: TP Seleccionar el color de piel (Fuente: Elaboración propia).

TAREA DE PROGRAMACIÓN	
Número de tarea: 5.	**Historia de usuario:** 4. Configurar aspectos físicos del personaje.
Nombre de la tarea: Seleccionar el color de piel**.**	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.5.
Fecha de inicio: 04/01/2019.	**Fecha de fin:** 07/01/2019.
Programador responsable: Eric Mañalich Cossio.	
Descripción: Implementar una funcionalidad para seleccionar un color de piel en una paleta con varios colores.	

Tabla 28: TP Editar las "blendshape" del cuerpo (Fuente: Elaboración propia).

TAREA DE PROGRAMACIÓN	
Número de tarea: 6.	**Historia de usuario:** 4. Configurar aspectos físicos del personaje.
Nombre de la tarea: Editar las "blendshape" del cuerpo.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2.
Fecha de inicio: 08/01/2019.	**Fecha de fin:** 22/01/2019.
Programador responsable: Eric Mañalich Cossio.	
Descripción: Crear un "Dictionary" para guardar la información de las "blendshape" del cuerpo e implementar la funcionalidad que permita editar sus valores.	

Tabla 29: TP Seleccionar peinado (Fuente: Elaboración propia).

TAREA DE PROGRAMACIÓN	
Número de tarea: 7.	**Historia de usuario:** 5. Configurar aspectos faciales del personaje.
Nombre de la tarea: Seleccionar peinado.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.5.
Fecha de inicio: 23/01/2019.	**Fecha de fin:** 25/01/2019.
Programador responsable: Eric Mañalich Cossio.	
Descripción: Implementar una funcionalidad para seleccionar un peinado.	

Tabla 30: TP Seleccionar el color de pelo (Fuente: Elaboración propia).

TAREA DE PROGRAMACIÓN	
Número de tarea: 8.	**Historia de usuario:** 5. Configurar aspectos faciales del personaje.
Nombre de la tarea: Seleccionar el color de pelo.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.5.
Fecha de inicio: 26/01/2019.	**Fecha de fin:** 28/02/2019.
Programador responsable: Eric Mañalich Cossio.	
Descripción: Implementar una funcionalidad para seleccionar un color de pelo en una paleta con varios colores.	

Tabla 31: TP Seleccionar el color de los ojos (Fuente: Elaboración propia).

TAREA DE PROGRAMACIÓN	
Número de tarea: 9.	**Historia de usuario:** 5. Configurar aspectos faciales del personaje.

Nombre de la tarea: Seleccionar el color de los ojos.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.5.
Fecha de inicio: 01/02/2019.	**Fecha de fin:** 03/02/2019.
Programador responsable: Eric Mañalich Cossio.	
Descripción: Implementar una funcionalidad para seleccionar un color de ojo en una paleta con varios colores.	

Tabla 32: TP Editar las "blendshape" de la cara (Fuente: Elaboración propia).

TAREA DE PROGRAMACIÓN	
Número de tarea: 10.	**Historia de usuario:** 5. Configurar aspectos faciales del personaje.
Nombre de la tarea: Editar las "blendshape" de la cara.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1.5.
Fecha de inicio: 04/02/2019.	**Fecha de fin:** 15/02/2019.
Programador responsable: Eric Mañalich Cossio.	
Descripción: Agregar al "Dictionary" la información de las "blendshape" de la cara e implementar la funcionalidad que permita editar sus valores.	

Tabla 33: TP Seleccionar atributos aleatorios (Fuente: Elaboración propia).

TAREA DE PROGRAMACIÓN	
Número de tarea: 11.	**Historia de usuario:** 6. Configurar aspectos del personaje aleatoriamente.
Nombre de la tarea: Seleccionar atributos aleatorios.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2.
Fecha de inicio: 16/02/2019.	**Fecha de fin:** 30/02/2019.

Programador responsable: Eric Mañalich Cossio.
Descripción: Implementar la funcionalidad que seleccione, de manera aleatoria, un modelo base en dependencia del sexo escogido, un peinado, el color de pelo, el color de ojos y el valor de todas las “blendshape”.

Tabla 34: TP Guardar la configuración del personaje (Fuente: Elaboración propia).

TAREA DE PROGRAMACIÓN	
Número de tarea: 12.	**Historia de usuario:** 7. Salvar la configuración del personaje.
Nombre de la tarea: Guardar la configuración del personaje.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2.
Fecha de inicio: 01/03/2019.	**Fecha de fin:** 15/03/2019.
Programador responsable: Eric Mañalich Cossio.	
Descripción: Implementar la funcionalidad que seleccione guarde la configuración del personaje para ser utilizada en el juego, esta funcionalidad también finaliza el componente.	

Tabla 35: Reiniciar a la configuración por defecto del personaje (Fuente: Elaboración propia).

TAREA DE PROGRAMACIÓN	
Número de tarea: 13.	**Historia de usuario:** 8. Reiniciar la configuración del personaje.
Nombre de la tarea: Reiniciar a la configuración por defecto del personaje.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 2.
Fecha de inicio: 16/03/2019.	**Fecha de fin:** 22/03/2019.
Programador responsable: Eric Mañalich Cossio.	

Descripción: Implementar la funcionalidad que seleccione, reinicie la configuración de las BlendShape.

3.2.1. Resultados del desarrollo del componente

Después de implementar todas las tareas de programación, se creó una versión "*beta"* de la aplicación, utilizando modelos de prueba, reflejada en las siguientes imágenes:

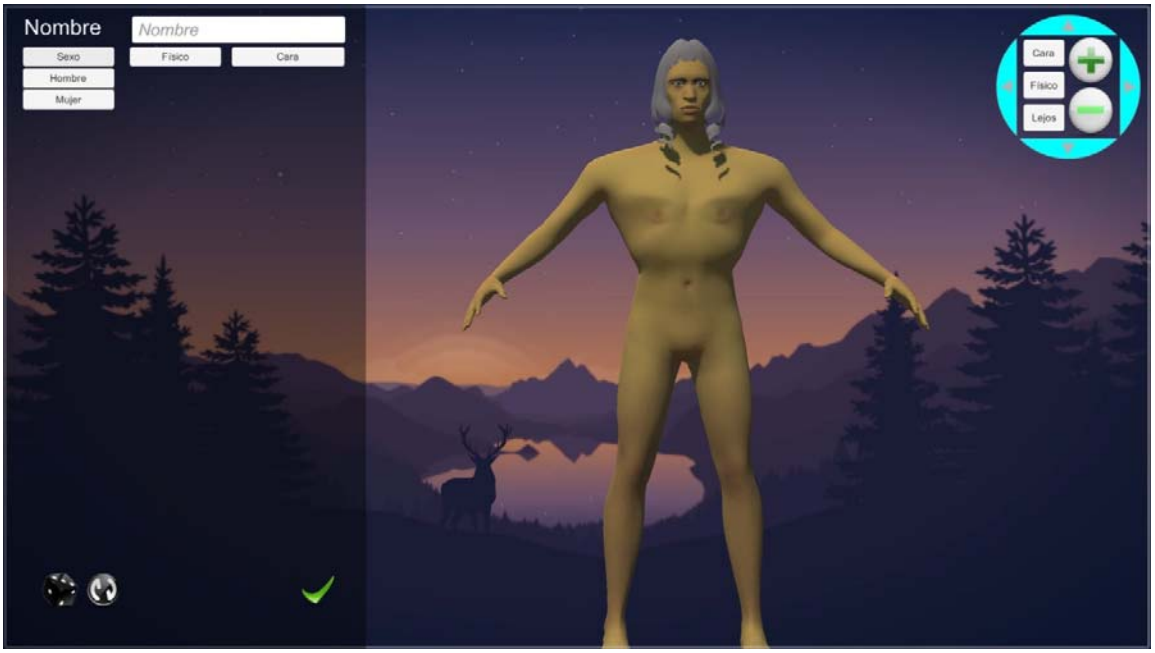


Ilustración 17: Resultados "Sexo" (Fuente: Elaboración propia).

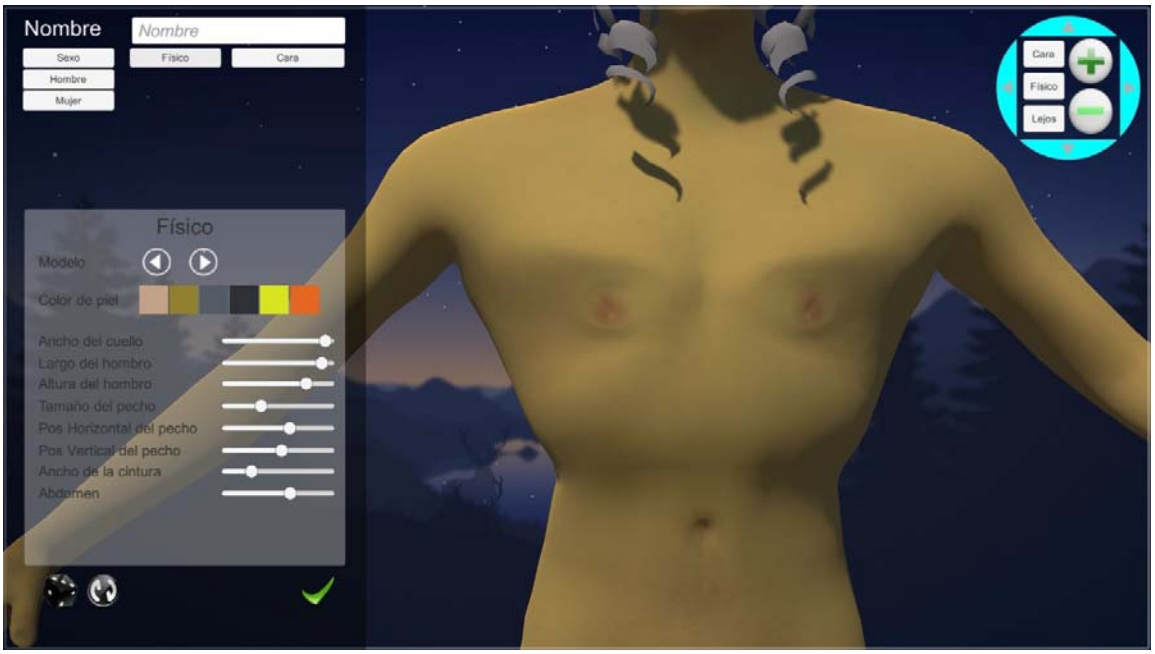


Ilustración 18: Resultados "Físico" (Fuente: Elaboración propia).

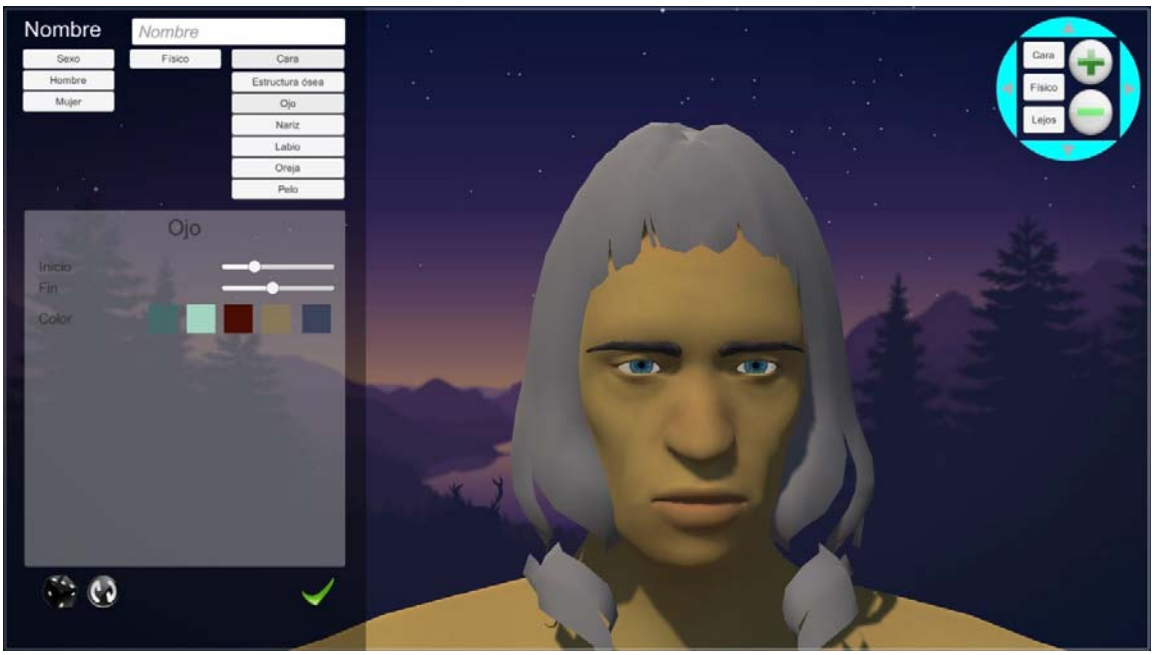


Ilustración 19: Resultados "Cara-Ojo" (Fuente: Elaboración propia).

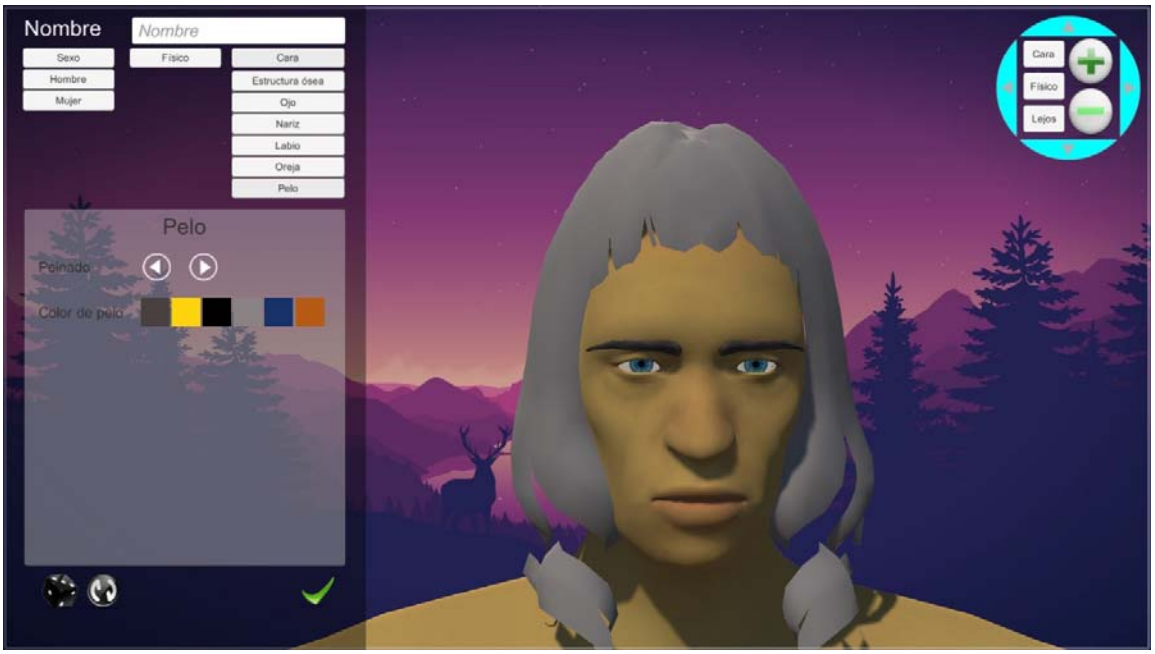


Ilustración 20: Resultados "Cara-Pelo" (Fuente: Elaboración propia).

3.3. Fase de Prueba

La prueba de software es un elemento fundamental para la garantía de la calidad del software y representa una revisión final de las especificaciones de los requisitos del sistema. Este proceso tiene dos objetivos distintivos (Sommerville, 2011):

- Demostrar al desarrollador y al cliente que el software satisface sus requisitos. Esto significa que debería haber al menos una prueba para cada requerimiento o característica que se incorporará a la entrega del producto.
- Descubrir defectos en el software en el que el comportamiento de este es incorrecto, no deseable o no cumple su especificación. La prueba de defectos está relacionada con la eliminación de todos los tipos de comportamientos del sistema no deseables, tales como caídas del sistema, interacciones no permitidas con otros sistemas, cálculos incorrectos y corrección de datos.

Uno de los pilares fundamentales de XP es el proceso de pruebas, el cual anima a los desarrolladores a probar constantemente tanto como sea posible (J. J. Gutiérrez, 2014).

3.3.1. Pruebas de aceptación y unidad

Las pruebas de aceptación XP, también las llamadas pruebas del cliente son especificadas por el mismo y se centran en las características y funcionalidades generales del sistema que son visibles y revisables por parte del cliente. Las pruebas de aceptaciones derivan de

las HU que se han implementado como parte de la liberación del software (Metodología XP, 2014).

Al efectuar una prueba de aceptación por cada tarea de programación también se ejecuta de forma simultánea las pruebas de unidad. En cada iteración de entrega de las HU al cliente, se realizan pruebas de aceptación. El objetivo es garantizar el cumplimiento de los RF.

3.3.2. Diseño de casos de prueba de aceptación

Las pruebas de aceptación fueron realizadas al culminar cada una de las dos iteraciones. En cada una el equipo se reunió con el cliente para detectar las no conformidades en cuanto a la respuesta del componente a las distintas configuraciones del avatar. A continuación, se muestran los casos de prueba de aceptación realizados para cada HU, dividido en las dos iteraciones de entrega del proyecto:

- **Iteración de entrega número 1:**

Tabla 36: PA Verificación del panel de configuración (Fuente: Elaboración propia).

CASO DE PRUEBA DE ACEPTACIÓN	
Código: HU1_P1.	**HU:** Mostrar panel de configuración.
Nombre: Verificación del panel de configuración.	
Descripción: Verificar la ubicación de los componentes de la interfaz de usuario, así como su ortografía.	
Condición de ejecución: Se debe tener buena ortografía y conocer las funcionalidades del componente.	
Pasos de ejecución: Seleccionar cada uno de los botones de la barra superior de la interfaz y comprobar que aparezca el menú de opciones asociado al botón correspondiente, así como la ortografía de los componentes.	
Resultados esperados: Todos los menús de opciones aparecen en correspondencia al botón seleccionado y poseen una correcta ortografía.	

Tabla 37: PA Comprobación del cambio de género del personaje (Fuente: Elaboración propia).

CASO DE PRUEBA DE ACEPTACIÓN	
Código: HU2_P2.	**HU:** Seleccionar género del personaje.
Nombre: Comprobación del cambio de género del personaje.	
Descripción: Comprobar que los avatares que muestra el componente correspondan con el género seleccionado.	
Condición de ejecución: Se deben tener avatares de ambos sexos.	
Pasos de ejecución: Seleccionar, dentro del menú “Sexo”, la opción “Hombre” o “Mujer” varias veces para comprobar si ocurren errores al cambiar de sexo.	
Resultados esperados: El avatar se muestra en correspondencia al sexo seleccionado.	

Tabla 38: PA Verificación de los controles de cámara (Fuente: Elaboración propia).

CASO DE PRUEBA DE ACEPTACIÓN	
Código: HU3_P3.	**HU:** Mover proyección del personaje.
Nombre: Verificación de los controles de cámara.	
Descripción: Verificar que la cámara se mueva en correspondencia a los botones seleccionados.	
Condición de ejecución: Se deben tener avatares de distintos tamaños.	
Pasos de ejecución: Seleccionar los botones “^” (arriba), “v” (abajo), “<” (izquierda), “>” (derecha), acercar y alejar. Cada uno debe permitir un máximo de cinco (5) movimientos y no debe desaparecer el avatar. Seleccionar “Cara” debe enfocar a la cara del avatar sin importar el tamaño. Seleccionar “Físico” debe enfocar al aspecto físico del avatar. Seleccionar “Lejos” debe mostrar el avatar completo.	

Hacer clic con el ratón y arrastrar debe girar el avatar en la misma dirección y con la misma velocidad del puntero del ratón.
Resultados esperados: Cada botón realiza la función deseada sin importar el orden, momento o cantidad de veces que sean tocados, girar el avatar no compromete las ediciones realizadas sobre él.

Tabla 39: PA Comprobación del cambio de modelos base del avatar (Fuente: Elaboración propia).

CASO DE PRUEBA DE ACEPTACIÓN	
Código: HU4_P4.	**HU:** Configurar aspectos físicos del personaje.
Nombre: Comprobación del cambio de modelos base del avatar.	
Descripción: Comprobar que se muestren los modelos base de los avatares en dependencia del sexo seleccionado.	
Condición de ejecución: Se deben tener al menos dos avatares de cada sexo.	
Pasos de ejecución: En la opción de “Físico/Modelo base” seleccionar siguiente y anterior varias veces, asegurarse que en ambos sentidos se completa el ciclo completo de modelos para comprobar si después del último vuelve al primero. Cambiar el sexo y repetir el paso anterior varias veces.	
Resultados esperados: Se muestran correctamente los modelos en dependencia del sexo, se muestran todos los modelos, no se desbordan los arreglos donde están almacenados los modelos, después del último modelo regresa al primero y viceversa.	

Tabla 40: PA Comprobación del cambio de color de piel (Fuente: Elaboración propia).

CASO DE PRUEBA DE ACEPTACIÓN	
Código: HU4_P5.	**HU:** Configurar aspectos físicos del personaje.
Nombre: Comprobación del cambio de color de piel.	

Descripción: Comprobar que se note el cambio de tonalidad de la piel en dependencia al color seleccionado.
Condición de ejecución: Se deben tener un avatar con textura en la piel y se deben de haber configurado los colores de la paleta de selección.
Pasos de ejecución: Seleccionar los colores de piel deseados en la paleta de selección.
Resultados esperados: Se muestran correctamente los tonos de piel en dependencia al color seleccionado sin destruir la textura del avatar.

Tabla 41: PA Comprobación de la configuración de los "slider" del cuerpo (Fuente: Elaboración propia).

CASO DE PRUEBA DE ACEPTACIÓN	
Código: HU4_P6.	**HU:** Configurar aspectos físicos del personaje.
Nombre: Comprobación de la configuración de los "*slider*" del cuerpo.	
Descripción: Comprobar que los "*slider*" editen los aspectos físicos correspondientes.	
Condición de ejecución: Se deben varios avatares con las Blendshape configuradas**.**	
Pasos de ejecución: Mover cada uno de los "*slider*" desde el mínimo al máximo, en todos los modelos, para asegurarse que editen su zona correspondiente.	
Resultados esperados: Los "*slider*" configuran correctamente a los avatares.	

- **Iteración de entrega número 2:**

Tabla 42: PA Comprobación del cambio de modelos de peinado (Fuente: Elaboración propia).

CASO DE PRUEBA DE ACEPTACIÓN	
Código: HU5_P7.	**HU:** Configurar aspectos faciales del personaje.
Nombre: Comprobación del cambio de modelos de peinado.	
Descripción: Comprobar que se muestren los modelos de peinado en la posición correcta.	

Condición de ejecución: Se deben tener al menos dos avatares de distinto tamaño y varios modelos de peinados compatibles con los avatares utilizados.
Pasos de ejecución: En la opción de “Cara/Pelo/Peinado” seleccionar siguiente y anterior varias veces, asegurarse que en ambos sentidos se completa el ciclo completo de modelos para comprobar si después del último vuelve al primero. Cambiar el modelo y repetir el paso anterior varias veces para ver si el pelo se ubica en la cabeza de forma adecuada.
Resultados esperados: Se muestran correctamente los peinados sin importar el tamaño del avatar, se muestran todos los peinados, no se desbordan los arreglos donde están almacenados los peinados, después del último peinado regresa al primero y viceversa.

Tabla 43: Comprobación del cambio de color de pelo (Fuente: Elaboración propia).

CASO DE PRUEBA DE ACEPTACIÓN	
Código: HU5_P8.	**HU:** Configurar aspectos faciales del personaje.
Nombre: Comprobación del cambio de color de pelo.	
Descripción: Comprobar que se note el cambio de tonalidad del pelo en dependencia al color seleccionado.	
Condición de ejecución: Se deben tener un modelo de peinado con textura y se deben de haber configurado los colores de la paleta de selección.	
Pasos de ejecución: Seleccionar los colores de pelo deseados en la paleta de selección.	
Resultados esperados: Se muestran correctamente los tonos del pelo en dependencia al color seleccionado sin destruir la textura del avatar.	

Tabla 44: PA Comprobación del cambio de color de los ojos (Fuente: Elaboración propia).

CASO DE PRUEBA DE ACEPTACIÓN	
Código: HU5_P9.	**HU:** Configurar aspectos faciales del personaje.

Nombre: Comprobación del cambio de color de los ojos.
Descripción: Comprobar que se note el cambio de tonalidad de los ojos en dependencia al color seleccionado.
Condición de ejecución: Se deben tener un modelo de ojos con textura y se deben de haber configurado los colores de la paleta de selección.
Pasos de ejecución: Seleccionar los colores de los ojos deseados en la paleta de selección.
Resultados esperados: Se muestran correctamente los tonos de los ojos en dependencia al color seleccionado sin destruir la textura del avatar.

Tabla 45: PA Comprobación de la configuración de los "slider" de la cara (Fuente: Elaboración propia).

CASO DE PRUEBA DE ACEPTACIÓN	
Código: HU5_P10.	**HU:** Configurar aspectos faciales del personaje.
Nombre: Comprobación de la configuración de los "*slider*" de la cara.	
Descripción: Comprobar que los "*slider*" editen los aspectos faciales correspondientes.	
Condición de ejecución: Se deben varios avatares con las Blendshape configuradas.	
Pasos de ejecución: Mover cada uno de los "*slider*" desde el mínimo al máximo, en todos los modelos, para asegurarse que editen su zona correspondiente.	
Resultados esperados: Los "*slider*" configuran correctamente a los avatares.	

Tabla 46: PA Verificación de la función "Random" (Fuente: Elaboración propia).

CASO DE PRUEBA DE ACEPTACIÓN	
Código: HU6_P11.	**HU:** Configurar aspectos del personaje aleatoriamente.
Nombre: Verificación de la función "*Random*".	

Descripción: Comprobar que los modelos bases, el color de la piel, el peinado, el color del pelo y los valores de los "*slider*" se editen de forma aleatoria.
Condición de ejecución: Se deben varios avatares con las Blendshape configuradas, varios modelos de peinado, varios colores de piel y pelo.
Pasos de ejecución: Al iniciar el componente bebe de salir un modelo aleatorio. Tocar varias veces el botón "*Random*" y confirmar que se editan las funciones del componente de forma aleatoria a excepción del sexo.
Resultados esperados: Los avatares se modifican de forma aleatoria.

Tabla 47: PA Comprobación de la función guardar (Fuente: Elaboración propia).

CASO DE PRUEBA DE ACEPTACIÓN	
Código: HU7_P12.	**HU:** Salvar la configuración del personaje.
Nombre: Comprobación de la función guardar.	
Descripción: Comprobar que el avatar configurado se guarde correctamente.	
Condición de ejecución: Se necesita un juego hecho en Unity para cargar el avatar.	
Pasos de ejecución: Acoplar el componente a un juego hecho en Unity y verificar que el avatar configurado se cargue correctamente.	
Resultados esperados: Los avatares modificados se cargan correctamente.	

Tabla 48: PA Comprobación de la función "Reiniciar" (Fuente: Elaboración propia).

CASO DE PRUEBA DE ACEPTACIÓN	
Código: HU8_P13.	**HU:** Reiniciar la configuración del personaje.
Nombre: Comprobación de la función "Reiniciar".	
Descripción: Comprobar que se reinicien las configuraciones del avatar.	
Condición de ejecución: Se deben tener un avatar configurado.	

Pasos de ejecución: Seleccionar el botón “Reiniciar” y verificar que los “*slider*” se encuentren en 0 (el indicador debe estar por la mitad)**.**
Resultados esperados: Los “*slider*” regresan a 0, se bebe reflejar en el avatar.

3.3.3. Análisis de los resultados de las pruebas realizadas

La estrategia de prueba describe el enfoque y los objetivos generales de las actividades de prueba.

En la presente investigación se utilizó el método de prueba de caja negra, que se lleva a cabo sobre la interfaz del software, para demostrar que las funciones del componente son operativas. En este método de prueba no se ve el código de la aplicación. Para desarrollar las pruebas de caja negra se escogió la técnica de partición de equivalencia. Dicha técnica permite examinar los valores validos e inválidos de las entradas existentes en el software. Se escogió como tipo de prueba: funcionalidad, este se centra en la validación de las funciones o métodos descritos en las HU. Las pruebas se efectuaron en un entorno de desarrollo real con el usuario final, esto hace referencia al nivel de prueba de aceptación. Como se prueba cada tarea de programación también se desarrolló el nivel de prueba de unidad con la técnica de flujo de datos mediante el método de caja blanca, que se lleva a cabo en el código de la aplicación. En la técnica de flujo de datos se insertan al software las variables de entrada, en este caso la información de los botones y los ‘sliders’, y se comparan los resultados obtenidos con los esperados.

Además, se efectuaron pruebas de regresión, que consiste en realizar las pruebas antes descritas al completar cada iteración de entrega para todos los casos de prueba diseñados. En caso de que se encuentren no conformidades, se corrigen y se efectúa otra iteración de pruebas. No se puede proseguir con la evaluación de las siguientes funciones, hasta que se corrijan los errores obtenidos en las pruebas de la iteración anterior. A continuación, se muestran los resultados de las iteraciones de prueba efectuadas:

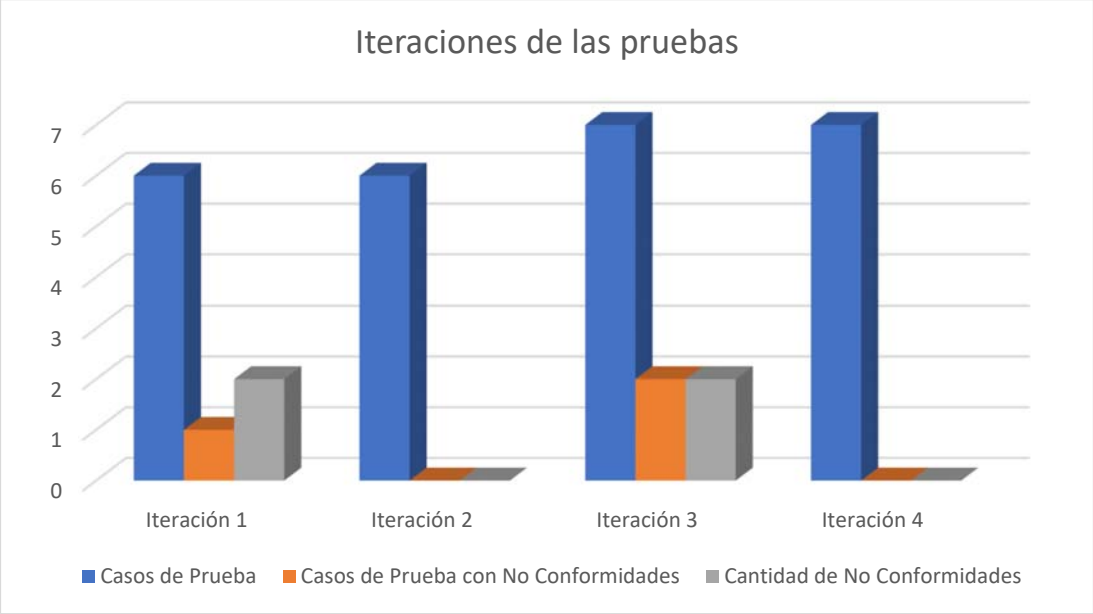


Ilustración 21: Iteraciones de las pruebas (Fuente: Elaboración propia).

En la primera iteración de entrega, el caso de prueba (CP) HU4_P5 tuvo dos (2) no conformidades (NC): el color de la tonalidad de la piel escogida no era visible cuando los avatares tenían textura sin transparencia; al cambiar el modelo base del avatar se reiniciaba el color de la piel. La iteración de prueba 1 y 2 de la gráfica corresponden a la primera iteración de entrega.

En la segunda iteración de entrega, el CP HU5_P7 tuvo una (1) NC: al cambiar el modelo base, desaparecía el modelo de peinado. La otra NC se encontraba en el CP HU6_P11: el botón “Random” no modificaba el valor de los “slider”. La iteración de prueba 3 y 4 de la gráfica corresponden a la segunda iteración de entrega.

3.4. Conclusiones parciales

En este capítulo se abordaron las fases de desarrollo y prueba de la metodología XP, permitiendo obtener los siguientes resultados:

- Las trece (13) tareas de programación desglosaron las HU para facilitar su implementación.
- Las pruebas de aceptación y unidad se les aplicaron a las trece (13) tareas de programación para asegurar una aplicación funcional.

Conclusiones generales

Con el desarrollo de la presente investigación se obtuvo un componente para la personalización de personajes 3D en videojuegos desarrollados para Unity, para lo cual:

- El componente implementado concede valor agregado a los videojuegos desarrollados en el centro Vertex; pues da libertad de escoger, a los usuarios, las características físicas de los personajes con los que van a interactuar durante cientos de horas.
- El componente concebido puede ser fácilmente utilizado en todos los videojuegos con personajes humanoides del centro Vertex y su desarrollo en Unity permite utilizarlo en aplicaciones multiplataformas.
- El diseño de los casos de prueba permitió detectar y corregir errores en las funcionalidades como: el cambio de la tonalidad de la piel no era visible con texturas sin transparencia; al cambiar el modelo base se reiniciaba el color de la piel y se perdía el modelo de peinado.

Recomendaciones

Para dar continuidad a la presente investigación se recomienda:

- Crear animaciones para que los personajes se muevan mientras se editan.
- Integrarlo con un módulo de inteligencia artificial que permita extraer los rasgos faciales de los jugadores con ayuda de un "*Kinect*".

Bibliografía

AWS. (2019). Obtenido de Trabajar con blend shapes: https://docs.aws.amazon.com/es_es/lumberyard/latest/legacyreference/anim-char-blendshapes.html

Baer, R. H. (25 de 4 de 1972). *United States Patent*. Obtenido de Television gaming apparatus and method: http://www.freepatentsonline.com/3659285.html

BioWare. (2014). *EA*. Obtenido de Dragon age inquisition: https://www.ea.com/games/dragon-age/dragon-age-inquisition

Blender. (2018). Obtenido de https://blender.stackexchange.com/questions/34758/is-it-possible-to-create-duplicates-of-a-mesh-that-can-be-deformed-and-used-as

Blueside, Phantagram. (2015). *MMOG*. Obtenido de kuf2.mmog.asia/en/?page_id=77

Bungie Studios, Vicarious Visions, High Moon Studios, Radical Entertainment. (2014). *Destinythegame*. Obtenido de https://www.destinythegame.com/

Carbine Studios. (2014). *Wildstar*. Obtenido de https://wildstar.es.jaleco.com/

CCP Games. (2016). *steampowered*. Obtenido de EVE_Online: https://store.steampowered.com/app/8500/EVE_Online/?l=spanish

CICE. (24 de 4 de 2018). Obtenido de La industria de los videojuegos: https://www.cice.es/noticia/la-industria-los-videojuegos/

Fenrir. (11 de 2018). *Cinco cosas buenas y cinco no tan buenas de Unity 3D*. Obtenido de https://laleyendadedarwan.es/2017/12/21/cinco-cosas-buenas-y-cinco-no-tan-buenas-de-unity-3d/

Game Career Guide. (29 de 4 de 2008). Obtenido de What is a game engine?: http://www.gamecareerguide.com/features/529/what_is_a_game_.php

Gamer Dic. (2013). Obtenido de Motor de juego: http://www.gamerdic.es/termino/motor-de-juego/

Guzmán, E. (9 de 11 de 2017). *Malavida*. Obtenido de Qué es Photoshop y para qué sirve: https://www.malavida.com/es/soft/photoshop/q/para-que-sirve-photoshop.html#gref

J, C. (2005). *Metodologías ágiles.*

J. J. Gutiérrez, M. J. (2014). *Pruebas del sistema en programación.* Sevilla: Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla.

Joskowicz, J. (2008). *Prácticas en Xtreme Programming.*

Julián Pérez Porto, A. G. (s.f.). *Definición .DE*. Obtenido de Definición de videojuego: https://definicion.de/videojuego/

La revista informática .com. (2015). Obtenido de Lenguaje de programación C#: http://www.larevistainformatica.com/C1.htm

La Vanguardia. (29 de 6 de 2012). Obtenido de Por qué enganchan los videojuegos: https://www.lavanguardia.com/estilos-de-vida/20120629/54317381414/por-que-enganchan-los-videojuegos.html

Lina María Montoya Suarez, J. M. (2017). *Análisis comparativo de las metodologías agiles en el desarrollo de software aplic.*

Liveplex. (2012). *Vendettagn*. Obtenido de https://sb.vendettagn.com/

Lucidchart. (2018). Obtenido de Qué es el lenguaje unificado de modelado (UML): https://www.lucidchart.com/pages/es/que-es-el-lenguaje-unificado-de-modelado-uml?a=1

Metodología XP. (25 de 5 de 2014). Obtenido de Funcionamiento: https://sites.google.com/site/xpmetodologia/marco-teorico/funcionamiento

Microsoft. (2017). Obtenido de Visual Studio Express Editions: https://web.archive.org/web/20070516193734/http://www.microsoft.com/spanish/msdn/vstudio/express/default.mspx

Neowiz, Neowiz Bless Studio. (2018). *Blessonline*. Obtenido de https://www.blessonline.net/

Observatorio del gabinete de tele-educación. (4 de 7 de 2018). Obtenido de ¿Qué es un motor de videojuegos?: http://blogs.upm.es/observatoriogate/2018/07/04/que-es-un-motor-de-videojuegos/

Palacios, O. S. (2013). *Módulo para la creación de videojuegos para jugadores virtuales de tipo batalla utilizando Unity 3D.* La Habana, Cuba: Universidad de las Ciencias Informàticas.

Patricio Letelier, C. P. (2015). *Métodologías ágiles para el desarrollo de software: Xtreme Programming (XP).* Valencia: Universidad Politécnica de Valencia.

Pressman, R. S. (2010). *Ingeniería de software, un enfoque práctico. Quinta edición.* McGraw-Hill Companies.

Retro informática. (2018). Obtenido de Historia de los videojuegos: https://www.fib.upc.edu/retro-informatica/historia/videojocs.html

Runewaker Entertainment, Runewaker. (2013). *Dragonsprophet*. Obtenido de www.dragonsprophet.com/

Sintya Milena, M. E. (2016). *Metodologia ágil de desarrollo de software programacion extrema.* Nicaragua: Universidad Nacional Autonoma de Nicaragua.

Sommerville. (2011).

Stack, P. (2005). *History of video game consoles*. Obtenido de http://content.time.com/time/interactive/0,31813,2029221,00.html

Tejeiro R., P. d. (12 de 8 de 2014). *Psico Gamer*. Obtenido de La psicología de los videojuegos: https://psicogamer.com/articulos/la-psicologia-de-los-videojuegos/

Temas especiales de computación. (2000). Obtenido de El lenguaje unificado de modelado (UML): http://profesores.fi-b.unam.mx/carlos/aydoo/uml.html

Unity 3D. (2018). Obtenido de Codificación en C# en Unity para principiantes: https://unity3d.com/es/learning-c-sharp-in-unity-for-beginners

Unity 3D share wordpress. (2014). Obtenido de https://unity3dshare.wordpress.com/2014/06/21/362-unity-assets-and-plugins/

Unity asset store. (2016). Obtenido de Character creation system (Ubrin): https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=2ahUKEwjew8ahroXgAhVJ11kKHf17DVQQFjAAegQIBBAB&url=https%3A%2F%2Fassetstore.unity.com%2Fpackages%2F3d%2Fcharacters%2Fhumanoids%2Fcharacter-creation-system-ubrin-56230&usg=AO

Visual Paradigm. (2019). Obtenido de About us: https://www.visual-paradigm.com/aboutus/

Webzen. (2014). *MMOG gratis*. Obtenido de Archlord 2: https://www.mmogratis.com/archlord-2/

Weigers. (2003).

XL games. (15 de 1 de 2013). *Steampowered*. Obtenido de https://store.steampowered.com/

Yee, N. (5 de 9 de 2015). *Hipertextual*. Obtenido de ¿Cuáles son los efectos psicológicos de tener tu propio avatar en un videojuego?: https://hipertextual.com/2015/09/efectos-psicologicos-de-tener-tu-propio-avatar