Universidad
Rey Juan Carlos

Escuela Técnica Superior de Ingeniería Informática

¡Al ladrón!

Diseño, modelado y animación de una galería de personajes para un juego 3D

Memoria del Trabajo Fin de Grado

Grado en Diseño y Desarrollo de Videojuegos

Autores:

Lucía Fresno Olmeda

Tutor: David Ortega del Campo

Noviembre 2024

Resumen

Este Trabajo de Fin de Grado consiste en la creación de varios personajes para un videojuego en 3D, desde su diseño visual y conceptual, hasta el desarrollo de sus animaciones en 3D y su inclusión en un motor de juego. Esto permitirá entender mejor cada una de las fases y conocer más a fondo este tipo de trabajos.

Para ello, se ha diseñado el videojuego '¡Al ladrón!' y se han detallado los aspectos principales del mismo, como sus mecánicas o sus personajes. Una vez hecho esto, el proyecto se ha centrado en el desarrollo de los enemigos del mismo. Estos se han creado siguiendo el flujo de trabajo habitual en videojuegos para el desarrollo de personajes en 3D. En primer lugar se han diseñado los personajes conceptual y visualmente, a continuación se han modelado y después, texturizado. Para poder animarlos, se les ha creado un esqueleto virtual (en la fase de rigging) y se ha asociado la malla del modelo a este esqueleto (en la fase de skinning). Por último, se han desarrollado manualmente cada una de las animaciones y, una vez terminadas, se ha importado cada personaje al motor de juego para que se encuentren ya listos para insertar en el videojuego.

Índice general

Índice de tablas

Índice de figuras

Nomenclatura

Animator controller Un elemento de Unity que permite definir el flujo de las animaciones.

Cartoon Estilo de dibujo o ilustración no realista.

Model-sheet Montaje visual en una página en la que se representan distintos aspectos de un personaje. El formato de esta página queda a decisión del diseñador.

Pixelart Técnica de dibujo o ilustración en la que se pueden apreciar los píxeles.

Retopologia En el mundo de gráficos 3D, redibujar los polígonos de una malla 3D ya existente con el fin de reducir su cantidad.

Script Nombre con el que se le conocen a los archivos de código en el campo de los videojuegos.

Trigger Conocido como 'disparador' en castellano, se refiere a un evento o condición que cuando se cumple hace que se ejecute una función o acción.

Turnaround Ilustración en la que se muestran varias vistas de un elemento. Se ilustran como mínimo una vista frontal y una lateral.

Capítulo 1

Introducción

En este proyecto se va a desarrollar una galería de personajes para el videojuego '¡Al ladrón!'. Se va a trabajar en los diseños y la estética de los mismos y en su desarrollo y movimiento en 3D.

Los personajes de un videojuego son uno de sus elementos más importantes: atraen la atención de jugadores y ayudan a definir la estética e identidad de un juego. A lo largo de este capítulo se va a poner en valor el arte en los videojuegos y se va a hacer un breve resumen de la evolución de los mismos, desde los primeros juegos que usaban gráficos muy limitados, hasta la gran variedad de estilos visuales de los juegos actuales.Además, se va a presentar el proceso de diseño y desarrollo de un personaje para un videojuego 3D, una de las técnicas más populares hoy en día. Por último, se describirá brevemente la estructura de la memoria a partir de este capítulo.

1.1. Importancia del arte en los videojuegos

En el artículo '*De la Usabilidad a la Jugabilidad: Diseño de Videojuegos Centrado en el Jugador*', se incluyen en el apartado artístico de los juegos a '*la calidad gráfica y visual, los efectos sonoros, la banda sonora y melodías del juego, la historia y la forma de narración de ésta, así como la ambientación realizada de todos estos elementos dentro del videojuego*' (Sánchez, Zea, Gutiérrez, y Cabrera, 2008) (Adobe, s.f.-a).

La estética o el arte gráfico de un juego está formada por varios elementos: sus interfaces, el diseño de los personajes, el de sus escenarios y de los elementos que se encuentren en él. Aunque a veces se pase por alto, la estética de un videojuego es uno de sus elementos más importantes, ya que puede ser uno de los factores que hagan que los jugadores se decidan a probarlo.

> '*Al analizar el éxito de los videojuegos y lo que le otorgan su actual atractivo, no se puede ignorar el factor gráfico y audiovisual (Sánchez Rodríguez, Alfageme González, y Serrano Pastor, 2010), siendo este uno de los motivos fundamentales de su popularidad*' (Solórzano Alcívar, Moscoso Poveda, y Elizalde Ríos, 2019).

En algunas ocasiones, como menciona Solórzano en su artículo, el diseño visual de un videojuego puede ser la causa principal de su éxito. Por ejemplo, el videojuego 'Gris'

(2018) (ver Figura 1.1.1) consiguió llegar a un gran público gracias al uso de una técnica innovadora en el momento: el uso de acuarelas para desarrollar el arte del videojuego. Por otra parte, existen otros videojuegos que, sin ser innovadores en este campo, han conseguido una gran fama por tener una estética visual muy reconocible. Algunos ejemplos podrían ser 'Journey' (2012) [86] o 'Undertale' (2015) [88] (ver Figuras 1.1.2 y 1.1.3).

Figura 1.1.1: Imágenes del videojuego 'Gris'.

Figura 1.1.2: Imágenes del videojuego 'Journey'.

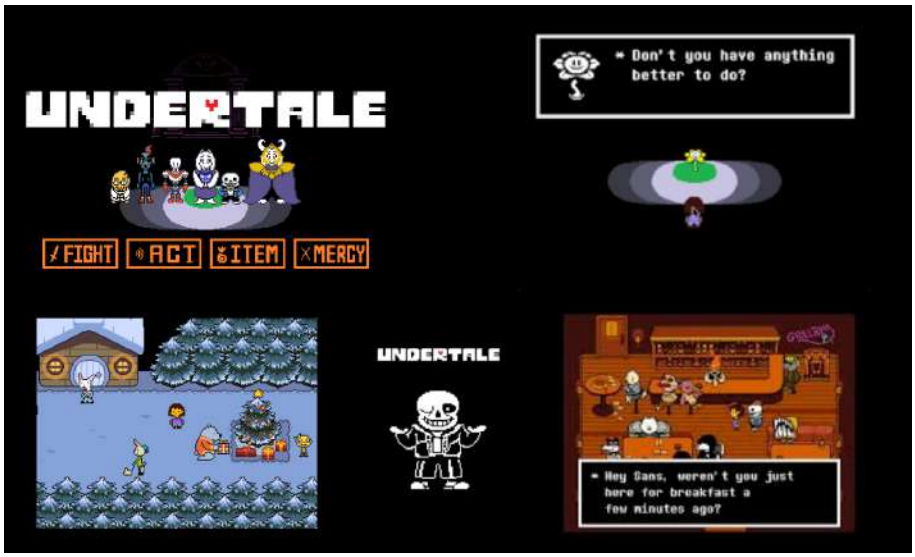


Figura 1.1.3: Imágenes del videojuego 'Undertale'.

El objetivo del arte de los videojuegos no es necesariamente conseguir un resultado 'bonito', sino conseguir llamar la atención del jugador, presentarle un mundo y que este entienda su funcionamiento y su papel dentro de él. Uno de los propósitos de la estética de un videojuego es, de alguna forma, lograr que el jugador se sienta partícipe en el juego. Algunas de las formas más comunes de incluir al jugador en el mundo del virtual son:

- Usando la cámara en primera persona, de forma que el jugador sentirá como si estuviera dentro del juego, ya que ese es su punto de vista.
- Construyendo personajes personalizables para que el jugador se pueda ver reflejado en ellos, ya sea por su aspecto físico, sus gustos o su personalidad. Uno de los ejemplos más conocidos de juegos que permiten personalizar a los personajes es la saga de videojuegos 'The Sims' [26], en los que el jugador puede editar cada detalle de su personaje. Sin embargo, es bastante común encontrar juegos en los que se puedan modificar, aunque de una forma más limitada, el aspecto del jugador. Por ejemplo, en 'Among Us' (2018) [34] o en 'Fall Guys' (2020) [46], el jugador puede elegir el color y accesorios de su avatar, como se muestra en la Figura 1.1.4.

Figura 1.1.4: Personalización del avatar en 'Among Us', 'Fall Guys' y 'Los Sims 4'.

No obstante, estas no son las únicas formas de hacerlo. A menudo los juegos no cuentan con avatares personalizables, sino que intentan que el jugador empatice con su protagonista. Tratan de atraer al jugador hasta tal punto que cuestione dónde acaba el personaje y dónde empieza el jugador; qué decisiones son suyas propias y cuáles pertenecen al personaje (Tato, 2017). Para ello, es vital que su diseño sea atractivo y el jugador pueda entender su personalidad, objetivos o historia.

Además de los personajes, también es importante que un juego cuente con un buen diseño de escenarios e interfaces y que todo ello se encuentre bien integrado. Esto es clave para que el jugador perciba que los distintos elementos de un juego forman parte de un 'todo'.

El apartado visual de un videojuego no sólo sirve para atraer a los jugadores, sino que también es una forma de darles información. Por ejemplo, un jugador puede deducir que el videojuego 'Phasmophobia' (2020) [37] es un videojuego de terror no apto para niños simplemente viendo la estética de su caratula, con colores oscuros y una tipografía tenebrosa. Por el contrario, el diseño visual de 'Animal Crossing: New Horizons' (2020) [55] indica al público que es un juego tranquilo, para todos los públicos. Esto se transmite gracias a su paleta de color y sus formas redondeadas. Por poner un último ejemplo, en el videojuego 'Rocket League' (2015) [61] se utilizan colores muy saturados y brillantes, con mucho contraste entre ellos. Esto da a entender al jugador que es un juego frenético y con mucha actividad. Todo esto se puede ver claramente en las imágenes de la Figura 1.1.5.

Figura 1.1.5: Imágenes de tres videojuegos que transmiten sensaciones muy distintas.

1.2. Evolución en el arte de videojuegos

En el año 1952, Alexander S. Douglas desarrolló lo que se considera el primer videojuego: 'Nought and Crosses' [1] (ver Figura 1.2.1). Este consistía en una adaptación digital del juego clásico de tres en raya, en la que el usuario jugaba contra la máquina (Belli y Raventós, 2008).

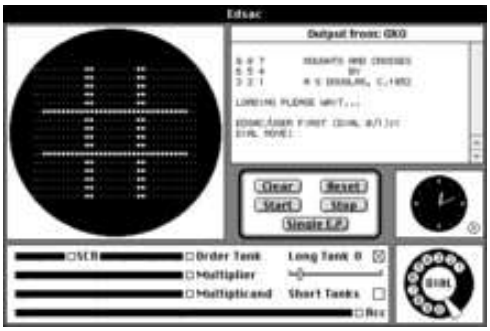


Figura 1.2.1: Captura de pantalla de una emulación del juego 'Nought and Crosses'.

Desde este primer juego, la industria de los juegos ha avanzado enormemente, dando pie a una gran variedad de juegos. El desarrollo de videojuegos ha estado muy influenciado por las innovaciones tecnológicas que se han dado a lo largo de su historia. De igual forma, el estilo artístico de los mismos ha ido evolucionando con los años (Solórzano Alcívar y cols., 2019) y (Belli y Raventós, 2008).

Los primeros juegos contaban con gráficos muy simples: formas simples y una gama de colores muy limitada. En general, la importancia del juego se centraba en la programación del mismo, su jugabilidad, dejando un poco de lado el apartado visual. Se buscaba conseguir un reto para las habilidades o intelecto del jugador (Solórzano Alcívar y cols., 2019). Algunos ejemplos de esta primera etapa visual podrían ser 'Spacewar!'(1962) [79] o 'Tennis for Two' (1958) [93] (ver Figura 1.2.2), juegos desarrollados para equipos muy distintos a los ordenadores de hoy en día.

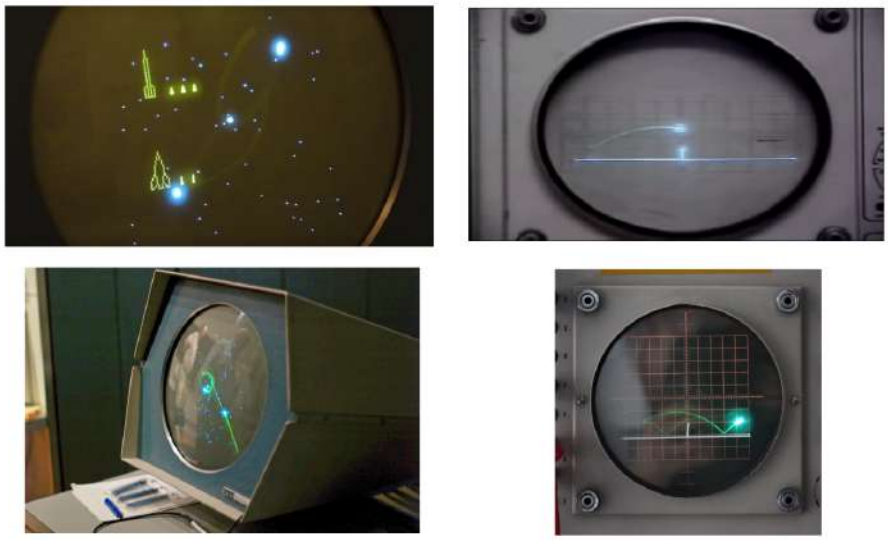


Figura 1.2.2: Imágenes del 'Spacewar!' (izquierda) y 'Tennis for Two' (derecha).

Durante los años 70 se crearon las máquinas recreativas, las cuales fueron ganando popularidad en los siguientes años. En la misma década, se comercializaron algunos de los primeros sistemas domésticos, como la Oddysey 2 o la Atari 2600. A medida que la industria crecía, la importancia de la estética en los videojuegos aumentaba: una portada o un diseño llamativo en una máquina arcade atraería a un mayor número de jugadores. Durante estos años nacieron juegos como 'Space Invaders' (1972) [84], 'Pac-Man' (1980)

[48] o 'Pole Position' (1982) [49]. Estos juegos tenían gráficos en 2D, muy pixelados y con colores muy brillantes, como se puede ver en las imágenes de la Figura 1.2.3.

(a) 'Space Invaders'.

(b) 'Pac-Man'.

(c) 'Pole position'.

Figura 1.2.3: Ejemplos de algunos videojuegos arcade.

En 1987, Nippon Electric Company lanzó al mercado la consola TurboGrafx-16 (comercializada con el nombre de PC Engine en Japón), la primera consola con un microprocesador de 16 bits (Adell, 2016). Esto significó un gran avance tecnológico en el momento y dio comienzo a la que se conoce como **era de los 16-bit** de los videojuegos. PC Engine tuvo bastante éxito en Japón, pero se vio derrotada por Nintendo y Sega, creadoras de las consolas Super Nintendo Entertainment System (1991) (conocida como Super Famicom en Japón) y Sega Genesis (1989) (llamada Mega Drive originalmente), respectivamente (Kent, 2016). En esta generación, estas dos últimas presentaron juegos de sus franquicias más famosas: Mario, con el videojuego 'Super Mario World' (1991) [51]; y Sonic en su juego 'Sonic The Hedgehog' (1991) [72]. También en estos años se publicaron algunos juegos que se conocen como clásicos a día de hoy, como son 'The Legend of Zelda: A Link To The Past' (1991) [50] o 'Chrono Trigger' (1995) [76].

Visualmente, los videojuegos de esta generación son muy distintos a los desarrollados hasta este momento. La paleta de colores aumentó significativamente, permitiendo más variedad y más detalle en los diseños de personajes y escenarios (Solórzano Alcívar y cols.,

2019). También se usaron fondos más complejos, con varias capas y distintas velocidades de desplazamiento que permitían una mayor sensación de profundidad. Algunos ejemplos de esto se pueden ver en la Figura 1.2.4.

Figura 1.2.4: Juegos de la era de los 16 bits.

Al mismo tiempo, en el campo de los videojuegos para PC se empezaron a desarrollar los primeros juegos en 2.5D y 3D. El 2.5D consiste en la combinación de elementos 2D y 3D para dar una sensación tridimensional. Existen distintas maneras de hacer esto, pero una de las más comunes es tener escenarios pre-renderizados en 3D y personajes y otros elementos en 2D. Por ejemplo, se usa 2.5D en 'Doom' (1993) [32] y 'Wolfstein 3D' (1992) [31], y 3D en 'Alone in the Dark' (1992) [33] (Pitko, 2022). A esta moda se sumaron también Nintendo y Sega, con algunos juegos para sus nuevas consolas, por ejemplo 'Donkey Kong Country' (1994) [52], publicado por Nintendo (Belli y Raventós, 2008). Estos juegos son representativos por usar gráficos 3D con aspecto muy pixelado (principalmente causado por texturas con poca calidad), una iluminación poco natural y elementos 3D formados por pocos polígonos, con poco detalle y formas angulosas, como se puede ver en la Figura 1.2.5.

Figura 1.2.5: Ejemplos de videojuegos en 2.5D y 3D.

En 1992, Sega publicó 'Virtual Racing' [73], el primer juego poligonal que tuvo popularidad. Este juego tuvo bastante éxito y provocó un gran cambio en la estética de juegos 3D: se fue dejando atrás el 2.5D y los gráficos pixelados, para empezar a usar una mayor parte

del juego en 3D, con modelos muy angulosos y texturas muy sencillas. Algunos ejemplos de estos juegos se pueden ver en la Figura 1.2.6.

Esta forma de usar gráficos 3D fue ganando popularidad a finales de los 90 con consolas como la Sony PlayStation o la Nintendo 64, pertenecientes a las generaciones de 32 y 64 bits de las consolas respectivamente (Belli y Raventós, 2008). En estas se publicaron juegos muy conocidos, como son 'Final Fantasy VII' (1997) [77] y 'Super Mario 64' (1996) [53].

Figura 1.2.6: Videojuegos con gráficos 3D poligonales.

A partir de ese momento, los avances tecnológicos permitieron gráficos 3D con más y más detalle. Se pasó de tener sombras duras a conseguir iluminaciones más realistas; de modelos 3D muy angulosos a otros con aspecto suave; y de texturas muy pixeladas y de baja calidad, a otras que permiten una gran cantidad de detalle. El objetivo durante las décadas de los 2000s, 2010s e incluso hasta la actualidad, ha sido conseguir imágenes lo más realistas posible, lo cuál se ha ido consiguiendo y perfeccionando a lo largo de los años a pasos agigantados. Esto se puede comprobar con las adaptaciones modernas de juegos antiguos, como se puede ver en la Figura 1.2.7.

Figura 1.2.7: Comparación en los gráficos del 'Final Fantasy VII' (1997) y del 'Final Fantasy VII: Remake' (2020).

En la actualidad, cualquier persona puede tener acceso a un ordenador en el que crear un videojuego. Esto abrió el camino a juegos mucho más variados y experimentales. Además de videojuegos con estética realista, existen muchos otros estilos visuales que exploran todo tipo de técnicas y de formas de transmitir su mensaje. Hoy en día se pueden encontrar juegos con estética retro como 'Cuphead' (2017) [81], que recuerda a las primeras películas de animación; juegos que usen cel-shading en su iluminación como 'Zelda: Breath of the Wild' (2017) [54]; que usen un estilo visual en pixelart, como 'Octopath Traveler' (2018) [78]; o incluso que combinen varias técnicas, como se hace en 'Inscryption' (2021) [22]. Gracias a todos los avances tecnológicos que ha habido hasta el momento, y los que habrá en un futuro, los desarrolladores pueden experimentar con muchas técnicas visuales y conseguir transmitir su mensaje de una manera única. Estos ejemplos se pueden ver en la Figura 1.2.8.

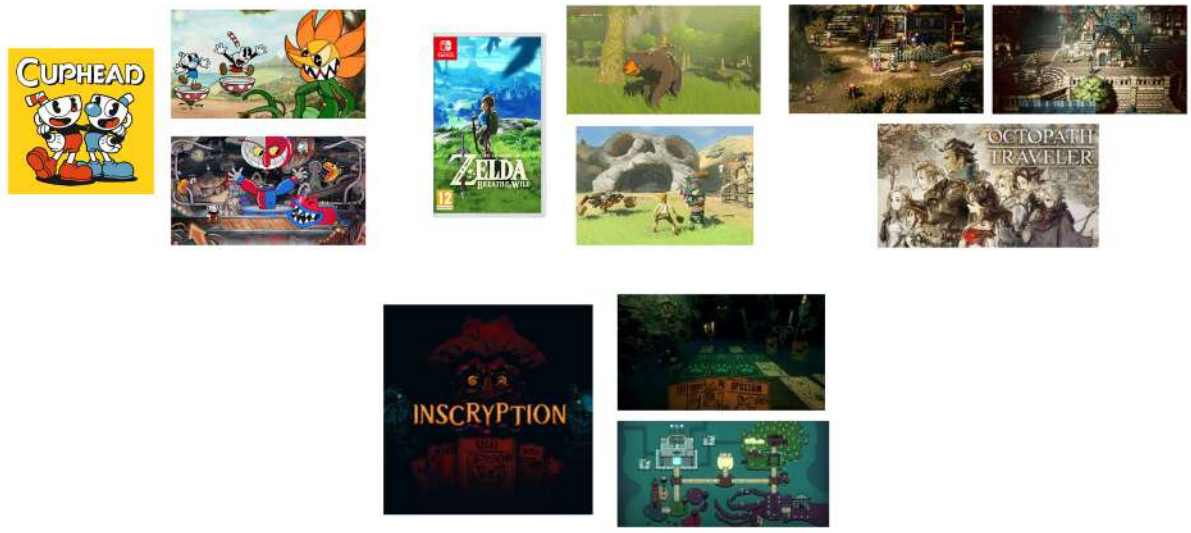


Figura 1.2.8: Algunos ejemplos de videojuegos actuales.

1.3. Flujo de trabajo en la creación de personajes 3D

Según Guillermo Tato, '*la narración normalmente gira alrededor de un individuo (o un objeto/animal dotado de personalidad)*' (Tato, 2017). Más adelante en su artículo, dice lo siguiente:

> '*Este personaje nos guía a través de una trama activa (interna o externa) y nos posiciona dentro del universo de la narración. Le da sentido (en relación casi siempre a las motivaciones y objetivos del personaje)*' (Tato, 2017).

De acuerdo con estas afirmaciones, el personaje protagonista es uno de los elementos más importantes en un videojuego, especialmente si se trata de videojuegos narrativos. Su estética, al igual que la de los videojuegos, ha estado muy unida a lo largo de la historia a los avances tecnológicos del momento: desde un estilo sencillo en pixelart a personajes 3D hiperrealistas o más cartoon .

El diseño y desarrollo de personajes para videojuegos es un proceso complejo con varias etapas que involucran a diferentes equipos de trabajo, como diseñadores, animadores o programadores. El flujo de trabajo a seguir es distinto cuando se trata de juegos 2D o juegos 3D. Además, estas etapas pueden variar ligeramente de un estudio de videojuegos a otro, dependiendo del número de personas que trabajen en el estudio, sus especialidades u otros factores.

Hoy en día es habitual ver que la creación de personajes 3D siga el siguiente flujo de trabajo:

1. Concept art y diseño del personaje.
2. Modelado 3D.
3. Texturizado.
4. Rigging y skinning.
5. Animación.

1.3.1. Concept art y diseño

A la hora de diseñar un personaje, independientemente de si se trata del protagonista, el enemigo o un personaje secundario, el objetivo es reflejar algunas de sus características en su apariencia física: su personalidad y carácter, su edad, su pasado, su ideología, etc..

Se define como concept art a '*las ilustraciones mediante las cuales se define la presentación visual de un diseño, una idea, una escena o un estado de ánimo antes de si representación final*' (Coterón, 2012). El concept es el primer paso a la hora de definir la identidad visual de un videojuego, ya que se usa para experimentar, proponer distintos diseños e ideas para los personajes, escenarios y otros elementos del juego (Oltra, 2020).

Antes de empezar a hacer concept, es importante definir lo máximo posible el proyecto que se está desarrollando. Por ejemplo, se debe definir si se busca de un estilo realista o cartoon, el género al que pertenece el juego, las sensaciones que se quieren transmitir, el público objetivo del mismo, etc. Todos estos datos ayudarán a los diseñadores a definir una identidad visual en la que encajar sus diseños.

A partir de las ilustraciones creadas como concept art, se comienzan a diseñar de una forma más detallada los elementos del juego (personajes, objetos, escenarios, etc.). El concept y el diseño van muy de la mano, por lo que en algunas ocasiones se consideran una misma etapa.

A la hora de hacer concept y diseñar es habitual hacer uso de softwares de dibujo digital o de edición de imagen, como Photoshop [4] o Clip Studio Paint [19], aunque algunos artistas prefieren hacer los bocetos iniciales de forma tradicional, con papel y lápiz. Existen muchos ejercicios que ayudan a los diseñadores a llegar al aspecto final del personaje o elemento que estén diseñando. Algunos de los más populares son:

- Dibujar una gran variedad de siluetas o bocetos y elegir las que sean más interesantes y transmitan mejor la esencia del personaje.

- Hacer bocetos de las distintas partes del objeto o personaje. Por ejemplo, probar varios peinados o distintas prendas de ropa y complementos. Un ejemplo de esto se puede ver en la Figura 1.3.1.
- Pintar el personaje o elemento con distintas paletas de color, para elegir la más atractiva.
- Especialmente si se están diseñando personajes, dibujar distintas expresiones o acciones que pueda tener.

Figura 1.3.1: Concept art para el personaje de Zelda en 'The Legend of Zelda: Tears of the Kingdom'.

Una vez terminado el diseño, se debe hacer un turnaround que facilite el trabajo del equipo de modelado. Además, cuando se diseñan personajes es bastante común montar una model-sheet (en español, hoja de personaje), en la que se añaden más detalles del diseño, como puede ser una ilustración más detallada o una ilustración en la que se muestre en relación a otros elementos del juego. El objetivo de una model-sheet es presentar su apariencia y sus cualidades de forma rápida. Al finalizar esta etapa se deben haber definido claramente el aspecto de los personajes o elementos que se hayan diseñado, de forma que, si una persona distinta se encarga de hacer los modelos, sea capaz de hacerlo sin ningún problema.

1.3.2. Modelado 3D

Una vez se ha definido la apariencia del personaje, se hace el modelo 3D de este. En esta etapa es importante tener en cuenta en qué tipo de proyecto se va a emplear el modelo: en videojuegos, suele ser importante que el modelo no tenga una cantidad excesiva de polígonos, ya que esto comprometería el rendimiento del mismo; sin embargo, esta preocupación no es tan tangible en el mundo del cine o series. El equipo también debe saber si

el personaje que estén desarrollando se va a animar o no, ya que esto condicionará la forma de la malla del modelo. Por ejemplo, si el personaje va a tener varias expresiones faciales, la malla de la cara debe tener geometría suficiente para que al mover las distintas partes, la cara no se deforme. Además, en caso de que el personaje vaya a tener animaciones, lo ideal es modelarlo en pose A para que la malla no sufra demasiadas deformaciones al animar.

El trabajo de los modeladores es plasmar el diseño del personaje o elemento en un modelo 3D. Una de las prácticas más habituales es usar como referencia el turnaround proporcionado por los diseñadores, colocando las distintas vistas de la ilustración para que sirvan como guía a la hora de modelar. Usando estas referencias podrán crear el modelo 3D, buscando siempre tener una malla equilibrada y con mayor cantidad de polígonos en zonas que vayan a necesitar moverse al animar, como pueden ser las articulaciones o la cara.

Hay dos caminos que se pueden tomar a la hora de hacer un modelo 3D: primero esculpir y después hacer la retopología , o directamente modelar modificando los vértices, caras y aristas de un objeto básico. Ambas opciones tienen sus ventajas y desventajas. Al esculpir, se modela el objeto como si fuese plastilina. Esto puede ser fácil y se pueden conseguir resultados muy naturales, pero requiere hacer retopología posteriormente y puede ser un proceso muy tedioso. Por otra parte, conseguir un modelo natural a través del modelado tradicional es más complicado, pero evita la necesidad de hacer una retopología, ya que se tiene más control sobre la geometría del objeto.

La decisión de usar una técnica u otra suele ser del modelador, pero también depende del software que se esté utilizando y del resultado que se quiera obtener. Si se busca conseguir modelos con formas orgánicas, suele ser más sencillo usar la técnica de esculpido. Los softwares más populares para esculpir son ZBrush [45] y Cinema 4D [44]. Para modelar elementos no orgánicos, suele ser mejor modelarlos de forma tradicional. Algunos de los softwares más usados para esto son Maya [13] y 3ds Max [12]. No obstante, ambas técnicas se pueden usar para cualquier modelo, e incluso existen muchos programas de modelado que permiten las dos de forma cómoda. Por ejemplo, el software **Blender** [15] es uno de los más usados hoy en día, ya que es gratuito y proporciona una gran variedad de herramientas: de esculpido, de modelado tradicional, de texturizado, etc.

1.3.3. Texturizado

Una vez terminado un modelo 3D, comienza el trabajo de texturizado, que consiste en dar color, distintos materiales o acabados a las distintas partes del personaje. Esta fase puede ser muy compleja, especialmente si se busca un nivel de detalle muy alto, por ejemplo, en personajes con un estilo hiperrealista.

El trabajo de texturizado se puede dividir en dos pasos: el mapeo de coordenadas y la aplicación de texturas.

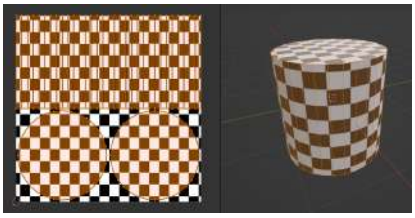
Mapeo de coordenadas

El mapeo de coordenadas consiste en asociar cada zona de la malla del modelo 3D una zona de una textura en 2D. A la imagen resultante de este proceso se le conoce como mapa de coordenadas, o mapa UV (Sastre, Rocha, Pequeño, y López, 2020). Este proceso

se puede realizar en casi cualquier software de modelado 3D, por ejemplo, en Blender o 3ds Max.

A la hora de hacer el mapeo UV en un modelo, es muy útil imaginar que la textura es una tela que cubre todo el modelo. El objetivo es dividir esa tela por piezas marcando dónde estarían sus costuras e intentando que estas costuras sean poco visibles. Al dividir un modelo por piezas, se pueden colocar todas ellas en un mismo plano y pintarlas sin que ninguna se solape ni se deforme.

Muchos softwares cuentan con herramientas que hacen este proceso de forma automática, pero suelen cometer fallos. Es por eso que, aunque estas herramientas puedan agilizar el trabajo, es conveniente revisar el mapeado y corregir lo que sea necesario de forma manual. Para comprobar que un mapeo está bien hecho, es bastante común usar un tipo de textura conocida como 'checker' o 'UV checker'. Estas son imágenes con cuadrículas de al menos dos colores (a ser posible, muy distintos entre ellos) y, en algunas ocasiones, con letras o números escritos en cada cuadrado. Si el mapeado es correcto, al aplicar esta textura al modelo no debería salir ningún cuadrado ni texto deforme, como se puede ver en la Figura 1.3.2.



(a) Objeto con un **buen** mapeo de UVs y una textura checker.

(b) Objeto con un **mal** mapeo de UVs y una textura checker.

Figura 1.3.2: Demostración del uso de una textura UV checker y de un buen y mal mapeo de UVs.

Uno de los objetivos que se deben tener en mente a la hora de hacer el mapeo, es intentar aprovechar el espacio de la textura al máximo. Por lo tanto, se deben intentar colocar todas las piezas del modelo optimizando el espacio en la textura. Cuanto más espacio en el mapa ocupe una pieza, mayor calidad tendrá su textura. Teniendo en cuenta esto, si una zona del modelo necesita tener mucho detalle o va a ser muy visible, es interesante que ocupe más espacio que una zona que apenas va a ser visible. Al igual que ocurría con la tarea anterior, la mayoría de softwares tienen herramientas para hacer esto de forma automática. Sin embargo, en este caso sí suelen dar buenos resultados, aunque conviene revisarlos siempre.

Aplicación de texturas

Existen varias formas de aplicar texturas a los objetos 3D: pintando directamente en la imagen 2D de la textura o usando programas como Substance 3D Painter, que permiten pintar en los objetos directamente, como si se tratara de pintar una figura de arcilla a mano. Este segundo método es el más popular hoy en día y hay varios softwares que permiten esta funcionalidad, aunque Substance 3D Painter es uno de los más utilizados.

Aplicar una textura a un modelo no consiste simplemente en darle color, sino en darle todo tipo de detalles y acabados. Es muy común usar materiales ya existentes en las distintas partes del modelo, por ejemplo, materiales de piel o de distintos tipos de tela para la ropa. Un material define la forma en que las superficies responden a la luz y su aspecto general (Sastre y cols., 2020). Para ello, un material combina varios tipos de mapa de textura. Algunos de los mapas más usados en los materiales son:

- **Mapa de color**: mapas que definen el color base de los objetos.
- **Mapa especular**: mapa en escala de grises que define cuánto refleja la luz en una superficie. En zonas en que el mapa se acerque al blanco, el objeto reflejará más la luz y ofrecerá un aspecto similar un objeto. Por otro lado, si se acerca más al negro en una zona, reflejará menos luz y dará una sensación parecida a la goma.
- **Mapa de desplazamiento**: mapa en escala de grises que permite simular relieve en ciertas zonas del modelo en las que no exista ese relieve en la geometría. Las tonalidades más oscuras en estos mapas representan una mayor profundidad, es decir, como si la geometría se 'metiese hacia dentro'. Por el contrario, los tonos claros dan la sensación de que parte de la geometría tenga un relieve hacia fuera.
- **Mapa de relieve** (o bump mapping, en inglés): mapa que se usa para modificar las normales de un objeto, consiguiendo, por ejemplo, superficies rugosas o arañadas.

Herramientas como Substance 3D Painter [5] permiten usar materiales de una amplia biblioteca, combinarlos y modificar sus propiedades para conseguir el efecto deseado. Una vez terminado el texturizado, se deben exportar los distintos mapas de textura utilizados en el modelo, para poder incluirlos luego en próximas fases del proyecto, por ejemplo, en el motor de juego o en el software donde se vaya a animar.

1.3.4. Rigging y Skinning

Las fases de rigging y skinning son las que permiten que los animadores puedan mover las distintas partes del personaje. Ambas tareas están muy relacionadas entre sí y deben hacerse una tras otra, por lo que en muchas ocasiones se agrupan en una misma fase en el proceso de animación 3D.

El rigging consiste en crear un esqueleto virtual que se adecúe al modelo 3D. Este esqueleto está formado por varios huesos y articulaciones, de forma que al mover estos, se moverá el personaje. La mayoría de softwares de animación cuentan con esqueletos humanoides prefabricados, los cuales se pueden modificar para que encajen con el modelo desarrollado, por ejemplo, haciendo los huesos de sus brazos o piernas más cortos o añadiendo más huesos en caso de que sea necesario (Stoneham, 2012). Es importante entender que el esqueleto virtual no tiene porqué parecerse a un esqueleto real, sino que debe tener los huesos necesarios para controlar todas las partes del personaje que vayan a tener movimiento. Por ejemplo, es habitual crear huesos (no existentes en la vida real) para animar el movimiento del pelo o de las distintas partes de la cara, como as cejas o la boca. Por el contrario, no se suelen definir los huesos de las costillas, ya que no se mueven de forma independiente al resto del tronco.

El proceso de skinning consiste en asignar la influencia que tiene cada hueso del esqueleto en cada vértice del modelo. Cada vértice puede verse afectado por varios huesos, en mayor o menor medida. Por ejemplo, un vértice que se encuentre en el centro del antebrazo, normalmente solo estará afectado por este hueso; pero un vértice cerca de la muñeca, puede estar influenciado también por el movimiento de los huesos de la mano. Algunos softwares, como por ejemplo Blender, permiten hacer este proceso de forma automática, ya que puede ser un trabajo muy largo y tedioso. No obstante, siempre es recomendable revisar y corregir el resultado, ya que suelen tener errores, especialmente en zonas en las que haya muy poca distancia entre huesos.

Al finalizar este trabajo, se hacen usos de animaciones sencillas, que sirvan como prueba de que la malla se ajusta correctamente en todo momento. Además, en muchas ocasiones se desarrollan una serie de controladores que permitan al animador generar los movimientos únicamente modificando ciertos valores, en lugar de mover directamente los huesos del personaje. En la Figura 1.3.3 se puede ver un ejemplo de los resultados que se deberían obtener al terminar estas etapas.

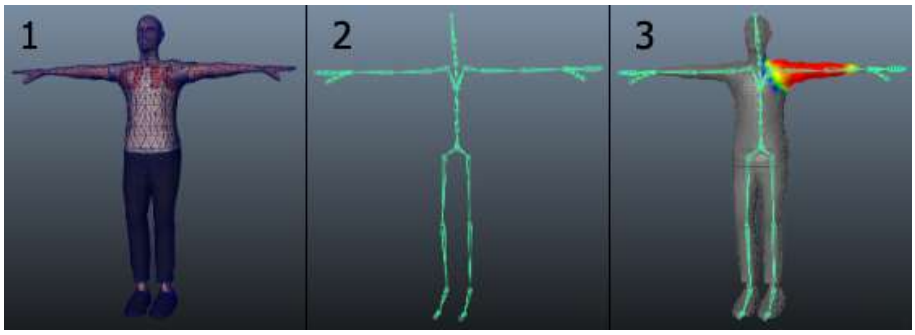


Figura 1.3.3: Imágenes del resultado de las fases de modelado, rigging y skinning.

1.3.5. Animación

La etapa de animación consiste en dar movimiento al personaje. Para ello, el animador irá moviendo los huesos del personaje y grabando las distintas poses de la animación. Esto se puede hacer en programas como Blender, 3ds Max o Cinema 4D, entre otros. En videojuegos, las animaciones para las distintas acciones de un personaje se hacen por separado, de forma que en el motor de juego se pueda decidir en qué momento ejecutar cada una. Es muy común tener animaciones que se ejecutan en bucle, por ejemplo, para que un personaje ande o corra se ejecuta en bucle una animación de un ciclo de caminado, es decir, un paso. No obstante, acciones como 'atacar' o 'coger algo' no suelen hacerse en bucle, por lo que estas no se preparan para que se ejecuten de esa manera.

Antes de comenzar a animar es muy importante buscar referencias del movimiento que se quiere realizar. Estas referencias pueden venir de videos de internet, de vídeos tomados por los propios animadores o de otras animaciones ya existentes, y permiten al animador entender mejor el movimiento. Además, es interesante que los animadores conozcan los Doce Principios de la Animación (ver Figura 1.3.4), definidos en 1981 por Ollie Johnston y Frank Thomas (Adobe, s.f.-a; Johnston y Thomas, 1981). Estos principios ayudan a que las animaciones sean más interesantes y fáciles de comprender para el espectador/jugador y son especialmente interesantes cuando se está desarrollando un proyecto con una estética no realista.

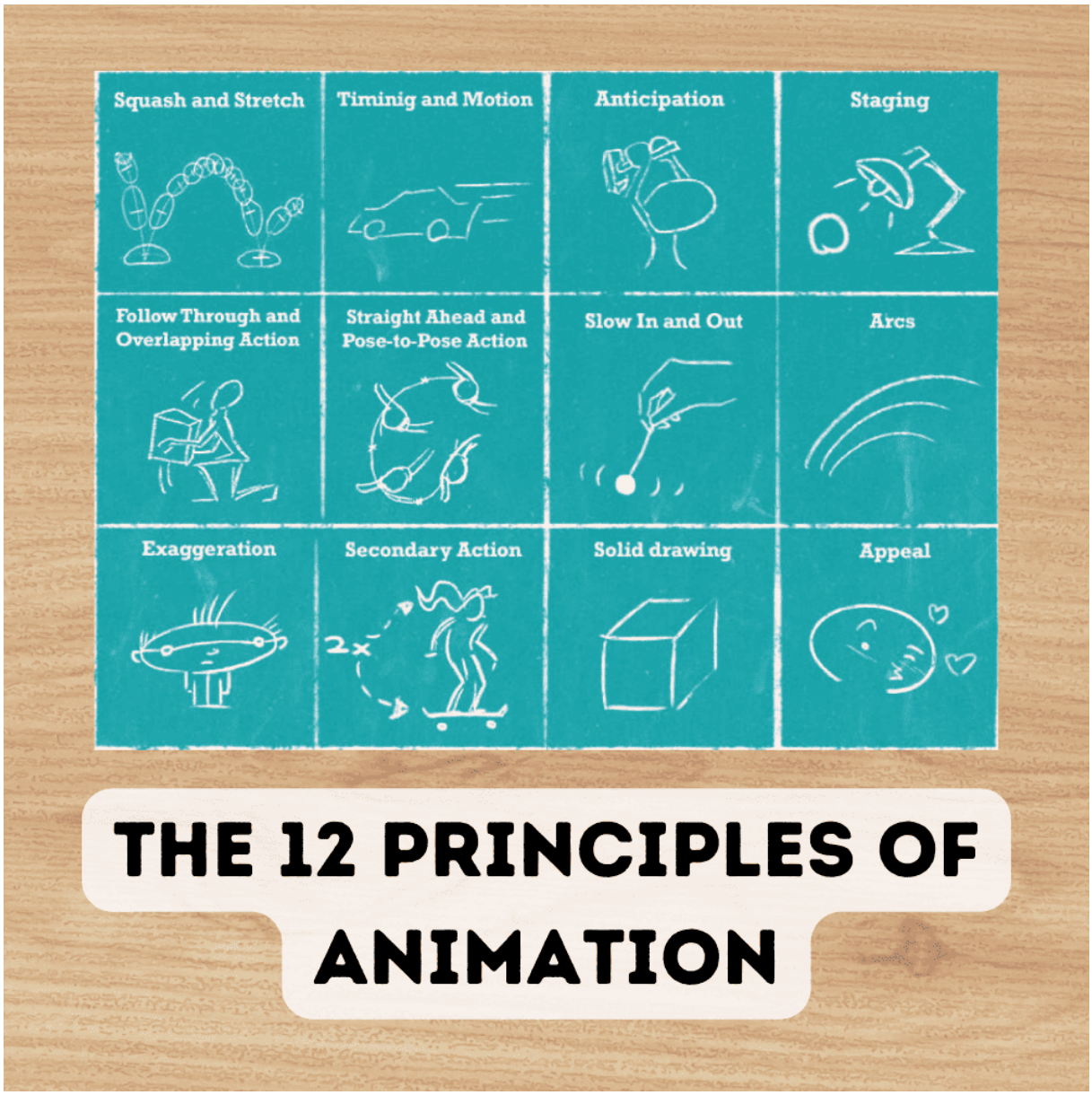


Figura 1.3.4: Doce Principios de la Animación de Ollie Johnston y Frank Thomas.

Los principios más relevantes a la hora de hacer animaciones para videojuegos son:

1º. Estirar y encoger: consiste en deformar a los personajes u objetos con el movimiento para conseguir que estos parezcan pesados y flexibles.

2º. Anticipación: sirve para preparar al espectador antes de la acción que vaya a realizar el personaje.

5º. Acciones complementarias y superpuestas: aporta realismo a los movimientos de los personajes. Estas acciones se refieren a las distintas partes del personaje que se mueven a distintas velocidades, o que continúan el movimiento una vez el personaje se ha parado. Este efecto se suele apreciar bien en el pelo o ropa holgada del personaje, que comenzará a moverse ligeramente después que este y le costará más frenarse.

6º. Acelerar y desacelerar: los objetos en movimiento ganan y pierden velocidad gradualmente. Esto mismo se puede aplicar a personajes, que tardarán unos instantes en coger velocidad al inicio de la acción, e irán frenando a medida que esta termina.

7º. Arcos: de forma natural, la trayectoria de objetos en movimiento suele formar arcos. Usar arcos en los movimientos de los personajes les dará fluidez.

8º. Acciones secundarias: ayudan a que los personajes se muevan más naturales. Por ejemplo, mover los brazos del personaje mientras anda, hace que parezca más humano.

9º. Sincronización: definir correctamente la duración de una animación o de las acciones dentro de ella harán que esta se vea más realista.

10º. Exageración: hace que las animaciones sean más interesantes para el espectador. Se puede exagerar de forma más sutil o más evidente, dependiendo de la estética que se quiera conseguir.

12º Atractivo: intentar que todos los personajes sean atractivos e interesantes para el jugador/espectador, sin importar si son protagonistas, antagonistas o secundarios.

Una vez hechas las animaciones, se pueden exportar y dar por terminado el desarrollo del personaje y se podrá incluir en el motor de juego para continuar ahí con el trabajo.

1.4. Objetivos

El objetivo principal de este trabajo es la construcción de una galería de personajes para un videojuego, desde el diseño hasta la inclusión en el motor de juego, pasando por las distintas etapas descritas en la sección 1.3. Para poner en práctica este proceso se va a plantear el diseño de un videojuego y desarrollar el trabajo en algunos de sus personajes, detallando sus diseños y haciendo algunas de sus animaciones básicas. Esto servirá para ejemplificar y entender mejor el flujo de trabajo que se utiliza hoy en día en la industria de los videojuegos.

Para completar este objetivo principal deben llevarse a cabo una serie de objetivos específicos:

Obj 1. Diseñar el videojuego.

Se debe plasmar el diseño del juego en un Game Design Document (abreviado como GDD). En este documento se deben exponer algunas de las características principales del mismo, como el género al que pertenece, sus mecánicas o los personajes que se pueden encontrar en el mismo. Además, se debe decidir qué personajes serán los que se van a diseñar más detalladamente y animar en el proyecto.

Obj 2. Diseñar y desarrollar en 3D tres de los personajes del juego.

Se debe llegar a un diseño final de cada uno de los personajes elegidos, crear un modelo 3D de los mismos y texturizarlo.

En primer lugar, se deben diseñar conceptualmente cada uno de los personajes y definir algunos rasgos de su personalidad. A partir de estas características se realizarán bocetos hasta llegar al diseño final, el cual se debe reflejar en una ilustración turn-around que muestre como mínimo una vista frontal y otra lateral.

A partir del turn-around, se debe modelar cada personaje y, posteriormente, texturizarlo. Para ello, se debe seguir el proceso habitual de hacer el mapeo de UVs y aplicar las texturas al modelo.

Obj 3. Implementar las animaciones básicas de los tres personajes.

Se deben preparar los modelos para animar, haciendo el rigging y el skinning de los mismos. Una vez hecho esto, se deben elegir al menos tres acciones para los personajes e implementar las animaciones para estas, idealmente las mismas acciones en todos ellos.

Obj 4. Incluir a cada personaje, junto con sus animaciones, en el motor de juego Unity [90].

Se deben exportar los modelos 3D de los personajes junto con sus animaciones e importarlos en el motor de juego Unity. En Unity, se deben crear controladores para las animaciones, de forma que se pueda cambiar de una a otra usando controles en el teclado. De forma que los personaje queden preparados para su futura inclusión en el videojuego.

1.5. Metodología

Una metodología de trabajo es un modelo que indica en qué orden se realizan las distintas tareas del proceso de desarrollo (Ojeda y Fuentes, 2012). En el campo del desarrollo informático son muy populares dos familias de metodologías de trabajo: las metodologías tradicionales y las metodologías ágiles.

Las **metodologías tradicionales** afrontan un proyecto entendiéndolo como un 'todo', sin divisiones. Su característica mas identificativa es que se definen los requerimientos del proyecto al inicio del mismo, y no se suelen modificar en ningún momento (Montero, Cevallos, y Cuesta, 2018). Esta familia de metodologías fomenta un flujo de trabajo bastante rígido, sin prácticamente comunicación con el cliente hasta el final del proyecto, cuando se realizan las pruebas y la evaluación del resultado. Esta rigidez puede provocar algunos problemas, por ejemplo, que se puedan encontrar errores al final del desarrollo, cuando ya son difíciles de solucionar, o que el cliente quiera hacer cambios en el diseño una vez terminado el trabajo. Otra de las características principales de esta familia de metodologías es la gran importancia que se le da a tener buena documentación del proyecto. Además, estos modelos están ideados para grupos de trabajo grandes y con una gran variedad de roles, por lo que es importante que todas las personas implicadas en el proyecto puedan consultar la documentación en cualquier momento (Canós, Letelier, y Penadés, 2003). Esto puede ocupar gran parte del tiempo de los desarrolladores, por lo que no siempre es una ventaja.

A esta familia pertenecen metodologías como la metodología incremental o la metodología en espiral, pero, sin ninguna duda, la más popular es la metodología en cascada (Rivas, Corona, Gutiérrez, y Hernández, 2015). Este modelo propone cinco etapas: análisis de requerimientos, diseño, implementación, verificación y mantenimiento del proyecto. Estas etapas deben realizarse de forma secuencial y en ese orden (Montero y cols., 2018). Gráficamente, se suele representar como se muestra en la Figura 1.5.1.

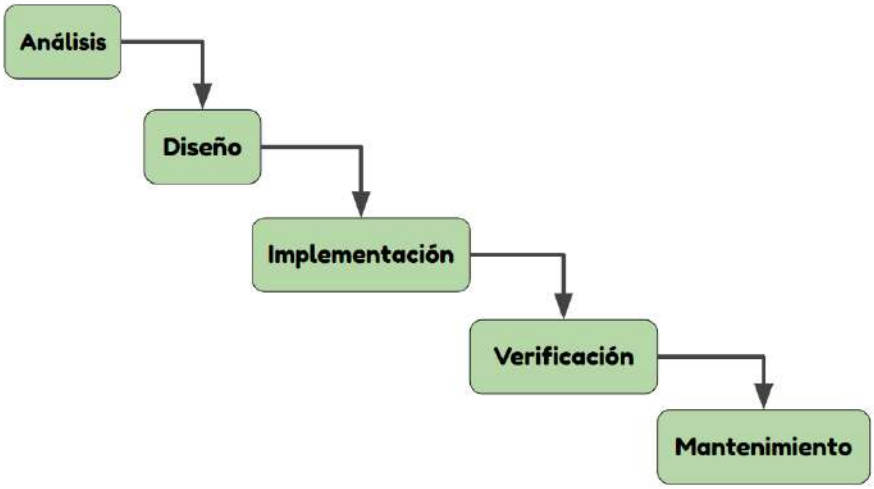


Figura 1.5.1: Etapas de la metodología de desarrollo en cascada.

Por otra parte, la familia de metodologías ágiles destacan por ser muy flexibles y estar bien preparadas para cambios (Canós y cols., 2003). Es por esto que este tipo de metodologías son ideales en proyectos de innovación o en los cuales se quiera tener mucha retroalimentación por parte del cliente, de forma que el resultado final se adapte completamente a las necesidades de este.

Una de las principales características de las metodologías ágiles es concebir el proyecto como un conjunto de tareas más pequeñas. De esta forma, el equipo de desarrollo podrá repartirlas y enfrentarse a ellas más fácilmente, además de ir revisando su progreso constantemente. Al contrario que en las metodologías tradicionales, las ágiles recomiendan equipos de trabajo pequeños y no dan tanta importancia a la documentación, centrándose más en la comunicación dentro del equipo y con el cliente.

La metodología Scrum es una de las metodologías ágiles más conocidas. En esta metodología, el proyecto se organiza en sprints (o iteraciones), los cuales representan un periodo de tiempo de trabajo. En cada sprint se deben realizar una serie de tareas, las cuales se deben revisar al final del mismo en una reunión que recibe el nombre de 'Sprint Review' (en español, revisión del sprint). En la reunión se debe comentar cuánto se ha tardado en hacer las tareas frente al tiempo estimado para las mismas, las dificultades que se hayan encontrado, etc. con el fin de adaptar los siguientes sprints si fuera necesario. Además, una de las características más significativas de Scrum es que el equipo debe tener reuniones diarias, en las que todos los miembros informarán de sus avances o plantearán cualquier cuestión que necesiten (Schwaber y Sutherland, 2013). Al final de cada sprint, se presentan los avances hechos al cliente, de forma que este pueda dar retroalimentación y proponer cambios si lo ve necesario.

Scrum es una metodología que busca dar mayor importancia a las personas sobre a los procesos y herramientas (Turley y Rad, 2019). Es por esto que su ideal es tener grupos de

trabajos pequeños, en los que la comunicación sea fácil y fluida dentro del propio equipo y con el cliente. Estos equipos deben estar formados por desarrolladores, un Scrum Master, que se encarga de que todos en el proyecto conozcan y lleven a cabo la metodología Scrum, y un Product Owner, cuyo trabajo es asegurarse de conseguir el mejor resultado posible por parte del equipo (Scrum.org, 2016).

Teniendo en cuenta los aspectos más significativos de ambas metodologías, se ha decidido planificar el proyecto como un desarrollo ágil con algunas adaptaciones para este proyecto concreto. Las razones que han motivado esta decisión son las siguientes:

- Es una familia de metodologías con las que la persona encargada del proyecto ha trabajado en varias ocasiones, por lo que le será fácil de implementar.
- Revisar el trabajo realizado al final de cada sprint puede ayudar a encontrar fallos y poder corregirlos antes del final del proyecto, además de evitar que se cometan los mismos fallos en un futuro.
- El equipo de trabajo está formado por únicamente una persona, por lo que invertir mucho tiempo en producir documentación puede ser ineficiente.
- Se puede dividir el proyecto fácilmente en varias tareas, de forma que sea más sencillo enfrentarse al trabajo.

Por estos motivos, se ha planteado una metodología en la que el trabajo se divide en varios sprints. Todos los sprints tendrán una duración similar y al final de cada uno de ellos se realizará una pequeña reunión de revisión en la que se podrán hacer cambios en las tareas o la duración de futuros sprints, intentando estimar lo mejor posible la duración del proyecto.

Adicionalmente, se va a hacer uso de un **Tablero Kanban**, una herramienta que permite ver claramente el progreso que se vaya haciendo en el trabajo. Estos tableros tienen una estructura con tres columnas: '*To Do*', '*Doing*' y '*Done*', como se puede ver en la Figura 1.5.2 (Yacelga y Cabrera, 2022). Se va a utilizar la herramienta 'Trello' para crear y modificar este tablero a lo largo del desarrollo.

Figura 1.5.2: Estructura de un tablero Kanban.

En conclusión, para alcanzar los objetivos expuestos en la sección anterior se ha planteado una metodología ágil, formada por varios sprints.

1.6. Planificación

Siguiendo la metodología descrita en la sección 1.5 y con el fin de completar los objetivos descritos en la sección 1.4, se han definido una serie de tareas para el proyecto y se han repartido en varios sprints. Además, se ha hecho una estimación inicial de la duración de cada uno de los sprints, teniendo en cuenta las tareas que los componen y que se harán jornadas de alrededor de 6 horas de trabajo.

- **Sprint 1** (11 días)

  El primer sprint del proyecto es significativamente más corto que los demás, ya que en este únicamente se ha planificado una tarea: desarrollar un el diseño del juego. Al final de este sprint, se debe haber resuelto el primer objetivo específico y se habrá plasmado en un documento todo el diseño del videojuego, del que partirán las siguientes etapas del trabajo.
  Obj 1. Diseñar el videojuego.

- **Sprint 2** (25 días)

  El segundo sprint se centra en el diseño y desarrollo de uno de los personajes elegidos. En este sprint se incluyen tareas correspondientes a cada una de las acciones descritas en la sección 1.3: el diseño conceptual y visual del personaje, su modelado, su texturizado, el rigging y skinning de este y la creación de tres animaciones. Además de la tarea de inclusión de este personaje en el motor de juego.

  Al finalizar este sprint se habrá completado parte de los objetivos específicos 2, 3 y 4 (ver sección 1.4).

- **Sprint 3** (25 días)

  En el tercer sprint, el trabajo se centra en desarrollar el segundo personaje, completando las mismas tareas definidas para el segundo sprint. Las tareas correspondientes a esta iteración contribuyen asimismo a los objetivos 2, 3 y 4.

- **Sprint 4** (25 días)

  Este último sprint se realizarán las mismas tareas que en los dos anteriores, pero esta vez para el último de los tres personajes. Una vez se haya terminado este sprint, se podrán dar por finalizados los objetivos específicos restantes y, en consecuencia, el objetivo principal del proyecto.

  Obj 2

  Diseñar y desarrollar en 3D tres de los personajes del juego.

  Obj 3

  Implementar las animaciones básicas de los tres personajes.

  Obj 4

  Incluir a cada personaje, junto con sus animaciones, en el motor de juego Unity.

  Objetivo principal

  Construir una galería de personajes para un videojuego.

1.7. Estructura de la memoria

Tras este capítulo introductorio en el que se han presentado las motivaciones del proyecto y los objetivos que se van a perseguir en este, en los capítulos 2, 3, 4 y 5 se van a describir el desarrollo y los resultados obtenidos en las tareas de los sprints 1, 2, 3 y 4, respectivamente. El trabajo en estos capítulos corresponderá con lo descrito en la sección 1.6, en la que se ha explicado la planificación y los tiempos de trabajo estimados para este proyecto. Por último, en el capítulo 6 se dará por finalizado el proyecto, citando los objetivos cumplidos y proponiendo una serie de futuros trabajos.

Capítulo 2

Diseño del juego - Sprint 1

Como se describió en la sección 1.6, el primer sprint sprint es el más corto de todos e incluye solo una tarea, el diseño del juego. En la industria de los videojuegos es muy común plasmar el diseño de un juego en un Game Design Document (abreviado en inglés como GDD). Un GDD es un documento en el que se detallan las distintas características del juego para que todos los miembros del equipo de trabajo puedan acudir a él en caso de que tengan alguna duda de diseño (Conway, 2021). Normalmente, en un GDD se explican, como mínimo, el género al que pertenece el juego, sus mecánicas, sus personajes y su sinopsis, aunque la estructura de este documento puede variar a gusto del equipo de diseño.

En este sprint, se ha creado un GDD en el que se definen algunas de las características más importantes del videojuego.

2.1. Game Design Document del videojuego '¡Al ladrón!'

Título: ¡Al ladrón!

Diseñadora: Lucía Fresno Olmeda

Género: '¡Al ladrón!' pertenece al subgénero *endless runner*, también conocido como *infinite runner* (corredor infinito).

En este subgénero, perteneciente al género de videojuegos de plataformas, los jugadores avanzan de forma infinita por un camino, evitando obstáculos. El objetivo en estos videojuegos es aguantar el máximo tiempo posible en la partida. Este tipo de juegos no cuenta con una condición de victoria, por lo que la meta de los jugadores siempre será batir sus propias marcas, o las de otros jugadores. Sin embargo, siempre tienen una condición de fin de partida muy clara, en general, haber chocado con varios obstáculos.

Plataforma: Dispositivos móviles táctiles

Sinopsis y contenido:

'¡Al ladrón!' es un videojuego con una estética cartoon/infantil. En este juego, el usuario controla a un niño que está en un bosque recogiendo frutos. Al inicio de la partida, suenan

las campanas de un pueblo próximo y el niño se da cuenta de que es tarde y debe volver a casa, por lo que emprende su camino.

En su camino, varios animales intentarán robarle los frutos de su cesto. Para evitar esto, el niño debe estar atento y girarse si oye ruidos raros, espantando así a los animales. En el momento en que el jugador siga avanzando, los animales volverán a acechar su cesta. Además, debe tener cuidado de no tropezarse con ninguna roca, ya que se le caerá una pieza de fruta si lo hace.

Su objetivo será llegar lo más lejos posible sin que le roben. Si el jugador pierde tres frutos, terminará su partida.

Categoría:

El videojuego pertenece al género o *endless runner*, un estilo de juego muy popular en smartphones y tabletas en los últimos años. Algunos de los títulos más conocidos son 'Subway Surfers' (2012), 'Temple Run' (2011) o 'Zombie Tsunami' (2012) (ver Figura 2.1.1).

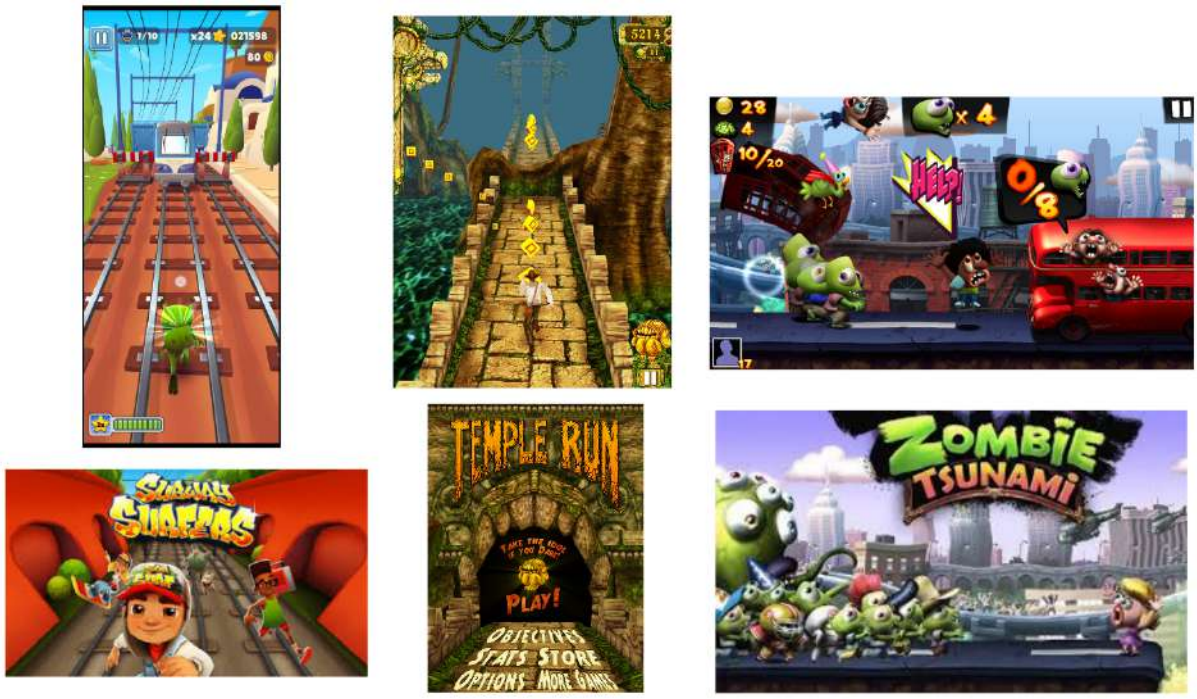


Figura 2.1.1: Ejemplos de videojuegos del género endless runner.

Para conocer mejor las fortalezas y debilidades de '¡Al ladrón!', se ha desarrollado el siguiente diagrama DAFO 2.1.

Análisis interno	**Análisis externo**
Debilidades	**Amenazas**
El equipo de desarrollo es muy pequeño, por lo que no cuenta con gente especializada para cada ámbito. Esto puede resultar en fallos o dificultades en algunos apartados técnicos.	Al ser un género tan conocido, es probable que el público compare este juego con otros parecidos, desarrollados por empresas más grandes y con más recursos.
Fortalezas	**Oportunidades**
Cuenta con algunas mecánicas innovadoras en este tipo de juegos: el jugador debe estar atento al sonido para evitar a los enemigos y debe tener cuidado de no gastar toda su energía.	Fácil para los usuarios, ya que conocen el funcionamiento gracias a otros juegos de este mismo género. Se puede dar a conocer como una 'innovación' del mismo.

Tabla 2.1: Análisis DAFO

Licencia: Videojuego creado a partir de una idea original.

Público: Público casual. Principalmente, personas de entre 13 y 30 años.

Mecánicas:

'¡Al ladrón!' es un videojuego en 3D con cámara en tercera persona. Sus controles son muy comunes en juegos *endless runner*: arrastrar el dedo a izquierda o derecha para moverse de un lado a otro; y un toque en la pantalla para mirar a los animales.

El sistema de puntuación es sencillo: se ganan 10 puntos por cada 5 segundos que el jugador pasa en la partida. Además, pasado un tiempo en la partida, empezarán a aparecer objetos que otorguen puntos en el camino y el jugador podrá cogerlos pasando por encima de ellos.

El sistema de guardado aparecerá en futuras versiones del juego, cuando este esté disponible para jugar *online* y entrar en rankings con otros jugadores. Sin embargo, para la primera versión no habrá funcionalidades de guardado y cargado de partida, ya que es un juego con partidas cortas e individuales.

Por último, es interesante conocer en detalle la jugabilidad de '¡Al ladrón!': una vez termina la cinemática inicial, el jugador se moverá automáticamente hacia delante por un camino. Este camino tendrá varios 'carriles', y el jugador podrá pasar de uno a otro moviéndose hacia los lados, para poder evitar los obstáculos que aparezcan en ellos. Al mismo tiempo, el jugador debe escuchar con atención y, si oye el ruido de algún animal cerca, debe girarse para espantarlo y que no le robe la fruta.

El jugador contará con una cantidad limitada de 'energía', la cuál se gasta cuando se gira para vigilar a los animales y se va recargando con el tiempo. Si pierde toda su energía, los animales le alcanzarán y le robarán fruta.

A medida que vaya avanzando la partida, el personaje se moverá más rápido, lo que dificultará el camino al jugador.

Estados del juego: La primera versión del juego cuenta únicamente con cuatro estados: menú principal, estado de juego, menú de pausa y menú de fin de partida.

La navegación entre estas cuatro pantallas es sencilla: se pasará al estado de juego desde el menú al pulsar el botón 'jugar'; al terminar la partida, saldrá el menú de fin de partida,

y el jugador elegirá si volver a jugar o si salir del juego. Además, si durante la partida el jugador pulsa el botón 'atrás', llegará al menú de pausa, donde podrá continuar su partida o salir del juego. En el diagrama de la Figura 2.1.2 se muestra el comportamiento descrito.

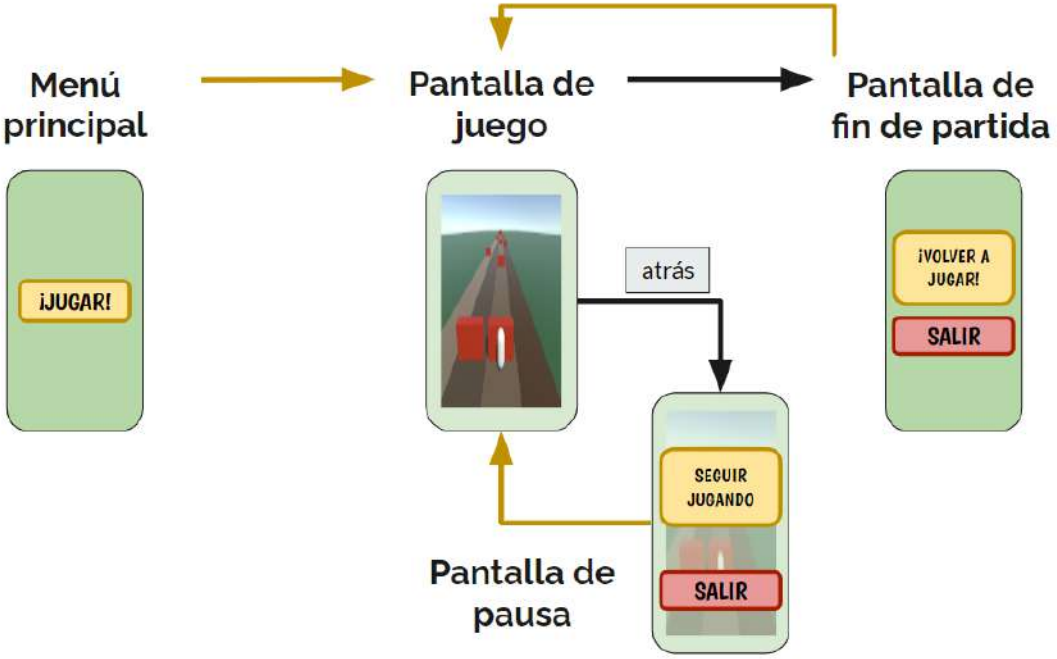


Figura 2.1.2: Diagrama de flujo de las pantallas

Interfaces:
En la primera versión del juego se pueden encontrar interfaces en cuatro pantallas, todas ellas con un estilo cartoon desenfadado, en las cuales predominan los colores verde y amarillo. En la Figura 2.1.2 se muestra el diagrama de flujo de la aplicación, además de un pequeño boceto de las distintas pantallas.

Las interfaces de '¡Al ladrón!' son:

- Pantalla de menú principal: es desde la cuál el jugador puede empezar una partida.
- HUD durante la partida: en esta interfaz el jugador podrá visualizar su puntuación y las 'vidas' que le quedan.
- Pantalla de pausa: interfaz en la cual el jugador tendrá la opción de seguir jugando o de salir del juego.
- Pantalla de fin de partida: es en la que se muestra la puntuación de la partida y se da la opción de de jugar otra vez o salir.

Niveles: No hay niveles diferenciados, pero la dificultad del juego irá aumentando durante la partida, a medida que el jugador avanza.

Personajes: '¡Al ladrón!' tiene a un único personaje principal: el niño al cuál controla el jugador. Para el primer prototipo no se ha desarrollado en profundidad su aspecto, pero sí se han reflejado en la Tabla 2.2 algunos aspectos claves del mismo.

Nombre	El personaje no tiene nombre propio.
Sexo	Masculino.
Edad	8 años.
Cualidades	Es un niño inocente, bueno y obediente. También es curioso, pero algo miedoso.
Trasfondo	El niño estaba en un bosque cerca de la casa de sus abuelos recogiendo frutos, para después hacer pasteles con ellos. Cuando oyó las campanas de la iglesia del pueblo sonar, se dio cuenta de que era tarde y decidió poner rumbo a casa. Es en este momento en el que comienza el juego.
Rol	Es el personaje que controla al jugador. Su único objetivo es que no le roben la fruta los animales, por lo que se dedica a huir de ellos.

Tabla 2.2: Ficha de personaje del niño

Enemigos: '¡Al ladrón!' Cuenta con tres enemigos: animales que intentan robarle la fruta al niño. Su papel en el juego es sencillo: perseguirán al jugador hasta acercarse lo suficiente para robarle, a menos que él se gire para vigilarlos, en cuyo caso se esconderán o huirán. Estos tres NPCs conforman la principal dificultad del juego.

Estos tres enemigos son tres animales que roban fruta a hurtadillas. Antes de diseñarlos, se han definido una serie de requisitos que cumplir:

- Deben estar basados en animales reales.
- Deben tener un tamaño pequeño o mediano, para conseguir una sensación inocente y divertida.
- Deben ser muy distintos entre sí, de forma que sus animaciones fuesen radicalmente diferentes.
- Deben tener algún elemento común que los vinculase visualmente.

Con todo esto en mente, se fue definiendo el concepto del grupo de enemigos. En primer lugar, se decidió que cada uno de los animales elegidos perteneciese a un grupo de los vertebrados: un mamífero, un ave y un reptil.

Dado que los NPCs cumplen un papel de ladrones, se decidió enfatizar esta característica. Es por esto que los animales elegidos son: un mapache, comúnmente asociado con ladrones; un mosquero cardenal, cuyo plumaje forma una especie de antifaz negro; y un camaleón, radicalmente distinto a los dos anteriores y capaz de robar con la lengua.

Estos personajes se presentarán en el juego como el 'Club del antifaz negro'. Este antifaz será un elemento común en los tres animales, de forma que el espectador pueda identificarlos como ladrones y los pueda entender como un grupo.

Además de todo esto, se debe tener en cuenta el ambiente del juego a la hora de diseñar los personajes. Se busca un estilo sencillo, muy cartoon y un poco infantil, de forma que los animales encajen con el estilo general de '¡Al ladrón!'.

Habiendo los aspectos más generales de este grupo de enemigos, se va a desarrollar cada uno de ellos por separado en sus respectivos sprints del proyecto.

Música y sonidos:

La música en el videojuego es una sintonía alegre y rápida, que irá aumentando su velocidad acorde a la dificultad de la partida. Se usa una canción que se reproduce en bucle y que aporte cierta sensación de adrenalina al jugador. A parte de esto, se pueden escuchar varios efectos de sonido en distintas situaciones que pueden ocurrir a lo largo de la partida, por ejemplo, cuando al jugador le roban fruta.

2.2. Revisión del sprint 1

Tal y como se explicó en la sección 1.5, al terminar cada iteración se va a realizar una revisión del trabajo hecho en la misma. En este sprint se han definido los aspectos más relevantes del juego, plasmándolos en un GDD. Además, se ha decidido que los personajes que se desarrollarán y animarán en los siguientes sprints serán los enemigos del juego, realizando para cada uno de ellos una animación para cuando persiga al jugador, otra para cuando intente robarle fruta y una última para esconderse. Todo este trabajo se ha llevado a cabo en el tiempo estimado (11 días), por lo que no se harán cambios en la planificación de las siguientes iteraciones.

Para dar este sprint por terminado, se ha reorganizado el tablero en Trello y se han movido a la columna 'To Do' las tareas correspondientes a la siguiente iteración.

Capítulo 3

Personaje Mapache - Sprint 2

El segundo sprint del proyecto se centra en la creación y animación del primero de los enemigos de '¡Al ladrón!', el mapache.

3.1. Diseño del mapache

El primer paso a la hora de diseñar un personaje es recopilar referencias para el mismo. En este caso, el personaje está basado en un animal, por lo que se usarán fotografías del mismo. Sin embargo, no se busca un diseño realista, por lo que es importante también inspirarse con ilustraciones de otros artistas para entender cómo estilizar el personaje, buscando un estilo sencillo y simpático.

Se ha usado Pinterest para la búsqueda de imágenes, y se ha creado un tablero en la herramienta PureRef (ver Figura 3.1.1), para poder moverse fácilmente por las imágenes.

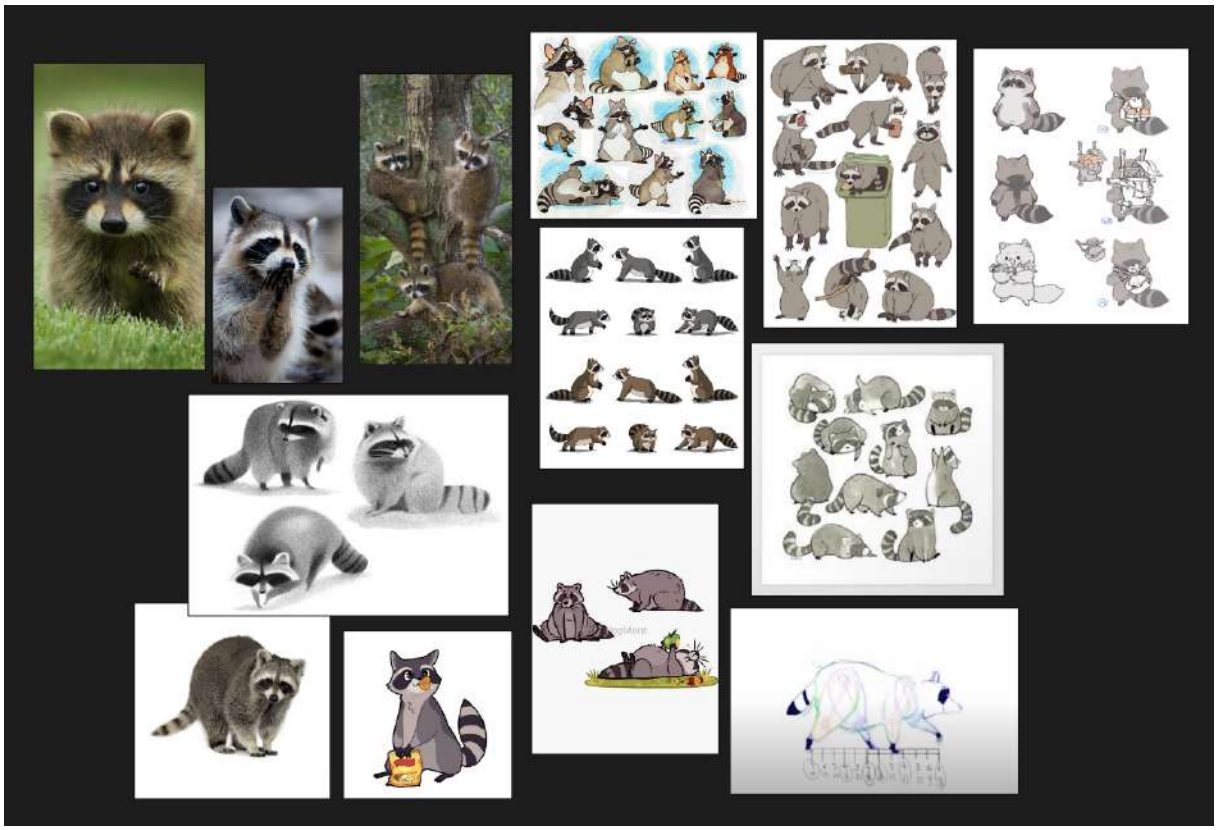


Figura 3.1.1: Referencias para el mapache

Por otra parte, se ha reflejado en la Tabla 3.1 algunas características de la personalidad del mismo. Esto ayudará a definir ciertos detalles de su diseño y, posteriormente, de su movimiento.

Especie	Mapache (Procyon)
Cualidades	Es torpe, gracioso y tontorrón. Es muy inocente, pero también bastante asustadizo, siempre sale corriendo si oye un ruido raro.

Tabla 3.1: Cualidades del mapache

Teniendo en cuenta estas referencias, se han realizado una serie de bocetos hasta llegar al aspecto final deseado. Del diseño final, se ha hecho un turn-around (ver Figura 3.1.2), de forma que sea sencillo el proceso de modelarlo en 3D.

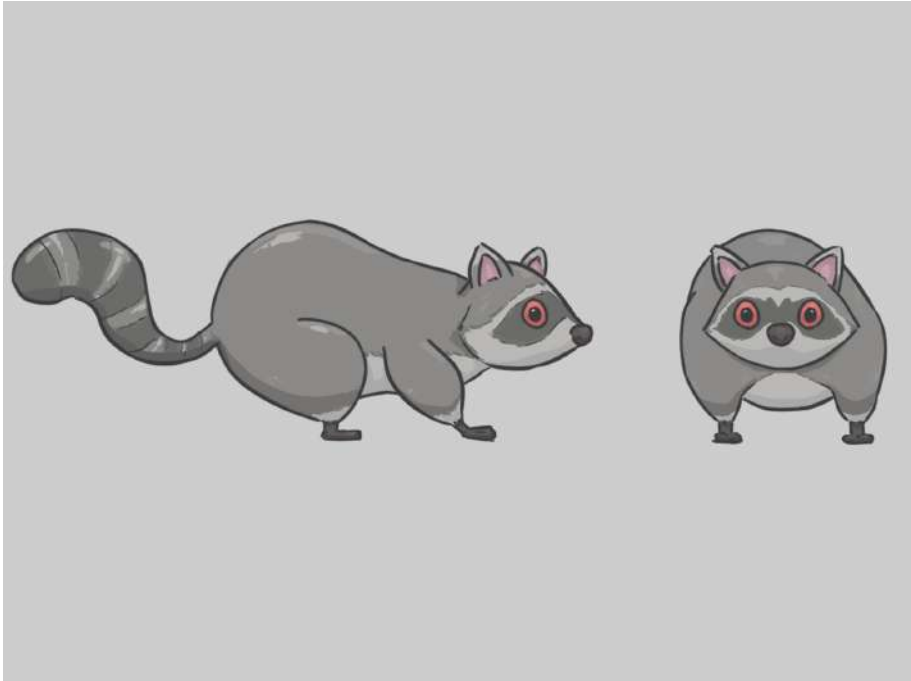


Figura 3.1.2: Turn-around del mapache

3.2. Modelado 3D del mapache

Como se explica en la sección 1.3, los dos caminos más populares a la hora de modelar un personaje en 3D son: esculpir y hacer retopología, o modelar de forma tradicional. Para este proyecto, se va a usar la segunda opción, ya que se trata de un diseño sencillo y se busca una baja poligonización, común en la industria de los videojuegos y que facilitará el trabajo a la hora de hacer el skinning de los personajes.

Para agilizar el trabajo de modelado, lo primero que se ha hecho es fijar el turn-around del personaje en las vistas ortográficas disponibles en el programa de modelado, y usarlas de referencia. En este proyecto, para el desarrollo de los modelos y animaciones se ha usado el programa Blender (versión 3.6). En la Figura 3.2.1 se puede ver un ejemplo de cómo se configurarían el entorno 3D antes de empezar a modelar.

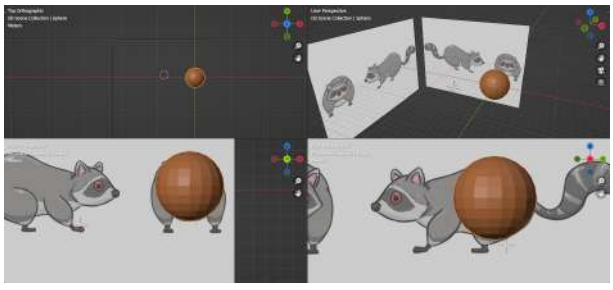


Figura 3.2.1: Configuración del entorno en Blender antes de comenzar a modelar.

A partir de este momento, se han ido modelado los personajes. Para reforzar la estética infantil, los personajes se han modelado por piezas, como se muestra en la Figura 3.2.2. Esta decisión estética ayuda a que los animales recuerden a marionetas infantiles, como

pueden ser ‘Los Lunnis’ o ‘Epi y Blas’. Además, esto facilitará el trabajo a la hora de animar.

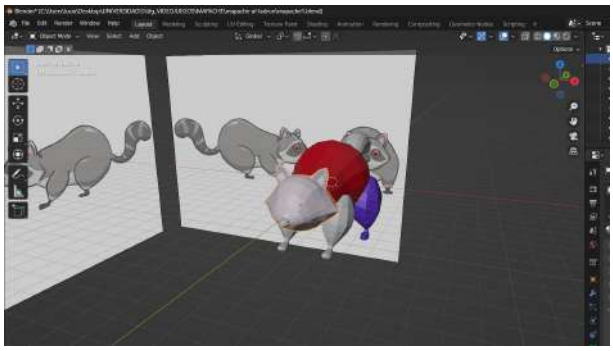


Figura 3.2.2: Modelado del mapache por piezas

De esta forma, se han desarrollado los modelos de los tres personajes. En la Figura 3.2.3 se pueden ver varios ángulos del modelo final del mapache.

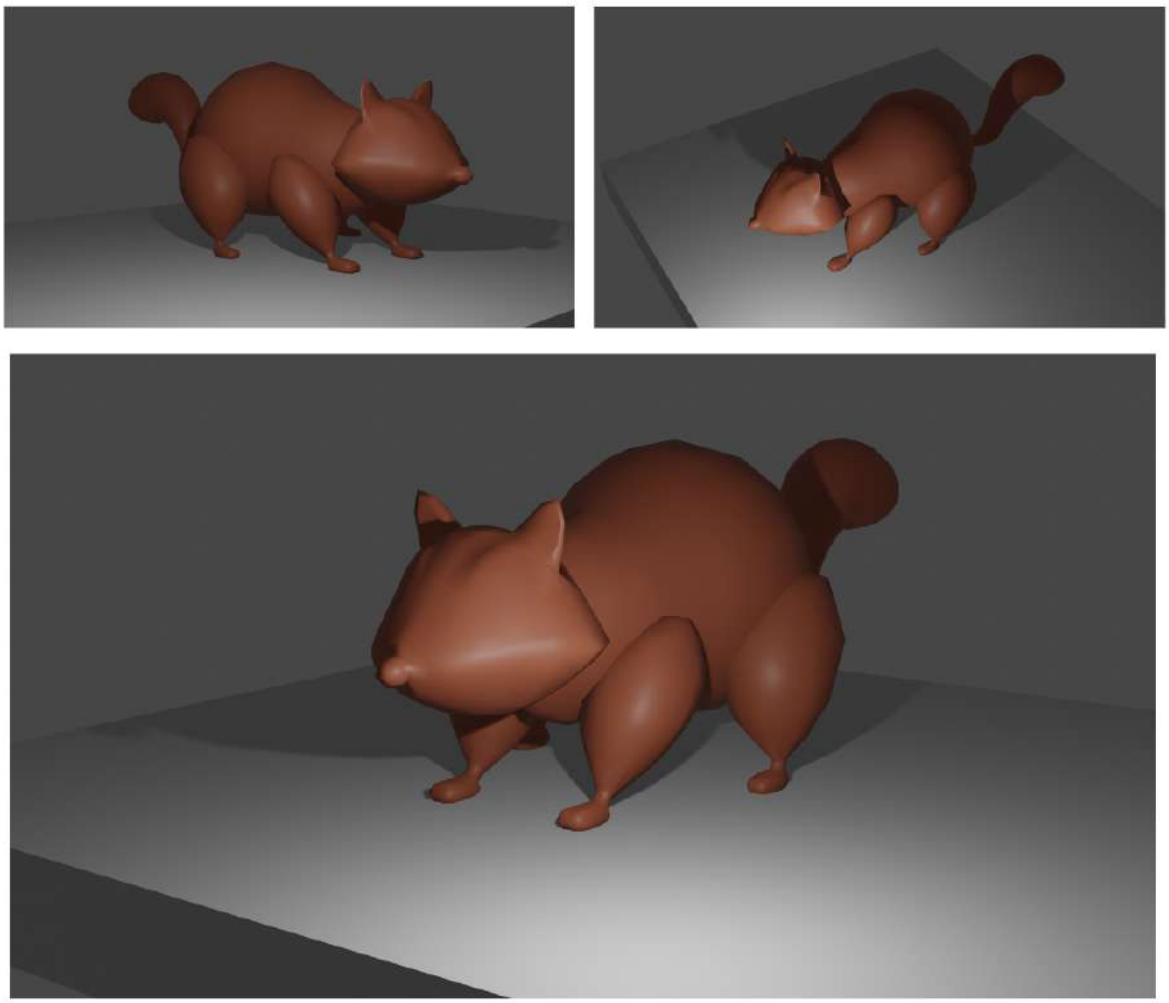


Figura 3.2.3: Modelo 3D del mapache.

3.3. Texturizado del mapache

Para texturizar el modelo, lo primero que se ha hecho es juntar todas sus piezas en una sola, ya que el diseño no tiene mucho detalle y no tendría sentido texturizar cada pieza por separado. Una vez hecho esto, se obtiene un único modelo y se ha hecho el mapeado de coordenadas en este. Como se puede ver en la Figura 3.3.1, se han colocado las distintas piezas en el mapa de UVs de forma que se aproveche al máximo el espacio de la textura. Además, se ha usado una textura checker para comprobar que no había errores.

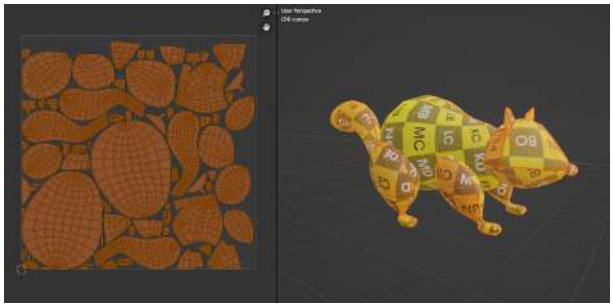


Figura 3.3.1: Mapeado de UVs del mapache.

Tras hacer el mapeado de UVs, se ha exportado el modelo como un archivo FBX, un tipo de archivo muy usado en el mundo de los gráficos 3D. Este se ha importado en el software Substance 3D Painter, donde se terminará el trabajo de texturizado.

Para este modelo, se ha decidido utilizar un material base con aspecto similar al plástico, que recuerda al de los juguetes infantiles. Se han editado las propiedades del material y usado varias capas de color para conseguir el resultado deseado, pintando a mano todos los detalles del personaje como son, los ojos, las orejas o las distintas tonalidades de pelo.

Por último, se han exportado desde Substance el mapa de color, el especular y el de normales, de forma que se puedan incluir de nuevo en Blender usando el editor de nodos de este software (ver Figura 3.3.2). Al hacer esto, el proceso de texturizado se da por terminado y se pueden empezar a realizar las tareas relacionadas con la animación del personaje. Tras este trabajo, el resultado final es el que se muestra en la Figura 3.3.3.

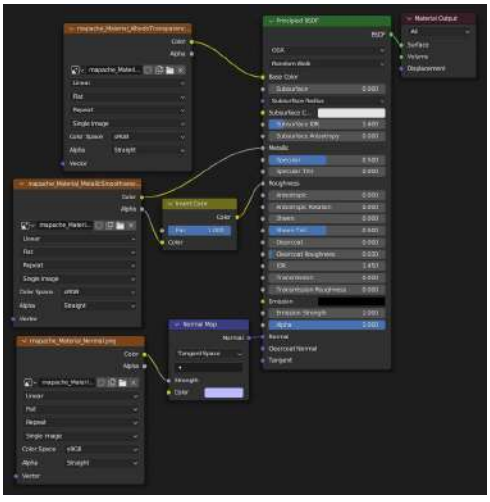


Figura 3.3.2: Editor de nodos de Blender para la inserción de las texturas del mapache.

Figura 3.3.3: Mapache texturizado en Substance 3D Painter.

3.4. Rigging y skinning de mapache

Antes de realizar las animaciones, se deben preparar los modelos para ello. Esto es lo que se hace en las fases de rigging y skinning. Como se explicó en la sección 1.3, estas etapas suelen mencionarse juntas, ya que no tienen sentido una sin la otra, pero se refieren a tareas distintas.

Rigging

El rigging consiste en crear un 'esqueleto' virtual adaptado al modelo 3D correspondiente. La mayor dificultad a la hora de hacer el rigging de este proyecto ha sido que los personajes no son humanos, por lo que se ha requerido una pequeña investigación previa de cada uno de ellos y de cómo realizan sus movimientos, de forma que se pudiese desarrollar un

esqueleto que permitiera hacer las animaciones requeridas, sin tener una cantidad excesiva de huesos.

En el caso del mapache, se han recopilado una serie de imágenes y se han usado vídeos para entender su movimiento. Algunos ejemplos se pueden ver en la Figura 3.4.1.

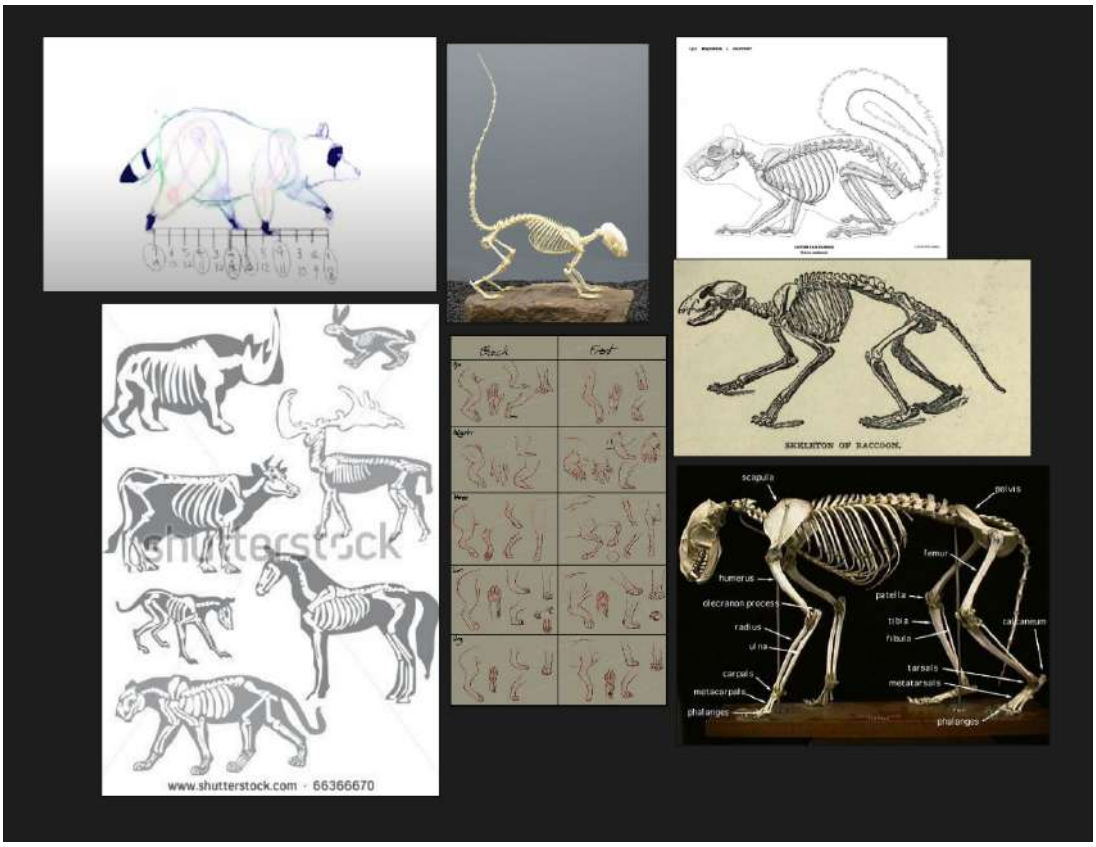


Figura 3.4.1: Referencias del esqueleto de un mapache.

Usando estas referencias, se ha desarrollado el esqueleto del mapache como se muestra en la Figura 3.4.2.

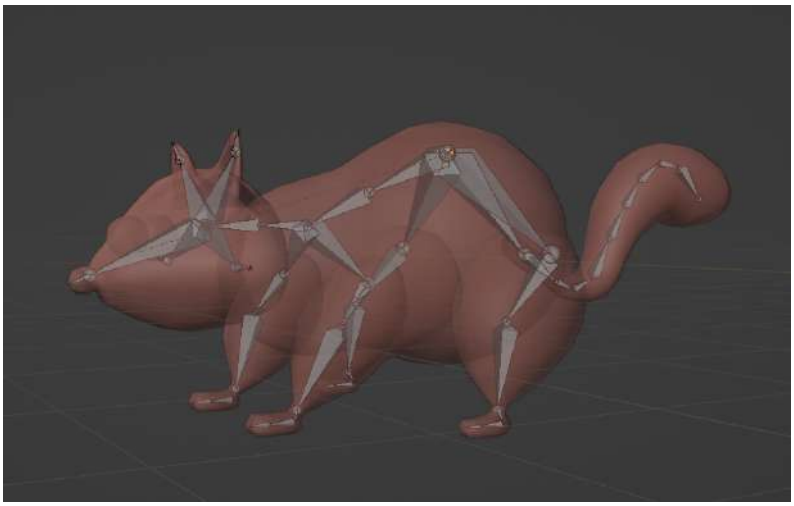


Figura 3.4.2: Rig del mapache.

Skinning

Inmediatamente después del rigging, se realiza la etapa de skinning. En esta, se asigna la influencia que tiene cada hueso del esqueleto en cada vértice de la malla. Es decir, cuánto se moverán esos vértices al mover el hueso correspondiente.

El software Blender cuenta con una herramienta de skinning automática. Sin embargo, esta no ofrece un resultado del todo correcto, por lo que es conveniente revisar y corregir el skinning proporcionado de forma automática. En la Figura 3.4.3 se puede ver el proceso de skinning del mapache. Al seleccionar un hueso del esqueleto, se representan con colores la influencia que tiene el mismo en las distintas zonas de la malla, siendo rojo la máxima (1) y azul oscuro la mínima (0). El valor de influencia que tiene cada hueso en cada vértice se ha ido corrigiendo manualmente, hasta llegar a un buen resultado.

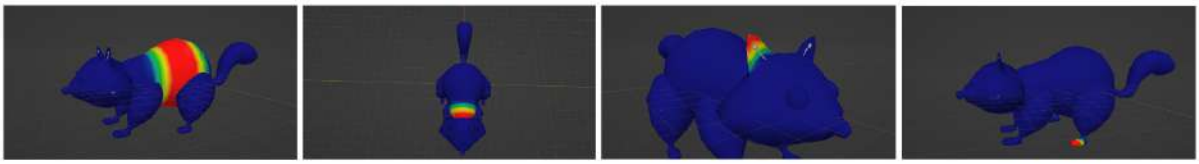


Figura 3.4.3: Skinning del mapache.

Para terminar esta fase, se han realizado algunas pruebas, en las que se han hecho movimientos exagerados con el personaje, para comprobar que el rigging y el skinning eran correctos.

3.5. Animaciones del mapache

Cada uno de los tres personajes cuenta con tres animaciones, correspondientes a las tres acciones que realizarán en el juego: perseguir al jugador, robarle fruta y esconderse cuando este se gire. Cada uno de estos animales hará estas acciones de forma única, ya que son muy distintos entre sí. En esta tarea, se explica el proceso llevado a cabo para crear las animaciones del mapache.

3.5.1. Acción de perseguir al jugador

Las animaciones de perseguir al jugador son las más sencillas de estos personajes, ya que consisten únicamente en un ciclo de caminado, el cuál se programará en el motor de juego para que avance la distancia correspondiente. La complejidad de estas se encuentra en poder reflejar la personalidad de cada animal en su forma de andar. Para conseguir esto, se han tomado todo tipo de referencias, en su mayoría, de vídeos. Estas se han usado tanto para entender el movimiento de cada animal en sí, como para comprender cómo darle personalidad al mismo.

La animación del mapache tiene una complejidad respecto a otros personajes: camina a cuatro patas. Para entender mejor el movimiento de las patas y la cadera, se han usado varios vídeos de referencia, tanto de mapaches como de otros cuadrúpedos como perros. Los vídeos que han sido más útiles para esta tarea son:

- '*Racoon Walk Cycle*' (DinoRPG, 2018).
- '*Question – Raccoon on Run*' (Kumar, 2013).
- '*quadruped locomotion*' (Art, 2020).
- '*Raccoon Walking and Climbing (Slow Motion Animation Reference)*' (Ref, 2014).

La personalidad torpe y graciosa de este personaje se ve muy reflejada en su caminar. Esto se ha conseguido con varios detalles: levantando mucho las piernas delanteras al andar, de forma que parezca que va dando pisotones; moviendo la cola y caderas de un lado a otro, de manera despreocupada; y moviendo ligeramente las orejas y los mofletes, mostrando así que está despreocupado. En la Figura 3.5.1 se muestra la animación desarrollada para esta acción.

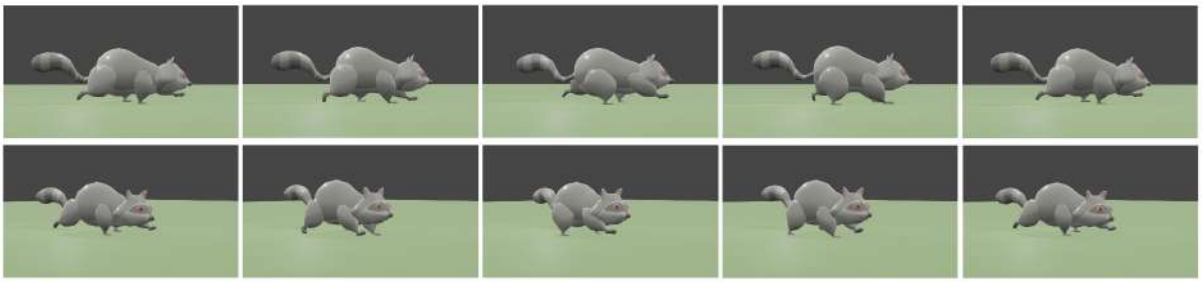


Figura 3.5.1: Animación de perseguir del mapache.

3.5.2. Acción de robar al jugador

Cuando los personajes estén lo suficientemente cerca del jugador, intentarán robarle rápidamente una pieza de fruta de su bolsa. Cada uno de ellos lo hará de forma distinta.

La animación del mapache consiste en que este se acerca corriendo al jugador, cotillea lo que hay en su bolsa, se mete en la boca uno de los frutos, y huye corriendo de ahí. El hecho de que se quede un instante mirando en la bolsa, en lugar de robar rápido y salir corriendo, denota la inocencia de este personaje. Sin embargo, cuando es consciente de que el jugador está justo ahí, huye muy asustado. Además de las referencias usadas anteriormente para entender la forma de andar y correr de un mapache, también se han usado vídeos que les muestran comiendo o cogiendo cosas. Uno de los vídeos que han sido más útiles en esta tarea es '*ASMR | Raccoon eating noises*' (Raccoon, 2020). Se puede ver el resultado de esta animación en las Figuras 3.5.2 y 3.5.2.

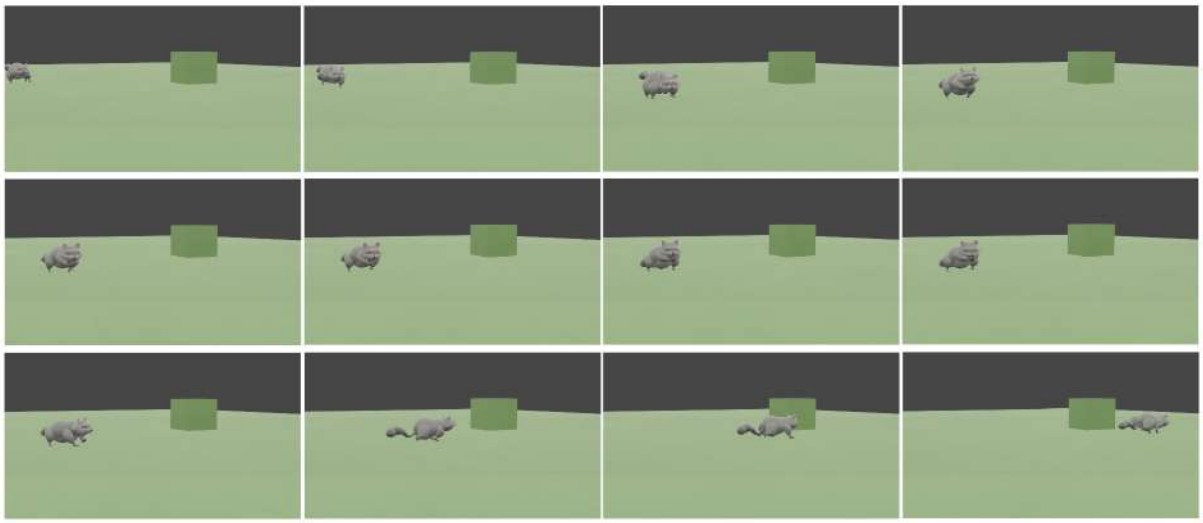


Figura 3.5.2: Animación de robar del mapache.

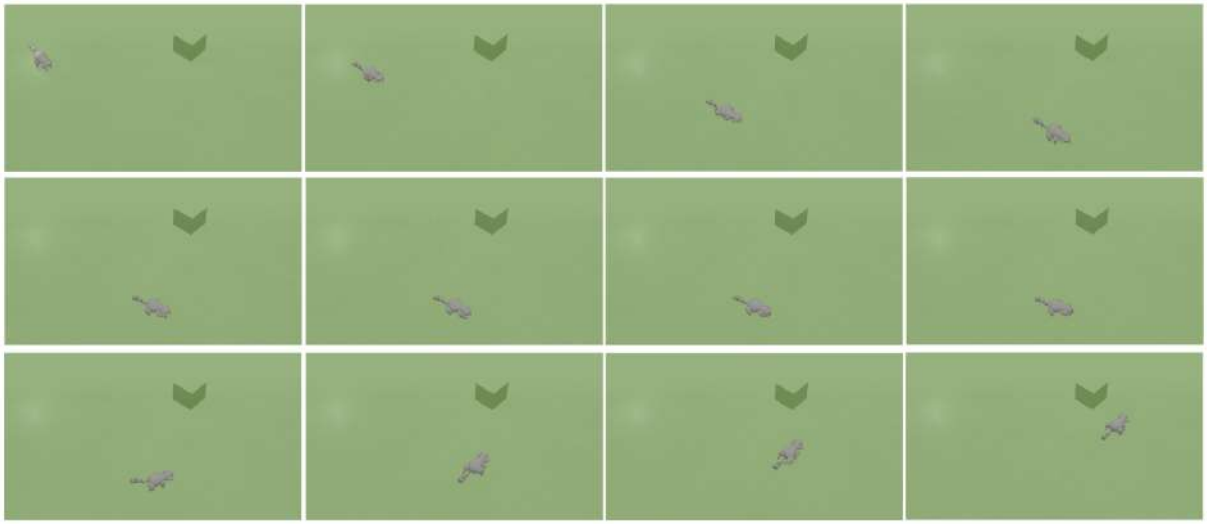


Figura 3.5.3: Animación de robar del mapache (vista superior).

3.5.3. Acción de esconderse

El último grupo de animaciones corresponde a la acción de esconderse. Esto ocurrirá cuando el jugador se gire para vigilar a los animales que le persiguen. En ese momento, el mapache, muy asustado, saldrá corriendo atropelladamente para esconderse detrás de alguna roca o arbusto que haya en el camino. Para simular este obstáculo, en la animación se ha usado un cubo, el cuál se podrá sustituir en un futuro por el elemento deseado.

Para esta animación se han usado principalmente las referencias de las animaciones anteriores y alguna más centrada en el movimiento a la hora de correr, como por ejemplo, '*How to animate an animal TROT cycle!*' (Kuzillon, 2024a). Esta última animación se muestra en las Figuras 3.5.4 y 3.5.5.

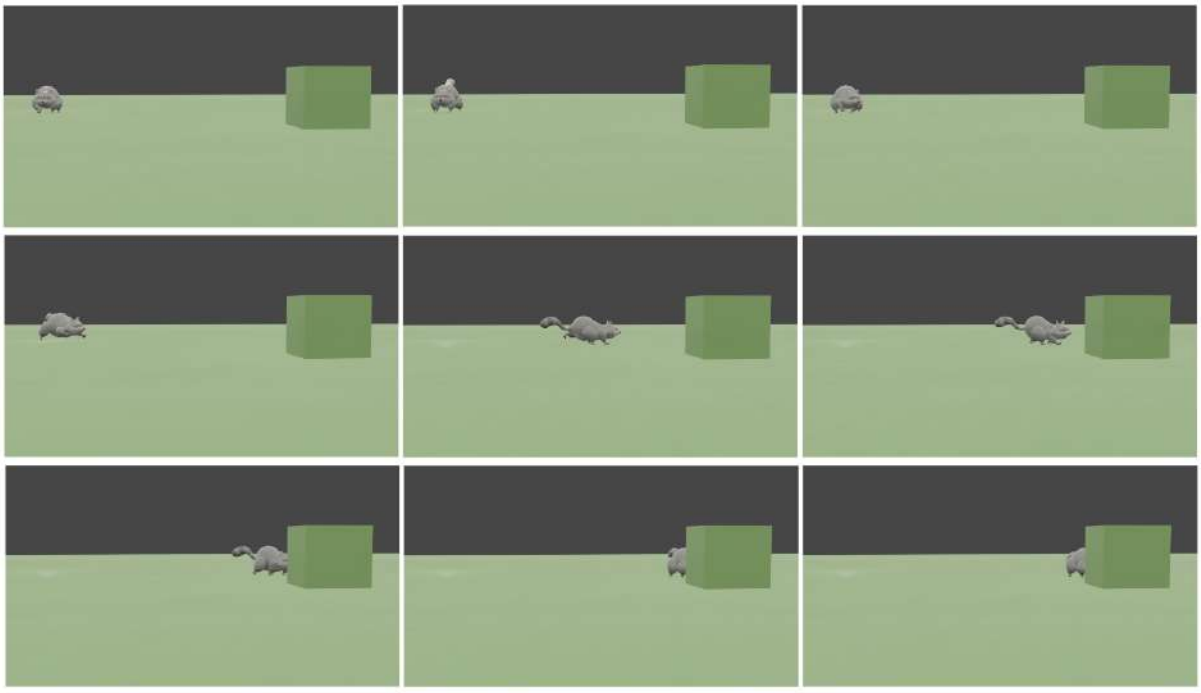


Figura 3.5.4: Animación de esconderse del mapache.

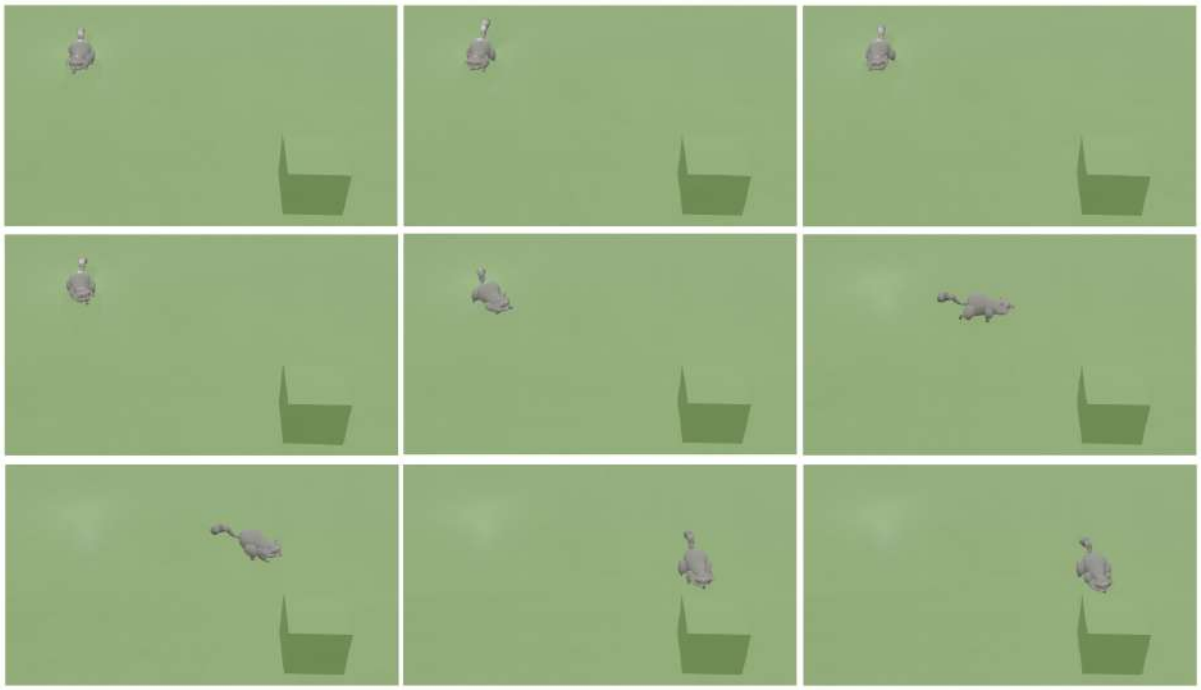


Figura 3.5.5: Animación de esconderse del mapache (vista superior).

3.6. Inclusión del mapache en el motor de juego

El trabajo a realizar con este personaje termina con su inclusión en el motor de juego. Para este proyecto se va a usar Unity [90], un motor de juego gratuito para uso personal y con el cuál ya se ha trabajado en ocasiones anteriores.

Lo primero que se ha hecho ha sido exportar el modelo con sus animaciones desde Blender [15] como un archivo FBX (un tipo de archivo estándar para elementos en 3D). Al importar este archivo en Unity, se han escogido las animaciones finales y ajustado el frame de inicio y de finalización de cada una de ellas. Además, se ha creado un material en el que se han incluido las texturas del mapache y se ha asociado este al modelo.

Tras hacer todas estas preparaciones, se ha preparado un controlador para alternar entre las animaciones del mapache. Esto se ha implementado usando un animator controller y un script . En el animator controller se ha definido el flujo de las animaciones:

- Por defecto, el personaje ejecutará la animación de perseguir en bucle.
- Si se activa el trigger con el nombre 'hide', el personaje cambiará a la acción de esconderse.
- Si se activa el trigger con el nombre 'steal', el personaje pasará a la acción de robar.
- Al terminar la animación de esconderse o de robar, volverá a la de perseguir.

Por otra parte, en el script se han asociado algunas acciones en el teclado con los triggers de animación de forma que, al pulsar 'Espacio' se activa el trigger 'hide' y al pulsar 'Shift Izq', se activa 'steal'. En la Figura 3.6.1 se muestra el animator controller y el script creado.

```
public class EnemyController : MonoBehaviour
{

    public Animator anim;

    void Start()
    {
        anim = GetComponent<Animator>();
    }

    void Update()
    {
        if(Input.GetKeyDown(KeyCode.Space)){    //hide

            anim.SetTrigger("hide");


        }else if(Input.GetKeyDown(KeyCode.LeftShift)){
 //steal
            anim.SetTrigger("steal");

        }
    }
}
```

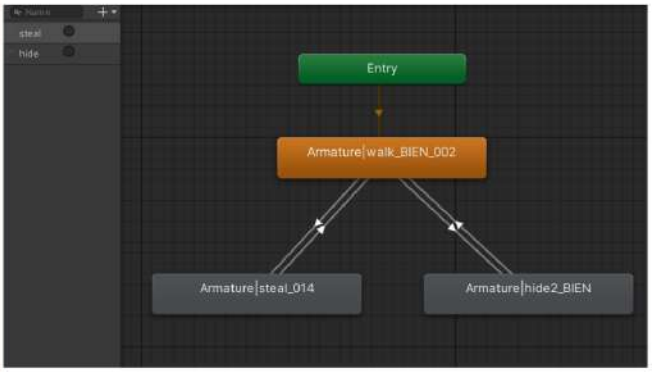


Figura 3.6.1: Script y animator controller usado para cambiar entre las distintas animaciones.

Una vez hecho esto, el mapache ya está preparado para que se pueda incluir en el juego en un futuro, programando más detalladamente su comportamiento y su movimiento por el escenario. En la Figura 3.6.2 se muestra al mapache incluido en Unity.

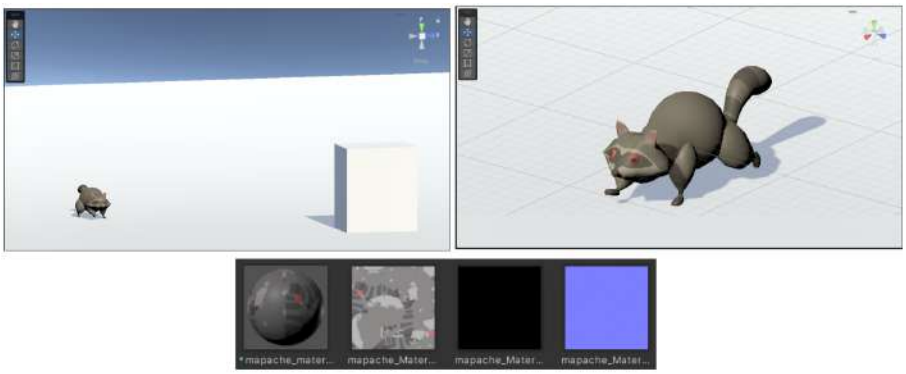


Figura 3.6.2: Mapache insertado en Unity.

3.7. Revisión del sprint 2

A lo largo de este sprint se ha desarrollado por completo el primero de los enemigos del videojuego, de forma que se ha completado un tercio de los objetivos específicos 2, 3 y 4 (ver sección 1.4. Se ha definido su diseño, modelado en 3D y realizado las tareas necesarias para su animación y su inclusión en el motor de juego. De esta forma, al final del sprint se puede presentar una escena en Unity con este primer personaje.

El resultado obtenido es satisfactorio, sin embargo, planificación de este sprint no ha sido del todo acertada. El mayor problema es que se hizo una estimación de tiempo demasiado optimista: se estimó que el trabajo se terminaría en 25 días, pero su duración real han sido 27. Para solucionar este problema en futuros sprints y estimar de la mejor forma posible la duración del proyecto, se ha modificado la duración de los dos sprints siguientes a 27 días. Este cambio se ha reflejado en Trello, donde se ha dispuesto todo para dar comienzo al siguiente sprint.

Capítulo 4

Personaje Mosquero Cardenal - Sprint 3

En este sprint se va a crear el segundo de los enemigos del juego: el mosquero cardenal. Para este propósito se han seguido los mismos pasos que en el sprint anterior.

4.1. Diseño del mosquero cardenal

La estilización de un ave es ligeramente más compleja que la de un mamífero, por lo que se han usado un mayor número de referencias de ilustraciones y de animación, que ayudasen a simplificar al animal.

De igual forma, se ha creado un tablero de PureRef (ver Figura 4.1.1), recogiendo las referencias encontradas en Pinterest.

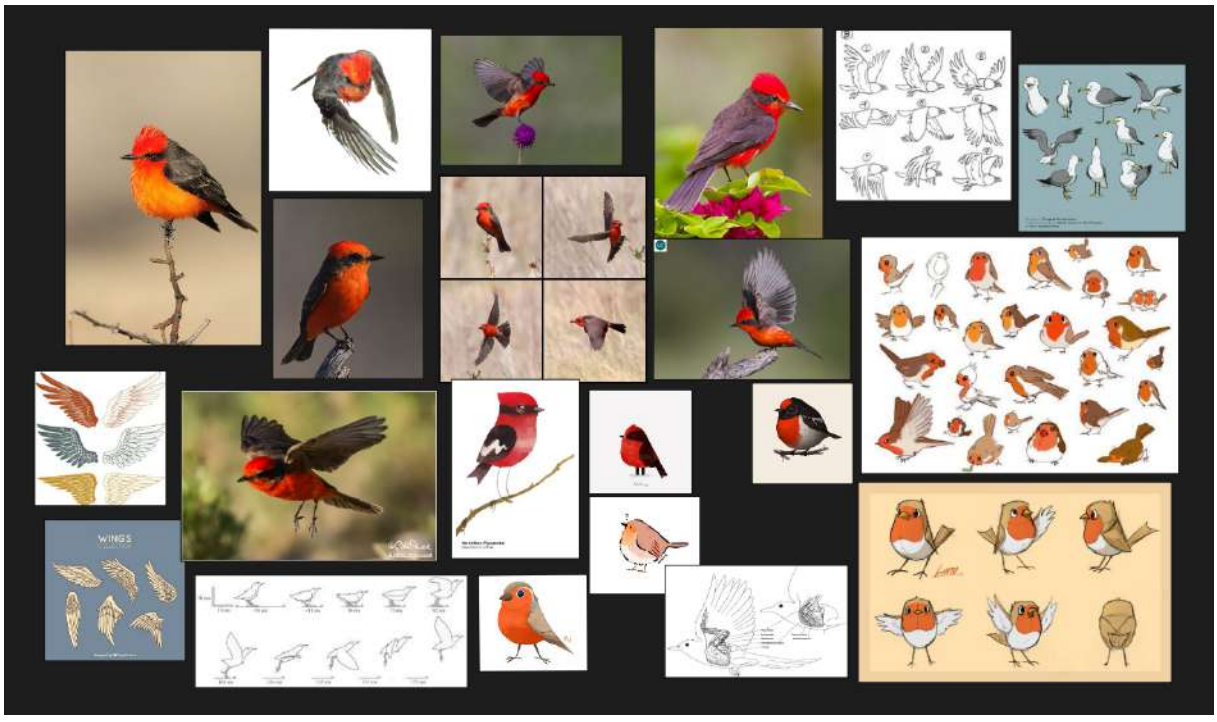


Figura 4.1.1: Referencias para el mosquero cardenal

Tras esto, se utilizó una tabla similar a la anterior para recoger algunos aspectos de la personalidad del animal (ver Tabla 4.1).

Especie	Mosquero cardenal (Pyrocephalus obscurus flammeus)
Cualidades	Es atrevido, alegre y ágil. Es el más valiente de los tres animales y le gusta demostrarlo. Es un poco orgulloso e incluso chulo en algunas ocasiones.

Tabla 4.1: Cualidades del mosquero cardenal

Una vez hecho esto, se realizaron algunos bocetos y un turn-around con el diseño definitivo. Este se puede ver en la Figura 4.1.2.

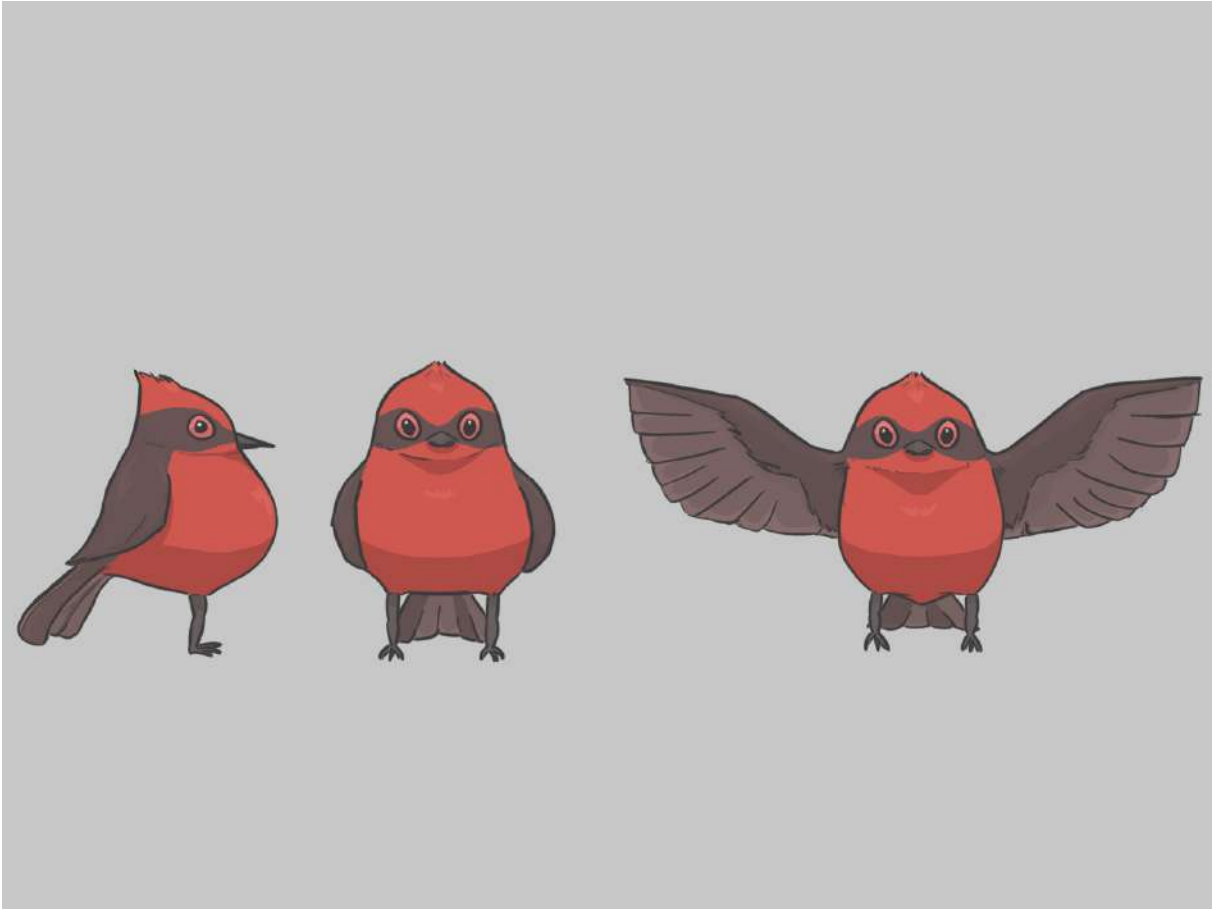


Figura 4.1.2: Turn-around del mosquero cardenal

4.2. Modelado 3D del mosquero cardenal

Antes de comenzar a modelar, de nuevo, se fijó el turn-around del personaje en las vistas ortográficas, de forma que se pueda tomar referencia.

Una vez hecho esto, se ha modelado el pájaro por piezas, separando las alas y la cabeza del resto del cuerpo para poder animarlos fácilmente , como se puede ver en la Figura 4.2.1.

Por este mismo motivo, se ha modelado el personaje con las alas abiertas y estiradas. En la Figura 4.2.2 se muestra el resultado final de esta tarea.

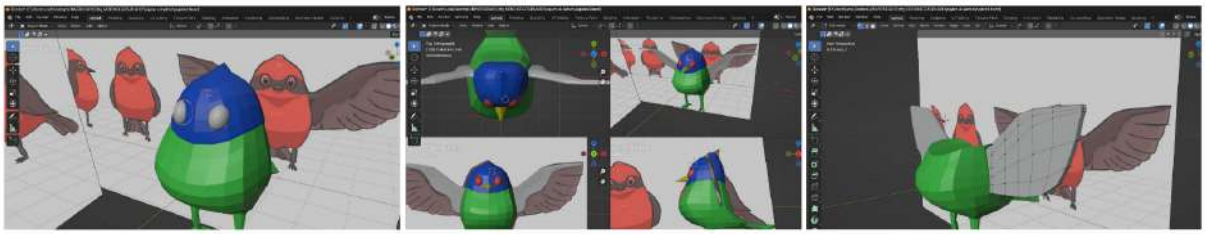


Figura 4.2.1: Proceso del modelado del mosquero cardennal.

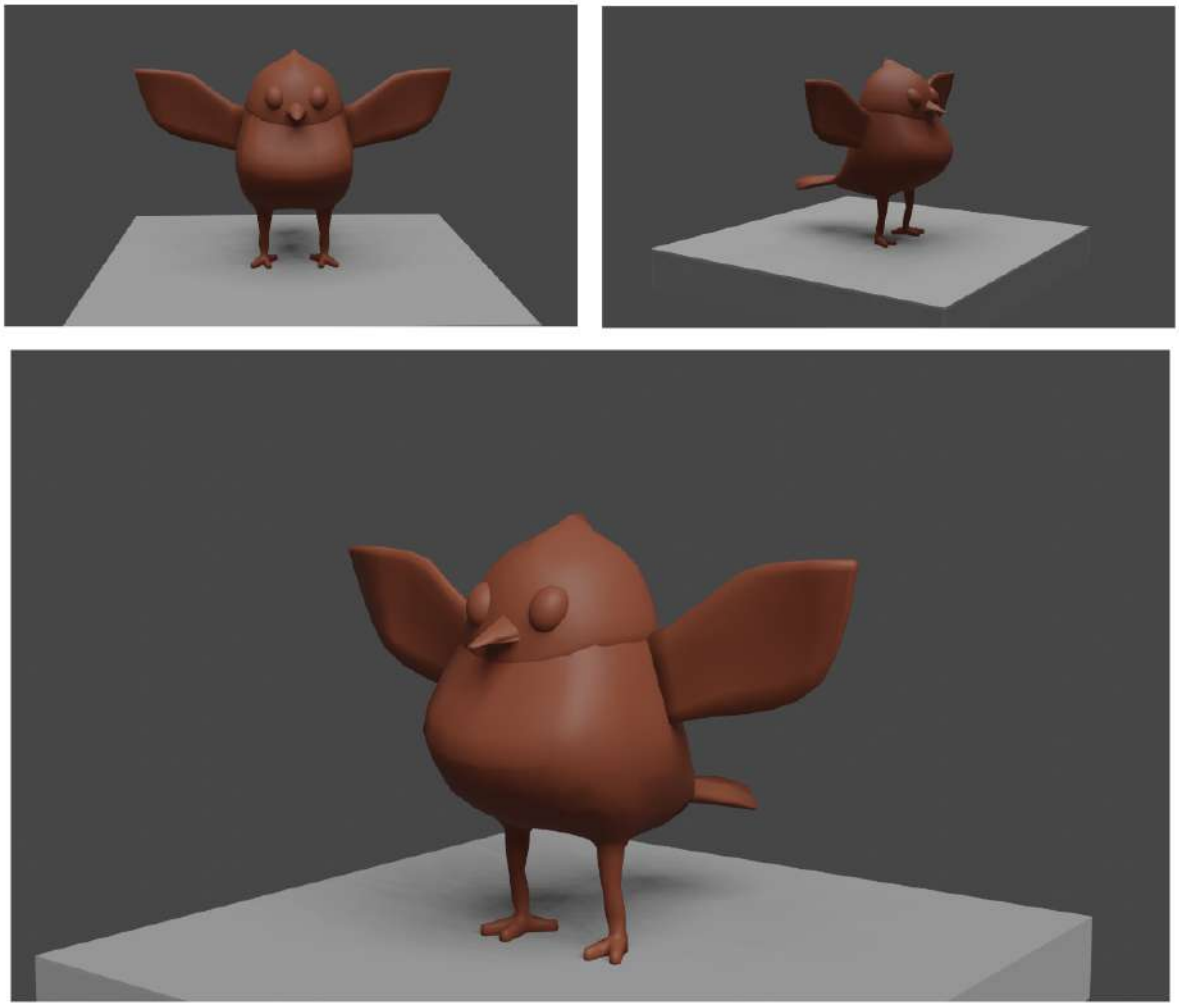


Figura 4.2.2: Modelo 3D del mosquero cardenal.

4.3. Texturizado del mosquero cardenal

La fase de texturizado comenzó juntando todas las piezas del modelo y haciendo el mapeado de UVs del mismo hasta que la textura checker se mostrase bien en todo el personaje (ver Figura 4.3.1).

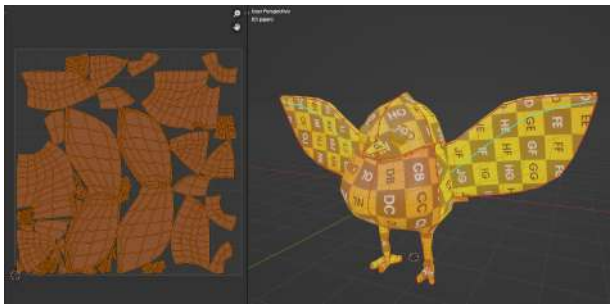


Figura 4.3.1: Mapeado de UVs del mosquero cardenal.

Tras esto, se exportó el modelo y se pasó a trabajar en el software Substance 3D Painter. En este programa, se ha aplicado el mismo material que al mapache, uno con aspecto de plástico, y se han ido añadiendo varias capas de color hasta conseguir el resultado deseado que se muestra en la Figura 4.3.2.

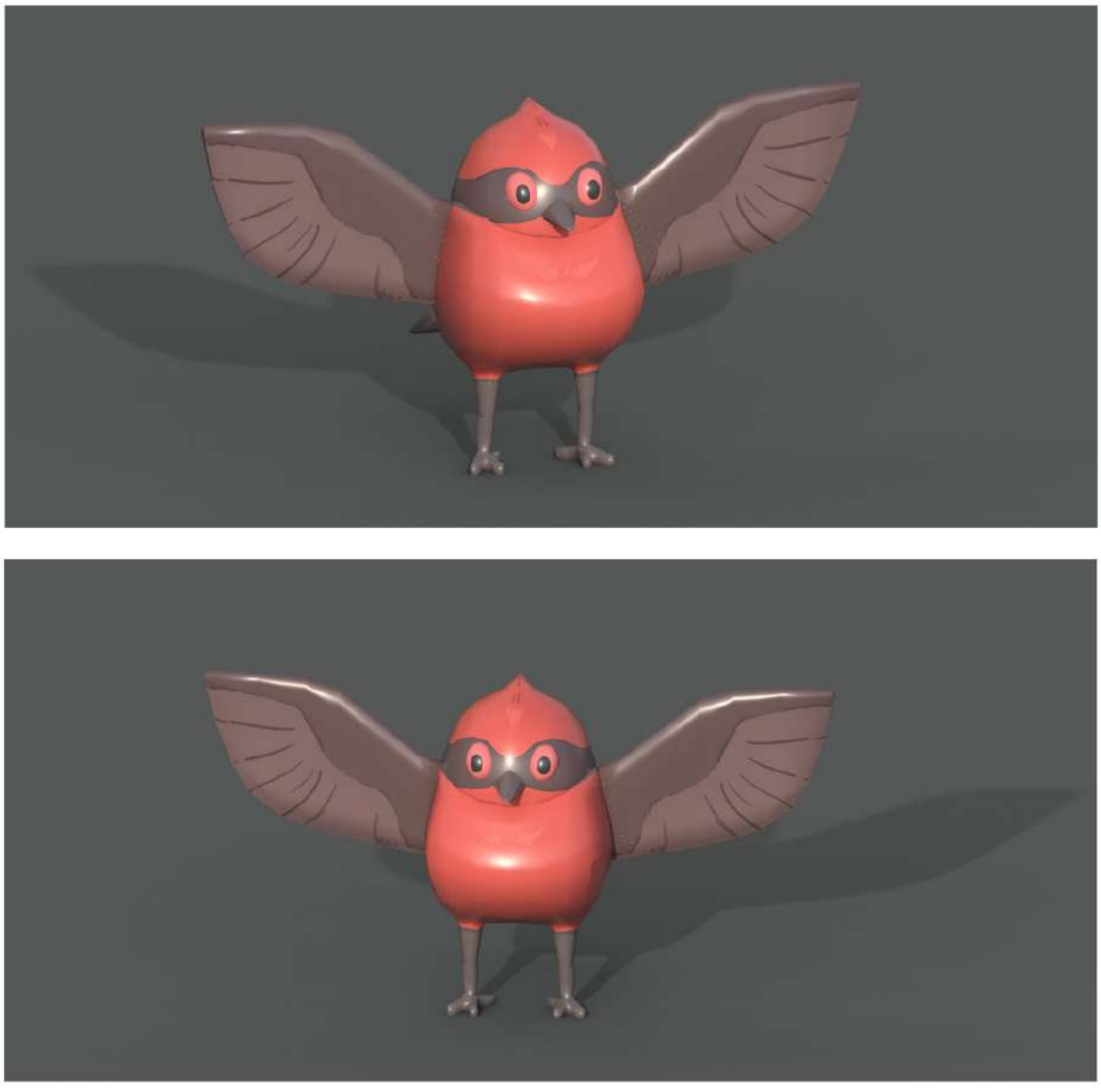


Figura 4.3.2: Mosqueo cardenal texturizado en Substance 3D Painter.

Igual que en el caso anterior, se formó un diagrama de nodos en Blender con las texturas exportadas desde Substance 3D Painter y se dió por finalizada esta tarea.

4.4. Rigging y skinning de mosquero cardenal

Estas tareas han resultado algo más complejas para este personaje que para el mapache, ya que el esqueleto de los mamíferos resulta más sencillo de simplificar. Sin embargo, el proceso que se ha seguido es muy similar.

4.4.1. Rigging

En primer lugar, se han añadido varias imágenes de referencia a una escena en Pureref (ver Figura 4.4.1).

Figura 4.4.1: Referencias del esqueleto de un mapache.

La mayor dificultad a la hora de crear el esqueleto virtual de este personaje han sido las alas, por lo que, además de las referencias anteriores, se han usado vídeos de animaciones de referencia. Algunos de los más útiles han sido:

- '*How to animate FLIGHT*' (Kuzillon, 2024b).
- '*Animation - The Mechanics of Bird Flight*' (of Aaron Blaise, 2016).
- '*Bird Flying 2D animation*' (Singh, 2021).

Gracias a todas estas referencias se ha desarrollado el esqueleto como se muestra en la Figura 4.4.2, de forma que se puedan mover las distintas partes de las alas, sin tener huesos innecesarios.

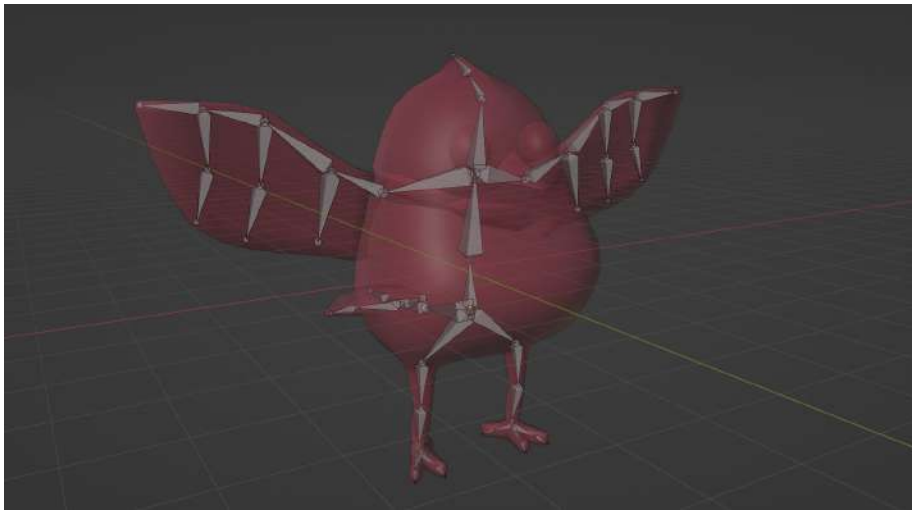


Figura 4.4.2: Rig del mosquero cardenal.

4.4.2. Skinning

La fase de skinning se ha llevado a cabo igual que en el mapache: se ha obtenido una primera versión de forma automática y se ha editado esta hasta conseguir la versión final que se muestra en la Figura 4.4.3.

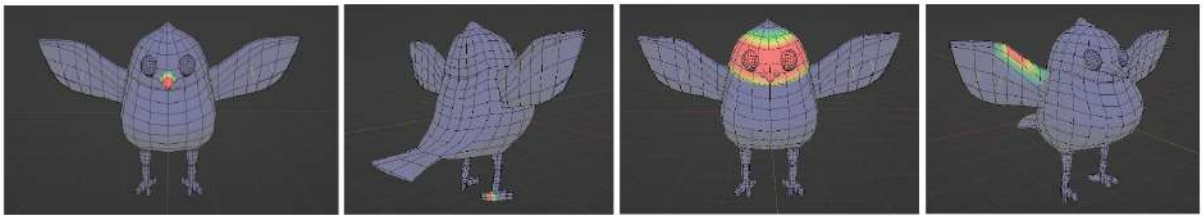


Figura 4.4.3: Skinning del mosquero cardenal.

4.5. Animaciones del mosquero cardenal

Por último en este sprint se van a realizar las animaciones básicas de los enemigos de '¡Al ladrón!'.

4.5.1. Acción de perseguir al jugador

La animación de caminar del mosquero cardenal presenta una dificultad: la estructura y movimiento sus patas, radicalmente distintas a las de los humanos y mayoría de mamíferos. Para entender su funcionamiento se usaron referencias de todo tipo de aves, desde gallinas hasta pájaros más pequeños. Los vídeos que fueron más útiles para esta tarea son:

- '*Chicken Walk Cycle: A breakdown*' (Gómez, 2021).
- '*Birds Walking - video reference for animators*' (Animation, 2019c).

Una vez animado el movimiento principal, se han ido añadiendo acciones secundarias para dar mayor interés a la animación y mostrar el carácter ligeramente chulesco del animal, demostrando la confianza que tiene en si mismo. Se ha animado un pequeño balanceo en la cola y la cabeza, además de hacer que ande sacando pecho y mirando ligeramente hacia arriba, como si mirase por encima del hombro. Por último, un detalle clave es que sus alas se han colocado de forma que caigan a los laterales del animal como si las llevase metidas en los bolsillos del pantalón con un gesto despreocupado.

En la Figura 4.5.1 se muestran varias imágenes de esta animación.

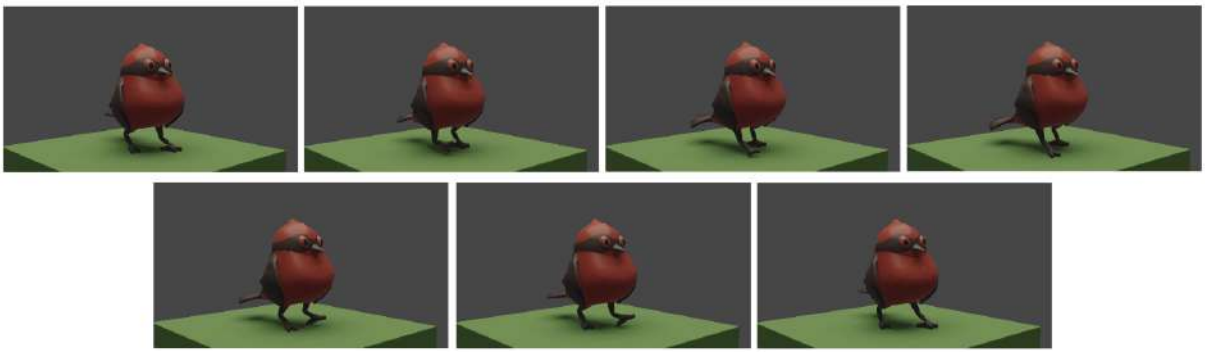


Figura 4.5.1: Animación de perseguir del mosquero cardenal.

4.5.2. Acción de robar al jugador

La forma de robar de mosquero cardenal es muy distinta a la del mapache. Este se acerca volando al jugador, agarra la fruta con sus patas y huye. Sus movimientos son mucho más fluidos y decididos que los de su compañero, mostrando su valentía y agilidad. Esta animación ha requerido un gran trabajo de investigación y un análisis profundo del vuelo del animal. Más concretamente, la mayor complejidad se ha encontrado en el momento de alzar el vuelo y de coger la fruta sin dejar de volar. Para imitar bien esto, se han usado referencias de varias aves distintas, prestando especial atención al movimiento de pájaros pequeños. Además de las mencionadas anteriormente, se han usado varios vídeos de pájaros reales:

- '*Birds Take Flight - video reference for animators*' (Animation, 2019b).
- '*Birds In Flight - video reference for animators*' (Animation, 2019a).
- '*bird reference*' (references, 2021).
- '*Hummingbirds in Slow-Motion | High-Speed Wings | Wild to Know*' (Mater, 2020b).

En las Figuras 4.5.2 y 4.5.3 se muestran secuencias de la animación final desarrollada para esta acción desde distintos puntos de vista.

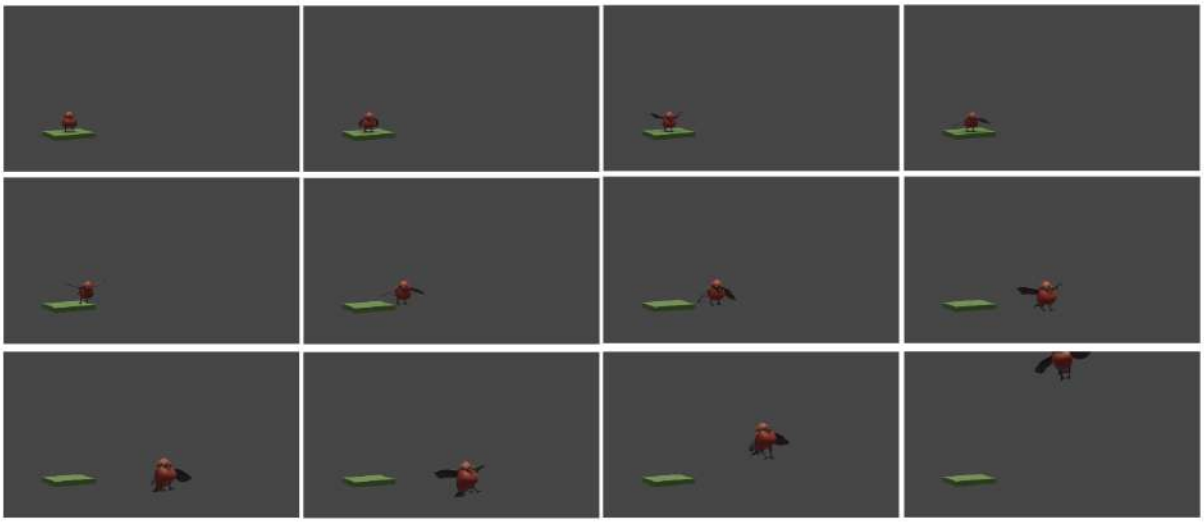


Figura 4.5.2: Animación de robar del mosquero cardenal.

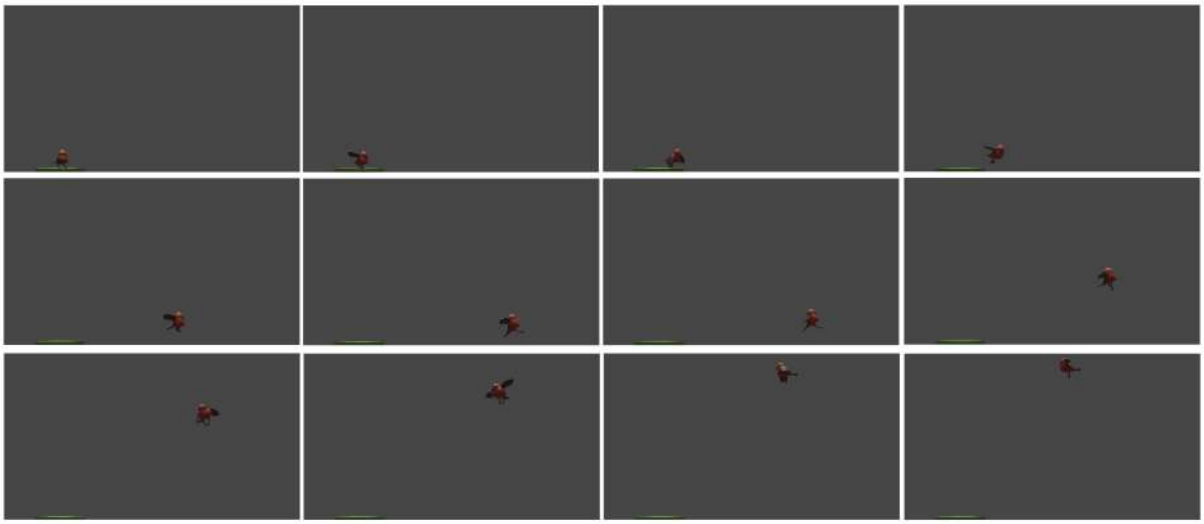


Figura 4.5.3: Animación de robar del mosquero cardenal.

4.5.3. Acción de esconderse

Cuando el jugador se gire a espantar a los animales, el mosquero cardenal, muy tranquilo y seguro de sí mismo, saldrá volando procurando que le jugador no llegue a verle. La forma de volar es ligeramente distinta a la de la animación de robar, ya que en este caso vuela de una forma mucho más vertical, para salir cuanto antes del campo de visión del jugador. El resultado final de esta animación se puede ver en las Figuras 4.5.4 y 4.5.5.

Figura 4.5.4: Animación de esconderse del mosquero cardenal.

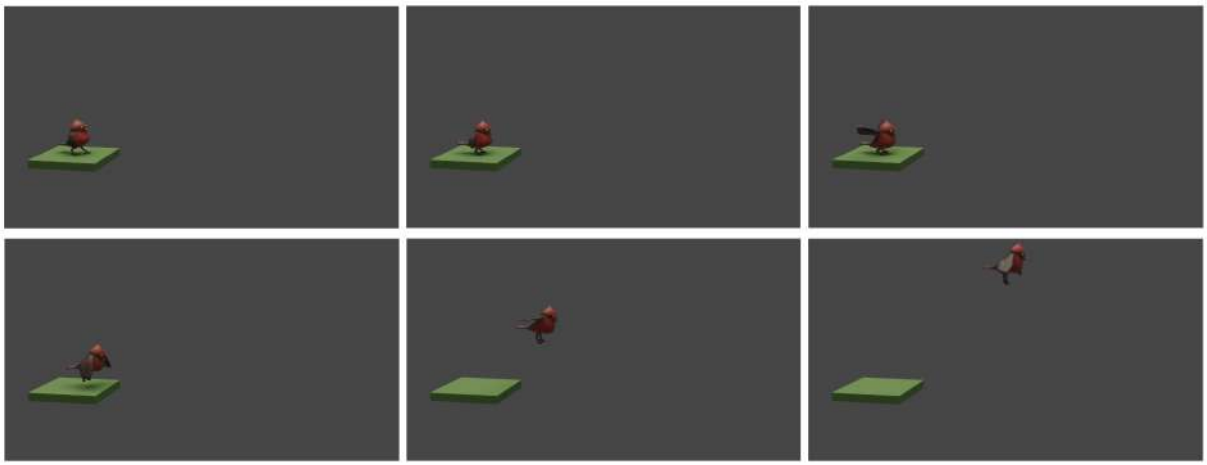


Figura 4.5.5: Animación de esconderse del mosquero cardenal.

4.6. Inclusión del mosquero cardenal en el motor de juego

Por último, se ha incluido al mosquero cardenal y a sus animaciones en Unity. Estas se han exportado desde Blender e incluido en el motor de juego. Una vez aquí se le han aplicado las texturas y ajustado las animaciones para que funcionen correctamente. Esto se muestra en la Figura 4.6.1.

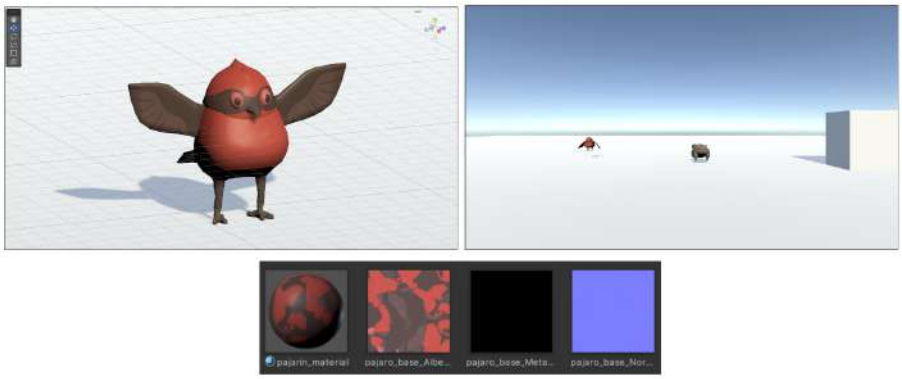


Figura 4.6.1: Mosquero cardenal insertado en Unity

Después de estas preparaciones, se han utilizado las mismas herramientas que con el personaje del mapache: un animation controller y un script para controlar el funcionamiento de las animaciones (ver Figura 3.6.1). El animation controller que se va a utilizar funciona exactamente igual que el del mapache, pero sustituyendo las animaciones de este por las del mosquero. Por otra parte, el script que se ha asociado a este es el mismo que el del mapache, ya que se reutilizará este en los tres personajes.

4.7. Revisión del sprint 3

Antes de dar por terminado este sprint, se ha hecho una revisión de todo el trabajo realizado. Como se había planificado, en esta iteración se han llevado a cabo las tareas relacionadas con la creación del segundo enemigo de '¡Al ladrón!', un mosquero cardenal, contribuyendo a los objetivos 2, 3 y 4 del proyecto (ver sección 1.4. La mayor dificultad que se ha encontrado en estas ha sido el hecho de modelar, hacer el rigging y animar las alas del animal, pero se ha podido conseguir buenos resultados gracias al uso de muchas referencias.

En cuanto la duración de este sprint, se ha cumplido con lo estimado en el anterior sprint, por lo que no se hará ningún cambio en la estimación de la próxima iteración. Para finalizar, se han colocado las tarjetas del siguiente sprint en la columna de 'To Do' del tablero Kanban y se ha dado por finalizada la tercera iteración.

Capítulo 5

Personaje Camaleón - Sprint 4

El cuarto sprint del proyecto se centra en el diseño y desarrollo del tercer enemigo de '¡Al ladrón!': el camaleón, el último integrante del 'Club del antifaz negro'. Una vez se haya terminado esta iteración, se habrán completado todos los objetivos del proyecto y se podrá dar por finalizado el mismo.

El proceso que se va a seguir para la creación de este personaje es el mismo que el llevado acabo en los dos sprints anteriores.

5.1. Diseño del camaleón

Al igual que ocurre con el ave, estilizar un camaleón puede resultar complejo. Por esto mismo, se han utilizado muchas referencias de ilustraciones y animación, incluso de otros anfibios. Las principales imágenes utilizadas se pueden ver en el siguiente tablero de PureRef (ver Figura 5.1.1).

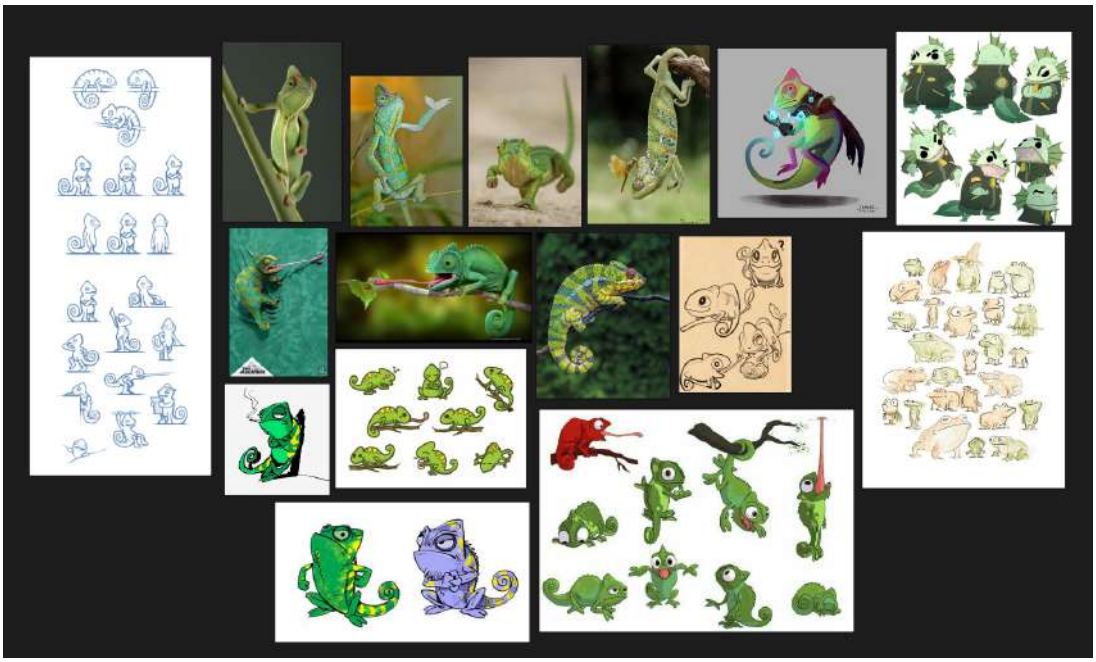


Figura 5.1.1: Referencias para el camaleón

Una vez más, con estas referencias en mente se han definido algunas cualidades del personaje (ver Tabla 5.1) y se han realizado una serie de bocetos, hasta llegar al diseño final, que se muestra en la Figura 5.1.2.

Especie	Camaleón (Chamaleonidae)
Cualidades	Es infantil y muy amigable. Esto le hizo conocer a sus compañeros y pasar a formar parte del 'Club del antifaz negro', aún sin tener uno de forma natural. Es muy miedoso y cauteloso, se camufla constantemente.

Tabla 5.1: Cualidades del camaleón

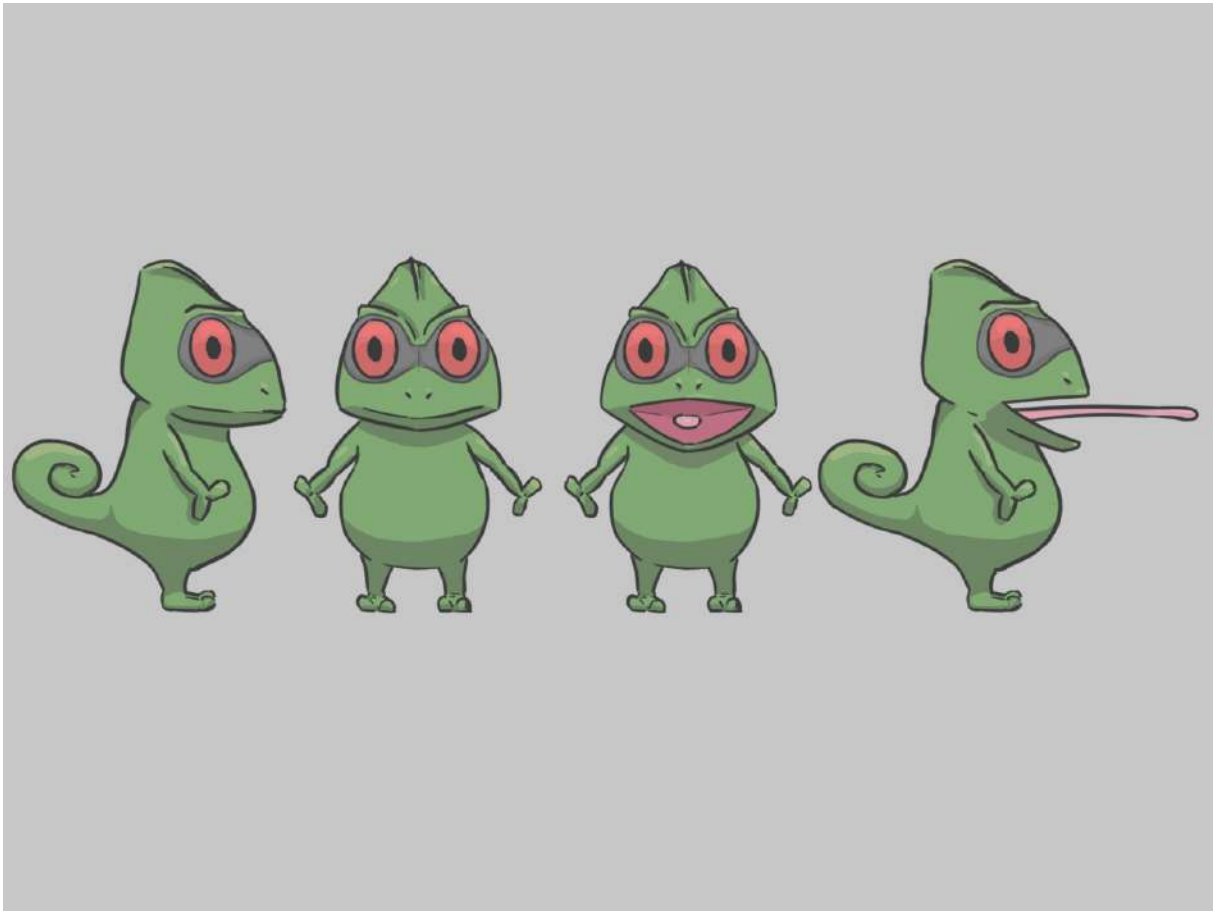


Figura 5.1.2: Turn-around del camaleón

Como se puede observar, se ha decidido hacer a este personaje bípedo. Esta decisión se ha tomado con objetivo: que el camaleón se moviese de forma muy distinta a los dos enemigos anteriores. Dado que la personalidad de este es algo similar a la del mapache, se ha decidido aumentar las diferencias entre ellos haciendo que uno sea cuadrúpedo y el otro bípedo. Además, la personalidad del mosquero cardenal y la de este personaje son radicalmente distintas, por lo que su forma de andar no será similar.

5.2. Modelado 3D del camaleón

Como se hizo en ocasiones anteriores, se preparó el entorno 3D en Blender para poder usar el turn-around del personaje como referencia en el modelado.

Gracias a estas referencias se han ido modelando las distintas partes del camaleón hasta obtener un buen resultado. Para que la malla no tenga problemas a la hora de animar al personaje, este se ha modelado en 'pose A' y con la boca abierta y la lengua fuera. Además, igual que en los anteriores, se han modelado algunas piezas del personaje por separado. Algunas imágenes de este proceso se muestra en la Figura 5.2.1.

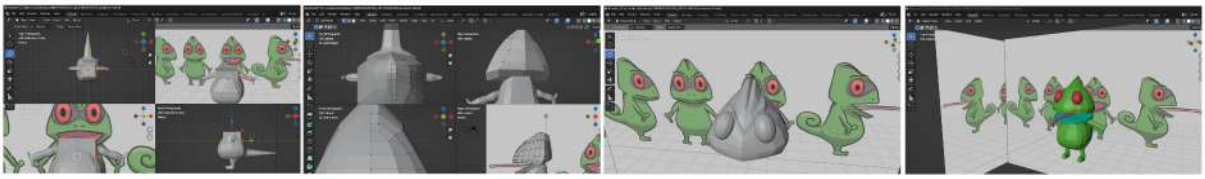


Figura 5.2.1: Proceso del modelado del camaleón.

En la Figura 5.2.2 se muestra el resultado final.

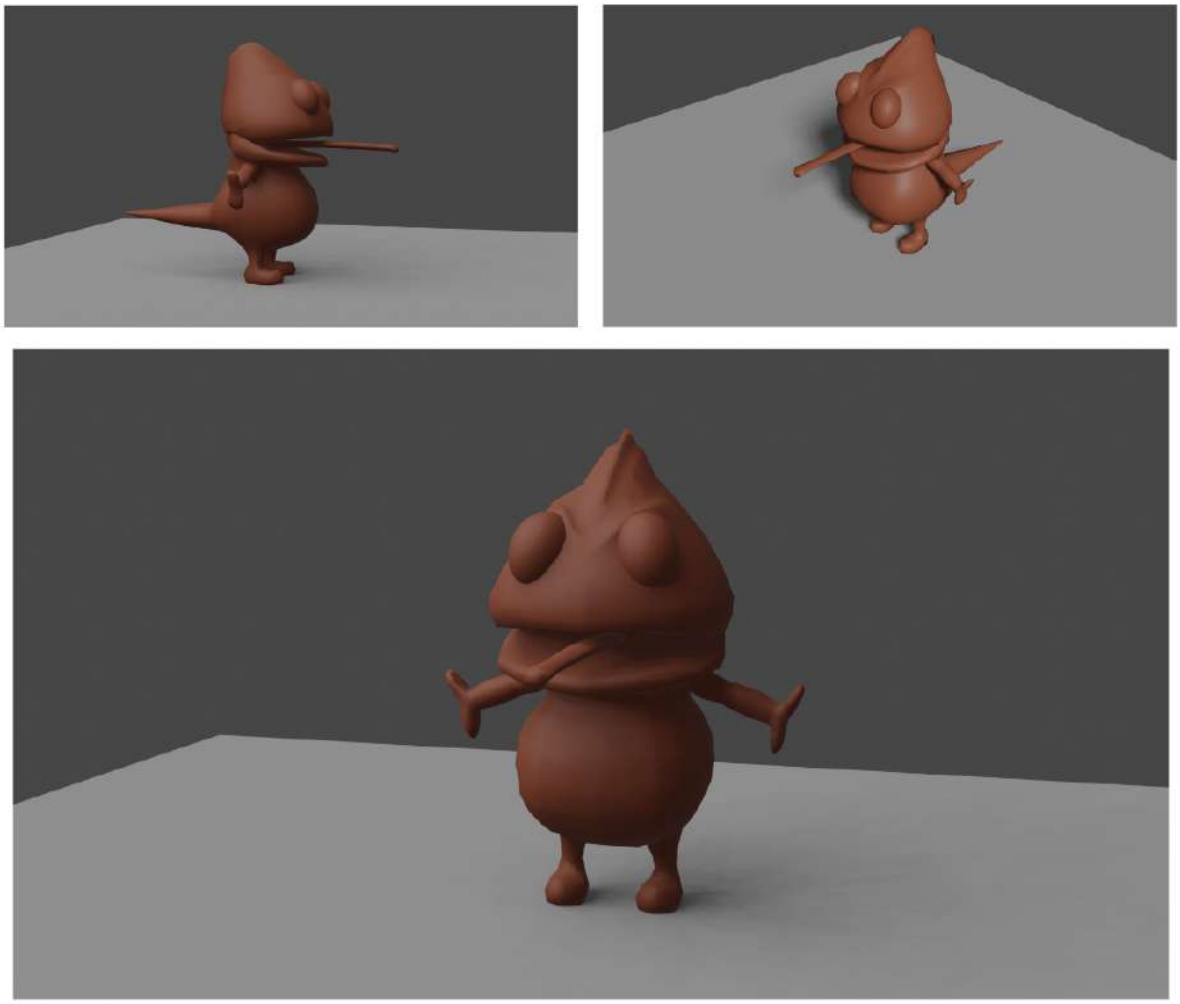


Figura 5.2.2: Modelo 3D del camaleón.

5.3. Texturizado del camaleón

Antes de comenzar a aplicar las texturas, se juntaron todas las piezas del modelo y se sacaron los mapas de UVs del mismo, usando una textura checker para comprobar el resultado del mapeado (ver Figura 5.3.1).

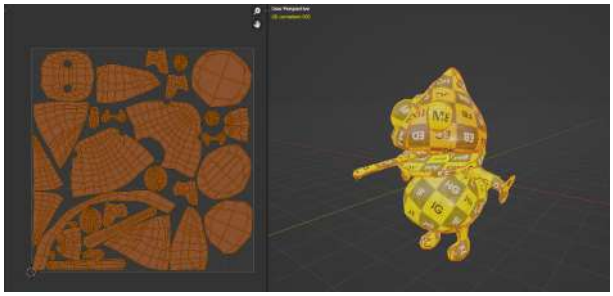


Figura 5.3.1: Mapeado de UVs del camaleón.

Ya con los mapas hechos, se exportó el modelo de Blender y se comenzaron a aplicar las texturas en Substance 3D Painter. Una vez más, se ha usado un material parecido al plástico y se han ido pintando manualmente las distintas capas para conseguir el aspecto deseado para el camaleón que se muestra en la Figura 5.3.2.

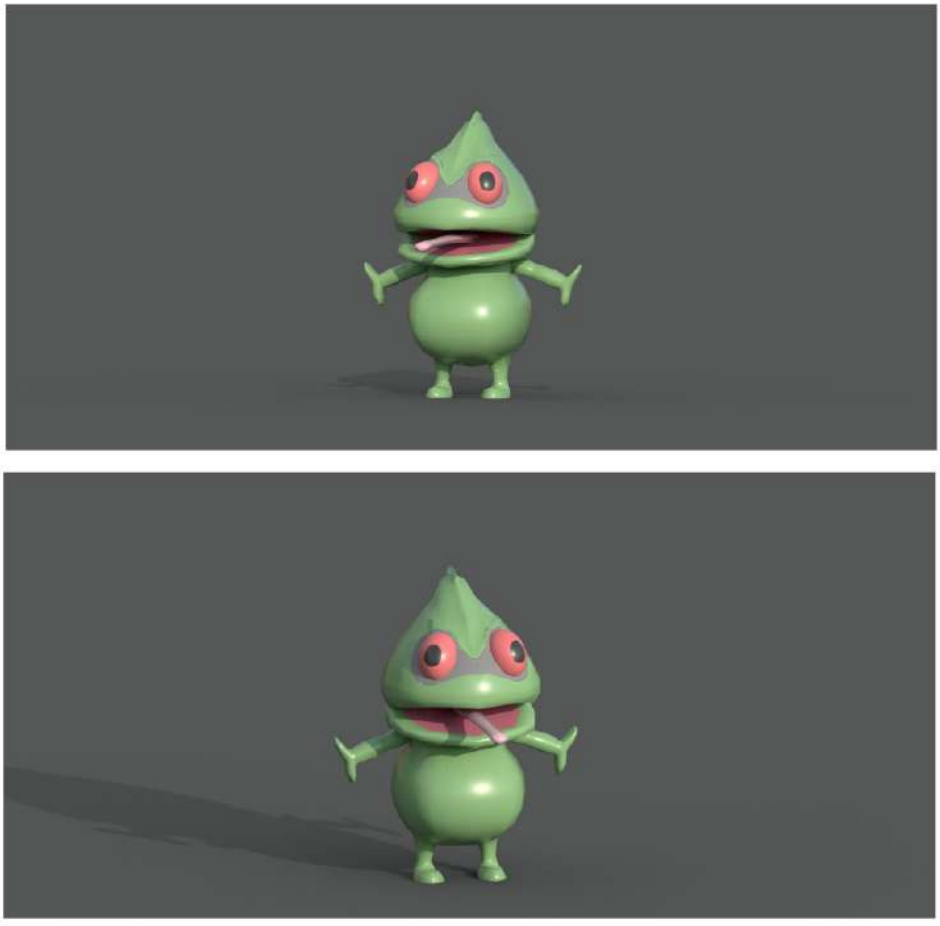


Figura 5.3.2: Camaleón texturizado en Substance 3D Painter.

5.4. Rigging y skinning de camaleón

El proceso de rigging y skinning del camaleón ha sido el más sencillo de entre los tres personajes, ya que la estructura de su esqueleto es la más similar a una persona humana.

5.4.1. Rigging

Para el rigging de este personaje no se han necesitado referencias, ya que se ha usado de punto de partida un esqueleto humano. Este esqueleto se ha modificado para que la longitud de las extremidades coincidiese con las del camaleón. Además, se han creado algunos huesos para el movimiento de la mandíbula, la lengua o la cola, entre otros, y se han eliminado varios huesos de las manos. El esqueleto final se muestra en la Figura 5.4.1.

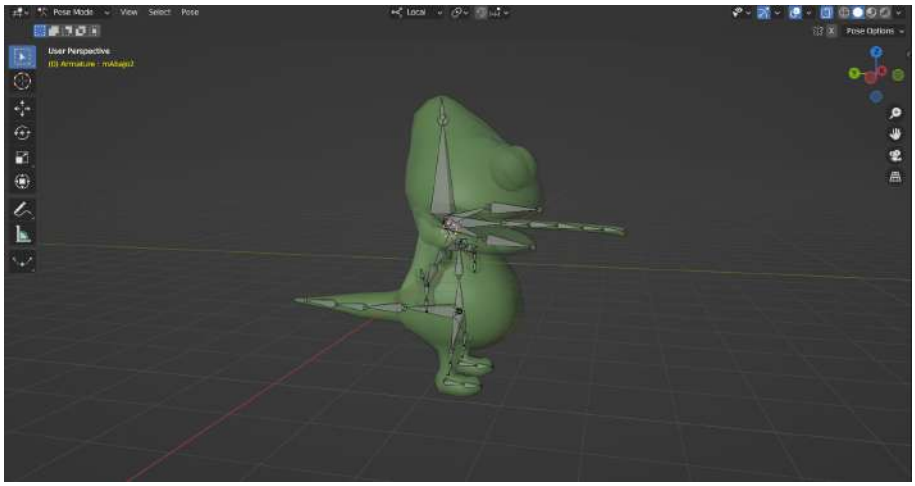


Figura 5.4.1: Rig del camaleón.

5.4.2. Skinning

Siguiendo el mismo proceso que se utilizó en los personajes anteriores, se ha usado la herramienta de skinning automático de Blender, y se ha modificado este hasta que el personaje se moviese correctamente. Se pueden ver algunas imágenes del resultado de esta tarea en la Figura 5.4.2.

Figura 5.4.2: Skinning del camaleón.

5.5. Animaciones del camaleón

Al finalizar esta tarea se habrán terminado las animaciones de los tres personajes y se habrá completado el tercer objetivo del proyecto:

Obj 3.
Implementar las animaciones básicas de los tres personajes.

5.5.1. Acción de perseguir al jugador

El camaleón es un personaje miedoso y cauteloso. Este camina a dos patas de forma muy similar a una persona, por lo que la mayor dificultad era mostrar su personalidad. Dado que es muy temeroso, en lugar de andar tranquilo, este personaje anda a hurtadillas y en tensión, procurando no hacer mucho ruido. Es decir, da pasos pequeños, con la cadera baja y despacio. Además, usa las manos para mantener firme el equilibrio y evitar hacer ningún sonido. Todas las referencias usadas para esta animación han sido vídeos personas o personajes humanos:

- '*16 WALK CYCLES - DIFFERENT ATTITUDES*' (to Be an Animator Animation Tutorials, 2021).
- '*Sneak Cycle Reference*' (Patterson, 2018).
- '*Sneaky - 3D sneak walk animation*' (annimates, 2023).

Para dar más interés a esta animación y enfatizar que el personaje está tenso, la cola de este se mueve ligeramente, pero siempre se encuentra estirada y pegada a su cuerpo. Se puede ver una secuencia de esta animación en la Figura 5.5.1.

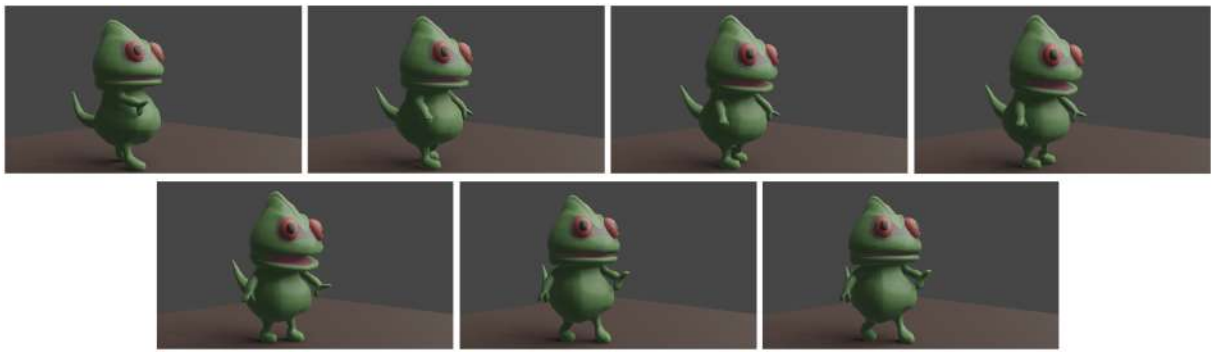


Figura 5.5.1: Animación de perseguir del camaleón.

5.5.2. Acción de robar al jugador

La animación del camaleón es la más infantil y cómica de las tres, ya que roba la fruta sacando su lengua. Este movimiento es muy típico de estos animales y se han usado un gran número de referencias de vídeo para hacerlo. Las dos que más han ayudado a entender el movimiento para poder estilizarlo son las siguientes:

- '*Chameleon Tongue In Slow Motion | One Life | BBC Earth*' (Earth, 2013).
- '*Fast Tongues l Chameleons in Slow-motion*' (Mater, 2020a).

La mayor dificultad en esta animación ha sido ajustar la velocidad a la que se mueve la lengua. Tras varias pruebas se ha llegado a la animación que se muestra en la Figura 5.5.2.

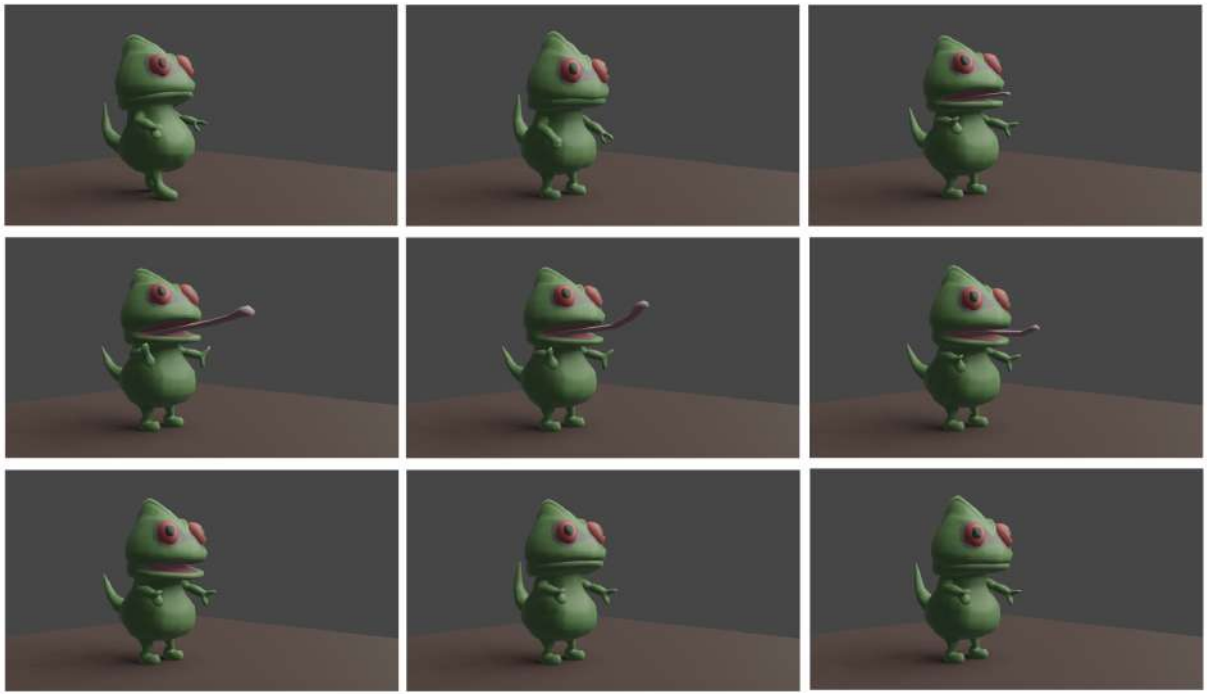


Figura 5.5.2: Animación de robar del camaleón.

5.5.3. Acción de esconderse

Para terminar, el camaleón, tras asustarse, se vuelve invisible. Esta animación es muy distinta a las demás, ya que no consiste únicamente en animar el esqueleto del personaje, sino su textura. En todos los personajes se han usado materiales 'Principled BDSF', los cuales cuentan con un canal 'Alpha' que refiere a la opacidad del objeto (donde 1 es completamente opaco y 0, transparente). Este canal es el que se ha animado, de forma que el personaje se haga progresivamente menos visible a partir del momento en el que se asusta. Para que este cambio de visibilidad no sea tan brusco, y atendiendo a los 12 principios de la animación, se ha animado a modo de anticipación al camaleón sobresaltándose.

Para conseguir esta animación se han buscado referencias para el movimiento del personaje al asustarse y para la animación de texturas. Los que han resultado más útiles son:

- '*Scared Animation*' (Valdez, 2011).
- '*TOM SCARED AND RUN ANIMATION*' (BOOK, 2022).
- '*Blender Tutorial - Animate transparency*' (Study, 2021).

Gracias a estas se ha podido conseguir el resultado que se muestra en la Figura 5.5.3.

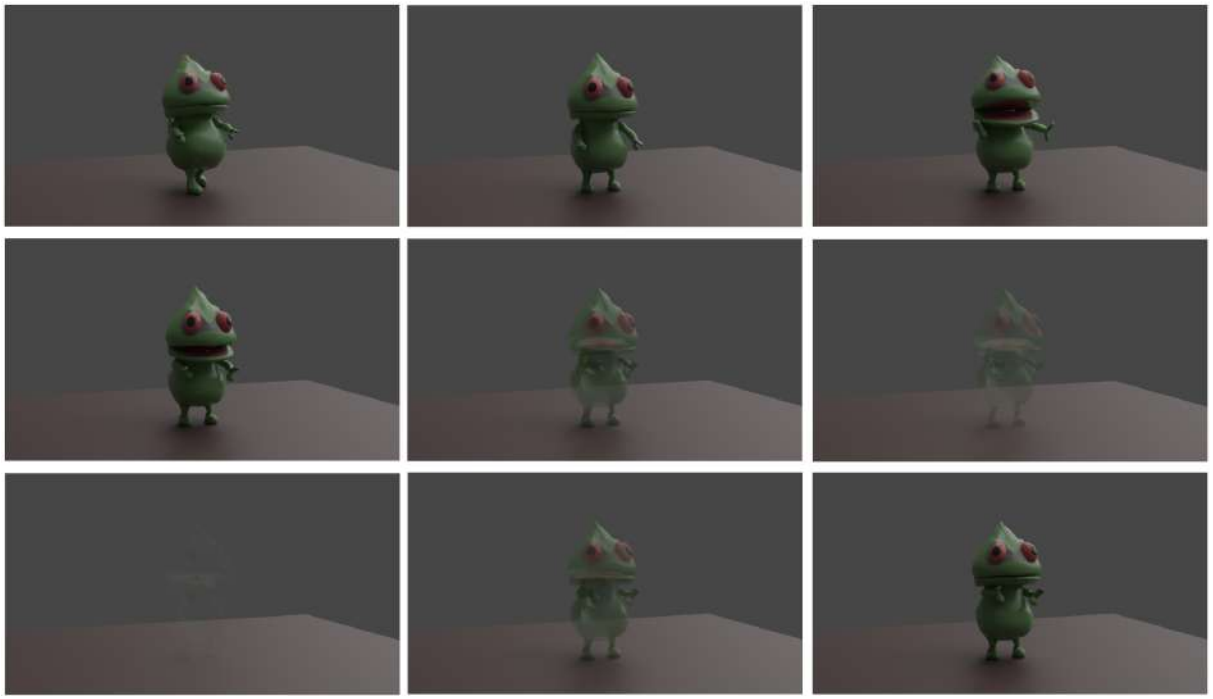


Figura 5.5.3: Animación de esconderse del camaleón.

5.6. Inclusión del camaleón en el motor de juego

La inclusión en Unity de este último personaje se ha hecho de la misma manera que los dos anteriores. En primer lugar, se ha exportado de Blender e inmediatamente después, importado en Unity. Se han asociado las texturas al modelo y se han ajustado los frames de inicio y de final en cada animación. Para terminar, se ha desarrollado un animator controller con la misma estructura que en los otros personajes y se ha hecho uso del script que asocia los comandos de teclado con los triggers de animación (ver Figura 3.6.1).

Además, este personaje requiere un trabajo extra: una de las animaciones requiere que el personaje se vuelva invisible temporalmente, y esto debe hacerse en Unity. Para conseguir esto, se creado nuevas curvas en cada una de las animaciones, en las que se ha modificado el canal Alpha de cada material, el correspondiente a la transparencia del mismo. Como se ve en la Figura 5.6.1, en la animación de esconderse el valor del alpha baja (hatsa 0, donde es completamente transparente) y vuelve a aumentar. De esta forma, el camaleón se hace invisible progresivamente y se vuelve a hacer visible de la misma manera. Por el contrario, en las animaciones de perseguir y robar, la curva correspondiente a este valor es constante, ya que siempre tiene valor 1.

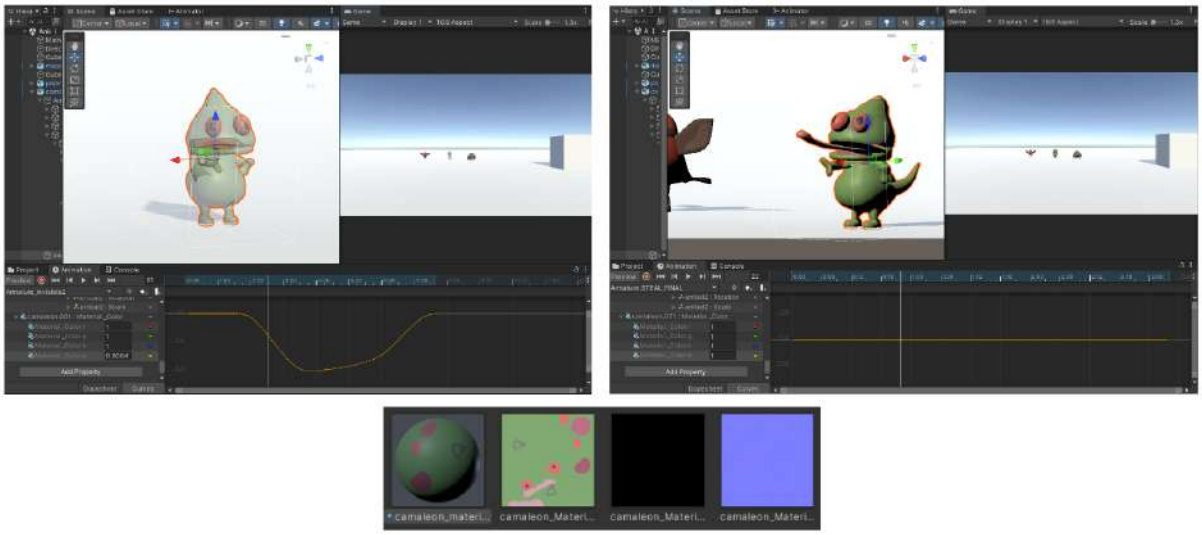


Figura 5.6.1: Camaleón insertado en Unity y curvas de animación del canal alpha.

5.7. Revisión del sprint 4

La última iteración del proyecto ha tenido una duración de 25 días, lo estimado inicialmente. Esta diferencia respecto a los sprints de los dos otros personajes probablemente se ha dado porque el movimiento del camaleón es el más parecido al de una persona humana, por lo que el rigging y la creación de las animaciones ha resultado algo más sencillo que el de los otros dos animales.

Una vez terminado el trabajo realizado en este sprint, se puede dar por finalizado el proyecto, ya que se han completado los tres objetivos específicos restantes:

Obj 2

Diseñar y desarrollar en 3D tres de los personajes del juego.

Obj 3

Implementar las animaciones básicas de los tres personajes.

Obj 4

Incluir a cada personaje, junto con sus animaciones, en el motor de juego Unity.

Y, por lo tanto, se puede dar por finalizado el objetivo principal del proyecto:

Objetivo principal

Construir una galería de personajes para un videojuego.

Capítulo 6

Conclusiones

En este capítulo se presentan las conclusiones del proyecto: los objetivos que se han cumplido en este y los posibles futuros trabajos que pueden llevarse acabo, siguiendo con la línea de trabajo presentada en este.

6.1. Objetivos cumplidos

Al inicio de este proyecto se planearon una serie de objetivos que se han ido completando a lo largo de los distintos sprints.

Obj 1. Diseñar el videojuego.

Se ha diseñado el videojuego '¡Al ladrón!', un juego del género endless runner en el que el jugador debe escapar de unos animales que intentarán robarle fruta de un cesto que lleva consigo. Los detalles de este videojuego se han plasmado en un Game Design Document, en el que se ha incluido información del mismo, por ejemplo, el género al que pertenece, su sinopsis, sus mecánicas o las interfaces que tendrá. Además, se han planteado los personajes que se han diseñado y animado en los sprints posteriores del proyecto.

Obj 2. Diseñar y desarrollar en 3D tres de los personajes del juego.

Se han diseñado conceptual y visualmente los tres enemigos del videojuego: un mapache, un mosquero cardenal y un camaleón. Los tres juntos forman el 'Club del antifaz negro' y se dedican a perseguir al jugador y robarle fruta.

Se han definido una serie de cualidades para cada uno de los personajes y se ha hecho un diseño visual de los mismos usando una gran variedad de referencias. Se ha reflejado este diseño en una ilustración turn-around, la cuál se ha usado después como guía para modelar a los personajes en 3D en Blender [15].

Por último, estos modelos se han texturizado, haciendo primero el mapeo de UVs en Blender y, posteriormente, pintando las texturas en Substance 3D Painter [5].

Obj 3. Implementar las animaciones básicas de los tres personajes.

Se han desarrollado tres animaciones para cada uno de los personajes, correspondientes a las tres acciones que realizan estos personajes en el juego: perseguir al jugador, robarle y esconderse de él. Las animaciones de cada personaje se han diseñado dependiendo de las características del diseño de cada uno de ellos.

Obj 4. Incluir a cada personaje, junto con sus animaciones, en el motor de juego Unity [90].

Se han exportado los modelos 3D y las animaciones de cada uno de los personajes y se han importado en el motor de juego Unity. Se han configurado sus animadores y se les han asociado controladores para poder cambiar de una animación a otra. De esta forma, el personaje ejecuta continuamente la acción de perseguir, al pulsar 'Espacio' se ejecuta la acción de esconderse y al pulsar 'Shift Izquierdo', la acción de robar, para que se puedan incluir en un futuro en un videojuego.

6.2. Futuros trabajos

Una vez terminado este proyecto se plantean algunos posibles futuros trabajos con los que seguir en este hilo de desarrollo.

El primero de ellos consiste en el desarrollo del juego y la inserción de los personajes en el mismo. Esto se ha comenzado a desarrollar en el Trabajo de Fin de Grado del Grado en Ingeniería de Computadores (Fresno, 2024), en el que se ha desarrollado un prototipo del videojuego y los comportamientos de los personajes. Como futura contribución sería interesante insertar las animaciones de los personajes en este prototipo.

Además, sería interesante diseñar e implementar más animaciones para los personajes ya existentes. Por ejemplo, se podrían animar distintas maneras de moverse dependiendo de las circunstancias: esconderse siempre que tengan un obstáculo cerca en lugar de salir volando o camuflarse, o avanzar a distintas velocidades dependiendo de la distancia a la que se encuentre el jugador.

Estos trabajos servirían tanto para continuar el proyecto y poder presentar un videojuego más terminado, como para practicar y mejorar en el campo del desarrollo de elementos 3D.

Referencias

A. S. Douglas. (1952). *Noughts and crosses.* Descargado de https://videojuegos.fandom.com/es/wiki/OXO

Adell. (2016, Enero). Historia de los videojuegos: La generación de las 16 Bits. *https://www.destinorpg.es/2016/01/historia-de-los-videojuegos-la.html*.

Adobe. (s.f.-a). *Historia de los 12 principios de la animación.* https://www.adobe.com/es/creativecloud/animation/discover/principles-of-animation.html. (Consultado en noviembre de 2024)

Adobe. (s.f.-b). *Photoshop.* Descargado de https://www.adobe.com/es/products/photoshop/landpa.html?gclid=Cj0KCQiA88a5BhDPARIsAFj595iu_Gwiz7U7i3T3k6oq6_-ezxhXqgnxVE0nMwqr7xyl9rJWNE6XzxsaAikkEALw_wcB&mv=search&s_kwcid=AL!3085!3!717364996486!e!!g!!photoshop!1445901735!56657232416&mv=search&mv2=paidsearch&sdid=2XBSBWBF&ef_id=Cj0KCQiA88a5BhDPARIsAFj595iu_Gwiz7U7i3T3k6oq6_-ezxhXqgnxVE0nMwqr7xyl9rJWNE6XzxsaAikkEALw_wcB:G:s&s_kwcid=AL!3085!3!717364996486!e!!g!!photoshop!1445901735!56657232416&gad_source=1 (Accessed: 2024-11-11)

Adobe. (s.f.-c). *Substance 3d painter.* Descargado de https://www.adobe.com/es/products/substance3d.html (Accessed: 2024-11-11)

Alvarado, A. C., y Planells, A. J. (2020). *Ficción y videojuegos: teoría y práctica de la ludonarración.* Editorial UOC.

Animation, J. M. (2019a). *Birds in flight - video reference for animators.* Descargado de https://youtu.be/qThIyj1mLfs

Animation, J. M. (2019b). *Birds take flight - video reference for animators.* Descargado de https://youtu.be/CJHP6dPjuGY

Animation, J. M. (2019c). *Birds walking - video reference for animators.* Descargado de https://youtu.be/NlbFbdYuX-w

annimates. (2023). *Sneaky - 3d sneak walk animation.* Descargado de https://youtu.be/pfRoAkC8HSE

Art, F. S. M. (2020). *quadruped locomotion.* Descargado de https://youtu.be/tLrRlXxM5Yw

Autodesk. (s.f.-a). *3ds max.* Descargado de https://www.autodesk.com/es/products/3ds-max/overview?term=1-YEAR&tab=subscription (Accessed: 2024-11-11)

Autodesk. (s.f.-b). *Maya.* Descargado de https://www.autodesk.com/es/products/maya/overview?term=1-YEAR&tab=subscription (Accessed: 2024-11-11)

Belli, S., y Raventós, C. L. (2008). Breve historia de los videojuegos. *Athenea Digital. Revista de pensamiento e investigación social*(14), 159–179.

Blender Foundation. (s.f.). *Blender.* Descargado de https://www.blender.org/download/ (Accessed: 2024-11-11)

BOOK, C. (2022). *Tom scared and run animation.* Descargado de https://youtu.be/dLkRHZOsSRM

Canós, J. H., Letelier, P., y Penadés, M. C. (2003). Metodologías ágiles en el desarrollo de software. *Universidad Politécnica de Valencia, Valencia*, 1–8.

Cave, T. G. D. (2023). *How to export animation from blender for game engines like unity or unreal.* Descargado de https://youtu.be/kUn4jUj4gOk

CELSYS. (s.f.). *Clip studio paint.* Descargado de https://www.clipstudio.net/es/?gad_source=1&gclid=Cj0KCQiA88a5BhDPARIsAFj595gppsTQqSmAuER-IaIEiweODvkwqdk4hSAhgVkNVJ_Z0_Ql-4nBIxQaAt4qEALw_wcB (Accessed: 2024-11-11)

Conway, A. (2021). Game design document. *Game Design and Development 2021.*

Coterón, L. S. (2012). Arte y videojuegos: mecánicas, estéticas y diseño de juegos en prácticas de creación contemporánea. *PhD diss. Universidad Complutense de Madrid.*

Daniel Mullins Games. (2021). *Inscryption.* Descargado de https://www.inscryption.com/

DinoRPG. (2018). *Racoon walk cycle.* Descargado de https://youtu.be/fe9rvulVq2I

Earth, B. (2013). *Chameleon tongue in slow motion | one life | bbc earth.* Descargado de https://youtu.be/z3oh73amxQo

Eddie Meardon. (s.f.). *¿qué son los diagramas de gantt? un resumen de los diagramas de gantt y de cómo pueden ayudar a gestionar proyectos de metodología ágil.* https://www.atlassian.com/es/agile/project-management/gantt-chart. (Consultado en noviembre de 2024)

Electronic Arts. (2014). *The sims 4.* Descargado de https://www.ea.com/es/games/the-sims/the-sims-4

Fresno, L. (2024). *Diseño e implementación de comportamientos para personajes del videojuego '¡Al ladrón!'.* (Trabajo de Fin de Grado. Grado en Ingeniería de Computadores. Universidad Rey Juan Carlos.)

García, F. G. (2006). Videojuegos y virtualidad narrativa. *ICONO 14, Revista de comunicación y tecnologías emergentes*, *4*(2), 1–24.

Gómez, P. (2021). *Chicken walk cycle: A breakdown.* Descargado de https://youtu.be/z3FvkwGcWtU?t=18

Hernández, J. G. (2019). La teoría narrativa del videojuego: Intertextualidad, hipertexto y videojuego. *Revista laboratorio*(18).

id Software. (1992). *Wolfstein 3d.* Descargado de https://wolfenstein.fandom.com/es/wiki/Wolfenstein_3D

id Software. (1993). *Doom.* Descargado de https://doom.fandom.com/es/wiki/Doom

Infogrames. (1992). *Alone in the dark.* Descargado de https://aloneinthedark.fandom.com/wiki/Alone_in_the_Dark_(1992)

Innersloth. (2018). *Among us.* Descargado de https://store.epicgames.com/es-ES/p/among-us

Johnston, O., y Thomas, F. (1981). *The illusion of life: Disney animation.* Disney Editions New York.

Kent, S. L. (2016). *La gran historia de los videojuegos.* Nova.

Kinetic Games. (2020). *Phasmophobia.* Descargado de https://www.kineticgames.co.uk/phasmophobia

Kumar, A. (2013). *Question – raccoon on run.* Descargado de https://youtu.be/Nv-IHWL890w

Kuzillon. (2024a). *How to animate an animal trot cycle!* Descargado de https://youtu.be/Y-3E0IK2G-U

Kuzillon. (2024b). *How to animate flight.* Descargado de https://youtu.be/yRwpCkQzd4U

Maida, E. G., y Pacienzia, J. (2015). Metodologías de desarrollo de software. *Tesis de Licenciatura en Sistemas y Computación. Facultad de Química e Ingeniería "Fray Rogelio Bacon".*

Mater, T. (2020a). *Fast tongues l chameleons in slow-motion.* Descargado de https://youtu.be/IWZNORWXNQ8

Mater, T. (2020b). *Hummingbirds in slow-motion | high-speed wings | wild to know.* Descargado de https://youtu.be/x5Wjz23EasA

Maxon. (s.f.-a). *Cinema 4d.* Descargado de https://www.maxon.net/es/cinema-4d (Accessed: 2024-11-11)

Maxon. (s.f.-b). *Zbrush.* Descargado de https://www.maxon.net/en/zbrush?srsltid=AfmBOorbibMPz9emWYuqexz3epQlEswwUlMgtC5twBv9LRbxrbQYgddT (Accessed: 2024-11-11)

Mediatonic. (2020). *Fall guys.* Descargado de https://www.fallguys.com/es-ES

Montero, B. M., Cevallos, H. V., y Cuesta, J. D. (2018). Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software. *Espirales revista multidisciplinaria de investigación*, *2*(17).

Namco. (1980). *Pac-man.* Descargado de https://pacman.fandom.com/wiki/Pac-Man_(game)

Namco. (1982). *Pole position.* Descargado de https://namco.fandom.com/wiki/Pole_Position

Nintendo. (1991a). *The legend of zelda: A link to the past.* Descargado de https://www.nintendo.com/es-es/Juegos/Super-Nintendo/The-Legend-of-Zelda-A-Link-to-the-Past-841179.html?srsltid=AfmBOoq2H59h3nTvFlhMBz3797H8mk7jT6fmcoc0KZDrZbTUGDaiJwQb

Nintendo. (1991b). *Super mario world.* Descargado de https://www.nintendo.com/es-es/Juegos/Super-Nintendo/Super-Mario-World-752133.html?srsltid=AfmBOoq97r26zHrT3yiaN6G0LPThbz8O8VTNvJyK8WAcKomxx_XGP3d2

Nintendo. (1994). *Donkey kong country.* Descargado de https://www.nintendo.com/es-es/Juegos/Super-Nintendo/Donkey-Kong-Country-276896.html?srsltid=AfmBOopyWCPvzXiW2p_Xx6r9ATWDGTL6ZbRiUFV357id-N9Vxr-UKo_o

Nintendo. (1996). *Super mario 64.* Descargado de https://mario.fandom.com/es/wiki/Super_Mario_64

Nintendo. (2017). *The legend of zelda: Breath of the wild.* Descargado de https://zelda.fandom.com/es/wiki/The_Legend_of_Zelda:_Breath_of_the_Wild

Nintendo. (2020). *Animal crossing: New horizons.* Descargado de https://animalcrossing.nintendo.com/es/

of Aaron Blaise, T. A. (2016). *Animation - the mechanics of bird flight.* Descargado de https://youtu.be/2FR982O37dw

Ojeda, J. C., y Fuentes, M. d. C. G. (2012). Taxonomía de los modelos y metodologías de desarrollo de software más utilizados. *Universidades*(52), 37–47.

Oltra, A. G. (2020). La revolución de los píxeles: arte y videojuegos en un mundo digital. *Eviterna*(7), 88–102.

Patterson, K. (2018). *Sneak cycle reference.* Descargado de https://youtu.be/oCGJFG_Df3w

Pitko, C. (2022). *Between gaming dimensions: An analysis on 2.5d and pseudo 3d animation techniques throughout video game history* (Ph.D. thesis). California State Universit.

Psyonix. (2015). *Rocket league.* Descargado de https://www.rocketleague.com/es-es

Raccoon, T. T. (2020). *Asmr | raccoon eating noises.* Descargado de https://youtu.be/gRE09r3fAtQ

Raju, P. (2019). *Character rigging and advanced animation.* Springer.

Ref, S. (2014). *Raccoon walking and climbing (slow motion animation reference).* Descargado de https://youtu.be/uohw49UDu_4

references, A. (2021). *bird reference.* Descargado de https://youtu.be/mh09uYX_uCI

Rivas, C. I., Corona, V. P., Gutiérrez, J. F., y Hernández, L. (2015). Metodologías actuales de desarrollo de software. *Revista de Tecnología e Innovación*, *2*(5), 980–986.

Sánchez, J. G., Zea, N. P., Gutiérrez, F., y Cabrera, M. (2008). De la usabilidad a la jugabilidad: Diseño de videojuegos centrado en el jugador. *Proceedings of INTERACCION*, 99–109.

Sánchez Rodríguez, P. A., Alfageme González, M. B., y Serrano Pastor, F. J. (2010). Aspectos sociales de los videojuegos/social aspects about video games. *Revista Latinoamericana de Tecnología Educativa-RELATEC*, *9*(1), 43–52.

Sastre, C., Rocha, R., Pequeño, J. M., y López, D. (2020). *Diseño y creación de personajes: desde el boceto hasta su realización 3d.* Parramón.

Schwaber, K., y Sutherland, J. (2013). La guía de scrum. *Scrumguides. Org*, *1*, 21.

Scrum.org. (2016). *¿qué es scrum?* Descargado de https://www.scrum.org/resources/what-is-scrum (Consultado en mayo de 2018)

Sega. (1991). *Sonic the hedgehog.* Descargado de https://sega.fandom.com/es/wiki/Sonic_the_Hedgehog_(1991)

Sega. (1992). *Virtual racing.* Descargado de https://segaretro.org/Virtua_Racing

Singh, P. (2021). *Bird flying 2d animation.* Descargado de https://youtu.be/pITFvYHaevM

Solórzano Alcívar, N., Moscoso Poveda, S., y Elizalde Ríos, E. (2019). Evolución de videojuegos y su línea gráfica _ un enfoque entre la estética y la tecnología _. *Ñawi: arte diseño comunicación*, *3*(2), 125–145.

Square Enix. (1995). *Chrono trigger.* Descargado de https://www.square-enix-games.com/es_XL/games/chrono-trigger

Square Enix. (1997). *Final fantasy vii.* Descargado de https://finalfantasy.fandom.com/es/wiki/Final_Fantasy_VII

Square Enix. (2018). *Octopath traveler.* Descargado de https://www.square-enix-games.com/en_EU/games/octopath-traveler

Steve Russell. (1962). *Spacewar!* Descargado de https://es.wikipedia.org/wiki/Spacewar!

Stoneham, B. (2012). *Como crear arte fantástico para videojuegos.* Norma.

Studio MDHR. (2017). *Cuphead.* Descargado de https://cuphead.fandom.com/es/wiki/Cuphead_(videojuego)

Study, B. (2021). *Blender tutorial - animate transparency.* Descargado de https://youtu.be/TGNCuoas2k0

sugarspice. (2024). *pascal carrying tangled with zero lines.* Descargado de https://youtu.be/J61jLpRAqs0

Taito. (1978). *Space invaders.* Descargado de https://spaceinvaders.fandom.com/wiki/Space_Invaders

Tato, G. (2017). *Análisis del personaje en el cine y en los videojuegos. inmersión y empatía.* Universitat d'Alacant. Vicerectorat de Cultura, Esport i Llengües.

Thatgamecompany. (2012). *Journey.* Descargado de https://thatgamecompany.com/#games/journey/

to Be an Animator Animation Tutorials, C. P. I. W. (2021). *16 walk cycles - different attitudes.* Descargado de https://youtu.be/8XDrUs5kMZI?t=342

Toby Fox. (2024). *Undertale.* Descargado de https://undertale.com/

Turley, F., y Rad, N. K. (2019). *Los fundamentos de agile scrum.* Van Haren.

Unity Technologies. (s.f.). *Unity.* Descargado de https://unity.com/es (Accessed: 2024-11-11)

Valdez, J. C. (2011). *Scared animation.* Descargado de https://youtu.be/vbJF3GAivLw

Velazquez, A. D. P., y otros. (2018). Aproximación a la función del color y forma en el diseño de personajes. *Universidad de Guanajuato.*

William Higinbotham. (1958). *Tennis for two.* Descargado de https://en.wikipedia.org/wiki/Tennis_for_Two

Yacelga, A. R. L., y Cabrera, M. A. C. (2022). Uso de tableros kanban como apoyo para el desarrollo de las metodologías ágiles. *Universidad y Sociedad*, *14*(S2), 208–214.