Integración de media con Unity

PID_00248065

Andreu Gilaberte Redondo

Universitat Oberta de Catalunya

Índice

Introducción

En los módulos anteriores hemos visto como crear, editar y producir los principales *media* que utilizamos en un videojuego. Básicamente hablamos de modelar escenas 3D, crear personajes, aplicar texturas, animar, iluminar, crear e integrar una banda sonora con todos sus componentes: voz, diálogos, banda musical o efectos de sonido.

A diferencia de otros medios audiovisuales, en los que los diversos *media* se crean e integran en un producto cerrado, en un videojuego los diversos elementos que forman los *media* se reproducen, integran y ejecutan con frecuencia en tiempo real. Normalmente respondiendo a la interacción del usuario con el juego.

Unity permite esta integración e interacción en tiempo real. Y a diferencia de softwares de creación audiovisual, parámetros finales del videojuego como la iluminación, la texturización, la animación o el sonido acaban siendo el resultado de combinar componentes y funcionalidades del programa con los *media* creados en la fase inicial.

Vamos a ver los recursos y procedimientos mediante los cuales Unity trabaja la interacción de los *media*. En concreto nos centraremos en el flujo de trabajo de *assets*. Dicho flujo de trabajo facilita que *assets* procedentes de una variedad de fuentes en utilicen en Unity. Estos *assets* incluyen gráficos y sonido que han sido creados con programas externos, también archivos Package procedentes de otros desarrolladores, *assets* que tenemos en nuestra *Asset* Store y los *assets* estándar incluidos en Unity.

Enlace

Empezando a conocer el entorno de Unity y su flujo de trabajo.

https://docs.unity3d.com/es/current/Manual/UnityOverview.html

Objetivos

Después de haber estudiado y trabajado este los contenidos que se presentan en este módulo, seréis capaces de:

1. Saber explorar y seleccionar recursos para Unity en los principales portales contenedores.

2. Saber identificar los Media específicos que sean susceptibles de ser aplicados a un proyecto determinado.

3. Saber integrar Media externos a un proyecto dentro de un entorno Unity.

4. Conocer y practicar con técnicas de modelado con programas específicos.

5. Conocer y practicar el funcionamiento del sistema de huesos

6. Saber aplicar acciones a los personajes en base a los principios de cinemática.

7. Saber optimizar geometrías de modelos aplicando técnicas de compresión y reducción de mallas y polígonos.

8. Saber definir y construir diferentes elementos orgánicos de una escena dentro de un entorno Unity.

9. Conocer y aplicar los principios de los ciclos de animación a un proyecto integral en Unity

10. Saber utilizar un lenguaje básico de programación para trabajar con assets pre definidos para la optimización de las animaciones y los elementos propios del control de personajes de Unity.

1. *Assets*

Habitualmente, en los proyectos de videojuegos tenemos la necesidad de utilizar elementos diversos, como luces, texturas, animaciones, etc. La complejidad de la elaboración de un proyecto con más envergadura hace que el tiempo de desarrollo de cada elemento se multiplique de forma exponencial. Ello nos lleva a la necesidad de valorar el hecho de poder utilizar elementos ya creados por otros desarrolladores y que se pueden importar en nuestro proyecto personal. Es lo que llamamos en Unity, *assets*.

Según el diccionario de términos de videojuegos:

Asset Del inglés ***asset*** (activo, recurso). Cada uno de los elementos que componen el juego (animaciones, modelos, IA, sonidos, etc).

Podemos ver en el manual de Unity la descripción de *asset*:

> «Un *asset* es una representación de cualquier ítem que puede ser utilizado en su juego o proyecto. Un *asset* podría venir de un archivo creado afuera de Unity, tal como un modelo 3D, un archivo de audio, una imagen, o cualquiera de los otros tipos de archivos que Unity soporta. También hay otros tipos de *asset* que pueden ser creados dentro de Unity, tal como un Animator Controller, un Audio Mixer o una Render Texture.»

Unity dispone de un contenedor de *assets* que permite cargar los elementos necesarios en cada momento del flujo de trabajo en el proyecto de un videojuego. La tienda virtual es el espacio donde se pueden encontrar una serie de *assets* que se encuentran clasificados según varios criterios se encuentra en:

<https://www.assetstore.Unity3d.com/en/>

1.1. Importación de *assets*

Unity es un programa extremamente potente en cuanto a la generación de interacción en videojuegos, y como tal tiene una comunidad extensa de desarrolladores que están constantemente publicando *assets* que pueden ser útiles y ahorrarnos un buen tiempo de desarrollo.

Según el manual de trabajo *online* de Unity:

«Importando desde la *Asset* Store

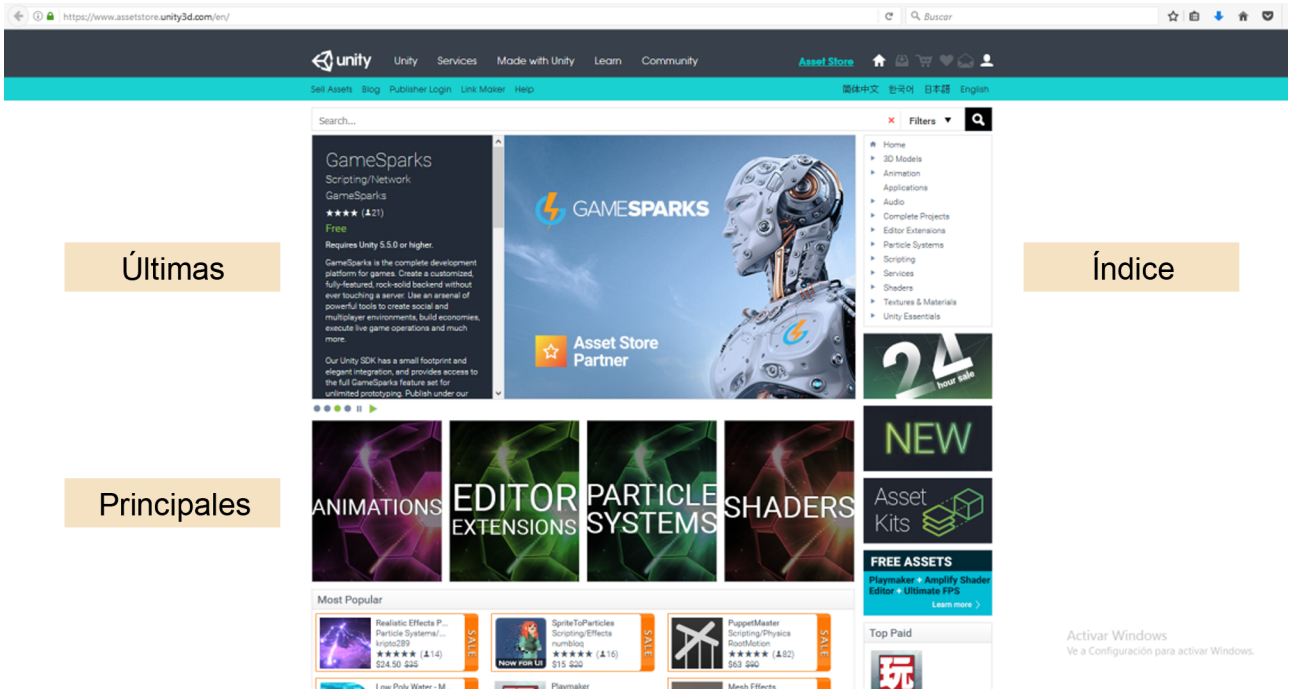
El *Asset* Store de Unity es el hogar de una creciente biblioteca de *assets* comerciales y gratuitos creados por Unity Technologies y miembros de la comunidad. Hay una gran cantidad de *assets* disponibles, desde texturas, modelos y animaciones hasta ejemplos de proyectos completos, tutoriales y extensiones del editor. Estos *assets* son accesibles desde una interfaz simple dentro del Editor Unity y son descargados e importados directamente en sus proyectos.

Los usuarios de Unity pueden volverse publicadores en el *Asset* Store, y vender contenido que estos han creado. Este proceso es documentado en más detalle más tarde de esta sección.»

1.2. Sistema organizativo de la *Store Unity*

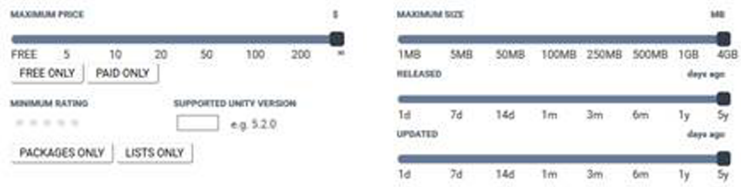
La tienda virtual de *assets* de Unity está organizada de forma que se divide en apartados en los que es posible buscar el tipo de complemento que el usuario precise en un momento dado. En el siguiente gráfico mostramos las partes principales de dicha organización.

Imagen 1. Tienda virtual de Unity



Cabe destacar que en el apartado de índice de complementos, tenemos acceso a un filtro específico que facilita la búsqueda del *asset* que buscamos. Se trata de un filtro que nos permite por ejemplo buscar solo los *assets* que no son de pago o los que presentan un determinado tamaño. Ello supone una facilitación de los aspectos más convenientes a la hora de seleccionar un determinado *asset*.

Imagen 2. Filtro de *assets*



Este filtro nos permite por ejemplo buscar solo los *assets* que no son de pago o con un determinado tamaño. Facilitando los aspectos que más nos convenga a la hora de seleccionar un determinado *asset*.

1.3. Acceso a la *Asset Store* e importación

Pasamos a comentar el proceso para la importación de *assets* en Unity. Algunos de ellos son de pago, otros gratuitos. Todos responden a la labor de una comunidad extensa de desarrolladores que constantemente publican novedades. Tenemos una descripción detallada del proceso a seguir en la página Importando *Assets*, de la que citamos a modo de introducción que:

> «Los *Assets* creados afuera de Unity deben ser traídos a Unity al tener el archivo ya sea guardado directamente a la carpeta "*Assets*" de su proyecto, o copiado a esa carpeta. Para muchos formatos comunes, usted puede guardar su archivo fuente directamente a la carpeta *Assets* de su proyecto y Unity será capaz de leerlo. Unity va a notar cuando usted ha guardado nuevos cambios al archivo y va a re-importarlo si es necesario.»

Continuando con la información que ofrece el manual, recomendamos la lectura del apartado referido a los ajustes de importación.

> «Cada tipo de *asset* que Unity soporta tiene un conjunto de *Import Settings* (ajustes de importación), que afectan cómo el *asset* aparece o se comporta. Para ver los *import settings* de un *asset*, seleccione el *asset* en el Project View. Los *import settings* para este *asset* van a aparecer en el Inspector. Las opciones que son mostradas van a variar dependiendo en el tipo de *asset* que fue seleccionado.»

Asimismo, es importante la información referida a la AssetStore, no únicamente como consumidores que acuden a una fuente de información sino también como posibles proveedores activos de la misma.

> «El *Asset* Store de Unity es el hogar de una creciente biblioteca de *assets* comerciales y gratuitos creados por Unity Technologies y miembros de la comunidad. Hay una gran cantidad de *assets* disponibles, desde texturas, modelos y animaciones hasta ejemplos de proyectos completos, tutoriales y extensiones del editor. Estos *assets* son accesibles desde una interfaz simple dentro del Editor Unity y son descargados e importados directamente en sus proyectos.»

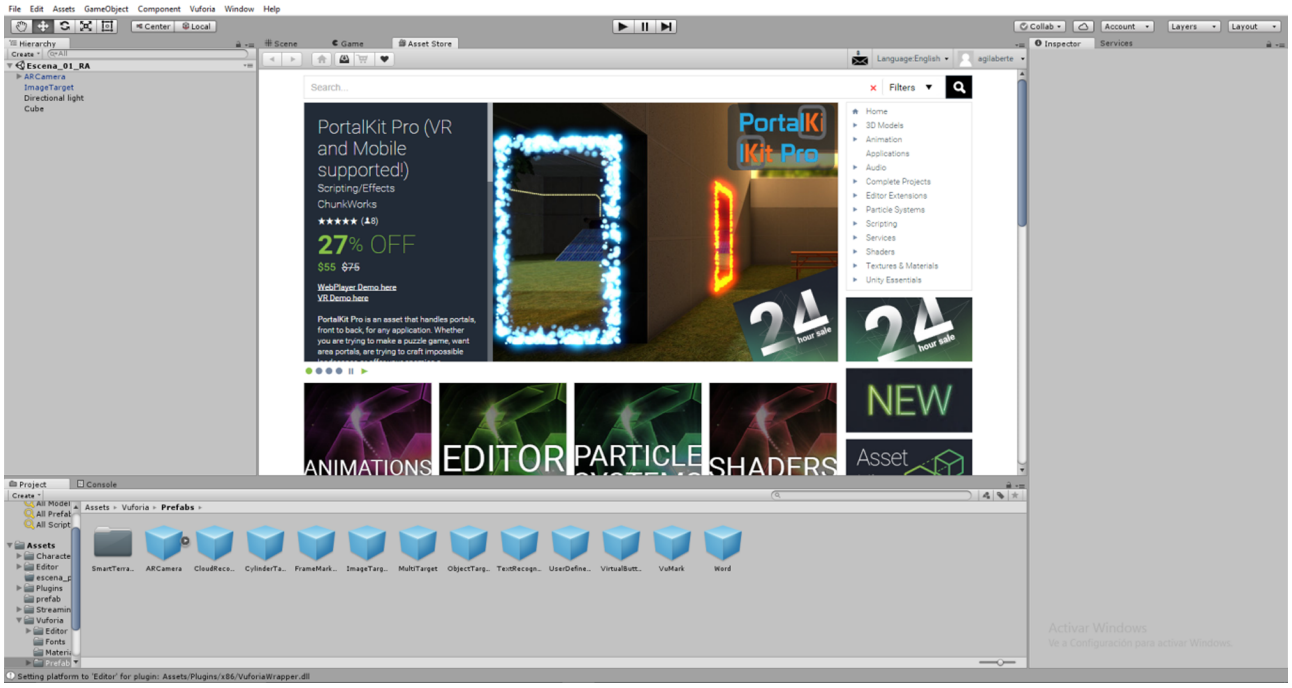
Existen dos formas de trabajar con los *assets* de la tienda de Unity. Una es abrirla directamente des de la interfaz del programa y otra es accediendo a la página web antes descrita.

Enlace

Importación de assets desde la tienda de Unity:

https://docs.unity3d.com/es/current/Manual/ImportingAssets.html

Imagen 3. Entorno de Unity – Acceso a la biblioteca de *assets* de Unity



1.4. *AssetsPackges* (Paquetes de *assets*)

El proceso de elaboración de escenas y personajes, así como elementos auxiliares, como luces, texturas, etc., son laboriosos y consumen gran cantidad de tiempo y dedicación. Hemos hablado que en la tienda de Unity encontramos elementos individuales para nuestro proyecto. Estos elementos están divididos por temas y es fácil encontrar aquellos que más nos convenga para importar a nuestro trabajo. Pero lo que realmente hace interesante la diversidad de la tienda, es la posibilidad de poder importar un proyecto entero, es lo que denominamos ***assets packges***. Estos *packs* están formados por varios archivos que están correctamente insertados en un proyecto, de manera que el usuario pueda utilizarlo inmediatamente después de su importación.

Son conjuntos formados por varios elementos dispuestos en una o varias escenas y totalmente editables.

«Los packages de Unity son una manera útil de compartir y re-utilizar proyectos al igual que colecciones de *assets*; Los *Standard Assets* de Unity e ítems en la *Asset* Store de Unity son proporcionados en packages (paquetes), por ejemplo. Los Packages son una colección de archivos y datos de proyectos de Unity, o elementos de proyectos, los cuales son comprimidos y almacenados en un solo archivo, similar a archivos Zip. Al igual que archivos Zip, un *package* (paquete) mantiene su estructura original de directorios cuando es desempacado, al igual que los meta-datos acerca de los *assets* (al igual que ajustes de importación y vínculos a otros *assets*).

En Unity, la opción del menú *Export Package* comprime y almacena la colección, mientras que el *Import Package* desempaqueta la colección a su proyecto de Unity abierto.»

Enlace

Como se definen los packges en el manual de Unity:
https://docs.unity3d.com/es/current/Manual/AssetPackages.html

1.5. Elementos diversos de Unity para Video Juegos

El usuario que ya domina la exportación y importación de los diferentes elementos *assets* disponibles en la tienda de Unity, podrá ahorrar mucho tiempo en la consecución del proyecto global, así mismo, habrá situaciones que tendrá de crear y trabajar sus propios elementos, específicos para un proyecto en concreto. En este caso, vamos a guiarnos por medio del manual en línea de Unity e iremos repasando los diferentes tipos de elementos que podremos trabajar directamente en el entorno Unity.

Enlace

El manual completo de Unity lo encontraremos en:

https://docs.Unity3d.com/Manual/index.html

Nota: La interface presentada en el manual puede no ser la misma que la versión de Unity utilizada.

2. Animación externa

2.1. Creando personajes con Fuse

Fuse es un programa de modelado de personajes de Adobe.

http://www.adobe.com/es/products/fuse.html

En un principio, Fuse está preparado para modelar personajes de forma muy sencilla con resultados realmente profesionales, con la posibilidad de animar a estos personajes en Adobe Photoshop como se puede apreciar en la información de la web de Adobe:

https://helpx.adobe.com/es/photoshop/using/fuse.html

Pero nosotros vamos a modelar en Fuse y luego animar en Unity, de forma que tendremos a disposición una nueva opción para conseguir modelos personalizados de forma muy rápida y con buenos resultados.

Nota

En su versión gratuita, Fuse dispone de menos elementos de personalización. El programa fase beta está disponible con el paquete de Adobe Creative Cloud.

La interface de Fuse es muy intuitiva y siguiendo pocos pasos, podemos ir creando las diferentes partes del personaje, así como cambiar algunos parámetros mas pormenorizados de forma a adaptar el personaje al estilo que nos convenga.

La primera fase de trabajo consiste en montar el personaje en cuatro partes:

- Cabeza
- Dorso
- Piernas
- Brazos

Imagen 4. Sistema de modelado de cuerpos

El programa por si mismo adapta cada parte de cuerpo y crea la totalidad del personaje lista para posibles animaciones. Con lo cual, puedes crear composiciones naturales o completamente diferentes:

Imagen5. Diferentes configuraciones de modelado de cuerpos

En el apartado de *Customize* del programa, podemos ajustar algunos parámetros del personaje de manera que se adapte al resultado buscado. Podemos trabajar con seis grupos de personalización:

- Brazos
- Rostro
- Cabeza
- Piernas
- Dientes
- Dorso

Así, por ejemplo, podemos crear diferentes configuraciones detalladas del personaje principal.

Imagen6. Configuración detallada inicial

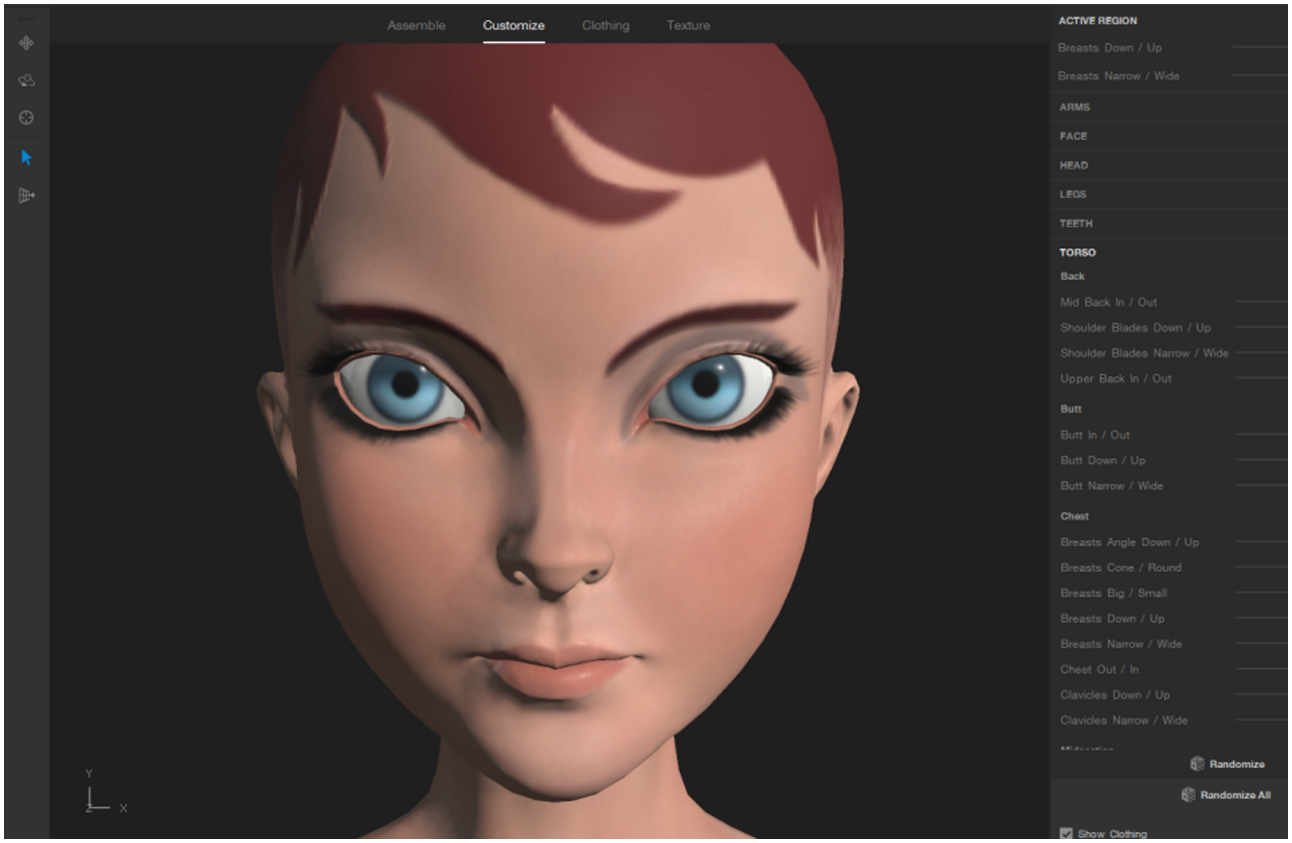
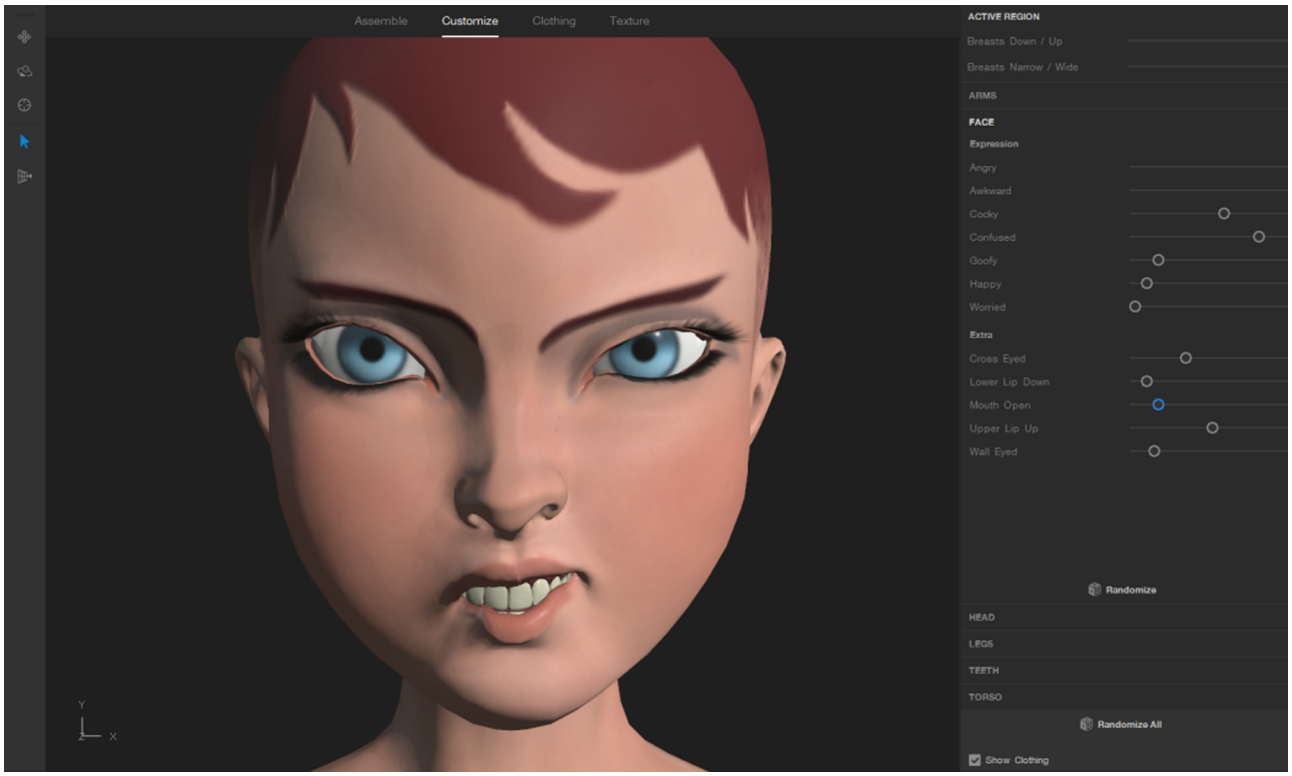
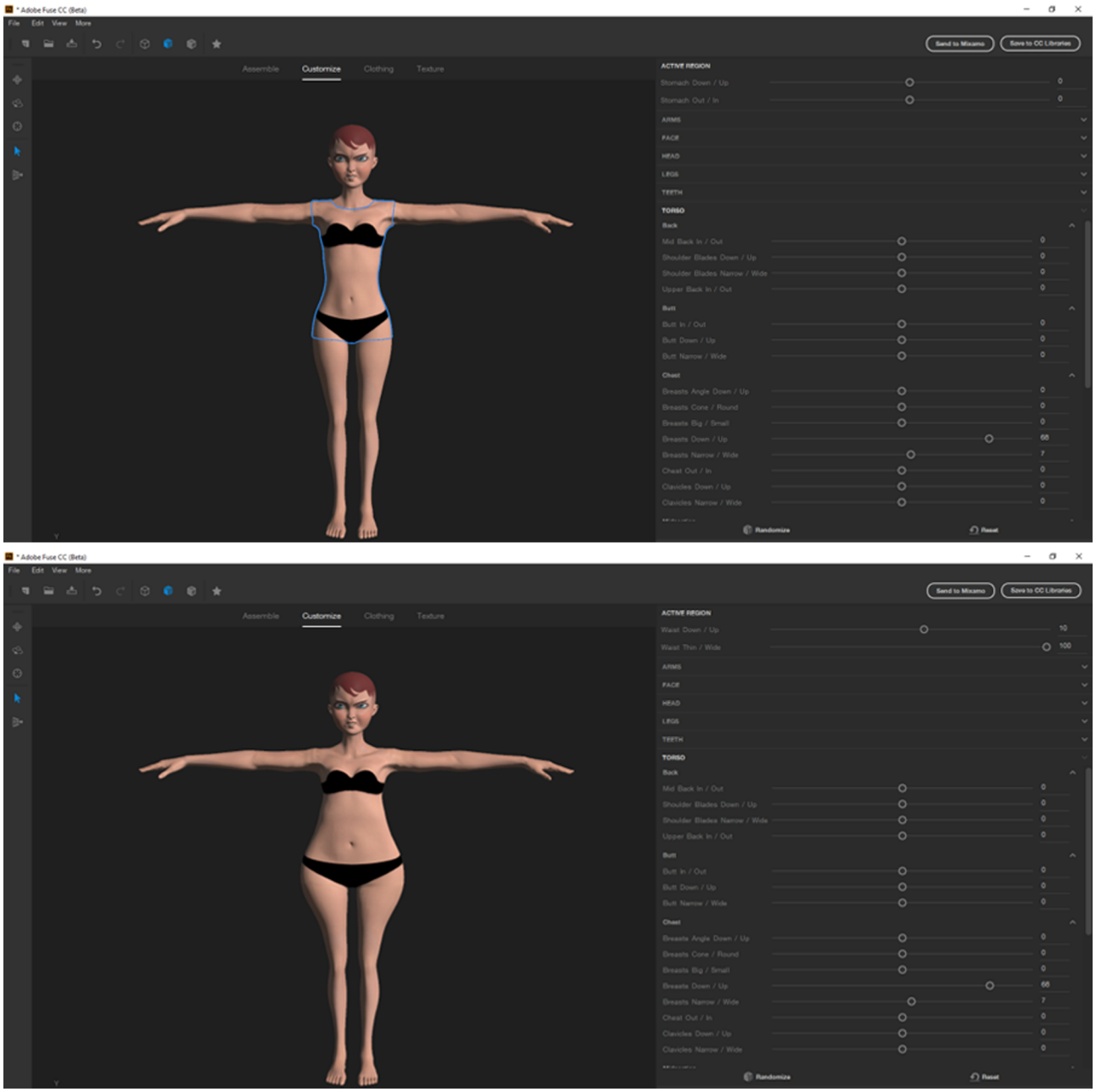


Imagen 7. Cambios en el rostro



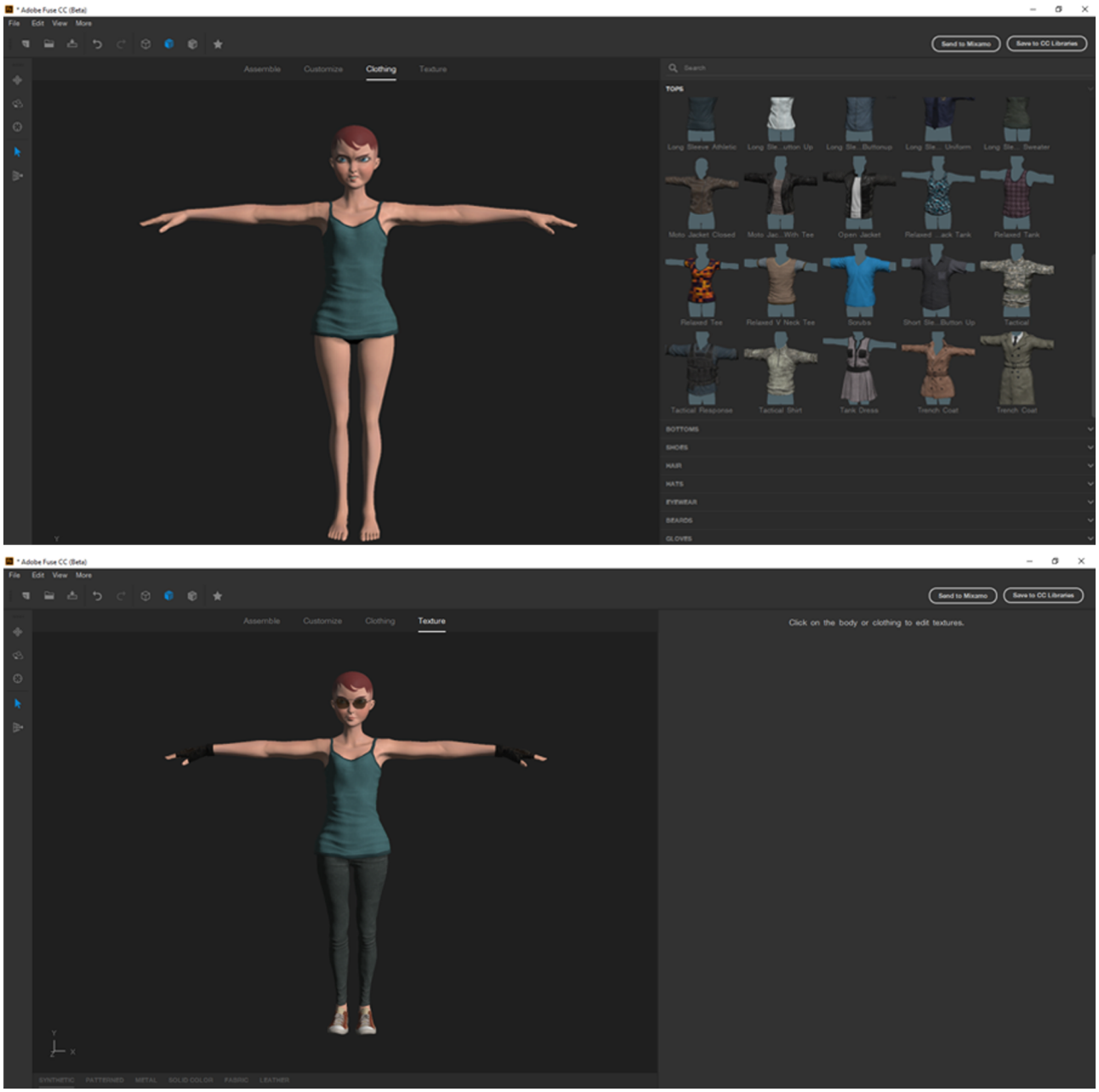
Si clicamos en cada zona del personaje, también podemos configurar i/o modelar estas partes con las opciones de trabajar con el ratón o con el menú deslizante de la parte de la derecha del programa.

Imagen 8. Cambios en zonas



Paso seguido, podemos vestir el personaje y trabajar algunas texturas del mismo, como piel, cabello, etc. De igual manera, vamos a trabajar por partes separadas del cuerpo hasta llegar en el resultado final.

Imagen 9. Resultado final



Cuando el modelo este completamente acabado, tenemos dos opciones de guardarlo, bien en la biblioteca de Adobe o bien enviar a Mixamo. Pues vamos a centramos en la segunda opción.

Enlace

Exportación y modelado en Mixamo. Breve apartado con explicaciones básicas de cómo trabajar modelos de personajes:

https://docs.unity3d.com/Manual/UsingHumanoidChars.html

2.2. Animando y preparando en Mixamo

En Fuse has elaborado el modelado de tu personaje con un amplio abanico de posibilidad de personalización. Pero este personaje carece de huesos con lo cual, animar-lo sería casi imposible. Mixamo es el complemento ideal para poder trabajar el sistema de huesos del personaje recién creado.

Mixamo es una plataforma en línea que permite a los desarrolladores crear rigs de forma fácil e intuitiva.

https://www.mixamo.com/

Una de las características más sorprendentes de Mixamo es el Auto-Rigger. El programa lee el personaje y según sus características, crea de forma automática el sistema de huesos. Otra característica importante de Mixamo es la amplia biblioteca de animaciones que dispone, donde el usuario puede agregar la que sea más apta para su proyecto y luego, la puede personalizar de forma a detallar todavía más el tipo de animación asociada.

En los prójimos puntos vamos a trabajar con nuestro anterior modelo que hemos preparado en Fuse, ahora en Mixamo.

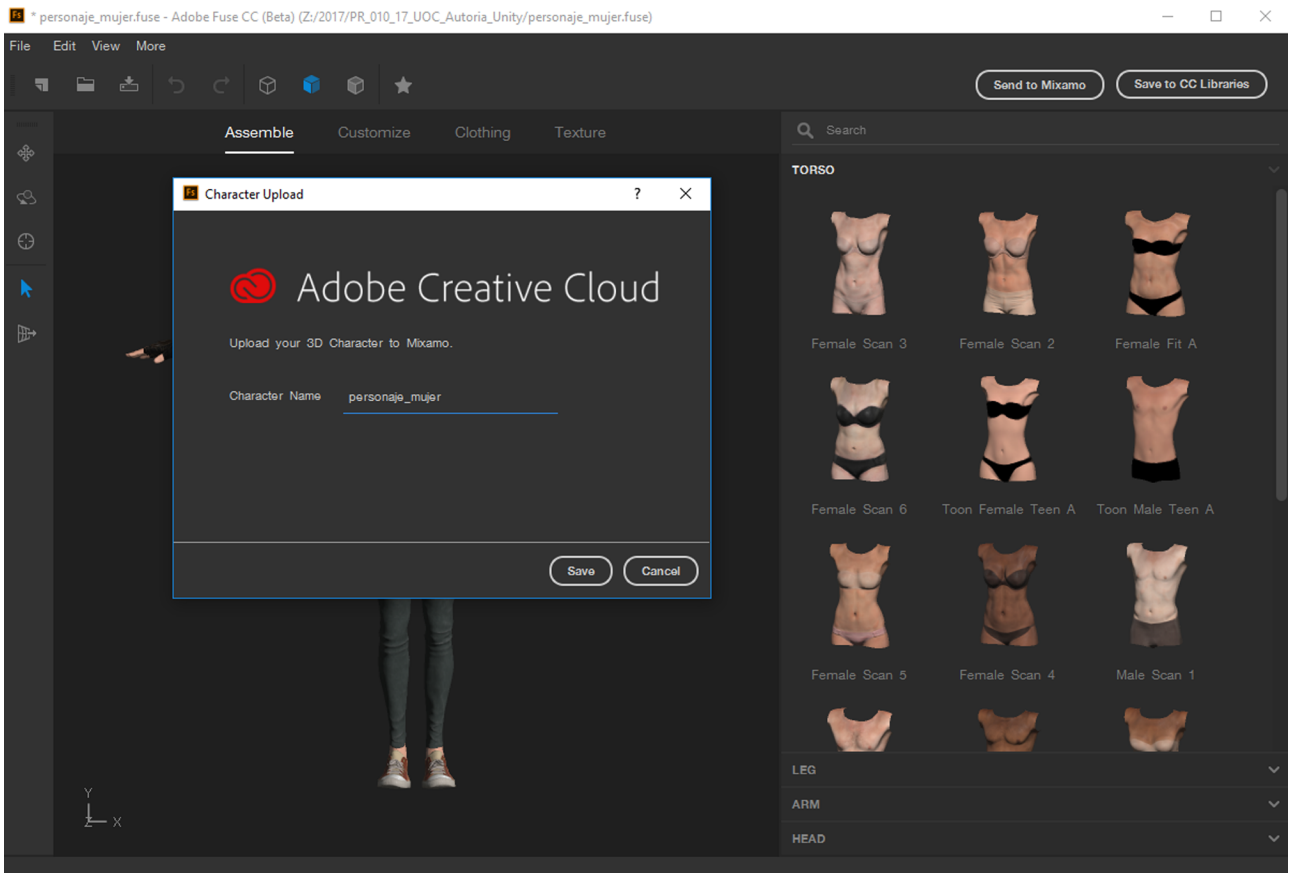
2.3. Exportación de Fuse a Mixamo

Desde la misma interface de Fuse, podemos exportar el modelo al entorno de Mixamo para trabajar su sistema de huesos y animación.

Localizado en el canto superior derecho de la pantalla, el botón *send to Mixamo* prepara el personaje para su exportación.

Si ya tienes una cuenta de ID de Adobe, la exportación se hará con este código, sino es recomendable la alta en Adobe ya que Mixamo es un complemento de la compañía Adobe y necesita la identificación del ID para realizar la exportación.

Imagen 10. Exportación a Mixamo desde Fuse



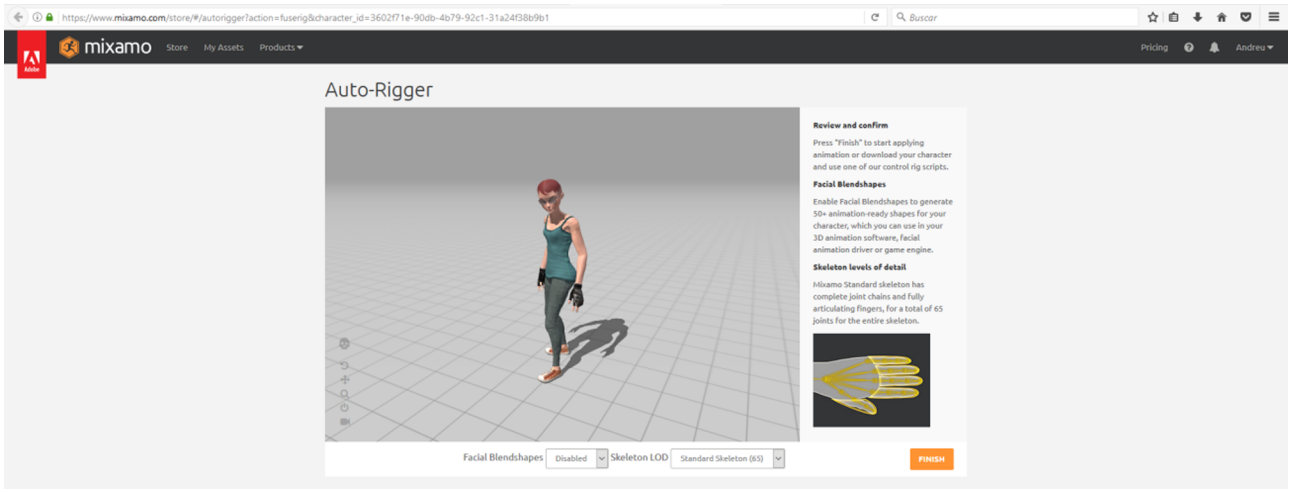
Mixamo suporta también otros formatos diferentes que los generados por Fuse:

Los personajes en 3D pueden encontrarse en formatos de archivo como **fbx**, **.bvh**, **.zip**, **.obj**.

2.4. Creando el sistema de huesos

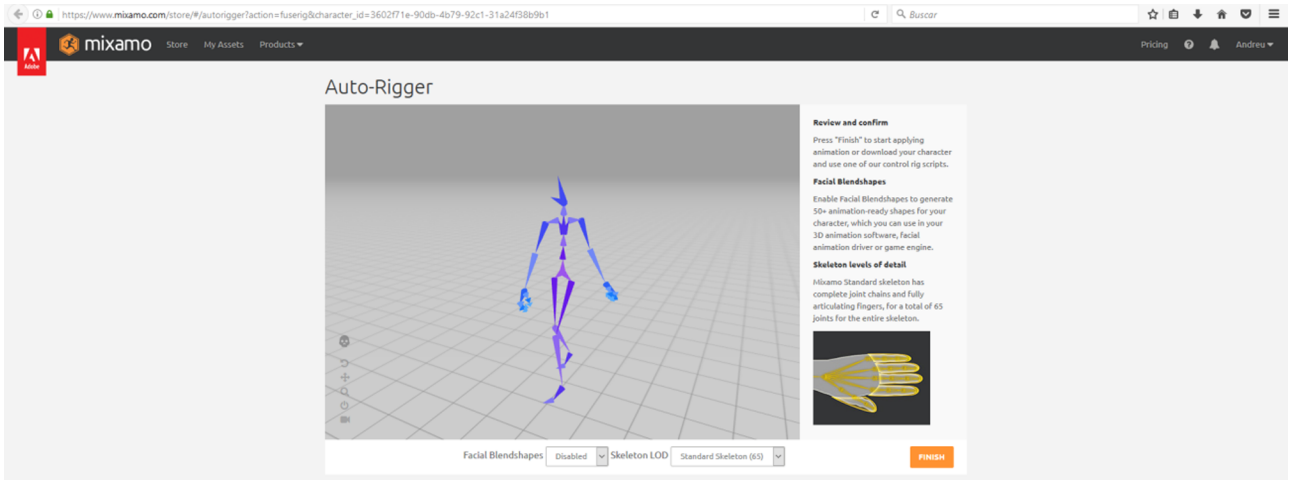
Una vez importado en Mixamo, lo primero que vemos es una ventana animada con nuestro personaje importado de Fuse en posición Idle. (posición estática, inicial)

Imagen 11. Entorno Mixamo inicial



En el menú lateral, tenemos varias opciones de control inicial de como visualizar el personaje en esta fase. Podemos mover, rotar, cambiar la posición de la camera, desplazar, hacer un acercamiento o inclusive, ver solo el sistema de huesos asignado.

Imagen 12. Visualización en modo huesos



Otro punto importante a destacar es la pequeña ventana de la derecha, donde vemos una mano con una sombra. En el menú inferior de la pantalla animada, tenemos dos cajas de opciones. Nos centremos en la segunda, denominada skeleton LOD.

Hay cuatro opciones disponibles:

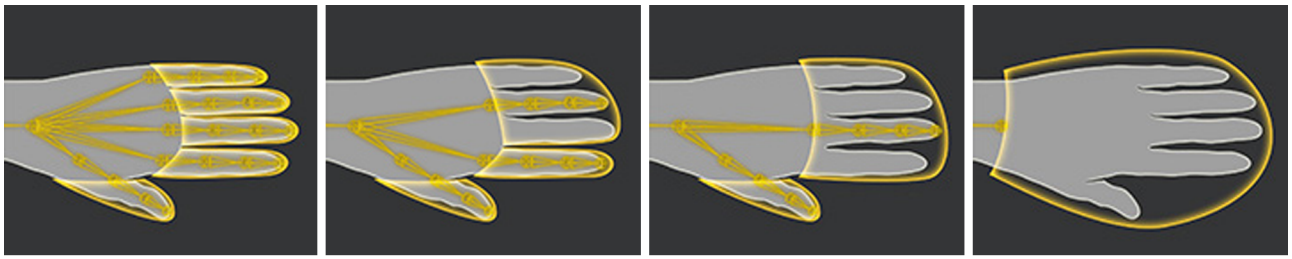
- Standart Skeleton (65)
- Chain Fingers (49)
- Chain Fingers (41)

- No fingers (25)

Lo que determinan estas opciones es el comportamiento de los huesos de las manos del personaje. Si tenemos un proyecto en lo cual necesitamos que el personaje recoja objetos o empuñe otros, necesitaremos un sistema de huesos más eficaz y por supuesto, mas articulaciones en cada mano. Por tanto, tendríamos que seleccionar un sistema de huesos *Standart Skeleton*, que asigna un valor alto a la cantidad de huesos de la mano.

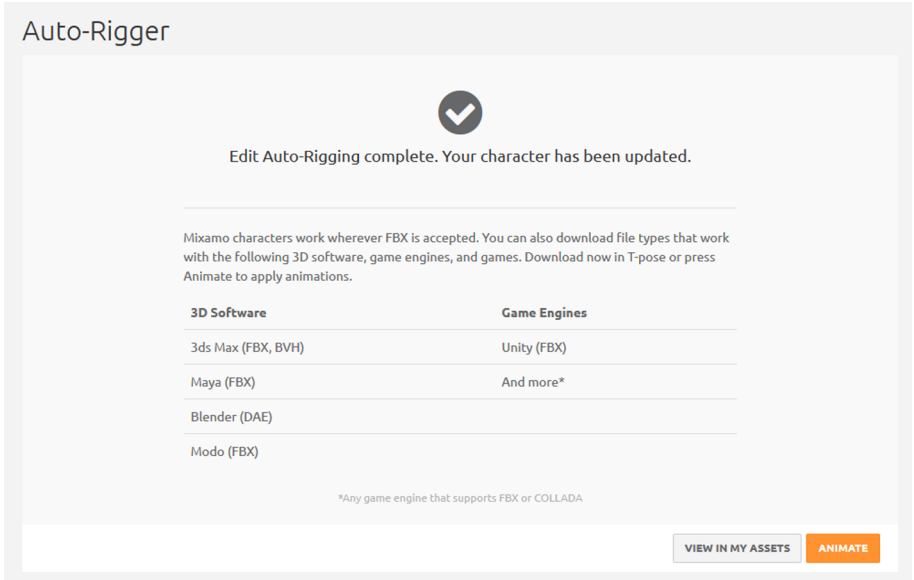
Una forma rápida de ver como altera el comportamiento de cada opción, es la ilustración asociada a cada opción.

Imagen 13. Huesos de la mano



Después de definir las características de las manos y facial del personaje 3D, ya podemos hacer el *upgrade* completo del sistema de huesos.

Imagen 14. Upgrade sistema de huesos



En este punto podemos seleccionar la opción de exportar el personaje en uno de los formatos estándar en la posición en *T* para poder animarlo en Unity o podemos seleccionar una de las diversas animaciones disponibles en la biblioteca de Mixamo.

En el caso de querer animar el personaje, podremos escoger el tipo de animación que queramos y en este punto, también podremos personalizar esta animación por medio de controles de la propia interface de Mixamo.

Para trabajar el sistema facial en Mixamo, hay disponible una serie de recursos que podrás crear de forma sencilla diversas expresiones.

https://www.mixamo.com/faceplus

Si el usuario lo prefiere, también puede utilizar scripts per realizar la animación en otros programas, como Maya o 3D Max.

https://www.mixamo.com/scripts

En esta web encontraras tres tutoriales explicativos referente a la exportación de los modelos para animarlos en los programas antes mencionados:

1) Maya Control Rig. Un script fácil de usar que agrega una plataforma de control a cualquier personaje *Auto-Rigged*, haciendo posible importar animaciones de Mixamo en ese equipo, editarlas y exportarlas. Ahora con un equipo facial para los personajes de Fuse.

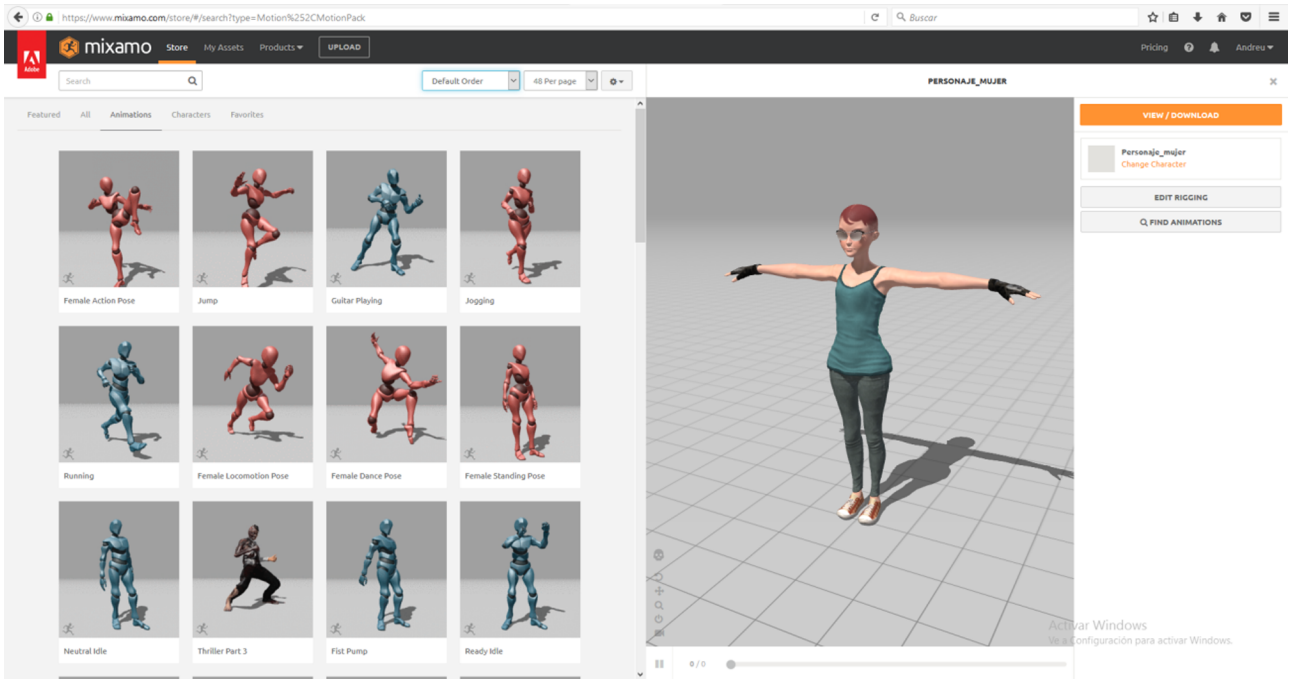
2) Auto-Biped. Script de un solo clic donde podrás convertir automáticamente cualquier personaje Auto-aparejado en el sistema de esqueleto Autodesk 3ds Max Biped.

3) Auto-CAT. Script de un solo clic donde podrás convertir automáticamente cualquier personaje Auto-aparejado en el sistema de esqueleto de Autodesk 3ds Max CAT.

Vamos a seguir animando nuestro personaje que hemos creado en Fuse y hemos exportado a Mixamo.

En la interface del programa, disponemos de la estructura de la biblioteca de Mixamo separada por diferentes filtros, para que el usuario encuentre la animación mas coherente para su proyecto en particular.

Imagen 15. Selección de la animación de biblioteca



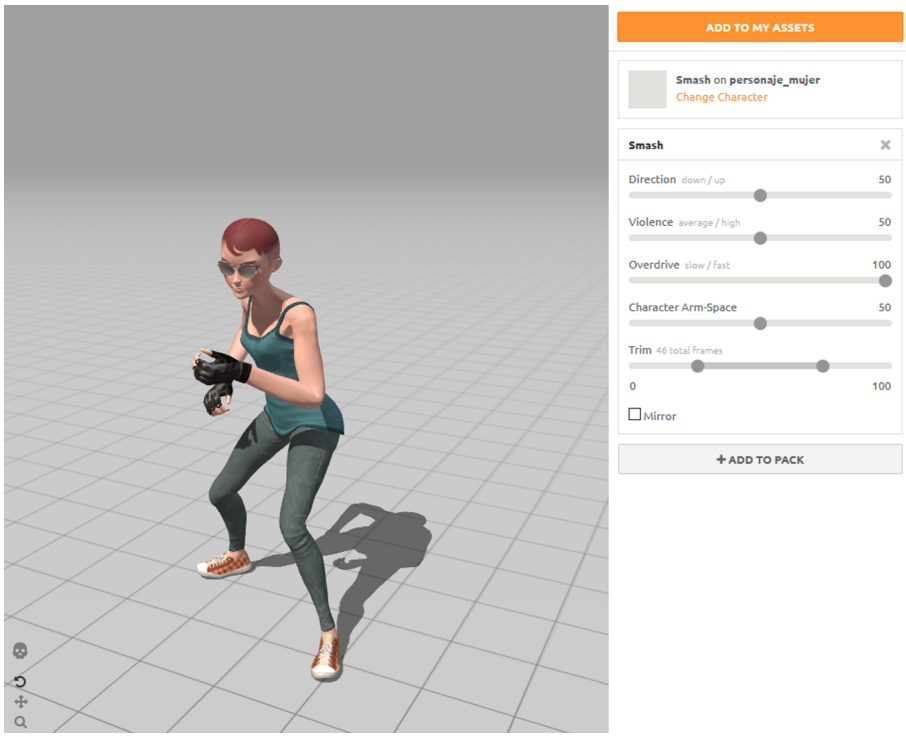
En este ejemplo, vamos a utilizar la animación Smash para dotar el personaje de movimientos de lucha. Basta seleccionar el tipo de animación y, de forma automática, se aplican los movimientos al personaje.

Imagen 16. Aplicación de una animación

Podemos personalizar la animación estándar utilizando los controles de desplazamiento en el menú lateral de la aplicación, de forma que podremos llegar al resultado que estamos buscando según lo que queremos hacer en el proyecto de Unity. Cada animación tiene unos valores pres estipulados de acuerdo con los movimientos determinados de cada animación. Por tanto, si la animación tiene movimientos de saltos, por ejemplo, tendremos controles específicos. En nuestro caso, disponemos de cinco opciones de configuración:

- Dirección del golpe
- Violencia del golpe
- Velocidad del golpe
- Distancia del golpe en relación a los brazos
- Recorte del golpe

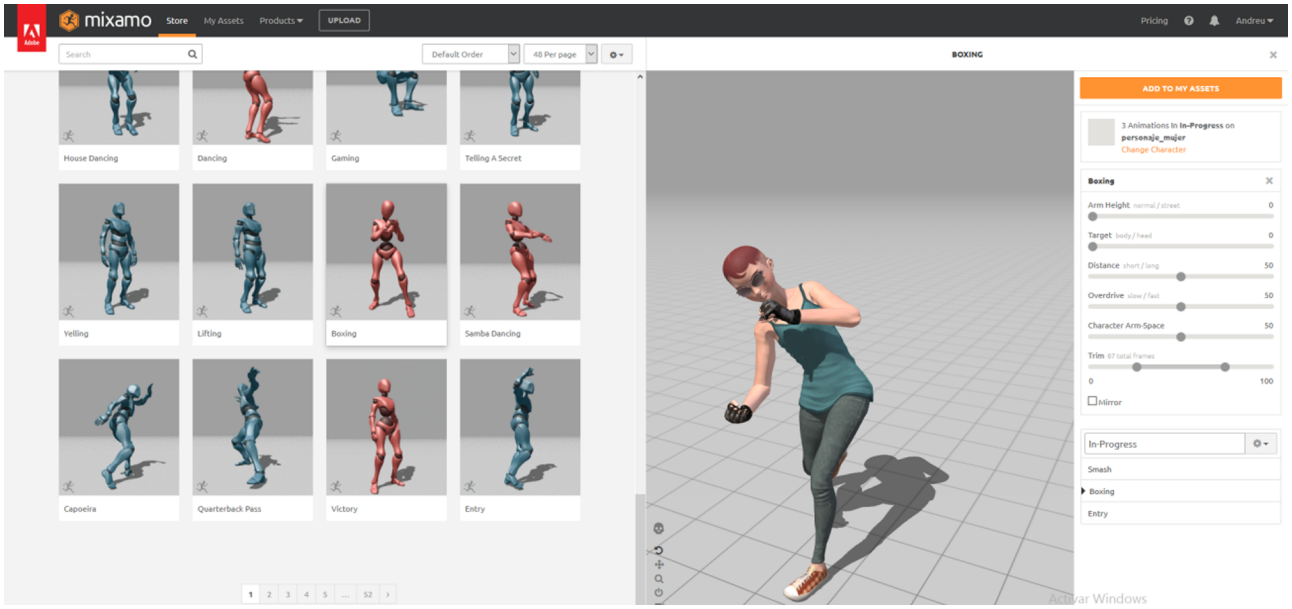
Imagen 17. Configuración de la animación



Otro aspecto interesante de la biblioteca de animaciones de Mixamo, es la posibilidad de exportar varias animaciones para un mismo personaje, que en el entorno de Unity aparecerán de forma separada para que puedas aplicarlas en diferentes momentos del juego. Este complemento posibilita que un único personaje tenga varias animaciones anidadas y que se exporten de forma conjunta. Para esto, en la parte inferior del menú lateral de cada animación seleccionada (lateral derecho) disponemos de la opción *+Add to pack*.

Podremos tener cuantas animaciones queramos y se apilaran en el historial del personaje.

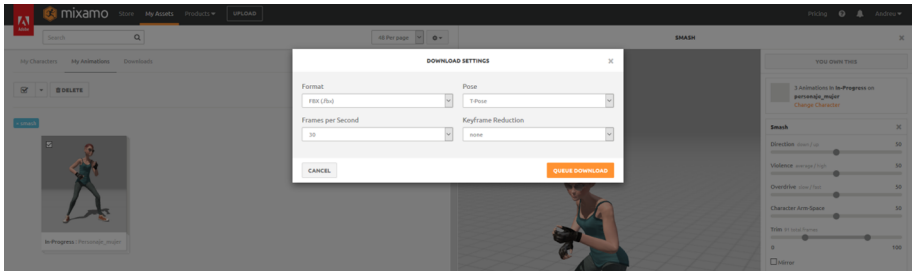
Imagen 18. Anidamiento de animaciones en un único personaje.



Finalmente podemos exportar todo el contenido de Mixamo para empezar a trabajar en el entorno de Unity.

Una vez más, podemos determinar algunos aspectos de configuración para la exportación.

Imagen 19. Configuración de la exportación



Llegados a este punto, podemos ya trabajar directamente en Unity para determinar otros aspectos de la animación del personaje.

2.5. Reducción de polígonos

En algunos casos, será necesario pensar en una reducción de los polígonos generados en cada modelo, ya que el peso excesivo puede ser un gran impedimento para el motor grafico tanto de Unity como del producto final.

En el manual de Unity, encontraremos un apartado específico de rendimiento de gráficos, que nos ayudará a determinar la mejor forma de trabajar los diferentes elementos del juego de manera que, sin perder demasiada calidad final, puedan tener un peso ideal para un buen desarrollo del mismo.

Algunos de los puntos que están especificados en el manual son:

- Ubique los gráficos que tienen alto impacto
- Optimización CPU
- GPU: Optimizando la geometría de un modelo
- *Lighting performance* (rendimiento de iluminación)
- GPU: Compresión de Textura y mipmaps
- Distancias LOD y cull per-layer
- Sombras en tiempo real
- GPU: Tips para escribir shaders de alto rendimiento

Enlace para la gestión de gráficos

Mejorar el rendimiento de los juegos:
https://docs.unity3d.com/es/current/Manual/Optimizing-Graphics Performance.html

Una simple lista de verificación para hacer su juego más rápido

Mantenga la cantidad de vértices debajo de 200k y 3M por frame cuando construya para PC (dependiendo en el GPU objetivo).

Si usted está utilizando uno de los shaders integrados, escoja uno de las categorías **Mobile** o **Unlit**. Estas funcionan en plataformas no necesariamente de móviles, pero son versiones simplificadas y aproximadas de más shaders complejos.

- Mantenga el número de materiales diferentes por escena bajo, y comparta tantos materiales que pueda entre objetos como sea posible.
- Configure la propiedad Static en un objeto que no se mueva para permitir optimizaciones internas como *static batching*.
- Utilice una única luz pixel light (preferiblemente directional) afectando su geometría, en cambio de varias.
- Use la iluminación (*static*) en vez de usar una iluminación dinámica.
- Utilice formatos de textura comprimidos cuando pueda, y utilice texturas 16-bit en vez de texturas de 32-bit.
- Evite utilizar *fog* (niebla) cuando sea posible.

- Utilice *Occlusion Culling* para reducir la cantidad de geometría que está visible y *drawcalls* en casos de escenas estáticas complejas con mucho occlusion. Diseñe sus niveles con *occlusion culling* en mente.

- Utilice *skyboxes* para distancias de geometría "falsas".

- Utilice sombreadores de píxel o combinadores de texturas para mezclar varias texturas en vez de un enfoque *multi-pass*.

- Utilice variables de precisión *half* cuando sea posible.

- Minimice el uso de operaciones matemáticas complejas como *pow*, *sin* y *cos* en *pixel shaders*.

- Escoja *fewer textures per fragment* (menos texturas por fragmento).

En el momento de modelar, también hay que tener algunos puntos en mente para minimizar el impacto del peso de los personajes y elementos en la escena.

- Utilización un solo Renderizador Skinned Mesh

- Utilización de tan pocos materiales como sea posible

- Utilización del menor número de huesos posible.

- Cantidad de polígonos

- Cinemáticas hacia adelante y hacia atrás separadas

Para más información, consulte el material en línea.

Enlace sobre modelado de personajes y rendimiento

Mejorar el rendimiento de los juegos con un modelado mas austero.
https://docs.unity3d.com/es/current/Manual/ModelingOptimized Characters.html

Los modelos exportados en formato *.fbx tienen la facilidad de contar con un reductor de polígonos online. Se trata de Decimator, una herramienta de Mixamo que reduce de forma considerable la cantidad de polígonos de los modelos 3D importados.

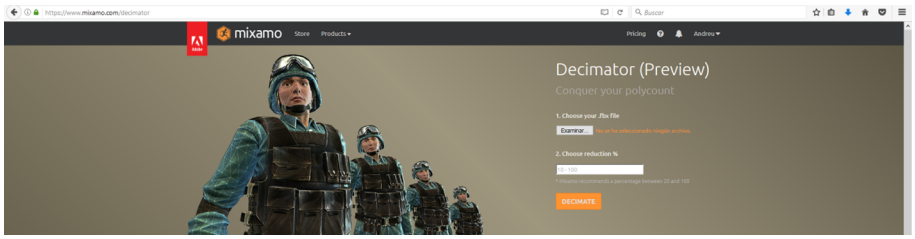
https://www.mixamo.com/decimator

Imagen 20. Diferentes tasas de reducción de polígonos

100% = 5330 polígonos	75% = 3997 polígonos	50% = 2665 polígonos	25% = 1337 polígonos

Basta hacer el *login* con la cuenta ID de Adobe, subir el archivo fbx y seleccionar la cantidad de reducción poligonal.

Imagen 21. Web principal de Decimator



Veamos cómo funciona. El *polycount* de la malla poligonal se reducirá de la manera más óptima posible, en base a su porcentaje de reducción deseado. Tenga en cuenta que esta solución es sólo para la reducción de policromo – Los huesos y texturas todavía no se reducen. Para mejores resultados, se recomienda porcentajes de reducción por encima del 20%.

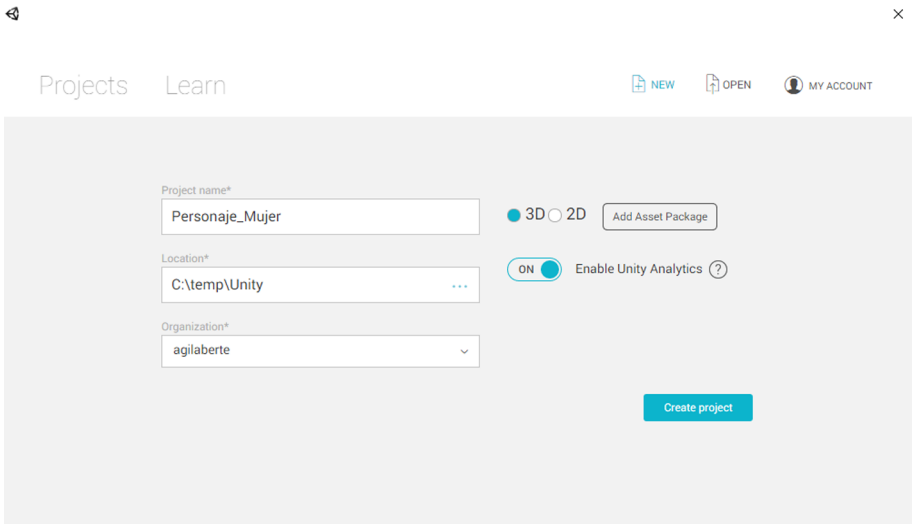
2.6. Importando y animando elementos de Mixamo en Unity

En un nuevo proyecto de Unity, lo primero que haremos será cargar el personaje modelado de Mixamo. En el punto anterior hemos aprendido como aplicar animaciones de la biblioteca de Mixamo para el personaje. Para este apartado vamos a trabajar en un modelo que no se ha aplicado ninguna animación de Mixamo y esto se debe a que vamos a explorar la tienda de Unity para aplicar otras animaciones predefinidas. De esta forma, el usuario podrá conocer diferentes vías para poder animar el modelo 3D.

2.6.1. Pasos previos

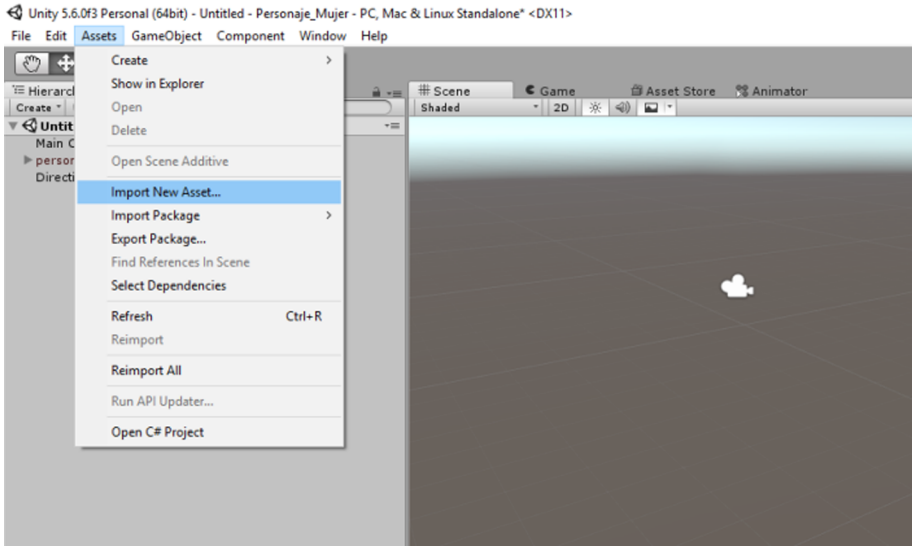
Por ahora no vamos a cargar ningún *asset* y solo vamos a definir el proyecto como 3D.

Imagen 22. Nuevo proyecto en Unity



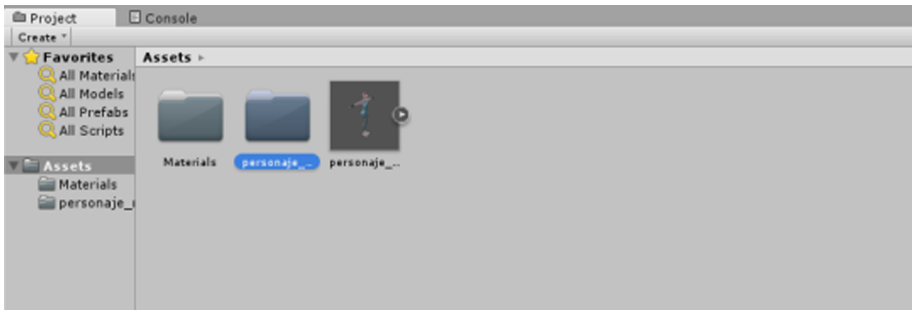
Importamos el *asset* (archivo fbx) de Mixamo.

Imagen 23. Importación del archivo fbx



En la parte inferior del programa, podemos ver que en la carpeta *assets*, tenemos el modelo, el mapa de materiales y texturas del personaje.

Imagen 24. Carpeta de *assets* de Unity.



Enlace sobre importación de *assets*

Conceptos de importación archivos assets en entorno Unity:

https://docs.unity3d.com/es/current/Manual/ImportingAssets.html

Si tienes algún modelo 3D elaborado con otro programa de modelado, como por ejemplo Blender, 3D Max, Cinema 4D, SketchUp, etc., Unity permite importar estos archivos de forma nativa. En estos casos, es importante identificar algunos parámetros de importación como las mallas, los polígonos o la optimización de la importación. El manual de Unity dispone de una serie de apartados específicos para estas configuraciones.

Enlaces sobre importación de modelos

Conceptos de importación modelos originarios de otros programas de modelado 3D.

https://docs.unity3d.com/es/current/Manual/HOWTO-importObject.html

https://docs.unity3d.com/es/current/Manual/ModelingOptimizedCharacters.html

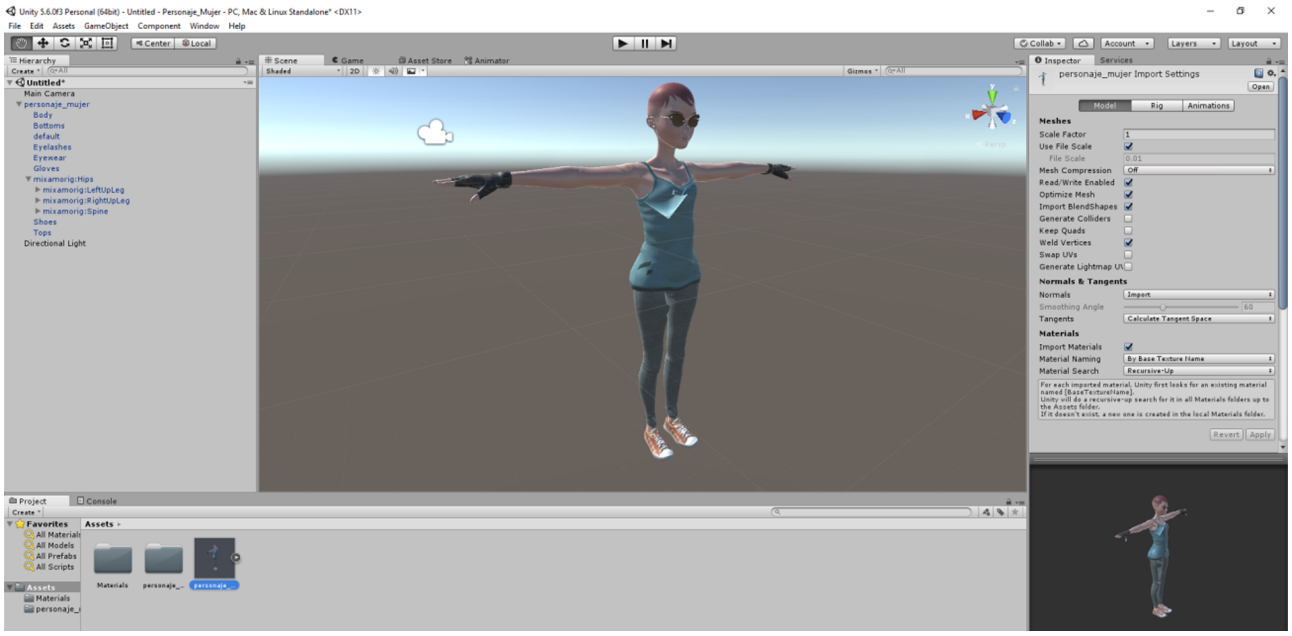
https://docs.unity3d.com/es/current/Manual/class-Mesh.html

Lo primero que haremos será comprobar que el sistema de huesos que hemos importado de Mixamo esta correcto y que toda la estructura del modelo responderá a las animaciones que le asignemos. En este sentido, vamos a *rigear* el modelo insertado en la escena.

Seleccionamos el modelo en la pestaña de *assets* y, en el menú lateral de Inspector, seleccionamos la opción de *rig* asignando como tipo de animación, un modelo humanoide.

Luego seleccionaremos el botón *configure* para acceder a la comprobación del sistema de huesos.

Imagen 25. Comprobación sistema de huesos.



Este es un paso importante para corregir posibles problemas que el personaje pueda tener en los diferentes puntos de control del esqueleto.

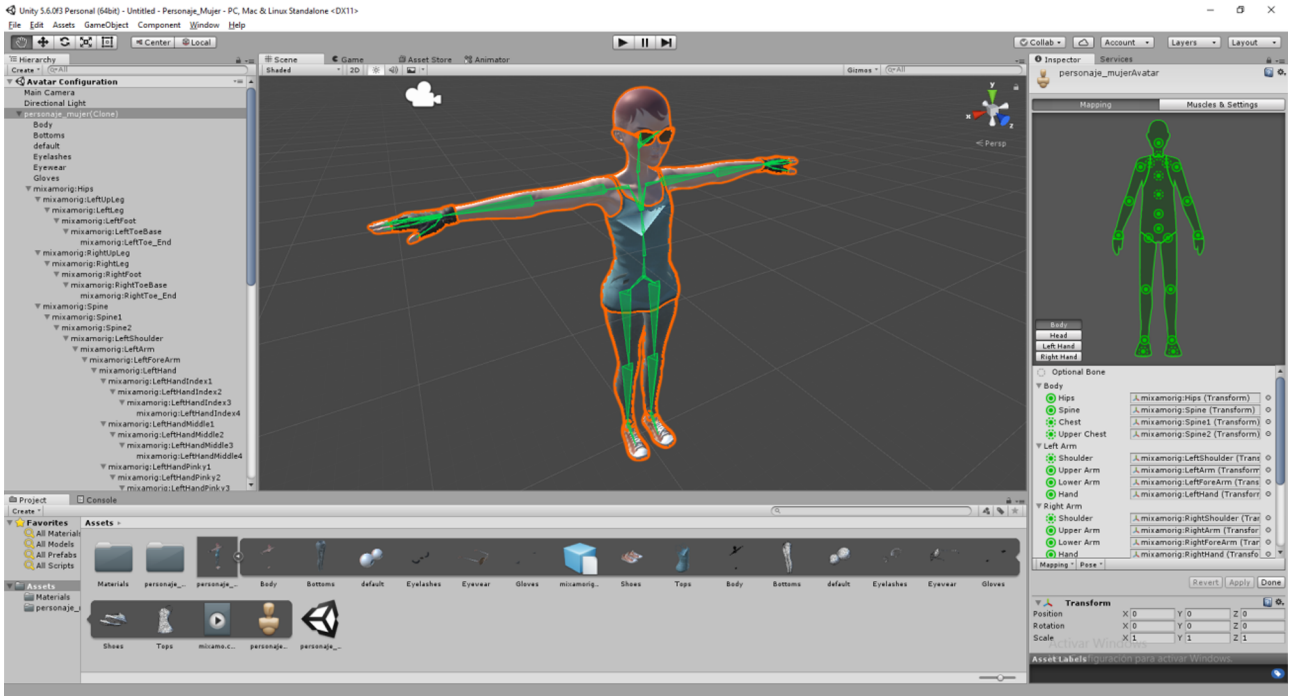
La forma de verificar si todos los huesos están bien definidos es por medio de la visualización de los *checks* de cada punto del modelo, que están identificados en el menú lateral derecho.

Enlace sobre configuración de huesos y músculos

Permite verificar y ajustar valores de configuración de huesos y músculos del personaje:

https://docs.unity3d.com/es/current/Manual/MuscleDefinitions.html

Imagen 26. Verificación sistema de huesos

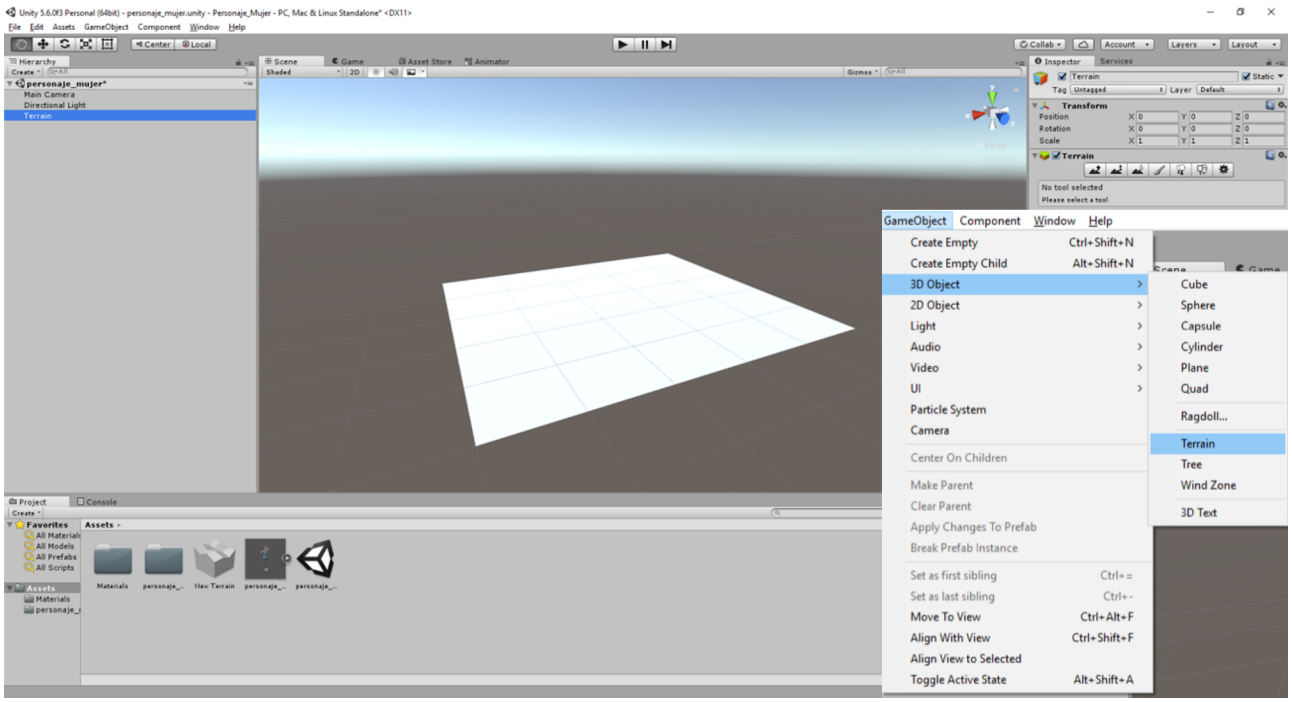


2.7. Terrenos

Antes de colocar el modelo en escena y para que tengamos una superficie de apoyo donde el personaje pueda animarse, insertamos un terreno.

Unity permite configurar el terreno con diferentes herramientas. Todas estas son configuraciones propias del programa, donde el usuario ira construyendo su superficie según las necesidades del proyecto en que está trabajando.

Imagen 27. Herramienta terreno de Unity



> «El sistema del Terrain de Unity permite agregar vastos paisajes a los juegos. En el tiempo de ejecución, la renderización de terreno es altamente optimizada para una eficiencia de renderización mientras que en el editor, una selección de herramientas está disponible para crear terrenos de manera fácil y rápida. Esta sección explica las varias opciones disponibles para terrenos y cómo hacer uso de estos.»

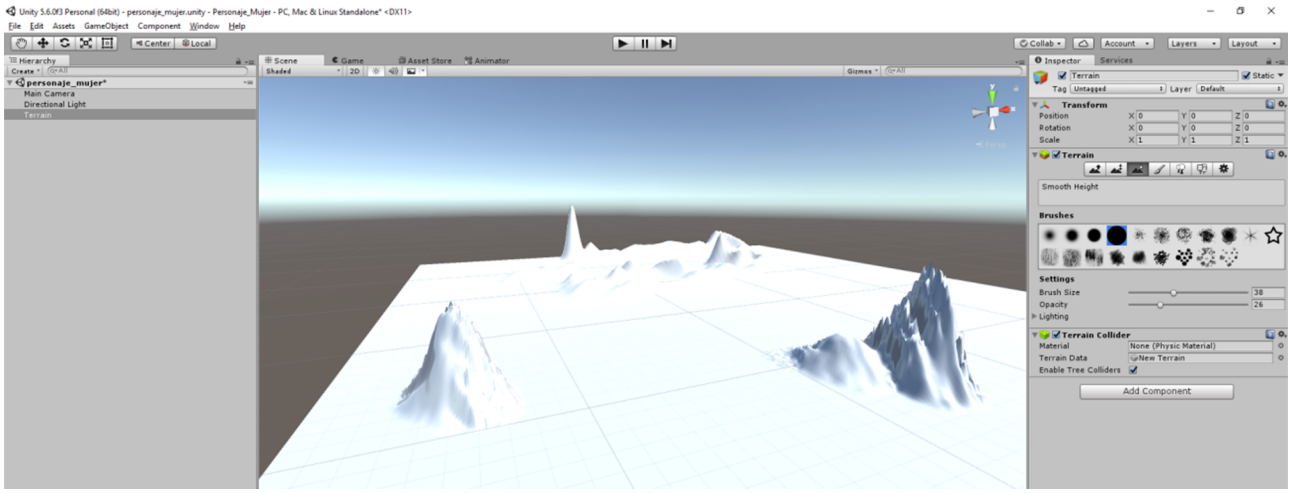
Como podemos observar, en el manual de Unity encontraremos mucha información relativa a la construcción de terrenos.

Enlace sobre cómo construir terrenos en Unity

Construcción y configuración de los terrenos en Unity:
https://docs.unity3d.com/es/current/Manual/script-Terrain.html

La altura y las depresiones, son formas naturales que tienen que estar presentes en un tipo de terreno natural. Las herramientas internas de configuración de Unity, posibilitan la construcción de estas y otras formas por medio de un sistema de selección y arrastre en la ventana de la escena para tal de transformar el terreno.

Imagen 28. Construcción del terreno



Otro punto importante cuando trabajamos con terrenos, es la posibilidad de insertar texturas, de forma que la superficie adquiera un manto que simula una determinada característica estética del terreno.

> «Usted puede agregar imágenes de textura a la superficie de un terreno para crear coloración y un detalle fino. Ya que los terrenos son objetos tan grande, es una práctica estándar utilizar una textura que se repite sin problemas y agruparla encima de la superficie (generalmente la repetición no es perceptible desde el punto de vista de un personaje cerca del suelo). Una textura servirá como la imagen de "fondo" sobre el paisaje pero usted también puede pintar áreas de diferentes texturas para simular diferentes superficies de suelo tales como pasto, desierto y nieve. Las texturas pintadas pueden ser aplicadas con una transparencia variable para que usted tenga una transición gradual entre un paisaje de césped y una playa arenosa, por ejemplo.»

Enlace sobre las texturas en terrenos de Unity

Aplicación de técnicas de texturas en terrenos generados en Unity:

https://docs.unity3d.com/es/current/Manual/terrain-Textures.html

Las texturas que se apliquen en el terreno se pueden producir con programas de edición de imágenes, como Photoshop o similar, o también podemos utilizar texturas de bibliotecas.

Como ya hemos visto anteriormente, Unity dispone de una tienda amplia de *assets* y dentro de la categoría de texturas, podremos encontrar las específicas para terrenos.

Utilizaremos el *asset* gratuito QS Materials Nature – Pack Vol.01.

Imagen 29.*Asset* de texturas para terrenos

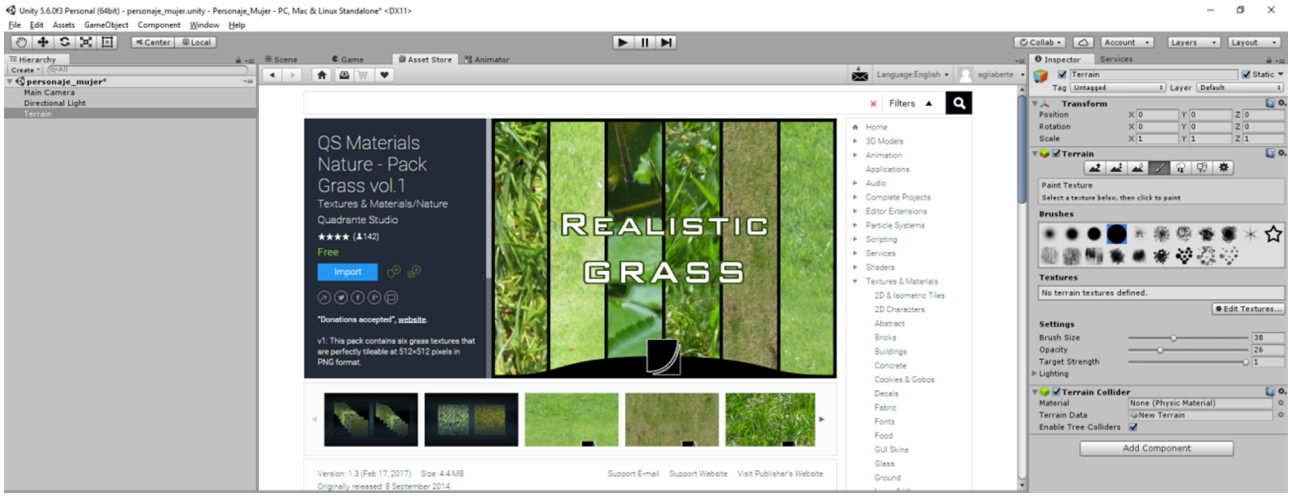
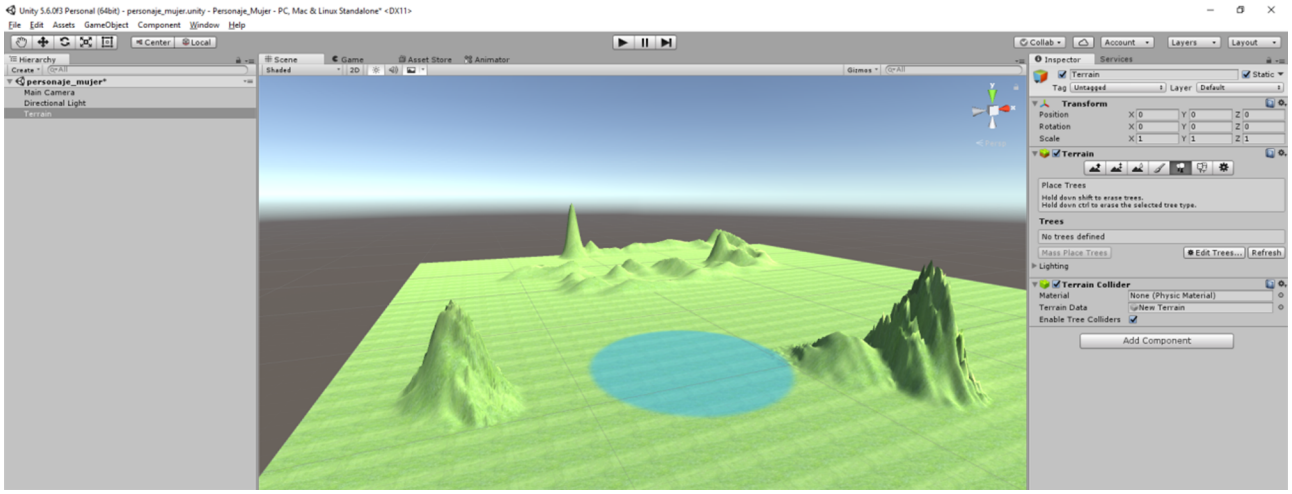


Imagen 30.*Asset* de texturas para terrenos



Una vez cargado el *asset*, tenemos a disposición varios elementos que podremos añadir en el terreno. Estos elementos quedan guardados en la carpeta específica de *assets*. En este caso, en la carpeta QS.

2.8. Creación de arboles

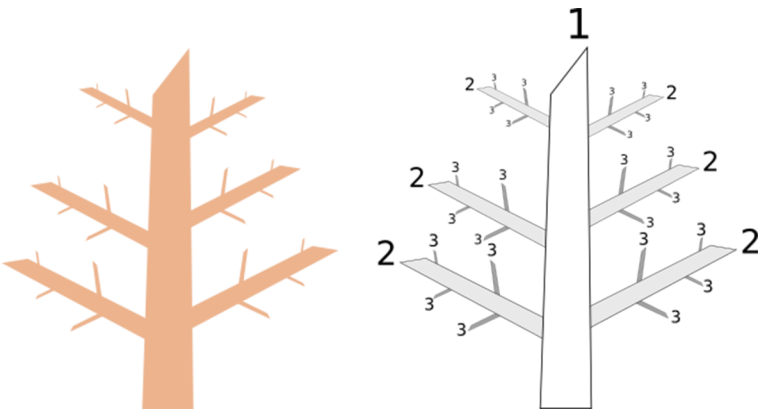
En ocasiones, el juego planteado necesita elementos arbóreos para un mejorado realismo de la escena o inclusive, la interacción del jugador con estos elementos. Unity dispone de un editor de origen para dibujar y crear diferentes interacciones con los objetos arbóreos.

Es una herramienta potente que pone a disposición del desarrollador varias opciones para crear un elemento des de cero y con los parámetros que cree oportuno según el tipo de juego que está dibujando.

Antes de entrar directamente en la ejecución del elemento, es importante identificar los principales elementos de un árbol para luego poder definir cada uno de los puntos que se quiera definir en el producto final.

El tronco (1) tiene ramas (2) que, a su vez, tiene sub-ramas (3); este proceso de ramificación continúa hasta que se produce la rama terminal. El tronco es considerado como el primer nivel del árbol, y luego las ramas que crecen directamente del tronco comprenden el segundo nivel. Cualquier rama que crezca del segundo nivel de ramas se junta y forman el tercer nivel, y así sucesivamente.

Imagen 31. Planificación de un árbol



Enlace sobre el concepto de árboles

Los conceptos básicos del elemento árbol se puede consultar en el siguiente enlace del material de Unity en línea:
https://docs.unity3d.com/es/current/Manual/tree-Structure.html

Enlace sobre la creación de un árbol

Forma básica de crear un objeto árbol estatico:
https://docs.unity3d.com/es/current/Manual/tree-FirstTree.html

Para crear un arbole en Unity, es necesario definir diferentes aspectos del mismo, como por ejemplo la cantidad de troncos, ramas y hojas, de acurdo con la jerarquía de la imagen anterior. Todo el proceso de creación de un arbole se puede consultar desde el manual de Unity:

Una vez el arbole esté definido, pasamos a configurar aspectos del mismo, lo que denominamos *Branch Group*.

Podemos modificar todos los elementos del árbol, como:

- Cantidad de hojas
- Distribución
- Geometría
- Forma
- Frondosidad

Enlaces sobre la modificación de un árbol

Configuración de diferentes elementos de un árbol:
https://docs.unity3d.com/es/current/Manual/tree-Branches.html
https://docs.unity3d.com/es/current/Manual/tree-Leaves.html

- Viento
- Crecimiento en el tiempo

Aparte de crear el arbol en sí, el usuario también puede crear diferentes interacciones de los elementos de este árbol con aspectos relacionados por ejemplo con el viento. En este caso hablaremos del método *Wind Zone*.

Enlace

Comportamiento de los elementos de un árbol con la fuerza del viento:
https://docs.unity3d.com/es/current/Manual/class-WindZone.html

Las Wind Zones son solamente usadas para animar hojas y ramas. Esto puede ayudar a sus escenas parecer más naturales y permite fuerzas (tales como explosiones) dentro del juego para verse como si estuvieran interactuando con los árboles.

Cuando "plantamos" arboles en nuestro juego, de forma automática estamos interaccionando con otro elemento de la escena, como es el caso del terreno. En este sentido, también podemos trabajar algunos aspectos relacionados con los arboles y el terreno de juego.

Uno de los principales problemas que el usuario tendrá cuando trabaje con gran cantidad de arboles es el rendimiento de los *renders* que se pueden generar. Para esto, Unity utiliza determinadas optimizaciones para equilibrar el *render* de salida con la cantidad de polígonos generados por una densa cantidad de elementos arbóreos.

Enlace sobre la optimización de grandes superficies

Reducción del tiempo de generación de fotogramas en zonas altamente pobladas de arboles:
https://docs.unity3d.com/es/current/Manual/terrain-Trees.html

2.9. Creación de césped

Aparte de los arboles comentados en los anteriores puntos, el diseño del juego puede necesitar otros tipos de elementos vegetales, como por ejemplo césped.

Para este caso en concreto, Unity trabaja con elementos 2D para pintar el césped sobre el terreno y para detalles más específicos del terreno, utiliza *meshes* estándar.

Enlace

Concepto de meshes estándar:
https://docs.unity3d.com/es/current/Manual/comp-MeshGroup.html

Para entender el funcionamiento de la creación de los elementos auxiliares en un terreno, Unity dispone de una serie de herramientas de control que son configurables según la necesidad del proyecto.

Enlace

Introducción de elementos auxiliares en el terreno, como césped o piedras:
https://docs.unity3d.com/es/current/Manual/terrain-Grass.html

2.10. Otros elementos naturales

Después de añadir terrenos, arboles y césped, tenemos la opción de trabajar la interacción de los medios naturales con estos últimos. Básicamente podemos configurar zonas de viento que interactúan con estos elementos.

"Usted puede crear el efecto del viento en su terreno agregando uno o más objetos con componentes Wind Zone. Los árboles dentro una zona de viento se doblarán de una manera animada y realista. El viento por sí mismo se moverá en pulsos para crear patrones naturales de movimiento entre los árboles."

Enlace

Introducción para la utilización de interacción de zonas de viento y partículas:

https://docs.unity3d.com/es/current/Manual/terrain-WindZones.html

https://docs.unity3d.com/es/current/Manual/class-ParticleSystem.html

Aparte de viento, en el manual de Unity también podemos consultar la generación de partículas para interactuar con elementos arbóreos.

2.11. El tratamiento de los gráficos en Unity

Este apartado se concibe como una guía de estudio de los tutoriales de Unity publicados en https://unity3d.com/learn/tutorials/s/graphics que constituyen una introducción a la forma en que el programa trabaja la iluminación y el renderizado.

A diferencia de otros productos audiovisuales, el vídeo, la fotografía o el clip de animación, en los videojuegos la iluminación recibe un tratamiento en el que la ejecución en tiempo real. Los cambios de iluminación dependen en parte de la interacción que pueda hacer el usuario durante la ejecución de un juego.

En el videojuego la iluminación responde a técnicas y modelos matemáticos que simulan el comportamiento de la luz, un comportamiento que varía en función de diversos parámetros, muchos de los cuáles responde a la interacción del usuario. Ejecutar todos los cálculos necesarios en tiempo real resulta excesivamente caro en términos de recursos computacionales. Exige excesivas prestaciones de los dispositivos. Para evitar este inconveniente se dividen las tareas, parte de los cálculos necesarios para crear una iluminación se llevan a cabo con antelación a la ejecución de un juego y parte de los cálculos se llevan a cabo en tiempo real.

> «Global illumination, or 'GI', is a term used to describe a range of techniques and mathematical models which attempt to simulate the complex behaviour of light as it bounces and interacts with the world. Simulating global illumination accurately is challenging and can be computationally expensive. Because of this, games use a range of approaches to handle these calculations beforehand, rather than during gameplay.»

Enlace

Descripción y configuraciones luces y renders.

https://unity3d.com/learn/tutorials/topics/graphics/introduction-lighting-and-rendering?playlist=17102

La combinación de técnicas en tiempo real y cálculos precomputacionales que se ponen en juego permiten la creación de escenas con una iluminación inmersiva y con unos tintes de realismo que evolucionan rápidamente. La verisimilitud de los videojuegos cada día es mayor.

Algunos tipos de luz, las luces direccionales, spot y puntuales, por ejemplo, acostumbran a ejecutarse en tiempo real. Permiten la Iluminación directa de una escena durante el juego. Se trata de iluminaciones duras que provocan sombras densas en las que apenas existen transiciones. Se trata de tipos de iluminación que resultan básicos en escenas en las que existen personajes en movimiento. Constituyen los procedimientos de iluminación en tiempo real, *Realtime Lighting*.

> «By default, lights in Unity - directional, spot and point, are realtime. This means that they contribute direct light to the scene and update every frame. As lights and GameObjects are moved within the scene, lighting will be updated immediately. This can be observed in both the scene and game views.»

Para iluminaciones más realistas se hace necesario la utilización de iluminaciones que hayan sigo calculadas con anterioridad, que hayan sido precompuestas o "cocinadas" previamente: *Baked GI Lighting*. Estas composiciones previas que muestran los efectos de la iluminación sobre los objetos estáticos se calculan con anterioridad a la ejecución y pueden combinar tipos de iluminación directa e indirecta.

> «These 'lightmaps' can include both the direct light which strikes a surface and also the 'indirect' light that bounces from other objects or surfaces within the scene. This lighting texture can be used together with surface information like color (albedo) and relief (normals) by the 'Shader' associated with an object's material.
>
> With baked lighting, these light textures (lightmaps) cannot change during gameplay and so are referred to as 'static'. Realtime lights can be overlaid and used additively on top of a lightmapped scene but cannot interactively change the lightmaps themselves.»

Enlace

Descripción de cómo aplicar las luces a una escena.

https://unity3d.com/learn/tutorials/topics/graphics/choosing-lighting-technique?playlist=17102

Los resultados de los cálculos se escriben sobre las texturas que se aplican a la escena con lo que se simulan efectos realistas de iluminación. Estas texturas no cambian durante la ejecución del juego. Se conocen como estáticas y no permiten interacción.

Un tercer tipo de técnica de iluminación lo constituyen la luz en tiempo real que ha sido calculada previamente y que permiten una actualización rápida durante el juego o durante la interacción. El cubo que se muestra en el manual visualizando como varia l iluminación en función del paso del tiempo o de forma asociada a las horas del día constituye un ejemplo de este tipo de iluminación.

> «Whilst traditional, static lightmaps are unable to react to changes in lighting conditions within the scene, Precomputed Realtime GI does offer us a technique for updating complex scene lighting interactively.
>
> With this approach it is possible to create lit environments featuring rich global illumination with bounced light which responds, in realtime, to lighting changes. A good example of this would be a time of day system - where the position and color of the light source changes over time. With traditional baked lighting, this is not possible.»

La precomputación permite llevar a cabo con anterioridad el cálculo de comportamientos complejos de la luz que tienen lugar durante la ejecución. Los cálculos se llevan a cabo en momentos en los que se puede disponen de mayores recursos en los dispositivos. Es un modo de lograr una reproducción más ágil y realista con una importante economía de

El proceso para precomputerizar se describe en el siguiente apartado.

«In Unity, precomputed lighting is calculated in the background - either as an automatic process, or it is initiated manually. In either case, it is possible to continue working in the editor while these processes run behind-the-scenes.

When the precompute process is running, a blue progress bar will appear in the bottom right of the Editor. There are different stages which need to be completed depending on whether Baked GI or Precomputed Realtime GI is enabled. Information on the current process is shown on-top of the progress bar.»

Enlace

Descripción de las técnicas de pre computerizar luces en una escena: https://unity3d.com/learn/tutorials/topics/graphics/precompute-process?playlist=17102

Por lo que respecta a las técnicas de renderizado, *rendering*, se distinguen básicamente el *Forward Rendering* y el *Deferred Rendering*.

En el primer tipo sobre cada objeto se llevan a cabo múltiples pasos de renderizado de modo que en cada uno de ellos se van aplicando los diversos tipos de iluminación que le pueden afectar. Se trata del proceso estándar de renderizado y el que utilizan la mayor parte de los dispositivos.

«The advantages of this approach is that it can be very fast - meaning hardware requirements are lower than alternatives. Additionally, Forward Rendering offers us a wide range of custom 'shading models' and can handle transparency quickly. It also allows for the use of hardware techniques like 'multi-sample anti-aliasing' (MSAA) which are not available in other alternatives, such as Deferred Rendering which can have a great impact on image quality.»

En el segundo caso, el *Deferred Rendering* la aplicación de las sombras y la mezcla de los efectos de luz sobre las texturas se lleva a cabo después que se haya construido un primer renderizado de los objetos de una escena.

«In 'Deferred' rendering, on the other hand, we defer the shading and blending of light information until after a first pass over the screen where positions, normals, and materials for each surface are rendered to a 'geometry buffer' (G-buffer) as a series of screen-space textures. We then composite these results together with the lighting pass.

This approach has the principle advantage that the render cost of lighting is proportional to the number of pixels that the light illuminates, instead of the number of lights themselves. As a result you are no longer bound by the number of lights you wish to render on screen, and for some games this is a critical advantage.»

Enlace

Descripción de las técnicas de renderizado de las escenas.

https://unity3d.com/learn/tutorials/topics/graphics/choosing-rendering-path?playlist=17102

3. Audio

3.1. Introducción

El audio es un elemento que puede definir una situación, una escena o las características de un determinado personaje. En el mundo de los videojuegos, el audio es un componente más de la escena, participando de forma activa en cada momento de la partida, generando sensaciones, expectativas y conclusiones.

Sin la correcta gestión del audio en cada etapa del proyecto de Unity, el juego quedaría sin un personaje más i perdería el interés por parte de los jugadores.

Podríamos definir dos tipos de audio: los efectos y las bandas sonoras.

1) Los efectos: El audio es complejo, viaja en el tiempo, incide en la fuerza del argumento y es preciso cuando es necesario. Cuando disparas una bala, el audio no solamente tiene que trabajar sincronizado con el gatillo sino tiene que acompañar todo el momento de la escena, des de la salida del cañón hasta el impacto deseado, y eso quiere decir que durante su trayectoria, tendrá un sonido específico para cada momento y espacio, tendrá interacción con objetos y elementos y tendrá un punto final. Por veces, este sonido tiene un tiempo muy curto de duración, pero los factores antes descritos, funcionaran de igual forma.

2) Las bandas sonoras: Otra cosa son las bandas sonoras, que acompañan y crean una atmosfera específica para cada tipo de acción o juego. Las bandas, en la mayoría de veces, trabajan en un segundo plano, dejando los efectos que trabajen en un plano principal. Así mismo, la música o audio de fondo, participa de forma activa, creando una narrativa a la acción, con lo cual, tiene tanta o igual importancia en cuanto a la selección y colocación en escena, en relación a los efectos.

En la documentación de Unity, encontraremos un apartado específico para explicar los conceptos del audio, con información sobre el audio 3D espacial, la mezcla y *mastering*, la importación y los efectos pre definidos, entre otros apartados, es lo que vamos a detallar a continuación.

Enlace

Descripción general del audio en Unity: https://docs.unity3d.com/es/current/Manual/Audio.html

3.2. Elementos principales

Hemos hablado de las características del sonido en un video juego, pero antes de entrar en materia, tendremos que identificar los elementos esenciales que participan en un sistema de audio. Unity siempre destaca a las fuentes y a los oyentes: *Source* y *Listener*.

Los *listeners* son los principales modificadores del audio en una escena de juego, pueden alterar la forma de escuchar un sonido de acuerdo con las características de la escena o del momento. También pueden cambiar la estructura del mismo sonido, cambiando y aplicando efectos que distorsionan o cambian completamente el sonido en relación a su origen y final.

Hay aspectos que tener en cuenta cuando se trabaja con el audio. El audio tiene propiedades dentro de una escena que pueden ser diversas: El audio se puede mover, puede chocar con otros elementos, puede mezclarse, puede cambiar de volumen o cualquier interacción con el medio. El desarrollador de videojuegos tendrá que tener todos estos y otros aspectos en cuenta a la hora de plantear su proyecto sonoro en Unity.

Enlace

Principales elementos de audio en Unity:
https://docs.unity3d.com/es/current/Manual/AudioOverview.html

3.3. Importación de elementos sonoros

En apartados anteriores, hemos visto como funciona la tienda de Unity y que tipos de *assets* dispone. Pues los elementos de audio, están presentes y podemos insertar cualquier elemento en nuestro proyecto de Unity. También podemos crear nuestros propios efectos y sonidos, ya que el motor de Unity permite también importar archivos creados en otros programas de audio. Lo único que tenemos que tener en cuenta son

los archivos que suporta Unity:

Formato	Extensión
MPEG layer 3	.mp3
Ogg Vorbis	.ogg
Microsoft Wave	.wav
Audio Interchange File Format	.aiff / .aif
Ultimate Soundtracker module	.mod
Impulse Tracker module	.it
Scream Tracker module	.s3m
FastTracker 2 module	.xm

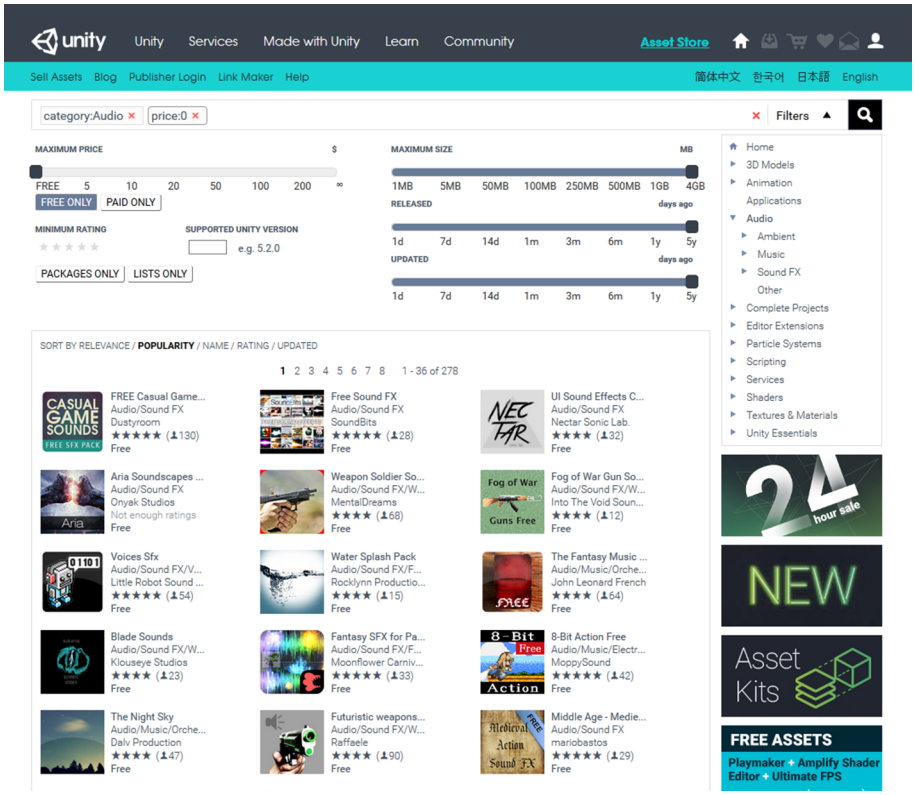
Enlace

Soporte de audio en Unity:
https://docs.unity3d.com/es/current/Manual/AudioFiles.html

Hay que tener presente que muchos efectos y bandas sonoras ya están disponibles de forma gratuita o de pagamento en la tienda de Unty, estos archivos se pueden importar de forma habitual en el proyecto de Unity.

Encontraremos efectos para armas, efectos para personajes, como pasos, caídas, voces o efectos tan diversos como explosiones o elementos de audio para objetos.

Imagen 32. Audios en la tienda de Unity



Una consideración muy importante a la hora de insertar efectos de audio en los juegos es la capacidad de interpretación de los mismos en cuanto a la velocidad de descarga en tiempo real. Unity propone los denominados Tracker Modules, que son archivos tratados en programas externos que disminuyen el peso en pro de una fluidez en el momento de acción del juego. Hay varios programas que trabajan este aspecto del audio, pero Unity recomienda dos:

- MilkyTracker para OSX
- OpenMPT para Windows

Para más información, consulta el blog:

https://blogs.unity3d.com/es/2010/06/29/mod-in-unity/

Enlaces

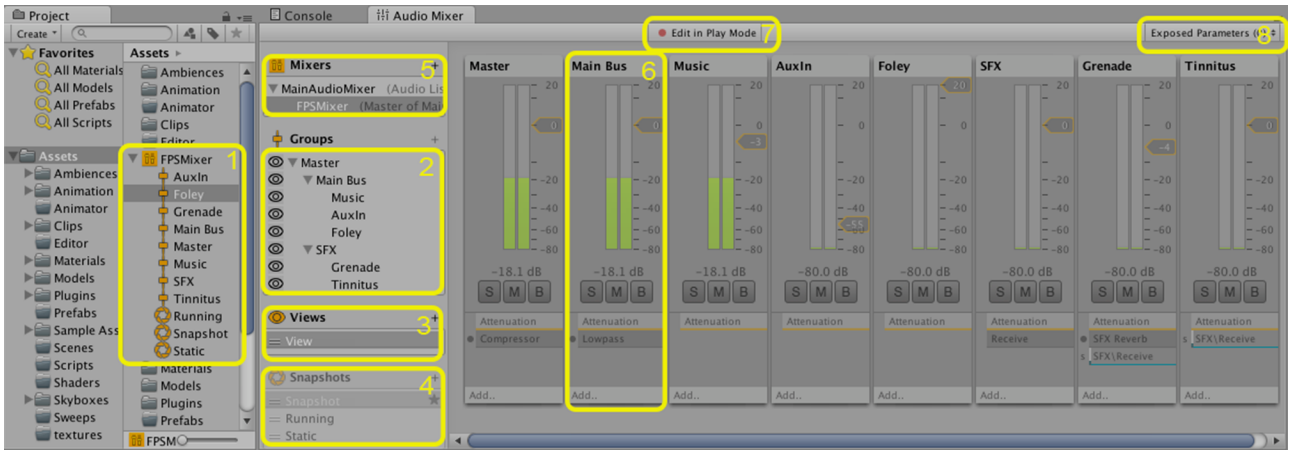
Forma de trabajar el audio para proyectos de Unity:https://docs.unity3d.com/es/current/Manual/TrackerModules.html

3.4. Trabajar audio desde Unity

También cabe la posibilidad de poder trabajar cualquier elementos de audio desde la interface de Unity, por medio de controles específicos que colocan el sonido en la escena de una forma muy concreta, siguiendo unos parámetros establecidos por el desarrollador del juego.

Las opciones de configuración del audio pasan por controlar el momento que se genera un determinado efecto o música, la velocidad, el tiempo de respuesta o la interacción del sonido con elementos del juego o con escenas definidas. Todos estos ajustes se pueden configurar des de la ventana del AudioMixer de Unity:

Imagen 33. Ventana de control de audio



1) El *Asset* - Contiene todos los AudioGroups y AudioSnapshots como sub-*assets*.

2) La vista de la Jerarquía - Contiene la jerarquía entera de mezcla de los AudioGroups dentro del AudioMixer.

3) La vista de los Mixers - Es una lista de los ajustes de visibilidad en caché del mezclador. Cada vista solo muestra un sub-conjunto de la jerarquía entera en la ventana principal del mezclador.

4) Snapshots - Es una lista de todos los AudioSnapshots dentro del AudioMixer *Asset*. Los Snapshots capturan el estado de todos los ajustes del parámetro dentro de un AudioMixer, y puede hacer una transición entre ellos en el tiempo de ejecución.

5) El Output del AudioMixer - El AudioMixer puede ser direccionado a AudioGroups de otros AudioMixers. Este campo de la propiedad le permite a uno definir el output Audiogroup para direccionar esta señal del AudioMixer.

6) Vista del AudioGroup Strip - Muestra una vista general de un AudioGroup, incluyendo los niveles actuales VU, ajustes de atenuación (volumen), Mute, Solo y ajustes del Effect Bypass y la lista de efectos DSP dentro del AudioGroup.

7) Edit In Play Mode - Permite editar el AudioMixer durante el modo de reproducción, o prevenir ediciones y permitirle al tiempo de ejecución del juego y controlar el estado del AudioMixer.

8) Exposed Parameters - Muestra una lista de los Exposed Parameters (parámetros expuestos) (cualquier parámetro dentro de un AudioMixer puede ser expuesto a script vía un nombre string) y los nombres con un tipo string correspondiente.

Todos dos detalles de configuración y ajustes del modo sonido se encuentran en el material de consulta de Unity.

Enlace

Configuraciones en el entorno propio de Unity con Audio Mixer:
https://docs.unity3d.com/es/current/Manual/AudioMixer.html

3.5. Filtros de audio

Aparte de las diferentes configuraciones que se pueden aplicar al sonido en cuanto a volumen, interacción, tiempo, etc., también podemos modificar un mismo sonido con diferentes filtros, de acuerdo con la escena que queramos destacar el audio, por tanto, no es lo mismo un efecto de pasos dentro de una habitación cerrada que los mismos pasos en un entorno exterior. El efecto sonoro es el mismo en los dos casos, pero lo que haremos será aplicar un filtro de acuerdo con lo que queramos transmitir y que se adecue a la escena actual.

Unity dispone de diferentes filtros para aplicar en los efectos:

- **Filtro de Audio Low Pass**. El sonido se propaga según la interacción del mismo con los elementos de la escena.
- **Filtro de Audio High Pass Filter** Evalúa las oscilaciones y resonancia de los sonidos.
- **Filtro de Audio Echo**. Propagación del sonido de acuerdo con el tipo elementos de la escena.
- **Filtro de Audio Distortion**. Simulación de distorciones sonoras.
- **Filtro de Audio Reverb**. Personaliza la reverberación del sonido.
- **Filtro de Audio Chorus**. Transforma el sonido en simulación tipo coro.

Enlace

Diferentes tipos de filtros para aplicar en los efectos sonoros: https://docs.unity3d.com/es/current/Manual/class-AudioEffect.html

4. Animación

4.1. Introducción

Unity trabaja con un sistema de animación complejo pero con un manejo substancialmente sencillo. Hay tres pilares básicos en que está basado el sistema de animación de Unity:

- Los clips animados
- Los avatares
- Los controles de animación

Pero, antes de entrar en la funcionalidad de cada elemento, sería interesante entender e interpretar la terminología característica de cada uno de estos elementos. Para esto, es conveniente leer el apartado de terminología de animación del manual de Unity.

Enlace

Breve terminología de los principales términos en materia de animación en Unity: https://docs.unity3d.com/es/current/Manual/AnimationGlossary.html

Por otro lado, también tenemos a disposición un apartado de preguntas y respuestas básicas en lo referente a la animación de Unity. Si te has planteado alguna de estas preguntas:

- ¿Qué es "Mecanim"?
- ¿Importa el orden de las capas que importo de los clips?
- ¿Cómo las animaciones que tiene Curvas se mezclan con aquellos que no?

Entonces encontraras las respuestas en el capítulo de Preguntas más Frecuentes de Unity.

Enlace

Breve apartado con las principales preguntas y respuestas respecto al método de animación de Unity.

https://docs.unity3d.com/es/current/Manual/MecanimFAQ.html

La animación de personajes y elementos en Unity es muy potente, el motor interno posibilita al desarrollador crear diferentes modos de animación, sean estáticas, como por ejemplo el viento que balancea las hojas de un árbol, o sean dinámicas, como por ejemplo un personaje que camina por una escena. En cualquier caso, la forma de animar en Unity responde a una interface que permite trabajar la animación de acuerdo con diferentes parámetros, como el

peso, una estructura en forma de jerarquía que sucede de acuerdo con variables del juego, transiciones, animaciones faciales, entre otros. Toda la animación de Unity está centrada en un sistema propio denominado Mecanim.

Enlace

Presentación del sistema de animación Mecanim en Unity y sus funcionalidades.
https://docs.unity3d.com/es/current/Manual/AnimationOverview.html

4.2. El flujo de trabajo

Cuando estamos animando personajes o elementos, son varias las situaciones que pueden desencadenar un movimiento. Como de una biblioteca de movimientos se tratara, estos personajes tienen varios tipos de movimientos definidos, por ejemplo coger un objeto, saltar, correr, caminar, girar la cabeza, etc. Estos movimientos son creados en programas externos, como Maya, Max, etc. y también de otras fuentes. Luego, estos movimientos son insertados en los personajes de Unity, es lo que llamamos Animation Clips. Esta técnica posibilita crear movimientos tipo en cualquier personaje que tengamos en Unity. En seguida, tratamos estos Animation Clips en estructuras tipo diagramas de flujo, lo que nos permite desencadenar las diferentes animaciones según parámetros definidos por el desarrollador del juego. Estos diagramas son los denominados *Animator Controller*.

Enlace

Como trabaja Unity el motor de animación:
https://docs.unity3d.com/es/current/Manual/AnimationOverview.html

4.3. Los Clips de animación

Como hemos comentado antes, los clips de animación pueden ser provenientes de programas externos, pero también de otras fuentes, como por ejemplo capturas de movimientos o librerías externas.

En cualquier caso, la flexibilidad de Unity reside en la forma de tratar estos movimientos en la escena. Podemos utilizar el tipo de movimiento importado tal y cual fue creado o podemos modificar en una línea de tiempo del personaje de forma que se adapte a nuestro proyecto. También tenemos la posibilidad de utilizar varios tipos de movimientos de diferentes fuentes en un mismo personaje. Tenemos la posibilidad de controlar todos los aspectos relacionados con el movimiento, como velocidad, posición, rotación o inclusive con sus propiedades, como color, forma, textura, etc.

4.3.1. Animaciones humanoides y no humanoides

La mayoría de juegos de la actualidad disponen de personajes con un sistema esquelético humanoide, quiero decir, cabeza, tronco, brazos y piernas. En este sentido, el motor de Mecanim trabaja de forma muy detallada cuando encuentra personajes con estas características. Los *clip animation* que hemos comentado en el anterior punto, tienen la particularidad de trabajar sobre este tipo de esqueleto, facilitando la importación de los movimientos en cualquier personaje que tenga una configuración similar. Cuando un personaje importa

Enlace

Crear y configurar sistema de animación humanoide
https://docs.unity3d.com/es/current/Manual/AvatarCreationand Setup.html

estos movimientos, Unity dispone de un configurador de huesos que mapea la estructura ósea del mismo para asegurar que los movimientos serán adecuados y precisos. En Unity, denominamos la creación del **avatar**.

Puede suceder que las animaciones que tengamos en el juego sean del tipo no-humanoide, en este caso, Unity también dispone de herramientas más especificas, no tan potentes como el caso que fueran del tipo humanoide, pero que presentan funciones muy interesantes a la hora de poder dotar de vida a nuestros monstros alienígenas. Hablamos de las animaciones Generic y Legacy. Esta última ya no se trabaja de forma habitual, ya que es una tecnología de versiones anteriores de Unity.

Cuando trabajamos con sistemas de huesos diferentes de los humanoides, es importante definir aspectos de puntos de gravedad, peso, extremidades, etc. Por eso, las animaciones genéricas son las ideales desde este punto de vista ya que permiten trabajar cada parte del sistema no-humanoide de forma particular.

Enlace

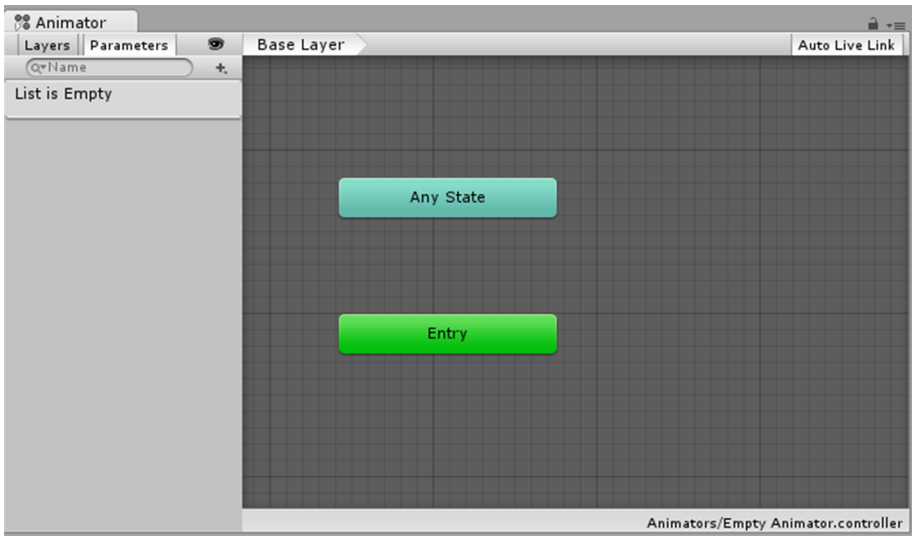
Crear y configurar sistema de animación no-humanoide
https://docs.unity3d.com/es/current/Manual/GenericAnimations.html

4.3.2. Animator Controller

Si queremos que nuestras animaciones trabajen de forma ordenada y respecto a una serie de parametros definidos conforme el juego trasncrurra, utilizaremos la interficie del Animator Controller que, como el propio nombre describe, podremos controlar las animaciones de las diferentes escenas según las condiciones que estipulemos en cada etapa del proyecto.

El sistema está basado en una estructura en forma de diagrama de flujo o en lenguaje Unity, *State Machine*. Se trata de un sistema de control visual donde podrás organizar las diferentes acciones a cada personaje o elemento. Su interface es sencilla y con una curva de aprendizaje rápida, básicamente tenemos la ventana principal donde estructuramos las animaciones y los eventos y un menú lateral que presenta un menú de capas y parámetros de la animación actual.

Imagen 34. Interface del Animator Controller



La forma de trabajar con los estados de animación, encontraremos detallada en el en manual de Unity, más concretamente en el apartado de máquinas de estado de animación.

Enlace

Métodos de trabajo para determinar estados de animación: https://docs.unity3d.com/es/current/Manual/AnimationState Machines.html

4.3.3. Rendimiento y Optimización

Dependiendo del tamaño del proyecto y su complejidad, antes de empezar a trabajar los personajes y las escenas, sería una buena practica pensar en el rendimiento del juego en el momento de la ejecución. Una buena planificación previa nos puede ahorrar problemas de sobrepeso del motor cuando tenga que calcular todos los parámetros del juego. Texturas, materiales y animaciones tienen que estar dimensionadas de forma que ayuden a que el juego se desarrolle de forma fluida y sin tirones. Hay algunos parámetros sencillos que podemos aplicar en las diferentes fases de trabajo para más adelante, poder ganar en rentabilidad de cálculo. Un ejemplo sería el caso de un personaje en una determinada escena que no tiene que coger ningún objeto o interactuar con las manos. En este caso concreto, no tiene sentido aplicar un sistema de huesos en sus manos, ya que no habrá interacción en este sentido. Unity ahorrará cálculo y el movimiento del personaje será más fluido. Reducir el número de huesos a las necesidades de cada personaje y escena, nos permitirá adecuar el elemento de forma más precisa. Podríamos aplicar el mismo concepto con las texturas del personaje o elemento.

Otro aspecto importante a tener en cuenta cuando estamos planificando un ahorro de medios, es el relativo a ordenar a Unity cuando tiene que trabajar las características de un determinado personaje. Unity está en constante evaluación de los elementos de las escenas por tanto, si en un determinado momento un personaje no aparece en la escena, tenemos que informar a Unity que deje de calcular sus propiedades y esto se puede hacer, por ejemplo, deshabilitando la propiedad *Update When Offscreen* del *mesh renderer*, de esa forma, Unity no actualizará los personajes que no estén en una escena.

Aparte de los consejos anteriores, hay que tener presente también algunos aspectos relacionados con las capas, las curvas escaladas y optimizaciones de escenas, todos estos puntos podremos ver en detalle en la documentación en línea de Unity.

Enlace

Claves para el rendimiento y optimización de personajes y escenas.

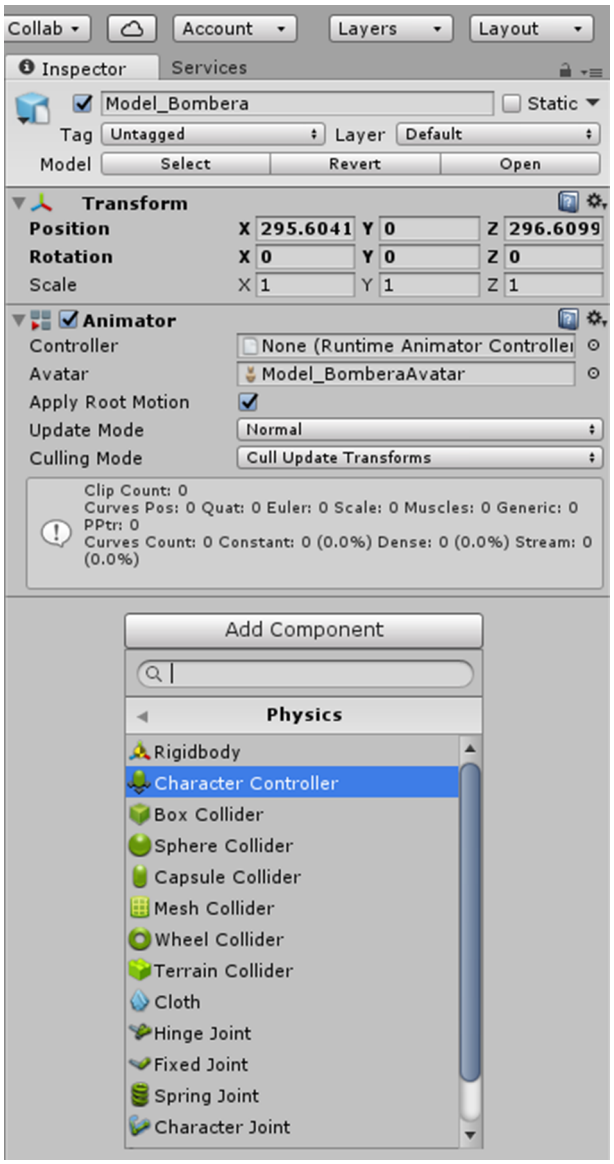
https://docs.unity3d.com/es/current/Manual/MecanimPeformanceandOptimization.html

4.3.4. Ciclos de animación – Ejemplo practico

En nuestro ejemplo de proyecto, ya hemos importado el modelo, hemos verificado su sistema de huesos, hemos añadido un terreno con texturas y ahora nos falta añadir el modelo a la escena y crear ciclos de animación.

Añadiremos a la escena un controlador del personaje (*characters controller*)

Imagen 35. Complemento Unity para control del personaje



Enlace

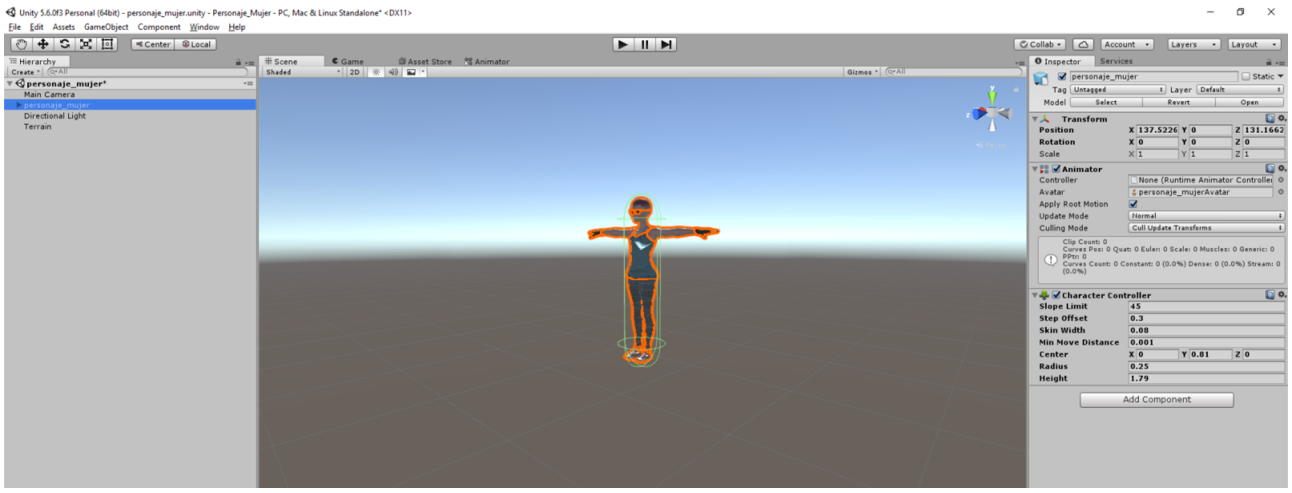
Complemento de control de la animación del personaje

https://docs.unity3d.com/es/current/Manual/class-CharacterController.html

https://docs.unity3d.com/es/current/Manual/CharacterControllers.html

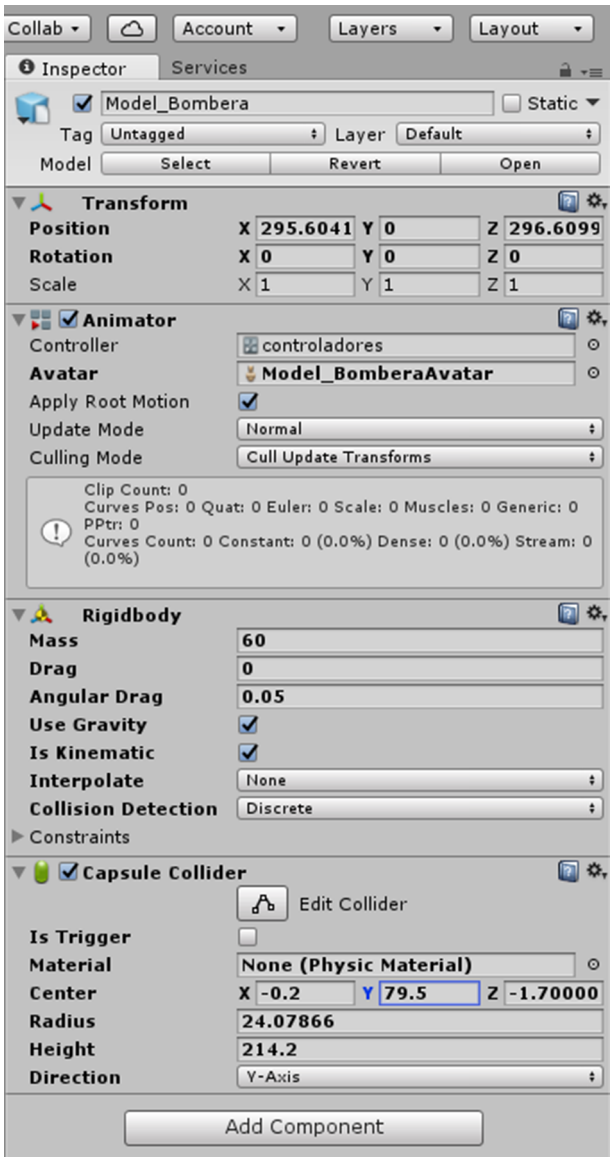
Configuramos el Controlador del personaje de forma que el mismo se encuentre dentro del contenedor (controlador personaje). Cambiamos las variables de radio y posición x, y i z.

Imagen 36. Configuración del Character Controller



Añadimos un complemento *rigidbody* y configuraremos su masa en 60. Esta masa es el peso aproximado que tendrá el personaje.

Imagen 37. Otros elementos para la animación



Para que la animación funcione de forma correcta, vamos a crear un script y añadiremos también un elemento *Animator Controller*, que funcionará como cuadro de funciones de la animación.

En la ventana de *asset*, botón derecho del ratón, seleccionamos la opción *CreateC# script*. Utilizaremos un script modelo de la web de Unity para definir los parámetros de la animación.

Enlace

Complemento en *C#*. Codigo para el controlador de personaje

https://docs.unity3d.com/ScriptReference/CharacterController.Move.html

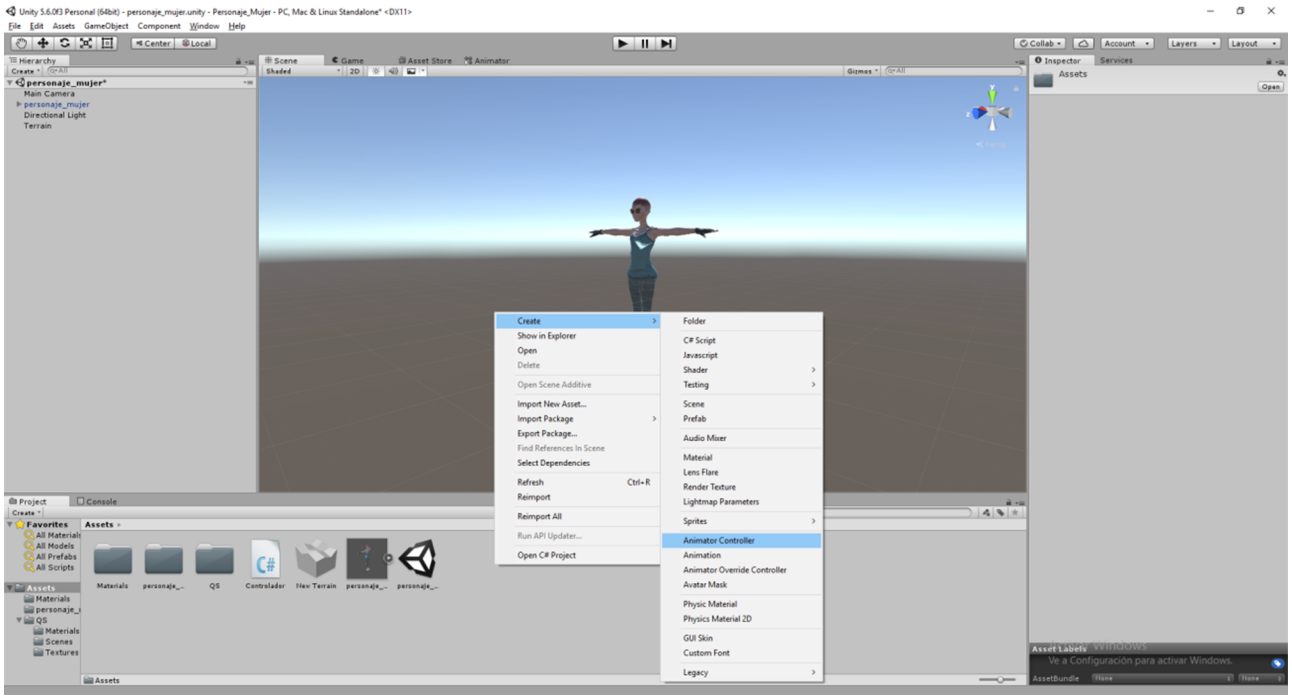
```
using System.Collections;
using System.Collections.Generic;
using Unity Engine;
public class NewBehaviourScript : MonoBehaviour {
```

```
    public float speed = 6.0F;
    public float jumpSpeed = 8.0F;
    public float gravity = 20.0F;
    private Vector3 moveDirection = Vector3.zero;
    void Update()
    {
  CharacterController controller = GetComponent<CharacterController>();
  if (controller.isGrounded)
  {
    moveDirection = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));
    moveDirection = transform.TransformDirection(moveDirection);
    moveDirection *= speed;
    if (Input.GetButton("Jump"))
  moveDirection.y = jumpSpeed;
  }
  moveDirection.y -= gravity * Time.deltaTime;
  controller.Move(moveDirection * Time.deltaTime);
    }
```

Arrastramos este controlador en el personaje.

Añadimos un *Animator Controller* i arrastramos el mismo a la ventana de *Inspector* en el componente *Animator* (casilla *Controller*).

Imagen 38. Complemento animator controller



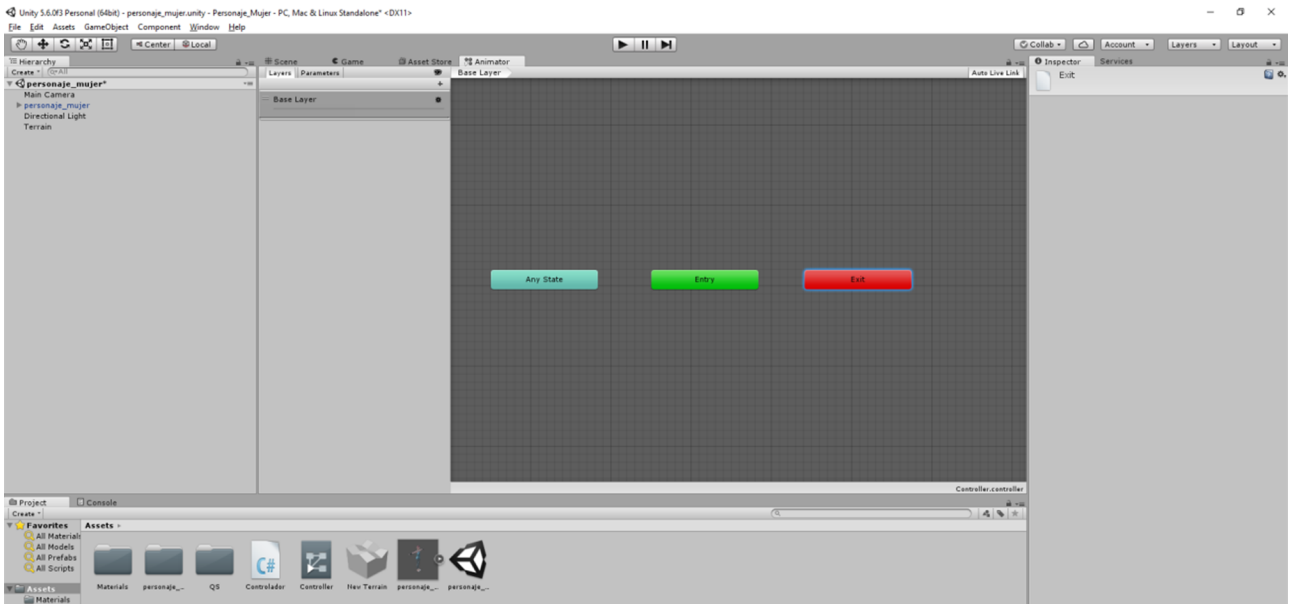
«Un *asset* Animator Controller es creado dentro de Unity y le permite a usted arreglar y mantener un conjunto de animaciones para un personaje u objeto. En la mayoría de situaciones, es normal tener múltiples animaciones y cambiarles entre sí cuando ciertas condiciones de juego ocurren. Por ejemplo, usted puede cambiar de una animación de caminar a una saltando cuando la barra de espacio sea presionada. Sin embargo, incluso si usted solo tiene un clip de animación, usted todavía necesita colocarlo a un animator controller para utilizarlo en un Game Object.

El *controller* tiene referencias a los clips de animación utilizados dentro de él, y maneja los varios estados de animación y las transiciones entre estos utilizando algo llamado un State Machine, que puede ser pensado como un tipo de diagrama de flujo, o simplemente un programa escrito en un lenguaje de programación visual dentro de Unity.»

Enlace

Complemento tipo asset para controlar la animación: https://docs.unity3d.com/es/current/Manual/class-AnimatorController.html

Imagen 39. Ventana de trabajo del complemento animator controller



Para entender de forma más precisa los elementos del Animator Controller, podemos encontrar más información en el manual de Unity.

Enlace

Interpretación de los elementos: https://docs.unity3d.com/es/current/Manual/AnimatorWindow.html

4.4. Utilizando ciclos de animación tipo *assets*

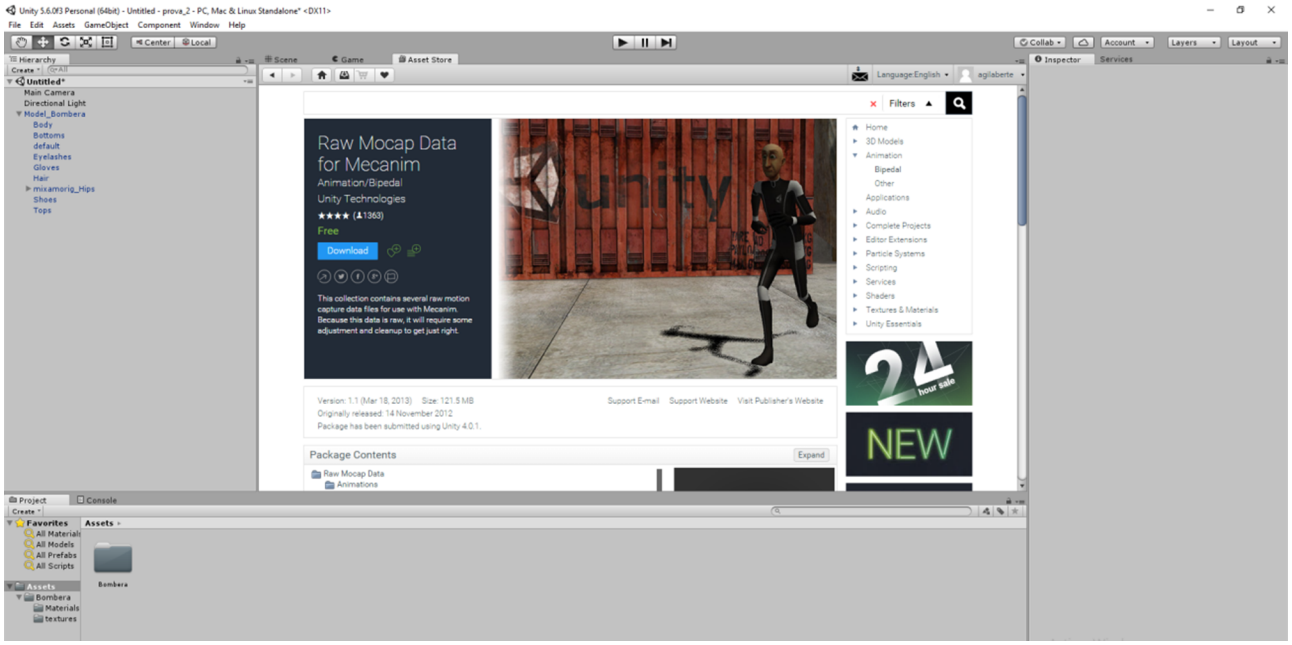
Unity permite importar clips animados o proyectos enteros o incluso construir una animación desde la misma interface del programa. De una u otra forma, una vez que la animación esté en el entorno de Unity, podremos acceder a diferentes herramientas para controlar estas animaciones en el espacio y en el tiempo.

En la tienda de Unity encontraremos personajes y animaciones ya listas para introducir en la escena del proyecto y podremos modificarlas según el criterio del juego.

En el capítulo 1 hemos tratado de la organización de la tienda de Unity y como importar estos *assets* a nuestro proyecto, ahora vamos ver los principales controles de animación.

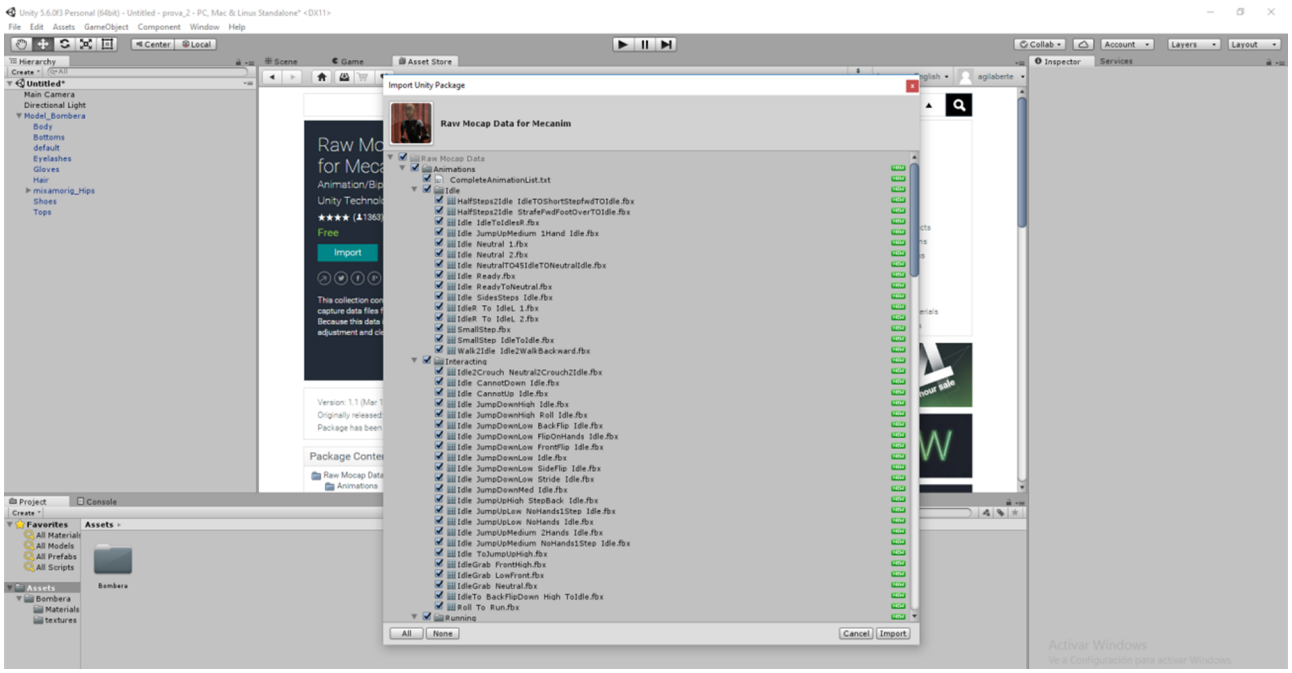
El primero paso para animar de forma automática a este personaje, es buscar un conjunto de animaciones ya realizadas en la biblioteca de Unity, pongamos un ejemplo de la lista de complementos gratis: *Raw Mocap Data* es el conjunto de animaciones más completo de la store de Unity.

Imagen 40. Conjunto de animaciones



Importamos todos los elementos asociados a la biblioteca de animación.

Imagen 41. Elementos a importar



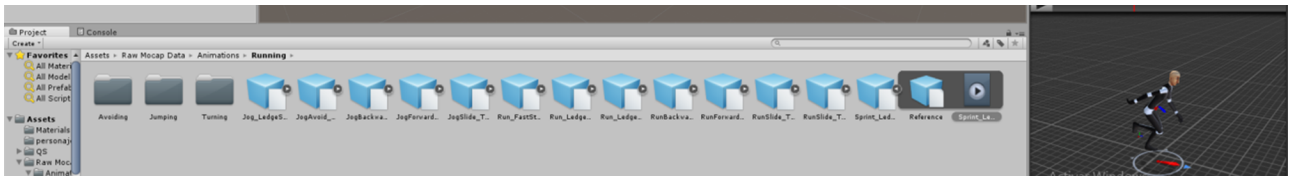
En la parte inferior del programa, veremos que dentro de la carpeta de *assets* ahora tenemos los ciclos de animación importados de la biblioteca de Unity, más concretamente los ciclos de *Raw Mocap Data*.

Dentro de esta carpeta, podremos ver todas las animaciones que se han cargado en el proyecto.

En este caso serán seis: *Idle*, *Interacting*, *Running*, *Sprinting*, *Strafing* y *Walking*.

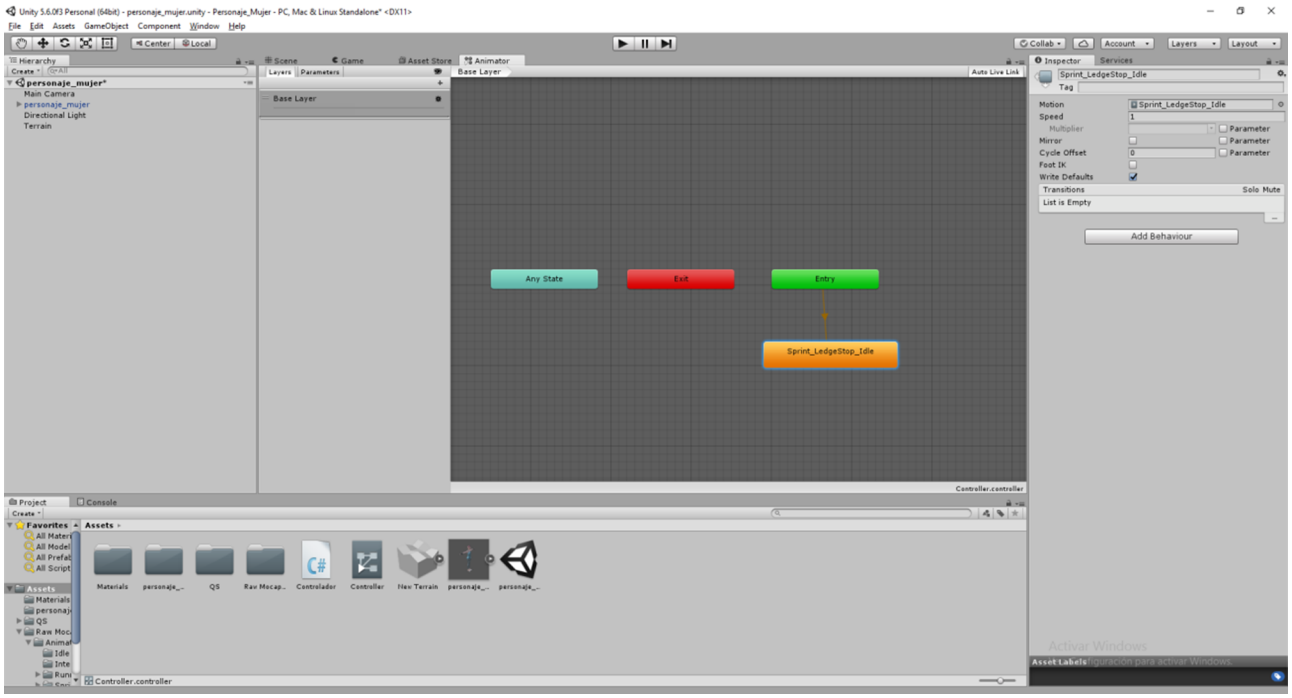
Si queremos visualizar cada una de esta animaciones, desplegamos cada carpeta hasta llegar en la animación que queramos ver y luego, seleccionamos el botón *player* al lado de cada animación.

Imagen 42. Visualizar los diferentes ciclos de animación del *assetRaw Mocap Data*



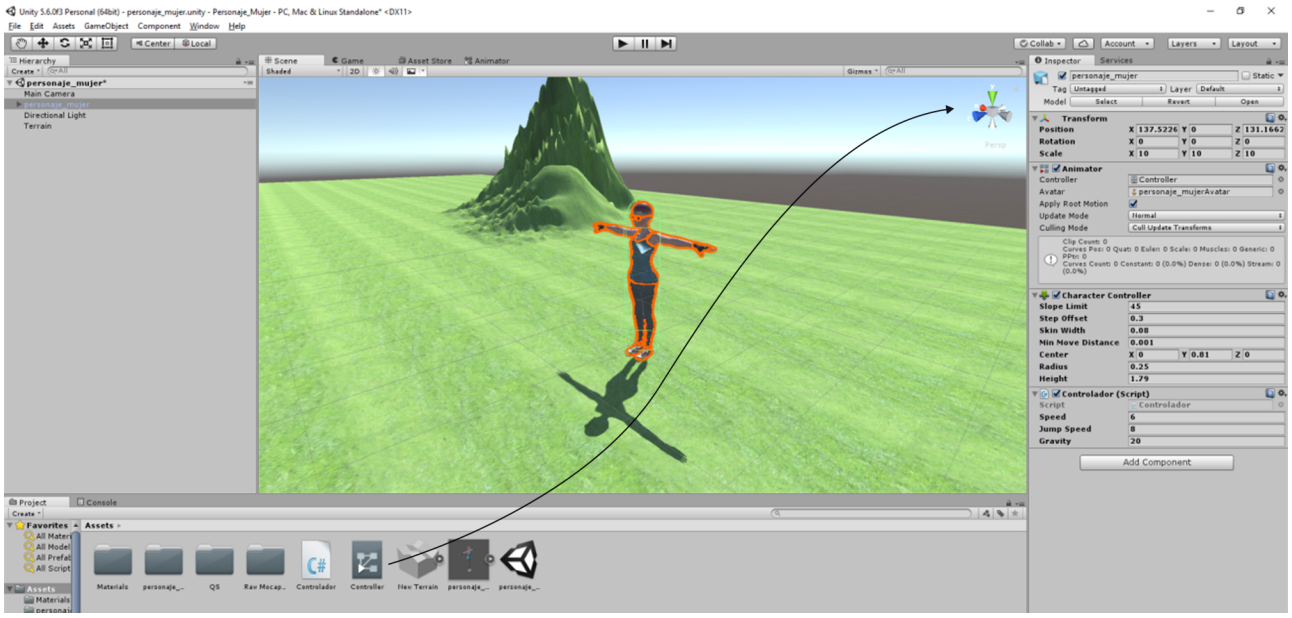
Paso siguiente será aplicar el ciclo al personaje que hemos traído de Mixamo. Para tal, basta arrastrar el ciclo de animación correspondiente a la ventana del *Animator Controller*.

Imagen 43. Asignar un tipo de animación al personaje



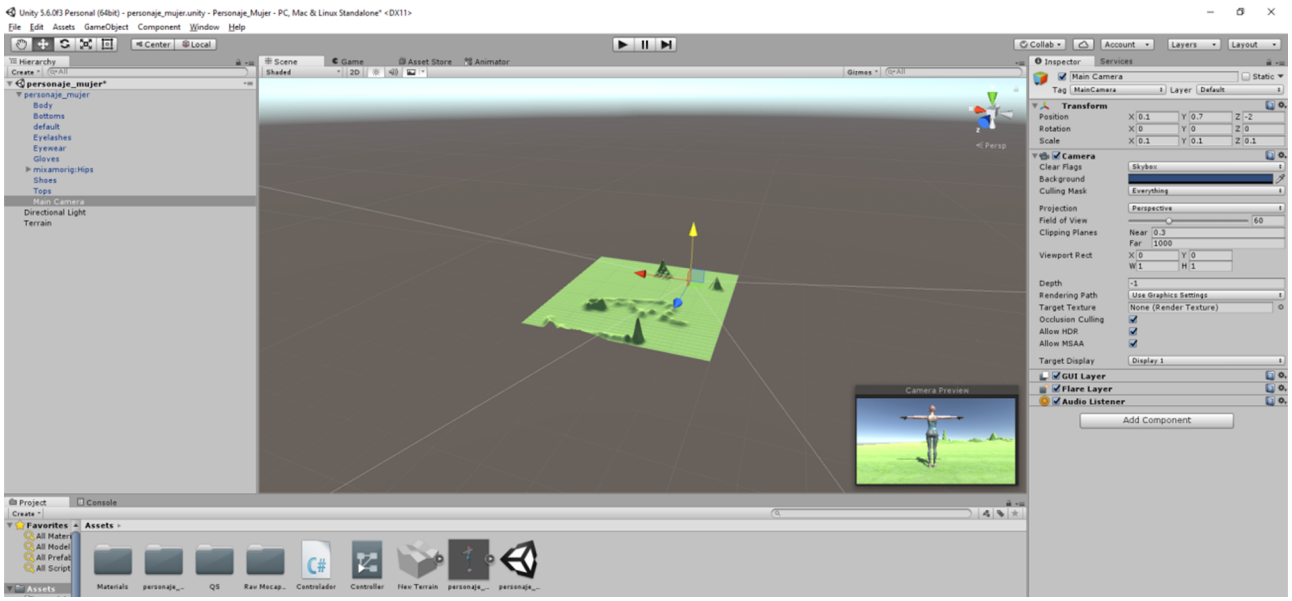
Por último y para que el personaje pueda animarse, arrastramos el *Character Controller* a la casilla *controller* del *assetAnimator Controller*.

Imagen 44. Asignación del Character Controller a Animator Controller



Por último, arrastramos la camera de la escena dentro del personaje y de esta forma, podremos hacer un seguimiento de la animación en primera persona.

Imagen 45. Elemento camera en la escena



Las cameras en Unity juegan un papel fundamental en cuanto a la narrativa del juego. Es importante saber determinar sus propiedades y ajustar sus parámetros en el espacio de juego.

Enlace

Descripción y configuraciones de cameras en Unity: https://docs.unity3d.com/es/current/Manual/class-Camera.html

5. Transiciones

> «Las *Animation transitions* (transiciones de animación) le permite al *state machine* (maquina de estado) cambiar o mezclar de un *animation state* (estado de animación) a otro. Las *Transitions* (transiciones) definen no solo el tiempo de cambio entre los dos estados debería tomar, sino que también bajo qué condiciones debería estar activa. Una transición puede ser configurada para que solo ocurra cuando una cierta condición sea cierta, y estas condiciones son basadas en los valores de los parámetros configuradas en el *Animator Controller*.
>
> Por ejemplo, su personaje podría tener un estado de patrullar y un estado durmiente. La transición entre patrullar y dormir puede ser configurada para que ocurra solamente cuando el valor del parámetro "alertness" caiga debajo de cierto nivel.»

Las transiciones son parámetros que están bien explicados en el manual de Unity y tienen un peso muy importante en la secuencia de cada escena, ya que funcionen con determinados puntos de interacción entre el u usuario final o los objetos que están interaccionando en la misma escena del juego.

Enlace

Descripción y configuraciones de las transiciones de las animaciones:

https://docs.unity3d.com/es/current/Manual/class-Transition.html

Resumen

En este módulo hemos trabajado diferentes aspectos y herramientas que componen el universo de los media para video juegos. El estudiante tiene como objetivo saber identificar los diferentes medias disponibles y determinar cuando y donde hay que utilizar elementos de diferentes procedencias de forma a completar el proyecto definitivo.

Unity es el centro del proyecto y dentro de un radio de acción encontramos diferentes elementos y programas que completan el proyecto definitivo.

Hemos estudiado la integración de elementos provenientes de bibliotecas externas y su compatibilidad en Unity. En este sentido, hemos explorado la biblioteca Store de Unity, analizando sus posibilidades y el sistema organizativo donde encontramos múltiples opciones de elementos que son exportables a Unity. Dentro de este conjunto de elementos, diferenciamos aquellos que son elementos únicos de aquellos que conllevan un conjunto de elementos agrupados en un único paquete, lo que denominamos paquetes de assets. Saber identificar la necesidad de cada momento del proyecto y recorrer a la biblioteca de Unity para encontrar elementos pre definidos constituye una competencia básica en el flujo de trabajo.

Respecto al modelado de personajes, hemos estudiado el programa Fuse de Adobe que permite la preparación de los personajes para su posterior exportación. La interface de trabajo facilita un entorno con una curva de aprendizaje fácil y objetiva. Completando el ciclo del modelado, hemos aprendido a exportar los personajes de Fuse a Mixamo, donde hemos podido generar un sistema de huesos completo y con una extensa biblioteca de movimientos definidos y aplicados en el personaje importado.

Para finalizar, hemos agrupado todos estos elementos de programas externos y hemos importado a Unity, donde hemos podido trabajar la integración de estos elementos en un entorno de juego definido por el propi alumno.

Una vez en el motor principal del entorno del juego, hemos aprendido algunas técnicas específicas para crear elementos determinados, como la creación de terrenos, césped o árboles.

En cuando a la interactividad de Unity, se han identificado algunas herramientas de control de personajes que definen algunos puntos clave en la animación de elementos en Unity. En este sentido, hemos visto de forma introductoria los principales aspectos relacionados con los gráficos, el flujo de trabajo en el entorno, los clips de animación o elementos más específicos de control, con las herramientas de Animator Controller, por ejemplo. El audio también ha

formado parte de este conjunto de conceptos básicos que hemos trabajado en este módulo, donde también hemos estudiado sus principales componentes y la forma de interacción con el entorno Unity.

Hemos comentado de forma genérica cada uno de estos elementos y ahora cabe al alumno explorar y desarrollar diferentes técnicas de trabajo en el entorno Unity y su universo de programas complementarios.

Actividades

Para asimilar todo el contenido de este módulo, se propone la creación de una escena de un juego de golf en Unity. Los pasos principales para poder finalizar la escena y aplicarle las técnicas específicas son los siguientes:

1. Crear un proyecto en Unity y, por medio de la herramienta de creación de terrenos, crear una plataforma rectangular de dimensiones libres.

2. Aplicar césped sobre este terreno y rellenar el entorno con árboles.

3. En Fuse, crear un personaje que sea un jugador de golf. Aplicar diferentes parámetros de modelado del personaje para que se adecue al personaje imaginado. Exportar el personaje final a Mixamo.

4. En Mixamo, crear un sistema de huesos completo (tipo Standart Skeleton). Buscar una animación pre definida relacionada con el golf y aplicarla al personaje.

5. Importar el personaje de Mixamo a Unity e integrarlo en el campo de golf.

6. Animar el personaje para que camine por el campo, haga paradas en el recorrido y lleve a cabo diversos movimientos de golf.

7. Trabajar la iluminación del entorno para que se vean bien las sombras.

8. Explorar y aplicar un sistema de script para que el jugador golpee una bola de golf.

Bibliografía

Biblioteca de contenidos de Unity

<https://www.assetstore.unity3d.com/en/>

Diccionario de términos de juegos

<http://www.gamerdic.es/>

Fuse

<http://www.adobe.com/es>

Manual de Unity

<https://docs.unity3d.com/es/current/Manual/UnityManual.html>

<https://helpx.adobe.com/es/support/fuse.html>

Mixamo

<https://www.mixamo.com/>

Reductor de polígonos para Mixamo

<https://www.mixamo.com/decimator>

Unity Blogs

<https://blogs.unity3d.com/es/2010/06/29/mod-in-unity/>