

UNIDAD 2:

Componentes del proyecto

Diagrama de organización



ucontinental.edu.pe



Tema n.º 1: Recursos en Unity

Sub tema 1.1 Gestión de recursos:

En Unity los recursos son todos los archivos que se copian o crean en la ventana Proyecto (Scripts, Mallas, Materiales, Texturas, Animaciones, Audio, etc.). Lo necesario para tu proyecto va a depender si se trata de un videojuego 2D o 3D. En las siguientes secciones describiremos los recursos necesarios en la producción de un videojuego 3D.

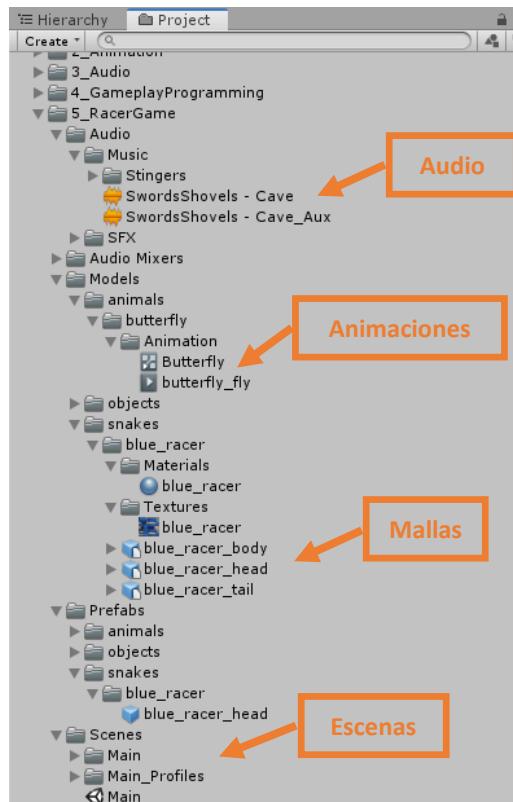


Fig N° 1. Recursos de un proyecto en Unity 3D

Sub tema 1.2 Mallas

Un objeto tridimensional está compuesto por mallas, el cual se edita en un software de modelado 3D. Hay muchas opciones entre gratuitas y de costo. Vamos a ver el uso básico de Blender, un software gratuito y muy completo de creación 3D:

1. En primer lugar, necesitamos instalar Blender, para lo cual tenemos que visitar la página www.blender.org y descargar la última versión. En este ejemplo se está usando la versión 2.8 beta.
2. Una vez instalado, lo abrimos, nos debe mostrar una ventana, donde hacemos click en cualquier parte de la grilla 3D para cerrar el panel con el logo de Blender.

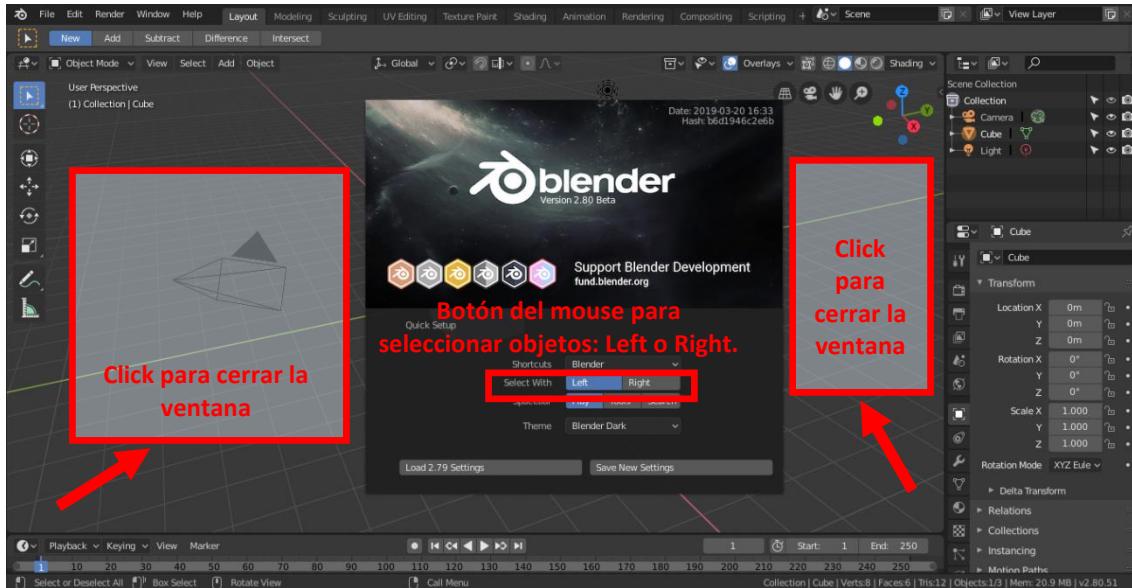


Fig N° 2. Pantalla inicio Blender. Adaptado del entorno gráfico de Blender 2.8 Beta , 2019

3. Interfaz de Blender

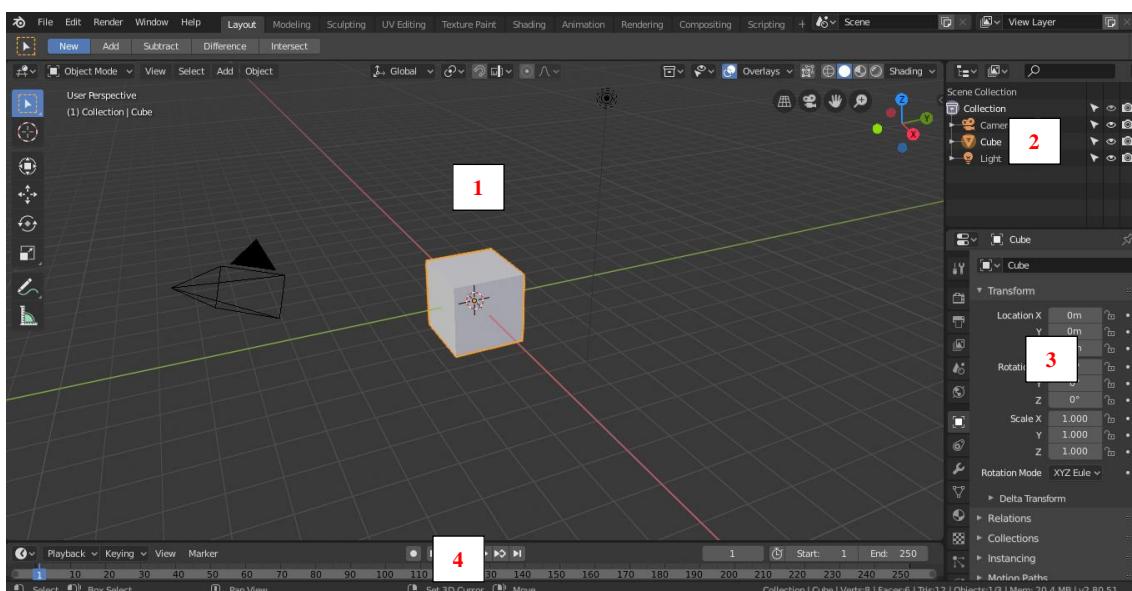


Fig N° 3. Pantalla inicio Blender. Adaptado del entorno gráfico de Blender 2.8 Beta , 2019

Elemento	Descripción
Ventana 1	ventana 3D Viewport , es el área donde puedes ver, navegar y crear objetos 3D
Ventana 2	ventana outliner , muestra en lista los objetos que hay en la ventana 3D viewport
Ventana 3	ventana properties , conjunto de propiedades del objeto actual seleccionado
Ventana 4	ventana timeline , línea de tiempo para animaciones

- En el centro hay un cubo, Blender lo crea por defecto, y lo selecciona automáticamente.

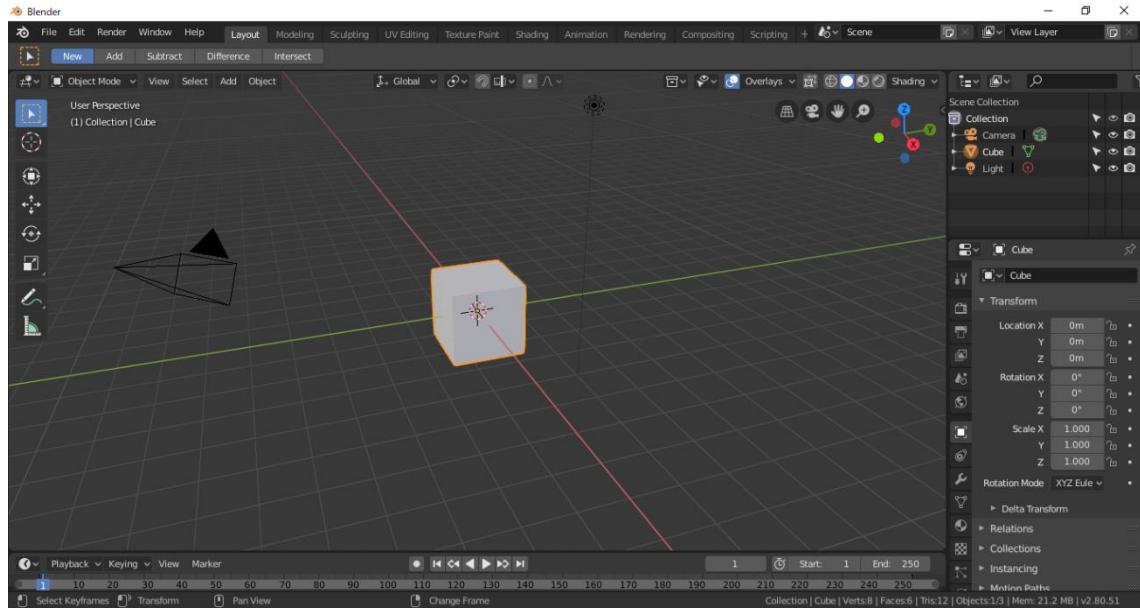


Fig N° 4. Cubo por defecto. Adaptado del entorno gráfico de Blender 2.8 Beta , 2019

Si no está seleccionado, dependiendo del botón de selección (click derecho o izquierdo) podemos seleccionar el cubo. Nótese que aparece un borde naranja alrededor del objeto seleccionado.

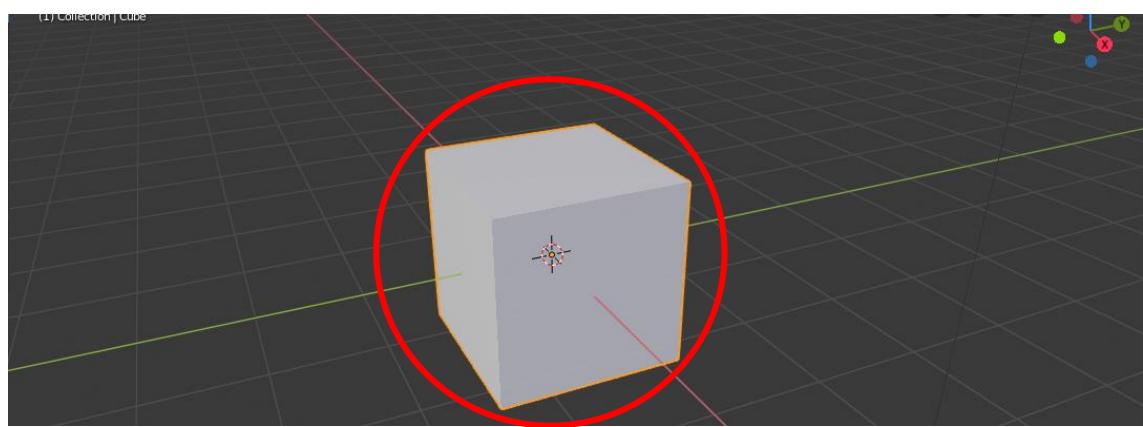


Fig N° 5. Cubo seleccionado. Adaptado del entorno gráfico de Blender 2.8 Beta , 2019

5. Usaremos la técnica de Extrucción (Extrude) para crear objetos 3D en este ejemplo. Para lo cual primero tenemos que pasar al modo de edición con la tecla TAB o eligiendo el modo en la esquina superior izquierda, lo que hará que el cubo tome una tonalidad naranja.

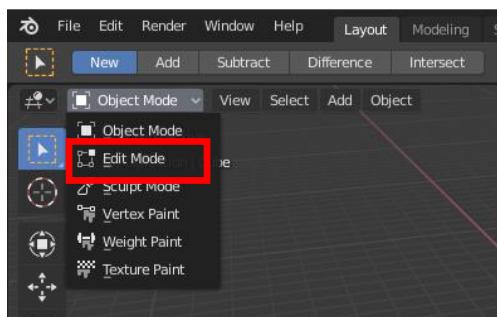


Fig N° 6. Elegiendo el modo edición de objetos

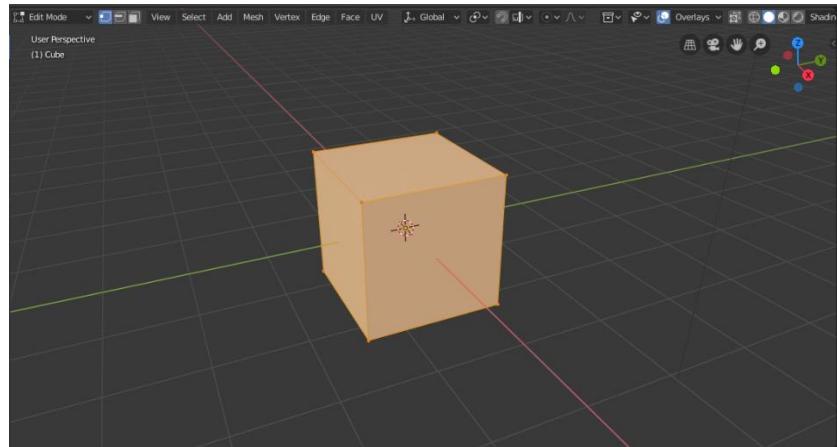


Fig N° 7. Objeto en modo Edición. Adaptado del entorno gráfico de Blender 2.8 Beta , 2019

6. Un objeto 3D está compuesto por Caras (**faces**), Esquinas (**Edges**), y Vertices (**Vertex**). En Blender elegimos el modo de selección en la parte superior izquierda:

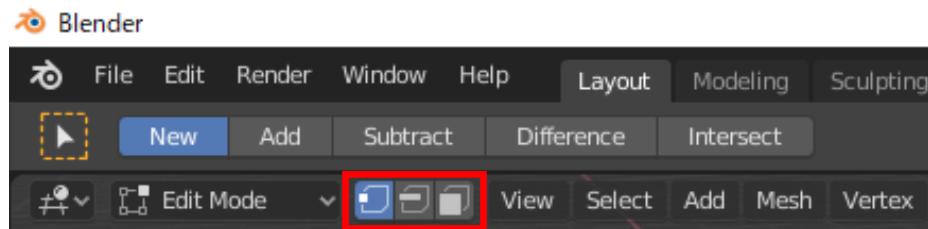


Fig N° 8. Elección del modo de selección

7. Para realizar un extrude en este ejemplo, elegimos el modo de selección de Caras y seleccionamos la Cara que queremos expandir, con el clic derecho o izquierdo, dependiendo del botón de selección.

TIP: Usando la tecla 'A' podemos seleccionar o deselegir todo.

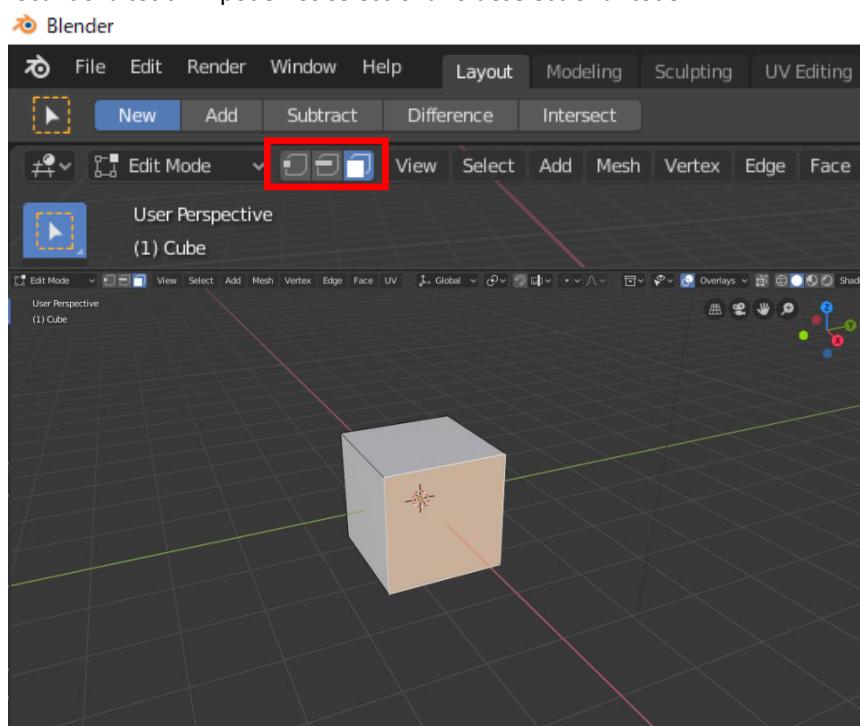


Fig N° 9. Seleccionando la cara a editar

- Utilizamos la tecla 'E' y arrastramos el cursor hacia una dirección para realizar una expansión de la Cara seleccionada y con el clic de selección en cualquier parte aceptamos el resultado.

TIP: Inmediatamente después de presionado la tecla 'E' podemos presionar un eje con las teclas X Y Z, para forzar la acción sólo en un eje. Esto también funciona con los modificadores de transformación (teclas G (grab, mover) R (rotación) y S (scale, escala))

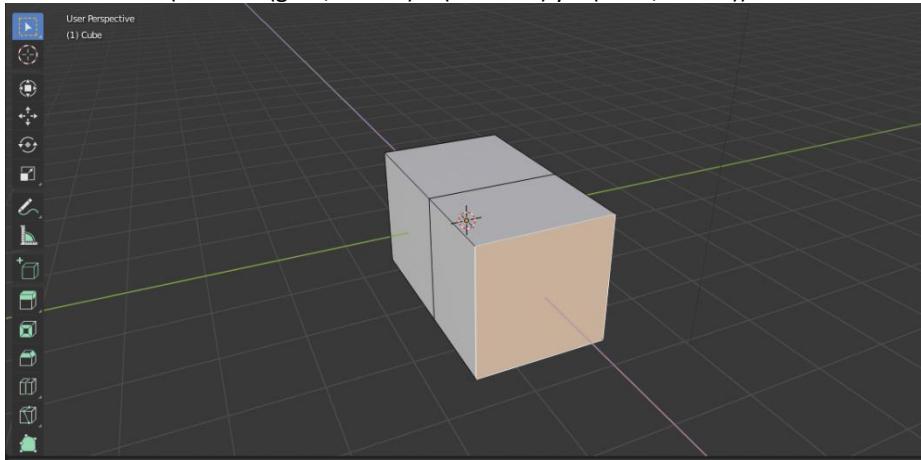


Fig N° 10. Aplicando la extrusión en el eje “x”

- Podemos seguir expandiendo las Caras que necesitamos hasta formar la figura deseada.



Fig N° 11. Objeto con extrusión aplicada en los ejes “x” y “y”

- Para crear el Snake en este ejemplo necesitamos de 3 partes (cabeza, cuerpo y cola). Para crear un nuevo objeto 3D utilizamos las teclas Shift + A, seguido de la opción Mesh -> Cube en la ventana 3D viewport, con el modo de objeto (object mode) activo.

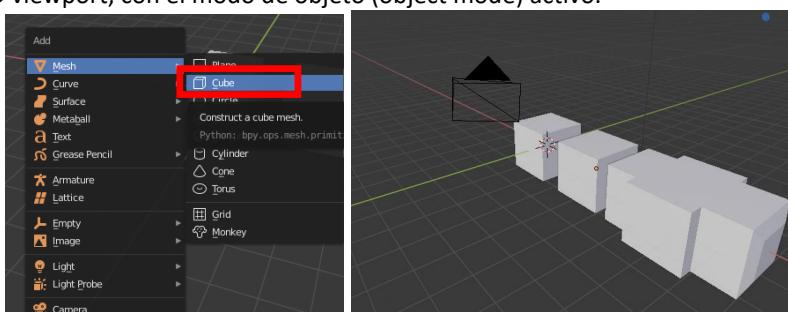


Fig N° 12. Agregando objetos cubo

- Seguimos transformando el cubo hasta tener las formas deseadas para la cabeza, cuerpo y cola

TIP: Podemos modificar las caras, las esquinas y vértices con las opciones de transformación del lado izquierdo de la ventana, o con las teclas rápidas: G (grab, mover), R (rotación) y S (scale,

escala).

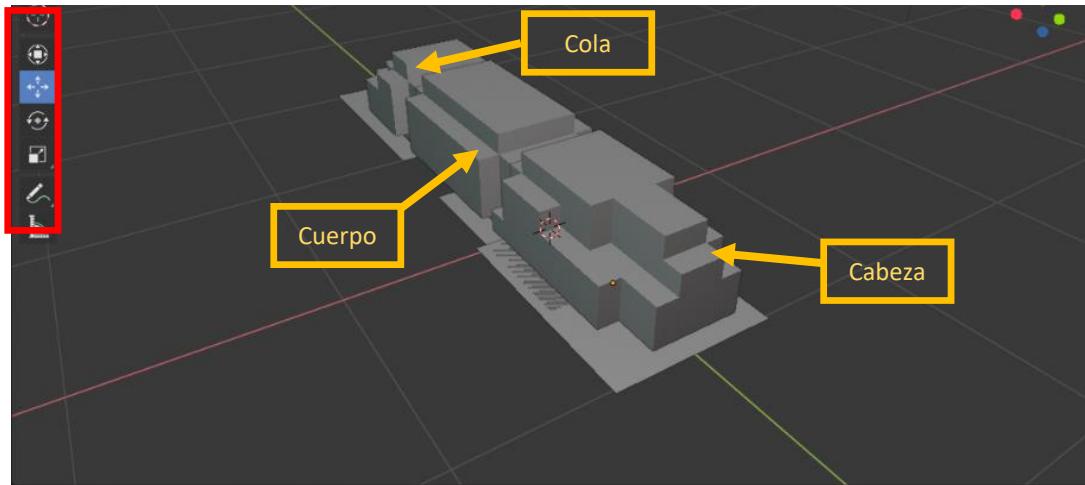


Fig N° 13. Objeto con los elementos definidos

12. A continuación aplicamos el texturizado del objeto 3D, para ello abrimos una nueva ventana desplazando el cursor del mouse a la esquina superior de la ventana (hasta que el cursor cambia a un símbolo "+") y lo arrastramos hacia la izquierda.

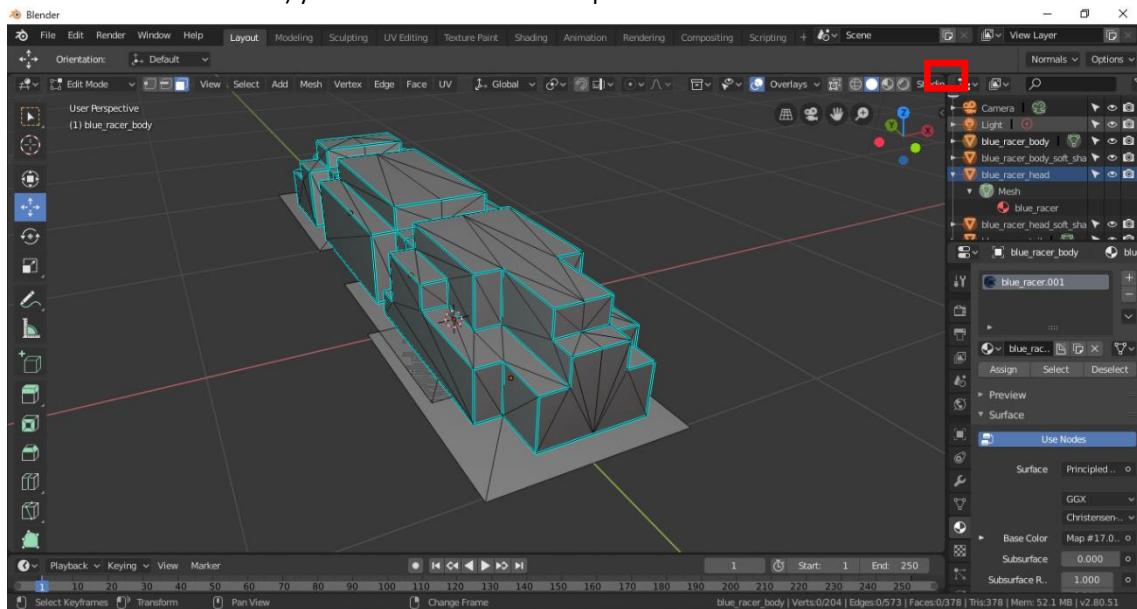


Fig N° 14. Objeto listo para texturizar

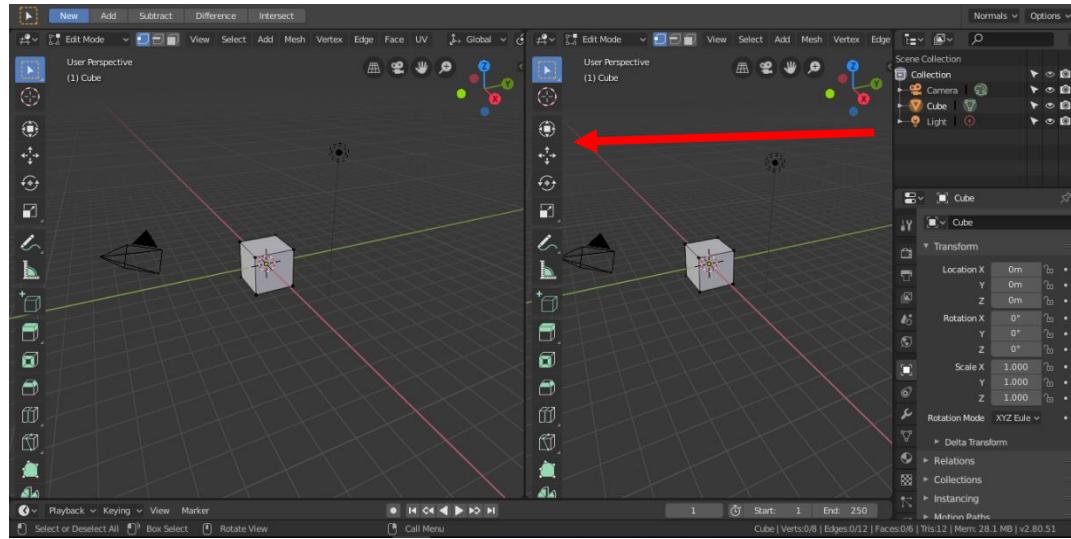


Fig N° 15. Ventanas divididas

- La nueva ventana la transformamos en ventana “UV Editor” para crear un mapa UV que sirve para situar una textura 2D en un modelo 3D

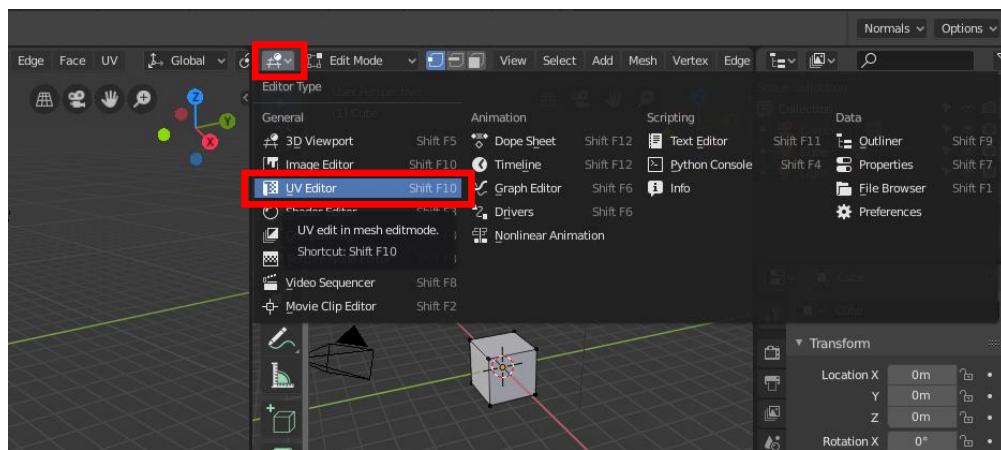


Fig N° 16. Selección del editor UV

- Si seleccionamos todas las caras (tecla A), podemos ver nuestra selección en la ventana UV Editor. Para este ejemplo vamos a usar la “proyección inteligente” de Blender, para lo cual teniendo todas las caras seleccionadas elegimos el menú UV -> Smart UV Projection, en la ventana 3D viewport.

TIP: Podemos seleccionar varios objetos al mismo tiempo manteniendo presionado la tecla Shift

TIP: Podemos mover, escalar y rotar las caras en la ventana editor de UVs al seleccionarlas



Fig N° 17. Activando Smart UV

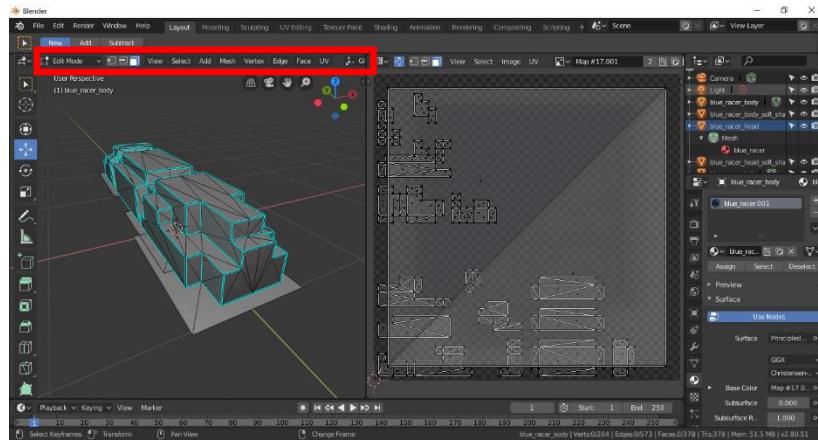


Fig N° 18. Objeto con Smart Uv preparado

15. Una vez hecha la proyección de los tres objetos seleccionados la exportamos usando el menú de la ventana UV Editor: UV -> Export UV Layout

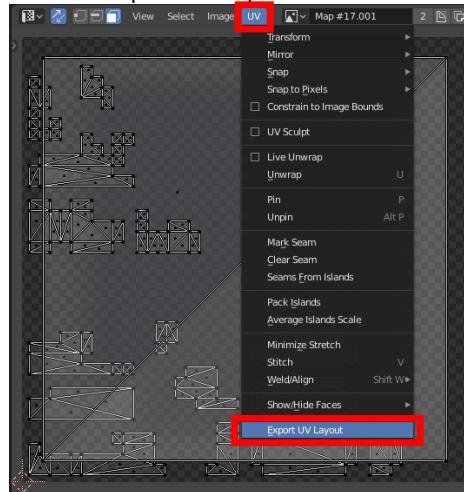


Fig N° 19. Exportando el Layout

16. La imagen generada podemos usarla como guía para pintarla con nuestro software favorito de edición de imágenes. En este ejemplo se está usando Krita, un software gratuito, el cual se puede descargar visitando: <https://krita.org/en/download/krita-desktop/>.

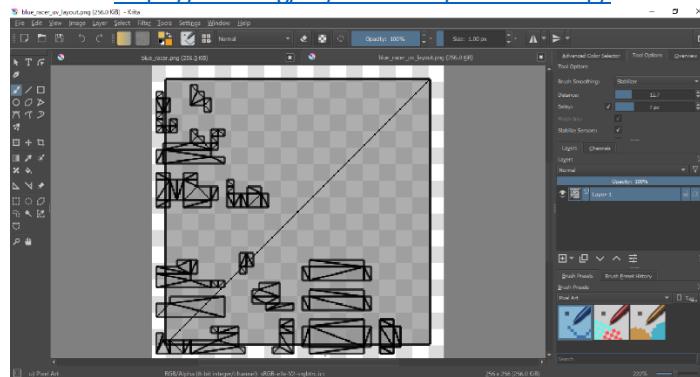


Fig N° 20. Pintado el layout en el editor Krita

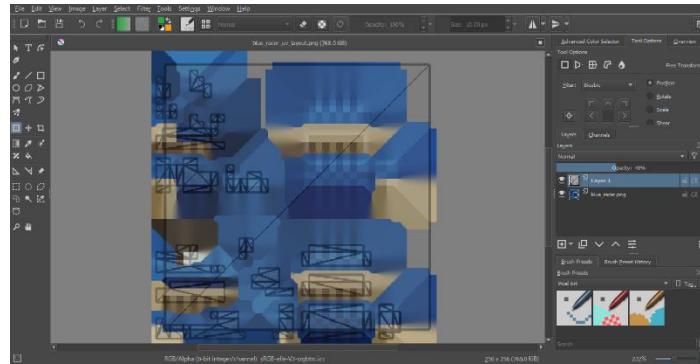


Fig N° 21. Objeto texturizado

17. Una vez hecha la textura de color la asignamos en Blender usando las opciones de Material en la ventana de propiedades. Elegimos la textura en la opción Color Base.

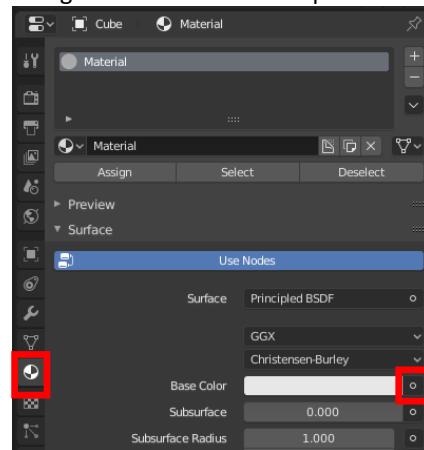


Fig N° 22. Aplicando el material al objeto de blender

18. Aplicación de la textura de Imagen

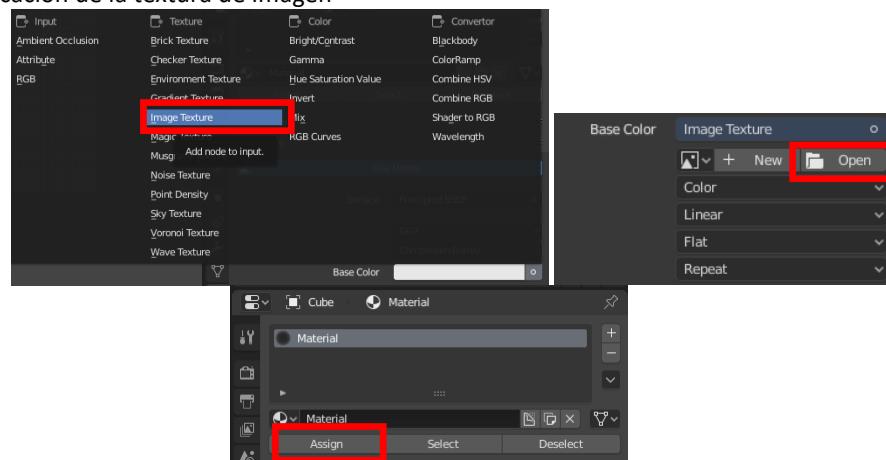


Fig N° 23. Aplicación de textura de imagen

19. Repetimos el proceso de asignación del material (agregamos un nuevo material si no existe) para cada parte (cabeza, cuerpo, cola).

TIP: Podemos asignar la textura de color en la ventana UV Editor para poder hacer correcciones o cambios moviendo las caras.

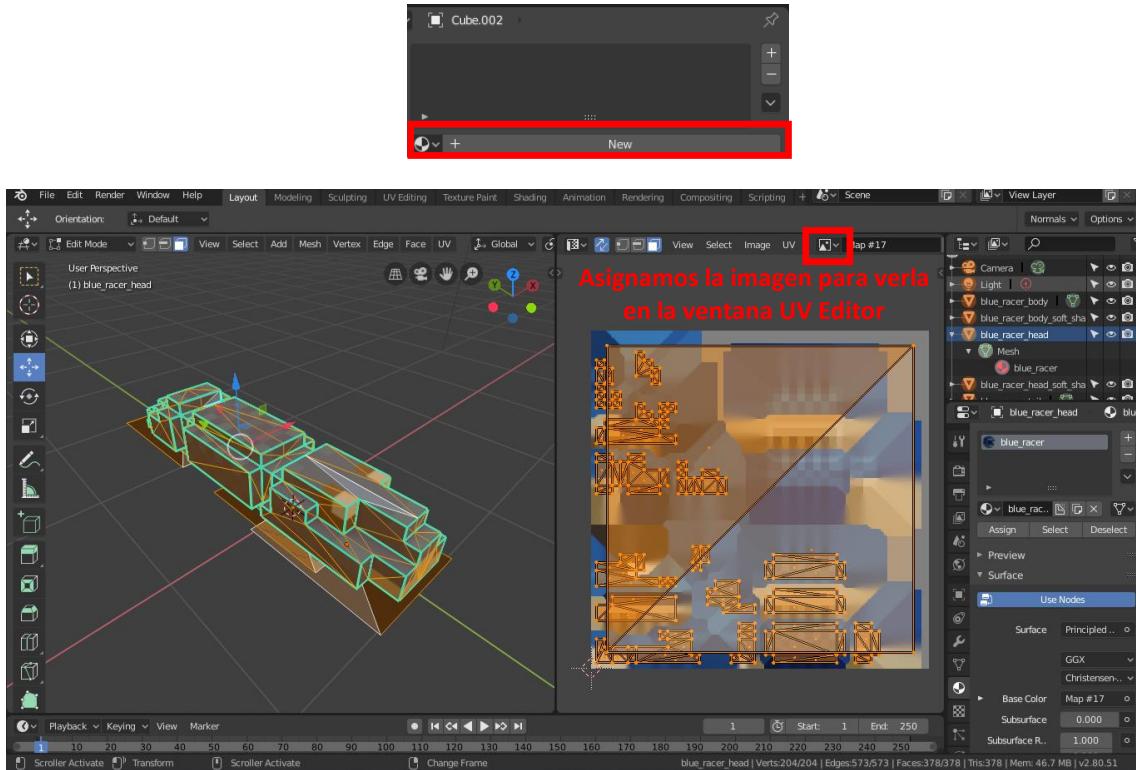


Fig N° 24. Visualizando el modelo texturizado

20. Para cerrar la ventana UV Editor desplazamos el cursor a la esquina superior derecha de la ventana 3D viewport (hasta que el cursor cambie a "+") y la arrastramos con clic en dirección de la ventana UV Editor. Podemos cambiar al modo de objeto para ver el resultado

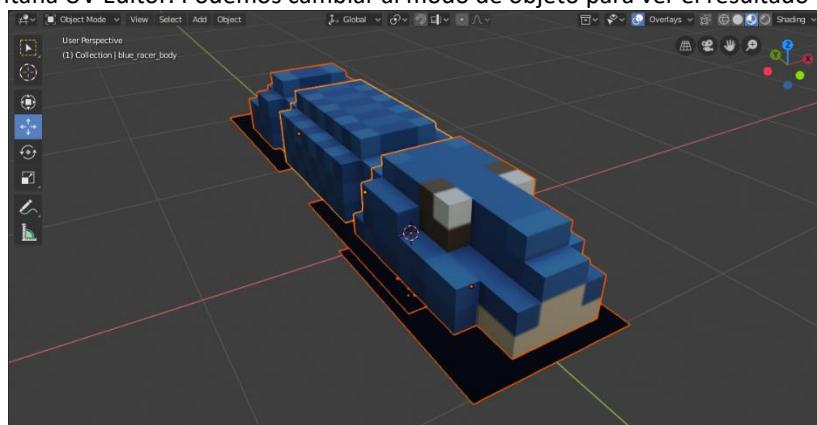


Fig N° 25. Modelo texturizado final

21. Finalmente exportamos nuestros modelos 3D en el formato estándar FBX soportado por Unity, enviándolo en la carpeta de nuestro proyecto
TIP: Para exportarlo por separado elegimos la opción “Selected Objects” en la ventana de exportación. Se tiene que repetir la acción por cada objeto, seleccionándolos individualmente.

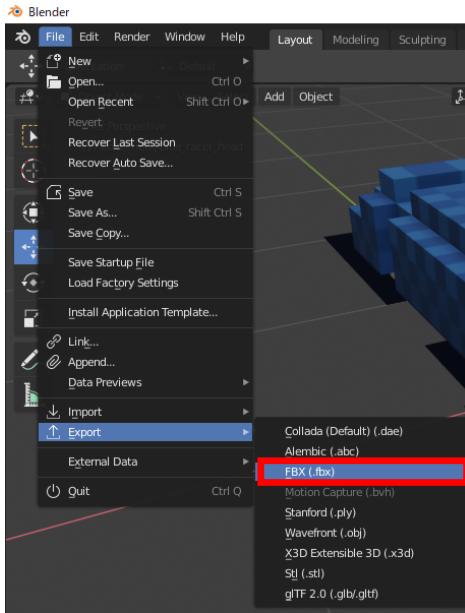


Fig N° 26. Exportando a formato FBX

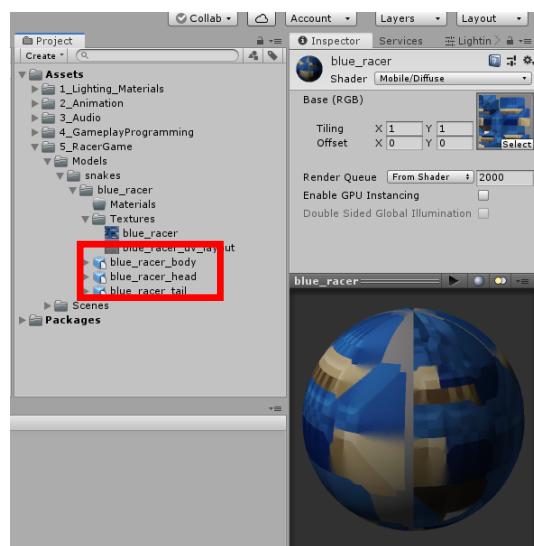
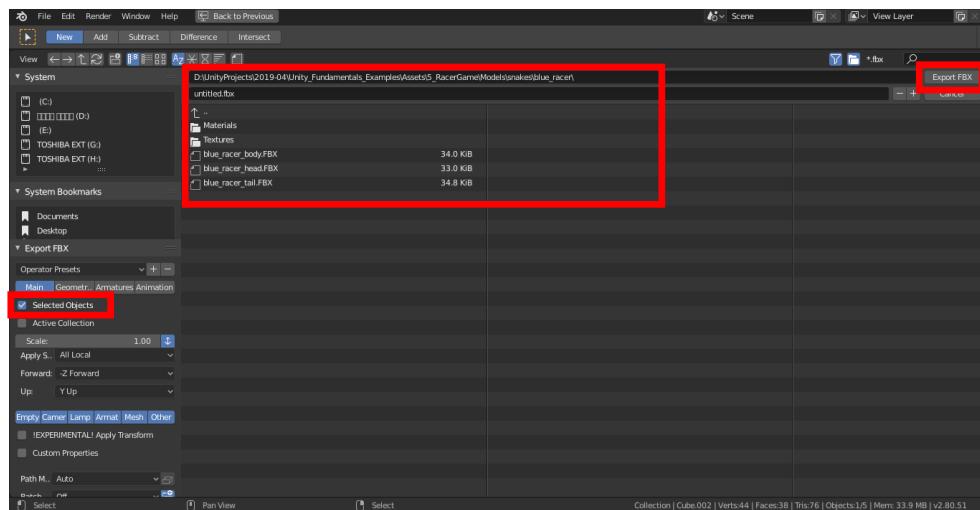


Fig N° 27. Modelo obtenido

Sub tema 1.3 Materiales

- Ahora vamos a crear un material para nuestros objetos, sólo es necesario uno para los tres ya que comparten la misma textura de color. Para ello en la ventana de proyecto en alguna carpeta (Materials en este caso) hacemos Clic derecho > Create > Material.

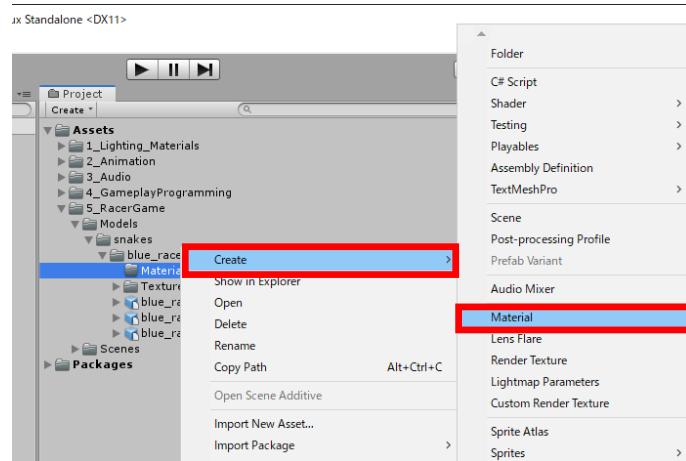


Fig N° 28. Inicio del proceso de creación de material

- Unity por defecto utiliza el Shader Standard, un shader es un grupo de instrucciones que corre en el GPU para dibujar los pixeles en pantalla. Un CPU puede contener algunos cores que pueden ejecutar varios hilos de procesamiento al mismo tiempo, pero un GPU puede contener cientos de cores. En la gráfica siguiente tenemos las tres partes necesarias para dibujar un modelo 3D en pantalla. Un modelo 3D es una colección de coordenadas (vértices) conectadas para crear triángulos, cada vértice contiene información de color, dirección (normals) y coordenadas para texturas (UV data). Los modelos necesitan materiales para ser dibujados, los materiales son contenedores con un shader en él y valores para las propiedades del shader.

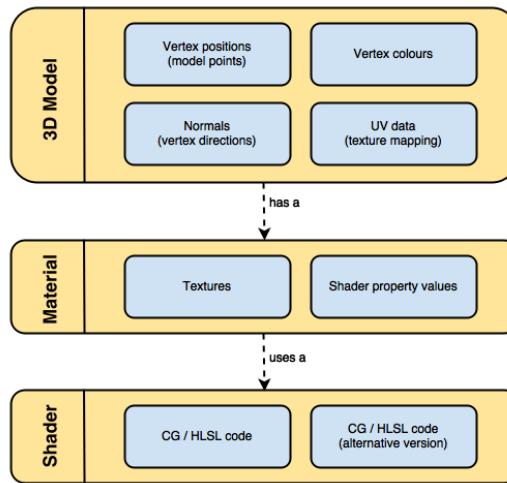


Fig N° 29. Componentes de un modelo 3D. Fuente: Unity technologies

- Un shader utiliza un modelo de luz (lighting model) que son algoritmos para determinar la luz en la superficie del modelo, en el caso del shader standard de Unity, utiliza un nuevo modelo de luz que simula la interacción natural de los rayos de luz con los materiales del mundo real. A continuación, describiremos las propiedades del shader Standard de Unity.

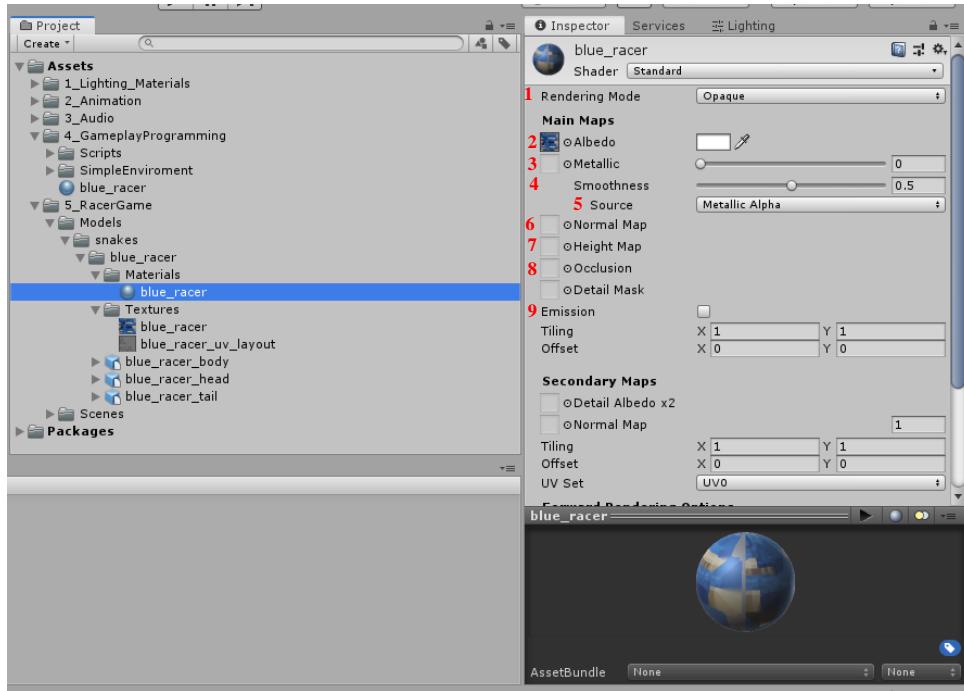


Fig N° 30. Ventana de propiedades del shader estandar

Elemento	Descripción
1	Rendering mode , permite elegir entre que el objeto use transparencia (diferentes modos de mezcla) o no
2	Albedo , este mapa de textura describe color sin ninguna mezcla de luz o sombras
3	Metallic , valor que define si un objeto es metálico (refleja/absorbe luz de todas direcciones) o no
4	Smoothness , valor que define si un objeto es suave (refleja/absorbe luz directa) o áspero
5	Source , define el origen del valor metálico (mapa metálico en escala de grises o el mapa de color) donde valores más claros hacen el objeto más metálico
6	Normal , especial forma de textura para agregar detalle como baches a los modelos que absorben luz como si representaran más triángulos
7	Height , al igual que el mapa de normales es usado para agregar detalle pero es más caro en uso de recursos GPU ya que desplaza áreas de la superficie visible para lograr un efecto de ocultación a nivel de superficie
8	Occlusion , usada para proporcionar información de que áreas van a recibir mayor o menor luz indirecta. Luz indirecta provienen de luz de ambiente o reflexiones.
9	Emission , controla el color e intensidad de luz emitido desde la superficie del objeto, que da la sensación como si el objeto fuera auto iluminado.

Fig N° 31. Propiedades del shader estandar

4. Si agregamos la textura de color en el campo respectivo, podemos ver los resultados en nuestro objeto

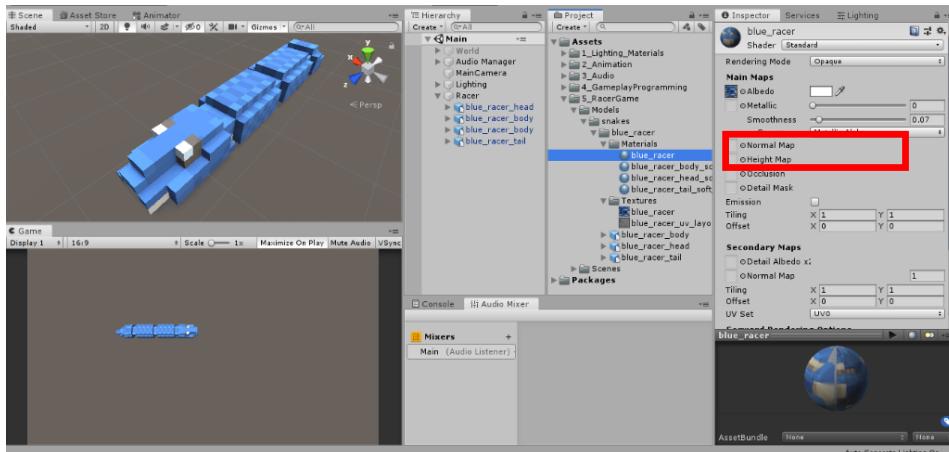


Fig N° 32. Agregando la textura de color

5. A continuación, veremos la diferencia entre los modos de renderizado.

El modo Cutout vuelve el objeto completamente invisible sólo si el valor alpha del color (A / 255) es menor al porcentaje de Cutoff.

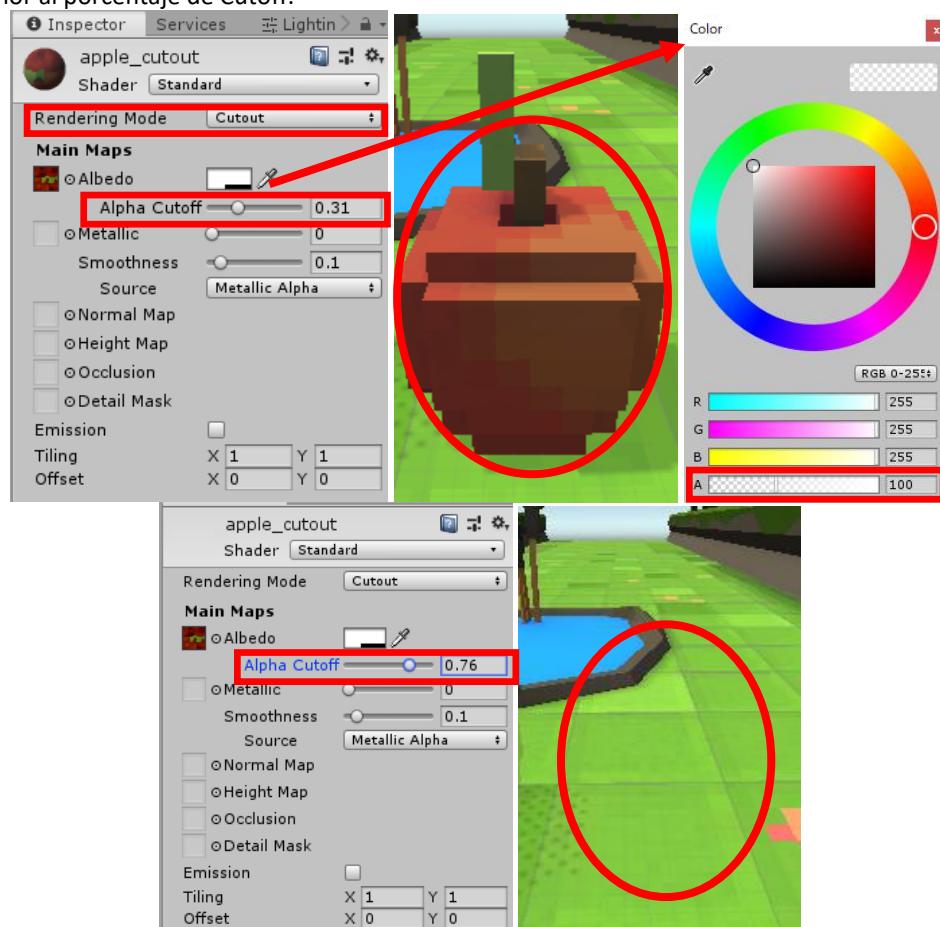


Fig N° 33. Aplicación del modo CutOff

El modo Fade vuelve el objeto completamente invisible solo si el valor alpha de color es 0

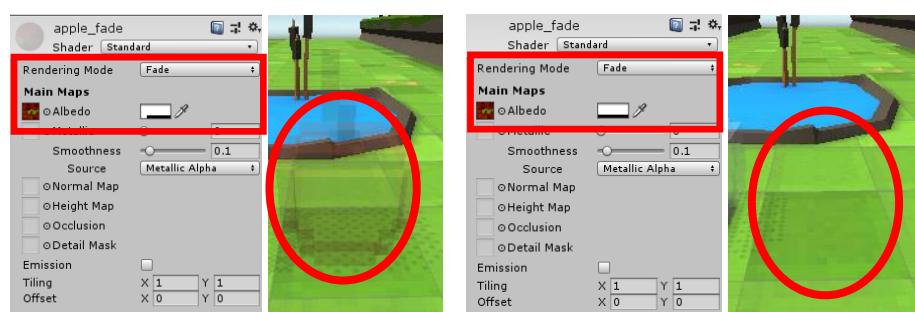


Fig N° 34. Aplicación del modo Fade

El modo Transparent simula la transparencia realista de objetos sin afectar reflexiones y luces



Fig N° 35. Aplicación del modo transparente

El modo opaco es el normal para objetos sólidos sin áreas de transparencia

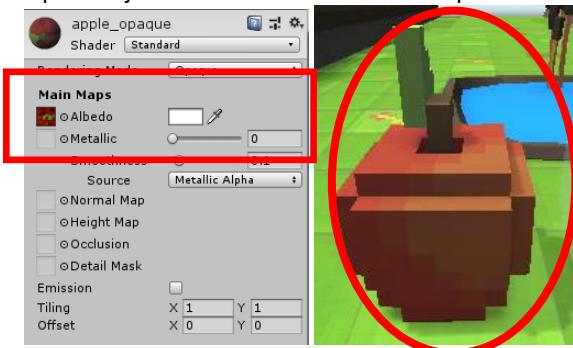


Fig N° 36. Aplicación del modo opaco

- Para finalizar este sub tema, veremos la creación de un tipo especial de material: SkyBox, que representa todo el entorno del cielo y el horizonte de la escena. Para lo cual creamos un nuevo Material y le ponemos un nombre referencial, y elegimos el shader: Skybox > Procedural

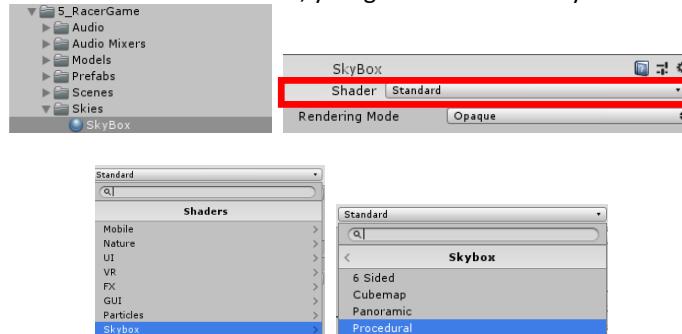


Fig N° 37. Creación de material SkyBox

- El material creado lo asignamos en la propiedad “Skybox Material” de la ventana de Iluminación, en el menú Window > Rendering > Lighting Settings.



Fig N° 38. Configuración de Luz

8. Podemos cambiar los valores de las propiedades del material Skybox hasta tener los resultados deseados

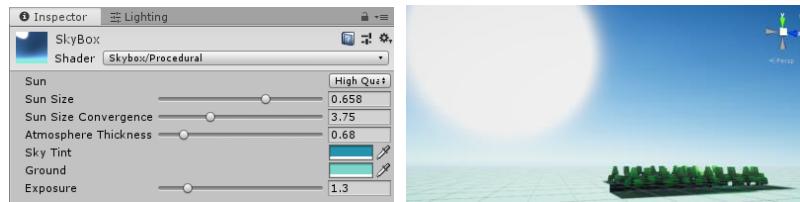


Fig N° 39. SkyBox finalizado.

Sub tema 1.4 Texturas

A continuación, vamos a ver la generación de texturas partiendo de la textura de color

- Al igual que el software de creación 3D, también hay varias alternativas, esta vez usaremos Shader Map, un software que cuenta con una versión gratuita no comercial, lo podemos descargar de <https://shadermapper.com/home/>. Al abrirlo solo tenemos que elegir el tipo de textura y arrastramos la textura desde el explorador de archivos hacia el ícono del centro.

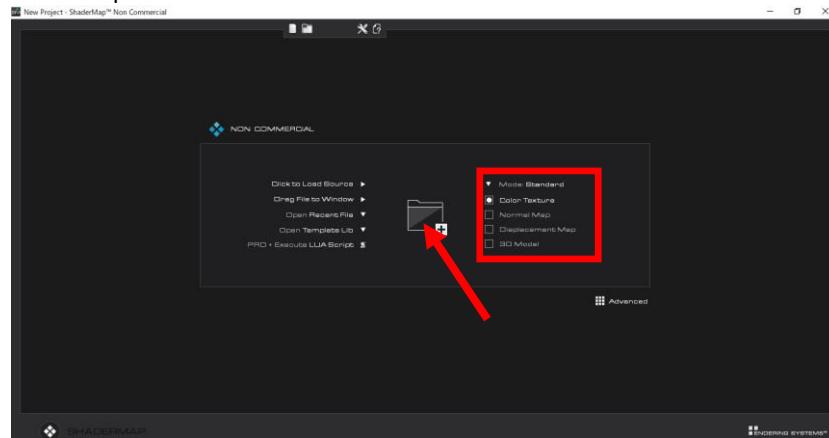


Fig N° 40. Iniciando en el uso de ShaderMap

- Shader map crea los siguientes mapas de forma automática, convenientemente para su uso con el shader standard de Unity: el mapa Normal, Specular (metallic), Ambient Occlusion (occlusion) y Displacement (height).

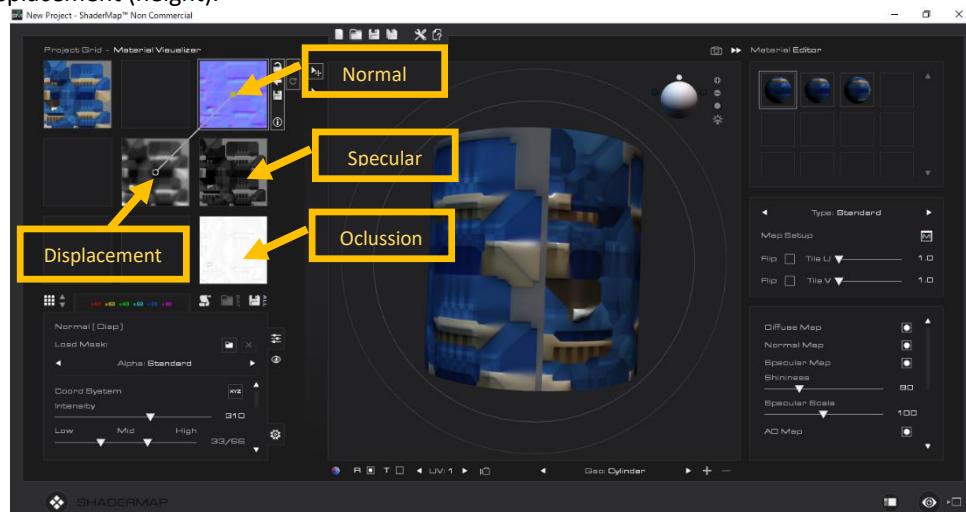


Fig N° 41. Iniciando la creación de mapas

3. Para guardar la textura generada seleccionamos con clic una de las generadas, presionamos el ícono de guardar y elegimos el destino.

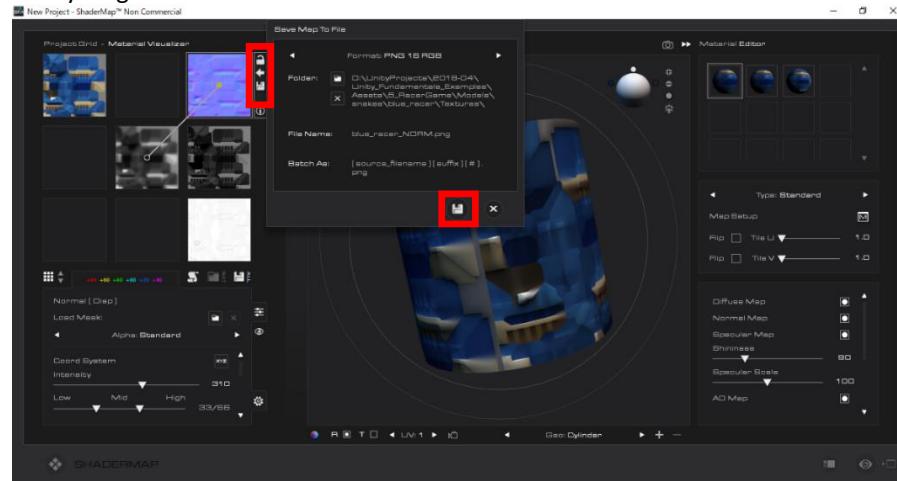


Fig N° 42. Proceso de guardado de mapas

4. Cabe destacar que en Unity solo el mapa de normales debe tener un tipo de textura especial entre las opciones de importación.

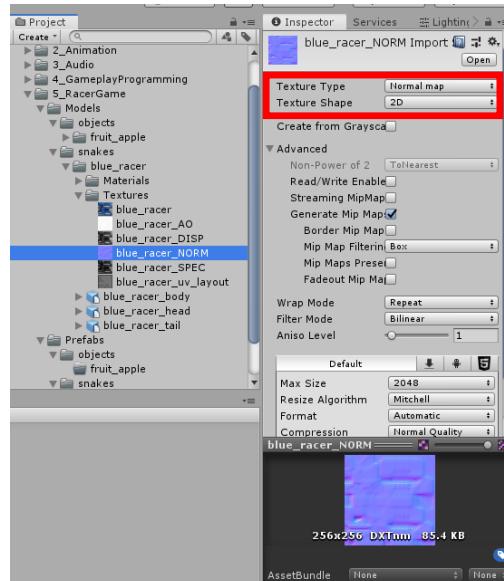


Fig N° 43. Asignando el tipo de textura

5. Una vez importado en el proyecto de Unity, podemos ponerlos en sus respectivos slots de texturas. Cabe destacar que en esta ocasión se está usando el shader "Standard (specular setup)", el cuál es muy parecido al Standard, la única diferencia es que el mapa Specular a diferencia del mapa Metallic acepta colores, permitiendo al usuario elegir el color tinte de absorción de luz. Nótese que en este caso el mapa Specular creado está en escalas de grises obteniendo un resultado muy parecido si usáramos un mapa Metallic.

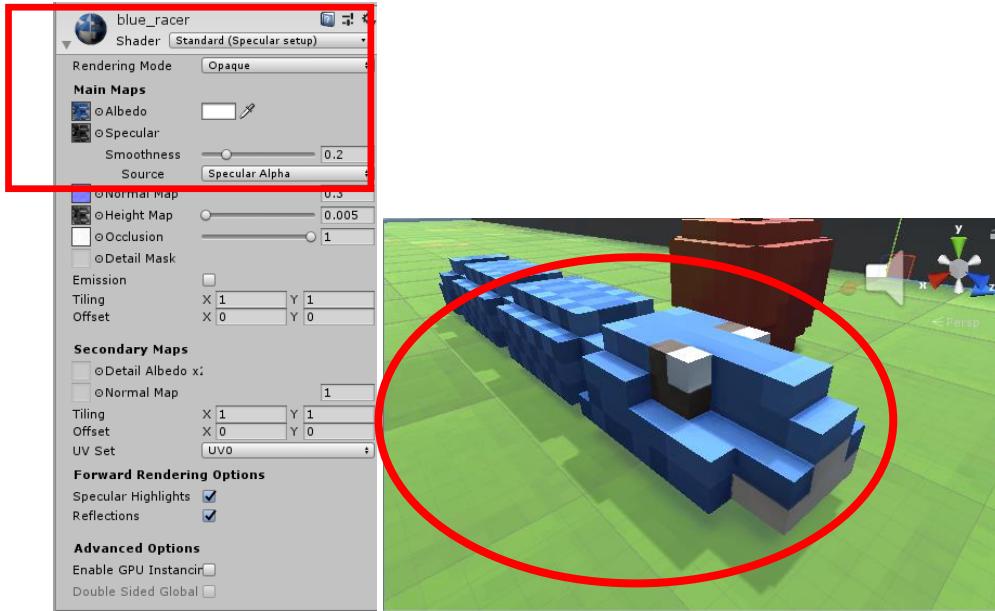


Fig N° 44. Mapa Specular

Sub tema 1.5 iluminación

En Unity se puede calcular iluminación en tiempo real y estática, que son calculadas y almacenadas previamente (baked lights).

1. Sólo los objetos señalados como “Lightmap Static” van a ser tomados en cuenta a la hora de generar el mapa de luces estático. Para un ejemplo posterior también se necesita que “Reflection Probe Static” este seleccionado. Esto se debe hacer para todos los objetos que no se muevan, por ejemplo, la mayor parte del terreno del mundo, rocas, etc.

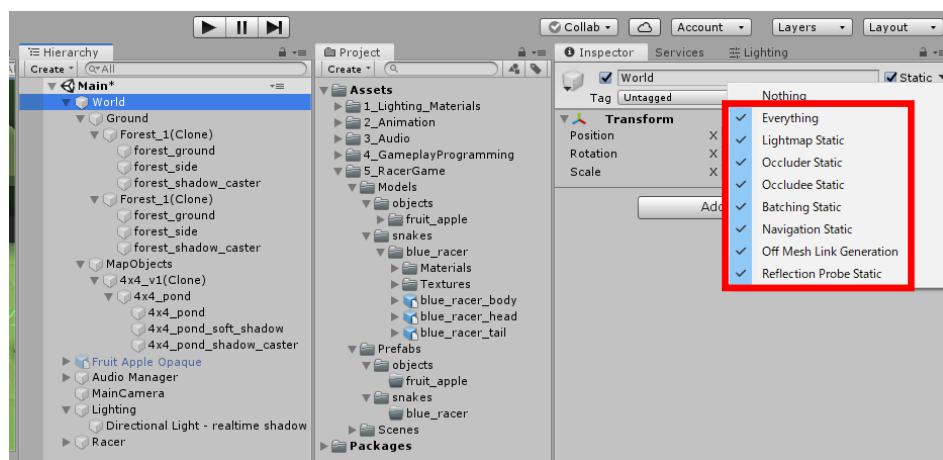
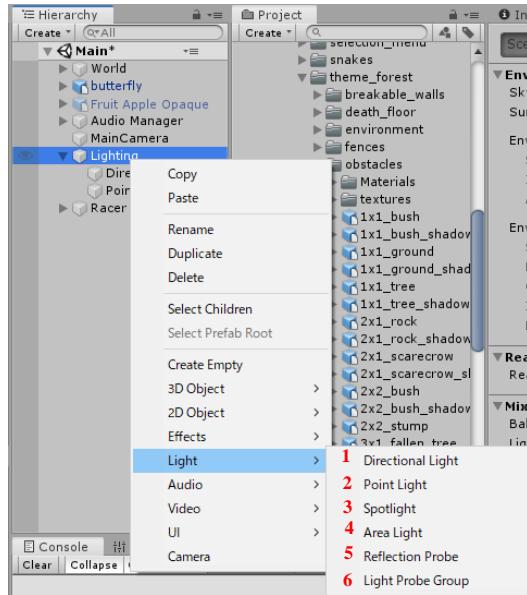


Fig N° 45. Asignación de objetos para Lightmap Static

2. Para crear un objeto de luz en Unity, vamos a la ventana de jerarquía > Clic derecho > Light > ...

Tenemos varias opciones descritas a continuación, todas tienen la propiedad de color e intensidad.



Elemento	Descripción
1	Directional Light , Generador de luz en una dirección, ignora posición.
2	Point Light , generador de luz en todas las direcciones, tiene un rango de alcance, como un foco
3	Spot Light , generador de luz en una dirección, tiene ángulo y rango de alcance, como una linterna
4	Area Light , generador de luz en forma rectangular, sólo disponible como mapa de luces, no en tiempo real
5	Reflection probe , generador de áreas de reflexiones, puede ser para objetos estáticos y dinámicos
6	Light Probe Group , guarda información de luz indirecta generada en el mapa de luces para enviarlo a objetos dinámicos

Fig N° 46. Asignación del tipo de iluminación

- Vamos a crear una directional light y un point light. Los íconos que se muestran ayudan a visualizar dirección en caso del directional light y rango en el caso del point light.

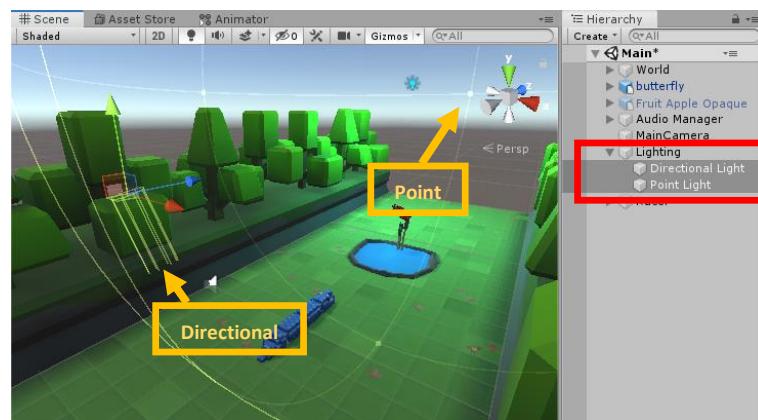


Fig N° 47. Creación de luz direccional y de punto

- Podemos modificar las propiedades hasta conseguir el resultado deseado. Cabe destacar que muchas veces sólo es necesario un generador de sombras, en este caso usaremos el Directional Light para generar las sombras de todos los objetos, además las dos luces están en modo Mixed, para que puedan afectar a objetos estáticos y dinámicos al mismo tiempo.

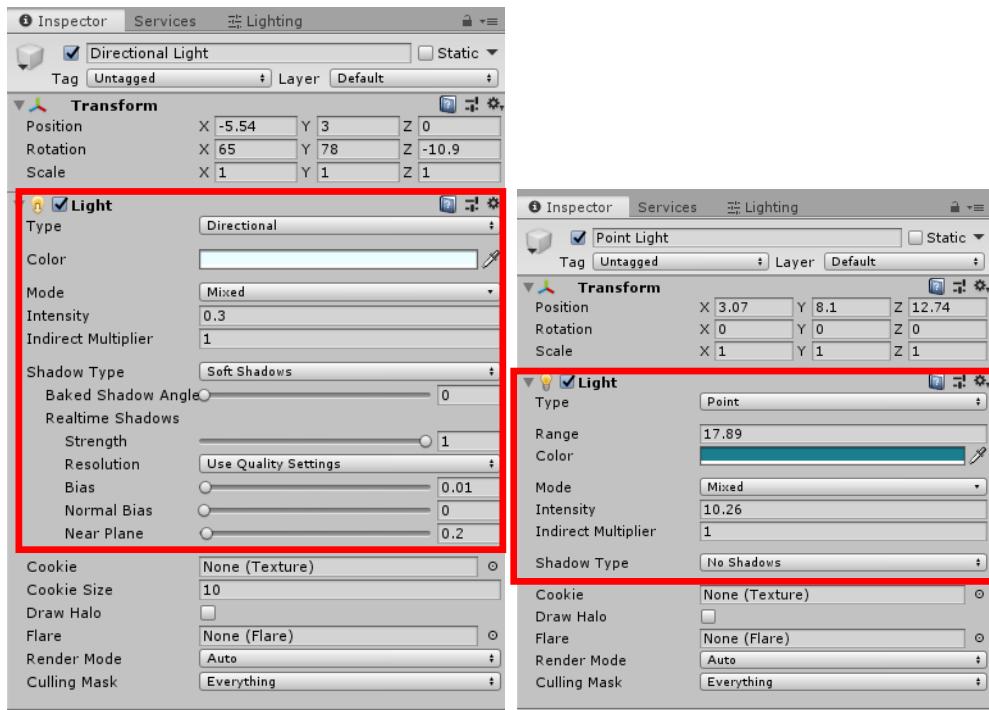


Fig N° 48. Parámetros de configuración de luz direccional y de punto

5. Ahora vamos a crear un Light **probe group** para almacenar la luz indirecta de los objetos estáticos y pasarlo a los objetos dinámicos. En la ventana de jerarquía > Light > Light Probe Group



Fig N° 49. Inicio de la definición de Light probe group

Podemos Agregar, Seleccionar, Eliminar y Duplicar usando las opciones del componente en la ventana de inspector. Las esferas probe se deben crear dentro del área de juego, es recomendable poner sólo en áreas críticas donde el jugador interactúe mucho más.

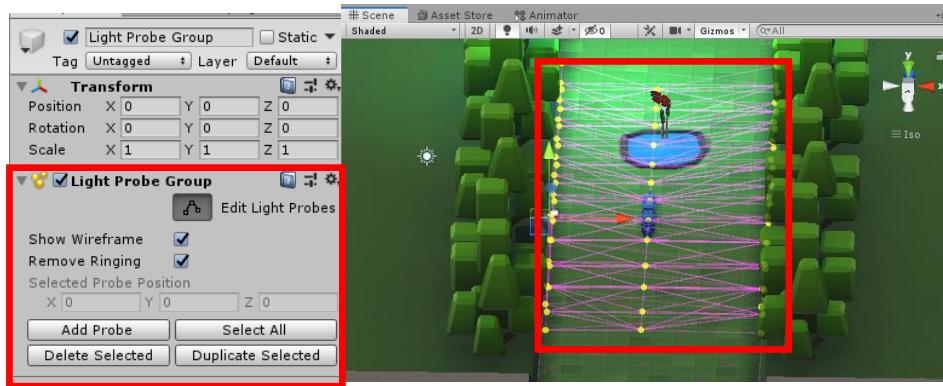


Fig N° 50. Configuración del Light Probe Group

6. A continuación, crearemos un área de reflexión en modo baked, para enviar información de color sólo de objetos estáticos a objetos con material con valores altos de metalicidad y suavidad (smoothness). De la misma forma, vamos a la ventana de jerarquía > Light > Reflection Probe, para crearlo. El área debe cubrir todos los objetos que queremos incluir. También podemos elegir modo en tiempo real, pero en esos casos es recomendable usar áreas más pequeñas.

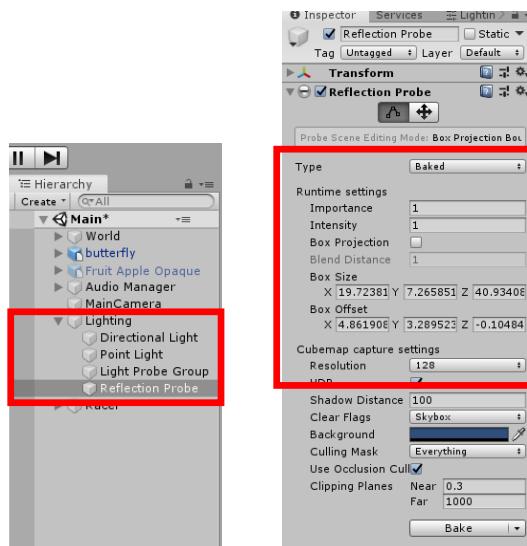


Fig N° 51. Configuracion de reflection Probe

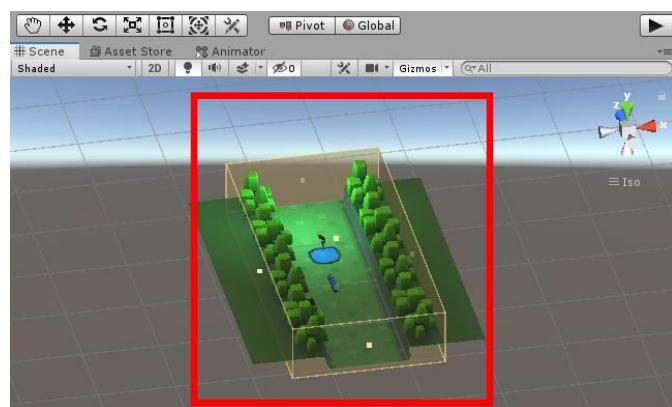


Fig N° 52. Resultado del reflection Probe

7. Finalmente vamos a generar el mapa de luces para lo cual abrimos dos ventanas en el menú Window > Rendering > Light Explorer y Light Settings.

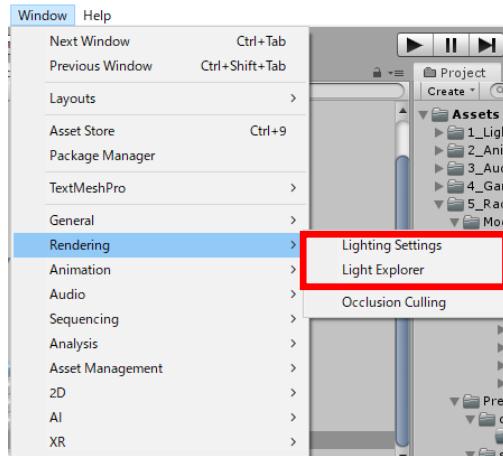


Fig N° 53. Generando Light Explorer y Light.

8. En la ventana Light Explorer podemos ver un resumen de todos los generadores de luces de la escena. Sirve de mucho cuando la escena es más compleja y se tiene un gran número de luces.

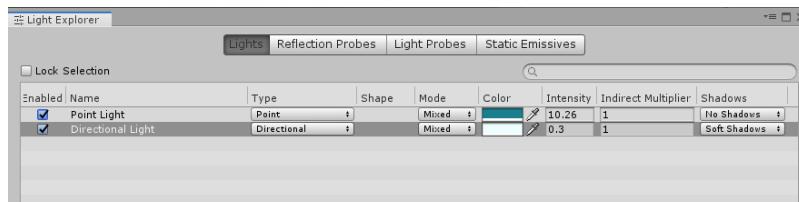


Fig N° 54. Resumen de los generadores de luces

9. En la ventana de Lighting Settings vamos a generar el mapa de luces, teniendo en cuenta algunas propiedades.

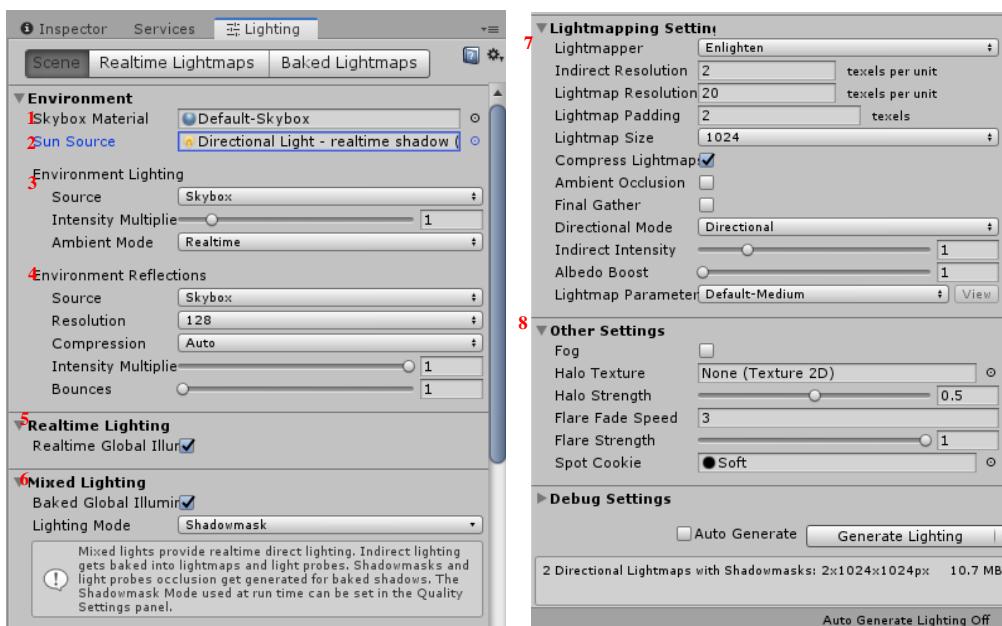


Fig N° 55. Generación del mapa de luces



Elemento	Descripción
1	El material que representa el cielo, lo que se ve detrás de la escena en el horizonte
2	La luz direccional que representa el sol cuando se usa un material de skybox procedural
3	Estas propiedades definen la luz que viene de un entorno lejano, la luz por defecto cuando no hay ninguna luz en la escena
4	Estas propiedades definen reflexiones globales que se generan junto con los Reflection Probe (sólo en tipo baked)
5	Define el uso de Iluminación Global en tiempo real, es decir trata de calcular la luz indirecta cuando el multiplicador (propiedad de las luces) en la escena es mayor a 0
6	Define el modo de horneado (baking) de luces directas e indirectas para luces mixtas en la escena (modo mixed)
7	Propiedades del horneado de luces por el backend de creación de mapas de luz: Enlighten y Progressive . La resolución del mapa de luces determina en mayor parte la calidad y duración del bake. (Por defecto el valor es 40 texels por unidad)
8	Propiedades para niebla, halo, llamarada y sombra generada por textura

10. Finalmente presionamos el botón Generate Lighting para generar el mapa de luces. El tiempo que toma dependerá del tamaño de la escena, la cantidad de luces, la resolución de luces indirecta y directa y la capacidad de la computadora.

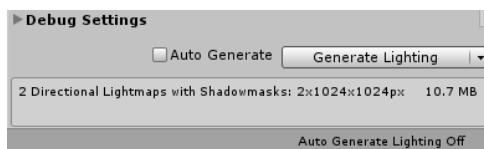


Fig N° 56. Generación del mapa de luces

Este es el resultado después de que el mapa de luces ha sido generado:

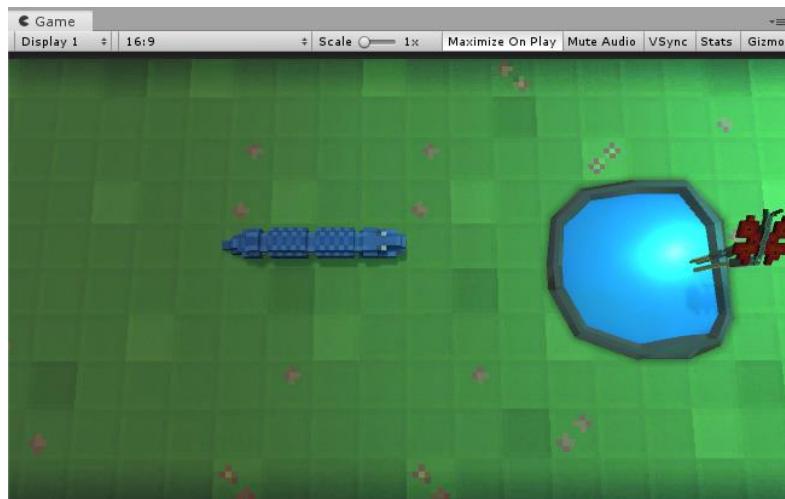


Fig N° 57. Resultado final del mapa de luces generado

Tema n.º 2: Animaciones

Sub tema 2.1 Animación

Vamos a utilizar el sistema de animación que viene con Unity, es muy completo y fácil de usar. Sirve para la mayoría de los casos cuando se necesita una animación que no cambia, por ejemplo, cuando siempre se transforma de un punto a otro predefinido.

1. En primer lugar, necesitamos un AnimatorController , elegir gameobject > Create > Animator Controller.
2. En este caso le pusimos el nombre de “Butterfly”.

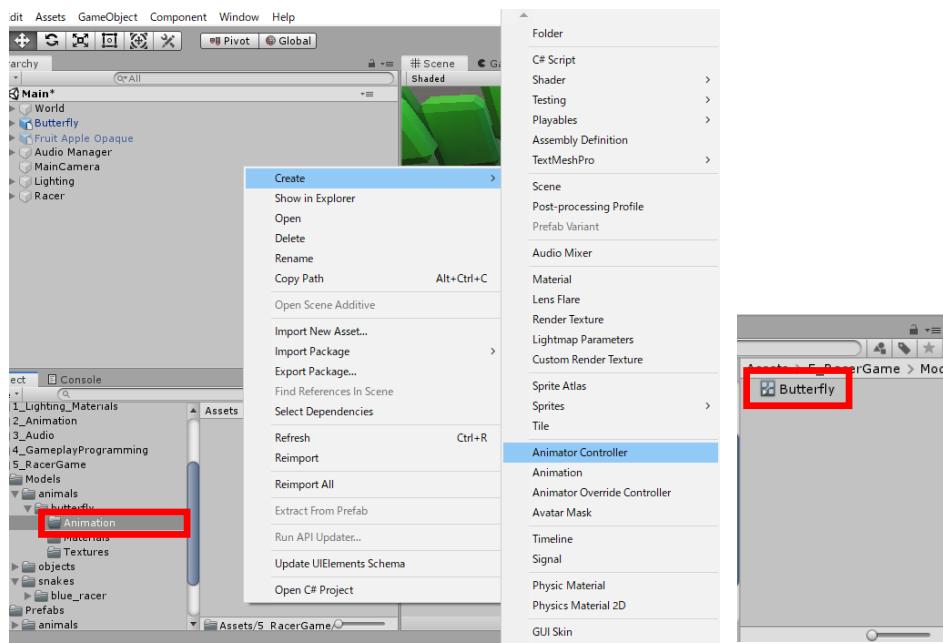


Fig N° 58. Creación de un animator controller

3. Una vez creado, agregamos el componente Animator al objeto que queremos animar y arrastramos el asset creado (Animator Controller) al campo en el componente.

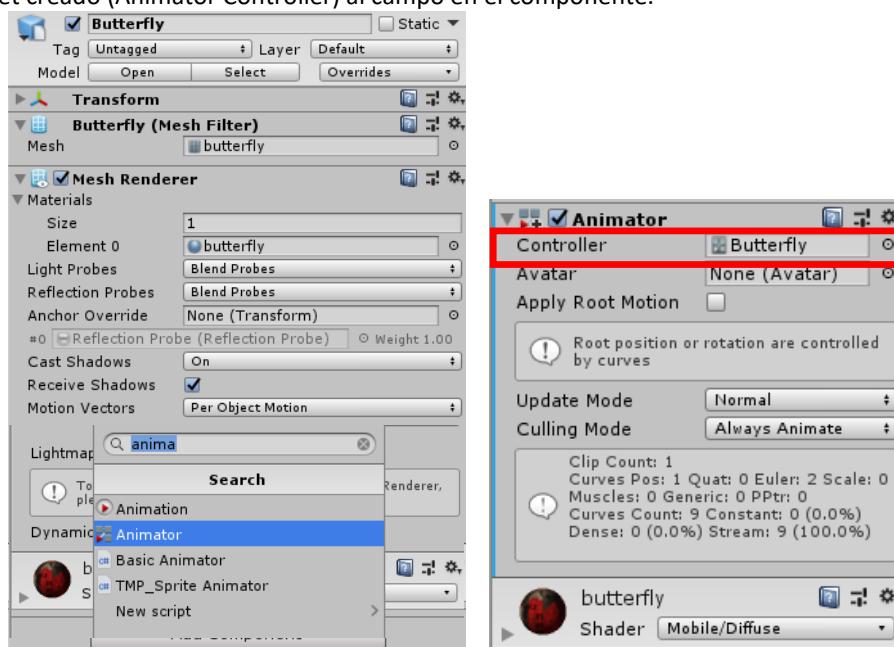


Fig N° 59. Asignacion de un animator controller a un objeto

4. Para empezar a animar primero necesitamos crear un archivo de animación, para el cual abrimos la

ventana Animator en el menú Windows > Animation > Animator, o haciendo doble clic al asset del animator controller creado previamente.

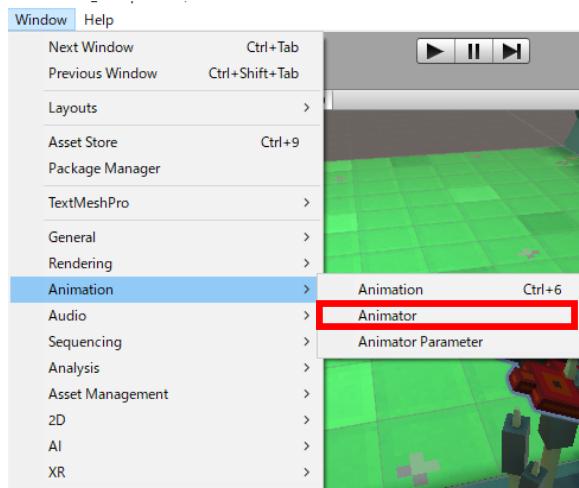
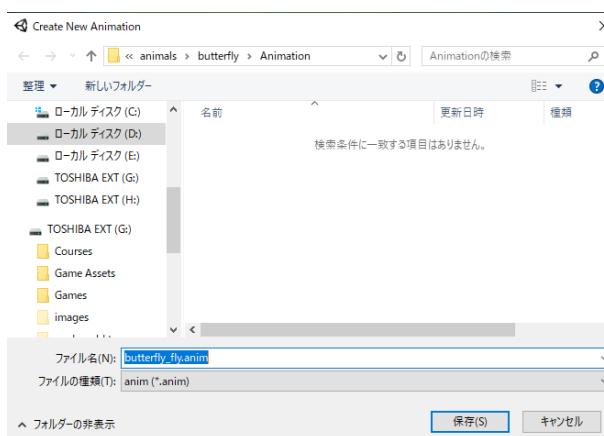
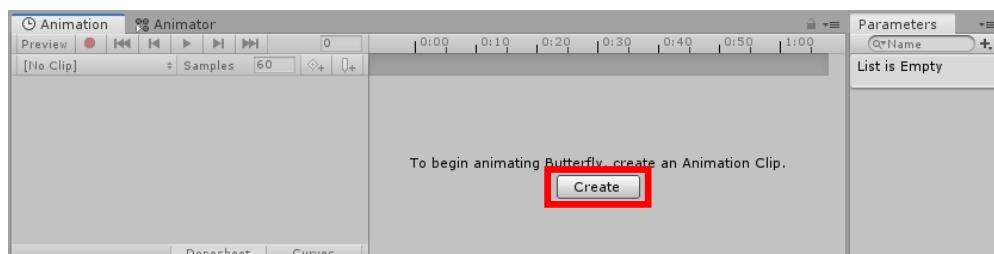
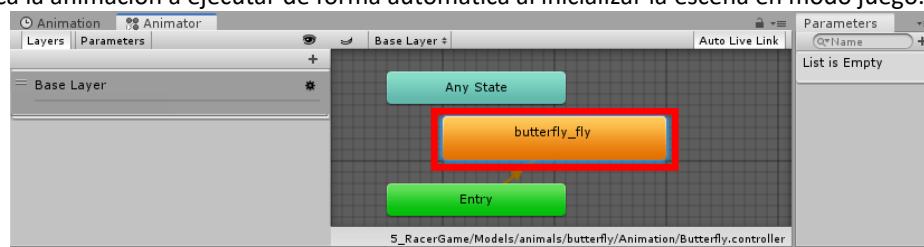


Fig N° 60. Creación del archivo de animación

- En esta ventana presionamos la opción “Create”, elegimos una ubicación dentro del proyecto y le damos un nombre referencial. En este caso se pudeo “butterfly_fly”, ya que esta animación representará la mariposa en vuelo.



- Una vez creada la podemos visualizar en la ventana Animator sombreado de color naranja, que indica la animación a ejecutar de forma automática al inicializar la escena en modo juego.



- Para poder animar necesitamos abrir la ventana de animación, en el menú Windows > Animation >

Animation.

Las animaciones consisten en cambio de valores de las propiedades como posición, rotación, escala, color, etc., en el tiempo. Para comenzar a animar presionamos la opción "Grabar" y modificamos valores de las propiedades en el inspector. En el ejemplo se modificó el valor de la propiedad "posición" del componente Transform.

TIP: Es mejor agregar el componente Animator en el padre según la jerarquía de los objetos a animar, para que permita mover al objeto padre a cualquier parte.

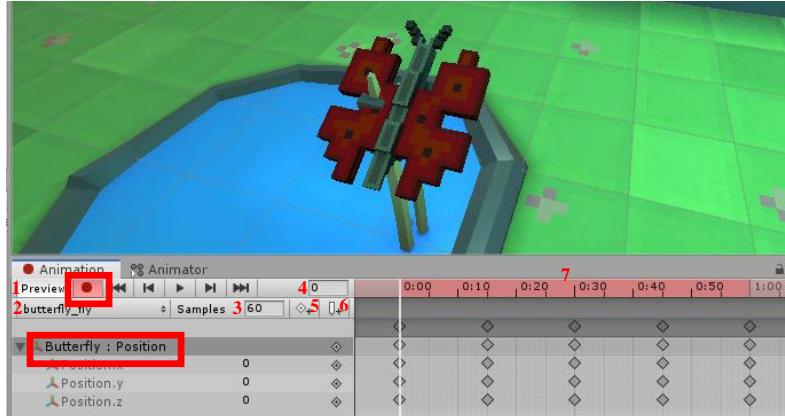


Fig N° 61. Elementos de animación

Elemento	Descripción
1	Opciones de control de ejecución de la animación. "Grabar" es el botón de color rojo.
2	Clip de animación actual
3	Número de cuadros por segundo a la que corre la animación
4	Número de cuadro actual
5	Opción para agregar un key-frame en la línea de tiempo
6	Opción para agregar un evento en el cuadro actual
7	Línea de tiempo

8. Seguimos modificando hasta obtener la animación deseada.

TIP: Para crear una animación infinita, se tiene que hacer que los valores de las propiedades del primer cuadro y el último sean iguales. Se puede copiar y pegar keyframes usando Ctrl + C y Ctrl + V, una vez las seleccionamos con clic izquierdo.

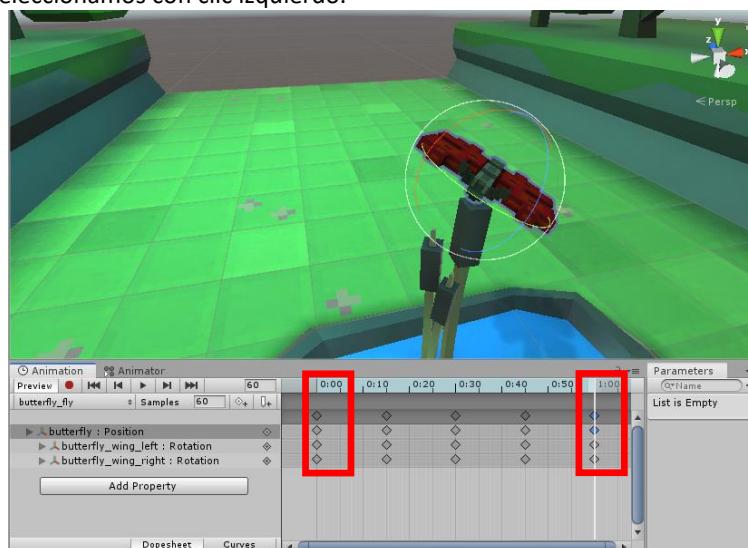


Fig N° 62. Escala de tiempo de la animación

Enlace al ejemplo final: https://www.youtube.com/watch?v=u_JVD3VtRrk&feature=youtu.be

Sub tema 2.2 Concurrency

Se define concurrencia como el procesamiento en paralelo de varias tareas por parte de los cores del CPU. Un claro ejemplo es el sonido, podemos ejecutar varios archivos de audio al mismo tiempo. A continuación, vamos a ver un ejemplo administrando audio en unity.

Sub tema 2.3 Sonido

A continuación, vamos a utilizar el Audio Mixer, que es un asset muy poderoso para el control completo de audio en Unity.

1. En primer lugar, lo creamos estando en la ventana de proyecto > Clic derecho > Create > Audio Mixer, y le ponemos un nombre referencial.

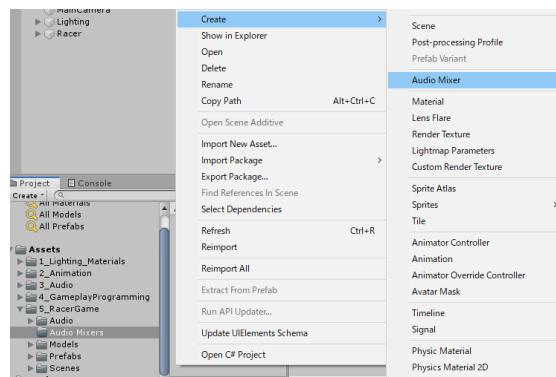


Fig N° 63. Creación de un Audio Mixer

2. Abrimos la ventana de Audio Mixer, luego elegimos el menú Window > Audio > Audio Mixer, o con la tecla rápida Ctrl + 8.

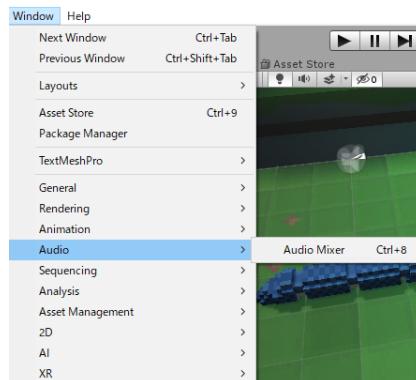


Fig N° 64. Inicio de la creación de audio Mixer

3. Esta es la ventana Audio Mixer. En ella creamos tres grupos: música, ambiente y efectos de sonido. Por defecto el grupo Master es el padre de todos los grupos, es muy útil para disminuir o aumentar el volumen global.

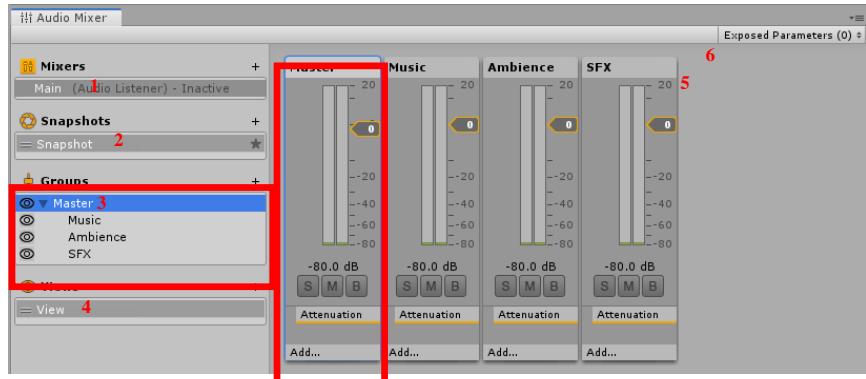


Fig N° 65. Audio mixer Unity 3D

Elemento	Descripción
1	Mixers , aquí se listan todos los audios mixers del proyecto, puede tener más de uno y un audio mixer puede estar incluido en otro, útil cuando se tienen demasiados grupos.
2	Snapshots , aquí se listan capturas de valores de los diversos efectos de los grupos. Por ejemplo, puedo tener uno para juego normal, y otro cuando está pausado (valor de attenuation (volumen) reducido)
3	Groups , aquí se listan todos los grupos de audios del audio mixer, los grupos son los que controlan la salida de los Audio Sources.
4	Views , aquí se listan las diferentes vistas de los grupos. Por ejemplo, puedo tener una vista donde sólo se muestran los grupos relacionados con música. Útil cuando hay demasiados grupos en un solo Audio Mixer.
5	Aquí se muestran los diferentes grupos y los efectos aplicados a estos. Atenuación o volumen, medido en decibeles (dB) es el efecto predeterminado en todos los grupos. También se cuenta con opciones de control S (Solo), M (Mute) y B (bypass effect o ignorar efectos, excepto attenuation)
6	Aquí se listan los parámetros expuestos para utilizarlos en un script de código. Se pueden exponer parámetros con el clic derecho del mouse en las propiedades del inspector

- Para agregar sonido primero necesitamos los archivos de audio, hay disponibles gratuitos en el Asset Store de Unity, o también se puede crear usando un software de creación de sonidos como BoscaCeoil (<https://boscaceoil.net/>), que es gratuito, el uso de este software va más allá del alcance de este manual, pero hay disponibles muchos tutoriales en línea. Una vez descargados o creados, necesitamos agregar un Audio Manager en la jerarquía, donde irán todos nuestros orígenes de audio (Audio Source).

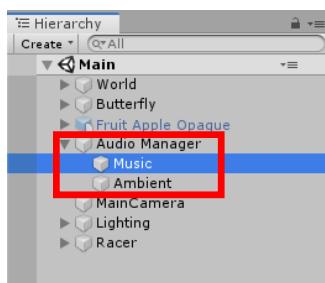
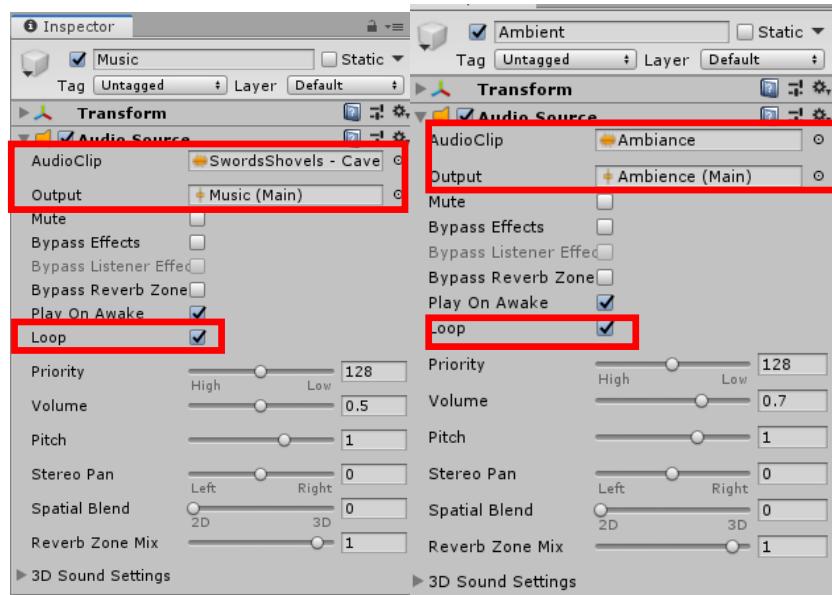


Fig N° 66. Audio Manager en Unity 3D

- En este ejemplo tenemos dos Game Objects debajo del Audio Manager, uno para música de juego y el otro para sonido de ambiente. Necesitamos agregar los componentes Audio Source en cada uno de ellos, seleccionar el clip de audio y el grupo del audio mixer respectivo (Music / Ambience), propiedad "output", y finalmente en caso de ser un sonido repetitivo, seleccionamos la propiedad "loop".



Enlace al ejemplo final: <https://www.youtube.com/watch?v=0YaewV4Ke2A&feature=youtu.be>

Tema n.º 3: Efectos de cámara e iluminación

Sub tema 3.1 Interfaces visuales

En esta parte vamos a crear un menú simple para cargar otra escena.

1. En primer necesitamos crear otra escena, usando la opción en la ventana de proyecto, Clic derecho > Create > Scene. Esta escena se puede llamar “MainMenu”.

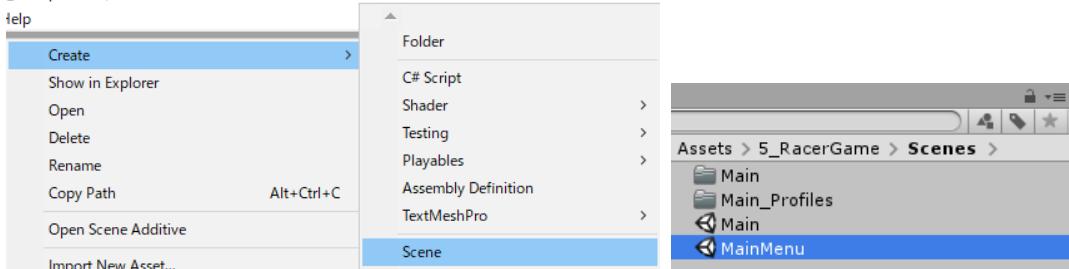


Fig N° 67. Inicio de un menu simple

2. En Unity, la interfaz necesita ir dentro del GameObject “Canvas”, por lo tanto lo creamos haciendo clic derecho en la ventana de jerarquía > UI > Canvas.

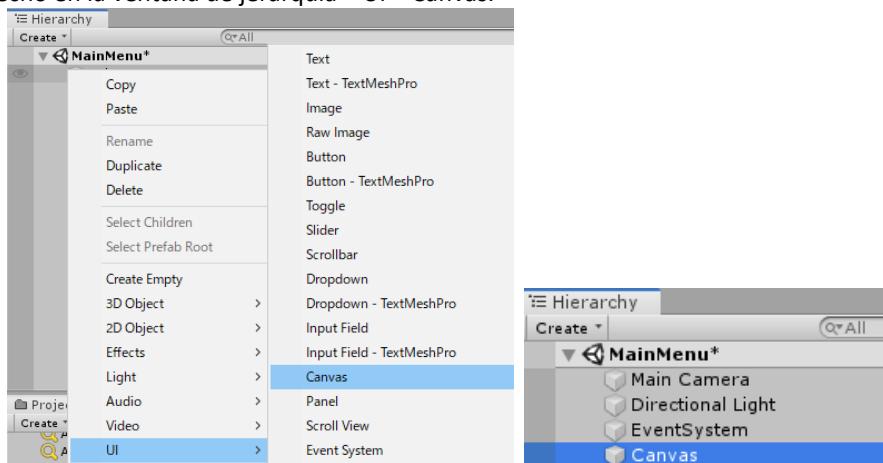


Fig N° 68. Creacion de objeto canvas.

3. El canvas por defecto se necesita editar en espacio 2D, pero en este ejemplo vamos a usar un canvas con modo de renderizado “Cámara” que nos permita aprovechar los objetos del mundo virtual, por ejemplo, cuando queremos agregar efectos de partículas directamente en la interfaz.

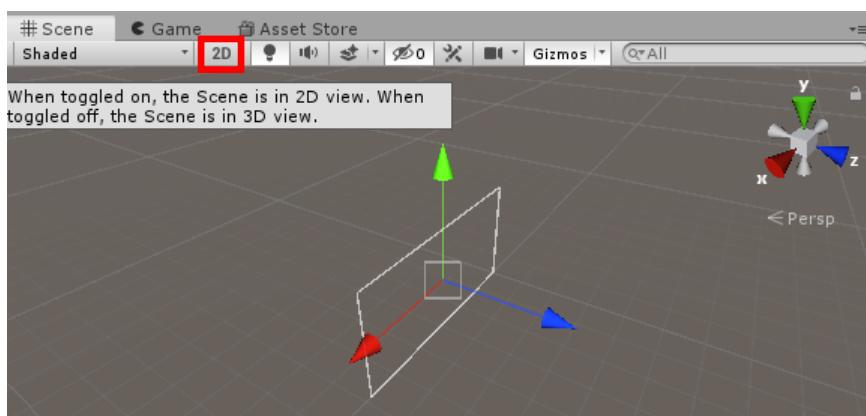


Fig N° 69. Canvas con modo rerenderizado

4. El canvas tiene tres modos de renderizado: (1) Screen Space overlay, que renderiza el canvas en la pantalla, siempre activo. (2) Screen Space camera, que renderiza el canvas usando una cámara

como referencia. (3) World Space, que renderiza el canvas como cualquier game object, teniendo en cuenta los ejes XYZ. Los tres modos usan pixeles como medida.

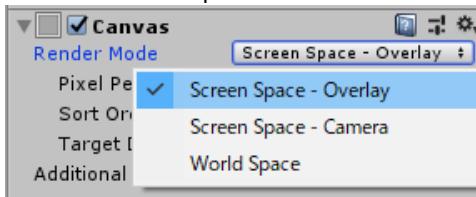


Fig N° 70. Modos de rerenderizado

- Cambiamos el modo de renderizado a Screen Space – Camera y elegimos el Main Camera de la escena como Render Camera. Las opciones de orden y ordenamiento son muy importantes en juegos 2D, en este ejemplo lo vamos a ignorar.

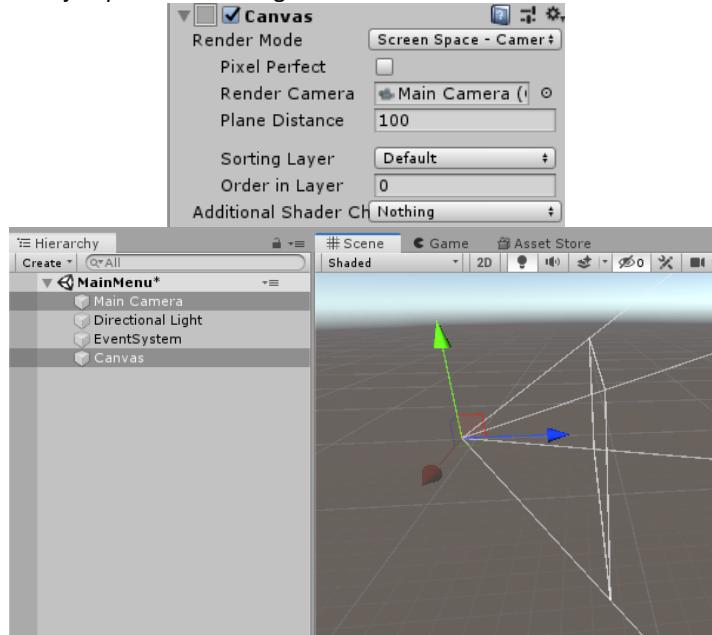


Fig N° 71. Cambio del modo de rerenderizado

- En este ejemplo vamos a usar un pedazo del mundo como fondo para el menú principal (opcional), pero de la misma forma podemos usar una imagen, color de fondo de la cámara o el mismo skybox como fondo.

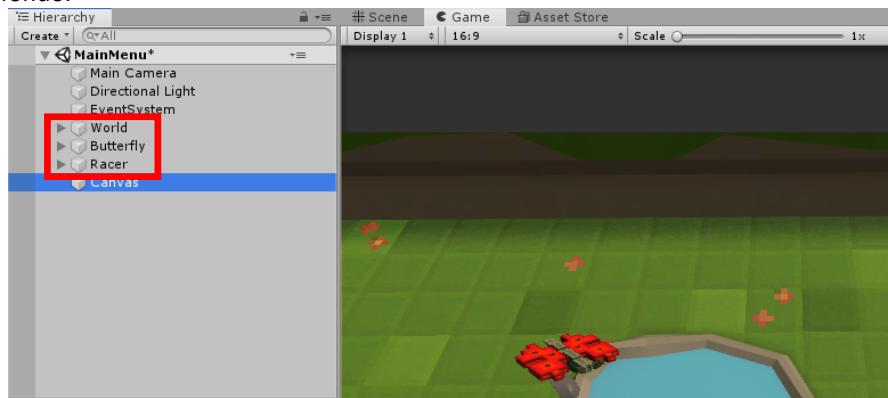


Fig N° 72. Cambiando el fondo del menú

- Para cargar la otra escena, usaremos un botón. Para crearlo, estando en la ventana de jerarquía > Clic derecho > UI > Button. Nótese que debajo en la jerarquía, el botón tiene un componente de texto.

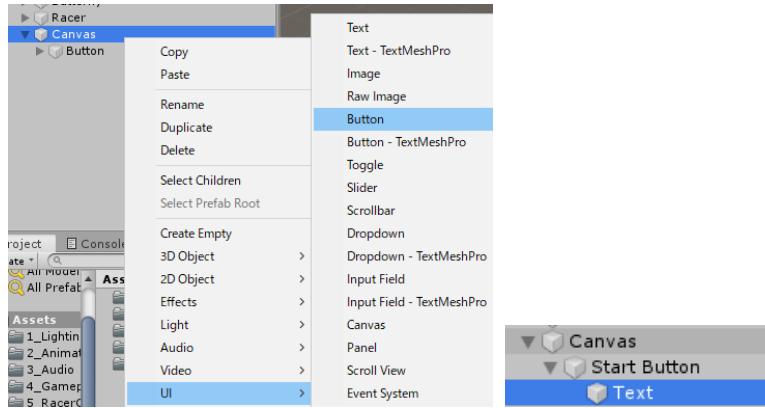


Fig N° 73. Agregando un componente de texto al menú

- Para ver correctamente en pantalla el botón creado, necesitamos reducir el valor de Plane Distance medido en unidades de Unity, el cual es la distancia de separación entre la cámara y el canvas.



Fig N° 74. Modificación del Plane Distance

- En el GameObject del canvas hay a un componente llamado Canvas Scaler, este componente mantiene el tamaño del canvas cuando el tamaño de la pantalla cambia en juego. (1) Constant Pixel Size, mantiene el mismo tamaño en pixeles. (2) Scale with Screen Size, escala los elementos de interfaz acorde al tamaño de la pantalla. (3) Constant Physical Size, mantiene un tamaño constante teniendo en cuenta un dispositivo físico como un smartphone. Para este ejemplo usaremos la segunda opción para que permita a los elementos de interfaz escalarse y mantener un tamaño normal si el dispositivo es muy grande o muy pequeño.

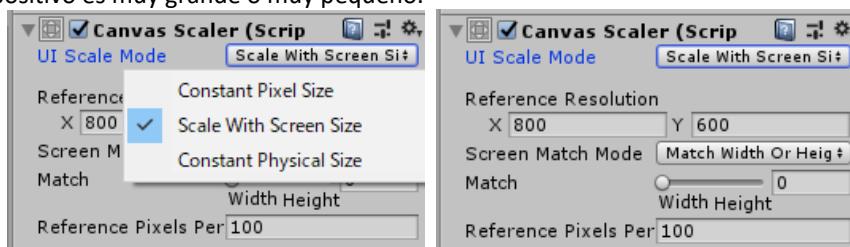


Fig N° 75. Uso del canvas Scaler

- Podemos modificar los colores de interacción del botón y el componente de texto.

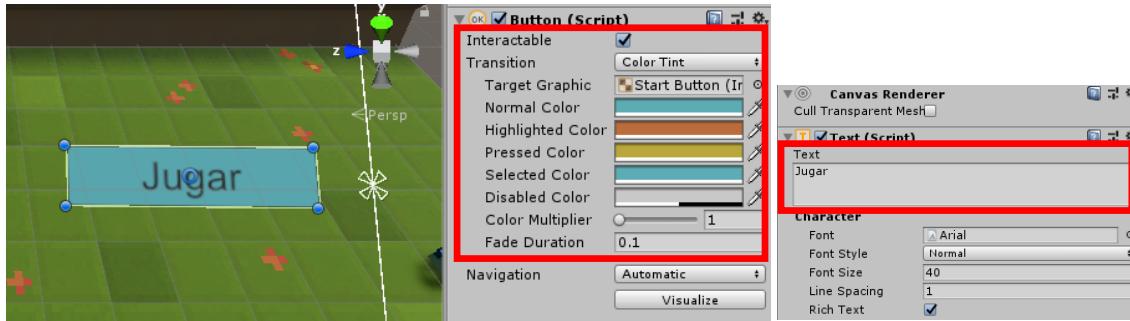


Fig N° 76. Modificación de propiedades del botón

11. Para que el botón pueda actuar como lo planeado necesitamos crear un script de código con la función de carga de escenas. Vamos a crear un script GameManager.cs. Nótese que Unity tiene un ícono especial para este script, siempre y cuando su nombre sea exactamente “GameManager”, para identificarlo fácilmente dentro del proyecto, este es el script controlador, como un árbitro del juego.

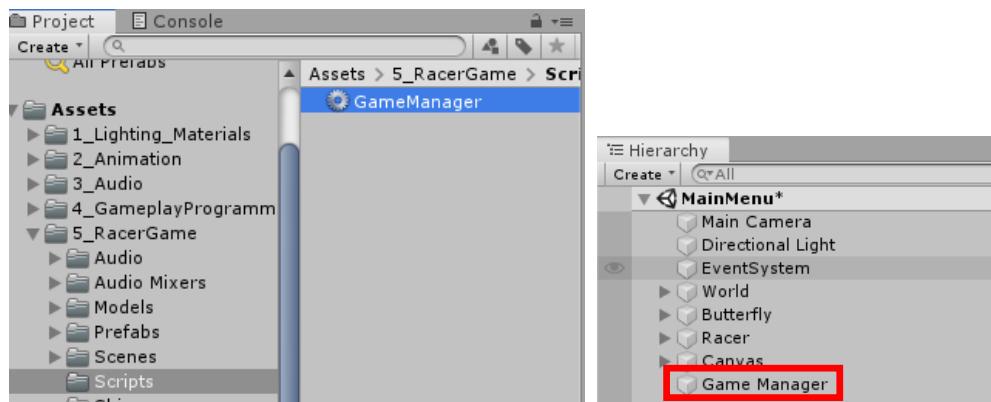


Fig N° 77. Creación de un game Manager para el botón

12. Antes de editar el script tenemos que agregar las escenas en la ventana “Builds Settings...”. Lo abrimos con el menú File > Build Settings...

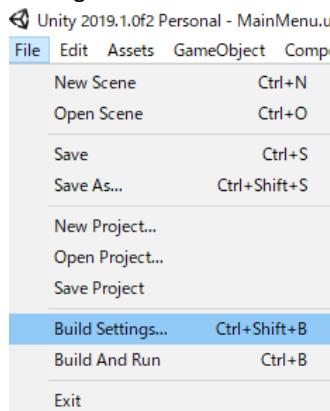


Fig N° 78. Agregar Ventanas al Builds Settings

Podemos usar el nombre o el índice, en este ejemplo usaremos el índice, así que tenemos que recordar el índice de la escena que queremos cargar con el botón, en este caso es el 1.

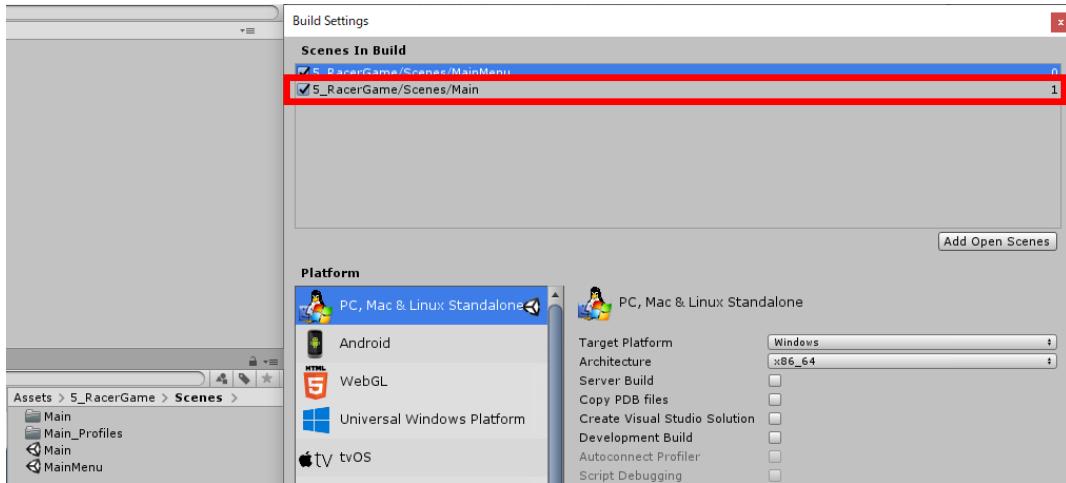


Fig N° 79. Asignando la escena a cargar por el botón.

13. Este es el script, sólo tiene un método y una línea de código en él. Este script carga una escena por índice (el argumento o parámetro “buildIndex”) de modo “single”, que significa que destruye todas las escenas actuales y carga la nueva escena. Además, se está usando el método estático LoadSceneAsync, que carga la escena usando otro hilo de procesamiento, útil cuando la escena es compleja y/o queremos mostrar el porcentaje de carga. Nótese que necesitamos el namespace “UnityEngine.SceneManagement”.

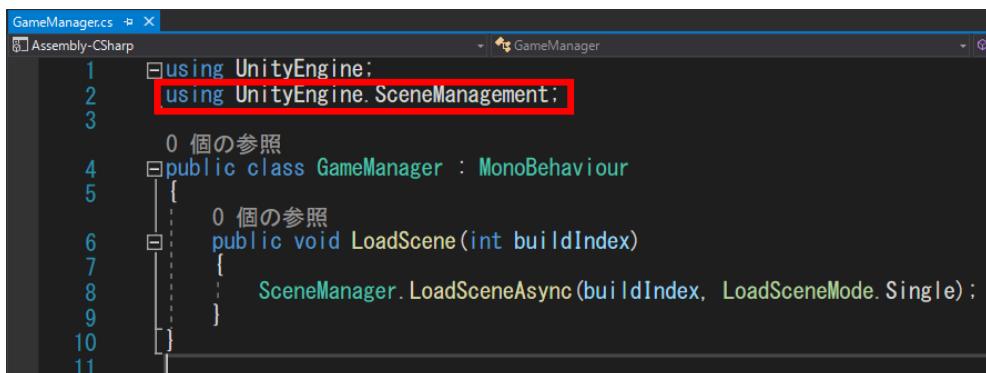


Fig N° 80. Script inicial de LoadScene

14. Finalmente, agregamos una acción al evento OnClick del botón. Arrastramos el GameManager al campo requerido y escogemos la función a llamar cuando el botón sea clickeado, en este caso la función es LoadScene(int) del MonoBehaviour GameManager. No olvidemos poner como parámetro 1, que es el índice de la escena que queremos cargar en la ventana Builds Settings.



Fig N° 81. Asignacion de evento al boton

Sub tema 3.2 Cámara

Unity tiene un paquete de efectos de post procesado aplicados en tiempo real a la cámara después de renderizar todos los objetos en pantalla. Veamos un ejemplo de cómo instalarlo y usarlo en Unity.

- Para instalarlo vamos al menú Window > Package Manager

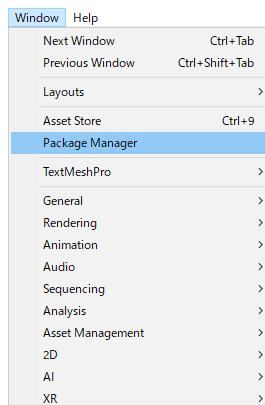


Fig N° 82. Aplicación de efectos de cámara

- Una vez que carguen todos los paquetes buscamos “Post Processing” y lo instalamos.

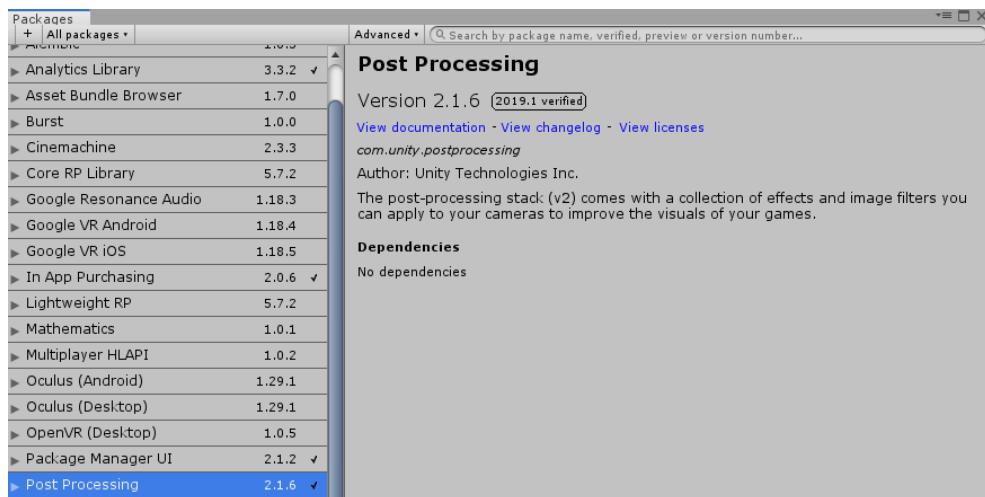


Fig N° 83. Instalación de PostProcesing

- Una vez instalado seleccionamos la Main Camera de la escena y agregamos el componente “Post Process Layer” y creamos el layer “PostProcessing” en la ventana de layers. Luego elegimos el layer creado como opción en la propiedad “Layer”. Este paso es muy importante, ya que esta cámara sólo activará los efectos de los volúmenes que tengan este layer.

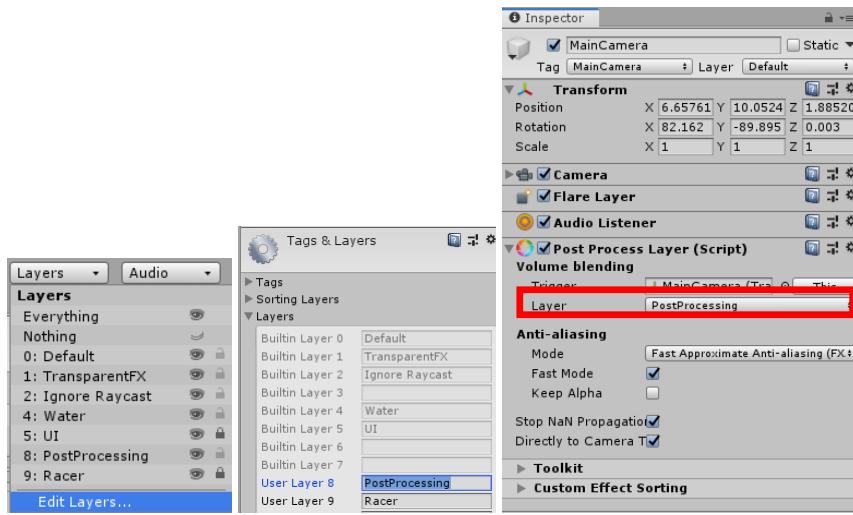


Fig N° 84. Agregación del componente Post Process Layer

4. Luego creamos un nuevo Game Object en la ventana jerarquía con el nombre “PostProcessing Volume” y le agregamos el componente “PostProcess Volume”. No olvidemos cambiar el layer al previamente creado. Elegimos la opción “IsGlobal” para aplicar este efecto de forma global indiferente a su posición. Y creamos un nuevo perfil de post procesado presionando la opción “New”.

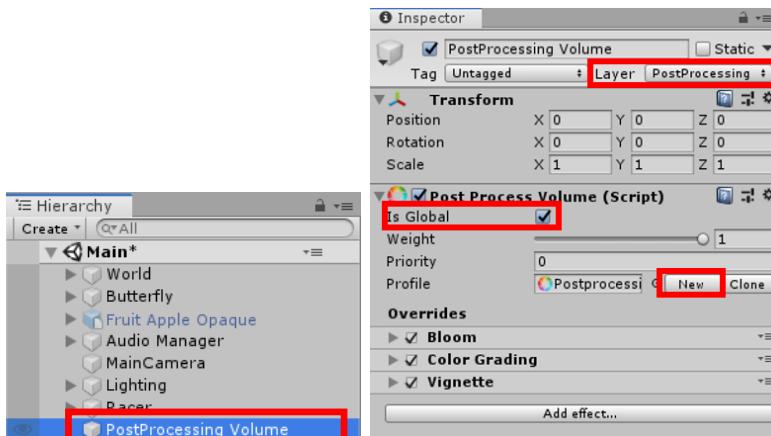


Fig N° 85. Agregando el componente PostProcess Volume

5. Al presionar esa opción Unity nos crea por defecto un nuevo asset llamado PostProcessing Volume Profile dentro de una nueva carpeta al lado del asset de la escena actual.

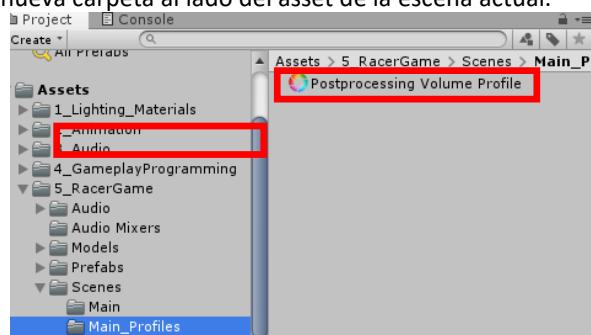


Fig N° 86. PostProcessing Volume Profile

6. Hay varios efectos disponibles, vamos a revisar tres de ellos. Para agregar un efecto presionamos la opción “Add effect...”



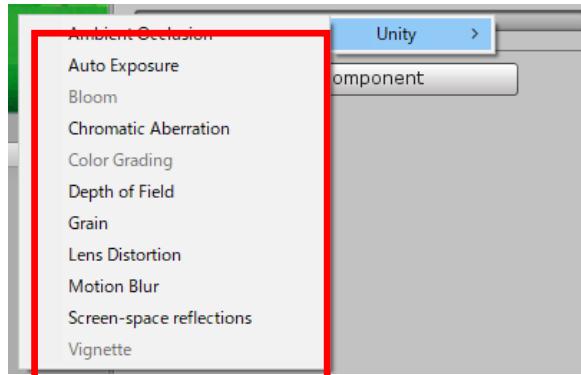


Fig N° 87. Efectos disponibles

7. **Efecto Bloom:** Produce franjas que se extienden desde los bordes de las áreas más brillantes de una imagen, lo cual contribuye a la ilusión de luces extremadamente brillantes. Más detalles: [Unity](#) [Github](#) [Wiki](#)

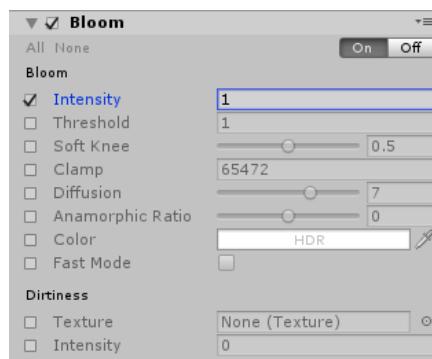


Fig N° 88. Propiedades del efecto bloom

8. **Efecto Color Grading:** Es el efecto de alterar o corregir la intensidad de luz de la imagen final. Es igual que aplicar filtros en programas de edición de imágenes como Krita o Photoshop. Más detalles: [Unity](#) [Github](#) [Wiki](#)

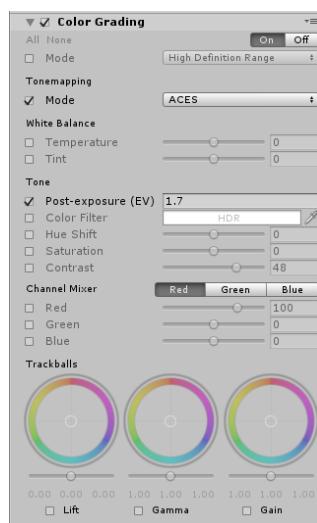


Fig N° 89. Efectos de Color Grading

9. **Efecto Vignette:** Viñeteado es el término usado para el oscurecimiento y/o desaturado hacia los bordes de una imagen en comparación con el centro. Se utiliza a menudo para efectos artísticos, como para enfocar el centro de una imagen. Más

detalles: [Unity Github Wiki](#)



Fig N° 90. Propiedades del efecto Vignette

- Este es el resultado final después de aplicar los efectos de postprocesado en este ejemplo, pero puedes seguir editando los valores, o usar otros efectos, hasta conseguir el resultado deseado.

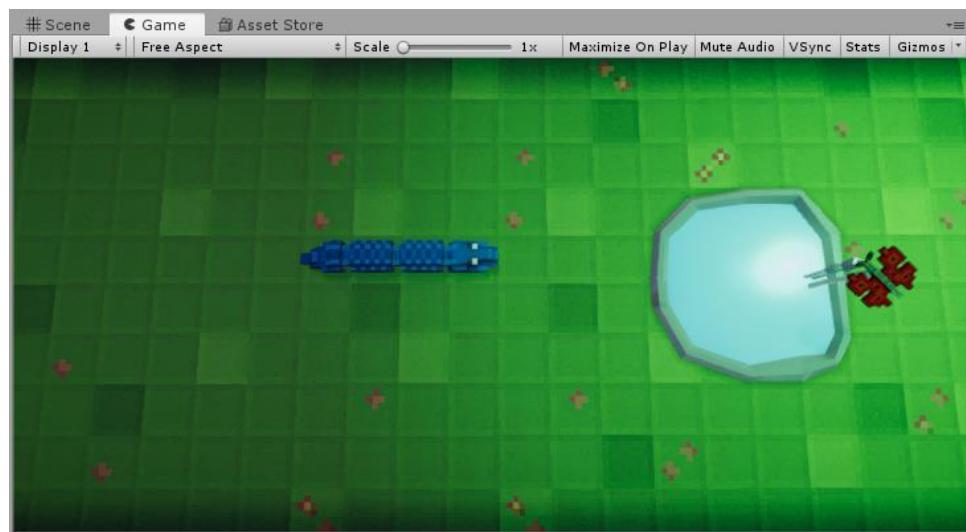


Fig N° 91. Resultado final aplicado al ejemplo



Glosario de la Unidad 2

Renderizar: verbo que expresa el dibujar los pixeles en la pantalla por el GPU usando el shader.

Skybox: se refiere al material y efectos aplicados en el elemento Cielo de las aplicaciones de realidad virtual.

Concurrencia : En el área de computación se refiere al hecho de que se requiere procesar varias tareas a la vez, lo que conlleva a dos posibilidades, la primera que se aplique multitarea(se desarrollen por turnos secuenciales) la segunda: se utilice procesamiento en paralelo por parte de varios cores del CPU.

Material: Forma de aplicar color a un objeto tridimensional de forma regular y uniforme a lo largo del mismo. Ejemplo: una pared, piso.

Texturas: Forma de aplicar color a un objeto tridimensional de forma irregular y diversa a lo largo del mismo. Se utilizan para dar color a un objeto difundido. Colorear una manzana.

Extrusión: Proceso utilizado en el modelado el cual consiste en generar nuevos vértices, aristas y caras a partir de anteriores, algo así como "hacer crecer el modelo".

Normal: El vector normal es un concepto geométrico que se utiliza en Blender, dado un plano que tenga todos sus puntos en un espacio euclídeo de dos dimensiones existirían dos vectores normales, ambos en dirección perpendicular al plano pero en sentidos contrarios. Es como si nos imagináramos una hoja de papel sobre una mesa, el vector normal de la cara de la hoja que está hacia arriba apuntaría hacia el techo y el de la cara que está en contacto con la mesa apuntaría hacia el suelo.

Motor de Video juego: término que hace referencia a una serie de librerías de programación que permiten el diseño, la creación y la representación de un videojuego.

(Incorpore 8 términos como mínimo ordenados alfabéticamente, la definición considerada debe de estar correctamente referenciada con una cita en estilo APA)

Bibliografía de la Unidad 2

Game character creation with Blender and Unity, Chris To TTen, Editorial: John Wiley & Sons, 2012

Unity 4.x Game Development by Example Beginner's Guide, Ryan Henson Creighton, Editorial: Packt Publishing, 2013.

Unity, Nicolás Arrioja Landa Cosio, Manuales USERS, 2013.

(Sin numeración, ordenado alfabéticamente y en formato de presentación APA)



Universidad
Continental



HUANCAYO

Av. San Carlos 1980
Urb. San Antonio - Huancayo

Teléfono: 064 481430

LOS OLIVOS - LIMA

Av. Alfredo Mendiola 5210
Los Olivos - Lima

Teléfono: 01 2132760

MIRAFLORES - LIMA

Jr. Junín 355
Miraflores - Lima

Teléfono: 01 2132760

AREQUIPA

Av. Los Incas s/n
Urb. Lambramani, José Luis
Bustamante y Rivero - Arequipa

Teléfono: 054 412030

CUSCO

Av. Collasuyo Lote B-13
Urb. Manuel Prado
Wanchaq - Cusco

Teléfono: 084 480070