TRABAJO FIN DE MÁSTER

DESARROLLO DE UN VIDEOJUEGO EN UNITY CON GENERACIÓN DE ESCENARIO PROCEDURAL

MÁSTER UNIVERSITARIO EN APLICACIONES MULTIMEDIA

Alumno: José Luis Morant Capellino
Tutor: Sergio Schvarstein Liuboschetz

Resumen

En esta memoria se presenta el desarrollo, diseño e implementación de los elementos fundamentales de un videojuego 3D y se emplearán técnicas para que los escenarios donde se desarrolle la acción se generen de forma procedural. Para cumplir este cometido se hará uso principalmente de Unity como motor de videojuegos principal, Visual Studio como IDE de desarrollo, Blender para el modelado de objetos 3D y Photoshop e Illustrator para el diseño de texturas.

El objetivo principal de este proyecto es ampliar los conocimientos que se adquirirán mediante el desarrollo del mismo sobre técnicas de generación procedural y generación de elementos personalizables o aleatorios.

Palabras clave: Unity, videojuego, procedural, modelado 3D, diseño, texturizado.

Contenido

1. Justificación y motivación

1.1. Justificación

Si echáramos la vista atrás, hace años, los videojuegos eran considerados como productos exclusivamente de entretenimiento, pero con el paso del tiempo los videojuegos han adquirido cada vez más importancia en el mundo tecnológico haciendo que las piezas de hardware de procesamiento gráfico generen un mercado muy demandado actualmente y verse repercutido indirectamente en mejoras en otros campos como pueden ser los especialmente científicos que hacen uso de estos avances.

Pero, además, el impacto de los videojuegos ha repercutido tan ampliamente en el sector de la industria del entretenimiento que en la última década ha superado a la industria del cine y de la música. Por poner un ejemplo la película “Avatar” que está catalogada como la más exitosa del cine, teniendo un presupuesto similar a “Grand Theft Auto V” de unos 250 millones de dólares no se acerca en nada a la recaudación del videojuego.

El mundo de los videojuegos también ha permitido experimentar e introducir nuevos saltos tecnológicos en formato de almacenamiento y reproducción de lectura (p. ej. Blu-ray), sistemas de controles que proporcionan nuevas formas de interacción (Wiimote de Nintendo), nuevas formas de experimentar y sumergir al jugador (VR, AR), …

Por lo que podemos decir que los videojuegos son más que un sistema de entretenimiento, sino que producen además mejoras tecnológicas y avances en formas de interacción.

1.2. Motivación

Desde siempre, el mundo de los videojuegos me ha fascinado y esto me ha empujado a aprender e intentar descubrir como poder desarrollar mis propios videojuegos, desde luego que son el motor que me impulso a ser informático y motivarme a leer e informarme de todo lo relacionado con la tecnología. Por eso, he decidido dedicar mi memoria al desarrollo de un videojuego, ya que le debo a ello tantas horas empleadas tanto de aprendizaje como de entretenimiento.

2. Análisis de mercado

2.1. Mercado de distribución de videojuegos mundial

Hace años, las consolas eran la plataforma preferida para disfrutar de los videojuegos, pero estos datos empezaron a cambiar mediante el auge de plataformas de distribución digital de videojuegos. Algo parecido está pasando en los últimos años, pero esta vez con el incremento en ventas de dispositivos móviles donde hoy en día es el tipo de plataforma que más distribución de copias vende.

Tanto las plataformas de PC, como de consolas se mantendrían igualadas hoy en día.

Resulta curioso ver como las consolas portátiles han pasado a un segundo plano (habrá que ver cómo evolucionan estos datos a partir de finales de 2019 con la incorporación de la versión únicamente portable de la Nintendo Switch Lite).

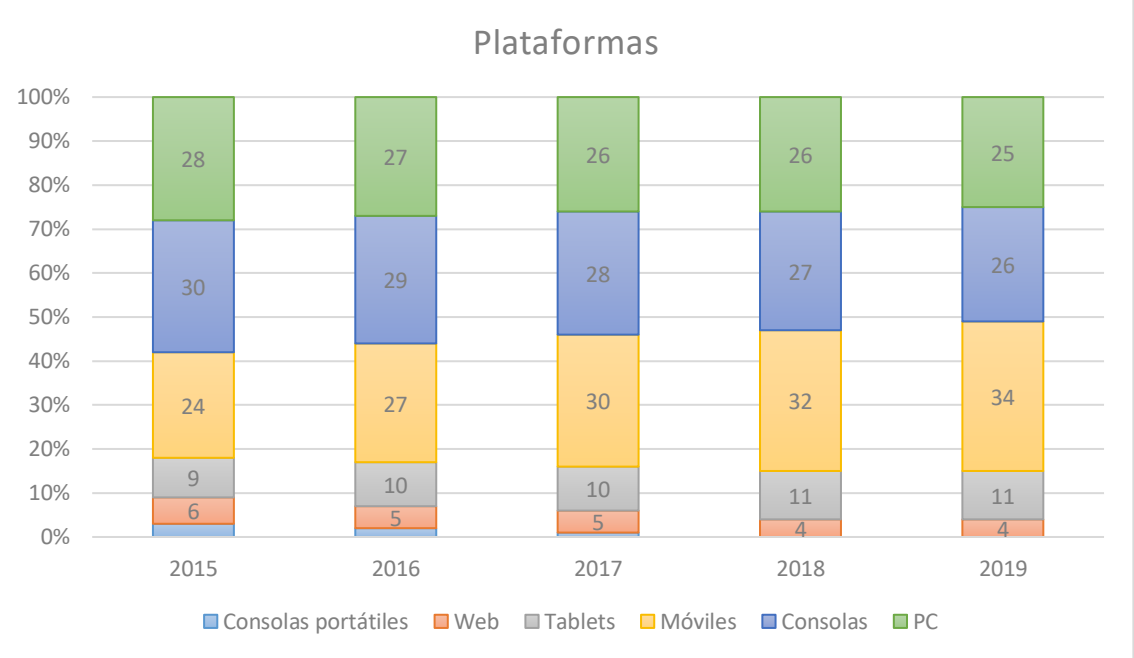


Ilustración 1. Distribución mundial de videojuegos. Fuente: Newzoo

En lo que compete a los ingresos de venta de videojuegos, se ha visto incrementado a nivel mundial, siendo el territorio asiático el que más nivel de ingresos genera.

El mercado europeo se situaría tercero muy ajustado junto al mercado norte americano y el mercado latinoamericano quedaría en último lugar bastante por debajo del resto.

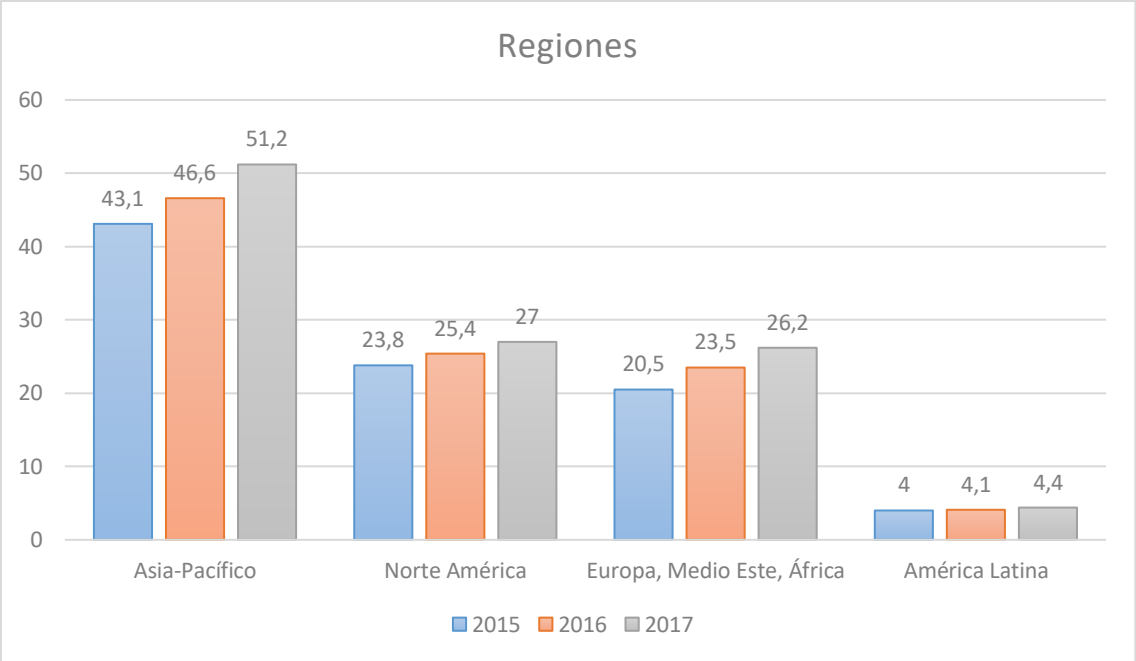


Ilustración 2. Mercado mundial en millones de ingresos por regiones. Fuente: Newzoo

Respecto al mercado de videojuegos en PC, no ha parado de crecer en el periodo de 2011 a 2019 (prácticamente se ha duplicado su nivel de mercado) y se prevé que continuará su evolución en los siguientes años.

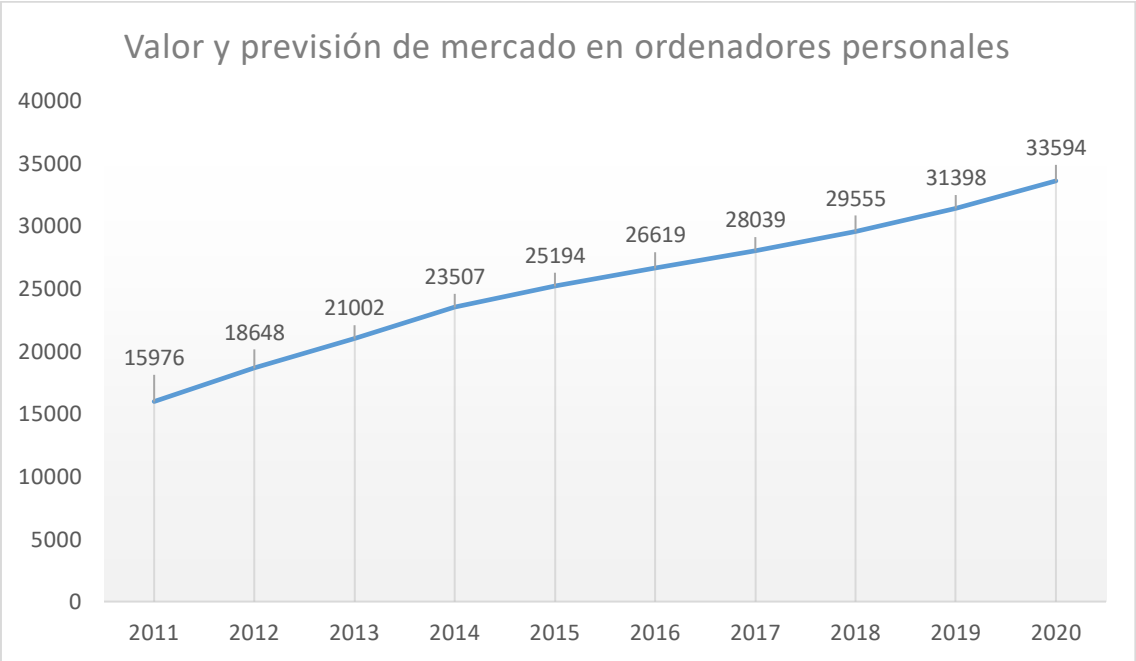


Ilustración 3. Valor y previsión del mercado en millones de dólares americanos.

Hay que mencionar que el mercado puede verse modificado en los próximos años mediante la posibilidad de unificación de plataformas y aprovechamiento de acceso al

entretenimiento de videojuegos mediante subscripción, como ha venido pasando en series y películas con plataformas como Netflix o HBO.

Todo esto podría ser posible mediante la incorporación de servicios de suscripción para videojuegos en la nube en la que grandes compañías como Google con su proyecto Stadia quieren poner en marcha.

Por lo que puede que la comparación que hoy en día existe entre plataformas hardware pase a formar parte de plataformas de subscripción de streaming.

2.2. Wave Race 64 como referencia

El primer videojuego de la serie Wave Race nació en 1992 para la consola portátil de Nintendo Game Boy. Este juego de temática de carreras de motos de agua se compone de gráficos monocromáticos y de una de cámara superior y técnicamente está limitado al de la consola portátil.

En el año 1996 saldría al mercado Wave Race 64 para Nintendo 64, este juego se compone de gráficos tridimensionales e incorpora un sistema de olas marinas que influyen en la jugabilidad. Este sistema se tendrá en cuenta como referencia para el desarrollo de este proyecto, y evidentemente el juego en general por su temática más que parecida con el descrito en este trabajo fin de master.

Ilustración 4. Imagen del videojuego Wave Race 64. Fuente: hobbyconsolas.com

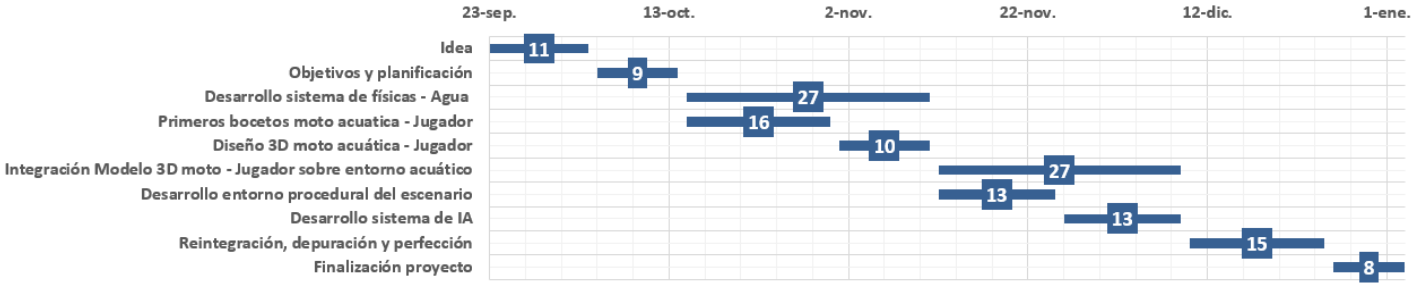
3. Objetivos y alcance

El objetico de este proyecto es el de conseguir desarrollar y diseñar las partes más esenciales y básicas de un videojuego con temática de carreras de motos de agua y no el de reunir en su conjunto una gran variedad de escenarios o detalles que podría disponer un videojuego comercial o completo ya que no sería posible en tiempo y forma, ni tampoco en forma de recursos (unipersonal).

Respecto al alcance que podría llegar a tener en vistas de futuro y post proyecto o después de cumplido el tiempo de trabajo de fin de master es el de catalogado como juego “indie” (videojuego independiente), cumpliendo con las premisas mínimas para ser considerado como videojuego completo y comercial, añadiendo variedad en forma de contenido tanto en mayor número de complementos decorativos, como variedad de diferentes tipos de juego y escenarios.

4. Planificación

4.1. Diagrama de Gantt



Nombre de la tarea	Inicio	Vencimiento	Días
Idea	23/09/2019	04/10/2019	11
Objetivos y planificación	05/10/2019	14/10/2019	9
Desarrollo sistema de físicas - Agua	15/10/2019	11/11/2019	27
Primeros bocetos moto acuática - Jugador	15/10/2019	31/10/2019	16
Diseño 3D moto acuática - Jugador	01/11/2019	11/11/2019	10
Integración Modelo 3D moto - Jugador sobre entorno acuático	12/11/2019	09/12/2019	27
Desarrollo entorno procedural del escenario	12/11/2019	25/11/2019	13
Desarrollo sistema de IA	26/11/2019	09/12/2019	13
Reintegración, depuración y perfección	10/12/2019	25/12/2019	15
Finalización proyecto	26/12/2019	03/01/2020	8
		Total:	149

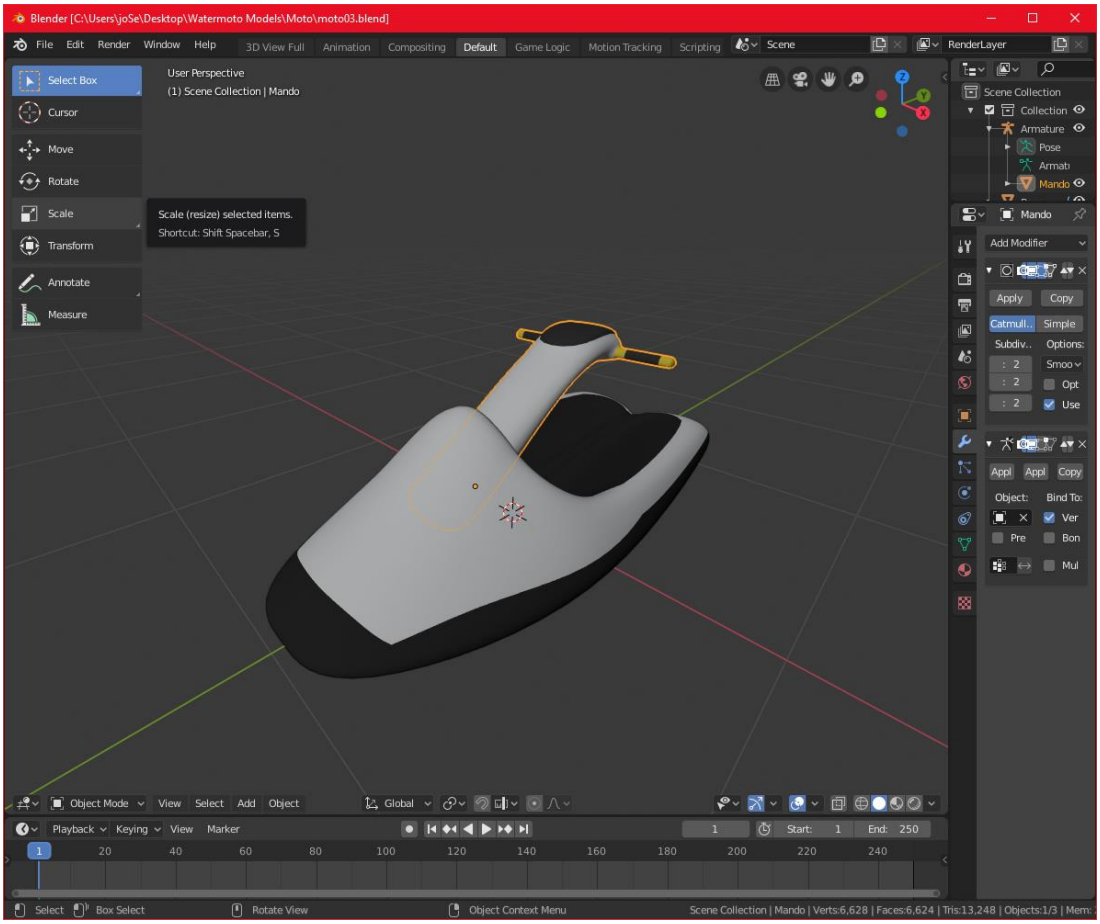
Los datos son sin contar la solapación de fechas, si reducimos el número de días por solapamiento queda en: **97 días totales de trabajo.**

5. Herramientas de trabajo

5.1. Blender

Es un software libre con licencia GNU que permite modelar y esculpir en 3D, animación además de permitir construir el mapeado UV que sirve para texturizar los modelos 3D que creemos.

Está programado en C, C++ y Python y es multiplataforma (principalmente Windows, macOS y Linux).



Servirá para la creación de modelos 3D de las motos de agua, personajes y elementos que compongan el escenario, así como el trabajo de rigging (Formar un esqueleto articulado que permite el movimiento de la malla) de los modelos que necesiten ser animados.

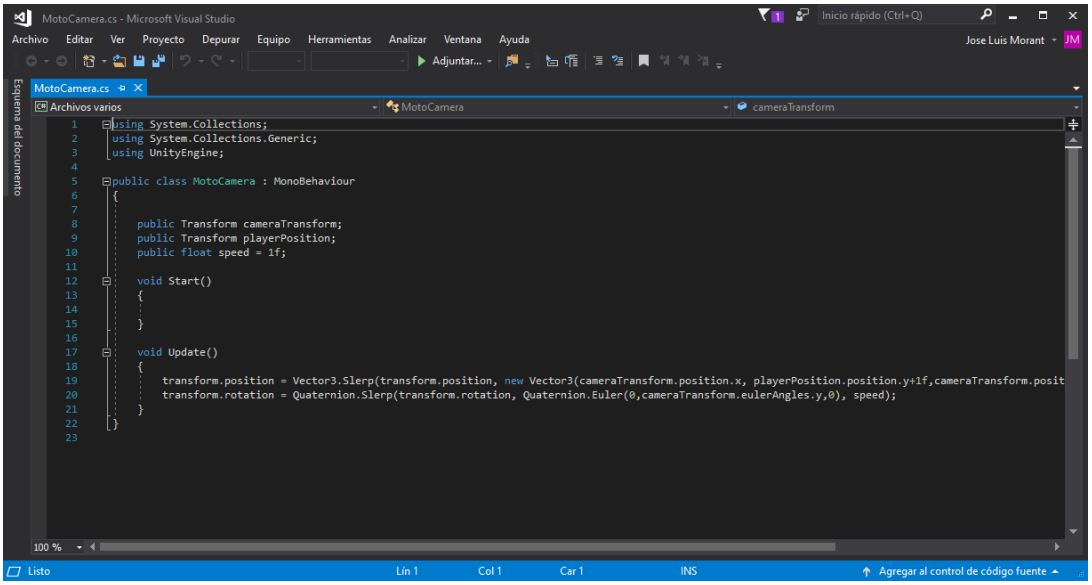
Como información adicional hay que mencionar que Blender está ideado también para el desarrollo de películas en animación 3D.

5.2. Visual Studio Community 2017

Es un Entorno de desarrollo integrado (IDE) desarrollado por Microsoft. Tiene las mismas características que la versión profesional, pero está pensado para estudiantes, empresas de pequeño tamaño y desarrolladores de software libre.

Esta programado en C++ y C# y puede ser instalado en sistemas Microsoft Windows y MacOS.

Se puede marcar para descargar desde las opciones de instalación e ya configurado para poder usado con Unity.

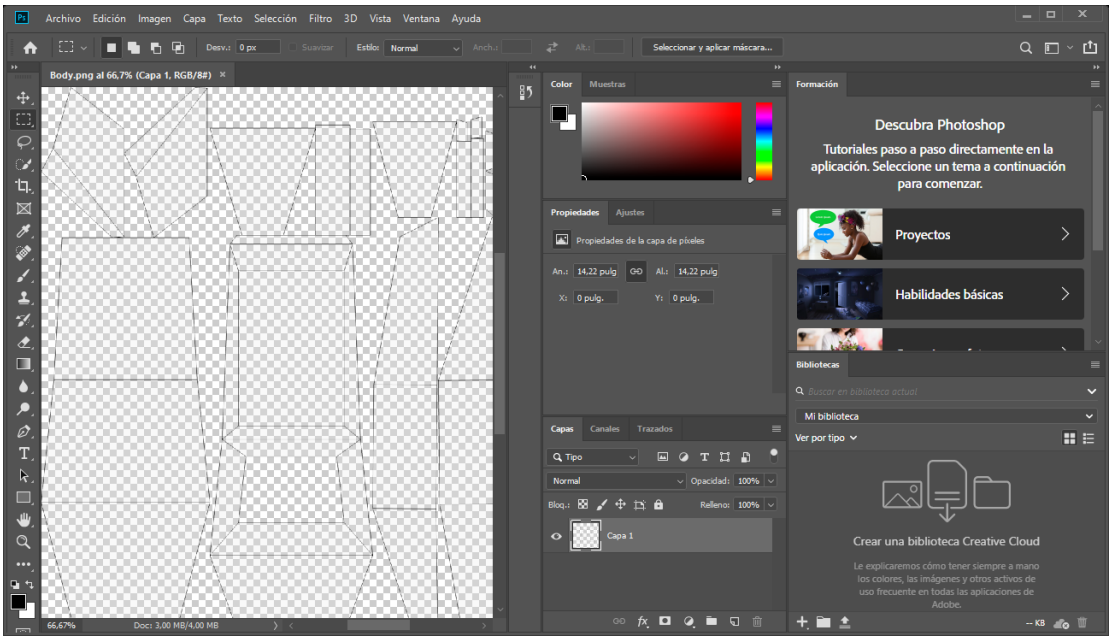


Para este proyecto se utilizará para realizar scripts en C#, pero Visual Studio Community soporta una gran variedad de lenguajes de programación como son C++, C#, Visual Basic .NET, F#, Java, Python, Ruby, PHP, ASP.NET, …

También existe una alternativa liviana llamada Visual Studio Code, más moderna y modular a la que se le pueden añadir las funcionalidades para facilitar el trabajo con Unity, pero me parece más cómoda para trabajar con proyectos web PHP o edición puntual de ficheros. Para este tipo de proyectos prefiero utilizar la versión Community.

5.3. Photoshop CC

Editor gráfico rasterizado y de retoque fotográfico conocido a nivel mundial. Esta desarrollado por Adobe Systems Incorporated y está programado con el lenguaje de programación C++. Puede ser instalado en sistemas Microsoft Windows y MacOS.

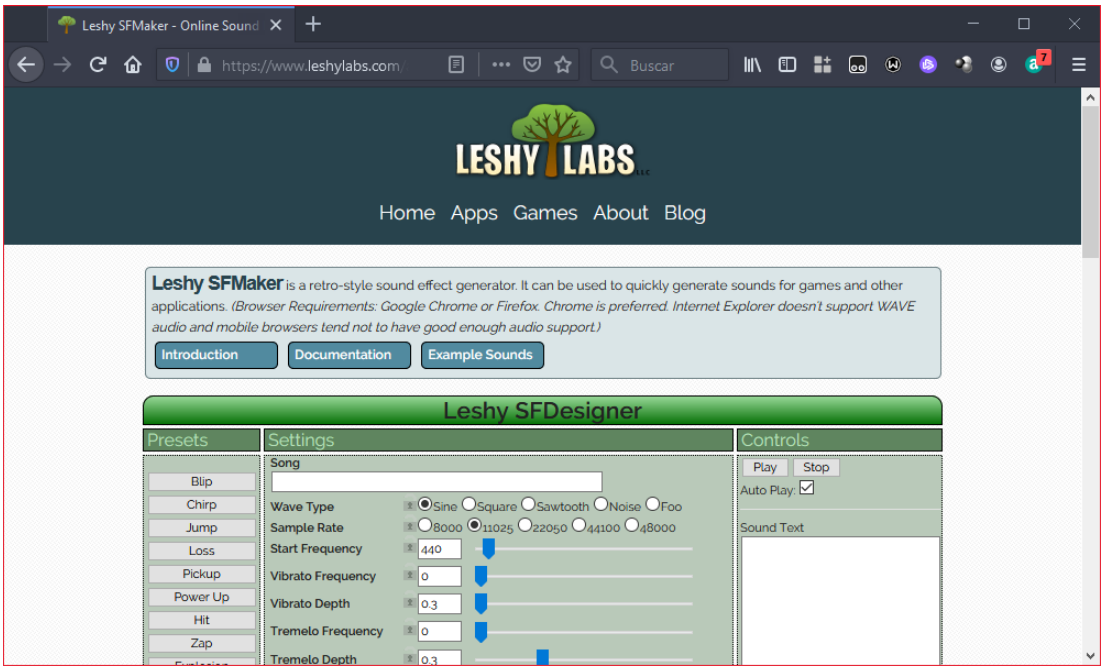


En este proyecto, Photoshop será usado principalmente para la creación y modificación de texturas de los modelos 3D, aunque también se empleará para gráficos 2D (sprites) para la parte de interfaz gráfica o para efectos de partículas, agua, …

Photoshop es un software propietario, pero en este caso se posee licencia de estudiante por parte de la UOC. Existe la alternativa libre GIMP, aunque la diferencia en cuanto a funcionalidades entre uno y otro es grande.

5.4. Leshy SFMaker

Es un editor online que permite crear efectos de sonido. Está desarrollado por Leshy Labs LLC y tiene soporte para los navegadores Google Chrome y Mozilla Firefox, dejando fuera a Internet Explorer.

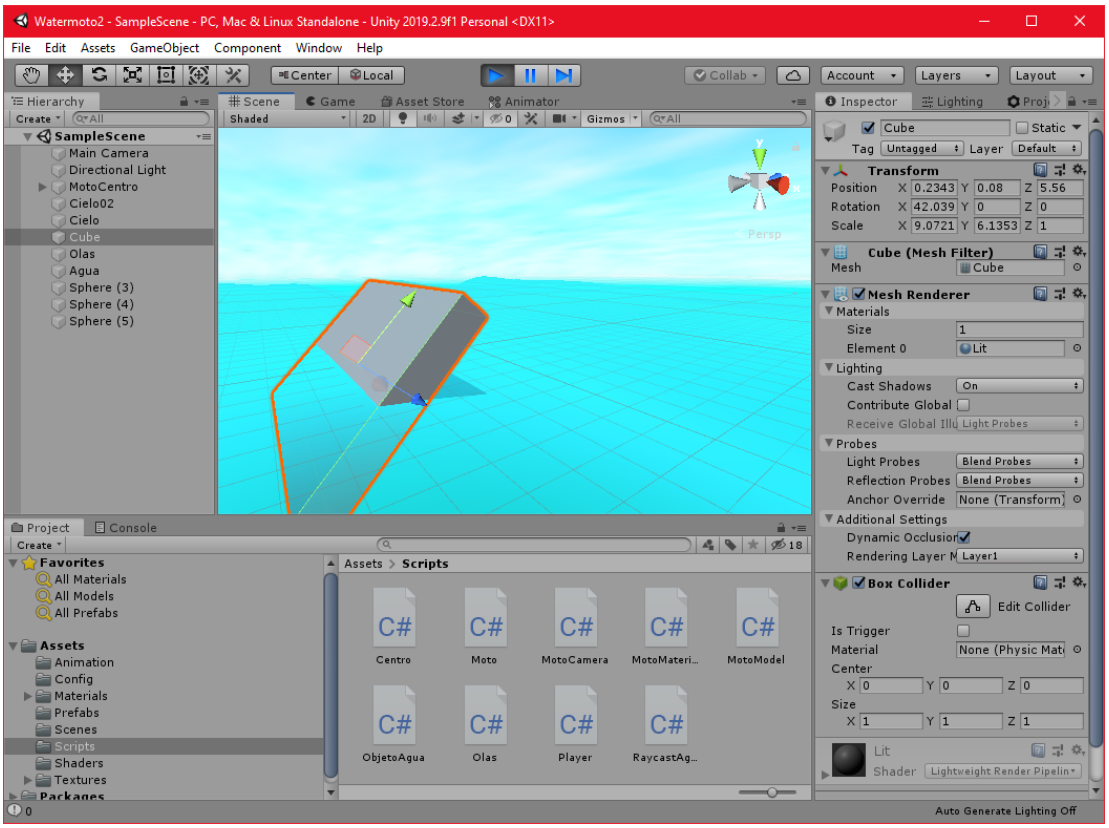


Será utilizado para la creación de efectos de sonido, como pueden ser recogida de ítems, saltos, etc.…

Permite exportar los efectos creados en formato .wav y es totalmente gratuito su uso.

5.5. Unity

Unity es un motor de videojuegos multiplataforma programado en C, C++ y C# desarrollado por Unity Technologies. Es compatible con sistemas Microsoft Windows, Mac OS X y Linux.



En Unity se recopilará todos los elementos que compongan las distintas escenas con sus modelos, código asociado, mapeado, materiales, texturas, sonidos, … dándole sentido al juego. Los distintos elementos que creamos o importamos en Unity son conocidos como 'Assets' y serán organizados en carpetas dependiendo del tipo de archivo.

Existen otros motores de videojuegos también accesibles para todo tipo de proyectos como puede ser Unreal Engine, pero Unity tiene un apoyo de la comunidad más grande y por lo tanto existe más documentación para desarrolladores.

Otra opción es CryEngine que es muy potente y ofrece características parecidas a Unreal Engine, pero es el que menos documentación y apoyo por parte de la comunidad posee.

Unity posee diferentes opciones de licencia: Personal, Plus y Pro.

Licencia	Precio	Descripción
Personal	Gratis	• Para principiantes o estudiantes. • Todas las prestaciones básicas del motor. • No permite quitar el logo de Unity 3D al cargar el videojuego
Plus	35$ / mes o 299$ anual	• Essentials Pack • Pantalla de inicio personalizable • Informes de ejecución • Unity Analytics ampliado • Gestión flexible de puestos • Editor UI Skin de la versión Pro • Priority Cloud Builds
Pro	125$ / mes, 1500$ anual o 2850$ cada 2 años	• Essentials Pack • Todas las prestaciones de la versión Plus • Servicios de nivel Pro • Tienes a tu disposición planes con soporte Premium y acceso al código fuente • No se aplican restricciones a los ingresos ni a la recaudación de fondos

6. Sistema acuático (oleaje marino)

En este punto se pretende aparte de mejorar el aspecto visual, añadir un apartado clave a nuestra jugabilidad, ya que este sistema de oleajes hará que nuestro personaje combine su movimiento con las interacciones y fricciones que provoquen las olas.

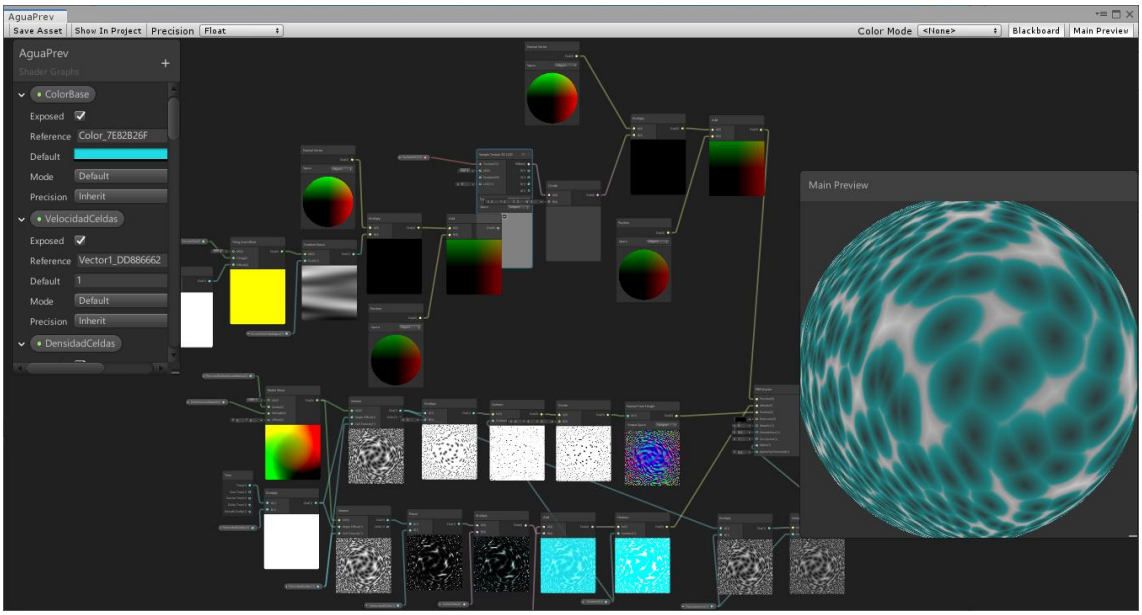
6.1. Uso de Shader Graph

En Unity podemos crear Shaders mediante la conexión de nodos en una red gráfica. Esto permite programar de forma más sencilla combinando datos, efectos, texturas, …

Mediante esta opción podemos modificar la malla de un objeto deformándola, por lo que nos vendrá muy bien para crear las olas.

Utilizando una textura como mapeado de deformación y haciendo que esta se mueva, podremos dar la animación de movimiento de las olas. Por lo que combinando este movimiento al de deformación tendremos conseguiremos fusionar movimiento y altura de las olas.

La principal ventaja de usar Shader Graph es que el proceso de cálculo lo realiza la GPU (núcleo de la tarjeta gráfica) por lo que esto se ve reflejado en un incremento de rendimiento y evitar cualgues en el uso de la experiencia.



Inconveniente: la modificación de la malla no es visible por la CPU, por lo tanto, si deformamos un plano, su colisión seguirá siendo plana para la CPU, aunque la actualicemos.

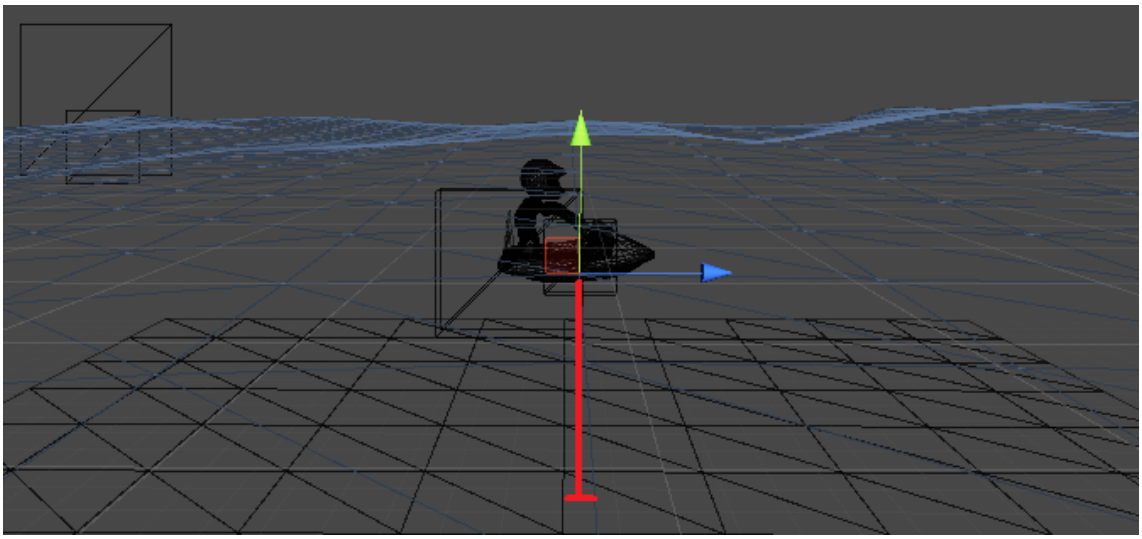
6.2. Combinación de Perlin Noise (CPU) con Shader Graph (GPU)

Mediante código y utilizando Perlin Noise podemos generar un mapa de ruido (tonos blancos y negros) que simulen la altura de las olas y produciendo un desplazamiento de esta textura su movimiento.

Por lo tanto, lo que intentamos hacer aquí es crear únicamente la textura mediante CPU, y aprovechar esta textura superponiéndola al plano que se deformará. Esta textura se la pasaremos también al Shader Graph que se encargara de hacer las modificaciones del agua.

Para que quede mas simple: tenemos 2 planos del mismo tamaño superpuestos, uno con la textura y el otro con el material Shader Graph al que le pasaremos la misma textura para que haga el efecto del oleaje marino.

El primer plano que solo contiene la textura no será visible por la cámara en el juego, y nos servirá para decirle al personaje la altura leyendo el punto de la textura donde este situado mediante Raycast (Una línea o rayo de luz que salga de la moto acuática y lea el pixel de textura justo situado debajo de su centro, obteniendo así el valor de altura dependiendo del valor de este pixel en escala de grises).



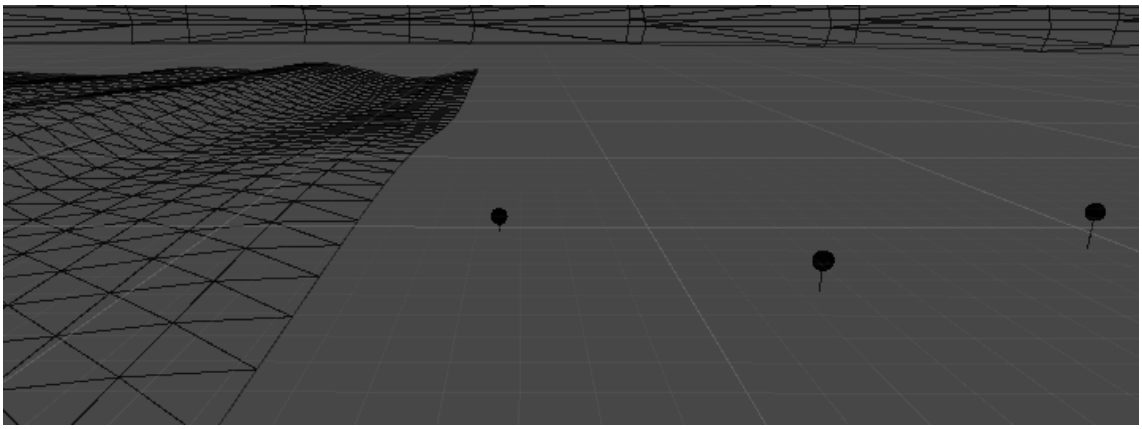
Inconveniente: Tanto el procesado de la GPU como de la CPU son variables, por lo que tenemos un problema de sincronización en la que, por ejemplo, la moto suba antes que la ola. Podemos intentar sincronizar estos movimientos, pero al final siempre acabaran por desincronizarse dependiendo de la carga de cada momento de cada componente.

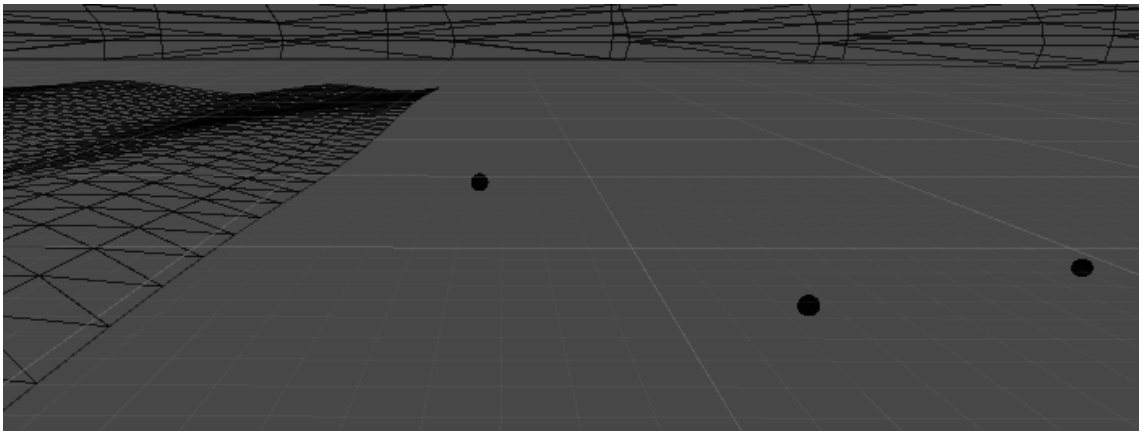
6.3. Solución: Uso exclusivo de la CPU

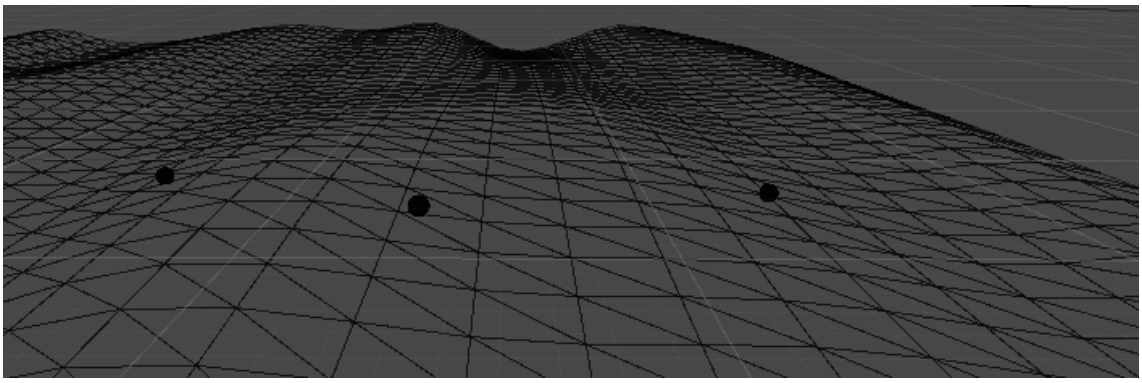
Esto provoca un empeoramiento radical del rendimiento, sobre todo porque estamos encargando a la CPU que deforme una gran malla.

Para evitar esto, lo que hacemos es engañar al espectador, creando una malla circular pequeña alrededor del jugador (hasta donde alcanza su visibilidad cercana). Las olas se encargarán de ocultar que por detrás de ellas hay o no más agua.

De momento tenemos solucionado el aspecto que altera al jugador, pero hay muchos más elementos que interactúan con el agua, como puedan ser otros jugadores controlados por la IA, o objetos que flotan en el agua. Para ello, haremos que si están fuera del rango donde existe la malla del agua se produzca un cambio de gravedad negativa a positiva y viceversa dependiendo de la altura media del agua. Esto provocará el efecto cuando los veamos a distancia que están realmente flotando sobre el agua.



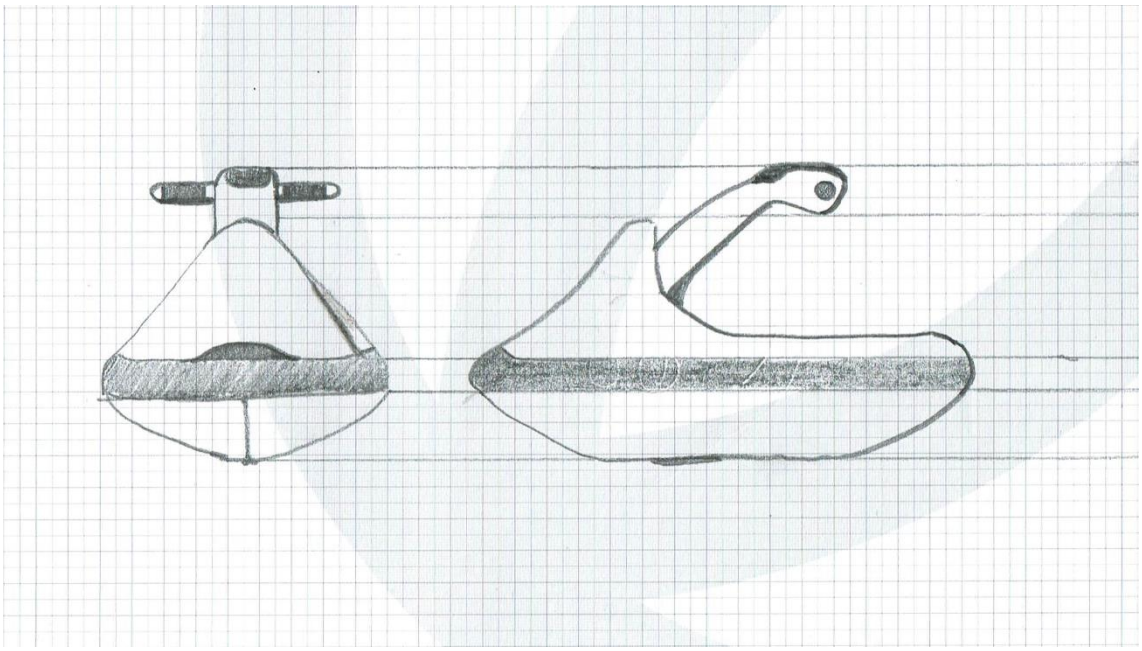




7. Diseño moto acuática

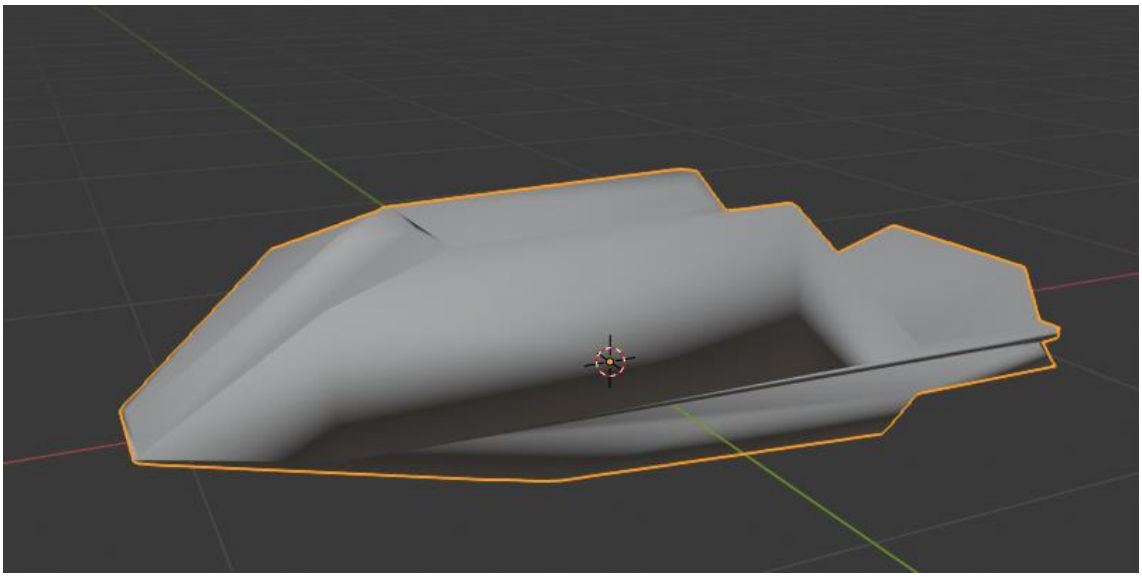
Se ha realizado un boceto a lápiz y papel de vista frontal y lateral, y mediante estos planos se ha desarrollado el modelado 3D en Blender. Para evitar excesos de polígonos y un acabado que resultara mas atractivo se ha partido de cero con un segundo modelo 3D.

7.1. Boceto

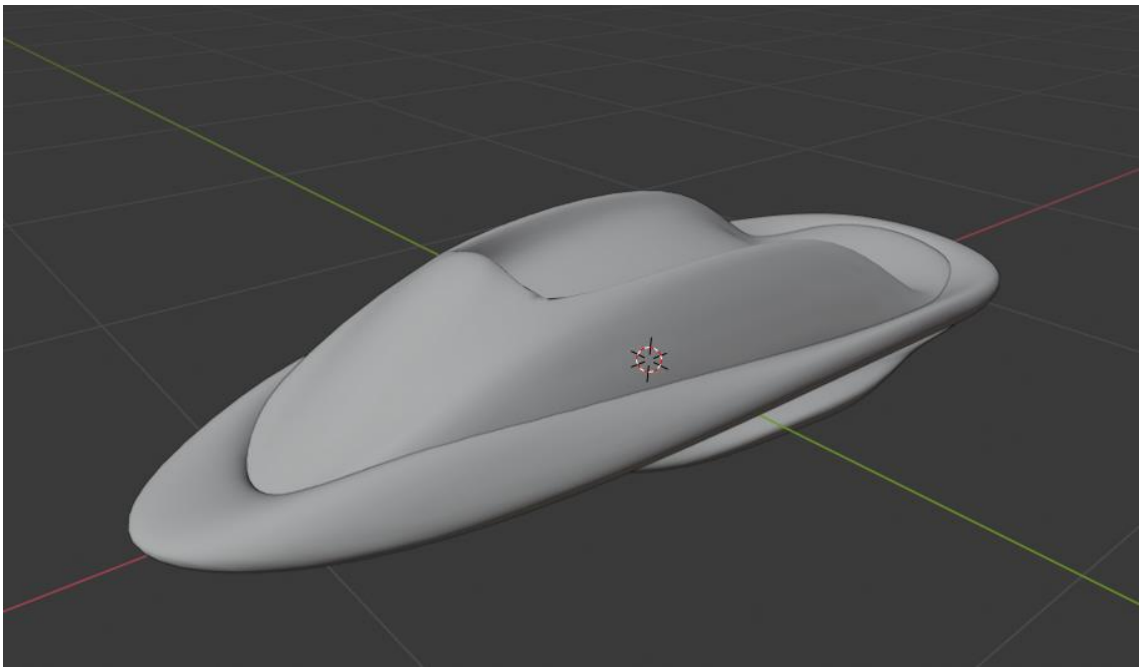


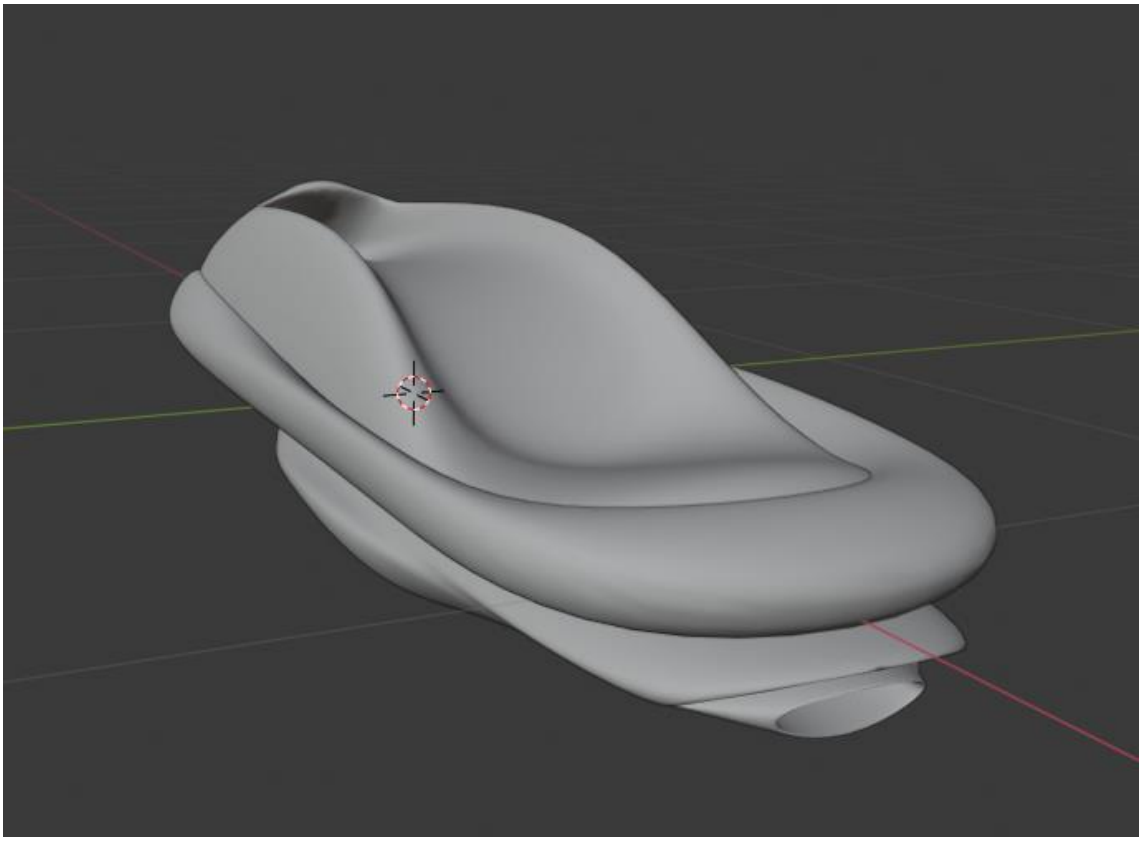
7.2. Primer modelo 3D de la moto

En la siguiente imagen podemos ver la base del modelo sin subdivisiones en su malla. En esta primera versión se ha modificado la base dibujada sobre el boceto dándole un aspecto más agresivo.



Y a continuación el modelo con subdiviones y la goma externa:

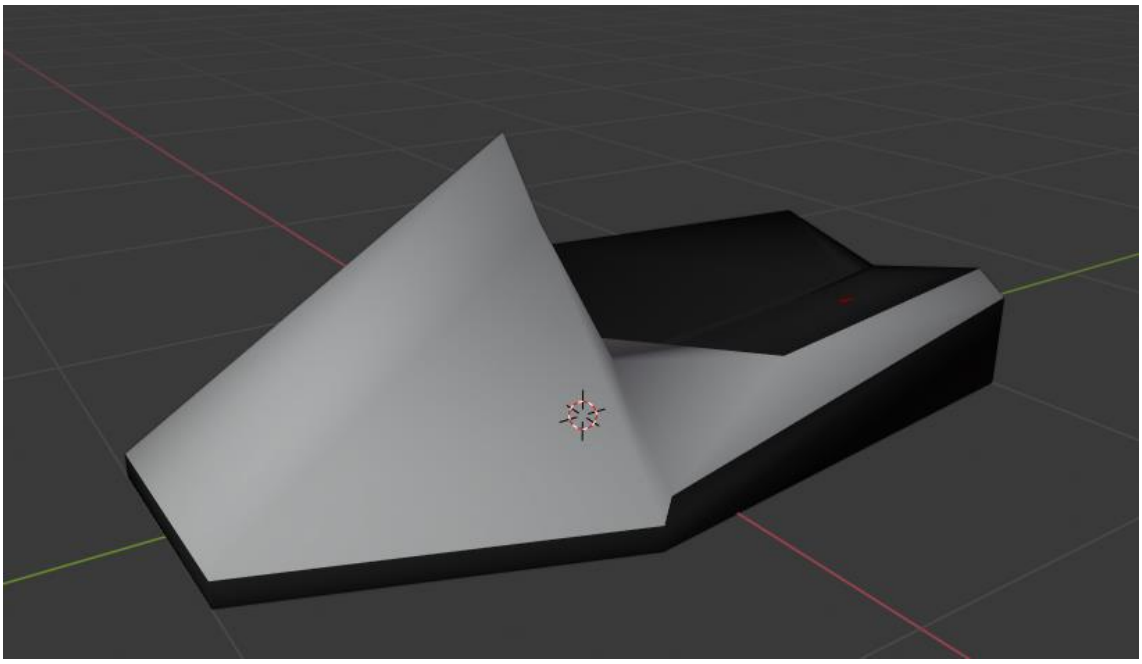


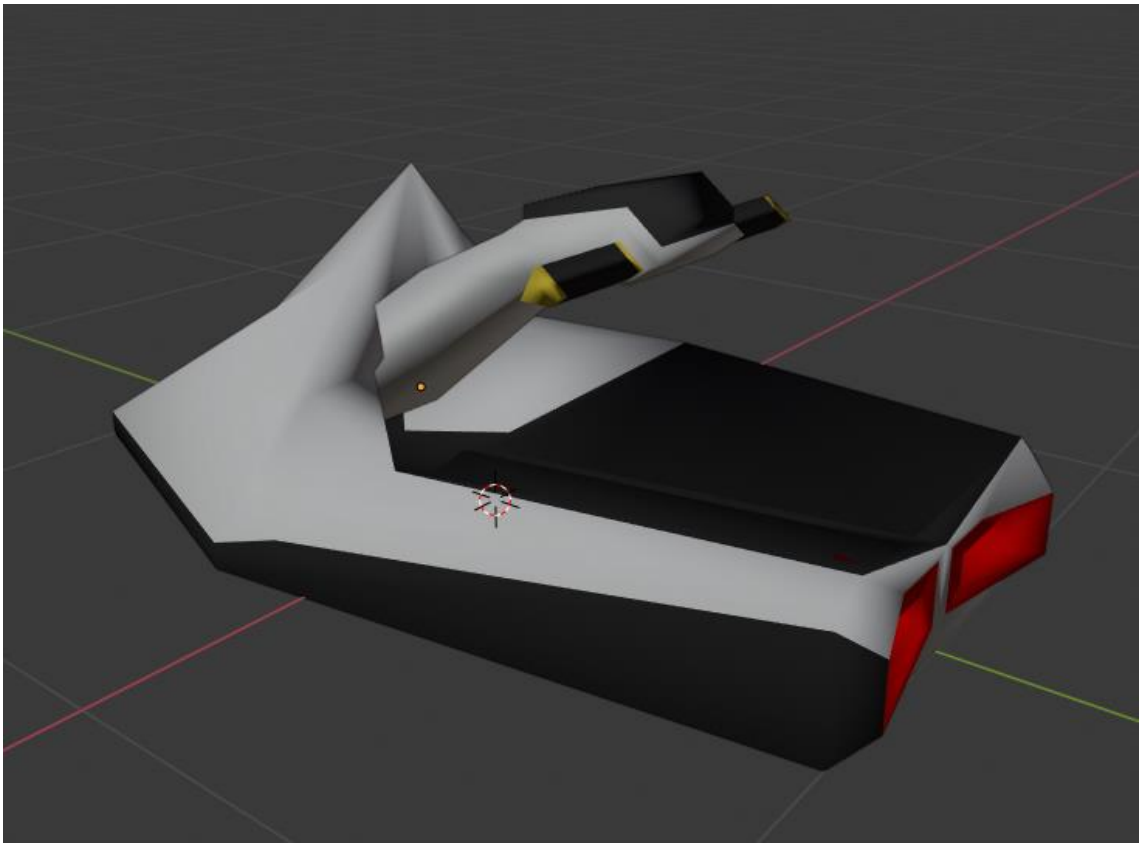


7.3. Segundo modelo 3D de la moto

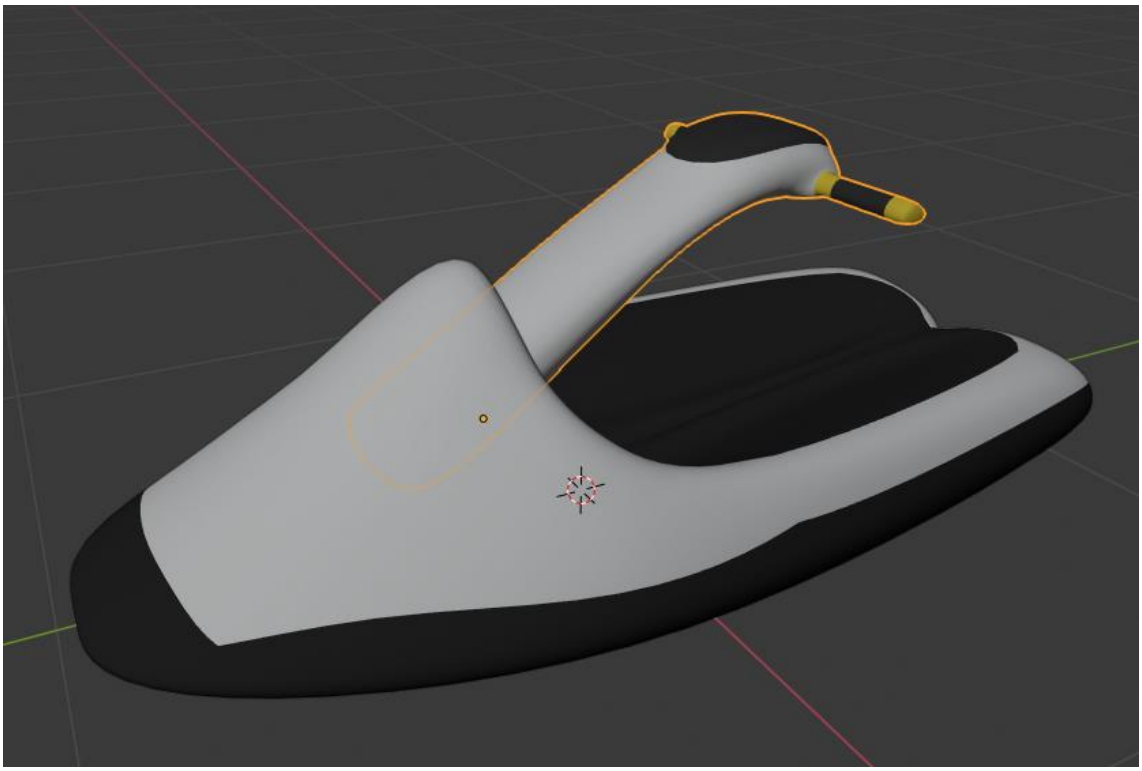
Como se ha mencionado anteriormente se reinicia el proceso de modelado de la moto acuática como se puede ver a continuación. (Los colores son por separación de materiales, no tienen nada que ver con el color definitivo de la moto de agua).

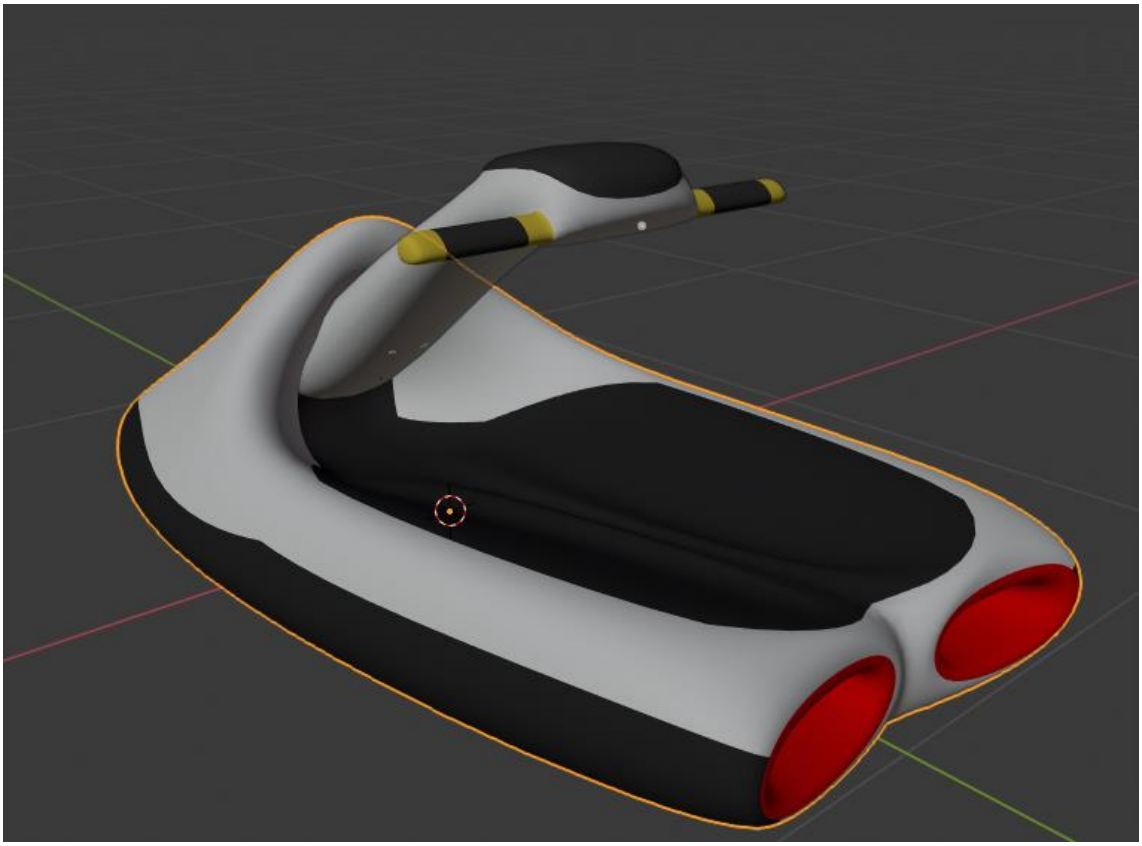
Bajo nivel de detalle:





Alto nivel de detalle:

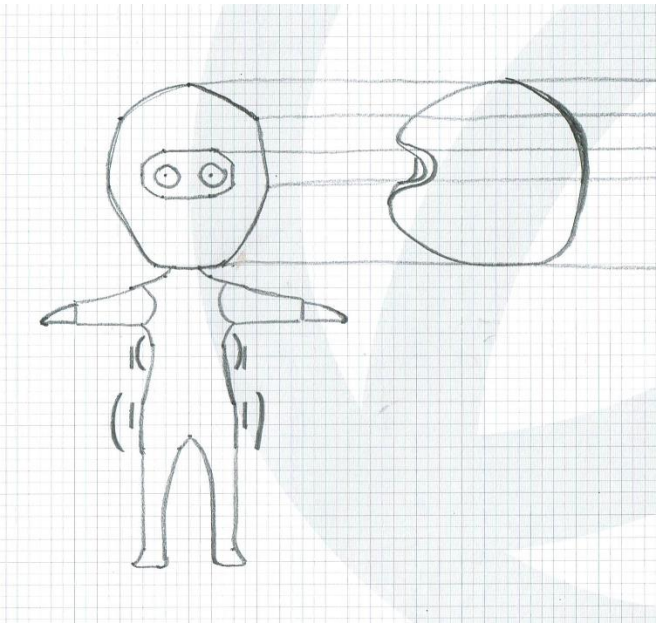




8. Diseño motorista

Como en el caso de la moto acuática se ha realizado primero un boceto y después se ha procedido ha realizar el modelado 3D en Blender.

8.1. Boceto



8.2. Modelo 3D del motorista

En el diseño del modelo 3D se ha mejorado las curvas, ya que en el boceto inicial quedaba muy plano y poco redondeado.

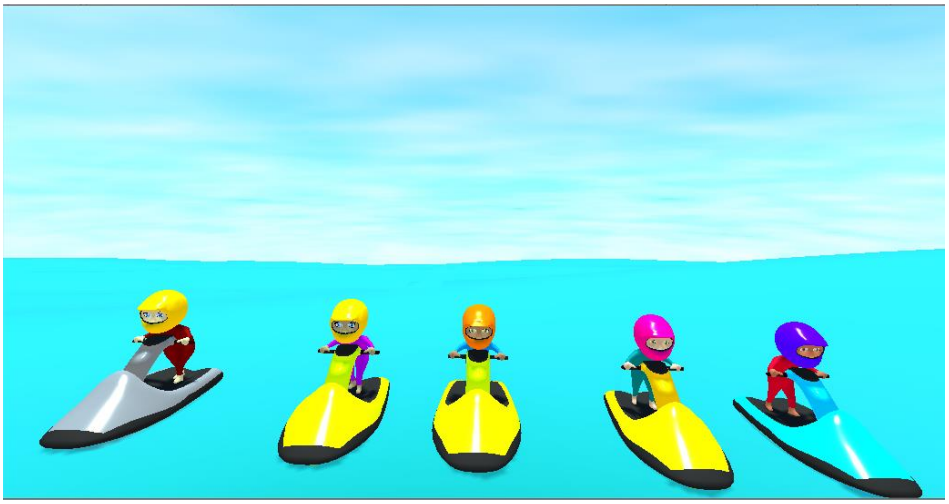
Bajo nivel de detalle:

Alto nivel de detalle:

El modelo dispone de modificadores (Shape Keys) que permiten modificar el cuerpo y así formar el cuerpo tanto de hombre como de mujer y tamaño, forma de los ojos y pestañas.

9. Puesta en escena de los modelos

Utilizando la variedad de materiales y texturas se ha desarrollado un script que permite modificar el tono de piel, ojos, pestañas, cabello y sexo del motorista y color de vestimentas y motos acuáticas quedando de la siguiente forma:



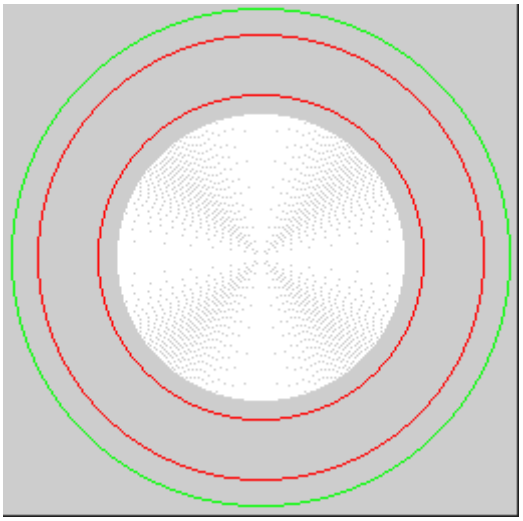
10. Entorno procedural del escenario (Isla)

El propósito de este punto es generar una isla central donde los motoristas rodearan. Para ello, se generan una serie de texturas que actúan como mapeado de altura.

El mapa de altura facilita la deformación de la isla, generando diferentes puntos de altura en sus vértices.

10.1. Límites

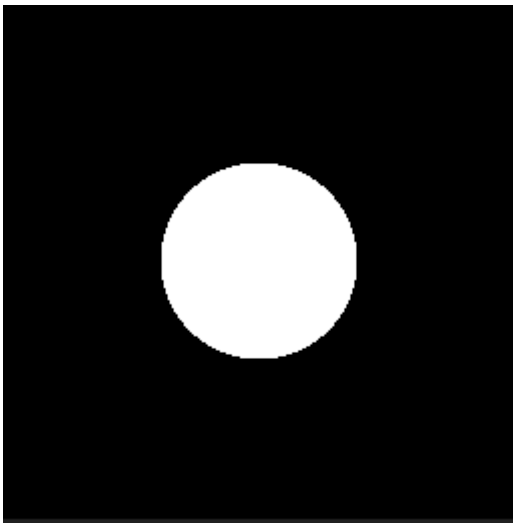
Hay que tener en cuenta que tamaño máximo puede tener nuestra isla, ya que hay que dejar el espacio donde los motoristas puedan navegar.



En la imagen anterior podemos ver una simulación donde el circulo blanco es la isla, la zona roja es donde competirán los motoristas y la zona verde es el límite máximo en la que la zona roja puede llegar a existir.

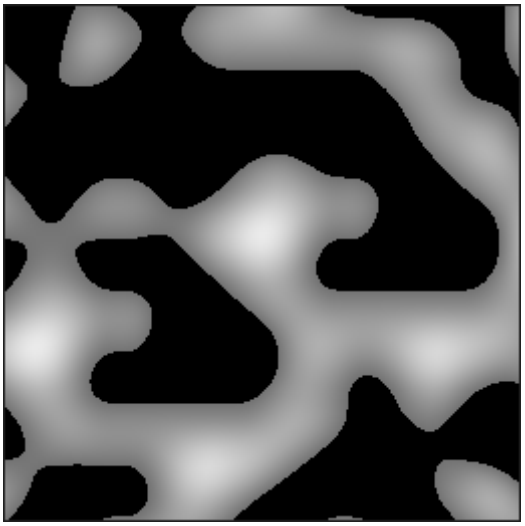
10.2. Máscara

Teniendo en cuenta los límites anteriores se almacenan al azar los distintos radios de distancia respecto al centro y se genera la máscara de la isla.



10.3. Mapa de altura

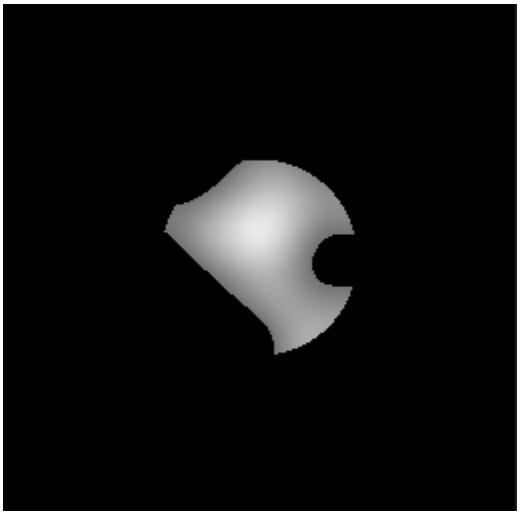
Utilizando y ajustando los parámetros mediante Perlin Noise, se genera un mapa de altura parecido a un sistema montañoso.



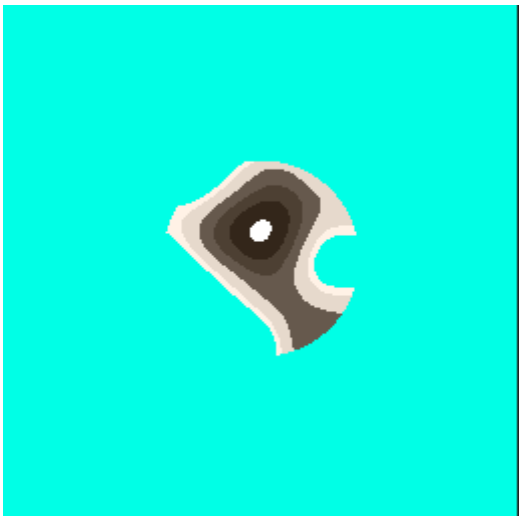
Los pixeles en negro indican el valor más bajo y el más blanco el más alto.

10.4. Multiplicación de mapa de altura y máscara

Como nuestra máscara de la isla esta rodeada de negro de valor 0, cualquier valor multiplicado por este será 0, y la parte de la isla es blanco al multiplicar por 1 nos devolverá el valor por el que lo multiplicamos. De este modo conseguimos generar los límites de la isla y el agua.



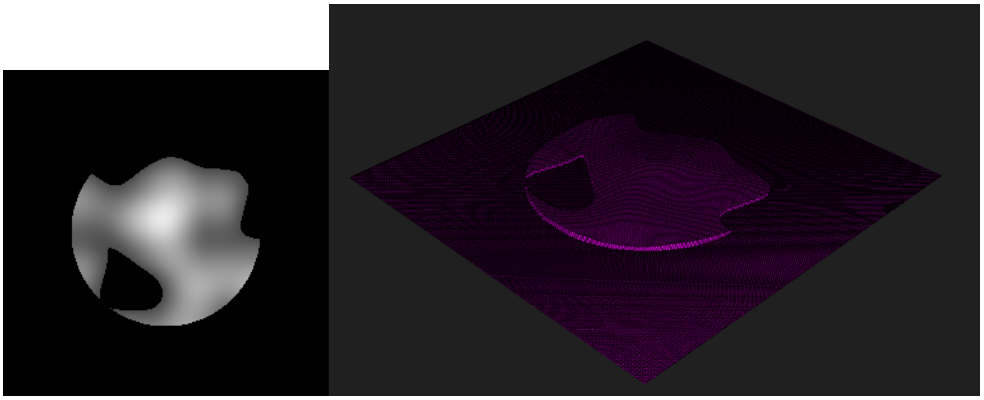
Dependiendo de estos valores, se hace otra pasada donde se indica los colores comprendidos para ciertos valores, generando un mapa de color.

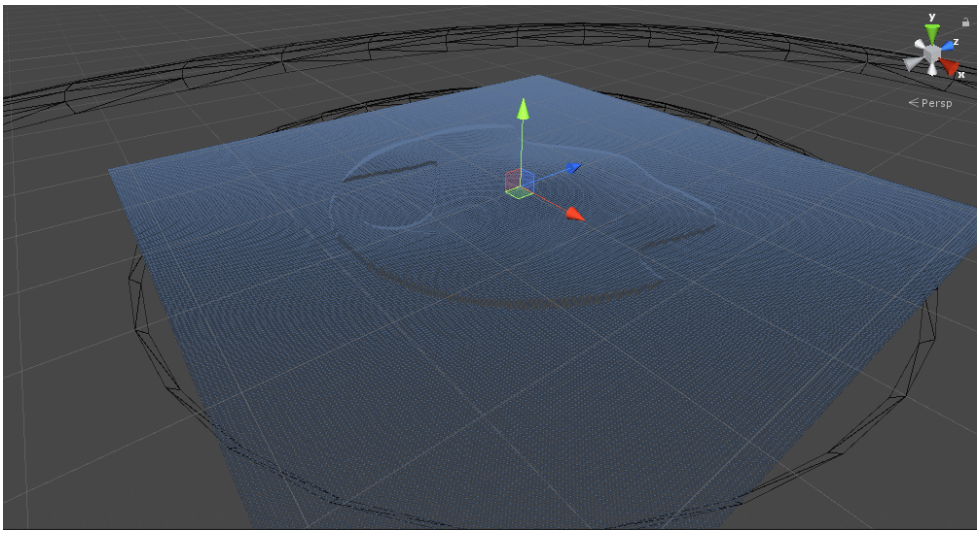


Mediante todo este proceso podemos generar islas diferentes para cada juego:

10.5. Isla 3D

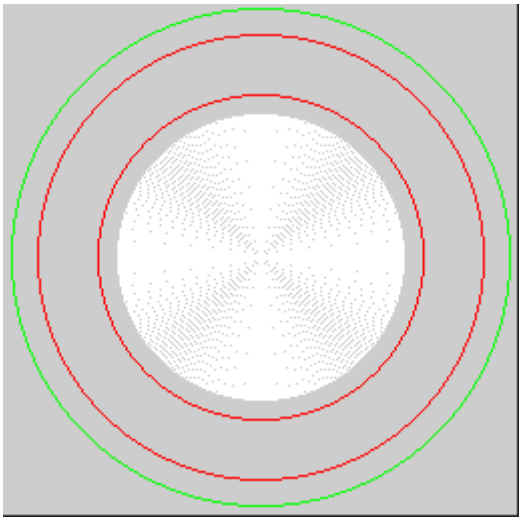
Mediante el mapa de altura generado anteriormente (no el de color). Se genera una maya 3D del mismo valor de vértices que pixeles tiene la textura y el valor de los pixeles nos modifica la altura de los vértices de la misma posición.




y
z
x
Persp

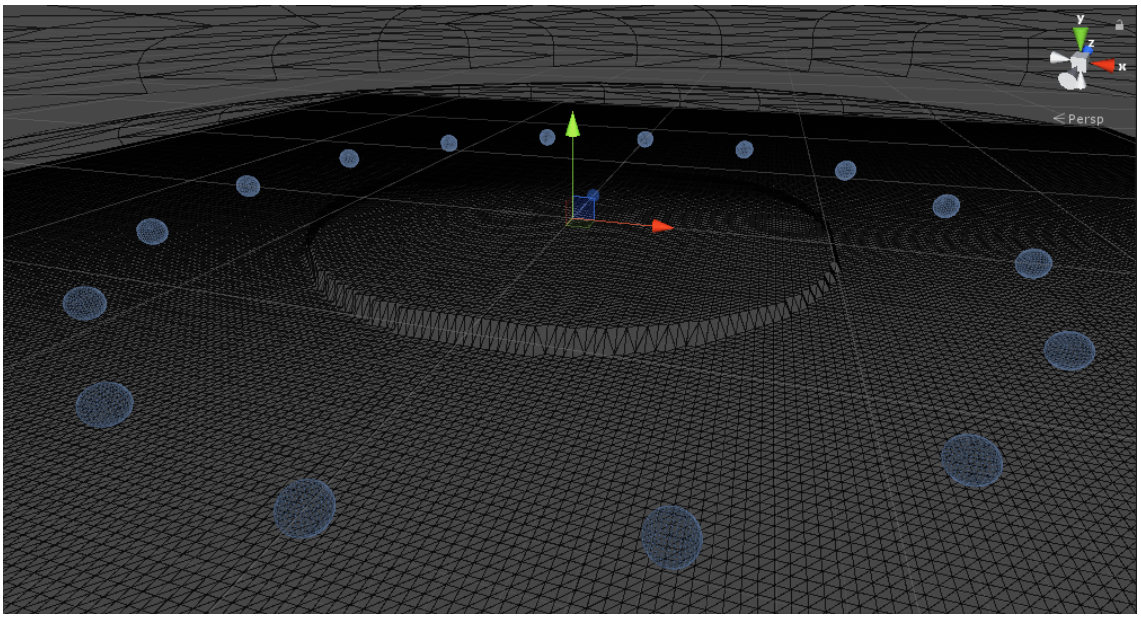
11. Sistema de IA

Teniendo la zona reservada donde los motoristas podrán dar vueltas (radio mínimo de acción y radio máximo). Zona roja vista en el punto anterior:



Se genera puntos en la zona media los cuales los motoristas seguirán de forma consecutiva.

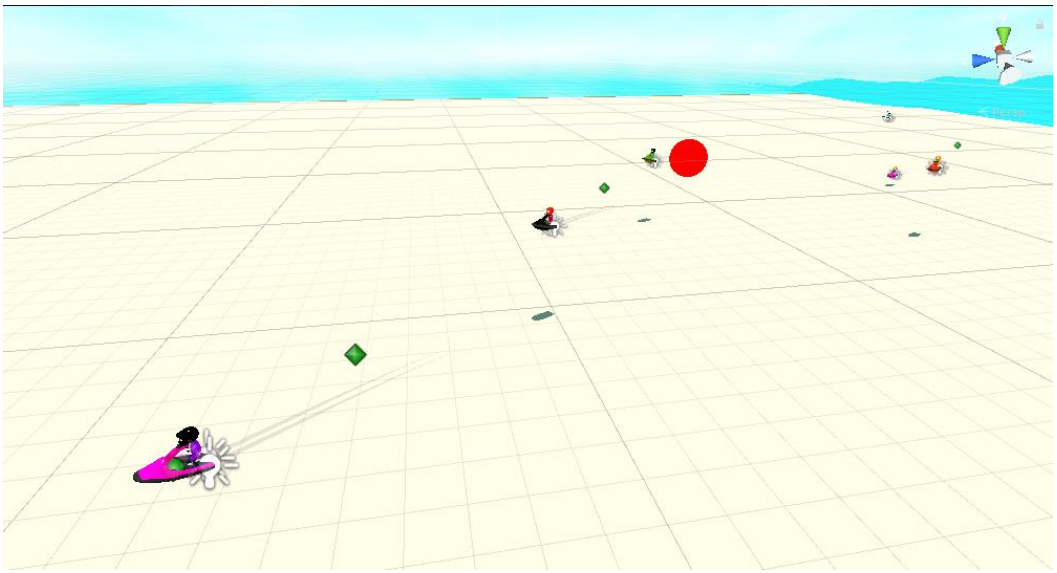
P1 → P2 → P3 → … → P final → P1 → P2 → …

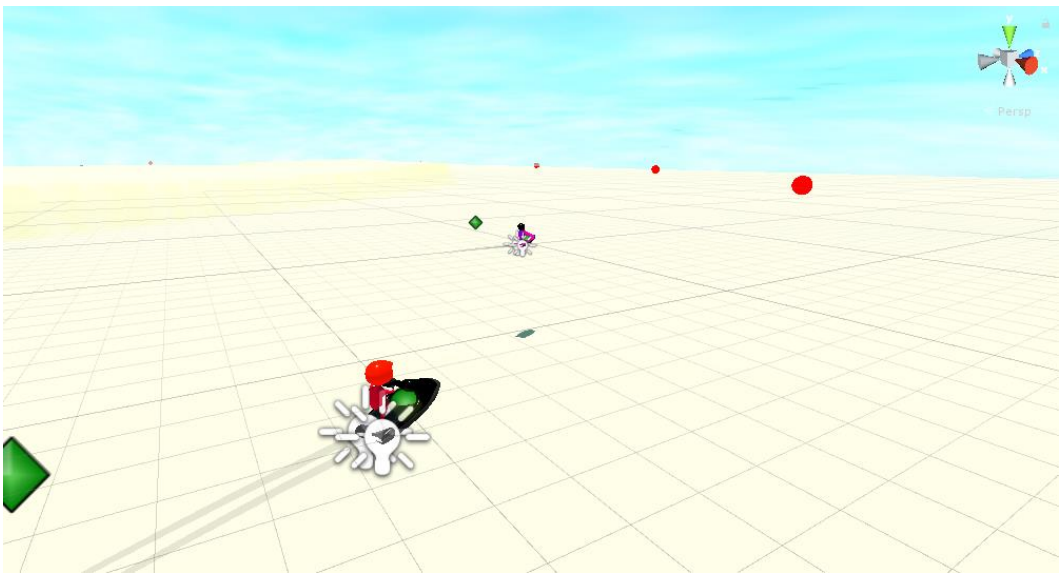


Si el motorista entra en la zona del punto 2 (P2), mirará y se desplazará hacia el punto 3 (P3), luego el punto 4 (P4), … Hasta que llegado al último punto contará una vuelta y volverá a empezar con el punto 1 (P1).

En el momento que se instancian los motoristas se les asigna un nivel de aceleración y habilidad de giro para diferenciar las capacidades de los diferentes motoristas manejados por la IA.

Vista en modo desarrollo:





12.Bibliografía

- Análisis Wave Race 64 - Hobby Consolas
  https://www.hobbyconsolas.com/reviews/wave-race-64-asi-analizo-hobby-consolas-hace-20-anos-97746

- Estadísticas de Videojuegos (wepc.com)
  https://www.wepc.com/news/video-game-statistics/

- Ventas de videojuegos (Newzoo)
  https://newzoo.com/insights/articles/global-games-market-reaches-137-9-billion-in-2018-mobile-games-take-half/

- Análisis Wave Race : Blue Storm - Vandal
  https://vandal.elespanol.com/analisis/gcn/wave-race-blue-storm/625#p-3

- ¿Qué es Blender 3D? - cgstudioscolombia.com
  http://www.cgstudioscolombia.com/blender/index.php/que-es-blender

- Blender – Wikipedia
  https://es.wikipedia.org/wiki/Blender

- Visual Studio Community - Microsoft
  https://visualstudio.microsoft.com/es/vs/community/

- Microsoft Visual Studio – Wikipedia
  https://es.wikipedia.org/wiki/Microsoft_Visual_Studio

- Visual Studio Code - Wikipedia
  https://es.wikipedia.org/wiki/Visual_Studio_Code

- Leshy SFMaker
  https://www.leshylabs.com/apps/sfMaker/

- Introducción Unity Shader Graph
  https://unity.com/es/shader-graph

- Creando Shader Graph
  https://docs.unity3d.com/Packages/com.unity.shadergraph@6.9/manual/Create-Shader-Graph.html

- Perlin Noise Unity
  https://docs.unity3d.com/ScriptReference/Mathf.PerlinNoise.html