Lost in the Abysm

Autores: Lauro Luján Omar Cacciagioni , Juan Ignacio Eizmendi , y Juan Emilio Nieva Casanova.
Tutores académicos: Ing. David Curras , Mg. Ing. Silvia Poncio , e Ing. Pablo Audoglio.

Universidad Abierta Interamericana, Facultad de Tecnología Informática
Av. Ovidio Lagos 944, Rosario, Santa Fe, Argentina
lauroomar_c@hotmail.com; {jieizmendi, juan.nieva94}@gmail.com
{david.curras, silvia.poncio, pablo.audoglio}@uai.edu.ar

Resumen: El presente trabajo es una reedición del trabajo "*Lost in the Abysm*" publicado en el Congreso Nacional de Ingeniería en Informática / Sistemas de Información (CoNaIISI) en el año 2016. Dicha reedición surge en el marco de las cátedras "Seminario de aplicación profesional" y "Trabajo final de ingeniería" de quinto año de la carrera Ingeniería en Sistemas Informáticos de la Universidad Abierta Interamericana de Rosario debido a los cambios que ha sufrido el mismo a lo largo del desarrollo, se busca mostrar el estado actual del proyecto comenzado en 2015.

Lost in the Abysm es un videojuego de rol de acción en tercera persona, realizado con la intención de poder aprender y aplicar la mayor cantidad posible de las técnicas y herramientas para el desarrollo videojuegos en tres dimensiones. Desde el modelado 3D, así como el diseño de la jugabilidad, la programación, la música, etc.

Palabras Clave: "Unity 3D", *Lost in the Abysm*, C#, "Kanban","Jugabilidad".

1 Introducción

El proyecto inicia en Octubre de 2015 luego de haber obtenido una satisfactoria experiencia en la realización de videojuegos en Unity3D[1].

Éste juego, llamado "*Lost in the Abysm*", empezó con el objetivo de aprender a realizar un videojuego complejo en tres dimensiones, objetivo que desde el inicio del mismo se está cumpliendo dado el aprendizaje constante en materia de desarrollo.

Así mismo, dicho juego comenzó teniendo únicamente un fin educativo pero a medida que se fue desarrollando se incorporó el fin comercial, apuntando a crear un juego "AAA"[1] y comercializarlo.

2 Características del juego

"Lost in the Abysm" es un videojuego de rol de acción en tercera persona ambientado en un mundo de fantasía medieval, narra la historia de un cazador el cual entra al abismo a través de un portal que encuentra en el bosque tentado por la promesa de poder que le hace un espíritu. Sus flechas y poderes son sus herramientas para abrirse paso entre los enemigos.

Fig. 1. Captura de pantalla del juego. Bosque.

2.1 Progresión del juego

A medida que el jugador avanza va ganando poderes y conociendo la historia del mundo, esto permite comprender a los enemigos y derrotarlos más fácilmente.

Para acceder a los poderes es necesario superar templos elementales, donde se pondrán a prueba los reflejos del jugador esquivando trampas y flechas enemigas.

[1] Juego con énfasis en la jugabilidad, presentación, historia, gráficos y audio. Usualmente con una alta inversión económica.

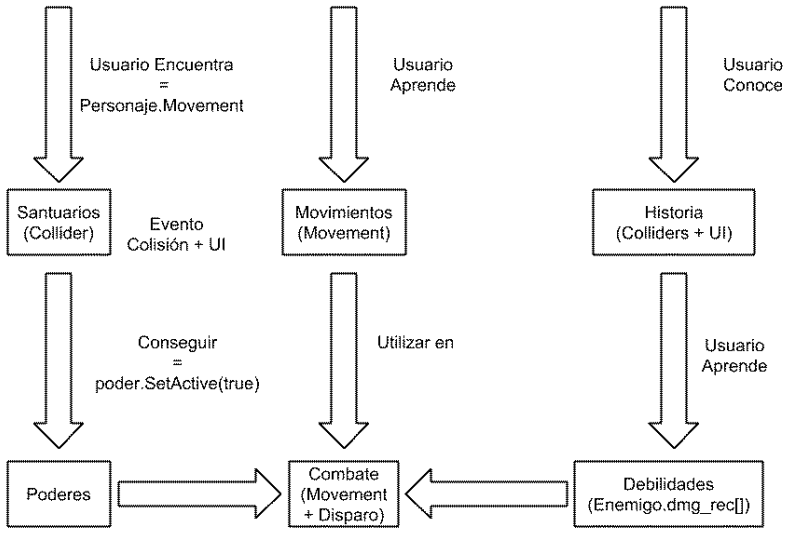

Fig. 2. Progresión del juego. Este ciclo se repite a lo largo de cada nivel ó zona.

2.3 Personaje

El personaje principal y los NPC (Non Playable Character) fueron diseñados y modelados utilizando SketchUp 8 debido a la estética del juego en Low Poly [2] y la facilidad del software de diseño.

Así mismo las armas y accesorios del personaje fueron diseñados con este mismo software.

2.4 Movimientos

El personaje cuenta con diferentes movimientos característicos de juegos en tercera persona existentes en la actualidad. El desplazamiento es controlado por las teclas W/A/S/D provocando que el personaje corra, al presionar una tecla específica el personaje puede realizar un desplazamiento rápido en cualquier dirección a cambio de energía.

3 Metodología de trabajo

4 Lost in the Abysm

Lo primero que se definió fue la metodología de trabajo. Se sabía que la jugabilidad[3] iba a sufrir grandes cambios a medida que se capacitaba en las diferentes áreas de desarrollo de juegos, por lo que se adoptó una metodología de las conocidas como ágiles. "Kanban"[4] brindó la flexibilidad necesaria para poder llevar a cabo este proyecto tan dinámico y complejo.

Esta metodología no tiene iteraciones y es poco prescriptiva, lo cual significa que hay menos reglas a seguir que en aquellas muy prescriptivas (Scrum o RUP). Requiere de un tablero (figura 3), y respetar siempre dos normas básicas: el flujo de trabajo y el trabajo en curso (WIP, *work in progress*).

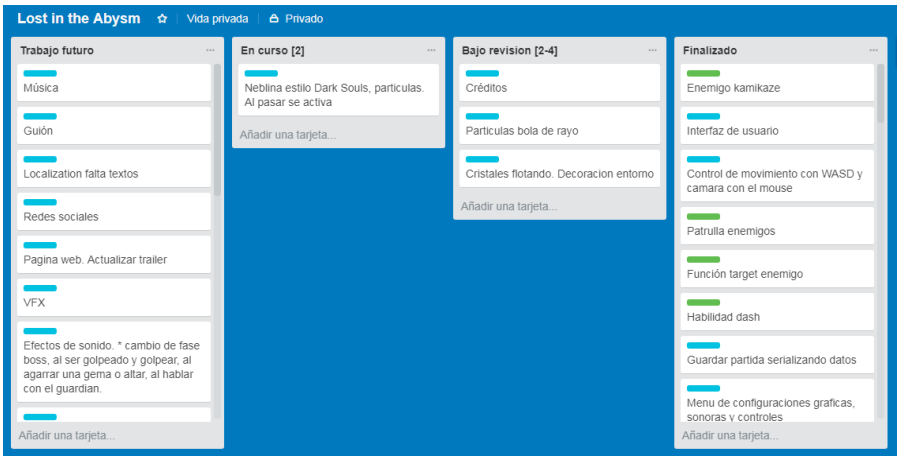

Fig. 3. Captura de pantalla del tablero "Kanban" que permite la gestión y avance de *Lost in the Abysm.*

Al tablero se accede por la web, y es actualizado diariamente.

Cada tarjeta representa una nueva funcionalidad a realizar (verde), un defecto (naranja) o una tarea (celeste), contienen un nombre y una descripción.

Las tarjetas se diseñan y crean en la sección “Trabajo futuro”. Luego se eligen las más prioritarias y se pasan a la columna “En curso”.

En esta columna, se desarrollan todas las tareas necesarias para cumplir con el criterio de aceptación de la tarjeta, y una vez terminado pasan a la comuna “Bajo revisión”, donde se verifica lo realizado. Una vez revisado la tarjeta finaliza su ciclo de vida y pasa a la columna “Finalizado”.

La herramienta utilizada para crear el tablero Kanban es Trello, la misma es gratuita y a través de extensiones de terceros permite la generación de reportes de progreso y mediciones de rendimiento en relación al tiempo, así como también permite poner un límite de tareas en cada lista.

Para la generación de las tarjetas en “Trabajo futuro” se utiliza la información contenida en el GDD (Game Design Document). Este documento describe en alto nivel el diseño completo del videojuego. Generalmente el GDD se finaliza antes de

comenzar el desarrollo del juego. Debido al carácter dinámico del proyecto y al trabajo iterativo y en constante revisión, se decidió ir desarrollando el GDD de forma progresiva con el desarrollo del videojuego.

3.1 Uso del framework Unity3D

Es uno de los motores de desarrollo de videojuegos más reconocidos del mercado, con un gran crecimiento en la industria. Incluye un editor 3D de gran ayuda para diseñar y desarrollar juegos [5]. Provee un sistema de partículas de gran calidad y un motor de física, entre tantas otras características.

Lost in the Abysm aprovecha muchas de las características que provee Unity3D. También se aprovechó el uso de herramientas desarrolladas por terceros que se pueden integrar al entorno de desarrollo.

Algunas de las características de Unity utilizadas en *Lost in the Abysm*:

- Efectos de partículas para generar los poderes, efectos naturales, etc.
- Motor de física para las flechas lanza das y otras funcionalidades.
- Optimización del performance, utilizando técnicas como "Lightmapping" y "Occlusion Culling".

3.2 Descripción técnica.

A continuación se describen los aspectos técnicos de desarrollo del juego:

Desarrollado con el motor de videojuegos Unity3D Personal Edition Versión 2017.3.1.

Lenguaje C# utilizando el IDE de desarrollo Microsoft Visual Studio.

Sprites e imágenes en Adobe Photoshop CS6 [6].

Modelos 3D en SketchUp8 y ProBuilder [7].

Música clásica obtenida de Musopen.

Sonidos ambientales extraídos de muestras obtenidas en internet.

Compatibilidad total con joystick.

Plataforma PC, con posibilidad de exportar a diferentes plataformas (Mac, Linux, consolas).

Metodología Agile Kanban, tablero Kanban creado con la aplicación web Trello.com [8].

4 Diseño e implementación

Los scripts son una parte esencial de Unity3D ya que definen los comportamientos de cada objeto en la escena.

Los siguientes cuatro scripts están asociados al objeto del personaje, y funcionan como diferentes capas para mantener una arquitectura organizada.

- Movement: Este script controla el movimiento del personaje. Para esto se basa en los inputs del teclado que se presionan.
- Disparo: Este script es el encargado de los ataques del personaje basándose en los inputs correspondientes, ya sea el disparo básico o las habilidades mágicas.
- Player HP: Este script controla el daño recibido del personaje aplicando a la barra de vida el daño causado por los enemigos.
- Poción: Este script reacciona al input correspondiente, controla las pociones de vida del personaje mediante el uso de una corrutina o hilo la cual puede activarse en paralelo múltiples veces, por lo que al presionar el input múltiples veces, según sea la cantidad de pociones disponibles, la cantidad curada aumenta y el tiempo que tarda en curar la cantidad establecida por cada poción disminuye.

4.1 Poderes

El personaje cuenta con un ataque básico y cuatro poderes que obtendrá progresivamente a medida que avanza en el juego.

- Flecha normal: Inflige daño. Costo de energía medio.
- Flecha de fuego: Inflige daño en el tiempo a través de una corrutina. Utiliza el sistema de partículas para crear una estela de fuego y una explosión al impactar. Costo de energía alto.
- Flecha de hielo: Inflige daño en un 50% y ralentiza la velocidad de movimiento del enemigo en un 50%. Utiliza el sistema de partículas para crear una estela de hielo y una explosión de nieve al impactar. Costo de energía medio.
- Flecha de tierra: Inflige daño e impide el movimiento del enemigo con el que impacta durante dos (2) segundos. Se utilizan las propiedades del componente NavMeshAgent para establecer la velocidad de movimiento del enemigo en cero (0) y el sistema de partículas para crear una estela de estalagmitas y una jaula de estalagmitas al impactar, simbolizando el impedimento de movimiento. Costo de energía medio.
- Flecha de aire: Inflige daño. La velocidad del proyectil es cinco veces más rápida que de la flecha normal. El costo de energía es bajo.

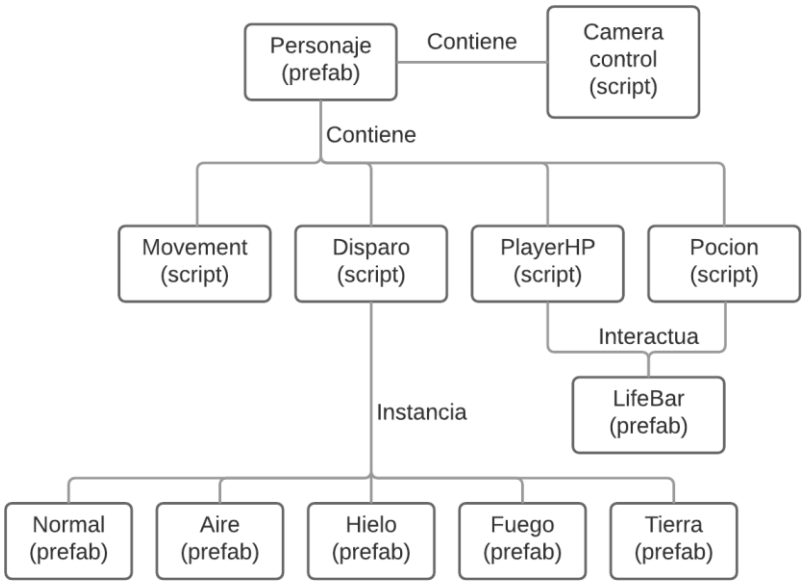

Fig. 4. Diagrama de relación de los componentes del personaje. Los prefab son objetos almacenados para ser instanciados, exceptuando Personaje y LifeBar que están disponibles en la escena desde el inicio.

4.2 Sistema de combate

El combate se puede dividir en dos modos de ataque:

- Modo de Mira: le permite al jugador disparar en cualquier dirección y utilizar una mira para aumentar su precisión (Véase figura 5).
- Modo de Combate: al encontrarse con un enemigo, el jugador tiene la posibilidad de fijar su objetivo al presionar uno de los inputs, de esta forma el objetivo quedará fijo y el personaje cambiará su sistema de movimiento para facilitar el combate con el enemigo (Véase figura 6).

Fig. 5. Modo de mira.

Fig. 6. Modo de combate.

4.3 Enemigos

La implementación del comportamiento de enemigos se realizó a través de scripts en los que se utilizan técnicas (como ser árboles de decisión) para definir el comportamiento de algunos enemigos y para el movimiento se utilizó el componente *NavMesh* y *NavMeshAgent*.

Utilizando el NavMesh se establecen las superficies transitables. El componente NavMeshAgent las detecta y limita el movimiento dentro de las zonas transitables, así como también permite controlar la velocidad de movimiento, la búsqueda del camino más corto hacia el objetivo, etc.

Dicho componente es utilizado para hacer que los enemigos patrullen por cada zona, ya que a través del NavMesh se pueden limitar las mismas.

En cuanto al combate en sí, los enemigos trabajan en base a un árbol de decisiones utilizando tres rangos de visión (los mismos son modelos 3D que funcionan como

triggers) y utilizando el método “Physics.Linecast” para determinar si entre el enemigo y el personaje existe algún objeto que bloquee la visión.

Los conos de visión determinan parte del comportamiento del enemigo de acuerdo al tipo de enemigo. En el caso de un arquero arquero el funcionamiento es el siguiente:

- Cono interno: Si el personaje está dentro y no hay ningún objeto bloqueando la visión el enemigo atacará a la vez que retrocede para alejarse del personaje.
- Cono medio: Si el personaje está dentro y no hay ningún objeto bloqueando la visión el enemigo atacará.
- Cono externo: Si el personaje está dentro y no hay ningún objeto bloqueando la visión el enemigo se acercará hasta entrar en el cono medio.

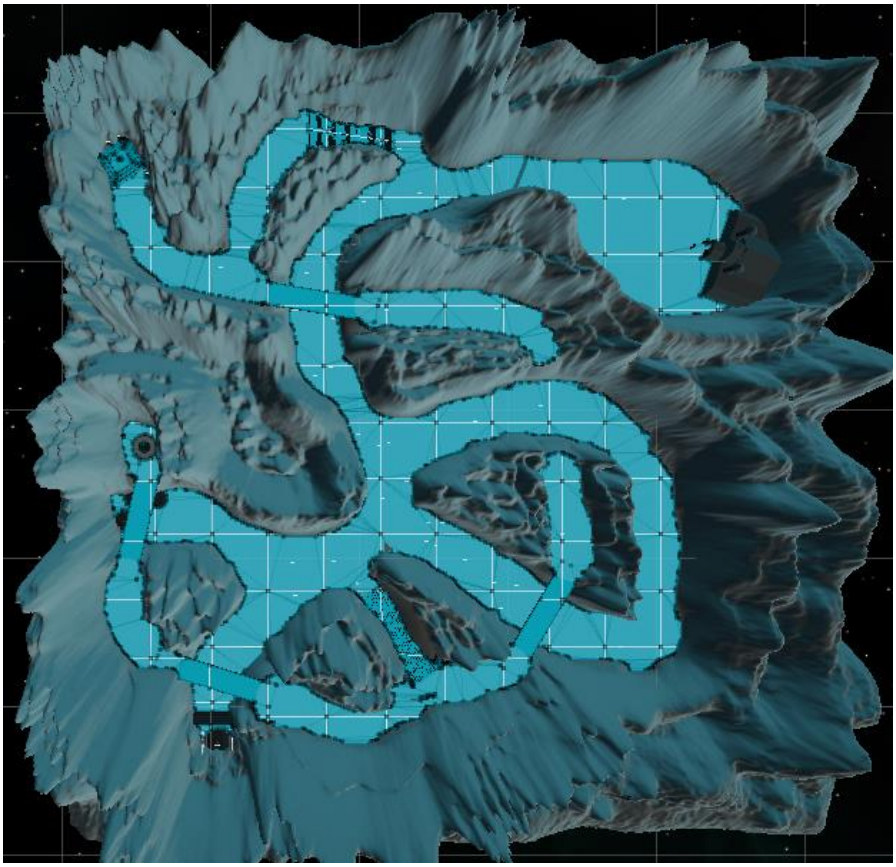

Fig. 7. Mapeado con NavMesh. Las superficies cubiertas en celeste son transitables por los enemigos.

4.4 Niveles

Lost in the Abysm tendrá 5 zonas (niveles), los cuales serán escenas diferentes.

Cada escena tiene sus propios objetos, que definen lo que existe en el nivel.

Quien se encarga del manejo de las escenas es el script "*GameManager*". A su vez, cada nivel tiene un "*LevelManager*" el cual, al derrotar al boss (jefe) del nivel recibe un mensaje enviado por el mismo, lo que activa el portal hacia el siguiente nivel.

Todos los niveles tienen una estética similar.

A continuación se describe el proceso de creación de cada uno de los niveles:

En primera instancia es creado el "heightmap"[9] en Photoshop, este height map es utilizado en la herramienta de Unity3D "terrain" para generar el terreno y a través de un complemento de tercero es exportado como modelo 3D para reducir los polígonos y realizar los retoques necesarios en SketchUp 8. Finalmente el modelo terminado es colocado en la escena de Unity correspondiente.

Una vez colocada la zona en su lugar se agregan los elementos estéticos y jugables relacionados con la historia.

Se ubican los enemigos y NPC[10] en el mapa.

Se realiza la optimización gráfica y de performance mediante técnicas como la disminución de distancia de renderizado, "Occlusion Culling", "Lightmapping" entre otras.

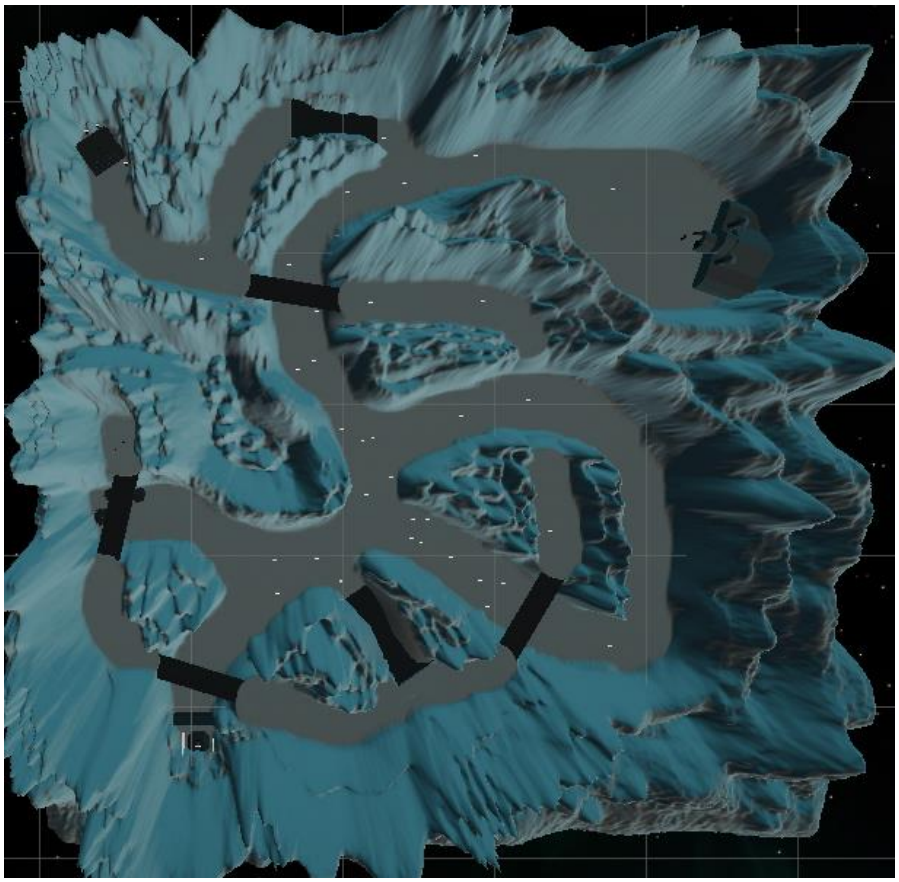

Fig. 8. Primera zona vista desde arriba.

4.5 HUD (Heads Up Display)

Se considera HUD[11] a toda la información que se muestra en pantalla. La podemos separar en las siguientes secciones:

- Indicador de personaje principal (salud, energía, poderes).
- Indicadores de los enemigos (salud, indicador de “objetivo fijado”).
- Tutoriales de poderes y movimientos.

El HUD es creado utilizando los componentes de interfaz que provee Unity3D. El diseño fue ideado y creado con Photoshop.

Los tutoriales de los controles son mostrados al inicio de cada zona.

5 Audio

La música en primera instancia es tomada desde Musopen, sin embargo está en progreso la creación de una banda sonora original en piano y sintetizador, buscando que la misma se adapte a la jugabilidad para así lograr que el jugador se sienta dentro del juego.

Los sonidos ambientales fueron extraídos de muestras obtenidas en internet.

6 Conclusiones

Se destaca la gran cantidad de áreas de desarrollo de juegos que está abarcando la creación de *Lost in the Abysm*, lo que permitió ampliar significativamente el nivel de conocimiento y apreciar la relación que existe entre cada una de estas áreas tan diversas.

La utilización de Kanban como metodología, permitió avanzar a medida que se progresaba y se capacitaba. La creación del tablero digital Kanban fue una incorporación muy valiosa, ya que permitió trabajar de forma virtual, manteniendo el proyecto organizado. Hubiera sido imposible llevar a cabo un proyecto de este estilo con una metodología estructurada y prescriptiva.

El desarrollo de *Lost in the Abysm* brinda la experiencia y confianza necesaria para poder encaminar nuevos y más ambiciosos proyecto en el futuro.

Referencias

1. Xie, J. (2012, July). Research on key technologies base Unity3D game engine. In *Computer Science & Education (ICCSE), 2012 7th International Conference on* (pp. 695-699). IEEE.
2. Jolma, V. (2014). Animated low poly characters.
3. González Sánchez, J. L. (2010). Jugabilidad. Caracterización de la experiencia del jugador en videojuegos.
4. Ahmad, M. O., Markkula, J., & Oivo, M. (2013, September). Kanban in software development: A systematic literature review. In *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on* (pp. 9-16). IEEE.
5. Unity Technologies, Unity Editor, http://Unity3d.com/Unity/editor/
6. Adobe, Photoshop website, http://www.adobe.com/es/products/photoshop.html
7. ProCore, ProBuilder website, http://www.procore3d.com/probuilder/
8. Trello website, https://trello.com
9. Heightmap, https://en.wikipedia.org/wiki/Heightmap
10. NPC, https://es.wikipedia.org/wiki/Personaje_no_jugador
11. HUD, http://es.wikipedia.org/wiki/HUD_(inform%C3%A1tica)