ESCUELA NACIONAL DE ESTUDIOS SUPERIORES
UNIDAD MORELIA

Administración de proyectos

Profesor: Héctor Alonso Guzmán Gutiérrez

Arquitectura

ENESoftware

Jorge Antonio Camarena Pliego
David Calderon Ceja
Keshava Tonathiu Sánchez Barbosa
Stephany Dzoara Vargas Mier

22/10/2019

Semestre 01-2020

Versión 0.1

Índice

Introducción

En este documento, se pretenderá planear con anticipación, con la intención de prevenir confusiones o desórdenes potenciales futuros en la implementación en Unity, la arquitectura del proyecto de una manera muy general.

Unity es un motor de juego Entidad-Componente-Sistema, lo cual significa que se trabajan muchos componentes globales que heredan casi todos de una misma clase conocida en este caso como MonoBehaviour, y que pueden ligarse a una entidad para proporcionarle más complejidad y atributos a su base. Y la parte del sistema se refiere sencillamente a que el programa corre continuamente en todo momento efectuando acciones sobre todas la entidades cada cierto intervalo de tiempo (usualmente pequeño) dependiendo de sus componentes.

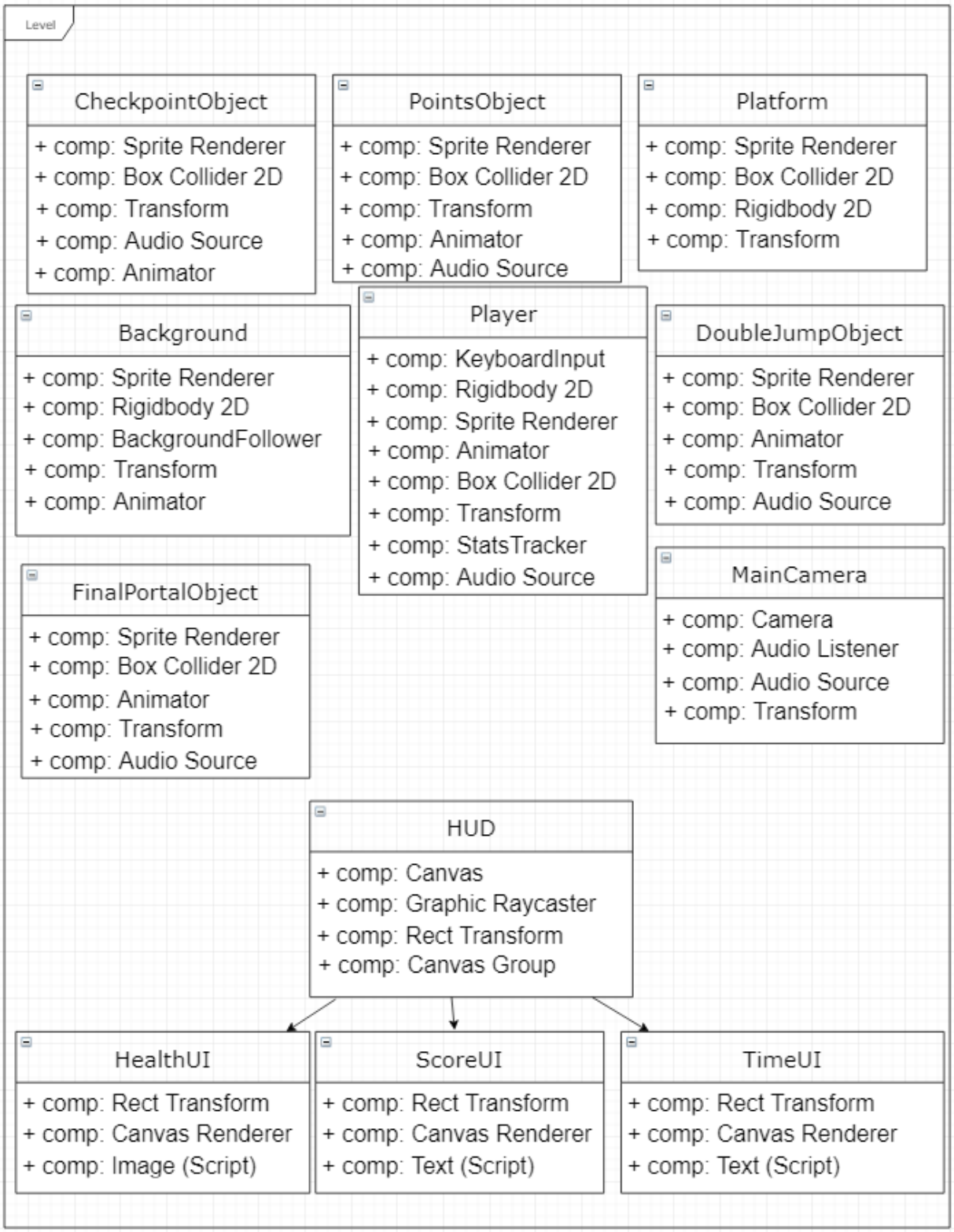
El problema con este paradigma que trabaja Unity es precisamente que sufre de la extensiva herencia desde una sola clase base, MonoBehaviour nuevamente. Este diseño a bajo nivel del motor tiene el propósito de proporcionar máxima libertad y facilidad a los desarrolladores, poniendo muy pocas restricciones y teniendo casi todas las clases e instancias las mismas propiedades.

Debido a este problema, es muy difícil representar la estructura de Unity, junto con la de nuestro propio proyecto en un diagrama de clases, dado que tendríamos casi todo apuntando a unas pocas clases. Será mejor entonces pensar en una representación similar a un diagrama de clases, pero ligeramente distinta, pensando posiblemente en los componentes como si fuesen atributos o métodos de alguna clase de objeto dentro del juego que nosotros mismos representaremos, de menos, con los “tags”.

Diagrama Entidad-Componente

En el diagrama a continuación se representan las clases de entidades o GameObjects distintas que estarán en el juego, especificando todos los posibles componentes que tendrían a la hora de la implementación. Se vuelve a hacer hincapié sobre el hecho de que este diagrama no es de clases, y que por ello las flechas usadas no representan herencia, sino jerarquía de objetos en una escena, tal que la cabeza de la flecha apunta al objeto hijo ligado a su padre, cuyos componentes los afectan o influyen sobre ellos también. Quizá pueda pensarse como una herencia de componentes de un objeto a otro.

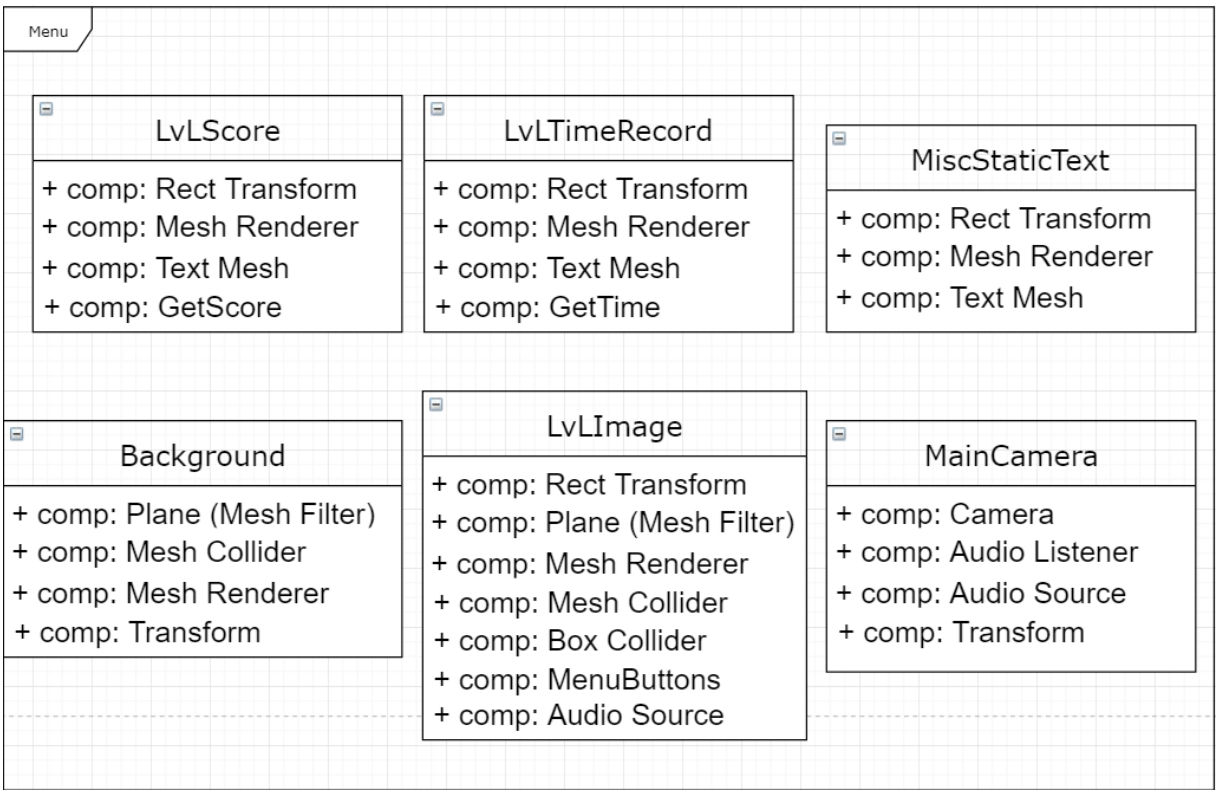
Diagrama de Escena de Nivel



Varios de los componentes especificados son scripts que deberán ser programados, como los componentes BackgroundFollower, KeyboardInput, StatsTracker (que llevará cuenta del número de vidas, si el checkpoint ha sido alcanzado, si es posible brincar, puntaje, tiempo restante, entre otras variables que deben estarse

actualizando).

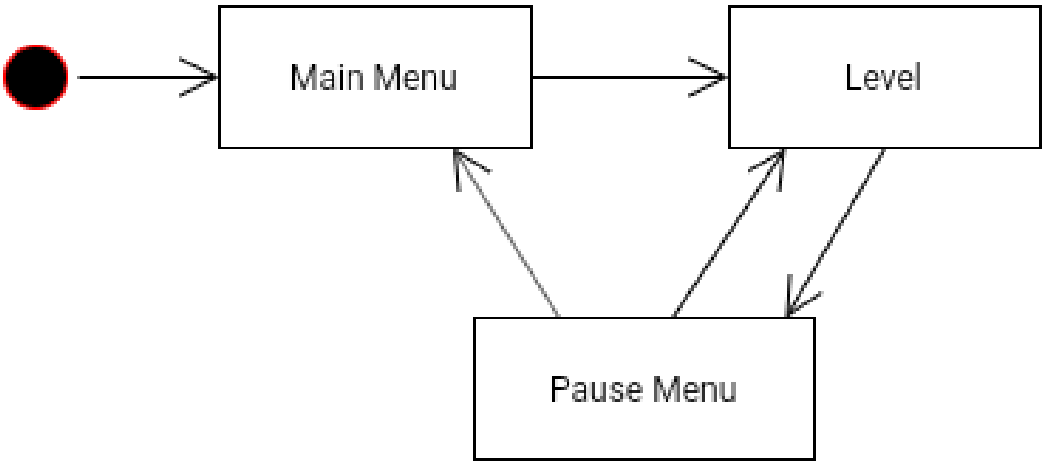
Diagrama de Escena de Menú Principal



Aquí los componentes a programar serían GetScore, GetTime (que obtienen y despliegan el score y el tiempo récord de un nivel en específico), MenuButtons (que determina qué hace cada botón, en este caso los objetos LvLImage que se comportarán como botones).

Diagrama De Navegación

En el diagrama de navegación que está a continuación se muestra como el jugador podrá pasar de una escena a otra de Unity dentro del juego. Es bastante sencillo y simplemente representa cómo el jugador inicia en la pantalla principal del juego Main Menu, a partir de la cual puede elegir entrar a cualquiera de los niveles que desee. Y la forma en que podrá regresar a dicha pantalla desde un nivel es a partir del menú de pausa del juego mientras se está jugando un nivel, sea porque el jugador lo abrió o por la finalización del nivel automáticamente.



Casos De Prueba

CP-1 Input

Caso de Prueba 1 (flujo normal)

Entrada	Resultado Esperado
Probar todas las teclas de movimiento definidas: “flecha a la izquierda”, “flecha a la derecha” y la barra espaciadora Y probar entrada del mouse en las escenas de Main Menu y Pause Menu	Se espera que el personaje se mueva en las direcciones correctas de izquierda, derecha y arriba (saltando) respectivamente, verificando que el salto solo pueda repetirse tras regresar al suelo. Y con la entrada del mouse esperar que todos los botones en Main Menu y Pause Menu sean activados al hacer click en sus áreas

Caso de Prueba 2 (flujo alterno)

Entrada	Resultado Esperado
Se prueban teclas distintas y entrada del mouse en múltiples escenas (Main Menu, Level o Pause Menu)	Se espera que no ocurra ningún error, bug o crash del programa

CP-2 Muerte

Caso de prueba 1 (flujo normal)

Entrada	Resultado Esperado
Probar la caída del personaje en algún vacío del suelo y el agotamiento del tiempo restante del nivel	El personaje entra en una animación de muerte y su GameObject correspondiente es destruido, desapareciendo de la pantalla

Caso de prueba 2 (flujo alterno)

Entrada	Resultado Esperado
Jugar un nivel sin caerse del suelo ni agotarse el tiempo	Que no se active la muerte del personaje y que el nivel sea finalizado correctamente desbloqueando el siguiente nivel (si no lo está ya) y registrando tiempo y puntaje récord

CP-3 Comportamiento de Objetos de Puntaje y Doble Salto

Caso de prueba 1 (flujo normal)

Entrada	Resultado Esperado
Hacer colisión con un objeto de puntaje y doble salto	Se espera que cada objeto efectúe su acción predefinida sobre la escena correctamente con el objeto de puntaje incrementando el puntaje del jugador, el de doble salto permitiendo un único salto adicional en el aire antes de volver a tocar una plataforma.

No hay flujo alterno

CP-4 Comportamiento de Objetos de Checkpoint

Caso de prueba 1 (flujo normal)

Entrada	Resultado Esperado
Llegar a la región del checkpoint y probar múltiples muertes y regresos al menú principal tras haber activado este objeto en un nivel	Se espera que al morir o al regresar al nivel desde el menú principal, que se le brinde la opción al jugador de regresar al punto del checkpoint o reiniciar el nivel desde cero en todas las ocasiones

Caso de prueba 2 (flujo alterno)

Entrada	Resultado Esperado
No llegar al checkpoint o perder todas las vidas antes de finalizar el nivel tras llegar al checkpoint	Se espera que bajo ninguna circunstancia se presente la opción de continuar el nivel desde su checkpoint