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

05/09/2019

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.

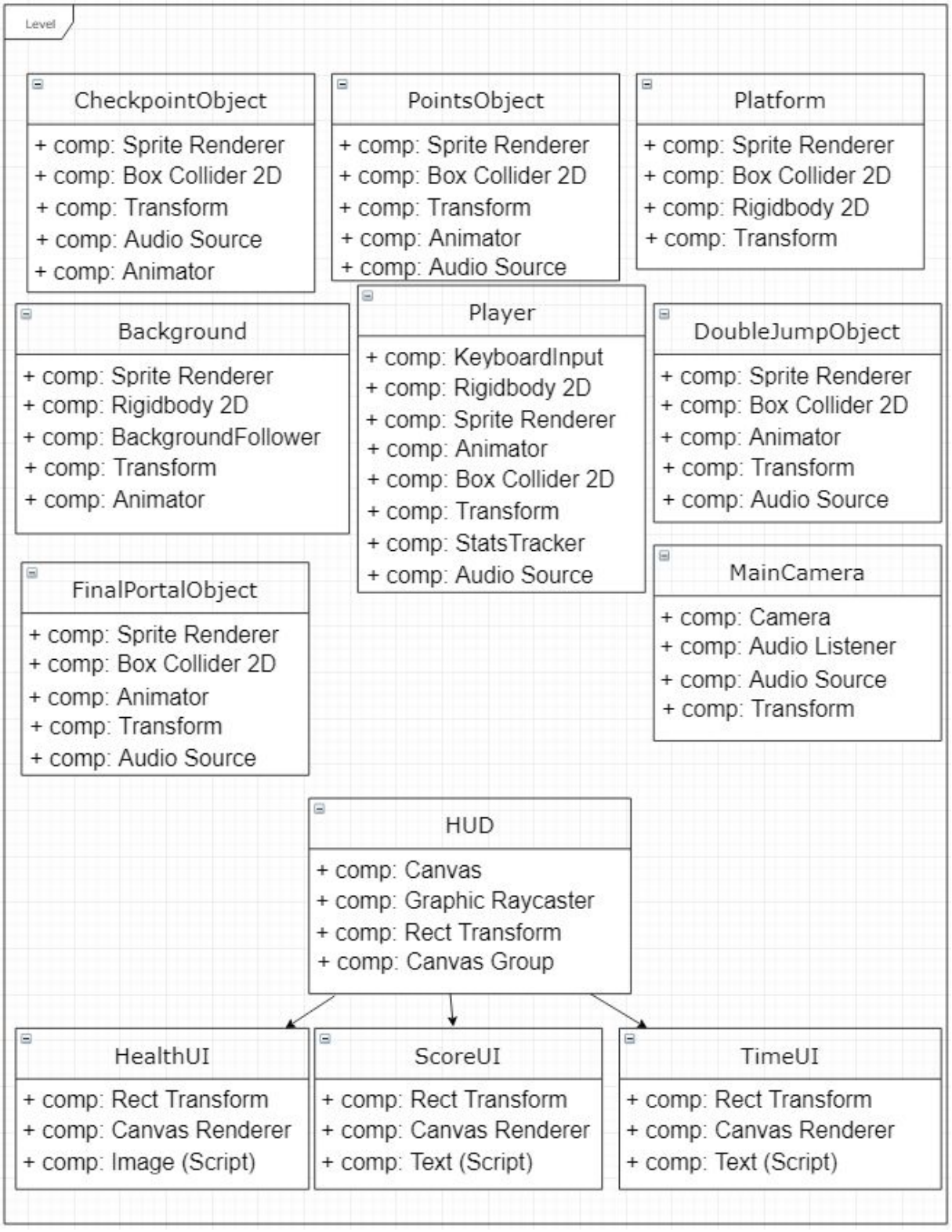
El problema con este paradigma que trabaja Unity es precisamente que sufre de la extensiva herencia desde una sola clase base, MonoBehaviour nuevamente. Este diseño a bajo nivel del motor tiene el propósito de proporcionar máxima libertad y facilidad a los desarrolladores, poniendo muy pocas restricciones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

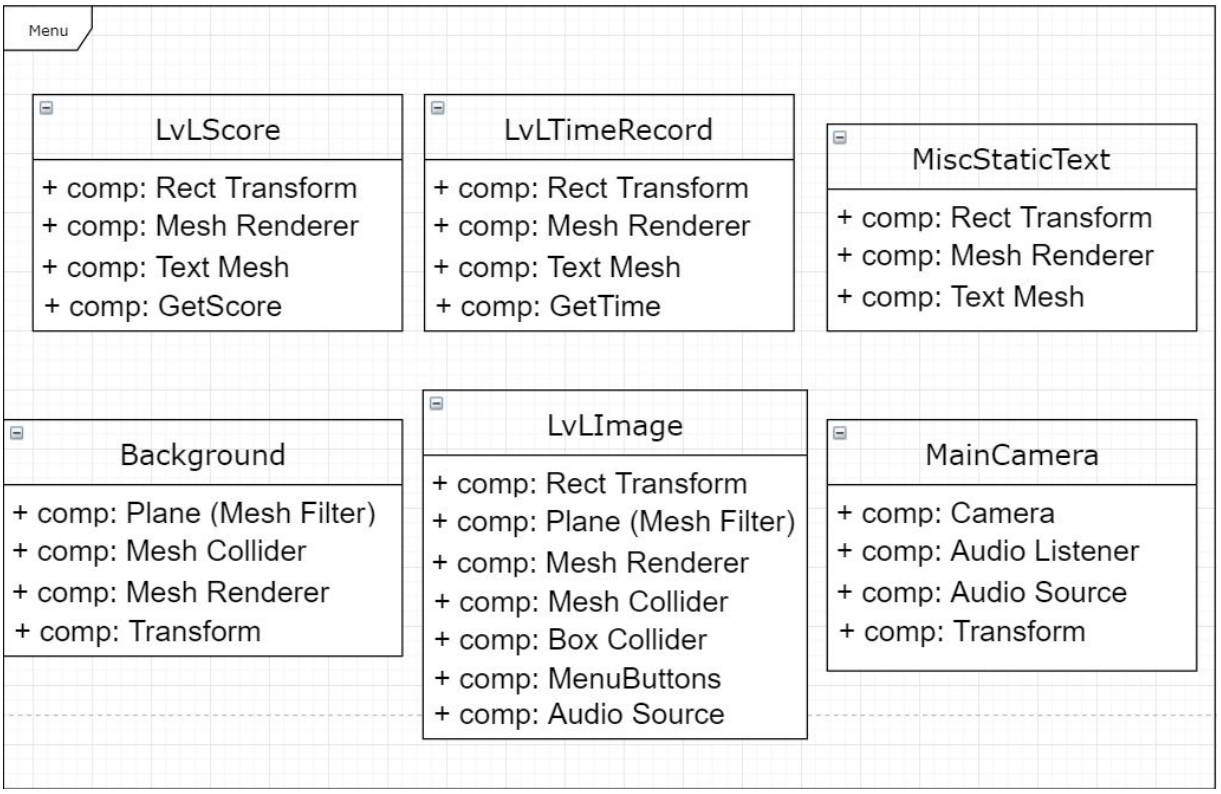
Diagrama de Escena de Nivel



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).