# **UNIVERSIDAD NACIONAL DE SAN AGUSTÍN**

**MULTIMEDIA Y REALIDAD VIRTUAL**

**Tower Defence**

****

**FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS**

**ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS**

**Integrantes:**

**· Chaucca Sacsi Yonathan**

**· Pauca Collanqui Gregorio**

**Arequipa, Lunes 07 de Julio 2019**

**Introduccion**

Tower defence es un juego de estrategia, donde los enemigos pasan por un laberinto y el objetivo principal es evitar que lleguen a la base. Eliminandolos con torres que los ataquen en cada tramo del camino.

La dificultad es que no se puede contar con torres ilimitadas, y mientras más enemigos se elimina se contará con más recursos para adquirir más torres.

Este juego se hará aplicando técnicas de Realidad Virtual,, además de utilizar los scripts en lenguaje C# para darle la lógica al juego.

**Desarrollo**

Para el desarrollo de este juego es necesario la utilización de tres herramientas de desarrollo.

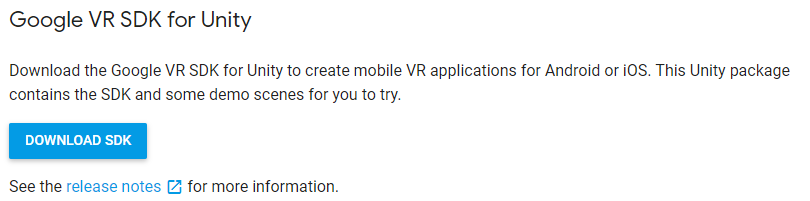
* Unity version 2017.4.24.f1
* Microsoft Visual Studio 2017
* Google VR





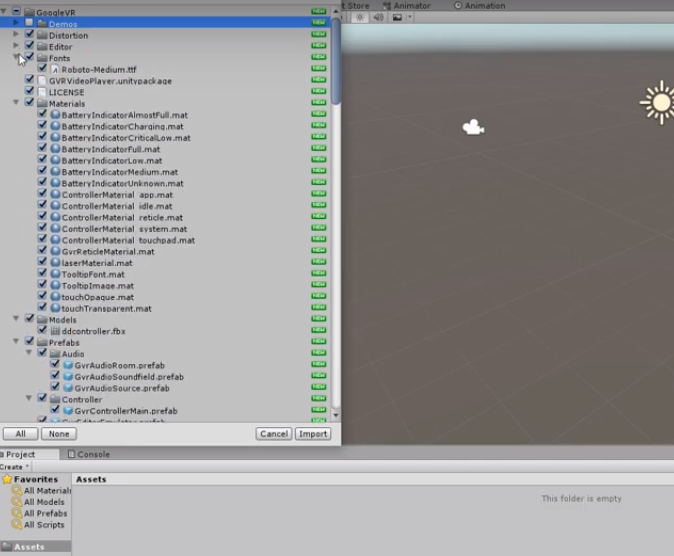
**Configuración de entorno UNITY**

Lo primero es crear un nuevo proyecto en UNITY 3D, y agregarle el Google VR SDK para el manejo de Realidad Virtual integrada con UNITY.

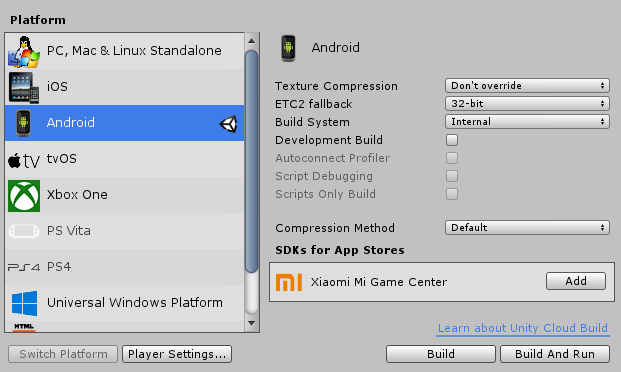


Importamos este API a UNITY.

De esa manera ya podemos usar esa herramienta de Realidad Virtual en UNITY.

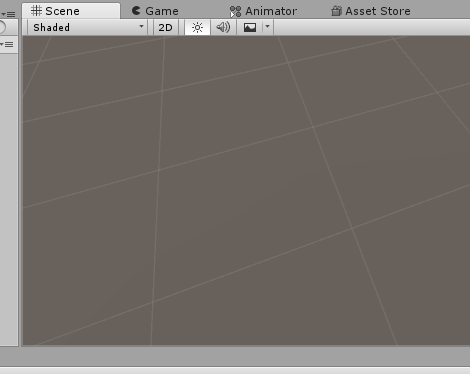


También configuramos UNITY para que nuestro proyecto sea compatible con cualquier dispositivo Android.

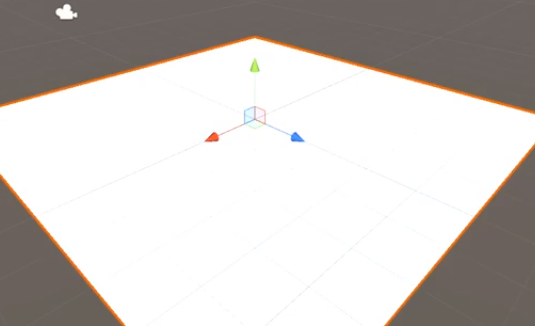


**Agregado de texturas y relieve**

Puesto que Unity nos muestra una escena estándar, tenemos que añadir texturas empezando por el fondo.

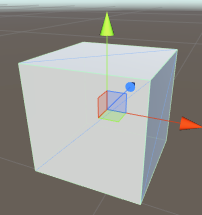


Para ello es necesario crear un Plano, donde ubicar nuestra escena y donde posicionar nuestra imagen de fondo.

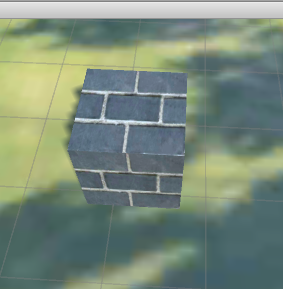


**Creando el laberinto**

Para poder crear el laberinto haremos uso de Cubos posicionados en forma de un camino cerrado. Como los Cubos por default vienen con un color estándar, aplicaremos textura también.



El resultado sería el siguiente:

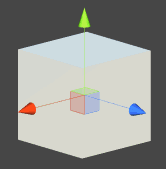
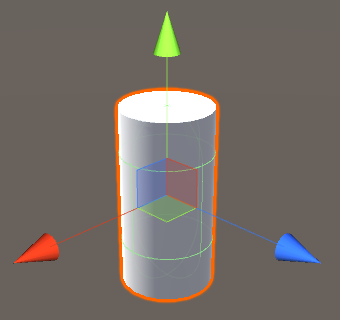


Una vez que tenemos los cubos, hacemos copias sucesivas de ellos para poder formar nuestro laberinto.

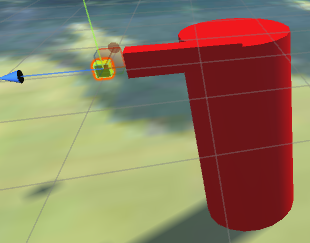
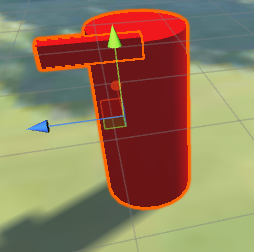


**Creando la torre**

Nuestra torre, la encargada de poder defender nuestro laberinto, se realizará usando un cilindro y un cubo.

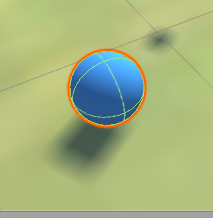
****

Modificando la longitud de sus lados, le damos la forma de una torre y de la misma manera la bala que está emitirá cada vez que un enemigo se acerque.



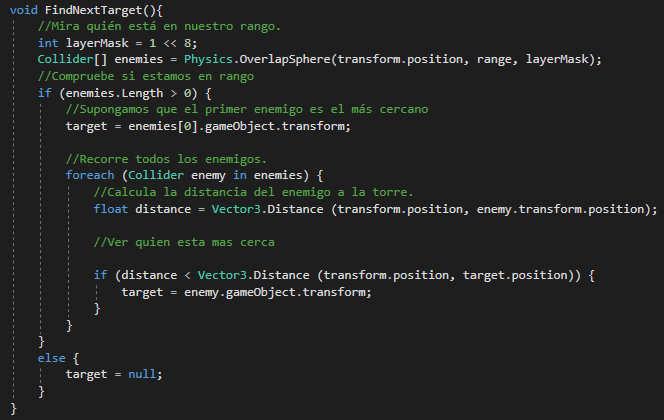
**Creando el enemigo**

Nuestro enemigo, en este caso será una esfera.

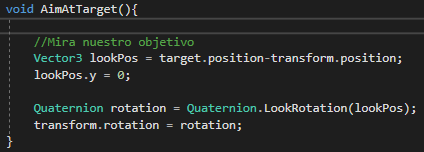


**Dando funcionalidad a la torre**

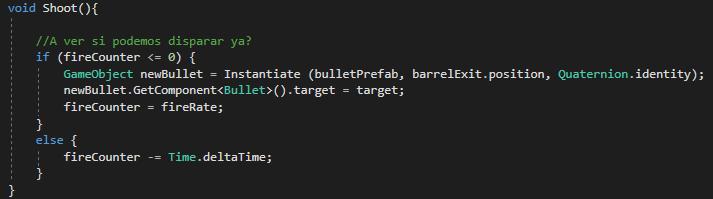
Nuestra torre tiene que tener una rango al atacar a nuestro enemigo. Para ellos creamos el método “FindNextTarget()”, que se encargara de buscar e identificar al enemigo más cercano de todos.



Como nuestro enemigo, estará moviéndose en el laberinto es necesario que la torre pueda seguir su trayectoria, es por ello creamos el método AimAtTarget().

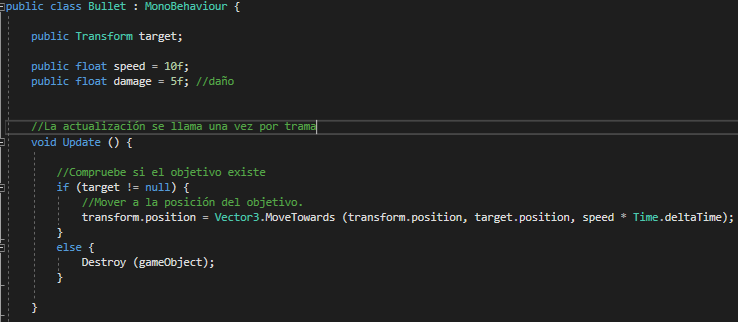


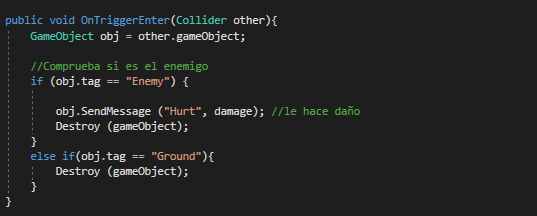
Y por último, nuestra torre debe atacar al enemigo disparando balas, siempre y cuando esté en su rango. Para eso creamos el método Shoot().



**Dando funcionalidad a la bala**

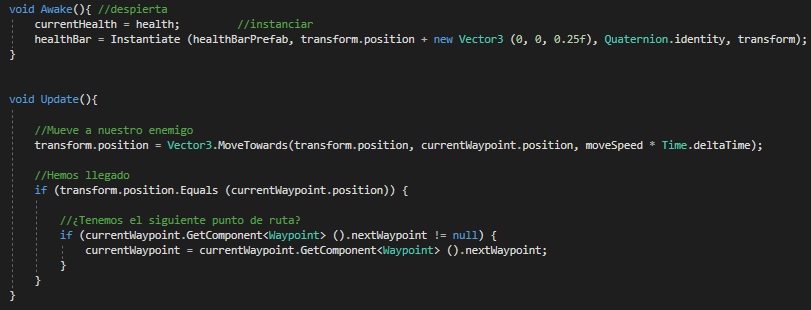
La bala que despide nuestra torre también, necesita funcionalidad ya que es necesario que pueda impactar al enemigo y compruebe si realmente es el enemigo es por ello que creamos el script Bullet, donde tendrá todas esas funcionalidades.

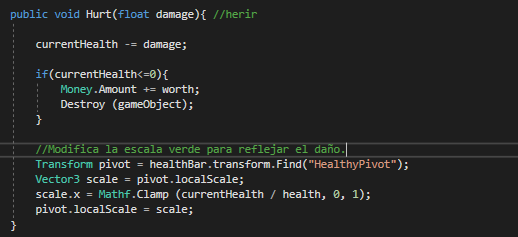




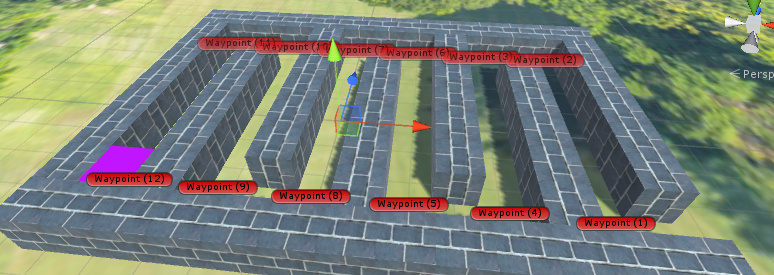
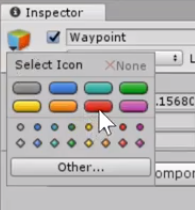
**Dando funcionalidad al enemigo**

Nuestro enemigo también necesita funcionalidades, como “Salud”, “Recorrido”, “Herido”, “Inicializar”. Para poder tener todas estas propiedades creamos el Script “Enemy”.





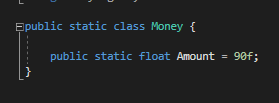
Para poder guiar al enemigo por donde ir, es necesario indicarle el camino y eso lo lograremos mediante objeto “Waypoint”, nuestro laberinto quedaría que esta manera.





**Recompensa por enemigo**

Por cada enemigo que se logra eliminar, se recibirá una recompensa, y esto se verá reflejado en un script “Money”.

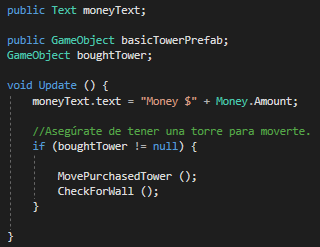


**Creando un cavas - espacio de compras**

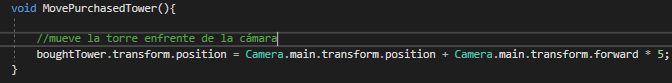
Para que el juego de torres sea más dinámico, creamos un canvas, con un botón de adquisición de torres y con un score que nos marcará el dinero y los enemigos.



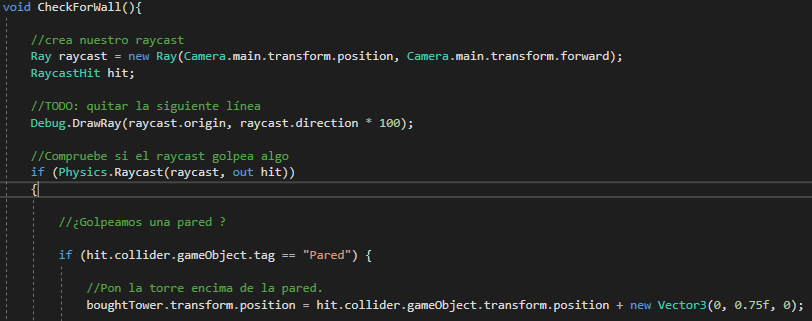
Para que nosotros podamos adquirir una torre, es necesario tener el dinero inicial, método que se llamará en todo momento.

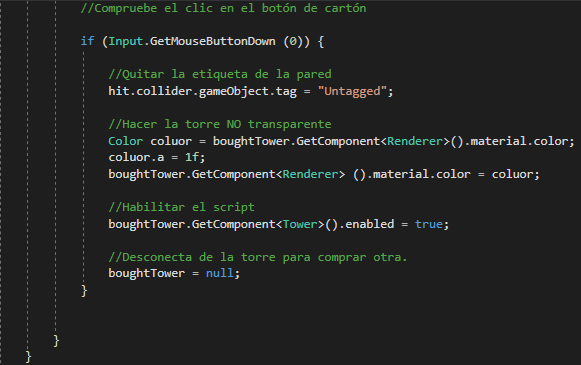


Con ayuda de la cámara, podemos adquirir y mover las torres MovePurchasedTower().



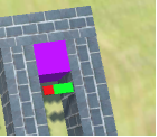
Nuestra torre tiene que colocarse en las paredes de nuestro laberinto una vez hayan sido adquiridas. CheckForWall().

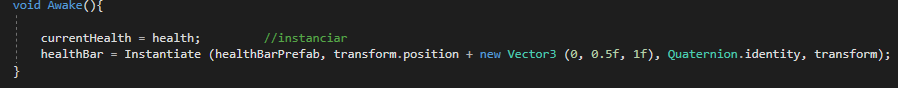


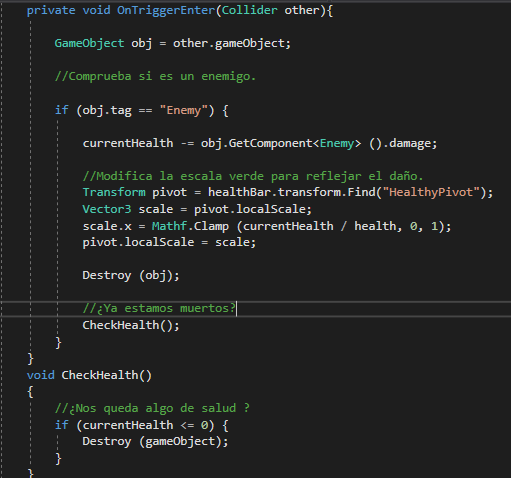


**Barra de vida de nuestra base**

Para que el juego pueda ser emocionante, es necesario fijar cuando ha de terminar. Y eso lo logramos poniendo una barra de vida en nuestra base, si gran parte de enemigos logran entrar, esa barra disminuirá y terminará el juego.







**Generador de enemigos**

Al iniciar el juego, es necesario que algo genere enemigos. Eso se logra con el Script “Spawner”, que hara uso del Script “Wave”



