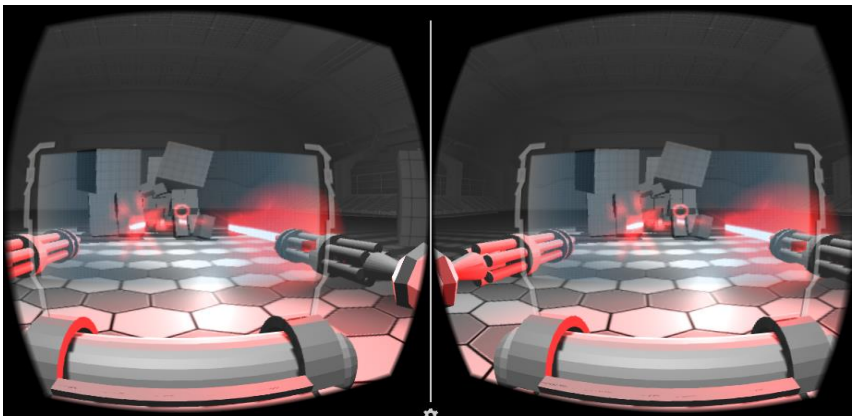
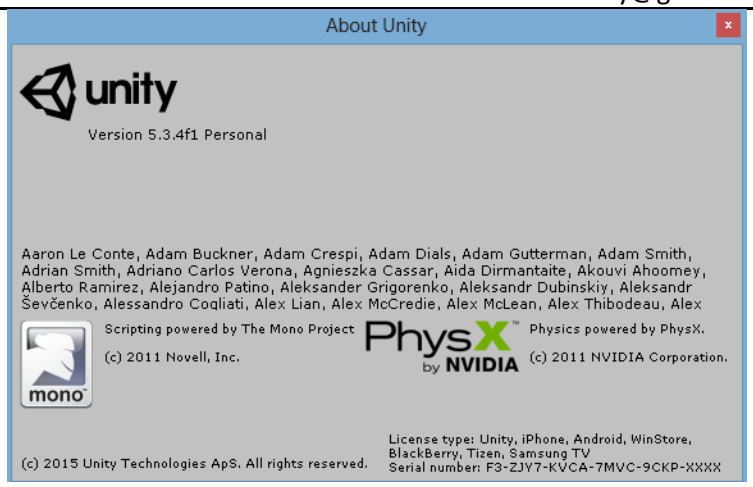
Cardboard VR Turret Controller FPS



1. GENERAL INFORMATION

DATE OF DOCUMENT	14/04/2016
NAME OF THE PROJECT	Cardboard VR Turret Controller FPS
AUTHOR	Michael Soler
UNITY VERSION	5.3.4.F1 PERSONAL
CONTACT	michael.soler.beatty@gmail.com



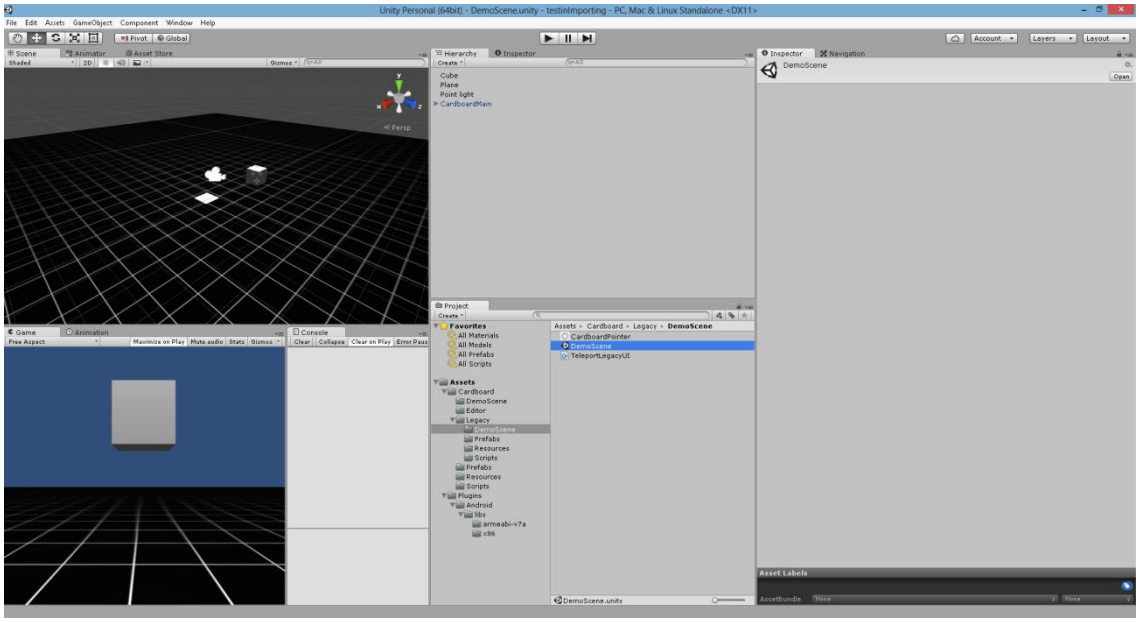
Index

2. IMPORTING INFORMATION

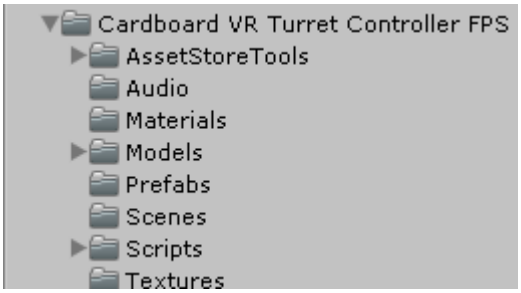
This package works with the "google cardboard" for UNITY that must be downloaded first using the following link:

https://developers.google.com/cardboard/unity/?hl=en

Once downloaded and imported to unity, your project should look like this:



Then, import our package to the project, which will leave you the following folder configuration:

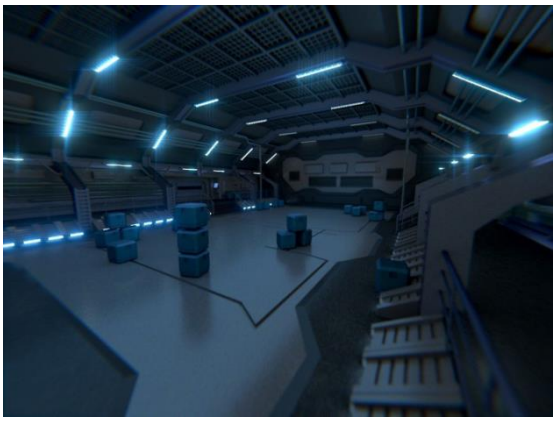


The gaze input collider must be disabled. There is no need to change collider or other game objects.

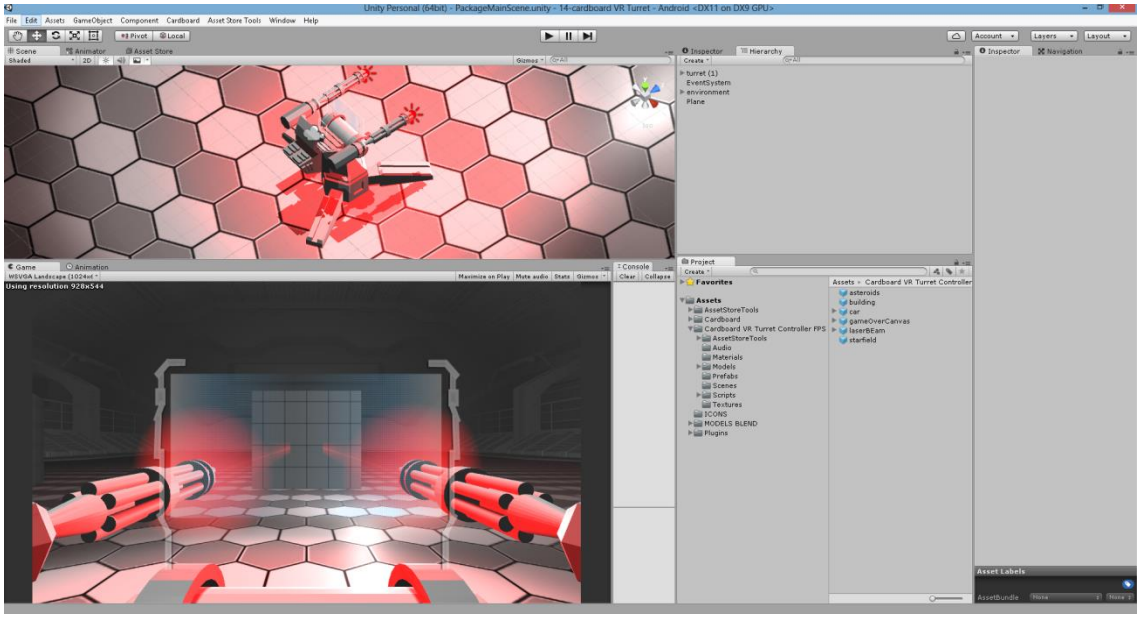
CARDBOARD SDK version="0.6"

Finally you can import the environment from this link:

https://www.cgtrader.com/free-3d-models/architectural-interior/other/the-sci-fi-hangar-scene



Once all is imported, you will find the following:



3. PROJECT DESCRIPTION

This is a VR experience for cardboard in which the user can blow boxes up using a turret. When the magnetic trigger is pressed, the two mini-guns will start pinning and the bullets will start flowing. When the bullets hit the boxes, they will explode into smaller boxes up to a limit size. The movement of the turret is controlled by the movement of the head as shown in the videos.

This package includes:

- 3D models of the turret.

- Textures and materials of the lasers and other objects.

- Turret movement script and shooting script.

- Realistic shooting sound effects.

- Ground hexagonal texture.

- Scripts that manage the creation of smaller debris (explosion effect of the boxes).

- A main scene with the basic space game.

- All gameobjects have a futuristic style.

- Documentation with importing information and script reference.

4. LAYERS, TAGS AND COLLIDERS

LAYERS

All objects are placed on the default layer.

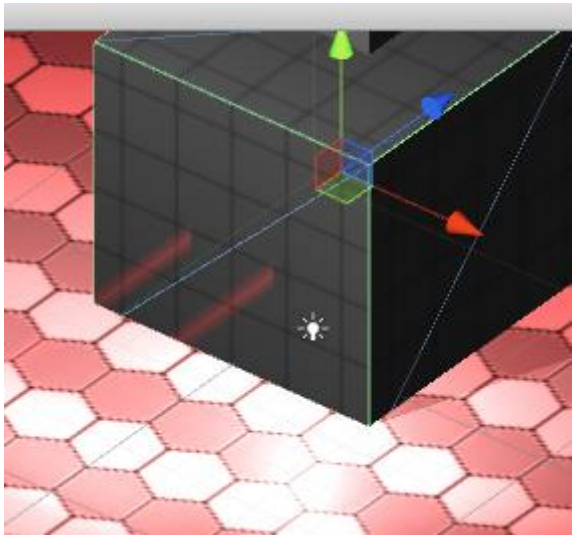
TAGS:

TAG	description
player	Player's cardboard head.
ship	Player's space ship.
environment	Buildings/other things.
asteroid	Debris with their own rotation

*Other gameobjects are untagged.

COLLIDERS

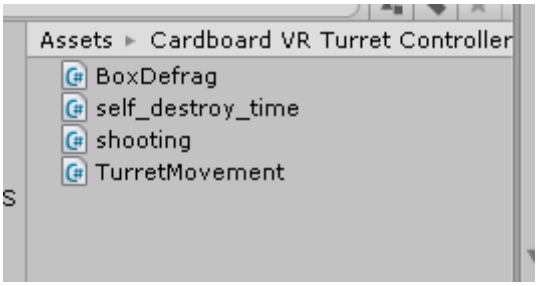
The boxes use a box collider, and the bullets use another box collider. These colliders trigger the exploding effects:



Check if the "gaze pointer cursor" collider is disabled:

It is important to import the cardboard package correctly, and check if these colliders are working properly.

5. SCRIPTING INFORMATION



We explain each script with some detail in the following table:

- **BoxDefrag.cs:**

  This script simulates the debris that are created when the user shoots at the boxes.

IMPORTANT VARIABLES
float u;→ this is the size of the side of the cube. **public Transform container;→** this is the parent for future boxes.
IMPORTANT FUNCTIONS

- **TurretMovement.cs:**

  It is used to move the turret with the head set.

IMPORTANT VARIABLES
public float[] alpha,beta,gama→ these are the arrays used to determine the limit position of the head to add forces and torque to the ship. **public float torqueCoef, forceCoef →** they are used to change the sensitivity of the movement of the ship.
IMPORTANT FUNCTIONS
Void FixedUpdate()→the position of the "seat" is updated using a Quaternion.lerp function. This allows to have a very smooth transition.

- **Shooting.cs:**

  This can be considered the main script of the package because it manages the bullet instances and sound effects.

IMPORTANT VARIABLES
public AudioSource audio; → this is the audio source with the bullet/minugun sound effect. **public GameObject prefab;** → this is the reference to the bullet prefab. **public float speed=40.0f;** → this is speed of the bullet. **public float timeBetwShots=0.5f;** → this is time between bullets. **public bool weapons_hot;** → this Boolean checks if the weapon is still hot and disables shooting capabilities. **public Transform[] weaponBarrel;** → these are the two starting points in which bullets will be created. **public float rotSpeed=0, maxRotSpeed=10;** → these are the speed limits for the rotation of the barrels.

public bool weaponOn=false;→ this is the Boolean that cheeks if the trigger event has occurred and the weapons are on.
public UnityEngine.UI.Image[] heatImages; → these are the heating bars that appear in the canvas.
public float heatingTime; → this value indicates the time that the weapon will work before stopping because of a temperature problem.
public float bulletDispersion; → a higher value will modify the trajectory of the bullets.
Light L1,L2; → these are the lights used to simulate the bullets.

IMPORTANT FUNCTIONS