Giorgio Giudice 4/27/2020

Final Project Report

For my final project I decided to use **a-frame**.

A-Frame is a JavaScript/HTML web framework that allows to code websites with VR capabilities. It handles multiple primitive shapes, exactly like blender. Since it is highly customizable through JavaScript (not without extreme pain), it is possible to handle collision management and handle/personalize any element in the code (in theory, because I did find several issues while doing it).

**Type of game:** Endless

**Controls:** Movement of the goggle if using a headset, or arrow keys (alternatively a/d) if used with keyboard. It is all handled with JS code in gameEngine.js

**Object of the game**: dodging as many bullets as possible by moving the cursor.

**Meshes**:

* Bullets are spheres
* The player’s cursor is a simple cylinder

**Animation**: The bullets are moving toward the user using “a-animation”. In Blender we could do that in the timeline, while here the position of the object is hard-coded in the html file (for example: from="0 0.6 -7" to="0 0.6 1.5"). Of course, everything is expressed in 3D coordinates.

**Mechanics of the game**:

There is a 50% probability for every bullet to appear on the screen, but the computer cannot shoot more than 2 at the same time to always offer the user a chance to escape. Of course, a great advancement would have been to customize the animation of the object so that the speed increases as time goes. I had some ideas about that, but handling a-frame objects with JavaScript is not as straightforward as it is with basic HTML tags. I think I could improve the game by adding that option.

My plan would be to use the funcion “shootBulletsRandomlyLoop” and have the paramenter “intervalLength” change over time.

**Problems encountered**:

* My main problem was that the game would get very slow after running for just a few seconds. The turning point was adding a JS function, removeBullet, that would remove the “child bullet” from the tree generated by the game engine after it goes out of sight.
* Physics: a-frame has theoretically some form of physics, mainly with add-on codes, but I haven’t found much documentation about it, and it requires some time to deeply understand it before using it proficiently.

**Collision Management**:

In the case of my game, when the position of the player is the same of the bullet, then it calls “gameOver()”. I found this to be the easiest way to handle it

**Conclusions**:

I would say that aframe is very easy to set-up and very compatible (it is indeed cross-platform). The problem is that its primitives are still not very customizable via JS or jQuery so there needs to be “hacks” to obtain the wanted results, and many times the best solution is to just find a compromise. I wouldn’t suggest a-frame as a good game development platform, but I would suggest it to make an impressive website with many interactive elements.