



SAPIENZA
UNIVERSITÀ DI ROMA

Engineering in Computer Science

Interactive Graphics – Final Project

Student name: *Alessandro Carnicella 1915407*

Course: *Final Project – Interactive Graphics Sapienza - 2020-2021*

Due date: *October 26, 2021*



Introduction

This report shows the solution adopted in the final project realization.

Game Introduction

First of all, we are going to introduce the protagonist of this game, red and white striped T-shirt, wool cap topped with a pompom. Wally is the protagonist of the children's book series born in the 1980's "Where's Wally?".

The historic game has as a difficult goal to spot Wally in a crowd. With the following project, the idea is to extend this world, up to now in 2d, into a new and three-dimensional world. In particular, this project simplifies the world of objects present in the classic picture books by Wally by hiding one of the distinctive elements of the character, the cap, within the three-dimensional space populated by lights and moving geometric figures.

The game evolves in 10 levels of increasing difficulty. The purpose of the game remains almost unchanged from the original, it is intended to challenge the player in the search for the Wally hat in this new dimension.

Technologies

Three.js

Three.js is a cross-browser JavaScript library and application programming interface (API) used to create and display animated 3D computer graphics in a web browser using WebGL.

Three.js enables the creation of accelerated 3D animations from graphics processing units (GPUs) using the JavaScript language as part of a website without relying on proprietary browser plug-ins. This is possible thanks to the advent of WebGL.

This library provides several features that have been used in the proposed project. In particular, features have been used for the following implementation: Single and multiple object Animation, Scene set up, Geometry creation, Asset load, Fly Control and Light.

Official documentation is available at the following link: <https://threejs.org>.

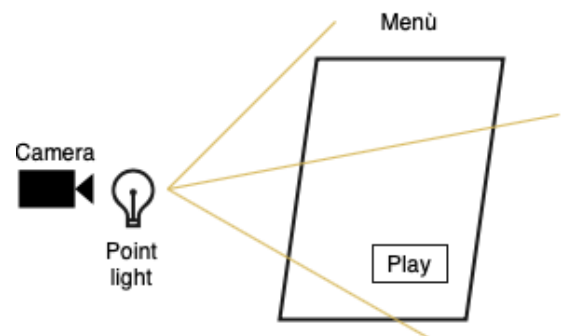
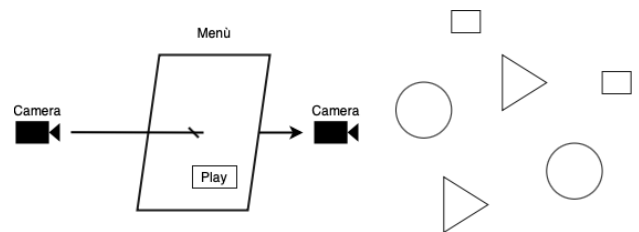
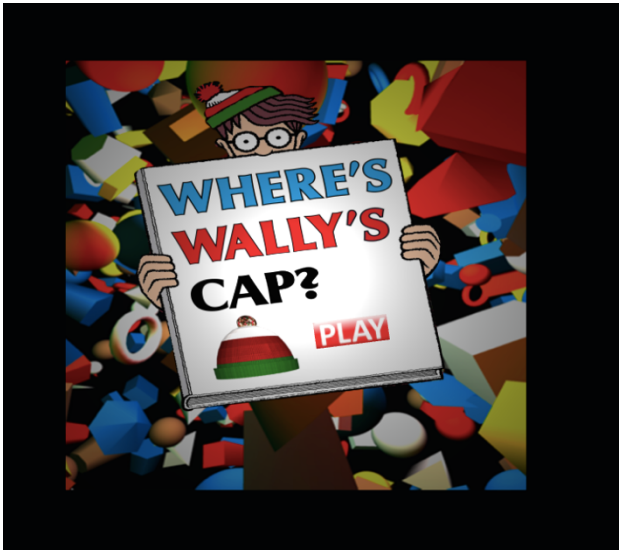
Game elements

Menù

A start menu has been created for each scene, thanks to which it is possible to access the game space. A hierarchical model has been created for each menu to which the various components that make up the menu are linked. Each node of the menu is in fact composed of different objects, in particular for each menu there is a point light that allows to illuminate the component, the texture dedicated to the level and the button for play. The light was placed on the same level as the cam which was also placed in the menu center.

By clicking on the play button, the camera is brought into the game space, making the various preloaded objects that make up the scene visible.

Menù Overview



Game Geometry and Texture

Depending on level, different geometries are created, this geometry has the purpose of confusing the player who is looking for the hidden object. For each scene, the level defines the number of geometries that will be created. At the beginning of the level the geometries are created and subsequently preloaded before the start.

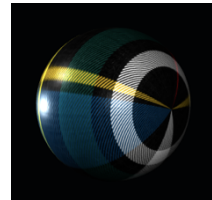
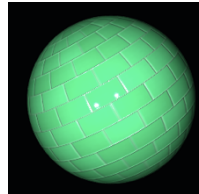
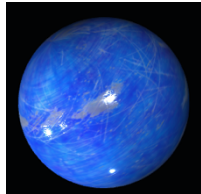
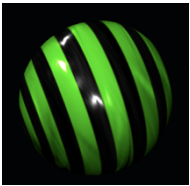
A node is created for each geometry, then the geometry that will be created is created by this possible geometry shapes:

- Ex. `THREE.BoxGeometry(1, 1, 1)`
 - `BoxGeometry(width : Float, height : Float, depth : Float, widthSegments : Integer, heightSegments : Integer, depthSegments : Integer)`
- Ex. `THREE.CircleGeometry(5, 32)`
 - `CircleGeometry(radius : Float, segments : Integer, thetaStart : Float, thetaLength : Float)`

- Ex. `THREE.ConeGeometry(5, 20, 32);`
 - `ConeGeometry(radius : Float, height : Float, radialSegments : Integer, heightSegments : Integer, openEnded : Boolean, thetaStart : Float, thetaLength : Float)`
- Ex. `THREE.CylinderGeometry(5, 5, 20, 32);`
 - `CylinderGeometry(radiusTop : Float, radiusBottom : Float, height : Float, radialSegments : Integer, heightSegments : Integer, openEnded : Boolean, thetaStart : Float, thetaLength : Float)`
- Ex. `THREE.SphereGeometry(15, 32, 16);`
 - `SphereGeometry(radius : Float, widthSegments : Integer, heightSegments : Integer, phiStart : Float, phiLength : Float, thetaStart : Float, thetaLength : Float)`
- Ex. `THREE.TorusGeometry(10, 3, 16, 100);`
 - `TorusGeometry(radius : Float, tube : Float, radialSegments : Integer, tubularSegments : Integer, arc : Float)`

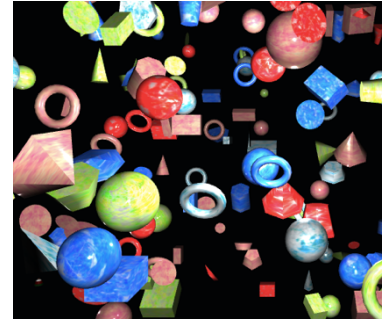
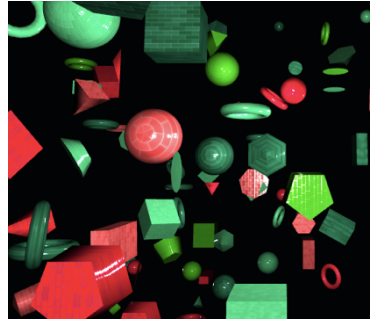
Each geometry created is associated with an initial position and its texture. The initial positions of the geometries are calculated randomly, and the figures fill the whole playing space. The textures are also randomly decided but different types of textures are provided for different levels of difficulty.

Here are some examples of the applied textures:



Once the geometries have been made, for some of them an animation is defined in addition, which on the whole makes it more difficult to identify the hidden asset.

Game
Geometry
and
Texture
Overview



Game Light

When the geometries are created, the lights are also waxed. For each level, in fact, different points of light are generated which help the player to identify the hidden object. The lights are generated in random positions and the number of lights generated changes according to the level.

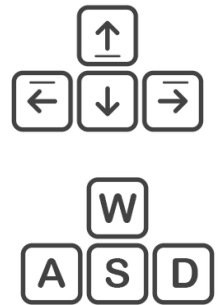
User Interaction

Movments

Movement within the three-dimensional space is made possible through the use of arrows keys, the use of the mouse and WASD keys. The combined use of the keys allows a fluid movement within the game space, in particular they allow the camera to orbit around any target.

The WASD keys allow you to move the camera on the x and z axes relative to the position of the camera. With the arrows keys it is instead possible to move the inclination of the camera on the x and y axes. The same movement of the arrows keys is also made available through the use of the mouse, making the movement experience more natural and intuitive.

Overall, the combination of the previous keys or mouse allows us to move on all three axes in a fluid way.



To realize the movement, we used the implementation of the flight control three.js, a function of the JavaScript three.js library that allows us to realize the movement within 3D models. FlyControls allows navigation similar to flight modes, with this library function we can arbitrarily transform camera into a 3D space without any limitation.

Asset (cap)

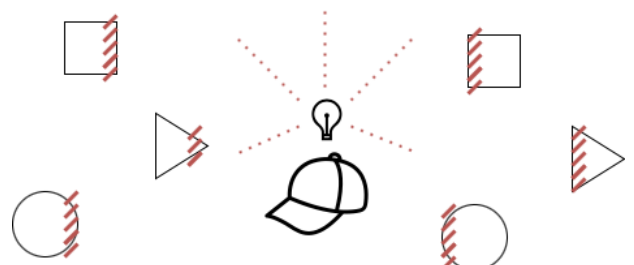
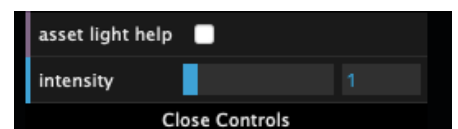
For each scene, in addition to the geometries, the asset that will represent the hidden object is also created. This time, unlike the previous ones, the asset is not generated through the functions for the construction of the geometries, but an already existing object is loaded. Also, in this case the object will have a position randomly assigned to the inside of the playing area.



The asset is associated with an animation that is started only when it is clicked, the animation represents the movement of the asset from one level to another. Thanks to the click on the asset, we have the possibility to change the level with the consequent loading of the menu of the next level and the creation of the geometries that will make up the level with their textures and positions.

Control menù


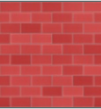





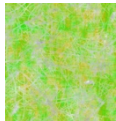
A control menu is also associated with the generated asset, this has the aim of helping the player in case he cannot find the hidden object. In fact, the asset is associated with a point light which makes the object more easily visible. Thanks to this menu it is in fact possible to change the intensity of the light associated with the asset and the color of the light emitted. By playing with these two controls, it is possible to make the hidden object easily visible.



Level description

The game consists of 10 different difficulty levels. Each level, depending on the difficulty, is made up of different elements. In general, a single scene is associated with each level, the components that define the play space are then added to the scene. Among the common elements that we find for the various levels we find the menu which for each level will present a texture created on purpose with the level that is about to start. The number of objects created for each level increases as the level increases, starting from a base number attributable to the succession of Fibonacci. Lights are also added to the scene, the number of which also in this case depends on the level and decreases as the levels advance and the presence of objects. For each level, the texture is also defined which, going to cover the various geometries, starting from the first levels in which there are no textures but only solid colors up to more difficult levels in which the texture of the textures greatly complicates the identification the hidden object.

The following table helps us to better understand the objects present for each level:

LEVEL	# GEOMETRY	# LIGHT	ex. TEXTURE
<i>LEVEL 1</i>	40	5	solid colors
<i>LEVEL 2</i>	46	5	solid colors
<i>LEVEL 3</i>	56	4	
<i>LEVEL 4</i>	72	4	
<i>LEVEL 5</i>	118	3	
<i>LEVEL 6</i>	130	3	
<i>LEVEL 7</i>	208	2	
<i>LEVEL 8</i>	318	1	
<i>LEVEL 9</i>	704	1	
<i>LEVEL 10</i>	1030	1	

Conclusion

This project integrates the use of the basic elements of WebGL, with the use of existing libraries and graphic JavaScript APIs. Based on this project, an interest was certainly born in the implementation of creative algorithm solutions useful for the development of apps aimed at web users. However, it is important to note that this research has mainly focused on learning the basic concepts of graphical web app development and for this reason an excellent starting point for future insights.