Interactive Graphics: Final project report

Maiorano Gabriele 1961926; Lauterio Davide 1934615;

September 16, 2021

Contents

Chapter 1

Introduction

As final project for the Interactive Graphics class we made a single player game in THREE.js built around the concept of collecting items of different type and disposing them in the appropriated container.

The initial idea comes from an oversimplification of the concept of the game **Slime Rancher** by "Monomi park" and Nick Popovich, with a real-world grounded re-contextualization. In the game you have to collect trashes from the city and recycle them in the correct way.

To make it more interesting, the players enemy is the time. A timer starts counting down at the start of each session, the objective is to collect the largest amount of trash, disposing it in the correct container, within the time frame.

1.1 THREE.js

Three.js is a javascript library that allows you to simplify the creation of 3d environments compared to the use of the classic WebGL. A few elements are needed to create the environment:

- The scene: the container of all the objects that will be rendered;
- The camera: which will allow the user to view the scene in space;
- Meshes: physical container of the objects that will be instantiated;
- Rendering engine: which will allow the scene to be rendered at 60 frames per second.

Furthermore, present in the development folder of THREE.js, we used:

- The loader file for 3d objects in gltf format;

- The loader file for audio and sounds;
- The orbit camera that allows you to better manage the focus of the camera with the mouse;

Chapter 2

Game Play

As written before the game objective is to collect the largest amount of trash, disposing it in the correct trash-bin, in a given time window, depending on the selected difficulty level. The Player controls a character whit a Third Person Controller style, always followed by the camera which is always behind it.

The forward/backward movement is controlled with the keys W and S, while the rotation around the vertical axe of the character is controlled via the keys A and D. The player can also adjust the vertical position of the camera with R and F keys. The interaction with the environment is made through mouse input.

The player can also gain more time to play by collecting the hourglasses that pops up in the environment, gaining more or less time depending on the difficulty level.

The player gets a point for each piece of trash disposed. Disposing of trash in the correct bin will award the player with double points.

- **Game Features:**
  - Random generation and placing of collectable items around the game environment;
  - Responsive User Interface that shows the trash type that the player is carrying, if it is;
  - Day/Night cycle to make the game more appealing and to ramp up the game difficulty when the night comes;
  - Added Music and sounds when the user collect or dispose trash;
  - Difficulty selector, which sets the number of items that spawns initially in the scene and the remaining game time.

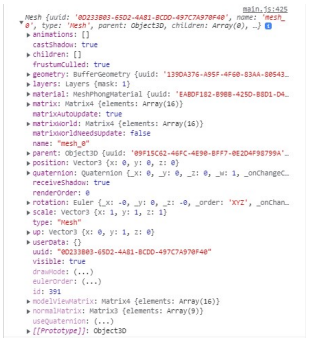
2.1 The Raycasting

To handle the interaction with the objects we implemented ray-casting to select objects from the scene hierarchy. Objects that are analyzed programmatically to determine if they are in the interaction range and if they are interactable at the moment.

The raycast returns a face of the object, we then recursively track that face up via parent nodes until we get the "group object" that is identified by the "trash type" parameter. That parameter allows us also to differentiate what to do when the interaction is possible.

The functionalities of the ray-cast make also possible to subdivide in different classes of interaction those objects by reacting only with objects inserted in predefined arrays, each used for a different group of objects. In this case we have collectables and dispose points (or trash-bins).

(a) Scene



(b) Console Log

Figure 2.1: Raycast interaction

Chapter 3

Graphics

Lighting

3.0.1 Night and Day cycle

The night-day cycle was created thanks to a dirLight main light that simulates the sun, whose position and orientation varies according to the time. Together with javascript functions, the shading effects have been inserted inside the html file via vertex and fragment shaders. Once the variable time reaches the value of about 4, the setting becomes dark, simulating the night. It is at this moment that 4 lights (whose intensity is increased at this time of day) turn on and partially illuminate the scene, simulating the lights of street lamps.

Models and Textures

All models were downloaded via the following sites: Sketchfab and Turbosquid and imported via gltf-loader in the Three.js scene along with their textures. Dimensions and colors of objects have been changed to be more visible to the user.

3.0.2 The main Character Structure

The main character contains, within its clothes meshes, a complex skeleton with 86 orientable bones. Together with the model we have inserted a skeletonhelper that allowed to better identify the bone structure of the model. This made it easier for us to create animations by accessing the quaternions variables of the single bone and adjusting their orientation. The model is composed of the relevant bones:

- rootJoint id:0 ;
- mixamorig Spine id:02;
- mixamorig Spine1 id:03;
- mixamorig Spine2 id:04;
- mixamorig Head id:06;
- mixamorig Left Arm id:15;
- mixamorig Left ForeArm id:16;
- mixamorig Left Hand id:17;
- mixamorig Right Arm id:47;
- mixamorig Right ForeArm id:48;
- mixamorig Right Hand id:49;
- mixamorig Right Upper Leg id:76;
- mixamorig Right Lower Leg id:77;
- mixamorig Right Foot id:78;
- mixamorig Left Upper Leg id:81;
- mixamorig Left Lower Leg id:82;
- mixamorig Left Foot id:83.

3.1 The user interface

The user interface is made all in HTML and css, instantiated via JavaScript code only, to keep it maintainable. Animations are made with css key-frames to avoid useless overhead on the load on the JavaScript part.

3.1.1 The main page

The main page contains :

- Information about the control system;
- The difficulty selector;
- A button to enter the developer mode to create/modify the animations;
- A button to start the game.

Once you click on start game, a loading screen appears (pure css) in which all the models are loaded and the scene is initialized. As soon as the loading is finished, the game begins.

3.1.2 The HUD

An indicator in the lower part of the screen shows the remaining time, while one in the upper-right corner shows the total score.

After collecting trash, a container in the upper-left corner shows the type of object collected and the maximum quantity that can be collected at the same time for that particular type. Once disposed in the correct bin an alert shows that the garbage has been disposed of and that you have earned double points for it.

Furthermore, to help the player, in addition to the P key to pause the game, we have inserted a button with the same purpose.

We've noticed how shadows make the game feel less stable. Consequently, we have decided to insert a button to deactivate them in order to improve their playability.

For some interactions, we made a user interface messaging system to avoid using the "alert" function which interrupts the game flow.

3.1.3 The game over page

When the time runs out, a gameover screen appears showing the chosen difficulty and the final score obtained. From this screen you can then click on try again to return to the main menu.

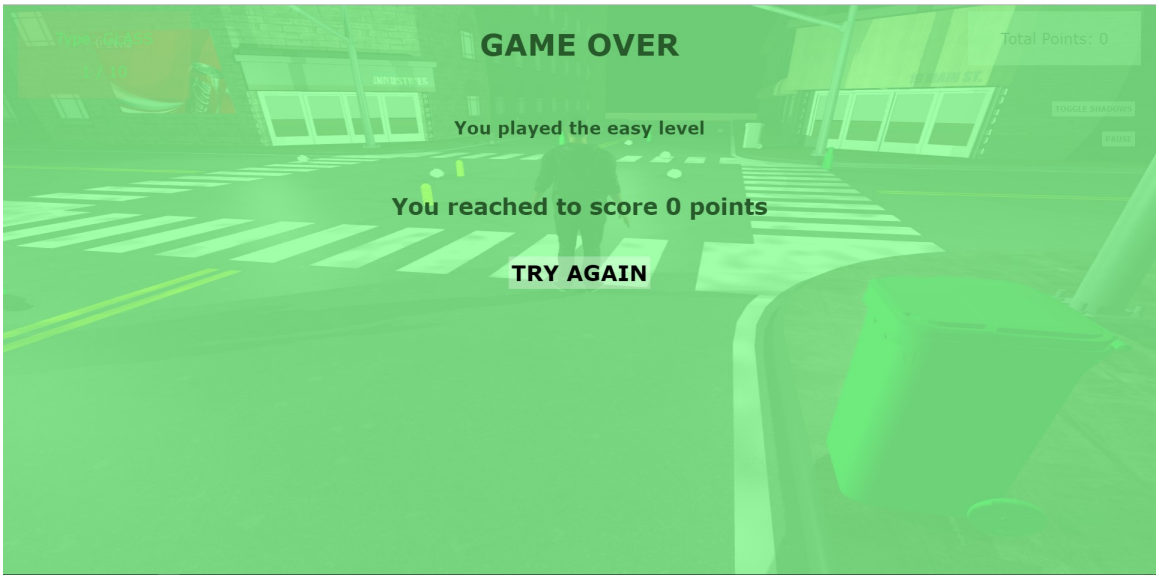


Figure 3.1: Game Over Page

Chapter 4

Animation

The animation of the main character is scripted via the THREE.js animator (Animation Action). Each animation clip is composed of a list of key-frame, each composed of a Quaternion per each relevant bone in the character model structure.
The animations we designed for the character are :

- Idle;
- Walk;
- Collect;
- Dispose.

We tweaked animation parameters, like spacing between key-frames or length and time scaling of the clip, to obtain the best and most natural animation possible with this technique.

To figure out the values for each key-frame we built a tool for animating the character.

4.0.1 The animation tool

The tool can be reached via a button in the upper left corner in the first screen of the game. It features a game view with only the main character displayed and the controls are replaced by a more handy orbit control allowing a detailed observation of the joints of the character, which tended to distort in the previous animation endeavor.

This tool will allow to design or modify animations, add or remove key-frames and print in the console the current status of the character's relevant bones or the full "data file" to replace in the "animationData" directory.

A function from the AnimationExec class will prepare the animationAction passing the correct clip and the main.js file will make the calls to start and stop those calls depending on the player agency.

The animation of the position was added in a later iteration since one of the animations required the character to crouch, so we expanded both the animation tool and the animation management scripts to accommodate this need. This seems to have introduced a bug that causes a weird crash in the execution when loading the scene in the animation tool, but we could not figure out if this was an issue related on our testing environment or a more general issue, neither we could asses which script is actually causing this malfunctioning since the error message is misleading and the error itself is really difficult to reproduce on purpose.

Figure 4.1: Animation Tool

Chapter 5

Music and sound effects

Music and sound effects are pre-loaded and stored in custom objects within a soundManager class instance when it is created.

Each sound is loaded in a THREE.Audio() object and associated to a non spatial listener, appended as a child to the scene camera object.

To allow easy tweaking of the sounds they are contained in custom objects containing also information about the audios, like how loud they should be played and if they are loopable music of "play-once" effects.

A background music is played when entered the game and paused when the Pause menu is showed. That background will be substituted by another music file when the player reach the game over state.

Also when interacting with the game via mouse input each class of object has it's own sound effect to be played as a feedback.

Chapter 6

Conclusions

This game is intended to be a show off of some of the THREE.js capabilities and its power in the game development field.

We purposefully didn't include external tools, like a physic engine or an animation engine, to show where possible the possibilities given by this single, powerful tool.

With greater understanding of it and with more experience in the field it can be a worth environment to experiment with prototyping and game design experiments due to the simplicity and freedom it may give.