Final Project Report

Course: Interactive Graphics
Professor: Marco Schaerf

Author: Elisa Foderaro, 1946278

Academic Year 2020-2021

Contents

1 Introduction **3**

2 Main Concept **3**

3 User interaction **3**
3.1 Main menu 4
3.2 Pop-up menu 5

4 Technical Details **5**
4.1 Environment 6
4.2 Scene 6
4.2.1 Bell Ringer 7
4.2.2 Walls and Floors 7
4.2.3 Ladder 8
4.2.4 Bells 8
4.3 Camera 9
4.4 Lights 9
4.5 Browsers 9

During the writing of this report, the same convention was adopted as in the textbook and "she" was used as generic, or gender-neutral, singular pronoun.

1 Introduction

This is a documentation report describing the techniques used in the Final Project of Interactive Graphics. The project is available here.

The first part of this report is a *user manual*, briefly describing the concept of the game and the user experience. Afterwards, all the technical details about the implementation are presented.

2 Main Concept

The final project was developed as an interactive 3D browser game.

Figure 1: A screenshoot of the game

It is split into two *phases*: in the first, the bells on the scene ring on their own and the user should carefully watch them because in the second phase, she has to play them. The aim of the game is to correctly remember the order in which the bells rang in the first phase and reproduce it in the second phase.

The sequence becomes longer at each level. The game is over when the user fails to remember the sequence for three times. An alert informs the user that she has lost the game, the page is reloaded and the main menu appears.

3 User interaction

The user can move the character of the bell ringer by using ***A-D keys*** or ***right-left arrows keys*** and can interact with the objects (the bells or

the ladder) by pressing the ***space bar***.

Pressing the space bar while the character is positioned in front of a bell, will lead to two different behaviors:

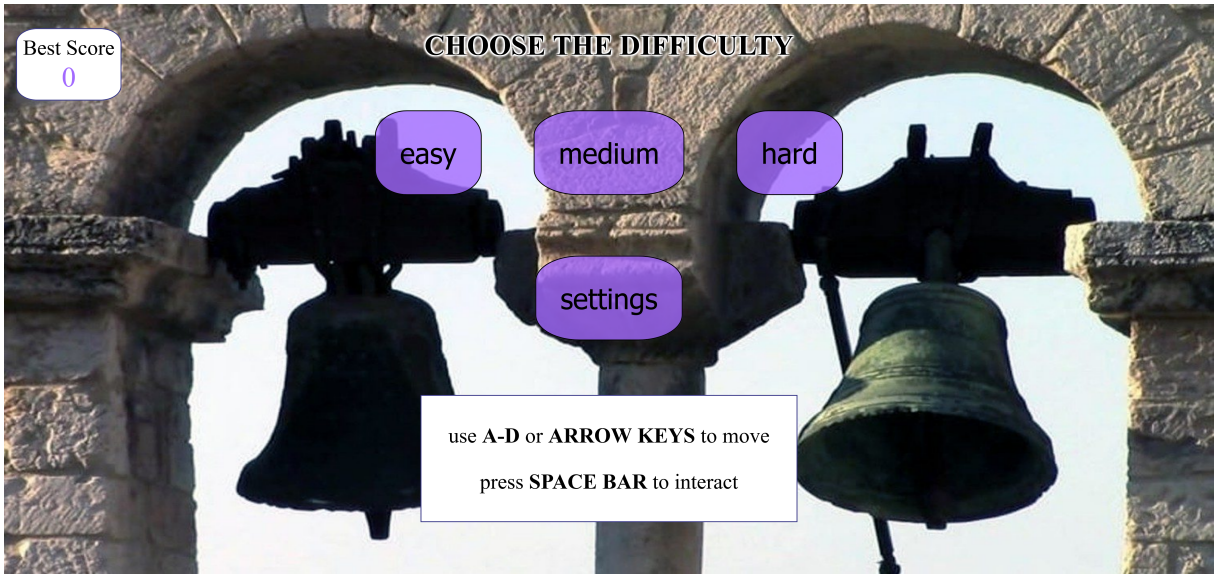
- if the bell is the right in the sequence: it will start the animation of the character *ringing* that bell.
- if the bell is **not** the right in the sequence: it will start the animation of the character *complaining* and the *number of lifes* will decrease by one.

If the bell was the last one in the sequence, an alert message asks the user to press `ok` to proceed to the next level.

Pressing the space bar while the character is positioned in front of the ladder, instead, will start the animation of the character changing floors by *going up* the ladder or *jump down*. During both animations, the user is not allowed to interact.

After pressing the ***esc key***, a dialog box appears to ask the user if she want to quit. If the user accepts, the page is reloaded and the main menu appears.

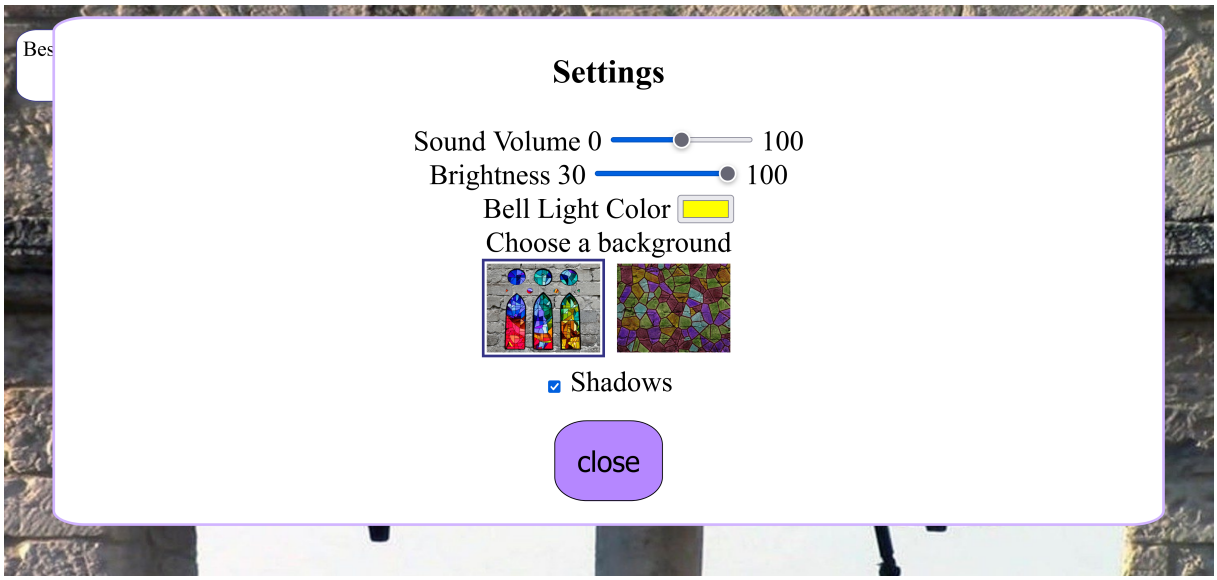
3.1 Main menu



Throught the main menu, the user can start a **new game** by choosing the *difficulty*, while seeing on the corner the *best score*, saved as a `localStorage` variable.

Indeed, the number of bells *increases* according to the difficulty chosen by the user; starting from 3 bells in the easy level, progressing to 4 in the medium level and reaching 6 in the hard level.

3.2 Pop-up menu



Throught the pop-up menu, accessible by push the `settings` button on the main menu, the user can set the *sound volume* and the *page brightness*, can choose the *background of the scene* and can decide if activate or not the rendering of the *shadows*.

In addition, the user can choose the *color of the light* inside the clapper, as explained in more detail below.

All those settings are saved as `sessionStorage` variables so that they will expire when the page session ends, *i.e.* when the browser is closed. It was chosen not to use cookies since this data only needs to be accessible on the client side. Using cookies, which are primarly for server-side reading, would have only resulted in a waste of bandwidth without benefits.

4 Technical Details

When the user clicks on the link, the page loads showing up a **progress bar**. In this phase, a `LoadingManager` has been used to load all *textures, models* and *sounds*. At the end of the loading phase, the *main menu* is showed.

4.1 Environment

The main environments used in developing this project are `HTML` and `JavaScript`, with a small use of `CSS`. In particular, some `JavaScript` libraries have been used, in addition to `WebGL`:

- `Three.js`, the world's most popular `JavaScript` framework for displaying 3D content on the web;
- `Tween.js`, for creating smooth animations.

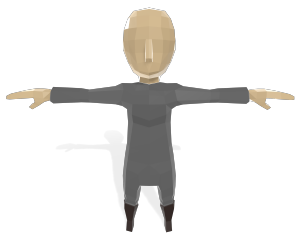
The project is organized through several ***modules***, according to `JavaScript ES6`. The file structure, disregarding the resources, is the following:

```
├── index.html            ...  the main page
├── index.css             ...  style for the main page
├── index.js              ...  main script to manage the game
├── popup.css             ...  style for the pop up menu
├── popup.js              ...  script to manage the pop up menu
│
└── modules/
    ├── loaders.mjs          ...  module to manage the loading of
    │                             textures, models and sounds
    ├── audio.mjs            ...  module to manage the audio
    │
    ├── models.mjs           ...  module to manage the adding of
    │                             models in the scene
    ├── animations.mjs       ...  module to manage the animations
    │
    ├── utils.mjs            ...  module which contains some useful
    │                             general functions
    ├── userInteraction.mjs  ...  module to manage the interaction with
    │                             the user
    └── gameManager.mjs      ...  module to manage the actual game logic
```

4.2 Scene

The following sections will describe individually each element that composes the scene. All but the bells are common to each difficulty.

4.2.1 Bell Ringer



It's the character that the user can play. It's a hierarchical model developed by quaternius at openGameArt.org.

The model is loaded with the other resources before the game starts and it is added directly as a *child of the scene*.

It is the model with the highest number of animations. It can **walk**, **climb the ladder** that connects the two floors, **jump off** from the first floor, **ring the bells** and has an animation for when the player **makes a mistake**.

All the animations are implemented in `JavaScript` by using `Tween.js`. The movements of the parts of the body (bones) are achieved by searching in the hierarchical structure the corresponding object and then creating a *tween* to move it smoothly. For more complex movements, a *chain* of tweens has been implemented.

Ultimately, all the animations are achieved by several chains of tweens, with each chain acting on a different part of the body.

4.2.2 Walls and Floors

For achieving the **ground floor**, a simple *Plane Geometry* mesh of dimensions 40x20, embedded into a 3D object named *GroundFloor* has been created.

Similarly, a *Box Geometry* mesh of dimensions 40x14x2, embedded into a 3D object named *FirstFloor* represents the **first floor**.

The material of both meshes is a *MeshPhongMaterial*, mapped with a wood-like texture loaded from an image at the same time as the other resources before the game starts.

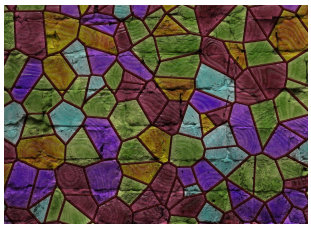
Both the 3D objects are then added as children of the scene.

For achieving the **lateral walls**, two *Plane Geometry* meshes of dimensions 20x30 has been created. For efficiency purpose, the geometry and the material are the same for both meshes.

The material is a *MeshPhongMaterial*, mapped with both a *color* map and a *normal* one to give the appearance of an old gray brick wall. The *normal map texture* has been generated with the same online tool already used in the second homework.

Then, both the meshes are added to the same 3D object named *Walls*, added as a child of the scene.

For the **background**, a *Plane Geometry* mesh of dimensions 40x30 has been created.



The material of the mesh is a *MeshPhongMaterial*, mapped with a texture that depends on the user's choice. The two possible backgrounds are shown beside and were created by overlaying some images on the wall texture.

Finally, a *Box Geometry* mesh of dimensions 40x1.5x1 has been added to the scene as child of the *FirstFloor* object to simulate the **beam** from which the bells hang on the first floor.

The material of the mesh is the same as that of the floor, *i.e.* a *MeshPhongMaterial*, mapped with a wood-like texture.

4.2.3 Ladder

The ladder that connects the two floors is a hierarchical model found in free3D.com. All the children in the object are characterized by a brown *MeshPhongMaterial*. Finally, before adding it to the scene, it was embedded into a 3D object named *Ladder*.

4.2.4 Bells

In addition to these objects, for each difficulty, a different number of *bells* is added as a child of the *Ground Floor* or of the *First Floor*, according to the desired position of the bell.

Each bell is a **clone** of a hierarchical model found in free3D.com. This because, by using the function `clone()`, the object can be loaded only once, together with the other resources.

Before being added to the corresponding floor, the bell is embedded into a 3D object called *Bell*, which contains as a child also the **rope** associated to the bell.

The rope is a simple *CylinderGeometry* mesh with a *MeshPhongMaterial* mapped with a rope-like texture loaded from an image.

The bells are the other animated objects on the scene. It has only the *song* animation, achieved with two chains of tweens, one involving the main bell and the other involving the clapper inside the bell. To provide a more realistic appearance, a *quadratic-in* easing function has been used.

When the animation starts, the *light* inside the clapper *turn on*, as better explained later, and the *sound* of a bell ringing *starts*.

4.3 Camera

A *perspective camera* has been used. It is initialized within `utils.mjs` with parameters:

```
fov = 45   aspect = 2   near = 0.1   far = 100
```

It was initialized in `position=(0, 5, 32)` but, during the animation of the character complaining, it **moves** to frame the character better, and then come back to the initial position. This movement has been achieved in the `animation.mjs` module through a tween involving the position of the camera.

4.4 Lights

Also the lights are initialized in `utils.mjs`. The main illumination is provided by a white `Ambient Light` and a white `Directional Light` positioned on top of the scene. The `Directional Light` can cast the shadows, depending on the choice of the user.

In addition, each time a bell is added to the scene, a `Point Light` is added *inside the clapper* of the bell. The *color* of the `Point Light` is the same chosen by the user throught the pop-up menu, `yellow` by default. To give the more realistic appearance of the light being emissed by the clapper, the `emissive color` of the material of the clapper has been set to the *same color* of the light. Both the `emissive intensity` of the material and the `intensity` of the light has been initialized to `0`. When the bell rings, they are switched, respectively, to `0.8` and `1` during the animation, and then go back to `0` at the end of it.

4.5 Browsers

The game has been tested on *Chrome*, *Firefox* and *Edge* browser and the performance are comparable.