Contents

1 Game Manual and General Description

1.1 Gameplay walkthrough recording

If you encounter some problems going forward into the game because the game is too hard or too heavy for your computer even with the lowest graphic settings, you can watch a complete walkthrough on it at this link(Google Drive shared .mov video). In this gameplay the player never dies and goes from the beginning to the end of the game with the final version of it. Everything that is not visible from this gameplay (e.g. what happens if the player falls down of if he's been hit by an enemy) can be discovered in game or reading this report.

1.2 Main menu and settings

In the main menu you can see three main buttons: start, settings and credits. This last button allow you to see information about the author of the game and let you know where you can find info about the imported models (actually into this document). Settings instead allow you to tune some parameters about the game that will be sent to the game when clicking the "start" button. The shadow parameters allow you to choose between maximum, medium and low, and this changes the shadow quality in the game. The "clouds" setting let you disable the moving windows in the game, while the "effects" and "music" buttons obviously allow you to disable this important characteristics of the game. An additional feature of the main menu is that you can click the background targets to make them fall.

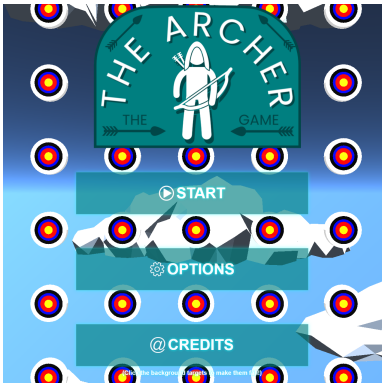

Figure 1: Home menu

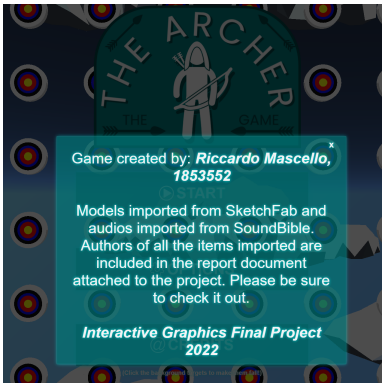

Figure 2: Credits

Figure 3: Settings menu

1.3 Commands

Move the archer using the WASD keys of the keyboard and jump using the spacebar. In order to move the camera visual you can move the mouse, and you can also zoom in or zoom out the camera using the mouse wheel, this will help you to have a better aim in some situations. You can also run using the shift key, while moving in any direction, to speed up a little bit the gameplay. Pressing the F key you will also shoot an arrow. When you shoot an arrow, the player will be rotated horizontally in the camera direction (maintaining although his relative direction with respect to the camera, more explanations during the technical discussion). This choices are the typical ones of a third person shooter game. When the player is in the nearbies of a terminal or an hologram projector (explained in details in one of the next chapters of the manual) he can interact with it using the E key.

1.4 Main components of the game

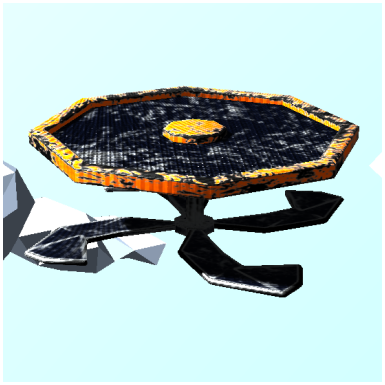

Figure 4: Floating Plaform

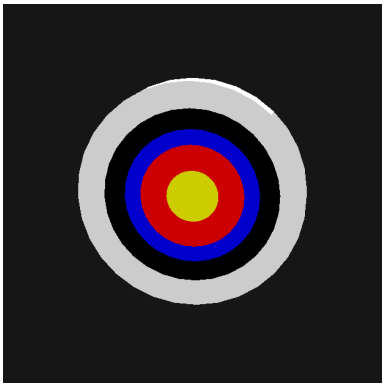

Figure 5: Target

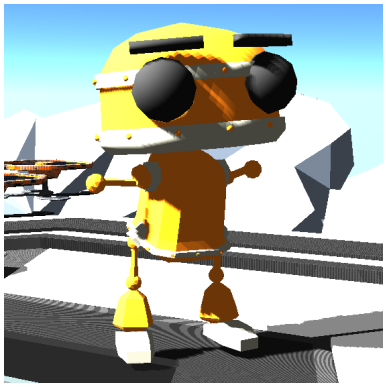

Figure 6: Robot Enemy

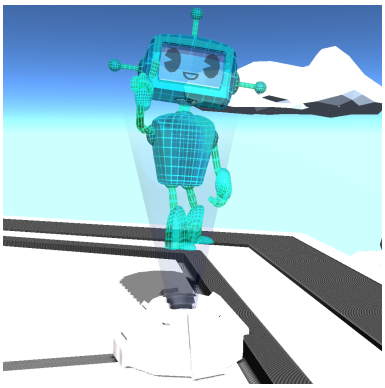

Figure 7: Robot Hologram

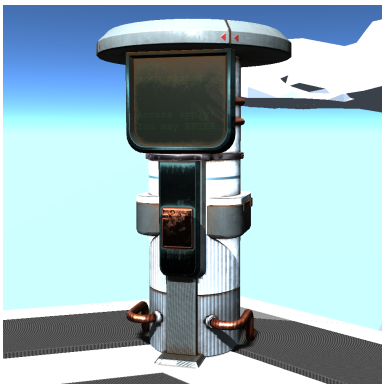

Figure 8: Terminal

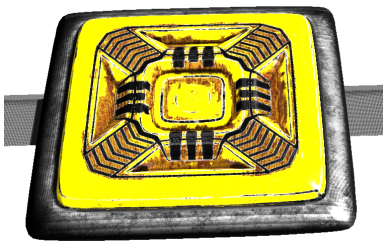

Figure 9: Pressure Plate

Floating platforms: The small floating platforms are very slippery and constantly moving up and down, this causes the player to easily fall from them. They are the main obstacle of the game (and also the hardest).

Targets: Try to hit these targets using your bow. Once you hit them you will unlock the next phase of the game, so they can be considered your main objective in this game. You should have a good aim because usually they're moving.

Robot enemies: Robot enemies are idle until you go sufficiently near to them. Once you activate them, they start running against you, and if they manage to touch you, you will be repulsed (usually falling down), they are not so fast but in some cases they work in group.

Holograms and Holograms projectors: Once the archer is sufficiently near to an hologram projector, he can interact with it, reading hints and information about what he has to do.

Terminals: Terminals are elements with which the archer can interact to trigger events, more details about each terminal obtainable from the hologram projectors in the game.

Pressure plates: Pressure plates activate positioning on them sufficiently heavy objects. You can simply stand up on them or put on it metal cubes that you find in the map.

1.5 Game sections

All these sections are characterized by robot holograms that will explain you what you have to do, to be short I will avoid to talk about them in each game section.

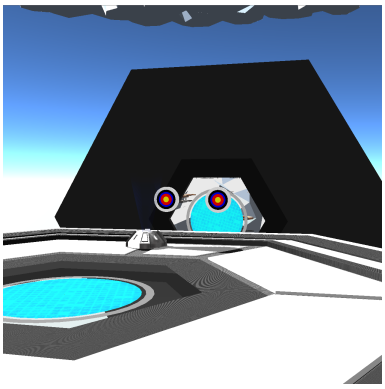

Figure 10: First section

Figure 11: Second section

Figure 12: Third section

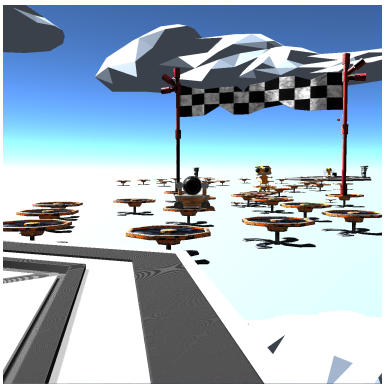

Figure 13: Fourth section

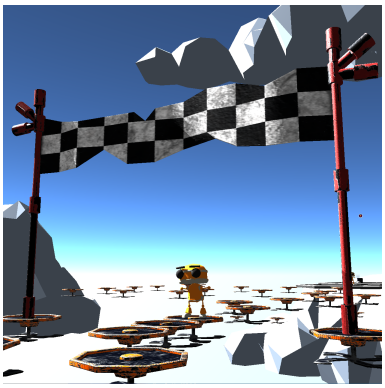

Figure 14: Fifth section

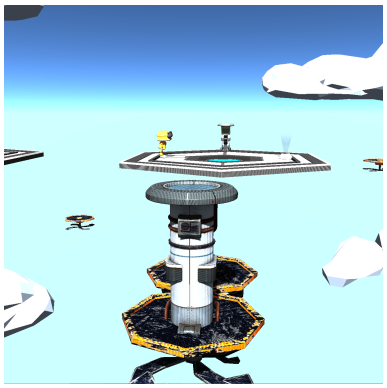

Figure 15: Sixth section

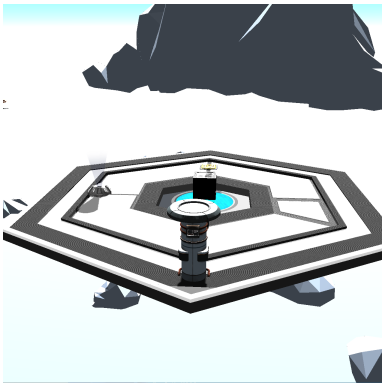

Figure 16: Seventh section

First section: The player starts on a platform where he has simply to shoot at two targets moving in front of him. Once he shot at them the platform in front of him will rotate to allow him to pass.

Second section: In this section the player encounters many floating platforms, only some of them are effectively reachable, while others will animate when both the targets moving in front of the player will be hit. More specifically six platforms will form stairs to go to the next section.

Third section: Here the archer will find three idle enemies that will be activated when going near them, the player will have to shoot them and make them fall in order to eliminate them.

Fourth section: Once the player shot out the enemies in the third section, he will face the threat of a shooting cannon in front of him. The cannon will shoot a cannon ball to the player, at every regular interval of time. The player will have to be fast in order to avoid to be thrown out of the platforms by the cannon balls.

Fifth section: The fifth is immediately after the fourth one, and it's characterized by a starting line and an enemy under it. Once the player will cross the starting line, the floating platforms under its feet will start falling at regular intervals. If the player will run and jump fast enough, he'll reach the next section.

Sixth section: Onto this main platform, the player will find a terminal, that, if activated, will activate some floating platforms for a certain amount of time. After that timeout, the platforms will fall and the terminal will be again available for use. So, once the terminal is activated, the main, objective is the one of running to activate the second one. This one will position a floating platform that will allow the archer to arrive to the next platform.

Seventh section: Once arrived to this platform, the player will see a pressure plate and a metal cube. The pressure plate can be enabled by the weight of the user or the weight of the metal cube. Since the metal cube could fall down there is also a terminal that allow the player to restore the initial position of this important object. The metal cube is useful to the player since the pressure plate will show up a target unreachable for the arrows of the player if he's standing directly on the pressure plate. For this reason he should move the metal cube directly on the pressure plate, enabling himself to shoot directly at the target. Once the target has been shot, the game is finally ended, the music will change and a text will be shown in the game.

2 Game Technical Description

2.1 Libraries

The entire game is based on the use of three main libraries, each one covering important fundamentals of the game environment, they are: ThreeJS, Cannon-ES and TweenJS.

ThreeJS: an interface library to create 3D graphics and visualize it. All the game is built on it.

Cannon-ES: a fork of the older CannonJS library. This last one is not maintained anymore, so the Cannon-ES is a better alternative. It is a physics engine that is useful to make the in-game environment interactions more realistic. The presence of gravity, contact materials and collisions make actions of the player work correctly (e.g. shoot arrows, jump and fall).

TweenJS: a javascript library that makes it available for the developer to use interpolation functions in keyframe animations. It also allows to repeat infinitely animations, create groups of them or simply chaining one to the other. All the animations of the game have been built using this fundamental library.

2.2 Models

All the models of this project have been imported from external sources with the GLTF format, using the GLTFLoader provided by default into the ThreeJS library. Here it follows a list of the models included into this project. Under each model you will find the link that brings you to the page from where the model has been downloaded. Following the links you will be able to see also the creator of each model. Some of these models contained already a hierarchical bone structure, that made easier the task of creating animations. In order to be able to manage all the bones of a structure a created a dedicated associative array to manage the rotations and movement of each component. On of the most important structure for example, was the one of the archer that contained many different bones, for each of which I created an entry in the associative array.

The functions that associate the bones of the hierarchical structure, to the entries of the associative array, have a name starting with the keyword *populate* (e.g. *populateArcherStructure*) and they perform this very basic but fundamental task. Once the structures have been populated the animations can be created using the library TweenJS, but the different behaviours of them will be discussed in the dedicated section "animations".

Figure 17: Archer LINK

Figure 18: Arrow LINK

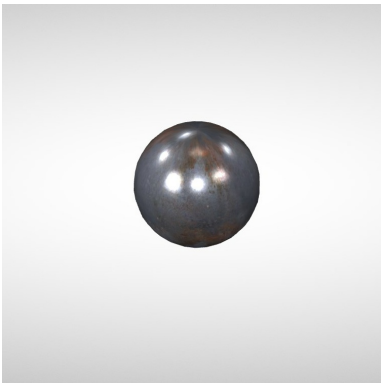

Figure 19: Cannon ball LINK

Figure 20: Cannon LINK

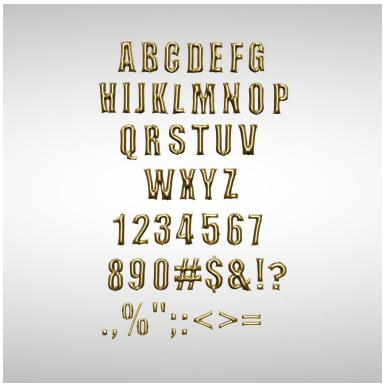

Figure 21: Letter balloons LINK

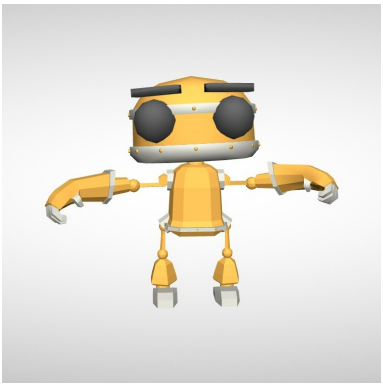

Figure 22: Robot Enemy LINK

Figure 23: Platform LINK

Figure 24: Hologram base LINK

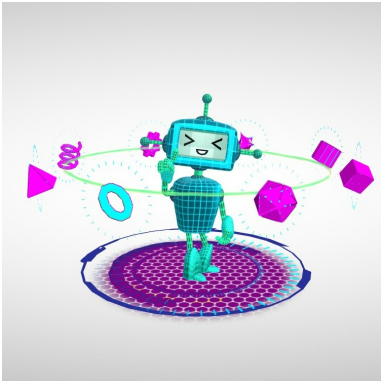

Figure 25: Hologram robot LINK

Figure 26: Metal cube LINK

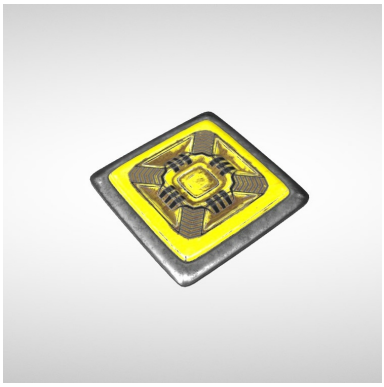

Figure 27: Pressure plate LINK

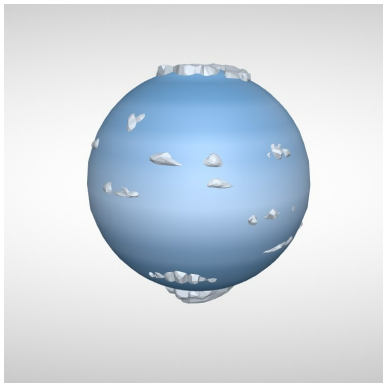

Figure 28: Skybox LINK

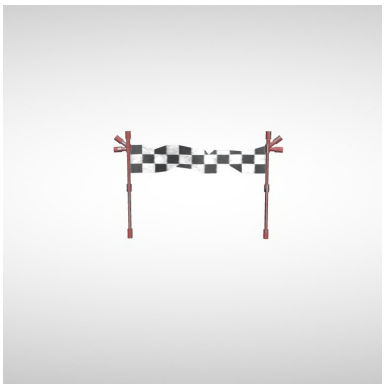

Figure 29: Starting line LINK

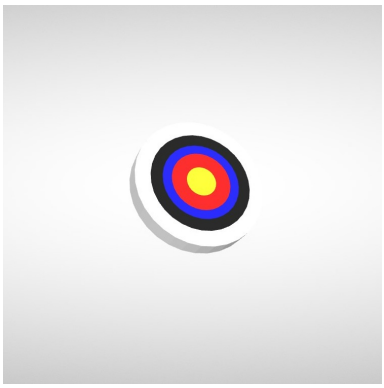

Figure 30: Target LINK

Figure 31: Terminal LINK

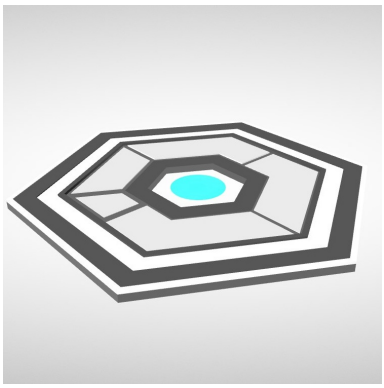

Figure 32: Main platform LINK

2.3 Textures

Each model imported contains dedicated textures, it means, like for the models and the sounds, I didn't create any texture, they were made by the models creator already mentioned with the links above. Inside each model there are many different types of textures, and they differ about what they are used for. I'll deepen how the terminal model is composed in terms of textures, since it is one of the most complete examples this project includes. It includes four different types of textures. Let's see for what each texture is useful considering the terminal example.

- The **base color** which represent the desired color of each pixel for the model at full luminosity, but the effective color will be obtained when applying the lighting model. The terminal contains two base color textures, one for the glass and one for the rest of the model

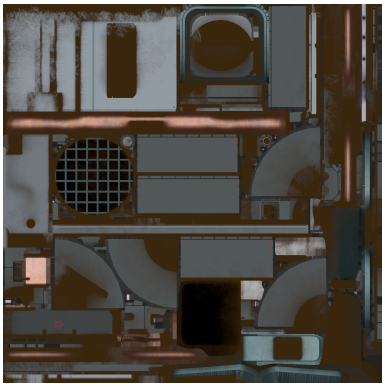

Figure 33: Terminal general base color texture

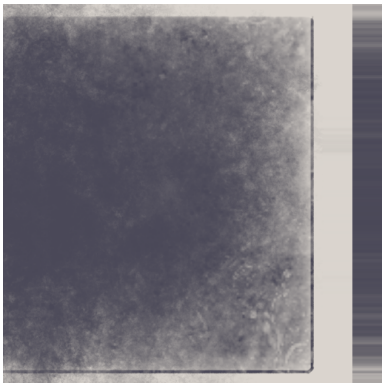

Figure 34: Terminal's glass base color texture

- The **normal map** is used to indicate the normals of each point artificially, to make it seems that you're facing an high poly surface, while instead it's only simulated by the lighting effect. So the normal map is describing the normal direction of each pixel of the surface, to create shadows or in general irregular surfaces.

Figure 35: Terminal normal map texture

- The **metallic roughness** texture indicates how sharply each point should reflect the light. The terminal contains two metallic roughness components, also in this case, one is for the glass and one is for the rest of the model.

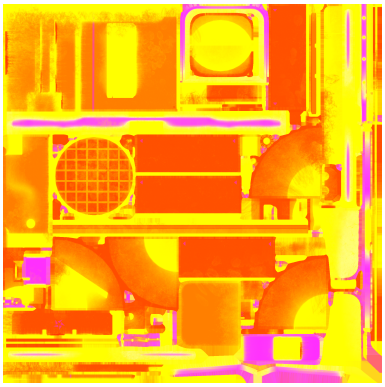

Figure 36: Terminal general metal roughness texture

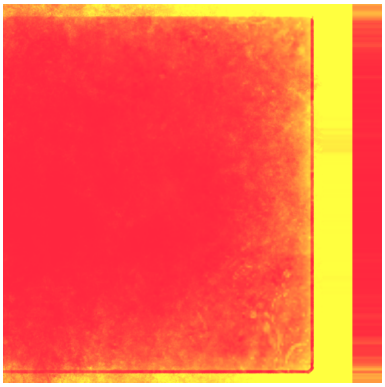

Figure 37: Terminal's glass metal roughness texture

- The **emissive** texture indicates that a model in particular points has to emit some light. The color and the intensity of the light are described into the texture.

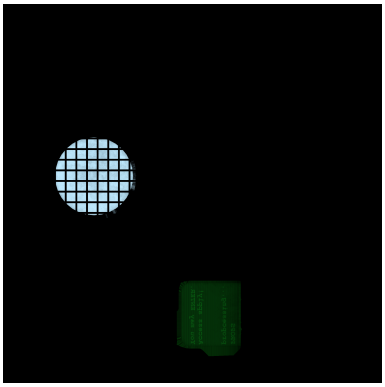

Figure 38: Terminal emissive texture

2.4 Wireframes

Each element in the game environment is characterized by three main components: the model (already mentioned in the previous section), the mesh and the body. These three components must be in the same position and oriented in the same direction in every moment, since they represent effectively the same entity. I decided to implement the mesh of the body as a simple wireframe that delimitates the physical body, to make it visible in some sense, in order to easily see if a body is colliding with another or to see if the hit-boxes are correctly implemented. There is an option (mainly for developers) in game to activate the wireframes of every body/model. Of the most important examples of wireframe that I had to design in order to make the shooting work is the mesh of the arrow. The mesh, so also the body, must be built according to the model dimensions and to the body dimensions. Since the wireframe is not visible by default, making the model not proportioned to the body and the mesh would mean having not expected behaviours for the user. For example in the case of the arrow, it would mean to hit an enemy even if the arrow model didn't effectively arrive to the enemy's body, or simply that the arrow model would penetrate the other models.

Figure 39: Arrow's body approximated by mesh

2.5 Bodies

2.6 Animations

2.7 Sounds

All the sounds available in this game have been imported from external sources, when mentioning each sound included into this project, it will be also provided the link from which the sound has been downloaded. There are two main categories of sounds in this game: the music audios and the effect ones. Both can be enabled or disabled through the settings inside the game menu. Technically speaking, two classes of sounds have been used, provided by the ThreeJS library: the first is the general Audio class, the second instead is a subclass of the previous and it is called PositionalAudio. The Audio class provide the basic functions to work with audios, but the PositionalAudio version is more sophisticated since it adds the possibility of embedding the audio inside a mesh. This means that when that sound is played it will come directionally from the mesh where it belong. As classic audios we have the two main themes of the game, one which is playing continuously (Main Theme Music Link) and the other playing only at the end of the game (End Theme Music Link). The only other non-positional sound is the menu click (LINK). For what concerns the positional audios instead we have all the other sounds:

- **the archer shot** (LINK), **the walk** (LINK), **the run** (LINK), and the **objective completed** (LINK) sounds emitted from the player's mesh,
- the **cannon shot** (LINK) emitted from the cannon's mesh,
- the **metal impact** sound (LINK) emitted from the enemy robot's mesh when this last one impacts the player or an arrow,
- the **terminal** sound (LINK) emitted from the nearest terminal's mesh when the player interacts with it,
- the **click** sound (LINK) emitted from the pressure plate's mesh when pressed or released,
- the **robot blip** sound (LINK) emitted from the hologram projectors when the player interacts with them.