Interactive Graphics 2021/2022 - Final Project

Marco De Luca - 2017104

September 25, 2022

Contents

1 Introduction **3**
1.1 Topic . . . 3
1.2 Project Requirements . . . 3
1.3 Hierarchical Models . . . 4
1.3.1 Human . . . 4
1.3.2 Cat . . . 4

2 Libraries Used **7**

3 The code files **7**
3.1 Main . . . 7
3.1.1 cat.js . . . 7
3.1.2 catAnimator.js . . . 7
3.1.3 environment.js . . . 7
3.1.4 human.js . . . 8
3.1.5 humanAnimator.js . . . 8
3.2 Utils . . . 8
3.2.1 Collision Detection: box.js . . . 8

4 Environment **9**
4.1 The TV Screen . . . 9
4.2 Lights . . . 9
4.3 Materials . . . 9
4.4 Textures . . . 10

5 Animations **10**
5.1 Cat Resting . . . 10
5.2 Human animations . . . 10
5.2.1 Human Walking . . . 10
5.2.2 Human Eating the Cheeseburger . . . 11
5.2.3 Human Sitting on the Couch . . . 12

6 User Interaction **12**
6.1 Walk . . . 12
6.2 Interact with the objects . . . 12
6.3 Change View . . . 13
6.4 Change colors . . . 13

7 Browser Compatibility **13**

8 Credits **13**

8.1 Human Model 13

8.2 Cat Model 13

8.3 TV Model with table 13

1 Introduction

1.1 Topic

The topic of the project is recreating a scene of ordinary life. In the scene we can find a person and his cat inside his living room, in addition to the furniture. The human can be moved by the user using the WAD keys or the keyboard directional arrows. In addition, when standing on one of the carpets, the human can interact with the nearby objects.

Figure 1: View of the Room.

1.2 Project Requirements

The project was realized only by myself. All the project requirements have been satisfied:

- **Libraries restrictions:** The project was realized using only ThreeJS and TweenJS;
- **Models:** Most models used were found on-line but no animation was imported.
- **Hierarchical Models:** There are two hierarchical models:
  1. **Human:** The human uses a hierarchical model found online, composed by 54 parts.
  2. **Cat:** The cat also uses a hierarchical model found online, composed by 36 parts.
- **Lights and Textures:** In the scene there is one ambient light and there is also a point light (the lamp). Regarding textures, the walls and the ceilings are made using a simple color texture, whereas the other objects in the room use more complicated textures composed by color, height, normal, roughness and ambient occlusion maps. Some of the textures were included with the models, while the others were created/edited by me.
- **User Interaction:** User can perform the following actions:
  1. Change the view using mouse;
  2. Move the human by using the WAD keys or the keyboard directional arrows;
  3. Interact with the nearby objects by using F and G keys.
  4. Change the human's body parts color by using the GUI;

5. Change the lights' color and/or intensity by using the GUI;

- **Animations:** As required, no animation was imported and they are instead implemented in JavaScript by exploiting the hierarchical models' structure.

1.3 Hierarchical Models

1.3.1 Human

The hierarchical model of the human is taken from sketchfab.com and is called "Rigged T-Pose Human Male". 54 parts were used for the animation of the model, although its rigging contains a few more which were not used in the animations (for instance the eyes' bones). Hands are made up by all the human fingers (thumb, index, middle, ring and pinky), each of them composed by three parts (the thumb has a third one to control the palm portion of the hand, the finger itself is composed by two parts). Feets are instead only composed by two parts each (the foot itself and a "toes" part), given that the human is wearing shoes.

Figure 2: The human model used.

1.3.2 Cat

The hierarchical model of the cat is taken from rigmodels.com. I personally rigged the skeleton of the model using Blender's Rigify addon. This allowed me to start from a fully completed skeleton, which I then resized and edited properly. Most of the head and feet bones in the resulting skeleton

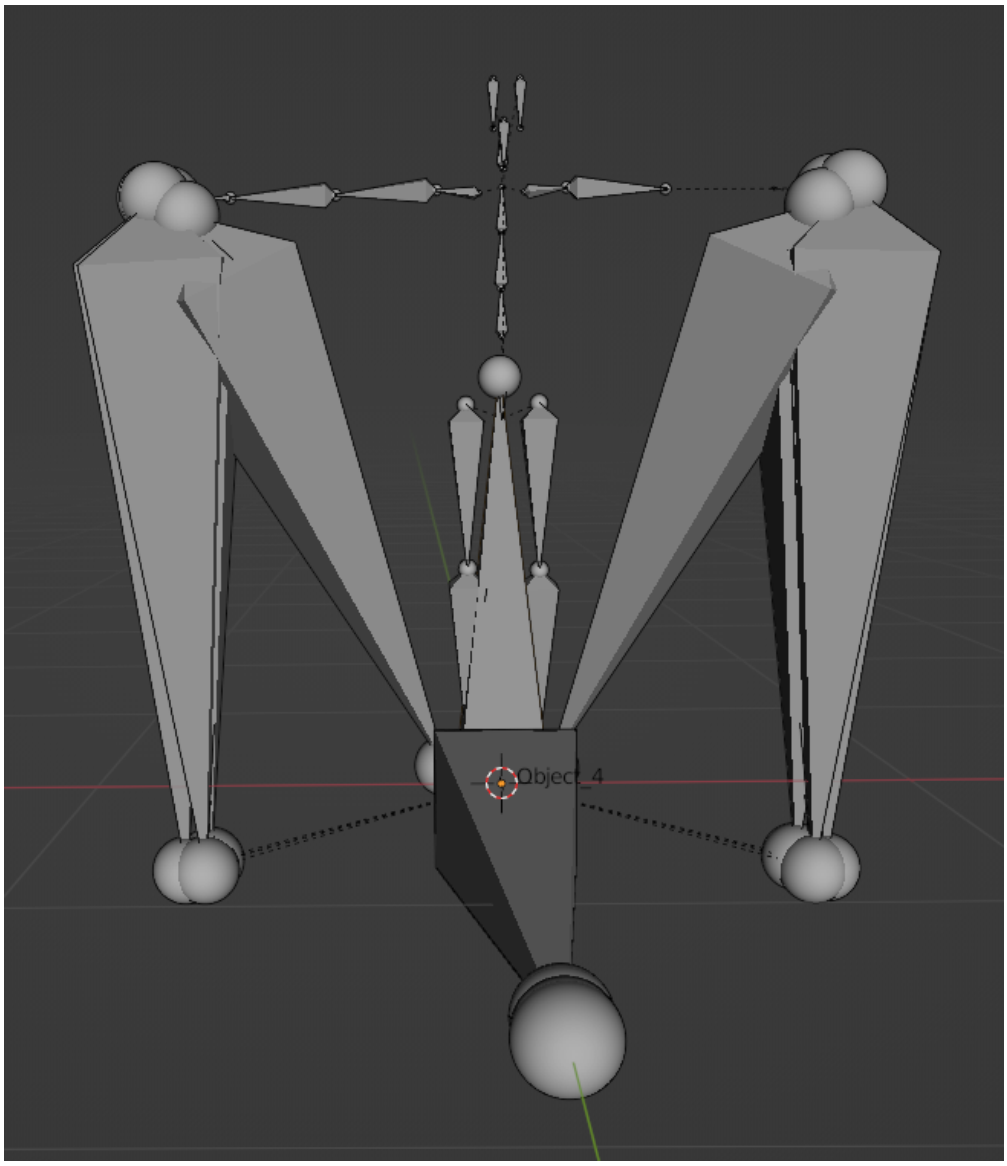

Figure 3: Skeleton of the human model.

weren't actually used in Javascript, although I still needed them to ensure a correct movement of the meshes. In total, the number of parts I used in JavaScript to animate the cat was 36.

Figure 4: The cat model used.

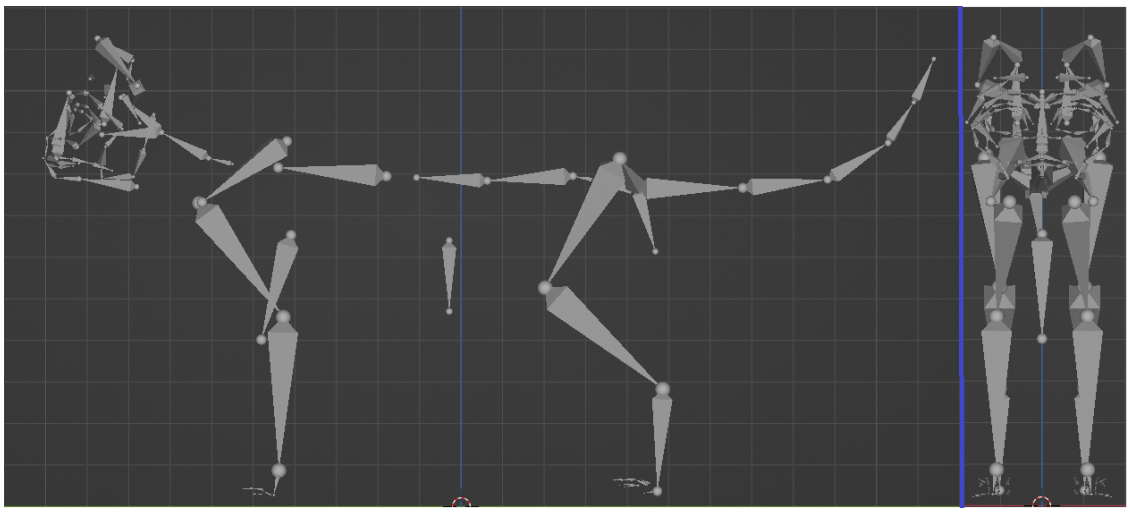

Figure 5: Skeleton of the cat model.

2 Libraries Used

The project has been implemented using the following libraries:

- **Three.js:** This is the main library, of which I used the 2022 Version;
- **Tween.js:** Library used to implement smooth animations;
- **Dat.gui.module.js:** Library used to implement the GUI (in the top right corner of the screen) to change the body parts' color and the lights' color and intensity;
- **OrbitControls.js:** Library used to implement the camera movement by using the mouse;
- **Stats.module.js:** Library used to show the current FPS/Latency/RAM Usage in the top left corner of the screen;
- **GLTFLoader.js:** Library used to load the Room, Cat and Human models which have been exported from blender in the .gltf format.
- **ColorGUIHelper.js:** This is a class supporting the GUI in the operation of assigning the color choosen by the user.
- **Reflector.js:** This class is used to setup the TV Screen so that it reflects the images.

3 The code files

3.1 Main

The main.js file coordinates all the others. It loads the 3d models, setups the renderer, the camera, the controls and the GUI.

3.1.1 cat.js

This library allows to manage the cat model: it loads the 3d model from the .gltf file (using the GLTFLoader.js library), then defines and initializes a variable for each of the 36 parts of the skeleton which will be used to animate the cat.

3.1.2 catAnimator.js

This library exploits the parts defined and initialized by cat.js to animate the cat. In order to move each part of the cat, I defined a configuration, which basically represents a Keyframe of the animation. The configuration is used to assign values to the quaternions of each part of the cat. The movement between the keyframes is then executed by using a set of tweens (i.e using the Tween.js library).
In the version of the project I'm presenting, the tween.js library and the configuration implemented are only used to change the cat from the "Standing pose" (The one shown in Fig. 4) to a "Resting pose" and then there is no further animation. I had initially planned to add a loop animation of the cat going back and forth between its resting pose and a stretching pose, but this was ultimately left unfinished because of time constraints.

3.1.3 environment.js

This class loads all the 3d models related to the room from the .gltf file (using the GLTFLoader.js library). It initializes every model by assigning them the proper textures and material properties. It also initializes the light. Lastly, it defines the walkable area for the human's movement and the areas where it is possible to interact with the objects. These operations are done by exploiting the **box.js** library I created.

3.1.4 human.js

This library allows to manage the human model: it loads the 3d model from the .gltf file (using the GLTFLoader.js library), then defines and initializes a variable for each of the 54 parts of the skeleton which will be used to animate the human. It also defines and initializes a variable for each of the meshes of the human. The user can use the GUI to change their color. Lastly, it normalizes the rotation angles of each part by making sure it is assigned a value between 0 and 2π.

3.1.5 humanAnimator.js

This library exploits the parts defined and initialized by human.js to animate the human. In order to move each part of the human, I defined a configuration, which basically represents a Keyframe of the animation. Regarding the configurations, the walk animation is defined by using quaternions, whereas the two animations related to the interactions are defined by using the XYZ rotation axis. I did this just because I wanted to understand if there was any difference in working with quaternions or the XYZ axis. The movement between the keyframes is then executed by using a set of tweens (i.e using the Tween.js library).
This library executes the following functions:

- It initializes the human to its idle pose and defines the tweens needed to perform the walk animation, the "Eat Burger Animation" and the "Sit on the couch" animation.
- It handles every aspect related to the walk animation: the variations in the position of the human, its orientation around the Y axis (i.e which direction the human is facing), starting, progressing and stopping the walk animation.
- It checks whether the human can actually walk forward or if there is an obstacle and therefore it needs to stop. It also checks if the human is currently in a spot where an interaction can be performed. Both these functions are supported by the box.js library I implemented, as better explained in 3.2.1.
- It allows the user to interact with the cheeseburger on the table and the couch. These animations are better explained in 5.1. It also allows the user to return the human to a standing position in order to be able to walk again.

3.2 Utils

3.2.1 Collision Detection: box.js

To prevent the human model from ending inside other objects while walking, I needed to define a walkable space. To achieve this, the space inside the room (on the XZ plane) where the human can't walk because of an obstacle has been delimited inside of a Box. When the user presses the W or the ↑ key, it tries to walk forward. The humanAnimator checks whether the human is about to walk inside one of the boxes I defined in environment.js, and if so, the animation is stopped. While being immediately outside one of the boxes the user can try to walk forward and the walk animation is in fact executed, but the human model does not move on the XZ plane.
Every box is defined by two variables, the top left corner (X;Z) coordinates and the bottom right corner (X;Z) coordinates: these are then used to define the rectangle composing the box. The box.js library also contains a method to check whether a point is inside or outside the box.
The same approach was used to determine if the human is positioned inside an area where an interaction can be performed. Lastly, it was used to check if the human is going to walk outside of the room or into the walls.

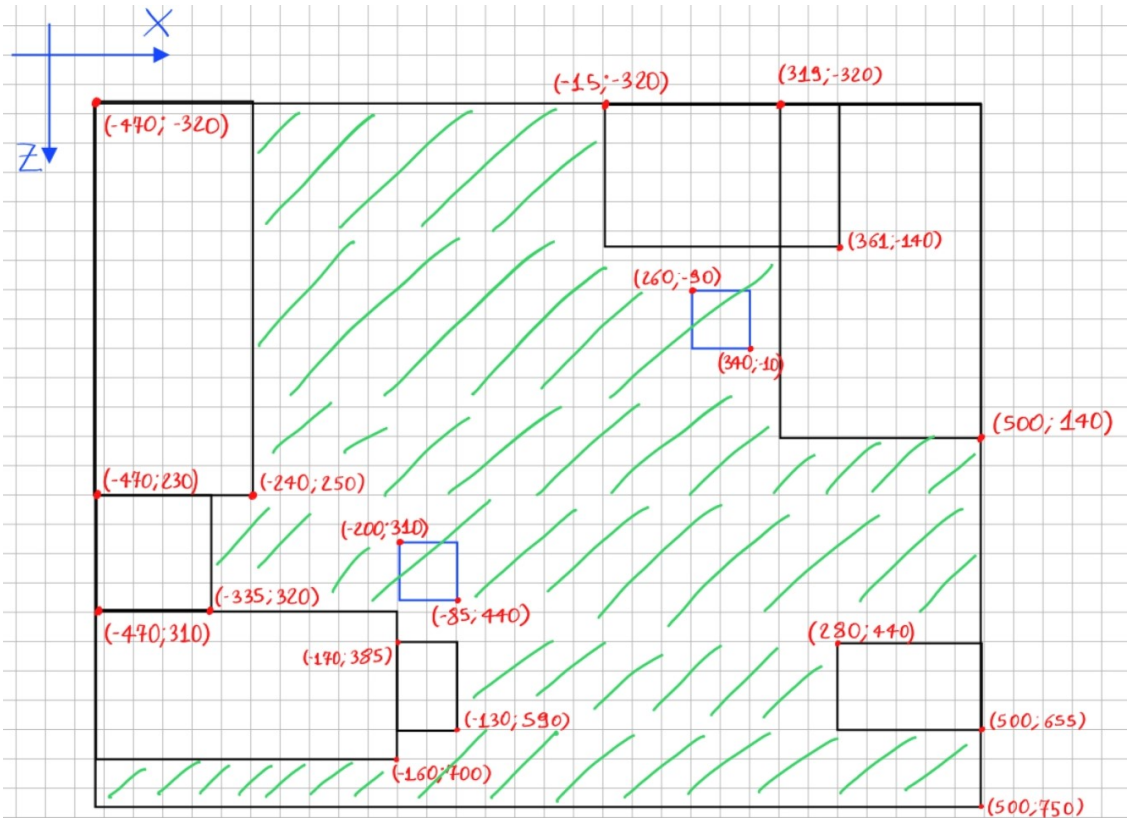

Figure 6: The boxes defining the obstacles in the room. The outer box represents the boundaries. The blue boxes are the areas where an interaction is possible. The user can walk in the green area

4 Environment

The walls, ceiling and floor 3d models were created by me using Blender. The burger, chairs, couches, Lamp, table, TV and the TV table are instead models I found online. I then put all these models together in a single 3d (.gltf) file by using Blender.

4.1 The TV Screen

To realize the TV Screen I added a vertical plane created by using the Reflector class (contained in /utils/reflector.js). This class comes from one of the examples of Three.js, and it was used to make it so the TV screen (which is turned off) reflects the image in front of it. The reflection color is darker, as to make it similar to a TV reflecting the image when it is turned off (therefore a black mirror), although the effect works better with a lower ambient light intensity.

4.2 Lights

Inside the room there are two lights: an Ambient Light and a Point Light. The latter is used to make it so the lamp produces light, therefore the lamp head in Three.js is made of a MeshStandardMaterial with an Emissive color. The color and intensity of both lights can be changed by the user thanks to the GUI. Changing the color/intensity of the pointlight also modifies the Emissive color/intensity of the material composing the lamp head.

4.3 Materials

The walls and ceiling are made with MeshPhongMaterial with only diffuse and specular components. The floor and the other objects instead use MeshStandardMaterial, which supports Physically Based

Rendering, giving a more realistic effect.

4.4 Textures

The objects realized using MeshStandardMaterial (except for the cheeseburger) are all implemented using a texture which contains the maps for color, normal, displacement, roughness and ambient occlusion. These are loaded using Three.js's textureLoader, and then repeated along the directions S and T 10 times for the floor and table textures, while only 3 times for the couch, pillow and metal textures.

5 Animations

5.1 Cat Resting

The only animation performed by the cat sets it in its rest pose. This happens during the initialization and is therefore not seen on screen. It is still implemented with tween.js but it is actually set to be completed instantly. As previously said, I had initially planned to add a loop animation of the cat going back and forth between its resting pose and a stretching pose, and the code of catAnimator was being set up to do this. This was ultimately left unfinished because of time constraints. Therefore, the only animation instantly moves the cat from the standing pose to its rest pose.

5.2 Human animations

The human has three animations, one for walking, one to interact with the cheeseburger and lastly one to sit on the couch. All these animations are handled by the HumanAnimator class, and everything is implemented by using the base Tween.Js only.

5.2.1 Human Walking

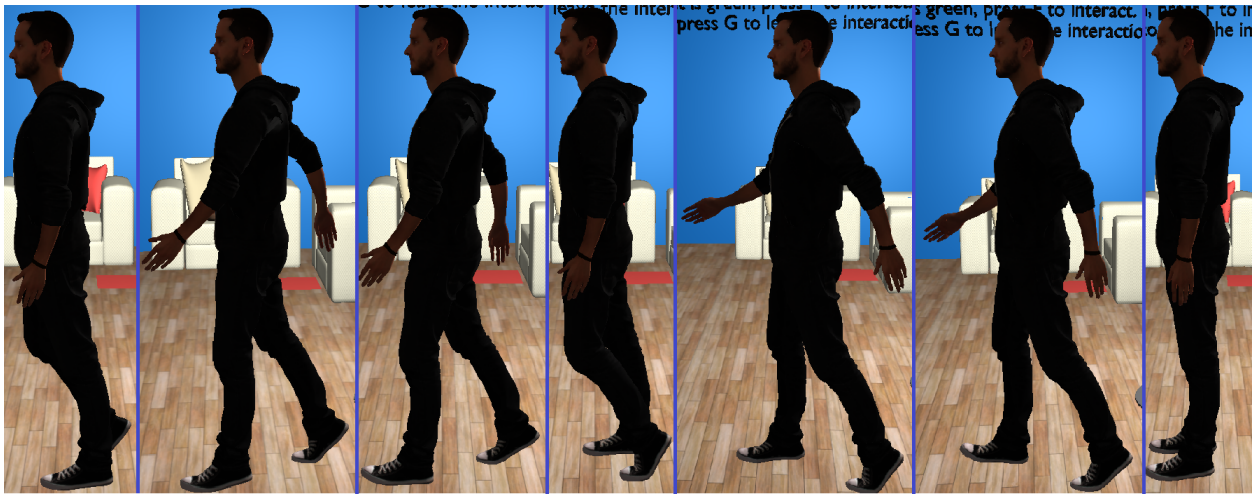

Figure 7: The walk 7 Keyframes. From right to left: Idle, StartRight, EndRight, ChangeLegRtoL, StartLeft, EndLeft, ChangeLegLtoR.

The walk animation is composed by 7 keyframes chained together: *Idle, StartRight, EndRight, ChangeLegRtoL, StartLeft, EndLeft* and *ChangeLegLtoR.* While the human is not walking, it is in the Idle pose. Once the 7 keyframes are completed, if the human needs to stop (either because the user is no longer pressing W or ↑, or because the human walked into an obstacle), then it returns to the idle pose. Otherwise (the human is still walking), the idlePose is skipped and a new cycle begins starting from the StartRight pose. While walking, the humanAnimator moves the human

by using a tween.
If the user releases the walk button while the human is walking, the current tween is immediately stopped and the human returns to the idle position.
If the user walks into an obstacle, the current walk animation is stopped and the human returns to the idle position. However, if the user keeps pressing the walk button (or if he releases it and then presses it again without changing direction) the walk animation restarts, although the human remains in the same position. There is no animation when changing the walk direction (the model simply rotates).

5.2.2 Human Eating the Cheeseburger

Figure 8: The 9 Keyframes. From left to right, top to bottom: Init, MoveChair, Idle, StepInit, StepStart, StepMiddle, StepEnd, Sit, MoveBurger.

This animation is performed when the user presses **F** while being on the carpet near the table. This animation is completely executed by the humanAnimator. It is composed by 9 keyframes: *Init, MoveChair, Idle, StepInit, StepStart, StepMiddle, StepEnd, Sit* and *MoveBurger*. In this animation, the human moves the chair, sits on it, grabs the cheeseburger and puts it in front of his face. Once the animation is completed, the humanAnimator loops continuously between two keyframes which are almost identical, except for a slight difference on the head rotation. This gives the impression that the human is eating the cheeseburger.
While the human is eating, if the humanAnimator detects that the user has pressed **G** (at any point after the interaction had started), it will perform the same animation as before but reversed. This will result in the human placing the cheeseburger back on the plate, standing up, putting the chair back under the table and then returning to the idle position.

5.2.3 Human Sitting on the Couch

Figure 9: The 3 Keyframes. From left to right: Init, Middle, End

This animation is performed when the user presses **F** while being on the carpet near the couch. This animation is completely executed by the humanAnimator. In this animation, composed by 3 keyframes, the human starts from an Idle position. Then he stretches back his right arm towards the armwrist, while sliding his left hand on his left leg. Once he put his hand on the armrest, he uses his arm to support himself while sitting, and in doing so he slides his arm on the armrest. At the end, he is sitted on the chair with an arm on the armrest and the left hand on his leg.
While the animation is performed, if the humanAnimator detects that the user has pressed **G** (at any point after the interaction had started), it will perform the same animation as before but reversed. This results in the human standing up from the couch and returning to the idle position.

6 User Interaction

6.1 Walk

Pressing **W** or ↑ moves the human forward (with respect to the direction he's currently facing). Pressing **D** or → turns the human right, while pressing **A** or ← turns him left. If the user makes the human walk into an obstacle (or out of the boundaries), he automatically stops and return to the idle pose. If the user keeps walking/resumes walking towards the obstacle, the walking animation will be performed but the human will not move.

6.2 Interact with the objects

There are two red carpets in the room. Getting close enough to one of them changes its color to green. When a carpet is green the user can press **F** to interact with the nearby object (namely the couch or the cheeseburger, depending on which carpet the human is on). This will cause a short animation of the human interacting with the object. Once the animation is completed, the user can press G to exit the interaction. While the human is interacting with one of the objects, it can't walk, and it won't be able to walk again until the user presses G and the animation completes (i.e it won't walk again until it returns to the idle pose).

6.3 Change View

The user can rotate the camera around by pressing the left mouse button. It is also possible to move the camera by pressing the right mouse button. Lastly, the mouse wheel allows to zoom in and out.

6.4 Change colors

There is a GUI in the top right corner of the screen. This allows the user to change:

- **Human Parts Colors:** namely the hair, skin, bracelet, jacket, shirt, pants and shoes color.
- **Lights:** It is possible to change the color and intensity of both the ambient and point light. Since the point light coincides with the lamp head, changing the color or the intensity of the point light will also change (respectively) the "Emissive" or the "EmissiveIntensity" properties of the lamp head.

7 Browser Compatibility

I tested the project on Windows 10, both locally (Python3 server) and in GitHub Pages on **Google Chrome Version 105.0.5195.127 (Official Build) (64 bits)**.
Github pages Link:
`https://sapienzainteractivegraphicscourse.github.io/final-project-marcodeluca/`
Please notice that on Github pages the project might take a little longer to load.

8 Credits

8.1 Human Model

This work is based on "**Rigged T-Pose Human Male w 50 Face Blendshapes**" (Link) by Mike Alger (`https://sketchfab.com/mikealger`) licensed under CC-BY-4.0 (`http://creativecommons.org/licenses/by/4.0/`)

8.2 Cat Model

This work is based on "**Cat 3D Model**" (Link).

8.3 TV Model with table

This work is based on **Samsung Smart TV 55 Zoll with Table** (Link).