Final project

Lorenzo Capobianchi 1815464

Jurassic escape

Introduction:

Jurassic escape is a 3D game belonging to the category of the endless run game, in which the user controls a velociraptor that runs forward and meets obstacles that must be dodged. The user simply collects points keeping the dinosaur alive as long as possible, indeed the game ends as soon as the dinosaur hits an obstacle.
In this case the user can not avoid an obstacle moving on the left or the right side or changing direction because the only way provided to dodge is jumping the obstacle with the right timing.

Environment:

The project is based on the Three.js library and the webGL renderer associated to this library.
All the base elements of the projects are created using Three.js, in fact the camera, the scene and the lights have been taken from this library. Also the ground and the sky of the game were built using the Three.mesh class of this library.

Technologies and models:

- **Technologies**
  The technologies that i used to develop this final project are:
  1. HTML
  2. CSS
  3. Javascript
  4. Three.js
  5. GLTFLoader
  6. FBXLoade
  7. Tween.js

  In particular HTML was used to build the homepage, the game over screen and to visualize the score during the game. CSS obviously helped in formatting and positioning the html elements on the screen.
  Javascript is the main programming language of the application, in particular the Three.js library is the core module used for the development and GLTFLoader and FBXLoader are the libraries that allowed me to import the models in the project.
  Tween.js is a tweening engine to make smooth animations in an easy way, in this project i used it to create the “run” and “jump” animations.

- **Models**

  Almost all the elements of the game are GLT or FBX models, only the ground and the sky are the exceptions and were built using the Three.mesh class of Three.js.

  All those elements can be found in the "resources" folder (not all the models contained in this folder were used in this project) and in particular are:

  1. resources/DesertPack/FBX/Cactus3.fbx
  2. resources/Dinosaurs/FBX/Velociraptor.fbx
  3. resources/Clouds/GLTF/Cloud(1,2,3).glb
  4. resources/DesertPack/GLTF/SmallPalmTree.glb
  5. resources/DesertPack/GLTF/BigPalmTree.glb
  6. resources/DesertPack/GLTF/Skull.glb
  7. resources/DesertPack/GLTF/Scorpion.glb
  8. resources/DesertPack/GLTF/Pyramid.glb
  9. resources/DesertPack/GLTF/Monument.glb

  The first model is the cactus used as obstacle in the game. It is the element that must be avoided by the user and can be found in cluster of 1-3 elements of different sizes.

  The second model is the velociraptor controlled by the player that can perform jumps to dodge the cactus.

  The third element is the set of clouds that can be spawned during the initialization of the game and that are placed in the sky.

  All the other models are just used to populate the ground environment, the user can not interact with them and their presence has only aesthetic purposes.

Technical aspects:

- **index.html**

  The homepage is very simple, it is composed by a container with a background image and it contains the name of the game and informations about how to play and how to start the game.

  When the user clicks the left button of the mouse the homepage is hidden and the game starts.

  In this html file we can also find the game-over layout and the score layout. The first one is hidden in first place and his only shown after that the user has lost the game, while the second is always visible except when we are visualizing the homepage.

- **main.js**

  This file is the core of the whole application. Here i used Three.js to create the base elements of the application such as the camera, the scene and the lights, but also the ground and the sky environments. The ground is a very simple Three.js plane geometry

with a color texture (simply created using the THREE.MeshStandardMaterial and assigning a color to this material), while the sky is a sphere geometry with a color that goes from a shade of blue to another.
The other elements of the game such as clouds, ground elements, the velociraptor and the obstacles are models imported in the application and managed in classes defined in different files. Finally they are instantiated in this file to put them togheter and correcty handle the game logic.
When the application starts all the elements of the game are instantiated but they are hidden by the homepage. When the user clicks the left button the homepage is hidden and the game starts and it keeps going until the velociraptor hits one obstacle. At this point the run stops and the game-over layout is shown to the user.
To start again the game the user has to click again the left button of the mouse as specified in the game-over layout. When the game restarts the whole world is reinstantiated and the score is set to zero.

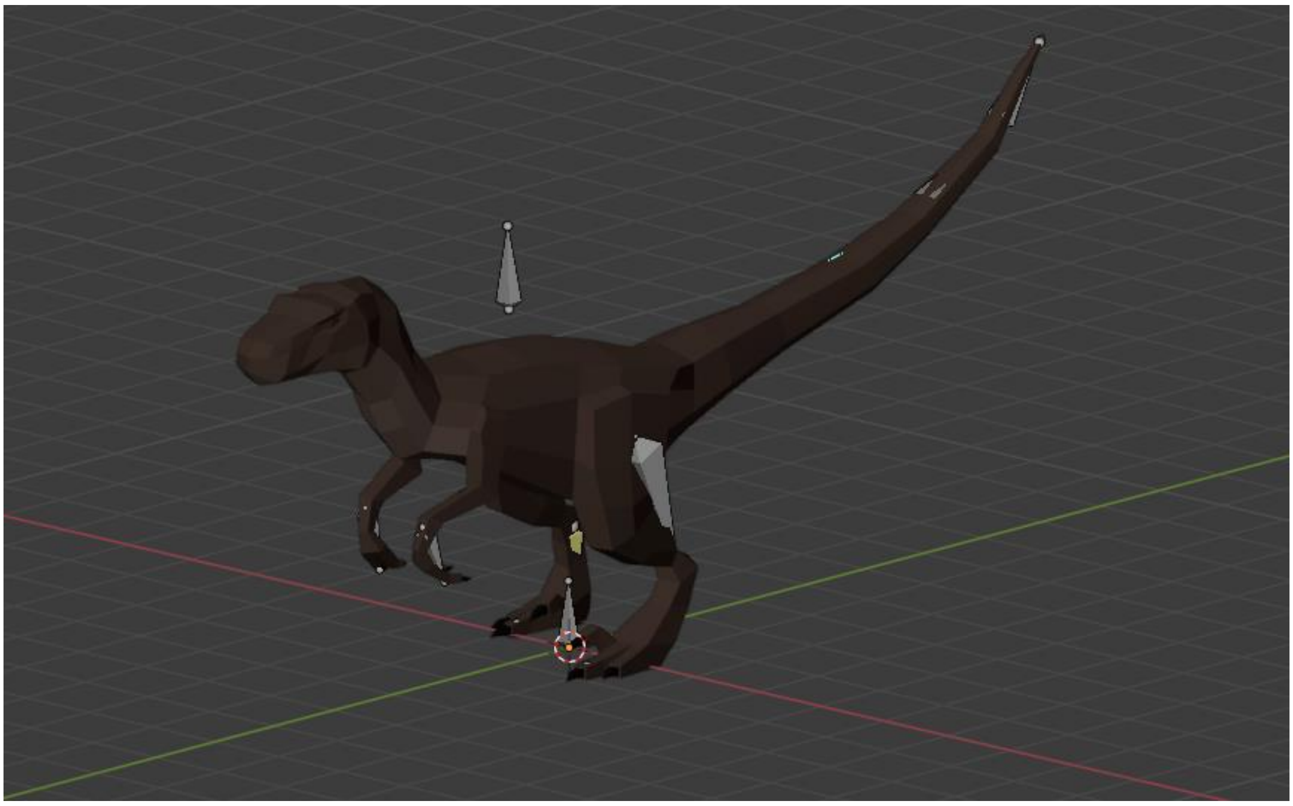
- **world.js**
  The obstacles are cactus generated in clusters of 1-3 elements in order to achieve different width of the obstacle to jump. Those cactus are models imported using the FBX loader and to which i attached an image texture. For each of them i created a box covering the area of the surface of the cactus used to evaluate the collision between the player and the obstacle that will lead to the end of the game.
  The obstacles are the elements performing the movement towards the player. In fact the game logic is inverted in the sense that we can see the animation of the velociraptor running into the obstacles even if it is clear that the items actually moving are the cactus instead of the velociraptor that is placed in one point and only moves on the y axis to perform the jump.

- **player.js**
  The obstacles are not the only elements based on a FBX model, in fact also the velociraptor is a FBX model, but in this case we only need to generate one dinosaur and as written before we do not even need to move it on the x and z axisses but only on the y axis to perform the jump.
  This element as the obstacles is associated to a box covering the area of its surface that is used to find the collisions that will determine the end of the game.
  This file contains two important functions: Update() and UpdateRun(), which are used to perform the jump and the animations of the player. I will discuss in a dedicated section the development of the animations.

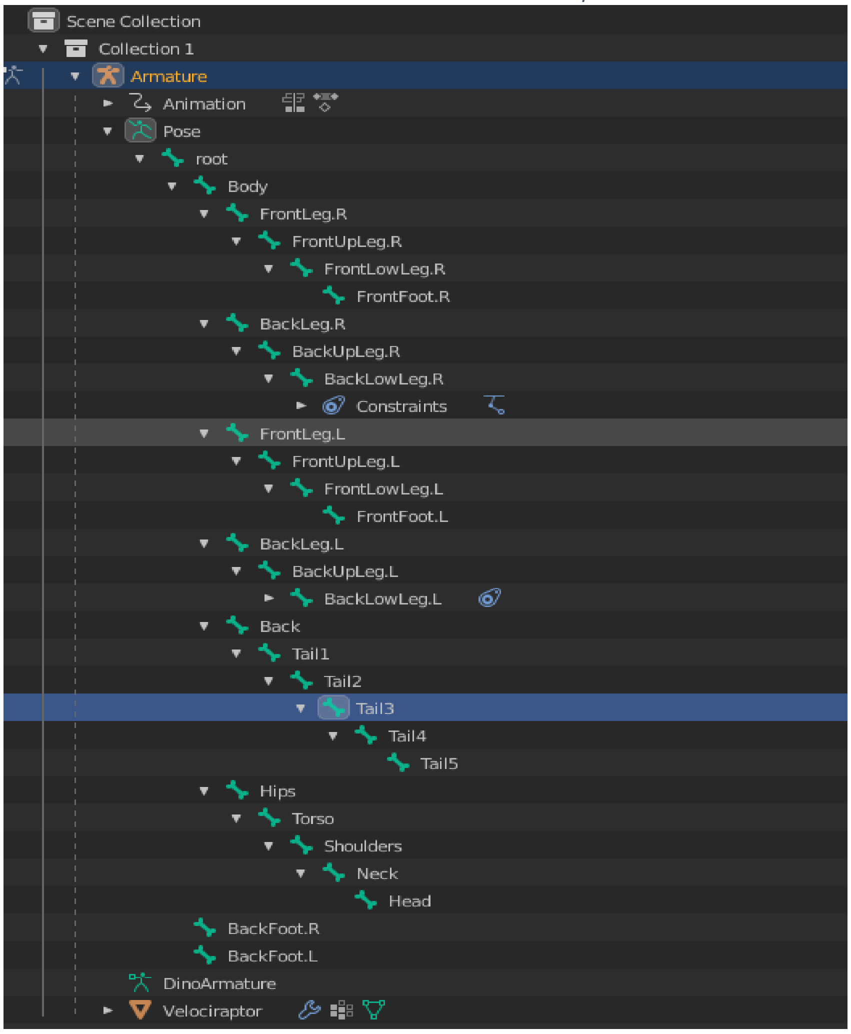
- **background.js**
  On the background we can see the clouds in the sky and several other objects such as pyramids and palms on the ground. All those elements are managed in this file and they basically are GLTF models imported in the project to which i applied an image texture . They are generated during the initialization of the application only for aesthetic purposes. They are not really useful, indeed the game would work even without them, but i used them to make the game more enjoyable, in fact without these items the background would be empty. Those elements like the cactus defined in world.js are moving toward the camera thanks to an "Update" function that modifies the x position of each element in every frame.
- **math.js**
  In this file we can find two auxiliary functions used to generate a random integer and a random number in a given range.

Animations:

All the objects contained in the project do not have animations, they just translate until they are not in the camera view anymore. The only exception is the velociraptor, which has two animations: one simulates the run of the character, while the other is associated to the jump . These are very simple animations defined in the player.js file.
The creation of the animations is a process that can be divided in several steps. The first thing that i did was move the character on the y axis after that the spacebar button is pressed, because in this way i could prevent the game from ending few seconds after the beginning, so that i could check the progresses on the other animations without reloading the page every time.
The jump animation was not complete because i wanted to add a rotation of the character during the ascending part of the jump and reverse the rotation during the descending phase, so that the velociraptor ends the jump touching the ground on his feet and without any inclinations.
At this point i used an application called "blender" to analize the structure of the velociraptor model that i downloaded, because to develop the animations i needed to know the organization of the internal nodes of the model and the name associated to each part.



1: Velociraptor on blender



```
Scene Collection
  Collection 1
    Armature
      Animation
      Pose
        root
          Body
            FrontLeg.R
              FrontUpLeg.R
                FrontLowLeg.R
                  FrontFoot.R
            BackLeg.R
              BackUpLeg.R
                BackLowLeg.R
                  Constraints
            FrontLeg.L
              FrontUpLeg.L
                FrontLowLeg.L
                  FrontFoot.L
            BackLeg.L
              BackUpLeg.L
                BackLowLeg.L
            Back
              Tail1
                Tail2
                  Tail3
                    Tail4
                      Tail5
            Hips
              Torso
                Shoulders
                  Neck
                    Head
          BackFoot.R
          BackFoot.L
      DinoArmature
      Velociraptor
```

These informations were useful since i had to move all the different parts of the leg to simulate a run and rotate the root element of the model during the jump. I had a further confirmation of the structure of the model using the fucntion “traverse()” and printing the name of every visited node. I defined a function called “UpdateRun()” in which using the methods defined in the tween.js library i managed to implement the animation of the run. In particular the result was achieved checking at each iteration the position of the right leg and moving it in the opposite direction if the movement in the previous direction was completed. The movement of the left leg was simply created reversing the movement done in the right leg at each iteration (if the right leg moves forward the left leg moves backward and viceversa).
Now the legs of the velociraptor simulate a run, but to make this animation more interesting and also a bit more realistic i decided to animate also the tail of the model, in fact during a real run of a velociraptor his tail swings left and right and so i decided to implement this movement in the animation.
The tail of the model is divided in 5 sections called “Tail1”, “Tail2”, “Tail3”, “Tail4”, “Tail5”. The first two sections are too close to the body and their movement is so small that i decided to ignore it at all.
All the other sections (Tail3,Tail4 and Tail5) swing from left to right and viceversa at each step performed by the velociraptor.
The animation of the run is now officially complet and i just needed to work a bit again on the jump animation. When the velociraptor performs the jump i change the y position of the model and rotate it to give the idea of the inclination of the jump, but i also stop the movement of his feet since they are not touching the ground. The run restarts as soon as the velociraptor touches the ground.

User interaction and manual:

This is a very simple game that proposes only one way of interacting: pressing the spacebar to jump with the right timing.
The homepage contains the information on how to start the game and also shows to the user the only command that he can use, in fact there is written that you need to click the left button of the mouse anywhere to start the game and just press the spacebar to play.
At the end of the game, when an obstacle has been hit by the player, the game-over layout makes clear to the user that he has to press again the left button of the mouse if he wants to play gain.