Michele Manglaviti, 1964287, Engineering in Computer Science, MINR

Interactive Graphics final project – deadline: 26/09/2021

GitHub team: mm_team

Participants: Michele Manglaviti, 1964287

Nate’s Adventure

1. Project Description

Nate is a treasure hunter who is looking for an ancient gem of inestimable value of which there are only legends.

After years of research, he comes to know that the gem was the hidden treasure of an ancient and now extinct population that used to live in the deep jungle and is still in the ruins of the temple where the population used to venerate it.

After a long and difficult journey, and with a new tiger friend as a pet, Nate is finally arrived close to the temple and the gem is very close, watch him making his last effort!!

1.1 How to Interact: Tutorial

The project is thought to be an animated scene in which you can interact with it using the mouse and the buttons console in the upper left corner.

Here’s the list of the commands and functions:

Hold LEFT button on mouse and move: Rotate the scene.

Hold RIGHT button on mouse and move: Move the scene in the chosen direction.

Scroll wheel: Zoom in/out the scene.

Start: start all the animations and the music.

TreasureAnimation: Stops treasure animation if clicked, restarts it if clicked again.

Reset: teleports Nate and restarts the animation from the beginning.

TigerSound: triggers the tiger verse when clicked.

Mute: mutes the music.

Unmute: unmute the music.

TreasureLightOFF: switch OFF the light positioned close to the treasure zone.

TreasureLightON: switch ON the light positioned close to the treasure zone.

SunLightOFF: switch OFF the sun light.

SunLightON: switch ON the sun light.

2. Project Architecture and Organization

The project has been developed using the Three.js environment for the building of the whole scene, included characters and textures, all has been developed using Three.

The animations have been developed using Tween.js library, used to animate Nate and the tiger, the treasure animation has been implemented using Three.js features.

All the libraries used in this project (Three.js, Tween.js) are in the “js” folder, in the repository you will find the “js” folder, the “resources” folder, css + html files and this document.

Textures, images and sounds are stored in the “resources” folder.

The project has been developed using Atom editor for html, css and JavaScript coding, using a python web server for testing the features on the localhost, and using the Three.js official site tutorials in order to comprehend the library.

The project has been developed and tested mainly using Google Chrome as web browser, it has been tested also on Opera, Mozilla Firefox and Microsoft Edge browsers.

I advise you that there could be a bug on the sound reproduction on Chrome after the eventual refresh of the web page (problem given by caching problems and also by the new policies of google on AudioContext).

The project has been deployed on GitHub pages: https://sapienzainteractivegraphicscourse.github.io/final-project-mm_team/

2.1 Technical Aspects

After having found the idea for the project, the first steps where to define the scene, the camera (PerspectiveCamera) and the renderer.

Once the scene has been set up, I started gaining confidence with the Three.js environment defining geometries for building the platforms that the character should use in order to reach the final objective.

In order to move inside the environment to have a better perspective, but also in order to add a fundamental interaction feature inside it, I added the OrbitControls feature from Three.js, in this way it was much easier to move inside the whole environment.

2.1.1 Environment

After having built the platforms. the next step was to build a way to connect the various platforms in order to allow Nate to have a way to go from one to another, so I built bridges and smaller platforms in order to create connections and obstacles to overcome.

Then, noticing that there was too much void spaces in the beginning platform (so the ground zone), I added some trees using cylinders as geometries for the trunk and dodecahedrons for the leaves in order to give the effect of an irregular surface.

The final step of the environment construction was to build the goal zone, so the temple where the treasure is.

Here I built some columns and broken columns in order to give the idea of ruins, and then added the temple roof, following the style of a canon classic temple.

Then, I added textures and colors to all the elements defined, in particular, the temple and the leaves of the trees have been defined using PhongMaterial, in a way to add lights effect and reflection to those elements.

2.1.2 Treasure

For the treasure, in order to create a gem, I defined an octahedron, the gem is located on a pedestal, and both are placed under the temple's roof.

The gem is animated and rotates around the X and Z axis.

Both the pedestal and the gem have been defined with PhongMaterial in order to be affected by the lights, the gem is green and his texture reminds an emerald, while the pedestal is made of a golden color.

2.1.3 Lights

In the whole environment I've defined three lights.

The first and the second ones are directional lights applied in order to illuminate the treasure zone, that emit a white color.

The third light is a directional light placed upon the scene that has the purpose to simulate the sun light, also this one emits a white color.

3. Hierarchical Models

Two hierarchical models have been defined inside the project, Nate and the tiger.

For Nate, the main node is the head, and he is composed by neck, body, upper arms, lower arms, upper legs, lower legs and a backpack.

In order to compose a hierarchical model, I used the function inside Three.js that allows to add objects in the scene connected to an object yet present in the scene, so in this case I added all the parts to the head (i.e. scene.add(head), head.add(neck)…).

Nate's textures are all color textures, except the face and the head, as the head is defined as a cube, each face has a different texture in order to form the face and the hair.

For the tiger, the model is composed by head, ears, body, anterior legs, posterior legs and a tail. The main node is the head.

The tiger's textures are images in order to reproduce the typical texture of a tiger, the head is a cube where the front face is a tiger's face and the other faces are covered by the typical

tiger’s texture, then also the ears have the front face with a color texture and the other faces with tiger’s texture.

In order to make a more realistic tail, I used TubeGeometry combined with a function for building a custom sinusoidal curve that applied to the tube geometry forms a sinusoidal tube.

4. Animation

As previously described, the animations have been implemented using Tween.js library, the animated objects are Nate, the tiger and the treasure (which is animated using Three and not Tween).

4.1 Nate

The most complex animation is the Nate’s one: The whole scene is quiet in the begging, except the treasure which is always animated, when the animation is triggered Nate begins to jump and walk on the platforms in order to reach the treasure, when he reaches the treasure he begins to cheer in a loop.

The first animation is the walking one, in which Nate moves his arms and legs in an alternate way (as the common human walk) in order to simulate a walking movement, he does it in a loop.

Then, the other animations are movements in all directions, jumps (made from up and down) and rotations in order to change the walking direction.

Then, there’s the final animation in which he cheers, jumps and makes pirouettes in a loop.

All the animations are connected using the Tween.js function “chain”, which allows an animation to start after the end of the previous one, forming so a chain of animations.

4.2 Tiger

The tiger’s animation starts in the same moment in which Nate’s one is triggered.

The animation is composed by the tiger walking in a loop around a tree, waiting for Nate to reach the treasure.

So is composed by a walking animation in which the tiger moves the anterior and posterior paws simulating the walking alternance of a real tiger, then from movements in all directions and rotations in order to change direction, and in the end an animation of the tail which moves in order to simulate the waggling of it.

4.3 Treasure

As yet described before, the treasure animation is done using Three.js code and is simply a perpetual rotation around the X and Z axis.

5. Interaction

In order to interact with the environment and the characters, I implemented some features in order to make changes and “play” a little bit.

Most of the interaction happens using buttons placed in a console in the upper left corner.

This feature has been implemented using the Three.js feature “dat.gui.module.js”, which has been imported inside the code.

5.1 Camera

As yet said, I’ve implemented a dynamic way to move the camera inside the scene, using OrbitControls, which allows the user to rotate, zoom or move the whole scene in order to change his perspective in every moment.

The camera can be moved using the mouse as described in the tutorial chapter.

5.2 Music

To add more emphasis to the context, I added some audio tracks to make the situation more “adventurous”.

Using the Three.js function, Audioloader and AudioListener, I added three audio tracks.

The first one is a soundtrack that starts when the **Start** buttons is clicked and all the animations are triggered, it automatically stops when Nate reaches the treasure.

The second one is the tiger’s verse, which triggers with the **Start** button and can be triggered by the user in every moment using the **TigerSound** button.

In the end, there’s a “victory music” which starts when Nate reaches the treasure, to celebrate the end of the long journey!

Using the **Mute** and **Unmute** buttons, is possible to mute and unmute the soundtrack during the Nate’s animation.

The sounds are triggered using the AudioListener function “play()”, which starts a track, while the Mute and Unmute functions are implemented by changing the volume of the track using the AudioListener function “setVolume()”.

5.3 Lights OFF/ON

In order to manage the lights, is possible to switch off and on the lights placed near the treasure, and the sun light.

The button **TreasureLightOFF** switches off all the lights near the treasure, and can be switched on again using **TreasureLightON**.

The button **SunLightOFF** switches off the sun light, and can be switched on again using the **SunLightON** button.

The process of switching the lights off and on is handled by removing the lights from the scene when switched off (“scene.remove(light)”), and adding again the lights to the scene when switched on (“scene.add(light)”).

5.5 Animations: Start, Reset, Treasure animation

In particular, the **Start** button described in the tutorial, triggers a series of actions, first of all, it starts the Nate’s and tiger’s animations, then triggers the tiger’s verse and the soundtrack.

The **Reset** button resets the Nate’s animation, starting it again from the beginning and restarting the music if Nate is yet arrived to the treasure.

This reset is possible by removing all the Tweens (animations) of Nate from his animation’s array by using the Tween function “removeAll()”, then Nate is literally teleported to the beginning and his animation starts again.

The **TreasureAnimation** button simply stops and resumes the treasure animation, this process is handled by using a Boolean flag which allows the rotation when true and stops it when false.

6. Final Considerations

Developing the project has been an interesting experience, choosing to use Three.js and Tween.js libraries has been fundamental as they are very versatile and make the webGL development more fluid and fast, allowing many alternatives and useful tools to experiment also during the development without going out of the main focus.

Surely this approach to webGL using alternative libraries is stimulating and I hope to extend my knowledge about it in the future.