SAPIENZA
UNIVERSITÀ DI ROMA

Interactive Graphics

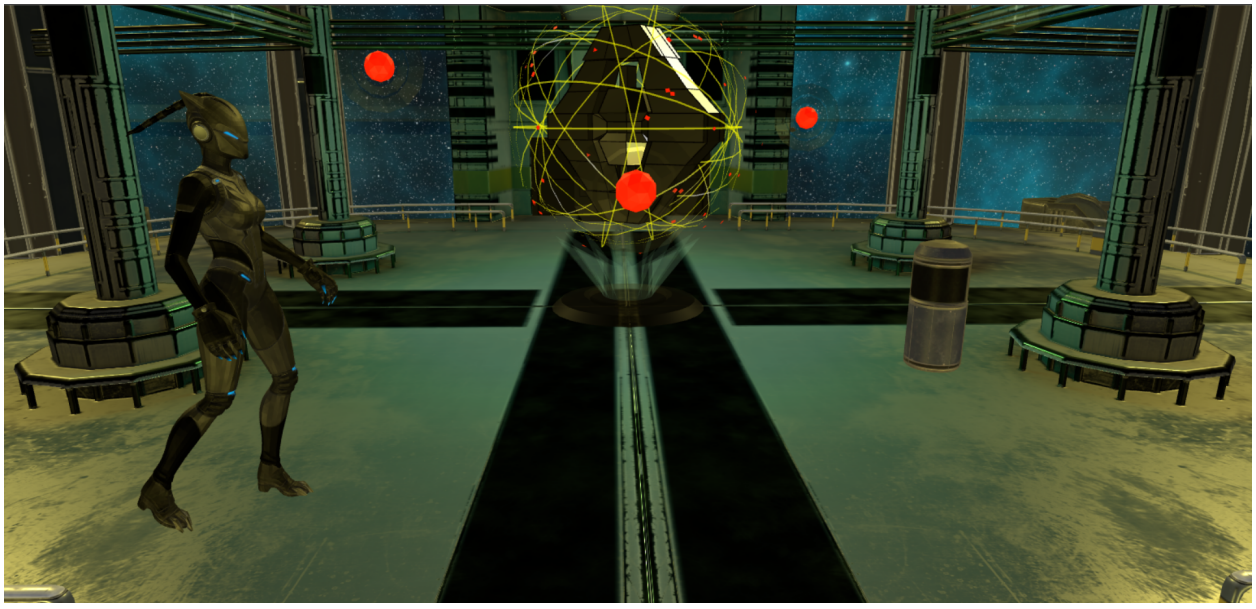
Final Project

Space Crystal

Nazgul Mamasheva

September 2023

In the “Space Crystal” mission the main character, Lynx, needs to complete the mission by finding the special crystal which is located in a hidden space station in galactic and returning it to home planet. After a long journey and many difficulties the main character finally found the location of the crystal. And now is facing the last challenge in the mission, the main character needs to defeat the guards that are protecting the crystal from non-allies.



The event is happening in the galactic space station. In the image above we can see the station, the main character stays on the right side, in the center we can see the crystal with its base under it and toruses. The red flying drones are the guards that can mortify anyone who will try to get closer to the crystal. They are flying around the crystal by protecting it, anyone will be mortified if they collide with them. So, the main character needs to be very careful and somehow get close to the crystal and take it away.
The crystal is in rotate mode all the time along with its base and toruses. The guards are in flying mode all the time. The main character has a humanoid physiology and characteristics, hence moves like a human.

Resources: Libraries, models and textures

The project utilizes several libraries and external resources for different purposes:

- Three.js is employed extensively for rendering tasks, including model and texture loadings, texture mapping, material specification, light model configurations, and the presentation of the scene graph. This includes the ability to establish hierarchical models.
- Tween.js is used for handling the animations within the project. It is also used for defining cyclic animations, thanks to its feature of tween chaining and properties for animations.

Hierarchical model and texture assets are stored in the local folders. These assets are collected from the web.

- Hierarchical models are used for organizing and structuring 3D objects in a scene, enabling complex object composition and simplifying transformations.
- Textures are applied to 3D object surfaces to enhance their appearance, add realism, and provide visual variety.

Hierarchical models

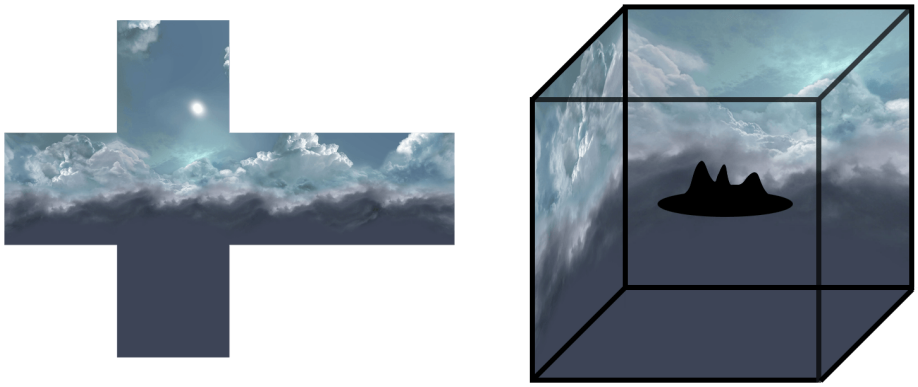
All complex hierarchical models in the project are external and they were downloaded from a 3D modeling platform website Sketchfab. Sketchfab provides 3D models in several converted file formats. I mainly dealt with glTF, since it was a recommended format along with glb from Three.js official documentation. The glTF and glb formats load models in the same way, but I preferred glTF, since it was possible to see the models' structures, their nodes, children nodes, their relations between them, meshes,

materials and other information in json format. Thanks to this glTF file inspection, I was able to utilize the necessary parts of the downloaded models without adding additional models. For the project two downloaded 3D models were used, one is for setting the environment, i.e. the space station model and the other for the main character. Thus, the station, a crystal along its base and toruses, guard drones are all from the space station model. The body parts of the main character were retrieved from the figure model. The models are loaded with GLTFLoader from Three.js.

The loadings of the models and textures have to be done first before executing any other modules. If we do not do the loading first we will face an empty web page or errors. For managing the model loadings I used the LoadingManager function from Three.js. After loading all necessities the LoadingManager will activate its onload callback function in which we can call our other functions, for instance init function, where we do our initializations and function calls with ready parameters.

Textures and Materials

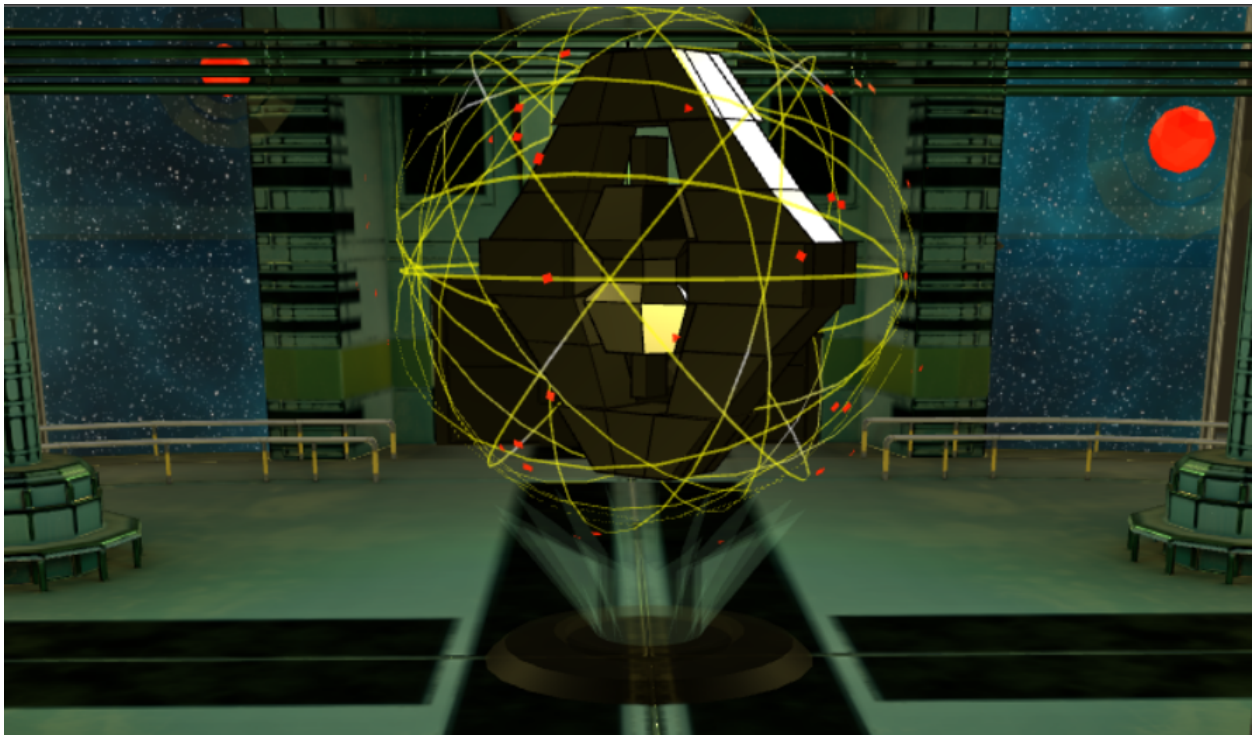
For a space environment I used a skybox texture. The example of it we can see below.



It is typically applied to a cube or a sphere, creating realistic and immersive 3D environments by providing a visually convincing background that envelops the entire scene. Skybox textures are often used in video games, simulations, and 3D rendering to provide a sense of context and immersion.

The skybox texture is loaded with the CubeTextureLoader function from Three.js. The six corresponding texture images were loaded as 6 sides (left, right, top, bottom, back, front) of a cubic scene, creating a sense of galactic.

The other texture that was used in the project is a texture for crystal. As we can see in the image below, the crystal has a metallic material and it reflects the lights coming from the surroundings. The original material of the crystal was a transparent dark gray material. To achieve the metallic reflection I mapped the metallic texture image to material with MeshPhysicalMaterial function from Three.js by setting corresponding values to its parameters such as reflectivity: 1.0, roughness: 0.3, metalness: 1.0.



Lights

Two types of lights were used in the project, one is AmbientLight and other is DirectionalLight.

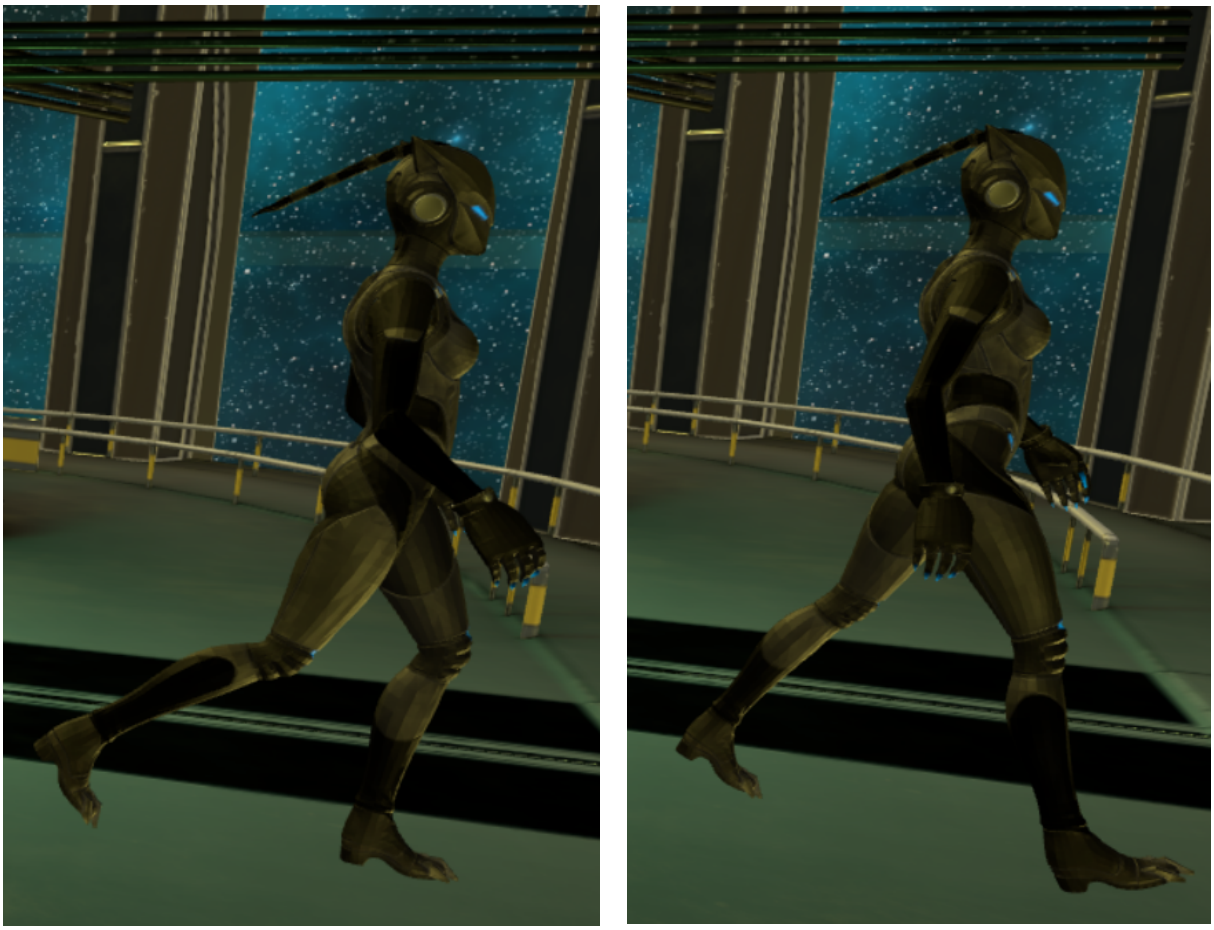
- AmbientLight is a light source that provides uniform, global illumination to all objects in a 3D scene, without casting shadows or creating specular highlights.
- DirectionalLight is a type of light source that emits light in a specific direction, like sunlight, to illuminate objects in a scene, casting parallel shadows.

Ambient light was given with specified color and with intensity of 1.0. Four lights of DirectionalLight were given with specifying shadow properties. They were positioned on four opposite locations in order to make all details on the scene viewable, since the scene is surrounded by dark blue 3D space texture. As we can on project scene images, the added lights enhanced the realistic view of an environment, reflecting the material properties of surroundings.

Animation

The animation was performed by using Tween.js library.

The human-like walking was animated by setting corresponding rotation values for each arm and leg joints of the figure per frame time. By adding the properties repeat(Infinity) and yoyo(true) on the Tween transitions it was possible to perform a swing effect and repeat it. Also, for easing, that defines how the transition will occur, a Quadratic.Out type was added, it creates a sharp transition at the end of transition with a realistic walking effect. Below we can see the example of walking animation, all arm and leg joints move simultaneously by their corresponding transition values.



The crystal, its base (a transparent conus type of object under crystal) and its toruses (yellow circles around crystal) rotate around themselves with no stop. The rotations are performed along the z-axis. To slow down the rotations the frame time was increased to 6000 milliseconds in their Tween transitions, while for the walking animation it is 500 milliseconds. For their smooth rotations without even a little stop on their original position, the Linear easing was added.

For flying animation of drones the chain property of the Tween was used. For each drone there were specified 5 tweens with 5 transition values, in this case the transition is a translation, not a rotation transition in previous animations. Thus, the first tween chains second tween, the second tween chains the third one, and so on, and the last

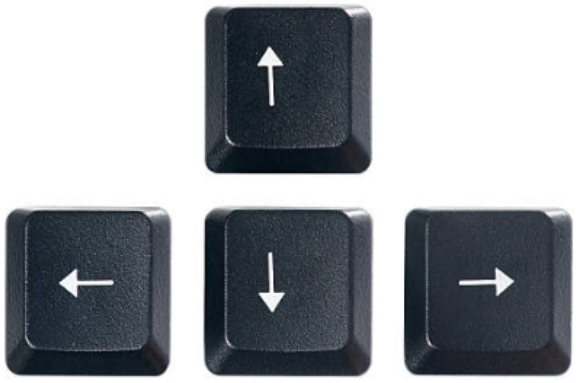
tween will chain the first tween, in this way creating a looping and an infinite transition. The transition values of the first and last tween should be the same values for smooth translation. For easing the Circular.InOut type was added in all chain tweens, since we want a circular motion effect during flying of the drones.

User interactions

For user interaction the orbit control was added in order to allow a user to see all sides of the scene by moving around the camera point view.

- OrbitControls is a utility in Three.js that enables interactive camera control, allowing users to orbit around a 3D scene using mouse or touch inputs.

To control the moves of the figure (a main character) the arrow keys on keyboard were introduced:



The Up and Down arrow keys make the figure move in z-axis, i.e. to the front or back sides. The Up arrow key is for moving to the back side, and the Down arrow key is for moving to the front side. The Left arrow key is for moving to the left side, and the Right arrow key is for the right side.