SGRAI Report

Sprint C

Class 3DE _ Group 32

1200616 - João Silva

1190516 – Diogo Marques

1191606 – Pedro Marques

1220417 – Ibon Eito

1220419 – Jakub Kielar

Date: 08/01/2023

Index

Part I – Increasing the realism of the graphical representation

I.1 Ambient light source

To simulate ambient light, we used the class 'AmbientLight' from the Three.js library. It is used to create a light that evenly illuminates all objects in a scene from all directions.

The first argument is the color of the light, and we used a white color for the light. The second argument is the intensity of the light, and we used the value of 1.

Lastly, we added the light to the scene.

I.2 Directional light source

To simulate directional light, we used the class 'DirectionalLight' from the Three.js library. It is used to create a light that shines from a specific direction and is diffuse, meaning that it scatters equally in all directions.

The first argument is the colour of the light, and we used a white colour for the light. The second argument is the intensity of the light, and we used the value of 1.

We then use the 'position' property to set the position of the light in the 3D space. We chose the coordinates (10,30,20).

Lastly, we added the light to the scene.

I.3 Shadow projection

To simulate shadow projection, we used the property 'castShadow' from the Three.js library. It is used to create a light that shines from a specific direction and is diffuse, meaning that it scatters equally in all directions.

Firstly, we enable the 'castShadow' property in the directional light and enable the shadow rendering in the render. In the directional light shadows, we set the properties of 'mapSize' width and height to 2048 and in the 'camera' properties near and far we set them to 0.5, 500 respectively and also add the field of view to render the shadow map to 500. Then we enable 'castShadow' and 'receiveShadow' in all the objects in the scene.

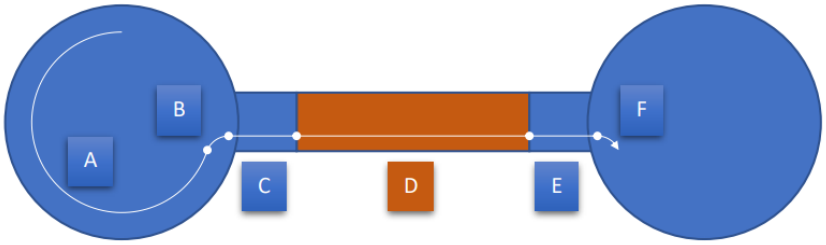
Part II – Importing the corresponding 3D model

To import 3D model of the truck in format '. fbx' we are using 'FBXLoader' from Three.js library. The loader is implemented in both automatic and interactive movements classes and is downloaded from the given uri path. Inside loading function fbxLoader.load() we scale an object to reasonable size and rotate the object inside the THREE.Group wraper to receive correct rotation on X, Y, Z axises given for our project with custom Z and Y axis position.

Part III – Automatic movement

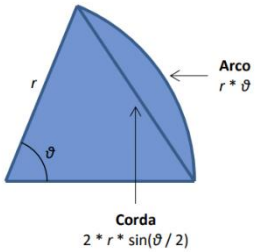
We divided the whole movement between many warehouses to modules which starts from the beginning of roundabout and finish just after taking the correct lane on the next roundabout on the way.

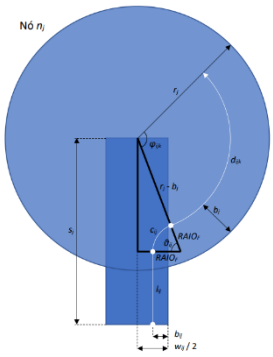
The modules are also divided, they consist of 6 parts: (movement around roundabout, taking off from the roundabout, movement from rounadbout to beginning of the slope, movement on the slope with appropriate tilt, movement from end of the slope to the next roundabout, entering roundabout on the correct lane).



We use different velocities for an object (angular, horizontal and vertical) to accomplish combined movement of all of them, they are calculated based on the type of the movement in the preparation of AutomaticVehicle class in function saveListOfSpeeds(). We also divide each movement to n frames to present it as an animated smoth movement. Function animate() in RoadMap.html is called recursively to make updates of the presented 3d visualization. This function calls a animateMovement() function on each of the animation frame. Function animateMovement() calls changes on every 10 frames to not overuse calculation capabilities. This functions do all the necessary velocity and angle calculations of the truck and then call updatePosition() function. This function based on velocities and angle of the truck update current position of truck for animation frame.

To acomplish correct size of the roundabout movement radius, we subtract half of the angular velocity as a preparation for circular movement and then add it just after finishing roundabout movement.





To make the movement of entering and taking off from the roundabout smoother and more natural circular movement we added values of RAIO for each roundabout. These values are responsible for the distance from the angle of entering and leaving roundabout. Controlling these values for each roundabout we can make the movement looks more smoother without sudden change of angle.

Part IV – Interactive movement

Using event listeners, we detect when the arrows are being pressed. In the update function, which is called in every animation frame, we get the key that is being pressed as well as the time between frames, to calculate how much we are going to move. Depending on the key we send to the truck class the info of the next movement. After calculating the new position of the truck, we need to detect if it is out of the road map. For the truck to be inside of the map it must be inside of one circle, inside of the exit of the warehouses or inside of a road. If the new position is inside one of those elements we accept the movement, if it's not, we return the truck to its previous position.

The way of calculating if it is inside these elements is following the tutorial in moodle:

- $(x'_P - x_i)^2 + (y'_P - y_i)^2 \leq r_i^2$.

If this condition is fullfilled, it is inside a circle.

- $x''_P = (x'_P - x_i) * \cos(\alpha_{ij}) + (y'_P - y_i) * \sin(\alpha_{ij})$;
- $y''_P = (y'_P - y_i) * \cos(\alpha_{ij}) - (x'_P - x_i) * \sin(\alpha_{ij})$.

onto correspondente à nova localização do pers
o ultrapassar os limites do rectângulo que o repr

- $0.0 \leq x''_P \leq s_i$;
- $-w_{ij} / 2.0 \leq y''_P \leq w_{ij} / 2.0$.

This one is to calculate that it is inside an exit road.

We follow the same formula to calculate if it is inside a normal road.

Part V – Other Improvements

We've also added fog to add more realism to the scene using the Three.js class "FogExp2" with the color white and the intensity 0.01.