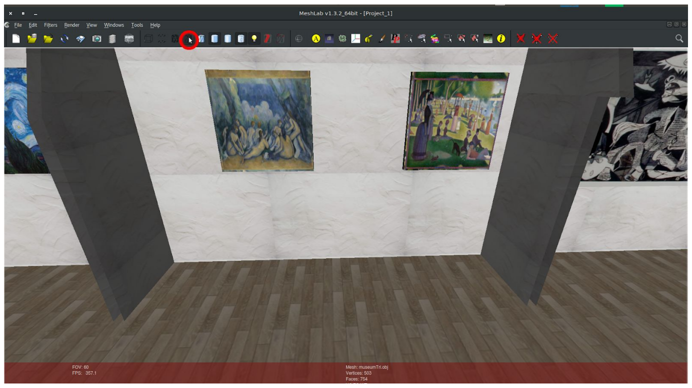
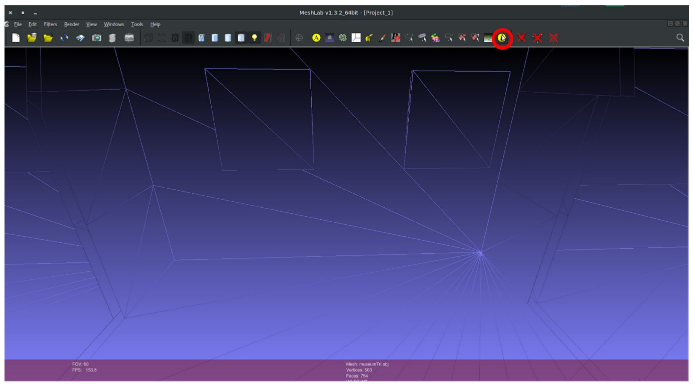
Assets and Hints v1.0

We suggest to use Meshlab (http://www.meshlab.net/) to inspect the 3D models, especially if you need to check and define in your code the position of the paintings (museum), cities (world) or doors (dungeon).

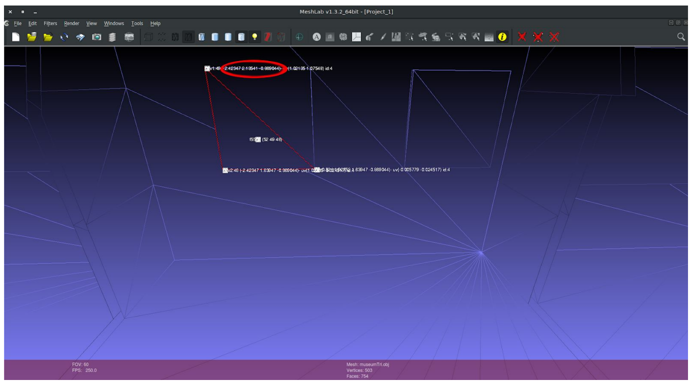
Example:



click on "HiddenLines" view to show the wireframed model



Then click on the info function



And select a triangle to show the position of the vertices

In general, project requirements are loose, to make your creativity free! All the efforts to do nice renders are evaluated and welcomed. Among the others, multiple lightnings shaders, user interaction to change camera point of view, scene graphs (when useful), and all the topics covered in the lectures are subject of evaluation.

Here some description of the assets and some additional detail for the project

In the exercise we provide the utils.js (most recent exercises contains the updated version of utils.js). However, you can use whatever library you want for the maths computation and assets loading, but DO NOT USE any library to ease the shaders design. Knowing how shaders works and how to program them is one of the most important aspect of the project.

<u>01 - Drone Simulator:</u>

You have to let the user choose to move the drone or the camera with the keyboard. Let implement different lightnings. Pay attention to drone-terrain collision. Optionally you can implement a procedure to “follow” the drone with the camera in third person

Drone model: no texture is released, so you can use a color or a texture of your own
Terrain mesh: this mesh represents the terrain the drone has to fly on. It is a regular grid on the x-y axis where the height is represented by the z coordinate.
To know when the 3D model of the drone hit the terrain, you have to check the height of the terrain corresponding to the drone position.
Hint: let define a safe area for the drone (e.g., a cube around the drone) to make the checking procedure easier.
Hint2: the grid is regular therefore to check the intersection you can store the height of the vertices in a regular bidimensional array of javascript (a matrix) and you retrieve the position you need to check just by computing the x,y using a formula like this (position to check - minimum grid coordinate)/(maximum grid coordinate - minimum grid coordinate) You can do something more refined but it is up to you.

<u>02 - Monster Truck:</u>

You have to let the user choose to move the vehicle or the camera with the keyboard. Let implement different lightnings. Optionally you can implement a procedure to “follow” the vehicle with the camera in third person.

Terrain mesh: this mesh represents the terrain the vehicle has to navigate. It is a regular grid on the x-y axis where the height is represented by the z coordinate.
To know to adapt the position of the vehicle to the terrain elevation, you have to check the height of the terrain corresponding to the vehicle position.
Hint: the grid is regular therefore to check the intersection you can store the height of the vertices in a regular bidimensional array of javascript (a matrix) and you retrieve the position you need to check just by computing the x,y using a formula like this (position to check - minimum grid coordinate)/(maximum grid coordinate - minimum grid coordinate) You can do something more refined but it is up to you.

03 - Museum:

The museum file contains both the structure of the museum and the paintings on the walls. When the user is near to one of the paintings, a card with the information about the painting is shown (hint: a 3D plane with the text captured as an image and put on a texture). Optionally you can let the user to choose if you pop up the infos or not. To check if a user is near to one painting we suggest to define and store an area around each painting.

Hint: use meshlab to check the position of the paintings to define the area around the paintings, and the position of the walls to make the user avoid passing through the doors

04 - Histograms:

In this case you do not need special assets, but you can just adapt the shapes from the exercises and the assignments to create bars. You use the data from the histogram.txt file or whatever data you want in the form of histogram.
Let the user navigate around the bars to inspect the histogram and implement different lightnings.

05 - Earth:

The asset is a sphere with the texture of the earth. The user has to choose among a set of cities (let choose at least 5 cities) and move the camera (or the world? make your choice) such that it points the city and a card with the information about the city is shown (hint: a 3D plane with the text captured as an image and put on a texture). We would prefer to see the animation of the world spinning from one city to the other.
Let implement the possibility to choose different lightnings.

06-Dungeon:

The dungeon model contains three maps (with different levels of difficulty) whose occupancy is described in Map asterix file.
The first model is the plain map. Here the user can just navigate the rooms.
The second map contains doors, so in this case you need to implement the door opening animation (if you notice each door has a lever on his left
The third map contains the doors as in the previous case but here one of the door has no lever on his side and in a section of the map you can see a key; here the user is requested to pick the key first and then he can open the door without the lever.
If you are alone, you can choose among the three maps If you are a group of two you have to implement at least the door opening animation of the second map.

Remember to implement different lightnings (chosen by the user).

07 - Solar System simulator

We provide a sphere with different textures of different planets. You have to combine them to create an object for each planet.
You have to provide the animation of the planet movement and you have to let the user navigate freely on the universe. Optionally you can also implement the possibility of viewing the entire solar system from one planet chosen by the user

08 - Air Hockey

The model contains the disk, two paddles and the table. You are not required to implement complicated physics, but just the physics for basic interaction, e.g., constant velocity and no acceleration. Is much more important to focus on implementing the possibility to change point of views and the lightning of the scene.

09 - Furniture

The set of furniture is rather limited and if you want you can download other models and use them. However the available models are enough to implement the application requested.
Given an empty room, the user has to choose the furniture to add, you have also to let the possibility to change the lightning of the room and the point of view of the camera

10 - City Map Generator

Among the assets you will find a set of road segments and city "parts". Feel free to use other assets to enhance the quality and variety of the scene
You have to give the user the possibility to change the lightning of the scene. The user can also change the point of view rotating the camera.