Report

Name: 吳岱容 Student ID: 310551067 Assignment: 2

Email: azusa871227@gmail.com

- ☑ ~~THIS MUST BE YOUR OWN WORK! YES~~

[10%] Introduction: //At least 100 words.

WORD COUNT: 141

In this work, we practice more advanced Ogre3D api, for example, the ray system, the particle system, and so on. We implement many tasks in this homework, combining the given source code into a larger project with a lot more functions.

The biggest part of this work is the ray system. The system detects the mouse action and lets the user interact with the objects by controlling their cursor. The ray system can help us turn the 2D coordination of our monitor into 3D coordination of the game scene.

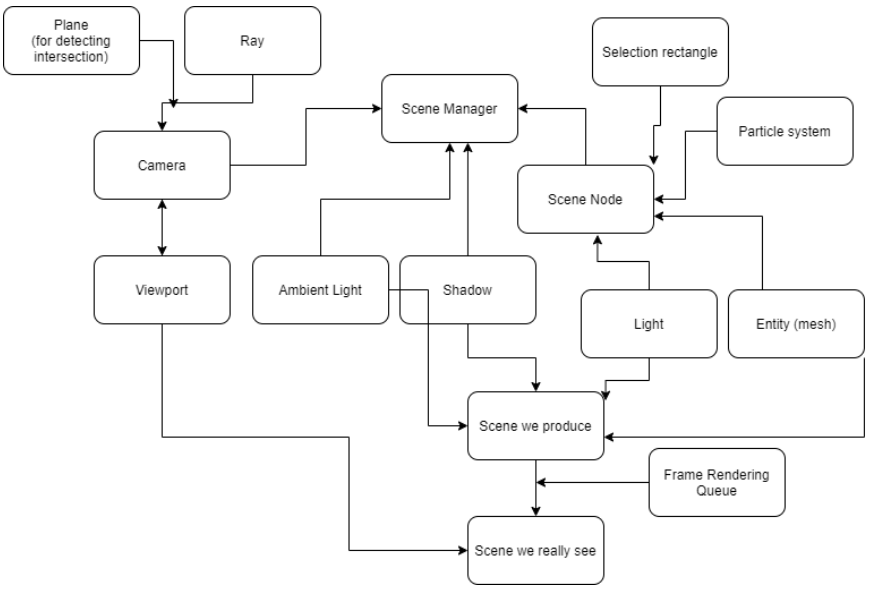
Also, we have a chance to learn queuing the rendering task, which help us updating the mesh state.

Besides all the above, we also need to implement a simple collision system in this work. We have to use our knowledge to come up with an idea to deal with the collision problems.

[10%] System architecture

WORD COUNT: 130

-[5%] Draw a diagram of the system. At least FIVE components.



-[5%] Describe in words about the system. At least 50 words.

We use a scene manager to manage components of the Ogre 3D engine. Lights and Entities are attached to the scene nodes, then it can be showed on the screen. Ambient light, shadow, lights and entities build the scene. Cameras decide how we will see the screen (direction, position, clip distance), and viewports output the camera image to become what we really see in the computer.

Particle systems also need to be attached to a scene node, we can use a child scene node of an entity, so the effect will be around the entity. In this work, we also have the ray system and plane object to help us do the mouse controlling task. Here, we also use the rendering queue to help us to do the animation update.

[30%] Methods:

WORD COUNT: 825

Task 1 :

Item 1 ~ 3, we have already done the same thing in assignment 1, so I just skip it. Item 4 ~6 , about creation of the fog, we just need to follow the api usage then we can finish it. Item 7, we also have done a similar job in assignment 1, but this time we need to create a ground, therefore, I create two plane mesh, and set another's y coordination be lower as requirement. Item 8, I use two for-loops to create the robot circles. I set the position of the robots as z-coordination = radius*sin(θ), y-coordination = 0, and x-coordination = radius*cos(θ). Two for-loops for different radius. Therefore, we will have two circles of robots, and in total 36 robots. Item 9, I just pick the first robot to be the biggest and change its scale. Item 10, I finished creating a sphere by calling api, and I stored its entity and scene node for other tasks. Item 11, I set a condition to detect if the selected object is the sphere or not. I store the sphere's scene node, so I can do this job easily. Item 12, I let the light to move in a circle, and I update its position in the frameStarted function. We have done a similar job in assignment 1, but the movement has acceleration. Therefore, this time I just simply assign velocity without acceleration. Item 13, I refer to the intermediate tutorial, but I alter some function. I set a flag to detect the direction of the corner rendering, so it can work like the spec requirement. Otherwise, I also alter the material to make it look like the demo program. Item 14 and 15, it is also very similar to the intermediate tutorial, but in the tutorial the pressed and released events seem to have strange performance. I set a boolean value to make sure it works well, and also detect the click type to make sure everything performs well. Item 16 and 17, I use the ray system and plan to find the destination's coordination in the game scene. I do the most robot-moving tasks in the function frameRenderingQueued. Since I already have the coordination of the destination, I use the lookAt function to let the robot look at the point before moving. Next, I can get the vector from the robot's position to the destination. I normalize the vector and use it as orientation. By using timeSinceLastFrame and velocity factor, I can make the robots move to the point. Item 18 and 19, I do not think I make a perfect collision system. I just do a very simple detection by using a circle equation. When the frame updates, I set some conditions to detect if the objects have collided with the other or not. If the object is in the collision circle of the other one, it has to change its velocity to stop or change its orientation. The collision circle of an object looks like: (objectB's x-position - objectA's x-position)^2 + (objectB's z-position - objectA's z-position)^2. If the equation is smaller than the square of the circle's radius, the object is in the circle. Item 20, I just adjust the moving velocity to make it look well. Item 21 and 22, the robots can only move when the bounding boxes are shown. As they reach the destination, I will disable the bounding box, and therefore, they will be idle. To disable

the bounding boxes we just need to call the api. Item 23, I didn't finish this job because my PC seems to have some problem using the OpenAL library. Item 24 and 25, I refer to the intermediate tutorial and create child nodes of each robot's scene node, and then attach the particle systems to them, so the particle effect will go with each robot. I set a flag to detect if the effect should be visible or not.

Task 2:

Most of the items can be easily implemented following the instructions. Item 3, it will cause some bugs if we set the value as the spec requirement, which we have already discussed in assignment 1. To fix it, we just need to set the camera position to a slightly different value in x-coordination.

Task 3:

Item 1, I set a flag to detect if the small map needs to be shown or not. If it needs to be hidden, I remove the viewport. If it needs to be turned on again, I call the createViewport01 function. Item 2, I just follow what the instructions do. We detect the mouse movement by using arg.state.z.rel and decide the camera position. Item 3 ~ 5, I modify some of the script in the material file and finish the key event to assign the correct texture. Item 6 ~ 10, in these items we all do similar jobs. We use key events and conditions to adjust some values.

[40%] Discussion:

WORD COUNT: 497

[2%] What do you draw the selection rectangle?

I take the selectionRectangle head file from the intermediate tutorial and use it in this assignment as the selection rectangle object. This class defines the some information of the rectangle and helps us render it.

When the mouse is pressed, we set the corners of the rectangle. The selection rectangle will change with the mouse movement, so we need to reset the corner when the cursor moves. Since the vertices have render order, if we do not follow it, the rectangle can be rendered. Therefore, I set a boolean value to define the direction.

[2%] How do you compute the intersection point between a ray and the plane?

When the mouse is released, the corners of the selection rectangle will be recorded. We can use these points and the ray system to transfer them into the game scene coordination, and then build the plane bounded volume, which can help us select the movables in our scene.

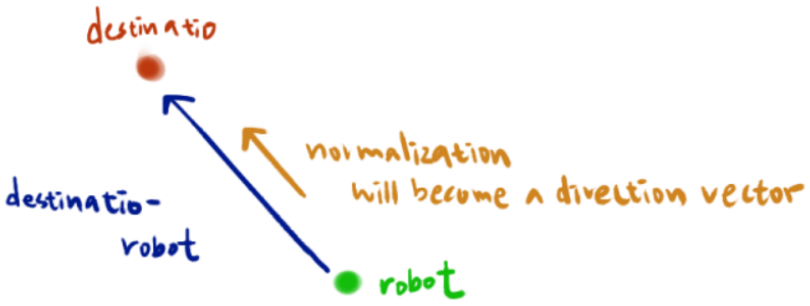
We can use screenToScene function, or intersect function and getPoint function to transfer the point from screen to scene and compute the intersection between the ray and the plane.

[2%] How do you disable skies for the second viewport?

In the function where we create the second viewport, we just need to call setSkiesEnabled(false), and the sky will be disabled, and then we can see the yellow background of the viewport.

[2%] How do you do so that the robots walk to the target position?

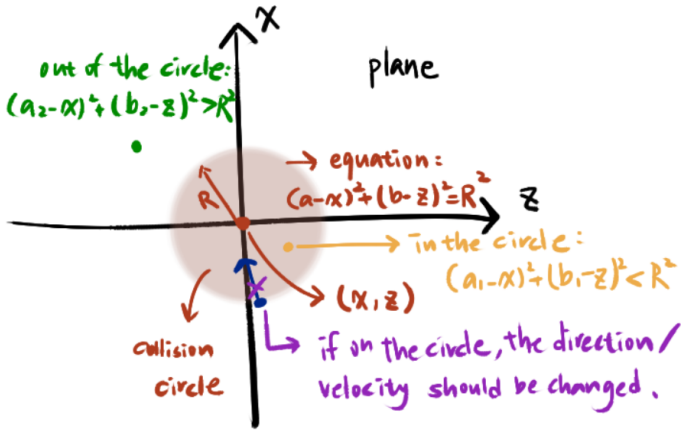
I get the position of the robots and the destination. If I minus the destination with robots' position, I will have a vector, and if I normalize the vector, I will have the orientation points from object to the destination. When the frame updates, I can use timeSinceLastFrame and some velocity factor with the translate function to make the objects look like moving to the destination.



[2%] The background color of the first viewport is set to black. However, why does the background color of the first viewport appear to be white?

It is because we set the fog color to be white. Fog is the object that will cast color on the far objects. The farther from the camera the object is, the closer color to the fog it looks. It looks like a white background, since we don't have any far objects. If we build some far objects, we can see the fog performing on the objects.

[5%] How do you do collision handling for the moving robots and the sphere? Draw a figure for illustration.



I use a circle equation to implement the collision area. As the picture shown above, there are three conditions: (1) in the circle (2) on the circle (3) out of the circle. We can use the equation in the picture and get the position of the objects to check the status. Condition 1 will not happen if the program performs correctly. Condition 3 will be taken as no collision. Condition 2 means that two objects have collided, and we need to do something to solve it. The best solution will need a lot of effort and physic and mathematical knowledge. In this assignment, I only change the velocity or change the orientation. It performs not very well. This doesn't work in every condition. More complicated conditions should be taken into consideration. Maybe I need to look into how other people implement this job.

Do you find any bugs in the demo program?

No, I don't find any bugs in the demo one. The error I encountered is the OpenAL problem.

[10%] Conclusion:

WORD COUNT: 134

In this assignment, we do more difficult and more challenging tasks than the last time. We have learnt how to use the particle system and entity animation to make our game look more fancy, and also we have learnt much about the ray system, which helps us implement a more complicated mouse control. Finally, we try to build a simple collision system by ourselves. It is quite difficult for me. It takes me a lot of time to come up with the idea, but it still doesn't do the very right job. However, I will say that the progress is fun for me. I seldom did tasks like this in the past because most of the game engines nowadays are equipped with physical systems. Combination of programming and physics is difficult, but also fun.

*****BONUS: The best report(s) will be received at most 10% extra points.**

*****Add figures to clearly illustrate your results. Add a proper description to each figure.**