CPT 205 Assessment 2

a three-dimensional (3D) Model

Module Code	CPT205
Module Name	Computer Graphics
Degree Programme	Information and Computer Science
Name	Liangyuting Zhang
Student ID	2142253

1. Design Description

This project implements a 3D model containing some elements like house, traffic light, train, car using graphics techniques and OpenGL functions.

2. Readme

Keyboard:

Press 'a' to move the camera round positive x-axis.

Press 'd' to move the camera round negative x-axis.

Press 'w' to move the camera round positive z-axis.

Press 's' to move the camera round negative z-axis.

Press '1' to change the traffic light to green.

Press '2' to change the traffic light to yellow.

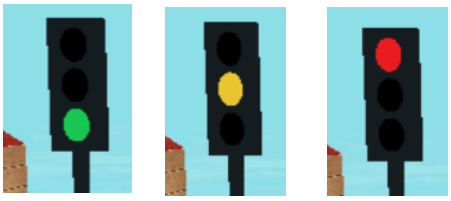
Press '3' to change the traffic light to red.

Press ' ' (space) to change the view.

Press ' q' to quit.

Mouse:

Click right to open the menu, which can control the traffic light using mouse.



(a) press'1' **(b)** press'2' **(c)** press'3'

Fig1.Interaction display

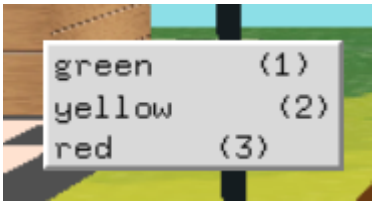


Fig2.Menu

3. Techniques

Lighting effects

Material Properties: glMaterialfv and **GL_SPECULAR** are used to set the specular reflection properties of the object, which is set to white. Then uses **GL_SHININESS** defines the shininess of the object. **GL_DIFFUSE** are used to set the diffuse reflection properties of object. All the properties are applied to **GL_FRONT_AND_BACK** surfaces. **GL_EMISSION** is set to light emitted by the object itself.

Light Source Configuration: glLightfv and **GL_POSITION** set the position of light; the last parameter is 1 indicating the directional light. Then uses **glEnable(GL_LIGHTING)** and **glEnable(GL_LIGHT0)** to enable lighting calculations and the first light.

Depth Testing: glDepthFunc(GL_LEQUAL) sets the depth testing function to less than or equal to, ensuring proper use of the depth buffer.

Texture mapping

The implement enables **glEnable(GL_BLEND)** to allow transparency in textures. Sets the **glBlendFunc** to use the source alpha component for blending. Then the **readImage** function is used to read external **BMP** pictures whose sizes are a multiple of 2. Also, A function similar to draw a checkboard is used to generate a pattern as a texture instead of reading picture.

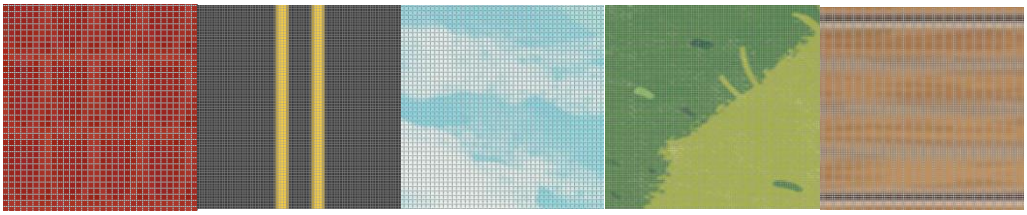


Fig3.Example of texture

Fig4.checkboard texture

glGenTextures are set to generate texture ID and specify it to an **GLuint array.** Then Binds the texture to the **GL_TEXTURE_2D** that set the parameters such as **GL_BGR_EXT, GL_UNSIGNED_BYTE.** The texture parameters are specified finally using **glTexParameterf**

glEnable(GL_TEXTURE_2D) and **glBindTexture** are invoked to map the object, and **glTexCoord2f** defines the 2D texture coordinates to map the vertex coordinates correctly.

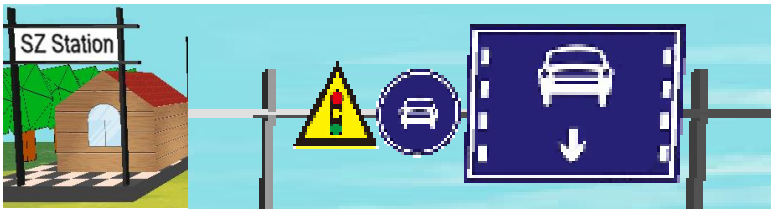


Fig5. Example of mapping effect

Viewing

glViewport is invoked to set the view size to window dimensions. Then the **glMatrixMode** is changed to **GL_PROJECTION**, and loads the identity matrix **glLoadIdentity()** to reset the projection matrix. The **gluPerspective** is used to Sets up a perspective projection.

glMatrixMode is modified to **GL_MODELVIEW** to affect the model-view matrix. The **glLoadIdentity()** is loaded to ensure that the model-view matrix is in its initial state. Then **gluLookAt** is invoked to set the viewpoint of the camera, including the position, look at point, and the upward direction of the camera.

Geometric modelling and Boolean operations

one(),inside(),fixup() and And() are implemented to go on Boolean geometric operations. **glCullFace** sets the faces to be culled. The **GL_DEPTH_TEST** is enabled and **glDepthFunc** are invoked to set various depth function.

The library **<functional>** and template class **std::function** are used to instore

function such as **cube(float...)**, which includes parameters to adjust the translate, rotation and scale. Then the **std::function<void(void)>** objects are used as the parameter of Boolean geometric operations above to implement the solid transformation.

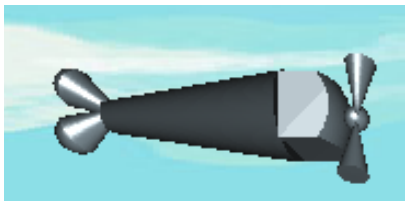


Fig6. Solid use Boolean operations

Hierarchical Modeling

Hierarchical modeling is designed to create complex structures, such as car, lorry and weathercock. The implement are divided into local transformations and global transformations. Local transformations affect only the part and its children, while global transformations affect all its children. The hierarchical modeling provide a ease of animation.

Fig7. Hierarchical Models

Dynamicity

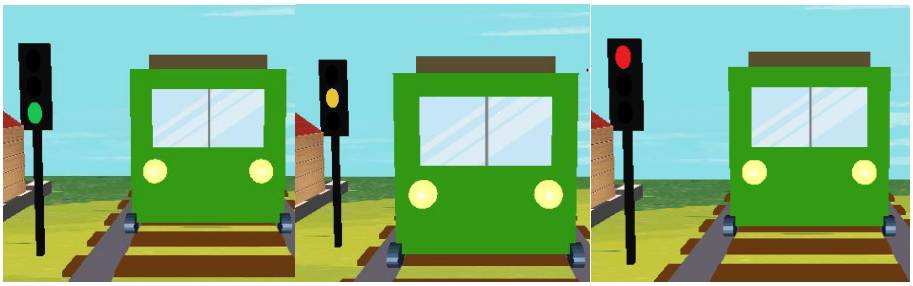
glutIdleFunc is invoked in main function to callback when the program is idle with **when_in_mainloop** as the parameter, and **glutPostRedisplay()** is used to redraw the current window, achieving the dynamic effect.

Also, **glutTimerFunc** is invoked in main function to call after the specified time interval with **OnTimer** as the parameter, achieving the dynamic effect.

Fig8. Dynamic effect of plane

Fig9. Dynamic effect of cars

The interaction of traffic light effects the velocity of train (the effects will display in .exe file).



(a) fast move **(b)** slow move **(c)** stop

Fig10. Dynamic effect of train

4. Screenshots

Fig11. Model display