Interstellar Car Park 3D v3.0

1. Introduction

The software is the second part of VG101 Project 3. The software demonstrates an interstellar parking lot which stops UFOs and alien spaceships. A human being driving a space car goes into the parking lot and stops his car in one of the slots. He should follow the instruction on the ground and reverse into the slot.

1. Quick Start\*

The program uses CMake to manage compilation and uses GCC as compiler. To compiler the project, please make sure that you have installed the following packages in your system: cmake, make, gcc and freeglut.

* Compile on Windows

To compile on windows, we recommend to use CLion, whose toolchain is mingw-64 with all required packages installed. After loading the project root, CLion will configure CMake and after that, just click “Compile” button to compile.

* Compile on Linux

On linux systems you can just use CMake to compile the program. Enter the project root and type the commands

>> mkdir build && cd build

>> cmake ..

>> make

Then the executable will be generated in the ‘${PROJECT\_ROOT}/bin’ directory. Before running the program, please make sure that the ‘data’ folder is along with the executable, or you will see nothing on the screen.

Now run the program and enjoy the future-feeling rendering!

1. Features and Controls

* Hold left mouse button and drag to rotate 3D scene!
* Press ESC to exit.
* A large parking lot with two regions (can extend easily).
* Highly-detailed UFOs which spin on itself while preceding in the reverse direction (compound motion).
* Fascinating spaceships of higher dimensional creatures whose shapes could not be observed (advanced rendering).
* A car follows the path guide and moves smoothly to an empty slot.

1. File Structures

The tree structure of files and directories of the project is shown below:

There are three main folders under the root directory: ‘bin’, ‘include’ and ‘src’. ‘bin’ stores the built executive and external assess required by the program. Header files are all in ‘includes’ directory, while source files are in the ‘src’ directory.

Files in ‘src’ and ‘include’ folders are separated in more detail. Five subfolders organize the code into different parts.

* ‘glm’

GLM is an easy-to-use header only math library foe OpenGL, which provides matrix and vector manipulations as GLSL does. In the project where we need a lot of matrix manipulations to calculate translations, curves, etc., we are not allowed to use external libraries. So we implement a ‘glm’ library by ourselves which contains basic functions like matrix mutilations and solving systems of linear equations and so on. It also takes the advantage of templates which make it type-independent. However, its implementation is different from the original GLM library.

* ‘models’

The code in ‘models’ directory is responsible for managing geometric and textural information of rendered objects. ‘model’ is their base class; ‘figures’ defines some basic 2D shapes; ‘rocket’ is a hard-coded rocket model (unused); ‘ms3d’ loads complicated models from Milk Shape 3D model files. Note, that a model doesn’t contain any transformation information.

* ‘objects’

Code in ‘objects’ defines objects that are actually presented in the scene. An object usually contains a model (which determines the object’s shape), along with its transformation (position, rotation, etc.). ‘object’ is their base class; ‘outline’ is a class of special objects that uses advanced render techniques, which is inherited by ‘spaceship’ to render the ‘merging shinning boundary’ effect. Some object like ‘camera’ doesn’t contain a model. It only does the look-at transform.

* ‘system’

Code in ‘system’ manages the life cycle of the program. ‘window’ opens an OpenGL window and sets up callback functions. ‘engine’ manages all objects in the scene and triggers all events like updating, rendering and mouse clicking, which actually ‘runs’ the program (just as a car engine). ‘grid’ maintains an abstract representation of the parking lot, along with some other functions like path finding and object generating.

* ‘utils’

‘spline’ in this directory implements an algorithm to solve cardinal splines, which is used in guiding the car along a smooth curve.

1. References

Some pieces of code or thoughts relating to concrete detail techniques refer to following websites:

* Designing pattern of the object system, flyweight pattern: <http://en.wikipedia.org/wiki/Flyweight_pattern>
* File structure of Milk Shape 3D models: <http://paulbourke.net/dataformats/ms3d>
* Reading .bmp images and loading to OpenGL texture: <https://stackoverflow.com/questions/41858408/how-to-load-texture-opengl>
* Look-at transform of objects: <https://stackoverflow.com/questions/6992541/opengl-rotation-in-given-direction>
* Introduction to spline: <https://blog.csdn.net/zju_fish1996/article/details/52735947>

1. Copyright Information

The model of UFO is downloaded from <https://www.turbosquid.com/> under the TurboSquid Royalty Free License.

All other models and textures are created by ourselves and everyone can use them without permission.

1. Contact

If you find bugs, have questions, or want to give me feedback, please contact contributors via email:

Zhang Zhenyuan: [lepetitsquarth@sjtu.edu.cn](mailto:lepetitsquarth@sjtu.edu.cn)

Hou Shuyuan:

1. Contributors

Zhang Zhenyuan is he who writes the ‘glm’ library. He sets up the basic structure of the project, and designs models for the car and spaceship (poor crafting).

Hou Shuyuan is responsible for creating the layout of the parking lot as well as the path finding strategy. She also creates some basic models.

They work together on Cardinal splines for the smooth motion of the car.