Computer Graphics Project

2110479 : Semester 1/2013

ZEGO

Mr.Jamorn Sriwasansak
Chulalongkorn University
254 Phayathai Road, Pathumwan, Bangkok
Thailand. 10330
zentojamorn@gmail.com

Mr.Suwapat Kittinanon
Chulalongkorn University
254 Phayathai Road, Pathumwan, Bangkok
Thailand. 10330
raiwait_wure@hotmail.com

Mr. Wichayut Eaksarayut
Chulalongkorn University
254 Phayathai Road, Pathumwan, Bangkok
Thailand. 10330
abatrozz@hotmail.com

Mr.Ariyawat Tanabodichalermrung
Chulalongkorn University
254 Phayathai Road, Pathumwan, Bangkok
Thailand. 10330
ariyawat_yy@hotmail.com

Abstract

Nowadays, Computer graphics software can be found anywhere. These technologies are manipulated and invented for various purposes. Creators love to build high-rises with blocks. We created the program to design the blocks to be easier and more comfortable.

Introduction

This project is part of 2110479 computer graphics subject. Along with tools are POV-Ray and Visual Express C++.

Our project are intent to be able to build and render the blocks as beautiful as possible. Hopefully, our project may benefits the users somewhat.

Background/Related Work

This project came from childhood. One of creators always dream that one day he would build high-rises from blocks with unlimited amount of blocks. It would be great to make such a program that simulate building blocks and can render it out for realistic look.

Approach

- Use POV-Ray to render with ray tracing.
- Use OpenGL to generate pre render model.
- Use c++ to manipulate objects and build OOP structure.
- Use OpenGL to make interface to interact between users and program.
- Use OpenGL and Win32 API to handle keyboard and mouse exception.

Experiment

From the beginning, our program are meant to make block building by load .obj and .zego and render with POV-Ray.

Google sketchup is the main tool to make models. We made a lot of models up to 14 models. After decorated with Maya, models are exported in format .obj and .mtl.

At the start up, program called GLM library to load .obj files. Even many functions are deprecated such as Texture loading but loading .obj and display function are working perfectly. Somehow we are working with blocks; there is no reason to load texture.

After we finish our loading the models are displayed on GuiGl library. GuiGl is special library written by us for harmony of rendering gui and 3d at the same time. For example, GuiGl can render 3d model over buttons and can easily adjust color of buttons and slider.

Then, the world building process started. There is the object, we called world. World is object which we use to handle blocks. At first, We handle blocks by creating array 32 x 32 of stack to keep tracking height of each node. (Node is small cylinder button on tip of block). After we try a lot of experiment, we feel a noticeable slow when we move the blocks down beneath the others blocks. We change from stack to AVL tree and our program become faster even we move the blocks down through 100 heights of blocks.

The blocks sequence is store inside stack so users can undo when they misplace block. This also lead us to save and load system. To save, we just write the whole stack into file. To read, we parse the code into stack.

We can't find any helpful library that could convert OpenGL model to .pov so we program it by ourselves. The library works as code generator. At first, our library make some mistake like turned blocks into wrong direction, misplaced blocks and lighting is not working at all. After we know that OpenGL and POV-Ray use different parameters for orientation and dimension. We recalculate the vectors and result in perfect library that convert from OpenGL models to .pov.

In POV-Ray, all of 14 models are already converted to .inc with poseray before the render button is hit. So when we render, there is no need to convert the blocks every time, the program render. Finally we called POV-Ray with system function in standard library of C which POV-Ray 3.7C is needed.

Conclusion

This project has help us a lot in understanding the way that how 3D or graphical programming use. In this case, we are using Visual C++ 2012 as a platform of our project which we started from nothing. We also use OpenGL library, GLUT library in this project to make it easier to build our 3D scenes and engine. By the time we were working with these library, we had faced some problem we didn't expected but we managed to fix those problems quite well. For engine part we are using those libraries we mentioned earlier to build our interface and scene display. Moreover, we are using POV-RAY to render our scene, make it perfect. But in order to render our frame, we need to create the .pov file for POV-RAY and so the final images are able to be rendered. To deal with this part we are using our main C++ language to create .pov file by generating new codes from all the data we have. (Blocks' position and camera's position) Once we have both our engine and POV-RAY's code, we're ready to go. We merge our Visual Studio and POV-RAY by calling POV-RAY from Windows' command prompt, put our new generated file as the argument and call render.

We have provided an instruction to use our software below.

1. please download POV-RAY version 3.7 beta here
   http://www.povray.org/redirect/www.povray.org/beta/povwin-3.7.RC7-setup.exe
   And install it.

(Note: Older version of POV-RAY is not working with our software. Please make sure you're downloading the right version)

2. Once you finished installing POV-RAY. You should be able to use our software without problem. Double click "CGLegoXZ.exe" to start our program.

3. To navigate through the scene, press W,A,S,D to pan the scene, press '2' to zoom in and 'x' to zoom out, right-click and drag to rotate the scene.
4. You can move the current block through the scene by pressing arrow key. To rotate the block, press 'page up' or 'page down'. And to place the block, press 'enter'. If there is a gap that our block can fit below, you can "dig" through it by pressing '<' or '>'
5. You can also choose different blocks and color of the blocks on the right side interface. If you feel that our interface is blocking the scene, you can press 'h' to toggle the visibility of the interface.
6. Once you finished editing your scene and would like to render, press 'Render' button (a white button located at below right of the program). Our program will run POV-RAY. (This might take a minute to render an image.) The image is placed under the same directory as the exe file.
7. If you would like to save the scene. You can press 'spacebar' with 'F1 to F10' to save your scene and you can load from the slot that you saved (F1..F10) by pressing 'F1 to F10'.

(Note: We have made a software demonstration here [Thai version] http://www.youtube.com/watch?v=L-jBhprDS0w There is no English version of video yet, we sorry for this inconvenient)

References

"OpenGL tutorial"
http://www.videotutorialsrock.com/

"GLUT-like Windowing, GUI, and Media Control toolkits"
www.OpenGL.org/resources/libraries/windowtoolkits/

"Persistence of Vision Raytracer"
www.povray.org

"Characters 3D models**"**
http://www.freebie3d.com/en/search/characters/

"3D PoseRay"
https://sites.google.com/site/poseray

"GLM Library"
https://code.google.com/p/glm/

"C++ Tutorial"
http://www.cplusplus.com/doc/tutorial/

"OpenGL Document"
http://www.OpenGL.org/sdk/docs/man/xhtml

"Screen capture issue in OpenGL"
http://stackoverflow.com/questions/4211486/OpenGL-pixel-data-to-jpeg

"AVL Tree tutorial and implementations"
http://www.oopweb.com/Algorithms/Documents/AvlTrees/Volume/AvlTrees.htm

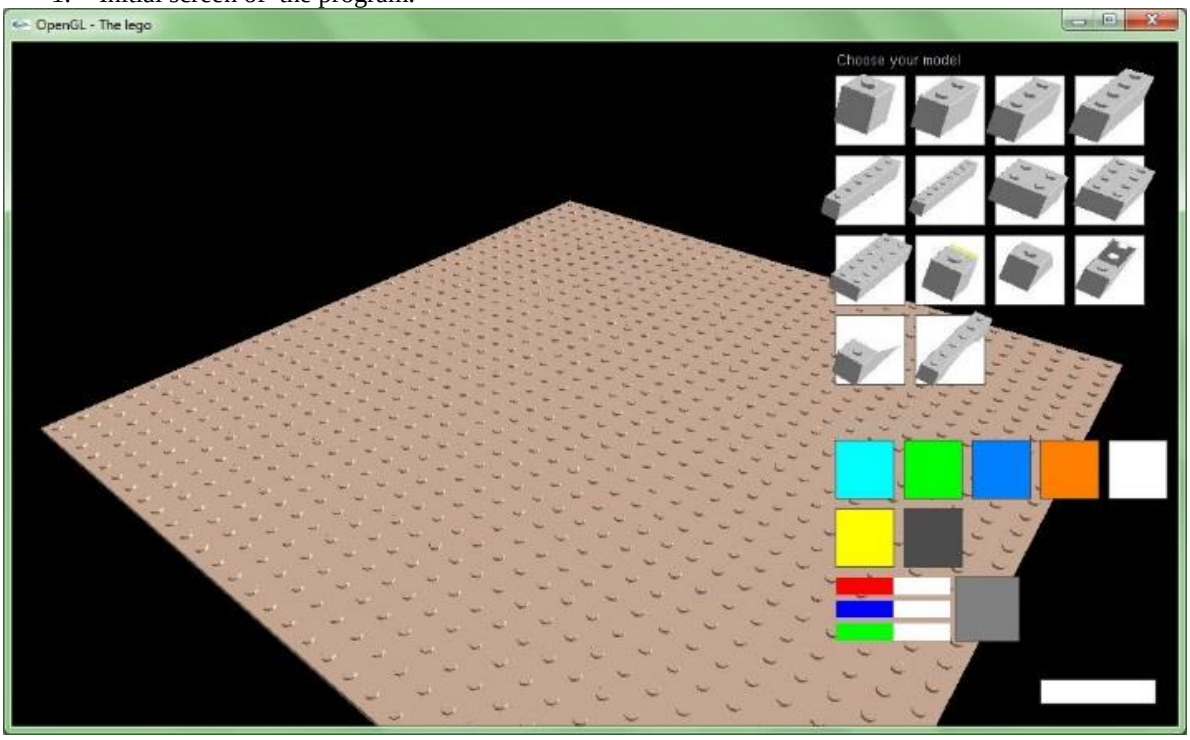
"std::vector complexity and function"
http://en.cppreference.com/w/cpp/container/vector

Supplementary

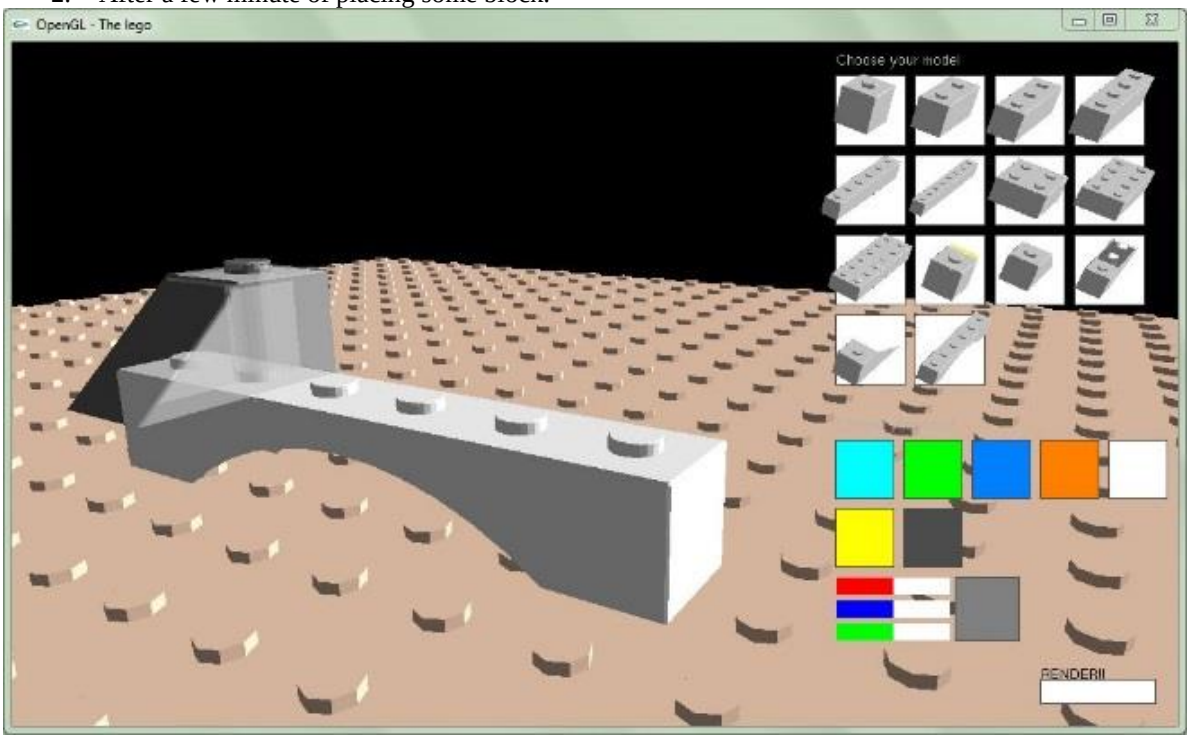
Compressed folder consist of files and necessary files for source code already attached to this document.

Example

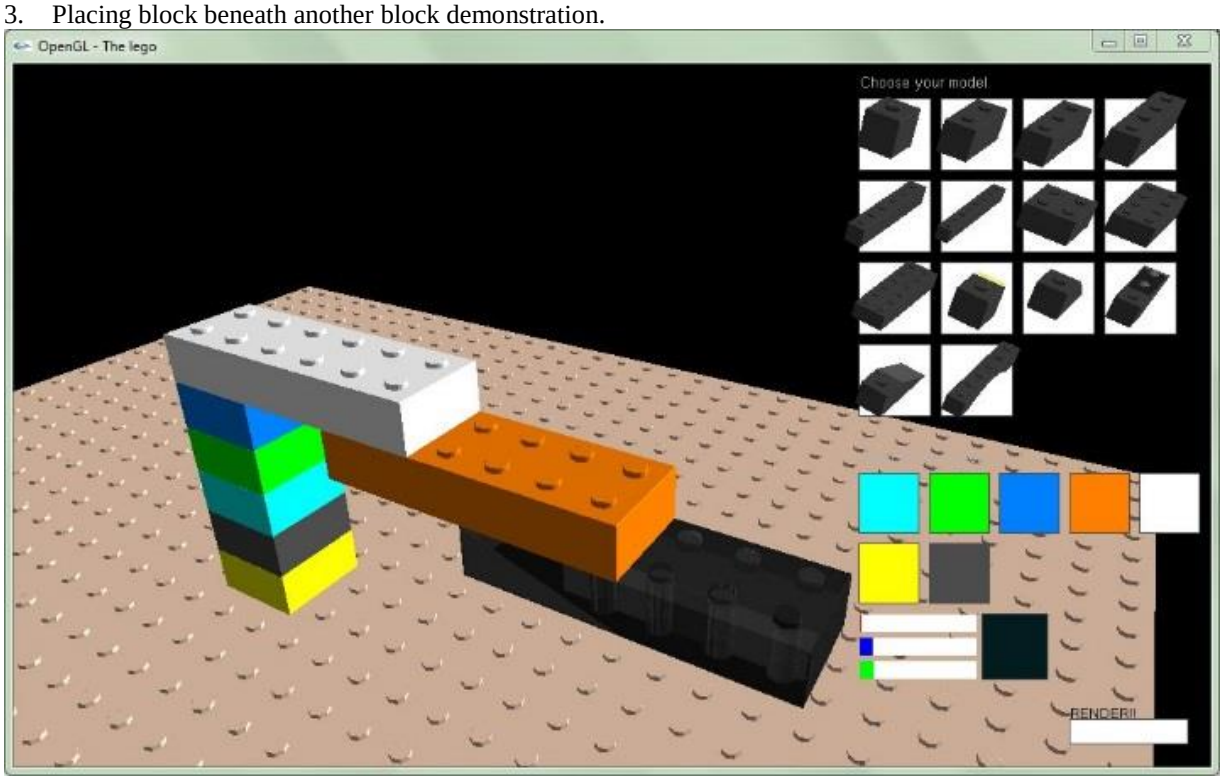
1. Initial screen of the program.



2. After a few minute of placing some block.



3. Placing block beneath another block demonstration.



4. After hit the render button, the povray will open in order to render the model with ray tracing and Zego will call default image viewer to open rendered picture.

5. Try loading previously built model after finish render