Middle East Technical University | Department of Computer Engineering

CENG 477

Introduction to Computer Graphics

Fall 2024
Assignment 3 - 3D Rendering with OpenGL
(v.1.0)

Due date: January 03, 2025, Friday, 23:59

1 Objectives

In this assignment you are going to implement a simple 3D game that resembles the well-known 2D tetris game. You are going to use OpenGL with GLSL shaders and you are expected to write your program using the C/C++ language. The purpose of this assignment is to give you hands-on practice to write a hardware-accelerated computer graphics program. This should complement the theory that you have learned in class and help you understand and solve various challenges of graphics programming. The demo video bundled with the homework package should give you a clear idea of what is expected from your programs.

Keywords: *OpenGL, GLSL, 3D tetris, GLM, GLEW, GLFW*

2 Specifications

1. You should name your executable as "tetrisGL".
2. Your executable will not take any input file. It should simply be run as "./tetrisGL".
3. There is a single game block which is a 3×3 cube. It should be possible to move this block left and right using the controls explained below. This block should fall towards the board at the bottom at a constant speed. It should be possible to speed-up and slow-down the fall as well as to fully stop it.
4. The board at the bottom must be a 9×9 grid. It must have a different color than the falling blocks. It should also be thinner (but it must have some thickness - so it should not be a plane).
5. Both the blocks and the board should have a regular grid-like structure with clearly visible border lines.

6. The following keys are used for game-play: **you should also display these letters on top left**

   A: Move the block left with respect to the current view

   D: Move the block right with respect to the current view

   S: Speed up the falling speed of the block

   W: Slow down the falling speed of the block

   H: Turn around the game area in the left direction

   K: Turn around the game area in the right direction

7. Note that although a block can only be moved left and right in the current view, changing the view and moving the block will actually make it move forward and backward with respect to another view.

8. You should use diffuse, ambient, and Blinn-Phong shading models for the shading computations. You are free to choose your own colors.

9. You are expected to have an ambient and at least one point light source. When you change the view, you can move the light with it or you can have multiple lights to provide lighting for each view. The positions, strength, and the distance of the lights are up to you – just try to make the final result look appealing.

10. You are given a template program which draws a single cube with white borders at the center of the viewport. You can build your homework using this template as a starting point.

11. The game-play should be similar to the demo video shared with you. The blocks should fall in a step by step manner, not continuously. The camera rotations should be smooth, not instantaneous. Blocks should not intersect with each other or with the ground. It should not be possible to move the blocks outside of the game area.

12. The number of steps a block can made before it touches the ground is up to you. In the demo video, a block makes 12 steps before it touches the ground and stops.

13. You should write current view on the top-left corner and the number of points in the top-right corner of the window. The points are equal to the number of block pieces that you collapsed (see the demo video). You should write "game over" when the game ends, which happens when a new block overlaps with the accumulated blocks on the board. **Also write pressed key**

14. **Standard tetris rules apply:** if you fully complete a level, the level should collapse and all the other pieces above it should go down until they land on a block or the ground. At this time, your game points should be updated. When a block touches the ground or lands on top of another block, a new block should be generated from the top-center of the scene. A newly generated block should immediately start falling down at the current speed of the game. If this block cannot be generated as this region is full with blocks, the game ends.

15. In general, feel free to make your own design choices as long as the game-play resembles the demo video shared with you and you support all of the controls. Step-by-step motion of the blocks and smooth rotation between the views is important.

3 Hints & Tips

1. Start early. The game is not difficult to implement but getting used to graphics programming may take some time.

2. It can be a good idea to write a simple bounding box class to detect if two blocks overlap. You can also write functions to check if a point is inside a region.

3. Remember that in the forward rendering pipeline, each frame is rendered from scratch. So do not try to use the contents of the earlier frame. Keep what needs to rendered in a data structure (e.g. `std::vector`) and render them each frame.

4 Bonus

You can get bonus points if you add new features to your game such as different block shapes, different game-play mechanics, or visual effects. But to get bonus points, you must first ensure that your program meets the required specifications.

5 Regulations

1. **Programming Language:** C/C++ for the main program and GLSL for shading computations. You can use any C++ version as long as you provide a makefile and it compiles on inek machines. You can use GLSL version 330, which is the version in the template code given to you.

2. **External libraries:** You can use C++'s standard template libraries such as `std::vector`. Additionally you can use GLFW for window management, GLEW for OpenGL extensions, and GLM for matrix operations. The template code given to you illustrates the usage of these libraries.

3. **Changing the Sample Codes:** You are free to modify the sample code provided with this homework. You are also free to not use it at all. The code is given to get you started. Although it is written with care, we cannot guarantee that it is bug-free.

4. **Development environment:** You can develop the game on your own computer/laptop but your code will be tested on inek machines. It is your responsibility to make sure that the submitted version compiles and works on ineks.

5. **Submission:** Submission will be done via ODTUClass. To submit, Create a "**tar.gz**" file named "tetrisGL.tar.gz" that contains all your source code files and a Makefile. The executable should be named as "tetrisGL". We should be able to extract, compile, and run it as follows:

   tar -xf tetrisGL.tar.gz
   make
   ./tetrisGL

   Any error in these steps will cause point penalty during grading.

6. **Late Submission:** Standard late submission rules defined in the syllabus apply. Do not expect postponement.

7. **Cheating: We have zero tolerance policy for cheating**. People involved in cheating will be punished according to the university regulations and will get 0 from the homework. You can discuss algorithmic choices, but sharing code between groups or using third party code is strictly forbidden.

8. **Forum:** Check the ODTUClass forum regularly for updates/discussions.

9. **Evaluation:** Your submissions will be evaluated based on the expected game-play features and on the overall playability of your game.

do everything in the video