Programming Project: Making a 3D Computer Game

This programming assignment is worth twice the weight of each previous programming assignment.

In this project, you can make any 3D computer game that you wish to, but I have provided the description of one default game (BlockOut) that you may attempt.

Deliverables and Schedule:

1. Short design document, describing your approach in detail. This document should contain a description of the various modules of your implementation. This is due on FRIDAY April 25th, 2003. If you are not attempting the default project, you must see me in my office before Friday.
2. Complete Project and Report is due on Monday May 5th, 2003

Default Project: BlockOut

The original implementation of BlockOut, by California Dreams is available in the Common Folder. Your implementation should mimic the original in terms of functionality, but may be different in terms of aesthetics. For example, you do not have to make a similar splash screen, or use the same fonts to display the score. However, in terms of basic functionality, the game should be the same, and should provide as similar ‘playing feel’ as possible.

For full credit, the following features must be implemented:

• ‘Flat’ block set of the game. To see all pieces in this block set, run the game after choosing ‘FLAT’ in the Choose Setup/Change Setup option. Six types of blocks are available in this block set.
• A grid, identical to the one in the original game, of 3×3×12 dimension. This is the only grid-size that you have to implement, unlike the original game, which offers a variety of grid sizes.
• Rotation functionality: Q, W, E keys should rotate the current piece counter-clockwise and A, S, D clockwise. You should generate rotations that have exactly the same ‘sense’ as in the original game.
• Translation functionality: Arrow keys should move the pieces.
• Restrictions on motion: Rotation and translations should be consistent with the dimensions of the grid. If, as a result of one of these movements, the playing block is going out of the grid, corrective action should be taken. In the original game, translations are automatically added to correct for a rotation going out of the grid.
• Maintaining the state of grid: When a layer is complete, it should be deleted, and the layers above it should be moved down by 1 unit.
• Collision detection: Your program should detect when a block has collided with the blocks at the bottom of the grid, and place it appropriately. Blocks, before they are placed are wire models; after they are placed at the bottom, they become solid (but the grid lines remain visible).
• Action on pressing the rotation/translation keys is ‘instantaneous’, while the block moves downwards at fixed increments. This functionality should be mimicked.
• Each level in the grid should appear as a different color.
• Game should end when the grid becomes full to the top.
Score should be maintained. A reasonable scheme is to grant points for each 'cube' played. At higher levels, points per cube are higher. In the original game, a small bonus is provided every time the pit is completely emptied. This is optional to implement.

Implement 10 levels, in increasing increments of speed. The user may start at any level. Level changes after playing a certain number of 'cubes', e.g. level 8 may start when 800 cubes are played. So if I start playing from level 8, I will stay at level 8 for a long time (till 800 cubes are played). But if I am coming from level 7, I may stay at level 8 only for a small time, because I had to play a lot of cubes at level 7 too. (A game piece is anywhere from 1 to 4 cubes).

Score must be maintained and displayed on the screen. The current level should also be displayed on the screen.

The playing window should resize reasonably, without distortion.

A screen shot of your program may look like the following:

Optional Functionality: The following features may be implemented, but are not necessary for obtaining full credit. However, if you implement additional features, they may compensate for short-comings in the basic tasks.
• Saving and displaying high scores table
• Sounds
• Depth bar on the side
• Introductory splash screen
• Help, menus etc.
• Additional block sets
• Variable pit sizes
• Accelerated fall of pieces by pressing space-bar
• Slow rotations (and variable rotation speeds)
• Demo mode (see demo mode in original implementation, in which the computer plays the game intelligently!)

The report accompanying the implantation should describe your design choices, and issues in implementation. A good report is one through which the reader gets a complete idea of how to implement the game and how to address significant implementation issues.

The project must be attempted individually. You cannot consult other people’s notes while coding, though you are allowed to discuss implementation issues with fellow students. Ask the TA or myself for help.