Tetris Documentation

1725448

January 2020

1 Build and Run Instructions

Simply execute the following command to build the game from source:

```
$> make
```

To run the program, execute this command with the following options:

```
$> ./tetris <difficulty> [2dmode] [width] [height]
```

- Difficulty (Required): 0 - Easy mode, 1 - Normal mode, 2 - Hard mode, 3 - Unfair mode
- 2D mode: 0 - 3D projection, 1 - 2D projection
- Width, Height: Dimensions of the game box (where Tetrominoes are dropped)

1.1 Controls

- W - Drop down
- A - Move Left
- S - Move down 1 block
- D - Move Right
- Q - Rotate Anti-clockwise
- E - Rotate Clockwise
- P - Pause/Unpause game

2 Current Features

The game is playable: it provides a game box to drop Tetromino blocks into and block movement within the box. The game checks for lines of cubes, which are removed and the cubes above are shifted down. Tetrominoes can move left, right, down, rotate left, rotate right and drop down to the bottom. A look-ahead copy of the current block is also provided to the player. Game over is checked by detecting any placed cubes outside the height of the box.

All seven Tetromino block types are available in the game, which are shown in Figure 1. These blocks come in six colours: Red, Green, Blue, Magenta, Cyan and Yellow.

Information signs are also provided next to the game box. It displays the current score and the next block selected. Points are gained per line completed and a multiplier is applied when multiple lines are completed at the same time: Given points per line p and multiplier m, when n lines are completed simultaneously, the points gained P are:

$$P = \sum_{i=0}^{n-1} pm^i \tag{1}$$

Difficulty modes are provided, which affect the speed of the blocks and the scoring parameters. There is also an "unfair" difficulty, which is the hard difficulty, but with half the box height.

The player can also pause the game by pressing the p key. When the window goes invisible, the game automatically pauses the game.

Graphical improvements to the game include a blue sky and green grass background and simple lighting, with a metallic shine applied to the cubes and Tetrominoes.

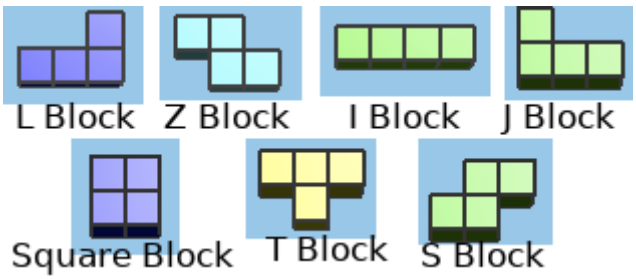

Figure 1: Block types provided in the game

3 Design Detail

Display lists are compiled to draw a 1 unit wide cube and a 1 unit wide cube frame. This is done by first specifying the positions of each vertex of the cube and then specifying its faces. Then, glGenLists() and glNewList() was called to start compiling OpenGL commands. After drawing each face of the cube, glEndList() was called to finish compiling. These display lists make up most of the objects in the game.

Tetromino objects consist of position data, dimension data, its angle and a 4-element array which specify the position of each of its constituent cubes, relative to the origin position - the bottom left corner. Tetrominoes have 4 angles: UP, RIGHT, DOWN, LEFT. An enum object is used to define these angles, so to maintain the anti-clockwise order. Rotation is done by adding the number of turns to the angle (Positive is anti-clockwise and vice versa), looping back when the angle is out of range. Therefore, each Tetromino subclass must implement each of the 4 orientation methods, which change the dimensions and the cube arrangement of that block. Each of the Tetromino's cubes are drawn by calling glCallList() on the cube and cube frame handles, after translating the model matrix to cube's position.

The game box is set up as a 2 dimensional array of Cube objects. When the current Tetromino block needs to move, collision is checked by performing the move on a copy of the block and testing each of its cubes if it is out of bounds or overlapping a cube in the Cube array. If so, the move is cancelled, unless it was moving down in which case its cubes are set into the Cubes array. Once set, only the rows where the Tetromino is placed in are checked if they are full of cubes. This saves from having to repeatedly check the whole board for complete rows. The lookahead block is created by making a copy of the current block and moving it down until it collides with a cube.

When drawing the status signs, the cube display list is reused here by translating and scaling the model matrix to the position and dimensions of the signposts and signboards before executing the display list.

There are 2 material types used for lighting: wood and metal. Wood has high diffusion, but low specular reflection - used for the sign and box border; metal has low diffusion and high specular reflection - used for the cubes and Tetrominoes. To allow such reflections, a normal is created using glNormal3fv() for each face of the cube in the cube display list. However, scaling affected the lighting, so glEnable(GL_NORMALIZE) was called to fix this problem.

4 Word Count

770 words