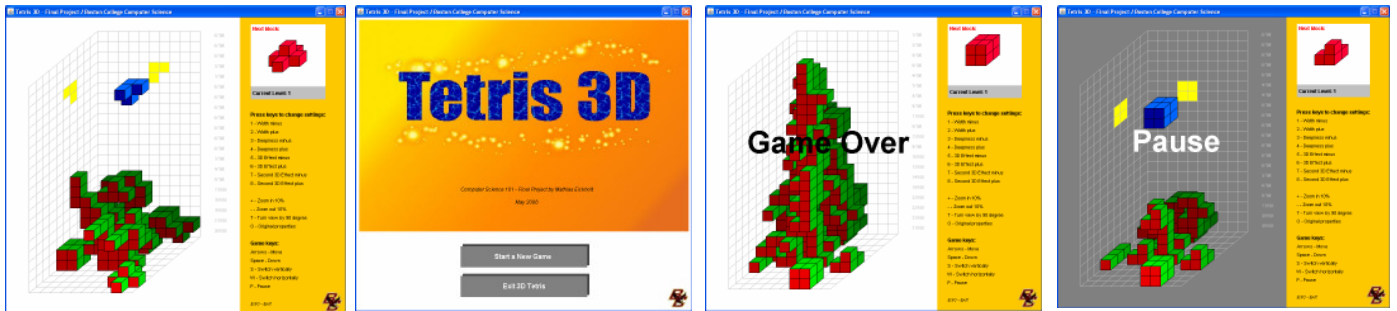


Final Project CS 101 – Final Project by Mathias Eickholt “3D Tetris”

Topic:

A 3-dimensional classic Tetris Game with variable size of the board and user-selectable perspectives (zoom, move around etc.).

Screenshots of the final program:



Program Features:

- It is a classical Tetris game with one more dimension
- The field size is 10 x 10 x 20 (whereby this is variable)
- There are so far 10 different block structures defined (in the class ball.java) – new ones can be add easily though
- As in a classical game a level disappears if there are all fields occupied. However since the difficulty of a three dimensional game is higher this parameter was set to 70 percent – that means when at least seventy of the 100 fields of one level are occupied the level is completed.
- Also similar to a classical Tetris game the level and the speed of the block structure coming down increases with every successfully fulfilled level (shown in the gray box on the top left)
- The game is over if any block is stuck in the top level.

Control:

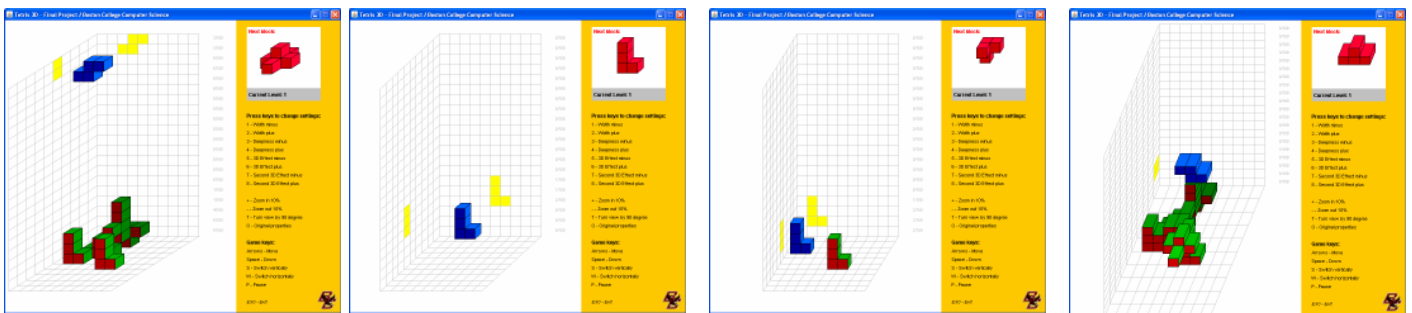
- The game starts with an opening screen with two buttons
- The user has to choose with the mouse “Start a game” or “Exit the program”
- The game itself is controlled with the keyboard as shown in the information field on the right of the main game screen (pressing “P” e.g. leads to a pause of the game)
- The Escape-Key ends both a pause and a current game after checking with the help of a dialogue box.
- To switch around the current structure actually two keys can be used: horizontally and vertically switching of the structure.
- To help the player the upcoming structure is shown in the top right of the window

Extras:

- A sound is played with every move of the current block (also called the “ball”) and when sending down a block completely. A separate sound class and sound objects are used therefore, so that also more than one song can be played simultaneously (new instances)
- A statistic on the right of the field shows the progress of each level. 35/100 means e.g. that 35 fields are occupied but 65 are still waiting for a block.

Graphical Options:

- As indicated on the right in the main window screen there are plenty of graphic options. Basically the paint() procedure of the window is overwritten and used to display the game field. Changing single parameters leads to a new display.
- Changing the zoom-level, moving the field and changing the angle of view are only a view possible options here (in the following: different perspectives):

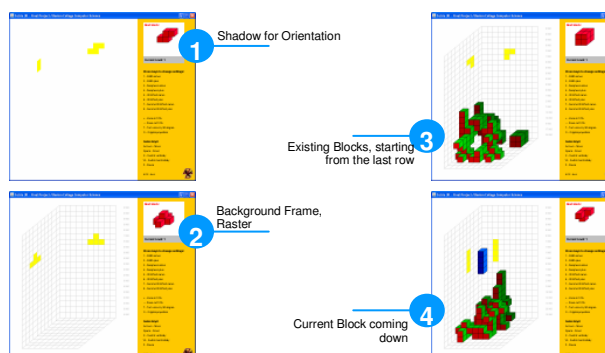


- Moreover, possible turns by each 90 degree of the whole field allows the user to see parts of the game which might be hard to detect with a single-view game (press “T”-key) – it’s at the end not more than a change of the angle of perspective.
- The colors used in the game can be changed easily in the parameter part of the program. At the moment blocks in the last rows have darker colors to strengthen the optical 3D effect

Painting Process:

- The field is basically painted in 4 different steps as shown in the following graphic. Different steps are important to display the 3D effect, helper etc.

The painting process...



```

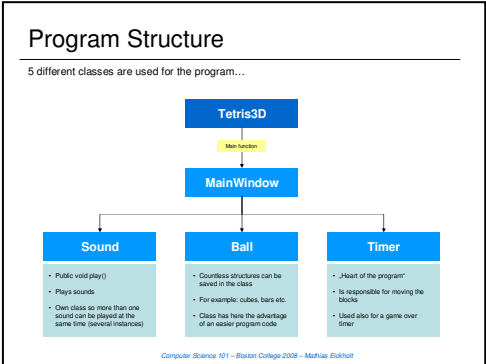
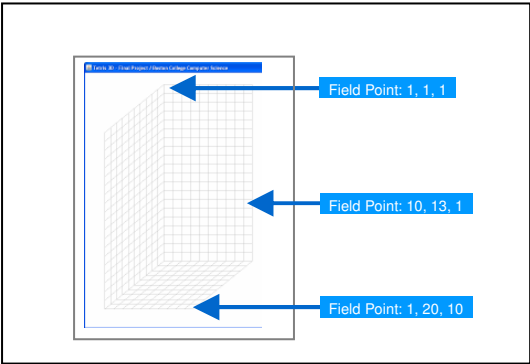
//calls different paint methods
public void paint(Graphics g)
{
    //paint new only if game not stopped
    if (start_game==true)
    {
        if (stop_game==false)
        {
            paint_backgr(); //white part and orange part on the right
            if (stop_ball==false) paint_orien(); //yellow helpers
            paint_field(); //raster and lines
            paint_blocks(); //blocks
            if (stop_ball==false) paint_ball(); //current structure
        }
    }
    [...]
}
g.drawImage (image, 0, 0, getWidth(), getHeight(), this); //double buffer
}
}

//update calls paint
public void update(Graphics g)
{
    paint(g);
}

```

Notable Programming Issues:

- To avoid a blinking or slow display a double buffer structure is used for the graphical output (image object)
- Almost all important variables and constants are defined in the first part of the main class – so the advanced used may change the preferences here.
- 4 classes were defined whereby the MainWindow class is the most important one.



- The field is saved in a 3-dimensional Array with the values 0 for empty and 1 for occupied (could be extended to different values for colors etc.)

```
public int[][][] aa = new int [10][20][10];
```

- To simulate a moving structure the Thread class is used (external)
- Switching around the “ball” was part of the hardest programming tasks. The ball is now also a 3D-Array with the size (1..4, 1..4, 1..4) but saved in an own object (class Ball.java). Switching the ball means therefore a 90 degree turn of this cube with empty and occupied fields.

```
public Ball bl = new Ball();
```

- The code is kind of complex and mostly based on arrays – it therefore should not be shown here. However it is extensively commented and can therefore be understood pretty quickly.

Further Ideas:

- Further ideas to improve the project might be a high score list which would probably increase the long term motivation.
- A block structure editor (colors, structures) could also be added
- Multiplayer functions

The program code is continuously commented which should help an advanced user or programmer to understand the basic idea of the game.