CSE 4200

Project

Joey Turczak

PUYO BLOCKS

Objective

The objective of this project was to create a game based on the popular Puyo Puyo video games using Opengl. The game is simple. It is played on a 6 x 12 grid. The player has control of 2 random blocks that are connected to each other with each block being one of 5 different colors. The blocks can be moved left and right and rotated clockwise and counter-clockwise. Once 4 of the same color blocks are touching, they will be removed from the game space and points are awarded to the player. Any blocks floating will be dropped and if there are 4 of the same color touching again, then the player will be awarded points with a bonus multiplier. The game is a game of endurance. The player plays indefinitely until they fill up the third column. The player can start the game over if they desire.

Abstract

The main Opengl features in this project include:

- gluSolidCube() and gluWireCube() and matrix transformations using glPushMatrix() and glPopMatrix() to display the entire game including the background and the game area. The transformations used include translating and scaling.
- Bitmap displaylist is used to create a font for displaying characters.
- Keyboard commands are used to move and rotate the player.
- The Opengl elasped time is used to control the speed of transformations and rendering to animate the game.

Introduction

The reason this work is significant is that it is showcasing what I have learned in this course. This includes defining the screen space and the world space, using colors, using timing for animations, creating and using fonts with bitmaps, and matrix transformations on world objects. Additionally, this work is significant in terms of organizing everything into the structure of a game including keeping track of the game state, using a recursive algorithm to determine if there is a chain of 4 or more connecting blocks of the same color, and displaying everything properly. This project was meant to be a challenge, and when I first decided to do it, I was not confident that I could pull it off. This is a culmination of much of what I have learned in this course and the final result far exceeds my initial vision and expectations for the project.

Software Package

The first thing I'd like to highlight is defining the world space. It's simple, but for this project, it was important to have a well defined space because the position of everything within the world space is important. I wanted the world space to be in a 10 by 16 space. That is set up in the reshape function:

```
void reshape(int w, int h)

{

    glViewport(0, 0, (GLsizei)w, (GLsizei)h);

    glMatrixMode(GL_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(-5, 5, -8, 8);

    glMatrixMode(GL_MODELVIEW);

}
```

The next thing I'd like to highlight is how I drew everything. Even though this is a 2D project, I used 3D objects. Everything on the screen with the exception of the text is drawn with either gluSolidCube() or gluWireCube() because I only needed rectangular objects. These are then shaped with matrix transformations. Here is a snippet of code which draws the main play area:

```
// Main play area
glPushMatrix();
glColor3f(0.0, 0.0, 0.0);
glTranslatef(-1.0, 1.0, 0.0);
glScalef(6.0, 12.0, 1.0);
glutSolidCube(1.0);
glPopMatrix();
```

Here we see that first the cube is scaled first and then moved into position and colored black. This happens with each element on the screen.

The next thing I'd like to highlight is how the player is drawn. The player consists of 2 cubes which can be oriented in 4 different ways. The cube at player's position stays the same when rotated, but the second cube can be rotated around. Here is the code for drawing the player:

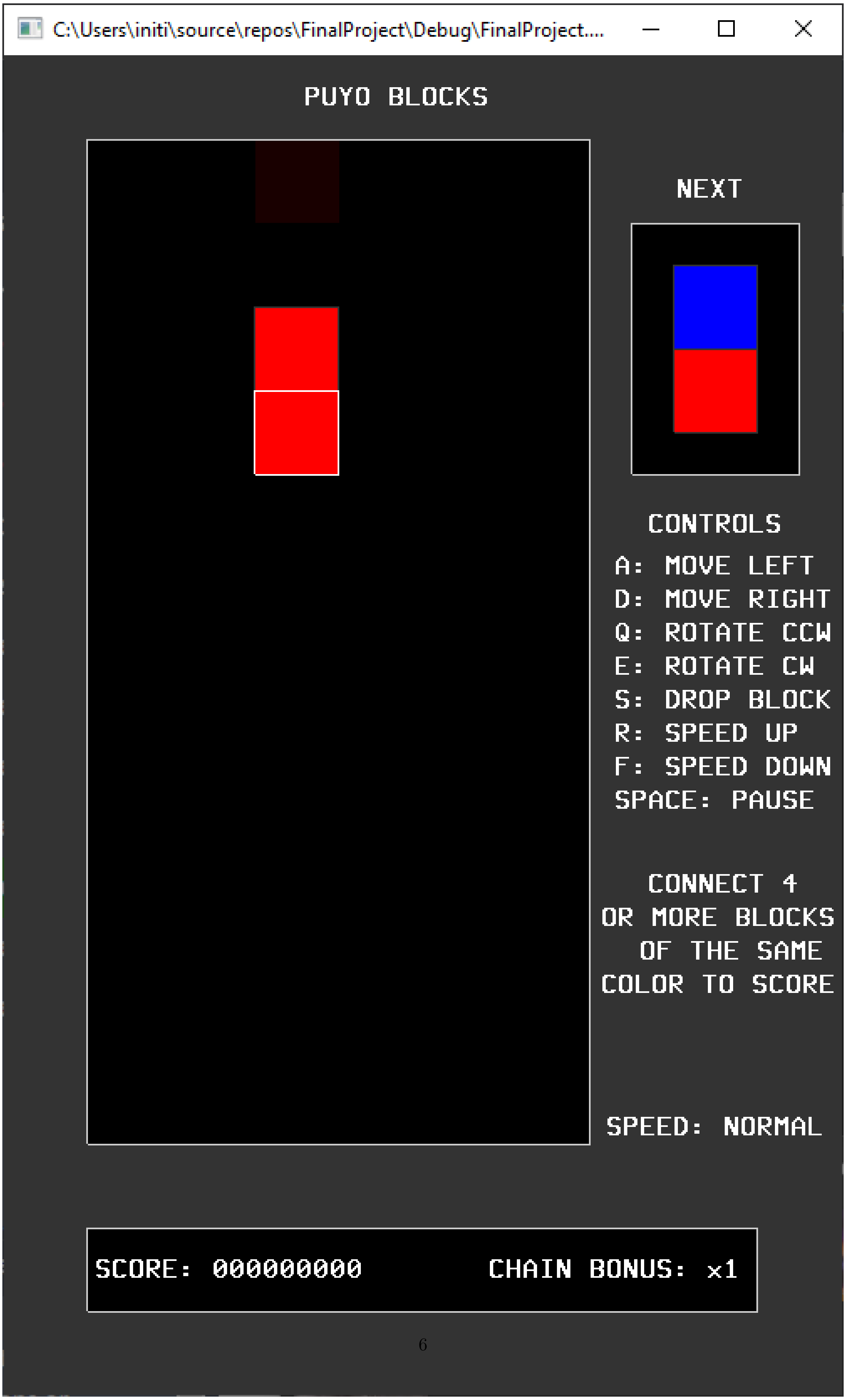
```
void drawPlayer()
{
    // Only draw in the frame
    if (position[1] < startPosition[1])
    {
        glPushMatrix();
        colorBlock(currentBlock[1]);
        translatePlayer(1);
        glutSolidCube(1.0);
        if (currentBlock[1] == empty)
            glColor3f(0.0, 0.0, 0.0);
        else
            glColor3f(0.2, 0.2, 0.2);
        glutWireCube(1.0);
        glPopMatrix();
    }
    glPushMatrix();

    colorBlock(currentBlock[0]);

    translatePlayer(0);

    glutSolidCube(1.0);

    if (currentBlock[0] == empty)

        glColor3f(0.0, 0.0, 0.0);

    else

        glColor3f(1.0, 1.0, 1.0);

    glutWireCube(1.0);

    glPopMatrix();

}
```

In this code it first draws the block that can be rotated and then draws the block at the player's position. The translatePlayer() function moves each block according to the orientation of the player. The blocks are not really rotated, but instead glTranslate() is used to move them into the correct position:

```
void translatePlayer(int blockNum)

{

    float x = gridCols[position[0]];

    float y = gridRows[position[1]];

    if (orientation == down)

    {

        x = gridCols[position[0]];

        y = gridRows[position[1]] + blockNum;

    }


    if (orientation == left)

    {

        x = gridCols[position[0]] - blockNum;

        y = gridRows[position[1]];
    }


    if (orientation == up)
    {
        x = gridCols[position[0]];
        y = gridRows[position[1]] - blockNum;
    }
    if (orientation == right)
    {
        x = gridCols[position[0]] + blockNum;
        y = gridRows[position[1]];
    }
    glTranslatef(x, y, 0.0);
}
```

Interface

The game starts immediately with the player block dropping. The block can be moved to the left by pressing 'a' and the right by pressing 'd'. The block will stay within the bounds of the play area. The block can also be rotated by pressing 'q' to turn clockwise and by pressing 'e' to turn counter-clockwise. Pressing the spacebar pauses and unpauses the game. Pressing 's' will drop the block fast. Pressing 'r' will speed up the game for harder difficulty. Pressing 'f' will slow down the game for easier difficulty. The interface shows the game area, the controls, the objective, the next block, the players score, and the player's score multiplier. The objective is to connect 4 blocks with the same color. They will disappear and then the score will be increased based on how many blocks were eliminated. The dark red spot indicates the failure point. If a block is occupying that space, that means game over. Here is a series of images of the game. The first is the beginning of the game. The second is a game in progress where 4 blocks have been connected and have turned white, indicating to the player that they will be removed. The final image is the game over screen.



C:\Users\initi\source\repos\FinalProject\Debug\FinalProject....

PUYO BLOCKS

NEXT

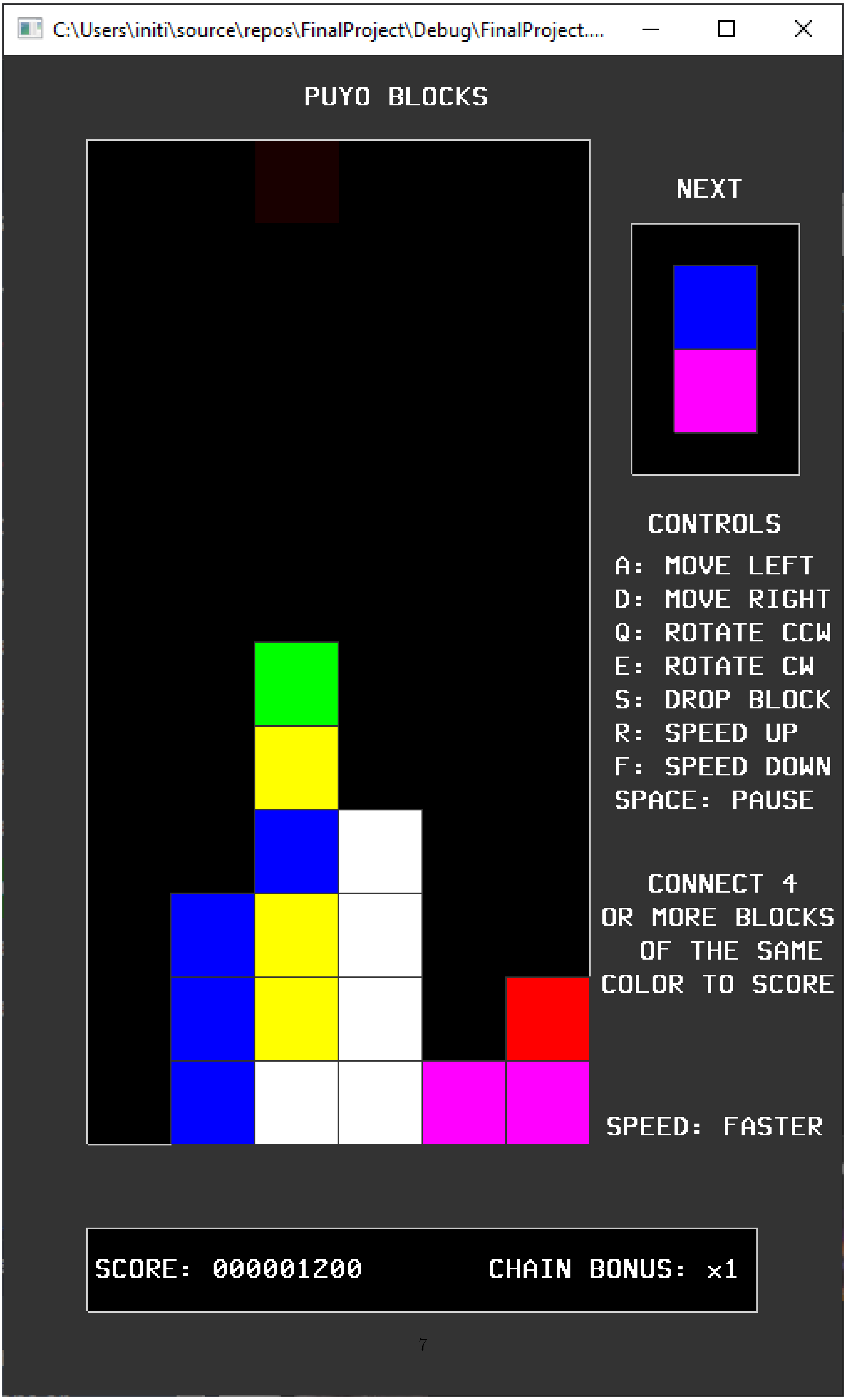
CONTROLS

A: MOVE LEFT
D: MOVE RIGHT
Q: ROTATE CCW
E: ROTATE CW
S: DROP BLOCK
R: SPEED UP
F: SPEED DOWN
SPACE: PAUSE

CONNECT 4
OR MORE BLOCKS
OF THE SAME
COLOR TO SCORE

SPEED: NORMAL

SCORE: 000000000 CHAIN BONUS: x1



C:\Users\initi\source\repos\FinalProject\Debug\FinalProject....

PUYO BLOCKS

NEXT

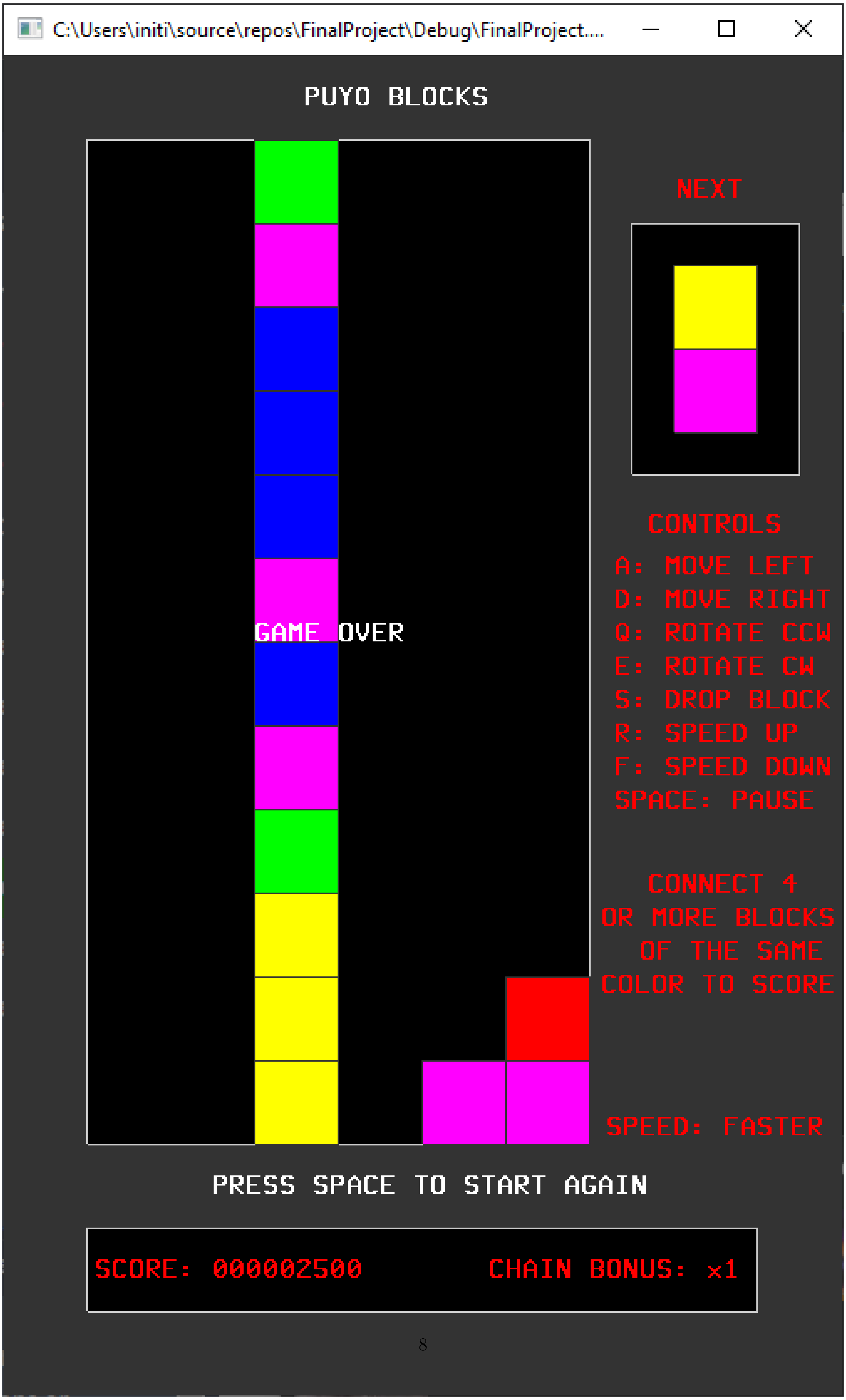
CONTROLS

A: MOVE LEFT
D: MOVE RIGHT
Q: ROTATE CCW
E: ROTATE CW
S: DROP BLOCK
R: SPEED UP
F: SPEED DOWN
SPACE: PAUSE

CONNECT 4
OR MORE BLOCKS
OF THE SAME
COLOR TO SCORE

SPEED: FASTER

SCORE: 000001200 CHAIN BONUS: x1


C:\Users\initi\source\repos\FinalProject\Debug\FinalProject....

PUYO BLOCKS

NEXT

CONTROLS
A: MOVE LEFT
D: MOVE RIGHT
Q: ROTATE CCW
E: ROTATE CW
S: DROP BLOCK
R: SPEED UP
F: SPEED DOWN
SPACE: PAUSE

CONNECT 4
OR MORE BLOCKS
OF THE SAME
COLOR TO SCORE

GAME OVER

SPEED: FASTER

PRESS SPACE TO START AGAIN

SCORE: 000002500 CHAIN BONUS: x1

Conclusions

Throughout building this project, I have learned a lot. I am much more comfortable with the matrix transformations on the 3D cubes and getting the results I expect. Setting up the world and designing the layout and the game was another learning milestone. I spent a good amount of time making sure the animation looked the way I wanted and I am very happy with the result. Overall, this project has been rewarding and it is definitely something I am proud of.

Going forward, there are many improvements that I can make. The following is where I would start:

- Sounds and music
- Textures
- More controls and options for the player

Resources

The primary sources for this project were the course materials, primarily the code from robot.cpp.

Here is where I found font bitmaps for numbers to add to the letters:

https://perso.esiee.fr/ perrotol/TheRedBook-1.0/chapter08.html

I used this resource for the quick_pow10() function which returns a power of 10 from a static array to cut down on computation time:

https://stackoverflow.com/questions/18581560/any-way-faster-than-pow-to-compute-an-integer-power-of-10-in-c