COSC345 Assignment 4
Developer documentation
&
Maintenance documentation

Ji Liu(2845570)
Bella Gao(3764478)
Andrew Wang(6686467)

Developer Documentation

Introduction

Our initial aim was to build a tile-matching puzzle game called Advanced Tetris which the player manipulates the traditional Tetris shape (L I O S Z T J) with up to four colours within each Tetris and eliminate blocks by matching more than 4 blocks of the same colour that connected.

Though it is based on traditional Tetris game that everyone has played in their life, with the left, right, accelerate and rotation movement function, our game using a different eliminating system and level passing system to judge if a player passes the level or fail.

Our level passing system is a block representing the player, and we have a chaser chasing the block. whenever player eliminating any blocks, these eliminated blocks will push the player represented block further away from the chaser, however, chaser will continue chasing the block no matter if eliminating has occurred. if the player represented block reach the pass point before chased by the chaser, player passes the level. if the player represented block chased by the chaser before the pass point, game over.

Resource Requirements

Communication

Our primary method of communication is through an app called Wechat, both mobile-based and desktop based. it has features for voice calls/video calls which could support better communication for special needs.
We also have a fixed time team meeting every Friday at 11 am-12 pm for team members to meet face to face and discuss ideas etc.

Compatibility

Minimum compatibility requirements are to run on macOS.

Version Control

We just write our version and combine all into one before and assignment. Every code writer is responsible for writing the comments for their corresponding code and explaining the corresponding code to another group member if required.

Team

Ji Liu

Programmer, organiser

good at Programming, algorithm and testing.
needs to improve The knowledge of C++ and third party library.

Bella Gao

Programmer, editor

good at: code-based problem solving, main language - Java. needs improve: limited knowledge in C and C++.

Andrew Wang

good at Programing and testing.
needs improve: Knowledge of C++ and python.

Risk Analysis

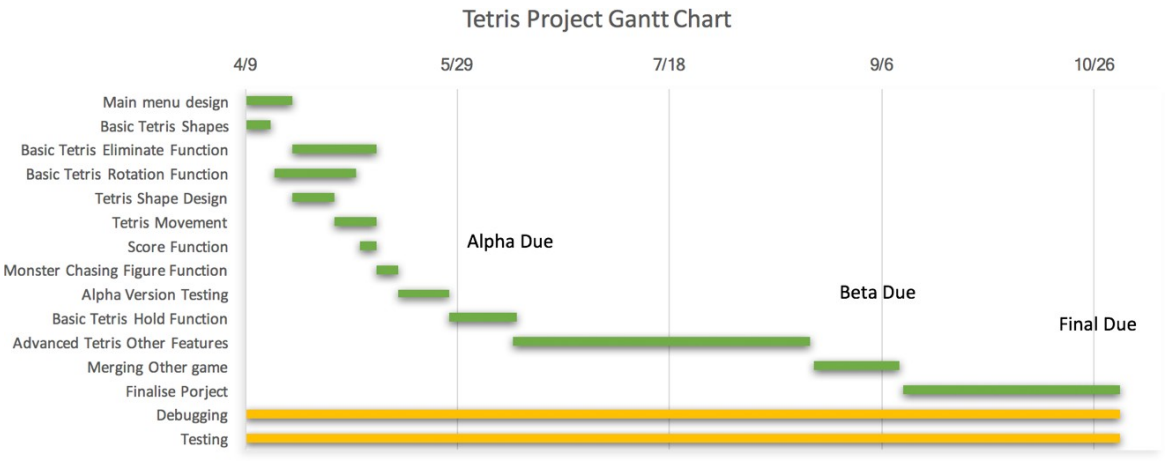
Risk: None of our group members has C++ programming experience, also the use of third party libraries are a big challenge for all of us.
Solution: We have found several tutorials including C++ and SFML, the third-party library which we are using for this project.

Risk: Project management including planning, scheduling and for this one-year-long project.
Solution: As we are all full-time student, the schedule and plan are flexible, so we can change these if necessary.

Risk: The update of the third party, there may be big version changes and/or updates.
Solution: We will examine the changes and updates of the third party library. If the changes are minor and don't have an effect on the project, we just ignore the update to ensure the stability of the project. If the changes are big, we will examine if the new features let the project more robust or easier to implement, we will deploy the new version of the third party library.

Risk: Medical emergencies - if a team member sick and no able to work, other team members could have a problem to understand the code or have double amount of work to do.
Solution: If any team member cannot work for any reason, they should inform the rest of the team as soon as possible. and the rest of the team can split the work remain so that no one has to work double. also due to our version control requirement, any code uploaded will need proper comments, these will help others to catch up on his code.

Project Scheduling



Standard Layout

We have decided to use K&R style in our code, since this is the default layout for the C++ programming language. Here is an example of K&R style (Source: Wikipedia)

```
int main(int argc, char *argv[])
{
        ...
        while (x == y) {
                something();
                somethingelse();

                if (some_error)
                        do_correct();
                else
                        continue_as_usual();
        }

        finalthing();
        ...
}
```

How our Program Will Differ

The similarity between our game and traditional Tetris:

It is based on the traditional Tetris game that everyone has played in their life. we also have 7 different types of Tetris, L O T S Z I J Tetris shapes with the left movement, right movement, acceleration and rotation for the blocks.

Differences between our game and the traditional tetras:

A different eliminating system:

Compare to the traditional Tetris game, we do not eliminate a line when a line is filled. we only eliminate the blocks that have the same colour and there are more than 4 blocks connected.

A different level passing system:

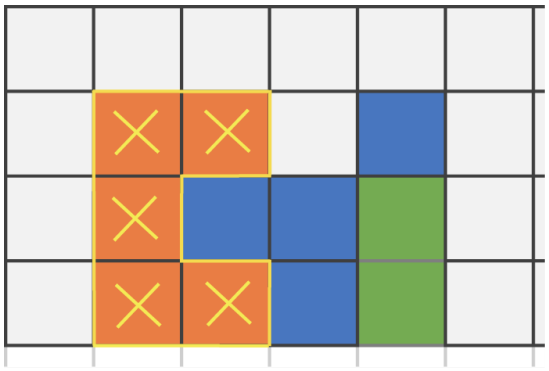
Our level passing system is a block representing the player, and we have a chaser chasing the block. Whenever player eliminating any blocks, these

eliminated blocks will push the player represented block further away from the chaser, however, chaser will continue chasing the block no matter if eliminating has occurred. if the player represented block reach the pass point before chased by the chaser, player passes the level. if the player represented block chased by the chaser before the pass point, game over.

A different gravity system:
unlike the traditional Tetris, if a line eliminates, the rest as a whole go down by 1 unit. In our game, whenever eliminating happens. any blocks above the eliminated blocks will go down and full fills any empty position so that there will be no gaps from top-down.

Current Features

In traditional Tetris games, players will be able to rotate the Tetromino, cancel a line and get scores for cancelling lines. But in Advanced Tetris, we are changing the game mechanism. Players now need to “give power” to a figure(currently represented a blue rectangle) to prevent the figure from getting eaten by the monster behind(currently represented a red rectangle). Without “power”, the figure can’t move forward and will eventually get caught by the monster(GameOver). “Power” can be gained by eliminating Tetrominos. The elimination of blocks in Advanced Tetris differs from the traditional Tetris games. Each Tetromino is now made up of four blocks(rectangles) with different colours. When a group of four or more blocks with the same colour are connected, this group of blocks will be cancelled. The figure will gain some power based on the size of blocks that’s been eliminated each time. Of course, the more blocks eliminated means more power the figure can get.



Blocks getting elminated

Improvement in new release

For the 1.0 version. We implemented some new features based on our future feature list in beta.

1. We now have an icon for our app
2. We now have about box for our app
3. New function - space for one-key dropping

Issues and Shortfalls

Instability:

As stated in the testing section, our app occasionally would crash and flashback during gaming due to lack of experience with testing in Xcode, the late timing when the problem showed up, and poor version control.

Audio issues:

Due to copyright issues, we decide to not add audio at this stage.

Space function issues:

As each member is only in charged of a few sections of the game, when we implemented the space function, it disturbs the behaviour of elimination of Tetromino. So we decided to play a trick which gets the distance from the Tetromino to bottom the highest Tetromino(if not then the bottom line), and then changed the y coordinate of that tetromino to that distance -1.

With this way, tetromino will use the same gravity system and elimination system.

There are of course drawback of this trick, that is when pressing space at -1 distance, the Tetromino should keep dropping to a lower level but instead, it will stay in the -1 level for another second.

Testing

While basic tests have been running by every member of the team, we have failed to debug an occasion where very occasionally our app would crash and flashback during gaming. This is due primarily to our lack of experience with testing in Xcode, the late timing when the problem showed up, and poor version control.

We first thought the bug was in new space function, after testing without new space function it still crashes from time to time. The bug could be existing from any time during the coding in the past 3 versions. We should do better testing and version control before releasing any new version instead of only fully testing before the final version.

we are aware of our current version is not substantial enough to deem the release fully ‘stable’. Poor version control has cost us a great price. Will not have the same issue happened to us again in the future.

Looking Forward

Our goals now would be to stabilize the game so our users do not have to be through a crashing experience. And possibly more functions if we get enough user feedback.

Maintenance Documentation

Code Explanation

void showWelcome(sf::Font font)

This function is to show the welcome screen.

Parameters:

font - the font of the text on welcome screen

void showPause(sf::Font font)

This function is to show the pause screen.

Parameters:

font - the font of the text on pause screen

void showOver(sf::Font font)

This function is to show the game over screen.

Parameters:

font - the font of the text on game over screen

bool checkOver()

check whether current blocks are exceeding playfield which means game over.

Returns:

true if game is not over

false if game should be over now

bool checkBlocksPos()

check for boundaries and collisions with other blocks.

Returns:

1 if no collisions or out of bounds

0 if collisions or out of bounds

void rotateBlock()

apply the rotation movements to the blocks

void fullLine()

Implements “gravity” into the game. It will go through all 200 grids to check any empty blocks blow existing blocks. When an empty block is found under an

existing block, the function will push the existing block down to fill the empty block. After the search and fill process is complete, there will be no empty block below any existing blocks.

int adjacentCount(int nValue, int nRow, int nCol, set<int> &sLst, bool bElm = false)

Check all adjacent blocks based on the current block's colour, row value, column value. When the block with the same colour is found, add it to the sLst. When the bElm is assigned true, eliminate all blocks within the list.

Parameters:

nValue - the colour value of the block that's been passed from adjacentCount2

nRow - the row value of current block inside the grid nCol - the column value of current block inside the grid

sLst - the list to contain all blocks that needs to be eliminated

bElm - the boolean that checks whether this block should be eliminated

Returns:

The number of blocks getting eliminated

int adjacentCount2(int nRow, int nCol, bool bElm = false)

Check the status of the current block and pass the values of the current block to adjacentCount.

Parameters:

nRow - the row value of current block inside the grid

nCol - the column value of current block inside the grid

bElm - the boolean that checks whether this block should be eliminated

Returns:

Pass the value to adjacentCount and get the The number of blocks getting eliminated

void checkElimination()

Go through all 200 grids, call the adjacentCount2 function to check the status of the block and its adjacent blocks. When the return value from adjacentCount reaches 4, eliminate the blocks, pass the value to scores, and calls the fullLine function to tidy up the grid.

sf setText(std:string info, int x, int y)

Set text with font, content, size, colour and position.

Parameters:

info - the target string

x - x position of the string

y - y position of the string

Returns:

Text

void horizMove(int distanceX)

left and right movement of blocks

Parameters:

distanceX - the amount to move.

void gameplay()

The main body of the game

int main()

The main function of the project. It calls the main body of the game and reset the game when last game is over

Changing minimum number of eliminations

In Line 389. If you want to change the number of blocks required for elimination. Let's say, you want elimination happens when X blocks are connected. Change both number 3 marked red to X-1.

```
void checkElimination()
{
  int nAdNum;
  for (int i = M - 1; i > 0; i--)
  {
     nAdNum = 0;
     for (int j = 0; j < N; j++)
     {
        nAdNum = adjacentCount2(i, j);
        if (nAdNum > 3)
```

```
        {
            tscore += nAdNum;
            fullLine();
            adjacentCount2(i, j, true);
            fullLine();
        }
    }
    if (nAdNum > 3)
    {
        i=M-1;
    }
  }
}
```

Changing text colours

Modify the array in Line 38 – 44. The first colour is the default colour of this game.

```
sf::Color fontColours[] = {
  sf::Color::Red,
  sf::Color::Blue,
  sf::Color::Yellow,
  sf::Color::Green,
  sf::Color::Magenta
};
```

If the length of the array is changed. In Line 643, change 5 to the new length of the array as well.

```
int idx = level % 5;
```

Changing font

Put font file into fonts folder and modify the file name in Line 452.

```
if (!font.loadFromFile("fonts/sansation.ttf"))
```

Changing text size

In Line 88, change 24 to the new font size.

```
text.setCharacterSize(24);
```

Changing enemy and players speed

To change the enemy's chasing speed, modify Line 629. The variable level is the current level of the game. The last 0 is the vertical speed, please do not modify it.

```
enemy.move(0.16 + level * 0.04, 0);
```

To change player's speed, modify Line 704. Variable tscore is the number of blocks eliminated just now. The last 0 is the vertical speed, please do not modify it.

```
player.move(tscore * tscore / 2, 0);
```