

# COSC345 Assignment 3

## Developer documentation Beta

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

# 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 colors with in each Tetris and eliminate blocks by matching more than 4 blocks of the same color that connected to each other.

Though it is based on traditional Tetris game that basically 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 pass the level or fail.

Our level passing system is that a block representing the player, and we have a chaser chasing the block. every time when 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 pass 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 a app called Wechat, both mobile based and desktop based. it has features for voice calls/video calls which could support a better communication for special needs.

we also have a fixed time team meeting every Friday 11am-12pm for team members to meet face to face and discuss ideas etc.

## **Compatibility**

Minimum compatibility requirements are to run on MacOS.

## **Version Control**

basically we just write our own 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 improve: The knowledges 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 member has C++ programming experience, also the use of third party libraries are also a big challenge for all of us.

**Solution:** We have find 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 is flexible, so we can change these if necessary.

**Risk:** The update of third party, there may be big version changes and/or updates.

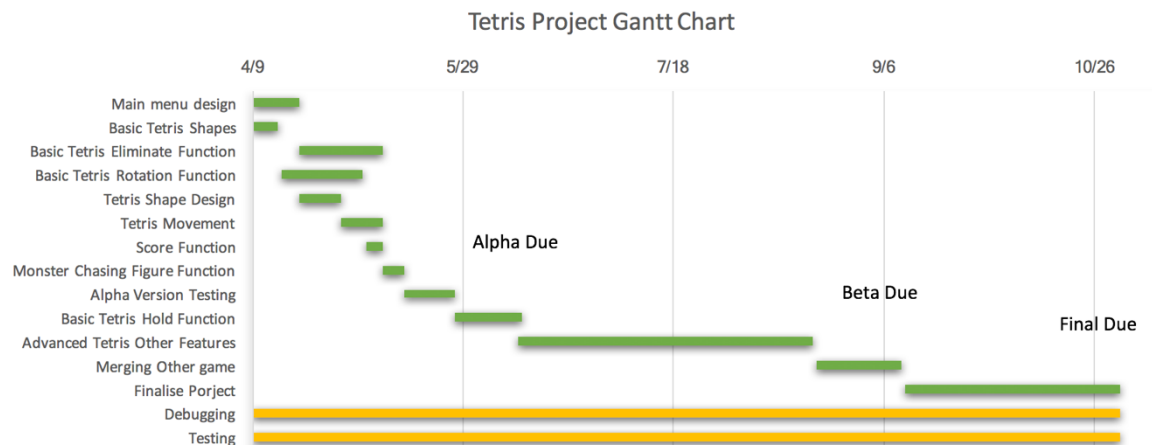
**Solution:** We will examine the changes and updates of the third party library. If the changes are minor and don't have effect on the project, we just ignore the update to insure the stability of the project. If the changes are big, we will examine if the new features let the project more robust or 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 member could have 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 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

## **Similarity between our game and traditional Tetris:**

It is based on traditional Tetris game that basically 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 full filled.

We only eliminate the blocks that has the same colour and are connected with more than 4 blocks.

### **A different level passing system:**

Our level passing system is that a block representing the player, and we have a chaser chasing the block.

Every time when 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 pass 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**

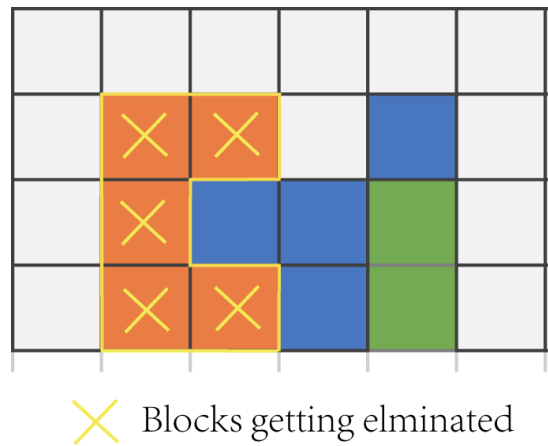
In traditional Tetris games, players will be able to rotate the Tetromino, cancel a line and get scores for cancelling lines.

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 by four blocks(rectangles) with different colours. When a group of four or more blocks with the same colour are connected, this group of block will be cancelled.

The figure will gain some power to move forward based on the size of block that’s been eliminated each time. Of course, with more blocks eliminated, the figure can get more power to get away from the enemy.



## Features and improvement in Beta

For the beta version, we fixed the most of the bugs in alpha. We also implemented some new features based on our future feature list in alpha.

1. First block is no longer a non-sense shape, it is now one of the seven shapes.
2. Team has come to an agreement to abandon the plan of generating Tetris with eight random colored tiles as it creates too much difficulties for our game. it will be very hard for our player to eliminate Tetris and pass the level. The color layout for each tetris is now fixed.
3. We added a method to make sure between current block and followed block will not be the same at all times to improve the enjoyment of the game.
4. The game will finish and fail when the play field is full.
5. Elimination animation is added when blocks are eliminated which we believe would help player to gain a better understanding of our game rules easier than without animation.
6. Chasing player and enemy have a new looking feature which can well possibly add more interest to our game.
7. The color of the text will change when player levels up which gives a better hint to our player about what is happening in the game.



8. When player pass the game and enter the next level, the chasing speed of enemy will increase.

## Issues and Shortfalls

### **Platform issues:**

Although we had a solution to adjust the birthplace from top left corner to middle of the screen, we found that solution does not work the same way on different platforms, which had never happened before. Because previously different platforms have never produce different testing result before beta version, we never thought it would be an issue to consider.

Solution was originally produced and tested on windows visual studio 2017 perfectly with no unwanted feature shows on the screen at all, but later our other team members found that when running the code on macOS and Linux, the top left column sometimes produces unwanted shapes and sometimes only generate one unwanted shapes when game starts. It also behaves different between macOS and Linux.

Due to limited time and limited devices we decided to leave the birth place at top left for now as it does not make the game less playable.

### **Another platform issues:**

The way our team work is that each member working on their own and integrate our code together before assignment's due date.

It leaded to some small issues such as repeatedly type setting code every time when one of the team members updated their code but was never a huge problem before beta version.

But this time due to the platform behavior inconsistency(between windows, mac Os and Linux), it causing us too much energy to restore code and debug any conflicts.

Source Tree as a good tool should be considered at the beginning, it has many functions to help avoid code conflicts and can easily restore code even though it might takes extra time to train the team about how to use it.

## Looking Forward

Our goals now would be to implement some of the functions that we did not get a chance to finish off.

1. To further improve the animation for elimination.
2. Change the background color of the playfield.
3. Get the birth place issue fixed.
4. If possible, add audio to the game.

Although we would like to add more features and functions to the game, we may also worry about new features and functions that might take away the simplicity of the game, after all a game should be simple and easy to play and also full of fun.

## 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()**

This function implement “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.

**void initPlayerPos()**

Set the initial position of player and the enemy.

**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.

# **Usage**

Make sure SFML package is installed base on the tutorial from <https://www.sfml-dev.org/tutorials/2.5/start-osx.php#installing-sfml>

To run the program in terminal, run terminal then navigate to your download position and execute following 3 commands:

```
g++ -c main.cpp -ISFML/include
```

```
g++ main.o -o game -LSFML/lib -lsfml-graphics -lsfml-window -lsfml-system
```

```
export LD_LIBRARY_PATH=SFML/lib && ./game
```