

# Temtris

Curt Lynch and Hunter Johnson

## Abstract

Temtris is a Tetris clone we are developing as part of our CSCI 352 class.

## 1. Introduction

Temtris is a WPF clone of Tetris with our own spin added in. Tetris, is a game where 4 block section shapes fall from top down and you try to fit rows of these blocks together at an ever increasing pace until the screen inevitably fills causing game over. A score is computed based on how efficiently you clear the board as the game progresses, with more points being awarded for streaks of row clears and tetris row clears (4 simultaneous row clears). Our target audience is anyone looking to pass time with a casual game of Tetris. We are making Temtris as a duo as part of our CSCI 352 class. That being the case, we still want Temtris to be a fun and interesting game for anyone looking to play.

### 1.1. Background

A Tetrimino is one of the 7 classic Tetris shapes; these include the O, I, T, L, J, S, and Z shapes. Each Tetrimino is made up of four squares called minos. The matrix is the space where the Tetris game is played. A 'line clear' happens when a horizontal row is completely filled, causing it to be removed from the matrix. Lock-down is when a Tetrimino is locked into its current position and is no longer controllable.

While attempting to brainstorm ideas for our CSCI 352 class, the idea of a Tetris clone come up (with the awful misspelling Temtris). Eventually we settled on the idea and the name stuck.

### 1.2. Impacts

Tetris is a classic game that has had a major impact on the gaming community since its debut. Continuing to iterate upon Tetris and bring it back into the community repeatedly will only deepen its legacy. Temtris is a great way to do this while improving our skills and preparing us for working in teams and on larger projects in the future.

### 1.3. Challenges

This is the first moderately sized project either of us has attempted. This is also our first real foray into WPF and C# so there are some growing pains there as well. As for the project itself, I suspect getting the movement and rotations to feel like a proper Tetris game will take much more fine tuning than we have needed to use in the past. Rotating the pieces will likely also pose some sort of challenge.

## 2. Scope

Our initial goal is to implement the following typical parts of a Tetris game:

- Implement a main menu.
- Have controllable tetriminos (2x1 shapes initially) that fall into place in the matrix.
- Have rows that disappear when they fill up, i.e., the line clears.
- Keep score or track how well the player does in some way.
- Display the next piece to spawn.

For stretch goals, we will to add the following:

- Implement all 7 tetrimino variants.
- Implement a multiplayer mode. Having at least two clients with a competitive game mode over LAN or just sharing a keyboard.
- Different power-ups may be earned by removing a certain amount of one color from the board. Perhaps board clear (of that color) or maybe have the next five pieces all be 'L' shapes.
- Play background music and have sound effects for line clears, player actions, etc.
- Have random shapes form and fall into the matrix ( as opposed to the standard Tetris shapes.

## 2.1. Requirements

These requirements are things we feel make Tetris what it is. Without them the Tetris would ultimately not be a Tetris clone.

### 2.1.1. Functional.

- Users can control Tetriminos (rotate, move left or right).
- Tetriminos are created and fall from the top of the matrix.
- Row clears should happen as the rows are filled.
- Main menu allows the user to start the game.

### 2.1.2. Non-Functional.

- Main menu should display controls needed to play the game.
- User interface should be attractive and inviting.

## 2.2. Use Cases

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Start game	Player	Easy	1
2	Move tetriminos left or right	Player	Easy	1
3	Lower tetriminos into position	Player	Med	2
4	Rotate tetriminos	Player	Hard	3

TABLE 1. USE CASE TABLE

Use Case Number: 1

Use Case Name: Start game

Description: A player wishes to pass some time by playing Tetris. They start the game and click on the "Start Game" button from the main menu.

- 1) The Player launches the application.
- 2) The Player left-clicks on "Start Game" button.
- 3) The game screen appears and the game begins.

Termination Outcome: A game of Tetris is now running.

Alternative: A previous game of Tetris has just ended.

- 1) Player left-clicks on "Play Again" button.
- 2) The game screen appears and the game begins..

Termination Outcome: A game of Tetris is now running.

Use Case Number: 2

Use Case Name: Move tetriminos left or right

Description: A game of Tetris has been started and the player wishes to move the falling piece to the left or right. The player presses the left or right arrow on the keyboard and the active tetrimino moves accordingly.

- 1) The player presses the left arrow key on the keyboard.
- 2) The active tetrimino shifts one position to the left.
- 3) Player presses the right arrow key on the keyboard.
- 4) The active tetrimino shifts one block to the right.

Termination Outcome: The player has control over the left and right motion of the active tetrimino.

Use Case Number: 3

Use Case Name: Lower tetriminos into position

Description: The player is comfortable with the current trajectory of the active tetrimino and wishes to lower it into place. The player presses the down arrow key on the keyboard and the active tetrimino moves down into place.

- 1) The player presses the down arrow key on the keyboard.
- 2) The active tetrimino begins to fall faster.
- 3) The active tetrimino stops falling when it reaches inactive tetriminos and is set into place.

Termination Outcome: The player can speed up the falling of the active tetrimino.

Use Case Number: 4

Use Case Name: Rotate tetriminos

Description: The player wishes to rotate a tetrimino and presses the up arrow key on the keyboard. The active tetrimino rotates ninety degrees allowing the player to better control how it falls.

- 1) The player presses the up arrow key on the keyboard.
- 2) The active tetrimino rotates 90 degrees clockwise.
- 3) The player presses the up arrow key on the keyboard.
- 4) The active tetrimino rotates an additional 90 degrees clockwise.

Termination Outcome: The player has control over the rotation of the active tetrimino.

## 2.3. Interface Mockups

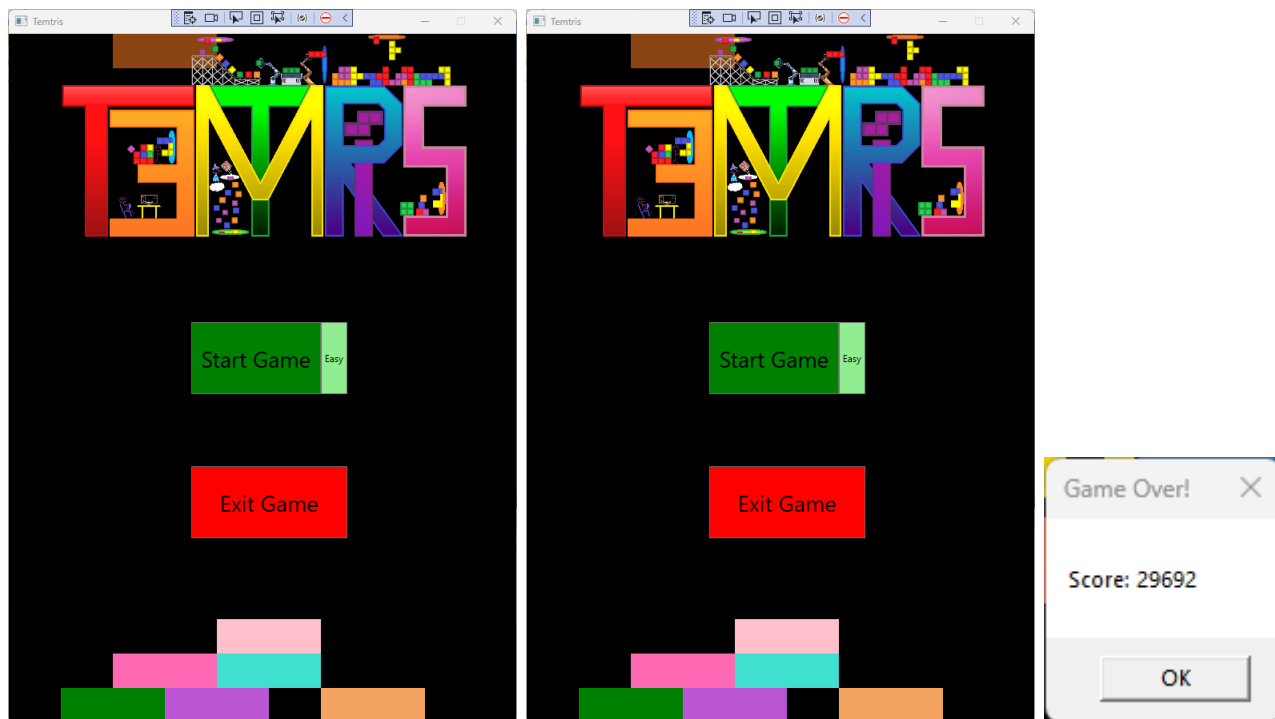


Figure 1. Temtris at the main menu(left), during a game(center), and the game over pop-up(right).

Todo: Tie interface mockups back into use cases and add game ending screen. Figure 1 shows various screenshots from Temtris as the user plays the game.

## 3. Project Timeline

As with most projects following the waterfall development cycle we wanted to solidify most of the design as early possible. This unfortunately left us with little time to actually implement the project and work out any rough edges in our design. If we had the chance to do this again we would've spent more time early learning WPF before attempting to outline a project using it.

## 4. Project Structure

The UI exists within the auto generated MainWindow class, which calls upon an instantiation of the TemtrisGame class to handle all of the game logic and keyboard input.

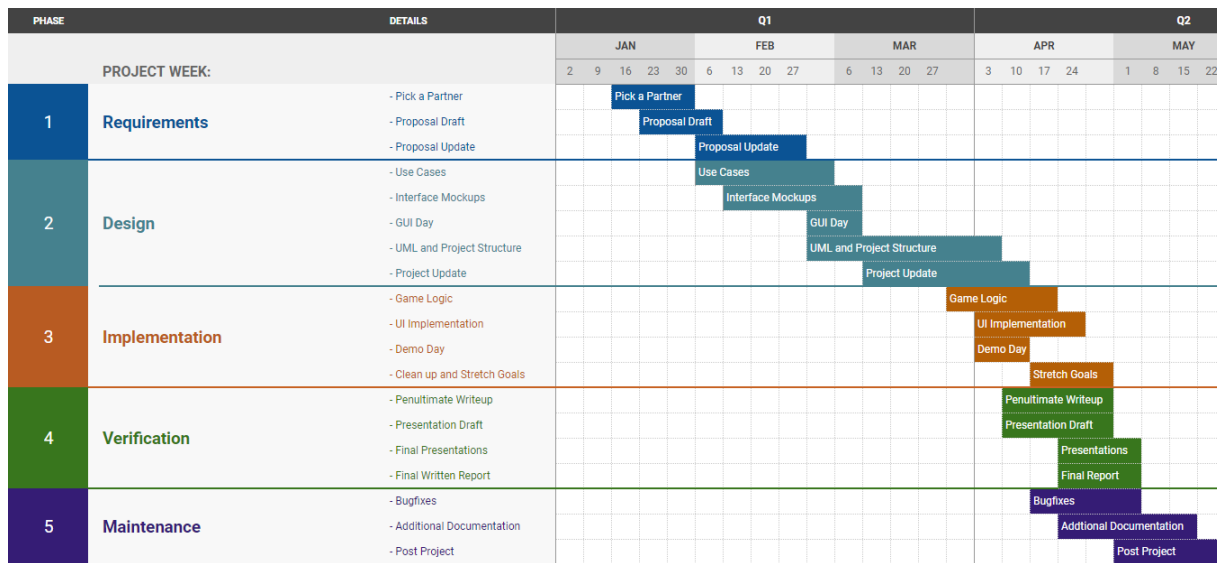


Figure 2. Project Timeline

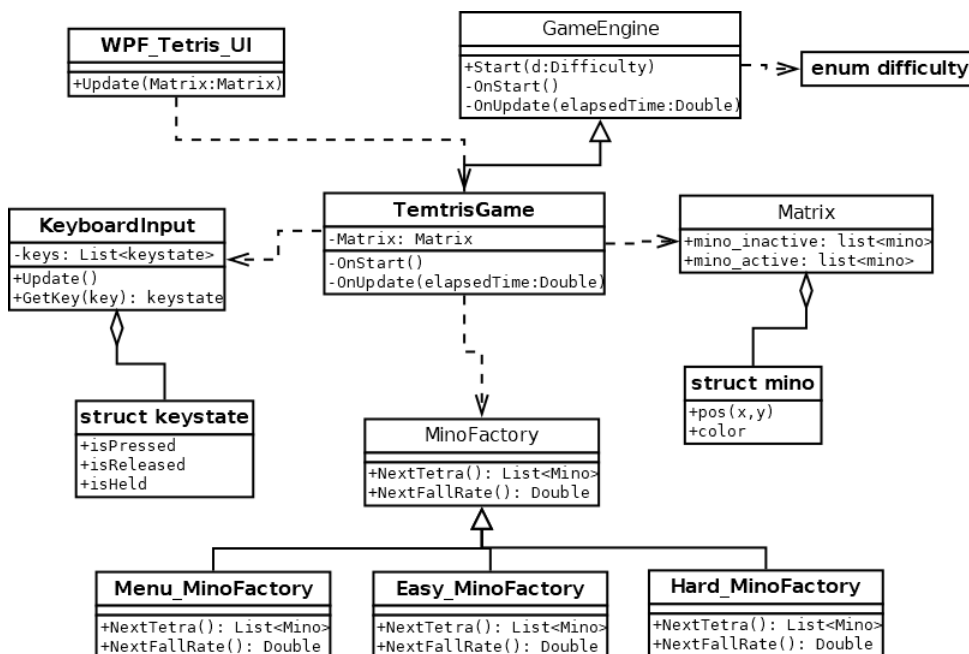


Figure 3. UML Diagram for Temtris

## 4.1. UML Outline

Figure 3 shows the breakdown of the class hierarchy for Temtris.

**MainWindow:** The auto generated class given when starting a WPF project in Visual Studio. This is the home of our UI. Utilizes TemtrisGame to get the layout and position of all Tetraminos.

**GameEngine:** This handles some of the more mundane non-Tetris aspects of the game engine.

**TemtrisGame:** The main Temtris class. Handles all of the game logic and keyboard input.

**Matrix:** A convenient data carrier. Stores all of the data for a given game data for a game of Temtris.

**KeyboardInput:** This keeps track of keyboard input and provides easy access to key state.

**MinoFactory:** Creates Tetraminos for the Tetris game, and provides updated fallRates. Each difficulty has it's own child class.

## 4.2. Design Patterns Used

TODO: clean up and explain a bit more. There's a template pattern in there through GameEngine/TemtrisGame and an abstract factory pattern through MinoFactory.

## 5. Results

This section will start out a little vague, but it should grow as your project evolves. With each deliverable you hand in, give me a final summary of where your project stands. By the end, this should be a reflective section discussing how many of your original goals you managed to attain/how many desired use cases you implemented/how many extra features you added.

### 5.1. Future Work

Where are you going next with your project? For early deliverables, what are your next steps? (HINT: you will typically want to look back at your timeline and evaluate: did you meet your expected goals? Are you ahead of schedule? Did you decide to shift gears and implement a new feature?) By the end, what do you plan on doing with this project? Will you try to sell it? Set it on fire? Link to it on your resume and forget it exists?

## References

- [1] H. Kopka and P. W. Daly, *A Guide to L<sup>A</sup>T<sub>E</sub>X*, 3rd ed. Harlow, England: Addison-Wesley, 1999.