Temtris

Curt Lynch and Hunter Johnson

Abstract

Temtris is a Tetris-like we are developing as part of our CSCI352 class.

1. Introduction

Temtris is a wpf application that we aim to be Tetris clone. 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. Temtris will include this regular mode with it's locked shape patterns, as well as a mode where the blocks are randomly generated within certain parameters. Our target audience would probably be the select few who call themselves "gamers", as well as those who are looking for a casual way to pass the time. 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.

Also, consider going into your personal connection with this project – why did you decide to do it? Whilst attempting to come up with ideas, the idea of remaking Tetris came up, and to clarify this is coming from Hunter, when I heard Tetris my brain went full monkey, "ooh ooh ahh ahh Tetris funny, seems more interesting than anything else we have thought up" and that's how we landed on Temtris.

1.2. Impacts

Temtris is a great way to improve our skills and better prepare us for working on more impactful projects in the future. Impact wise, towards the greater scale of coding, it has none, but for personal entertainment as well as great c# practice it will be quite impact full.

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.

2. Scope

Our initial goal is to implement the following typical parts of a Tetris-like: Implement a main menu with access to settings and game modes. Have controllable tetriminos that fall into place in the matrix. Implement all 7 tetrimino variants. 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. Play background music and have sound effects for line clears, player actions, etc. For stretch goals, we would like to add the following: Implement multiplayer? Have 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.

2.1. Requirements

As part of fleshing out the scope of your requirements, you'll also need to keep in mind both your functional and non-functional requirements. These should be listed, and explained in detail as necessary. Use this area to explain how you gathered these requirements.

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Add item to cart	Shopper	Med	1
2	Checkout	Shopper	Med	1

TABLE 1. SAMPLE USE CASE TABLE

2.1.1. Functional.

- User needs to have a private shopping cart – this cannot be shared between users, and needs to maintain state across subsequent visits to the site
- Users need to have website accounts – this will help track recent purchases, keep shopping cart records, etc.
- You'll need more than 2 of these...

2.1.2. Non-Functional.

- Security – user credentials must be encrypted on disk, users should be able to reset their passwords if forgotten
- you'll typically have fewer non-functional than functional requirements

2.2. Use Cases

This subsection is arguably part of how you define your project scope (why it is in the Scope section...). In a traditional Waterfall approach, as part of your requirements gathering phase (what does the product actually *need* to do?), you will typically sit down with a user to develop use cases.

You should have a table listing all use cases discussed in the document, the ID is just the order it is listed in, the name should be indicative of what should happen, the primary actor is typically most important in an application where you may have different levels of users (think admin vs normal user), complexity is a best-guess on your part as to how hard it should be. A lower number in priority indicates that it needs to happen sooner rather than later. A sample table, or Use Case Index can be seen in Table 1.

Use Case Number: 1
Use Case Name: Add item to cart
Description: A shopper on our site has identified an item they wish to buy. They will click on a "Add to Cart" button. This will kick off a process to add one instance of the item to their cart.

You will then go on to (minimally) discuss a basic flow for the process:

1) User navigates to page listing desired item
2) User left-clicks on "Add to Cart" button.
3) User cart is updated to reflect the new item, this also updates the current total.

Termination Outcome: The user now has a single instance of the item in their cart.

You may need to also add in any alternative flows:
Alternative: Item already exists in the cart

1) User navigates to page listing desired item
2) User left-clicks on "Add to Cart" button.
3) User cart is updated to reflect the new item, showing that one more instance of the existing item has been added. This also updates the current total.

Termination Outcome: The user now has multiple instances of the item in their cart.

You will often also need to include pictures or diagrams. It is quite common to see use-case diagrams in such write-ups. To properly reference an image, you will need to use the `figure` environment and will need to reference it in your text (via the `ref` command) (see Figure 1). NOTE: this is not a use case diagram, but a kitten.

After fully describing a use case, it is time to move on to the next use case:

Use Case Number: 2
Use Case Name: Checkout
Description: A shopper on our site has finished shopping. They will click on a "Checkout" button. This will kick off a process to calculate cart total, any taxes, shipping rates, and collect payment from the shopper.

You will then need to continue to flesh out all use cases you have identified for your project.

Figure 1. First picture, this is a kitten, not a use case diagram

2.3. Interface Mockups

At first, this will largely be completely made up, as you get further along in your project, and closer to a final product, this will typically become simple screenshots of your running application.

In this subsection, you will be showing what the screen should look like as the user moves through various use cases (make sure to tie the interface mockups back to the specific use cases they illustrate).

3. Project Timeline

Go back to your notes and look up a typical project development life cycle for the Waterfall approach. How will you follow this life cycle over the remainder of this semester? This will usually involve a chart showing your proposed timeline, with specific milestones plotted out. Make sure you have deliverable dates from the course schedule listed, with a plan to meet them (NOTE: these are generally optimistic deadlines).

4. Project Structure

At first, this will be a little empty (it will need to be filled in by the time you turn in your final report). This is your chance to discuss all of your design decisions (consider this the README's big brother).

4.1. UML Outline

Show the full structure of your program. Make sure to keep on updating this section as your project evolves (you often start out with one plan, but end up modifying things as you move along). As a note, while Dia fails miserably at generating pdfs (probably my fault), I have had much success with png files. Make sure to wrap your images in a `figure` environment, and to reference with the `ref` command. For example, see Figure 2.

4.2. Design Patterns Used

Make sure to actually use at least 2 design patterns from this class. This is not normally part of such documentation, but largely just specific to this class – I want to see you use the patterns!

Figure 2. Your figures should be in the *figure* environment, and have captions. Should also be of diagrams pertaining to your project, not random internet kittens

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 LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.