Tetrominoes

Application name:CS4540_Tetris

Hayden Henrie • Nhan Nguyen • Jaecee Naylor • Courtney Crum

u1057630 • u1219393 • u1183887 • u0829734

Abstract

Overall we feel like our code accomplished the exact thing we set out to make. We managed to get our Tetris game working in the single-player and on top of that we managed to make multiplayer work as well. We also added in two different chat systems, one of which is a global chat while the other is a chat room with two people. The chat room with two people is designed to match players in order to participate in a multiplayer game. Overall we managed to meet the required functionality we outlined in the original design document and after talking to Jim in class we also added in the two-player Tetris game concept. Some of our other cool features include the Player stats list, as well as a high score list that keeps track of every player's highest scores. We have also added fun alerts for losing, aspects to the game, and email confirmation.

These features are useful to a real user because it allows them to accomplish the task of playing Tetrominoes by themselves or competitively synchronously with another user, being mindful all of our users should be players of Tetrominoes. On top of this, real users can track where they are on the high scoreboard and stats board. This adds a fun component to our application for real users. Users can be authenticated using email confirmation. Real users can also chat between other real users discussing the game and at times a little "trash talk" in competitive gaming is fun!

URL

http://ec2-54-86-215-51.compute-1.amazonaws.com/ (Jaecee's URL)
https://ec2-18-215-231-153.compute-1.amazonaws.com/ (Courtney's URL)

Please keep in mind seeded users are gamegorl@tetrominoes.com, gameboi@tetrominoes.com, and gamealien@tetrominoes.com. All seeded passwords are "password" (no quotations). High score and player statistic information is seeded for gamboi and gamealien. If you were to register on deployed websites, email confirmation code will cause an error because AWS does not allow spoofing. This means we cannot send emails for confirmation in deployment mode. We wanted email confirmation to be shown, so code was left in and you can register with email verification locally. This means registering on deployment does not have email verification. If you would like to test registration code with email confirmation please build the program locally. Before building the program locally comment out Areas → Identity → Pages → Account → Register.cshtml.cs line 101 ("await _signInManager.SignInAsync(user, isPersistent: false);"). Also uncomment out Areas → Identity → IdentityHostingStartup.cs lines 46-51 where commented out. With these changes, email confirmation works locally.

Secondly, it was not intended for someone to play themselves using multiplayer on AWS. This means if you are trying to play multiplayer on one computer, have one site being ran in one browser and another in incognito because these are separate sessions. This way the game will not think you are in the same session and send the same info. This can cause errors because of how signalR depends on IP. We do not want two users to have the same IP.

Introduction

Tetrominoes is a Tetris application. It was initially designed to support single player games of Tetris. For this we used JavaScript and Canvas rendering. During this time we implemented the beginning of the game of tetris including all of the blocks and orientations, piece randomization, and falling tetrominoes (blocks). We decided to use and array of arrays for the board and whenever a row was filed we would white out the board, let other blocks cascade down and increment the score by 100 points. During this time, we also began to focus on creating the database for both game data (i.e high scores, player statistics, etc.) and user authentication.

After discussing our project with Jim, we realized we needed to expand the scope of our project. This is when we began to work on implementing multiplayer as well as chat rooms for matching users. For this part of the project, we used signalR and more JavaScript. At first this was a bit of a challenge, but as we began to understand signalR it became much easier. After we had a working matchmaking and room selection, we added in sending two player data over signalR in order to create two player Tetris. Throughout this process, we were also working displaying data stored in the game data database in the high scores and player statistics view. We felt it would be fun for users to be able to comment on one another statistics, so we added this functionality as well. On top of this, we implemented email confirmation that would authenticate users when logging in. We did not feel it would make sense for users of our application to have roles beyond a registered, authenticated user. This is because we consider all users who are registered and authenticated with our application to be “players.” It would not be helpful for roles to exist if there would only be one role, “players.” Because of this, we did the authorization of our app by letting only logged in and authenticated users visit the Dual Player, High Scores, and Player Statistics web pages. Everyone else is free to access any part of the application. We also have significant database seeding to show initial data that we can work with. All in all, our application is designed to best support the idea of an online Tetris application.

Feature Table

*Any lines unaccounted for were for general bug fixing and prebuilt implementations.

Feature Name	Scope	Primary Programmer	Time spent	File/Function	LoC
Single Player Tetris code	UI	Nhan	24 hours	Tetris.js	410
Multiplayer Tetris code	UI	Hayden	14 hours	Tetris.js	230
Global Chatroom	UI	Courtney	8 hours	Chat.js chatroomhub.cs	104
Private chat room	UI	Courtney	8 hours	Chatroom.js chatroomhub.cs	60
Player statistics	UI and DB	Nhan	4 hours	HomeController.cs tetris.js	258
Comments on player statistics	DB and UI	Jaecee	8 hours	stats.html	127
Email confirmations	Other	Jaecee	4 hours	Emailsender.cs	200
Database seeding	Back-end	Jaecee	6 hours	DbInitializer.cs	259
Database creation	DB	Courtney and Others	8 hours	All files in Model folder	85
Bug fixing and combining each other work	Other	Hayden	14 hours	Pretty much every file	N/A
Views	UI	Jaecee and Others	9 hours	All .cshtml files	468
Documentation	Other	Jaecee	3 hours	Every file	N/A

Individual Contribution

This does not include the line counts of prebuilt/scaffolding code as well as inclusion of external code such as signalR. And all of these numbers were pulled from github.

Team Member	Time Spent on Project	Lines of Code Committed
Hayden	28 hours	844
Nhan	28 hours	912
Jaecee	33 hours	941
Courtney	29 hours	817

Summary

I believe our team performed in the Superior to Good range and we really all worked very well together. We made plans with the team by meeting up at least twice a week and were constantly in contact with each other over a program called Discord. This allowed each of us to program our own content at our own pace while also working towards the same final goal. In the beginning we were constantly hitting roadblocks and whenever that happened there was always another teammate available to help and explain or help with the problem. This helped us much more smoothly program as a group and after a while kept the bug fixing and merging code down to a minimum.

Overall we feel like the multiplayer and chat rooms are the most important features that we implement as they provided the most functionality and usability of the program. We also felt that the LOT really helped us to create and grow this program because it gave us a way to test and develop similar programs with two very different programs. In the end, we feel like we have implemented everything we have learned in this class and feel like all our features were useful and interesting.