Battleship Bois:
Eric Naegle
Christopher Bordoy
Tom Nguyen

Final Project: Battleship

Web Software Architecture

06 December 2019

Abstract

We have re-created the classic table-top game “Battleship.” The game places the ships for you, and you play against an AI opponent. In the real-life version of the game, the user calls out the coordinates of the spot they want to hit, but in this game, the user just has to click on the grid. The user can see both grids in this version, but the AI’s ships are hidden from the user in the back end. The user is also able to use power ups rather than just firing one missile at a time, and their scores are shown on a high score page. Their victories also earn them money that they can spend on more power ups in later games, so there’s an incentive for the player to keep playing. The AI is intelligent, so the game is challenging and fun to play.

URL

At this time, our project is not deployed on AWS, so there is no URL for the game. The game can be played by building out and running the Visual Studio Solution that we turned in.

Introduction

Responsive Grid System

There are two grids, one for the AI and one for the player. These grids are highly responsive. When you hover the mouse over the enemy's grid, the mouse changes to a target. When a hit or miss occurs, the grid changes colors. When the game is restarted, the grids are wiped.

Animations

Along with a colorful grid system, there were also Html Canvas animations implemented into the game. When you shoot a spot on the enemy's grid, or when they shoot you, the grid doesn't just turn red for a hit and grey for a miss — there's also a little animation to go along with it. If there's a miss, a splash animation is shown. If it's a hit, an explosion animation is shown. This makes the game feel more interactive and exciting.

Intelligent AI

The AI can't see the player's ships. We could have implemented it where the AI peeks at the location of the player's ship and challenges them that way, but we figured that it would be more challenging and rewarding to implement a smart AI that is just as blind as the player is. So the AI shoots randomly, skipping one square at a time to search more efficiently, and when it hits a spot on the player's ship, it stops firing randomly and hunts down the rest of that ship in the nearby squares until the ship is sunk. This AI can

be beaten by a player that is trying hard, but if you're not careful the AI can beat you pretty easily, especially if you don't use power ups.

User Logins

A full login system was implemented into the game just like in the Learning Outcome Tracker. The reason for this is two-fold: first, the server can support multiple people playing at once, because each logged-in player has their own Game object running in the GameService for them; and the second reason for logins is so that you can keep track of your money and inventory, which is persistent between games. The user login page is also really professional. We went above and beyond on that page to leave a good first impression on the game. There were some stretch goals with the login page as well when it comes to the Remember me box generating an authentication token and the forgot password being implemented.

High Scores

We have your accuracy percentage displayed on the high scores page after you win, along with the date of your win. Everyone who logs in the game can see these high scores, not just you.

Power ups

We implemented a powerup called Cannon Barrage that fires in 10 random spaces at once. (You have $1000 to start when you create an account just so that you can use a bunch of powerups while you grade it.) We also added two more power ups to the store page, but they were stretch goals that we didn't have time to implement.

Money and Inventory

You need to have money to buy power ups. Money is given to you after every victory, and you can use the money on the store page to buy more power ups, and they are added to your inventory in the DB.

Audio

Cool background music is playing throughout the game. We know that not everyone wants to listen to it, so we also supplied a functioning mute button. One downside of this is that we couldn't figure out how to

Dragging-and-Dropping

After we got the game to place the ships randomly for you, we thought it would be cool for the player to be able to drag and drop their own ship images on to the grid. This proved to be overly-ambitious for us, and it wasn't completed, but you can still drag and drop ships onto the grid, and they won't just drop where you leave them, they will actually snap intelligently into the nearest grid square. It doesn't affect the gameplay, but the snapping feature is still impressive.

Deploying to the Cloud

Unfortunately we weren't able to get this implemented. We started a bit too late on trying to integrate this and it didn't succeed due to technical difficulties.

Feature Table

Feature	**Scope**	**Programmer**	**Time Spent**	**File/Function**	**Lines of Code**
Animations	Front	Tom	8 hours	grid.js	200
Intelligent AI	Back	Eric	10 hours	Grid.cs	60
User Logins	Front, Back, DB	Chris	12 hours	Login.css, site.css, Identity scaffolding, _RealLoginPartial.cshtml, _LoginPartial.cshtml, _Layout.cshtml, HomeController.cs,	650

				login.cshtml, register.cshtml, loginviewmodel.cs, registerviewmodel.cs	
High Scores	Front, DB	Tom and Eric	4 hours	HighScoreController.cs, HighScoreIndex.cshtml	130
Power ups	Front, Back, DB	Eric and Tom	8 hours	Grid.cs, grid.js	300
Money and inventory	Front, Back, DB	Tom	6 hours	GamePlayController.cs, GameStoreController.cs, BattleshipDBContext.cs, GamePlayViewPage.cshtml, GameStore(index file),	250
Audio	Front	Chris	1 hour	Site.js, music.mp3	10
Dragging and Dropping Ships (this was a hot mess that didn't make it fully into implementation)	Front	Chris	25 hours	GameView.css, grid.js, GamePlayViewPage.cshtml, site.css	310
Store	Front, Back, DB	Tom	6 hours	GameStore(folder), GameStoreController.cs, BattleshipDBContext.cs	270
Game Logic	Back	Eric	20 hours	Cell.cs, Grid.cs, Game.cs,	820

				GamesService.cs, Ship.cs	
Grids	Front	Tom and Chris	10 hours	GamePlayViewPage.cshtml, grid.js	550

Individual Contribution

Note: if you find the number of features/lines of code a team member has added/modified to be small, argue why this is less important than the added capability provided by those lines.

TEAM MEMBER	TIME SPENT	LINES OF CODE
Eric Naegle	34	~ 1000+
Chris Bordoy	40	~ 1000+
Tom Nguyen	40	~1100 +

Summary

Summarize your team project and provide any other insights you would like us to have about the team, process, and/or project. Reiterate the most important parts of the project. In effect, argue what utility you have provided and why your project is worthwhile.

We feel like we have made a convincing and high-functioning version of a classic table-top game. The cool thing about our game is that it isn't just playable, it's challenging and rewarding. In previous CS classes, we made games that were playable, but not fun. This game plays like a real game, and it takes skill and concentration to win, and we've given players a way to compete against each other through high scores, even though there is no PvP multiplayer.

Suggest at what level you think your team performed: Superior, Good, Adequate, Needs Work, Poor. Explain your decision.

We performed “Good.” If we had hit all of our stretch goals and bonus objectives (multiple power ups, dragging and dropping your own ships), then we probably would have said “Excellent.” We collaborated really well and had good communication.