Tower Defense Game

Programmer's Manual

1. Introduction

Tower Defense Games are a genre of games with a devoted following. There is a simplicity to the concept of this genre : defending a home base against waves of attackers : that appeals to many different kinds of players. In homage to this genre, we have designed a simple addition to the genre that we hope people will enjoy.

1.1 - Purpose

The purpose of this game is ultimately to provide some entertainment to the player and a sense of accomplishment upon attaining a new high score. The goal of the gameplay is to stop the enemies, which spawn and move towards an end goal, from reaching that goal.

1.2 - Scope

Team name: Team Tower Defense

Summary: Our group wants to create a tower defense game in a web application format, using a C# backend and a React js frontend.

Platforms: Web browser (Firefox, Chrome, Edge)

How it functions: Utilizes a network connection for players to access it.

Any other software interactions: Just the browser used to access it.

How many players: Single player, with a multiplayer leaderboard.

The system will control the enemies which the player has to defeat in each round.

Programming languages: React js front end, C# backend with connecting web API, and EF Core ORM using SQL database

Interface: GUI interface in the web browser, mouse based interactions, with some keyboard usage to insert information about the player.

Features:

- Login necessary to play, player navigates to browser and enters name.
- Playable level where enemies move along predefined path, and player can place towers along that path to try and defeat enemies.
- Players are scored when they play, and the points are stored in a database of high scores which are viewable to anyone.

1.3 Definitions, acronyms, abbreviations

As this project was relatively simple in scope, there are no specific abbreviations or acronyms.

1.4 References

These are resources that were used to educate us about our project :

Canvas : https://medium.com/@pdx.lucasm/canvas-with-react-js-32e133c05258

Using React Hooks : https://dev.to/droopytersen/reactify-vanilla-js-libraries-with-hooks-fgd

Passing State using Hooks in React : https://stackoverflow.com/questions/40722382/how-to-pass-state-back-to-parent-in-react

Drag and Drop : https://www.youtube.com/watch?v=nfZvg4n3zzQ

https://www.youtube.com/watch?v=pxUv81-ymgU

https://engineering.datorama.com/mastering-drag-drop-with-reactjs-part-01-39bed3d40a03

Game Design :
https://www.raywenderlich.com/2709-how-to-make-a-tower-defense-game-tutorial

https://levelup.gitconnected.com/rpg-game-with-react-redux-html5-part-1-build-a-tile-map-9144fd867830

Animation / Graphics :
https://dev.to/ptifur/animation-with-canvas-and-requestanimationframe-in-react-5ccj

https://stackoverflow.com/questions/29017379/how-to-make-fadeout-effect-with-pure-javascript

2. General description

2.1 Product perspective –

This product (Tower Defense Game) is being developed as a requirement for CSC 478 Software Capstone at University of Illinois – Springfield. Tower Defense games are a popular and easy – to – play genre of game that our team all enjoys to pass time and provide mental stimulation.

2.2 Product functions –

The Tower Defense Game will have three essential components – Towers, Enemies, and the Game Board on which the other two components will interact. Players will have currency, which they will use to purchase towers, which will fire projectiles at enemies, which are trying to move from point A to point B. If enemies reach point B, the game is over. For each enemy defeated, players will earn points as well as currency used to purchase more Towers.

2.3 User characteristics –

The end users for the Tower Defense Game will be those who navigate to the website, enter an identifier for high score purposes, and click Play.

2.4 General constraints –

Game will be limited to one player. Game will be unavailable offline. Placing towers will require a mouse input for drag-and-drop – at this time, hotkeys via keyboard are not supported.

2.5 Assumptions and dependencies –

It is assumed that users have Google Chrome installed and an internet network connection.

3. Specific requirements

Pages

Login Page

1.0.0 - The Login Page should display Game Logo and should have space for the user to enter their name.

1.0.1 Enter Name button should send saved name to the database.

Game Page

1.1.0 - Game Page should display Game Board, user interface including displaying player resources, score, waves, frames per second (FPS), and Towers available for placement on Game Board.

Score Page

1.2.0 - Score Page should display contents of database containing player high scores and names.

1.2.1 - The Home button should bring the user to the Home page.

Home Page

1.3.0 - Home Page should display buttons Play and Highscores.

1.3.1 - Play button should bring the user to the Login Page.

1.3.2 - Highscores should bring the user to Score Page.

Game Board

2.0.0 - The Game Board should consist of a set number of square cells in which players will place Towers and Enemies will appear.

2.0.1 - Each Cell should be uniform in size and provide context for where Towers, Projectiles, and Enemies are in relation to each other.

Enemies

3.0.0 - Enemy types, speeds, paths

3.0.1 - All enemies should have a slightly variable maximum speed and a fixed minimum speed.

3.0.2 - All enemies should remove one life from the player if they reach the end.

3.1.0 - Basic Enemy ("Death Star") should have medium health and medium speed.

3.2.0 - Fast Enemy ("Ghost") should lower health and fast speed and provide more resource upon being defeated.

3.3.0 - Tough Enemy ("Metal Angel") should be slower than Basic Enemy and have more health than Basic Enemy.

3.4.0 - Boss Enemy ("No Face") should be faster than a Basic Enemy, and have the highest health.

3.4.1 - This enemy will also remove 5 lives from the player rather than one.

Towers

4.0.0 – Towers

4.0.1 - Towers should be able to be placed on set placeable tiles.

4.1.0 - Basic Tower should fire one projectile with medium damage.

4.2.0 - Slow Tower should fire one projectile at a lower fire rate, lower range and lower damage than Basic Tower.

4.3.0 - AOE Tower should fire one projectile at every enemy in range, lower than a Basic Tower and damage lower than a Basic Tower.

4.4.0 - Sniper Tower should fire one projectile at a higher range than Basic Tower, with higher damage and slower attack speed.

4.5.0 - Poison Tower should fire one projectile with less range than a Basic Tower and less damage initially, however it adds a damage over time to the enemy.

4.6.0 - Bank Tower should generate $5 per round.

Projectiles

5.0.0 – Projectiles (speed, size, types, collision handling)

5.0.1 - Each Projectile should match the color of the tower that fired it.

5.0.2 - Projectiles should disappear when colliding with an enemy.

Game Logic

6.0.0 - Game Logic (enemy spawn behavior, scaling, movement, game end)

6.0.1 - Enemies should spawn at a rate of 1 per round, increasing by 1 each round.

6.0.2 - The first four rounds should spawn basic enemies only.

6.0.3 - Starting with the 5th round, the game should have a chance to spawn Ghosts (fast enemies) as well as Metal Angels in the 10th round (tough enemies).

6.0.4 - Every 5 rounds, a Boss enemies equal to wave number divided by 5 should spawn.

6.0.5 - Every 5 rounds, enemy strength should multiply.

6.0.6 - When a player's lives reach 0, the game should end.

6.0.7 - The End Game button should immediately end the game and send player score to High Scores database.

6.0.8 - The Pause Game / Play Game button should toggle game state between active and paused.

6.0.9 - When the game ends, the player's score should be sent to the database.

Money

7.0.0 - Money (player resource)

7.0.1 - Enemies should provide the player with money upon defeat. Money is used to purchase towers.

Score

8.0.0 - Score (player accrued accomplishment)

Enemies should add score to player total when defeated.

Sound

9.0.0 - Sound (music)

9.0.1 - Music should play when player checks a box and stop when player unchecks the box.

Drag & Drop

10.0.0

- Player should be able to click tower element, drag to location, and place tower.

Timer

11.0.0

- A timer should display on screen to show time elapsed.

Back-End Architecture

12.0.0 - High score controller handles HTTP get method to retrieve top 10 high scores and calls corresponding repository method.

12.0.1 - High score controller handles HTTP post method to add a high score and calls corresponding repository method.

12.0.3 - High score repository handles method call from controller to retrieve scores from the database.

12.0.4 - High score repository handles method call from controller to insert new score into the database.

12.0.5 - High score domain object maps the object to its database relational object.

12.0.6 - High score request object maps the json in the HTTP request to a C# object for use in the controller.

12.0.7 - Program.cs registers the high score repository for dependency injection.

12.0.8 - Program.cs adds a DbContext with connection string to allow use of SQL database.

12.0.9 - Program.cs adds a CORS rule to allow cross origin communication between the react app and the web API.