Tristan Leigh, Sam Boswell, Jonah Tuchow
Software Design
Jadrian Miles
May 26, 2015

Design Document

Description

We are building a game in which you defend your base from a wave of enemy circles. The goal is to build towers that will destroy the circles before they get to your base. You start with 10 lives, and you lose a life each time a circle reaches your base alive. You get gold from each enemy circle you destroy so you can build more towers to destroy more circles.

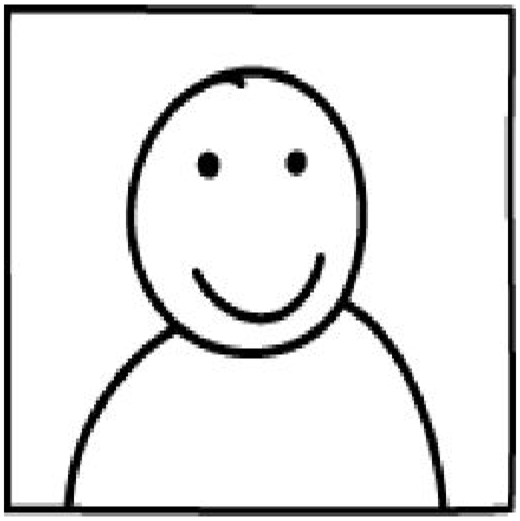
Features

- One tower that will shoot enemies
- Multiple types of enemies (with different amounts of health)
- One wave that starts with easy enemies, then has harder enemies, and ends with a boss enemy
- One stage
- Basic colors for the stage layout
- Towers can be upgraded once

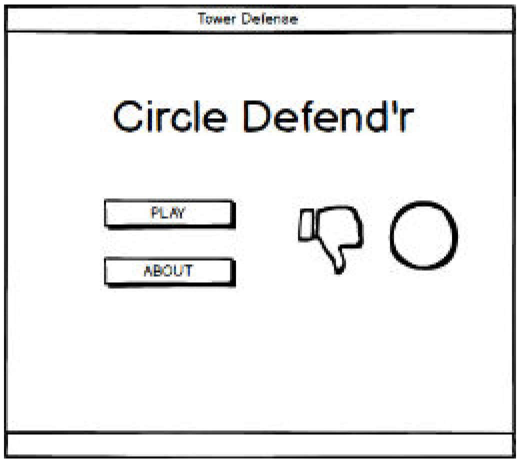
Stretch Features (may or may not be implemented)

- Multiple towers with different attack speeds and damage per attack
- Two stages; each stage has a different path for the enemies
- Towers can be upgraded twice
- Multiple waves

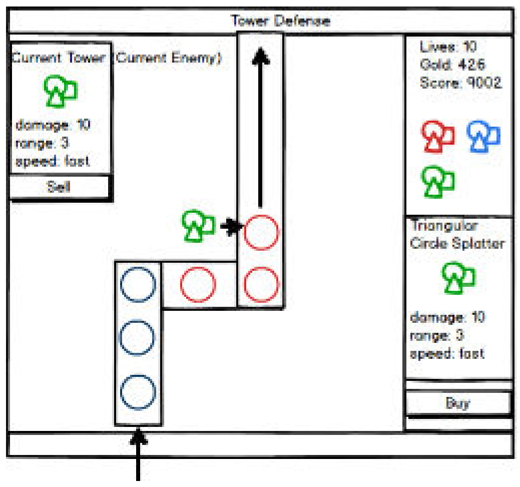
<u>Storyboard</u>



Timmy is a 12 year old boy who just came home from his day at school. He wants to go up to his room and waste some time with a relatively amusing tower defense game.



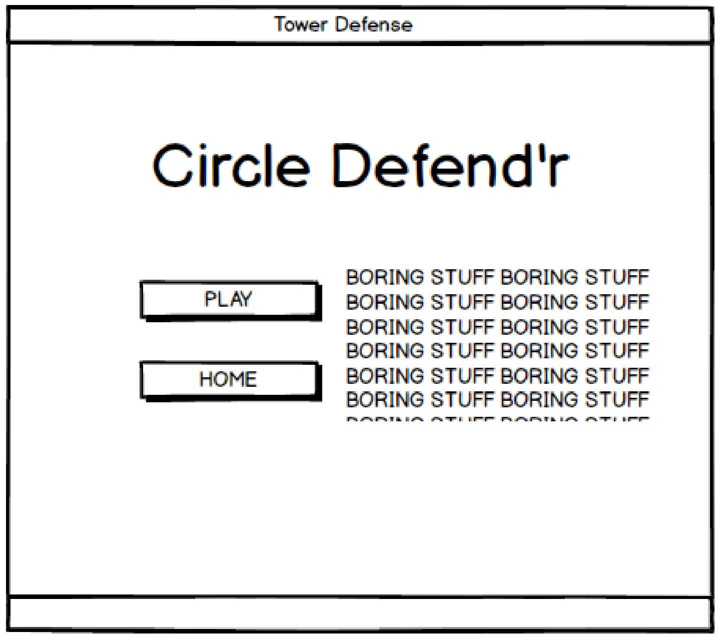
Lucky for him, he has Circle Defend’r on his computer. He doesn’t care about boring “about” information so he clicks on the play button immediately.



He places a tower in the middle and watches the tower shoot the circles.

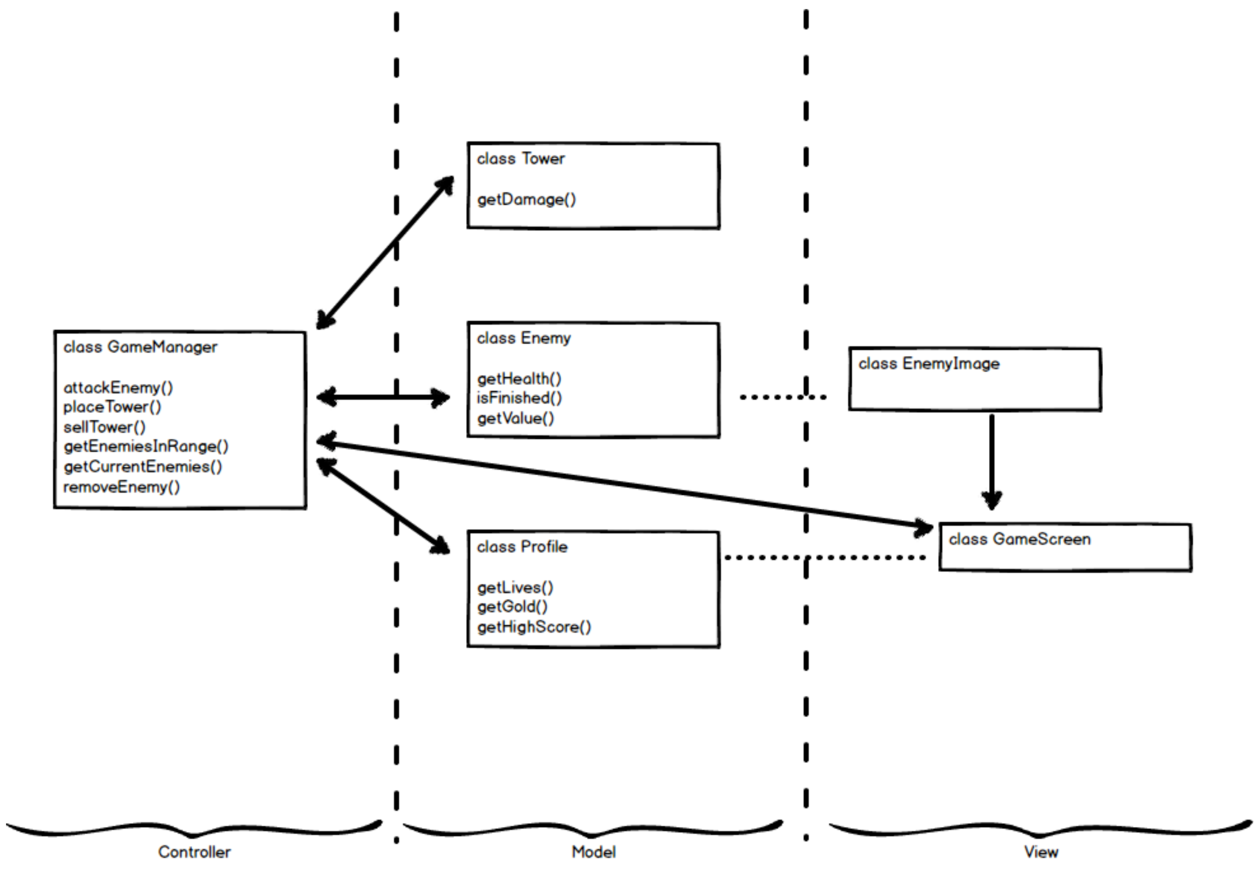
———————————————————————————————--

Jimmy is a Tower Defense game enthusiast. He just wants to see information about the game, and doesn’t actually want to play it. He clicks “ABOUT”.


Tower Defense
Circle Defend'r
PLAY
HOME
BORING STUFF BORING STUFF
BORING STUFF BORING STUFF
BORING STUFF BORING STUFF
BORING STUFF BORING STUFF
BORING STUFF BORING STUFF
BORING STUFF BORING STUFF

He reads the information and is content.

System Architecture



Signatures of Major Software Components

Model:

public class Tower {

    public Tower ( ) { }

        Constructor that initializes a tower.

    public int[ ] getRange( ) { return this.range; }

        Returns the range of attack that a given tower can reach to attack enemies. This information will be returned in a list of integers: minX, maxX, minY, maxX.

    public int getDamage( ) { return this.damage; }

        Returns the damage that a given tower inflicts upon enemies.

    public int getCost( ) { return this.cost; }

        Returns the cost to buy a given tower.

    public int getSpeed( ) { return this.speed; }

        Returns the speed of attack of a given tower.

    public string getStats( ) { return this.stats; }

        Returns the stats of a given tower.

    public int getX( ) { return this.xCoordinate; }

        Returns the x-coordinate of the tower on the grid.

    public int getY( ) { return this.yCoordinate; }

        Returns the y-coordinate of tower on the grid.

}

public class Enemy {

    public Enemy( ) { }

        Constructor that initializes an enemy.

    public int getHealth( ) { return this.health; }

        Returns the current health of a given enemy.

    public int getSpeed( ) { return this.speed; }

        Returns the speed of a given enemy.

    public int getValue( ) { return this.value; }

        Returns a certain amount of gold of a given enemy.

public boolean isFinished( ) { return this.isFinished }

Returns whether an enemy has finished the path without dying.

public int getX( ) { return this.xCoordinate; }

Returns the x-coordinate of a given enemy.

public int getY( ) { return this.yCoordinate; }

Returns the y-coordinate of a given enemy.

}

public class Profile {

public Profile( ) { }

Constructor that initializes a profile for a player.

public int getLives( ) { return this.lives; }

Returns the amount of lives a player has during the current game session.

public int getGold( ) { return this.gold; }

Returns the amount of gold a player has during the current game session.

public int getHighScore( ) { return this.highScore; }

Returns the high score a player has during the current game session.

public boolean canPlaceTower( ) { return this.canPlaceTower }

Returns whether a player can place a tower in a specific spot.

public Tower[ ] getUserTowers( ) { return this.getUserTowers }

Returns a list of towers that a player has during the current game session.

public void buyTower( ) { }

Subtracts money from the player's total gold amount, and places a tower in the chosen spot.

}

View:

public class GameScreen {

public int [ ][ ] getGrid( ) { return this.grid; }

Returns the main grid window for the game.

public int [ ] getPath( ) { return this.path; }

Returns the path that the enemies will take on the grid.

public string displayStats(Tower tower) { }

Displays the stats of a given tower.

}

public class EnemyImage {

public int enemySize( ) { return this.size; }

Returns the size of the enemy, indicating strength.

public string enemyColor( ) { return this.color; }

Returns the color of the enemy, indicating type (air, ground, or both).

}

Home Screen and TowerImage are FXML files

Controller:

public class GameManager {

public void upgrade(Tower tower) { }

Removes the tower from the grid and replaces it with the new tower the player has chosen.

public void attackEnemy(Enemy enemy) { }

Depletes a given enemy's health once hit by a projectile from a tower.

public List<Enemy> getEnemiesInRange(Tower tower) { return this.enemies; }

Returns a list of enemies in a given tower's range.

public int deadEnemy(Enemy enemy) { return this.gold; }

Returns the amount of gold an enemy yields and adds it to the player's total gold amount.

public void sellTower( ) { }

Removes a given tower from the grid and adds gold to the player's total gold amount.

public void placeTower(Tower tower, int xCoordinate, int yCoordinate) { }

```
                *Places a given tower in a given coordinate point on the grid.*
        public <List>Enemy getCurrentEnemies( ) { return this.numEnemies; }
                *Returns the number of enemies left in the wave.*
        public void removeEnemy( Enemy enemy )
    }
```

Tests

Example:

```
    public class TowerTest {
        private Tower myTower;

        @Test
        public void setUp() throws Exception {
            myTower = new Tower();
        }

        @Test
        public void testGetRangeShouldNeverBeNegative() throws Exception {
            assertTrue(myTower.getRange() >= 0);
        }

        @Test
        public void testGetRangeShouldReturnCorrectRange() throws Exception {
            expected = [1, 10, 1, 10];
            actual = myTower.getRange();
            assertEquals(expected, actual);
        }
    }
```

```
public class EnemyTest {
        private Enemy myEnemy;

        @Test
        public void setUp() throws Exception {
                myEnemy = new Enemy();
        }
        @Test
        public void testGetSpeedShouldNeverBeNegative() throws Exception {
                assertTrue(myEnemy.getSpeed() >= 0);
        }

        @Test
        public void testGetSpeedShouldReturnCorrectSpeed() throws Exception {
                expected = 5;
                actual = myEnemy.getSpeed();
                assertEquals(expected, actual);
        }
```

<u>Tests In Tower Class:</u>

- public Tower ( )
- public int[ ] getRange( )
- public int getDamage( )
- public int getCost( )
- public int getSpeed( )
- public string getStats( )
- public int getX( )
- public int getY( )

Tests In Enemy Class:

- public Enemy( )
- public int getHealth( )
- public int getSpeed( )
- public int getValue( )
- public boolean isFinished( )
- public int getX( )
- public int getY( )

Tests In Profile Class:

- public Profile( )
- public int getLives( )
- public int getGold( )
- public int getHighScore( )
- public boolean canPlaceTower( )
- public Tower[ ] getUserTowers( )
- public void buyTower( )