GRADUATE PROJECT PROPOSAL

Mobile Application:
Tower Defense Game

William Elmes

Fall 2015

INTRODUCTION

For my graduate project, I will be developing a video game as a mobile application. This game will be a Tower Defense style game, which is well-suited for mobile devices both in terms of performance and functionality. First, it is important to clarify some concepts that will illustrate the goals of my project.

One of the most basic concepts that needs to be defined is that of a Tower Defense game. A tower defense game is a simple strategy game in which the players tries to stop single-minded enemies from reaching their destination. The enemies simply follow the shortest path from their start location to their destination, and (usually) will not retaliate against attackers. The player's task of destroying these enemies can be accomplished by constructing stationary “towers” which will automatically attack nearby enemies. If too many of these enemies reach their destination, the player loses. By carefully choosing where to place towers, the player can increase the distance enemies have to traverse (as towers can form the walls of a “maze”) and increase the window of opportunity to attack nearby enemies (as towers can only attack enemies within a certain range of themselves). By destroying enemies and completing rounds of each level within the game, the player is awarded resources that can be used to construct additional towers or upgrade existing towers to become more powerful. Each round of a level becomes increasingly difficult. Once all rounds of a level are completed, the players moves onto the next level, which will clear all of the player's towers and present a new field of battle to the player. The ultimate strategy of these games stems from the placement and composition of the towers constructed by the player. Levels begin very easy, with little sensitivity to these elements, but later levels quickly escalate in difficulty, requiring a well-formed plan in order to survive every round of the level and win.

The next important issue to clarify is why my project will be a mobile application rather than a traditional computer game. While Tower Defense games are perfectly viable as computer games (and indeed were first created as one), they are an ideal match for mobile devices in many ways. Most notably, mobile games are mostly played in brief, fragmented sessions during downtime activities such as waiting in a long line. Tower Defense games are an especially good match for this niche, as they are broken down into levels which are further broken down into rounds, granting the player frequent opportunities to put down the game and pick up where he or she left off at a later time with minimal interruption to game progress. Another reason why Tower Defense games as a mobile application are a good match is because of user input. Mobile applications typically use a touchscreen as the primary source of user input, and tower defense games tend to have exceptionally simple controls, perfect for touchscreen input. Finally, Tower Defense games work well as a mobile application due to their light-weight performance requirements. Thankfully, Tower Defense games are well-suited to simple 2-dimensional sprite-based graphics, which almost any mobile platform can handle. With the mechanical-simplicity and natural fragmentation of gameplay, a Tower Defense game is an ideal fit for a mobile application.

With these concepts defined, I can lay out the objectives of my project and the activities that I will undertake to accomplish them.

PROJECT OBJECTIVES

The overall objectives of my project will be straightforward: create a mobile application Tower Defense game with 2D sprite graphics, a profitable monetization model, and is simple to learn yet difficult to master. These objectives will require me to define my development environment, how my application will generate profit, and my target audience.

I will be developing my game as an iOS application in Objective-C using the Xcode IDE. I am

already familiar with Objective-C and Xcode, so the learning curve will be minimal. My other relevant option would be developing my game as an Android application, however, I do not have the same degree of experience on that platform. As for which game engine I will use, I have two main options: Sprite Kit and Cocos2D. Sprite Kit has the advantage of being of being developed by Apple, so we can be sure of its performance and integration with iOS. Additionally, Sprite Kit can be fully integrated with Xcode, which will make it much easier for me to use with my background. On the other hand, Cocos2D is open source and has the support of Apportable, which would allow me to port my game to Android (for a fee). For this project, I will keep the scope focused on iOS and Xcode, so I will use Sprite Kit for my game engine.

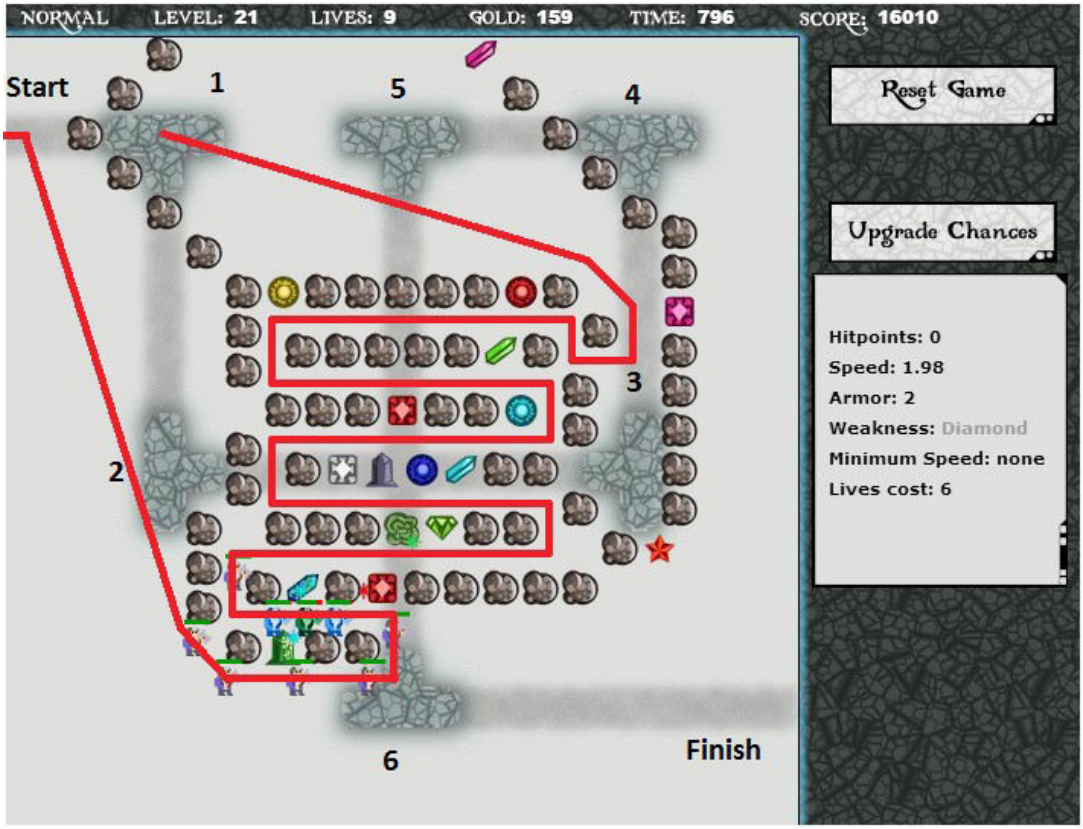
Next, I will need a way to monetize my game in order for it to generate a profit. The simplest way to accomplish this will be to create a system in which the player only has access to a limited number of lives that slowly replenish over time. Each time the player loses on a level, a life is lost. If the player runs out of lives, gameplay options become limited for the player until the player receives another life. In addition to waiting for a life to automatically replenish itself, players can also purchase extra lives via an in-app store. With this monetization model, I will be able to publish the game as a free download on the iOS App Store, but will be able to recuperate and surpass the publishing fees over time. By tracking players' progress through levels, and determining the rate at which most players are using the automatically replenishing lives, I can adjust the rate at which lives are replenished, and the cost of purchasing additional lives.

Lastly, my game must be well-suited for my target audience. The target audience for my game will be people who usually play video games in their spare time (typically traditional computer games), but who are also looking for a mobile game that more closely emulates the level of challenge required for those games. My game will be easy to learn and intuitive for anyone with experience with video games, but will also offer an optional level of challenge for players that seek it.

PROJECT ACTIVITIES

In order to accomplish the objectives of my project, there are several key activities that I must complete: I will need to implement efficient algorithms for key game-processes such as pathfinding, I will need to create innovative gameplay that will hold the players' interest, and I will need to implement an intuitive user interface that will be easy to use with a touchscreen. In order to explore these activities, I will first look at how other similar games have done this.

The first Tower Defense game I looked at was Gem TD (shown on the next page). In this Tower Defense game, you are presented with an open area for the player to construct towers in. In this area is a path that the enemies will attempt to follow (starting at the top left of the screen, leading around the screen, and finishing on the bottom right). The player is not allowed to build on the corner points of this path (shown in the grey, craggy ground) and must always leave a path from each corner of the path to the next (regardless of if that path actually follows the drawn path in light grey or not). This game is one of the simplest types of Tower Defense games, but demonstrates the strategy of creating a “maze” the best. As enemies are created at the start point, they must traverse the corners of the path in order, but thanks to the player's maze shown in the screenshot, we can see that that shortest path from the start point to the first corner becomes much longer. Likewise, to reach the second point of the path, the enemies must retrace their steps back out of the maze, and back through it once again to reach the third point. This repeats until the enemies eventually reach the finish point (if they have not been destroyed), creating an exceptionally long path compared to the default one. We can also see the elements of the simple user interface of this game, including the game statistics along the top of the screen, and the interactive elements along the right side.



Gem TD: Screenshot of the "maze" strategy in action with path from starting point to first corner.

GemCraft: Screenshot of gameplay.

The next Tower Defense game I looked at as an example was GemCraft (shown on the previous page). Here, we can see the path that the enemies take is much more straightforward. Enemies are created at the structure on the lower left, and traverse the dirt path to the structure on the center of the top of the screen. The player is not allowed to build towers on this path, and must resort to placing fewer but more powerful towers at key points. In this game, the construction of towers themselves is slightly more complicated. The player still creates towers on the map, but the towers do not attack the enemies. Instead, the player also creates "gems" which vary in power and special effects that can be placed in towers to attack enemies. Gems can be combined and used in interesting ways to allow the player to handle a variety of different enemies, and can be moved from tower to tower for an additional layer of strategy. Looking at the user interface for this game, we can see that the game statistics are once again along the top of the screen and the interactive interface elements are along the right side. We also can see information regarding upcoming enemies along the left side the screen, allowing the player to plan ahead.

Kingdom Rush: Screenshot of gameplay.

The final Tower Defense game of note that I found of importance was Kingdom Rush. In this game, enemies also follow a predetermined path (in this case starting at the bottom of the screen, then either taking the left or right path to reach the top of the screen), but the "plots" where the player is able to construct towers is also predetermined. However, the roles of different towers vary greatly here. While most towers simply attack nearby enemies as usual, some towers deploy troops on the path of the enemies. The enemies will engage these troops in combat, holding the enemies in place to allow the other towers more time to attack them. Additionally, we can see a panel on the bottom left of the screen that gives the player some abilities that can be activated during combat which will summon additional

troops to distract enemies for longer, or deal a large amount of damage to enemies in a small area. Both of these new mechanics are innovative and interesting, and show how small changes to gameplay can dramatically change the overall game. The user interface for this game is slightly different than that of the other games, as this game was specifically designed as a mobile application. Still, we see the game statistics in the upper left corner of the screen, but interactive elements along the bottom of the screen. Most of the interface elements are dynamic, and only shown when the corresponding tower or enemy is selected.

My game will incorporate gameplay and mechanics from several different games. The most defining characteristic for a Tower Defense game will be the pathing of enemies. I will use the "maze" strategy shown in Gem TD for my game, as I believe it requires the most strategy from the player. This will require a flexible pathfinding algorithm in order to allow enemies to navigate from their starting point to finish point. I will use the A* algorithm, as it is both efficient and flexible enough for my needs. This will also require me to use the tower construction methods of Gem TD, that is, allowing the player to construct a large number of towers in order to form a maze. However, I will use a mechanic similar to GemCraft, where the player's towers do not attack the enemies themselves. Instead, the towers will house objects similar to gems that can be placed in towers to attack enemies. Rather than just a straightforward construction method for these objects, a small amount of randomness will be incorporated into their construction, and the player will be able to combine the objects to form more powerful ones with new effects. This will force the player to adjust on the fly, and change each level every time it is played, giving the game some replay value. Finally, the user interface will have to be close to that of Kingdom Rush, as smaller interface elements will not work well with a touchscreen of a mobile device. Dynamic menus will allow me to fit more interactive elements onto the screen based on context. Additionally, as shown in the screenshots of each of these games, many elements are in the same place on the screen. Game statistics are always along the top of the screen, enemy information is along the left side, and interactive elements are along the right side. Following these subtle cues will allow my game's user interface to remain intuitive to experienced players, but still easy to pick up for new players.

By incorporating successful elements from other interesting Tower Defense game along with my own creative spin, I hope to develop an engaging and entertaining Tower Defense game for both new player and experienced ones.

PROJECT RESULTS

For the final deliverable of my graduate project, I will provide a fully-functioning and ready to publish Tower Defense game that will run on iOS mobile devices, my final report on my project, my application's source code, and a brief user's manual which will detail how to play the game and compatibility and installation information.

BIBLIOGRAPHY

Game In A Bottle. "GemCraft | Strategy Games | Play Free Games Online at Armor Games." GemCraft. Armor Games, n.d. Web. 10 Oct. 2013. <http://armorgames.com/play/1716/gemcraft>.

Holko, Peter. "Gem Tower Defense." Gem Tower Defense. N.p., n.d. Web. 10 Oct. 2013. <http://www.gemtowerdefense.com/>.

Ironhide Game Studio. "Kingdom Rush: Media." Kingdom Rush: Media. Ironhide Game Studio, n.d. Web. 10 Oct. 2013. <http://www.kingdomrush.com/play.php>.

Lester, Patrick. "A* Pathfinding for Beginners." A* Pathfinding for Beginners. Policy Almanac, 18 July 2005. Web. 05 Dec. 2013. <http://www.policyalmanac.org/games/aStarTutorial.htm>.