T.O.A.D: Tower Offense Android Development

A Senior Project

presented to

the Faculty of the Computer Science Department

California Polytechnic State University, San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science

by

Jeffrey Bryan

June, 2014

Table of Contents

Image Index

Abstract

TOAD is a tower offense game developed for Android smart phones and tablets. TOAD was developed to demonstrate applied principles of computer science and the software design process. The game engine was written as several components to handle various functions and allow for portability to other platforms. The end result is a fully functional game that can be used as a base to further expand on the idea and be showcased on Google Play.

Introduction

In the ever growing world of computer and mobile gaming, one category has been consistently on top of the popularity list: tower defense gaming. Quick and immediately satisfying, these wave based games provides users a simple outlet for fun. The normal style for these types of games is for the user to place towers along a preset path to prevent the preset enemies from reaching the end and hurting the user. Various enemy types and towers keep this interesting and allow for expansion and many hours of entertainment.

Project Goals

The goal of this project was to take the standard tower defense game and turn it around: allow the user control of the attacking enemies and keep the towers as a preset element. TOAD is the fully functioning and playable base for a tower offense game. It allows the user purchase, arrange, and launch minions along the tower protected path to the end. After the amount, types and order of the minions are decided, the user can then cast spells to help their minions reach the finish line. Upgrades are available to evolve the minions, allow for more to be bought, and increase the chances of reaching the end of the level.

Why Android

Android apps are developed in Java and Google provides plug-ins for Eclipse to ease the process of app creation. Their one time flat rate to publish on the Google Play store allows easy distribution to outside parties, unlike Itunes' $99 per year. Also, Android's OS is available on a large variety of mobile devices from many different publishers. Plus, not having Android programming experience makes for a challenging yet rewarding coding experience.

Related Projects

Kingdom Rush [1]

A very popular tower defense game. Ported for Android, iOS, and web based flash, Kingdom Rush has a variety of enemy and tower types and is a shining example of how a standard tower defense game can work. It's simple game play and achievements keep users intrigued. However towers can only be placed in designated slots and only four tower types are available.

Figure 1: Kingdom Rush

Villainous [2]

One of the few implementations of a Tower Offense game. Made for web based flash, Villainous provides many different upgrades, full game, increasing difficulty and many different levels. Though hosted on many sites, the game is only available through a flash-enabled web browser.

Figure 2: Villainous-before launching troops. Troops and order determined in menu on right of the screen

Figure 3: Villainous-after launching troops

User Guide

Adding Minions

The main feature of TOAD is the minion selection and organization screen. Users have a certain amount of money to spend on the various types of minions they have unlocked. Each minion has their own strengths and weakness; some have high health but are very slow and don't earn much gold. Another is fast and has increased durability but costs much more than a normal minion.

Figure 4: Minion selection screen

Order of minions is just as important as the types of minions. Sending stronger minions first will draw fire away from weaker ones. Of course, having a higher number of minions will keep the towers from focusing too much fire on any one minion. After minions are added to the queue, TOAD users can click and drag minions into their desired orders.

Launch

After finalizing minions the user sends the minions down the path. The minions automatically know where to go. As they race down the path they will gather gold for the user to spend on any subsequent waves that the user may need to send to complete the level.

Figure 5: Main level screen

Cast

Users are encouraged to use their ever-recharging mana to cast spells to help their minions down the path. Three spells are initially available for the user to cast.

Figure 6: Freeze spell

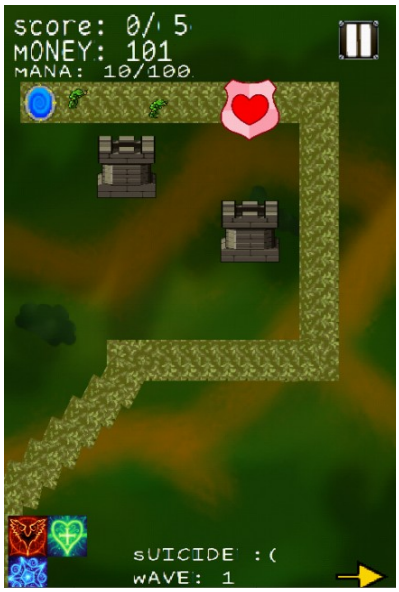

Figure 7: Heal spell

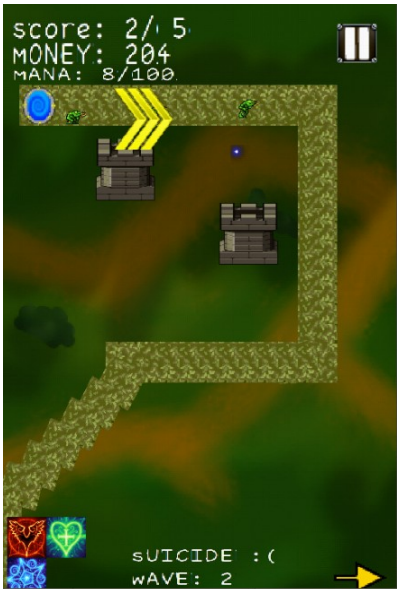

Figure 8: Flash spell

Mana capacity and recharge rates can be increased in the upgrade screen. Many levels are not able to be completed without using spells.

Upgrade

Upgrades can be bought to increase minions' likeliness to reach the end of the increasingly

Figure 9: End level screen

Figure 10: Upgrade menu

difficult levels. Gems are earned through the completeness of each level. Even if the user doesn't complete a path, they can still earn gems based on how far their minions get along the path and how many lives they took. Completing a level earns a large gem boost.

Technical Specs

Framework

A general game object was used to instantiate any calls or functions made to the Android OS. The main TOAD objects have little or nothing to do with the Android system so the base of the game can be moved or updated onto different systems.

Separate screens each have their own logic to render and update the game. Updates are based on frame rate, more specifically the time between the frames. The main game screen calculates all the vectors dealing with minion movement and bullet trajectory, along with calculating money and deploying spells.

Graphics

OpenGL is the standard high performance graphics library for Android developers. Android supports multiple versions of OpenGL, however TOAD uses one of the earliest versions, GL10, as this was implemented in Android APK 1, so all devices running Android would be able to run the graphics.

Assets in TOAD are saved as raw jpg and png files. These are read in with the AssetManager into bitmaps to be loaded and drawn as sprites. Because of the potential of many minions and bullets being on the screen at once, the assets are small and simple to render. The assets are statically loaded once and are only reloaded when TOAD loses screen focus, such as when the screen is locked or another application is started.

TOAD uses bounding boxes and circles to detect actions in the game. These were developed in the framework along with a general Overlap Tester. The Overlap Tester and shapes take care of receiving touch input and converting the input into game world coordinates.

Levels and Path Generation Algorithm

Each level has a tower listing and path definition(s). Towers are determined ahead of time and are repopulated every wave. Towers in the listing have a wave of appearance, disappearance, and waves where they might get upgraded to a stronger type, all in order to simulate a live player.

Paths are based on a series of control points. This implementation was chosen because it allows for an arbitrary size and style of path. Minions move naturally to the next control point in the list. This allows for maps that include looping and allows each minion to only need to index into the static path, rather than each minion having a copy of the path. It also allows easy branching capability for the paths: build up all possible paths once ahead of time as static variables, and then give each minion a index into which path they need to take.

Saving Data to SDcard

Data such as the current gems for upgrades, already purchased upgrades and completed levels are saved. These are written to a file on the SDcard. This allows the game instance to be preserved between closing and reopening of the app.

Target APK

Developing TOAD required the game to target a specific version of Android's OS. TOAD's minimum APK is 2.3.3. This allows the game to run on many older devices that still implement this version of the OS and still have full functionality on newer versions of Andriod's OS.

Because of the varying screen sizes, resolutions and other hardware running the app, it was much simpler to start with an older APK and scale upward. Targeting a smaller, set screen resolution allowed for easy movement of assets across the screen, easier touch input detection and response, and was able to scale to any size resolution relatively easily by multiplying by a few constants, mostly the screen size.

Future Improvements

While designing and implementing this application not all of the ideas made it into the first version of TOAD. Some future plans for the project:

- Artificial Intelligence for tower placement and strategy
- A more complete compliment of minion and tower types
- A back-story for the game (partially implemented in the help screens)
- 2 player competition mode where one player controls minions and the other controls tower placement and upgrades

Conclusion

Overall TOAD was a success. I became even more familiar with Java and was able to program for the Android platform, though I never had any experience with it. I was able to use the skills that I had gained during my term at Cal Pol. TOAD also gave me experience with working on long term projects beyond what was learned in one quarter courses. TOAD is a hands-on real-world application of the principles taught in classes.

I have always enjoyed the genre of tower defense an have always wondered about what a tower offense game would be like, since there are not many available. The mobile platform was a great experience as a solo project. The amount of easily accessible information to understand the Android was more than enough to breakthrough onto a platform I had never coded for.

Though not all of the features that I wanted were able to get done, I still plan on improving TOAD and posting it to the Google Play store as a free download. Future improvements on the game will help me hone my skills and let me work in other areas that I had not attempted before. TOAD ended up being both a great learning experience and a great game.

Bibliography

[1] "Ironhide Game Studio." Ironhide Game Studio. Ironhide Game Studio, n.d. Web. 25 May 2014. <http://www.ironhidegames.com/>

[2] "Villainous | Armor Games." Armor Games. N.p., 29 July 2011. Web. 24 May 2014. <http://armorgames.com/play/12143/villainous>.

[3] Zechner, Mario, and Robert Green. *Beginning Android Games Second Edition*. New York, N.Y.: Apress, 2012. Print.