Tower Defense

**Chapter 1 - Introduction**

Our group’s theme was “games”, and we settled on a “tower defense” game. The term “tower defense” usually refers to the genre, more can be read about it here: https://en.wikipedia.org/wiki/Tower\_defense. It is meant to be a casual game for all ages. It is by no means an intense strategy game, just something to pass the time.

**Chapter 2 - Requirement Analysis**

The requirements for this game are as follows:

* At least api 15 (Ice cream sandwich)
* Vibrate support/permission

**Chapter 3 - System design**

Development was very quick, the main game loop was the first to be implemented, then sound effects, and a settings menu. The game processing/rendering loop runs 30 times per second, which is a good tradeoff between usability and battery/device performance.

After the game loop and drawing surface was implemented, a utility class called Vector2f was created, this was solely to hold coordinates of game objects as floats instead of integers, which is crucial if you want movement to be smooth. The Vector2f class also contains utility methods such as computing the distance between two points on the screen.

An enemy class was then created with basic properties that all enemies will have. The map class was then created with checkpoints for the enemies, and the map graphic was drawn out to match those checkpoints. The update method (called on every game object for every frame) was then implemented on the enemies, so that they would follow the path.

The tower class was then implemented, with an update method to track the enemies as they walk by the tower. The projectile class was implemented alongside this to be shot at an enemy by the tower. The tower class was updated to shoot these projectiles at the enemies, and the enemies were updated to have a health value that is drained when being hit. Towers would get shot at a lot even though they were going to die by a bullet that was already headed for them, so another health variable was added to drain the health when the bullet was shot, to ensure other bullets would aim at enemies who weren’t dying soon.

The User Interface class begun development at this point, starting with the ability to place towers and sell towers. Soon followed different subclasses of towers and projectiles to fill up the slots of the UI. A lot of different aspects of the UI were added, like the buttons and screen to world coordinate translation, wave numbers, lives, and coins. The Wave class was also created, allowing enemies to be spawned procedurally, using a wave number in the constructor, enemies would be spawned with differing speed, delay between enemies, and health. Every 5 waves there is also boss enemies with more health.

Saving and loading was also added, because of the way upgrading towers is done, it is only necessary to save the level of the tower, type, and position, and not its stats. Also, the waves are procedural so it is only necessary to save the wave number. The game is saved to file instead of the shared preferences of the application. The game is only saved between rounds, so people cannot cheat by restarting their wave after having gained some coins.

There is a SoundPlayer class that is capable of playing sounds and has looping capabilities (for background music). The SoundPlayer class inherits from AsyncTask and uses the MediaPlayer class to play the sounds. Background music has been implemented. The track that we used can be used for noncommercial purposes. Other sound effects are on the way.

The most recent addition was vibration, when an enemy escapes your towers, the device will vibrate. This requires the android.permission.VIBRATE permission, declared in the android manifest.

**Chapter 4 - Testing**

There were no official unit tests done on this game, as they are kind of tricky to implement on an android device. Beyond that we all tested the game, and the few bugs that did appear got fixed eventually. As far as we know, there are no major game-breaking bugs present in the final version.

**Chapter 5 - Conclusion**

Some new concepts that we learned were saving to files in android, using a canvas to draw bitmaps to the screen, rotating bitmaps using matrices, and recoloring bitmaps using a ColorFilter.

The AsyncTask class was used for playing music and sound effects, and may be used for other things down the road now that we are familiar with it.