TURTRIS

Project Team 69: Jason Dang, Anthony Nhat-Nguyen, Wisely Kong, and Michael Shaffer

Overview

For our project, we wanted to make a mobile game that combines some of the best elements from Tetris and Bloons Tower Defense. Thus, Turtris was born!

Turtris is a strategic defense game created using Swift and SpriteKit where the player tries to prevent crates from building a tower and reaching the top of the screen, similar to Tetris. They can do this by placing turrets, which fire bullets to destroy the crates before they land. For every crate eliminated, the player earns money that they can use to buy more powerful turrets. Gradually, the crates fall faster and faster (increasing waves/difficulty), making the game more and more challenging over time!

Goals

The primary goal of this app is to create a fun game that is simple and visually appealing. Here are some of the features that we wanted it to have:

- Turrets that can be placed and fired
  - One of the core features of our game is the ability for the user to place and fire turrets. This is more fun than just clicking the crates to make them disappear, although the user can do that as well.

- Crate physics
  - Using SpriteKit, we made the crates fall via gravity and land on top of eachother. This is what allows for our game to end, as eventually, they will stack to the top of the screen.

- Upgrade mechanic/Turret shooting mechanics
  - When the player destroys a crate, they are rewarded 1 ‘Money.’ This can be used in the upgrade panel to buy stronger turrets, which have advantages such as increased clip size (shoot more bullets in a row) or decreased cooldown. Having only one turret is terribly boring so we had to incorporate upgrades so that the player can use different turrets with different shooting mechanics.

- Waves/progression
    - To allow for progression in our game, we added a round system. After a certain amount of crates are destroyed, a timer will appear, and the current round will end. Then, after a short time, a new round will begin with increased gravity.

- Multiple Scenes
    - To make the game look more visually appealing, we added a couple of different scenes. First, there is a main menu with the title, and an animated background representative of what the player will see in the real game. From there, the player can press a button to start the game. There is also a separate scene for the upgrade page, which can also be reached via a button once the player has started. Finally, even though we found some copyright free assets online, it is always good to give credit where credit is due, so we created a 'Credits' page to give credit to the artists.

- Continuous play
    - We also tackled the scenario where crates have destroyed the player's turret which causes the player to be unable to place a new turret. It would be horrible to have one crate end the game so we made it so that players can spend 1 'Money' to delete a crate.

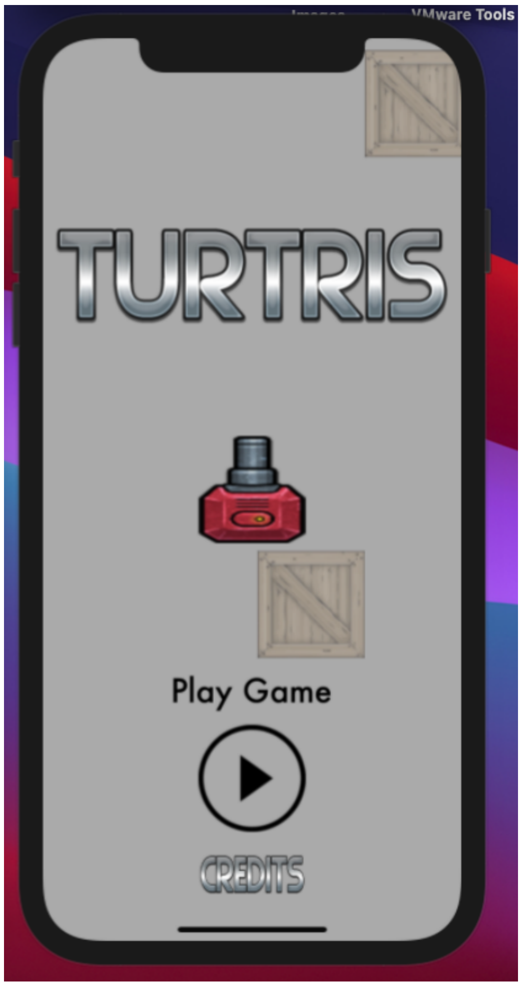
- Background Music
    - Every game has some sort of music playing in the background. Otherwise the game would be terribly bland. Fortunately, Michael is proficient with music so he was able to create a background music that relates to the art of our crates and turrets.

- Game over
    - Since we did not want the game to go on forever, we made it so that it ends once the crates fill/reach the top of the screen. This makes it more exciting, since the player has to constantly focus in order to keep blocks from stacking up.

- Better looking turrets
    - At first we started with turrets that were hand drawn (stick figures, pretty much). But we realized that there were opportunities to use copyright-free material to help fix the look of our game.

User Interactions

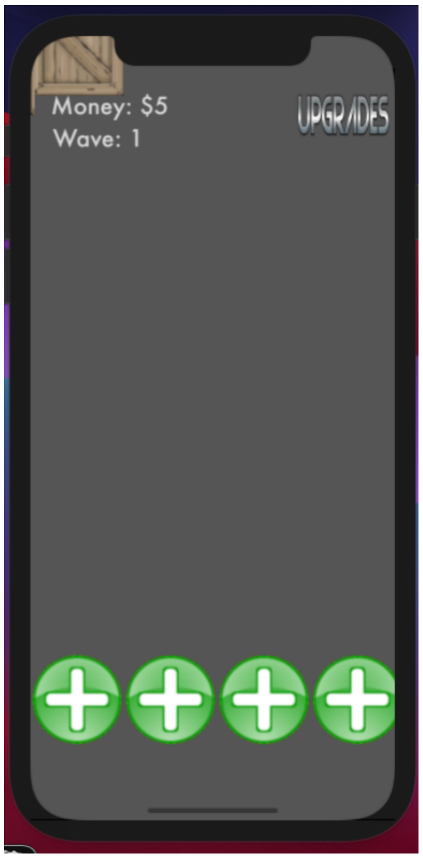
- Main Menu

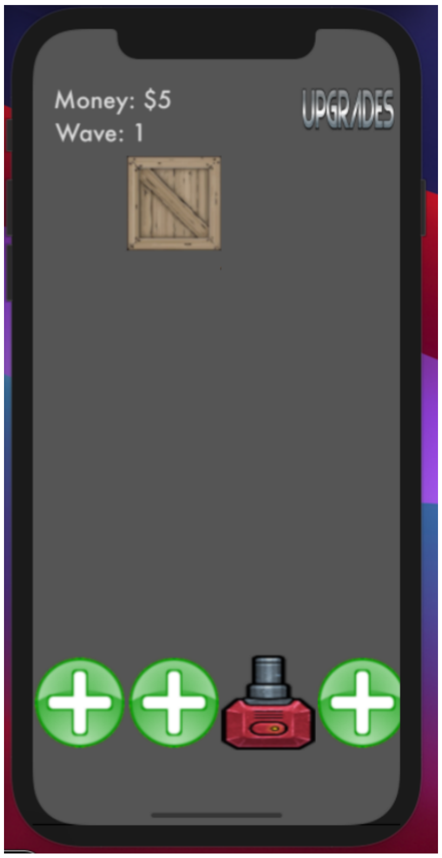


This is the title screen that the player first sees when they open the app. It has an animated background, with crates falling as they would in the game.

Once the player loads this screen, they can choose to either play the game by clicking the ‘Play Game’ button, or view the Credits by clicking the ‘CREDITS’ button.

- Gameplay
    - Initial Placements



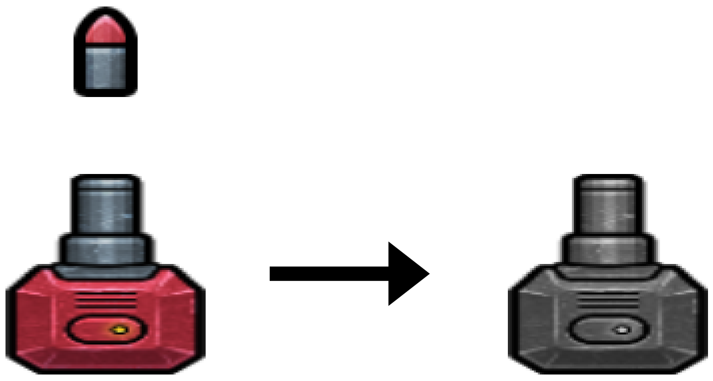


Once the player starts the game, they will see this screen. They can then click on the green ‘plus’ buttons to place their initial turrets.

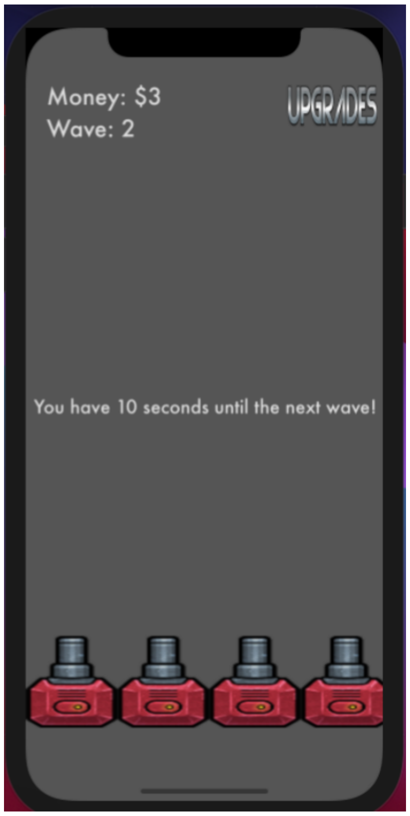
The initial placements are free, so the player will not be deducted any money. Here is what would happen if a player started the game, and then clicked on the third plus button.

    - Shooting Turrets

When a turret is clicked, it will fire a bullet. The turret will then turn gray, signifying that it is on ‘cooldown’ and cannot shoot for a short period of time. If a bullet collides with a crate, the crate will disappear and award the player with 1 money.



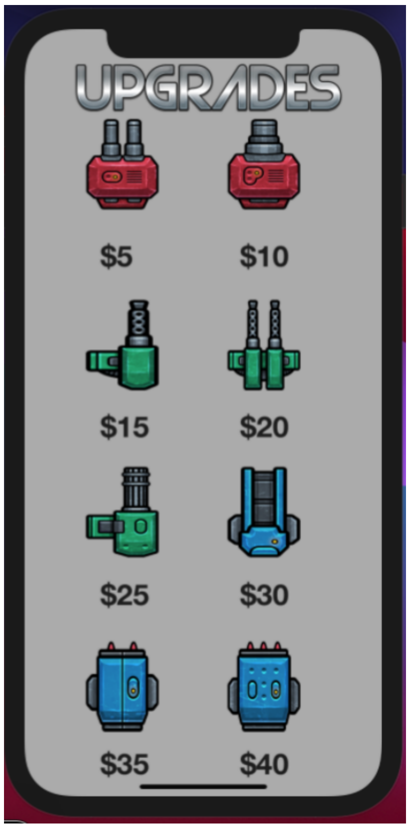
- Wave Complete



You have 10 seconds until the next wave!

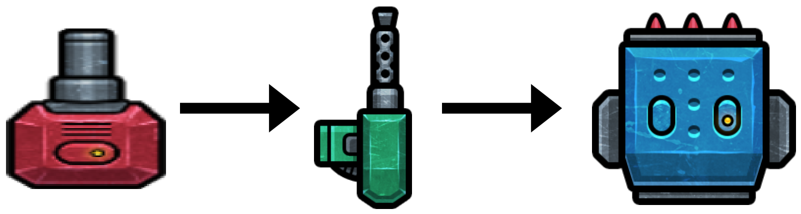
After the completion of a wave, the player will be prompted that the wave is finished. They will then be given 10 seconds to prepare for the next wave by upgrading their turrets. The 'Wave' counter in the top left corner will then increase, and the next wave will begin.

- Upgrade Menu

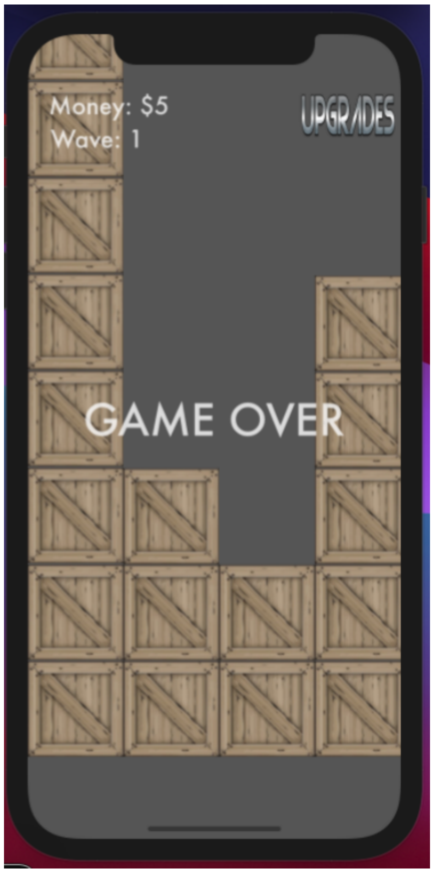


When a player selects the "Upgrades" button in the top right corner of their screen, they are shown this menu. Here, they can use their money to upgrade one of their turrets. Once they select an upgrade that they can afford, the next turret that they click on will be changed.

When a turret is upgraded, either it's clip size (how many bullets it can fire before going into cooldown) may change or its cooldown time will decrease. For some of the later upgrades, both improvements will be made. The final turret has no cooldown, so it can be fired as quickly as the player can click.

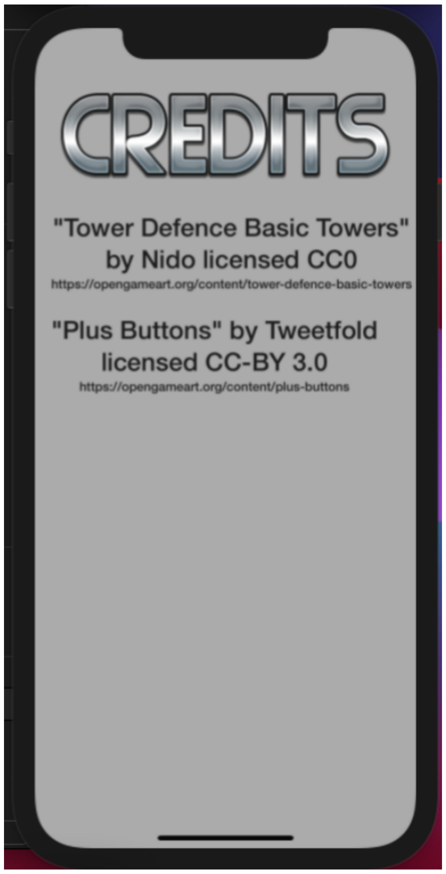


- Game Over



When a crate finally reaches the top of the screen, the game is over. A prompt will appear on the screen, and then the player can click anywhere to begin a new game.

- Credits Page



CREDITS

"Tower Defence Basic Towers" by Nido licensed CC0
https://opengameart.org/content/tower-defence-basic-towers

"Plus Buttons" by Tweetfold licensed CC-BY 3.0
https://opengameart.org/content/plus-buttons

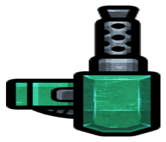
If the user clicks the “Credits” button at the main menu, it will bring them to this page. This is simply there to source all of the copyright free artwork that we used within this project.

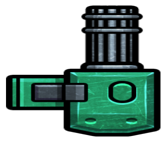
Upgrades (Details)

This turret can only shoot one bullet before it has to reload/go on cooldown. The cooldown is around 3 seconds.

This turret can shoot two bullets before it has to reload/go on cooldown. The cooldown is around 3 seconds.

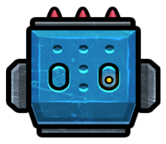
This turret can shoot three bullets before it has to reload/go on cooldown. The cooldown is around 3 seconds.


This turret can shoot three bullets before it has to reload/go on cooldown. The cooldown is around 2 seconds.


This turret can shoot four bullets before it has to reload/go on cooldown. The cooldown is around 2 seconds.

This turret can shoot four bullets before it has to reload/go on cooldown. The cooldown is around 1 second.

This turret can shoot five bullets before it has to reload/go on cooldown. The cooldown is around 1 second.


This turret is the best of the best. It has no need to reload/go on cooldown. You can shoot as many bullets as you want. Go crazy!

Development Process

We went through a lot of different ideas when we were brainstorming for this project. The first, which we almost ended up pursuing, was a recipe app that gave you suggestions based on what ingredients you had available in your fridge/pantry. We decided against this, as we thought that it would be difficult to sync our app with an online recipe database. It also seemed a little too basic, so we opted to develop a game instead. It would provide us with more of a challenge and we can use material we didn't learn in class. The recipe app would simply have been done in SwiftUI. After careful consideration, we decided on a mashup between two classic games, Tetris and Bloons Tower Defense.

The largest hurdle that we experienced during this project was trying to make the app without SpriteKit. After all, it wasn't taught in class. For our first milestone, we coded a modification of the "Raindrops" example that we were taught in class. However, while this allowed for the blocks to fall, it made stacking blocks and shooting bullets extremely difficult. We could not figure out a way to use SwiftUI and Swift for collision detection, physics, and sprite mechanics. As we mentioned before, we tried to avoid SpriteKit since we had no prior knowledge of it. We only knew about it because everytime we type up games in Swift, SpriteKit always pops up in some way.

However ,after our first milestone, we completely scrapped our code and started from scratch. We kicked the bucket and decided to use SpriteKit to implement the crate physics. So we pretty much had to learn something entirely new without a lot of time. It was a lot of hard work, but it definitely paid off. Implementing physics and collisions were much easier with SpriteKit than trying to do everything with basic Swift functions. Furthermore, we found that using free assets that we found online as our SpriteNodes would be much easier having to create the rectangles manually within our view. Nearing the end of the milestone 2 deadline, we had falling crates and shootable turrets. But we didn't have any menus which was part of our milestone 2 goals. This ended up being one of the hardest parts for milestone 2 since we weren't exactly familiar with scenes and scene transitions. We could implement the physics and collisions, but this had us stuck. After scouring through the SpriteKit documentation, we finally found a way to transition between scenes. Using that information, we added a main menu and the upgrade menu, although the buttons in the upgrade menu did not do anything yet. Another problem with the upgrade menu is that we decided to design it using a .sks file because it was easier to make an SKScene with that since we could drag and drop nodes without manually coding in positions. The reason why this was a problem was because we didn't have enough knowledge of SpriteKit to know how to interact with nodes in a .sks file which leads into milestone 3.

For milestone 3, we needed to turn our app from an idea into an actual game. We could place turrets, shoot, stack crates, and switch scenes, but the game still wasn't fleshed out. There were no upgrades, playing the game was too easy (no turret shooting mechanics so the basic turret was able to constantly spam bullets), and no game over mechanic. One of the hardest aspects about milestone 3 was figuring out how to maintain the state of the game when switching scenes. Everytime we switched to the upgrade menu and went back to the game, the whole game restarted as if it’s the first time the player was playing the game. If the game restarted every time that we switched to the upgrade or main menu, it would not be that much fun. It took an extremely long time, we were nearing the deadline, but we finally figured it out. After storing the game state, we worked on creating a working currency system and upgrade system. Once we figured out how to interact with nodes from a .sks file, it allowed for the player to upgrade their turrets, which added some much needed variety to the game. Finally, we implemented a wave system for progression, and the end-of-game controls for when the crates reach the ceiling. As part of our stretch goals, we were able to implement better looking turrets and a greater variety of turret upgrades. Additionally, we also added background music to our game.While there are some more things that we wished to add, we are happy with the current state of the game.

Future Directions

The biggest improvement that we could make in the future would be some new core gameplay mechanics to add variety. We experimented with several ideas, such as swiping to move the boxes from one lane to another, or having the boxes be different colors to indicate how many hits it takes to break them. This could make the game a lot more complex and mentally challenging for the player.

There could also be some simple UI improvements, such as a pause/play button and a high score page. It also seems like this game would be more suitable for an iPad. We severely underestimate how difficult it would be to implement this type of game with the limited space of an iPhone screen. Unless we were to make the crates and turrets tiny, it would be very difficult to add in more lanes.

Additionally, we could eliminate the need for the upgrade menu because we can integrate the upgrades directly to the turrets themselves. For example, we use a tap gesture on the turret to bring up a mini menu that has each new turret’s upgrades in a circular fashion. The player can then select a specific upgrade and the turret that the player tapped would immediately upgrade to a new one.