Richard Nguyen

April 4th, 2018

ISTA 451

# Shark Tank Report

## Game Instructions

The main objective for the player is to guide the small fish to safety while avoiding hostile enemies, such as sharks and blowfishes. There are a set of tools/skills available to the player to help reach this objective, which will be listed below. Each enemy will have a predetermined range of sight that will detect the player. The shark will chase the player until it is not longer in it’s range of sight. The blowfish, if triggered will cause the player to instantly lose the game, like a land mine. If at any point in the game the player were to collide with the enemy shark’s head, even if stunned, will cause the player to lose.

* Single Target Stun: This will prevent any enemy from moving for 3 seconds.
  + 6 second cooldown.
  + Max of 2 enemies can be stunned at a time.
* Screen Shake: A mass stun that will affect every enemy on the map.
  + Usable once per level.
* Hiding:
  + Prevent enemy vision from detecting the player.
  + The enemy can still collide with the player resulting in a lost for the player.
* Feeding:
  + The player can spawn up to two fish food onto the map.
  + Only sharks will chase after the food.
  + Sharks within 20 units will chase after the food.

## Game Design

Game Name: Shark Tank

Genre: 2D top-down puzzle game

Target Platform: Mobile devices

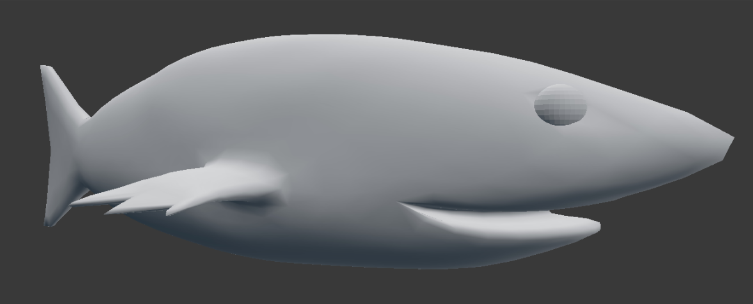
Target Audience: Younger children around 6-10

Target pricing: FREE!!!

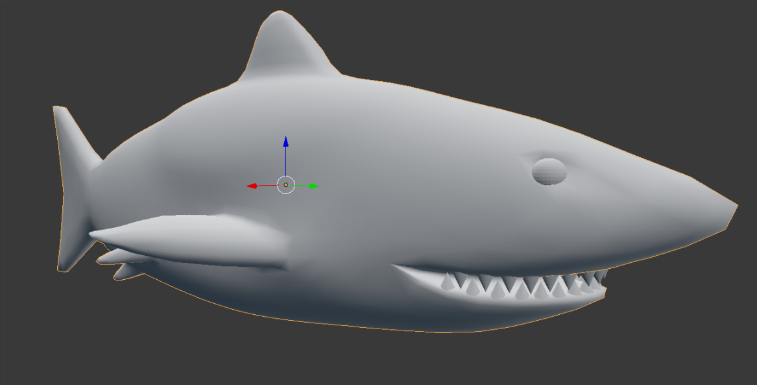
Game Engine: Unity

A Unique selling point is key to any successful game. Shark tank’s uniqueness involves how the player interacts with the phone. The main goal of Shark Tank is for the player to feel like the phone itself is a fish tank. Shark Tank emulates any action a human could perform on a fish tank in real life. This includes the ability to feed the fish, tap the glass, and even shake the whole fish tank. Although shaking the fish tank is not recommended in real life.

## Character Design



* Player



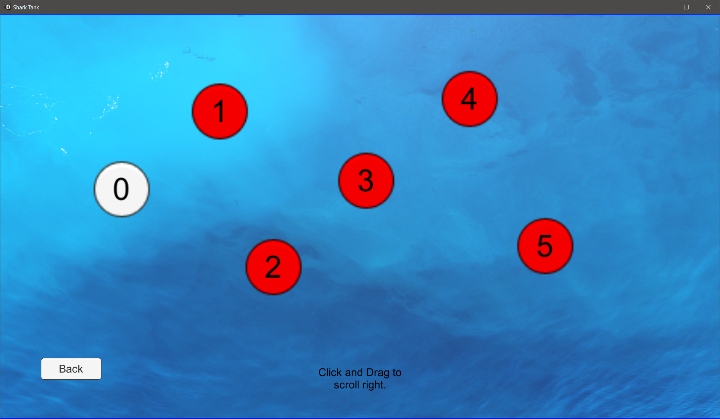
* Shark

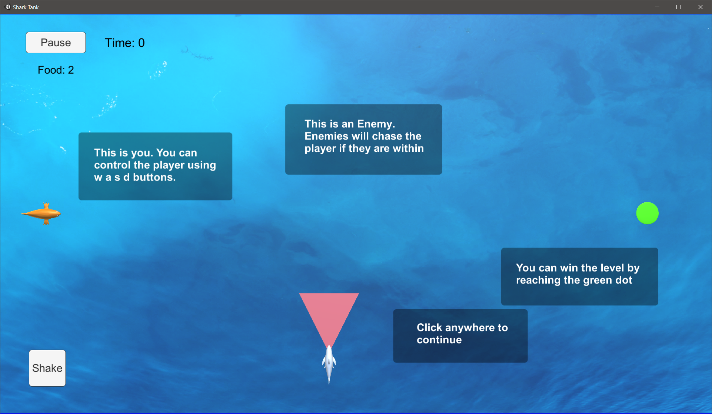
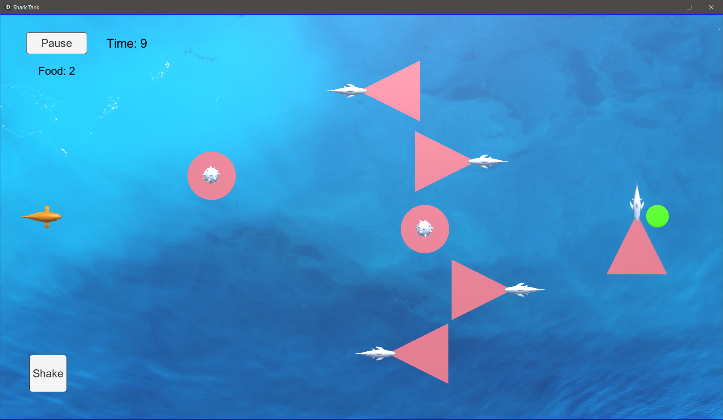


* Blowfish

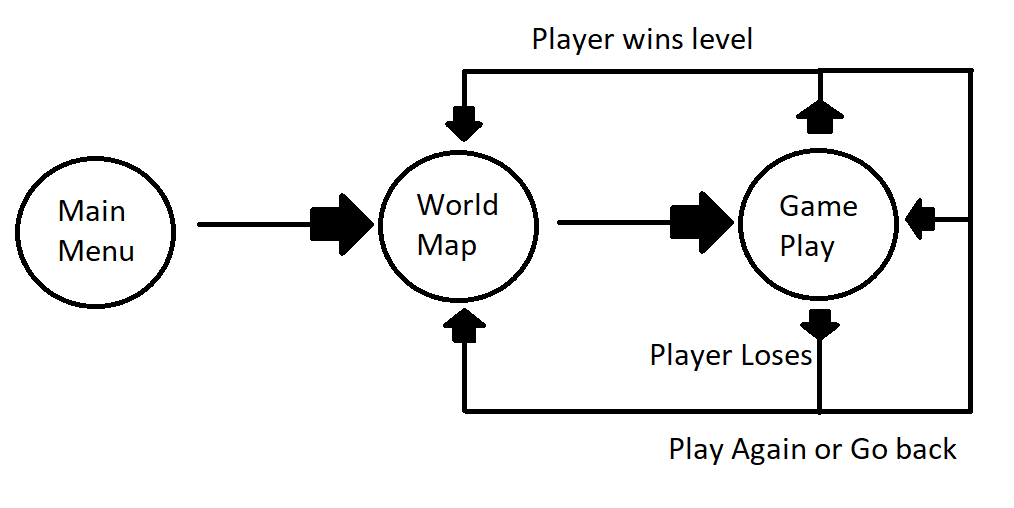
Each character design was made using 3d modeling software, blender. I was personally very satisfied with how character came out during first draft. Future revisions were not needed in my opinion.

## World Map Design



The world map consists of 10 levels total. Each level progressively gets harder and harder. Level 0 is the tutorial level, showing basic controls to the player. Level 6 is when the blowfish enemy is introduced into the game.

## Flowboard



## Challenges

Many challenges were faced during the development of Shark Tank. The most notable difficulty was the AI development. AI control was the highest priority and first of development. Without good AI, the game would be almost meaningless. Shark Tank’s AI core foundation uses Unity Navigation mesh component to navigate the map. Early stage development involved setting proper pathing for enemies which leads to our first challenge. The enemy would initially avoid the player and redirect its pathing; this defeats the whole purpose of Shark Tank. Initially the enemy would follow the player by using transform translate. This was solved by using the player’s position as a destination for the navigation mesh rather than direct transform translation.

Another minor issue were enemies following the player even after the player had left the vision range. After further testing, the issue was caused by how navigation mesh would follow points, in this case the player. Navigation mesh controls the enemy by setting destination points and will calculate the next point even before the current destination is reached. Once the shark’s remaining distance was less then 0.5 units it would assign the next point that has already been calculated. Since the shark never reached its destination it continued to indefinitely follow the player. This was solved by using a “detection” variable alongside Unity “OnTriggerStay” function. If the player is detected and the trigger is still colliding with the player, the shark will continue to follow the player. Otherwise reassigning the destination to the previous pathing point.

The last challenge faced with AI development was the food. Once a food is spawned by the player, all sharks within a 20 unit distance would attempt to eat the food. The first shark to reach the food would consume it and destroy the game object from the map. This caused issues with navigation mesh control, resulting in a back and forth pathing between a non-existing food and its old destination point. This was fixed by having a script send a message by calling a function on all shark game objects on the map. It would let each shark know that the food was consumed and to resume normal behavior.

Other minor issues

* Shark would still chase a player when it was hiding. This was fixed by turning off box collider trigger for the player
* Food would spawn behind buttons. This was fixed by placing invisible objects behind the button. Since food would spawn when the object “background” was clicked.
* Clicking enemy to stun was rather difficult and didn’t work at first. Adding a “Enemy touch” invisible game object above the shark fixed this. Allowed for easier stunning control for the player.
* Issue with null pointer exception occurring in build but not unity runtime with food being destroyed. Fixed by adding null checkers when chasing food.
* Character would still move during pause time. Fixed by setting Time.timeScale to 0.

## Testing and Debugging – Feedback

Similar feedback will be addressed under one feedback

“does not seem to be super unique.”

* The game during testing session was incomplete. After adding more player interactions such as the screen shake and the ability to spawn food, I believe this will address the issues of uniqueness.

“I do think that you should not be able to stun more than one or two sharks at once though.”

* After further testing, I came to the same conclusion, that stunning was an overpowered feature. Stunning was limited to a max of 2, except for screen shaking that stuns all enemies.

“I did like the fish and shark models though they might benefit from some color”

* I later added orange coloring to the player. The sharks I believe should be left white. This seems more realistic and still stands out from the background.

“Adding sound to the game”

* During the testing session, the game was incomplete. The final build includes sounds for the main menu, losing, winning and moving the player.

“You also had to option to press space to become invisible to the sharks for a couple seconds.”

* To keep the game more realistic and difficult, I believe invisibility is not needed. The current feature “hiding” adds a similar feature but keeps the game difficult. The player must calculate the pathing to know when hiding is a good use.

“It wasn’t particularly challenging, but there were only a few levels done”

* Five more levels were added since testing session and difficulty is increased per level.

“UI was alright, but a little blurry”

* I tried to keep the UI to a minimal since this game was developed for mobile devices. Fixing some scaling issues to address the blurriness.

“The AI wasn’t incredibly smart, and it seemed like they were just following a path.”

* The target audience for this game was intended for young children around 6-10. The game design was intended for the enemy to follow a set path. Randomize pathing would increase the difficult scope higher then I wanted too. This would also pull the game away from a puzzle game. Puzzle games, in my opinion, may possess multiple solutions but must offer the ability for the player to analyzes the level. This would not be possible if it was random, essentially it would become more luck then skill.

“just make the character a little faster than the sharks so you can escape, and it will give more of a chase feeling.”

* The game design is set to where the player and the shark have the same linear speed, but the shark has faster rotational speed. I want to force the player to have to use stun to escape. In hopes that it keeps the game engaging rather then just knowing you can out run everything.

“There was one occasion where a shark caught me, and it didn’t do anything, so maybe work on the collisions.”

* After further testing, I could not reproduce this situation. It’s possible that you were so close to the shark mouth right before detection that it turns by an unnoticeable amount and caught the player.

“more elements to add different challenges to present to the player instead of just avoiding more and more enemies. Perhaps barriers or hazardous areas that should also be avoided, or the implementation of different enemy types and power ups, could be helpful.”

* An additional enemy was added to the game, the blowfish. The blowfish is unique because it doesn’t chase food and will instant cause the player to lose. Due to time constraints, additional features like power ups couldn’t be added. The game design is better suited for an open world rather then a maze type world in my opinion. The main feature gameplay is maneuvering and running away. I believe obstacles would hinder the main gameplay.

“The only bugs were the inability to pause the game during tutorial mode and the ability to choose later levels from the level select screen without completing prior levels.”

* This was fixed in the final version of the game.

Additional feedback that involved adding more features to game that would improve the game overall was unable to be added due to time constraints hitting development. Potential DLC is being considered.

## Technical details and assets

All components and assets **excluding sounds and images** were developed by me.

**Scripts**

BlowFishController.cs: Control the blowfish enemy

* Start: Initialize variables.
* Update: Control AI behavior.
* StunEnemy: Stun the enemy for 3 seconds.
* GoToNextPoint: Calculate the next point in the enemy pathing.
* HiddenPlayer: Let the enemy know if the player is hiding for detection behavior.
* OnTriggerEnter: If the player collides with blowfish while not hiding, will result in a lost.

BoundsCheck.cs: Handles map edges

* LateUpdate: Make sure the player stays within the play area.

Data.cs: Player data

* No functions in this script. Holds player data on unlocked levels.

DestroyFood.cs: Handle food behavior

* Start: Initialize variables.
* OnTriggerEnter: If the shark eats the food then let all sharks know the food is gone.

GamManager.cs: Control UI and manage game

* Start: Initialize variables.
* Update: Calculate total play time for a level.
* PlayerWon: Toggle win overlay for player and unlock next level.
* PlayerLost: Toggle lost overlay.
* Continue2Handler: Return to world map after completing level.
* AgainHandler: Replay the level.
* QuitHandler: Return to world map.
* ContinueHandler: Unpause game.
* PauseHandler: Pause game.

Guide.cs: Manage guide scene

* Start: Initialize variables.
* BackHandler: return to main menu.

MainMenu.cs: Manage main menu

* Start: Initialize variables.
* DeleteHandler: Delete user data.
* QuitHandler: Close game.
* GuideHandler: Load guide scene.
* PlayHandler: Create player data if needed and load world map scene.

Player.cs: Manage player actions

* Start: Initialize variables.
* Update: Control player behavior
* StunAll: Shake screen ability. Stun all enemies in the map.
* MovePlayer: Helper function for Update(). Allows for player movement. Also manage player movement sound.
* OnTriggerStay: Manage collision with other objects. Also let sharks know if the player is hiding and will notify GameManager script that the player has lost or won.

SharkController.cs: Manage Shark behavior

* Start: Initialize variables.
* Update: Control shark behavior.
* Closestfood: Find the closest food to the shark.
* StunEnemy: Stun the enemy when clicked.
* GoToNextPoint: Calculate the next point in the enemy pathing.
* FollowPlayer: Follow the player.
* HiddenPlayer: Let the enemy know if the player is hiding for detection behavior.
* ResetEnemy: Return shark to normal speed.
* AddFood: Called by the player script and adds food to the food list array.
* AteFood: Called by DestroyFood script. Lets the shark know the food is no longer there or finds the next closest food if it exists.
* OnTriggerStay: Increase shark speed to chase the player if the trigger is colliding
* OnTriggerExit: Reset the shark to normal speed when it doesn’t detect the player anymore.

TutorialManager.cs: Manager the tutorial in level 0

* Start: Initialize variables.
* NextHandler: cycle through the tutorial for the user.

WorldMap.cs: Manage level selection

* Start: Initialize variables.
* BackHandler: Load the main menu.
* LevelZeroHander: Load level 0.
* LevelOneHander: Load level 1.
* LevelTwoHander: Load level 2.
* LevelThreeHander: Load level 3.
* LevelFourHander: Load level 4.
* LevelFiveHander: Load level 5.
* LevelSixHander: Load level 6.
* LevelSevenHander: Load level 7.
* LevelEightHander: Load level 8.
* LevelNineHander: Load level 9.
* LevelTenHander: Load level 10.

**Outside Assets**

All Sounds and images were used to help enhance user experience and overall feel of the game. All assets are free for personal and commercial use.

* Player wins by fins
  + <https://freesound.org/people/fins/sounds/171671/>
* Player lost by deleted\_user\_877451
  + <https://freesound.org/people/deleted_user_877451/sounds/76376/>
* Wrong action by Gabriel Araujo
  + <https://freesound.org/people/GabrielAraujo/sounds/242503/>
* Level start by plasterbrain
  + <https://freesound.org/people/plasterbrain/sounds/243020/>
* Video trailer music by djgriffin
  + <https://freesound.org/people/djgriffin/sounds/172561/>
* Main menu background music by Mrthenoronha
  + <https://freesound.org/people/Mrthenoronha/sounds/369920/>
* Player movement sound by InspectorJ
  + <https://freesound.org/people/InspectorJ/sounds/398702/>

Images

* Background image used for all scenes by Jaymantri
  + <https://www.pexels.com/photo/water-blue-ocean-5412/>

Unity documentation was referred to when needed for development

* <https://docs.unity3d.com/Manual/index.html>