The Giving Child

Group #2

Sean Zusi, Mark Reifsteck, Noah Deibert, Gett Villanueva, Han Nguyen, Coleman Hoyt

Project: Reef Defender

June 20th, 2017

Introduction:

Our client in a non-profit group called The Giving Child, whose goal is to raise money for various charities and global causes through making mobile games for kids. TGC has a wide array of available games already. When purchased, 90% of profits go directly to charity, while the other 10% of profits go towards developing new games. Our client's vision for this project was to create a mobile app game for toddlers and kids that helps them learn about the importance of keeping our oceans clean. Our game is based around swiping trash off of the screen as it floats through a coral reef. There are five main parts of the game: Toddler Mode, Levels 1-3, and The Reef. Toddler mode is essentially level 1 without an end condition. The three levels are based around cleaning up the trash that spawns with various difficulties. The Reef is an extra feature where you can view all of the special fish you have collected through playing the game. This project was split into two separate teams. Team #1 focused on the menus and navigation as well as level one, while Team #2 focused on The Reef, Level 2, and level 3.

Requirements:

Requirements Completed by Team #1:

- Toddler Mode: Endless mode where trash spawns, unable to lose level.
- Level 1: Essentially the same as Toddler Mode, however, you can lose by missing enough trash

Functional Requirements- Team #2:

Two defined levels:

  - Level 2: tap floating trash while avoiding tapping on fish. Fish have health bars, and lose health if trash collides with them
  - Level 3: level 2, with T/F and multiple choice questions integrated in some way. Possibly through power ups where if the player answers the question correctly, the power up is attained

Non-Functional Requirements:

- Android and/or iOS game
  - Client has asked for both. Assuming no setbacks, Unity should be able to export to both platforms.
- Built using Unity

    - Unity was decided on 5/16 as the platform of use because of its ability to port to android and iOS and because of its functionality as a game development tool
- Pricing: $1-$3
    - This all comes down to the quality of the app in the end and how much TGC thinks is appropriate for it. We also have the option of making the game free, and just using the in app donate button. This button would not be an in-app purchase, but would link to TGC's main page to donate. This will be used in order to lessen the chance of unwanted purchases being made.
- Release to App Store and/or Google Play
- Original artwork
- Sound effects and music to create a more stimulating game

Risks:

- Porting between iOS and Android
- Difficulty of compatibility between teams
- Over complicating the mechanics (it is for toddlers after all)
- Not Finishing (too large of a scope)

Definition of Done:

Specifically Team 2 (Definition of "ready for integration"):

- A full bank of questions to be used on level 3
- Two distinct levels ready for integration with the other team

Full Game:

- A finished game that is ready to be placed on the Apple and Android marketplace.
- The game has three distinct levels
- When opening the app, the user can choose which age group to play as
- App has been tested on both platforms.

Extra Features:

- The Reef was added as an extra feature during development. During gameplay, a special fish will occasionally swim across the screen. When tapped, the fish will be sent to the Reef, which is a separate mode in the app where you can see all of your collected fish swimming around. This was added to give the game another level of achievement.

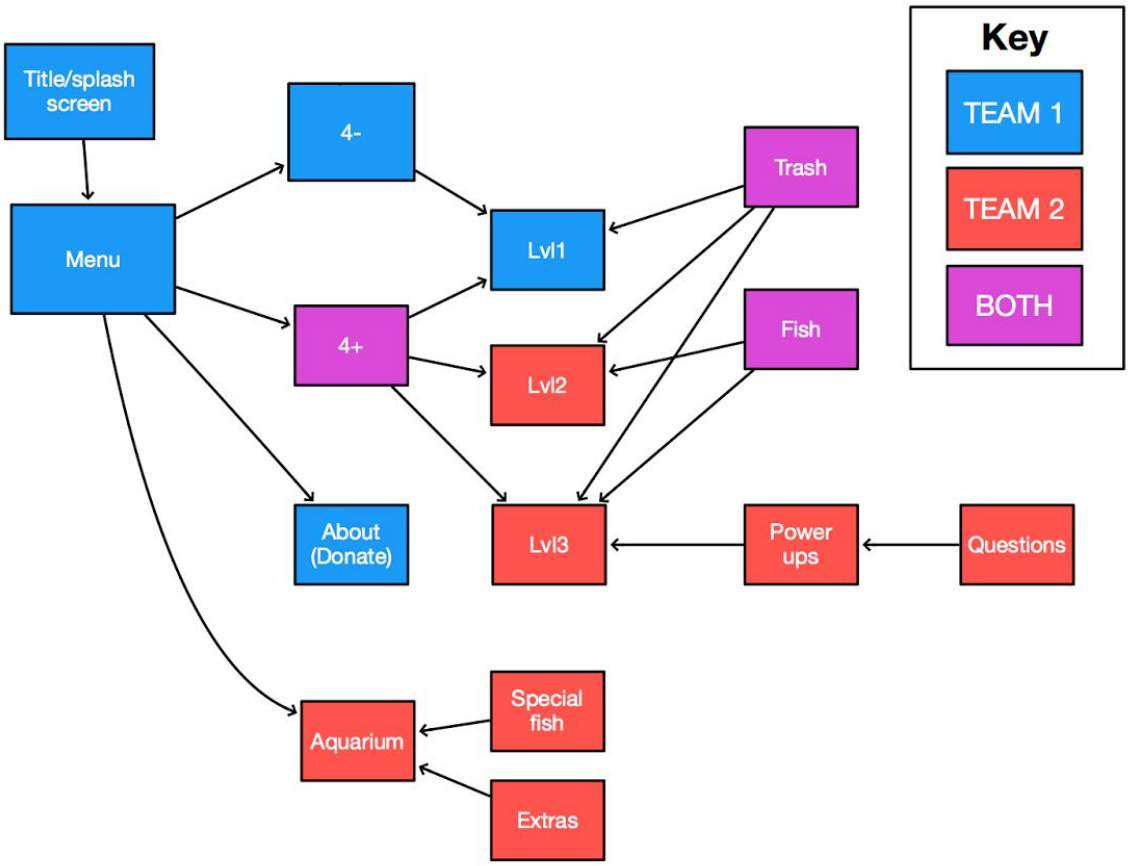

Figure 1: Team Responsibility Breakdown

Figure 1 shows the breakdown of responsibility between the two teams working on the project.

System Architecture:

- Unity is a compiled platform written in C++. All of our C# scripts and objects are compiled for the target system. On Android, this means the app is running on top of the Java interface, regardless of the chip architecture the system has. The Java interface in turn runs on Linux, a free operating system that runs on virtually any chip. iOS is different because apps run directly on the Apple chips, and the code is compiled to run directly on the device. This makes it compatible with Apple's proprietary hardware.

Technical Design:

Aquarium Saving:

In the course of creating the game we ran into a design challenge: we wanted the fish that you unlock for your reef, (shown in Figure 3) to be persistent and to have names that would last between runs of the application. Thus we decided to have the game contain a file where which fish you had unlocked could be saved. There are a number of considerations that we made when designing how the reef would handle save files.

Anti-Cheat:

In order to prevent the user from cheating, we decided to use a serialized data object saved as a .dat file. This means that there is no text file that a user could simply change a line of text that read "fishUnlocked = false" to "fishUnlocked = true" in order to unlock a fish. In theory someone could still cheat by exchanging save files, however, this layer of security will most likely put off any would-be cheaters.

Scaleability:

We decided to let each fish keeps track of its own save data. This allows the application to be more easily scaled. If we decide to add additional fish into the game, we need only to duplicate the in game object and the script will keep track of the files behind the scenes. There is no need to add and search through to an increasingly large file for the single fish data that is needed.

Ease of Use:

A data object was used for the ease of use of the programmers involved. We created a new object without any methods and only public variables in order to keep track of the variables that were being used in each save, as well as to allow all the data to be serialized all at once. This way it is also very clear what data is being saved and loaded.

Overall we decided on the algorithm demonstrated by the following flowchart (Figure 2) in order to implement these features. Each fish calls their own save function on the aquarium scene being loaded and calls the save function before each time the scene is unloaded.

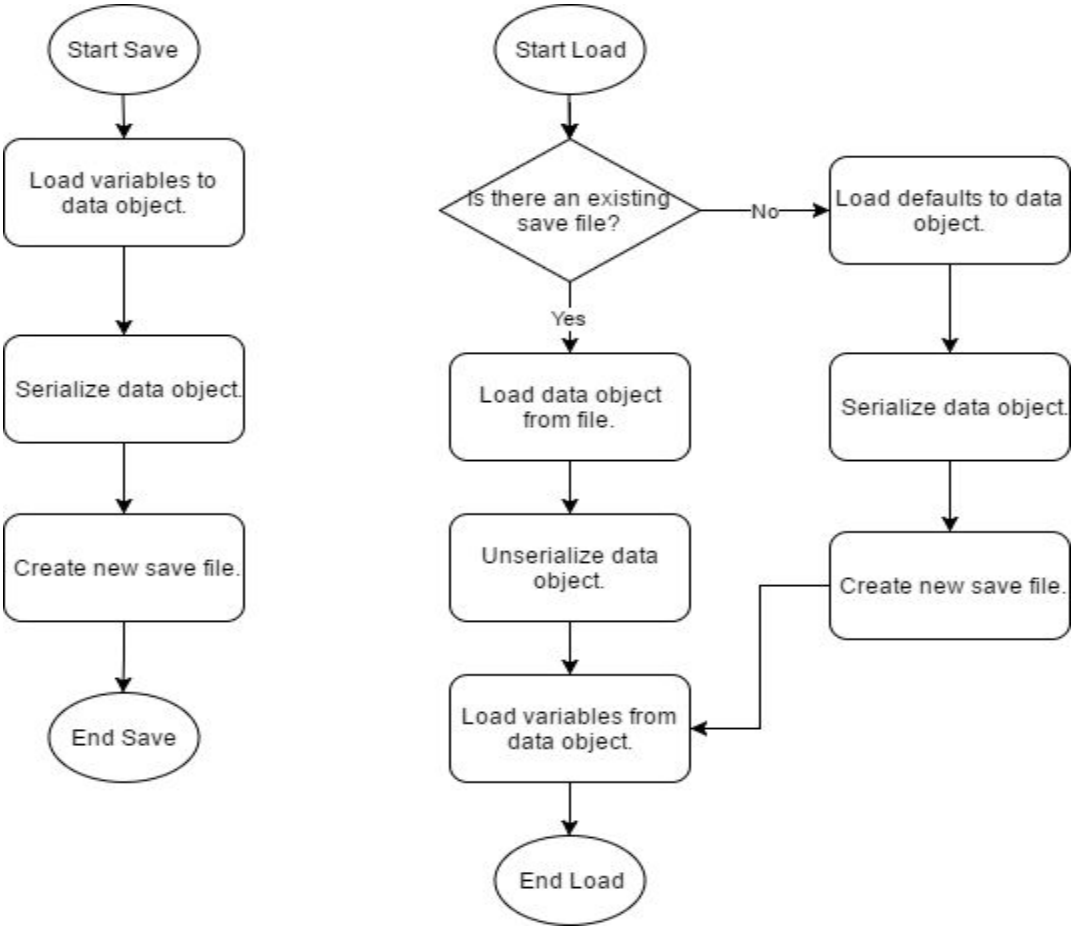

Figure 2:
Flowcharts demonstrating the saving and loading features for the reef.

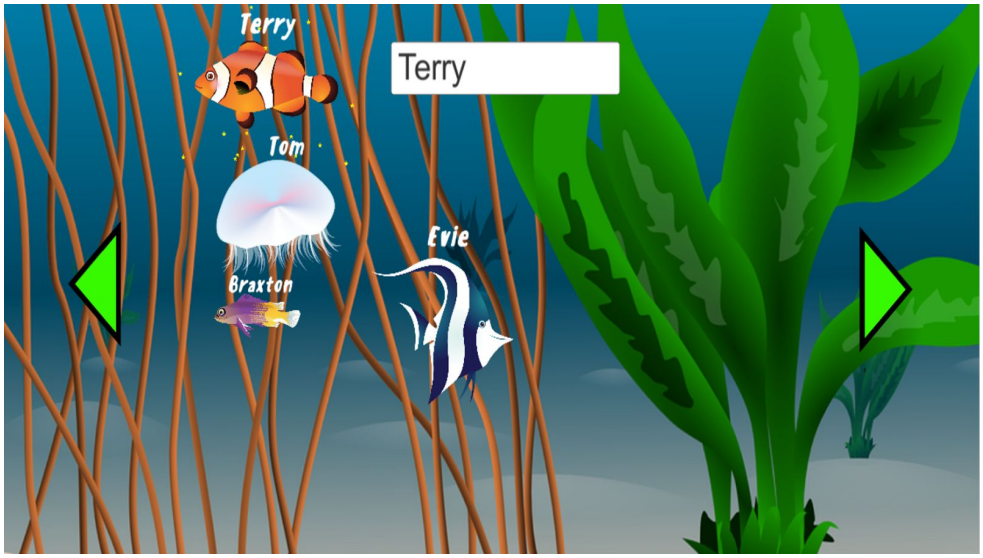

Figure 3:
The reef showing multiple unlocked fish

Design Decisions:

General:

Platform Choice:
The first major decision our group made was which development platform was going to be used. Our client asked for our game to be portable to both iOS and Android, thus we decided to not use either platform's native coding. Initially, we were going to use Game Maker Studio, however, a license that allowed us to port to a mobile platform was too costly. After some debate, we settled on Unity as it met all of our requirements.

Premade Assets vs. Handmade Assets:
Another decision that we had to make was whether we were going to use the assets, such as fish, that Unity provides to you through their store or make our own. We currently have a mix of the two. All of the trash and some of the fish are currently pre made, and a few special fish are handmade. Premade assets are easier to use because you can just download them, however custom handmade assets look better and are easier to fit to our project.

Donate Page vs. In-App Purchases:

For security reasons, we needed a way to prevent toddlers from accessing a donation page and using the phone or tablets saved payment information to donate without the parent's consent. Instead of having in-app purchases, there will be a donate button that opens a web browser and navigates to the client's website where a donation may be given, manually entering payment information.

Team Specific:

Question Bubble Actions:
The questions will pop up in multiple choice format. It was decided that multiple choice is optimal for both the diversity of the questions able to be asked, while still being suitable for toddlers. Short answer questions were deemed too difficult for toddlers, as it would probably have to check for spelling. Popping up a keyboard might also convolute the screen. True or false questions would be too simple as it's just a 50/50 shot at a correct answer.

Upon successfully answering a question, a power-up will be given to the player and activated immediately. These power-ups vary and may clean up all trash on the screen, slow down time, or restore the health of your 3 fishes. For increased difficulty, incorrectly answering a question will result in a huge wave of trash entering the ocean.

End Conditions of the Game:
Another decision we had to make was about how the user would lose the game. Loss conditions for the Toddler Mode and level 1 will be decided by the other group. For level 2 and 3, there will be 3 fish swimming around on the left ¾ of the screen. These fish will have health bars and will lose health when trash is touching them. Once a fish losses all of its health, it will die. When all 3 of your fish have died, you will lose the game. On the other hand, collecting the required amount of trash will successfully "clean up" the area and complete the level.

Since the end conditions will be fairly easy to achieve, there will also be an endless mode with no winning condition. The endless mode will be endurance based, and instead of keeping your fish alive, you will need to continually collect trash in order to stay alive.

Results:
The goal of our project was to create a functional game for kids that would teach them about ocean pollution in a fun and informative way. Our application meets our client's functional requirements, and goes beyond them by having many added features past what was required. Our project is planned to be released on both iOS and Android, thus our game was tested on both platforms for usability and stability. Our application works well on both platforms, and any problems encountered, such as pop-ups appearing too small, were easily fixed.

Lessons learned:

- Working as a group in-person was extremely useful for keeping everyone on task and getting work done quickly. Our combined efforts while working together were much greater than working individually.
- Unity is a very powerful game development software. Along with many helpful tutorials and manuals, it was able to easily handle all of our functional requirements through its many features.
- Communication with your co-workers is key to a successful project. We were able to stay focused on our individual work, because we communicated well.

Testing:

- While the majority of our testing happened in the Unity desktop environment with a mouse and/or trackpad, we periodically uploaded the game to our personal mobile phones to test the game for the goal platform. Most of the touchscreen features in the game are suited fine to a cursor and button, except for dragging objects around the screen.
- This led to minor issues with graphics scalability. While there have been no issues with mobile phone hardware's capability to support the graphics coded in the desktop, some text boxes did not fit correctly, and some of the HUD was not suited well to the smaller display. This was fixed by scaling the graphics for the mobile platform instead of the desktop.