The Giving Child

Clean Water Game

Chris Card – Dylan Chau – Maria Deslis – Dustin Liang – Tony Nguyen – Gurpreet Nanda

Contents

Introduction:

Client description:

The Giving Child is a non-profit organization whose goal is to teach children that giving is fun and rewarding, in hopes of improving future generations and reducing the problems that the developing world is facing. The organization was inspired by the founding directors' experiences as nannies, seeing children conflicted and not fully comprehending the idea of charity and giving for the benefit of humanity.

Their first goal is to educate children about how the lack of clean water is negatively impacting the economic development of many countries. Without local access to clean water, many women and young girls are sent off to get water, a journey that takes up to 3 hours to get to the nearest source of water. This water is often sitting out in the open, such as ponds or rivers, and is exposed to all kinds of germs. Time spent gathering water, is time they can't spend learning to read, write, earn an income, or take care of their family. And the walk is not just difficult, it is dangerous. The women are alone and are carrying up to, if not more than, 40 pounds of water. Many get hurt and sometimes they are even attacked. The young girls who have to carry this water are still growing, and this causes their spines to fuse and deforms their bodies, making their futures even more difficult. When, and if, they make it home, the water they have collected is contaminated leading to diarrhea, dehydration, and even death, especially in young children.

To introduce this issue to children in a quick and attractive way that will motivate them to get involved in the matter, *The Giving Child* decided to create a fun, educational Android game called *Revolution: H2O*. All of the proceeds and donations generated by the game will be divided such that 90% of the proceeds will go to the charity of their choice, and 10% will be used for overhead and future app development.

Product Vision:

The Android game titled *Revolution: H2O* was designed to educate children from 8-13 years old about the water problems of other children throughout the world, and to inspire them to take action through fun, motivating gameplay. The game's goal is to both entertain and educate children about the lack of clean water in developing countries and how providing access to clean water can improve the quality of life. To engage children, the game has a young female protagonist and storyline that is both understandable and relatable. The levels are challenging, but also beatable to keep children engaged enough to replay the game. In the game, they learn how to alleviate the clean water problems that children in developing countries are facing, in a fun and interactive way. While there are other runner-style games on the market, none of them are trying to teach children about the clean water issues in the developing world and the solutions that exist to help improve such a morbid situation. Our goal was to develop the lite version during field session so that it can be launched in the fall. This version includes a single "act" which consists of a simple story of the main character, and five game scenes/levels with various obstacles and ways to traverse each level. The lite version will help *The Giving Child* gain visibility and make it more likely that people will consider paying for their future releases, as well as donate to The Giving Child or other existing charities that strive to improve the developing world.

Requirements:

The application had several main requirements that had to be met in order to satisfy the client's project proposal.

1. **It must educate the user about clean water issues:** The game teaches the importance of clean water and how clean water is a necessity in the developing world. The gameplay has to incorporate clean water and the struggles that developing countries have to endure due to the lack of clean water.

2. **It has to be consistently entertaining:** The game needs to keep the children engaged for the duration of the gameplay and also needs to be complex enough to sustain a fair amount of replay value.
3. **The game needs to provide a challenge:** It should be easy enough for an 8-year-old, but should also provide enough of a challenging to entertain a 13-year-old.

Functional requirements:

- The application sounds can be muted and unmuted
- Users of the app are able to visit *The Giving Child* donation site
- The game has multiple levels
- Report bugs - email error messages if error is detected
- Game can be paused - if options menu in level is selected

Non-functional requirements:

- Has to be easy to navigate through the menus
- Minimum Android OS is 2.3 Gingerbread
- If bugs occur, we have to be able to reference the errors to be able to fix them.
- Target ARM processor powered devices

Technical Design:

The most prominent design decisions are related to game interactions and level design. For the game interactions, the team decided to use swiping and single tapping gestures. Through the usability studies, the team found that swiping gestures are natural for kids, with a shallow learning curve. The game's primary interaction is jumping, which is executed by either swiping up or tapping the screen once. Swiping up to jump is an intuitive and easy gesture for jumping; however, the team also included tapping to jump for those who feel that tapping is a better quicker and easier action then swiping. By incorporating both types of controls for jumping, the team was able to cover a wide variety of gamer preferences. The next major gesture is swiping forward to sprint. The gesture from swiping left to right to make the character sprint has been intuitive for the users we have tested, and by including a sprint action the team was able to design more challenging and entertaining levels for the more mature children of the target audience. The last gesture the team incorporated was a single tap to start the character's running motion. This allowed the user the ability to prepare before starting the gameplay rather than being thrown into the game right away.

The level design was challenging, but the team decided to iterate the levels with increasing difficulty such that Scene 1 is easier than Scene 5. In each scene, the team introduce at least one new gameplay element. In Scene 1, the user is introduced to the basic jumping game play, hills and pits that they have to jump on and over, as well as the general collectables. However, in Scene 2, the user is introduced to floating platforms that they may jump onto and level-specific collectables. Scene 3 introduces the user to character sprinting and falling platforms. The sprinting feature allows the character to run quickly across falling platforms as well as jump even farther distances. Also in Scene 3, the level introduces the user to alternate paths to complete the level, one being more challenging than the other. In scene 4, the level introduces simple traps and difficult ways to attain more points, while Scene 5 combines the elements of the other scenes to engage the user in a more challenging experience.

Within each scene, the user is given a certain number of health points that are decreased as they hit the sides of obstacles. The goal is to avoid all the obstacles while trying to collect as many of the game's collectable objects. If the player falls into a bottomless pit or loses all their health points, a "Game Over" screen appears, and allows the player to try again or quit. All levels are designed around the protagonist's story to inform the user about clean water problems in developing countries. To emphasize this notion, scenes are preceded by cut scenes which describe the character's story through a comic strip. The cut scenes not only describe the story, but also explains to the users the interactions needed to complete the level and what objects the user needs to collect in the level.

Design & Implementation Decisions:

Initially, the team was unsure of how to implement the graphics and animation for the application. Some team members researched Java Canvas class for animation, but after attempting to implement basic visuals with the class, it seemed to be a very raw method of drawing and animating, and would spend a lot more time working on the physics and collision detection than on the game itself. Other team members researched several different Android game engines, including Unity3D and AndEngine. While Unity3D is now free for mobile deployment and has a great GUI for development, the team did not have knowledge in 3D modeling, 3D animation, and creating the necessary textures for the game objects. In addition, Unity3D requires the publisher to purchase a $1,500 license if the publisher receives $100,000 in a fiscal year. As a result of the team's inexperience with 3D modeling, the decision was made to use a 2D model.

The goal was to find an open source 2D game engine that would allow future teams to continue development for the game without having to purchase expensive software. It was important for the game to have these five requirements:

1. The engine is able to draw and animate 2D graphic sprites easily.
2. The engine is able to deal with physics and collision detection in a trivial matter
3. The engine is able to easily to incorporate audio in the game
4. Easily traverse menus and “game scenes”
5. Implement simple player controls.

While initially, the team did not consider all of these to be of such importance, we found that utilizing an engine with these features would allow us to focus more on a quality product design and less on the complexity of our implementation.

Ultimately, the decision was made to use three external libraries that are not present in the default java or android library. We are using *AndEngine* as the main game development framework as well as the Box2D physics engine, *AndEnginePhysicsBox2DExtension*, to create the majority of our game. We choose to utilize this particular 2D game engine because it incorporates the 5 features previously listed, and is easier to learn and use than OpenGL ES API. It is also open source and has a large, active community, and is constantly receiving updates, which means future development is not an issue. There are also many tutorials available on the Internet as well as easy-to-read documentation. With the Box2D extension, we don't have to spend additional time to implement a physics engine.

We are also using ACRA to do bug reporting. This will allow us to collect crash data from users to help debug the app. We used this for bug reporting because it was simpler and easier to use then having to implement our own bug reporting system. When the application runs into an error or crashes, ACRA emails the java stack trace, the app version and Android OS version to the client so that the game can be improved and updated on the Google Play Store.

When deciding on how we should create the character and how we should animate her, we decided to use animation sprites which are basically digital flip-books. Sprites are easy to create and animate, and can be used across many devices—we don't have to worry about the graphics not displaying on various devices. After researching sprites and how to create them, we decided that we needed to have a minimum of six frames for one running sprite to ensure that the animation would be smooth during gameplay. The size and quality of icon graphics however, are predetermined requirements of Android such that the quality is consistent across devices of various resolutions.

Results:

The Android game was tested with several children between the age of 8 to 14, and on different android devices to ensure that the experience is thoroughly enjoyable, regardless of the device the user has. The game software ran smoothly on all tested devices, with high frame rates, and very responsive controls.

The first test subject was a 9-year-old girl, using a 4.2.2 Jelly Bean device. She thought the game was really fun and continuously tried to beat each level. The game occupied her for over half an hour, and she remained intensely focused on the game during that time. The swiping motions were extremely intuitive for her, and she picked them up almost immediately with very little instruction. She initially noticed that the game played like *Temple Run*, where you lose a life whenever you collide with obstacles, making it more challenging to master the level. She liked scenes 4 and 5 because they were challenging and fun, but she disliked the beginnings of those scenes because she kept getting stuck in the first quarter of the level. She also thought that if she beat the previous levels, that level 5 would be easier. Another issue she had with the game was that the long, falling platforms fell too quickly for her reaction time. She also suggested that having more than 3 hit points would make the game more enjoyable. Overall, she claimed that her experience with the game was positive.

The second test subject was a 14-year-old male, using a 4.2.2 Jelly Bean device. He said that the game was fun, but when asked if he would play it again, he said, “not quite” because it became frustrating for him, especially during scenes 4 and 5. He learned that the basis of the story was that the character was thirsty and had to search for clean drinking water. He said his favorite part was that the player could dash and jump long distances. During his time with the game, he said that the game was “intense,” and that it kept his attention. He disliked the first level because the square floors didn’t provide enough space. He was unable to complete the levels because of the difficulty; however, he did say he would share it with his friends.

The third test subject was an 11-year-old female, using a 4.2.2 Jelly Bean device. She enjoyed the game thoroughly, and said that she would play the game again (she did play the game again). She knew pretty quickly that she needed to reach the goal at the end of the level and collect objects along the way. She enjoyed the colors of the levels, character, and collectables. She also liked the menu music and the game music. She disliked that there was not yet a point score system, and that the version she played did not have a story. When asked if she would share it with her friends, she said “maybe.”

The last test subject was a 10-year-old male using a device with 4.0.4 Ice Cream Sandwich. He enjoyed the game, and said he would likely play it again. He learned that the character didn’t have clean water and wanted to build a well. He enjoyed that the character was a female, and thought that the character was cute. He didn’t like that the character would sometimes hit an obstacle and jump into a pit, causing a game over. He said that he would share the game with his friends.

What needs to be done in the future is to build the full version of the app and insert additional “acts” and “scenes” that the client has envisioned. This includes the addition of an upgradeable village that represents the increase in overall prosperity of developing countries due to clean water access. Also, the Box2D physics engine allows for vertices to be implemented for all sprites, meaning that polygon bodies can be utilized for better collision detection. There could also be improvements to the method in which levels and textures are loaded so that only the necessary resources are in use during each level. This would improve the overall performance of the game and allow bigger, more complex levels without lag. In addition, a leaderboard section would encourage users to play the game more to achieve a higher score. Lastly, the character should be able to perform a “power slide” as an additional ability for avoiding obstacles in further levels.

Appendix:

System requirements:

- Signed for Android version 17 (Jelly Bean), with a minimum version of 8 (Gingerbread)
- *AndEngine:*
    - Open source graphics engine for android located at: https://github.com/nicolasgramlich/AndEngine.git
        - Use branch: GLES2-AnchorCenter

- *AndEngine Physics Box 2D Extension:*
    - Open source physics extension for AndEngine located at: https://github.com/nicolasgramlich/AndEnginePhysicsBox2DExtension.git
        - Use branch: GLES2-AnchorCenter

    Both *AndEngine* and *AndEnginePhysicsBox2DExtension* have to be open in your working directory/workspace and be included in the project's library to use the API for the game development.

- *ACRA:*
    - Open source android bug reporting library should already be included in the project directory.
    - May need to added it as a library to your project

Coding conventions:

- All .java files have a header in the style of:

```
/**
* Authors: Chris Card, Dylan Chau, Maria Deslis, Dustin Liang, Gurpreet
Nanda, Tony Nguyen
* Date: 5/13/13
* Version: 1.0
* Description: Insert description of the file
*
* History:
*    5/31/13 original
*/
```

- All xml files have the have a header in the style of

```
<!-- Chris Card | This is act1 scene 2 level xml for the and
engine it defines what objects will be displayed for the level |
5/24/13 -->
```

- Singleton patterns for managers and the application class for context and consistent contexts