EMBER’S INKLINKO

Interactive Media and Game Development

Major Qualifying Project Report
Submitted to the faculty of
Worcester Polytechnic Institute, Worcester, MA
In partial fulfillment of the requirements for the
Bachelor of Science degree

In Cooperation with Eduardo Baraf, Studio Director
Disney Interactive, Palo Alto, CA

Submitted To:
Professor David Finkel (Advisor)

Submitted by:
Bryce Jassmond
Corinne Kennedy
Andrew Lukas
Christian Walker

Date of Submission: 30 March 2014

Advisor Signature
Abstract

*Ember’s Inklinko* is a casual iOS game designed in Unity at Disney Interactive in Silicon Valley. In eight distinct levels of increasing difficulty, players earn points by launching up to six balls through a field of gems and coins to reach the golden bucket. *Ember’s Inklinko* was developed as a WPI Major Qualifying Project by Interactive Media and Game Development (IMGD) and Computer Science (CS) majors. This report describes the design, development and analysis of that effort.
Acknowledgements

We are deeply grateful to our advisor, Professor David Finkel, whose tireless efforts among companies in Silicon Valley made possible this first Disney Interactive experience for WPI’s MQP students. In addition we appreciate the advice and counsel that he provided during both our PQP planning process and our on-site work in California.

We also sincerely thank Ed Baraf, Studio Director at Disney Interactive who was our project manager and whose industry knowledge and generosity of personal time and assets ensured us a successful project and product. We are also grateful to other Disney professional staff, such as Bennie Booysen and Nick Gallant, who helped bring our game to a professional status.
Authorship

Bryce Jassmond:
- Introduction
- Mechanics
- Playtesting

Corinne Kennedy:
- Art
- Literature Review/Background

Andrew Lukas:
- Pre-Qualifying Project
- Early Work
- Gameplay
- Analysis of Development

Christian Walker:
- Audio Design

Brooke White:
- Appendix D: Official Playtest Report

Anything not mentioned above specifically was written by the joint effort of Bryce Jassmond, Corinne Kennedy, Andrew Lukas, and Christian Walker.
# Table of Contents

Abstract .................................................................................. i
Acknowledgements ................................................................... ii
Authorship ............................................................................... iii
Table of Contents ...................................................................... iv
List of Figures ........................................................................... v
List of Tables ............................................................................ vi
Introduction ............................................................................... 1

## Literature Review/Background

- Casual Games ....................................................................... 2

## Project Procedure

- Pre-Qualifying Project ............................................................. 5

## Ember’s Inklinko

- Early Work ........................................................................... 6
- Gameplay ............................................................................... 6
- Art ........................................................................................ 9
- Mechanics ........................................................................... 18
- Level Design ........................................................................ 24
- Audio Design ....................................................................... 32

## Results and Discussion

- Playtesting .......................................................................... 34
- Analysis of Development .................................................... 35
- Conclusion ............................................................................ 36

## References ............................................................................. 37

## Appendices

- Appendix A: Software List ................................................... I
- Appendix B: Progress Reports ............................................. II
- Appendix C: Milestones ....................................................... XLI
- Appendix D: Official Playtest Report .................................... LII
List of Figures

Figure 1: Csikszentmihalyi’s Flow Theory ............................................................. 2
Figure 2: Coins used for guidance ......................................................................... 7
Figure 3: Coins used for just points ....................................................................... 7
Figure 4: Ember from *Hidden Worlds* ................................................................... 9
Figure 5: Ball texture with swirl ........................................................................... 10
Figure 6: Reflect Map ............................................................................................. 10
Figure 7: Coin close-up ......................................................................................... 11
Figure 8: Original gem cracking sprite sheet ....................................................... 12
Figure 9: Final gem cracking sprite sheet ............................................................. 12
Figure 10: Regular paint bucket sprite ................................................................. 13
Figure 11: Regular paint splatter sprite sheet ....................................................... 13
Figure 12: Bottom buckets with goal bucket’s lid closed ........................................ 13
Figure 13: Bottom buckets with goal bucket’s lid open ......................................... 13
Figure 14: Paint splatter sprite sheet for goal bucket ........................................... 14
Figure 15: Background – Chrona’s House ............................................................. 15
Figure 16: Level 1 with Chrona’s House background ........................................... 15
Figure 17: Original menu design ........................................................................... 15
Figure 18: Jiung’s Concept menu .......................................................................... 15
Figure 19: Final menu screen ................................................................................ 16
Figure 20: Confused Ember – 1 Star ...................................................................... 16
Figure 21: Neutral Ember – 2 Stars ...................................................................... 16
Figure 22: Happy Ember – 3 Stars ...................................................................... 17
Figure 23: Sad Ember – No Stars .......................................................................... 17
Figure 24: Level Handler main methods breakdown ........................................... 20
Figure 25: Preliminary test - Tightly packed gems ............................................... 25
Figure 26: Preliminary test - Thinly packed gems .................................................. 25
Figure 27: Final gem spacing design ..................................................................... 25
Figure 28: Preliminary - Large Gems .................................................................... 26
Figure 29: Final - Gems slightly larger than ball .................................................... 26
Figure 30: Rectangular, octagonal, triangular, and hexagonal gems .................... 26
Figure 31: Powerful shot ...................................................................................... 27
Figure 32: Weak shot ............................................................................................. 27
Figure 33: Level 1 - Instructional finger and ensured win ....................................... 29
Figure 34: Level 2 - Extra buckets and new gem pattern ..................................... 30
Figure 35: Level 3 - Red gem that opens the goal bucket’s lid when hit .............. 31
List of Tables

Table 1: Scoring.........................................................................................................................7
Table 2: Scoring in possible Level 3 play through.................................................................8
Introduction

This project and paper was completed by a team of four Worcester Polytechnic Institute seniors. The team consisted of two programmers and two artists. The programmers were Bryce Jassmond, majoring in Interactive Media and Game Development (IMGD) and Computer Science (CS), and Christian Walker, majoring in IMGD. The artists were Corinne Kennedy, majoring in IMGD, and Andrew Lukas, majoring in IMGD. This project was completed as the team’s major qualifying project.

The project was sponsored by Disney Interactive, specifically the mobile/social division. They are located in Palo Alto, California, and have more than 400 employees. They hired the team to work on a free-to-play casual mobile game prototype with the goal of having a potential $10 million in revenue if made into a complete game. There were to be about two months of pre-project planning done off-location and nine weeks of focused, on-location development.

The team arrived on-location with two game ideas after the pre-project period. At an initial meeting, elements from both ideas were taken and combined into one game design that is similar to games like Pachinko\(^1\), Peggle\(^2\), and Papa Pear Saga\(^3\). Disney’s intellectual property, from their recent game, Hidden Worlds\(^4\), was to be used in the game. The result was the game prototype that came to be Ember’s Inklinko, a game in which players tap or drag with their finger to aim and shoot balls at gems, bouncing the ball off them and racking up score, and trying to land at least one ball in a goal paint bucket at the bottom of the screen. Players are awarded stars based on whether or not they got a ball in the goal paint bucket and, if they did, their score. Following the meeting, the nine weeks of continuous development commenced, in which bi-weekly meetings kept the project on-track. At the end, a formal playtesting session was conducted in order to evaluate the project’s progress.

This paper discusses the process of how Ember’s Inklinko was developed. It first reviews the research conducted during the pre-project phase. It then discusses the early work and changes to initial game concepts. Next, it delves into details about the final game’s gameplay, the design of the art, the development of the background mechanics, the level design, and the audio creation. Finally, the results of the formal playtesting session are discussed and the overall development cycle is analyzed.

\(^1\) Pachinko. (2014, March 31).
Literature Review/Background

Casual Games

Before actually designing a casual game for Disney Interactive, we felt it appropriate to research the literature specific to that industry in order to better understand the market from the developer’s point of view. We also wanted to identify the factors that were critical to effective game design. The following highlights our research and findings.

Casual games are video game experiences that are entertaining, easy to learn and require little commitment of time. They often involve short, flashy sections of play that can be interrupted and resumed at will. For that reason, the casual game needs to be designed in a way that attracts the attention and interest of the player in the first few minutes of the game and should be structured as a series of levels that can reward the player immediately.\(^5\)

![Figure 1: Csikszentmihalyi’s Flow Theory](image)

It is also important that the game design keeps the player engaged in an area between anxiety and boredom as he or she progresses through increasingly difficult levels of the game. This is consistent with Flow Theory developed by psychologist, Mihaly Csikszentmihalyi.\(^6\) Figure 1 illustrates the optimal game flow between a player’s ability and the game’s level of difficulty. From a starting point of the game at position A1, if the difficulty increases beyond the player’s skill level, he or she will quit from

---


frustration. If the player’s skill level exceeds the level of difficulty, he or she will quit from boredom.

Casual games have shorter development cycles than other games. This is because casual games employ a few simple mechanics or concepts and generally have 2D, as opposed to 3D, graphics. For example, King.com’s famous game Candy Crush Saga\(^7\) is a match-three puzzle game. This means all the player has to do is align three or more candies of the same shape and color. There are not many graphics in the game aside from the user interface which shows the score and remaining moves, the candies the player has to match, simple background images, and the occasional animated sequence. A triple-A title, such as Bethesda’s The Elder Scrolls V: Skyrim\(^8\), has rich storylines, and detailed 3D graphics. Triple-A titles can take years to be completed, but a casual game can be conceptualized, developed and released within a few months to one year.

The casual game genre also has a different target audience from larger games. Traditionally the video games market has been primarily male-dominated, attracting young to middle-aged adults. In recent years, that market has been changing and now 45% of all gamers are women with the average gamer being 30 years old.\(^9\) Casual games specifically attract about 75% female players and 72% are over 35 years old.\(^10\) In addition to the fact that casual games are inexpensive, quick to play, and easy to pause, other reasons why casual games appeal to a female audience are they usually don’t feature the violence of the traditional video games targeted toward young men nor do they require the physical prowess of fast hand control to play.\(^11\)

Casual games often use visuals, pop-ups, and established conventions, such as stars or coins, to guide the player to the intended goals with minimal use of text. In the early levels of Where’s My Water?\(^12\), collectible rubber ducks are used to help indicate to the player where to send the water to reach the bathtub. According to our mentor at Disney, Ed Baraf, the best games are the ones that “show” rather than “tell” the player how to play.

These games also reward the player for success and usually do not present instantaneous loss conditions at the very beginning. In Farmville, players are given coins for taking care of crops and animals which can be used to purchase more crops

\(^{7}\) Candy Crush Saga. (2012, April 12).
\(^{8}\) The Elder Scrolls V: Skyrim. (2011, November 1). Bethesda.
or livestock. There is no way to lose in the game. If the crops die the player can simply replant, but there is no consequence for letting them wither.

Casual games need to have a low starting price as players tend to be cost-conscious. In order to address this, some companies will mask the real cost of virtual goods by first requiring the purchase of a premium currency such as gems or gold and then listing the cost of everything in that currency. Another monetization technique is dividing the game into two sections. The first part of the game consists of easier levels, but the player always has the option to pay for more challenging levels as well as more content. A common business model for casual games is free-to-play. Instead of being released in their entirety, they tend to have ongoing development, and go to market sometimes with only 50% of their final content. One of the challenges in designing free to play games is capturing the interest of the player in order to later open up his or her wallet. Then new features have to be introduced in order to keep the consumer engaged for extended periods of time. Newer features are often priced at a premium. These techniques allow casual games to be released at a low cost yet still make money.

With the advent of mobile devices, the casual games market has grown tremendously. Gaming on smartphones and tablets is dramatically increasing the time consumers have available to play games. It is projected that smartphone and tablet gaming will capture 27.8% of the global games market in 2016, an increase from 17.4% in 2013. This growth opportunity provides strong evidence that there is great potential for further game development in the mobile segment of the industry.

---

Project Procedure

Pre-Qualifying Project

In early November 2013 our project team began weekly planning meetings at WPI with our advisor, Professor Finkel who is responsible for the Silicon Valley MQPs. This was a team of four students representing both the art and technical aspects of Interactive Media and Game Development who were interested in experiencing an off-campus game development project at Disney Interactive.

The purpose of the planning meetings was to provide structure for designing a Disney-themed casual game prototype which if further developed by Disney would have a $10M revenue potential. While our initial research and design efforts focused on classic Disney movies with a target audience of young children, through both our research of casual games and information from our sponsor, we realized that the actual target audience for casual games is adult women.

Our PQP goal was to conceptualize several game ideas by mid-December 2013. We developed two different design concepts. The first was a marble puzzle game where the player would direct a marble to a goal by placing obstacles in its path to guide it. The second was a Break Out-style\(^{16}\) game where with a paddle at the bottom of the screen the player breaks a series of bricks.

In December we began having Skype meetings with our Disney project manager, Ed Baraf, who is a Studio Director at Disney Interactive in California. With his advice and counsel we refined our game concepts and gained a better understanding of what worked and didn’t work from a gameplay perspective. Neither of our first two concepts was actually developed. In the end, we combined elements of each and added other ideas.

Additionally, this pre-qualifying project helped with software planning. The game engine, Unity, was chosen because it is free and the whole team has had at least minimal experience with it. Autodesk Maya and Photoshop were primary art programs for three-dimensional modeling and two-dimensional images respectively. A full list of software used during this project is available in Appendix A: Software List.

Ember’s Inklinko

Early Work

Once we arrived in California, our initial assignment was to build a “toy”, a game that didn’t have specific goals, but instead demonstrated the physics of the engine and the problem solving skills of the programmers. The purpose of this activity was to demonstrate what our team was capable of designing. With a new IP, instead of classic Disney, and clarity of our target audience, we were given art assets from Disney’s very successful Facebook and mobile game Hidden Worlds. From those assets we quickly identified and adopted the key character for the game that we would create. The advice from our mentor was to get core mechanics to operate as solidly as possible and build from that. Biweekly meetings with our mentor allowed us to quickly move from concept to design. Milestones from the early meetings are in Appendix C: Milestones, and daily progress reports are in Appendix B: Progress Reports.

Gameplay

Ember’s Inklinko is a Pachinko style game similar to Peggle and is designed to be played on mobile iOS devices. The game uses an elfin character named Ember from Disney’s Hidden Worlds to launch six balls into play from the top of the screen. The objective is to bounce a ball into the goal, a gold paint bucket, at the bottom of the screen which when accomplished, advances the player to a subsequent level. In addition, the player wants to earn as many points as possible in each level.

Points are earned in several ways. The player is awarded 500 points for each ball that goes into the goal bucket. In addition there are several other buckets at the bottom of the screen which award lesser point values when balls fall into them. As each ball falls, it bounces off red, blue or grey gems on the way to the buckets at the bottom of the screen. When multiple blue gems are struck, a multiplier of 0.1 is applied to the score; this is a combo bonus.

In all levels after the second level, the goal paint bucket is lidded, preventing the player from immediately reaching the goal. In order to open the bucket, the player must clear all of the red gems in the level by striking them with a ball. Exploded red gems give 100 points and add another 25 to the combo bonus. All red gems must be cleared, in addition to hitting the goal bucket, in order to advance to the next level.

Striking each blue gem with the ball awards 50 points, but unlike red gems, clearing them is not a requirement to beat a level. Grey gems cannot be broken and do not give
the player points. They behave as obstacles for the ball to bounce off of. In some cases not all six balls are used in a level because the goal bucket has been hit before they were launched. Five hundred points are also awarded for each ball is saved in this manner. This is called the ball bonus and is a reward for accuracy and efficiency.

![Figure 2: Coins used for guidance](image)

![Figure 3: Coins used for just points](image)

Gold coins appear in every level and are worth 250 points each to the player when collected. In early levels, they are aligned in a way that guides the player to aim the ball launch (Figure 2). In subsequent levels, the coins appear among the arrangement of gems, no longer guiding the player’s shots, but still awarding points (Figure 3).

The points awarded by all buckets, including the goal bucket are added up into the “end bonus.” At the end of each level, stars are awarded to the player based on whether the goal bucket has been reached and on the total score for the level with a maximum of three gold stars being awarded. The score is calculated from the base number of points for each gem cleared (50 for blue and 100 for red) plus the combo bonus plus the ball bonus. Table 1 summarizes the scoring rules, and Table 2 provides an example of how a score is calculated for a specific level 3 play through.

### Table 1: Scoring

<table>
<thead>
<tr>
<th>Basic elements</th>
<th>Points</th>
<th>Combo points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal (gold paint) bucket</td>
<td>500 per ball landed</td>
<td></td>
</tr>
<tr>
<td>Pink paint bucket</td>
<td>400 per ball landed</td>
<td></td>
</tr>
<tr>
<td>Green paint bucket</td>
<td>100 per ball landed</td>
<td></td>
</tr>
<tr>
<td>Blue paint bucket</td>
<td>200 per ball landed</td>
<td></td>
</tr>
<tr>
<td>Blue gem</td>
<td>50 for each strike</td>
<td>Adds 0.1 to multiplier</td>
</tr>
<tr>
<td>Red gem</td>
<td>100 for each strike</td>
<td>Adds 0.5 to multiplier</td>
</tr>
<tr>
<td>Coins</td>
<td>250 each collected</td>
<td></td>
</tr>
<tr>
<td>Extra Ball Bonus</td>
<td>500 each</td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td>Action (or per item value)</td>
<td>Score</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>Ball 1</td>
<td>Collects 3 coins @250</td>
<td>750</td>
</tr>
<tr>
<td>Strikes 1 red gem – opens goal bucket</td>
<td></td>
<td>100 + adds .5 to multiplier</td>
</tr>
<tr>
<td>Strikes 7 blue gems @50</td>
<td>Activates multiplier</td>
<td>350</td>
</tr>
<tr>
<td></td>
<td>gem 1 150x1.5 =75-50 =25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>gem 2 50x1.6=80-50=30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>gem 3 50x1.7=85-50=35</td>
<td></td>
</tr>
<tr>
<td></td>
<td>gem 4 50x1.8=90-50=40</td>
<td></td>
</tr>
<tr>
<td></td>
<td>gem 5 50x1.9=95-50=45</td>
<td></td>
</tr>
<tr>
<td></td>
<td>gem 6 50x2.0=100-50=50</td>
<td></td>
</tr>
<tr>
<td></td>
<td>gem 7 50x2.1=105-50=55</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sum of multiplier</td>
<td>280</td>
</tr>
<tr>
<td></td>
<td>Lands in red bucket</td>
<td>300</td>
</tr>
<tr>
<td>Ball 2</td>
<td>Collects 3 coins @250</td>
<td>750</td>
</tr>
<tr>
<td></td>
<td>Lands in Goal bucket</td>
<td>500</td>
</tr>
<tr>
<td>Balls 3,4,5,6 (Ball bonus)</td>
<td>Balls not used 4@500 each</td>
<td>2000</td>
</tr>
<tr>
<td>Total for Level 3</td>
<td></td>
<td>5030</td>
</tr>
</tbody>
</table>

The player aims by dragging with his or her finger across the touch screen. When the player lifts his or her finger a ball is launched into the game field from Ember’s slingshot. Once a ball is released, the player cannot change its trajectory directly. The game allows multiple balls to be in play simultaneously. This allows the player to increase combo bonus, bounce balls off of one another to adjust their direction, or clear multiple red gems at once. This also allows for a measure of strategy as the player can clear a red gem while another ball moves toward the gold paint bucket or can clear a path to the gold bucket with one ball and reach it with another.

The gameplay is currently at a complete prototype stage. There are not chunks of the game missing, and the game itself is mostly self-guided. Players can interact with the game and receive a complete experience from it, playing through levels and trying to achieve a high score with very few errors that break the game.
Art

Art Direction

The art of Ember’s Inklingo is done in a painterly style, one where the forms are made of color rather than outlines to create the forms. The overall design is based on Hidden Worlds, a recent game by Disney Mobile released on Android, iOS, and Facebook. In Hidden Worlds, the character Ember uses a slingshot to craft in-game items (Figure 4).

![Figure 4: Ember from Hidden Worlds](image)

Our game required a launcher, a method of dropping the ball into the level, and Ember’s slingshot was exactly what we needed. One of the lead artists for Hidden Worlds, Bennie Booysen, gave us access to many assets from their game, Ember included, to use in our own, and Ember became the main character in our game. The style is reminiscent of classic Disney movies.

Dynamic Art Assets

Most of the art assets used in the game are 2D sprites. A sprite is a simple 2D image displayed on a plane, usually with a transparent background. Sprites are often laid out in a grid on a single image file and can be used for animations; this is referred to as a sprite sheet. With sprite sheets, the game engine can go through the arranged list of sprites and either display a desired sprite or play them in succession to animate the image. The only 3D assets used in the game are the ball and the collectible coins.

The 3D ball consists of three separate art assets: the spherical model, called a mesh, a texture file, and a reflection map, or reflect map. The sphere was made in the 3D modeling program Autodesk Maya and then exported from there as an object file,
which is a standard file extension for 3D objects. The object file was loaded into Unity, along with the 2D texture file, the color of the ball (shown in Figure 5), and the reflect map, shown in Figure 6. The texture and the reflect map were applied as a material, which itself was wrapped around the sphere mesh. The swirl made the movement and roll of the ball more obvious; the swirling motion draws the player’s eye to the ball after it has been launched so the player can track its progress.

![Figure 5: Ball texture with swirl](image1)

![Figure 6: Reflect Map](image2)

The ball uses Unity’s pre-existing physics engine, which allows for more realistic bouncing and makes the game easier to code. Having the ball as a 3D object also allows it to take advantage of Unity’s lighting engine. This way, when the ball bounces, rolls, and rotates, the highlights and shadows function naturally in relation to the light rather than being part of the texture and rolling when the ball rolls in an unnatural manor or needing to be a separate sprite. The ball’s reflect map allows it to appear shiny.

The reflect map is a default texture from the 3D modeling program, *Pixologic ZBrush*, and is a photo of a desert that has been set to grayscale and warped to appear spherical. The sun in the image becomes a strong highlight and the desert floor becomes areas of shadow. In addition to making the ball appear shiny, the reflect map also gives the ball a greater illusion of depth.

The coins in Figure 7, like the ball, also have a mesh and texture but did not require a reflect map. The coins are 3D to make the rotation animation easier; having actual depth to begin with means they just had to be programmed to spin when implemented rather than being drawn out frame by frame as the perspective changes. This also allows the coin to rotate smoothly and attract the player’s eye.
There were several advantages to using sprites instead of 3D objects for the majority of the objects in our game. First, using sprites allowed us to create objects that have different states and display state changes visually (e.g., a gem cracking when hit). Second, 2D sprite animations, such as Ember firing the ball, were also easier to create than equivalent 3D animations. 2D assets are significantly less memory-intensive than 3D assets and they can be colored by the Unity engine, which will be discussed later. The gems, for example, are 2D sprites, but they have 3D colliders, an invisible 3D mesh that can tell the engine if another 3D mesh has entered its space, to allow them to interact with the ball. Starting out as 3D objects, the gems took up too much memory when a significant number of them were in a scene. By making them into 2D objects we drastically reduced the memory usage of the game.

We wanted to have the option to show the difference between gems with varying health, for example a gem that will shatter in one hit as opposed to a gem that will shatter in three hits. This was done by making a sprite sheet with the gem cracked to varying degrees, shown in Figure 8. The gem on the left is undamaged and will take more hits to break than the gem on the right, which will break the next time it is hit by the ball. It is more efficient to show these state changes with sprites because only one object, the gem cracking sprite sheet, is required. This one sprite sheet is less memory intensive than a 3D model and four separate texture files. It proved to be more difficult to dynamically switch textures on a gem model to show state change because changing the texture of one gem in a level often changed all of them.
We wanted the gems to have different colors. Since the sprites could be assigned a color in the engine, only one sprite of each of the two shapes, octagonal and rectangular, needed to be made rather than several sprites of different colors or textures for the models. This came with a challenge of its own: Unity assigns color additively, which means it takes the color you want the sprite to become and layers it over the colors that already exist in the sprite. If we had a blue gem and wanted to change it to red, it would turn purple. We solved this problem by making the gem sprites predominantly white and using grays to add shading. We had some help from another Disney artist who worked under Bennie to make the gems match the artistic style of our game (Figure 9).

![Figure 9: Final gem cracking sprite sheet](image)

**The User Interface**

For *Ember’s Inklinko*, we needed to do something with the bottom of the screen. It used to be purely ball death; if all the balls launched reached the bottom without hitting the goal zone, the player would lose. Ed recommended we make it such that reaching the bottom of the screen is less penalizing. We decided to implement bonus points in four zones at the bottom and make a fifth zone the goal. To represent these zones, we used buckets from the crafting menu in *Hidden Worlds*. Memory space was saved by having one bucket of paint (Figure 10), duplicating it, and assigning it a color, rather than making many paint buckets.

The paint in the paint buckets at the bottom of the screen and the splatters (Figure 11) that appear when a bucket has been hit are white sprites, like the gems, and are likewise colored by the engine.
The goal bucket needed to be unique and draw the player’s eye. It needed to be immediately apparent to the player that the bucket is the goal and whether it was accessible or inaccessible because of red gems in the level (as indicated on its side). It sits on top of a stool to give it some height variance from the other bonus point buckets (Figure 12), making it even more obvious that the goal bucket is important, even when the bucket is lidded and inaccessible.

The goal bucket has a shine that is absent from the other buckets. This informs the player that the goal bucket is now open and reachable, shown in Figure 13. The shine on the bucket has a very simple animation to make it seem as if the shining light is spinning, which also helps to draw the player’s eye to the goal bucket.
This lets the player know how many red gems need to be broken before the bucket is opened. The goal bucket’s gold paint splatters, shown in Figure 14, are unique in that they have a subtle glitter texture to them, which makes it clear that the goal has been reached.

![Figure 14: Paint splatter sprite sheet for goal bucket](image)

*Backgrounds*

The backgrounds (such as in Figure 15) were pieced together from different assets in *Hidden Worlds* by Bennie. They all have a large area of sky with a little bit of Inkspire, the island the Inklings live on in *Hidden Worlds*. These backgrounds allow the gems to pop on the sky section while grounding the whole scene with the island at the bottom. The large open area of sky keeps the background from being too cluttered so the player is more aware of the gems than the background while tying together the look of the whole game (Figure 16).
The main menu and end screens were based on some concept pieces by Jiung, another artist working under Bennie. She took our original menu screen (Figure 17) and gave us a concept piece that better represented the game and matched the art style (Figure 18). We took that concept and made a few adjustments (Figure 19). The ball was changed to match the one used in the game, and different button textures and animation were added. For the end screen, we chose to emulate one of Jiung’s designs: just Ember, the score, and the stars are displayed on a black overlay, semi-transparent layer of color, over the level screen.
Getting the animations for the end screen proved difficult; we were given four animations of Ember making faces, seen in Figure 20, Figure 21, Figure 22, and Figure 23, as Adobe Flash documents. These animations add character to the game and make the player feel either really happy about winning or sad about losing in a way that encourages replaying the level. However, Unity does not accept Flash animations as valid assets. We had to convert the animations to sprite sheets using a program called SWFSheet. Since each animation had over 100 frames, the game used too much memory and crashed when it was loaded. To solve this problem, the size and quality of these animations was drastically reduced.
We wanted to use stars to show how well a player did on any given level. This is a convention shared by many casual games including *Hidden Worlds*. Simply completing the level grants the player the first star, like in Figure 20, and the other two will fill based on how well the player scored (compared to a calculated threshold). In Figure 21, the player beat the level and got enough points to reach the first threshold but did not get the high score. Therefore, the player only has two full stars. In Figure 22, the player has achieved, or exceeded, the high score threshold for the level and was awarded three stars. The player did not reach the goal bucket in Figure 23 and earned zero stars.

We were unable to get the already animated stars from *Hidden Worlds*, although we did get a still image. Instead, we animated the filling in of a star in Flash by layering a rectangle over the color of the star and animating the rectangle to uncover the star. We then converted the flash animation to a sprite sheet and deleted the rectangle. This gave us a sprite sheet with the color appearing gradually. Next, we put that over a wooden star-shaped frame, thereby making the frame appear to fill with color. However, the sprite sheet stars caused a problem in the game; Unity can only display one sprite from the sheet at a time. Since the stars fill based on score, falling just short of the threshold for achieving a full star will occasionally cause it to display as full. To avoid confusion, a shine behind filled stars was added.

**Summary**

The assets came together at the end to form the cohesive look of *Ember’s Inklinko*. Everything in the game, from the UI to the gems, feels like they are from the same world and the same game. Our technical adjustments helped make the game look its best and work well with the engine and the code.
Mechanics

*Ember’s Inklino* has many background mechanics that are not apparent through gameplay, used both for development as well as in-game. As new needs arose, they were iterated and changed until they worked as desired. These mechanics include the physics implementation, level handler, custom buttons, audio handler, scoring objects structure, and modular gems.

Physics

This game primarily used built-in, realistic physics functionality. By using *Unity*, the physics engine *PhysX* by *Nvidia* was included by default. This included basic rigid body physics with both three dimensional and two dimensional collisions as well as indicating triggers, colliders that do not automatically affect movement. While the game mostly consists of two dimensional sprites, early implementations used three dimensional meshes and colliders, which proved difficult to change in the project’s time frame when the sprites were introduced; thus, the mesh colliders were kept for the sprites, which were essentially of the same shape. For several elements that were rectangular, such as the invisible walls around the levels, built-in box colliders were used to reduce calculations needed for comparatively complex meshes. The final colliders used were planar meshes, used sparingly for kinetic interactions but mostly for user input, such as buttons. All of these are tools built-in that provide a realistic physics simulation.

To create a more fun experience, creative changes and additions were made to the built-in physics functionality. The ball physics in particular needed many iterations to generate a desired feel from its bounces. Gravity strength and the physics material bounciness given to the ball were continuously edited in an effort to find that desired feeling, but ultimately, an artificial clamp on the maximum speed of the ball was what achieved it. The ball launcher, Ember’s slingshot, that shot the ball also included a custom trajectory calculation that showed exactly where the ball would be in small time steps, taking the artificial maximum speed into account. This enabled a player to aim the ball prior to shooting it. These built-in realistic physics combined with our creative additions gave the game a fun and understandable experience.

Level Handler

Each level is controlled in the background by a level handler so it may run smoothly. When a new level is loaded, the level handler initializes right after the audio handler. While doing so, it grabs and saves each key element in the scene, such as the audio handler, camera, and launcher, and creates other parts, such as a temporary title banner and score tracker. For objects where it is appropriate, the handler calls their initialize functions in a specified order, having each one start when they need to. If
there was a problem or missing item earlier in the sequence, such as a missing audio handler, the level would throw a custom error and break out of the level, letting the developers know what needs to be fixed. Once the level is fully initialized, the handler continues to manage all the objects by a smart update, which occurs a number of times per second, based on frame rate, to keep the game running; not only do the objects update in an appropriate order to prevent race conditions, but certain object updates may be completely skipped if the game is paused or the level has ended, saving on computation time. A detailed order of operations within the level handler is available in the flowchart of Figure 24. This control over the operations of the level and its objects allows the game to run smoothly.

The level handler also reduced the development and running cost of scripts. In Unity, scripts are attached to an object in order to run, unless they are static and utilized by other scripts. Initially, all objects were independent, and whenever one's script needed to access information from another object, such as the buckets needing to know the active ball count and shots remaining from the launcher in order to trigger the level's end, the object would have to search the scene for the one holding the desired information. To combat this, a static storage script was created, which allowed objects to access all desired information without searching. The downside to this static storage script was the size and inflexibility of it; any addition to the game that required shared information was added to it, making finding a particular part that needed editing difficult. The level handler provided a cleaner solution, letting each script know only as much as it needed to know, which limited the use of public variables and costly searches as well as made scripts concise and compact. In cases where an object's script needed to know information from another source, such as the audio handler to play a sound or the level's score tracker to add to, the script could access the information straight from the level handler, which keeps track of all important assets in the level. The organization and optimizations of the scripts were made possible only by the level handler.
Figure 24: Level Handler main methods breakdown
Custom Buttons

Customized buttons were the most efficient option for Ember’s Inklinko. While there are built-in buttons and alternative solutions that may be purchased, making ones to fit the game’s needs was found to be both more efficient than the built-in ones, which ran draw functions several times within single frame, and more cost effective than the purchasable add-ons, favoring time over monetary cost as the game’s budget was nothing. The custom button was structured such that actions may be plugged in for various user inputs, such as press, drag, and release, as long as they adhere to a base set of rules: the action script needed to inherit from a base action script and, in order to do something unique, needed to redefine the base script’s initialization and Action functions. This made them modular to accommodate all necessary functionality. This modularity and cost effectiveness made custom buttons the most efficient choice for the game’s user interface.

The button acts like one might expect a button to in a game. If given a graphical object, the button will demonstrate interaction by shrinking the graphic when pressed and held down, and return it to the original scale when released. Additionally, a sound is played whenever the button is pressed and released. For most basic buttons, all behavior is applied by assigning an action script to only the released action slot, but there are slots for actions on press and drag. How the actions are implemented will be discussed in the following paragraphs, but the scaling, sounds, and actions on user input are all features commonly found in game buttons.

One of the simplest buttons in the game is the activation button. When given an object, it will toggle the object’s active state, causing the object to disappear and become unable to be interacted with. This is used for the level select window and credits button on the main menu. The pause action is similar to the activation button, but uses a level’s pause menu instead of a given object as well as toggling the level handler’s pause. This is used on the options button in a level.

A common action, level changes, are handled in several different ways. The general level select action loads a level by a number given to it, but if given a text object with an integer string value, it will load the level interpreted by the string. To assist the level select action that is given a text object, an iterator action button increments or decrements a count that, when given a text object, is parsed to the object’s string value. These are used in the level select window on the main menu. Variations of the level select strip down the functionality to only the necessary parts; the retry action reloads the current level, the main menu action loads the main menu, and the next level loads the next level, defaulting to the main menu if there are no more levels. These are used on the respective buttons on the options menu and stats screen in levels.
An interesting button action is the skip stats action. This is only used on the stat screen, and will cause the stat sequence to skip to the end by displaying the total score and triggering each star to fill up its required amount. This action is applied to the press slots of the stat screen buttons, but also to the pressed action of an invisible button that overlays the entire stat screen, allowing the player to skip the stats by tapping anywhere. For just this invisible button, the action additionally turns off the press and release sounds.

The most interesting button of all is the ball launcher. This button is invisible, like the stat screen button, and covers the whole screen except for the user interface at the top. The button has two actions, one for aiming and one for firing. The aim action runs both when pressed down and when dragging, telling the launcher to run its aim function, which turns Ember’s slingshot to face the player’s finger and shows the trajectory of a fired ball. The firing action runs when released, firing a ball in the direction the slingshot is facing. Both actions replace the button sounds with slingshot press and release sounds in their respective slots to fully make the player feel they are using a slingshot.

Custom buttons were the best choice for this game. They were the most efficient choice, in time and money, out of the available options, and were modular enough to accommodate all user interactions, taking the basic presses and releases and acting appropriately on each one. Without them, development would have been slower.

Audio Handler

After several iterations of an audio system, it eventually consolidated into one, centralized audio handler. Initially, each object had sounds attached by default, which proved inefficient both for development and during play. If a new sound were to be tested, it would need to be replaced for each object by hand, and since the sources always existed during play, resources were taken up by them. Changing the sound clips to variables, to be played from one audio player, improved the resource usage during play, but did not fix the development and testing cost; even with a script to assign the sound effects at runtime, it was time consuming to change the sounds, as well as messy to assign in practice. Finally, a central library was formed with a dictionary of sounds keyed with strings. This allowed scripts to use a hard-coded string reference to a sound and sound effects not only be differently named, but also switched in only one location, vastly improving development time. An additional improvement to this system was the ability to instantiate an audio source only as long as a clip was playing, which freed up memory and allowed each sound to have adjusted volume and pitches. At the slight cost of reallocating memory frequently at runtime, this gave flexibility to how audio was used, and would not be possible without the centralized audio handler.
Structure to Scoring Objects

*Ember’s Inklinko* has numerous objects that affect score; gems increment score when hit and increase in value from an increasing combo, coins award a constant score when touched, buckets give scores based on their labelled value, and extra balls award a constant value at the end of a level. They are all different, but share similar features, so a basic inheritance protocol was needed to give them structure. They inherit methods and parameters that any object involved in scoring needed to know about, so scripts can assume general properties. For example, by inheriting from a base class, the level handler can easily track all of them after a single search while initializing, as shown in Figure 24. Additionally, as art was constantly changing alongside code development, general terminology, such as “launcher” and “ballEndZone”, let assets be swapped out with minimal effort. Giving much needed structure, the inheritance protocol for scoring objects was important for the game’s development.

Modular Gems

Extending the structure of scoring objects, gems in particular were very modular. The developers could change a gems strength, breakability, key status, and combo increase amount without delving into the gem’s script. As a core element to the game since the beginning, this gem modularity allowed the gems to be easily changed and tested, which led to quicker level design. A fine example of how art and audio work together, the gems begin as white and colorless, waiting for a script to assign them color based on the previously discussed properties when they are initialized. In previous versions of the game, they were additionally assigned sound effects based on the same properties, but this was taken out with the implementation of the audio handler. In this final version, without having to worry about running down a list of preset gems or changing all of said presets when an asset was changed in earlier versions, designers could move a single base gem into a level and change a few simple properties to vastly change its behavior without much effort. For these development needs, the gems needed to be as modular as they were.

Summary

All of these background mechanics were necessary for the game, even though they are not readily apparent to a player. They kept the game’s assets organized neatly and running efficiently, keeping the game fun to play. Lack of any of those parts would have made development slower and hindered the design process.
Level Design

Level design is the part of the game development cycle where the physical setting and the elements with which the player interacts are designed. The level designer’s role is to create a fun and engaging experience for the player by skillfully manipulating the various elements of level design. In the case of Ember’s Inklinko, those elements include gems, balls, coins and paint buckets that are integral components of the game.

Several considerations were evaluated in creating the action of the game. They included the density, size and shape of the gems; the breakable vs. unbreakable elements; the speed and control of the ball; and the goal placement. Then it became the challenge in the design of specific levels to ensure that the arrangement and interaction of the elements appropriately evolved in complexity in each game level while the player’s skill level advanced to ensure interest, engagement, challenge and fun. Discussion of each of these aspects of level design follows.

Gems

Density

How densely or loosely packed the gems were arranged affected the play of the game. Using level designs from preliminary builds with gems densely spaced, we realized that when gems were placed close together, the movement of the ball was impeded (Figure 25). This made the ball action rather boring to watch. When placed too far apart, there was not enough “bouncing” or interaction between the ball and the gems and we risked losing the interest of the player (Figure 26). The optimal spacing between gems in the final design was found to be roughly a gem’s width as shown in Figure 27.
Size

The size of the gems also affected play. If the gems were too large, they effectively cluttered the screen and actually interfered with play (Figure 28). If they were too small, they were difficult to see and interact with. We found that the best gameplay was achieved when the gems were slightly larger than the ball (Figure 29) and the action then of striking the gem was a more satisfying experience for the player.
Shape

The shape of the gems was also an important level design consideration. The game requires that the ball is launched in a way that allows it to bounce off the gems. Rectangular and octagonal shapes (Figure 30) provided the best surfaces for a predictable “bounce” of the ball, allowing those gems to “explode” on impact. Other gem shapes, such as triangles and hexagons (Figure 30), were considered but rejected because after a few bounces off them, predicting the ball’s motion and direction became too chaotic. This was likely a function of the difference in the degrees of angle among the shapes.

![Rectangular, octagonal, triangular, and hexagonal gems](image)

Breakable vs. Unbreakable

Consideration had to be given to the interaction of the ball with the gems to ensure that there was adequate visual action to optimize gameplay and create an enjoyable experience. Effective level design required that there be ample numbers of breakable gems in order to produce an element of action and sense of accomplishment on the
part of the player. Red and blue gems in the game were the breakable elements which created exploding action and added excitement to the game. The gray gems were unbreakable but interacted with the ball to alter and sometimes guide its direction.

**Ball**

The speed of the ball was affected by gravity and the game engine’s built in “bounciness.” With a higher speed, the player could not follow the ball’s progress through the gems. At a slower speed we risked having the player become bored and leave the game. Even at slower initial speeds, the more the ball bounced, the faster it accelerated. This required assigning a speed cap in order to keep the ball at a reasonable velocity. If action was too bouncy, the ball’s movement was chaotic and difficult to follow or predict; not bouncy enough and the ball felt lethargic so the game wasn’t exciting or engaging.

The player affects the movement of the ball by controlling when it is fired, the angle at which it is fired, and the power of the shot. In Figure 31 the player is aiming down to the left with a powerful shot as indicated by the placement of the trajectory “dots.” In Figure 32 the player is aiming down and to the right with a weak shot where the trajectory “dots” indicate that the ball is likely to fall with minimal power behind it.

![Figure 31: Powerful shot](image1.png) ![Figure 32: Weak shot](image2.png)

There is some predictability of a ball’s action when the player understands the relationship between the ball and the shapes of the gems. An experienced player can fire one ball and have it bounce off gems or walls to direct it to its desired target.

**Coins**

Gold coins are other elements in the game. They serve two purposes. Sometimes they direct the player to important targets, such as red gems. In every case, coins serve to award points to the player when a ball passes over them while in play. Points earned from the collection of coins offers the player additional ways to maximize their score and experience a sense of success.
Buckets

During preliminary game design we realized that we needed to have some type of receptacle for a launched ball. Having a ball just “die” at the bottom of the screen was not a particularly rewarding outcome. A bucket became the ideal target toward which to fire a ball. It was shortly after this design decision that we were offered the opportunity to use art assets from Disney’s *Hidden Worlds* and several buckets became part of *Ember’s Inklindo*.

The first bucket that is introduced in the game is a gold paint bucket. It becomes the primary target of the game, the goal bucket. For each level of play the only way to advance is to hit the goal bucket. In the first level of *Ember’s Inklindo*, there is a single bucket, the gold goal paint bucket, placed at the bottom center of the screen. Its solitary and central placement signifies its importance to the player.

Four additional buckets, green, pink, red and blue, are added to all the subsequent game levels, where reaching the goal bucket becomes more challenging. Each of the four has a lesser point value, but hitting them prevents the player from feeling that a ball was wasted even if they missed what they were aiming for. This serves as another way of keeping the player engaged.

For all but one level, the goal bucket is positioned in the bottom center of the screen. At Level 3 it is moved to the far right where the vertical arrangement of unbreakable gray gems and the positioning of coins advise the player that there is a required sequence of play in order to overcome an additional challenge, opening a lidded goal bucket. An important consideration in level design was to create incremental levels of difficulty to maintain the interest of the player. Introducing a lidded bucket for levels 3 through 8 as well as a specific sequence in order to open the lidded bucket introduced the necessary complexity and play strategy to ensure engagement and a fun experience.

Trial and Error

The optimal layout of the elements with which a player interacts requires a certain amount of trial and error. In an article describing the design of *Peggle 2*, author Russ Pitts reinforces the fact that there are no hard and fast rules in level design; it is up to the level designer to arrange the elements in ways that allow for exciting and engaging game play.¹⁷

Designing the Levels

Level 1

In level design it is essential to engage the player’s interest with the very first game experience so that he or she will continue. This means visually providing clear directions to get the player started as quickly as possible. Level design considerations on the first game screen involved showing the player where to start. An animated hand with a pointing finger directs the player to the slingshot and the six game balls (Figure 33).

In addition the alignment of three gold coins, which are traditional elements in casual game design, guides the player to an effective aim trajectory. In Level 1 there is only one paint bucket at the bottom of the screen, the goal bucket. Its singular placement at the center makes it obvious to the player that this is his or her desired target. The balls, once launched, strike the blue octagonal gems and are directed toward the goal bucket by the funnel-like arrangement of unbreakable gray gems and land in the bucket. The player is immediately rewarded with points for striking the gems and hitting the goal and advances to the next level.

![Figure 33: Level 1 - Instructional finger and ensured win](image)

Level 2

Effective level design requires a balance of challenges that are difficult enough to continue to engage the player’s attention, but are not so difficult that the player will give up.
Level 2 introduces additional paint buckets of different colors and point values at the bottom of the game screen. The title of Level 2, “More Buckets?” appears at the top of the screen to indicate to the player that there are additional reward opportunities (Figure 34). Rectangular gems placed in a wavy pattern offer the player a new challenge.

![Figure 34: Level 2 - Extra buckets and new gem pattern](image)

Level 3

Game complexity increases as red gems are introduced in Level 3, which is titled “Red Gems are Key.” This title provides an important clue to the player that striking the red gem is a critical aspect of the game. In addition there are gray gems that are now aligned in a vertical pattern and the goal paint bucket has been moved to the far right. Two arrangements of gold coins indicate that there are now two areas toward which the player should aim (Figure 35).
Subsequent Levels

The first 3 levels introduce all of the elements that the player will encounter while advancing through the eight levels of play. With each successive level of play we have built increasingly challenging gameplay while still using all of the now familiar elements. The more complex arrangement of those elements and the player’s increasingly skilled interaction with them in subsequent levels creates excitement, challenge and fun.

Summary

Level design involves attending to a number of details in the development of a game. The gameplay elements require thoughtful planning and integration in order to ensure a gamer’s engagement. With *Ember’s Inklino* we believe that we have done well in attending to all of those factors.
Audio Design

Audio design is the process of selecting, acquiring, manipulating and generating audio elements. It is the creation of a complete soundscape; the music, sound effects, and ambient noises that make up the game’s aural experience. It is used to set the mood of a game, evoke emotion and otherwise immerse a player in the game world.

A large part of the sound design process of Ember’s Inklinko was obtaining or synthesizing sounds to match the soundscape and the universe of Hidden Worlds. Our audio assets were acquired from either Freesound.org or Nick Gallant, an audio professional at Disney. Some were synthesized in an audio editing and production tool known as Audacity with the help of external filters.

The soundscape of Hidden Worlds uses percussion and woodwind instruments to create a primal and mystical sound. The sounds selected tended to have a slower tempo, and were high on acoustic effects such as reverberation, or echo. Hidden Worlds lacked synthesized sounds or metallic instruments, creating a soundscape evoking nature and tribal civilizations. The soundscape of Hidden Worlds avoids aggressive bass and drum-heavy pieces that would not fit with the whimsical, playful and mischievous nature of the Inklings.

There were also technical considerations that factored into audio decisions. Nick told us that the iPad’s speakers render high-pitched sounds as perceptibly much louder than intended. These technical limitations of the iPad combined with human perceptual differences of tonal highness or lowness, known as pitch, required drastic reductions in objective amplitude, or loudness, for certain sounds such as the coin, to achieve comparable subjective volume to other sound assets.

Considerable effort was also expended in cleaning up and making the sounds consistent, a process called mastering. Audio elements needed different techniques to make them sound like they came from the same environment. For example, a few sounds had very loud and very soft portions that needed to be made less extreme. The technique to make the loud and soft portions less extreme is called compression.

Dynamic range compression, the technique used to process these sounds, involves increasing the volume of sounds below a certain loudness threshold and reducing the volume of sounds above another threshold. The thresholds are set independently for each sound, as the human ear perceives a sound’s volume dependent on its frequency,
and around the ~3000 Hz frequency range a sound is perceived to be much louder than a sound that is higher or lower pitched.\(^\text{18}\)

Many mobile devices have a frequency cutoff that truncates the lowest part of the spectrum where bass would sound louder. Some sounds with considerable bass either had the very low and high ends of their frequency spectrum cut off or, due to the nature of lower-pitched sounds and aforementioned iPad frequency playback limitations, simply did not feel loud enough even when compressed and then tuned to a reasonable maximum amplitude. The sounds that did not level well needed to be pitch-shifted upward. This is generally set to -1 decibels for most audio, as it is well above the point where humans can start to perceive a sound, or hearing threshold, but well below the threshold of pain where a sound starts to become painful to listen to at any frequency.\(^\text{19}\)

Although we were already aware of these techniques, Nick was able to guide us as to when each form of volume leveling was superior. For example, he commented since the sounds for breakable bricks sounded too soft but had very drastic differences in volume from one part of the sound to another, compression would likely be needed to make the whole sound more audible. He also assisted with providing high-quality vocalizations and music from *Hidden Worlds*, and some assets from another game he has worked on based on the movie *Frozen*.

One of the most effective parts of our soundscape involved the addition of vocalizations to Ember’s animation on the win/loss screen. The voices were minimally processed with volume leveling and compression, but a few needed a bit of further manipulation. The loss vocalizations were slightly time-stretched, or made slower, to increase the emotional impact. The vocalizations were compelling enough in the preliminary play-testing that most of the players commented positively on the win-loss screen. We determined that voiceovers were necessary to match player expectations that would surface when they watch Ember’s animated expressions in the win/loss screen. Changing the music for the loss screen to a slower, lower-pitched, somber tone was also required so that it was distinguishable from the cheerful ‘win’ music. The playtesters enjoyed the voiceovers and the music changes, and thought it added emotional depth to the end of the game. This test showcases the cumulative effect of our efforts. Overall, this experience and effect on the player are the most important factors; the sounds of *Ember’s Inklings* help immerse the player in the Inklings’ world, add unique feedback to each action the player does, and help reinforce positive and effective gameplay.


\(^{19}\) Sound Intensity. (2014, March 13).
Results and Discussion

Playtesting

Playtesting Ember’s Inklinko had two main forms, informal and formal. Throughout the development cycle, Ed Baraf and our academic advisor, David Finkel, played the current game build during meetings that occurred multiple times each week. The feedback from them continuously guided the next development steps and brought the game to where it is now. At the very end of the project, the game had the chance to be playtested formally, providing valuable feedback from the target audience. If there were more time allotted to the project, this feedback would also be used to further the game.

The formal playtest was conducted in a special pair of rooms: the playtest room and the observation room. The playtest administrator, who was the only person to directly interact with the playtester, conducted the session while the team observed from inside the sound-proof observation room, separated from the playtest room by a two-way mirror. Microphones and cameras connected to two monitors allowed close observation of playtester actions. In between play sessions, the administrator would return to the team and discuss the results, making adjustments for the next play session if necessary. The official report, filled out by the administrator, Brooke White, is shown in Appendix D: Official Playtest Report.

The results showed promise for the game. Most of the playtesters found the game to be enjoyable, or at least show potential to be enjoyable when it was complete. They understood how to learn to shoot at least a single ball without outside assistance. Lastly, they enjoyed seeing Ember’s emotions in the statistics screen, finding them cute and causing them to want to know more about Ember.

The results also revealed many aspects that need to be worked on. One of the main problems the testers had was they did not have a solid mental map for the game. For example, just by hearing the name of the game Cooking Mama\(^{20}\), the player can guess the game involves something about preparing food. Conversely, “Ember’s Inklinko” does not readily invoke any preconceptions about the game. Players didn’t know exactly know what they needed to do first, and couldn’t connect the meaning of the gems and paint buckets. Having not played Hidden Worlds, they did not know who Ember was or what the purpose of the stars could be. A second problem was the

---

scoring system. All of the testers could see and identify what caused score, but none could come up with the optimal scoring strategy. Finally, the instructions were not always clear. Some players were hesitant to press the ‘play’ button without seeing tutorials or hints about how the game was played. Also, for those that didn’t discover the ability to shoot multiple balls by accident, the hints to use the ability were not seen or understood. Along the same lines, some couldn’t figure out the exact purpose of the red gems. They saw that they stood out and were important, but, due in part to the lack of a mental map, some did not learn that they unlocked the goal bucket. All of these issues indicated aspects of the game the team would work on if there were more time on the project.

The experience of playtesting and the feedback gained from it was invaluable. It not only helped guide development and future plans, but it also indicated important design considerations the casual game genre in general that were not apparent during the pre-project research. It taught the team that having the target audience play a game was essential for the game’s development to succeed.

Analysis of Development

*Ember’s Inklino* is a successful casual game prototype. We were able to hit most of the critical success factors of casual game development. It is fun and entertaining, relatively easy to learn, is able to be played in small increments of time, and grabs a player’s attention very quickly. Our sponsor was impressed that our team was able to accomplish as much as we had in so short a time. Being onsite at Disney Interactive in Silicon Valley gave us access to professional artists who were able to help us bring the quality of *Ember’s Inklino* to a level that is reflective of Disney’s reputation.

However, there are a number of things we would do differently. Although we have a successful game prototype, with more time we could have implemented additional features and levels. We could have gone to the artists at Disney, whom we were reluctant to bother because everyone was busy with project deadlines, for assistance earlier on, possibly allowing some minor art issues currently present to be cleaned up and allowing for more intricate art assets. More complex mechanics that were envisioned early on could have been implemented properly instead of being partially implemented then scrapped, such as randomization. Given more time, we would have addressed the issues revealed during playtesting, especially the score balancing, instructions, and mental mapping.
Conclusion

We learned that casual game development could be completed in as little as two months. We were able to devote all of our focus to one project with minimal distraction, enabling us to get more done in a day than we would have been able to accomplish in a week while taking other classes. By working in a professional work environment, we were able to feel not like students, but like other employees of a major company. We realize that completing our MQP at a company like Disney was an experience that will serve us well as we begin our respective careers.
References


Appendices

Appendix A: Software List

Unity – https://unity3d.com/
Free game engine that supports 3D and 2D games written in languages like C#, Java, or Boo

3D modeling and animation tool

2D image creation and manipulation tool that can handle animations.

Turns SWF movie files into sprite sheets

Zbrush – http://pixologic.com/
3D sculpting tool

Audacity – http://audacity.sourceforge.net/
Free audio editor and recorder
Appendix B: Progress Reports

The following pages include daily progress reports this team sent to Ed Baraf to keep track of the game’s development.

Previous to the first progress report (Recap):

Christian:

Implemented phasing blocks that turn invisible and visible on a timer
Implemented blackhole blocks that pull your ball in
Implemented a combo multiplier that increases when bricks are destroyed and decreases when the ball touches ground
Implemented a system for painting and keeping track of ink splattered across the screen

Bryce:

Implemented a cannon launcher for the ball
Implemented a “bomb” that blows up bricks around it when struck, and added the texture
Constructed the level frame and background
Ball return

Christian & Bryce:

Collaborated on many of the tasks

Andy & Corinne:

A bunch of assorted level mockups
mockups of specific items/features
placeholder models
1/10/2014:

Christian:

Implemented scoring system using the score multiplier combo value
Implemented portals that teleport a ball and release it using its current momentum

Bryce:

Built a level that includes all currently implemented features
Tweaked portals to look more appealing and helped w/ debuggin

Andy & Corinne:

Rough UI layout for level
Rough level select
Title Screen mockup
Vortex texture
1/13/2014:

All:

Had Milestone 1 Meeting
Had group discussion/meeting after the Milestone 1 Meeting

Bryce Jassmond:

Prepared PowerPoint and Prototype Build for Meeting
Researched porting from Unity to iOS devices
Submitted request for a ticket to IT for gear to build to iOS devices
Debugged with Christian

Corinne Kennedy:

Completed 3 level mockup sketches
Textured 4 placeholder models created by Andy
Had a meeting with Andy to discuss art, style, and thematics
Started 1 detailed model of a pen and texture

Andy Lukas:

created manipulatable mockup sketch
created 6 placeholder models
had a meeting with Corinne to discuss art, style, and thematics
started 1 detailed model of a book
tested several iterations of level 1

Christian Walker:

Designed a very simple level 1 with just breakable and unbreakable bricks
Implemented win condition: Break all blocks in time limit
Implemented win condition: Get ball in target zone
Implemented cross-level persistence of score
Implemented loss screen (win screen TBA)
1/14/2014:

All:

Project Meeting with Prof. Finkel

Bryce Jassmond:

Prepared iOS script for when we obtained a Mac
Prepared Agenda for Meeting
Worked with Christian on debugging and testing certain dynamics
  -- Made a brick-circle level
Changed level-changing mechanic to work with Christian’s scenes
Picked up the Mac for porting to the iPad in the morning.

Corinne Kennedy:

Finished Dip Pen launcher model
Created detailed brick sample model and texture
Created a model for the ball and texture
Playtested variations of level one.
Made more placeholder items for testing.

Andy Lukas:

Made several detailed brick samples
Made more placeholder blocks for level testing
Playtested variations of level one

Christian Walker:

Integrated triangle brick
Created score threshold win condition
Created win screen and updated transitions between them
Tested a 1-ball mechanic where the floor destroys the ball
Tweaked a scene to use the triangle blocks and debugged their meshes
Debugged scenes to work with asset destruction on transition (Bomb will not work with current code -- need to fix).
1/15/2014:

All:
Milestone Meeting
Team Design Meeting after Milestone Meeting

Bryce Jassmond:

Finished setting up iPad for build testing (confirmed to work)
Added shortcut to main menu from win/loss screens
Tested level mechanics passed to me

Corinne Kennedy:

Finished breakable brick
Debugged texture/object imports
Playtesting
Created a backdrop for the game
Created a model and texture for the black hole brick
Reduced size of breakable brick from 300+ poly to 6.
Created a model and texture for an extra life “angel” brick

Andy Lukas:

finished unbreakable brick
made hexagonal gem brick
started explosive brick
started pyramid brick
debugged object/texture import issues
more playtesting, debugging

Christian Walker:

Integrated unbreakable/breakable brick and pen art assets and assorted colliders and replaced stock shapes with them
Implemented “# balls” win condition dynamic
Optimized TryAgain.cs menu, removing severe resource drain
1/16/2014:

Bryce Jassmond:

Converted layout to portrait
Changed iOS input to be drag to aim, release to fire
Worked with Andy to add octagonal bricks/pegs to test
Fixed scene issues
Taught Andy Unity

Corinne Kennedy:

Researched art styles employed by other games
Picked style choices from games
*Castle Crashers*: Thick outlines and no perfect straight lines
*Candy Crush*: 3D rendered textures for shine and depth
*World of Goo*: Simplified backgrounds and color washes
*Bejeweled*: Gems instead of bricks
Created a very simple level layout in Photoshop combining style choices
Playtested a few block layouts
Came up with a way to theme the game’s art and “story”
Created more concept art to fit theme

Andy Lukas:

Learned Unity from Bryce
Researched art styles
Created ‘gem’ style blocks
Created several styles along ‘gem’ design
Experimented with different block layouts in Unity
Started designing levels

Christian Walker:

Implemented key blocks and tied them to the main goal of the games
Developed system to reposition and re-aim for future “glove” re-aim star mechanic or power-up
Tried a few brick layouts for the first level and tweaked physics
Rectangles seem to have less interesting angles than triangular or hexagonal shapes
Changed timer to show number of balls remaining instead
Helped Andy debug Unity tag interface issues
1/17/2014:

All:

Milestone 3 meeting
Post milestone meeting

Bryce Jassmond:

Project organization and synchronization (assets, prefabs, scenes/levels)
Brick strength (for multiple hits)
Made balls hop if they remain stationary for too long

Corinne Kennedy:

Iterated on layouts in Photoshop
Created a texture for a black hole
Textured a spring
Began modelling a chest
Looked into games that use the style we were looking for. Found Bloons Tower Defense 5.
Began retheming the layouts in Photoshop to mesh better with Inklings.
Andy Lukas:

Designed more levels
Tweaked various level mechanics (ball size, bounciness, number of key bricks vs number of balls)
Made spring model
Began modelling a chest

Christian Walker:

Randomization script that can randomly replace N breakable bricks in a scene with key bricks. Replacement with identical type bricks to come later.
Level Select that looks through all scenes and generates buttons that load the correct scene.
Fixed score and combo counter that was broken when new assets were imported
Modified Andy’s nested octagons level to have a black hole at its center and a nicer win zone
1/21/2014:

All:
Progress Meeting with Prof. Finkel

Bryce Jassmond:

Wrote meeting agenda
Improved GUI to be more informative
Re-factored the ball counting and limits

Corinne Kennedy:

Rethemed level concept to mesh with Inklings
Refined style
Modeled a key
Made a placeholder texture for the key.
Modeled an ink bottle.
Modeled the Inkling’s Airship

Andy Lukas:

Tweaked ball size
Tweaked gravity
Tweaked bounciness
Designed several more levels
Assisted with debugging

Christian Walker:

Changed randomization of key blocks to select only from the same type of blocks
Designed level w/ force arrows
Key block tagging and coloring system; Will be replaced with method of texturing later.
Helped Andy figure out a few Unity features incl. physics manipulation
1/22/2014:

All:
Milestone 4 meeting
Post milestone team meeting

Bryce Jassmond:

Added score spawning on bricks breaking
Worked on implementing assets
Worked on synchronization
Assisted team with miscellaneous tasks
Started a color changer for the ball/balloon

Corinne Kennedy:

Made placeholder textures for ink bottle and airship
Submitted Beauty and the Beast background for levels
Made a Mulan and Aladdin background
Researched how to add outlines to 3D models in Unity
Modeled a balloon
Started a Little Mermaid background.
Was introduced to Benny.

Andy Lukas:

Designed more levels
Helped with bottle and airship
Continued trying different gravities, ball bounciness, ball size
Discovered/debugged more versatile way to import objects
Rigged airship's articulating arm
Helped with coloring of scores
Met Benny

Christian Walker:
Altered score display to dynamically modify color and size based on the value of the score
Altered loss screen to correctly show when a game is lost instead of "Time Up"
Tweaked randomization script to include pre-existing keys
Adjusted black-hole and force arrow levels to have consistent coloring and health
Differentiated dangerous and non-dangerous black-holes through scripted color
1/23/2014:

All:

Bryce Jassmond:

Added counter for block health (if applicable)
Helped with adding visual items in the level GUI
Imported art assets
Researched 2D libraries
Tested simple version of a button
Researched animations and animator controllers
Tested strange physics in a level - funny results

Corinnee Kennedy:

Modeled and textured a rubber band ball
Remodeled and textured 3 gem types (breakable, key, and unbreakable)
Helped rig book
Helped animate book
Modeled and textured a rope ladder
Play tested level one
Rigged rope ladder
Play tested physics adjustments

Andy Lukas:

Designed level 1
Ironed out physics issues
Continued tweaking physics
Made placeholder key
Helped rig book
Helped animate book

Christian Walker:
Implemented ball stock and key gems left as images instead of text and integrated it with the existing gameover system
Helped Andy figure out how to tweak ball physics and eliminate the wall hug bug
Researched available 2D sprite engines
Helped implement and optimize new art assets
Playtested several builds with different gravity and collision elasticity
1/24/2014:

All:
Milestone Meeting 5
Playtesting with Alisa
Team Meeting

Bryce Jassmond:
Made structure for primitive buttons
Main Menu
Stat pop up w/buttons
Gem synchronization
Bug fixing w/Christian
Level editing w/Andy
Finished integrating Corinne and Andy's book animation

Corinne Kennedy:
Made quick level layout concept sketches
Retextured bricks and gems to match style more closely
Modeled Belle Blue
Modeled Gaston Red

Andy Lukas:
Put together more levels from concept sketches
Learned how to model bottles
Helped model Gaston Red
More physics tweaking

Christian Walker:
Key randomizer can now be restricted by area
Randomizer can take arbitrary limits on which objects can be keys
Randomizer can now have maximum and minimum limits on how far keys can be apart
Tweaking physics w/ Andy
Temporarily disabled sound
Bugfixing w/ Bryce (incl. win/loss race condition)
1/27/2014:

All:

Bryce Jassmond:

Cleaned out assets
Organized asset hierarchy
Tweaked win/loss popup
Tweaked physics

Corinne Kennedy:

Retextured ball.
Made a placeholder icon for the game
Remodeled the airship (base only)
Made ball more round
Made a background for the menu screen
Looked over Hidden Worlds art assets provided by Benny with Andy
Began working on a saga map layout

Andy Lukas:

Tweaked physics
Assisted in retexturing ball
Modelled and textured a crate
Started re-making levels using new assets
Looked over Hidden Worlds art assets provided by Benny with Corinne
Helped tweak physics more

Christian Walker:

Added system for keeping track of new types of score
Added a score bonus for balls left and separated “bonus” combo score from base score
Fixed persistent bug where a loss would appear after a win if all balls were destroyed
Slightly restructured cannon handling to remove duplicate execution of code
Helped debug win/loss screen persistence
1/27/2014:

All:

Meeting with Prof. Finkel
Milestone 6 Meeting
Post Milestone Meeting

Bryce Jassmond:

New assets and prefabs
New UI implementation

Corinne Kennedy:

Worked on saga map layouts
Began modeling Ember
Began working on getting assets for backgrounds

Andy Lukas:

Grabbed buttons and menu backgrounds from Hidden Worlds
Rebuilt more levels from new assets
Assisted with randomizer upgrade
Debugged randomizer upgrade

Christian Walker:

Fixed point spawning to only count breakable bricks for the purpose of score
Added texture swapping to key randomization engine
Assisted with asset integration
UI backend tweaking
1/30/2014:

All:

Breakfast meeting
Milestone meeting
Post-Milestone discussion

Bryce Jassmon:

Yet more art and asset integration and organization
Builds and synchronization
Began bottle display in stat screen
Texture fixes

Corinne Kennedy:

Made a new menu background
Made cracked brick/gem textures
Finished hand that pokes screen if player is idle
Made backgrounds for levels
Helped come up with titles for levels

Andy Lukas:

Redesigned early levels
Tweaked scoring
Fixed bottle animation
Made better bucket
Bugtesting
Made level titles

Christian Walker:

Made octagons crack upon impact if they have more than 1 health
Made it simple to edit multiplier/score code in the editor
Added pegs to prevent the ball from falling in-between the buckets
Capped the velocity of the ball at a configurable value
1/31/2014:

All:

Bryce Jassmond:

Stayed home sick, but still completed:
Fixes to combo
Added a given font into the game
Made selecting the strength of gems easier (for strength 1-3)
Reduced the time a ball was still before jumping
Worked a bit on the paddle
Added button animations
Helped integrate coin (still is buggy)
Managed and synchronized team work
Fixed pause button functionality

Corinne Kennedy:

Out sick

Andy Lukas:

Designed pause button
Designed more levels
More physics tweaking (with ball max speed)
Assisted with paddle tweaking/bugfixing
Made coin model
Helped fix key gem cracking

Christian Walker:

Implemented pausing functionality as a per-object event.
Added pause behavior to the ball, the cannon and the point spawn items.
Bug-fixed pause button to not fire balls when paused or when the pause button is clicked (later converted to iPad).
Added paddle that moves back and forth and hits ball back automatically.
Debugged kinematic paddle interaction with other kinematic objects
Added coin points script
Improved randomized and non-randomized gem and brick cracking, replacing each with the new cracked textures
2/3/2014:

All:

Team meeting over breakfast
Presentation meeting over lunch

Bryce Jassmond:

More asset integration and synchronization (primarily buttons)
Helped with bucket editing
Helped with brick score and coin editing
Options menu
Edited pause feature

Corinne Kennedy:

Updated menu screen background
Manipulated a PSD of Ember to make a sprite sheet
Concepted a chute for displaying/loading balls
Worked on improving the icon

Andy Lukas:

Made more buttons
Attempted to debug bucket scale randomization
Level design
Learned how to import and animate sprites
Helped with icon

Christian Walker:

Edited bucket code to work with the new bucket images and without pegs
Added configurable scene transition (fade to color)
Update combo to round to the nearest tenth
Fixed combo off-by-one where combo was incremented on hit but score was only added on destruction
Tweaked point spawn size and color thresholds to suit the new values
Worked on fixing point color spawning system on first hit and on coin hit
2/4/2014:

All:

Team meeting over breakfast
Worked on outline for paper

Bryce Jassmond:

Bug fixed coin points and pause w/Christian
Integration, synchronization, and builds for meetings

Corinne Kennedy:

Finished a new iteration on the Icon
Added tweens to Ember shooting animation
Made a paint bucket

Andy Lukas:

Worked on implementing Ember sprites
Helped implement paint bucket art
Made a ball chute (for balls left)
Found some potential sounds

Christian Walker:

Finished fixing color of text spawned by coin (was updating color before calculating new score)
Fixed coin color changing one brick’s color incorrectly w/ Bryce
Fixed pause functionality not restarting on level change w/Bryce
End-of-level bonus fixed
Combo multiplier going below 1 fixed
Helped debug a sound effect not importing into Unity
2/5/2014:

All:

Team meeting over breakfast
Milestone meeting

Bryce Jassmond:

Asset integration and synchronization
Ball chute implementation (balls roll down and get deleted as they are shot)
Minor fixes to various mechanics such as ball jumps

Corinne Kennedy:

Adjusted Ember animation
Made a UI frame object
Changed Home button to more closely resemble a house
Made a new potential goal object
Playtested to see if score thresholds for stars were achievable
Made paint splatter particles
Took screenshots of game

Andy Lukas:

Worked on finding, editing, and implementing audio
Helped teach Unity integration to Corinne
Integrated and synchronized assets
Helped integrate ball chute
Playtested score threshold numbers
Various debugging

Christian Walker:

Added sound selection randomization
Added pitch randomization
Added score-based volume randomization
Added preliminary procedural calculation of score thresholds
Score thresholds testing and tweaking
2/6/2014:

All:

Team meeting over breakfast
Edited presentation

Bryce Jassmond:

Work on debugging icon (unsuccessful thus far)
Fixed ball chute use in iPad
Implemented coin sound effect
Identified issues with physics
Implemented misc. assets
Redesigned early levels w/Andy

Corinne Kennedy:

Retextured Airship Base
Made button icons for sound on/off music on/off and help.
Made particles for gem breaking
Made a sprite sheet for a vortex for the book opening
Added Ember’s name to the main menu screen

Andy Lukas:

Found coin placeholder audio
Redesigned early levels w/Bryce
Score tweaking
Found physics bugs
Added spinning coins to main menu

Christian Walker:

Out sick, but completed:
Ball maxspeed clamp on first bounce
Work on debugging icon
Made game wait for all balls in play to die to show "You Win" screen
Made player get extra score for balls after the first that go in book
Made book destroy ball on collision if it’s open
2/7/2014:

All:

Team meeting over breakfast

Bryce Jassmond:

Fixed the icon appearance on the iPad
Asset integration and synchronization
Tested fixes and levels
Rolling scores on stat screen
Rudimentary filling bottles that begin when a threshold is met

Corinne Kennedy:

Out sick

Andy Lukas:

Cleaned up and implemented gem shattering
Learned how to set up particles for bottom (paint) buckets
Implemented paint splash particles

Christian Walker:

Implemented idle detection w/ finger icon and animation to show “tap here”
Implemented point spawning on buckets
Fixed book open delay caused by bricks waiting for a sound to play before destroying themselves
Fixed Coin chain display bug when several coins are collected in a row
Researched information on pausing animation using the controller
2/10/2014:

All:

Team meeting over breakfast

Bryce Jassmond:

Editing paper
Worked on refactoring code base:
  - GUI handler
  - Button actions

Corinne Kennedy:

Worked on the paper.
Added more paint splatters to spritesheet
Created more backgrounds
Adjusted vortex animation
Textured the coin
Made white textures for gems

Andy Lukas:

Worked on the paper
Arranged paint splatters on spritesheet
Added vortex when the book is open
Added vortex particle burst when the book opens
Implemented paint bucket art, new vortex art
Helped with slide/ramp mechanic
Made level 3 easier

Christian Walker:

Implemented sliding mechanic. Works pretty well for the most part, with configurable angle of incidence.
Adjusted parameters of slides and force factors.
Scripted an animation for a gem growing and then shrinking toward the book when you collect it.
Worked on the paper.
2/11/2014:

All:

Team meeting over breakfast
Meeting with advisor

Bryce Jassmond:

Edits to presentation
Agenda
Implementing assets for demo build
Assisted Christian with trajectory calculations
Implementing and testing trajectory components in current project
Worked on refactoring stat screen

Corinne Kennedy:

Created cracking sprite for gems and bricks
Created shattering sprite for gems and bricks
Created paint particles for golden bucket
Created golden bucket sprite sheet
Helped come up with/test level designs
Found physics/slide mechanic bugs

Andy Lukas:

Helped with gem/brick sprites
Helped with bucket/paint sprites
Attempted to redesign early levels again
Made ball trail, guide line look good
Built level designs
Found physics/slide mechanic bugs

Christian Walker:

Implemented trajectory calculations and display/scaling of resultant line
Added gravitational calculations and a skeleton of collision prediction to the trajectory of the arc
Changed the sliding algorithm to only slide on rectangular bricks
Attempted debugging various parts of the sliding
Assisted with tuning the boost velocity vectors in sliding
2/12/2014:

All:

Team meeting over breakfast
Milestone meeting
Post milestone team meeting

Bryce Jassmond:

Finished refactoring stat menu
Refactored pause and level transitions
Refactored ball
Refactored buckets
Finished refactoring all basic functionality (minor bugs yet to fix)

Corinne Kennedy:

Adjusted title position on menu
Made a new finger sprite animation to replace model
Made concept sketches for potential level designs

Andy Lukas:

Implemented new version of gem sprites
Testing and bugtesting of Christian’s scripts
Level design/redesign

Christian Walker:

Added “death delay” to sliding bricks so that gravity can’t take over
Rendered delayed bricks unscorable but collidable
Worked with sliding mechanics to hug convex shapes more instead of rocketing over
Researched methods of getting the ball to grip a surface without repeatedly colliding
Working w/ Andy to tune script parameters and fix odd bounces
2/13/2014:

All:

Team meeting over breakfast

Bryce Jassmond:

Finished refactoring to the point the project has switched over to that version
   - Few new features missing implementation
Bug fixes
Documented nearly all the code
Worked on audio testing and filling in hooks
Edited UI

Corinne Kennedy:

Remade paint bucket sprites using Hidden Worlds assets
Updated main menu to better match the master version.
Retextured ball
Listened to and gave feedback on audio assets
Began updating icon to better match the master version.

Andy Lukas:

Reimplemented gems
Implemented new paint buckets
Started rebuilding levels
Listened to, gave feedback on audio assets

Christian Walker:

Obtained and edited or synthesized various sounds, including:
Various Music
Win/Loss screens
Magic book opening
Interface sounds; select/move/pause
Analyzed soundscape of Hidden Worlds to see what fit
Iterated over the sounds with Andy and Corinne
2/14/2014:

All:

Team meeting over breakfast

Bryce Jassmond:

Made a handler for gems to be automatically colored and assigned sounds depending on their settings
Fixed how gems destroy and score
Added functionality to the goal and it’s scoring
Added a main menu handler
Filled most of the audio hooks with trial sounds
Implemented misc. assets and bug fixes

Corinne Kennedy:

Created UI items
Iterated on Menu Screen
Iterated on Icon
Worked on ball explosion
Helped with level design/playtesting
Listened to and gave feedback on audio

Andy Lukas:

Rebuilt/redesigned/playtested levels
Listened/gave feedback on audio
Helped with ball explosion

Christian Walker:

Many more iterations of audio stuff, filtered through Andy and Corinne
Worked on tuning existing and new acquired sounds with HPF/LPF/noise removal/etc to mask recording quality issues
Pullback/firing sounds
About 20 different types of brick breaking sounds, a few of which were selected
2/18/2014:

All:

Team meeting over breakfast
Prepped for and had advisor meeting

Bryce Jassmond:

UI integration and fixes
Misc. asset integration
Misc. script changes

Corinne Kennedy:

Fixed UI issues
Worked on final presentation
Fixed icon
Level Design
Worked on making gems shinier

Andy Lukas:

Worked on final presentation
Level design
re-made gem shatter animations
Started re-making Ember animation

Christian Walker:
Continued iterating on audio
Submitted 13 different button sounds for review
Worked w/ Andy to shorten the slingshot sound
2/19/2014:

All:

Team meeting over breakfast

Bryce Jassmond:

Refactored/corrected trajectory display
Audio task list
Finger implementation
Work on re-implementing aiming to work with possibly new animations
Extra balls fire and burst (with score) when the level ends

Corinne Kennedy:

Made more adjustments to gem sprites.
Listened to and gave feedback on audio
Played Peggle to get ideas for level designs
Worked on list of assets to ask artists for

Andy Lukas:

Worked on re-animating Ember
Put together requested asset list
Level design
Re-implemented gem sprites
Listened to and gave feedback on audio
Made and imported ball explosion animation

Christian Walker:

Around 35 new iterations of different sounds
New crack/pop/explosion sounds
New wall bounce and bucket sounds
Improved above with feedback from Andy
2/20/2014:

All:

Team meeting over breakfast

Bryce Jassmond:

Re-implemented key gems heading to the goal in a level
Talked to Benny about assets
Checked out issues with pitch implementation

Corinne Kennedy:

Researched articles on Peggle and other pachinko style games that dealt with level design.
Level design
Playtesting
Helped with Alum Gathering speech
Talked to Benny and Dave about assets
Transferred gem sprites to Benny and Dave.

Andy Lukas:

Level design
Helped debug gems going to book
Practiced Alum Gathering speech
Talked to Benny and Dave about assets

Christian Walker:

Re-implemented sliding pitch w/ number of hits functionality
Made a level 2-3 prototype that looks like a cloud
Made a ‘slide machine’ level
Discussed and helped start debugging audio handler
Talked to Nick about audio
2/21/2014:

All:

Team meeting over breakfast

Bryce Jassmond:

Added more fine-tuned audio managing to the Audio Handler
Made changes to scorables objects for better score handling
Made gems change pitch based on combo (temporarily disabled)
Corrected errors related to ball trails trying to be destroyed after program exiting
Added audio library w/methods to Audio Handler
  - Allows us to play audio clips by reference instead of making sure objects
    have them
Fixed issues with score scrolling being called too many times.
Inserted several new temporary sound handles to test Nick's audio

Corinne Kennedy:

Worked on paper
Listened to and gave feedback on audio

Andy Lukas:

Out sick

Christian Walker:

Edited Nick's sounds (splicing, volume normalization and noise removal)
~20 new sounds for clicks, breaks and other impact sounds
Mastered the remaining sounds to have consistent volume levels and ends/stops
Down-mixed and tweaked a few tracks
Continued researching sliding methods that don't interfere or depend on the physics
engine
Worked on polishing the levels I made yesterday
2/24/2014:

All:

Team meeting over breakfast

Bryce Jassmond:

Worked on audio changes and re-imports
Integrated stars
Re-made end screen

Corinne Kennedy:

Iterated on main menu background
Added reflect map to ball material
Worked on final presentation powerpoint
Worked on paper

Andy Lukas:

Made filling star animation
Worked on final presentation powerpoint
Cleaned up and normalized all audio assets
Implemented Dave’s gem treatment
Implemented new finger/hand animation
Worked on paper

Christian Walker:

Out sick
2/25/2014:

All:

Advisor meeting

Bryce Jassmond:

Finished implementing and cleaning audio cues in scripts
Added the desired goal bucket implementation
Added lids to buckets
Minor work on final presentation
Implemented finger animation/movement
Delved into memory issues with our game for iPad 1
More work on stat screen

Corinne Kennedy:

Worked on final presentation
Attempted to work with Ember animations
Changed color of breakable gems to be more distinct from ball
Made level design concept
Playtested level

Andy Lukas:

Worked on final presentation
Attempted to work with Ember animations
Reimplemented old levels
Assisted with audio
Attempted to find solutions to memory issues

Christian Walker:

Added ~20 sounds for ball hits in the rack, unbreakable brick sounds, finger tap/swipe
Researched methods for reducing Unity memory usage (Overflow in the iPad 1)
Tweaked coin levels to have a more natural volume falloff and lower max. volume
2/26/2014:

All:

Team meeting over breakfast
Edited list of pre-playtest tasks

Bryce Jassmond:

Adjusted gems shattering when a level ends
Background implementation
Made the launcher aim at the instructional finger while it is active
Edited point spawn location for explosions
Edited pause and stat windows
Cleaned animation handling on Ember
Added level title banner
Fixed little things from meeting
  Name, gem shatter speed, finger size, trajectory dots

Corinne Kennedy:

Level designs
Adjusted Next button
Helped with new ball explosion anim

Andy Lukas:

Implemented new rectangle shatter animation
Separated Ember firing animations
Brought back old levels
Tested fixes, new levels
Re-made ball explosion animation
Started on Ember emotes

Christian Walker:

Added animation controller that handles when Ember pulls back her slingshot, is idle, or fires the ball instead of just looping the animation endlessly
Implemented new animation handling so that stars fill proportionally to how much of that threshold was achieved instead of as discrete units
Gave feedback on new UI design and levels
2/27/2014:

All:

Team meeting over breakfast
Edited list of pre-playtest tasks

Bryce Jassmond:

Fixed stars’ filling so that they wait for the previous star to complete before filling
Adjusted time delay on stat screen
Experimented with different bucket layout + fixed layering
Made buckets not play a sound if opening/closing on initialization
Added hook for bucket lid closing
Made the stats/explosions wait for gems to be fully shattered
Worked on asset integration and synchronization
  - Stat screen with Ember’s emotes

Corinne Kennedy:

Out sick
Updated ball texture

Andy Lukas:

Edited Ember animations to be framed
Made framed ember emotes into spritesheets and implemented
Updated goal bucket
Updated ball texture and explosion
Updated gem textures and animations
Added key gem slots to goal bucket
Updated powerpoint presentation

Christian Walker:

Algorithmically generated drop shadows that can be selectively applied to text meshes
Added shadow update code so that the shadow stays in sync if text changes
Changed shot indicator dotted line to only display while the screen is being pressed
Added platform detection code to change aspect ratio appropriately
Added code to skip animations and score scrolling if the screen is tapped
2/28/2014:

All:

Team meeting over breakfast

Bryce Jassmond:

Overnight:  
Cleaned assets  
Fixed Buckets  
  - Balls won’t go through lid  
  - Gem slots fill up  
Stat Screen  
  - Tweaked speed  
  - Fixed broken animation controller on star  
  - Added shine when a star is fully filled  
Rewrote a script to automatically maintain aspect ratio for any iOS device  
  - Based heavily on a resource documented in the script  
  - Edited Button so this is testable in the editor as well as the iOS device  
Corrected stats skipping to end (with stars) when tapping in the stat screen  
  - All buttons skip when pressed down as well  
Added shine to the goal bucket when it opens  
Started on rebuilding levels - Base + Level 1  
Bottom killzone in case there are no buckets (i.e. level 1 and something goes wrong)  

At work:  
  - Gem fix for shadows  
  - Level design  
  - Played some Papa Pear for inspiration

Corinne Kennedy:

Final presentation revisions  
Level designs  
Attempted to find better gem colors  
Playtesting

Andy Lukas:

Attempted to find better gem colors  
Re-built/redesigned early levels  
Attempted to balance scoring
Tried to find what's using large amounts of memory

Christian Walker:

Quantized some PNGs to 256-bit color to avert memory issues
Recompressed PNGs using zopfli and TruePNG to improve memory usage
Added occlusion culling to existing assets
Fixed an issue of the build not properly compiling for Windows
Prepared Windows build for Nick to do an audio pass on
Helped Andy do score balancing
Research into unity memory usage; as despite these optimizations and drastic reduction in asset size, the memory usage is near constant.
3/3/2014:

All:

Team meeting over breakfast

Bryce Jassmond:

Hide trajectory if aiming when the goal is met
Asset integration
Discussed audio with Nick
Main menu animations
Credits slide on main menu
Worked a bit on aspect ratio checks
Fixed sound error and shine effect at the stat screen

Corinne Kennedy:

Separated menu screen image into components for animation.
Playtesting.
Animation for the shine.
Researched texture masking in Unity.

Andy Lukas:

Adjusted score thresholds
Re-ordered levels
Tried to come up with methods of adding shadow to main menu Ember

Christian Walker:

Cleaned up and reworked volume, pitch and dynamic range for Nick’s new assets
Reworked coin and brick sounds further to make them subjectively softer and louder respectively
Smotherer animation techniques implementation for the main menu w/ Bryce
Discussed audio with Nick
Texture masking shader code and research w/ Andy
Helped figure out what was going wrong with scaling on the iPhone
Scripted fix for shine display (stops showing at end of level)
3/4/2014:

All:

Team meeting over breakfast
Level testing
Playtesting
Post-playtest meeting

Bryce Jassmond:

Audio work + script work based on audio-related or -revealing hiccups
Level editing, pruning, naming, and re-arrangement
  - Coins in each level-

Corinne Kennedy:

Andy Lukas:

Christian Walker:
Implemented post-playtest adjustment of firing power dependent on drag distance
3/5/2014:

All:

Team meeting over breakfast

Bryce Jassmond:

Fixed an issue with the variable speed of the launcher
Fixed camera aspect ratio calculations for alternate devices
Cleaned up extra assets
Edited Hypothetical Future Plan to include our points and plans from before the playtest
Worked on final game archive (Unity project, build, and IPA file)
Worked on paper formatting and section layout

Corinne Kennedy:

Worked on final paper

Andy Lukas:

Worked on final paper
Found last-minute bugs

Christian Walker:

Worked on final paper
Appendix C: Milestones

Milestone 1
01/13/14
Design & Mechanics Prototype

Core Concept
- Casual mobile casual game
- Setup Map
- Limited Lives/Atempts by Energy (wechselable)
- Limited Moves
- Retry
- Challenges (Consecutive levels)
- Player Level Generation
- Victory Depending or Desired Difficulty
- For Campaign Mode
- Crafting
  - Random Drop Rewards
  - Used for Upgrades
  - Framed with Holographic" for Disney IP
- Target Audience
  - Women, 35+

General Concept
- Brick Breaking
  - Various Brick Types with Different Effects
  - Brick Heaven
- Limited Skills
  - Item Activation "Total"
  - Paltwitch Dependable
- Scoring
  - High scores
  - Increased EXP
- Various Level Goals
  - Score Threshold
  - Goal Reach
  - Timed play with In
- Mechanics
  - Ovis: Obtained through Shop or Purchase
  - Used for Upgrades

Mechanics Prototype
- Physics Demonstration
  - Normal
  - Special
  - Force "Yin"s"
  - Physics
- Examples of Brick Types
- Principle Skill (AX) and Action
- Principle Skill (AX) and Action
- Scoring Demonstration
- "Ghost Prototype"

Concept Art

Undecided Concepts
- Lava Intervise
  - Inaccessible
  - Manipulable Elements
- Active Reactions
  - Stick with Censor
  - Dropper
  - Include a Puddle
- Ship Features
  - Rock Ball Bank Up
  - Refillable Bottles
  - Replaced by Launcher
  - Except for Pink Balls
  - Bloated Bottom
  - Similar to Princess Bottom
<table>
<thead>
<tr>
<th>Undecided Concepts</th>
<th>Next Milestone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Skill Limit</td>
<td>Choose:</td>
</tr>
<tr>
<td>Only 1 Ball Active vs. Any Amount</td>
<td>- Tassie</td>
</tr>
<tr>
<td>- Current Factions</td>
<td>- Time Frames</td>
</tr>
<tr>
<td>- Kill vs. Specific Areas</td>
<td>- Key Decisions</td>
</tr>
<tr>
<td>- One Shot per Turn</td>
<td>- Documentation</td>
</tr>
<tr>
<td>- Gun Shot, shotgun replaces Ask view gizmo that measures up trajectory</td>
<td></td>
</tr>
</tbody>
</table>
Milestone 2
01/15/14
iPad, "Level 1", and Art

Current Build
- iOS
- Multiple "Level 1"s
- Capability for Testing Variables
- Suggested: Re-factor for Monday
  - Code is currently hard to extend
  - Better organization will help streamline development

Art
Milestone 3
01/17/2014
Picking a direction

Current Build
- Moving more towards modern style
- Incorporating Drew places and designs to determine what’s fun and relating

Art
- Started developing a style based on:
  - Minigames
  - A mix of pixels and vectors
  - Realistic and abstract elements

Style:
- Fable-like
- Cartoony, angular lines
- Quests for exploration and discovery
- 3D terrain and obstacles
- Simplified background melodies

Theme
- Wanted to solidify a theme story idea to solve the game
- Original concept was a dungeon
- Concept was later changed to a castle
- Potential for more elements in future levels
- Levels are omnidirectional and imaginative

Gameplay Progress
- Ported to portrait view for the iPad
- Changed the terrain to show vertically downward
- Sticks work best interacting shapes in the how these they collide compared to other polygons, inspired by thinking of Hyrule and Nuggs
- Powering up is “overworld” after filling special tiles – that allows you to re-win the bell, areas. Not currently on iPad
- Core mechanics: action, puzzle
- Several levels were designed of level selection feature. Some are on the iPad
- Some preliminary art assets changed
Milestone 4
A puzzle game with levels and an interactive UI.

Art Style
- Pixel art
- 8-bit graphics
- Hand-drawn elements
- Platformer elements
- Boss battles
- Level completion rewards
- Dynamic particle effects

Art Assets
- Object readiness for levels, such as 3D models of the whale and bottles
- Backgrounds taken from classic Disney movies

UI
- Simple controls for new players
- Buttons
- Icons
- Score
- Showing temporary numbers when a block is destroyed, indicating score

Level Design
- Defined core mechanics: gravity, wall active, bounce
- Class but doesn't suit for	roles
- Planned: designing decorative levels based on a background

Prototype
- Platformer
- Large number of available levels
- Two different types of levels
- Basic reflection
- Living wall visual interactive
Discuss

- Is the game fun?
- Is the game understandable?
- Unique?
  - Gonna innovate
- Is the game on the right path?
  - Commrury
  - An why
- What needs to be done for milestone 3?
Milestone 5

Dialing in

Level Design

- More structured levels
- Attempted to design levels 1-6
- Very easy to win, but still possible to lose
- No random generation in early levels

Art

- Continued working on assets
  - Main menu and trailer backgrounds
  - The three main enemies in different poses
  - The game's logo
  - Modified version of the unique letters
- Will eventually have an ending title with a depilation hanging from the ceiling so the logo won’t be rotating to weird
- Also worked on some quick level layout sketches.

Prototype

- Primitive Menu w/ Buttons
- Organized levels
- Primitive Stat Screen
  - Pops up at level end with options (bug: beginning)
  - Text objects in scene currently overlay (bug)
- Book Opens
  - On getting all keys
  - Will bounce when closed
    - Will pass through when open
    - Triggers End
    - (doesn’t pause yet)
- Ball and Key Gem indicators

Procedural Generation ~ Randomization Tuning

- Can now declare area where key-blocks will spawn
- Can now restrict key-blocks to block categories
- Can now designate minimum and maximum spread between key-blocks
- Future: Extend randomization engine to include block generation
- Future: Dynamic parameter tuning ('Seesaw difficulty') and difficulty scaling
Milestone 6

Prototype Update
- More messing with physics
- Better UI
  - Still need menu button in levels
  - Still need font
- Levels
  - Updated
  - Extra (1-5 +misc.)
- Some Art Asset Integration
  - Airship, Ball

Level Design
- Shape bugs and ball
- More tiles with each level
- Rebuild levels with new assets
- No access to assets

Art
- Same game assets for art assets mentioned
- Begun working to create art on character envelopes; bringing some elements together
- Begun reusing the brushes to make the best use of the asset's detail
- 3D asset model instead of using a letter
Milestone 7

UI Changes
- Changed in-app UI
- Consistent outside
- Ball count above center
- Keep score with the book
- Green screen
- Dice when they reduce in strength
- Does not work for large values

Ball Buckets ~ Easing failure
- Randomly generated: bucket with weighted width based on size
- Increase multipliers and variety
- Special buckets such as multiplying "impact" as a multiple of the average point value for purposes of length
- Global vs. local consideration for weighting?
- More or less accurate?

Scoring
- Still have live score and results
- Added separate statistics to the level and results
- Skilled fill up to indicate goal achievement
- Currently at just fill up

Art
- New items and icons
- Seamless animation
- Colors and shading
- Felt texture added for transparency and thickness in physical
- Items not available in test above
- Improved rendering
- Images of elephant on screen generated
- Sketched has been added but may not be implemented

Planned
- Levels
- Initial state
- Light reflection
- Automatic background fading
- Transitions
- Game audio
- Moving score placeholders
- Maximum speed for the ball

3/20/2014
Questions

- Upcoming presentation
- What is the research background?
- What topics can’t we talk about?
- Paper
  - To complete the project for LII, we have to write a project report
  - You (Diana) will have time to go over it and read all the topics you don’t want going public.
  - Quiz any office time as used for it.
Preliminary Score Threshold

- Currently based on maximum possible theoretical score (hit all bricks with one ball)
- Does not factor in "buckets" or the end of ball bonus—these are bonuses for the player
- Factor in number of bricks
- Future: Factor in average distance of bricks to other bricks (clusters/local density would be preferred, but that is quite difficult)

Milestone 8

Art

- Adjustments to main view background
- New and improved graphics
- Improved loading and game state initialization
- Level design
- Ember animation
- Level design

Level Design

- Now much easier to build levels with multiple gems strengths
  - 1-3 hits
  - Haven't re-made all levels with this in mind yet
- Coin added
  - Used to show player "recommended" route
  - Still needs to be added to most levels
- Started experimenting with ways to have the player launch multiple balls at once

Fixes and Clean-up

- UI
  - Animation
- Options menu (levels, home, mode) w/ pause
- Physics Tweaks
  - Added a max speed component
- Scoring fixes
  - Fixed many visual and mathematical bugs
- Ball Trail, Tapered
- Ball Jump Timer, reduced

Misc.

- Ball Ramp
  - Shows how many balls left
  - Ember will take a ball from the ramp to fire
- Moving Paddle
  - Made, not present
- Audio
  - Still working
Appendix D: Official Playtest Report

The following pages contain the official playtest report written by Brooke White, who conducted the playtest sessions the team observed.
Executive Summary

- While there are some challenges with the UX and learnability of key features – for the most part, players were able to actually “play” this game, learning controls and getting better over time.
- Players unanimously confused by scoring ruleset
- Lack of clear mental map about why Ember was shooting balls into paint buckets led to player confusion on “goal” of the game. Teaching ability to shoot multiple balls was inconsistent and confusing
- Some players struggled for a long time on “opening” the buckets via hitting the red gems
- Aiming and “shooting” controls for the most part felt smooth to players. However all players instinctively tried to stretch aim to shoot faster, or push back up toward slingshot to “shoot more gently”.

Goals

- Evaluate new game playability
- Assess UX and UI
- Evaluate if players are able to cognitively scoring

Protocol

1:1 Standard Usability, Think-Aloud Protocol

Target

- 21-40 women mobile players
- play hidden object games and/or puzzles

<table>
<thead>
<tr>
<th>InkLInko - 2/4/14</th>
<th>Name</th>
<th>Gender</th>
<th>Age</th>
<th>Games Played</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:30-11:00</td>
<td>Alexandra</td>
<td>Female</td>
<td>21</td>
<td>Bejeweled, Candy Crush Saga, Flappy Bird, Dice with Buddies</td>
</tr>
<tr>
<td>11:15-11:45</td>
<td>Nina June</td>
<td>Female</td>
<td>27</td>
<td>Paplinko, Angry Grandma Toss, Fruit Ninja, Adventure Town, Design This Home, Animal Voyage, Hunger Games Catching Fire</td>
</tr>
<tr>
<td>Morning Floater</td>
<td>Michelle</td>
<td>Female</td>
<td>25</td>
<td>Diablo III, Bejeweled, Bubble Mania, Tetris, Snake, Candy Crush Saga, Angry Birds, Plants vs Zombies</td>
</tr>
<tr>
<td>Time</td>
<td>Name</td>
<td>Gender</td>
<td>Age</td>
<td>Games</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>--------</td>
<td>-----</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>12:15-12:45</td>
<td>Teresa</td>
<td>Female</td>
<td>31</td>
<td>Naughty Kitties, Pudding Monsters, Plants vs Zombies 2, Hay Day, Bubble Seasons</td>
</tr>
<tr>
<td>1:00-1:30</td>
<td>Kisha</td>
<td>Female</td>
<td>21</td>
<td>Bejeweled, solitaire, Temple Run, Diner Dash</td>
</tr>
<tr>
<td></td>
<td>Kimberly</td>
<td>Female</td>
<td>30</td>
<td>Skyrim, Rockband, The Sims 3, Words with Friends, Angry Birds</td>
</tr>
</tbody>
</table>

### Results

<table>
<thead>
<tr>
<th>Usability Findings</th>
<th>Recommendations / Team Plans</th>
<th>Priority (click to sort)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Players unanimously confused by scoring ruleset</td>
<td>• Players unclear how to optimize score (use fewest balls? knock out most pegs?)</td>
<td>1 - HIGH</td>
<td></td>
</tr>
<tr>
<td>• Players unclear about relationship of stars to score</td>
<td>• Some usability issue w/ Star image looking &quot;filled&quot; when not completely filled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Players expressed delight with Ember's facial expressions and vocalizations. They didn't want to &quot;disappoint&quot; Ember.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lack of clear mental map about why Ember was shooting balls into paint buckets led to player confusion on “goal” of the game.</td>
<td>• &quot;Golden&quot; bucket was clearly important to players</td>
<td>1 - HIGH</td>
<td></td>
</tr>
<tr>
<td>• Money / coins were a strong pull for players to try and get</td>
<td>• Recommend providing a clear mental map for players on why Ember is shooting balls into bucket and why one bucket is more important to hit than others. Scoring should then reinforce this mental map.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Red gems mostly stood out as different and special - but players unclear what purpose of gems was.</td>
<td>• Recommend using different iconography on red gem (like key or switch) to clarify the purpose of this gem.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Teaching ability to shoot multiple balls was inconsistent and confusing</td>
<td>Consider staggering level balance to help teach multiple ball usage / requirement (i.e. you could only GIVE the player one ball each level for the first two levels, and then make a big deal about giving them 2 balls - and balance so usage of two balls at once is the only way to solve the puzzle)</td>
<td>2 - MEDIUM</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>- Some players repeatedly tap (like they were individually popping) the gems. This early on frenetic tapping confused these players about actual gameplay</td>
<td>- Will need to clearly tutorialize usage of multiple balls - but recommend not allowing multiple balls for the first few levels</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Some players only ever use one ball at a time - even with repeated clues from moderator</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Some players struggled for a long time on &quot;opening&quot; the buckets via hitting the red gems</th>
<th>Recommend clarifying mental map between bucket, bucket top and red gem</th>
<th>1 - HIGH</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Some players never put together the red gem with the buckets. It took actually losing the round for these players to understand that the bucket wasn't &quot;open&quot;.</td>
<td>- Recommend clearly showing that the bucket wasn't open (the glowing was mostly noticed by players, but the lid wasn't noticed - and the lid is actually the thing they need to learn)</td>
<td></td>
</tr>
<tr>
<td>- Lack of mental map between bucket, bucket lid and red gem contributed to this issue</td>
<td>- Progression in the last 2 sessions where the level order was switched worked better for training players on using multiple balls</td>
<td></td>
</tr>
<tr>
<td>- Once players did finally figure out the usage of the red gems, they were able pretty easily to put together that they needed to hit multiple red gems to open the bucket</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Several of the players hesitated for a long time on the opening screen - hesitant to press &quot;play&quot; without any background knowledge of what type of game it was or whether they would understand how to play it.</th>
<th>Recommend changing button on very first load of game to &quot;start&quot; or something like that, or a &quot;play&quot; arrow (many Disney mobile games use the play arrow)</th>
<th>2 - MEDIUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Players didn't come into the game with any knowledge of what type of game it was or familiarity with the main character or setting.</td>
<td>- Recommend splash screen on initial load (or even all loading) that shows image of gameplay - or better - a little animation of gameplay</td>
<td></td>
</tr>
<tr>
<td>- &quot;Play&quot; meant to these players that they would be dumped immediately into a game and that made them scared.</td>
<td>- Recommend setting up &quot;goal&quot; of the game via splash screen and strong mental map</td>
<td></td>
</tr>
</tbody>
</table>
Aiming and “shooting” controls for the most part felt smooth to players. However all players instinctively tried to stretch aim to shoot faster, or push back up toward slingshot to “shoot more gently”.

- Some challenges with aiming on left side - unclear if issue is programmatic or the trail of coins isn't placed correctly in these levels.

Unclear if the intuitive desire to control velocity of the shot was because of strong mental map of slingshot or the aiming vector dots contributed (players didn't use word slingshot when describing).

| 2 - MEDIUM |

A couple of major UI / navigation elements are yet to be developed

- Players had no sense of "where they were" or "what level they were on" or "what their goal was"
- Little goals of each of the levels weren't clear to players without experimentation, title of level somewhat helped
- Level scoring was unclear, and relationship between number score and stars was unclear

| 3 - LOW |

In discussion, players talked about understanding implicitly the saga style maps as both "overall goal" and "where I am in meeting this goal"

- Team is planning on a saga style map
- Team is planning on "in between screens" where goal of each level is clear
- Recommend using these in between screens as a good way of merchandising possible boosts / powerups (ala Candy Crush)