December 2011

Picturiffic - Porting a Flash Facebook Game to iOS

Kevin Richard Scannell
Worcester Polytechnic Institute

Kyle Horn
Worcester Polytechnic Institute

Michael Nicholas Johnson
Worcester Polytechnic Institute

Stephen Andrew Heitman
Worcester Polytechnic Institute

DMO MQP 4099

Picturiffic Mobile

Research and development in porting from Facebook to iOS devices

A Major Qualifying Project Report
Submitted to the faculty of
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements for the
Degree of Bachelor of Science in:

Computer Science
&
Interactive Media and Game Development

by
Stephen Heitman, Kyle Horn,
Michael Johnson, Kevin Scannell

Advised by:
Professor Dean M. O'Donnell
Professor David Finkel

Sponsored by:
Large Animal Games

Abstract

Large Animal Games, a leading social game developer, is interested in mobile devices as a new medium for their games. They wanted to port their puzzle game, Picturiffic, from Flash to HTML5/Javascript. Based on our research we found that a more feasible approach was to use Adobe AIR to publish a native mobile application. We completed two major tasks: design and creation of a proof-of-concept desktop application as well as research and preparation to publish our new build with Adobe AIR.

Contents

Table of Figures

Executive Summary

One of New York City's leading developers for social and mobile games, Large Animal Games, had an interest in researching HTML5 and Javascript, with the intent to port their Flash-based Facebook game, Picturiffic, from the desktop browser to mobile devices like the iPhone and iPad.

We began the project with a trip to New York City, which was originally delayed due to inclement weather. Having lost the first week of the project, we were finally able to meet with the Picturiffic team at Large Animal Games headquarters to go over the specifications for the project as well as their conventions and procedures that we would adhere to while working with them.

When we got back to WPI we began researching the plausibility of using HTML5 and compatible animation software to successfully port their game. We then returned to the Picturiffic team with the conclusion that HTML5 and Javascript didn't have appropriate animation software that was developed enough to complete the project that Large Animal Games originally envisioned. Together with Picturiffic's producer and engineer, we decided to shift gears and revert back to Flash as the animation medium and use Adobe AIR, which was a recent release from Adobe that would allow us to publish standalone applications that would run on most devices, including iOS. The two most popular devices that run on iOS are the iPad and the iPhone 4/4S. We also decided that the two members of the tech team would split, where one would integrate Picturiffic with Adobe Air and the other would work with the artists to prepare the newly designed build to integrate with Adobe Air later.

Another part of the initial design process for this project was rethinking the user interface. Since we would be working with touch screen devices and different screen resolutions, rearranging the interface was crucial for an enjoyable and fluid user experience. Mike and Kyle, the two artists on the team, researched different interface conventions, both by playing through popular mobile games as well as reading into psychological and rhetorical implications of interface design. After they had a good understanding of popular mobile conventions and mechanics, they engaged in a design and feedback

process with the main artist from Large Animal Games which eventually boiled down to interface designs for Picturiffic on a mobile device, particularly targeting the iPhone 4 and 4S.

While Kevin worked on getting an Adobe AIR version of Picturiffic running, Stephen, Kyle, and Michael worked on a standalone desktop version that implemented the new interface designs for the mobile application in the original game.fla file.

Kevin worked on the back-end code, preparing a separate build where the Actionscipt code and XML files would function properly inside of Adobe AIR. Due to the way that Large Animal structured the Flash project to compile and load, issues arose when it came to compiling and running AIR packages, such as not enough memory and latency issues. The game needed to be reorganized such that it would run off of data we provided it, rather than searching for data externally. Additionally, code could only be contained in one file to satisfy a requirement imposed by Apple.

While we did run into issues publishing the game to Adobe AIR, we were ultimately able to successfully merge the back-end code and the new UI together. Once we had a functional build running, it allowed us to start the process of debugging and optimizing the implemented designs so the game would look and run flawlessly on the iPhone as a native iOS application.

Literature Review

When we considered the overhaul that we would be doing to the interface for Picturiffic we ultimately knew careful research was going to have to be conducted in order to regulate our design pipeline. There were three main areas of focus for our user interface research which included existing games, specifications and processing power of mobile devices and psychological implications of our designs.

We played a handful of popular mobile titles that developers have released for both the android and iOS market. These titles included Angry Birds by Rovio Mobile, Hanging with Friends and Words with Friends by Zynga Inc., Text Twist 2 by RealNetworks Inc., and Plock by MetroGames. These titles served as reference points for techniques in interface design. These games offered some of the best insight into designing a robust and successful user interface.

In order to familiarize ourselves with the mobile devices we were targeting we also looked into their particular specifications. Wikipedia gave us a table with all of the iOS devices and their exact specifications. Interestingly, this article offered more usable information on one webpage than, for example, Apple's own site. This article gave us the exact dimensions and resolutions of the different iPhone screens as well as the specifications on their memory and processing hardware.

In order to understand the implications of a user interface design we looked into Jeff Johnson's *Designing with the Mind in Mind: A Simple Guide to Understanding User Interface Design Rules*. The book asserted that as humans we have a natural tendency to organize visual information as well as simplify it for our own understanding. The Gestalt principles alluded to how one can use this innate behavior to both avoid bad design and improve good design. We found that the proximity, similarity and figure-ground principles were the most applicable to the work we would be doing.

On the tech side, we reviewed Adobe's documentation for Flash and AIR, and used various online accounts from other developers for insight into the quirks and dangers of mobile Flash development. We also turned to Apple's official documentation to research the pros and cons of iOS development, through both HTML5 and AIR.

Additionally, we turned to various blogs and forums, like IndieFlashBlog and StackOverflow as we encountered problems with our development. As it turns out, many of our concerns were well-known issues, which Adobe is working on fixing in the AIR platform, since it has only recently expanded to include iOS development. Community resources like Adobe's forums, therefore, provided a good deal of insight into the best ways to get around problems based on others' experimentation.

When researching how to optimize Flash for better performance, we encountered a forum post on kirupa.com which gave details on how to minimize memory usage in the context of Flash games for mobile devices. This helped us to better understand the performance impacts of the elements already in the game, as well as those that we were adding. This understanding included the memory costs of using bitmap images, but not vectors. Thankfully, the hiland.com knowledge base had some helpful hints regarding the differences between the two forms of image files, and this knowledge resulted in our careful allocation of the two in the game to improve performance.

Introduction

Background

Large Animal Games (L.A.G.) is a game development company that creates casual social games that are easily accessible to large audiences on social networking sites like Facebook and Bebo. The company was founded in January 2001 by Wade Tinney and Josh Welber and is based in New York City, New York.

Figure 1: Large Animal Games Logo

L.A.G. is the creator of popular Facebook games such as Bumper Stars, Lucky Strike Lanes, Bananagrams, and dozens of others, created both independently and collaboratively with companies such as Microsoft, LEGO, Mattel, MTV, Discovery Kids, Cartoon Network, New York Philharmonic, Swatch, Arm & Hammer, Henkel Brands (Tone) and others. With over 90 games to their name, they have become a large part of the casual gaming scene over the past decade.

One interesting aspect of the company comes from their proprietary TOGA system, which handles much of the background work involved in the development of social games. TOGA is a cross-network social games platform which allows L.A.G. to deploy a single game to several social networks without worrying about site-specific details like Facebook Connect. This allows the development teams to focus on the game creation, without having to worry about deployment.

We worked on one of their beta projects called Picturiffic. Picturiffic is a new and exciting puzzle game from Large Animal Games created in Flash and released on Facebook. The team involved in Picturiffic's creation includes the owners Wade Tinney and Josh Welber, the game designer, Jon Keefer, company art director, Brad MacDonald, the producer, Andrew Burrows, the programmer, Yossi Horowitz, the artist, Shiho Hoshino, and Diana Hsu, the content creator. The game play is described as being "[like] Hangman with a visual clue". The player is presented with a phrase which has several letters missing and an image which is partially obscured by the missing letters from the phrase. The picture becomes an additional clue to solving the phrase as it is revealed.

Visiting Large Animal Games

The four of us and Large Animal Games decided that the best way to kick off the project was to meet in person, establish a game plan and go over specifics for the next fourteen weeks. The two groups initially agreed on meeting Monday, August 29th, and Tuesday August 30th in New York, but after warnings about Hurricane Irene hitting the state on Sunday, August 28th, we decided it wasn't a good idea to travel through a storm of that caliber and pushed the trip to the next weekend.

On Tuesday, September 6th, we met the team at Large Animal Games and jumped straight into meetings with the group that we would be working with on the project. This core Large Animal group consists of the design lead, Andrew Burrows, the tech lead, Yossi Horowitz, and the art lead, Shiho Hoshino.

Andrew started the day off by outlining the trip. He explained to us that there was a general meeting planned for the morning, an art-specific meeting planned for the afternoon, a tech-specific meeting Wednesday morning, and general wrap-up Wednesday afternoon before we left to return to WPI on Wednesday night.

Andrew then went into detail about the project. We would be working on the game Picturiffic, with the desired goal of porting the game from Facebook to iOS, using HTML5 and Javascript. He explained how the game was a variation of Hangman, where the player tries to guess a specific phrase with a picture hint covered by the letters used in the phrase sitting next to it. We discussed HTML5's weaknesses when it comes to iOS, and after deliberation with the New York group, agreed to report their research findings during the week after the trip regarding HTML5/Javascript and its compatibility with iOS native app development.

We got to sit in on some early morning meetings about the past week's sprint, which is a small set of short term goals that the team agrees to meet within a certain time period. This organization method is part of a long term project planning method known as "scrum". A scrum is an iterative method that allows the team to build a game slowly and react to feedback, rather than trying to plan the entire game all at once. They also included us in their sprint planning meeting, where they committed to goals for the upcoming sprint.

In the afternoon, Large Animal Games held an art meeting where they discussed the various implications for the art team in this project. Shiho Hoshino, the lead artist on the project, addressed the art guidelines for the game and the visual style they used to define Picturiffic. Shiho also addressed various research and design tasks that the art team would be doing. Some of those tasks included researching animation tools compatible with HTML5, designing the user interface to suit the different display size, and adapting the game to the play style that is inherent in mobile devices. They also set up a schedule to stay in contact and review work as the project progressed.

On the second day, we sat down with the Large Animal team to go over the technical specifications of the project. Yossi went over some of the guidelines and standards for coding at Large Animal and Stephen and Kevin went over the Large Animal wiki to understand the different technical aspects of Large Animal, such as TOGA. Yossi also talked about some of the potentially useful software to keep in mind when converting ActionScript code to HTML5 and Javascript.

After lunch, the two teams met to address our final comments and concerns about the project. Here, the two groups went over possible systems for project management and tied up loose ends for topics that trailed off during the other meetings. At this final meeting, we also went over the specific gameplay aspects of Picturiffic.

Picturiffic

Picturiffic is a Facebook game developed by Large Animal. The player is supposed to guess a specific phrase from a grid of letters that cover a picture. If the player guesses the next letter correctly, then it disappears from the letter bank, revealing more and more of the picture, until the letter bank is gone and the phrase is complete.

Behind the letter grid is a picture that is intended to give the player a hint as to what the phrase could possibly be. For instance, if the puzzle's phrase is "A Man's Best Friend", then the picture might be of a man petting a dog. This picture is covered by the letter grid, which can be anywhere from 4x4 tiles to 6x6 tiles. The area of the grid is always the same, but the size of the tiles inside the grid depends on the number of tiles required to complete the phrase.

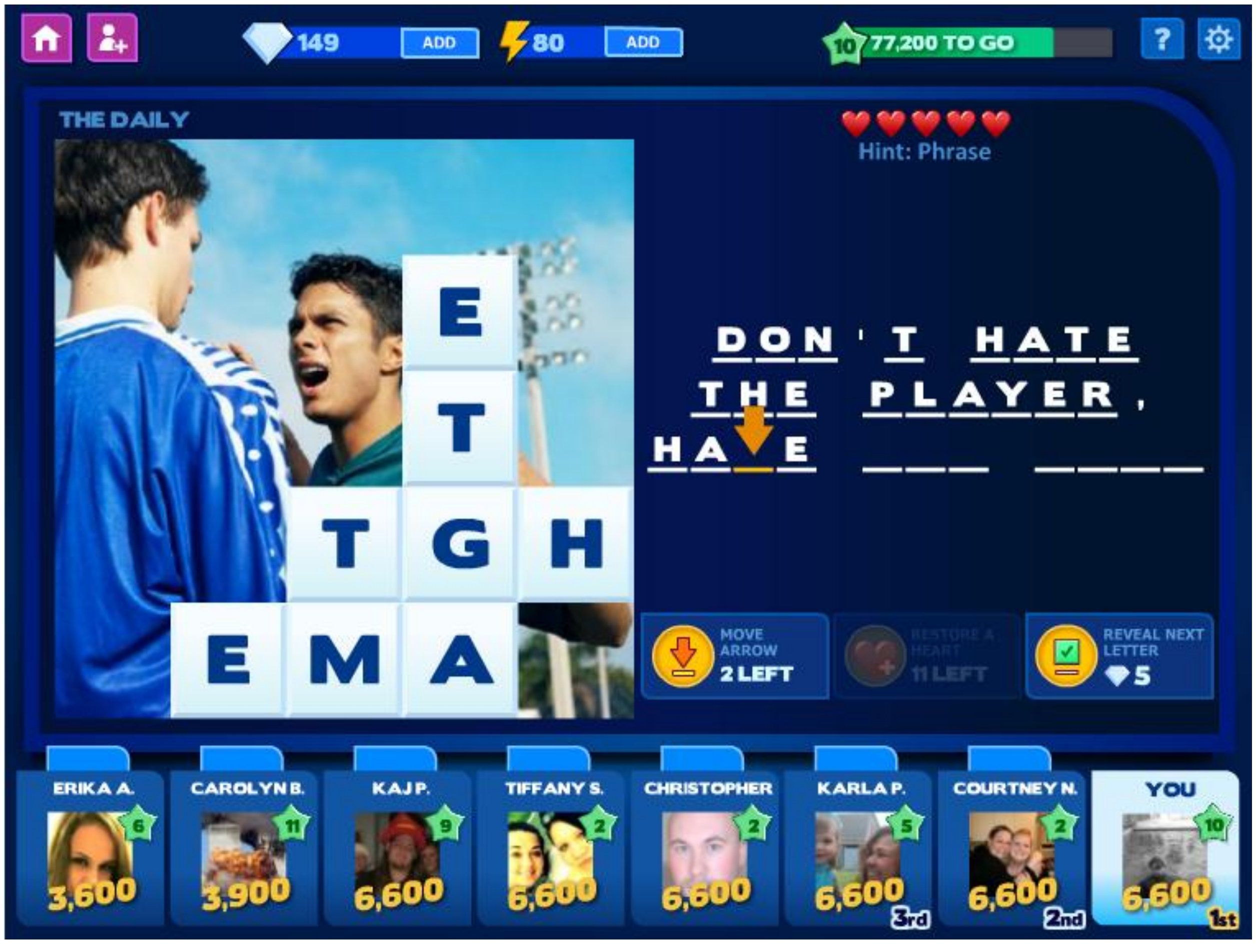


Figure 2: Picturiffic Puzzle Screen

Each day, the player is prompted to play the “Daily Puzzle”, which is a free puzzle that allows players to stack up against one another. Large Animal emphasizes the importance of the Daily Puzzle by bringing it to the player’s attention a few times as they start up the game each day. The player’s interest in the game stems from playing the Daily Puzzle once a day. The Daily Puzzle enables players to gain experience points without spending any of their resources, which fosters a sense of unimpeded progress which is highly rewarding to the player.

As they play the game, players are granted two different kinds of currency: energy and diamonds. Players use energy to play puzzles in Game Show mode. Diamonds are used to buy new categories of puzzles or lifelines that the player can use during a puzzle to make solving it easier. There are several ways to earn both energy and diamonds. At the beginning of each session, the player is prompted with a grid that contains prizes in the form of currency and lifelines. The more friends the player invites the more prizes there are in the grid. You can also receive diamonds for every extra life you have after successfully completing a puzzle. Additionally, after earning enough points by completing puzzles to level up, the player’s energy is replenished, to a maximum of 50. Players can also use real money to purchase energy and diamonds.

There are three types of lifelines to assist players: Reveal Next Letter, Move Cursor, and Regain a Heart. The Reveal Next Letter charm does exactly what it says, revealing the next letter in the phrase but awarding no points. Move Cursor allows the player to move the cursor to any unsolved space in the phrase, and then returns the cursor to its original position. Regain a Heart allows players to get another chance at solving the puzzle. During each puzzle the player is given five hearts, representing five chances to guess a correct letter. If they guess incorrectly, then a heart is lost, and if the player runs out of hearts they lose the puzzle.

Another important aspect of the game is the sense of community; that is, experiencing a sense of camaraderie by playing Picturiffic against friends and stacking up against each other. Large Animal's goal is for players to play a puzzle, see that they beat their best friend, and then post a notification to their wall, encouraging the friend to fire up the game and try to beat their score. It's in this spirit that Picturiffic has Leaderboards, a panel at the bottom of the screen which paints the player among seven others -- friends if they have played the specific puzzle previously or just other competitors otherwise. This gives the player a sense of competition when playing Picturiffic: instead of playing against the game and just trying to avoid running out of chances, the player must also consider the progress made by previous competitors and friends as he/she plays the puzzle. In addition to the competition Picturiffic offers, the game also rewards players who invite their friends to play. Picturiffic grants Energy and Diamonds to players who invite their friends to play, create puzzles for their friends, and create notifications on their wall to advertise to their friend list.

Players are given the ability to create their own puzzles to share with their friends in an opportunity to gain extra Energy and Diamonds each day. The first puzzle players create each day earns them an additional 10 Energy, and players earn 1 Diamond for each friend who plays their created puzzle. Create A Puzzle allows players to make puzzles by crafting phrases about their own pictures in their picture albums, or pre-determined pictures if they don't feel comfortable using their own pictures in a puzzle.

Art Research

Animation Software Research

Since the tech team planned on porting Picturiffic to HTML5, it was important that the artists find a program that was capable of producing suitable animations that would be compatible with both HTML5 and the power of the mobile devices the team was targeting. The artists initially invested their research into two programs; Adobe Edge and Sencha Animator, both of which seemed promising at first glance.

Adobe Edge was the first choice for producing the animations in Picturiffic. This was Adobe's premier HTML5 animation software, which is very similar to Flash. We found that the tools were very limited as far as reproducing the same effects as those used in the Facebook version of Picturiffic, but there was the possibility of baking some of the animated effects right into the static images, like the drop shadow for instance. Unfortunately though, due to its extremely underdeveloped state and its inability to cooperate with the tech team's tools, the team decided that the inherent risk of using a preview version of this software could compromise the goals of the project. The main concern was that since Edge was in a preview state, any update to the software during the course of the project could potentially render our previous work obsolete. (Adobe Labs)

Sencha Animator was the second most promising software that the team could have used to produce animations. It exported animations as style sheets in CSS3, which were compatible with HTML5 (Sencha Animator). While the software was far more developed than Adobe Edge, it was quickly evident that there was no logical way to address the animation code so that it would be malleable from the artist's side. CSS3 doesn't provide the level of control needed to manipulate animations in a game. Sencha Animator works well for content like ads, which can loop indefinitely, but not as well with games, which animate in response to user input. For this reason, Sencha Animator needed to be abandoned.

Another option for animating elements in the game was by way of procedural animation. This meant that the artists would generate a series of static images that comprised a whole animation and the programmers would use code to trigger and play the individual frames. This was a less favorable option because it's much harder to optimize animations this way, if we wanted to tweak different aspects. This also added an unforeseen amount of work for the coders, since the team couldn't anticipate how difficult it would be to optimize different animations.

After all the considerations were made for supporting animations in HTML5 it became evident that there wasn't a program that was reliable enough and that would be suitable to accomplish the project's goals. Every animation software that was specifically for HTML5 was still in a very crude and undeveloped state. We couldn't rely on experimental software to carry us through to the end of the project. With that notion in mind, the group began to look for alternative solutions to port Picturiffic to a mobile device.

Resolutions of Target Devices

Since we were going to deploy a version of Picturiffic on smart phones, one primary concern was the varying resolutions and aspect ratios of all the different mobile devices on the market. Since the primary target was iOS, each Apple product was analyzed for their technical specifications. We found that for the first three generations of the iPhone the dimensions were the same at 480x320 pixels with an aspect ratio of 3:2. The iPhone 4 resolution was double in size at 960x640 pixels but still maintained the same aspect ratio of 3:2. Essentially this just meant that the pixel density on the screen was higher. The iPad and iPad2 however, had a larger resolution and different aspect ratio at 1024x768 pixels and a ratio of 4:3. (Wikipedia) Considering the end product would ideally support all of these resolutions we had to consider how we were going to handle the resizing of assets and the interface in game. This brought about the concern for using bitmap versus vector assets.

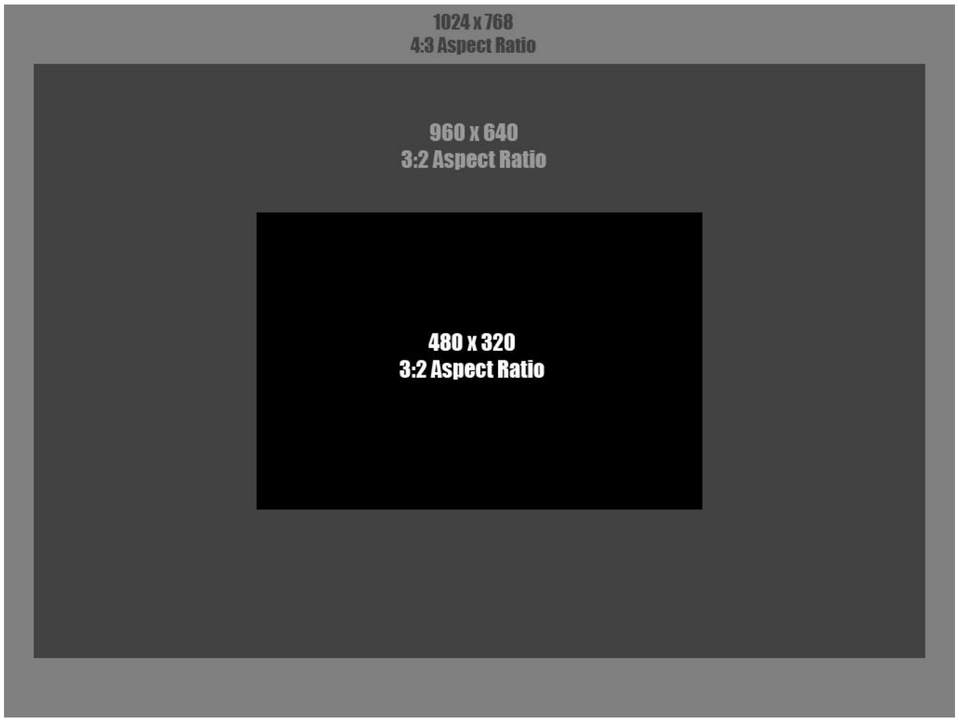


Figure 3: Various iOS device resolutions

Vector vs. Bitmap Assets

In order to maintain the quality and visibility of the game's assets they were going to have to be re-sized to support the varying display properties of the different devices. As a result the team had to assess the optimal format for their assets that would yield this promising retention. The burning question was whether or not we would use vector images or bitmap images.

Vector images are mathematically calculated lines and curves that are independent of their resolution. Resizing a vector image changes the equation that draws the different elements of an asset so there's virtually no loss in quality when you scale that image up or down. The complexity of the image determines the size of the file - the more detail, or vector shapes, that you have the larger the file size. For bitmap images it's not the complexity of the image that determines the file size but the actual dimensions of the image itself. (Anonymous)

Bitmaps are spatially mapped arrays of pixels, which don't use equations to store the image like vector images. This makes them less flexible than vector images in terms of rescaling the image. Generally, bitmap images can scale down without a loss of quality, but when they're scaled up the individual pixel information is magnified making the image appear blurry and "pixelated". Their ability to achieve subtle color variations make them very useful for representing photo realistic images. (Anonymous)

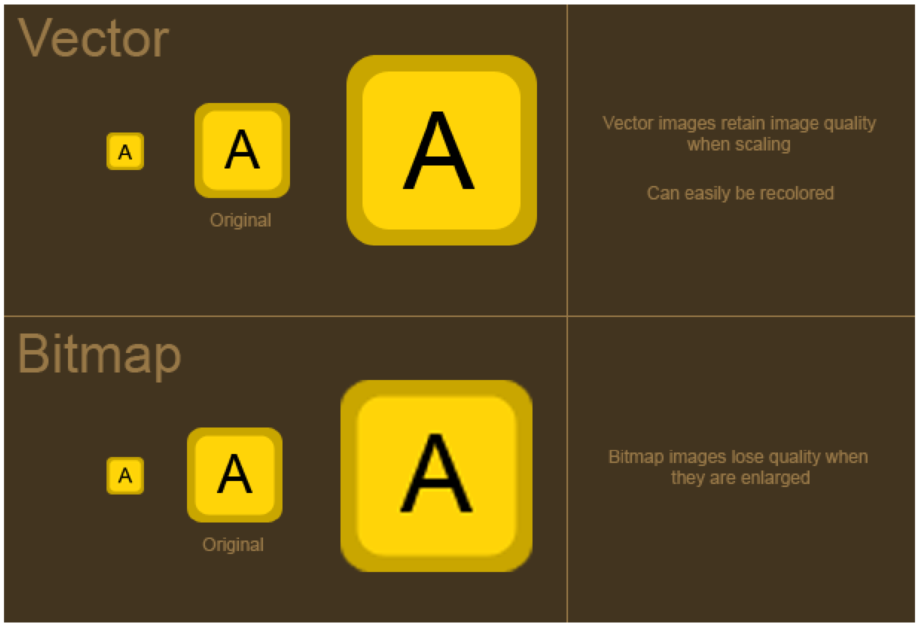

Vector
A
A
Original
A
Vector images retain image quality when scaling
Can easily be recolored
Bitmap
A
A
Original
A
Bitmap images lose quality when they are enlarged

Figure 4: Comparing Bitmap and Vector Assets

Each format had notable pros and cons that made it difficult to determine which was better than the other. It seemed that we would have to carefully choose assets that would benefit from being one format rather than the other. Since most assets would have to be scaled up to support the mobile device resolutions, the vector format seemed like the best choice. But since Picturiffic also uses photographs, we would inevitably have to use bitmap images at some point. There was also the question of performance and how drastically the different image formats would affect the overall quality of the game. We concluded that it would be best to use both bitmap and vector images in order to create a balance between the quality of the assets as well as the performance of the game.

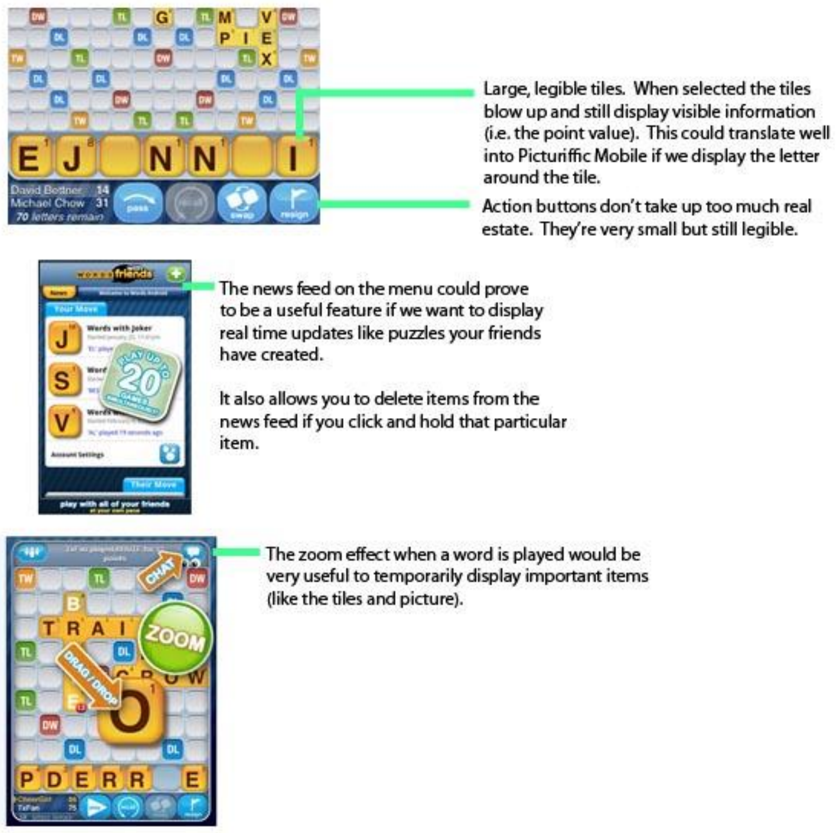
User Interface Design Research

As part of the initial research the artists investigated a variety of popular mobile games available on Android and iOS devices. The most influential games were Words with Friends, Hanging with Friends, Angry Birds, Plock and Text Twist 2. Playing these games helped to assess the different ways in which the team would be able to think about and iterate the redesigning of Picturiffic's user interface. Things such as layout, menu nesting and button behaviors were a few key characteristics that were being analyzed. We also looked into the psychological implications of user interface design and how we could use design principles to create an intuitive interface.

Words with Friends (Zynga)

Words with Friends is a mobile version of Scrabble that you can play against contacts on your phone as well as friends from social networking sites like Facebook. Words with Friends' overwhelming popularity as well as intuitive design made it a key game to aspire to in terms of the interface design. It was also beneficial that Words with Friends was a text based game that used letter tiles just like Picturiffic.

The main feature that we liked was the size and legibility of the tiles. When they are pressed, and essentially covered by your finger, they blow up in size so that critical information like the letter's point value is still visible. Similarly we could use a design feature like this to display the letter outside of the tile space the user covers up with their finger.


Large, legible tiles. When selected the tiles blow up and still display visible information (i.e. the point value). This could translate well into Picturiffic Mobile if we display the letter around the tile.

Action buttons don't take up too much real estate. They're very small but still legible.

The news feed on the menu could prove to be a useful feature if we want to display real time updates like puzzles your friends have created.

It also allows you to delete items from the news feed if you click and hold that particular item.

The zoom effect when a word is played would be very useful to temporarily display important items (like the tiles and picture).

Figure 5: Words with Friends interface analysis

We also liked how the text and action buttons were small enough to preserve valuable screen real estate but large enough to still be legible and usable. This was a notable feature that could translate well to elements in Picturiffic like the leader boards and lifelines.

Angry Birds (Rovio)

Another popular game that we chose to research was Angry Birds. While the actual game play interface was minimal and seemingly unrelated to Picturiffic it offered great reference for menu design and nesting elements in collapsible menus.

Since we were going to have to be very careful in how we allocated screen space in the mobile version of Picturiffic, having collapsible menu or nested elements inside a menu was going to be essential.

Angry Birds does a great job of using non-intrusive collapsible menu buttons with simple graphics and animations to hide functions like toggling the sound on and off and help information. They also use the in game pause menu to display similar buttons as well as access to the level selection screen and a button to restart the level. Player information is also only displayed in a pop-up menu after a level is complete.

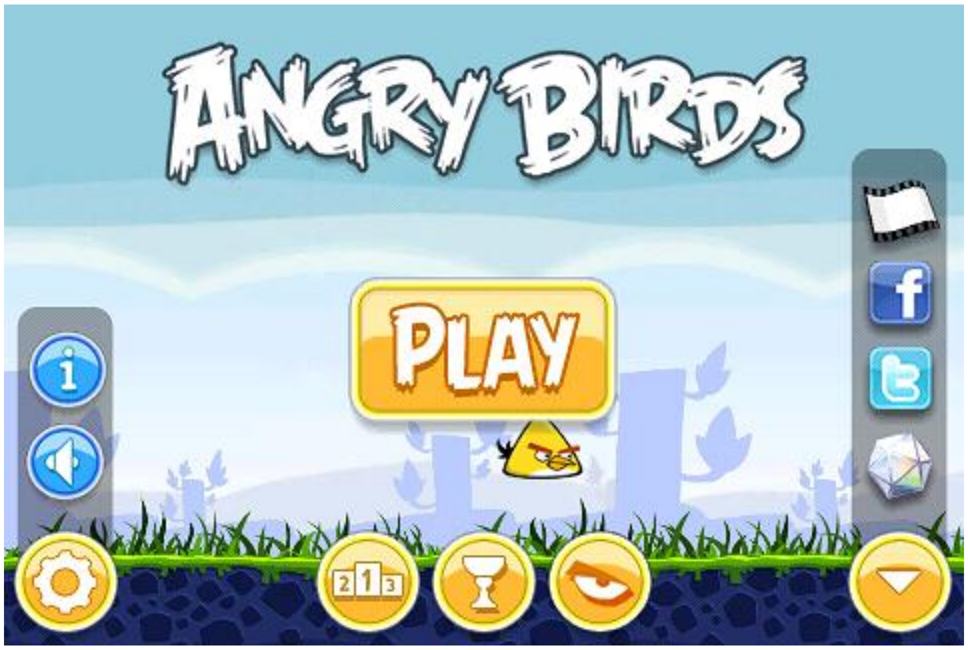


Figure 6: Angry Birds menu screen

One thing in the Facebook version of Picturiffic that takes up valuable screen space is all of the player information. Relocating this information inside of a collapsible menu button like Angry Birds may allow us to use that space in a more effective manner.

Text Twist 2 (RealNetworks)

Text Twist 2 was another game we decided to look into for reference for user interface design. This was another good reference because it too was a letter-based puzzle game and its game play mechanics were similar to Picturiffic.

One of the first notable features was the menu. Rather than having to select a specific element in the menu list you could select a space outside of the menu and a bar would highlight the item in line with your finger. This virtually eliminated any concern for precision with a touch screen system.

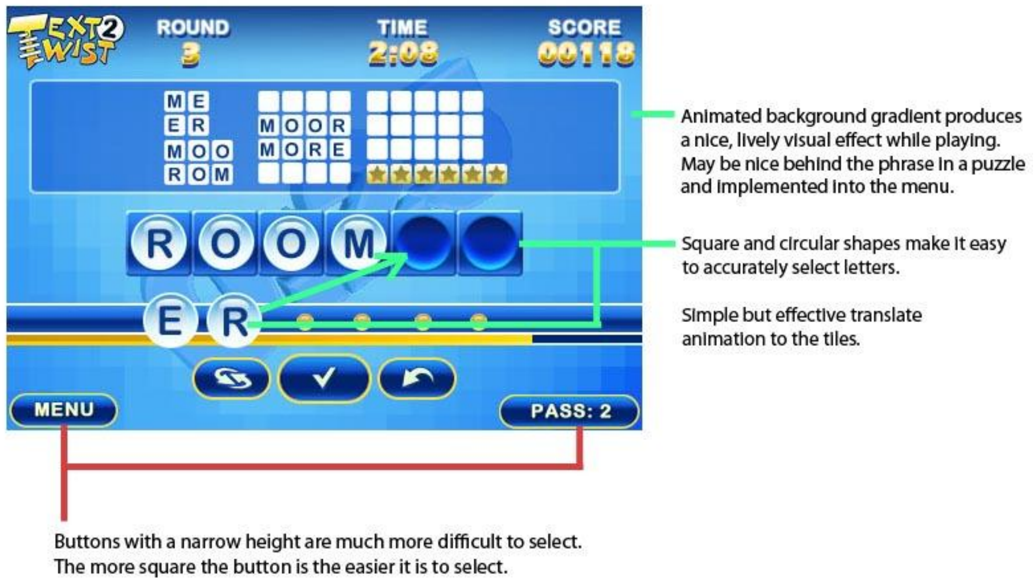

TEXT 2 TWIST
ROUND
3
TIME
2:08
SCORE
00118
MENU
PASS: 2
Animated background gradient produces a nice, lively visual effect while playing. May be nice behind the phrase in a puzzle and implemented into the menu.
Square and circular shapes make it easy to accurately select letters.
Simple but effective translate animation to the tiles.
Buttons with a narrow height are much more difficult to select. The more square the button is the easier it is to select.

Figure 7: Text Twist 2 interface analysis

While the layout of Text Twist 2 is unlike what we envisioned for Picturiffic it still offered some notable features for the game play screen, specifically concerning visual effects, animation and button sizing and shape.

The background of the game play screen had an animated gradient which conveyed a lively and visually pleasing aspect to the screen. While this is a fairly minor detail for the core game experience itself it is a nice, subtle effect that would blend nicely with the style of Picturiffic.
Another notable feature was the size and shape of the letter tiles. Their round and square shapes coupled with their visually prominent size made them easy to select without error. They also implemented a very simple translate animation so when the tile gets selected it “flies” to its container.

On the contrary, we noticed that some of the menu and action buttons had a flatter, more rectangular shape which made them more difficult to select compared to circular and square buttons.

Overall, Text Twist 2 left us with a good sense of how we would approach the shaping and sizing of buttons as well as aesthetic details like the background and letter tile animations in Picturiffic.

Plock! (Metrogames)

An additional reference game was Plock!, which provided examples of several more features that were under consideration for the mobile version of Picturiffic. Plock generally made excellent use of screen space, with the vast majority of the screen being taken up by game play elements or menu buttons.

In addition, their buttons changed size and color nicely in the menu, which made it very obvious which button you were selecting. This consequently made the menus easy to use and visually appealing; a feature we wanted to replicate in some fashion.

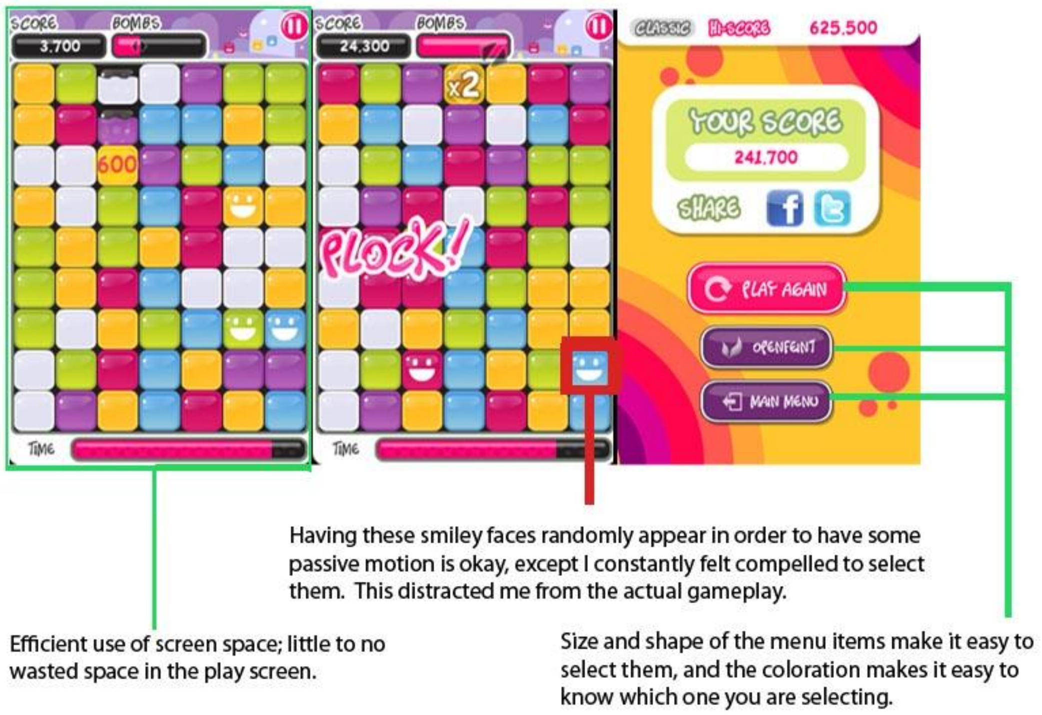

SCORE
BOMBS
3,700
600
SCORE
BOMBS
24,300
x2
PLOCK!
TIME
TIME
CLASSIC
HI-SCORE
625,500
YOUR SCORE
241,700
SHARE
PLAY AGAIN
OPENFEINT
MAIN MENU
Having these smiley faces randomly appear in order to have some passive motion is okay, except I constantly felt compelled to select them. This distracted me from the actual gameplay.
Efficient use of screen space; little to no wasted space in the play screen.
Size and shape of the menu items make it easy to select them, and the coloration makes it easy to know which one you are selecting.

Figure 8: Plock interface analysis

As a final note, the game had background motion that gave a nice sense of energy and life to the screen. Unfortunately, they emphasized this motion to the point where it became distracting from the game elements and game play itself. We hoped to utilize motion to a point that it would make the screens more visually exciting, while avoiding distraction from game play.

Hanging with Friends! (Zynga)

Hanging with Friends also gave us several ideas for our implementation of Picturiffic Mobile. The size of their letter tiles was very usable, allowing them to be easily read and selected but small enough to leave space for the rest of the game elements.

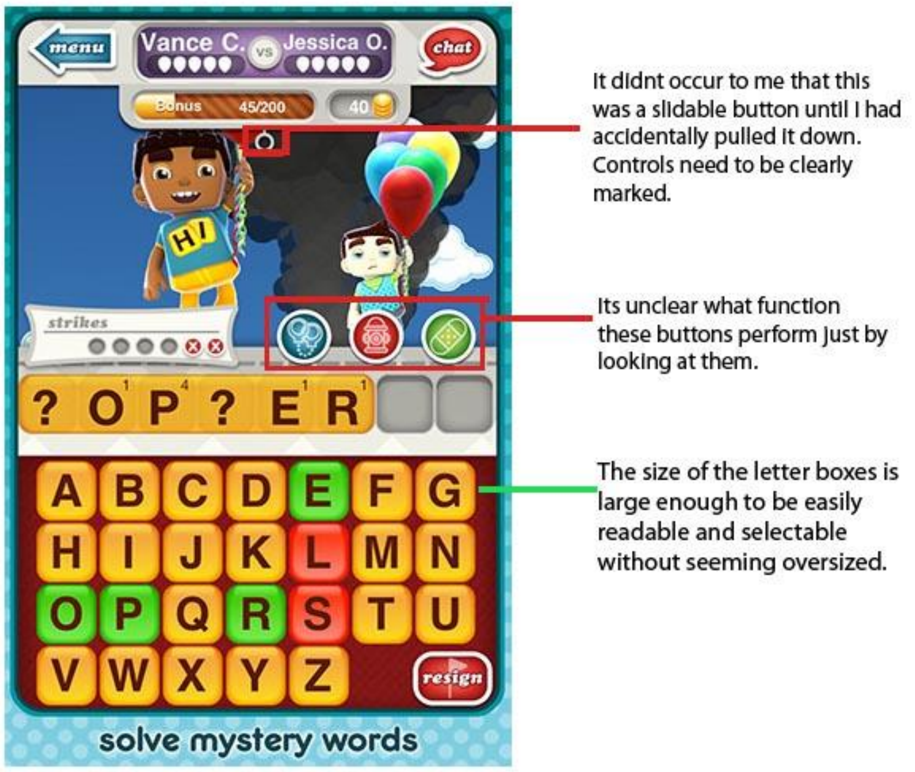

menu
Vance C.
VS
Jessica O.
chat
Bonus 45/200
40
It didnt occur to me that this was a slidable button until I had accidentally pulled it down. Controls need to be clearly marked.
strikes
Its unclear what function these buttons perform just by looking at them.
? O P ? E R
The size of the letter boxes is large enough to be easily readable and selectable without seeming oversized.
resign
solve mystery words

Figure 9: Hanging with Friends interface analysis

They also used a system of expandable and retractable menus which saved a large amount of screen real estate for menu items that did not need to be on the screen at all times. This idea was very applicable to Picturiffic Mobile, as our screen space is very limited.

There were also several features which did not present themselves very well. The slider that was used to bring down the expandable menus was not an obvious button and we did not know it existed until we accidentally used it. This was a good example of why buttons need to express their functionality clearly; otherwise players may not know to use them.

Along a similar line of thought, Hanging with Friends also used several buttons that contained no text, relying instead on a small icon to represent what they did. The problem was not that they didn't use text, but rather that the chosen icons meant very little to us as new players. We realized that we would have to be very careful with our icons to make sure they would be understandable.

Designing with the Mind in Mind

Another realm of visual design that the artists explored was the psychological and rhetorical implications behind user interfaces. By understanding how the human brain and visual center operate they would be able to cater to the natural tendencies of users to create a more intuitive and effective interface design.

Our most basic and obvious intention for the user interface design was to make it as intuitive as possible. In order to learn which design principles would allow us to do so we looked at a book called Designing with the Mind in Mind by Jeff Johnson. This book was filled with essential guidelines, both psychological and visual, for building a user friendly interface.

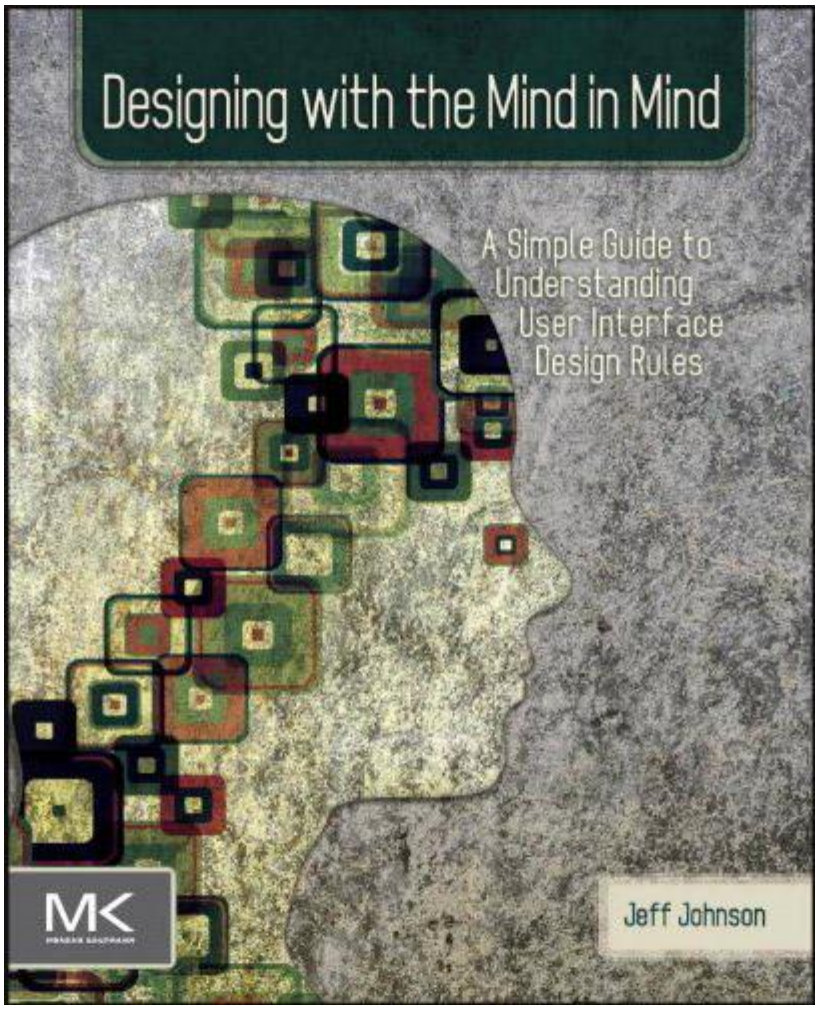

Designing with the Mind in Mind
A Simple Guide to Understanding User Interface Design Rules
MK
Jeff Johnson

Figure 10: Designing with the Mind in Mind cover

The tricky part of this design process was that we were working within the confines of an already established art style. This meant that basic guidelines in the book, like those about color, would be out of our control. Instead we looked towards design principles associated with the layout, structure and textual aspects of the interface, and even those were already prominent in Picturiffic.

The book asserted that as humans we have a natural tendency to organize visual information as well as simplify it for our own understanding. The Gestalt principles alluded to how one can use this innate behavior to both avoid bad design and improve good design. We found that the proximity, similarity and figure-ground principles were the most applicable to the work we would be doing. (Johnson)

The proximity principle states that the distance between objects can determine how we interpret their functions and whether or not we perceive them as groups (Johnson). This made us consider the spatial relationships and boundaries between different objects in the interface. Something like the Daily Puzzle box that we designed for the home screen groups all of the relative information to the Daily Puzzle together. This prevents the user from confusing things like the timer with other game modes. We wouldn't want them to think that they couldn't play any game mode at all until the timer was reset. We also apply the proximity principle in our settings menu design. Since we relocated all of the player information into this menu it was essential that we displayed it as a group. By confining the different player currencies and level status bar to a separate bounding box on the screen it's made clear that they're all related chunks of information.

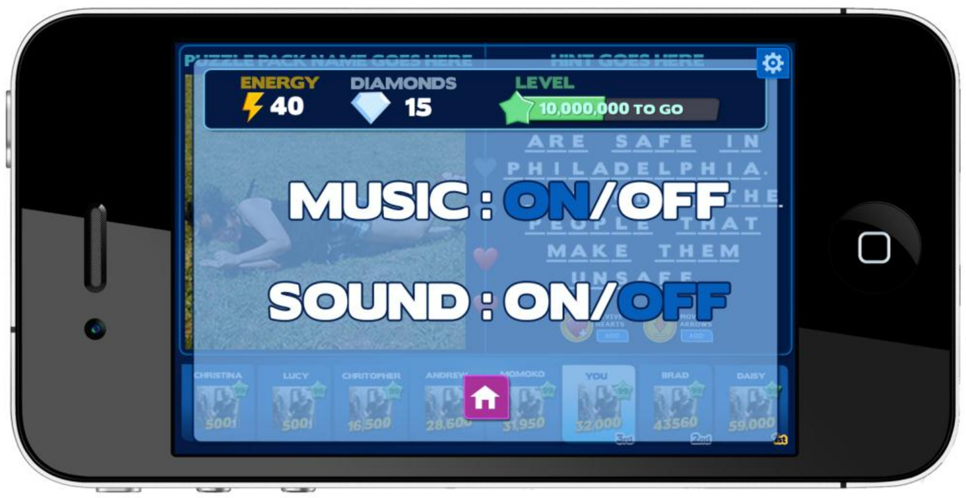


Figure 11: Grouping Player Information using Gestalt Principles

Elements of proximity were also already implemented in Picturiffic. Things like the hearts and charms were all closely grouped together with one another which made them seem related in function and purpose.

The similarity principle states that objects that are similar in appearance appear grouped. This is important for the user because it allows them to easily identify the functions of different elements on the screen (Johnson).

This principle was somewhat already established in Picturiffic. Things like buttons all had similar shapes, with two rounding corners and two sharp ones. This allows the user to quickly realize what a button is on the screen versus what isn't. It was important for us to maintain design principles like this because it not only allows new users to more easily interpret objects on the screen but it also allows previous users of the game to readily jump into the game play because they're able to recognize the features they're already familiar with.

Since some elements in the interface already adapted both of these principles in the Facebook version of Picturiffic it was important for us to maintain a consistency throughout our new designs. Without consistency we would be wasting the user's brain power on figuring out what does what rather than playing the game itself.

One last principle that we adopted was the figure-ground principle. This states that we naturally split our visual field into figures (foreground) and the ground (background). By careful visual manipulation we can bring elements to the foreground temporarily, focusing the user's attention for a brief moment. (Johnson)

The most notable use of this principle is in our settings menu design. When the user clicks the settings button a semi-transparent box opens up over the game board (which is now temporarily the background) and focuses the user on the new foreground elements that make up the settings menu. This also helps maintain the orientation of the user and their goals because it doesn't directly replace the existing information (i.e. the game play) but instead temporarily hides it behind a transparent box.

As stated previously this was a tricky task because we were working with an established design. We had to maintain a certain consistency with things like color, shape, spacing and animations but at the same time we had to implement this consistency in a new visual structure. There was an intimate balance between the consistency and intuitive nature of our designs. Often we would try to make minor changes to the art treatment to make the design more intuitive but that would detract from the original art conventions of the game. So the challenge really stemmed from the necessity to redesign an existing design while staying true its original nature.

A few other notable ideas that came from this book involved our tendency to perceive structure. By creating a visual hierarchy in an interface the user is more able to scan and interpret information presented to them (Johnson). In the original art treatment for Picturiffic there were quite a few visual elements that established a visual hierarchy. An example is the use of the purple button. Purple buttons are meant to appear as a more important action that a user can take. Another example are the exclamations. These are the chunks of information that fly into the screen and often take prominence as the foreground on the screen. The combination of their appearance and their animation make them seem like a higher order in the visual hierarchy.

Another challenge we had was the textual aspect of the interface design. Naturally, we're not wired to read so poor design and wording of textual information can often detract from the intuitive nature of an interface. The author suggested minimizing the need for the user to read, which is also something echoed by Large Animal Games. He also suggests to avoid complex or tiny font, patterned backgrounds and centered blocks of text, and instead, enforce the reading by using plain language, saying what you need to say in a mild and simple manner, as well as using a font format that

All of these principles serve to create a more clear and concise communicative style of information. It is not the individual principles that accomplish this thought. It is actually the fusion of these principles which allows the design to communicate itself in a way that is natural to understand.

Technical Research

Changing Focus

While the artists were researching their tools and design choices, the tech team researched the feasibility of HTML5 development. As it turns out, MobileSafari, Apple's mobile web browser, does not fully support HTML5 due to concerns about the amount of data transferred by web pages (Apple Inc.). Specifically, MobileSafari includes some restrictions on audio played by websites. First, audio files may not be pre-loaded or played automatically (Apple Inc.). They may only be played as the result of user input, such as a button press. Secondly, only one audio file may be loaded at a time. This prevents the use of simultaneous audio, such as background music and sound effects playing concurrently (Apple Inc.).

The solution to these problems was to release the game as a native iOS application ("app"), parallel to its HTML5 release. One option for supporting this parallel release was to use a piece of middleware, such as Phonegap. Phonegap takes HTML and Javascript files and compiles them into a file usable by mobile devices, such as .apk files for Android devices, or .ipa files for iOS devices (Adobe Systems Inc.). Because Phonegap compiles the files for the web version into the mobile device's native language automatically, the code only needs to be written once, and could then be deployed in multiple ways.

With a workaround in place, we turned to the task of actually generating code. At our sponsor's request, we experimented with the Google Web Toolkit (GWT), which compiles Java code into Javascript for use with HTML documents on websites. GWT itself worked well and allowed us access to HTML5 elements such as the canvas. The HTML5 canvas is one of its biggest features. It allows the developer to draw shapes and colors on a virtual space in creative ways quickly and easily with code.

The next step was to create a pipeline to integrate art assets created by the art team, such as animation, with the code generated by GWT. We looked into two HTML5 animation tools: Adobe Edge and Sencha Animator. Adobe Edge stood out as a very promising tool. It provides artists with tools similar to those of Adobe Flash, such as the timeline and stage, and outputs animations as HTML5 webpages. However, Edge is still in a preview stage, and not complete enough to integrate its animations into other projects. Because it can only output Javascript and HTML, Edge doesn't interface well with GWT. Additionally, the animations need to be static; there isn't yet a way to use dynamic elements from the page. Because of this, the team had to abandon Edge despite its potential. Sencha Animator is another HTML5 animation tool in development. However, Sencha creates animations through CSS3, and appears to be much better suited for content like advertisements, which can be played continuously and don't depend on user input.

We didn't like any of the options available enough to commit to any of them, so we explored other options. The final HTML5 approach we considered was procedural animation. The artists would plan out the timing and positioning of animations, and the programmers would hard-code them into the game. In the end, we decided that this route would devote too much of our time to working out animation and not enough to making the game fun, playable and polished.

One of the main issues we ran into during this initial research stage was completing the circuit of the pipeline, because the only Macs to which we had access were not sufficiently upgraded for mobile development. This was because the Xcode version and the current OS version were not updated sufficiently to support mobile development. Since these Macs were in the IMGD Lab and owned by WPI and not the project team, we were unable to get quick turnaround on getting the appropriate software upgrades, which hindered our ability to research the complete pipeline efficiently.

Switch to Flash/AIR

With no clear pipeline to unite the code and art assets, the team decided to pursue Flash development instead of HTML5. While MobileSafari doesn't support the Flash Player plug-in, it is possible to use Flash to create native apps. In September 2010, Apple lifted the restriction on third-party app-creation software. As a result, Adobe added a packager to their AIR platform to allow programs written in Actionscript to be released on iOS devices.

After meeting with Andrew and Yossi to present our research and conclusion, we shifted gears from HTML5/Javascript to inheriting their existing code base and preparing it to funnel through Adobe AIR's mobile API, which we would then deploy on iPhone.

Flash apps on iOS work differently from other uses of Flash. Normally, Flash compiles and executes actionscript commands at run-time. This is how Flash works on traditional web-browsers and in apps published through Air for Android devices. Apple, however, forbids the Just-in-Time compilation actionscript usually relies on. Instead, when code is published through Air for iOS, it is compiled into Objective-C, making it compliant with Apple's app requirements (Holguin).

After a discussion with Yossi, the tech side decided that the best course of action would be to split the effort into two separate focuses: one would work with the artists on making sure that the new UI for mobile worked correctly and had the correct specifications, and the other would work with the backend code, prepping it for AIR and the export into iOS. Additionally, due to the extremely large code base, both tech members had to take it in and understand how it all tied together, especially since the way Large Animal uses Flash was different than the team was used to.

Tech Conventions, Flash Formatting and Structure

Large Animal Games is a company that has been around for a decade; as a result, they have a well-defined coding standard documented for any new employees that join their team. Their documentation is password-protected and mostly confidential, but they have permitted us to discuss some aspects of their tech conventions.

They have several pieces of documentation dedicated to rather universal standards of coding - naming conventions, formatting, layout, as well as specific instructions depending on the language in use. Most of their games use Actionscript, Javascript, or Java, but they do use PHP and C++ occasionally.

After we went over their documentation requests for universal coding standards, there were quite a few documents on their company-specific tools, like TOGA. TOGA is their network tool that allows them to sync their games with multiple platforms seamlessly without reproducing too many assets. The documentation involving TOGA was well in-depth and explained how we would expect to see TOGA used, which benefited us in understanding the code base more.

Adobe Flash is an extremely useful tool, allowing developers to create robust and exciting applications using simple transitions, animations and tweens to animate movie clips, buttons, and graphics. Actionscript allows programmers to take art assets that designers/artists have created and manipulate them with calls and functions designed for simplicity. The remarkable thing about the program is that it leaves significant room for improvement with a developed API in Actionscript 3.0, so it is possible to develop a software system that handles large game undertakings.

Large Animal Games has developed a large number of games, many of which use a similar code base called Songla. Most of their Flash games have a common code base that contains logic that is reused by each game, such as physics, audio, basic display, events handling, and state machines. On top of this common code base, Large Animal develops game-specific logic, based on the game. For example, Picturiffic's gameplay takes place mostly in the UI and the puzzle, so most of the top-end code pertains to either UI logic or puzzle-solving logic.

Picturiffic has several .swf files that relate to it: game.SWF, gameAssets.SWF, audio.SWF, Loader.SWF, and intro_animation.SWF. If the game runs in a browser, the Loader.SWF is the initial file that starts up, which calls intro_animation.SWF to display to the user while game.SWF and gameAssets.SWF are brought into the stage to load. After these two files are finished loading, the intro_animation.SWF finishes, and the game is started up.

The game.SWF is compiled by building the code base inside of FlashDevelop. It essentially builds an empty stage with all the code in the background (which calls gameAssets.SWF onto the stage). gameAssets.SWF is compiled by publishing the game.FLA that the artists work in constantly, editing layers and establishing the aesthetics of the game. When game.SWF is loaded, it brings gameAssets.SWF into its stage. gameAssets.SWF is essentially the same file as game.SWF except that instead of an empty stage, gameAssets has all the assets for Picturiffic on the stage, and instead of a full code base, gameAssets has code stubs loaded in it (which is basically just headers for each function so that it thinks the function is all there, when it's just a shell). When game.SWF loads gameAssets.SWF, the stage is replaced, but when gameAssets.SWF tries to replace the code base, game.SWF basically says "We've already got code here, so we win" and gameAssets.SWF's code disappears.

Technical Development

Moving to AIR

Once we decided to use Adobe AIR instead of HTML5, the existing codebase needed to be modified to run on its own, independent of any website. To that end, we configured the game to be compiled by a script provided by the FlashDevelop toolkit/editor. This allowed the various .swf files to be packaged together for installation as a standalone game.

The next thing that needed to be done was to find a way to feed the game data about the player. Normally, the website provides this data via the player's account, such as on Facebook. While running as an independent app, however, the game cannot access any sort of account like that. The solution we found was to have the game behave like it does during debugging and read in data from an xml file.

At this point, the game was able to run as a stand-alone application on desktop computers. The next step was to port it to iOS, which required a bit more work. While the core code remained the same, its organization became a problem. Originally, the game was composed of 5 .swf files: Loader.swf, Intro_animation.swf, Game.swf, GameAssets.swf and Audio.swf. As it turns out, AIR on iOS doesn't run the same way as other versions of the platform due to restrictions from Apple (Holguin).

Normally, AIR uses .swf files as needed, executing actionscript code at runtime. iOS apps forbid just-in-time compilation, which is how scripting languages like actionscript run their code. Therefore, AIR must compile the game into bytecode, including the entire AIR runtime into each game. In contrast, AIR is installed as its own application on Android or desktop machines, which loads .swfs as needed.

AIR's compilation process can only use code from a single .swf file (Holguin). This prevents additional .swf files from serving as anything other than sources of assets. Any code in the additional files is simply ignored at runtime.

With this in mind, we did away with Loader.swf and Intro_animation.swf. Since we feed the game fake player data via the xml file, we don't need the loading sequence to request this data, and can jump right into the game itself. The drawback to this approach is that the game does take a while to load on iOS, and without the animation sequence can appear frozen to users.

The biggest concern with the iOS build is its memory usage. Because of the organization of gameassets.swf, all the objects in the game are instantiated as soon as the game loads and are simply moved on- and off-screen as needed. The ideal method of making the game for iOS would involve converting gameassets.swf to a .swc library and instantiating instances of each symbol as needed (Krcha). However, the amount of work this change would entail and the late discovery of its performance benefits made it infeasible for this project.

Stand-alone Proof of Concept

We had to convert the UI from the desktop resolution to mobile, so we devoted some time to developing a stand-alone application that demonstrated how elements such as buttons and pictures would look on the mobile device to give the artists an idea of how the main Flash file needed to be structured to accurately depict the Picturiffic application. Stephen did most of the work in this aspect, as Kevin was working on the back-end for the project.

This stand-alone application was developed for Android, purely due to convenience -- Apple requires at least a developer's certificate to publish applications to iOS devices, where Android has a much more relaxed system when it comes to debugging and testing applications locally.

The stand-alone application demonstrated a few features; first, it showed how pictures and buttons displayed on the screen; this way, the artists could give their constructed elements a specific height and width and see how that translated onto the mobile device. Additionally, the application demonstrated sample code that tested the functionality of touch methods; for example, long press or swiping. We didn't know if we would use any of these touch features, but we implemented them in case we ever needed to come back to them in the future.

The stand-alone application allowed Stephen and the artists to work separately from the back-end code while Kevin worked on the shift from the desktop application to the Adobe AIR version that would be published to the mobile device. This split allowed two different aspects of the project to run simultaneously while still keeping the merging of assets relatively simple.

Large Animal Games Flash Standards

Large Animal Games Art Conventions (Games)

In order to create an end product that would reflect the original feel and nature of Picturiffic we needed to familiarize ourselves with the conventions and style that Large Animal Games created. While we were visiting Large Animal Games in New York City we held two separate meetings, one for the art team and one for the tech team. In these meetings the Picturiffic team shared design documents and some company wiki pages that went into detail about the different conventions they used related to specific things like the art style of Picturiffic all the way to general guidelines for something like the library structure in Flash.

For the art team we were primarily concerned with preserving the art style which was done by studying and mirroring Shiho's vision. We found out that doing so was not as easy as it seemed.

Shiho shared a document with us that outlined the vision statement and design principles, as well as file management for the repository, FLA structure and game architecture tools. It also discussed the User Interface conventions as they fit within the confines of Facebook.

Without exposing the specific details of the document, it began by establishing what Picturiffic was as well as what it wasn't. This helped set the stage for the driven purpose and feeling behind the game. It then went on to a section that established some basic visual principles used throughout the game. These included techniques to emphasize social elements, important actions, reduce copy and repetition. This section alluded to ways in which Picturiffic is made to be intuitive as well as innovative with its design. Among the most critical of principles was visual reinforcement of text and creating a visual hierarchy so as to influence to user to realize what's the most important.

The next section was more technical and concerned with file management and performance. It highlighted that understanding how the repository works was essential because it would allow us as a team to work on the same files in an organized fashion - making it easier to develop and maintain. It also briefly mentioned to concern ourselves with performance issues and the target file size for SWFs. This section of the document was more tuned to the nature of making sure that we're aware of these things, rather than a "how-to" guide.

The last section was concerned with the interface conventions for Picturiffic. While we would be developing for a different platform it still contained useful information that we could translate to our project. It talked about the different typefaces and their implications as well as color schemes and how each color could be used to create a visual hierarchy of information. It also went into detail about the styling of animations which outlined how animations should be used to reinforce social activity, create a sense of a dynamic environment and how they should coincide with the way their asset looks. It also discussed implications of different aspects of the interface and game play. Things like the differences in buttons, the use of exclamations and the player panel were among the elements discussed. It was evident that each design choice had a specific reason or goal behind it.

While some of the features highlighted in this document were outdated it still offered us great insight to the thought and purpose behind Picturiffic's art conventions, both visually and technically. All of this information served as a useful guide for us when we started designing the new interface for the mobile version.

Large Animal Games Flash Format

Since we decided to make the switch from HTML5 to ActionScript and Flash it became necessary for the art team to familiarize themselves with the main FLA and its structure since we would be using the existing content from Large Animal Games.

Essentially the entire game, as far as screen content and animations, is contained inside one .FLA. This main FLA contains no code and instead uses external code to manipulate the game. This is made more feasible by a strict naming convention for objects and layers within the FLA file, and a file hierarchy for the objects in the library. Each object is also on its own layer, in order to facilitate the finding and animating of individual components. There is also heavy use of nested objects, with potentially dozens of objects nested within larger objects to create a tiered structure that naturally organizes the game in a logical manner. This structure is made of four main tiers, or levels of detail within the game's FLA.

The first tier of the FLA consists of the GUI layer and several layers with debugging interfaces for developer use. On the GUI layer is the GUI object which consists of every object in the entire game.

The second tier is within the GUI object. Here you will find objects which represent each game screen, dialog box, and navigation panel that you will see in the game. There is no animation mapped to the timeline on this tier, because all screens and dialogue boxes contain their animations on their own timeline.

The third tier is made up of the contents of all objects in the second tier. Generally this is the tier in which you will find timelines with screen transition animations, buttons, and most other individual game element objects. The majority of animation in the game is here on this tier.
The fourth tier is inside those objects. This is where you will find the inner workings of the simple game objects. An example of what you could find on this tier would be a button, which maps each possible button state to its own individual timeline.

User Interface Design and Development

Designing the User Interface

The new interface for Picturiffic Mobile had to be both visually appealing and practical to use on the smaller screens of a mobile device. This meant that text had to be large enough to read without filling the whole screen so that the buttons and other game elements would have enough space on the screen to be usable. The size of the buttons also played a major role in our design, since they had to be small enough to fit together on the screen while being large enough to be easily selectable via the imprecise touch input that the mobile devices use. If the proper balance in size wasn't found, then it might result in an interface that causes frustration for the player.

Because the mobile version would have a smaller screen size, all of the game elements needed to be expanded to take up a larger portion of the screen in order to meet the size requirements that we had set for usability. In order to make these changes to the UI, we had to remove some aspects of the original Picturiffic design that were purely visual in nature. This mostly affected the design of the Home screen, where large portions of the original Facebook version were taken up by art that didn't contribute to the functionality of the design.

There were several aspects of the original Picturiffic's Puzzle screen that were not conducive to the mobile version's screen size either. The inclusion of experience, diamond and energy counters was an unnecessary use of valuable screen real-estate. Because those resources are not used during play, they could be removed from sight and placed in a collapsible menu, thereby freeing the space for other elements to expand into.

Another inefficiency of the original design was the lifeline buttons. Each lifeline had a use button in addition to an "add" button, which was used for buying more Lifelines if you ran out. Since the user wouldn't be able to purchase lifelines in our build, the add buttons were taking up unnecessary space.

To successfully port the game to mobile devices, all of these inefficiencies had to be resolved. It was not simply a quick fix for most of them, however; a careful design process was required to iterate on and implement any new ideas we came up with to get closer to a mobile-friendly design.

Design Process

In order to properly design the UI to fit the mobile resolutions, Large Animal Games decided that using a step-by-step procedure would be the most effective way to design it. Initially, basic sketches were made of different UI design ideas which did not include the finer points of the L.A.G. art conventions. These were purely for design and not intended to be visually complete. The second step was proof-of-concept images designed to loosely follow the conventions with finer attention to the visual aspect of the designs. Finally, compositions of various screens were made using the refined design ideas created in the first step, but composed of mostly finalized art assets which were essentially at the level required for the final product's art.

Our process for advancing through these stages involved a substantial amount of feedback from the art team for the original Picturiffic game. After we finished several basic sketches of each of the three main screens, we sent them to L.A.G. for general feedback involving the overall layout. These sketches represented varying ideas such as variable sizes for game elements, collapsible menus, and centralized buttons intended to make the screen more symmetrical and appealing. For those sketches that garnered positive feedback for their design ideas, we moved on to proof-of-concept images for a finer look at their visual layouts.

The first of these was a home screen composition, which separated the game's modes of play into large non-uniformly sized sections of the game screen for a more dynamic look than standardized buttons. It got positive feedback for the text on the Daily Puzzle section which had a sort of glare effect which made it a non-static color, and more interesting to look at. It was also much better about focusing on the game modes themselves, and having them fill up the majority of the screen. Unfortunately it didn't do a good job of making the buttons look like actual buttons, and a lot of the text was skewed oddly or simply plain looking. Due to the nonstandard shapes of the buttons, it also had more empty space than we wanted, and this was only made worse by the single color in the background. This resulted in low contrast, which made the important elements on the screen blend in more than they should have. We had to make significant changes when making a composition of the design for further feedback.

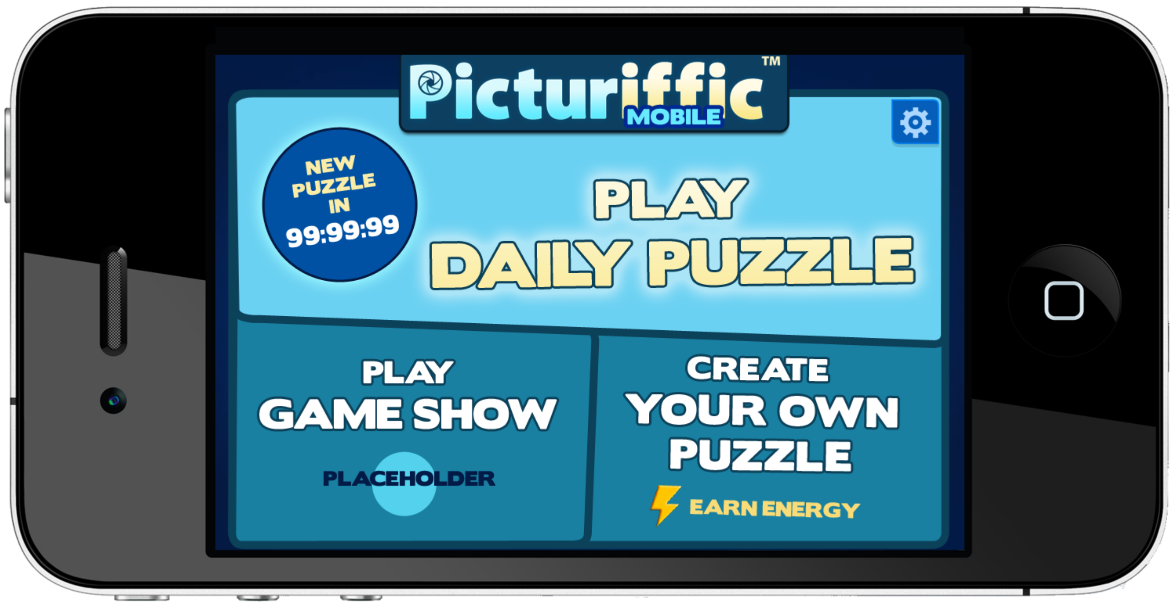


Figure 12: Adjusted Home Screen for dynamically sized buttons

We also included two designs focused on leaderboard designs, which were essentially the same except for a glow effect. They both made mostly effective use of contrast, but otherwise had many new ideas that simply didn't work as well as we had hoped. We had moved the life heart icons to the center line between the picture and the phrase, which made them seem too much like decoration and not enough like game elements. The experience, diamonds, and energy indicators were moved off the main play screen to save space, and the leaderboard was made to look more like a separate item, rather than simply laid on top of the rest of the game. Removal of the indicators caused some confusion, but a greater issue was the detachment of the leaderboards, which caused the screen to look too fragmented and lose cohesion. In addition, the 1st, 2nd, and 3rd place players were highlighted with red, green, and blue glow effects. These color choices seemed arbitrary, and caused confusion about which color represented each rank. We had also inadvertently changed the header text color slightly, leading to an awkward lack of contrast and generally displeasing look. Unfortunately the lifeline buttons were also too small, making them hard to read and likely would've been far too small to select on a mobile device. These were all problems that we later addressed in the composition of the design.

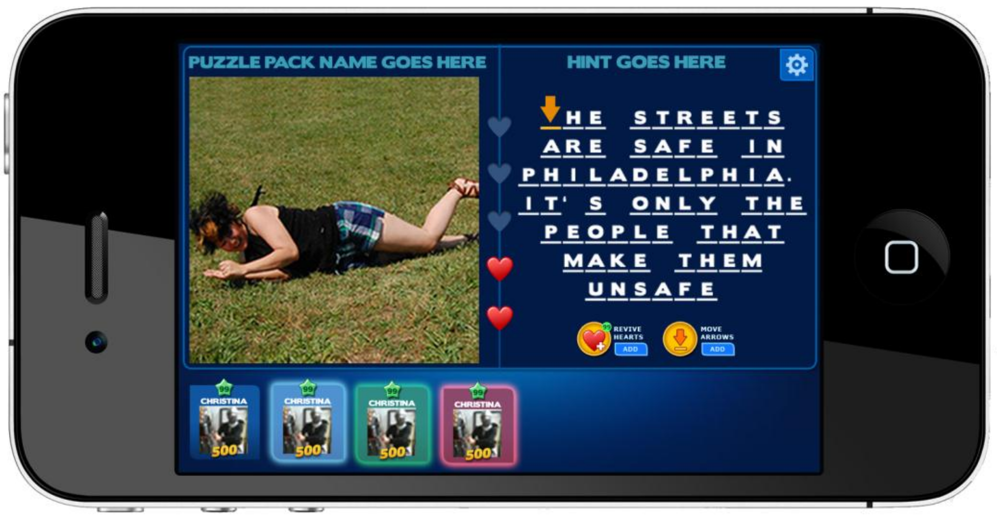

PUZZLE PACK NAME GOES HERE
HINT GOES HERE
THE STREETS ARE SAFE IN PHILADELPHIA. IT'S ONLY THE PEOPLE THAT MAKE THEM UNSAFE
REVIVE HEARTS
ADD
MOVE ARROWS
ADD
CHRISTINA
500
CHRISTINA
500
CHRISTINA
500
CHRISTINA
500

Figure 13: Leaderboard with glow effect

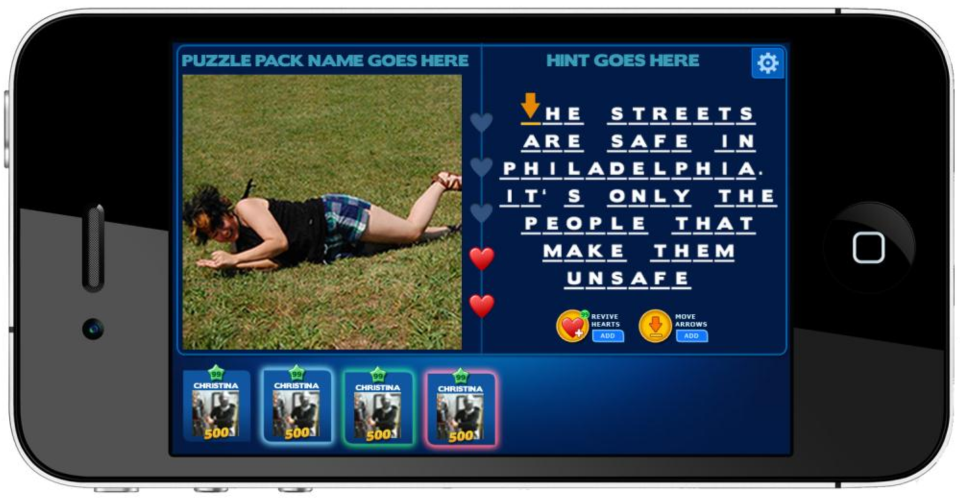

PUZZLE PACK NAME GOES HERE
HINT GOES HERE
THE STREETS ARE SAFE IN PHILADELPHIA. IT'S ONLY THE PEOPLE THAT MAKE THEM UNSAFE
REVIVE HEARTS ADD
MOVE ARROWS ADD
CHRISTINA
500
CHRISTINA
500
CHRISTINA
500
CHRISTINA
500

Figure 14: Leaderboard without glow effect

Our last round of solid feedback from L.A.G. was focused on two compositions; one for the daily puzzle screen and one for the home screen. The overall feedback was that we had been innovative in our approach, but we had strayed from the original Picturiffic art conventions and visual style. While innovation is usually a positive property, in this case it could potentially alienate existing Picturiffic players. Because we wanted to remain visually similar to the Facebook version as much as possible, these two compositions would need to be adjusted to visually match the original game more closely.

At this stage, the home screen design was mostly solidified, but there were still a few remaining problems that needed addressing. The main issues that stood out were the choice of words for the text and the precise sizing of the buttons. Some contrast was also lost from previous iterations, but overall it was a large improvement over the original compositions. We had worked to remove text from the screen by replacing relevant words with icons, such as Diamonds and Energy text below the Create Your Puzzle button.

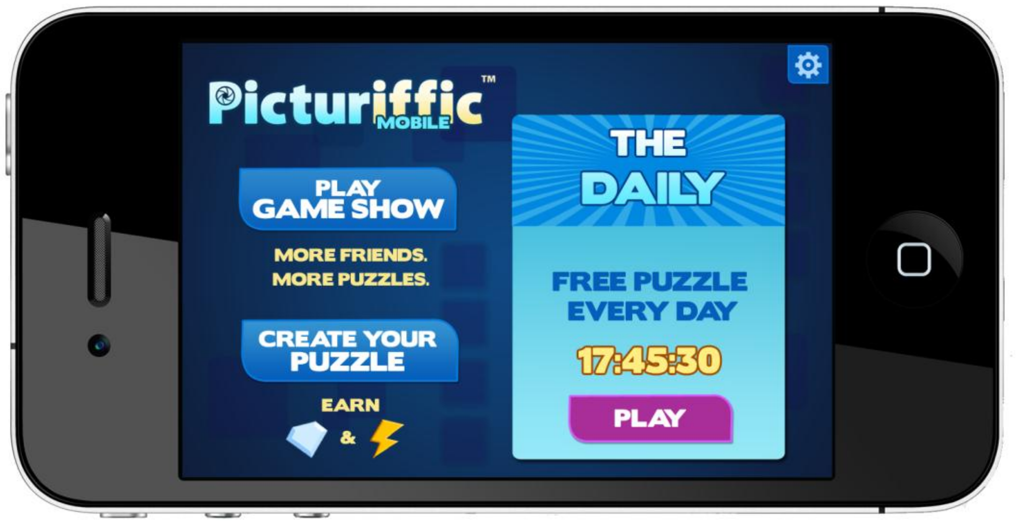


Figure 15: Final Home Screen with minimal text

The play screen was similarly complete, except we were still tweaking the location of the heart indicators. For this composition, they had been placed below the puzzle picture, which caused logistical issues when considering how the leaderboard animation might temporarily cover that space.

We had also implemented a new way of indicating leaderboard information, which was not immediately understandable for anyone who had not participated in its design. Because it was sufficiently different from the method used for the Facebook version, it would likely cause confusion and needed to be further refined.

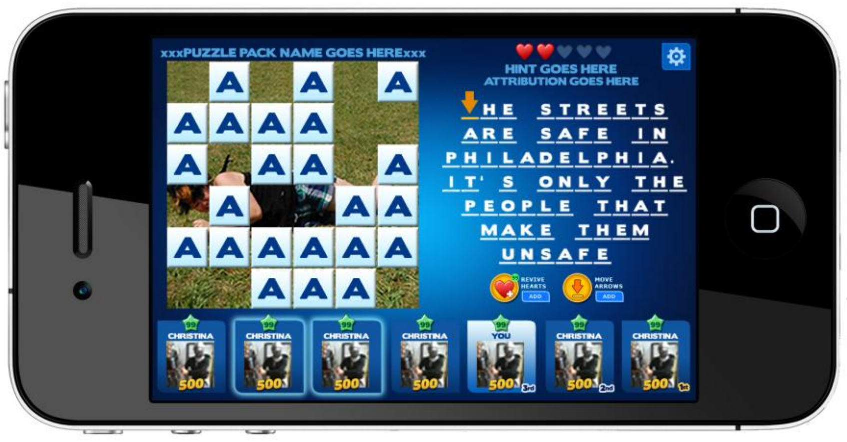

xxxPUZZLE PACK NAME GOES HERExxx
HINT GOES HERE
ATTRIBUTION GOES HERE
HE STREETS ARE SAFE IN PHILADELPHIA. IT'S ONLY THE PEOPLE THAT MAKE THEM UNSAFE
REVIVE HEARTS
ADD
MOVE ARROWS
ADD
CHRISTINA
YOU
500
3rd
2nd
1st

Figure 16: Final Daily Puzzle Layout

Implementing our Designs

Since we were designing the mobile version of Picturiffic in Flash, Large Animal Games gave us copies of the entire file library for Picturiffic. So, instead of creating the assets and animations from scratch we would implement our designs into their already existing FLA files and modify the existing code to work with the new build. This meant that we would have to adhere to their pipeline and our creative liberties would be limited as far as our process of design.

The art team's first priority was to make sure they understood how the main game FLA was structured so that they could properly edit and navigate through the file and implement their screens. Since the entire game was contained in one FLA, and every animation for a particular screen was on one timeline there was a lot of nesting of objects and animations that made it very intimidating to navigate through at first. We quickly became familiar with the layer and library structure of the file and we were able to begin the implementation of our newly designed screens.

The first and most obvious task was to resize the stage to the resolution of our target device which was the iPhone4. We specifically chose to design for the iPhone4 since it had the same aspect ratio as the earlier generations but a larger resolution of 960 by 640 pixels. (Wikipedia)

With a larger stage it now became necessary to realign all of the existing elements on the screens. Then we stripped away assets that we wouldn't need on the newly implemented screens. We were essentially breaking everything down so we could start building it back up.
The next logical step was to import any new assets to the appropriate folder in the asset library and properly name the symbol and instances of those assets according to Large Animal Games' standards. From there we implemented our designs based on our final approved compositions on the appropriate layers. We also implemented a similar style of animation to the new assets we brought into the game. Since we were focusing on delivering a working version of the Daily Puzzle game mode, we took time to cull animations that wouldn't be applicable to our build, like the bonus puzzle.

In order to review the work we did, Stephen would publish a desktop build that would allow us to get a general idea of the spatial arrangement and overall look of our new designs. Once we did several passes on our target screens we merged with Kevin to begin the process of bringing the build to the iPhone.

Optimization

We dedicated some time to researching how to optimize the performance of Flash projects, specifically on mobile platforms. One specific idea was to divide the FLA into scenes, theoretically splitting the load times between them and lowering the maximum memory usage of the game.

In *Flash CS5: The Missing Manual, Fourth Edition*, it is explained that while scenes are excellent for organization, they provide little to no benefit for the audience of the flash project. This is because Flash doesn't treat scenes the way we had anticipated; when it publishes the game into a SWF file, it stores everything in one big timeline. This means using scenes doesn't actually provide the performance boost that we had thought it might, although it does make the file more organized and easy to work with.

In fact, the only method of optimization explained by the book is that of multiple SWF files. The idea is that if you divide the project into multiple SWF's and have one main file load the others, it reduces the amount that the user has to have loaded into memory at any given time. As a side note, this is how we had mistakenly hoped scenes would work.

Through further research, we discovered that the methods of optimization are highly dependent on whether your performance bottleneck is due to CPU or memory deficiency. Specifically, how animated sprites are loaded or drawn on the game's canvas. We discovered several different methods to achieve better performance. (RossD20Studios)

The first option was pixel Blitting. Essentially, what previously was a bunch of MovieClip objects is shoved into one large bitmap object. The code tells it which part of the bitmap is needed for each object within it, and loads that. This drastically reduces the amount of RAM needed for loading all the objects in your game, but unfortunately it suffers large performance drops when used in high-resolution mobile devices like the iPad, probably due to high CPU load. (flashgameblog)

The second option was using a different MovieClip object (with bitmap data, not vector images) and moving those individually around the stage. While this method seems to work fine in situations like the iPod example above, it requires substantially more RAM than pixel Blitting. (RossD20Studios) This option is closest to what the Facebook version of Picturiffic currently does, except Picturiffic uses a fair number of vector data as well as bitmaps.

It's important to note that these two options can be combined, by using multiple MovieClip objects, while having each object contain a single bitmap that uses Blitting for the animation.

Flash has two methods of manipulating bitmap data; the draw method and the copyPixel method. In general, draw is slower than copyPixel. The draw method is used to convert vector data into bitmap data which reduces RAM usage but increases CPU load (and actually runs faster in CPU render mode), and the copyPixel method is used to simply load existing images that must be stored as BitmapData. (RossD20Studios)

There is also a performance distinction between CPU and GPU render modes for mobile devices within the publish settings. Choosing the right mode for the optimization methods you use can make a huge difference. There are also serious disadvantages to the Android GPU render mode; it doesn't support filter rendering for objects on the stage. This means all the glow and drop shadow effects used in Picturiffic aren't supported. In addition, any objects on or near the filtered object will also suffer decreased performance. Apparently it is still possible to achieve filter effects by using the draw method on the filtered objects off-stage, drawing it as a bitmap, and then placing it on the stage. (RossD20Studios)

Conclusion

Results

Originally the project's intentions were to port Picturiffic from Facebook to an HTML5/Javascript environment. Our research concluded that the pipeline and software available to do such a thing was too convoluted and underdeveloped for us to confidently execute our goals.

A large portion of the project was spent working within the constraints of a company atmosphere, and as a result, we gained a lot of insight into how to operate in a corporate setting. We may not have had the schedule that the employees at Large Animal Games did, but when it came to the research and the work we did, everything had to be within the scope of the project and the conventions laid before us by the company. This exposure gave us a clear understanding of how to work with constraints, which will inevitably assist us in future employments. We can say that we were tasked with taking an existing game, and without sacrificing much, porting it from one powerful processing system to another lightweight yet robust one.

In the end, we created a functional build for the iPhone4 in Adobe Flash and Adobe AIR. We spent a large portion of development time trying to get the build to compile and publish properly so we unfortunately did not have the time to optimize the build so that it would run fluidly on the device. Users are capable of loading the game and, as a pre-set account, playing through the Daily Puzzle mode successfully. The art assets have been re-arranged to better function on the smaller mobile display, and the code has been adjusted to take advantage of AIR's API.

To see our latest version as of December 15th, 2011 as an Android app, visit the following URL: http://users.wpi.edu/~kscannell/Picturiffic.apk. To see our latest version as an iOS app (to install, contact Large Animal Games directly for permission): http://users.wpi.edu/~kscannell/Picturiffic-test-interpreter.ipa.

Reflection & Recommendation to Large Animal Games

Looking back on the project there are things that we loved about it and there are also some things that we wish we could have done differently.

We originally looked at this sponsored project as a way to get real world, industry experience. It was also appealing to have a sponsor because it seemed that it would help direct the project and keep us realistically focused on our goals so that we would assuredly produce a functional product. We realized that we would be sacrificing some aspects of creativity but it seemed like a more valuable experience, and it was.

We got to try on the shoes of Large Animal Games and work as an outsourced developer for one of their major products. Just from visiting them we got to experience the dynamic of a game studio and all the branches within. We got a taste of what it's like to really be an artist, an engineer and a producer at Large Animal Games and how all of those roles work together to create a game.

While this was an extremely educational and valuable experience, there still were things that we would have liked to do differently. One of the major drawbacks of the project was that we weren't aware of what our project was about until we met with Large Animal Games. As a result we lost valuable time for research that may have helped us assess obstacles and execute our solutions sooner.

(The following is the document that we sent to Large Animal Games along with the project).

DISCLAIMER: Anything we put in this document is meant purely as a suggestion based on our experience -- by no means are we saying "If you don't do this, everything breaks!"

As it stands right now, Picturiffic works pretty well as a Flash application that runs in the browser. The way the project is structured allows for a fluid pipeline where the artists don't need the assistance of the engineers to make changes to the aesthetic feel of the game. The way Picturiffic uses multiple swfs to enable this process works well in the scope of the browser.

The multiple-swf approach poses several problems in mobile development though, specifically on iOS. First, iOS forbids just-in-time compilation, which AIR handles by compiling the actionscript into bytecode directly. A side effect of this process is that only one .swf file can contain code.

Secondly, the existing loading process doesn't work outside of a webpage. The solution we found was to use fake flash variables from channel_settings.xml to give the game the player information it needs to run. However, this creates the problem that the player's account is, essentially, hardcoded into the game. A method needs to be found to update that xml file on first run to connect to the player's Facebook account. However, it seems that Adobe's Facebook for Actionscript API isn't well-developed for mobile use yet.

Another issue that arises when making the move from the browser to a native app on iOS is memory. As Picturiffic runs right now, all the assets are loaded into the game at the beginning and are visible or invisible depending on whether or not they are currently in use. Because the computer is capable of running many difference processes at once, this goes unnoticed when running the game in the browser. In the mobile space, however, memory issues were one of the first things we noticed. The game simply loads too many assets to be able to maintain them all in memory while still running the game at an acceptable frame rate. The solution to this issue is to compile the game assets into a .swc file instead of a .swf file and instantiate each object as it's needed, then destroy it when it isn't.

On a less technical note, one of the major drawbacks of the project was that we weren't aware of what our project was about until we met with Large Animal Games. As a result we lost critical time for research that may have helped us assess obstacles and execute our solutions sooner.

Had we also been able to visit Large Animal Games sooner, say, at the beginning of summer, we could have used that extra time to do our research and been ready to start development in A-term. We spent a lot of valuable time looking into alternative methods for development instead of using that time for content creation. While the research itself is valuable too, it would have been beneficial to rule out HTML5 from the beginning so we could have had more time to develop and optimize our application.

Large Animal Games also wanted us to preserve the way the game looked, loaded and ran in the mobile version. This was a major obstacle because it meant that we had to adjust to their conventions and pipeline. If we could do it again, we would insist on rebuilding a prototype of the game from scratch for the mobile device with our own conventions. While this might have caused us to stray from Large Animal Games' vision for the game a bit, it would have allowed to develop a more efficient and fluid application that might better convey how the mechanics of Picturiffic could translate to mobile devices.

While this is wishful thinking, it could have been beneficial to relocate to New York for a term and work on nothing but this project. This way we would have had more immediate contact and feedback for our work. It seemed like the Picturiffic team lost interest in the project as we encountered set back after set back. While we understand that the Picturiffic team had their own priorities, we think if we closed the distance between our teams we could have optimized the dynamic between us.

Bibliography

Adobe Systems Inc. "About Phonegap." 2011. Phonegap. 05 09 11 <http://phonegap.com/about/>.

—. Adobe Edge Preview. n.d. August 2011 <http://labs.adobe.com/technologies/edge/>.

Anonymous. Highland Marketing. 2010. September 2011 <http://www.hiland.com/knowledge_base/helpful_hints/bitmap_vector_images_understanding_difference/>.

Apple Inc. "Safari HTML5 Audio and Video Guide." 05 07 2011. Apple Developer Guide. 02 09 2011 <http://developer.apple.com/library/safari/#documentation/AudioVideo/Conceptual/Using_HTML5_Audio_Video/Introduction/Introduction.html#//apple_ref/doc/uid/TP40009523-CH1-SW1>.

Campos, Jonathan. Optimization Techniques for Air for Android Apps. 8 September 2010. 25 November 2011 <http://www.unitedmindset.com/jonbcampos/2010/09/08/optimization-techniques-for-air-for-android-apps/>.>.

Flashgamesblog. Blitting - the art of fast Pixel drawing. 8 April 2010. 25 November 2011 <http://www.flashgameblog.at/blog-post/2010/04/08/blitting-the-art-of-fast-pixel-drawing/>.

Games, Large Animal. "Picturiffic Style Guide." Picturiffic Style Guide. Large Animal Games, September 2011.

Grover, Chris. Flash CS5: The Missing Manual. O'Reilly Media, Inc., 2010.

Holguin, Antonio. Swfhead. 21 March 2011. 10 November 2011 <http://swfhead.com/blog/?p=1102>.

Johnson, Jeff. Designing with the Mind in Mind: A Simple Guide to Understanding User Interface Design Rules. Morgan Kaufmann, 2010.

Krcha, Tom. Flash Realtime. 13 June 2011. 30 October 2011 <http://www.flashrealtime.com/compiling-big-air-swf-to-ios/>.

Przybylski, Ross. How to Optimize Flash Sprite Animations for Mobile. 19 January 2011. 25 November 2011 <http://www.kirupa.com/forum/showthread.php?359867-How-to-Optimize-Flash-Sprite-Animations-for-Mobile>.

RealNetworks. "Text Twist 2." Text Twist 2. RealNetworks, Inc., 13 October 2010.

Rovio. "Angry Birds." Angry Birds. Rovio Mobile, 25 August 2011.

Sencha Inc. CSS3 Animator. n.d. August 2011 <http://www.sencha.com/products/animator>.

Wegheleire, Koen. Adobe AIR Mobile: Application performance optimization on Android. 13 November 2010. 25 November 2011 <http://blog.newmovieclip.com/2010/11/13/adobe-air-mobile-application-performance-optimization-on-android/>.>.

Wikipedia. 2011. September 2011 <http://en.wikipedia.org/wiki/List_of_iOS_devices#iPhone>.

Zynga. "Hanging with Friends." Hanging with Friends. Zynga, Inc., 8 December 2011.

Zynga. "Words with Friends." Words with Friends. Zynga, Inc., 8 December 2011.