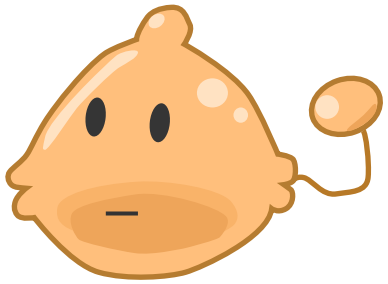
Kyong Art Website Recreation
&
AgiJagi Friends
iPhone Game Application



Hannah Nye

Senior Project

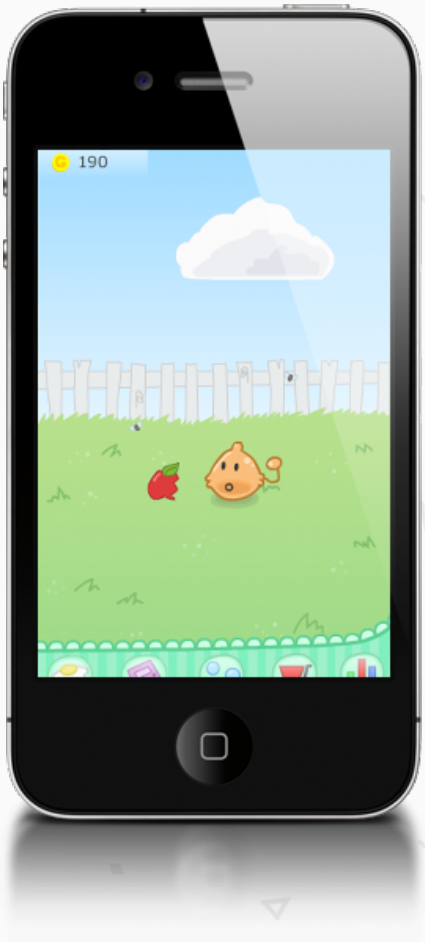
May 2, 2012

AgiJagi Friend Apple Store App

AgiJagi Friend was a application development project that was featured in the Apple store for a year. It is a virtual pet game that was reminiscent of old hand carried virtual pets. It allowed you to grow your own pet by feeding, playing, cleaning, and designing. Depending on how you treated your pet, it could grow up into various forms. All of the illustrations and coding were created by me, both created in Illustrator and Flash.

The app ran for $.99 for a year, and sold over 100 apps with minimal marketing.

View my official announcement blog post here.



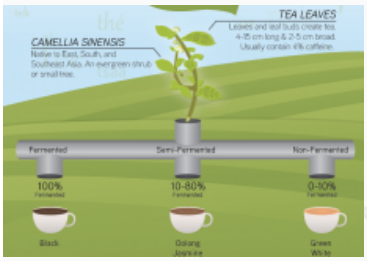
Posted Under

App Flash Illustrations

1 2 3 4

Portfolio

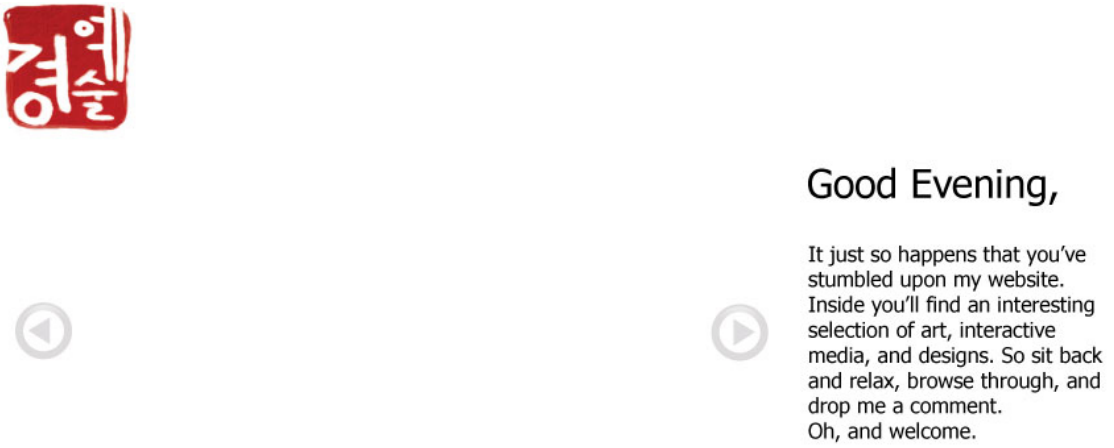
Sort Items: All // 3D // App // Flash // Illustrations // Print // Web Design



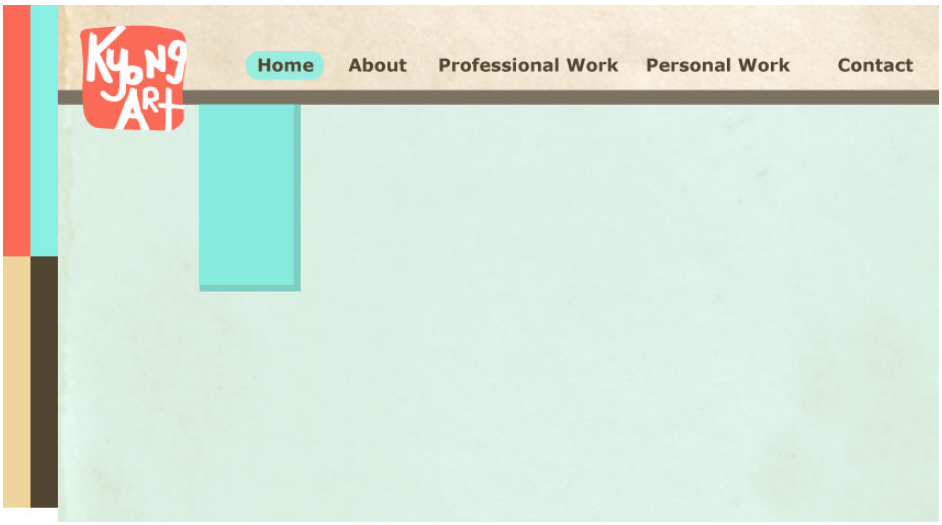
Kyong Art Website Recreation

The first part of my senior project was to recreate my entire personal website, titled Kyong Art. I decided this was a great idea because I really want to focus on web design and interactivity in my career field and I knew that my latest web design at the time was not representing my full capabilities.

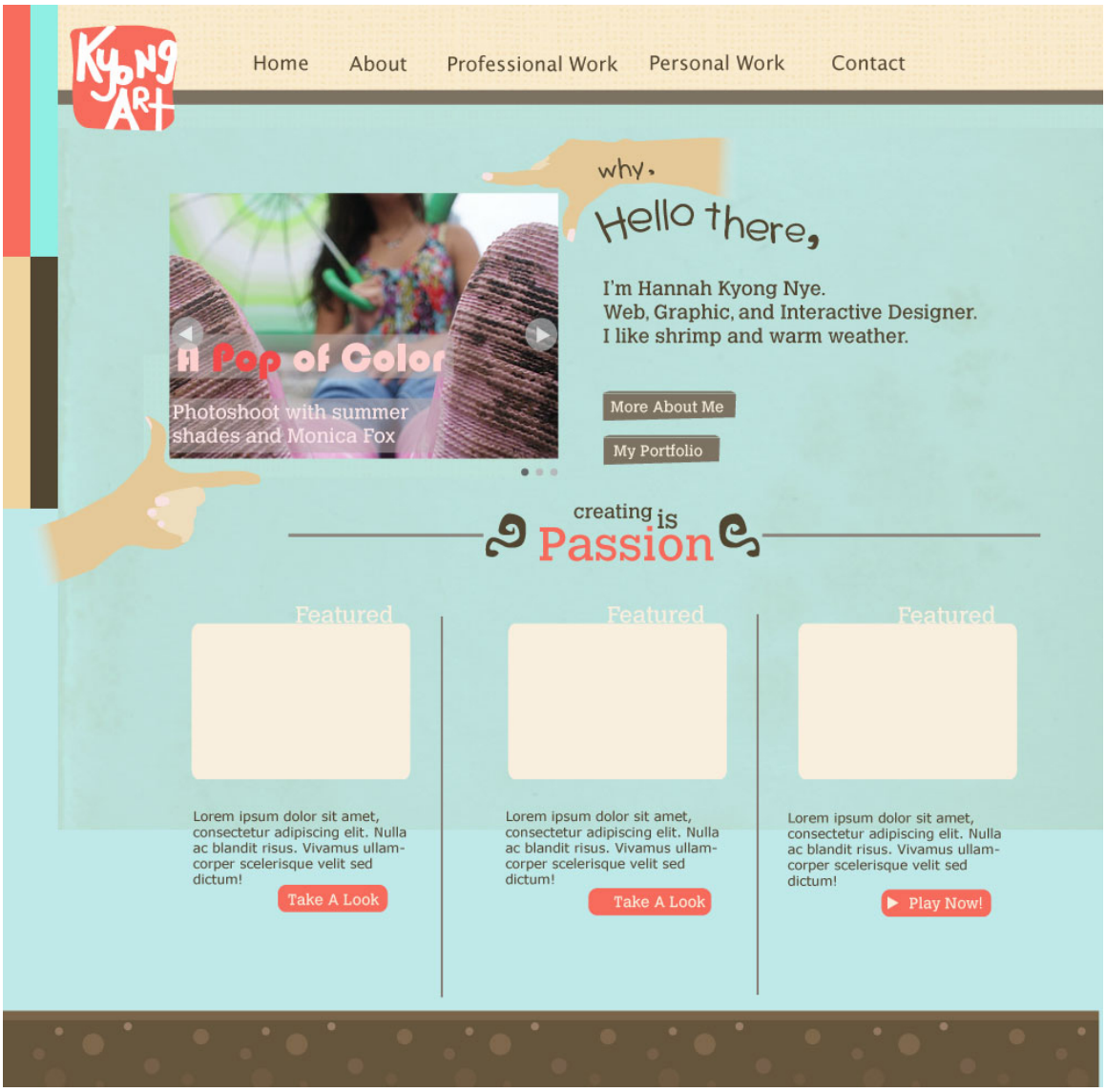
The first concept that I focused on was my logo. I wanted to create something that represented my cultural backgrounds, but was also catchy and artistic. After doing some research and initial sketches, I stumbled upon an image of a Korean rubber stamp drawing that was decorated. I fell in love with the idea and wanted to use the same style in my logo. My first creations were a dark red with Korean writing. I planned to have this the center of the web design, contrasting with a simple white background and text.



However, after sketching a few ideas, I realized that the contrasting white space was not what I really enjoyed as an artist. I started to sketch more ideas and kept the same sketchy feeling that I had in the logo. Eventually, I played with a color scheme that I really enjoyed – deep corals, mints, tans, and browns.



After a while I became attached to the idea of a pair of hands outlining and being the main focus of the website. I added more buttons, a background texture, and a slider to the design. I also decided to keep three main focuses at the mid-center of the page: Graphics, Websites, and Interactivity.



After consulting some friends and other artists, they felt that the hands were a bit too busy and that I could do without it. Honestly, I did not believe them at first but eventually understood what they meant and agreed that the hands were a little distracting. I took them out and worked on the whole design some more. I added some tabs at the bottom for social media : Facebook, Twitter, and Wordpress. Lastly, I also added a small design for an interactive bug that sings on a leaf at the very bottom. The front page also has two rectangle boxes that feature my most recent blog posts (connected to Wordpress), my recent Instagram photos, and Tweets.

Kyong Art

Home | About | Professional Work | Personal Work | Contact

why,
Hello there,

I'm Hannah Kyong Nye.
Web, Graphic, and Interactive Designer.
I like shrimp and warm weather.

More About Me

My Portfolio

creating is Passion

Graphics/Photography

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ac blandit risus. Vivamus ullam-corper scelerisque velit sed dictum!

Take A Look

Web Design and Coding

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ac blandit risus. Vivamus ullam-corper scelerisque velit sed dictum!

Take A Look

Interactive Design

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ac blandit risus. Vivamus ullam-corper scelerisque velit sed dictum!

Play Now!

Blog Entries

February 18, 2012
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ac blandit risus.
Read More...

February 27, 2012
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ac blandit risus.
Read More...

Lastest Instagrams

Twitter

Twitter
twitter

twitter Post-Bowl Twitter analysis
blog.twitter.com/2012/02/post-b...
pic.twitter.com/M0AtLQVd
19 hours ago · reply · retweet · favorite

twitter Join the conversation

The rest of the pages follow the same top navigation menu, with the same repeating background and the bottom footer. It also follows the same fonts that I used on the first page – Calvert MT Std (a favorite) and Verdana.

As for the coding, it integrates HTML, CSS, JQuery, Javascript, and PHP. I created all of the HTML and CSS in Dreamweaver, with distinct tables to hold information and used different classes to change certain elements on the page. I also used the CSS to place certain elements on the page; for example, the logo was placed within a div to hover in the position that it's at. I also made sure that at any given window width, that the main content will grow and shrink accordingly. The top navigation and logo combined stay in the same place, but do not exceed over 950 pixels.

The JQuery and Javascript were combined together to create both the slider and the Lavalamp (scrolling menu background). Both we're downloaded and integrated into my website, which was probably the most difficult part of the process. There was a problem with connecting two different j-query packs to the right effect. Eventually, after intense research online and playing around with the Javascript held in the main html document, I was able to combine both of the Javascript codes to make both of them work together.

Lastly, I created a contact page, "Get in Touch", with PHP. It's a simple page that allows the user to leave in their name, email, and message. It catches if the name, email, or message space was left blank and warns the user, asking them to fill out the field properly.

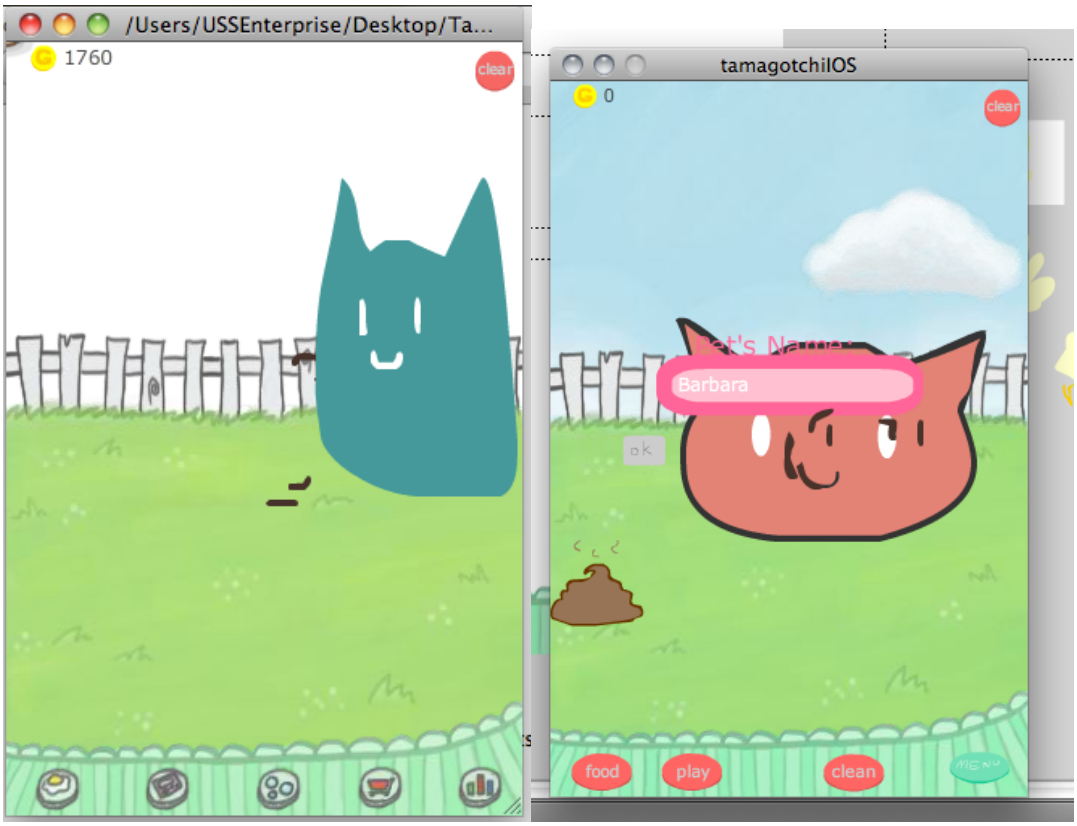
Overall, Kyong Art is an inviting website with a calm but artistic color scheme that reflects my personal creativity as an artist and as a web designer.

AgiJagi Friends - iPhone Application

AgiJagi means "Charming" in Korean. Therefore, the true name of the app is Charming Friends. I figured that AgiJagi would be a catchy, new, and creative word that

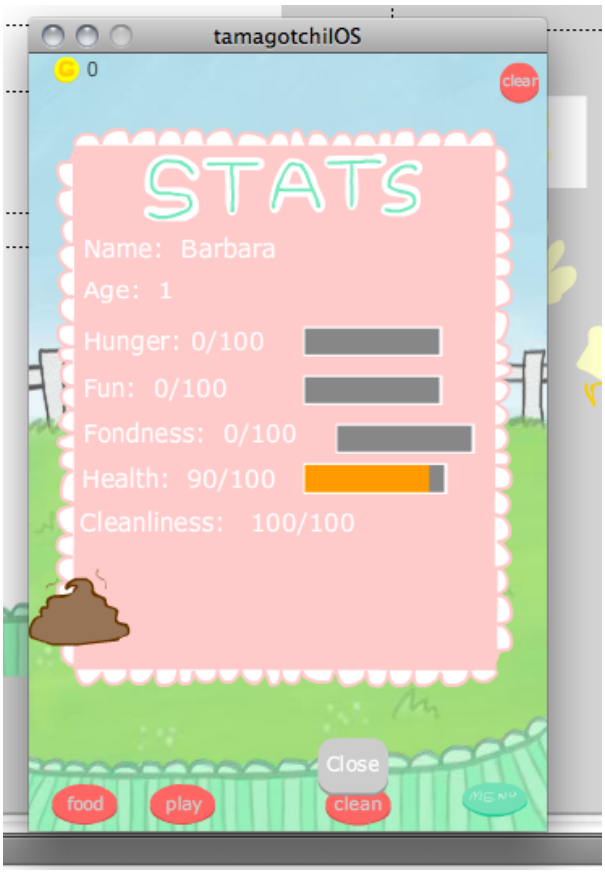
would interest users. The idea for this game first happened when I was looking for games in the app store that reflected the same abilities of an original Tamagotchi (electronic pet), but was up to date and had new capabilities. I eventually found something similar to what I wanted, but it was badly translated from Japanese and even though it was well made, still did not feel like the games I used to carry around as a child. While I had planned to create a Flash website for my final senior project, I figured that creating an iPhone application would be even more endearing, interesting, challenging, and an overall fun experience.

One of the first ideas that I really wanted to integrate with my new app was the ability to change the background depending on the time. I wanted to show the user that it was nighttime or daytime, depending on what time it was on their phone – which usually reflects real time. This was the first hurdle I went over, after refreshing my mind online with tutorials I was eventually able to get it to work, and it was the first of many side projects that eventually made my game what it is today.



Throughout the entire process of creating this game, I really began to understand how coding languages work and function; Flash makes it easier to understand, for me, because of the visual aspects. For example, I learned how to use the button and all of its different Event Listeners. About 80-90% of images that appear on the screen, are Movie Clips imported to the stage at load time. I relearned how to use variables and things like Math.random (for my game).

After creating time differences, I went into developing variables for the pet. I created different categories that change depending on how much the user takes care of the pet. This includes Hunger, Love, Cleanliness, Health, and Fun. For example, feeding the pet food will increase its hunger and could either decrease or increase its health, depending on the type of food and likewise do the same with love.

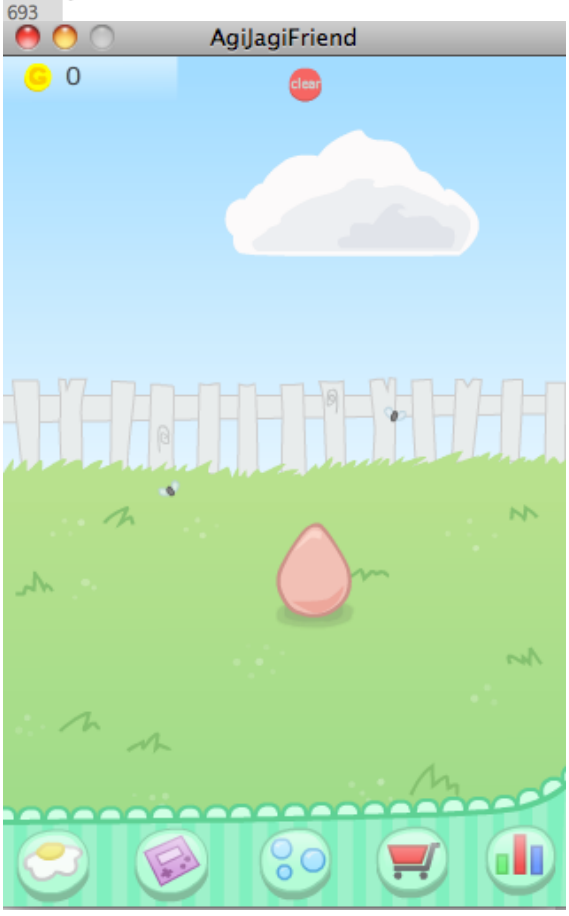


With all these variables, I wanted to make sure that when a user opened the program again that they were able to keep the same stats and pet on stage. This included extensive research online, after looking through many tutorials and commentaries I was finally able to find a way to save data to the system itself. Therefore, I have a regular variable and a saved variable. By making the saved variable equal the regular variable, the saved variable will always equal the regular variable (which is being updated whenever the user interacts). When the user opens the program again, the initial code goes through and makes sure that the regular variable equals the saved variable, thus making sure that the variables are always the same and creating a chain that is always linked.

```
24
25
26  // open a local shared object called "myStuff", if there is no such object - create a n
27  var savedstuff:SharedObject = SharedObject.getLocal("myStuff");
28  // manage buttons
29
30  //btnSave.addEventListener(MouseEvent.CLICK, SaveData);
31
32  addEventListener (Event.ENTER_FRAME, SaveData);
33
34      function SaveData(MouseEvent){
35          savedstuff.data.newage = age;
36          savedstuff.data.currentdate = my_date.date;
37          savedstuff.data.currentdatehours = my_date.hours;
38          savedstuff.data.secondanimal = mc;
39          savedstuff.data.petname = newpetname;
40          savedstuff.data.hunger = food;
41          savedstuff.data.gold = gold;
42          savedstuff.data.fun = fun;
43          savedstuff.data.fondness = fondness;
44          savedstuff.data.health = health;
45          savedstuff.data.poop = poopMC.x;
46          savedstuff.data.cleanliness = cleanliness;
47          savedstuff.data.bg = bg;
48          savedstuff.data.pettype = pettype;
49          savedstuff.data.f1 = F1;
50          savedstuff.flush();
51
52          //trace("saved time is",savedstuff.data.currentdate);
53      }
54
55
56  // Backgrounds!!! ----- new settings, NOT time bgs
```

Another part of code that I used was a Timer. This comes in play the first time the user presses adopt. Basically, it starts a timer of 30 seconds. At the end of 30 seconds, something happens (once, twice, or as many times as you want it to occur. In this case, it only happens once). At the end of the EggTimer, a movieclip is added to stage and the egg hatches, allowing the user to name their new pet. By clicking the OK button, the function savename saves the pets name and that name is kept throughout its entire lifespan and can be seen used throughout the game.

```
667 ▼ //var eggTimer:Timer = new Timer(2000,1);
668   //repeats forever.. takes 10 secs to get to food scene.., lo
669   eggTimer.addEventListener(TimerEvent.TIMER, timerEgg);
670           function timerEgg (e:TimerEvent):void{
671               pettype = 1;
672               animalegg.x = -200;
673               animalegg.y = -200;
674               babyAnimal.x = 150;
675               babyAnimal.y = 250;
676               Food.mouseEnabled = true;
677               playMenu.mouseEnabled = true;
678               cleanMenu.mouseEnabled = true;
679               shopMenu.mouseEnabled = true;
680               menu.mouseEnabled = true;
681               //event1.start();
682               eventsMC.x = -10;
683               eventsMC.y = 10;
684               eventsMC.gotoAndStop(155);
685               //newNameMC.y = 410;
686   //          newNameMC.x = 100;
687   //          newName.y = 410;
688   //          newName.x = 90;
689   //          savename.y = 410;
690   //          savename.x = 130;
691
692   }
693
```



The buttons at the bottom of the app represent the buttons that would usually be pressed in a regular, handheld virtual pet. There are five different categories: Food, Games, Clean, Shop, and Stats.

(Example of first buttons)

Food holds several different foods, including donuts, frozen yogurt, and a cheeseburger. Each food has different qualities and is unlocked at certain ages. If the pet is old enough, they pass the first “if” statement. If the user has enough Gold, they pass the second “else if” statement and the movieclip for the Food menu is closed, along with all the buttons, and the pet goes to and plays a part of it’s own movieclip that shows it eating.

(Example of first food graphics)

(Current food graphics)

The second category, Games, has only one game at the moment. I found the code for a simple Rock, Paper, Scissors game online and then adapted it to include my own strings and graphics. Basically, the computer chooses between three different numbers at random, all of which represent rock, paper, and scissors. The computer chooses after the user has chosen their own option. If the option is a win (rock > scissors, paper > rock, scissors > paper) then the events movieclip plays a clip saying "Win!" and reveals a Play Again button. This also happens with all other outcomes.

Clean allows the user to pick two options. The first one is a button with a broom on it, this allows the user to clean up poop. However, the broom button will only work when there is a poop movieclip on stage. If there is not, then the warning text will notify the user. If the user sees the poop movieclip on stage, but closes the program, the X and Y positions of the poop movieclip is saved and store into data until the program is opened again and the need to clean is still there.

The second option of clean is to give the pet a bath. This is a favorite because it allows the user to rub over the pet with their finger and rub back and forth like they are actually petting it. Basically, it uses a mouse over, mouse out, event listener to create this effect. Every time the user mouses out, Health and Cleanliness gain some points.

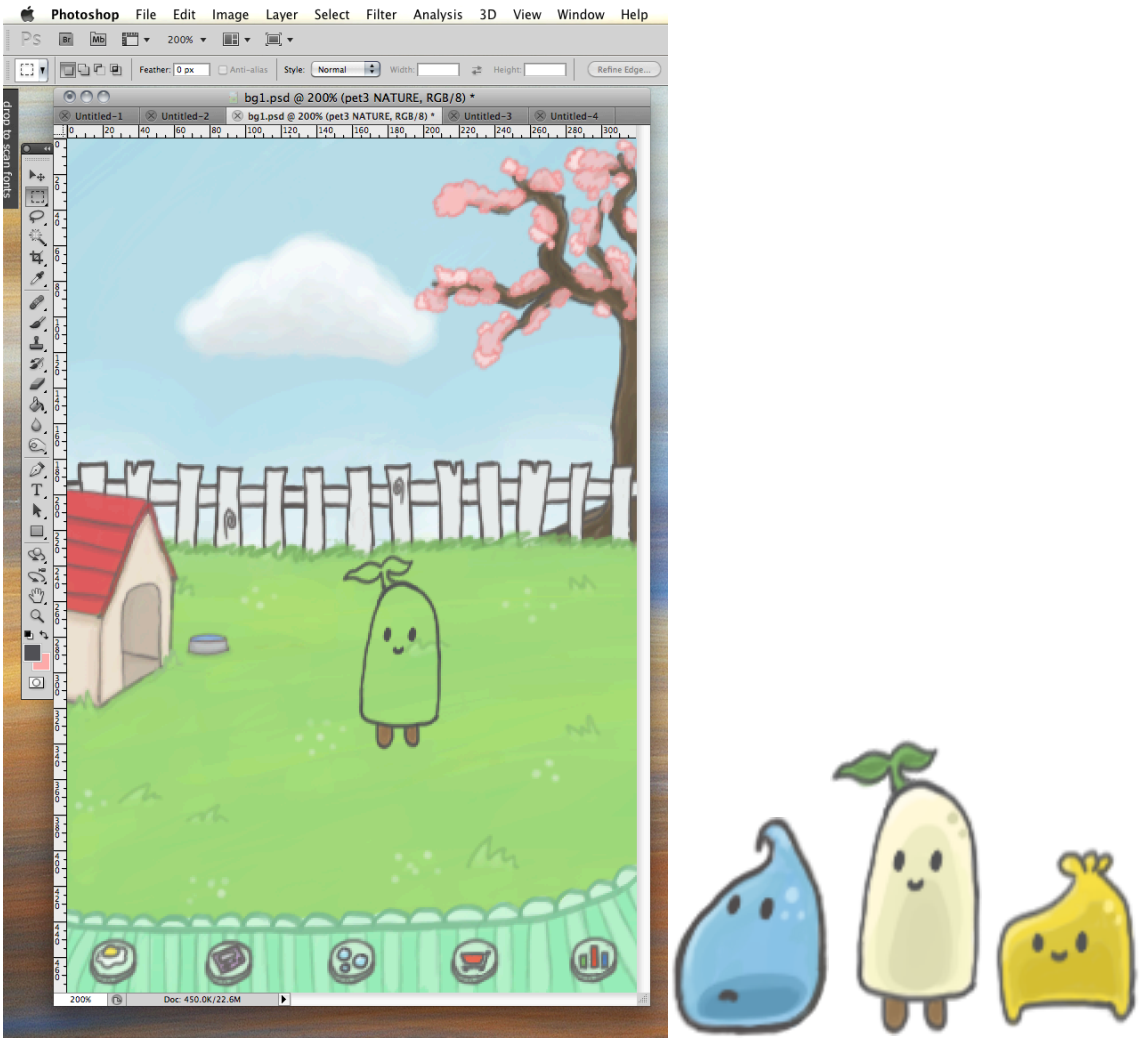
Shopping is another vital part of this game. It allows the user to change scenery and customize their own stage. The first part of the Shop is the backgrounds. The

backgrounds change the overall scenery of the location that the pet is in. More backgrounds become available depending on age and they get pricey as the age goes up also, making them hard to get. The second part of shopping is the furniture. This is a recent addition to the game and one that I think adds customization that all users will enjoy. Basically, different furniture is unlocked at different ages and after purchasing an item, users are then able to “Redecorate” and drag around the piece of furniture to the location they want it to be in. This allows the user to place it anywhere on the stage, giving them full customization of their own scene.

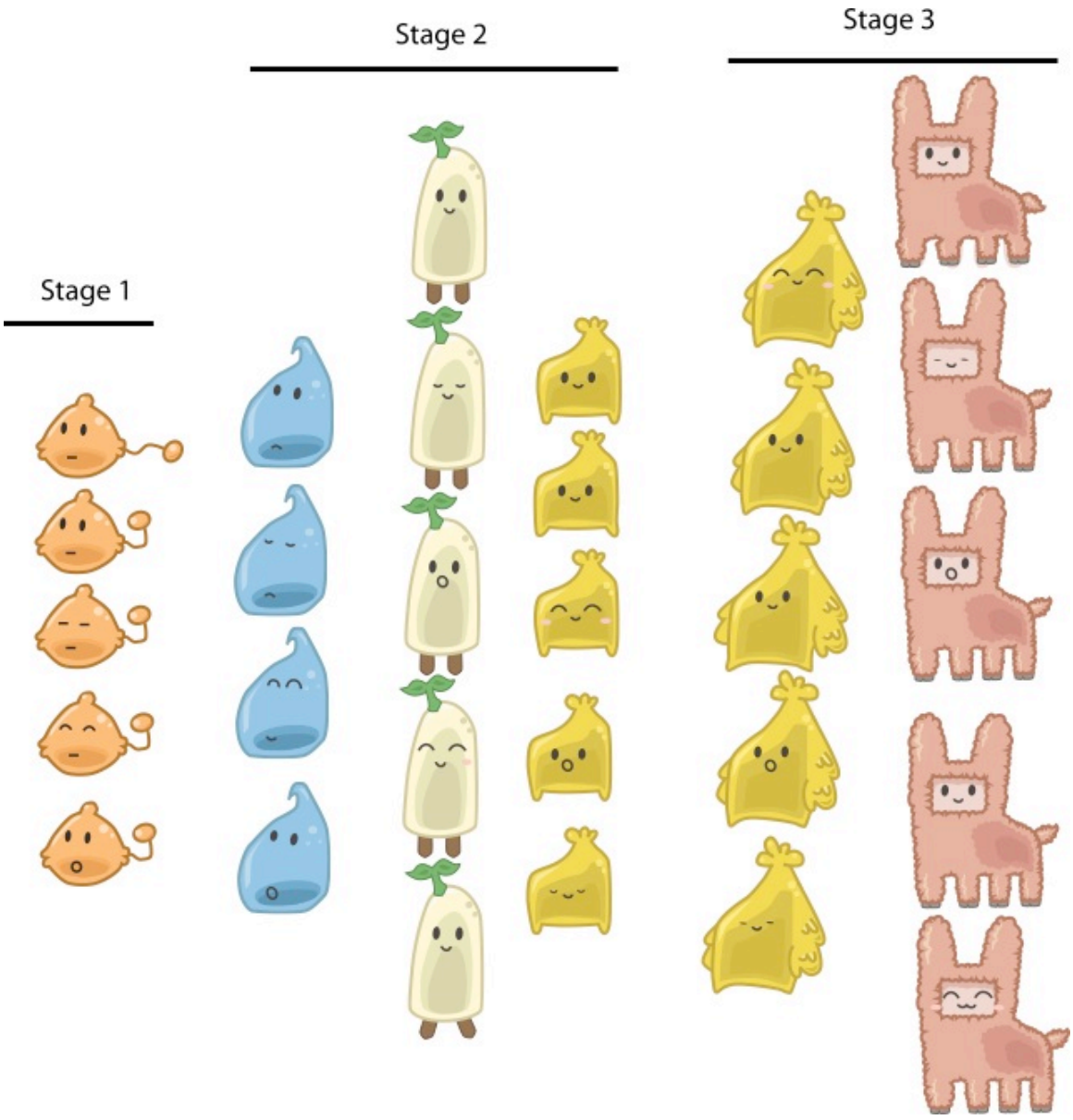
The last button is the Stats, this basically holds a meter for each of the variables and the name of the pet. The more that the variables increase, the more that the meter does also. This happens by creating the effect of “if (variable >= 50) { meter.gotoAndStop(5); }”. Basically, it stops at a different part of the movieclip.

```
515     if (fun <= 9) {
516         funscaleMC.gotoAndStop(1);
517     } else if (fun == 100) {
518         funscaleMC.gotoAndStop(11);
519     } else if (fun >= 90) {
520         funscaleMC.gotoAndStop(10);
521     } else if (fun >= 80) {
522         funscaleMC.gotoAndStop(9);
523     } else if (fun >= 70) {
524         funscaleMC.gotoAndStop(8);
525     } else if (fun >= 60) {
526         funscaleMC.gotoAndStop(7);
527     } else if (fun >= 50) {
528         funscaleMC.gotoAndStop(6);
529     } else if (fun >= 40) {
530         funscaleMC.gotoAndStop(5);
531     } else if (fun >= 30) {
532         funscaleMC.gotoAndStop(4);
533     } else if (fun >= 20) {
534         funscaleMC.gotoAndStop(3);
535     } else if (fun >= 10) {
536         funscaleMC.gotoAndStop(2);
537     }
538
539
540     if (fondness <= 9) {
541         fondnessscaleMC.gotoAndStop(1);
542     } else if (fondness == 100) {
543         fondnessscaleMC.gotoAndStop(11);
544     } else if (fondness >= 90) {
545         fondnessscaleMC.gotoAndStop(10);
546     } else if (fondness >= 80) {
547         fondnessscaleMC.gotoAndStop(9);
548     } else if (fondness >= 70) {
549         fondnessscaleMC.gotoAndStop(8);
550     } else if (fondness >= 60) {
```

Graphics wise, I initially created everything in Photoshop – I wanted to a create a paint like effect that would be different from most app designs. However, I quickly realized that vectorized images would work a lot better with the high retina screens that the iPhone offers today. After creating all the graphics all over again, I added more to what I had.

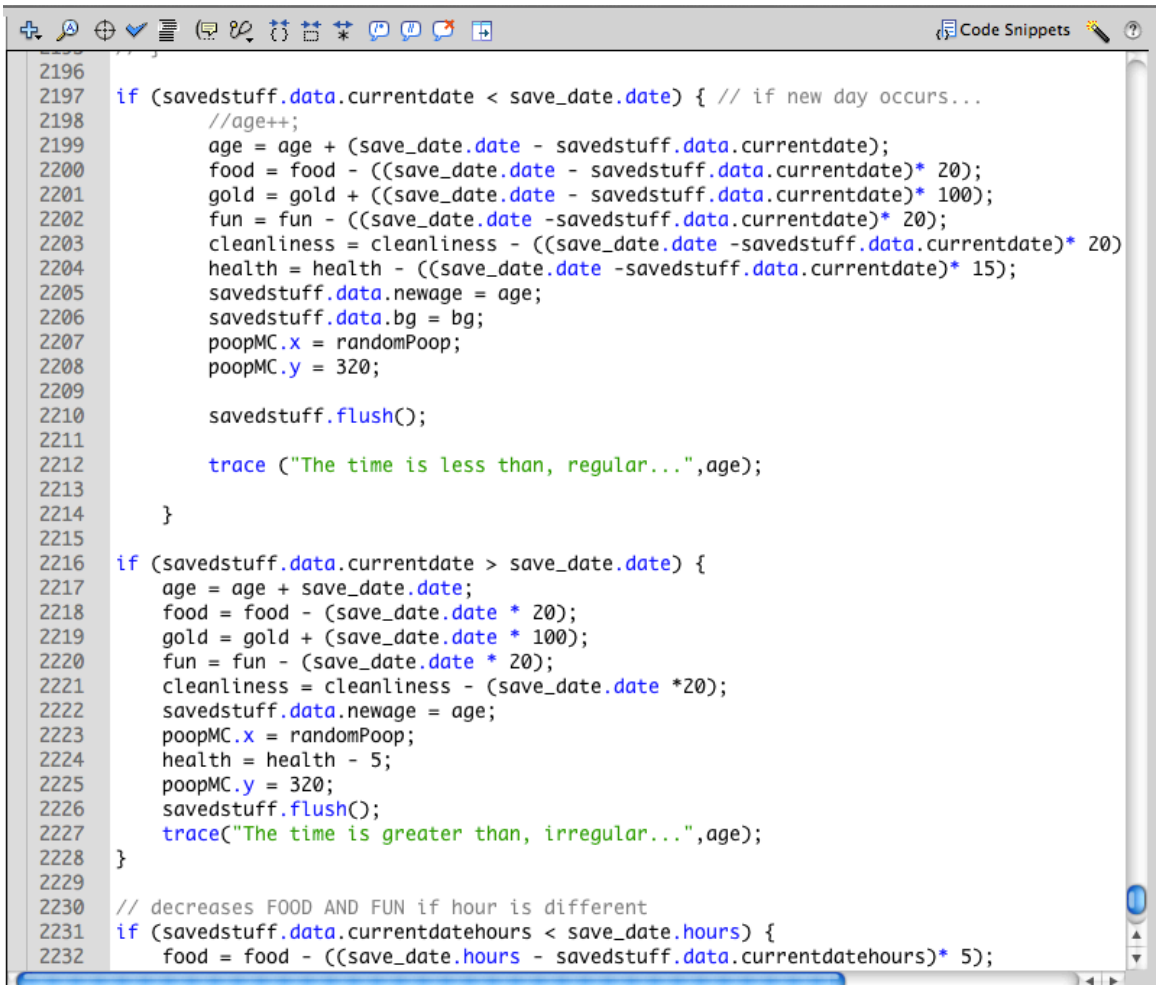


The main focus for my game is the pets that you raise and evolve. I made about five different poses for each pet evolution stage. Overall, I currently have six pets completed and fully in use in the game. They all evolve at ages 5 and 12, depending on how well you take care of them (what their stats are). If the stats are low, the pet is going to be sad looking. If the stats are high, the pet is going to be happier looking. Overall though, the evolutions do have some random elements and I eventually plan to make over 100 different pets that the user could try and evolve into.



As fun as the game was to make though, there were still some subjects that I was not able to fix. For example, after purchasing the Apple Developer ($105 – including tax), after reading through a tutorial for uploading flash made iOs games, I found that some of the graphics had not converted over like I planned. Though most of the graphics look fine, some of them look a little pixelated. I found out that it might be because of the iPhone 4's HD retina display. I searched online to find out how to fix my problem, but the closest answer I got was "change the vectors to 2x resolution". I was very confused, and even after creating bigger graphics to import into Flash, a couple images

were still a little blurry. I plan to work on this more in the future and to hopefully find a remedy. Also, the beginning/intro screen for the game lags a bit; the cloud in the background chops its way across the screen. The last shortcoming is probably one that I will always continue to search for a solution to; the time settings. Basically, the age variable changes with each new day. However, since variables do not recognize monthly and yearly dates, I had to code it in a way in which it changes every time the date is increases. Then, depending on how much it increased, it would multiply by a certain number to subtract from the statistics. The big problem I had was when the dates change from the end of the month (example: 31) to the first of the month. Trying to subtract the number of days that have increased was not getting me the right amount of numbers, so I had to use the number of days that have increased (1,2,4,7) as the number that would be multiplied and then subtracted from the stats. It was a confusing process, that can have some complications. For example, if a user did not visit their pet everyday and the last time they visited was on the 29th (out of a 31 day month) and they opened the app again on the 2nd of the next month, the age would only increase by 2. This is because the program detects that the saved date is greater than the current date. It then only takes the current date (2) and does calculations with the statistics. It's a process that I hope to eventually fix by finding a calculation or script that can detect a new date and how many days, without going through that hard process. Either that, or I could eventually add a movieclip that shows the pet dying after two days of neglecting it. I should also point out, that if a user changes the date on their own device, they will be able to manipulate the date and age on the game.




```
2196
2197  if (savedstuff.data.currentdate < save_date.date) { // if new day occurs...
2198          //age++;
2199          age = age + (save_date.date - savedstuff.data.currentdate);
2200          food = food - ((save_date.date - savedstuff.data.currentdate)* 20);
2201          gold = gold + ((save_date.date - savedstuff.data.currentdate)* 100);
2202          fun = fun - ((save_date.date -savedstuff.data.currentdate)* 20);
2203          cleanliness = cleanliness - ((save_date.date -savedstuff.data.currentdate)* 20)
2204          health = health - ((save_date.date -savedstuff.data.currentdate)* 15);
2205          savedstuff.data.newage = age;
2206          savedstuff.data.bg = bg;
2207          poopMC.x = randomPoop;
2208          poopMC.y = 320;
2209
2210          savedstuff.flush();
2211
2212          trace ("The time is less than, regular...",age);
2213
2214      }
2215
2216  if (savedstuff.data.currentdate > save_date.date) {
2217      age = age + save_date.date;
2218      food = food - (save_date.date * 20);
2219      gold = gold + (save_date.date * 100);
2220      fun = fun - (save_date.date * 20);
2221      cleanliness = cleanliness - (save_date.date *20);
2222      savedstuff.data.newage = age;
2223      poopMC.x = randomPoop;
2224      health = health - 5;
2225      poopMC.y = 320;
2226      savedstuff.flush();
2227      trace("The time is greater than, irregular...",age);
2228  }
2229
2230  // decreases FOOD AND FUN if hour is different
2231  if (savedstuff.data.currentdatehours < save_date.hours) {
2232      food = food - ((save_date.hours - savedstuff.data.currentdatehours)* 5);
```

In the future, I eventually plan to connect the game to the Internet. With PHP script, I will be able to store data online so users will be able to connect with other players, see high scores, and possible “visit” other players homes to view each others different layouts. I also have several new ideas for mini games and would like to add in some such as, growing vegetables for money, being able to move your character through a 2D environment, and a butterfly catching game. On top of this, I also do plan on adding more foods, backgrounds, and furniture frequently. Last, I would also like to add different stats in the future. I would probably add something like “Smart” or “Knowledge”, so the user would be responsible for making the pet read or taking them to school.

In conclusion, I am very happy with the outcome of my application – it was exciting to see it working on my phone for the very first time. I plan to release the game in the Apple app Store within the next week or two for free. Depending on how well it does, I may start charging .99 cents after multiple updates.