Interactive Level Design for iOS Assignment Delivery: A Case Study

Anson BRONW, Alireza TAVAKKOLI, Donald LOFFREDO, and Hashimul EHSAN

University of Houston-Victoria

Victoria, TX 77901, USA

ABSTRACT

This paper presents an application of an iOS-based online gaming assignment in a real classroom. The core concept of the project is a gameplay environment involving two players that have full control over creation and modification of levels. This level design mechanism was implemented in an iOS-based game in the area of genetics and based on an existing written assignment. The game includes support for both instructors, who have the ability to create and post assignments and students, who can take the assignments. Two trials of the iOS application consisted of in-class testing of twenty- one students. Students first took the original paper assignment, followed by the iOS version. Start times, end times, and grades were recorded for both versions. A comprehensive study of the grades and times for the iOS version of the assignment versus the paper version was conducted and is presented in this paper. Our Study showed that the iOS version was completed much faster in nearly every case while a strong delivery mechanism is needed to ensure student grades and completion of the assignment will not be affected. These results are not unexpected due to some major difference between the two formats. Future updates and additions will address any currently existing issues.

Keywords: Interactive Level Design, iOS, E-Learning, Artificial Intelligence, Mobile Cellular Devices, Statistical Significance Tests.

1. INTRODUCTION

The video gaming industry has grown rapidly and shows no signs of slowing. Gaming and simulation not only take advantage of the newest computing technology, but they also are the driving force behind many new innovations. The goal of this project is to harness some of these technologies and techniques in order to create a unique learning experience. The goal of this project is to test a mobile game implementation of an existing paper-based assignment and compare the results in order to find the best delivery method.

Motivations

Educational games have been used for quite some time, but they are not considered mainstream in the gaming community. Entertainment gaming in general, however, is pervasive. New technologies in digital software and hardware allow the creation of games with a level of interactivity and immersion that was simply not possible in the past. Touch screens and motion sensing technology are now commonplace in the gaming industry and keep improving with every new iteration of the technology. Many students already carry these devices daily and using them for assignment delivery could allow work to be done from almost anywhere without paper or pencil. A paperless approach could save instructors from wasting time and resources on repetitive grading.

Some educators are embracing this new technology and the idea that students of this generation are more technology-inclined. Earhart Middle School in California recently announced promising results of a year-long study in which an Algebra I curriculum was taught entirely on iPad devices [1]. An app called HMH Fuse: Algebra I was developed by Houghton Mifflin Harcourt (HMH) exclusively for the Apple iPad. HMH is currently the world's largest provider of materials for pre K-12 learning. The results of this pilot program showed a 20 percent increase in standardized test scores when compared to traditional text book users.

HMH Fuse: Algebra I [1] was designed with the capabilities of the iPad in-mind from the beginning. The software fully utilizes the touch-screen functionality, portability, and multimedia capabilities of the device in order to achieve a rich, interactive experience. Using the devices' internet capabilities allowed instructors to see student progress in real-time and tailor instructions based on his or her strengths and weaknesses. While this application is not necessarily a game, it still utilizes the iPad device to its fullest extent and applies many of the same methods that are being explored in this paper.

Another example was found that details the experiences of a high school that used an educational video game to teach history. In this case, a game titled

Making History was used to teach students about World War II [2]. The study found that using a game shifted the learning environment from teacher-centered to student-centered. This allowed students to be more active and engaged in their learning. Students were noted to be much more enthusiastic about what they were learning and took to the gaming environment readily. The article also points out some examples of several non-educational games being used for teaching [2]. For example, Civilization III, a cooperative turn-based strategy game, was used in an after school program to teach students various historically accurate facts and rule-sets.

Literature Review

Most standard video games contain level content that has been created previously during development. These levels generally cannot be modified by the player community. Some designs allow content to be created by a player and shared with the rest of the world. Few games, however, allow multiple players to participate on a limited basis in the level creation process in real-time. Littlebigplanet and Littlebigplanet 2 by Sony Computer Entertainment Europe allow players to create and share an unlimited number of levels with anyone in the world [3]. The "Create" mode allows players to collaboratively build a level over a network. All of the tools the game creators used to build the main game levels are available to users for their own creations. At the time of this writing, the Littlebigplanet online level database contains nearly 7 million user-created levels [3]. Players are able to use a variety of helpful filters when searching for levels in order to find what they are interested in.

Minecraft by Mojang AB takes the collaborative building approach and applies it to an open, procedurally-generated sandbox world [4]. Multiple players can join together and build the entire world with almost no limits [5]. The limitless nature of Minecraft has caught the eye of some instructors who are now using it to teach a multitude of different lessons in the classroom.

Joel Levin, a computer teacher at Columbia Grammar and Prepatory School has been using Minecraft to teach a full unit to second graders [6]. The open-ended nature of the game allows Levin to teach a large variety of subjects. Levin can create his own lessons by modifying the game world and adding content where needed [6]. In light of his success in using Minecraft in the classroom, Joel Levin has started a program called MinecraftEdu with the intention of getting more teachers to use the game in their classrooms [7]. MinecraftEdu is partnered with Mojang AB and provides bulk copies of Minecraft at a discount as well as custom game versions for teaching, lesson plans, inservices and training and other helpful tools [7].

Sean C. Duncan points out that Minecraft was not developed as a teaching game, yet more teachers are turning to it for this purpose [8]. Duncan suggests the reasoning behind this is due to the fact that while Minecraft is a game itself, it allows other games to be built within it in a sense.

An online community called Massively Minecraft [9] is working to create a thriving learning-oriented guild for Minecraft. The Massively Minecraft guild is targeted towards children between the ages of 4 and 16 and facilitates learning by assigning objectives for its members to complete. The Massively Minecraft program currently has 300 members worldwide and continues to expand by adding servers and maps [9].

The Minecraft game itself provides the tools to do almost anything by default. For example, one player was able to create a 16-bit ALU (arithmetic logic unit) computer entirely in-game using various blocks [10]. The creator of this in-game computer has extended it further by giving it the ability to compile and run code [10].

Many classrooms implement computers with projectors for displaying presentations and other demonstrations. SMART Board technology combines a white board with touch-screen capabilities to enhance learning and teaching experiences [11]. Instructors can use their fingers as mouse pointers to manipulate a computer via the projected screen output. The SMART Board also utilizes special pens that allow instructors to draw digitally on the screen and save their notes [11].

Fatih Gursul and Gulsah Tozmaz conducted studies involving 20 classrooms with SMART Boards to determine their benefits and drawbacks [12]. Results of Gursul and Tozmaz studies showed that when used correctly, SMART Boards are excellent at increasing student attentiveness and interaction while giving teachers the tools to create more enjoyable lessons. Detriments to SMART Board use include both technical issues and the difficulty in transferring saved data between machines or other teachers. The boards were found to be most effective in specific kinds of lessons involving animations displaying activities that cannot be conducted in classrooms [12].

Tablet computers and smart-phone or PDA devices are gaining popularity in the classroom. Studies have been conducted to compare tablet devices versus traditional laptop computers in a collaborative classroom environment. A study involving graduate engineering students found that tablets were highly preferred over laptops due to their ability to function as digital paper [13] Students were better able to express ideas and collaborate while writing on the device in a natural way with a stylus [13].

The Apple iPad tablet device was specifically used in a study conducted in a United States history course. An eleventh grade class was tested using an application called Explore 9/11, which is designed to teach students about the World Trade Center attack in 2001 [14].

The application uses audio and visual elements to provide more immersion in the learning experience. A total of 75 students were tested with 49 using the iPad and 25 being taught with traditional methods. The results of the study concluded that students using the iPad application generally scored slightly higher than those who did not. While grades were not much improved with the iPad, student surveys praised the format due to the collaboration and interaction it brought to the learning experience. Instructors also found that the iPad made bringing in outside sources much easier than with traditional methods [14].

Mobile serious games have also been implemented on laptops. Classmate PC [15] devices were used in a Chilean 8th grade classroom to develop problem solving and collaboration skills [16]. The games used in the study had somewhat static level design, but one used real-time strategy elements which allowed the gameplay to change and evolve. Students were split into groups of 4 and challenged to grow and maintain various animal species in a virtual environment collaboratively. Findings showed that students using the mobile game were better at plan execution than those in the non-equivalent control group [16]. This, however, is the only significant difference. Researchers suggest that finding the proper conditions for using mobile serious games is key in their effectiveness [16].

Massively Multiplayer Online (MMO) games are becoming an attractive educational platform. These types of games support online play on a large scale with a social emphasis. An MMO game engine was used to create a prototype online game to be tested by 25 students in the University of La Laguna's Computer Engineering Department [17]. Students were first given instruction on how to play followed by tasks to complete cooperatively with a focus on communication. Early activities proved difficult due to the complexity of the interface, but were completed once students had adequate time to learn. Results of a student survey showed that 79 percent believed that the activity increased the appeal of the course [17]. Researchers also tested secondary school students using non-educational games such as World of Warcraft [18] in order to teach server administration and leadership skills. Research concluded that video games in general, not just those built for education, can be useful in teaching many skills that transfer to the outside world as well [17].

2. DESGIN

Our design incorporate aspects of interactive level design, but the style of game is completely different. The new game implements a formerly written assignment in an online, interactive form. Elements of web-based teaching frameworks are also incorporated to allow interaction between students and instructors. The aim is to allow instructors to create an unlimited number of assignments on-the-fly from anywhere and push them to students. As long as the student has an iOS device with an active internet connection, he or she can receive and take assignments from anywhere. Instructors can immediately see the results and provide feedback. The overall feel is similar to that of the Blackboard learning system [19] combined with a video game.

3. IMPLEMENTATION

This section discusses the frameworks and software used in this project followed by detailed explanations of the user interface and algorithms used.

Platform

The implementation of the design is carried out on the iOS platform, which currently includes the iPhone, iPad, and iPod Touch mobile devices. This platform was chosen because of the incredibly high amount of iOS devices currently in use as well as several other features that make for great interactive games.

The way a user interacts with these devices is generally very intuitive and functions the same across all of the supported hardware. A touch interface allows various finger gestures, such as pans and rotations that come naturally to most users. Accelerometers allow users to tilt and rotate the device to interact with objects on-screen. Networking and social capabilities enable users to easily interact with one another in both turn-based and real-time settings. The game implements natural touch-based gestures as well as networking to provide a cooperative learning experience on handheld devices.

Another aspect that made the iOS platform attractive was the lack of fragmentation. Fragmentation occurs when many different device and operating system versions are released, which makes programming extremely difficult. On a fragmented platform, processer speeds, RAM, and screen resolutions can all drastically alter the performance of an application or inhibit its ability to run altogether. Most iOS applications run on all previous device versions without modifications. There are exceptions, but this specific project has no such problems.

Technology

This section discusses the technologies used in implementation and deployment of the proposed game-based assignment including the interface design, networking infrastructure, and arts and asset creation.

Interface: The main interface is created with the UIKit framework, which is provided with the iOS SDK. UIKit includes classes designed for touch-based interfaces in iOS applications. Common controls such as textboxes and labels are provided, as well as event handling, drawing model, windows and views. UIKit was utilized in this project to speed of development time and provide a cohesive, familiar iOS experience for users.

The phenotype creation tool partially utilizes Cocos2d-iPhone, a free 2D game engine created by Ricardo Quesada [20]. Cocos2d can be used alone to

create 2D games without the use of InterfaceBuilder or UIKit objects. The framework handles much of the more difficult openGL programming and sections portions of the game into scenes and layers.

Networking and Back-End: The networking portion of the game makes use of the Parse framework [21]. Parse framework provides an easy-to-use web back-end for both Android and iOS devices. Installing the Parse framework allows applications to perform the following tasks efficiently.

- Creating user accounts
- Uploading objects containing any data
- Uploading files
- Querying objects
- Viewing geographical data
- Sending push notifications

Parse also includes its own pre-built user interface templates for creating an account, logging in and viewing tables of data. The account creation and login windows are fully customizable and allow login via custom user account, twitter and Facebook. Fields can be added to query the user for additional information such as phone numbers and email addresses.

Artwork: Logos and background images can also be changed and customized. The included table window provides the functionalities that enable the programmer to query any classes and adds pull-to-refresh and pagination functionality. Parse also includes security features that utilize access-control lists and anonymous user account creation. The majority of the artwork for the project was created using Adobe Illustrator [22]. Vector images are preferred due to their ability to be scaled without loss of quality. Vector graphics are created by combining geometric primitives, such as lines and circles. Individual pixels are not saved, only information that describes how to draw the vector [23].

This is important for future porting considerations since higher resolution images may be needed for new devices or platforms. The vector art only needs to be resized before being exported to an appropriate format, which in this case is Portable Network Graphics, or PNG. PNG images are lossless, compressed images with an alpha channel [24].

Only the hair was not created in Illustrator since primitive shapes made hair types hard to distinguish. Instead, Adobe Photoshop was used for this task to provide more realism in order for the different types to be more readily distinguishable [23]. The PNG images were imported into Photoshop and the hair was drawn on a layer on top of the face. The face was removed before exporting the hair only as its own image.

Testing and Deployment: Deployment of the application was completed with the aid of the Test Flight service [25]. Test Flight is a web-based program that streamlines the beta testing process. iOS applications that are in the beta-testing phase must be compiled with a provisioning profile tailored to specific devices in order to prevent piracy.

TestFlight allows recruitment of testers and tracks all of their current device models and software versions. The program works by sending an email or link to the desired tester, which they then navigate to on their device. Navigating to this link collects information about the user and their device so that they can then participate in beta-testing. Testers can install the application over-the-air without having to sync their device. Adding a new build automatically notifies testers that it is available.

In addition to these features, a TestFlight SDK can be installed which returns live feedback as testers are using the application. Through the SDK, applications can be remotely logged, ask users in-app questions, generate crash reports, and prompt users to update when a new version is available. The TestFlight SDK was not used for this project, but will be added to future versions.

Original Assignment

A written biology assignment titled You've Got the Cutest Little Baby Face serves as the basis for this paper [26]. The assignment is designed to teach students what phenotypes and genotypes are and how both are related. Students are explained the overall idea behind these concepts in the assignment introduction. From there, the assignment is broken down into seven steps.

- Step one asks the student to pick a mate with whom they would like to have a baby.
- Steps two and three ask the student to determine their own phenotype along with their partners' phenotype and attempt to guess both genotypes.
- Next, the student will determine which gametes will be passed onto his or her offspring.
- Steps five and six require the student to determine the phenotype and genotype or their baby.
- Finally, step seven requires the student to draw a picture of their baby to the best of their ability.

The assignment contains details of twenty-five different traits that students are expected to look at to determine phenotype expression. Drawings of each phenotype are listed per trait along with all possible matching genotypes and information about types of inheritance. Each step that requires a student to record information contains a simple table with columns for: trait, phenotype and genotype.

The goal of this project was to convert this assignment into a more interactive cooperative mobile-based simulation.

Assignment and Game Differences: The interactive game keeps most of the original ideas of the assignment intact while adding more interactive

and collaborative elements. One of the biggest differences is the way the assignment is created and published to students. This step is implemented in the iOS based assignment to keep the student performances in both paper and iOS based assignments similar. This is essential to enable us to perform a comparative study of the two assignments.

In the original assignment, the students create the initial parent phenotypes before determining the appearance of the offspring. The game version has the instructor create the parents so that the student may attempt to determine the parent and offspring genes. This change was made in order to allow the instructor to be more involved with the assignment, rather than leaving it as a self-teaching experience for the student that focuses on a fun, creative drawing at the end. The paper-based assignment asks the students to simply select values from a table to use to create the parent phenotypes and does not adequately test their knowledge in this area.

Layout

Upon first opening the application on an iOS device, users are greeted with a login screen containing username and password fields. For the purposes of this study only, there will be pre-created student and instructor accounts. The final program will allow creation of accounts of either type. **Figure 3.1** shows the layout of the instructor view.

Fig 3.1. Instructor View

Instructor View: Instructor accounts will have the option of associating specific user accounts to their own. Successfully logging in with an instructor account will present the user with a label that displays their account type, buttons for entering various parts of the application, and a label displaying amount of new submitted assignment submissions received.

The topmost option for instructor accounts will allow creation of a new phenotype. The new phenotype button will open a new view that contains a two-dimensional representation of a human face. At the top of this screen is a menu bar with two buttons that allow the user to either return to the home view or save the current phenotype.

Single tapping on the human face in a specific region will select that particular trait or set of traits for editing. The phenotype (trait) editing mode adds a horizontally scrolling list to the bottom of the view, which contains all of the traits for the current selected region, along with buttons for changing their phenotypes and genotypes. Scrolling left or right with one finger on the bottom of the view, moves between traits. Some traits can be changed with intuitive finger gestures without having to tap on buttons.

For example, the eye region has several gesture-controlled movements. Eye separation and height on the face can be controlled with simple panning. A two-finger pinch controls the size of the eyes and a two-finger rotation controls eye slantedness. Any of the gesture-controlled traits have certain thresholds that automatically change the phenotype and genotype. Once a phenotype is created, tapping the save button will save it with a custom name to the application specific Documents folder on the iDevice.

The Saved Phenotypes view is comprised of a UITableView that utilizes a custom table cell in order to add extra data fields to each row. This table is populated by using a for loop to generate an array of all PNG images in the root/Documents directory. Parsing the saved phenotype name from the filename allows it to be displayed alongside a thumbnailed image of the phenotype for easy recognition. The thumbnail is displayed with a UIImage control set to resize the image to fit the aspect ratio.

Since the SavedPhenotypesView is set as the delegate for the UITableView, it receives notifications when an item is either selected or deselected, which allows a selected count to be kept. Whenever this selected count is equal to two, the Create Assignment button is made active. This keeps the user from accidentally trying to create an assignment with the wrong number of phenotypes. When the Create Assignment button is clicked, a reference to the two PLIST files are passed to the Assignment Detail view.

Assignment details consist of a name and description, which are represented by UITextViews that allow editing. The details box contains some placeholder text that explains how the assignment works at the highest level. It is up to the instructor to add or change the details if needed to fit the particular assignment.

The next option on the main menu displays a list of any saved phenotypes on the iDevice. Phenotypes are displayed in a scrollable list with a name, creation date, and thumbnail. Underneath the list is a menu bar with options for editing phenotypes, deletion, and assignment creation. Edit and delete function as expected. The Create Assignment button opens a new view with the purpose of creating new assignments and pushing them to students. This option is only available when two saved phenotypes are simultaneously selected. The assignment creation view presents the instructor with two input boxes for giving the assignment a name and description. Once the details are given, the assignment can be pushed to students.

The last option available to instructors allows the viewing of assignments submitted by students. Here, the instructor is shown a scrollable list of assignments have been turned in along with the names of the students. Tapping one of these assignments shows both the answers for each of the twenty five traits and a visual representation of the offspring phenotype the student has created. This view also allows the instructor to post a grade for the current assignment. A label displays the number of questions the student answered correctly and the total number of questions in the assignment. A textbox is automatically populated with a suggested grade that can be changed if necessary. The following flowchart shows the hierarchy of windows in the instructor view:

Student View: The student view is very similar to the instructor view, but lacks the option to create and view phenotypes. The label at the bottom displays amount of new assignments waiting to be completed.

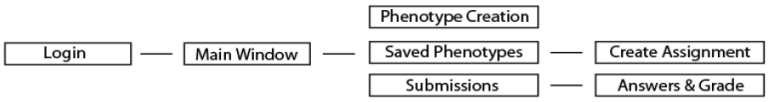


Fig 3.2. Instructor View Flowchart

Students have two options, the first of which allows the viewing of assignments in a scrollable list. The listed assignments display an assignment name and posting date. Tapping an assignment will first show the student a view containing the assignment name and description.

At the top of this view is a back button and a forward button that continues to the rest of the assignment. The next view displays the visual phenotype for Parent A as well as a scrollable list with text labels for each phenotype and genotype. The genotype tab displays the name of each of the twenty-five traits as well as the phenotype entered by the instructor.

The genotype label is blank and has to be filled in by the student by clicking the corresponding button. Each button press cycles to the next genotype. Once each genotype is set, the student can move on to the next parent. The next view is the same as Parent A, but for Parent B. These views can be looked at during the assignment.

Tapping the next button on the Parent B view displays a third window that is the same as the parents, but does not show any phenotype. The student can view the genotypes they have entered in the previous two screens here and attempt to figure out which genes the child will inherit. Again, the genes are selected by clicking the corresponding button to cycle through the choices.

The next and last page of the assignment displays a phenotype window very similar to the phenotype creation screen found in the instructor view. It is in this area that the student will see what the offspring phenotype will look like based on the information given on the previous screen. Once finished, the student will tap the submit button to submit the assignment and be returned to the main menu. The following flowchart shows the hierarchy of windows in the student view:

Fig 3.3. Student View Flowchart

Class and Method Descriptions

The following section describes the classes used in the application and details the methods and algorithms.

Login Screen: After the application is launched, the login screen is presented to the user. User credentials are maintained and manipulated by the Parse framework. When the login button is clicked, the following occurs:

Main Window- MainMenuController

The main window is a subclass of UIViewController and is the first window seen upon launching the application. In addition to allowing both students and instructors to login, this window also serves as the Instructor and Student views.

Login Screen: After the application is launched, the login screen is presented to the user. User credentials are maintained and manipulated by the Parse framework. When the login button is clicked, the following occurs:

Main Menu: The main window is changed to the main menu after a successful login. From here, instructors and students can navigate anywhere in the program where they have permission as well as logout. A label at the bottom of this window shows either new assignments or new submissions, depending on the current account type.

The getNewSubmission method utilizes the query abilities of the Parse framework. A simple query to of the ”submission” class name is made with a filter set to only return a count of the objects that do not yet contain a ”grade” field.

The getNewAssignments method is more complicated. The pseudocode below demonstrates how this method works.

New Phenotype- PhenotypeViewController

The New Phenotype view is created using a combination of UIKit and Cocos2d. First, a UIViewController subclass is pushed onto the top of the stack. Then Cocos2d is initialized by handing the window to a singleton called CCDirector, which handles Cocos2d windows and scenes. This allows UIKit elements to be placed on top of a Cocos2d scene seamlessly.

The Cocos2d scene is used for displaying the main phenotype image, which is divided into many CCSprite objects. CCSprite is a subclass derived from CCNode, which is the main element used by Cocos2d for drawing objects to the screen. CC-Sprite contains several operations that are useful in this project.

CCSprite objects are created and added to the scene for each part of the phenotype that needs to change independently of the rest. The scale method of CCSprite is used to change some of the traits that have multiple size values. Both the *x* and *y* scale values can be changed independently, which is useful in changing shapes of the eyes, nose, and mouth. Using these scaling techniques made possible the use of only one image.

Other traits consist of multiple images that are changed using the texture class member of the CCSprite class. Traits that have different color values were created in white to start with, which allows the color to be set programmatically. Cocos2d uses subtractive color tinting for sprites, which means that images with an RGB value of [*255, 255, 255*] can be set to any color by lowering combinations these values.

UIKit GestureRecognizers are used for detecting finger movements. Three types of gestures are monitored for this project: rotation, panning, and pinch.

- Rotation occurs when two fingers are placed on the screen and turned. This gesture controls some of the facial traits that need to be rotated, such as eye slantedness. Rotation is accomplished by taking the rotation and adding it to the current rotation of the corresponding CCSprites.
- Panning controls horizontal movement of some facial features.
- Pinching with two fingers is used to either increase or decrease the scale of the CCSprite.

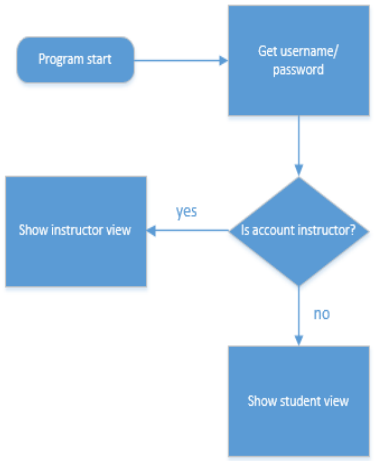


Fig. 3.4: Log in algorithm.

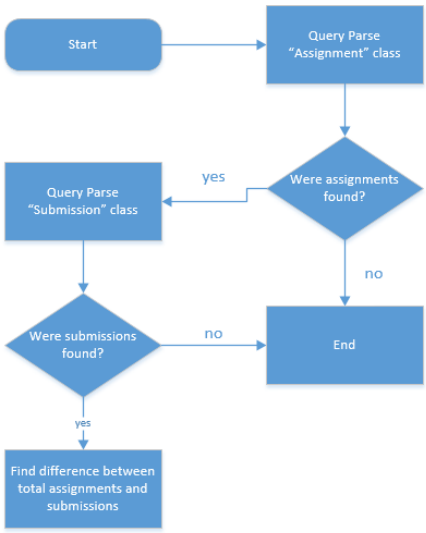


Fig. 3.5: The getNewAssignment Algorithm.

These different methods are mainly used on incomplete dominance traits that have three different possible values.

Each genotype value is held using an integer with the different genes enumerated to allow easy manipulation. Any time a phenotype button is clicked the corresponding genotype is checked and set accordingly with a switch statement. When a phenotype is changed, a method is called in the Cocos2d scene which changes the CCSprites to match.

When saving a phenotype, the following two files are created:

- phenotype+name.plist

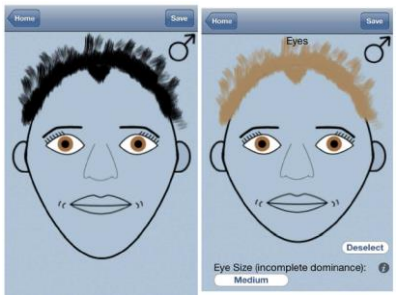
- phenotype+name-phenotype.png

The first file is created by adding the values of each phenotype and genotype to an NSMutableDictionary and then calling its save method. The image file is created using code within the Cocos2d scene. First, an instance of CCRenderTexture is instantiated with the size of the screen bounds. This class is a rendering target that allows any other children of the scene to be drawn to it with a simple method call.

CCRenderTexture also supports saving itself as either a JPEG or PNG image. In this case, PNG is used in order to keep the alpha channel intact. These two files are saved to the application's/Documents directory, which is a sandboxed location that is unique to application and cannot be accessed by outside means.

Saved Phenotypes – Phenotype List View Controller

The Saved Phenotypes view is comprised of a UITableView that utilizes a custom table cell in order to add extra data fields to each row. This table is populated by using a for loop to generate an array of all PNG images in the root /Documents directory. Parsing the saved phenotype name from the filename allows it to be displayed alongside a thumbnailed image of the phenotype for easy recognition.



(a) (b)

Fig 3.6. Phenotype Creation Screen.

The thumbnail is displayed with a UIImage control set to resize the image to fit the aspect ratio. Since the SavedPhenotypesView is set as the delegate for the UITableView, it receives notifications when an item is either selected or deselected, which allows a selected count to be kept. Whenever this selected count is equal to two, the Create Assignment button is made active. This keeps the user from accidentally trying to create an assignment with the wrong number of phenotypes. When the Create Assignment button is clicked, a reference to the two PLIST files are passed to the Assignment Detail view.

Assignment Creation – Create Assignment View Controller

Assignment creation consists of a name and description, which are represented by UITextViews that allow editing. The details box contains some placeholder text that explains how the assignment works at the highest level. It is up to the instructor to add or change the details if needed to fit the particular assignment.

Clicking the Post Assignment button uses four nested Parse saveInBackground calls to upload the two phenotype PNG images and corresponding PLIST files. Each upload uses a block, which is a segment of code that can be passed like an object. When the first upload is finished, the next upload within the block is started and so forth. After the final file is finished uploading the assignment object is created on the server with relations to the uploaded files. The creation date and a unique ID are automatically created and the program returns to the main menu.

Assignment List – List Assignment View Controller

The assignment list is created using a custom UITableView template provided by the Parse framework. Assignments are queried and sorted by creation date and only shown if the instructor name field matches that of the student's user account instructor field. When a list item is tapped, the assignment objectID is passed to the Assignment Description View Controller.

Assignment Description – Assignment Description View Controller

Assignment descriptions are displayed whenever a student first accesses an assignment. UILabels display both the assignment name and the assignment description, which are pulled from the server using the objectID passed from AssignmentViewController.

A check is performed first to see whether or not the assignment has already been completed by the student. This check is simply performed by querying the Parse server for an assignment submission with the student's name. If the assignment has been submitted, another check is made to see if a grade field exists on the submission object. If this field exists the grade is displayed, otherwise a message tells the student the assignment has not yet been graded.

Assignment – Parent View Controller

The ParentViewController class is used for the majority of the assignment windows. Both the Parent A and Parent B windows use this same class as well as the Child input window. References to each instance of this class are saved in the first instance of it which allows the user to move back and forth between without losing any of the data they may have entered. The windows are not released until the first Parent window is popped off the top of the stack.

A reference to the assignment object name is passed between all the instances so that Parse can be queried for the required PLIST and PNG files. Labels are updated by using a for loop to iterate over the twenty-five different traits. Phenotype and genotype strings are stored in arrays in the same order that they are enumerated to make matching them easy.

If the user taps the button to change to the next page, first a check is performed to see if all of the genes have been entered. If so, the genes are stored in an NSDictionary and passed to the next instance of the ParentViewController before it is pushed onto the stack. Upon leaving the Child window, three NSDictionaries are populated and ready to be uploaded. The below figures show the different sections of the assignment covered by ParentViewController.

The submission list works exactly the same as the assignment list. When a submission is selected, the reference to the object is passed to the SubmissionViewController.

Submission – SubmissionViewController

The submission window shows the child phenotype with a tab to view their answers. The genotype tab shows each of the twenty-five traits along with the parent genotypes found in the assignment, the parent genotypes the student submitted and the child genotypes.

If an answer is found to be wrong, the label has its font color set to red so that it can easily be found. This gives the instructor a simple way to visualize the wrong answers.

These answers are placed on a UIScrollView that can be moved up and down to view all the answers. At the bottom of this window is a label that displays the total number of questions and the number of questions the student answered correctly. The grade is obtained by starting with a total of seventy-five and subtracting one for every wrong answer. This value is then divided to get the grade percentage, which is used as a placeholder in the grade textbox.

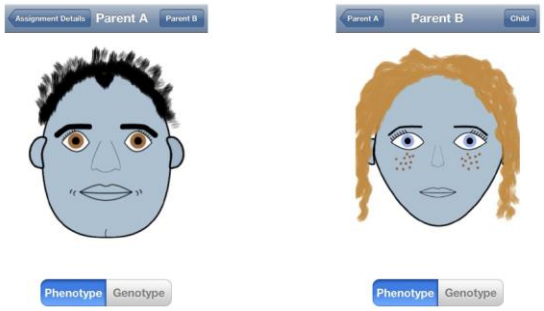


Fig 3.7. Parents Phenotype Screens: Parent A(left), Parent B(right).

Checking submission parents for incomplete dominance traits is simply done by comparing the assignment parent genotype value with the submission parent genotype value. Dominant-recessive traits are different in that two different parent genotype values can show the same phenotype.

For these traits, the student is correct if his or her answer is either value. If the student misses both values, a point will be deducted from their grade.

Grading the child portion of the assignment requires more complicated methods. Both dominant-recessive and incomplete dominance traits both utilize the same algorithm to check the validity of the child genotype.

The following method uses the genes for Eye Shape as an example:

The two polygenic traits, hair color and eye color, require different methods since they involve more genes. Hair color involves four genes, which gives it the most possible outcomes. Instead of using if else statements, patterns were found that make the process much easier.

The assignment suggests that hair color depends on the number of dominant alleles, so the algorithm depends on these counts. Below is the algorithm that demonstrates the patterns used for grading hair color:

The above algorithms were each derived and tested using results found by using Punnett squares to calculate the outcomes of parent crosses.

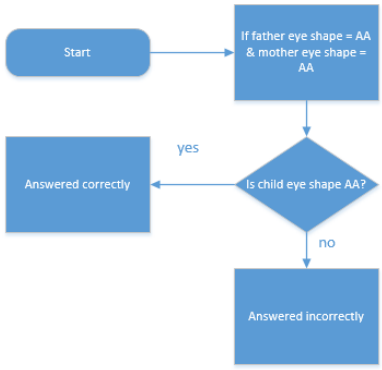


Fig 3.8. A Simple Grading sub module

Global Data – GameState

A singleton class called GameState is used to maintain certain variables across different parts of the

application. Since only one instance of this class can exist, it is perfect for cross-communication. The combined use of a UIKit ViewController and Cocos2d scene for the phenotype creation window requires both objects communicate with one another. Both of the object references are stored inside the GameState singleton when they are first instantiated and remain there for the remainder of the application running time.

4. RESEARCH METHOD

A class of 21 biology students were chosen to complete the written version of the assignment followed by the iOS version. Students were emailed directions on how to sign up for TestFlight and test on their own devices. Students without devices were allowed to complete the assignment on a borrowed iPhone. Start times, finish times and grades were recorded for both for comparison. As of this writing, results are still being gathered. The below tables and figures show the results of both the paper and iOS versions of the assignment.

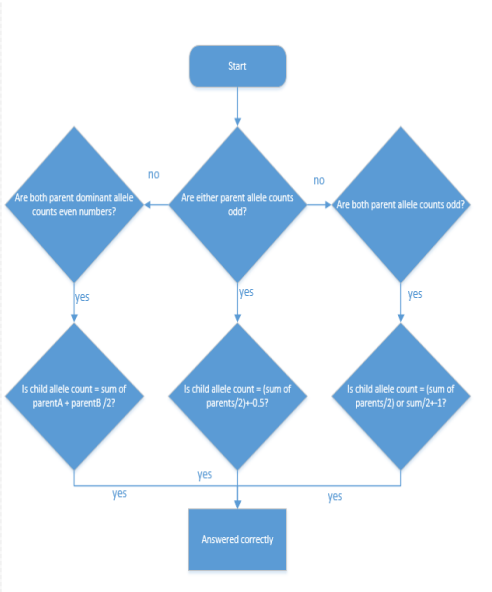


Fig 3.9. A Complex Grading Sub module

Participants: The convenience sample consisted of 21 (10 male, 11 female) undergraduate biology student volunteers matriculating at a south Texas state university ranging in age from 18 to 41. The average age of the participants was 21 years.

Materials and Procedure

A written biology assignment titled You've Got the Cutest Little Baby Face serves as the basis for this paper (Averill, 1993). The assignment is designed to teach students what phenotypes and genotypes are and how both are related. Students are given instructions to complete the assignment. The assignment is broken down into seven steps.

- Step one asks the student to pick a mate with whom they would like to have a baby.
- Steps two and three ask the student to determine their own phenotype along with their partners' phenotype and attempt to guess both genotypes.
- Next, the student will determine which gametes will be passed onto his or her offspring.
- Steps five and six require the student to determine the phenotype and genotype or their baby.
- Finally, step seven requires the student to draw a picture of their baby to the best of their ability.

Table 3.1.Participants Demographics

No.	Gender	Age	Ethnicity	Major
1	M	19	3	CJ
2	F	19	1	NUR
3	M	19	3	CJ
4	F	19	3	EDU
5	F	19	5	CJ
6	F	19	3	CJ
7	F	22	1	EDU
8	M	18	3	BIZ
9	M	41	1	EDU
10	F	19	3	NUR
11	F	18	4	BIO
12	M	19	3	BIO
13	M	18	1	BIO
14	F	19	3	BIO
15	F	19	5	PSY
16	F	19	3	NUR
17	M	19	2	CJ
18	F	22	3	EDU
19	M	19	3	UND
20	F	19	3	MAT
21	F	19	2	UND
22	M	19	3	UND
23	F	35	3	UND
24	M	19	2	PSY
25	M	20	2	BIO
26	M	24	2	BIO
27	F	19	3	BIO
28	F	19	1	COM
29	F	20	3	BIO

Table 3.2. Ethnicity Legend

Code	Ethnicity
1	African American
2	White non-Hispanic
3	Hispanic
4	Asian American
5	Other

The assignment contains details of 25 different traits that students are expected to look at to determine phenotype expression. Drawings of each phenotype are listed per trait along with all possible matching genotypes and information about types of inheritance. Each step that requires a student to record information contains a simple table with columns for: trait, phenotype and genotype.

The goal of this project was to convert this assignment into a more interactive cooperative mobile-based simulation.

Research Design and Statistical Analysis

The study's design was a repeated-measures experiment. Participants completed the biology assignment (mentioned above) first by paper-and-pencil and then, one week later, by iOS devices. A repeated-measures (RM) analysis of variance (ANOVA) was used to make comparisons.

The independent variable was assignment method: paper-and-pencil versus iOS. There were seven dependent variables with time measured in minutes:

Table 3.3. Results Details

No.	Paper time	iPhone time	Paper Grade	iPhone Grade
1	1:05	0:04	85	49
2	0:53	0:10	84	79
3	0:47	0:06	92	41
4	0:42	0:10	98	56
5	7:15	0:08	92	48
6	0:30	0:11	89	51
7	0:53	0:06	93	52
8	0:41	0:12	94	60
9	0:34	0:04	34	40
10	0:39	0:05	94	48
11	1:13	0:05	93	52
12	0:40	0:09	82	55
13	1:00	0:23	86	51
14	0:39	0:18	88	88
15	0:30	0:12	29	91
16	0:25	0:12	86	76
17	0:48	0:06	81	72
18	2:30	0:17	98	92
19	0:37	0:12	97	97
20	8:00	0:05	81	63
21	1:08	0:13	89	60
22	-----	0:13	0	91
23	-----	0:05	0	39
24	0:45	-----	90	0
25	0:16	-----	32	0
26	0:45	-----	90	0
27	2:30	-----	82	0
28	1:15	-----	90	0
29	-----	-----	0	0

1. pGrade: grade on the paper-and-pencil version of the biology assignment
2. iGrade: grade on the iOS version of the biology assignment
3. pTime: time to complete the paper-and-pencil version of the biology assignment
4. iTime: time to complete the iOS version of the biology assignment
5. mTime: time to grade the paper-and-pencil version of the biology assignment
6. iscoreTime: time to grade iOS version of biology test = 0 (scored instantaneously by iOS)
7. (pTime + mTime): total time to complete and manually grade the paper-and-pencil version of the biology assignment

5. EXPERIMENTAL RESULTS

Table 3.4. Grade Statistics

Platform	Min	Max	Avg.	Std.	Median
iOS	39	97	63	19	56
Paper	29	98	82	21	85

There was a statistically significant difference between participants' pGrade (M = 84.05, SD = 18.27) and iGrade (M = 62.90, SD = 17.65), $F(1, 20) = 13.88$, p = .001, partial eta squared = .41, with participants scoring significantly higher on the paper-and pencil version of the biology assignment.

Table 3.5. Time Statistics (h:mm)

Platform	Min	Max	Avg.	Std.	Median
iOS	0:04	0:23	0:09	0:04	0:10
Paper	0:16	8:00	1:25	1:54	0:46

There also was a statistically significant difference between participants' pTime (M = 89.95, SD = 125.13) and iTime (M = 9.90, SD = 5.04), $F(1, 20) = 8.46$, p = .009, partial eta squared = .30, with participants taking significantly more time on the paper-and-pencil version of the biology assignment than on the iOS version of the assignment.

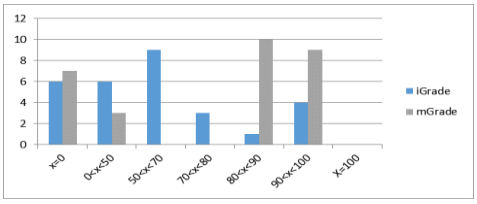


Fig. 3.10. Paper (mGrade) and iPhone (iGrade) Grade Comparison

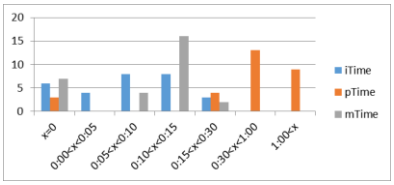


Fig. 3.11. Paper Submission (pTime), iPhone Submission (iTime) and Paper Grading (mTime) Time Comparison.

Finally, there was a statistically significant difference between participants' total time to complete and (researcher) grade the paper-and-pencil version of the biology assignment (pTime + mTime) and the time to complete the iOS version of the assignment (iTime + 0 time to score) = (iTime), $F(1, 20) = 11.23$, p = .003, partial eta squared = .36, with participants taking longer to take and (researcher) grade the paper-and-pencil version of the biology assignment (M = 102.43, SD = 125.52) than to take and grade (0 time) the iOS version of the biology assignment (M = 9.90, SD = 5.04).

6. DISCUSSION AND FUTURE WORK

We see that the iOS grades are lower than the paper grades, which could be attributed to several factors. Given that the paper-based assignment was the first taken, it could be an indication that it did not effectively teach the material. Students taking the paper version were able to look at instructions and examples on the same page while choosing phenotypes and genotypes. Different delivery methods might also be added to match the needs of the specific class or grade-level. The iOS assignment is also different than the paper version in that there are essentially three times the number of questions. This is an even larger factor when grading is taken into consideration. In the paper version human error and time constraints can influence the results. The iOS version is graded by computer and doesn't miss questions or offer leniency.

While both versions of the assignment have no time constraints, the iOS version was completed much quicker in nearly every case. This could also contribute to an overall lower average grade. Students may have not paid as much attention as necessary to earn a good score. Changes to the game may aid the students in staying focused. A deliberate slowing of question delivery could force students to think more carefully about each one and not rush.

Screen size limitations could also contribute to lower grades. The iPhone and iPod Touch devices both have a relatively small screen, which makes presenting a large amount of information on-screen difficult. Due to this limitation, the test had to be broken up into several windows. An iPad-specific version would have more screen space to work with and may better present the assignment to the user. However, this is not a factor yet since no students took the assignment on an iPad.

Obviously, careful planning and testing is required to fully utilize this new technology in the classroom. Attention must be paid to the target population and their skill level. Observation of students as they complete the assignments will help to better understand how different users interact with the various features. Further testing will aid in determining exactly what should be changed and added to the game in order to provide the best experience for all users.

A dedicated iPad version is underway as well as a version for Mac iOSX. Artwork and user interface elements will be improved to provide a better experience for both instructors and students. Future projects will be based on the existing engine that will be tailored for different courses and teaching methods.

REFERENCES

[1] Houghton Mifflin Harcourt, "Student math scores jump 20 percent with hmh algebra curriculum for apple ipad; app transforms classroom education," 20 January 2012. [Online].

[2] C. Harris, C. Mong and W. Watson, "A Case Study of the in-Class use of a Video Game for Teaching High School History," Computers and Education, 2011.

[3] Sony Computer Entertainment Europe, "Little Big Planet," 2012. [Online]. Available: http://www.sony.com.

[4] Procedural Content Generation Wiki., "Procedural Content Generation Wiki," 2012. [Online]. Available: http://pcg.wikidot.com/.

[5] Mojang AB, "Mojang AB.," 2012. [Online]. Available: http://www.mojang.com.

[6] A. Webster, "Educational building blocks: how Minecraft is used in classrooms," Artstechnica, 2011.

[7] MineCraftEdu, "Bringing MineCraft to the Classroom," 2012. [Online].

[8] S. Duncan, "Minecraft, Beyond Construction and Survival," Well Played, 2011.

[9] Massively Minecraft, "Massively Minecraft," 2012. Available: http://www.massivelyminecraft.org/.

[10] M. Schramm, "Working 16 Bit Computer Built Inside Minecraft," Joystiq, 2010.

[11] SMART Technologies, "Smart Boards for Education," 2012. [Online]. Available: http://www.smarttech.com.

[12] F. Gursul and G. Tozmaz, "Which One is Smarter? Teacher or Board.," Procedia Social and Behavioral Sciences, 2010.

[13] C. Alvarez, C. Brown and M. Nussbaum, "Comparative Study of Netbooks and Tablet PCs for Fostering Face-to-Face Collborative Learning," Computers in Human Behavior, 2011.

[14] E. Garcia and A. Friedman, "There's an App for That:" A Study Using Apple iPads in a United States History Classroom," Studies in Teaching 2011 Research Digest, 2011.

[15] Intel, "Classmate," 2012. [Online]. Available: http://www.intel.com/learningseries.

[16] J. Sancehz and R. Olivares, "Problem Solving and Collaboration using Mobile Serious Games," Computers and Education, 2011.

[17] C. Gonzalez-Gonzalez and F. Blanco-Izquierdo, "Designing Social Videogames for Educational Uses," Computers and Education, 2012.

[18] Blizzard Entertainment, "World of Warcraft," 2012. [Online]. Available: http://www.blizzard.com.

[19] Blackboard Inc., "Blackboard," 2012. [Online]. Available: http://www.blackboard.com.

[20] R. Quesada, "Cocos2d for iPhone," 2012. [Online]. Available: http://www.cocos2d-iphone.org.

[21] Parse, "The Mobile App Platform for Developers," 2012. [Online]. Available: http://www.parse.com.

[22] Adobe Inc., 2012. [Online]. Available: http://www.adobe.com.

[23] A. Quint, "Scalable Vector Graphics," IEEE Multimedia, pp. 99-102, 2003.

[24] G. Randers-Pehrson, Portable Network Graphics Specifications v1.2, 1999.

[25] TestFlight, 2012. [Online]. Available: http://www.testflightapp.com/.

[26] E. Averill, "You've Got the Cutest Little baby Face," Favorite Labs from Outstanding Teachers, 1993.

[27] Houghton Mifflin Harcourt, "HMH Fuse," January 2012. [Online]. Available: http://www.hmheducation.com/fuse/.