ABSTRACT

Jungle Gym was born out of a goal to build a Visual Programming Language (VPL) that is compatible with iOS devices in order to raise and stimulate an interest in computer science and programming in children. This goal was accomplished through a web-based integrated development environment (IDE) where users can interact with the VPL to create projects, and an associated native iOS application where users can download and run their projects. The hope is that this product will mimic the success that is seen in other VPLs, such as Scratch [9] and App Inventor [10], and that JungleGym will impart a positive view of computer science and programming on the target audience. Related research studies have shown the effectiveness of VPLs with children in both generating interest in programming and in general their performance with programming. Unlike existing VPLs, which are either entirely web-based or Android-based, this project aims to fill the gap of iOS specific VPLs by being iOS specific and incorporating iOS-specific features such as the touch screen and camera.

1. INTRODUCTION

While computers and the Internet have been ingrained into many aspects of children’s lives, most children are still not exposed to programming and computer science. The goal of JungleGym is to give means for children to find and develop an interest in programming and computer science by using the fact that most children have access to an iOS device, whether that be an iPhone or iPad. The new visual programming language allows users to create applications for iOS, Apple’s mobile operating system. iOS was the chosen platform because Apple products are extremely popular and there is no current VPL geared towards creating iOS applications, while some exist for Android and web development.

VPLs are programming languages that allow users to code using a graphical interface as opposed to a more traditional text-based one. This allows users, especially those just starting to program, to focus on the core ideas of programming and without having to struggle with syntactical specifications that come with traditional programming languages. Some examples that are further explored in Sec. 2 of this proposal include Scratch, App Inventor, and Blockly [5]. In order to have a greater impact on the target audience, JungleGym uses a new VPL designed specifically for iOS development because the graphical nature of a VPL allows users to ignore the language-specific knowledge required to code in Objective-C. Since the target audience is elementary to high school students with little or no prior experience in coding, eliminating any need for programming-specific knowledge is essential.

The application was designed to be easy to use; the emphasis was not on generating actual code but teaching larger coding principles through a drag-and-drop “build-it-yourself” interface. The distinguishing factor of the application is the feature of publishing the user-designed iOS application to an Apple device. In order for users to view a product reflecting their work, the user-created application is hosted and viewed through the iOS application.

The software has the capability to create simple applications. Some examples of the types of application that can be created are a simple photo editing app where the user can upload or take a photo and then can draw on it with colored lines, simple games such as Frogger and apps that allow for simple data entry into a table such as a to-do list. These applications were chosen to familiarize new users because they all work with different aspects of the iOS device. The photo editing app works with the camera and touch screen. Frogger works with the touch screen (tapping the screen to signal the frog to move) or the accelerometer (shaking the phone forward to signal the frog to move). The to-do list works with iOS tables and data entry.

The JungleGym-created application that will be focused on in this paper will be that of a simple paint program where users interact with a drawing pad and can use the iOS device’s touch screen to paint pictures.

2. RELATED WORK

One of the earlier attempts in the field of visual programming languages is Alice, a software for 3D projects released in 1998 under Carnegie Mellon University. Alice was designed to support object-based programming, while abstracting away the need to use specific syntax. In The Design of Alice [1], Stephen Cooper analyzes the pedagogical and technical strengths and weaknesses of Alice as a tool for teaching programming. He found the program to be successful in motivating students and appealing to a wider and previously underrepresented audience. It is also effective in helping students to understand calling methods and passing parameters. Alice was also found to have a gentle learning curve, as children were able to learn to use the program successfully in less than an hour. However, there were some areas where Alice fell short. Cooper found that, specifically in Alice 2, object-oriented concepts such
as inheritance and polymorphism are limited. For example, all classes inherit from the Object class, and newly created classes cannot inherit from any class other than the Object class [1].

Another major VPL in use to teach programming is Scratch, an application developed in 2003 at the MIT Media Lab. Scratch adopted the drag and drop editing used by Alice while also adding more user-friendly visuals to appeal to grade school students without prior exposure to programming. Scratch uses a drag and drop puzzle format where code pieces “snap” together, such that only valid code pieces can be connected and specific language syntax does not become a barrier for learning. Scratch also introduced an online repository where users can share their projects as well as view the projects of other Scratch developers around the globe [8].

A study done on the effectiveness of using a Scratch-based course as a precursor to a Java-based course for at-risk students (defined as students with weak mathematical backgrounds) showed that the program was effective in retaining students in the computer science curriculum, as the retention rate was raised from 33 percent to 57 percent. There was also a demonstrated improvement in students’ performance, as the pass rate of the higher level computer science course increased from 39 percent to 74 percent. The Scratch program was praised due to its inclusion of more advanced programming concepts such as concurrency and event-driven programming, as well as basic concepts such as variables and arrays [13]. In fact, the idea of being able to introduce children to programming through Scratch led Berkeley to build their own VPL, Snap! (previously BYOB [Build your own blocks]), which is based on an implementation of Scratch [4]. Because VPLs are constructed in a way to make programming easier, they are seen as easing the cognitive processes behind this type of problem solving and reducing the mental effort required to code [14]. These benefits are likely why VPLs work well with children as it gives them the same problems that they would encounter in more traditional programming but reduces the actual complexity to a state where they are able to come up with a solution.

The main difference and improvement between already existing VPLs and JungleGym is the platform on which projects are run. Both Scratch and Alice are limited to developing and running applications on a computer. Another VPL, App Inventor, enables users to develop on their computer for a smartphone setting, however it is limited to Android devices [12]. App Inventor uses an external library, Blockly, in order to build the blocks used for their VPL. Thus JungleGym fills the obvious smartphone void and specifically targets developing applications that can be run on an iOS device. Building on what Blockly already does, additional drag and drop elements that would correspond to different aspects specific to an iOS device, such as the camera, the music player, accelerometer, touch screen, etc were implemented.

3. SYSTEM MODEL

The block diagram (Fig. 2) has three distinct sections (Computer based GUI, cloud-based database, and iOS application) with two additional smaller steps to connect the components. These three components are the the basics of what was needed to complete JungleGym and that the two smaller steps successfully link all the components together and allow JungleGym to be fully functional.

The first component is the computer based GUI. For this part of the project a web based GUI where people sign up as users and use the VPL was built. The first stage of the web based GUI where people sign up as users and use the VPL is a screen designer where users build the view and layout of the application itself. Here the must list all view elements specifically as they can only use preexisting view elements when building the code for their projects. After the screen designer, the user is taken to the code environment, which allows them to see the blocks that they are using, as the VPL is an extension of Blockly, and will give them a sense of how their application will work, given they understand how the blocks are fitting together. They then be able to upload their content to their user account. This action requires the first of the two mini steps that interlink components. This first step counts for a major part of JungleGym’s functionality as it is the compilation of the VPL code. This step takes the user’s code, the extended Blockly, and convert it into proper javascript and takes the screen designer output and constructs a properly formatted JSON file, which it then uploads to the second component of JungleGym: the database.

The second component of JungleGym is the database. A cloud-based database service was decided upon because it was the easiest option and would allow the focus of work to be on other parts of the VPL as opposed to spending a lot of time setting up a JungleGym-specific database. JungleGym uses Parse.com [11] as its database service. Parse provides a simple way to create the necessary tables with the appropriate schema and provides a simple way of connecting to
both web based and mobile applications.

Figure 2: Block Diagram

The third and final component is the iOS application. This is the part of JungleGym that allows users to actually use the projects that they built with the VPL via the web component. The application allows users to sign in with the same credentials and then will display all their currently uploaded projects for them to select from. After picking a project to run, the application will query the database for that project and then dynamically construct a new application view filled with the content of the project so that the user may actually try out the project that they wrote.

With these three components fully interlinked a person can sign up as a user and then proceed to use the VPL to build projects that will be stored on the database associated with their user account so that they can then use the iOS application to then download and run their projects. Thus a user is able to fully utilize every aspect of the application. As such, anything else that is added and could be add would be to further improve the user experience as it will be extra from the core functionality. Such improvements will be discussed in Sec. 6.

4. SYSTEM IMPLEMENTATION

4.1 iOS Application

As discussed earlier in the paper, JungleGym has a native iOS application that connects to a cloud database and allows for users to login and download and run their projects built in the web component of JungleGym. When the application opens it brings the user to the login screen, where they can enter their user credentials to enter into their account. It should be noted that these are the same credentials that they will use when using the web application to build projects with the VPL. Also visible on the starting view of the application is a button labeled “Sign Up,” which takes the user to another view where they can create an account. This option exists on both the web and iOS applications in order to ensure that if someone finds JungleGym on the App Store they can sign up and log in and be informed of JungleGym’s web component. If the user uses the Sign Up view to sign up with JungleGym then they are also logged in when they submit the request.

When a user successfully logs in they are taken to the main view of the application. The main view of the application is a table view, where each cell in the table corresponds to a project that they have built and uploaded to their account. When the user clicks on a cell in the table view, the files associated with that project are downloaded and the application begins construction of the project. The user is then taken to a new view, where they can run their project. The layout of this view is designed by the user in the screen designer portion of the web component and is thus a dynamic view. The functionality of this view is also dynamic as it is the code that they produced when working with the VPL. When the user is done, they can simply hit the cancel button that is on the view and go back to the table with all their projects. This view is dynamic in layout as the screen designer produces a JSON file that lists all the view elements with their positioning and text. When the view is being constructed that file is downloaded and parsed to create all the elements that it lists. This dynamic functionality was accomplished by having the code designer produce javascript and using the JavascriptCore library and a JSObject in Objective-c. Thus the application can download the javascript file and map the responses of different view elements to functions in the javascript file.

For example, given a simple paint program where a user has colors to select from while drawing on a canvas, an iOS view could be constructed as in (Fig. 3). The view for this program is very simple as not much is needed. The main white space of the view acts as the drawing canvas and is thus meant to be empty and white. The buttons on the bottom of the view correspond to pen colors used in drawing and when clicked, will trigger the javascript functionality to change the color of the pen used. The title bar of the view is also set by the screen designer, which will be mentioned more later in the paper. The only parts of this view that are not dynamically constructed from the generated JSON object are the status bar, where the carrier information and time are listed, and the cancel button, which is a static button meant to allow the user to return to their list of programs.

4.2 Web Application

The computer based GUI system uses a Python backend with a Flask [2] web framework. Flask was chosen over other more extensive frameworks such as Django [3] because of its flexibility in being able work with different database
types (specifically NoSQL which is what Parse.com uses). Flask also has access to optional extensions which allows for more involved web components to be abstracted away while still allowing for the application to stay light, devoid of extensions that are not necessary. One such extension that JungleGym uses is Flask-Login, which handles user session management while giving the freedom of user authentication type (for example, OpenID versus the more traditional username and password pair) and how the user is loaded, i.e., database specifications. To simplify the visual interface, Bootstrap [6] was used as a simple HTML, CSS, and JS framework due to its responsive nature.

The screen designer portion of JungleGym’s web component, displayed in Fig. 4, is where users build the views for their projects. The screen designer has a simple layout, where users will add the view elements they want by clicking on the add button associated with the type of view element they wish to add. When they are adding an element, they must provide a name, a label, and x and y coordinates. The name is used to reference that view element in the code designer component, as discussed later in the paper. The label is the text that appears on the view element, typically to denote what that element does. And the x and y coordinates are simply for placement on the screen. If the user makes a mistake or decides they want to remove a view element, all the elements are listed in a table in their respective view element section and all have a remove button associated with them.

It is here, in the screen designer, where the JSON object that the iOS application uses to dynamically construct the view is constructed. After the user has completed their design and they submit it to move onto the next step, the code designer, the JSON file is uploaded to their user account as a new project and the relevant view elements are passed to the code designer as it is loaded to allow the user to actually reference parts of the layout they just constructed.

As mentioned in Sec. 2, JungleGym’s VPL is an extension of Blockly. An example of the Block Factory which allows for the creation of custom blocks is pictured in Fig. 5. The preview window gives a preview of what the block will look like in the VPL editor, while the language code window specifies how the block will render the block shape. The generator stub gives a starting point for turning the block into Javascript. Since every block name must be unique, to avoid collisions JungleGym’s VPL will be following the naming convention that each block name must be prefixed by its category, so for example the camera block will be prefixed by the media category.

In Fig. 5, the camera block allows for the user to choose between the various cameras on the phone (front camera, back camera, and screenshot). It exports the choice as JSON object by editing the generator stub, which can then be used by the Objective-C code to invoke the correct camera. Also, because this particular block allows for top and bottom connections, it can be used as a condition for if and while blocks, such as the probably common block “if button is pressed, take picture with CAMERA”.

Like the example camera block, similar blocks were made with the other media function of the phone, specifically the camcorder and sound recorder, as well providing the ability to import already existing sounds, vibrations, and images on the phone. Another type of iOS specific block that will be made is the “Open Application” block, which, similar to the camera block, will allow the user to open a separate application on the phone. Restrictions will be put in place to prevent the invocations of (1) applications that are not appropriate or that do not relate to the end goal of the overall VPL app and (2) applications that the end user of user-created application does not have on their iOS device.

The code designer portion of JungleGym’s web component, as displayed in Fig. 6, has another very clean and simple layout to allow the user to focus on actually coding their project. All the graphical elements are sorted by a relevant keyword in the sidebar and the user must simply click on
the keyword for the side bar to slide out and present them with the different possible blocks they could be looking for. They then just drag them into the editor portion of the web page and begin fitting blocks together. When they are done coding their project, they hit another submit button and their code is compiled into javascript and the javascript file is uploaded to the database in the same project as the JSON object they generated in the screen designer.

Figure 5: Blockly Block Creation

Given the same simple paint program as discussed in Sec. 4.1, Fig. 6, displayed below, shows the VPL code associated with that project. Here, the names of buttons (Red, Black, Blue), are given to the code designer after the user has created them in the screen designer and has submitted their view design. Also demonstrated is the “whenCanvas-dragged” block, which was added to JungleGym to work with the touchscreen of an iOS device. The variables used with that block aren’t specific to any view element, so they are not required to be listed in the screen designer. These are the only blocks needed to construct the simple paint program, Drawpad, which only allows a user to select a pen color (using the top 3 blocks in the figure) and to draw on the canvas.

4.3 Tutorials

Initially, there were the basics for four initial tutorials that all work through a sample program and then give an open ended description of how the user can expand the project and make it more complex. The four tutorials were of different difficulty levels. The tutorial themselves will be used mostly by children so the aim was for them to be extremely detailed but also easy to understand. The tutorials were to be highly visual and include pictures for each step. Because the actual functionality of JungleGym was more important during development than the construction of the tutorials, the tutorials remain more on the basic side. They can easily be edited to be more detailed and visual simply by taking the time to work on them. However, despite not working extensively on them, the tutorials themselves are actually an important aspect of JungleGym as it is meant as an educational tool for children and thus the children would need a starting point.

The first tutorial is a simple program that acts as this language’s version of Hello World and consists of two parts. The first part is simply displaying the words “Hello World” on the screen. The purpose of this tutorial is to give new users a feel for how to create a new project, deploy a project, and load it on the phone. This step will not involve the code designer but will focus on using the screen designer. The user will have to add a label to the screen, choose its position, and change the text. The second part of the first tutorial will involve interaction with the code designer. This part is also where the concept of a variable is introduced. Part two of the tutorial involves creating a project such that when a button is pressed the number on the screen increases by one. This is where users will see the integration of the code and screen designers. They will have to use the screen designer to create a button and label to use and will use the code designer to actually add functionality to these view elements. If a user wants to go further with the first tutorial, they are encouraged to add a button that decrements the list by one when clicked.

The second tutorial introduces users to using an iOS device’s keyboard. This tutorial guides users through making a project for simple data entry to create something similar to a to-do list or a shopping list. Using the screen designer, users will have to add three components to their application view: a label, a text field, and a button. Users will also have to add functionality for the keyboard. In the code designer, users need to use the “when button is clicked” action that was described in tutorial one. They then add the action for...
reading the input field text and adding that text to the end of the label variable so that it appears on screen. If a user wants to expand the complexity of this tutorial they are encouraged to add a variable that keeps track of how many inputs have been added and numbering them, thus making the list a numbered list.

The third tutorial introduces users to using the camera feature and an iOS device’s touch screen capabilities. The project the users build with this tutorial can take a picture, draw on the picture, then save it to their iOS device. This means that in the display editor, users will have to select having camera capabilities, add a button for taking a picture, and add a button for saving the picture. In the code editor the first step is similar to the first tutorial where they have to add a button and then they will drag the “call camera” into the “when” statement. Then users will have to add a “when figure is dragged” action that will pass to the users the following variables “current x”, “current y”, “previous x”, and “previous y”. Users will then add the “draw line” action that needs to be passed the following variables: “current x”, “current y”, “previous x”, “previous y”, and color. Users will then need to add another “when button is clicked” that will call the “save to image file” action. To further build on this project, users are encouraged to add functionality for having multiple colors that a user can switch between instead of just one. Even further, users are encouraged to add the ability to send the image via text or iMessage from the application itself to add even more complexity to this project.

The fourth tutorial is the JungleGym version of Frogger. This tutorial introduces the idea of image objects, interaction between objects, moving an object, and the use of the iOS device’s clock. In the screen designer, users have to add images for the frog, the cars, the background, a button, and add capability for the clock. In the code editor, users select the “On Clock Tic”. In this action, users have to add “If Statements” that check if the frog object is touching one of the car objects. In this action, the x-axis position of each car will change and there is also a check to see if the car has made it to the other side of the screen and if so it will change directions on the x-axis. There is a “When Button is Clicked” action that changes the frog object’s y-axis position. This button also checks to see if a car object was touched or if the frog object has reached the end of the screen, meaning that the user has won. To expand on this tutorial, users are encouraged to create three separate “On Clock Tic” actions so each car moves at a different speed. Users could also add a label that only appears on winning that shows the amount of time spent or displays a victory message. A more complicated expansion of this project given to users is to remove the frog object itself and to use the touch screen and their finger to maneuver around the cars.

5. SYSTEM PERFORMANCE

Because JungleGym is an expanded version of Blockly, the VPL itself is a full language. After construction of the tutorials used to base the work around, it was evident which iOS
specific features should be incorporated into Blockly right away. As such, the added iOS blocks work with the existing blocks and can be interweaved in with other Blockly code. Because the code designer portion of the web component is just working with JungleGym's VPL, it's a very simple interface and is a very streamlined process.

The screen designer is the more intricate portion of the web component of JungleGym. Because it needed to ensure that the view elements correspond to actual code written, the screen designer was constructed to take this into account. When a user makes a project using JungleGym, they must first use the screen designer to create the layout for their project. This is so that the code designer is auto-populated with view element options so the user can only refer to view elements that they have actually created. Using the screen designer is easy as one must simply click on an add button for the type of view element they wish to add and then to fill out the labeling and coordinates for the element. When the user is done working on the layout of the screen, the JSON serialization is created and uploaded to their user account. They are then taken to the next step, the code designer.

The code designer, while still a vital aspect of the web component, is less intricate as it is mostly borrowed from the Blockly library. Save for the addition of the iOS specific blocks, the code designer simply displays Blockly as it is given by Google. This section of the web GUI is meant to be simple and mimic the way actual Blockly is constructed, on a clear environment with the code acting as a puzzle.

When the user is done with this part of their project, their code is compiled into javascript and also uploaded to their user account.

Everything that a user writes and designs through the web GUI is uploaded to their user account when they submit. For each project, the JSON and javascript that they generate as the go through the web GUI's development pipeline are stored together in the project tables of the database. Every time a user uploads a project and an entry is made in the project table, the user's project column, which is an array of project id's, is updated and the new project id is added.

The design of the iOS application portion of JungleGym is kept simple so as not to remove from the experience of the user running their own projects. Really the only functionality it has is to show a logged in user a table of all their designed projects and then to construct the view for the project that is selected. The user is then taken to the new view where they can interact with the project that they built. When they are done, there is a cancel button that appears on every project-related view so the user can return to the list of project that they have built and select another one to run.

6. FUTURE WORK

Because JungleGym is separated into two major components, there is work that can be added to both the web component and the iOS application and work that can be added to improve the integration of both parts of the project. As
JungleGym uses Parse.com for the database component, it is limited to their usage restraints unless a database is implemented specifically for JungleGym or another database host is used.

In terms of overall language design, as in Alice, JungleGym still cannot completely encapsulate object-oriented ideas such as polymorphism, encapsulation, and inheritance. Being able to support these concepts would result in a much more complete system overall.

Future work on the iOS application includes finding a way to allow users to create multi-view applications as currently JungleGym only supports single view applications. Also, while it does properly parse a JSON object to construct the layout of the view, it only accepts a few parameters and it can be improved to allow for more customization of application views.

The web application can be improved by working to add more blocks specific to the use of iOS device features. Also, as more iOS devices with different sized screens are released, JungleGym will need to have a way to specify the size of the screen the person intends to design with the screen designer. Other possible additions include creating a code previewer such that the user can view what product their current code will compile to as they program.

Another improvement that would be to allow users to edit existing projects. The case could be that the user constructs something but after running it on the phones wants to make a change. Instead of having to redo the whole project it would be better to allow them to just make the simple changes.

Designing the look of tutorials and lesson plans still need to be done to give users a means of learning how to use the VPL to create their own apps and use them on their iOS devices. Because the iOS application will create a dynamic app based off JSON data, these tutorials won’t be the only applications that users will be able to make; however, they will give a good starting point for new users and for use in the classroom when it get tested on children in a classroom setting.

Another addition would be to make JungleGym social. Adding a social network component where users can add their friends and share their applications. This could also be extended to add the ability for collaboration; multiple users could potentially work together to develop applications. This would also have the benefit of introducing students to the concept of peer programming.

7. ETHICS

As JungleGym grows to support more features of the parent language (Objective-C), the possible JungleGym projects will become much more complex, allowing for more varied tutorials to give a broader educational experience. However, as JungleGym becomes more encompassing of traditional iOS development, it will become easier to use it for development for purposes other than educational reasons, taking advantage of its ability to circumvent the traditional barriers that come with developing for iOS put in place by Apple. This poses not only a legal risk, but also a security one, as people could potentially introduce malicious applications made through JungleGym. A possible mitigation for this problem would be to have some type of system in place to screen JungleGym applications for malicious or undesirable content before allowing the application to be shared with others.

Other than security and legal issues, JungleGym does not introduce any other ethical complications. As it is a free educational tool, users should not find themselves financially at risk nor should they worry of identity compromise. There is also no reason for users to worry of any physical or mental risks when using the VPL.

8. CONCLUSIONS

8.1 Evaluation and Results

Five main criteria based on standards developed by James Kiper, Elizabeth Howard, and Chuck Ames [7] were used as a basis for evaluating the VPL system. Those criteria were the visual nature, the functionality, or how well varied functions are supported, the ease of comprehension, the level of paradigm support, or how well the VPL supports the native language of Objective-C, and the scalability of created projects, or how well larger as well as smaller projects can be created using the VPL. Due to the intended audience of grade school children, it was decided to focus especially on the first three of these criteria in order to allow users to use varied functions and commands that well represent more traditional iOS development while still having an easy-to-use method for learning.

In relation to the visual nature metric, JungleGym utilized different techniques to promote this metric. For example, color was used effectively to help differentiate the types blocks with groups of blocks such as variable blocks, conditional blocks and function header blocks having the same color. Different colors were assigned to function headers, variables, conditional, etc. Another would be the actual shape of the VPL blocks. They are shaped like puzzle pieces and will only fit together if the code is viable.

In relation to the functionality metric, JungleGym puts priority on implementing Objective-C functions and features that are commonly used in various types of applications. This way, it encapsulates the core functionality that most developers for iOS devices require. For example, JungleGym includes swipe detection, buttons, labels, camera usage, internal clock usage, etc.

In relation to the ease of comprehension, JungleGym is easier to understand then it's sister language, Objective-C, especially for the targeted audience of younger students. Because of its graphical nature and the puzzle-like structure, JungleGym has a lower learning curve than more traditional programming languages, including Objective-C. As JungleGym is meant to be an educational tool, it is essential to maintain this low learning curve.

8.2 Outcomes

What JungleGym was able to accomplish was creating an environment where students can explore and delve into computer science and app development. As mentioned previously, VPLs have been proven to be one of the best methods for teaching children about computer science. Since they are syntax independent, VPLs allow students easily learn the logic behind coding. Given that JungleGym is iOS specific, students are learning this logic in a way that is relevant to their everyday lives. Through JungleGym, students are being given an opportunity that did not exist previously; they are able to develop iOS apps without the need for an Apple computer, Xcode, an expensive developers license, or going
through the App Store. Students are encouraged to explore their creativity through the development of a wide range of applications.

9. REFERENCES