

# **Mobile and Web Game Development: Using videogames as an education and outreach tool**

Fernando I. Jaime

Mentors: Nancy J. Leon, Austin Fitzpatrick – 172F

Jet Propulsion Laboratory Summer 2012  
Minority Student Programs  
East Los Angeles College

Few tools reach out to capture the imagination and interests of children like video games do. As such, the development of educational applications that foster young minds' interest in science and technology become of the utmost importance. To this end, I spent my summer internship developing outreach and educational applications in conjunction with JPL's Space Place team. This small, but dedicated, team of people manages three NASA websites that focus on presenting science and technology information in such a manner that young children can understand it and develop an interest in the subjects. Besides the websites, with their plethora of educational content presented through hands-on activities, games and informative articles, the team also creates and coordinates the distribution of printed material to museums, astronomy clubs and a huge network of educators.

Over the course of my internship I collaborated with every member of the team to produce two educational game applications in Flash, which are to be incorporated on NASA's Space Place and Climate Kids websites. The team members are Nancy Leon, team lead,, Austin Fitzpatrick, lead programmer and my technical mentor, Diane Fisher, content writer, Alex Novati, technical illustrator, Laura Lincoln, partner liaison, and my fellow intern and schoolmate Joseph C. Meneses.

NASA's Space Place website focuses on teaching children about stars and galaxies, the sun, the Earth, and the solar system, as well as the people and technology that make space exploration possible. The Climate Kids website focuses on climate change science and the effects of changing conditions on people, animals and the environment. It also profiles individuals with "green" careers and provides educator resources. The SciJinks website is sponsored by the GOES-R satellite program, and it

teaches children about weather and other Earth science topics, space weather, and satellite technology.

The first game application produced for this summer project was developed in conjunction with my fellow intern Joseph C. Meneses. Because of our inexperience with the ActionScript 3.0 language and the Flash authoring tool, our technical mentor Austin Fitzpatrick encouraged us to acquire hands-on experience by beginning to write code from the start of the internship. In such manner we learned the basics of developing in Flash, and as is to be expected our first assignment required several rewrites to ensure its usability and ease of maintenance by the Space Place team after we are gone.

This first assignment was titled Solar System Scramble, a tile-based puzzle game similar in feel to scrabble in which users are presented with visual clues to help them guess a solar system object (for example Sun, Venus, comet) based on clues. For this assignment my task consisted of developing a user interface that would provide a mechanism for serving the visual clues to the player. With the help of my mentor I designed a software architecture that enabled me to feed the game engine XML files containing the puzzle answers and the image URL's for the corresponding clues. By putting the game content into an XML file, the application can be easily adapted for use on the other two websites.

I achieved this great reusability and ease of modification through the use of Flash's native loader and URLRequest classes. This approach allowed me to load the image clues into the user interface however it was not sufficient to hold together the three clues pertaining to each answer. So I devised a custom class of name BitMapLoader that would act as a large bitmap image with the clue images side by side. Furthermore, I

positioned the BitmapLoader in the user interface in such a way that only one clue would be visible at a time, with a sideways scrolling animation effect each time the user requested a new clue or went back to one. The control system for such scrolling consisted of simple buttons, each one corresponding to one of the three clues assigned to each puzzle. Once enabled by using the next clue button, the users can toggle among the different clues by simply clicking their corresponding button. This button feature uses an imported function of the open source TweenNano library, which permits the “tweening” (movement) of graphical objects on Flash’s stage according to the parameters desired by the programmer. I made these buttons using one of my mentor’s custom classes, which enabled me to easily assign different frame labels to the buttons. In such a way the buttons could display different animations depending on their state —that is faded out if the clue has not yet been revealed, a flash of color when pressed and full opacity when selected. I was also responsible for the transitions between the title screen, game screen and game over screen for this application. This task was made relatively easy by the fact that the screens (instances of custom classes themselves) extended a custom class developed by my mentor that made fading in and out an object in Flash as easy as binding a simple function call to a button click. The custom class developed by my mentor made use of Flash’s native effects library, but in a much more intuitive and simple way.

The greatest problem I encountered in the first assignment was learning the quirks of ActionScript 3.0 and the way Flash handles elements on the “stage” (the blank canvas in which the game or animation plays). Once I understood that all objects in a Flash movie are children of the stage, and that those children may have children themselves,

referenced through the constructors of their corresponding classes, I was able to start handling “screens” and placing their corresponding children objects inside them.

For my second assignment I worked independently from Joseph in a “20 Questions” type game, in which the player is presented with a selection of endangered animals and asked to select one mentally. The application then asks several questions regarding characteristics of their animal, such as “Does it eat meat?”, “Can it fly?” and more. The application then guesses the animal. If the guess is incorrect, the player is presented with the list of discarded animals from which they select the animal they were thinking of to see which question concerning their animal they got wrong. Afterwards they are directed to a screen that briefly describes their animal and explains why it is endangered. Likewise if the computer guesses correctly the player sees the same information about their animal.

At any point during the series of questions the player has the option to click on a “Learn more” button to see which animals have the characteristic in the question. In addition, the audiovisual effects of the application —such as computer “beep” sounds, a typewriter effect for the text and a “talking monitor”— were carefully planned in order to keep the player engaged with the content.

XML files became once again the weapon of choice to handle the feeding of data to the application in order to make its design reusable for the Space Place team. Two XML files were constructed. One file contains a listing of the animals, each one with a set of true/false characteristics, a URL string pointing to an image of the animal and finally a tag with a description of the animal. The second XML file contains a listing of all the possible questions as well as descriptions for the characteristic the question is

asking for. Example: “Is it a mammal?”, “A mammal is a warm blooded animal. It gives birth to live babies (rather than laying eggs) and nurtures its young with milk from the mother.” When the application initializes, the XML listings are pushed one by one into objects of the custom classes “animal” and “question,” which allow for variables to capture each one of the characteristics of the items in the XML files. The animal and question objects in turn, are pushed into an array of questions and an array of animals to be used by the application’s logic handler class, the “game screen.” In addition, the question objects are assigned an “ID” value through the numeric value of the question array length. That is, the first question pushed into the array will be question “0,” and the second question pushed will be question “1” and so on.

Using the two arrays created, the game screen class will ask the first question at random. Based on the answer to that question, the “psychic computer” begins to implement a simple algorithm that asks the remaining questions according to a value determined by how many animals the question may eliminate. This is made possible thanks to the self-evaluating mechanisms contained within the animal and question objects. Each animal object contains an “answers” array of true or false values passed in from an XML file that correspond to each one of the questions in the game. Using this array, the animal objects can make use of a class function that takes in a question to determine if the animal would pass it by simply checking the true or false value that the animal has for that particular question. The animal objects are able to identify which question they are being asked by using the id value assigned to the question objects at the moment of creation. Furthermore, the question objects make use of an evaluating function that takes in as a parameter the current array of animals in the game screen

object, and by using the pass/no pass function in each animal object and comparing the number of animals that pass the question to the number of animals in the array, it assigns them a value. The more animals a question can eliminate the more valuable it is. The question objects evaluating function gets called each time a new question will be asked to the player, as part of sorting function that the game screen object implements in order to sort the questions array such that the question with most potential to eliminate the most number of animals gets asked. This approach was taken in order to ask as few questions as possible, ensuring that the game feels like a game, not a test.

Other programmatic aspects of the game worth mentioning are the audiovisual effects. Because trivia games tend not to be too fun after a couple questions, there needed to be a way to engage the young players besides the questions and teaching them about their favorite endangered animals. Thus, with the help of my mentor I created several custom classes to handle the audiovisuals; the most visible among these is the “talking monitor class.” The application contains several screens that feature a computer monitor with moving “voice lines.” These “voice lines” start moving at the same time that the text fields on the screens begin filling in one letter at a time and stop once all of the text fields are completely shown. The effect of filling in the text fields one letter at a time was achieved through a custom class named “typewriter,” that handle insertion of string characters into the text fields as well as producing a “beeping” sound each time this happened. The sounds are selected at random from an array of sounds contained within the music and sounds custom class.

It is very important to point out that the development of an educational game application is not the work of the programmer alone. For both of my assignments I

worked not only with my mentor Austin Fitzpatrick in programming issues, but also with Diane Fisher for the content of the assignments and with Alex Novati for the illustrations and the graphics that make the applications look so good. As a result of this collaborative effort not only did I acquire programming skills over the course of my internship but also professional and teamwork skills. I participated in the meetings coordinated by the team lead Nancy Leon, in which the entire team contributed towards making my project and all their other assignments a success.

*This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the Minority Student Program and the National Aeronautics and Space Administration.*