ACTIONSCRIPT 3.0 PROGRAMMING FOR HIGH SCHOOL STUDENTS

A Creative Work Report

Submitted as a Partial fulfillment of the requirements for

MASTER OF ARTS DEGREE IN EDUCATION

With concentration in Instructional Technologies

By

JAMES BRIANO

San Francisco, California

December 2010

CERTIFICATE OF APPROVAL

I certify that I have supervised the creative work, "ActionScript 3.0 Programming For High School Students," by James Briano, and that in my opinion it meets the criteria for approving a culminating study submitted in partial fulfillment of requirements for the Master of Arts Degree at San Francisco State University

Dr. Brian Beatty
Department of Instructional Technologies

Dr. Peggy Benton
Department of Instructional Technologies

Approved by COE Graduate Committee

ABSTRACT

ActionScript 3.0 Programming for High School Students

James Briano

San Francisco State University

The purpose of this project is to teach high school students fundamental Object Oriented Programming concepts using Flash ActionScript. A series of eleven tutorials, grouped into three sections, are presented in web page format. The complete body of tutorials should take approximately three hours to complete. Each tutorial includes a skills list, a sample of a completed project, embedded screen shots of the programming development environment, a review of the major concepts and important ActionScript, and a self-check for understanding. Each of the three sections culminates with a special project which challenges learners to apply newly acquired skills. The tutorials were evaluated by current high school Flash animation students, industry experts, and high school teachers. Feedback verified these tutorials are an effective tool for teaching high school students the basics of ActionScript 3.0 programming.

I certify that the Abstract is a correct representation of the content of this creative work.

Chair, Creative Work Committee

Date

TABLE OF CONTENTS

Background Information

Introduction

ActionScript 3.0 is the programming language native to Adobe Flash. Like JavaScript, C++, and Ruby, ActionScript is an Object Oriented Programming (OOP) language. Once learners have mastered the essential concepts of the OOP paradigm, it is a simple task to apply those concepts to any other programming language that follows the same paradigm.

This report outlines the instructional design process utilized to create web-based tutorials on the fundamentals of ActionScript 3.0 programming in the Adobe Flash Integrated Development Environment (IDE). These tutorials are designed to slot into a comprehensive Flash animation and object oriented programming course. Although not a requisite, it is assumed that learners have prior experience with Flash timeline animations.

Background

I am currently enjoying my tenth year teaching computer applications courses at a San Francisco Bay Area public high school. Although curriculum for teaching essential productivity software is readily available, I have been displeased with the offerings for higher-level computer courses such as animation, programming, and web design. Additionally, due to extreme budget cuts there are few resources to acquire expensive training materials or subscription-based training libraries (Swanson, 2010). By combining funds from the unified school district and local Regional Occupation Program, we were able to outfit a classroom with computers loaded with the requisite software. Unfortunately, we weren't provided funds to purchase appropriate instructional materials, and I was left to join the growing ranks of educators who purchase or create their own supplies (Balderas, 2010).

Purpose

The overarching goal of this project is to create an ad-supported, web-based curriculum for teaching the object oriented programming paradigm to high school students. Not only will the tools presented in this report aid me in teaching ActionScript to my own students, but other instructors and self-learners will have access to the same set of proven instructional resources.

Significance

Currently, less than sixty-five percent of K-12 schools offer introductory computer courses, and far fewer provide access to higher-level programming courses (Wilson, Sudol, Stephenson, Stehlik, 2010). Developing applications for the desktop, web, and mobile devices requires a thorough understanding of the OOP paradigm. Unfortunately, there are few free or affordable resources for teaching OOP languages to high school students. Furthermore, much of the available materials are not appropriate for the high school audience (either the scope is too complex, the materials are poorly designed, or the curriculum is not comprehensive). The tools presented as part of this report are intended to fill that resource gap and introduce young learners to the OOP paradigm by presenting a straightforward transition from more traditional tweened animation to ActionScript 3.0 animation. Because ActionScript 3.0 is an ECMAScript compliant language, learners can easily apply ActionScript concepts to JavaScript (which truly *is* ECMAScript) and other popular OOP languages (Yard, Elst, & Jacobs, 2007).

Front End Analysis

Introduction

Prior to embarking on the design and implementation of the instructional content, I conducted an analysis of the learner characteristics, the desired outcome, constraints or barriers to achieving the desired outcome, and available media and distribution options. Data collection took place through a variety of written and verbal interviews with current learners, subject matter experts, and high school instructors.

Learner Analysis

Unique considerations must be made during the learner analysis because the instruction is intended to be used primarily by public high school students. In most public high schools, upper-level students have an opportunity to enroll in elective courses of their own choosing. However, students are not guaranteed a place in their first-choice elective. Because all students *must* be in a classroom, and all classes are constrained by maximum and minimum enrollment numbers, students may actually find themselves enrolled in a course they had no interest in simply because there was nowhere else for them to go. As a result, it is guaranteed that, in a public high school environment, learners in the ActionScript 3.0 programming course will have vastly different skill sets and intrinsic motivation. Additionally, it is not unusual to encounter schedule changes that place students in a class many weeks after the start of the semester, without any explanation to the instructor or student. Learners may be forced to catch up with their contemporaries on their own.

Students entering the course with less than desirable skill sets may create particular disruptions and an unsuccessful learning environment. Students who “get it” and could typically

follow along with the instructors sequential instruction (although ActionScript is an Object Oriented Programming language, the fundamentals are still very much sequential) are left out if the instructor continually breaks the instructional flow to assist students who have missed one or more steps. The key to creating a successful training program for all students is not to lower the learning objectives, but rather to lower the difficulty in achieving the objectives, where possible. Throughout the project development, I found the need to continually remind myself of the intrinsic cognitive load required to learn any programming language. Some learners simply will not come equipped with the skills to master the material. To put it bluntly: Learners who lack basic algebra skills cannot be expected to fully grasp the material.

Content Analysis

A detailed content analysis involved researching texts, online tutorials, and secondary and post-secondary curriculum (A sample post-secondary syllabus can be found in the appendix). The following texts were reviewed for content sequencing and delivery strategies:

- ActionScript 3.0 for Adobe Flash CS4 Professional Classroom in a Book
- AdvancED ActionScript 3.0 Animation
- Designing with Web Standards (3rd Edition)
- Foundation Actionscript 3.0 Animation: Making Things Move!
- HTML5 For Web Designers
- Object-Oriented ActionScript 3.0

The analysis was utilized to guide learning objectives and expectations, content chunking, and lesson pacing. Additionally, the author's "voice" was evaluated for writing that was either too technical or informal.

By the end of the training, learners should be able to abandon all stage-based design tools and create objects with code, modify object properties, and update object properties during runtime. Students will be able to describe other core concepts of Object Oriented Programming, including inheritance and encapsulation. Additionally, this project should include a key "Rite of Passage" for learning a new programming language: The bouncing ball (Webster, Yard, & McSharry, 2007).

After codifying the expected learning outcomes, the content analysis focused on the skills students should come equipped with at the start of the training. Learners can create vector assets with the tool set included in the Flash IDE and animate multiple objects using either timeline-based "Motion Tweens" or "Classic Tweens." Learners can access and manipulate essential MovieClip properties: X and Y position, scale, tint, and alpha. Finally, learners should have received an introduction to controlling the timeline with interactive buttons.

Based on this analysis, it is recommended that instructional materials should ease students into the desired learning outcomes by making incremental connections between what they already know how to do, and the correlating script which actually underlies tool-based production work. Students should follow a sequenced set of tutorials which guide them through building a project which, ultimately, adheres to industry standard programming practices.

Instructional Goals and Objectives

Upon completion of *ActionScript 3.0 Programming for High School Students*, learners will be able to employ industry-standard coding practices to:

1. Add an object to the Stage using AS 3.0
2. Position the object using Flash (x,y) coordinates
3. Declare variables and update at runtime
4. Add multiple instances of a Class to the Stage
5. Create a “mover” function to dynamically update the position of multiple objects

Media Selection and Delivery

Appropriate media selection is crucial to reach successful learning outcomes. Many traditional modes of content creation and delivery where eschewed in order to ensure learners truly master the desired skills. For instance, videos may require students to pause and replay while swapping between the instructional media and their own digital toolkits. Books, either ePub or paperback, do not allow for the display of motion graphics.

The natural choice is therefore to create and deliver content primarily as it is found natively: As Flash.swf files embedded in HTML web pages styled with CSS. The ability to embed motion graphics allows embedded examples to demonstrate, rather than describe, exactly how a project should function at various stages of testing and development.

Code examples are primarily presented as bitmapped “screenshots”, disallowing learners from simply copying and pasting the contents. Learners will develop a requisite attention to detail, troubleshoot their own code, and get into a coding “rhythm”.

Although instructional content may be distributed on CD-ROM or other storage media, the preferred distribution method is online. Content distributed online is easily updated to meet learners’ changing needs. In some cases, the instructional materials may be altered in

real-time to help learners reach the desired outcomes. Additionally, the web facilitates collaboration with an instructor and others who can assist when learners hit inevitable barriers.

Design and Development

Following a thorough front end analysis, the design and development of training tools proceeded smoothly. During this phase the instructional strategy was codified, a content outline was created, and instructional materials were drafted.

Instructional Strategy

Instructional sequencing is designed to ease students toward standards-based programming practices. Learners experience success early, and create a series of increasingly complex programs as they build an understanding of Object-Oriented Programming concepts. By slowly introducing students to coding practices, students will make fewer syntactic or typographic errors. Additionally, it will be easier for learners to spot and correct the inevitable errors on their own. It is important to remember what may seem like an insignificant error to a beginning learner will bring an entire scripting project to a standstill. However, making a few mistakes is an important aspect of the learning process. Therefore, instructional materials will, on occasion, intentionally direct students to make common errors, then teach strategies for locating and correcting those errors.

A student who can follow instructions, complete assigned projects, recite core concepts, and ace multiple-choice assessments may not be prepared to tackle real-world programming challenges. The act of programming is a skill, and can only be truly mastered by performing the skill. I employed an instructional strategy which disallows students the ability to simply copy and paste blocks of code. Instructional materials utilize screen capture techniques to present code as image files, rather than selectable text. By forcing students to hand-code their own projects, they begin to develop their own workflow and troubleshooting skills.

More importantly, by chunking the material learners will focus on, and conceptualize, what each successive block of code is actually doing. Experience has shown that young learners today are prone to lightly scan, or altogether skip, large blocks of text. Breaking code examples into smaller sections allows for more concise explanations, followed by targeted formative assessments.

Development Process

Instructional materials are created with Adobe Flash, Dreamweaver, Fireworks, and Captivate. The development process begins by creating proof-of-concept example files in Flash. Examples are evaluated on levels of complexity and either split or combined, with an effort to ease the difficulty of learning for students. Once a clear pathway to a desired outcome is established, each project is re-created from start to finish and instructional materials are created. Screen-shots are cropped and annotated in Adobe Fireworks. The resulting bitmap images are then embedded in web pages (XHTML 1.0 Transitional) and written instructions are created. Where appropriate, Adobe Captivate is utilized to capture mouse movements and narration. Working Flash .swf files are embedded at key points to give students examples of exactly how their own projects should be functioning at a given point in the lesson. Finally, summative assessments are created to help ensure students have mastered new concepts.

After the instructional materials were created, they were tested with an industry expert, and piloted by students. Students were asked to log the amount of time spent on each project, and a general average was used to determine how long each project should take a typical user to complete. It's important to remember that these materials are meant to be used in the "typical" high school student classroom. A classroom facilitator must work within very finite

class times, and must understand about how long each project should take students to complete. Additionally, students in most public high school environments cannot be expected to have access to the Flash software outside of the classroom. Once pilot data was returned and analyzed, the expected completion time for each project was added to the instructional resources.

Content Outline

Course content is chunked into three main sections: *Setting the Stage for ActionScript*, *Animating a MovieClip with ActionScript*, and *Working with Multiple MovieClips*. Students begin by creating a single asset, and add functionality as they progress through subsequent lesson. The entire unit should be completed sequentially, as advanced projects build off and reference work created in prior lessons. Estimated completion times and a list of core skills are included for each subsection.

ActionScript 3.0 Programming For High School Students **-3 ½ Hours**

- Setting the Stage for ActionScript Programming
  - The Flash Coordinate System – **10 Minutes**
    - The Cartesian Coordinate System
    - Origin Point
    - The Flash Coordinate System
    - Shape x,y positions
    - Object x,y positions
  - MovieClip Instance Names – **10 Minutes**
    - Symbol Names
    - Instance Names
    - Reserved Names
  - Set (x,y) Coordinates with ActionScript – **15 Minutes**
    - The ActionScript Window
    - Compiler Errors
    - Compilers
- Animating a MovieClip with ActionScript

- Add a MovieClip to the Stage with ActionScript – **20 Minutes**
  - Properties
  - Export for ActionScript
  - Class name
  - Classes
  - addChild
- Animate a Single MovieClip with ActionScript – **20 Minutes**
  - Event Listeners
  - Events
  - ENTER_FRAME
  - functions
  - left brace
  - right brace
  - var
- ScreenWrap an Animated MovieClip – **20 Minutes**
  - if statements
  - stage.stageHeight
- Create a Bounce Animation – **20 Minutes**
  - decrement
  - else if statements
  - variable*=-1
- Challenge Project

- Working with Multiple MovieClips
  - Add Multiple Instances of a MovieClip to the Stage – **25 Minutes**
    - Replace All
    - for loop
      - init
      - condition
      - next
    - stage.stageWidth
  - Target Multiple Objects with an Array – **20 Minutes**
    - Array
    - push
    - index
    - init function
  - Wrap Multiple Objects – **20 Minutes**
    - e:Event (event:Event)
    - Math.random

- Bounce Multiple Objects – **25 Minutes**
  - int (integer)
- Challenge Project

Formative Evaluation

Formative evaluations of the completed project materials were conducted by industry experts and typical end-users. Feedback was utilized to check for errors, ensure content adheres to standard practices, and guide revisions. Evaluations were conducted by design and education subject matter experts, and by a group of typical end-users.

Industry Expert Review

I conducted a one-on-one review with a public high school digital media arts instructor. The instructor completed each of the lessons and served as a technical reviewer. Aside from helping edit a few typos, the most important suggestion was the materials include more "multimedia." The instructor wished to hear more narration, or a lecturer's voice guiding him through the projects. Although he appreciated the very concise use of text, he felt it was still too much for "auditory learners" more accustomed to viewing video instruction.

Following his recommendations, I incorporated several Adobe Captivate examples in early projects. My challenge was to find areas where a full motion screen capture was appropriate for teaching computer programming. In the future, it may be beneficial to capture mouse movements highlighting the code, and record verbal explanations.

Pilot Test

Pilot tests of the ActionScript 3.0 Programming curriculum were conducted with three high-school students currently enrolled in a Flash animation course. Students were hand-selected based on their interest in learning the material, and had demonstrated a level of responsibility. Upon completion of the coursework, each student submitted a written response to a questionnaire, their completed Flash files, and an estimate of how long each project took

to complete. (See sample questions in appendix). Feedback was generally positive: “I learned some really interesting stuff, and it was all VERY easy to comprehend.” At least one tester was surprisingly thorough in his evaluation, reporting grammar errors (previously overlooked by teachers, experts and me) and half-a-dozen broken links. A second eagle-eyed tester spotted an error in a formative “self-check for understanding”.

All three students mentioned the first “Challenge Project” was hardly a challenge at all, and deemed it unnecessary. After further discussion, we decided the material was easy enough to comprehend and complete, and any additional project would be superfluous. Therefore, the Challenge Project for Section 1, Setting the Stage for ActionScript Programming, has been removed from the final project.

Summary and Conclusions

Summary

The serious lack of higher-level programming courses in secondary schools is due, in large part, to extreme budget cuts. Equipping and maintaining a computer laboratory with current hardware and software is extremely costly. By creating access to free, age-level appropriate training materials, the overall cost of fielding an object oriented programming course can be lowered.

Conclusions

I am pleased with the outcome of this instructional project. However, the proof is in the pudding, and I have yet to present these tutorials to any high school students other than a hand-selected pilot group. In the future, I would like to include more screen capture videos with narration and detailed verbal explanations. I think the key is to get over my fear of listening to my own recorded voice, and just go for it.

While researching available instructional resources for this project I was again dismayed at how challenging it was to find materials appropriate for the high school audience. With much time and effort, I was able to pick and choose among a multitude of free online resources when looking for solutions to my own programming challenges. I can only imagine the difficulty a young learner, who hasn't developed the skills to locate valid resources, might have. Actually, I do have an idea, based on emails from self-learners who finally reach out to me for help with their personal programming projects: *"Your [sic] a genius! How do you learn all this stuff???" "This is just how I learned from the tutorials I watched online. Do you recommend something different?"*

It took me a very long time to complete this body of tutorials. It's rather disconcerting to know it takes approximately two hours to create fifteen minutes worth of curriculum. I have created this type of instruction resources before, and it would be beneficial to train an assistant to help with various aspects of the design process. My future goal is to apply more of the skills I've acquired as an instructional designer. I plan to continue mastering the skills required to communicate a project outline to an assistant, who goes on to develop the instructional materials.

References

Balderas , K . (2010, October 8). California Teachers Paying for Their Own Supplies and More. [Electronic Version]. *Time Magazine*

Keith, J. (2005). *DOM Scripting: Web Design with JavaScript and the Document Object Model*. Berkeley, CA: friends of ed.

Swanson, R. (2010). *California's Students Shortchanged in Final Budget Deal*. Retrieved December 7, 2010, from http://www.cta.org/About-CTA/News-Room/Press-Releases/2010/10/20101007_2.aspx

Webster, S., Yard, T., McSharry, S. (2007). *Foundation ActionScript 3.0 with Flash CS3 and Flex.* Berkeley, CA: friends of ed.

Wilson, C., Sudol, L. A., Stephenson, C., Stehlik, M. (2010). *Running on Empty: The Failure to Teach K-12 Computer Science in the Digital Age.* Retrieved December 7, 2010, from http://www.acm.org/runningonempty/fullreport.pdf

Yard, T., Elst, P., & Jacobs, S. (2007). *Object-oriented ActionScript 3.0*. Berkeley, CA: friends of ed.

Appendix 1: Textbooks Used During Front End Analysis

Adobe Creative Team. (2008). *ActionScript 3.0 for Adobe Flash CS4 Professional Classroom in a Book*. San Francisco, CA: Adobe Press.

Peters, K. (2008). *AdvancED ActionScript 3.0*. Berkeley, CA: friends of ED.

Peters, K. (2007). *Foundation Actionscript 3.0 Animation: Making Things Move!* Berkeley, CA: friends of ED.

Keith, J. (2010). *HTML5 For Web Designers*. New York, NY: A Book Apart.

Appendix 2: Web Sites Used During Front End Analysis

ActionScript.org: http://www.actionscript.org/

Adobe K-12 Career and Technical Education:

http://www.adobe.com/education/solutions/k12/careerteched/

A List apart, for people who make websites: http://www.alistapart.com/

Flash ActionScript, Multimedia 279: http://www.csm.flashclass.org/?page_id=473

Flashkit, A Flash Developer Resource Site: http://flashkit.com/

Intro to ActionScript 3.0: http://sccottt.com/teaching/svc/2009-autumn/intro-to-

actionscript/downloads/syllabus.pdf

Interactive Scripting – CGR C/092:

http://people.artcenter.edu/~vanallen/scripting/2004fa/index.html

Lynda.com: http://www.lynda.com/

Programming Foundations for Digital Media, DGM6108:

https://www.spcs.northeastern.edu/files/syllabi/20112520203.pdf

intro to **ActionScript 3.0**

School of Visual Concepts
Wednesdays, 11/4 - 12/2
6:30 - 9:30pm

Scott Thiessen, instructor
scott@sccottt.com
sccottt.com/intro-to-actionscript

syllabus

Course Description

Intro to ActionScript 3.0 is a five week course designed for people who have some experience using Flash and who want to use ActionScript to create more interactive, more dynamic, and more interesting Flash work. Students will gain a strong foundation in the latest version of ActionScript and many more general object-oriented programming principles that they will be able to take and run with. This course assumes no prior programming experience.

Course Objectives

- Become comfortable writing your own ActionScript code
- Gain firm understanding of core ActionScript concepts and syntax
- Understand basic object-oriented principles (objects, methods, properties)
- Learn to effectively use the ActionScript documentation
- Become equipped to eagerly dive deeper into ActionScript on your own

Recommended Reading / Materials

*ActionScript 3.0 Visual Quick Start Guide ***,*** by Derrick Ypenburg
Essential ActionScript 3.0, by Colin Moock
Learning ActionScript 3.0, by Rich Shupe and Zevan Rosser
Notepad & pen for notes
USB flash drive to take work with you

Course Schedule

Week 1
Nov 4

AS3 vs AS2; Functions, methods, and parameters; Comments; Communicating with symbol instances; Instance properties and methods; Dynamic and input text; custom functions; Variables; Data types

Week 2
Nov 11

Events; Button event handling; Object-oriented programming concepts; Class based ActionScript
Project: Building a simple mini-site
In the Book: Chapters 4, 7

Week 3
Nov 18

Conditional statements; Arrays; Objects; Looping; Math
Project: Creating a quiz game
In the Book: Chapters 11, 12, 13, 14

Week 4
Nov 25

Advanced OOP concepts; Working with Display objects; Other classes and libraries
Project: Enhancing mini-site / quiz
In the Book: Chapter 5, 18

Week 5
Dec 2

Loading symbols from the Library; Loading external images and swfs; Creating a preloader using Events
Project: Creating a dynamic slideshow
In the Book: Chapters 15, 16

Appendix 4: Sample Questions for Pilot Students

1. What grade are you in?
    a. Frosh
    b. Sophomore
    c. Junior
    d. Senior
2. Describe your interest in computer programming.
    a. I want to learn it soooooo bad!
    b. No offense, but I'm not interested.
    c. I guess it might be cool to learn some.
3. Describe your feelings about the overall design of the web site.
    a. The site is too confusing!
    b. It's simple and easy to follow.
4. How difficult were the projects?
    a. Too easy! I did stuff, but never learned anything.
    b. Too hard! I did stuff, but never learned anything.
    c. Pretty good. I think I learned a bit about programming.

Appendix 5: Sample Questions for Industry Expert

1. How did you find the overall design of the web page tutorials?
    a. Fairly logical, and easy to follow
    b. I was rather confused at times, but figured it out.
    c. You might want to start over
2. What are your thoughts on the level of difficulty the information offered?
    a. It wasn't too difficult.
    b. It was a crazy amount of confusing information
    c. It was a bit too watered down and simplified to be considered "learning".
3. Did the information conform to industry coding standards?
    a. Very much so!
    b. Yes, for the most part
    c. No, not really
    d. What standards?
4. Did YOU learning anything while evaluating the tutorials?
    a. Actually, I did!
    b. Nothing I didn't already know

Appendix 6: Sample Questions for One-on-One Review

1. Do you think these projects are appropriate for a high school student?
2. Did you have any trouble navigating the web site?
3. Do you think the tutorials would help a high school student learn the basics of computer programming?
4. What could be done to help lower the difficulty some students will have with this material?