Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 224 (2009) 133–140
www.elsevier.com/locate/entcs

Animation and Interactive Programming: A Practical Approach

Phillip Benachour[1] and Reuben Edwards [2]

Department of Communication Systems
Infolab21, Lancaster University, UK

Abstract

This paper describes a work in progress of using animation software tools to teach programming principles. The motivation behind this work is to encourage students in higher education who do not see themselves as serious programmers to engage with some of the concepts and methods used in the teaching of programming. In addition this work was used in workshops to engage with local and regional secondary schools focussing on the use of animation as a tool for learning programming principles. The results presented in this paper are based on feedback from first year undergraduate students. Initial feedback from teachers, pupils and schools has been very positive and requests for additional visits have been made. This has created an opportunity to further engage with these schools for additional workshops. Analysis and feedback from these schools will be presented in the workshop.

Keywords: ActionScript, interaction programming, hands-on learning

1 Introduction

The teaching of new concepts and hands on practical skills such us programming can be combined with animation and visuals to great effect. The use of learning objects to deliver programming skills to multimedia students and other disciplines has been explored and implemented [1]. Modern programming languages taught at university level is generally seen as a major obstacle for new undergraduates and such a perception of difficulty is commonly cited as a reason for disengagement with the subject as a whole [3]. A number of papers in the literature have introduced Actionscript as a suitable tool for teaching introductory programming. For instance, citeCrBo:2006:ActionScript compared Actionscript code with more complicated code in Java for similar programs and concluded that Actionscript not only teaches the fundamental of programming and concept of object-oriented development to

[1] Email: p.benachour@lancaster.ac.uk
[2] Email: r.edwards@lancaster.ac.uk

doi:10.1016/j.entcs.2008.12.057

the students but also enables them to find the errors in the smaller tasks which is easier to solve. In a more recent paper [2], Actionscript has been recommended as a useful step up to high-level languages such as C++. Actionscript is seen to be easier to learn due to the immediate visualization it provides. In addition citeLeEd:2007:GamesFirst argued that Actionscript is widely used in the real world e.g for designing animations and 2 dimensional games.

The aim of this paper is to describe and evaluate a work in progress on using animation and interactive programming to engage first year university students, from diverse educational backgrounds and subject disciplines, with programming principles. This work in progress focuses on encouraging students to learn progressive programming tasks from simple commands and assignments to designing a game. The design of a BreakOut computer game is perhaps one of the most useful exercises the students get involved in when learning programming and scripting concepts. Students learn about collision detection between the ball, paddle, bricks and boarders, user interaction by moving the paddle using keyboard controls, update of scores using dynamic text and the use of sound to make the learning of programming experience more enjoyable and engaging.

2 Using Actionscript for Animation and Scripting

Actionscript uses traditional coding techniques but allows the user to see how each piece of code effects the running or execution of the program, allowing the user to have an instant visual understanding of what the code is doing. To help with coding errors Actionscript uses a syntax checker and will inform the user of errors either before or as they run a program. This is in contrast to other programming or scripting languages such as Javascript where errors are not so easily identifiable and the simple omission of a comma can cause the entire program to fail. A Javascript program will also only execute when the whole program is complete, where as with Actionscript code written for a specific action can be run even if the program, as a whole, is unfinished.

2.1 Teaching the use of Increment and Decrement Operators

Perhaps the most obvious and easiest example to work with when beginning to use Actionscript for animation, is to use the increment and decrement operators. Students are encouraged to experiment with moving objects in a two-dimensional and three-dimensional world. The following simple script allows an object to move in the x direction with incrementing values of x by one pixel every time the frame is rendered and displayed.

```
onClipEvent (load) {
  this._x = 50;    // sets the initial x position of our clip
}
```

Now try the following code but only for the X-axis:

```
onClipEvent (enterFrame) {
  this._x = this._x+1;  // this moves our clip 5 pixels to
    //  the right every frame
}
```

Students are then asked to record their answers on the following questions:

- What value would you assign "this._x" to in order to centre it in the middle of the stage?
- What would you change in the code above to stop your circle from moving?
- What would you change in the code above so that your ball moves in the opposite direction?

The students are then asked to introduce a "y" variable to the Actionscript code and give their movieclip the following action:

```
onClipEvent (load) {
  this._x = 50;
  this._y = 50;
}
onClipEvent (enterFrame) {
  this._x+=1;
  this._y+=1;
}
```

The routine above enables students to understand how two-dimensional animation works using the following questions as a guide:

- What happens now and why?
- What would the values of "this._x" and "this._y" be set to put your circle in the centre of your movie clip?
- What sort of increments/decrements would you set "this._x" and "this._y" to in order to get the ball object to travel in the four possible directions from the centre.

2.2 If and If ... Else Statements

Using if and if ... else statements for conditional testing checks whether a condition is true or false. The ball object in the example above can be used to do this. Students are asked to think of ways to stop the ball object moving when it gets to a certain pixel on the screen. A routine like the one below can achieve this task very easily:

```
onClipEvent (enterFrame) {
if (this._x<300) {
    this._x += 1;
  }
}
```

Once students get the idea that they can modify the code to make the ball object move upwards and downwards once the value of "this._x" reaches 300 pixels, they are asked to experiment with different values of the x co-ordinate and then the y co-ordinate. An alternative way to assess students understanding is to invite them to comment on a piece of code already written such as the one below:

```
onClipEvent (enterFrame) {
  if (this._x<300) {
    this._x += 1;
  } else if (this._x>300) {
    this._y += 1;
  }
}
```

Using if and if ... else statements can be used more imaginatively by testing whether an object is moving within a defined space or by testing for collision detection. The design of a breakout game is an example in case where the ball is required to bounce off the edges or a paddle. Students are initially given code which does not take into account the variable ballRadius, once the type the code and see it working they realise that the ball object goes over the edges set. In order to solve this problem ballRadius is included as part of the if statements for the x and y directions as shown in the code below:

```
var screenWidth = 550;
var screenHeight = 400;
var ballRadius = 10;
ball._x = ball._x + ballSpeedX;
ball._y = ball._y + ballSpeedY;

if (ball._x<ballRadius) {
  ball._x= ballRadius;
  ballSpeedX*=-1;
} else if (ball._x>screenWidth - ballRadius) {
  ball._x = screenWidth - ballRadius;
  ballSpeedX*=-1;
}

if (ball._y<ballRadius) {
  ball._y= ballRadius;
  ballSpeedY*=-1;
} else if (ball._y>screenHeight - ballRadius) {
  ball._y = screenHeight - ballRadius;
  ballSpeedY*=-1;
}
```

2.3 Using Functions and loops for layering of bricks in a breakout game

A function is defined using the function keyword, followed by the name of the function, and then a list of parameter names within the following brackets. In the example code below the function laybricks is passed three parameters which provides the information that allows the layering of bricks.

```
function layBricks(numWide, name, row) {
  eval(name)._width = screenWidth/numWide;
  eval(name)._height = 0.25*screenWidth/numWide;
}
```

In this function, a loop is created that lays numWide bricks:

```
function layBricks(numWide, name, row) {
  eval(name)._width = screenWidth/numWide;
  eval(name)._height = 0.25*screenWidth/numWide;

  for (i=0; i<numWide; i++) {
  }
}
```

This loop, begins by setting a variable called i, to 0, and then loops numWide times, and creates a new name, based on the original brick name, so that we have a unique variable name for each of the new bricks we are going to create. Figure 1 below shows output produced once the game is completed.

```
nextLevel = 100;
numBricks = 0;

function layBricks(numWide, name, row) {
  eval(name)._width = screenWidth/numWide;
  eval(name)._height = 0.25*screenWidth/numWide;

  for (i = 0; i < numWide; i++) {
    brickName = name + i;          // trace(brickName);
    duplicateMovieClip (name, brickName, nextLevel++);
    eval(brickName)._y = 100 + row*eval(name)._height;
    eval(brickName)._x = (i + 0.5)*eval(name)._width;
    numBricks++;
  }
}

layBricks(10, "yellowBrick", 0);
layBricks(10, "orangeBrick", 1);
layBricks(10, "blueBrick", 2);
layBricks(10, "greenBrick", 3);
```

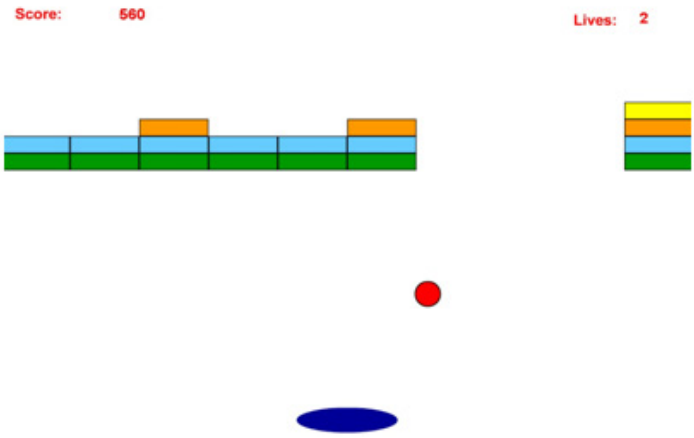

Fig. 1. BreakOut Game

3 Evaluation and Student Feedback

The first year course in Information and Communications Technology is common to all first year undergraduates studying for a communications degree. The course is taught taking into consideration the needs of both potentially highly numerate and technologically aware students as well as students who may have no previous experience of ICT. The first year intake has between 60-70 students, most studying for programmes on Information Technology, Media and Computer Communications. The aim of the survey below is to carry out a quantitative and qualitative evaluation of how effective Actionscript has been in helping students engage with programming and scripting principles, the data and feedback collected was intended to focus on the following three areas:

- Previous experiences of using programming and scripting
- Experience of computer animation, game design, and game play
- Evaluation of Actionscript as a tool to learn programming principles and for application development

From the cohort of students studying the course, 67% said they had some previous experience of programming and 47% with scripting. When asked if they found some programming concepts easy to implement, 82% said that they felt that learning to program without visualisation was hard and slowed their progress. A majority said that they decided to carry on with their courses at school/college because it was too late to change. Many felt that they found difficulty in acquiring fluency in programming.

When considering the distribution of the type of programming languages used, 10%, 16% and 26% said they have had experience of C, C++ and Java respectively. It is also worth noting here that the remaining 48% who have not used C/C++, Java have used other high-level languages such as Pascal, Prolog, C#, VB.NET, and Delphi. We can summarise from this that two-thirds of first year students have some experience of programming using a high-level language at a basic level

which is a positive outcome in as far as recruiting students with programming skills. However, this experience was proved to be very basic and engagement was an issue as pointed out in the previous paragraph. Of the 47% of students who said they have scripting experience, 30% said they had used Actionscript before but only at an introductory level (using a button for controlling the start of an animation), 35% said they have used Javascript. The remaining 35% have used VBScript 18%, PHP 12%, ASP 5%. Figure 1 shows the distribution of programming and scripting previously used at school/college by first year students.

All student (100%) of this group found Actionscript an easier tool to visualize and understand basic programming and scripting.

For students who had no previous experience of programming and scripting languages 33% and 53% respectively, engaging them in learning programming principles was rather easier than anticipated. The students had no prior experiences and there was no bias towards a particular programming or scripting language. It was also noted that for this group, the majority tended to have had more experience of using video and audio editing tools with some good experience of Web design and image editing. 80% of this group found Actionscript a useful tool to understand basic programming and scripting.

In order to find out how effective Actionscript has been in engaging and helping student to learn programming principles, success rates of the questions put to the students during the practical sessions were measured. The results of these are shown in Table 1.

4 Conclusions and Further Research

The results attained from the practical assessments has shown that correct responses have progressively improved over time. Although this was somewhat expected it also demonstrated student commitment and engagement as well. The challenges presented when designing the breakout computer game is an example of how well the students have adapted to using Actionscript.

References

[1] Jones, R., *Designing Adaptable Learning Resources with Learning Object Patterns*, Journal of Digital Information **6** (2004).

[2] Leutenegger, S. and J. Edgington, *A Games First Approach to Teaching Introductory Programming*, in: *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education* (2007), pp. 115–118.

[3] McCracken, M., V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. Kolikant, C. Laxer, L. Thomas, I. Utting and T. Wilusz, *A multi-national, multi-institutional study of assessment of programming skills of first-year cs students*, SIGCSE Bull. **33** (2001), pp. 125–180.

Questions used	**% correct**
What value would you assign "this._x" to in order to centre it in the middle of the stage?	53%
What would you change in the code above to stop your circle from moving?	46%
What would you change in the code above so that your ball moves in the opposite direction?	58%
What would the values of "this._x" and "this._y" be set to put your circle in the centre of your movie clip?	65%
What sort of increments/decrements would you set "this._x" and "this._y" to in order to get the ball object to travel in the four possible directions from the centre.	60%
Use if and if ... else statements to control the movement of the ball when it reaches a position on the x-axis	51%
Use if and if ... else statements to control the movement of the ball when it reaches a position on the x-axis and y-axis	60%
Write the code you would add to make the ball bounce from the bottom and top wall (y-axis)	71.4%
Write the code you would use to modify the gameloop code to take into account the height of the ball	85.7%
Reflecting the ball off the paddle and back up the screen	100%
Explain how you would add a 5th layer of bricks -with a score of 50 points- to the screen. What additional code would you use?	93%
What code would you add to generate sound for your game?	90%
Recall the different approaches of programming in Javascript and Actionscript. Which programming style do you prefer?	90% said Actionscript

Table 1
Student success rate in the practical sessions