

Developing Understanding of Programming Principles using Flash Actionscript

Graham Routledge, Amin Aminaei, Phillip Benachour

Department of Communication Systems
Infolab21, Lancaster University, UK
g.routledge@lancaster.ac.uk, a.aminaei@lancaster.ac.uk,
p.benachour@lancaster.ac.uk

Abstract: Over the last few years, the IT industry has witnessed a growth -driven by the success of the Internet- in the requirement of skills based on the emergence of the technology and the creative media industries. This has provided a unique niche market where, a combination of programming and artistic knowledge, is a requirement. As a result, many of the courses focusing on multimedia and information technology need students to develop programming and scripting skills to develop Web applications, databases, 3D modelling of objects, animation and games. For students who do not see themselves as serious programmers, some of the concepts and methods used in the teaching of programming can be difficult to grasp. It is important as a result to find ways of engaging students in activities where programming is seen as a tool to demonstrate other concepts and ideas visually which can in turn help in the teaching of programming principles. In this paper, the authors describe a number of activities used to engage multimedia students in the learning of scripting and programming. By the use of animation, mixed reality gaming, computer game design and research seminars, first year students are able to grasp the fundamentals of computer programming and combine it with their creative artistic side to produce digital multimedia content.

Keywords: Teaching Animation and Programming, Hands-On Learning, Engaging students in the learning of programming principles.

1 Introduction

For many students who have an interest in combining their creative artistic side and the use of digital software technologies, the need to gain skills in programming and scripting is growing. Over the last few years the convergence of technology and art has become a reality. Scripting languages have been supported in many animation, 3D modelling and virtual studio design software packages including Adobe Flash, Alias Maya and Adobe After Effects.

The teaching of new concepts and hands on practical skills such as 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 (Jones, 2004; Spaul, 2006). For students who require scripting and programming skills to develop multimedia applications but do not see themselves as serious programmers, some of the concepts and methods used when writing programming code can seem very abstract and difficult to understand. Modern programming languages taught at university level are 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 (McCracken et al 2001). Visual programming has been explored as a method to engage students with programming focussing on the Alice programming tool (Cooper et al, 2000; Cooper et al, 2003). McDermott et al (2007) argued that students can face difficulties acquiring programming skills when they enter a course of study with little confidence in their own ability to use symbolic reasoning.

For many students entering higher education to study information and media technology with no or little programming experience, it is not always obvious what a 'function' is or how an 'if', or 'else if' statement or an 'array' works and where it should be used. In general, basic programming should not be too difficult for beginners, but concepts can be difficult to visualise for some students, while for others the written example code alone is enough to build a mental image of what is happening within the program and this often leads to varying learning rates for different students. This can have an effect on many academic disciplines not just multimedia software development. The following example illustrates the case where the same physical event is described both textually and pictorially. The textual description is as follows: "As the hammer is thrown upwards an observer at a fixed point in space observes the hammer moving relative to them. The movement of the hammer is dynamic and has a rotational component around the hammer's centre of mass. The hammer moves with this rotational component along a parabolic path under the influence of gravity." The pictorial description is shown in Figure 1 below.

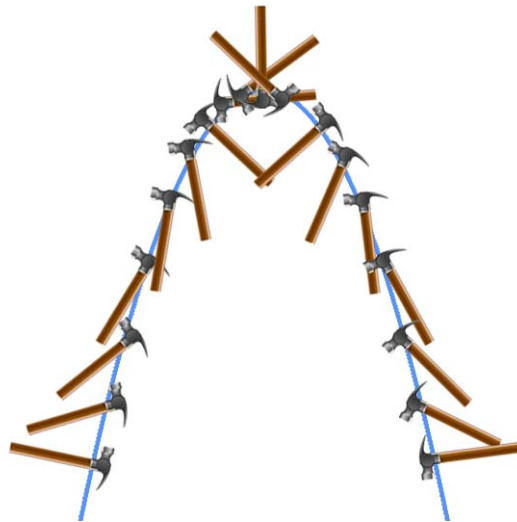


Figure 1: Diagram of the dynamic motion of a hammer

The coloured line represents the parabolic curve as the hammer falls under the influence of gravity. Diagram and text after Ohanian, *Physics*, 2nd Edition, W.W. Norton & Company Inc. 1989. Chapter 12 – Motion of a rigid body, p 297

The textual description may be clear enough and understandable to a few but with the addition of a picture or diagram, as shown in Figure 1 above, the description is reinforced and made much clearer for many more students who were unable to mentally visualise the motion of the object. It is now possible for software users to create professional looking 2-Dimensional and interactive computer animations, particularly for the internet. Adobe Flash and Actionscript is one such product that allows the user to create interactive animations in a straight forward and easy to use environment.

Flash 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, in the same way as the image in Figure 1 is an additional aid to the understanding of the descriptive text. 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. A number of papers in the literature have introduced Actionscript as a suitable tool for teaching introductory programming. For instance, (Crawford and Boese, 2006) 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 the students but also enables them to find the errors in the smaller tasks which are easier to solve. In a more recent paper (Leutenegger et al, 2007), 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 Leutenegger et al (2007) argued that Actionscript is widely used in the real world e.g for designing animations and 2 dimensional games.

2 Using Actionscript for Animation and Scripting

Adobe Flash is a highly flexible animation and programming environment that can be used to create animations, cartoons, web pages, computer games, business applications either through its drawing and animation tools or using its built-in scripting language, Actionscript. Since Flash provides an integrated drawing and programming environment it is ideally suited for creating prototypes of applications that are going to be implemented in other languages/environments. For example, before creating an application for a mobile phone, a prototype can be created and built on the desktop in Flash. Actionscript is derived from the Javascript programming language, although it has grown considerably since the early versions of the application. Nowadays it is possible to create standalone executable applications in Flash8, e.g. databases for business applications, streaming and animation.

First year students are introduced to a Flash environment that presents a work area and stage for the animations, a frame by frame timeline that can contain numerous content layers, toolboxes and effects panels. Objects can be placed on the stage at any particular frame in the time line in order to create a sequence of events. This coupled with the ability to use Actionscript programming to control or manipulate the objects allows this to be a very powerful animation tool.

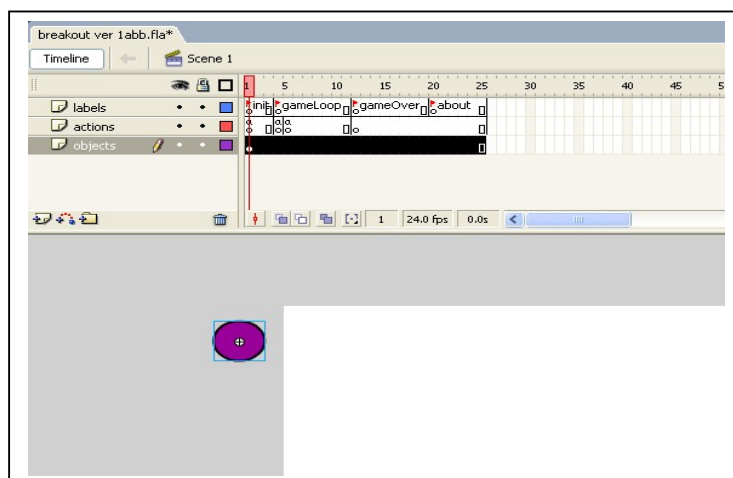


Figure 2: view of an object created on the stage in Adobe Flash

Initially students find their way around the software by creating simple objects and placing them in the work area or on the stage. Some of these objects may be converted to *graphics*, *movieclips* or *buttons* to be used later in the animation. Their first task is to create simple layered animation using the timeline, of a bouncing ball against a static background. The timeline organizes and controls a Flash document's content over time in layers and frames. Layers are like multiple film strips stacked on top of one another, with each layer containing a different image or object that appears on the Stage. Students progress onto creating further layers with animated objects in order to create a realistic animated scene.

Next the *button* symbol is used to control and play the animation. The *button* symbol is interactive and can be used to work with effects such as roll-over and mouse-click animations. Here a simple routine using ActionScript is introduced linking an *instance name*: `play_btn` to the *button* symbol and the function `play()`. Before they add the code, the students need to give the button a unique instance name –in this case: `play_btn`. The instance name enables them to target the *button* symbol with ActionScript code. If they don't name the *button*, their code doesn't have a way of targeting the *button* from the timeline.

```
stop();
play_btn.onRelease = function() {
play();
};
```

The script above allows a user to play an animation when the *button* is clicked. Initially the animation will not play since the first line in the code is `stop()`. Once the button is clicked and the mouse is released the code `play_btn.onRelease` executes and the function `play()` will run. This is the first example of Actionscript that the students get to use.

2.1 Handling Events

The students are encouraged to notice how to interpret the `play_btn` instance in the second line of code. The `onRelease` event in the ActionScript code which refers to the action when a user clicks and then releases the mouse over the `play_btn` instance is also discussed and explained:

```
play_btn.onRelease = function() {
```

Events are actions that occur while a Flash output file (SWF) is running or playing. An event such as a mouse click or a keypress is called a **user event** because it occurs as a result of direct user interaction. An event that a Flash output file generates automatically, such as the initial appearance of a movie clip on the stage (`onClipEvent`), is known as a **system event**

because it isn't generated directly by the user. In order for an application to react to events, a user must use *event handlers*. *Event handlers* is ActionScript code associated with a particular object and event.

Students are encouraged to do some research on the two types of events that Adobe Flash supports and make the distinction between them. These being:

- User events, which are actions taken by the user (e.g., a mouse click or a keystroke)
- System events are events that happen as part of the internal playback of a movie (e.g., a movie clip appearing on stage).

And that the Actionscript objects that can receive events are:

- Movie Clips
- Buttons
- Objects

From this point, students have a basic understanding of how the timeline works, how to animate an object and how to use Actionscript to control animation by using the symbols, instances, and an event handler.

2.2 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. This simple technique allows an movie clip like a ball to move in the x and y direction of the stage. The use of dynamic animation speeds up the process as the animation occurs in one frame and follows certain rules or in this case the script. Engaging students in learning and developing programming skills involves capturing their imagination and their interest in digital animation. 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;           // this 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.

2.3 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. The following routine initially tests whether the ball object has reached the screen width and height and will reverse direction when the 'if' test returns a true value:

```
var screenWidth = 550;  
var screenHeight = 400;  
var ballRadius = 10;  
  
ball._x = ball._x + ballSpeedX;  
ball._y = ball._y + ballSpeedY;  
  
if (ball._x<0)  
{  
    ball._x=0;  
    ballSpeedX*=-1;  
}  
else if (ball._x>screenWidth)  
{  
    ball._x = screenWidth;  
    ballSpeedX*=-1;  
}  
  
if (ball._y<0)  
{  
    ball._y=0;  
    ballSpeedY*=-1;  
}  
else if (ball._y>screenHeight)  
{  
    ball._y = screenHeight;  
    ballSpeedY*=-1;  
}
```

Students are initially given the code above which does not take into account the variable `ballRadius`, once they 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:

```

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;
}

```

2.4 Using 'for loops' within a function for grid creation in tile-based games

Many games tend to have a large map (an area that defines the world of the game). Most Tile-based worlds in Flash are going to be in either top-down view (Pac-Man) or 3D isometric view. The way tile data is stored and manipulated is exactly the same for both of those views, but the way the tiles are displayed on the screen is not (Makar, 2002). The use of 'for loops' can be used to help create tiles in the top-down view and store information about those tiles. For the example used here, students are required to use nested loops within a function.

```

function buildGrid() {
    for (var j=1; j<=game.rows; ++j) {
        for (var i=1; i<=game.columns; ++i) {
            var name = "cell"+i+"_"+j;
            var x = (i-1)*game.spacing;
            var y = (j-1)*game.spacing;
            game.path.attachMovie("cell", name, ++game.depth);
            game.path[name]._x = x;
            game.path[name]._y = y;
            game[name] = {x:i, y:j, name:name, type:type, clip:game.path[name]};
        }
    }
}

buildGrid();

```

This function above uses nested loops. The **outer loop** loops through the number of rows. In each iteration of the **outer loop**, the **inner loop** loops through for each column. Each tile (which we call a *cell* here) is named uniquely by using the **row** and **column** of the cell as part

of that cell's name. For instance, if the cell belongs to column 8 and row 6, the name would be cell8 _ 6 and the intended position of the new movie clip is calculated. Then a variable called **type** is created with a value of 1. This refers to the frame that the **tile** will display. Next, the movie clip is created and positioned. An object is also created to store information about the cell that was just created, such as its type, its name, and a reference to the movie clip it represents. The final line of Action in the script above is **buildGrid ()**. This line calls the function that was just examined to create the grid.

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

Function calling and the passing of parameters is an important part of programming fundamentals not only because they are efficient at executing code within the function itself but also because they can be called a number of times from a program. 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:

```

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);

```

3 Learning and Teaching Outcomes

3.1 Designing a BreakOut 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 whole experience more enjoyable and entertaining. The breakout game includes all of the elements required for teaching the fundamentals of design and programming discussed in section 2.

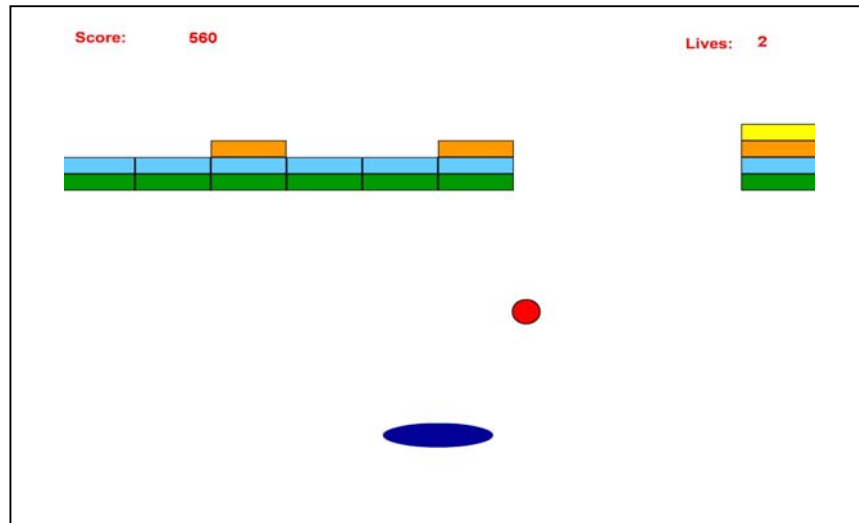


Figure 3: BreakOut Game

By now students have practiced a number of the techniques needed for this game. The BreakOut game has a series of layered rectangular bricks that span the width of the screen as shown in Figure 3 above. Each layer of bricks has a different colour and value for a points system. The object of the game is to keep the ball in the air by means of a horizontally moving paddle knocking out as many bricks from above as possible. If the ball passes below the paddle because the player did not hit the ball in time a life is lost from the original three lives. When all three lives are lost a *Game Over* screen appears and if the players points tally is equal to the total values of all the bricks (ie. All the bricks have been knocked off the screen) a *You Win* screen appears. For both the *Game Over* and *You Win* screens a *Game Reset* button allows the player to play the game once more.

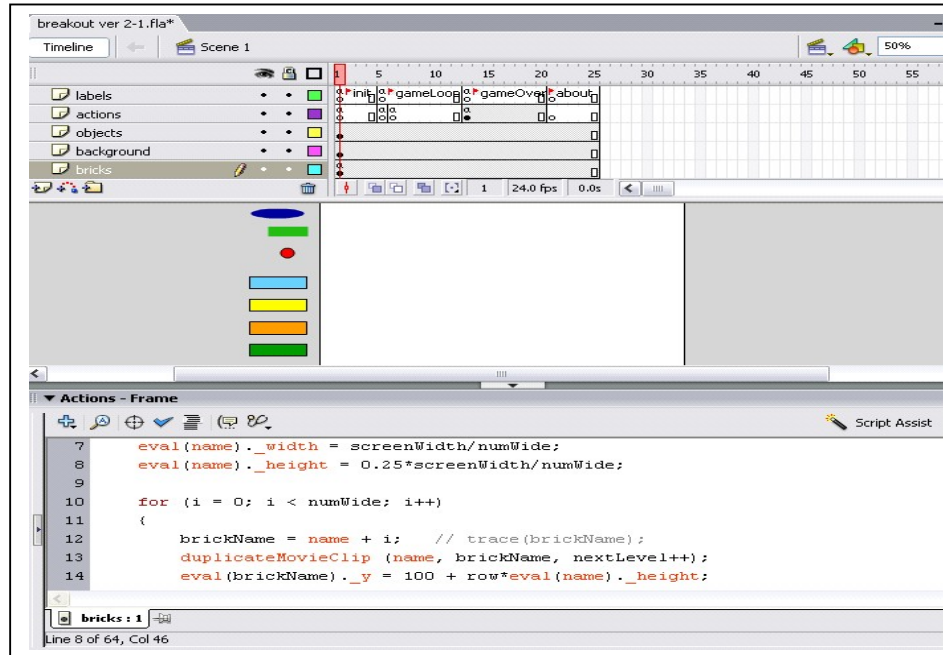


Figure 4: BreakOut Game in Adobe Flash IDE

There are four main outcomes to this exercise:

- Declare global variable specifying the speed and acceleration of the ball object
- Use if and if ... else statements to control the movement of the ball within a specified border, collision detection, and interaction with the paddle
- Use of functions to run the Bricks layering routine
- Use for loops to build the bricks in rows with different colours

Students are grouped in pairs and the game must be completed over three two-hour sessions of laboratory time. Students will get help and support but only if they attend the lab sessions. This encourages attendance and makes efficient use of the time available. Each group is required to demonstrate their work to the class and discuss the methods that they used to design and program the game. Students are allowed to discuss and share ideas with other groups but must not copy and paste code from other sources and must be able to describe how their own code works and how they went about writing their scripts. The emphasis is on understanding how a particular routine works and being able to describe code functionality and its purpose.

3.2 Designing a Tile Based Game

The use of practice based methods such as social and mixed reality gaming as an aid to design a tile-based PAC-MAN computer game was explored and implemented. The mixed reality game was an application developed by a researcher in the communication systems department by

the mobile research group (Rashid et al, 2006a; Rashid et al, 2006b). The application here requires participants to use mobile phones with radio frequency identification tags (RFID) to navigate a maze in the form of the PAC-MAN computer game. Groups of five students participated as the players in the game and another five other students were involved with filming the game. RFID tagging is a technology that is being used in many applications today ranging from stock monitoring in supermarkets to the tagging of goods when in transit. There are also social implications when using this technology which are centred around the privacy of individuals. Course work and research assessment for undergraduate students focussed on the invasion of consumer privacy when it was suggested by industry that the RFID tags can be imbedded in consumer devices (Benachour et al, 2007). This in turn would allow some goods manufacturers to track consumers and collect information on their shopping habits and their movements which some argued that such use of this technology invades human privacy. The five members who took part in the game were required to do further research of the technology and design a computer game based on PAC-MAN like game. The five students who were involved in filming the game were required to produce a short video of about two minutes. Each group –there were six groups in total- compared their work in order to share their experiences and exchange further ideas.

The groups involved in the game design were required to develop a two-dimensional grid where a character can be controlled by the arrow keys and find its way around the maze. The walls of the maze can be re-designed using an XML file that is read every time the game is started. The work presented was assessed as part of their summer term assignment. Students taking part said that they felt they understood more about the technology when they were taking part and designing the game. This allowed them to think of other gaming ideas and applications involving RFID technology. The students who were involved in filming the game used the skills and experience they have developed previously to produce more professionally looking videos. The requirements for the tile-based game were as follows:

- Tile creation and Management
- Creating the Grid and Storing Information
- X-Y Position character Detection
- Designing the maze wall and creating the XML file
- Adding sound with collision

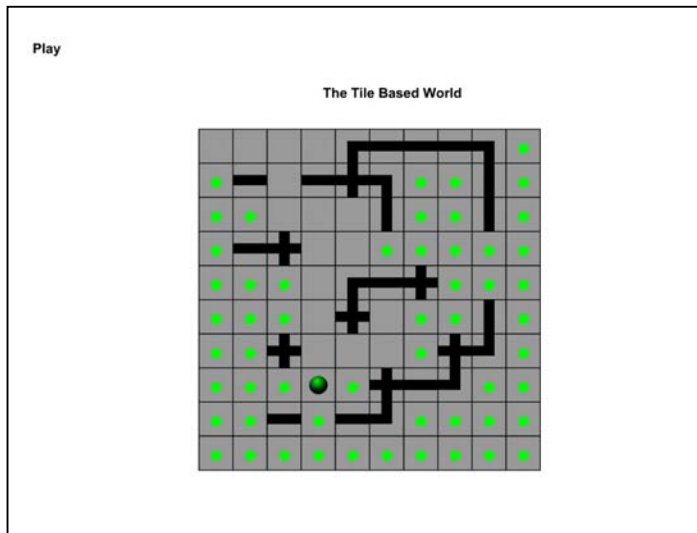


Figure 5: Tile-based Game in Adobe Flash IDE

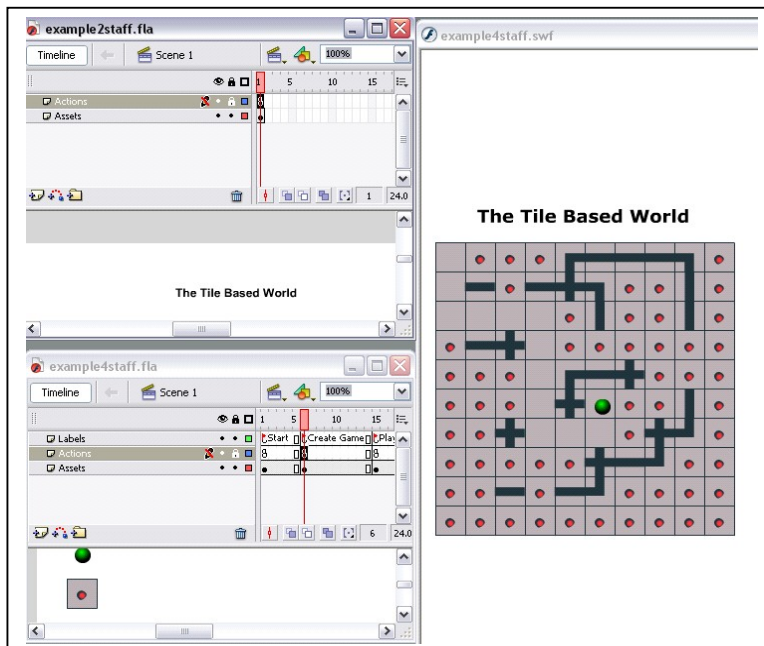


Figure 6: Tile-based Game in Adobe Flash IDE

Each group was required to demonstrate their game to the class and gave a presentation on the results achieved. For assessment purposes, each group was asked to demonstrate their game design ideas and how they went about building the game prototype in Adobe Flash. Each group was required to demonstrate their understanding of the overall functionality of the code they developed with less emphasis on specific routines. It was assumed at this stage that they

have demonstrate knowledge of programming fundamentals from the BreakOut game they previously designed.

4 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 course aims to introduce first year students to information technology from a theoretical and practical viewpoint and covers a number of areas relating to communication technology, digital telecommunications, the creative use of media technology such as video, sound editing, animation and computer game programming. It can also be taken as an elective course by first year students enrolled on any other degree discipline from the Arts, Music, Sciences and Media. Unique challenges are presented in the teaching of this course due to the diversity of the students participating as they do not have to be enrolled on a communication systems degree program. Assessment is based on exams and coursework weighted at 60% and 40% respectively.

When carrying 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. An important point to make here is that previous experience in programming refers to limited – and not extensive use- at school/college. Most students felt that they had a basic introduction to programming and were confident in writing very simple programs. More importantly, they felt that they needed more practice and time to consider themselves proficient programmers. 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 7 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.

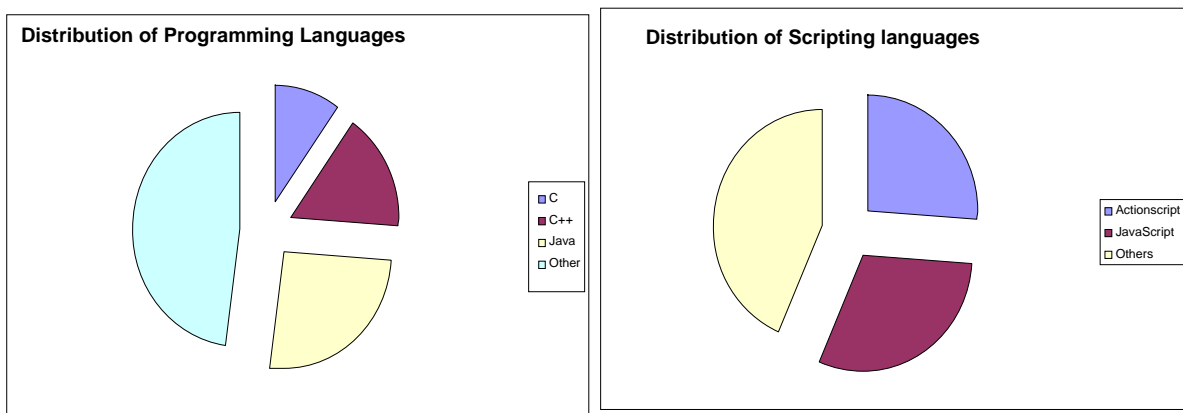


Figure 7: Distribution of Programming/Scripting

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.*

It was also felt important to find out how many students -with and without previous programming experience- had previously played computer games, used computer animation

and game design packages (shown in Table 1). This would give an indication of the group's overall experience of using animation tools and of their interest in computer games.

	Never Used	Used Occasionally	Used Frequently
Playing computer games	13%	47%	40%
Computer animation	73%	20%	7%
Computer game design	86%	7%	7%

Table 1: Students' previous experience

It was found that 87% were occasional –once weekly- to frequent –once daily- computer game players. That is 47% played games occasionally and 40% frequently. This suggests that a big majority of students have had previous experience with playing computer games within a dynamic 2D/3D environment and would have had the opportunity to understand and visually be familiar with game play and concepts such as velocity, acceleration, gravity, and collision detection. It can also be seen in Table 1 that 27% were occasional to frequent users of computer animation. In terms of using game design packages, only 14% have had some experience and have used tools such as GameMaker.

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 2.

Questions used	Percentage of correct answers
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 5 th 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 2: Student success rate in the practical sessions

5 Conclusions and Further Research

It is the first time that such an exercise was carried out with the intention of introducing programming principles to students with diverse background who do not see themselves as serious programmers. It has been shown that Flash Actionscript is a useful tool to engage

students in learning programming principles and develop software applications. The combination of laboratory exercises and group projects has given students the opportunity to engage in the learning of programming fundamentals by using animation, visual tools, and computer game design concepts. Visual programming was an essential part of the practicals as a method to engage students with programming as students can face difficulties acquiring programming skills when they enter a course of study with little confidence in their own ability to use symbolic reasoning.

When evaluating student feedback, the results show that a majority of students –with and without previous programming experience- will engage well with an application environment where visualisation is involved. A high majority of students with previous programming experience (82%) said that they felt that learning to program without visualisation was hard and slowed their progress. Many felt that they found difficulty in acquiring fluency in programming. For students who had no previous experience engaging them in learning programming principles was rather easier than anticipated. *80% of this group found Actionscript a useful tool to understand basic programming and scripting.*

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.

Further research is required on how effective this exercise has been in engaging students with programming in their second year. Some students will be taking C++ as part of their course and will use programming as part of their final year project. It will be interesting to find out how well the students will adapt to migrating from Actionscript to other programming languages.

References

- Benachour P., Longden N. (2007). Communications, Technology and Society –Use of Research Seminar Topics as part of an ICT Learning Program, In Proceedings of the 8th Higher Education Academy Information and Computer Sciences Conference (HEA-ICS), Southampton, UK.
- Cooper, S., Dann, W. and Pausch, R. (2000). Alice: A 3D Toll for Introductory Programming Concepts J. Comp. Sci. 15(5), 108-117.
- Cooper, S., Dann, W. and Pausch, R. (2003). Teaching Objects-first in Introductory Computer Science, Proc. 34th SIGCSE Technical Symposium on Computer Science Education, SIGCSE'03, 191-195.
- Crawford, S., Boese E. (2006), Actionscript: a Gentle Introduction to Programming, Journal of Computing Sciences in Colleges, Volume 21, Issue 3, Pages: 156- 168, USA.
- Dehaan J. (2005). Animation and Effects with Macromedia Flash, Macromedia Press, USA.
- Jones, R. (2004). Designing Adaptable Learning Resources with Learning Object Patterns. Journal of Digital Information, Vol 6, Issue 1, Article No 3005.
- Leutenegger, S., Edgington J. (2007). A Games First Approach to Teaching Introductory Programming, Association for Computing Machinery (ACM) Special Interest Group on Computer Science Education (SIGCSE) Bulletin archive Volume 39, Issue 1, Pages: 115-118, USA.
- McCracken, M., et al. (2001). A Multinational, Multi-institutional Study of Assessment of Programming Skills of First-Year CS Students. SIGCSE Bulletin 33(4).
- Makar J. (2002). Macromedia Flash Game Design Demystified, Macromedia Press; Pap/Cdr edition. USA.
- McDermott R., Eccelston G., Brindley G. More than a Good Story –Can You Really Teach Programming Through Storytelling? (2007). In Proceedings of the 8th Higher Education Academy Information and Computer Sciences Conference (HEA-ICS), Southampton, UK.
- Ohanian. (1989). Physics, 2nd Edition, W.W. Norton & Company Inc. Chapter 12 – Motion of a rigid body, p 297
- Peters K. (2006). Actionscript Animation: Making Things Move, published by Friends of Ed.
- Rashid O., Bamford W., Coulton P., Edwards R., and Scheibel J. (2006a). PAC-LAN: Mixed reality gaming with RFID enabled mobile phones, accepted for publication in ACM Computers in Entertainment.
- Rashid O., Mullins I., Coulton P., and Edwards R. (2006b). Extending Cyberspace: Location Based Games Using Cellular Phones, ACM Computers in Entertainment, Vol 4, Issue 1.
- Spaul M. (2007) The Use of RLOs to Deliver Programming Skills to Multimedia Students, taken from the higher academy Website at: http://www.ics.heacademy.ac.uk/projects/development-fund/fund_details.php?id=63.