Written by Billy D. Spelchan for www.BlazingGames.com

Chapter 9

First Games Summary

Contents

This is a simple summary of what was learned in this part of the book and some suggestions on how you can applied what was learned here towards your own projects.

- Chapter 5: Overview of First Games - summarized.
- Chapter 6: NIM - summarized.
- Chapter 7: Advanced Action Script - summarized.
- Chapter 8: Bomb Nim - summarized.

Chapter 5: Overview of First Games

Goals of this Part

This section is simply a summary of the goals behind this part of the book. With the understand that many people are not programmers, this chapter of the book leads people slowly into the world of Action Script. The first game we created in this part of the book is designed to have a minimal amount of Action Script in it. After taking a look at some more advanced features of Action Script, we then create a more Action Script oriented variation of our first game.

What is Nim

While I have seen this game go by other names, I was introduced to the game as NIM. The game is a very simple game where you want to be the player who grabs the last object. Apparently this game has been used within the mathematics of Game Theory to demonstrate certain principles. The game has already been posted on my Blazing Games site.

Nim design

We already know the how the game will play from the previous section. Also, we know that the game will be designed to use Flash animations instead of Action Script wherever possible. This leaves three questions. What is the object that is being removed, how many objects are there to be removed, and how does the player remove the objects.

While a variety of possible answers to these questions are provided, it is decided that for this version we will have 40 gems. This would also leave room for a space where the user could specify how many gems to take. As all the user has to do is select how many gems to take, the user interface is simply three buttons, with a text area to describe what is happening.

What is Bomb NIM

Bomb Nim is a variation of the Nim game. It plays the same way as the Nim game but with one major exception. Instead of trying to take the last object, the player is instead trying to make the other player take the last object. This minor change actually changes the strategy required to win the game by quite a bit. The game is using bombs as the objects as it makes more sense to not want a bomb.

This Time Its Action Script

Action Script will be used for four key aspects of the game.

- Initial layout of the game board. Forty bombs moving at once!
- Bomb Removal. Instead of forty labelled sequences, all of this can be controlled by scripting.
- Bomb animation. Three different bomb explosions, all controlled with Action Script.
- The user interface. Clicking on a bomb to remove it.

Chapter 6: NIM

Creation of the Gem

I outlined the steps taken to create the gem symbol. This was provided more to show people new to Flash how symbols are created.

NIM logo

Normally when I create a game, the title and menu screens are the last screens that are created. For this chapter, however, I am going to create the games title sequence first. The title screen is it's own scene. The scene starts off blank. Seven gems then grow onto the screen in what at first seems to be a random pattern. Once the gems are fully grown the letter N fades in behind the gems. This is followed by three gems appearing to create the letter I and another 7 gems to form the M.

Play Game Button

Having a fancy title is one thing, but some way to get the game underway is needed. This comes in the form of a Play Game Button. This takes advantage of Flash's button capability by having the button's click area be different from the actual shape of the button. The actual shape of the button, of course, is the erratic shape of the words "Play the Game."

Limiting Layers

In theory Flash lets you have as many layers as you desire. In reality, however, you should try to limit the number of layers that you use. There are three main reason that it is a good idea to limit layers: Designer overload, memory, and Flash workload.

In our nim game, we limit the layers by grouping gems into four layers of 10. We could also have had all the gems on a single layer, but when I originally wrote the game I had the gems in four layers so for this book I kept it that way.

Five Object Animating Limit

Those of you familiar with Flash probably already know that Flash has a tendency to act funny when you try having animation on a layer that contains more than one object. As that is the case how are we going to animate the gems? The trick is to have a separate layer for animation, and move the symbol to the proper layer after the animation is complete. Our game needs up to five objects animated at any given time, so five animation layers are created.

Player Panel

The player panel could have been designed as an all in one object, but that would have added a bit of complexity to the design and I wanted to try and keep this movie as simple as possible for those new to Flash. Instead, the panel is simply a rectangle with rounded corners. Once in it's final position we add the components on the panel. These components consist of a dynamic text message, a static text message, and three buttons labelled "1," "2," and "3." Some code is needed to make the buttons work.

Initializing the Game

At this point, we are now ready to get the game running. To start the game we have a bit of code on the first frame that prepares the game to run. The initialization code simply calls the initialize method and then sets up the games variables.

Updating buttons

The user interface has one potential problem. The buttons for players actions are accessible at all times. While it is certainly possible to ignore this problem, a little bit of Action Script can solve this problem. This is where the updateButtons function comes in.

Ending the Turn

The main endTurn function is designed to handle all of the game logic. This function will be called at the end of each gem removal sequence.

Gem Removal

The game revolves around the removal of gems, so we obviously need a way of removing gems. As the goal of this movie was to limit the amount of Action Script that is to be used, we will have to animate the removal of all the gems by hand. This is not that difficult of a task, only requiring that we have forty short (10 frame) animation sequences. The question is how do we stop all forty gems from being removed? Unfortunately, this will require a bit of action script at both the start and the end of each removal sequence.

Winning or Losing the Game

Both sequences start out with the final gem growing larger while moving to the left side of the screen. At the end of this sequence, in a labelled sequence called we have some code that directs the movie to the player win sequence or the computer win sequence based on who won.

Chapter 7: Advanced Action Script

The Switch Statement

Switch statements are a convenient way of having a large number of if then else statements. While the if statement can be used to accomplish the same thing, the switch statement tends to be easier to read than a large number of "else if" statements. The statement starts with a switch (variable) statement within the switch block, a number of case statements are made. A case is a particular value for the variable. Each of the case statements should end with a break statement, otherwise the code for that case will continue into the code for the next case. By purposely leaving out the break statement, you can create a group of cases that all execute the same code.

Arrays

Arrays are essentially lists of variables. There are three ways to initialize an array. The empty constructor creates an array that initially has no elements. The size constructor creates an array with a specific size. The pre-filled constructor lets you create an array that has it's initial elements declared. To access an element of the array, you use the name of the array variable with the element you want to access enclosed in square brackets.

While and Do

The while loop is simply a block of code that runs until the condition within the while statement is no longer true. The do statement is similar to the while statement except that it will always run the contents of the code block at least once.

For loop

For loops are a specialized version of the while loop but designed to count through a sequence of numbers. The for statement has three parts to it. The start condition, the end condition and the increment.

It's time for a Break

The break statement, which was used in the switch statement, can be used in while loops and for loops to exit the loop early. The continue causes code in the rest of the loop to be ignored and returns execution to the condition statement of the loop.

Boolean Arithmetic

We started this section off with a look at the binary numbering system that computers use. We then look at the more convenient hexadecimal system that is familiar to any programmer who has done low level programming. Finally, we took a look at various boolean operations. The AND operation requiring that to bits be set to be true. The OR operation requiring that one or both of the bits be set to be true. XOR is a specialized version of OR where only one of the two bits can be true to gain a result of true. Bit shifting is when the bits that make up a binary number are shifted one position.

Debugging: Output

The trace command can be used to send messages to the output window. In addition, the output window has commands that allow you to dump a list of variables and objects used in the Flash movie.

Debugging: Breakpoints

A breakpoint is essentially a marker. You can set a breakpoint on any line of code that will be executed. To use the breakpoints, run the movie by using the "Debug the Movie" command. When your program reaches one of the breakpoints that you set it will pause. To continue the movie, click on the play button. Alternatively you can step through the movie by using the step over or the step through commands.

Debugging: Watching variables

The debugger automatically shows you the values of every element in the movie. This list of variables is handled the same way as the movie browser component. In addition, Flash has a watch tab that lets you reduce the list of variables being watched to just the variables that you are interested in. Not only can you watch the variables as the program is running, but you can change the variables.

Chapter 8: Bomb Nim

Building a Bomb

Deciding on the type of bomb. As the game is being named bomb nim, it is obvious that we are going to have to use bombs as the object. There are a large number of options here. For the game I have chosen your typical Saturday morning cartoon style of bomb.

Lighting the Fuse

An animated fuse helps make the game more vibrant. The way this way created was by creating four different spark versions. The frames of spark animation then randomly go to a different version of the spark animation.

Exploding Bombs

Bombs need to be able to explode. To add variety to this, three different types of explosions are created. We have a flash explosion, a smoke explosion, and a cartoon explosion.

Bomb Highlighting

For managing the user interface, we need to be able to highlight the bomb. This can be done by creating an outline movie that is the shape of the bomb. The bomb movie then has this highlight movie underneath the bomb. A bit of code makes this movie visible when the bomb is to be highlighted and invisible when it is not highlighted.

Layout

Laying out the playfield is done by using only one bomb. That bomb is cloned in order to create forty different bombs. The positions of the bombs can be calculated algorithmically.

Math Behind Motion

We first take a look at the math behind motion. We then create a general purpose method that can be used to move an object.

Animating the Layout

An animated layout sequence. This takes the layout routine create in the layout section and changes it so that the bombs all start out on the edges of the screen and move towards their final position. Through the power of Action Script, we can move all of the bombs at the same time.

Player turn

To actually handle the selection of bombs, we are going to take advantage of two mouse controlling methods. The mouseMove function is used to highlight the bombs when the mouse is over a bomb that the player can select. The mouseUp function is used to actually take the bombs away.

Bomb Removal

As the game no longer has blocks of animation for each of the bombs, we now simply have a loop that loops through each bomb that was taken and causes it to move.

Computer Turn

The logic the computer uses for this game is fairly simple. If there are four bombs remaining, the computer takes 3. If there are three left, the computer takes 2. If there are two or one bomb left then the computer takes 1. Otherwise the computer takes a random number of bombs.

Game Over

The end of the game displays a message containing text describing who won. The continue button uses bomb symbols to add to the appearance of the button. With the game over screen done, we put together the title sequence. This sequence consists of a screen sized bomb exploding followed by the title of the game and a start game bomb button.