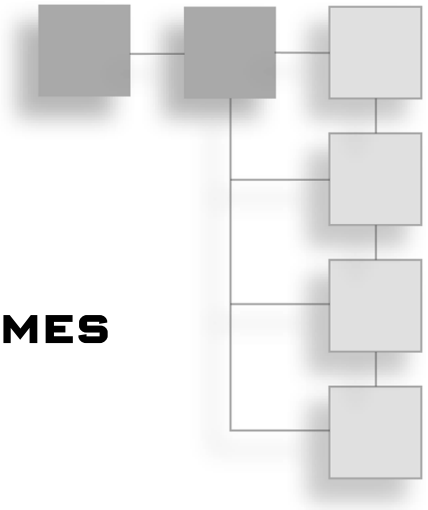
Chapter 15



Graphic Adventure Games

In this chapter, we will look at how to make your own graphic adventure games. These are different than graphic text adventure games covered earlier in the book; graphic adventure games do not include text entry and are wholly mouse-driven. You may think that any adventure game that uses a mouse-driven approach can be called a point-and-click graphic adventure game, and you would be partly right. Any graphic adventure game can be called a point-and-click adventure, but in this case, we are talking about a first-person graphic adventure game where the player cannot see his on-screen avatar, but instead he can walk around the game world looking through the eyes of the game's character.

Note

First-person games give the player the perspective of looking through the eyes of the character as he walks around the game world. Examples of first-person shooting (FPS) games include *Call of Duty* or *Medal of Honor*. Third-person games involve an avatar that the player can see on-screen; example games that follow this genre include *Mass Effect* or *Assassin's Creed*.

These types of adventure games were extremely popular in the 1990s with games such as *Myst* and *The Last Express* gaining many fans, but they have on the whole fallen out of favor with game designers who have shifted their attention to point-and-click adventure games. Games such as *Myst* had amazing stories and complex puzzles with many interesting and colorful locations to visit.

The key components of these types of adventure games include:

- **Graphics:** Amazingly detailed scenes with animations, mainly realistic graphics.

- **Locations:** Many places to visit and discover.
- **Puzzles:** Mouse-based puzzles, such as levers, buttons, moving of objects that would open/close a door, room, or access to clues.
- **Directions:** The ability to move in typical first-person directions such as north, east, south, west, up, and down.
- **Mouse cursors:** Mouse cursors are very important to let the player know he can move in a particular direction or activate a part of the scene.
- **Full motion video (FMV):** Some adventure games of this type might contain FMV of characters or events.

Creating Worlds

In this chapter, we are going to use photos to represent our game world. Although many of these types of games use pixel- or Photoshop-based art, it would potentially take a long time to create all the various angles needed to make even the simplest of games. If you don't have the art skills to create art-based scenes and locations, then using a camera is a great way of getting your game up and running quickly.

So we will be using an 8MP camera and converting the images to black and white to give them a particular style. If you don't have a digital or SLR-based camera, you can use a camera phone. Remember you can always swap out the images later if you find you can get access to a better camera or an artist.

Note

Some photo-based software allows you to add graphical effects to images, enabling you to give them their own unique feel and style that may better suit your game than an untouched photograph would.

The basic process for creating this type of game is as follows:

- **Design concept:** The design concept is a simple document you write to outline the game and what you intend to put in your game. This would include the places the player can visit and, if you are using photos as in this example, a list of locations to photograph.
- **Basic map plan:** The basic map plan is a top-down drawing of the places that the player can visit, very much like the maps generated in the Text Adventure Map object. You should draw out your map locations and the directions the player can move in so that you have a good idea how the world fits together.
- **Puzzles plan:** If you have a puzzle in a particular scene, you should document it and how it should work. It's important to note that if you are using photos, you

may need to take multiple pictures of the same scene, with the puzzle disabled/enabled. An example of this is: Perhaps you have a scene with a box so you may need a photo of the box closed, and then the player can open it, so you will need a second image of the box opened. The puzzle plan will help you document any additional images that you might need.

- **Getting content:** You will need to get someone to draw the content or you will need to go out and take photos of the locations that you intend to use.
- **Creating code:** Once you have your content, you would start putting it into MMF2 and then coding the game.

Park Life

In the following example game called *Park Life*, you will need to travel around a park environment and find a lost football. Here is a simple example story to explain the concept:

> *Billy was playing football in the park, when suddenly a storm appeared. Billy ran under a park building until the storm had passed. Billy then walked home and realized that he had forgotten his favorite football, and it is somewhere in the park. Your task is to help Billy find his football before it gets too dark.*

The story for this game isn't meant to be complex but to provide the player a reason to be in the park looking for a particular object. To play the game, you need to navigate the park and find the football.

The number of photos used in this game will be kept to a bare minimum, but the coding process for making the game will be the same as if it was 5 or 100 images. Of course, the more puzzles and locations you can include, the more coding you will need to do, but the process would be the same.

In this example, I have taken several photographs from around my local park, but it's important to note that I have taken them with the intention that the player can move forward, left, right, and backward, but the player will not be able to turn around on the spot. This ensures that the number of pictures I need to take is limited, as I don't need to worry about every location having images from all four directions.

As you can see in Figure 15.1, the player can decide to move from location A to B, and she will be facing forward. If the player then decides to move back to A, she will still be facing backward but will be effectively walking backward until she has reached her original starting location.

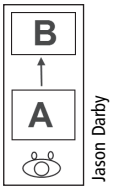


Figure 15.1
Always facing forward.

Some of the issues to consider when using photos rather than graphics include:

- **People, cars and animals:** If you are using photos as part of your game, you need to take into account any people, cars, animals, or other moveable objects that might cause you issues when creating your game. For example, perhaps you have in one shot a person in the distance, but when you move forward that person has disappeared (because he has moved out of the shot of the next image), but when the player backtracks, he will suddenly appear in the distance. The same issue exists for vehicles if, for example, you are making a game that has a player walking along a road. You may need to delay taking a picture until the picture is clear of such objects; this may require you waiting a period of time until the area is clear. Another issue with inadvertently including people in your images is that if you are making a commercial game or you plan to upload it, the person you photographed (if he can be identified) may not be happy being included in a product/game without his permission or knowledge.
- **Lighting:** If you intend to take pictures, a big issue to consider is the current lighting of the scene. If you take pictures over the course of time, the light that is available may change and can have a dramatic effect on the images that you have created. You may have to use a flash if lighting changes significantly in different areas where you are taking a picture.
- **Camera angles:** When taking several pictures, it is likely that you will change the angle from which you are taking some of the pictures and the inconsistency can cause problems. For example, in a scene, I had a football in a bag. As part of the game, the player would need to click on the object to remove the bag and reveal the ball. The issue was that I had decided, after lowering the camera, to take another picture and took it at a different angle. This situation illustrates the issues that can occur with photo adventure games. Firstly, I hadn't considered what happens if I changed my mind about a picture, and secondly I hadn't taken a tripod, which would have ensured all images were at the same height.

- **Continuity:** One of the issues in making a game using photos or videos of real-life places is continuity. This is a common problem in movies when doing work over a period of days, weeks, or months where things might change unnoticed by you. For example, you have a character that appears in the game, and you take photos of this character. As you continue to make the game, you accidentally change his coat or belongings, or the scene changes. Perhaps you took a picture of a road, and then a month later, there is a construction sign that has suddenly appeared. It can be difficult to spot these errors sometimes, which is why there are often "continuity experts" who work on TV shows and movies who watch for these types of mistakes, because the audience does notice even subtle changes to an environment.

Note

In *Park Life*, I changed all the images to be grayscale only, which helps reduce the issues with lighting.

Note

If you want to use graphics and animations within your game, you can still use photos to establish the basic structure of your game created in the first instance. You can use this structure as reference points or locations that you need to create within your game.

Creating a Map

When designing your game, you may want to create a map of your game's locations. This doesn't have to include puzzles or particular characters you will meet, but it is used to give you a general idea of how the player will navigate the world.

In this example, we can use the photos we have taken to create our map, but if you were creating this type of adventure game with graphics rather than photos, you could still use photos as reference or use line drawings. You can see an example of the *Park Life* game in Figure 15.2.

Game Functionality

The game is created by placing an individual image on a frame of its own. Each frame is either a location or an animation, and on each frame is a set of invisible boxes that will change the mouse cursor when the player moves over it. The player will then click to move in a particular direction or activate a simple event.

You can see the basic storyboard of the game in Figure 15.3, which also has the boxes already applied that can also be seen in the first frame of the game in Figure 15.4. In

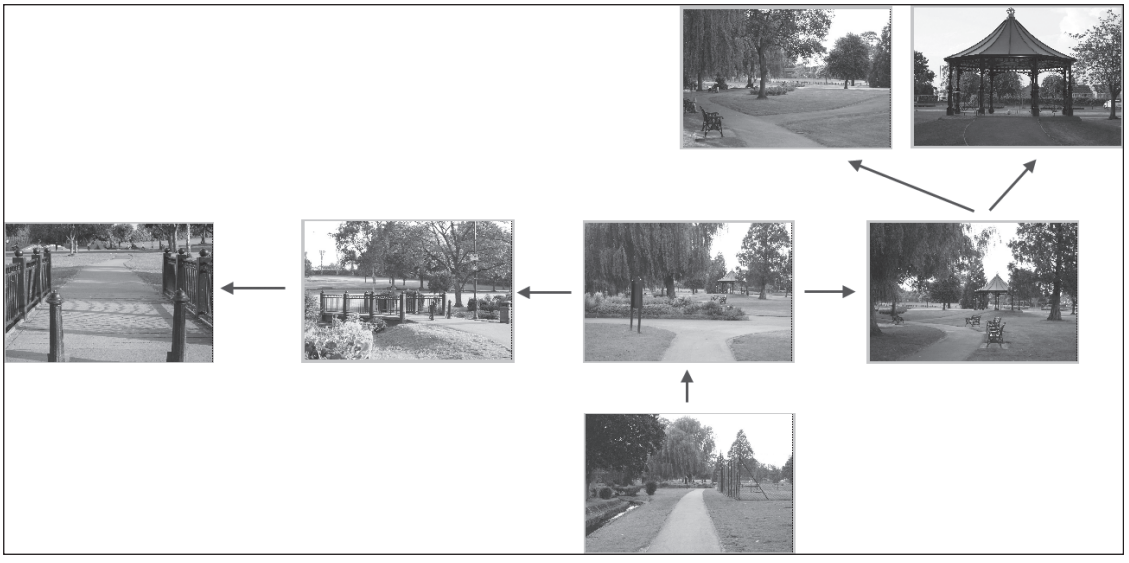


Figure 15.2
Park Life birds-eye view.

Figure 15.4, you can currently see a box that will be used to change the mouse cursor. Once we finish creating the game, this box will be made invisible to the player, but while designing it we will leave it on-screen so that we know what actions are available while putting the game together.

Note

You would most likely have objects that are interacted with all on the same frame. For example, in this game, the player clicks on a football within a bag to remove the bag. If you were creating the game using drawn images, you would place an image of a ball in a bag and then animate it to show only the ball, but the background would stay the same.

Setting Up the Game

You can download the basic structure of the game with the included photos in a file called PhotoAdventure.mfa. In this file you will find 10 frames, which already have all of the objects on each frame ready for programming.

We have the following objects:

- **Backdrop object:** These will contain all of our background pictures.
- **Active object:** The Active object will be used to create boxes that will allow us to check when the mouse cursor is in a particular place on-screen.

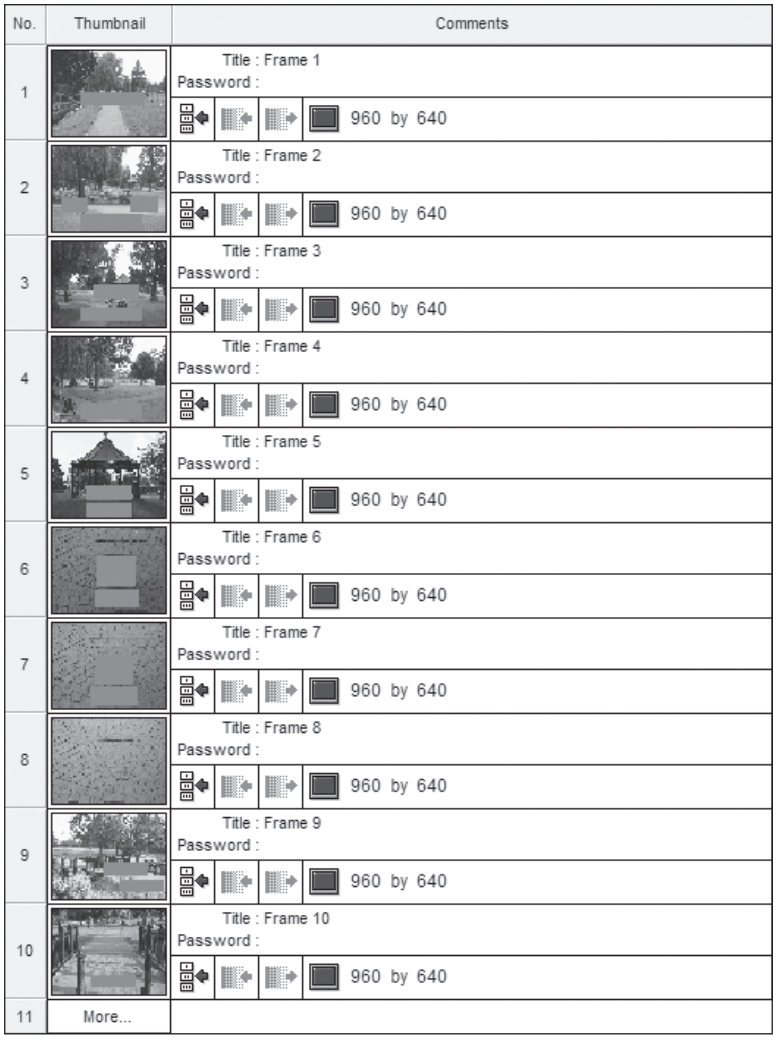


Figure 15.3
The Storyboard editor with all the frames set out.

- **Cursor object:** This will allow us to change the shape/image of the mouse cursor to provide players with more information on what potential actions are available to them.

We have 10 frames that contain the following locations, and we describe the purpose of each:

- **Frame 1:** The starting location for our adventure; you can only move forward from this location.

Figure 15.4
The first frame with a movement box placed on it.

- **Frame 2:** The second location you can visit. From here, you can go down the left path, down a right path, or return to the original location.
- **Frame 3:** If you selected the right path from frame 2, you will see a building going forward. You can go left or you can go back to frame 2.
- **Frame 4:** From here, you can only go backward in terms of gameplay, though in the picture, the actual path continues. If the game was larger, we might have considered having the player move in this direction. Of course, one of the difficulties in using real-life pictures is that many directions and paths may lead off one another, creating many paths.
- **Frame 5:** You are closer to the outdoor building. From here, you can go forward or backward. We ensured that the football was placed in the bag before the picture was taken in order to give the player a clue.
- **Frame 6:** A close-up of a bag, where you can just see the ball. We have a box over the ball to remove the plastic bag, and you can also move backward from here.
- **Frame 7:** If you were to click on the bag in frame 6, you would see the football in frame 7. If you click on the ball again, you have completed the game, or alternatively you could move backward.
- **Frame 8:** When you have clicked on the ball, you will be taken to this frame that shows just the floor. From here, the game will be over, and you will not be able to move in any direction.
- **Frame 9:** In frame 2, the path separated. If you choose left, you can then see a small bridge. From here, you can move forward or backward.

- **Frame 10:** You are now on the bridge seen in frame 9; you cannot move any farther forward, but you can click backward.

Setting Image Effect

When I took the photos for this game, I used a Canon 400D camera, and even though the photos came out quite well, changing the images to monochrome made the photos even better for this type of game. We could have applied this effect to our images using a graphics program, such as Photoshop, Fireworks, or MS Paint, but because we wanted the option of keeping the original photos within MMF2, we applied a graphical effect within MMF2.

Once we added all the pictures to the frames as backdrop objects, we changed their graphical effects by clicking on the object and accessing the Display Options tab. You can see the Effect option has been changed to Monochrome in Figure 15.5. If you access the properties of the Effect option, a dialog box appears (as shown in Figure 15.6), which gives you a number of possible effects that you can apply to an image. You can see there are a number of effects depending on if you intend to use software or

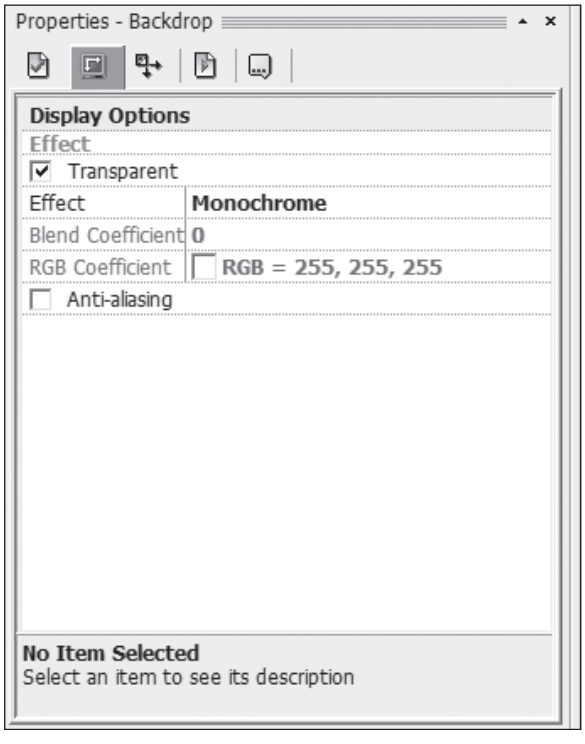


Figure 15.5
The Monochrome effect applied to an image.

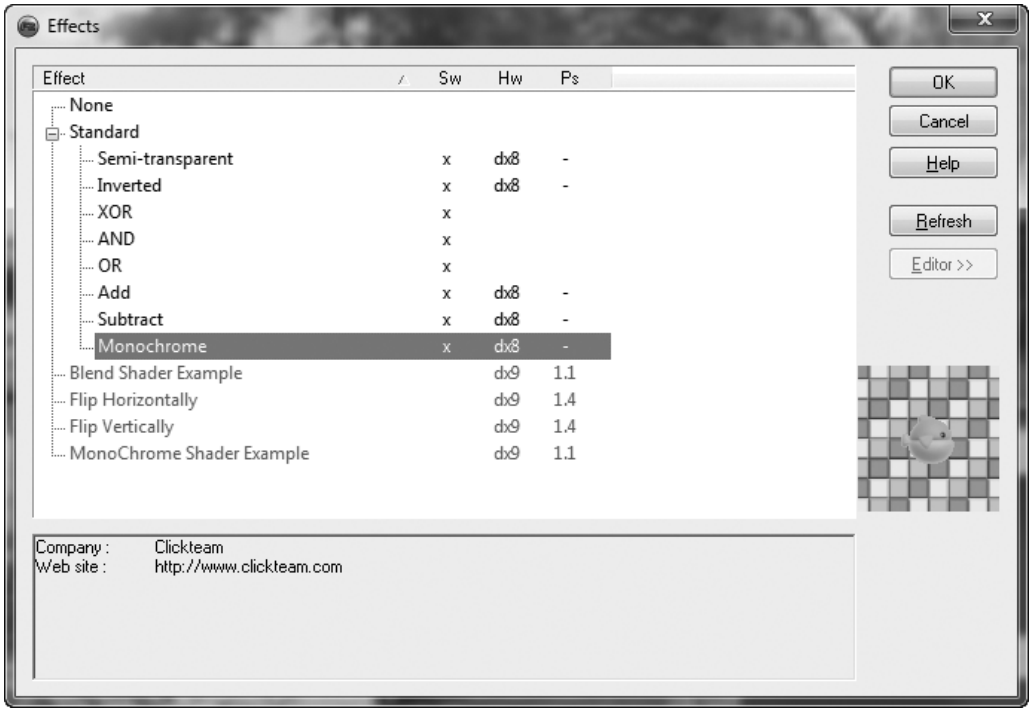


Figure 15.6
The Effects dialog box.

hardware effects. Hardware effects use the graphics card processor, unlike software which uses software to create an effect. The main consideration is that using the graphics card is faster, but it requires you to have a card that supports the effect you want to apply.

Note

The benefit of applying a graphical effect via MMF2 rather than from within a graphics program is that if you do it within the graphics program, you have physically changed the properties of the images. If you wanted to undo those changes, you would need to revert the images back to how they were before they were modified or keep multiple copies of the same images with the different effects applied. If you use MMF2's effects, you can temporarily apply an effect and then change it back if you decide you don't like it.

Note

Once you are happy with the type of effect you want to apply to your images, you can then do this through a graphics package and then re-import these images into MMF2. Doing so may reduce the size of the images stored within MMF2 and make your final game size smaller.

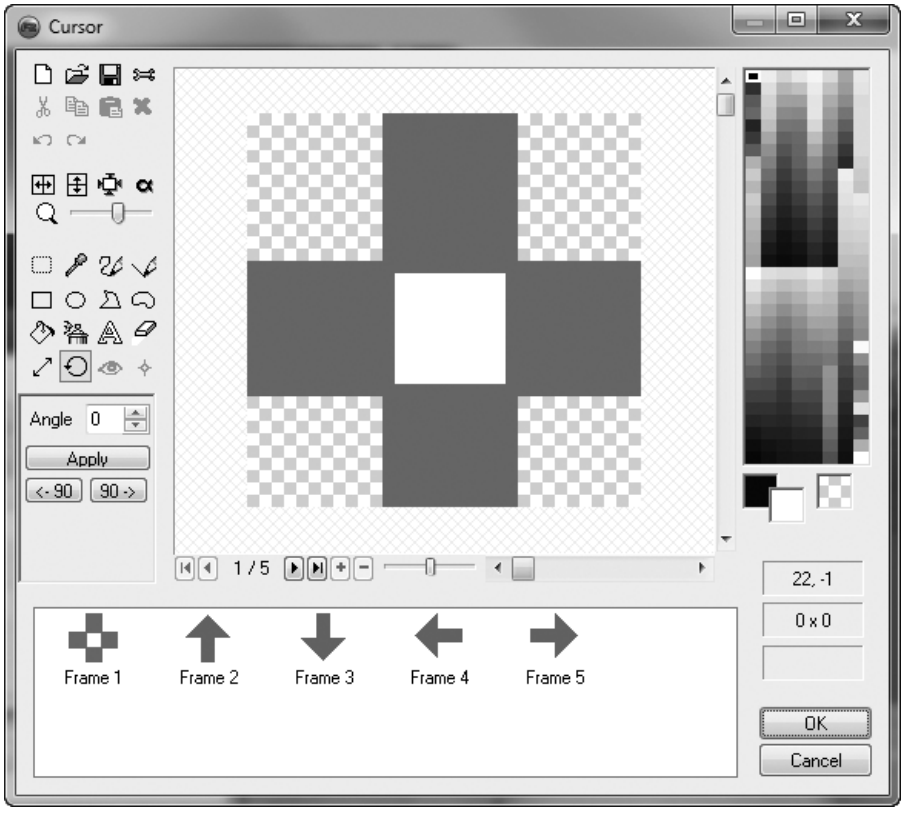


Figure 15.7
The different animations for our cursor.

Cursor Object

We have placed a Cursor object on every screen that will be changed depending on the box that it is currently over; to do this, we have set a number of images within the Cursor object and given them a specific name. You can see the images in Figure 15.7. We have five cursors: the first has the name "cursor" and is the default image when moving the mouse around the screen. The arrows are named "up," "down," "right," and "left," which indicate the direction the player can move in.

Adventure Game Code

In this section, we will take a look at the code required to create this game. The code in our game consists of the following key code:

- **Start of Frame:** On every frame, we set the mouse cursor to the cursor image. This ensures that we have the initial drawn cursor rather than a traditional Windows cursor.

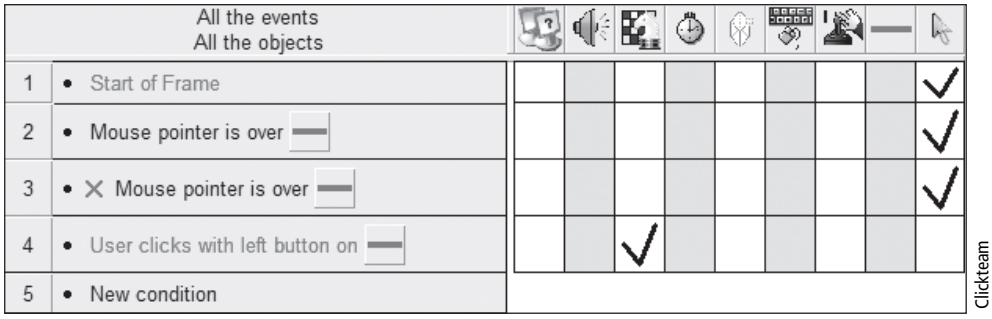


Figure 15.8
The Event editor for frame 1.

- **Mouse Pointer Is Over:** When the mouse cursor is over a particular active box, we then change the mouse cursor to the correct direction. So when the player is over a forward box, it will change the cursor to an up image.
- **Mouse Pointer Is Not Over:** When the mouse is not over a direction box, we need to reset it to the cursor "crosshair" graphic. We can do this by checking that the cursor is not over any boxes.
- **User Clicks On:** If the user clicks on a particular box, we will then jump to that particular frame.

In Figures 15.8 and 15.9, you can see the Event editor and the Event List editor for the code in frame 1. As you can see, the code is pretty straightforward, and in screens with additional directions, just duplicated further. If you add puzzles, then the code will get slightly more complex, but these types of computer games are quite easy to

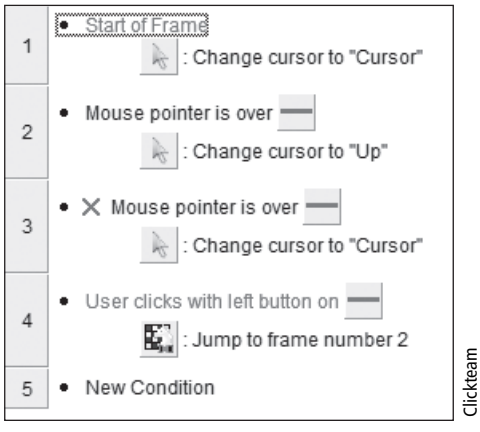


Figure 15.9
The Event List editor for frame 1.

get up and running. The complex part of creating the game is working out puzzles and story elements that players will enjoy.

If you wish to try out the game, you can load and play PhotoAdventure_Code.mfa. The game is short, but it should give you a good basis for creating your own graphic adventure. The PhotoAdventure_Code.mfa version contains all of the red link boxes which are used to change the mouse cursor and test for the player's input. You couldn't release a version like this, so it is important to hide any boxes such as these. We have created a file called PhotoAdventure_Complete.mfa, which hides all of these boxes.

Chapter Summary

In this chapter, we have looked at how you can create your own graphic/first-person adventure game. We have shown you that you can use photos if you don't have the drawing skills or if you want to get your game code up and running quickly while waiting for the art to be completed. You will be able to build your game quite quickly, allowing you to focus on adding more complex code for puzzles and story elements to entertain the players.

In Chapter 16, we will discuss the various editors and applications we have created to help you make your games quicker and easier.