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

Chapter 29

Game Concepts

Contents

Before we get into the creation of some Arcade Games, now would be the ideal time to take a look at some of the concepts and terminology used in the creation of arcade games.

- Lists - Linked lists
- Sprites - What the heck are these sprite things?
- Tile sets - a simple way of having multiple levels
- Scrolling - taking the tile set to the ultimate level.

Lists

In both of our arcade games we have situations where there are a various number of objects. We could have a large array that holds all the potential items. The problem with this is that it is inefficient to have arrays in which large number of elements are empty. Especially when you have to loop through the array for things such as collision detection.

In other programming languages having a variable sized list is the ideal situation for something known as a linked list or a doubly linked list. Sounds scary, but really it isn't. Before getting into a doubly linked list, let us first look at a linked list. This list starts with an object within the list. This object points to the next object in the list. To traverse the list you simply need to go to the first object on the list and use it's next object to get to the next object. The next object has a reference to the next object after it. You keep going to the next object (doing whatever it is that you are doing with these objects) until the next object is null.

A doubly linked list is very similar. The big difference is that objects also have a reference to the object before them. This makes it easy to traverse the list in either direction and also makes it easier to remove an object from the middle of the list.

Flash makes it possible to have the flexibility of linked lists without there complexity by having a very robust Array class. Lets take a look at some functions that make this work possible. The push and pop functions are used for adding a new element to the end of the array (push) and removing the last element from the end of the array (pop). The shift and unshift are the same but shift removes the first element instead of the last and unshift inserts the new element as the first element of the array.

The function that makes it possible to avoid using linked lists is the splice function. This function lets you remove elements from anywhere in the array. This alone would be great, but it also lets you insert elements into the middle of the array! The format of this command is simply

```
arrayname.splice(where, delete_count, optional variables to insert);
```

For instance, instance, if you had an array with 10 elements, to remove elements 3, 4 and 5 you would use

```
tenarray.splice(3, 3);
```

Sprites

People who try to get into game programming, especially arcade game programming, will hear the term "sprite" used quite frequently. What exactly are sprites? While there is a soft drink named "Sprite," game programmers are generally not talking about that. Nor are they talking about fairies or elves, though if they are developing a fantasy game there may very well be fairies and elves in the game. What they are talking about is a graphical object that can have transparent portions and which can be placed on the game screen and moved without damaging the images underneath them.

The term originally came from Atari (one of the earliest game manufacturers), though I am not aware of the specific reasons why they were called sprites. I assume that it had something to do with the magical nature they seemed to have. You see, in the old days (and in fact, even today) when you drew something on the screen, whatever was underneath what you drew was permanently destroyed. Having something that programmers could easily position on the screen without having the underlying image on the screen being destroyed must have seemed magical! At least hardware sprites would have been. Software sprites actually do destroy what is underneath them but they know how to restore what was underneath them when they move.

Sounds like a movie clip? That is one reason why flash can be easily used as a sprite. Flash uses software sprites in the form of layers of movie clips to draw the display. When a layer changes, Flash knows what areas of the screen have been damaged (most likely through a technique known dirty rectangle tracking but that's a bit beyond this book) and redraws only the damaged portions of each frame to keep the fast frame rates.

To sum this section up, Sprites and Movie Clips are pretty much the same thing. Movie clips should be thought of as very sophisticated sprites. While this may be a bit of a letdown, quite often techniques and algorithms have complex sounding names simply because people need to call them something. In fact, many advanced graphic algorithms are simply named after the person who came up with the technique.

Tile sets

I use the term playfield to represent the playable portion of the game. More importantly, the playfield can often be represented using something known as tile sets. Our String Along game will make use of a simple tile set based playfield. It is a very useful way of making complex levels out of simpler sets of tiles.

Essentially, the game's playable area (playfield) is broken into a grid. For argument's sake, let us say that the playfield is the full movie window, which is 640x480. Let's say that we want fairly large tiles (32x32), so the grid size would then be 20x15. This also means that there are 300 locations on the screen. If we create a group of tiles, lets say 64 so that the set of tiles can be easily represented using text, each tile being a different background, item, or wall and assigned each tile a number we would have the making of a tile set.

Each of the 300 locations on the screen could then contain a value between 0 and 63 (or 1 and 64 if you prefer) which would indicate the particular tile to use for showing that location. This 300 letters worth of data could be easily stored in a series of strings, allowing you to have a lot of levels with very little overhead.

As the sprites in the game can easily be translated into tile coordinates, you would also have a very easy way to tell where the player is and what type of tile they are entering. This can be used for things like eating dots or handling jumping in a platform game.

Scrolling

Tile sets can be of even greater use for larger worlds. If you wanted to, you could create a playfield that was 32x32, even though the viewable portion of the map is 20x15. By doing so, only a portion of the playfield, commonly called the viewport, would be visible. As the player moved, the other portions of the playfield would be shown. This is what is known as scrolling.

There are a few ways of implementing scrolling. One would be to actually have a movie that held the whole 32x32 tile set and scrolled. The second would be to have a smaller 20x15 playfield that when scrolled would just replace the values of the grid tiles with the appropriate tiles in the larger playfield map. The second method doesn't allow for smooth scrolling, so if smooth scrolling is wanted without the larger map, a compromise could be reached. By having a 21x16 or 22x17 tile movie you would be able to scroll the image smoothly. When the movie was scrolled to the point where it will no longer cover the screen, you would quickly reset the movie replacing all the tiles with their shifted version.

If you wanted really fancy scrolling, you could have multiple playfields (possibly with each playfield having it's own tile set). The upper playfield layers would have transparent and partially transparent tiles so portions of the lower layers would be visible. More to the point, each layer would scroll at different rates. This would give you a more 3D effect.

The biggest concern of having more than one layer of tiles would be memory usage and speed. While Flash is very fast, it does not have native support of playfields and tile sets. This means that the updating of the scrolling may not be as smooth as it would be on a platform that was designed specifically for handling such things.