[Version number, author, save date, and task days total are fields and can be updated with F9]

Background System Design Document, v3

Greg Sabatini

October 12, 1999

Introduction

This system must load in the backgrounds, both scrolling and non-scrolling, keep track of which ones are in memory, and blit them to the display.

Requirements

This system must be fast on both systems, but use the same assets. We will be using run-length encoded strips of 64x480 pixels to composite the backgrounds.

We have 2 possible methods for doing the background compositing.

The first system requires a background buffer located in video memory and two additional buffers in video memory for the scrolling worlds. These are the Left and Right buffers. We will initially composite the base 640x480 image onto the background buffer and the strip to the left and right of the image in the Left and Right buffers, respectively. When the user scrolls, we will first blit the portion of the Background buffer that is still in view to the Back Buffer. The new portion would be blit from the Left or Right buffer to the Back Buffer. The Back buffer then gets blit to the Background buffer. This gives us 3 blits, one of (640-x)*480 pixels, one of x*480 pixels, and one of (640*480) pixels. It is important to note that in this scenario we must use a power of two (divisor of 64) for the scrolling factor, x, so that the Left and Right buffers never are moved off of until we reshuffle the Left and/or Right Buffer. On the Mac we can still use the Underlay buffer as the Background buffer, but the Left and Right buffers will not exist, as the strips will be in system memory already.

The second system requires us to put a full 12 strips in video memory, individually. The strips are then composited directly to the back buffer. As the user scrolls, the strips get shuffled to remove those that are not in use keeping a 1 strip buffer on either side. This gives us an average case of 11 blits: nine of 64*480 pixels, one of x*480 pixels, and one (64-x)*480 pixels. Once again, it is important in this scenario to keep a power of two (divisor of 64) for the scrolling factor X. On the Mac, this scenario would have to be performed completely from system memory, however due to the RLE of the strips, could prove to be faster than the previous.

For a non-scrolling background, the same system will be used of strips and buffers, with appropriate simplifications done to gain as much speed and memory savings as possible.

This system is frame rate critical to the product. We can not afford the scrolling background to cause a 1 frame decrease.

This system must be closely integrated with the graphics system, as it is dependent on this system so that it doesn't have to clear the back buffer. Also this system must have complete access to the back buffer.

Structures/Classes

```
class CBackground {
        int XScrollPosition;
        LinkedList CBackgroundAssets; // Background
}

class CBackgroundAsset {
        pRLE
        pDDSurface
        int nID
        int Left, Right;
}
```

Functions/Methods

```
class CBackground {
        Load(szLabel);
        Blit(DDSurface BackBuffer,  int X); // Blit whole back buffer with arg for
world to screen offset
}

class CBackgroundAsset {
        Blit(DDSurface*); // Fast Blit RLE to DDSurface (for System to video)
        Blit(DDSurface*, int X); // Fast blit, blits DDSurface to DDSurface at X
        Blit(DDSurface*, int X, int L, int R); // Blits DDSurface rect with L and R to
DDSurface at X

        SetDDSurface(*DDSurface); // sets, clears DDSurface pointer.
        SetRLE(void*); // sets RLE Buffer
}
```

Diagrams

Schedule Task List

System Tasks	Duration	Dependent
Design Background system	2 Days	Design Document
Design Background Strip RLE tool	1 Day	Design Document
Code Background Strip RLE tool	2 Days	RLE tool designed
Test & Revise BG Strip RLE tool	2 Days	BG Strip RLE tool coded
Code Win32 CBackgroundAsset class	2 Days	Background system designed
Code Win32 CBackground class (both methods)	3 Days	Win32 CBackgroundAsset class coded
Integrate Win32 CBackground & CBackground Asset	1 Day	Win32 CBackground class coded
Test, Optimize, & Revise Win32 Background system	2 Days	Win32 Background system Integration, Win32 performance analysis tool
Code Mac CBackgroundAsset class	2 Days	Background system designed
Code Mac CBackground class (both methods)	2 Days	Win32 Background system coded, Mac CBackgroundAsset class coded
Integrate Mac CBackground & CBackground Asset	1 Day	Mac CBackground class coded
Test, Optimize, & Revise Mac Background system	3 Days	Mac Background system Integration, Mac performance analysis tool
Rework #1 BG Strip RLE tool	1 Day	BG Strip RLE tool complete
Test & Revise BG Strip RLE tool Rework #1	1 Day	BG Strip RLE tool Reworked #1
Rework #2 BG Strip RLE tool	1 Day	BG Strip RLE tool complete
Test & Revise BG Strip RLE tool Rework #2	1 Day	BG Strip RLE tool Reworked #2
Rework #1 Background system	2 Days	As Needed
Test & Revise Background system Rework #1	1 Day	Background system Reworked #1
Rework #2 Background system	2 Days	As Needed
Test & Revise Background system Rework #2	1 Day	Background system Reworked #2
Rework #3 Background system	2 Days	As Needed
Test & Revise Background system Rework #3	1 Day	Background system Reworked #3
Total	36 Days	

Memory

This system will require 64*12*480*2 = 720K of video memory, out of the 848K available. Additionally, it will require in the scrolling world approximately 3 megs of system memory: 8 world areas * 800 pixels wide * 480 pixels tall * 2 bytes per pixel * 50% RLE compression = 3000 K.

On the CD, the backgrounds will take up 12.3 megs of space:

Name	#	Size	Total
Scrolling World Area	8	800*480*2	6144000
Curriculum Backgrounds	4	640*480*2	2457600
Clicky/Bricky Backgrounds	4	640*480*2	2457600
Additional Backgrounds	3	640*480*2	1843200
		TOTAL	12.3 Megs

Risk Assessment

Speed here is the biggest potential problem. We spend time optimizing the background to be as fast as possible. Coding our 2 best solutions and using the better one will preoptimize the code. There is an additional time allocated for optimizing.

The Mac has many more potential problems than the PC. Our research indicates that the Mac abstracts video memory and doesn't allow us to manage it and use it for the very fast video-to-video memory blits. Once again, we may need to call in the Mac consultant for help optimizing our Background system.

QA & Test

If backgrounds are displayed properly, and the screen scrolls properly in the scrolling world, then this system works. If background graphics are garbled or pixels are repeated, or there is a noticiable chug in the game at regular intervals while scrolling, it is probably in this system.