Graphics System Design Document, v2

Randy Angle

October 20, 1999

Introduction

The graphics system provides three levels of support. The first area is setting the screen and video memory into a playable 640x480 pixel resolution with at least 15bit color and the low-level support for page flipping. This is accomplished by DirectDraw on the PC and DrawSprockets on the MAC. The second level of support is the handling of bitmaps or surfaces that contain the loaded assets. The third, higher-level, responsibility is to abstract the handling of graphical primitives in a cross-platform way. These graphic primitives will be sprites and base class functionality for graphic types discussed in other documents (ANIM sprites, procedural textures, etc.).

Requirements

The three areas of the graphics system will be handled by several classes. The CVideo class handles the screen functions and page flipping. The CBitmap class contains the single instance of an art asset and abstracts the concepts of surfaces or texture space. The CSprite class is used in conjunction with properties from the CObj class, for scene objects, and provides the base functionality for graphical objects. There can be several CSprite objects that will use a CBitmap as their displayed image.

The critical aspect of this design is that a single set of cross-platform assets can be used in a consistent and high performance way on both MAC and PC. PC frame rates on the target machine should be able to attain 30fps; the minimum system should attain 15fps. On the MAC these frame rates may be slightly less but a good goal would be 15fps minimum.

We are assuming that on the PC there will be 2Mb of VRAM for multiple display pages, background buffers, and static images like fonts and particle sprites. The MAC uses video memory slightly differently so these concepts will be abstracted accordingly.

Structures/Classes

```
class CVideo
{
private:
#ifdefined _WIN32_
        LPDIRECTDRAW4          pDDContext; //PC
        LPDIRECTDRAWSURFACE4   pFrontBuf; // chunk of video memory used for display
        LPDIRECTDRAWSURFACE4   pBackBuf; // video memory drawn to before a flip to display
#else
        DSpContextReference    pDSContext; //MAC
        CGrafPtr*              pFrontBuf;
        CGrafPtr*              pBackBuf;
#endif

public:
        CVideo();
        ~CVideo();
        BOOL    SetMode(void);         // finds and sets 640x480x15 (or 16) and inits buffers
        void    Blit(POINT Position, SIZE Size, WORD* PixelBuf);
        BOOL    Update(CBitMap* pWorkBuf);    // calls background update, does blit and flip
                // returns MSB values for current video mode
        void    GetMSB(WORD *RedMSB,WORD *GreenMSB,WORD *BlueMSB);
        void    Blit(WORD* pPixelBuf, POINT Position, SIZE Size, RGB16 ColorKey);
        void    Blit(WORD* pPixelBuf, POINT Position, SIZE Size);
}

class CBitMap
{
private:
        SIZE    Size;
        WORD*   PixelBuffer;

public:
        CBitMap();
        ~CBitMap();
        BOOL    Load(char* FileName, CVideo* pVid);
        void    SetSize(SIZE NewSize);
}

class CSprite
{
private:
        CBitMap* pImage;
        POINT   DrawnPosition;
        RECT    SubBox;
        RECT    ClipBox;
        int     Layer;
        RGB16   ColorKey;
public:
        CSprite();
        ~CSprite();
        void    Draw(CVideo* pVid, POINT* dst);       // draws the whole sprite
        void    Draw(CVideo* pVid, POINT* dst, RECT* SubSrc); // draws the sprite from a sub-tile
        virtual BOOL   Load(char* FileName);
        pure virtual void     Update(float DeltaTime) {;};
        void    SetPosition(long X, long Y);
        void    SetColorKey(RGB16 Color);
        void    SetClip(RECT* ClipBox);
        void    SetSize(SIZE Box);
        void    SetLayer(int newLayer) {Layer = newLayer;};
}
```

Schedule Task List

System Tasks	Duration	Dependent
Design CVideo Class	3 Days	Design Document
Code PC CVideo Class	4 Days	CVideo Class designed
Code MAC CVideo Class	4 Days	PC CVideo Class coded
Integrate CVideo Class	2 Days	CVideo Class coded
Test & Revise CVideo Class	1 Day	CVideo Class integrated
Rework #1 CVideo Class	1 Day	As Needed
Test & Revise CVideo Rework #1	1 Day	CVideo Class Reworked #1
Rework #2 Cvideo Class	1 Day	As Needed
Test & Revise CVideo Rework #2	1 Day	CVideo Class Reworked #2
Design CBitMap Class	1 Day	Design Document
Code PC CBitMap Class	2 Days	CBitMap Class designed
Code MAC CbitMap Class	2 Days	PC CBitMap Class coded
Integrate CbitMap Class	1 Day	CBitMap Class coded
Test & Revise CBitMap Class	1 Day	CBitMap Class integrated
Rework #1 CbitMap Class	1 Day	As Needed
Test & Revise CBitMap Rework #1	1 Day	CBitMap Class Reworked #1
Rework #2 CBitMap Class	1 Day	As Needed
Test & Revise CBitMap Rework #2	1 Day	CBitMap Class Reworked #2
Design CSprite Class	1 Day	Design Document
Code CSprite Class	2 Days	CSprite Class designed
Integrate CSprite Class	1 Day	CSprite Class coded
Test & Revise CSprite Class	1 Day	CSprite Class integrated
Rework #1 CSprite Class	1 Day	As Needed
Test & Revise CSprite Rework #1	1 Day	CSprite Class Reworked #1
Rework #2 CSprite Class	1 Day	As Needed
Test & Revise CSprite Rework #2	1 Day	CSprite Class Reworked #2
Total	38 Days	

Memory

Besides the class size, which is relatively small, the CBitMap class uses a pixel buffer that is loaded into system RAM, or VRAM, for image data. The only other memory is the instance of the various classes. This should amount to less than 1K of RAM for class overhead and width*height*2 per bitmap. A typical scene might use approximately 20, 64x64x2 images resulting in 163860 bytes of memory.

The CVideo class buffers are in VRAM and are included in the master memory map.

Risk Assessment

A real risk with the graphics system is that it could be very slow or take too much RAM. A significant and necessary amount of time has been allocated to make this system a reliable and fast as possible. Because the base engineering team has no practical experience with MAC graphics we are basing our estimates on the descriptions in the Apple Developers Guides. Hiring the MAC consultant will work to lessen this risk.

QA & Test

The graphics system is a core technology and will have several test/rework cycles. The QA department should pay special attention to the visual look of the on screen objects, the clipping behaviors at the edge of the screen, and the video frame rate on our minimum systems.