Particle System Design Document, v2

Randy Angle

October 19, 1999

Introduction

Provides animation without incurring additional art assets. This system allows for visual effects involving particles using simple 2D algorithms and small 2D art assets. The sprites will be kept in VRAM when possible and copied to the screen as quickly as possible. The algorithm will have several dial-able properties and can change over time.

Requirements

Spawning particle system creates a new instance of a particle class. It will use pre-loaded assets and a procedural path to display a unique animation. It must be able to sustain the drawing of the individual elements and calculate the movement using simple methods:

```
pParticle = new CParticle;
pParticle->Spawn(EPSPRITE_STAR,EPSTYLE_TORNADO,NumParticles,Speed,Xpos,Ypos,BoundingBox);
pParticle->NewOrigin(x+dx,y+dy);
pParticle->NewBox(BoundingBox);
pParticle->NewStyle(EPSTYLE_RAIN);
pParticle->NewSprite(EPSPRITE_DROP);
pParticle->NewNumber(NumParicles-10);
pParticle->NewSpeed(Speed+5);
pParticle->Update();
pParticle->Draw();
delete pParticle;
```

Particles always happen in a plane in front of the characters but behind cursors and drag objects. They can move in any direction and will be clipped if they would be drawn off screen, or outside of the bounding box. Any single particle system should be able to support up to 50 particles.

This module uses the graphics system. While very similar to the CSprite class, it may have features that would make it unique. Particle objects are attached to the Scene database as they are created.

Structures/Classes

```
typedef enum
{
        PSPRITE_STAR=0,
        PSPRITE_DROP=3,
        PSPRITE_MOON=6,
        PSPRITE_HEART=9,
        PSPRITE_CLOVER=12
} EPSPRITE;

typedef enum
{
        PSTYLE_TORNADO=0,
        PSTYLE_RAIN,
        PSTYLE_EXPLOSION,
        PSTYLE_FIREWORKS,
        PSTYLE_BUBBLES,
        PSTYLE_BUGS
} EPSTYLE;

class CParticle
{
private:
        CVRAMSprite    *pParticleList;
        long           ParticleX;
        long           ParticleY;
        int            ParticleSpeed;
        RECT           BoundingBox;
        Int            NumParticles;
        EPSPRITE       SpriteIndex;
        EPSTYLE        ParticleStyle;

public:
        CParticle();
        ~CParticle();
        Spawn(  EPSPRITE SpriteIndex,
                EPSTYLE ParticleStyle,
                int NumParticles,
                int Speed,
                long Xpos,
                long Ypos,
                RECT BoundingBox);
        NewOrigin(long Xpos, long Ypos) {ParticleX = Xpos; ParticleY = Ypos;};
        NewBox(RECT Box) {BoundingBox = Box;};
        NewStyle(EPSTYLE Style) {ParticleStyle = Style;};
        NewSprite(EPSPRITE SpriteIndex) {SpriteIndex = Index;};
        NewNumber(int Num) {NumParticles = Num;};
        NewSpeed(int Speed) {ParticleSpeed = Speed;};
        Update(void);
        Draw();
}
```

Schedule Task List

System Tasks	Duration	Dependent
Design Particle System	1 Day	Design Document
Code Particle System	3 Days	Particle Class designed
Integrate Particle System	1 Day	Particle Class coded
Test & Revise Particle System	1 Day	Particle System integrated
Rework #1 Particle Class	1 Day	As Needed
Test & Revise Particle Rework #1	1 Day	Particle Class Reworked #1
Rework #2 Particle Class	1 Day	As Needed
Test & Revise Particle Rework #2	1 Day	Particle Class Reworked #2
Total	10 Days	

Memory

The particle system uses a bitmap that is loaded into VRAM for sprite data. The only other memory is the instance of the particle system and the specific sprites for each individual particle. This should amount to less than 5K of RAM per system.

Risk Assessment

The real risk with the particle system is that it could be very slow to have multiple particle systems on screen at once. Every effort will be made to optimize the performance so that it is possible to have at least three systems of approximately 20 particles each. If necessary the number of particles could be fixed and the system would dynamically alter the number of visible particles each frame.

QA & Test

The particle system relies on much of the planned sprite technology. All the QA department should concern themselves with is does the presence of particles slow the game too much and is the correct sprite and style displayed on the correct events.