Computer Graphics and Programming

Lecture 13
Particle Effect

Jeong-Yean Yang
2020/12/8

Our World can NOT be Perfectly Modeled

- We start to understand our world by Ray tracing
  - One pixel color is the combination of many lights interaction
  - Simple Modeling implies Simple Result (Not Realistic)

- World is filled with Stochastic Processes
  - From the viewpoint of Ray Tracing, We omitted dusts in air.
  - Dust is modeled by Probabilistic Distribution as in Probabilistic Robotics ^^

- From the Multiple Objects Interaction,
  - The Concept of “Particle Effect” arises.

Our World works under Quantum Computing

- The basic of Material meets Quantum
  - Higgs-Boson is an elementary particle.
  - Quantum excitation of Higgs field, and so on.
- Small sized Object interacts with others
  - Cells Vs. Human
- Fluid dynamics and Heat transfer is modeled by mathematics.
  - But, boundary condition is simplified in many cases.
  - Explicit model Vs. Implicit model

Particle Effect

- Fire or Fluid movement is defined as
  - the integration of many Particles or elements.
- Multiple Object with **Random Movements** looks Realistic.
- Each particle has some attributes
  - Life cycle, Color variance, Size, moving direction
- Each particle works during its Life then it dies
  (After the death, a particle starts a new Life)

Example of "Fire" with uParticle

- uParticle is composed of 300 objects

```
class uParticle : public uObj
{
public:
    uParticle();
    ~uParticle();

public:
    void    Create(int n);
    void    Set(uParticle*);
    virtual void    Run();

protected:
    float   Random(float variance);

public:
    int     nLife, mLife, vLife;
    uVector dir, mdir,vdir;
    uVector pos, mpos,vpos;

    uColor color, mcolor, vcolor;
    float  scale,mscale,vscale;
    float  dscale,dmscale,dvscale;

public:
};
```

- uParticle is inherited by uObj
- Random with mean and variance

$$X \sim N(\mu, \sigma^2)$$

$$\mu : mean$$

$$\sigma^2 : \text{variance}$$

- mXXX → mean of XXX
- vXXX → variance of XXX

Particle is RECREATED after Death
uParticle::Set()

```
void uParticle::Set(uParticle *p)
{
    if (p->nLife<=0)
    {
        // Generate life
        p->nLife = rand()%vLife + mLife;

        // Generate Pos
        float x,y;
        x = Random(vpos.x)+ mpos.x;
        y = Random(vpos.y)+ mpos.y;
        p->Trans(x,y,0);

        p->dir.x = Random(vdir.x)+ mdir.x;
        p->dir.y = Random(vdir.y)+ mdir.y;
        p->dir.z = 0;

        p->color.r  = Random(vcolor.r)+mcolor.r;
        p->color.g  = Random(vcolor.g)+mcolor.g;
        p->color.b  = Random(vcolor.b)+mcolor.b;
        p->color.a  = Random(vcolor.a)+mcolor.a;

        p->color.r = MAX(0,p->color.r);
        p->color.g = MAX(0,p->color.g);
        p->color.b = MAX(0,p->color.b);
        p->color.a = MAX(0,p->color.a);

        p->color.r = MIN(1,p->color.r);
        p->color.g = MIN(1,p->color.g);
        p->color.b = MIN(1,p->color.b);
        p->color.a = MIN(1,p->color.a);

        if (vscale>0)   p->scale = Random(vscale)+mscale;
        else            p->scale = mscale;
        p->diffuse  = p->color;
    }
```

Function Random returns -1.0~ 1.0

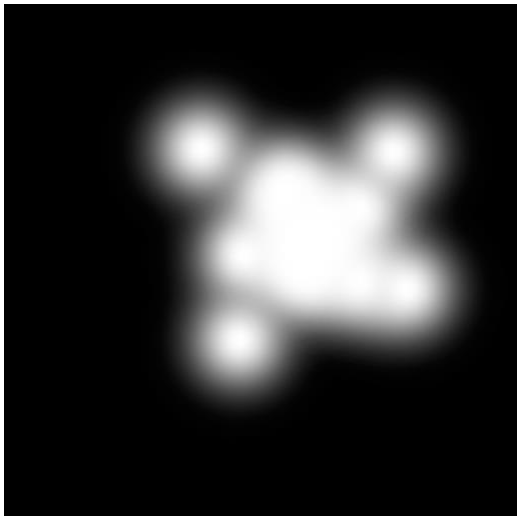
Position is varying
-vpos + mpos< Position< vpos+mpos

Template for Particle Effect :uGL-42-ParticleEffect

- GLSL is simple.
  - Shader.fsh and shader.vsh for only 2D texture mapping
- Gaussian distribution like texture named “gauss.png” is used for Every Particle

Gauss.png

300 particles
are overlapped



Particle effect

Key Point of Particle Effect is Overlapped Images

- OpenGL has Blending function for Mixing Images

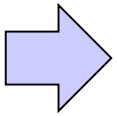
```
void uWnd::Draw()
{
    glClearColor(info.r,info.g,info.b,info.a);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);


    // draw objects
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

    glDisable(GL_DEPTH_TEST);
    glEnable( GL_BLEND );

    for (int i=0;i<objs.GetSize();i++)
    objs[i]->Draw();

    glEnable(GL_DEPTH_TEST);
    glDisable( GL_BLEND );

    glFinish();
}
```

Enable Blending

Disable Blending

Particle Works by Timer function

```
void uWnd::OnTimer(UINT_PTR nIDEvent)
{
    if (nIDEvent==1)
    {
        for (int i=0;i<objs.GetSize();i++)
        objs[i]->Run();
    }
    uGL::OnTimer(nIDEvent);
}
```



```
void uParticle::Run()
{
    int n = child.GetSize();
    for (int i=0;i<n;i++)
    {
        uParticle *p = (uParticle*)child[i];
        if (p->nLife<=0)
            Set(p);

        p->Run();
    }

    // particle move
    nLife--;
    uVector p = coord.o;
    p = p+ dir;

    //diffuse.a = diffuse.a*0.97;

    H.I();
    Trans(p);
    H = H.Scale(scale)*H;
}
```

Particle's life is reduced in every turn

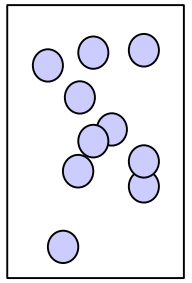
Particle moves toward some direction

Particle's size is varied

Ex1) Fire with Fixed Scale Particle

- Color =[R,G,B,A], Alpha, A is used for Blending.

```
p->vpos      = uVector(5,10,0);

p->vdir      = uVector(.1,.3,0);
p->mdir      = uVector(0,0.5,0);
p->vcolor    = uColor(0.3,0.1,0.1,0.1);
p->mcolor    = uColor(0.8,0.1,0,0.8);
p->mscale    = 0.5;
p->vscale    = 0;

p->Create(300);
```



20

10

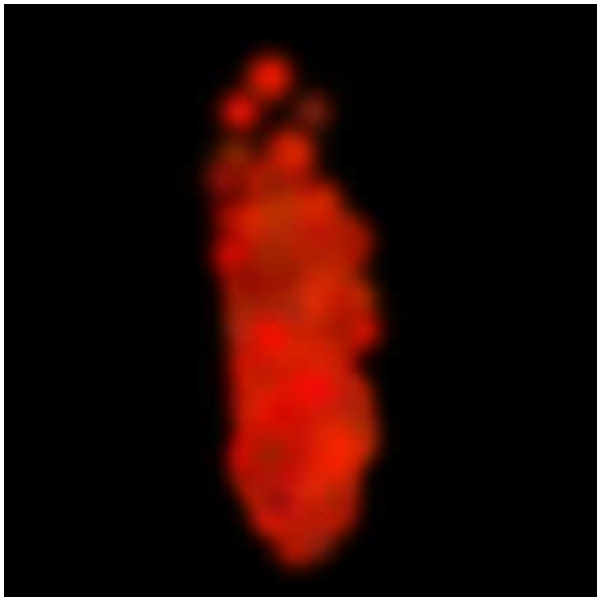
Particle's position will be limited with width=5 and height =10

Mean color is red

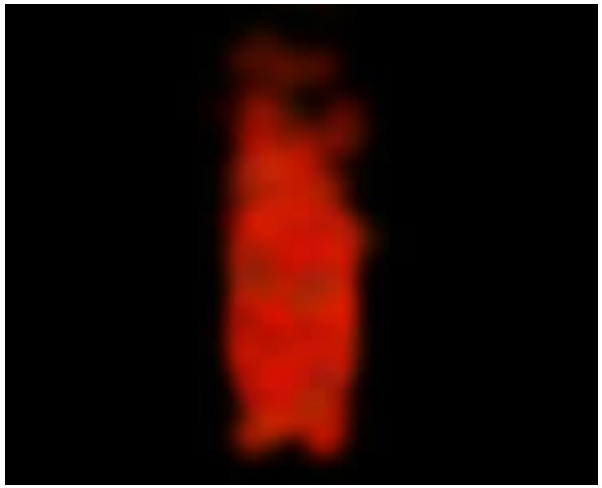
Red Color variance is higher.

Fire without Scaling Vs with Scaling

Create particle with a variety of scale factor

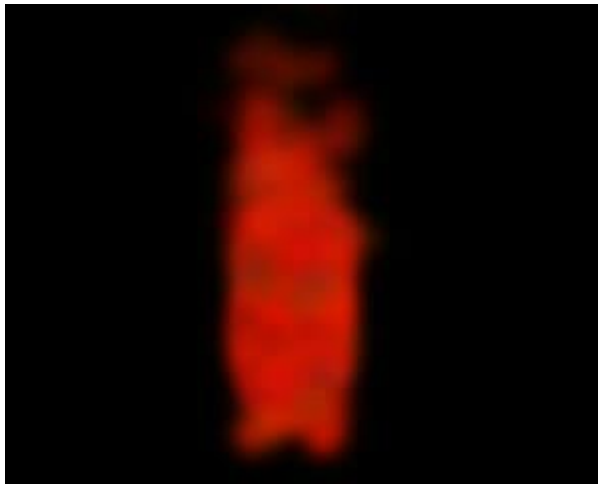


Scale is fixed
after creation

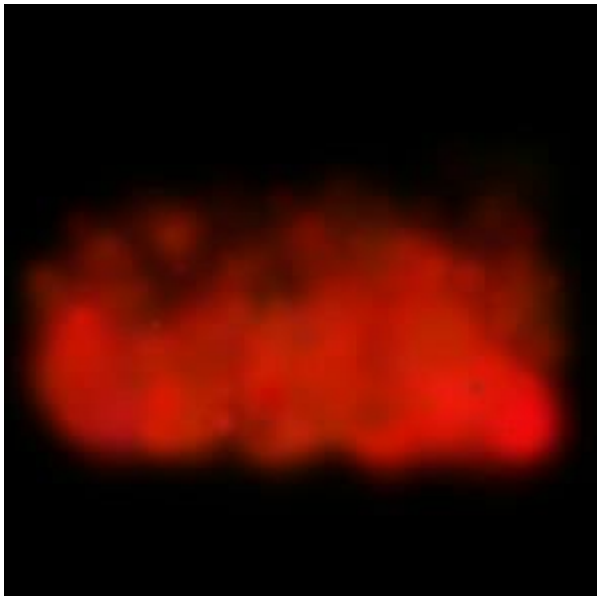


Scale is controlled
by life proceeding

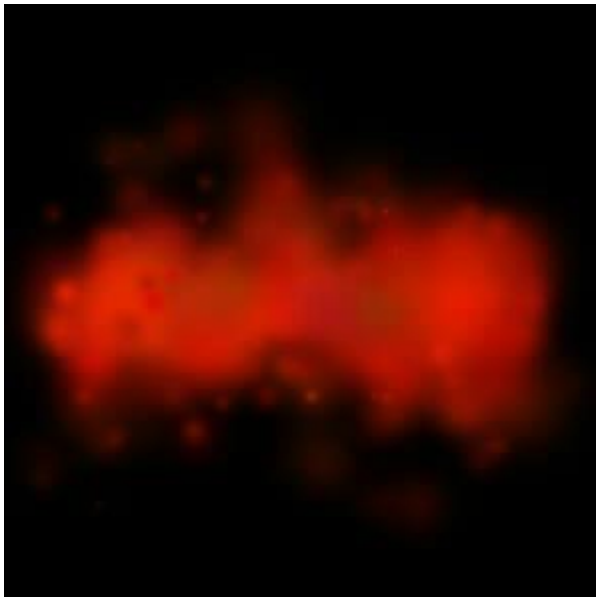
Ex 2) Initial Position and Direction Changes



Upward direction



Higher Position variance
+
Positive Y Direction



Positive and negative y direction variance

Thank you for Listening Graphics Class

Welcome to Homework Hell