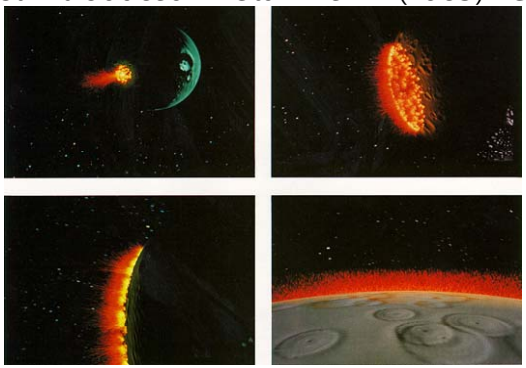
Particle Systems

305890
Spring 2011
5/23/2011
Kyoung Shin Park
kpark@dankook.ac.kr

Star Trek II (1983)

- Particle Systems can be utilized to simulate a wide range of phenomena such as fire, rain, smoke, explosions, and projectiles.
- It was first introduced in Star Trek II (1983) "Genesis Effect"

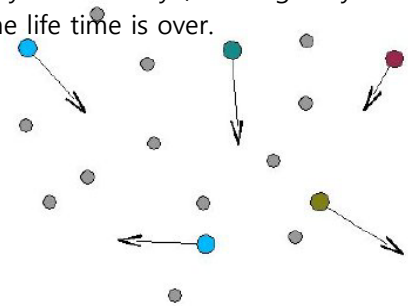


Particle Systems

- Particles and Point Sprites
- Particle System Components
- Particle Systems
  - Snow/Rain
  - Explosions
  - Smoke

Particles and Point Sprites

- Particle
  - A very small object that is usually modeled as a point mathematically.
  - Represented as a point with position, velocity, and color.
  - Each particle can be added in the particle system and moved in a physically realistic way (due to gravity or wind) and then died after the life time is over.



Particles and Point Sprites

- Particle
  - Use a **Point sprite** to represent a particle.
  - With point sprites (introduced in DirectX 8), graphics card only needs only one vertex for each particle.
    - Prior to point sprites, game developers had to create a rectangle (quad) and apply a texture to it for each particle.
  - Point sprites can be as small as one pixel to as large as we want
    - We can set each particle to be a different texture and/or size.
  - XNA automatically maps our texture to the vertex. It also makes sure that the texture is always facing the camera.

Particles and Point Sprites

- Creating the Particle Class

```
public class Particle {
    private Vector3 velocity;           // 속도
    private Vector3 acceleration;       // 가속도
    private float lifetime;             // 수명
    private Vector3 externalForce;      // 외부힘
    internal float Age;                 // 파티클이 없어져야할 지 결정
    internal bool IsAlive;              // 현재 살아있는지?
    internal VertexPointSprite Vertex;// 포인트 스프라이트
    internal float ColorChangeRate;
    internal float CurrentColorTime;
    internal int CurrentColorIndex;
 ..}
```

Particles and Point Sprites

```
public Particle() {
    Age = 0.0f;
    Vertex = new VertexPointSprite();
    CurrentColorTime = 0;       CurrentColorIndex = 0;
}
internal void Update(float elapsedTime) {
    Age += elapsedTime;
    CurrentColorTime += elapsedTime;
    if (Age >= lifetime)
      IsAlive = false;                          // 수명을 다했으면 죽음
    else {
      velocity += acceleration;                 // 속도업데이트
      velocity -= externalForce;                // 속도업데이트
      Vertex.Position += velocity * elapsedTime;  // 위치업데이트
    }
}
```

Particles and Point Sprites

```
// 파티클 초기화 메소드
internal void Initialize(ParticleSystemSettings settings, bool makeAlive) {
    Age = 0;
    IsAlive = makeAlive;
    // 랜덤 위치
    Vector3 minPosition = (settings.EmitPosition - (settings.EmitRange * .5f));
    Vector3 maxPosition = (settings.EmitPosition + (settings.EmitRange * .5f));
    Vector3 position = Utility.GetRandomVector3(minPosition, maxPosition);
    Vertex.Position = position;
    if (settings.EmitRadius != Vector2.Zero) {
            float angle = Utility.GetRandomFloat(0, MathHelper.TwoPi);
            Vertex.Position = new Vector3(
                position.X + (float)Math.Sin(angle) * settings.EmitRadius.X,
                position.Y,
                position.Z + (float)Math.Cos(angle) * settings.EmitRadius.Y);
    }
```

Particles and Point Sprites

```
    velocity = Utility.GetRandomVector3(                // 랜덤 속도
            settings.MinimumVelocity, settings.MaximumVelocity);
    acceleration = Utility.GetRandomVector3(            // 랜덤 가속도
            settings.MinimumAcceleration, settings.MaximumAcceleration);
    lifetime = Utility.GetRandomFloat(                  // 랜덤 수명
            settings.MinimumLifetime, settings.MaximumLifetime);
    if (settings.DisplayColorsInOrder) {
            Vertex.Color = settings.Colors[0];
            ColorChangeRate = lifetime / settings.Colors.Length;
    } else {
            Vertex.Color =                              // 랜덤 색
               settings.Colors[Utility.GetRandomInt(0, settings.Colors.Length)];
    }
    Vertex.PointSize =                                  // 랜덤 크기
          Utility.GetRandomFloat(settings.MinimumSize, settings.MaximumSize);
     externalForce = settings.ExternalForce;  // 중력이나 바람과 같은 외부힘
}
```

Particles and Point Sprites

- Creating the VertexPointSprite structure
  - If we only needed one size for all our particles, we could just use the VertexPositionColor struct already provided by XNA.

```
public struct VertexPointSprite {
    public Vector3 Position;  // 포인트스프라이트 위치
    public float PointSize;     // 포인트스프라이트 크기
    public Color Color;         // 포인트스프라이트 색
    public VertexPointSprite(Vector3 Position, Color Color, float
    PointSize) {
            this.Position = Position;
            this.Color = Color;
            this.PointSize = PointSize;
        }
...}
```

Particle System

- Particle System Engine
  - Particle system class manages the multiple particles' creation, initialization, remove, update, and draw.
  - This class will be an abstract class that inherits from the DrawableGameComponent class.
  - Each individual particle system (rain, bubbles, gas) will inherit from this base ParticleSystem class.

Particle System

```
// 전형적인 particle system을 일반화한 base class
pubic abstract class ParticleSystem : Microsoft.Xna.Framework.DrawableGameComponent {
    private Particle[] particles;                   // 파티클 배열
    private int lastParticleIndex = 0;
    private Effect effect;
    private VertexDeclaration vertexDeclaration;
    private VertexPointSprite[] vertices;
    private SpriteFont font;
    private SpriteBatch sb;
    private Rectangle titleSafeArea;
    private float rotateAngle = 0;                  // 회전각도를 0으로 초기화
    private int totalParticlesEmitted = 0;          // 배출되는 파티클수
    private int numberOfActiveParticles;            // 활성화된 파티클수
    protected ParticleSystemSettings settings;
    protected ContentManager content;
    public Matrix View;
    public Matrix Projection;
... }
```

Particle System

```
public ParticleSystem(Game game) : base(game) {
        content = new ContentManager(game.Services);
        settings = new ParticleSystemSettings();            // 1개 파티클 속성
}
public override void Initialize() {
        settings = InitializeSettings();                    // 파티클마다 셋팅초기화
        particles = new Particle[settings.Capacity];
        vertices = new VertexPointSprite[settings.Capacity];
        for (int i = 0; i < settings.Capacity; i++)  {
            particles[i] = new Particle();
            particles[i].Initialize(settings, false);       // 파티클마다 초기화
        }
        numberOfActiveParticles = 0;
        base.Initialize();
}
protected override void LoadContent()    {       // .... 생략
   effect = content.Load<Effect>(@"Content₩Effects₩PointSprites");
   vertexDeclaration = new VertexDeclaration(GraphicsDevice, VertexPointSprite.VertexEle
}
```

Particle System

```
public override void Update(GameTime gameTime)  {
    float elapsedTime = (float)gameTime.ElapsedGameTime.TotalSeconds;
    int particlesToEmitThisFrame = (int)(settings.EmitPerSecond * elapsedTime + .99);
    int particlesEmitted = 0;
    bool canCreateParticle;
    for (int i = 0; i < particles.Length; i++)    {
        canCreateParticle = false;
        if (particles[i].IsAlive) {
            particles[i].Update(elapsedTime);
            if (!particles[i].IsAlive)  {
                numberOfActiveParticles--;
                particles[i].CurrentColorTime = 0;
                particles[i].CurrentColorIndex = 0;
                canCreateParticle = ShouldCreateParticle(particlesEmitted, particlesToEmitThisF
            } else {
                if (settings.DisplayColorsInOrder) {
                    if (particles[i].CurrentColorTime > particles[i].ColorChangeRate) {
                            //due to rounding errors with floats we need to make sure
                            //we actually have another color
```

Particle System

```
                            //we actually have another color
                            if (particles[i].CurrentColorIndex < settings.Colors.Length – 1) {
                                particles[i].CurrentColorIndex++;
                                particles[i].SetColor(settings.Colors[particles[i].CurrentColorIndex])
                                particles[i].CurrentColorTime = 0;
                            }
                        }
                    }
                }
            } else { // 파티클이 살아있지않다면 새로 생성할지 여부를 판단
                canCreateParticle = ShouldCreateParticle(particlesEmitted, particlesToEmitThis
            }
            if (canCreateParticle) {
                particles[i] = CreateParticle();       // 새로운 파티클 생성
                particlesEmitted++;
                numberOfActiveParticles++;
                totalParticlesEmitted++;
            }
        } // 생략 ... }
```

Particle System

```
protected virtual void SetBlendModes() {
    GraphicsDevice.RenderState.AlphaBlendEnable = true;
    GraphicsDevice.RenderState.SourceBlend = Blend.SourceAlpha;
    GraphicsDevice.RenderState.DestinationBlend = Blend.InverseSourceAlpha;
}
public override void Draw(GameTime gameTime)  {
    GraphicsDevice.RenderState.PointSpriteEnable = true;
    GraphicsDevice.RenderState.DepthBufferWriteEnable = false;
    SetBlendModes();            // render states
    GraphicsDevice.VertexDeclaration = vertexDeclaration;
    PopulatePointSprites();
    if (numberOfActiveParticles > 0) {
        effect.Parameters["View"].SetValue(View);
        effect.Parameters["Projection"].SetValue(Projection);
         // 중간 생략..
    }
    base.Draw(gameTime);
}
```

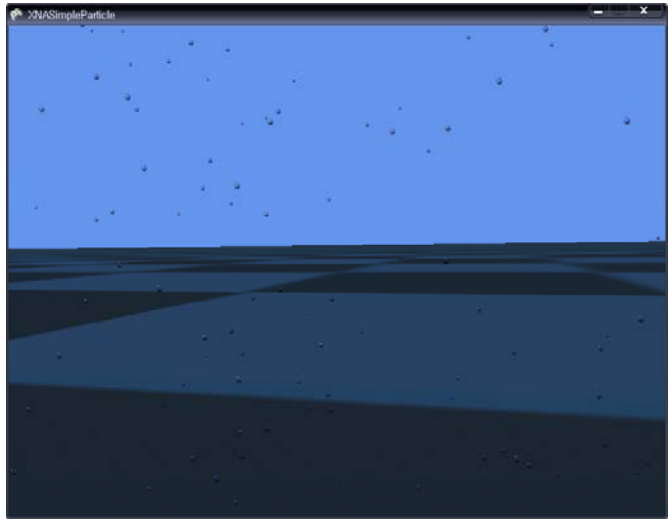
Rain

```
public class Rain : ParticleSystem  {
    public Rain(Game game, int capacity, Vector3 externalForce) : base(game)  {
        settings.Capacity = capacity;
        settings.ExternalForce = externalForce;
    }
    protected override ParticleSystemSettings InitializeSettings() {
        settings.EmitPerSecond = 1000;          // 초당 1000 파티클 방출
        settings.EmitPosition = new Vector3(0, 4000, 0);
        settings.EmitRange = new Vector3(4000, 0, 4000);
        settings.MinimumVelocity = new Vector3(0, -10, 0);
        settings.MaximumVelocity = new Vector3(0, -50, 0);
        settings.MinimumAcceleration = new Vector3(0, -10, 0);
        settings.MaximumAcceleration = new Vector3(0, -10, 0);
        settings.MinimumLifetime = 5.0f;   settings.MaximumLifetime = 5.0f;
        settings.MinimumSize = 5.0f;    settings.MaximumSize = 15.0f;
        settings.Colors = new Color[] { Color.CornflowerBlue, Color.LightBlue };
        settings.DisplayColorsInOrder = false;
        return (settings);
    }
```

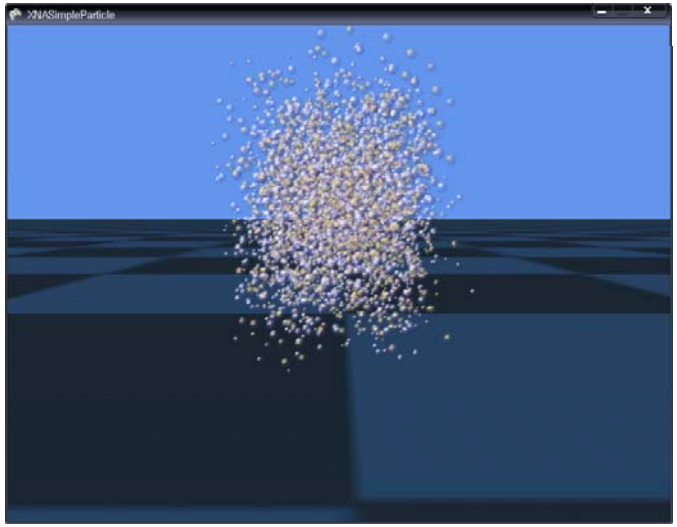
Rain

```
        protected override void LoadContent()  {
            base.SetTexture(base.content.Load<Texture2D>(@"Content₩Textures₩raindrop"));
            base.LoadContent();
        }
    }
// 파티클 데모 프로그램
public class Game1 : Microsoft.Xna.Framework.Game
{
    private Rain rain;          // 비 파티클 생성
    public Game1() {
        // 중간 생략..
        rain = new Rain(this);
        Components.Add(rain);
    }
    protected override void Draw(GameTime gameTime) {
        rain.View = camera.View;
        rain.Projection = camera.Projection;
    }
}
```

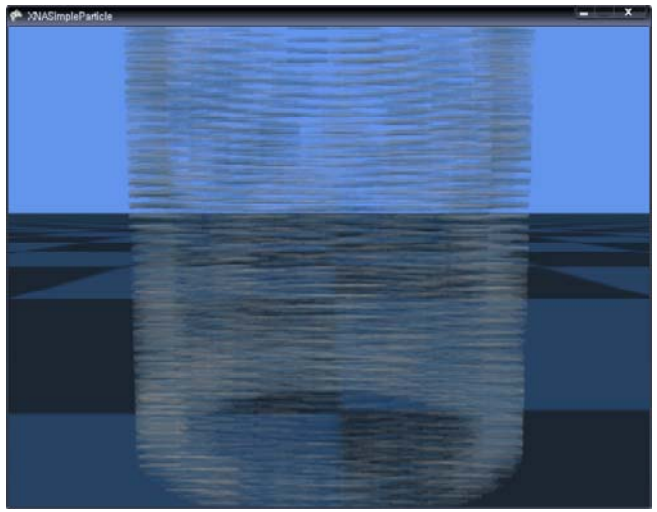
Example: Rain



Example: Bubbles



Example: Shield



Example: Gas