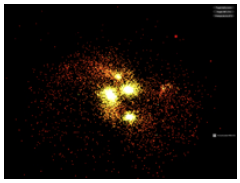
NBodyGravity Sample

Collapse All

This is one of 3 sample applications shown during the Advanced Real-Time Rendering in 3D Graphics and Games course at SIGGraph 2007. The Direct3D 10 sample shows N-Body particles system managed entirely by the GPU.



Path

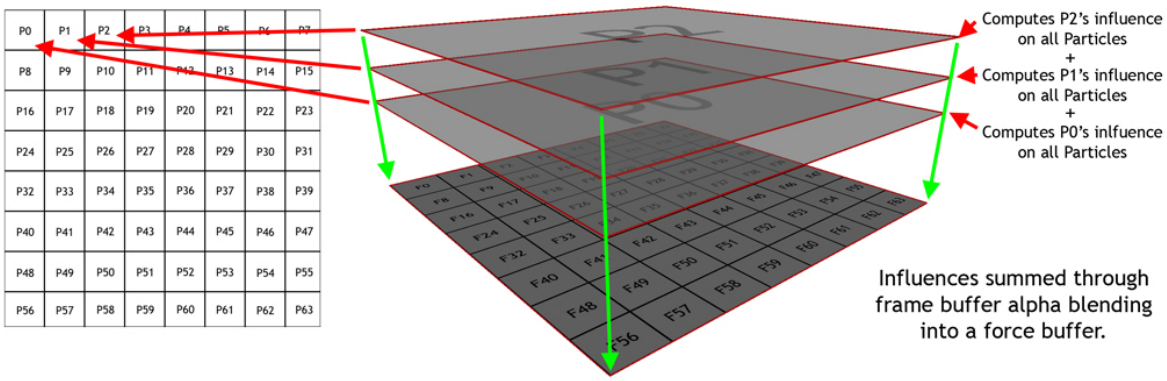
Source	*SDK root* \Samples\C++\Direct3D10\NBodyGravity
Executable	*SDK root* \Samples\C++\Direct3D10\Bin\ *x86 or x64* \NBodyGravity.exe

How the Sample Works

The sample calculates the interaction between all particles in the system. In this case, the interaction involves the influence of gravity between all possible particle pairs. The calculate the N*N interactions, we use a technique called force splatting.

Force Splatting

The goal of force splatting is to project the force from one particle onto all other particles during a single operation. In this case, the operation is the rendering of a quad primitive. We create a texture that acts and an accumulation buffer for all forces applied to the particles. This buffer will be the target of the rasterization operations that will accumulate particle forces. Each texel in the force texture holds the accumulated forces acting upon a single particle. We also create a stack of N quad primitives, where N is the number of particles in the system. The dimensions of dimensions of the quads are such that they will exactly cover the force buffer when rasterized. The four vertices of each quad in the stack contain a vertex element which identifies the exact particle represented by the quad. During rasterization, this interpolated vertex element is used to fetch properties of the particle from the particle texture or the particle buffer.



During the rasterization of a single quad, the forces are calculated between the particle being rasterized to and the particle represented by the vertex element in the vertices of the quad. Forces are accumulated by rendering successive quad with additive alpha blending enabled.

```
float4 PSAccumulateForce(PSForceIn input) : SV_Target
{
    float3 inplacePos = g_txParticleData.SampleLevel( g_samPoint,
                                                      float3(input.tex, 0),
                                                      0 );

    // use the law of gravitation to calculate the force of the input
    // particle on the particle we're rasterizing across at this moment
    float3 delta = input.pos - inplacePos;
    float r2 = dot( delta, delta );

    float4 force = float4(0,0,0,0);
    if( r2 > 0 )
    {
        // Calculate acceleration between the two caused by gravity
        float r = sqrt(r2);
        float3 dir = delta/r;
        force.xyz = dir * g_fG * ( (g_fParticleMass * g_fParticleMass) / r2 );
    }

    return force;
}
```

While inelegant less than elegant in terms of algorithmic complexity, the force splatting algorithm exploits the fast rasterization and alpha blending capabilities of modern graphics hardware without the need to continually recreate complex space partitioning structures on the GPU.

Force Splatting

To compute the gravitational force of all particles to all other particles, we use the method of force splatting mentioned above to accumulate all of the forces imparted on each particle in the system. In the particle update phase, this force is divided by the particle mass to determine the instantaneous acceleration of the particle. The equations of motion are integrated, and the particle system is updated.