Programming Graphics Hardware

Cloth Simulation on GPU

Cyril Zeller

NVIDIA®

Cloth as a Set of Particles

- A cloth object is a set of particles
- Each particle is subject to:
  - A force (gravity or wind)
  - Various constraints:
    - To maintain overall shape (springs)
    - To prevent interpenetration with the environment
- Constraints are resolved by relaxation

- For more details, see:

  Jakobsen, T. "Advanced character physics", *GDC 2001*

Force

- The equation of motion for each particle at position P(t) subject to force F(t) is integrated using Verlet integration:

$$P(t + \Delta t) = P(t) + k\,(P(t) - P(t - \Delta t)) + \Delta t^2\, F(t) / m$$

  Δt is the simulation time step

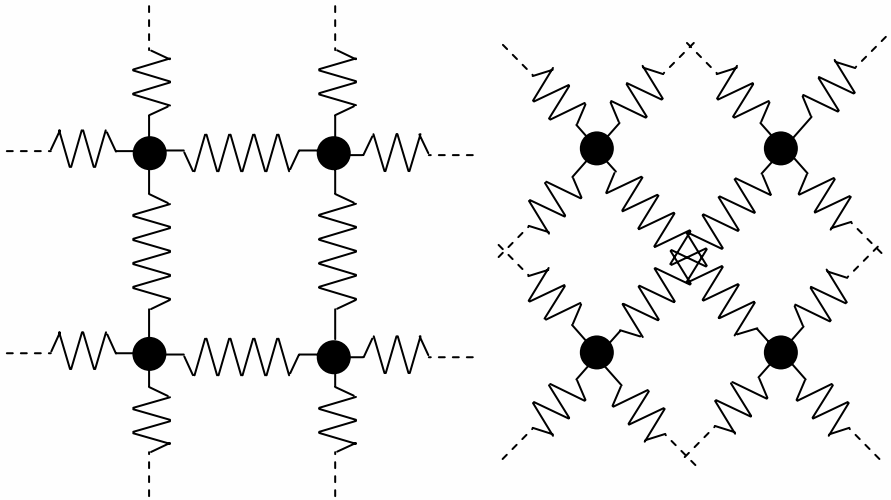
  k is an arbitrary damping coefficient very close to 1

  m is the mass of the particle

- No force is applied to fixed or user-moved particles

Distance Constraints

- **Particles are linked by springs:**



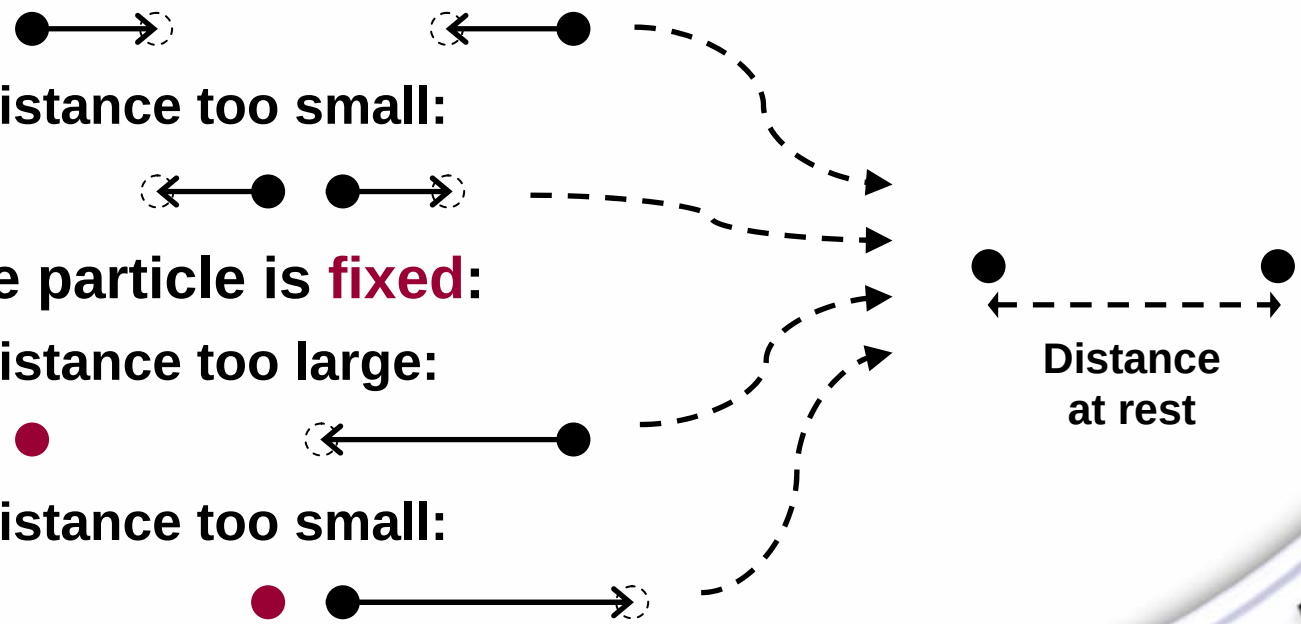
Structural springs

Shear springs

- **A spring is simulated as a distance constraint between two particles**

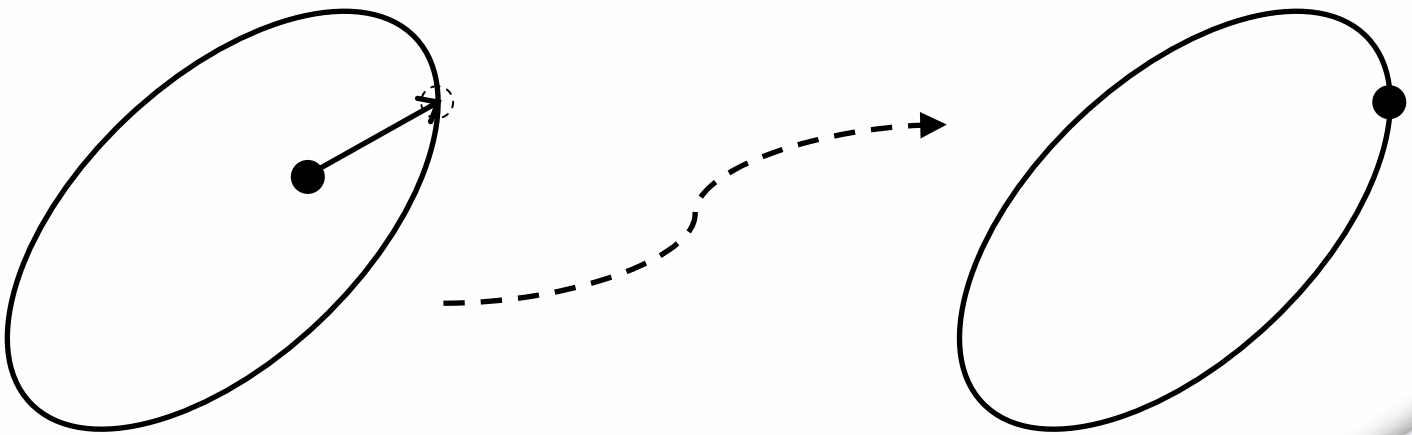
Distance Constraints

- A distance constraint between two particles is enforced by moving them away or towards each other:
  - If both particles are free:
    - Distance too large:
    - Distance too small:
  - If one particle is fixed:
    - Distance too large:
    - Distance too small:



Collision Constraints

- The environment is defined as a set of collision objects (planes, spheres, boxes and ellipsoids)
- A collision constraint between a particle and a collision object is enforced by moving the particle outside the object



Algorithm Outline

- For every simulation time step:
  - For every particle that isn't fixed or user-moved:
    - Apply force
  - For every relaxation step:
    - For every distance constraint:
      - Reposition particles
    - For every particle:
      - For every collision object:
        - If the particle is inside, move the particle out of the object

GPU Implementation

- The particle positions and normals are stored into floating-point textures PositionTex and NormalTex
  - The CPU never reads the content of these textures!
- At every frame:
  - GPU simulation: Compute PositionTex and NormalTex
  - Rendering:

Vertex buffer contains only texture coordinates

```
void VertexShader(float2 texCoord)
{
  position = tex2Dlod(PositionTex, texCoord);
  normal   = tex2Dlod(  NormalTex, texCoord);
   ...
}
```

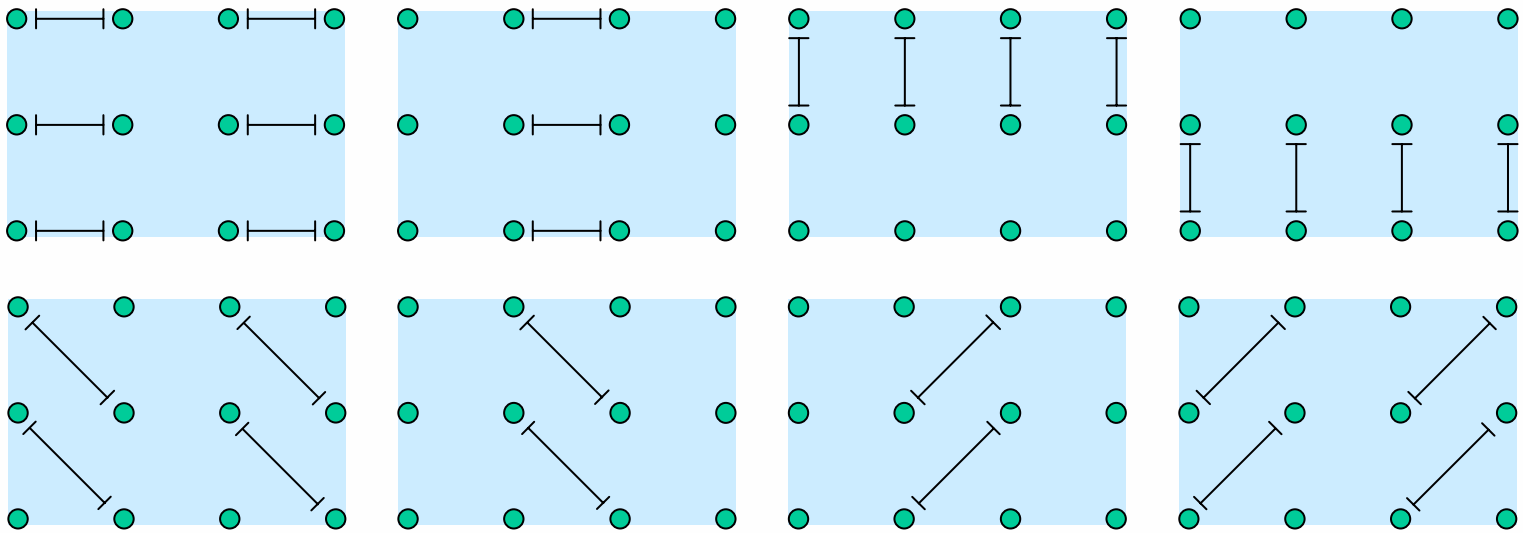
GPU Simulation: Force

- Three floating point textures:
  - OldPositionTex
  - CurrentPositionTex
  - NewPositionTex (used as render target)
  - Rotated after each draw call
- Draw fullscreen quad with:

```
float4 PixelShader(float2 texCoord)
{
  float3 oldPos     = tex2D(    OldPositionTex, texCoord);
  float3 currentPos = tex2D(CurrentPositionTex, texCoord);
  float3 force = ComputeForce();
  float3 newPos = currentPos
                + 0.99 * (currentPos – oldPos)
                + force * TimeStep * TimeStep; // Verlet
  return newPos;
}
```

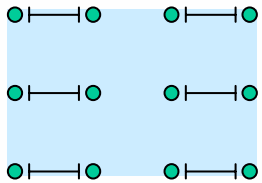
GPU Simulation: Distance Constraints

- **8 fullscreen quad draw calls to simulate the 8 springs attached to a particle:**



GPU Simulation: Distance Constraints

- **Stiffness** coefficients stored into 2D textures to deal with:
  - Cloth boundary
  - Fixed particles
  - Cut springs
- Fullscreen quad draw call for:



```
float4 PixelShader(float2 texCoord, float2 pixel : VPOS)
{
  float2 offset      = float2((pixel.x % 2 ? – Dx : Dx), 0);
  float3 currentPos  = tex2D(CurrentPositionTex, texCoord);
  float3 neighborPos = tex2D(CurrentPositionTex, texCoord + offset);
  float3 delta       = neighborPos – currentPos;
  float  dist        = length(delta);
  float  stiffness   = tex2D(StiffnessTex, texCoord);
  float3 newPos = currentPos
                + stiffness * (1 – DistanceAtRest / d) * delta;
  return newPos;
}
```

Equal to either 0.5 or 0

GPU Simulation: Collision Constraints

- Collision objects stored into 1D textures
  - 1 texture per geometric type
  - Can't index constant registers
- Draw fullscreen quad with:

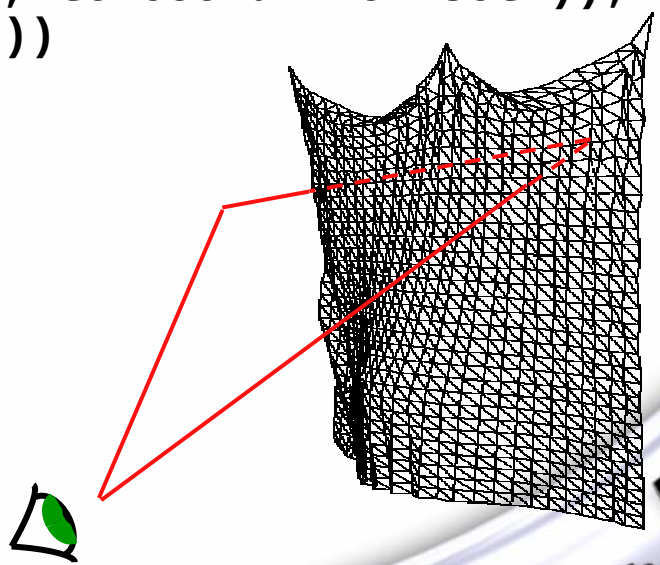
```
float4 PixelShader(float2 texCoord)
{
  float3 currentPos = tex2D(CurrentPositionTex, texCoord);
  for (int i = 0; i < NumPlanes;     ++i) ...
  for (int i = 0; i < NumSpheres;    ++i) {
    float4 sphere = tex1D(SphereTex, Ds * i);
    currentPos += SphereConstraint(currentPos, sphere);
  }
  for (int i = 0; i < NumBoxes;      ++i) ...
  for (int i = 0; i < NumEllipsoids; ++i) ...
  return currentPos;
}
```

GPU Simulation: Cloth Cutting

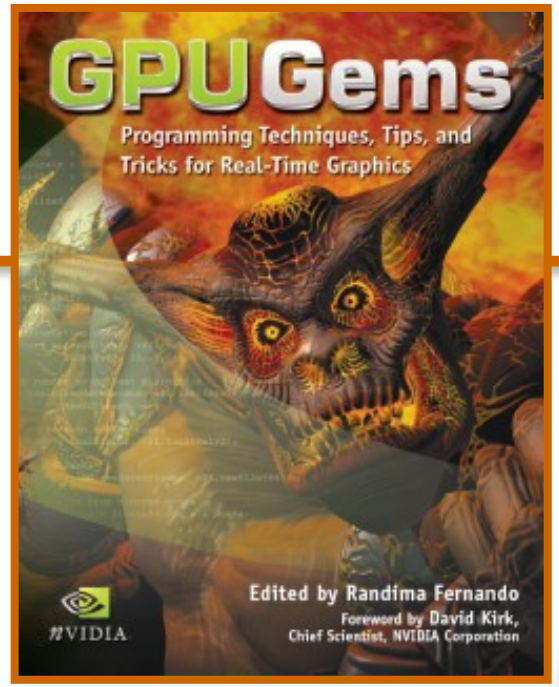
- Set a render target with one pixel per triangle and clear it to 0
- Draw fullscreen quad with:

```
float4 PixelShader(float2 texCoord, uniform Triangle Cutter)
{
  Triangle tri;
  tri.V0 = tex2D(CurrentPositionTex, texCoord);
  tri.V1 = tex2D(CurrentPositionTex, texCoord + offset1));
  tri.V2 = tex2D(CurrentPositionTex, texCoord + offset2));
  if (TriangleIntersect(tri, Cutter))
    return 1;
  else
    discard;
}
```

- Read back render target to CPU
- Modify stiffness textures
- Modify cloth index buffer



GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics



- **Practical real-time graphics techniques from experts at leading corporations and universities**
- **Great value:**
  - **Full color (300+ diagrams and screenshots)**
  - **Hard cover**
  - **816 pages**
  - **CD-ROM with demos and sample code**

For more, visit:
http://developer.nvidia.com/GPUGems

"*GPU Gems* is a cool toolbox of advanced graphics techniques. Novice programmers and graphics gurus alike will find the gems practical, intriguing, and useful."

Tim Sweeney

Lead programmer of *Unreal* at Epic Games

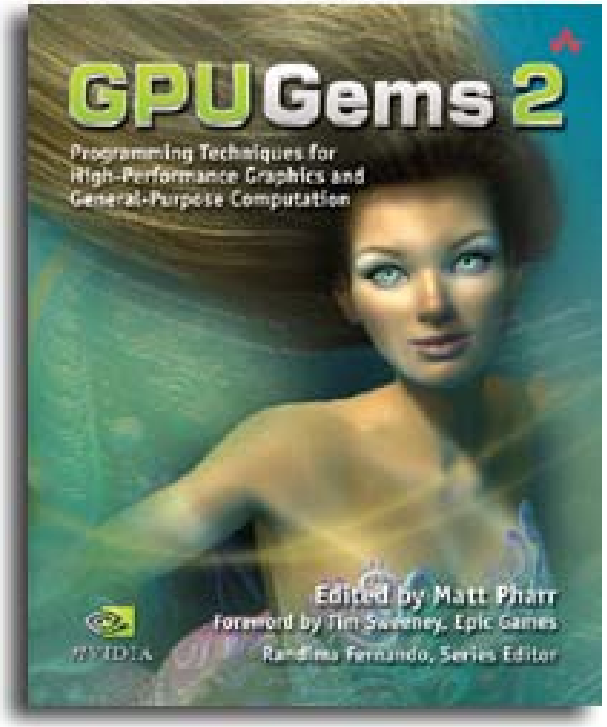
"This collection of articles is particularly impressive for its depth and breadth. The book includes product-oriented case studies, previously unpublished state-of-the-art research, comprehensive tutorials, and extensive code samples and demos throughout."

Eric Haines

Author of *Real-Time Rendering*

GPU Gems 2

Programming Techniques for High-Performance Graphics and General-Purpose Computation



- **880 full-color pages, 330 figures, hard cover**
- **$59.99**
- **Experts from universities and industry**

"The topics covered in *GPU Gems 2* are critical to the next generation of game engines."

— Gary McTaggart, Software Engineer at Valve, Creators of Half-Life and Counter-Strike

"*GPU Gems 2* isn't meant to simply adorn your bookshelf—it's required reading for anyone trying to keep pace with the rapid evolution of programmable graphics. If you're serious about graphics, this book will take you to the edge of what the GPU can do."

—Rémi Arnaud, Graphics Architect at Sony Computer Entertainment