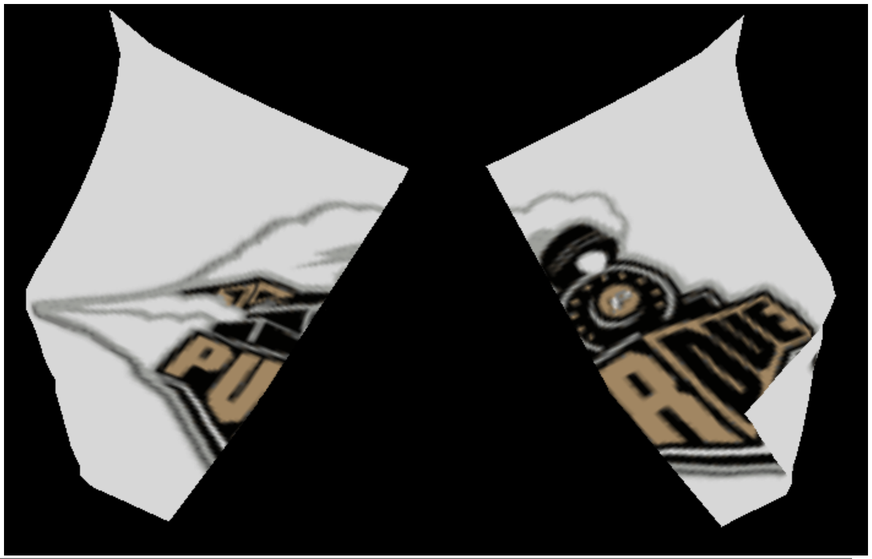
An Interactive and Rip-able Cloth

Nate Andrysco

Goal

- Have a virtual cloth that can be dynamically torn.

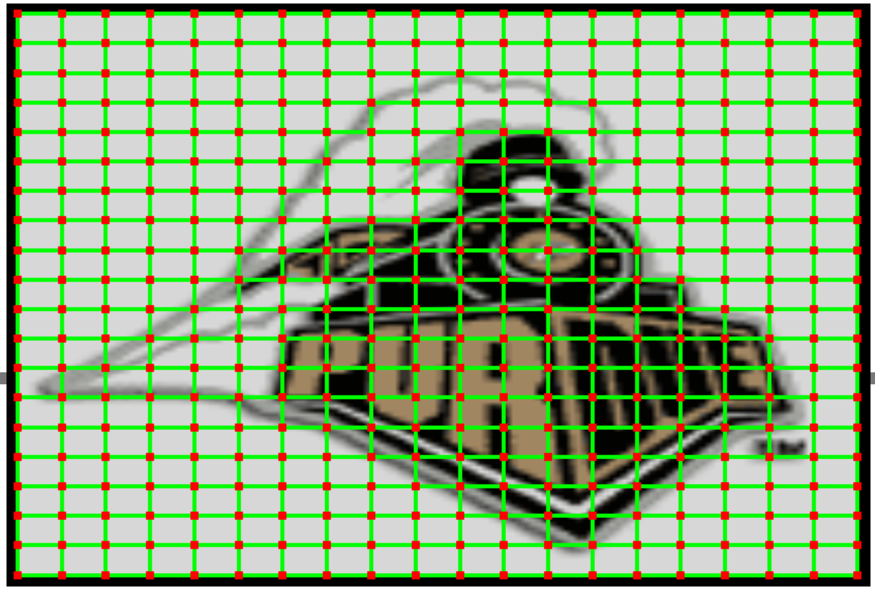


Intro to Cloth Simulation

- Real-life cloths are meshes of thousands, if not millions, of threads.
- Approximate using meshes of particles.
  - Speeds up simulation while maintaining realism.

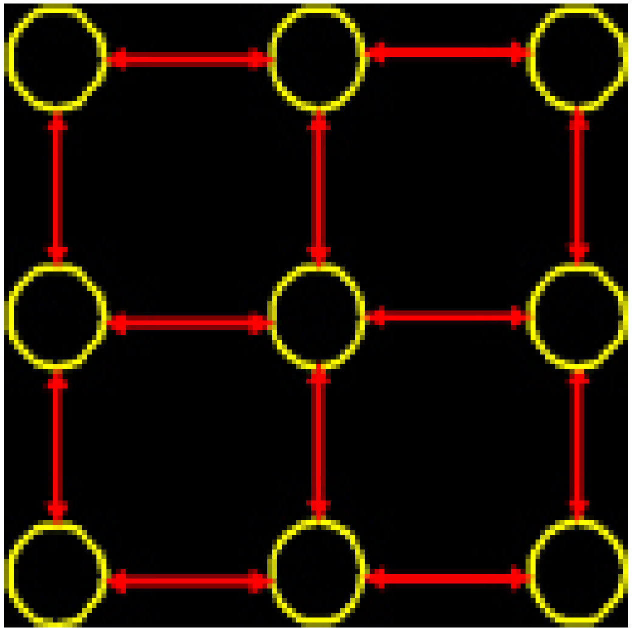
Mass-Spring Model

- Mesh of particles connected by virtual springs.
  - The springs are analogous to threads.
  - Particles are the intersection of two threads.

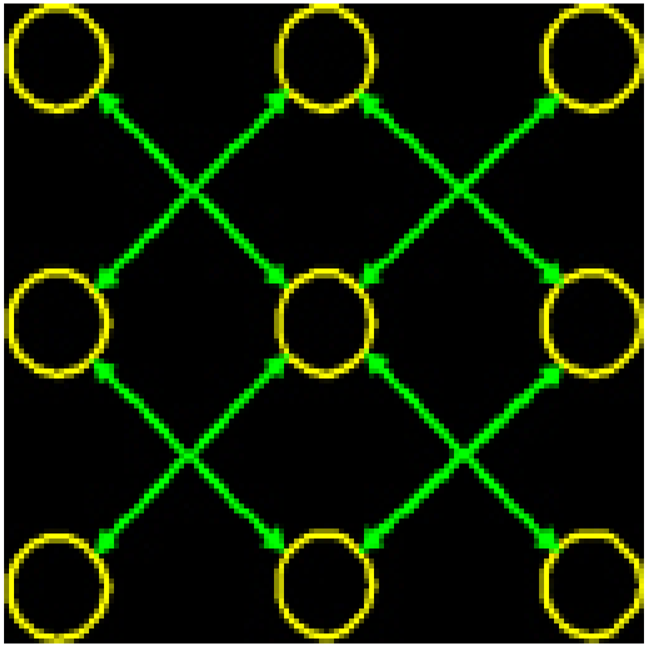


Types of Springs

Structural

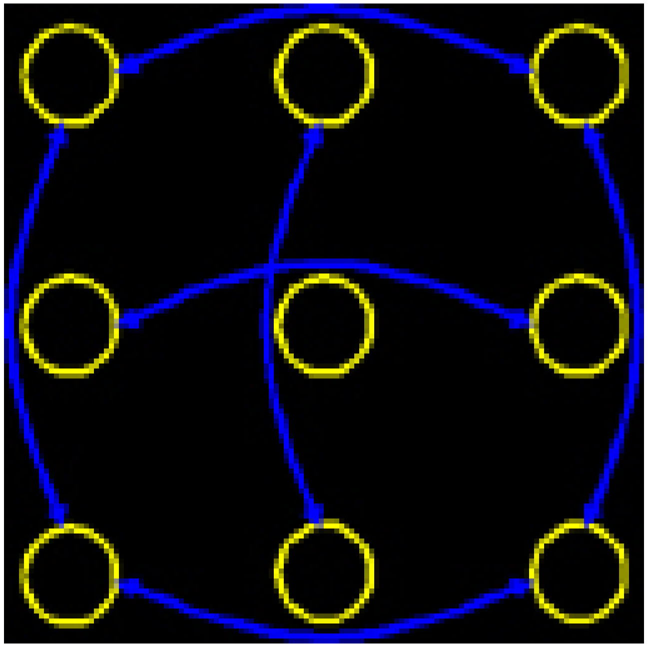


Shear

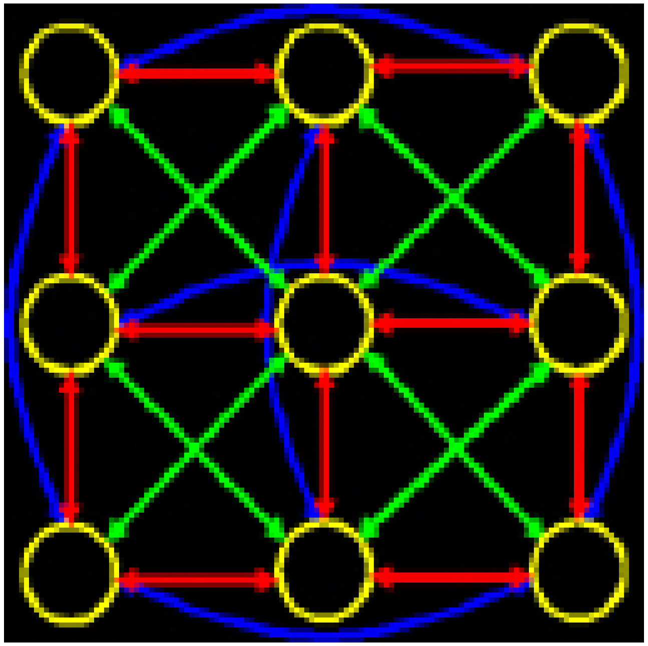


Types of Springs

Bend



All Together



Other Forces

- Add additional forces to make the cloth behave more realistically.
  - Gravity
  - Wind
  - Friction
  - Collision Response

Updating the Simulation

- Use integration to update the position of a particle given a time step, h, and the forces acting on the particle.
  - $v = dx / dt$, $F/m = a = dv / dt$
- Again, need to approximate using numerical integration.
  - Euler, Runge-Kutta 4th Order, Verlet

Collision Detection

- After position update, perform collision detection.
  - Collisions with external objects.
  - Collisions with the cloth's own triangles.

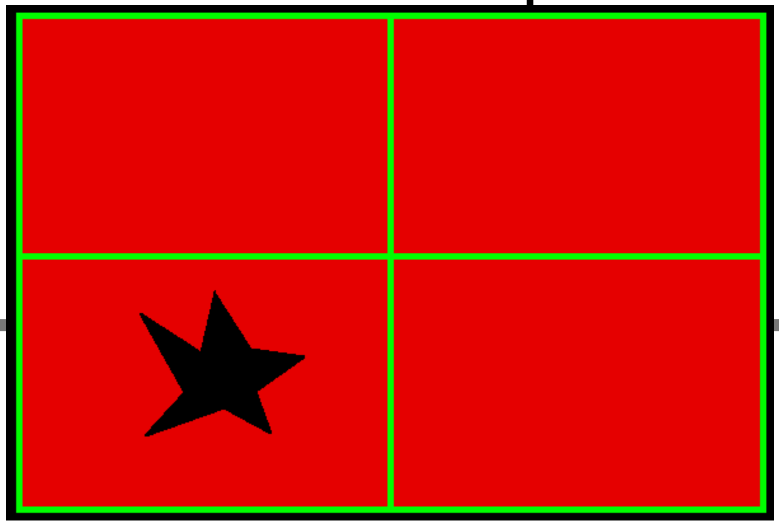
- Many papers devoted to both types of collisions.

Tearing a Cloth

- Why such a "tear-able" idea?
  - Animators must specify the ripping of a cloth by hand.
  - Incorporate into a game setting.
    - i.e. shoot a cloth in a first-person shooter

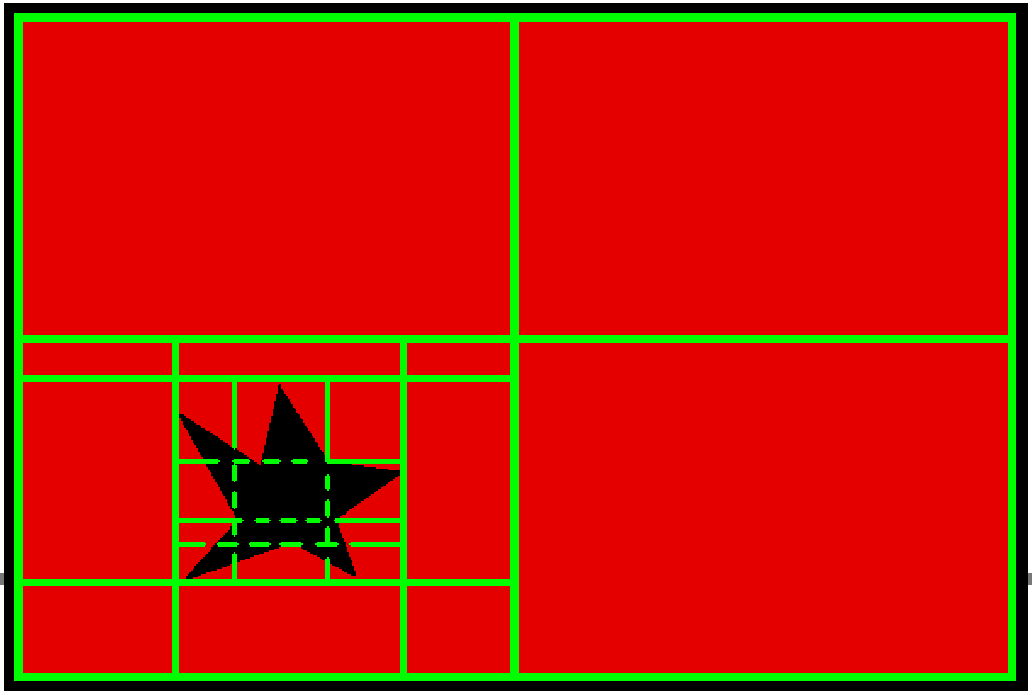
Difficulty

- If we had an infinite amount of resources, the problem would be easy.
- The particle approximations are what make it difficult.
  - What should we do if we have a rip in the middle of a quad?



Solution

- Refine the mesh where a tear occurs.
  - Keep realism while maintaining minimal complexity.

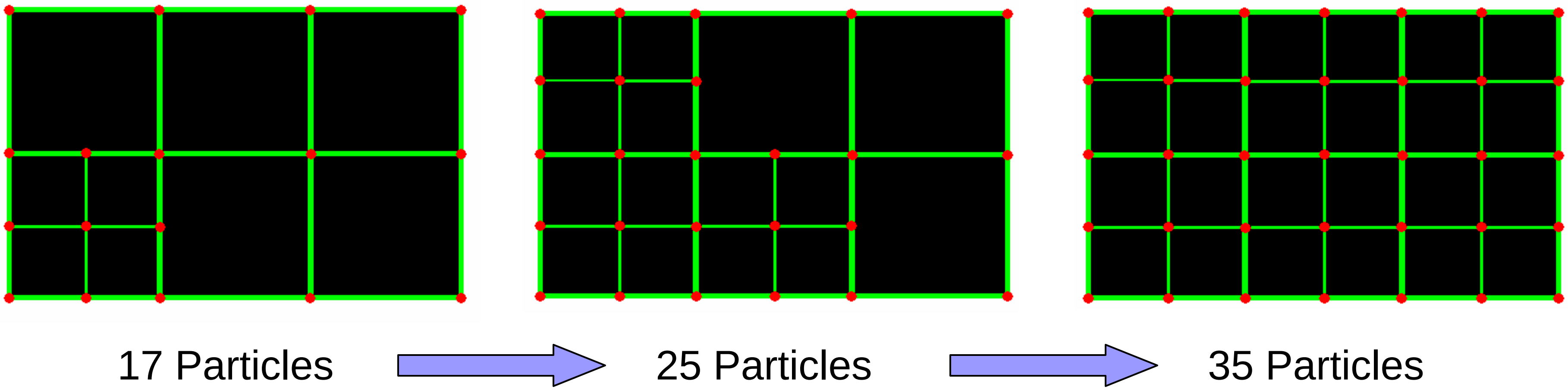


T-Junctions

- Problem:
  - After subdividing, T-junctions will occur.
    - When drawing quads, holes will appear.

T-Junctions

- Do not want to subdivide neighboring regions.
  - Significant increase in complexity

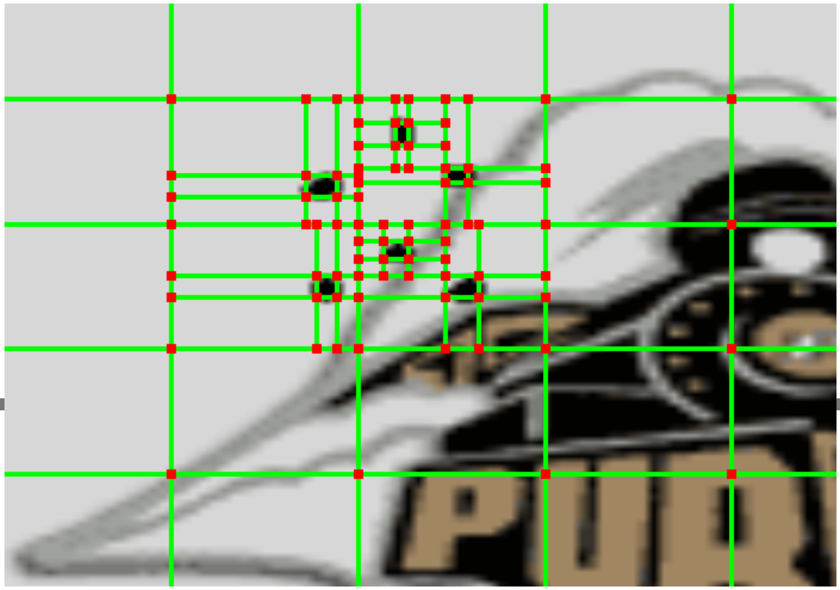


T-Junctions

- Solution:
  - Intermediate particles are interpolated.
    - Guarantees that intermediate particles always are positioned on the edges of neighboring grids that are not subdivided.
  - Negligible complexity added when updating position.
    - However, complexity is added to collision detector.

Initial Tests

- Simulating the cloth in a game environment.
  - The user can shoot the cloth as if he had a gun.



Future Work

- The ability to drop a heavy object onto the cloth and have it tear.
  - Or pull the cloth by its corners.
- Pour water on the cloth and have it respond realistically.
  - Or set the cloth on fire.

References

1: Baraff. Large Steps in Cloth Simulation

2: Lin, Tang, Dong. Cloth Simulation Based on Local Adaptive Subdivision and Merging

Questions ???

Euler

- Fastest but least accurate.
- Equation: $x(t+h) = x(t) + h*f(t, x(t))$
- Accuracy: First order Taylor series accurate -> $O(h)$
  - Not good enough for cloth simulation.
    - Instantly blows up, cloth requires much greater stability.
    - Simpler particle meshes can get away with it (i.e. often used in games for ropes)

Runge-Kutta 2nd Order

- Trades speed for better accuracy.
- Equations:

$$x(t+h) = x(t) + 0.5*(K_1 + K_2)$$

$$K_1 = h*f(t, x)$$

$$K_2 = h*f(t + h, x + K_1)$$

- Accuracy: 2nd order Taylor series accurate -> $O(h^3)$

Runge-Kutta 4^{th} Order

- Even slower but even more accurate.
- Equations:

$$x(t+h) = x(t) + 1/6 * (K_1+2K_2+2K_3+K_4)$$
$$K_1 = h*f(t, x)$$
$$K_2 = h*f(t + h/2, x + K_1/2)$$
$$K_3 = h*f(t + h/2, x + K_2/2)$$
$$K_4 = h*f(t + h, x + K_3)$$

- Accuracy: $O(h^5)$

Verlet

- Speed on the order of Euler but with the numerical stability of RK4.
- Calculate new position using the last two positions of the particle (which allows for the approximation of the velocity!).
- Can slightly modify the equation to simulate damping effects (i.e. wind resistance)
- See Wikipedia for more details.