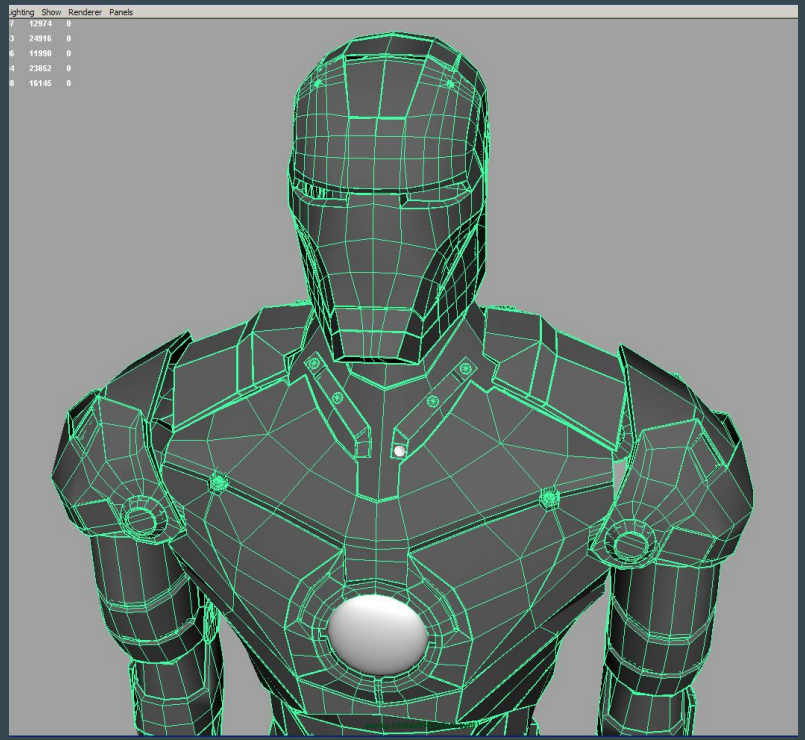
Geometry Destruction

CSE 5912

Alex Losego, Ryan Wachowski, Danny White, David Wright

Overview

What is Geometry Destruction?



- Objects in video games are represented by meshes of polygons
- Non-procedural destruction = pre-defined sub meshes
- Procedural destruction = modify the meshes at runtime
  - Random
  - Response to stimuli

Today's Overview

- History & Examples
- Applicable Algorithms
- Toolkits
- Gameplay Considerations
- Demos Throughout

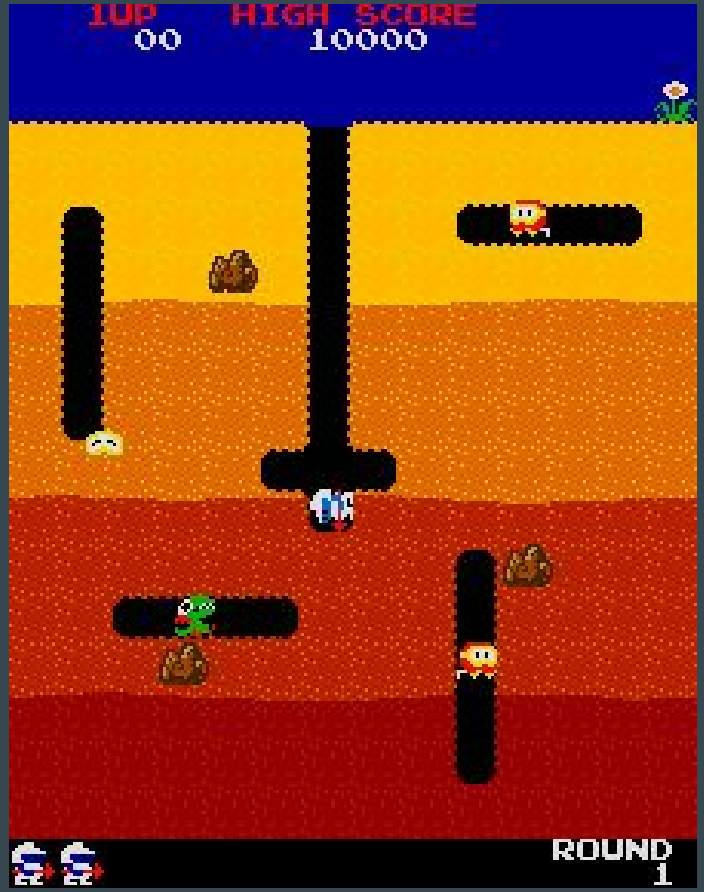
History & Examples

2-Dimensional, Procedural

- *Space Invaders*, 1978
- Arcade
- Barriers destructible by player & enemies
- Raster collision detection

2-Dimensional, Non-Procedural

- *Dig-Dug*, 1982
- Arcade
- Entire playfield is destructible
- Player stays on a grid



3-Dimensional, Non-Procedural

- *Super Mario 64*, 1996
- Nintendo 64
- Simple, predefined fractures and animations
- Particle effects

3D Non-Procedural Demo

3-Dimensional, Voxel-Based

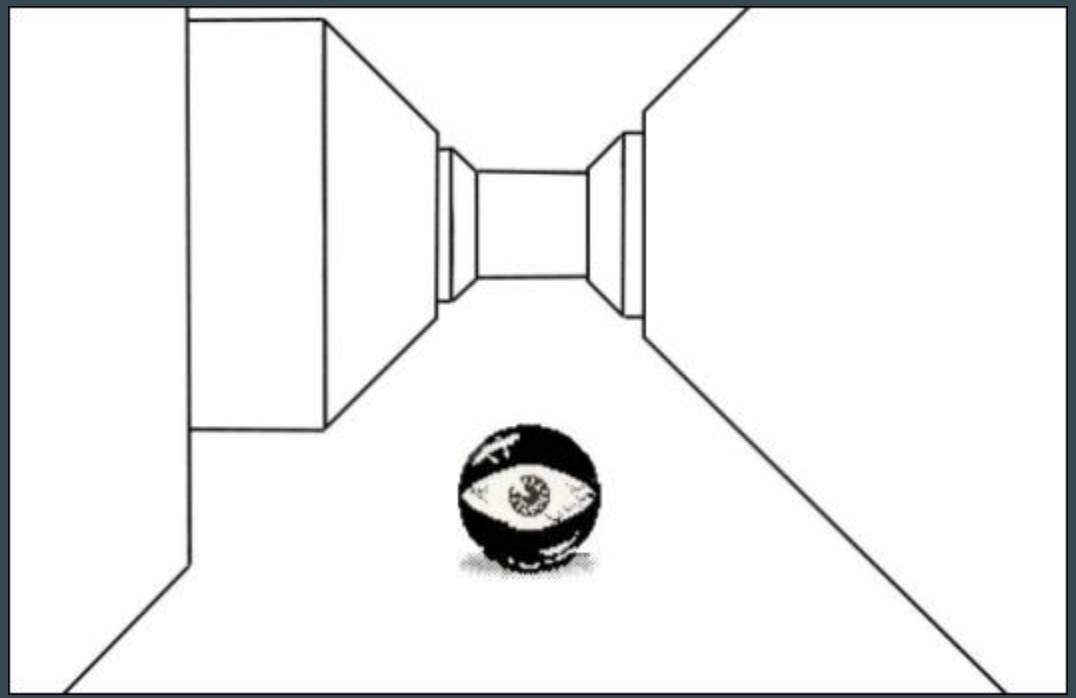
- *Minecraft*, 2011
- PC
- Procedural, but simple
- Particle effects

3D Procedural Demo

3-Dimensional, Procedural

- *Tom Clancy's The Division*, 2016
- PC, XBox One, Playstation 4
- Many different types of materials
- Fewer particles, more fragments

Games Without Geometric Destruction

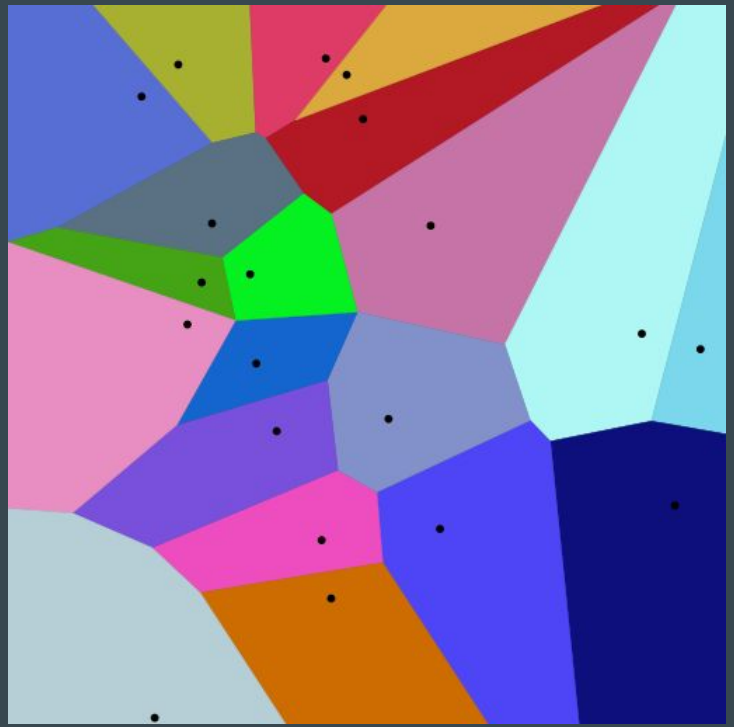


Applicable Algorithms

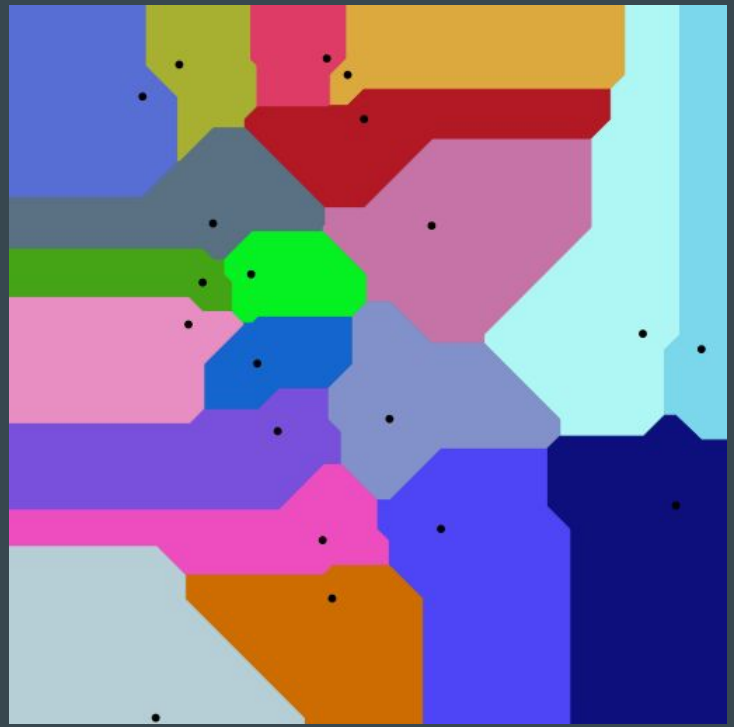
Mesh Tessellation - Voronoi Diagrams

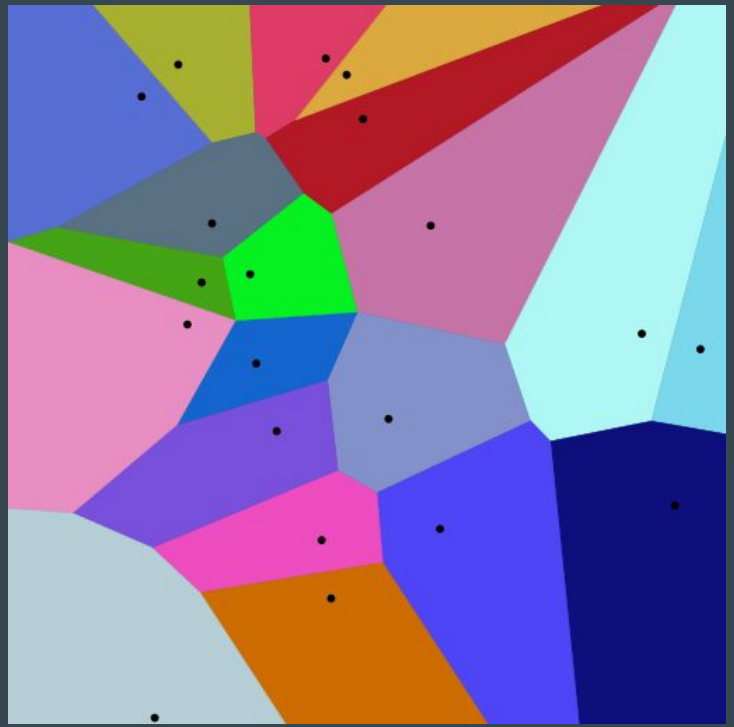
- “Human algorithm” is simple in nature

```
routine VORONOI_TESSELLATE(Plane, N) is
  Generate N points
  for each arbitrary-resolution point P* in Plane do
    Evaluate assignment metric for P* on each N point
    Assign P* to matched point
  endfor
  Fill generated regions
end routine VORONOI_TESSELLATE
```



Tessellation Metrics

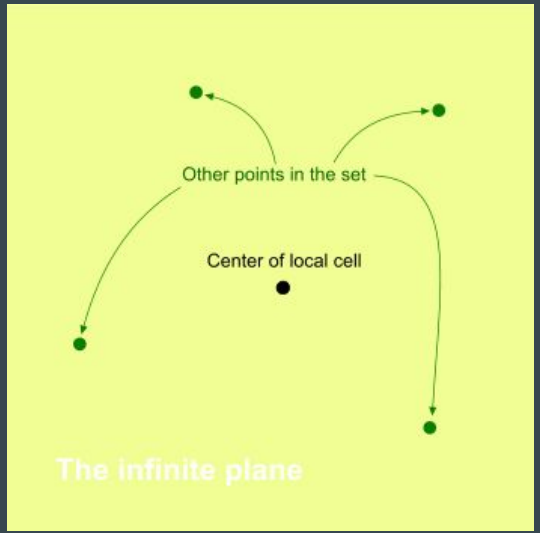




Brute-force Voronoi Diagram

- Brute force isn't *necessarily* bad—consider trade-offs.
- Amortized analysis: $O(n^2)$

```
for each point...
  for each other point...
```

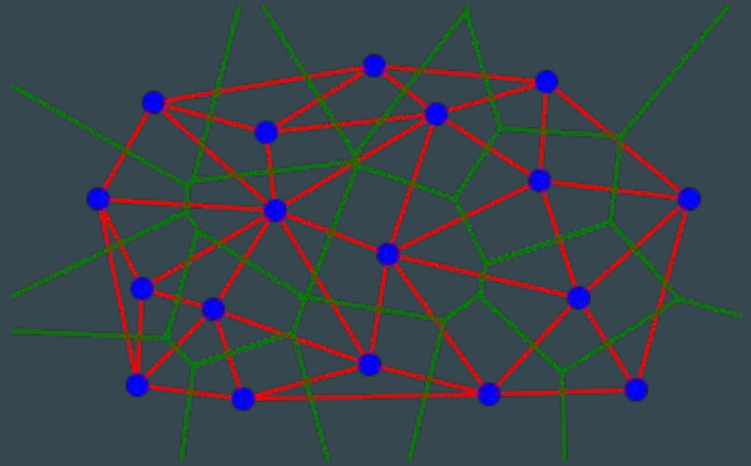


In Search of the Feasible Solution

- “Human algorithm” (obviously) ambiguous
- Second algorithm is slow, but a possibility
- Third algorithm..?

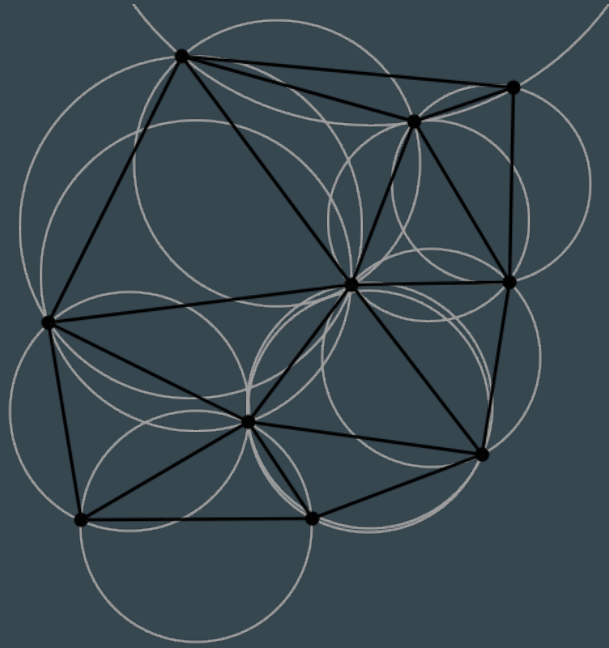
Voronoi Diagrams and the Delaunay Triangulation

- Delaunay and Voronoi: dual graphs
- Derive one from the other



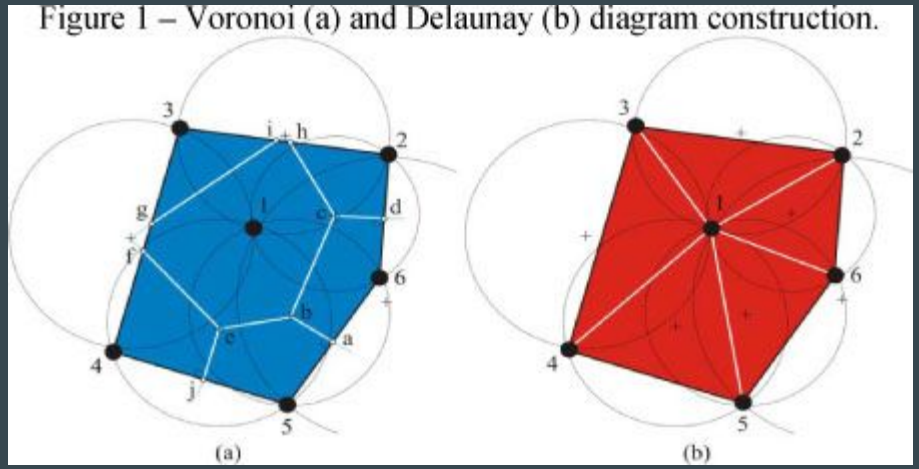
Delaunay Triangulation

- Disregarding degenerate cases...any set *P* can be triangulated
- Additional constraints for Delaunay
- Creates “well-formed” geometry



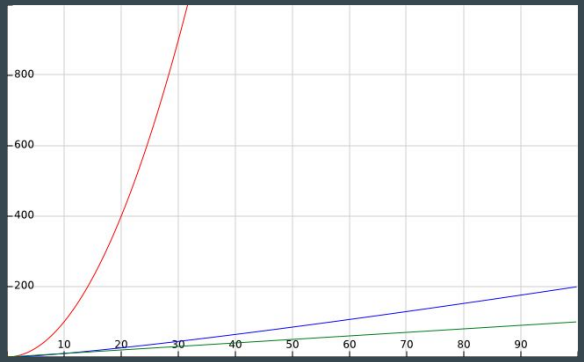
The Derivation

- Generate points
- Triangulate points
- Circumcircle centers → Voronoi vertices



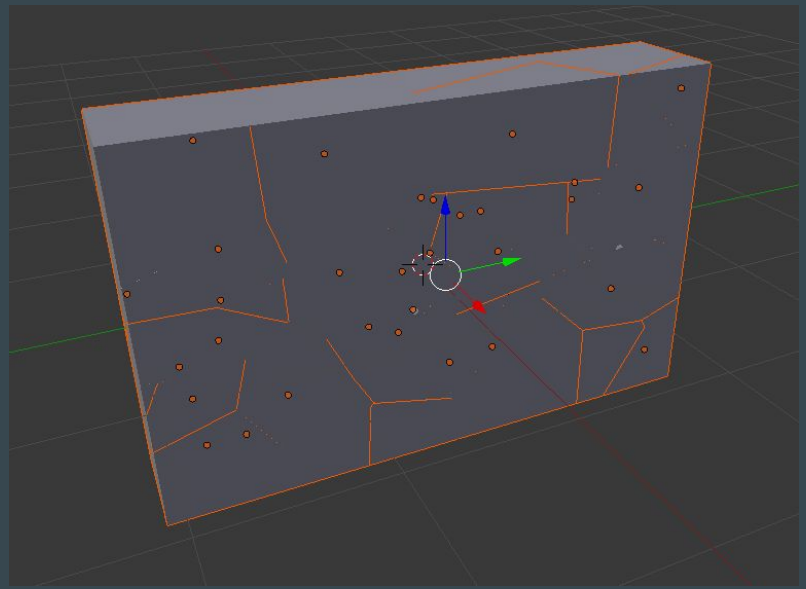
A Gold Mine

- Multiple variations exist
- $O(n \log n)$ complexities; $O(n^2)$ in worst-case for naive “flipping” implementation
- Some only work in two dimensions—choose wisely



Transitioning From 2D to 3D Shatters

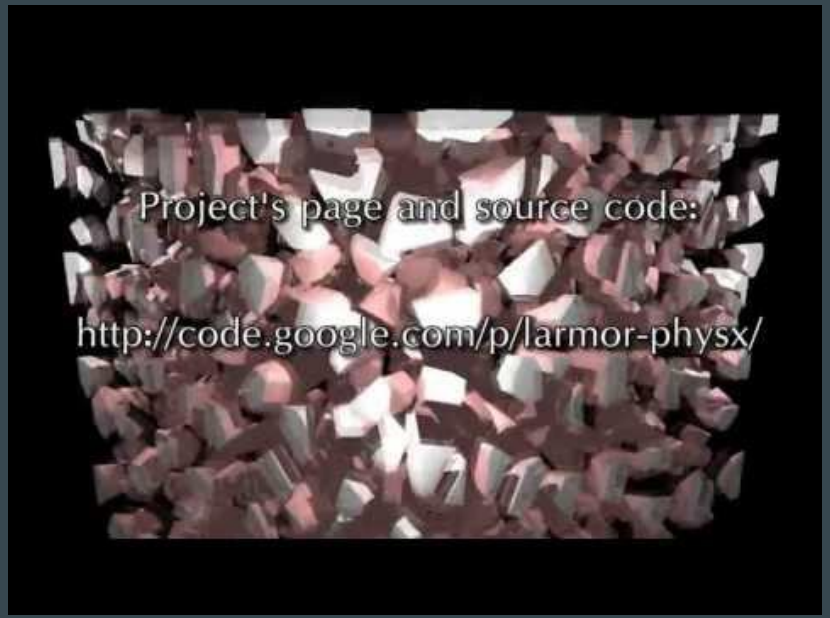
- Is not trivial
- Can be costly at runtime
- Major factor: reconstruction of exposed back-face surfaces (with DT)
- Possible solutions: bake it into model or animation (“it’s the artist’s problem”)



The End Result

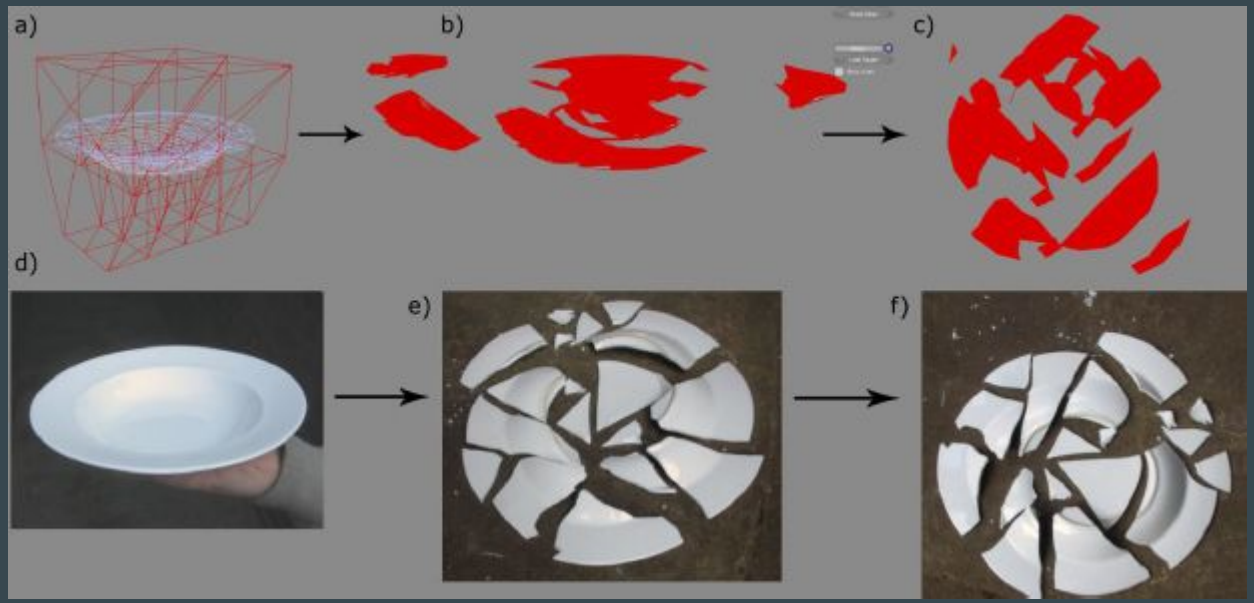
- Industry-standard destruction is (virtually) kept a secret
- Secrets are worth money of varying amounts

Voronoi Shatter in Three Dimensions



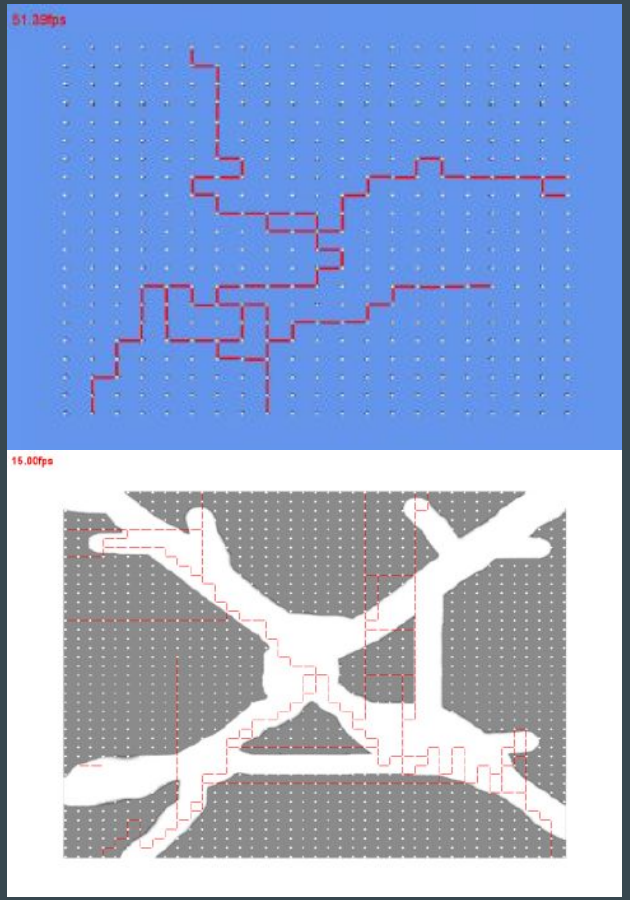
Cracking Algorithm

Objective: To accurately simulate a brittle material being fractured at its center point.

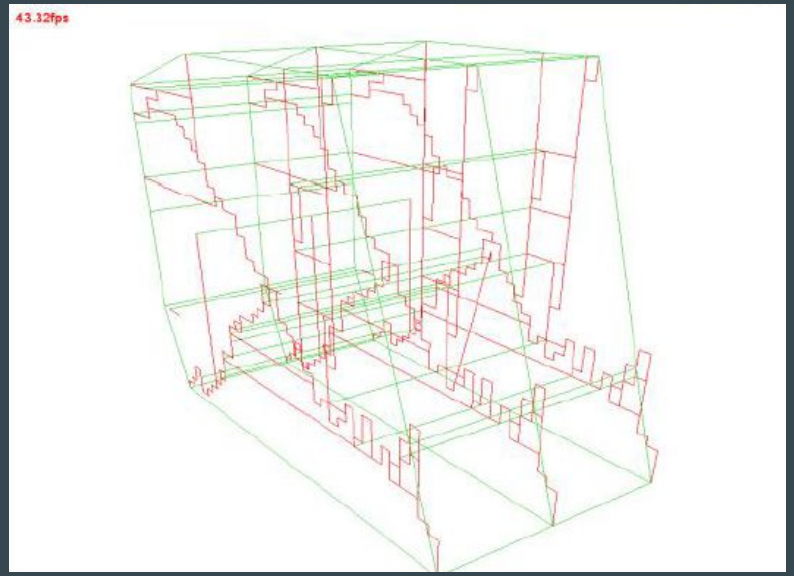


Cracking Algorithm

- Random walk from center of grid
- Random branching
- Weight nodes to form “strain map”
- Forms a basis for the 3D fracture pattern

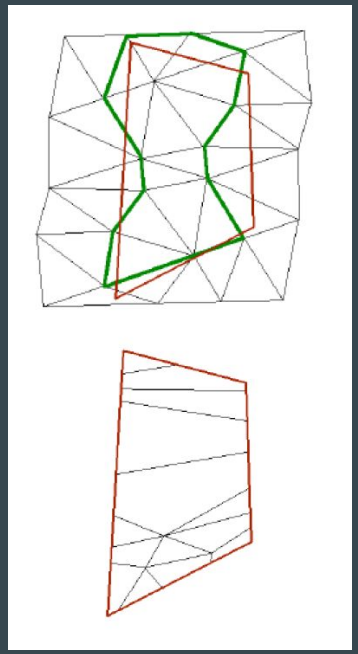


Cracking Algorithm



- Generate many 2D cross sections
- Smooth and sew them together
- This creates a 3D partition of the mesh, each being a fragment

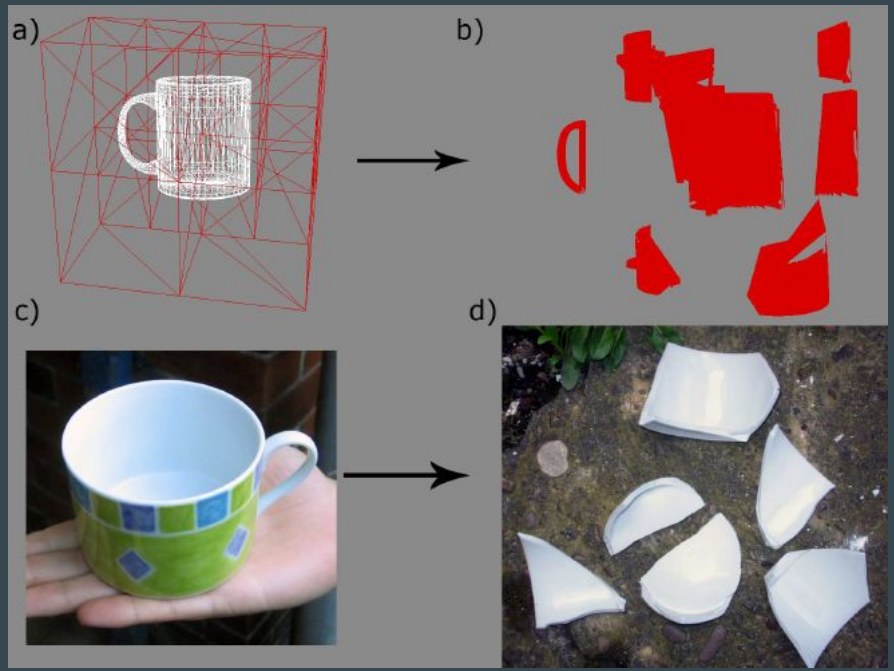
Cracking Algorithm



- Clip the mesh against the partition
- Remesh fragments
  - Matching Nearest Edge
  - No tessellation Clipping
  - Tessellation Clipping

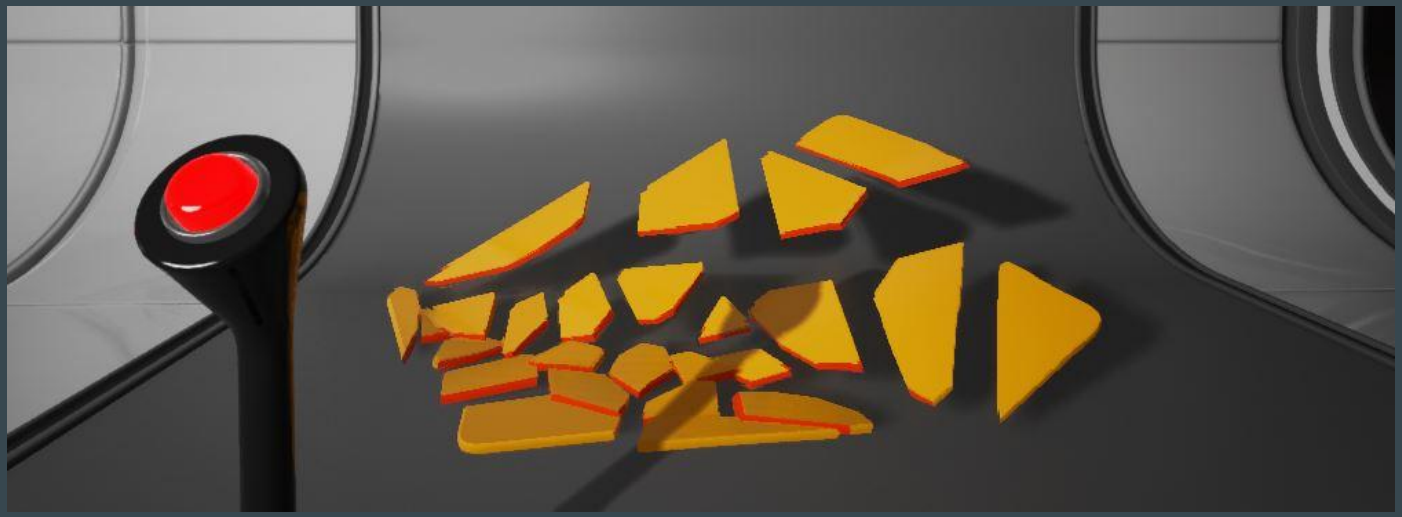
Cracking Algorithm

- Strain texture controls shape and style of fragments
- Resolution of cross sections controls size
- Can modify smoothing algorithm to create different textures



Toolkits

Unreal Engine: Built-In Destructibles



- Nvidia's APEX PhysX Lab for Destructible Mesh creation
- Destructible Properties Matrix

Toolkits in Unity: 2D

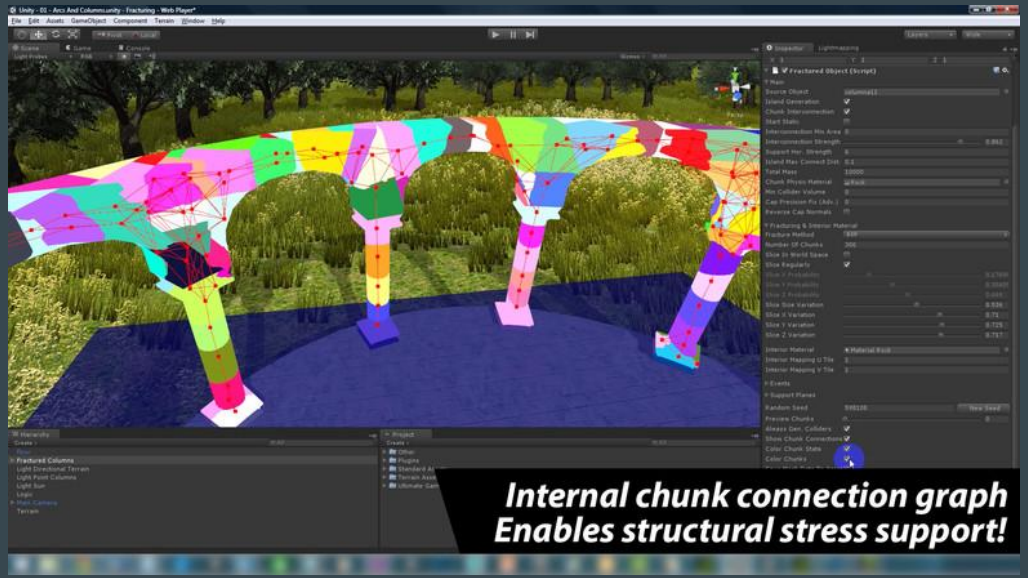
Destructible 2D: https://www.assetstore.unity3d.com/en/#!/content/18125

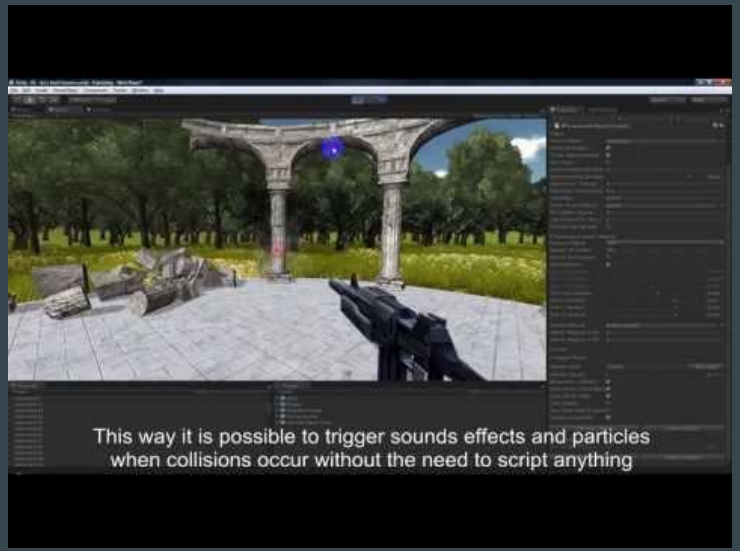
Note: $50 as of 3/6/2016

Toolkits in Unity: 3D

Fracturing and Destruction: https://www.assetstore.unity3d.com/en/#!/content/9411

Note: $60 as of 3/6/2016





Gameplay Considerations

Pros of Destructible Meshes

- Realistic responses to ordnance create an engrossing world
- Dynamic map arrangements and large-scale power struggles
- Small-scale gameplay element of destroying enemies' cover
- Enhance replayability since every round has different result

Screenshot from Battlefield 4

Cons of Destructible Meshes

Screenshot from Gears of War

- Performance cost
- Additional development work
- Resulting map may not be fun to play on
- “Destructible” characters--AKA gore--will cause an M rating and restrict your audience

Questions?