Introduction to Computer Graphics

Lecture Notes 3/27/2019

Announcements:

No class April 3rd (next Wednesday)

Assignment 2 has been extended to be April 2nd since the OpenGL problem description has changed. You will receive full credit if you load a stick figure and display it to the screen, but will receive extra credit if you do the original problem which is load a humanoid and display it to the screen.

The final project due date is May 5th

Homework 3 will have a problem on rigid body dynamics – specifically you will be expected to use one of the simulation platforms from one of the three mentioned in class:

- Bullet Physics https://pybullet.org/wordpress/
- Project Chrono https://projectchrono.org/
- Open dynamics engine https://www.ode.org/

Rigid Body Dynamics: Collisions:

Two types of collisions: Separating collisions and non-separating collisions (which solves the contact problem)

Separating Collisions: Two parts to the problem:

1) Collision Detection – to know that there was a collision
2) Collision Resolution – once we know that there is a collision, the collision needs to be resolved

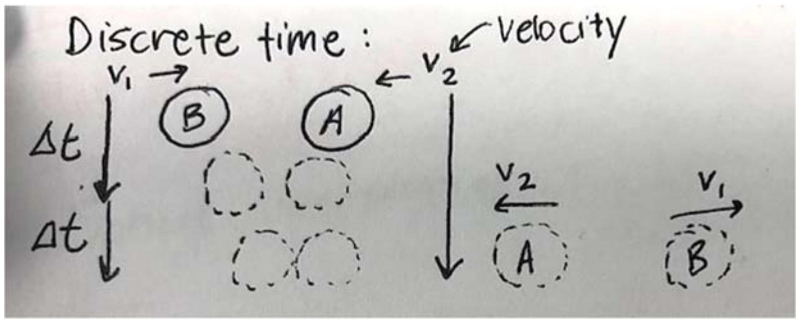
Several ways to solve these problems

Collision Detection:

Different ways to represent time when we are figuring out at which point in time there is a collision:

1) Discrete time – 0, Δt, 2*Δt, etc
    a. This requires Δt to be a small enough time step for an accurate collision detection of objects within that time step – the time step is limited by the object's mass and velocity
2) Continuous time – no time step description

Discrete time:



Here, at time step t, the objects become closer and closer to each other given their mass and velocity. And then they bounce away from each other once the collision is detected.

Δx is the minimum edge length of the bouncing box over all objects

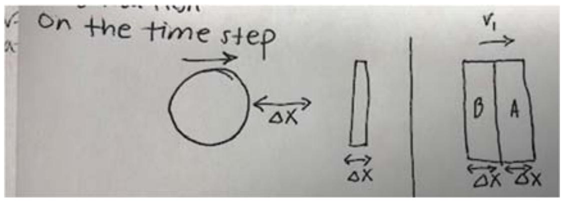
$|\vec{v}|$ is the maximum velocity over all objects

We put a condition on the time step as seen below known as the "Time Step Resolution"

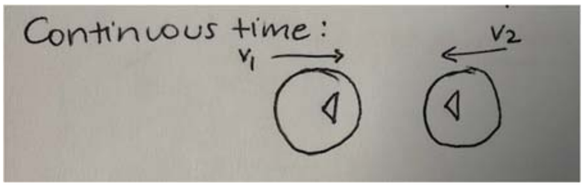
$$\Delta t < \alpha * \frac{\Delta x}{|\vec{v}|} * 0.1$$

Where α is known as the CFL constant → the Courant-Friedrichs-Lewy constant

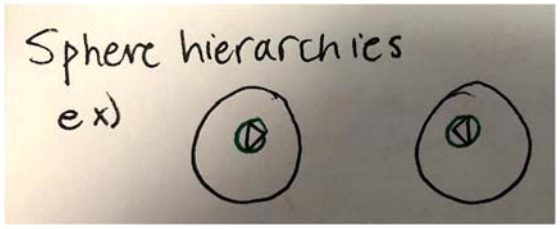
If α is less than 1, then you are guaranteed collision detection (and normally we use α = 0.5)



Continuous time:



Assuming we have the triangle mesh for each object, we compute the trajectory for the triangles in the mesh and do the same for the triangles in the other mesh, if assume they move with the same velocity, we get a cubic equation of whether they collide or not. If the cubic equation has a real solution, it will give the position at which the objects collide.

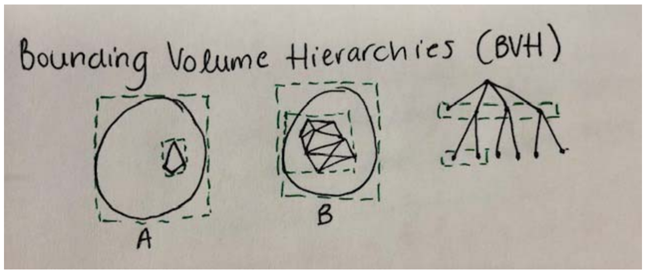


Sphere hierarchies – tries to minimize the cubic equation for each triangle in the mesh. We detect a collision by detection of the collision of spheres

→ just a simple check between the radius lengths of the two spheres.

How to detect the collision: Two types of object representation that we have available

1) **Mesh-based**
2) **Implicit-Surface**



Mesh-based:

Using Bounding Volume Hierarchies – these net bounding boxes group together some surface triangles in the object. The representation can be seen as a tree, where the entire object itself is the root and the entire surface patch are represented in a layer, and the children of each surface patch are subpatches. Essentially, to detect a collision you trace down the leaves of the tree correspond to triangles. This is testing box-box intersection until you test the leaf nodes making the test triangle-triangle intersection.

Convex Decomposition: a computationally NP Hard problem

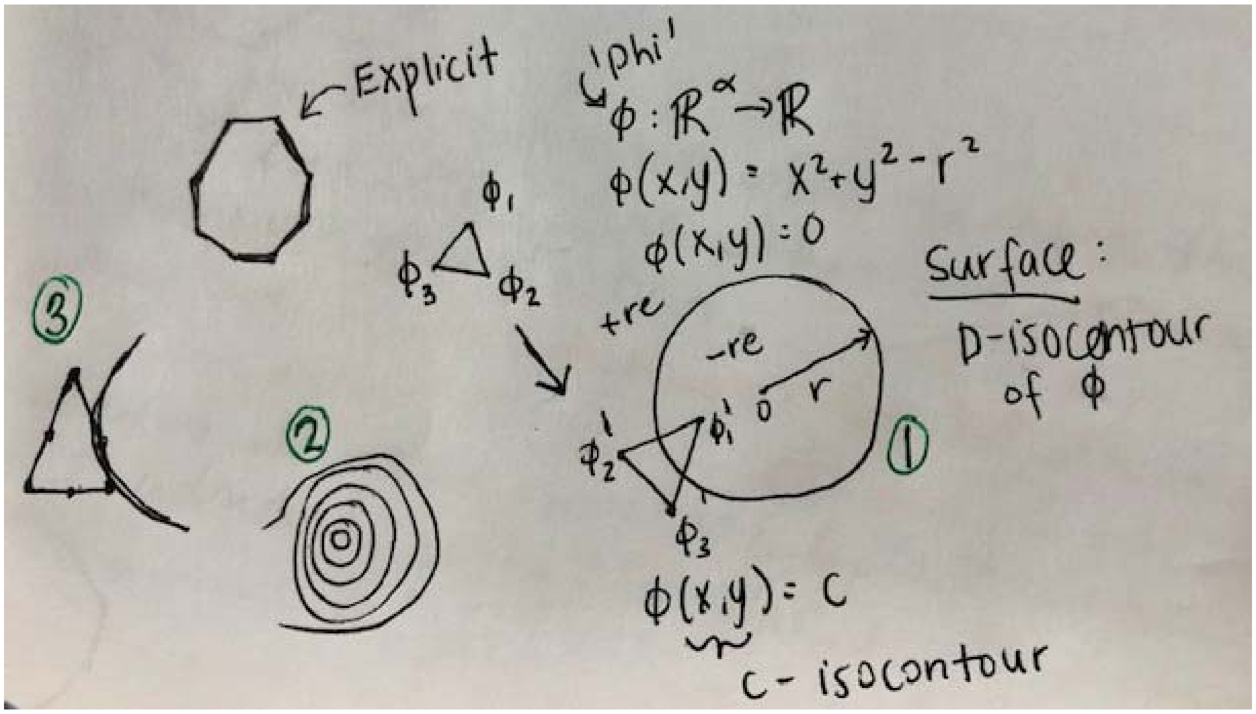
Convexity definition: a real-valued function defined on an *n*-dimensional interval is called convex if the line segment between any two points on the graph of the function lies above or on the graph. Also can be described as the two points can "see each other."

Approximate convex decomposition:

There are flaws in this methodology but you can only go so far with what you can resolve.

So we use implicit-surface object representation as a collision detection solution and approximate convex decomposition

Explicit → Individual points/triangles on the surface



See figure above and follow the numbering of the figures:

1) Shows the circle equation. We need to test what the value of phi is for the points and if phi is a positive value, then the surface is not intersecting with the sphere, while if it is a negative value, then the surface is colliding with the sphere.
2) Weather reports use isocontours in order to show the magnitude of the terrain/mountain. The isocontour is based on the height for the curve.
3) Explanation that the intersection of the points of a surface and another sphere, you may need to look at points on the edge of the surface for more precise collision detection.