Quiz 3

Victoria Deng

August 2020

Lecture 7 - Collision Detection

Collision Detection: Detection of an intersection between different objects in a virtual environment

- Need to prevent objects from going through each other as if they aren't there.
- Need to produce a realistic virtual environments by having objects react accordingly when a collision happens.

Continuous Approach (a priori): Using the trajectories of the objects to predict if a collision is going to occur

- This approach is robust, however has drawbacks
- When dealing with objects that have a lot of polygons, need to calculate all possible intersections for each polygon.
- Also requires a lot of calculations in an environment with a high number of objects moving.
- Computationally complex, which can affect performance/framerate.

Discrete Approach (a posteriori): Allows objects to collide, and reverses it back to where the collision happened initially in the engine.

- More efficient that continuous, but also has problems
- Tunnelling: occurs when a very fast object passes through another object, but the engine does not detect the collision.

Bounding Volumes: A simplified shape encloses the entire object, though it should be as small as possible.

Axis Aligned Bounding Box (AABB): The bounding box remains orientated with respect to the main axes. I.e if the object turns, the box turns with it.

- Pro: Translation invariant, it will not change if you move every point of the object by the same amount in a given direction. (i.e move object in it's entirety)
- Con: With any other movement, the box will no longer remain axis-aligned. (i.e moving one arm up but the rest of the body stays)
- Con: Simple to compute, but needs to be recomputed every frame due to input changes.

Oriented Bounding Box: Box is oriented with respect to the object.

- Pro: Transformation invariant for translation, reflection and rotation. If the object moves, the box moves with it.
- Pro: The volume is more compact than AABB, and has less empty space (false positives)
- Con: The computation of intersections is much more complex as box orientation is dynamic.

Bounding Spheres: Uses a sphere as a bounding box.

- Pro: Transformation invariation for every movement
- Con: More empty space in the volume
- Pro: Simple collision detection
- If $D \leqslant R1 + R2$, then the spheres are intersecting
  where D is the distance between circle 1 and circle 2, and R is the radius of each circle

Hierarchical Bounding Volumes: Has two phases, consists of multiple spheres bounding an object.

- Broad Phase: Is simple and has a low cost. The first pass on the bounding volumes, used to identify potential collisions, and mark objects as definitely not colliding.
- Narrow Phase: Is complex and has a high cost. Based on the identified potential collisions from the broad phase, exact collision tests are conducted to determine if the object will collide or not.
- For example, if the point is outside the broad phase, then it is not intersecting with the object. If the point is inside the broad phase, then it will advance to the narrow phase and check how many spheres it intersects (and deal appropriate damage etc.)

Hitboxes: Used in video games for real-time collisions. Are simplified bounding volumes that detect one-way collisions, like bullets hitting a character. Can be adjusted to be a tight or large bound to balance difficulty.

Lecture 8 - Illumination

Surface Types: Shading of a surface depends on the surface type.

Self-luminous: Object that has in itself the property of emitting light. Eg. Jellyfish

Refractive: Surface that changes the direction of light, caused by a change in transmission medium. Eg. Water or Glass

Translucent: Light interacts in a more complex way. Eg. Diamonds make light scatter.

Reflective (diffuse): Light is reflected from a surface in many different angles. Eg. Rugged surfaces like mountains.

Reflective (specular)': Light is reflected from a surface in the same angle as the incident array. Eg. smooth or glossy surfaces.

Isotropic: Relationship between the incoming direction of light is the same as the outgoing direction of light for the whole surface

Anisotropic: Relationship between incoming and outgoing direction differs over the surface.

Simple Illumination Models: Do not consider shadows, reflections or photon-based effects.

Full Ray Tracing: Considers all rays of lights and their recursive interaction between each object (computationally complex).

Decide Model Limitations: Decides how many times we will recurse (or allow for re-reflection)

Components of light: Three components of light.

Ambient Component: Simpliest kind of shading is from ambient illumination. Light comes uniformly from all directions.

The radiated light intensity (I) is given by:

$$I = I_a k_a$$

I_a = the intensity of the ambient light
k_a = the percentage of light reflected by the object

Diffuse (Lamebertian) Component: Brightness depends on the angle between the direction of the light source and the surface normal. Assumes that light is re-radiated uniformly.

Surface Orientation: The amount of light radiated towards the viewer is greatest when the surface normal is pointing straight at the viewer, and falls off according to the cosine rule as the surface slants away from the viewer. At the same time, more of the surface is seen within an angle as the surface slants away from the viewer, which also operates according to cosine rule. These effects compensate, so the intensity of Lambertian reflection is independent of surface orientation with respect to the viewer.

The Diffuse Illumination equation is given by:

$$I = I_p k_d cos(\theta)$$

I_p = intensity of light
k_d = coefficient of diffuse reflection (reflectivity)
We can also express it as a scalar product:

$$I = I_p k_d (\bar{N} \cdot \bar{L})$$

$\bar{N}$ = unit vector in the direction of the surface normal
$\bar{L}$ = unit vector in the direction of the light source

Distance: Also affects the intensity of the incoming light

Specular Component: Light reflected immediately on the outer boundary of a surface (highlight). Influenced by orientation, and distance of viewer and light source.

The spread of reflection is modelled by:

$$(cos\alpha)^n$$

If this equals to 1, then the viewer is at the direction of perfect reflection

α = angle of viewing direction
n = specular reflection exponent, controls the degree of spread
(high 100 = sharp highlight, low 1 = spread-out highlight)

Phong Reflection Model: Combines all three components

Lecture 9 - Particle Systems

Geometry Shader: Very useful in creating particle systems.

Particle System: A collection of many small particles that together represent a fuzzy object. Over a period of time, the particles are generated into a system, move and change within the system, then die from the system.

Collection of particles: A partcile system is composed of one or more individual particles. Each of these particles has attributes that directly or indirectly effect the behaviour of the particle or ultimately how and where the particle is rendered.

Stochastically defined attributes: All particles have the introduction of some type of random element via stochastic limits (bounds, variance, distribution). This can be used to control the particle attributes.

Attributes: Properties of particles.

- Emission (speed, spread, rate)
- Age (time that a particle has been alive)
- Lifespan (infinite, constant, variable)
- Opacity (static, dynamic)
- Colour (static, dynamic)
- Size
- Shape

Life Cycle: Phases of a particle in a particle system

- Generation
- Dynamics
- Extinction

Generation:

- Particles in the system are generated randomly within a predetermined location of the fuzzy object
- This space is termed the generated shape, and it can change over time
- Each attribute is given an initial value. They can be fixed or determined by the stochastic process

Dynamics:

- The attributes of each particle can vary over time.
- In general, each of the attributes can be specified by an equation with time as a parameter.
- Particle attributes can be functions of both time and other particle attributes.

Extinction:

- A particle will be destroyed once the age matches it's lifetime
- Can also be destroyed prematurely, from going out of bounds, colliding with an object, or if an attribute reaches a threshold. Eg. if a particle gets darker over time, once it hits black it's not going to get any darker and can be removed.

Rendering: Can use a 2D or 3D system. If we render the entire life cycle of a particle simultaneously, we would have static particles.

Sprites: 2D images, can be overlayed on top of a particle to simulate a texture.

Voxel': 3D pixels, also overlayed on top of a particle but results in being more realistic. Similar to a 3D gif on top of a particle. Will move with time.