Collision Detection

Motivation

- OpenGL is a rendering system
- OpenGL has no underlying knowledge of the objects it draws
- Collision Detection == Intersection Detection
Motivation

- Collision Detection
  - Identifying the intersection of 3D models
- Collision Response
  - Calculating the appropriate post collision response of the 3D models
- Assumptions
  - Closed 3D polygonal objects

Overall Approach

- Where is collision detection and response in code?
- In animation function
  - Usually in Idle or Display
  - Update object positions
  - If (CollisionDetection())
    - Collision Response()
  - Draw Objects
Collision Detection

- One part of the physics generally necessary in today's game environments

Basics
- Ray-Polygon Intersection
- Object motion vector is the ray
- Wall or other object is the polygon(s)
- Simple to implement
- Polygon-Polygon Intersection
- Can be expensive to calculate
- Separate from Collision Response

Point Polygon Intersection

- Assumption: The polygon is convex!
- Algorithm:
  - Draw lines from the point to the vertices of the polygon
  - Calculate the sum of angles between the lines
  - If the sum is 360° then the point is on the polygon
  - If the point is not on the polygon, the sum of angles will be smaller.
- Remarks:
  - The calculation is not precise
  - In one step you can pass the polygon
  - The larger the polygon, the smaller the change
  - Good for collision with large surfaces such as walls
Polygon Mesh Intersection

- Assumptions:
  - The object consists of a list of polygons
  - The object is closed
- Observation:
  - To move into the object you must collide with one of the object's polygons
- Algorithm:
  - Check collision with each of the polygons of the object using the point polygon intersection algorithm
- Remarks:
  - Works for large objects with large surfaces
  - Inaccurate for objects with small surfaces
  - Can be inefficient for objects with a lot of polygons

Bounding Sphere (BS)

- Finding BS:
  - Center is average of vertices
  - Radius is maximum distance from center
- Remarks:
  - Fast and easy to implement.
  - Lots of empty space within volume
  - Bad result for thin objects such as walls, polls
BS Intersection

- Point – BS Intersection:
  - Calculate the distance between the point and the BS center:
    \[ D = \Delta X^2 + \Delta Y^2 + \Delta Z^2 \]
  - If \( D < R^2 \), then there is a collision, where \( R \) is the radius of the Bounding Sphere

- Remarks:
  - Very easy to implement
  - Works for every distance

- BS – BS Intersection
  - If \( D < (R_1 + R_2)^2 \), then there is a collision

BS Intersection Cont.

- BS – Plane Intersection:
  - Calculate the distance between BS center and the Plane:
    - Let \( P \) be a point on the plane
    - Let \( n \) be the plane normal
    \[ d = (C - P) \cdot n \]
  - If \( d < R \) then there is a collision, where \( R \) is the radius of the Bounding Sphere

- More efficiently:
  - \( D = P \cdot n \)
  - \( d = C \cdot n + D \)
Bounding Cylinder (BC)

- **Finding BS:**
  - Flatten geometry to one plane (drop z coordinate)
  - Center is average of points (x, y only)
  - Radius is maximum distance from center
  - Find minimum and maximum z value for top and bottom planes

- **Algorithm:**
  - Collision if between z planes then within radius

- **Remarks:**
  - Fast and easy to implement.
  - Better representation for long objects such as: polls, humans, etc.

Axis Aligned Bounding Box

- **Finding AABB:**
  - Find maximum of each coordinate
  - Find minimum of each coordinate
  - Now have two opposite corners of the box

- **Point – AABB intersection:**
  - Check if the point coordinates overlaps.

- **Remarks:**
  - Fast and easy to implement.
  - Lots of empty space within volume
AABB Intersection

- AABB – AABB Intersection:
  - Calculate the distance between the 2 AABB’s centers (each axis separately):
    \[ D = C_1 - C_2 \]
  - If \( D < (\text{Size}_1 + \text{Size}_2)/2 \), then there is a collision, where \( \text{Size}_1 \) and \( \text{Size}_2 \) are the AABB’s sizes (3 values for the 3 dimensions)

![Figure 3. Axis-aligned bounding boxes in 2 dimensions.](image)

AABB Intersection Cont.

- AABB – BS Intersection:
  - Calculate the distance between the AABB’s edge and the BS center:
    \[ D^2 = \Delta X^2 + \Delta Y^2 + \Delta Z^2 \]
  - If \( D^2 < R^2 \), then there is a collision, where \( R \) is the radius of the Bounding Sphere

```c
for(i=0 ; i<3 ; i++) {
    if( C[i] < B.min(i) ) {
        s = C[i] - B.min(i);
        d += s*s;
    }
    else if( C[i] > B.max(i) ) {
        s = C[i] - B.max(i);
        d += s*s;
    }
}
```

![Figure 4. Closest point between a sphere and a box.](image)
Oriented Bounding Box (OBB)

- **Creating OBB:**
  - Define the object as axis aligned
  - Apply the rotation matrix on the object and on the bounding box

- **Finding OBB:**
  - Find mean of vertices, becomes center
  - Find the most distant vertex from the center
    - make it the first axis
  - Find the most distant vertex from the first axis
    - Make it the second axis
  - Calculate the cross product of the axes
    - Make it the third axis
  - Find the most distant vertex on the third axis

- **Remarks:**
  - Minimum amount of empty space in volume but a lot of calculations

OBB Intersection

- **OBB – OBB Intersection:**
  - For all possible separating axes, calculate the OBB’s projection to the axis:
    \[ r_x = a_x|A^T \cdot L| + a_y|A^T \cdot I| + a_z|A^T \cdot L| \]

- If there is an axis for which \[ |A^T \cdot L| > r_x + r_y \], then there is NO collision
Fast moving objects can cross between frames

Let

\[
\begin{align*}
\Delta A(u) &= A_0 + u \times \Delta v, \\
\Delta B(u) &= B_0 + u \times \Delta v,
\end{align*}
\]

Where:

\[
\begin{align*}
\Delta v &= A_1 - A_0, \\
\Delta v &= B_1 - B_0, \\
0 \leq u \leq 1
\end{align*}
\]

The objects collide when:

\[
\frac{\Delta B(u) - \Delta A(u)}{\Delta v} = \frac{(r_1 + r_2)^2}{(r_1 - r_2)^2}
\]

Remarks:

- We get quadratic equation
- There can be 0-2 solutions:
  - 0 – there is no collision between the objects
  - 1 or 2 – the solution with the smallest u is the collision point

Additional Information

- [http://www.realtimerendering.com/intersections.html#II247](http://www.realtimerendering.com/intersections.html#II247)
Collision Response

- Simple
  - Keep previous position

- Physics
  - Realism requires all Newton's laws
    - $F = ma$ (Force $F$, Mass $m$, Acceleration $a$)
  - Hook’s law
    - $F = -kx$ (Force $F$, Spring Constant $k$, Displacement $x$)
  - Gravity
    - $g = 9.8 \text{ m/s}^2$ (if you want)
  - Friction
    - $F_f = \mu F$ (Force Friction $F_f$, Coefficient of Friction $\mu$, Regular Force $F$)