CSC 240 Computer Graphics
Video 20: Collision Detection & Game Design Basics

Nick Howe
Smith College

Some slides & content courtesy Sara Mathieson

Collisions

What is meant by a collision in computer graphics?

- Two or more nominally solid objects occupying the same space
- Important in games and physical simulations
- May require action → how to handle?
  - *A priori*
    - Predict in advance
    - Solve for time & place
  - *A posteriori*
    - Detect & react
    - Discrete timesteps

Easier for games!
Collision Detection

Collision detection differs in **2D** and **3D**

- **2D collision detection**: may have hardware support
  - Sprite collisions
  - Color overpainting

- **3D collision detection**: handled in software
  - Object to object pairwise check: SLOOOOOW!
  - Partitioning & pruning schemes reduce work
  - Exploit temporal coherence

Collision Response

How to respond when a collision occurs?

- **Physics simulation:** add countering force
  - Proportional to degree of collision
  - Situation corrects over time

- **Game:** change position to avoid collision
  - Correct situation immediately
  - “Good enough” is ok
  - May also have in-game effects
Rough Collision Detection

Simple bounding box scheme – prelude to detailed check

- Akin to computer window overlap problem, in 3D
- Each object described by 3 intervals: $I_x, I_y, I_z$
- All three must overlap for collision
- Use a sorted list of endpoints for efficient checking
Detailed Collision Detection

How do we carefully check two objects for a collision?

- **Vertex method based on ray casting**
  - Cast ray from object center to each vertex
  - Compute first external intersection
  - If closer to center than vertex, record a hit
  - Run for both objects in pair

- **Slow and expensive – save for last resort**
  - Simple proximity often will suffice
    (bounding boxes or spheres)
Setup up collidable objects

```javascript
var movingCube;
var collidableMeshList = []; // objects the movingCube can collide with

// first purple box
var wall = new THREE.Mesh(wallGeometry, wallMaterial);
wall.position.set(100, 50, -100);
scene.add(wall);
collidableMeshList.push(wall);
var wall = new THREE.Mesh(wallGeometry, wireMaterial); // wireframe (not necessary)
wall.position.set(100, 50, -100);
scene.add(wall);
```

Collision Detection Code

```javascript
var originPoint = movingCube.position.clone();

for (var vi = 0; vi < movingCube.geometry.vertices.length; vi++) {

    var localVertex = movingCube.geometry.vertices[vi].clone(); // get vertex coordinates relative to the object
    var globalVertex = movingCube.localToWorld( localVertex ); // convert to world coordinates
    var directionVector = globalVertex.sub( originPoint ); // vertex - origin (vector subtraction)

    // cast a ray from the center of the object through the vertex
    var ray = new THREE.Raycaster( originPoint, directionVector.clone().normalize() ); // normalize to unit vector
    var collisionResults = ray.intersectObjects( collidableMeshList );

    // if we have at least one collision result, and the collision vector is less than the direction vector, HIT
    // note: collisionResults[0].distance is like our "t" value
    if ( collisionResults.length > 0 && collisionResults[0].distance < directionVector.length() ) {
        console.log('HIT'); // or do something else like move the object back to where it was, or delete the collided object
    }
}
```

Modified from demo: http://stemkoski.github.io/Three.js/Collision-Detection.html
Questions

1. Why are collisions easier to detect in 2D graphics?
   *If objects occupy the same pixels, they have collided.*

2. In 3D, why do you need to project rays from both objects to detect all collisions?
   *A corner of one object may intersect with a face of the other.*

3. Which boxes overlap?
   A. X: 2-7, Y: 10-12, Z: 0-4
   B. X: 6-9, Y: 6-9, Z: 3-10
   C. X: 4-8, Y: 8-11, Z: 2-5
   D. X: 7-10, Y: 3-7, Z: 0-4

   *A with C, B with C, and B with D.*
Game Design
A game is a tool designed for meaningful play.

- Meaning emerges from player actions within the game and the game’s response to those actions.
- Responses to actions should be both discernable & integrated.

**Discernable**: Result of game actions is communicated in a perceivable way.

**Integrated**: Player actions have immediate effects but also accumulate and affect the play experience in a continuing way.


[https://b985.fm/introducing-the-dice-gift-exchange/](https://b985.fm/introducing-the-dice-gift-exchange/)
Considerations

Theme: What is the objective of the game?
- Can the player “win” or is it open-ended?
- How is progress perceived and recorded?

Gameplay: What is the primary mechanic of the game?
- What controls are available to the player?
- What sort of view is available?

Production: What ingredients will go into this game?
- What technical challenges must be solved?
- What digital resources are required & where will they come from?
Review

After watching this video, you should be able to...

- Give two reasons for detecting collisions
- Describe two simple methods for checking possible collisions
- Describe and implement an algorithm for detailed 3D collision checks
- Identify important elements in any successful game

Music: [https://www.bensound.com](https://www.bensound.com)