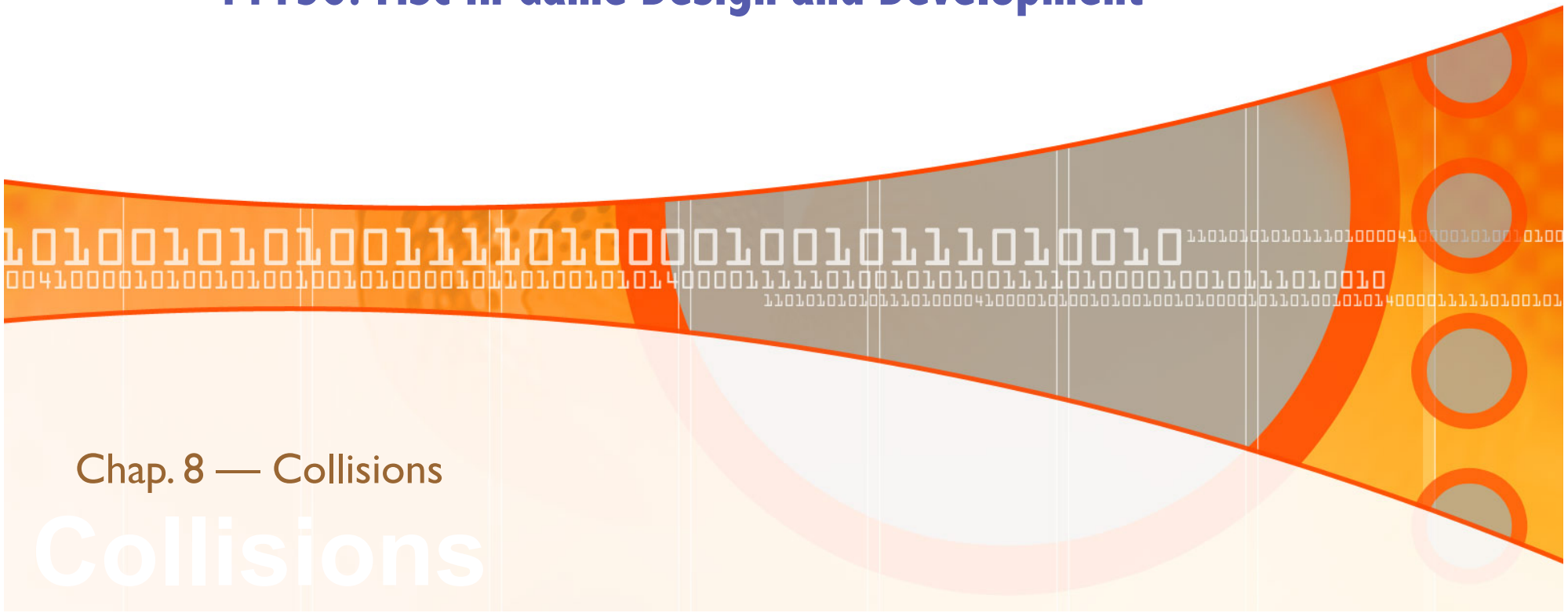
Video Games Technologies

11498: MSc in Computer Science and Engineering
11156: MSc in Game Design and Development



Chap. 8 — Collisions

Collisions

Outline

...:

- Introduction
- Touching and perception
- Collision detection
- Broad phase
- Narrow phase
- Point & mesh relationship

Collision detection: introduction

Remember:
- Physical laws do not exist in the virtual world by default.
- You must model and program them on computer.

***Definition*:**

– Collision detection concerns the detection of collisions between objects in the virtual environment.

***Goal*:**

– To stop objects moving through each other and the environment (see below).

***Why*:**

– Collisions are everywhere in computer games: between characters and characters, between characters and terrain, etc.

Modeling perception in games:

Why? It helps to create the illusion of intelligence.

Senses in games? Sound, Touch, Sight, ~~Smell, Taste~~.

***Examples*:**

- *You approach a bot silently*, from the rear but it immediately turns around and frags you with a chain gun. *Game bots do not really perceive*!
- *You run and hide*. It is impossible for an enemy to know where you are, but it nevertheless proceeds directly to your location.
- *You notice a guard in a guard tower*. It sweeps the ground with a search light. You follow a path that avoids the search light, but the guard still sees you.

Touch & collisions

***Goal*:**

- Detecting when two 3D objects collide.

***Examples*:**

- The player collides with the walls.
- The player walks up stairs.
- A projectile hits a game entity.
- A projectile hits a wall.
- The player collides with a game entity.
- The player goes somewhere they are not supposed to.

Two major issues in collision detection

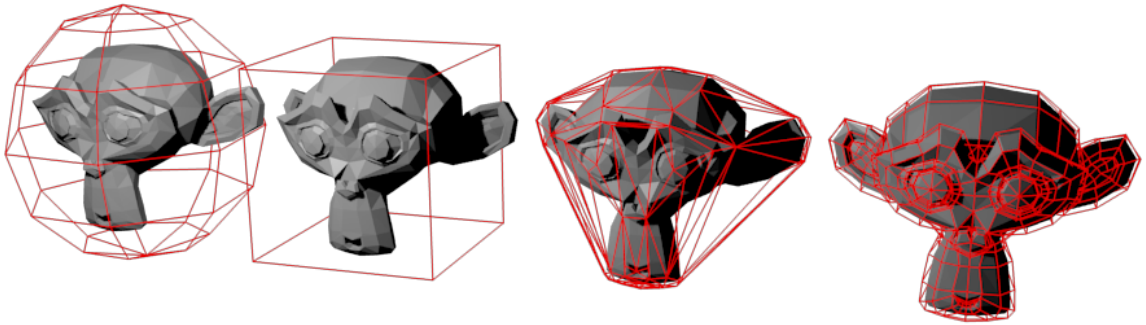
Large scale or object scale collisions:

- If you have *n* objects, object #1 may collide with (*n*-1) objects, object #2 may collide with the remaining (*n*-2) objects, and so forth. Thus, we end up having *n! collisions* potentially.

Narrow scale or polygon scale collisions :

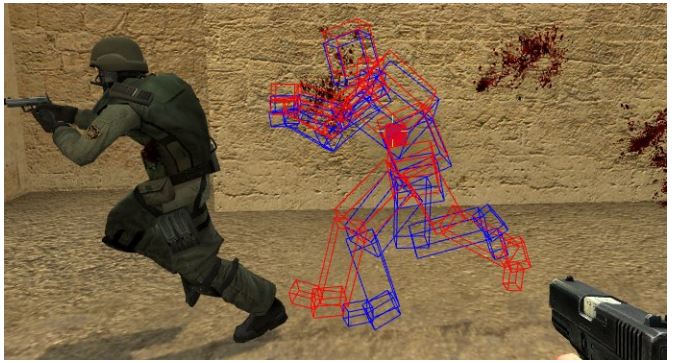
- True collision detection require computing the intersection between arbitrarily complex polygons.

Collision detection between two objects: 2 phases



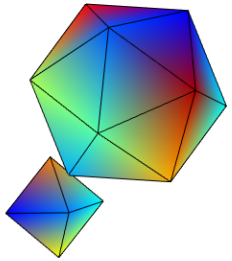
Broad phase:

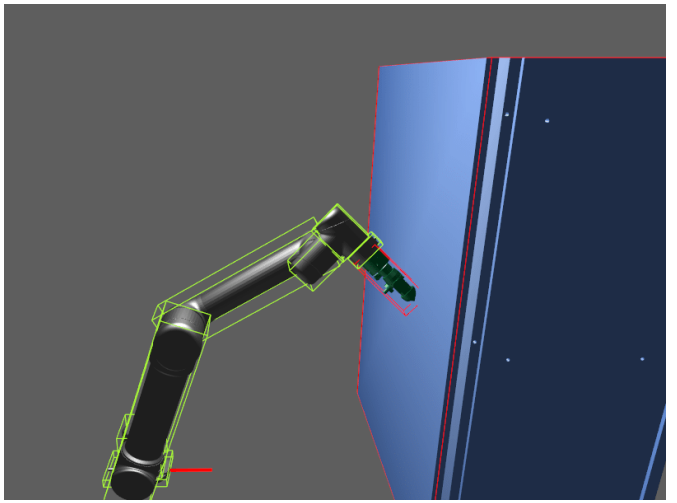
- Collision detection takes place with proxies (bounding volumes).



Narrow phase:

- Collision detection takes place using polygon-polygon intersection.





http://www.opengl-tutorial.org/miscellaneous/clicking-on-objects/picking-with-a-physics-library/

Collision detection algorithms: categories

-*We use bounding boxes/spheres to determine which meshes are colliding (**broad phase**)*

-*At some point we need to compare each triangle of each mesh whose bounding box/sphere collides to check for a true collision (**narrow phase**)*

Broad Phase:

- Point & bounding box
- Point & bounding sphere
- Bounding box & bounding box
- Bounding sphere & bounding sphere
- Box & sphere
- Ray & sphere
- Ray & bounding box

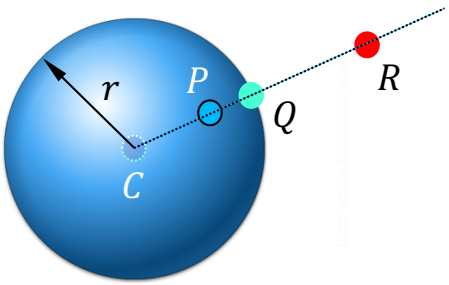
Narrow phase:

- Point & plane (triangle)
- Ray & plane (triangle)
- Plane & plane (triangle & triangle)

Broad Phase

Point & bounding sphere

Euclidean distance: $\|\cdot\|$



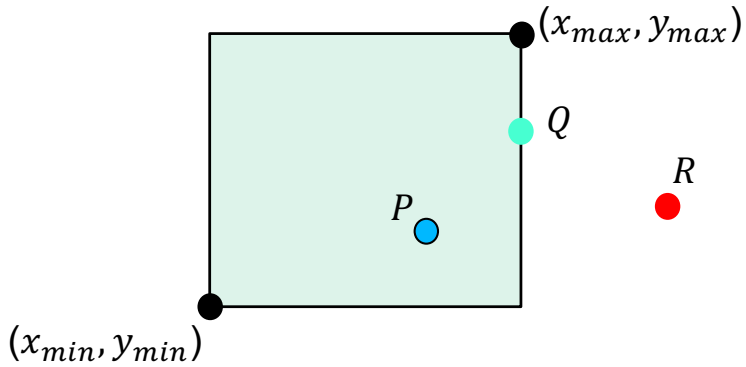
$$\|P - C\| < r \implies P \text{ is inside sphere}$$
$$\|Q - C\| = r \implies Q \text{ is on sphere}$$
$$\|R - C\| > r \implies R \text{ is outside sphere}$$

```
function isPointInsideSphere(point, sphere) {
  // we are using multiplications because is faster than calling Math.pow
  var distance = Math.sqrt((point.x - sphere.x) * (point.x - sphere.x) +
                           (point.y - sphere.y) * (point.y - sphere.y) +
                           (point.z - sphere.z) * (point.z - sphere.z));
  return distance < sphere.radius;
}
```

https://developer.mozilla.org/en-US/docs/Games/Techniques/3D_collision_detection

Point & AABB

Comparison operators: ≥ and ≤
Boolean operators: ∧ and ∨



$(x_P \geq x_{min}) \wedge (x_P \leq x_{max}) \wedge (y_P \geq y_{min}) \wedge (y_P \leq y_{max}) \Longrightarrow P$ is inside AABB
$(x_Q \geq x_{min}) \wedge (x_Q \leq x_{max}) \wedge (y_Q \geq y_{min}) \wedge (y_Q \leq y_{max}) \Longrightarrow Q$ is on AABB
$(x_R \geq x_{min}) \wedge (x_R \geq x_{max}) \wedge (y_R \geq y_{min}) \wedge (y_P \leq y_{max}) \Longrightarrow R$ is outside AABB

```
function isPointInsideAABB(point, box) {
  return (point.x >= box.minX && point.x <= box.maxX) &&
         (point.y >= box.minY && point.y <= box.maxY) &&
         (point.z >= box.minZ && point.z <= box.maxZ);
}
```

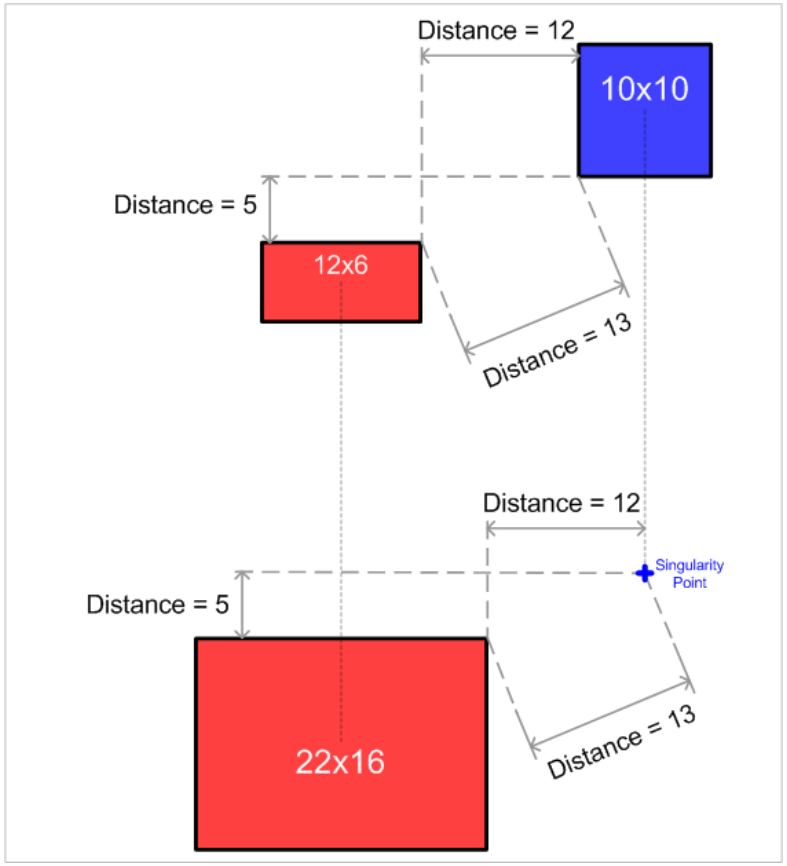
Box-box intersection

Algorithm 1 (box-box intersection):

1) Determine the center of the 2nd box (blue);
2) Add the length of the 2nd box (blue)to the length of the 1st box (red);
3) Add the height of the 2nd box (blue)to the height of the 1st box (red);
4) Check whether the center of the 2nd box (blue) is inside the 1st augmented box (point-box membership)

Algorithm 2 (point-box membership):

```
IN: point P
IN: left-bottom corner L
IN: right-top corner R
OUT: boolean value (either true or false)
1)  if
2)  (
3)      (p.x > L.x) and (p.x < R.x)
4)      and
5)      (p.y > L.y) and (p.y < R.y)
6)  )
7)       return TRUE;
8)  else
9)       return FALSE;
```



Narrow Phase

Geometry refresher

Point:

- $P = (x, y, z)$

2D Line:

- Start and end points: P and Q
- Slope and intercept: $y = mx + b$
- Cartesian: $ax + by + d = 0$

Ray:

- point O and vector $\vec{v}$
- $P(t) = O + t\vec{v}$, with $t[0, \infty[$

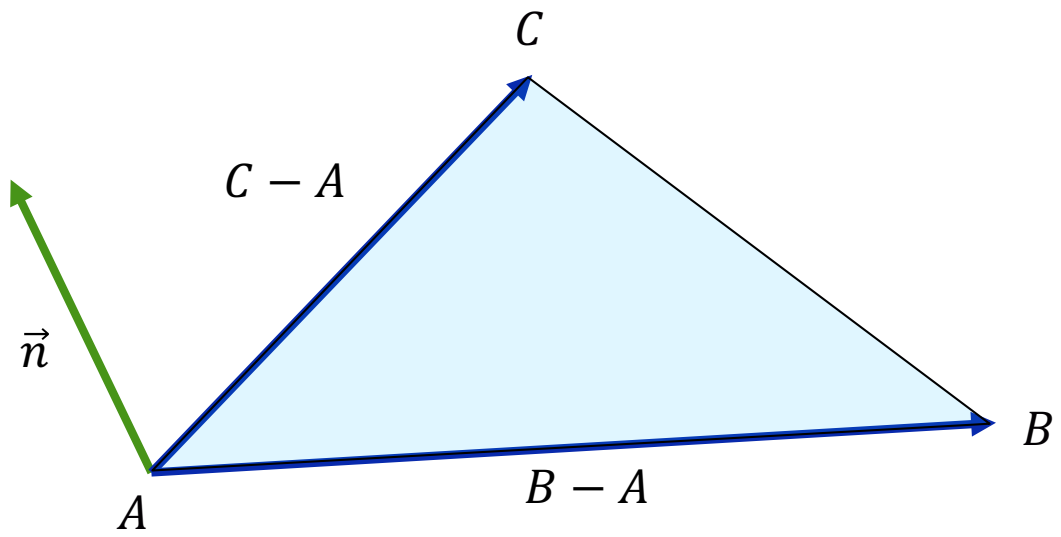
Plane:

- Point P and normal vector $\vec{n}$
- Cartesian: $ax + by + cy + d = 0$
- $\vec{n} = (a, b, c)$

Triangle nomal

Normal vector:

- $\vec{n} = (B - A) \times (C - A)$
- Length of n is twice the area of the triangle ($AB \sin \theta$)



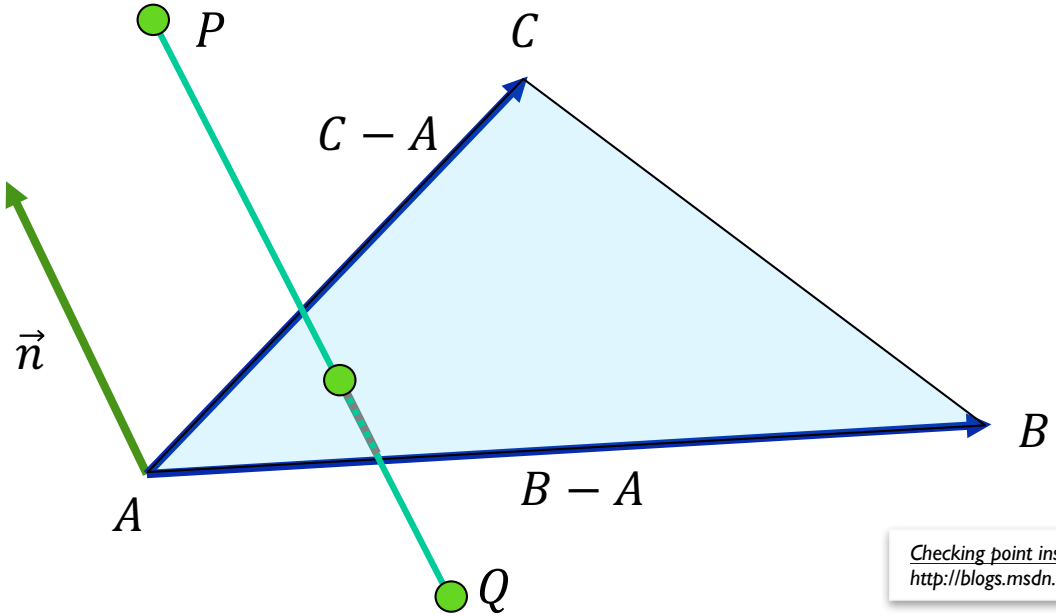
Segment-triangle intersection

Does the segment PQ intersect the triangle ABC?

- $\vec{n} = (B - A) \times (C - A)$
- First, compute signed distances of P and Q to the plane:

  $d(P, X) = (P - A) \cdot \vec{n}$ and $d(Q, X) = (Q - A) \cdot \vec{n}$
- If both are above or both are below the plane, return NULL.
- Otherwise, return the point $X = {Q.d(P,X) - P.d(Q,X)} / {d(P,X) - d(Q,X)}$

http://en.wikipedia.org/wiki/Plane_(geometry)



http://en.wikipedia.org/wiki/Line-plane_intersection

See the proof at:
http://www.di.ubi.pt/~agomes/tjv/proof.pdf

Checking point inside triangle:
http://blogs.msdn.com/b/rezanour/archive/2011/08/07/barycentric-coordinates-and-point-in-triangle-tests.aspx

Triangle-triangle intersection

Input: triangles ABC and A'B'C'

- Step 1: compute the plane equations for ABC:

$$\vec{n} = (B - A) \times (C - A) \quad \text{and} \quad d = -\vec{n} \cdot A$$

- Step 2: compute signed distances from vertices of A'B'C' to the plane of ABC:

$$d_0 = \vec{n} \cdot A' + d \qquad d_1 = \vec{n} \cdot B' + d \qquad d_2 = \vec{n} \cdot C' + d$$

  - Return NULL if all $d_i < 0$ or all $d_i > 0$

- Step 3: repeat Step 2 for vertices of ABC against the plane of A'B'C'
- Step 4: determine intersection points X and Y
- Step 5: Determine if segment XY is inside triangle or intersects triangle edge.

Summary:

...:

- Introduction
- Touching and perception
- Collision detection
- Broad phase
- Narrow phase
- Point & mesh relationship