Collisions

Some very simple collision checks

BACKGROUND

- In many cases, games require checking whether two objects collide.

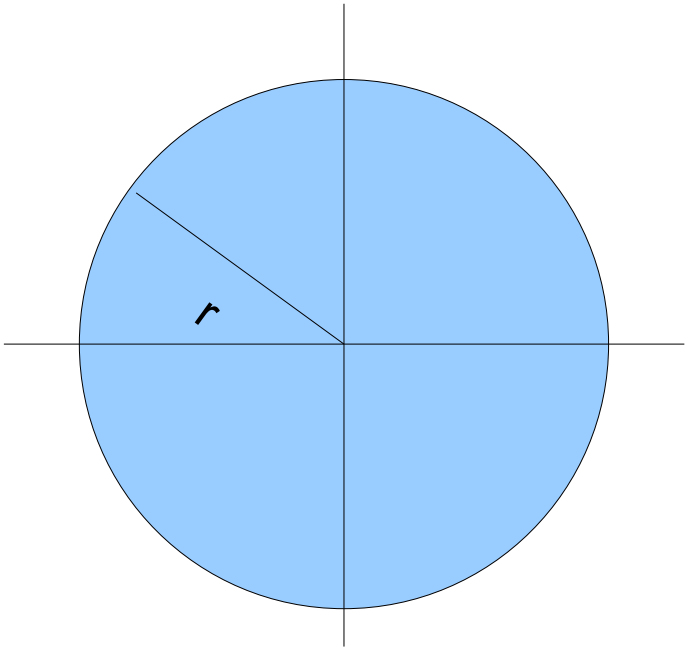
- There's two parts to a collision:
  - Collision detection
  - Collision response

COLLISION DETECTION

- Collision detection is basically checking whether objects overlap in their destination coordinates.
- Usually it's enough to check whether simplified collision hulls collide
  - Bounding box (or rectangle in 2d)
  - Bounding sphere (or circle in 2d)
- In many cases, axis aligned bounding boxes (or rectangles) are enough.

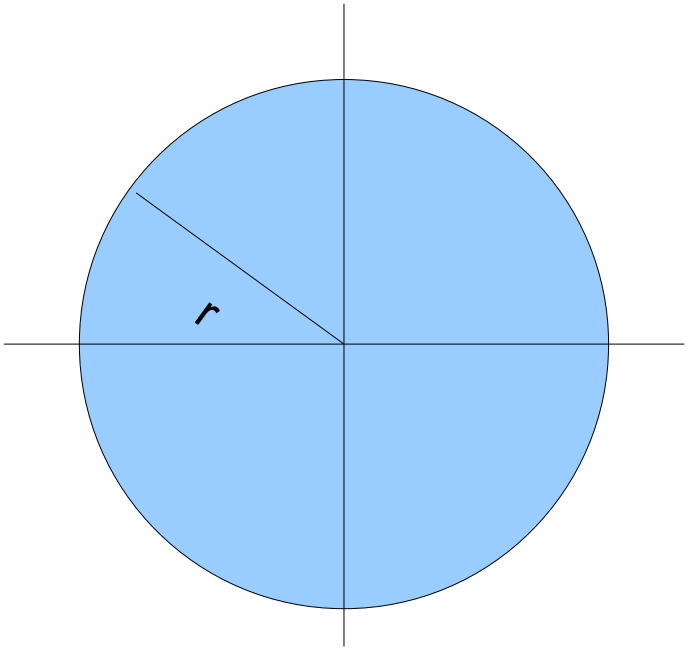
CD: POINT-CIRCLE

- Point – circle collision detection:
  - Calculate distance between circle center and point.
  - Check if distance is less than radius.



CD: CIRCLE-CIRCLE

- Circle – circle collision detection:
  - Calculate distance between circle centers.
  - Check if distance is less than sum of circle radii.



CD: POINT-RECT

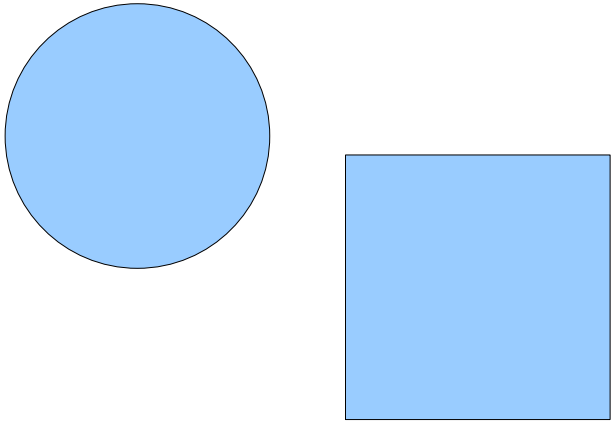
- Point – axis aligned rect collision detection:
  - Point.Y < rect.top? No collision.
  - Point.Y > rect.bottom? No collision.
  - Point.X < rect.left? No collision.
  - Point.X > rect.right? No collision.
  - Otherwise, collision.

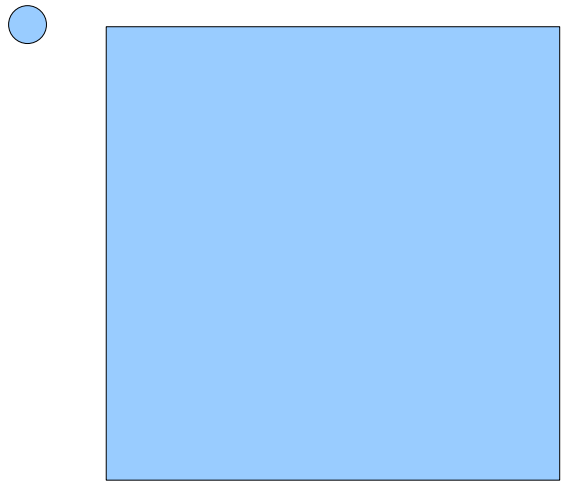
CD: RECT-RECT

- Collision between axis aligned rects:
  - rectA.bottom < rectB.top? No collision.
  - rectA.top > rectB.bottom? No collision.
  - rectA.right < rectB.left? No collision.
  - rectA.left > rectB.right? No collision.
  - Otherwise, collision.

CD: CIRCLE-RECT

- Reduce circle to point, taking radius and growing rectangle with it.
- Check for collision via point-rect check.





COLLISION RESPONSE

- Response, when collision is detected, depends on what is colliding and what is the desired result.
  - Player + missile -> remove missile, reduce health
  - Player + wall -> player slides along a wall

EXAMPLE: WALLS

- Find collision.
- Find wall normal.
- Figure out collision penetration depth.
- Push object back outside wall in wall's normal direction.