Game AI.

Part 1

https://www.youtube.com/watch?v=GLKSuuOvnTw 1

Basic AI components.

- Sensing
- Pathfinding
- States

Sensing

Sensing the player.

Sensing the world.

Pathfinding

Navigating the level.

States

Normal

Angry


104/106

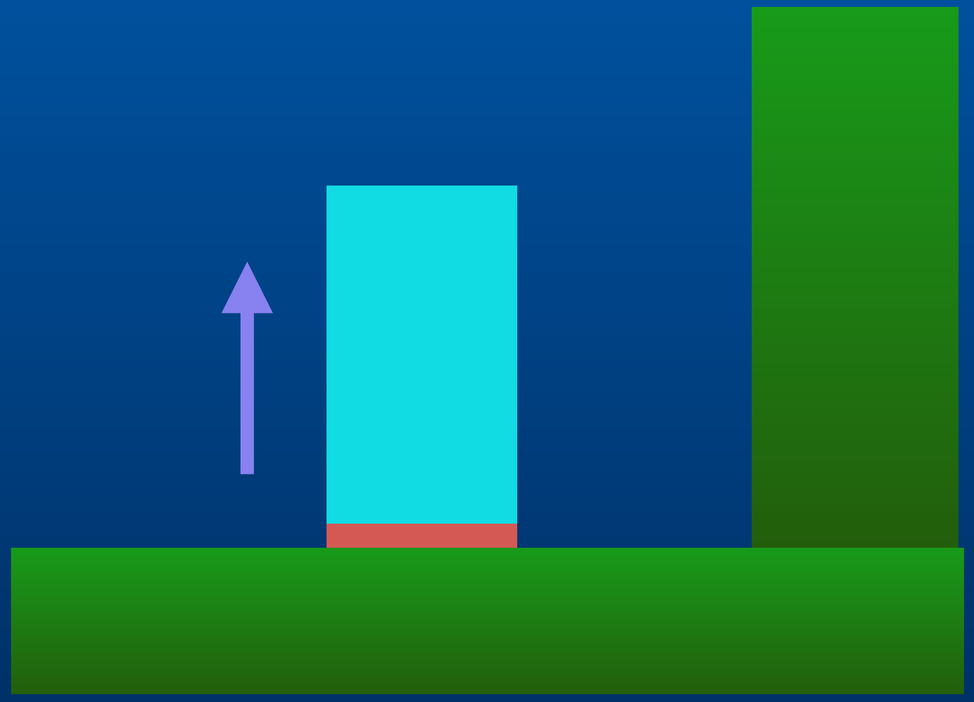

Z
Zz

Sensing basics.

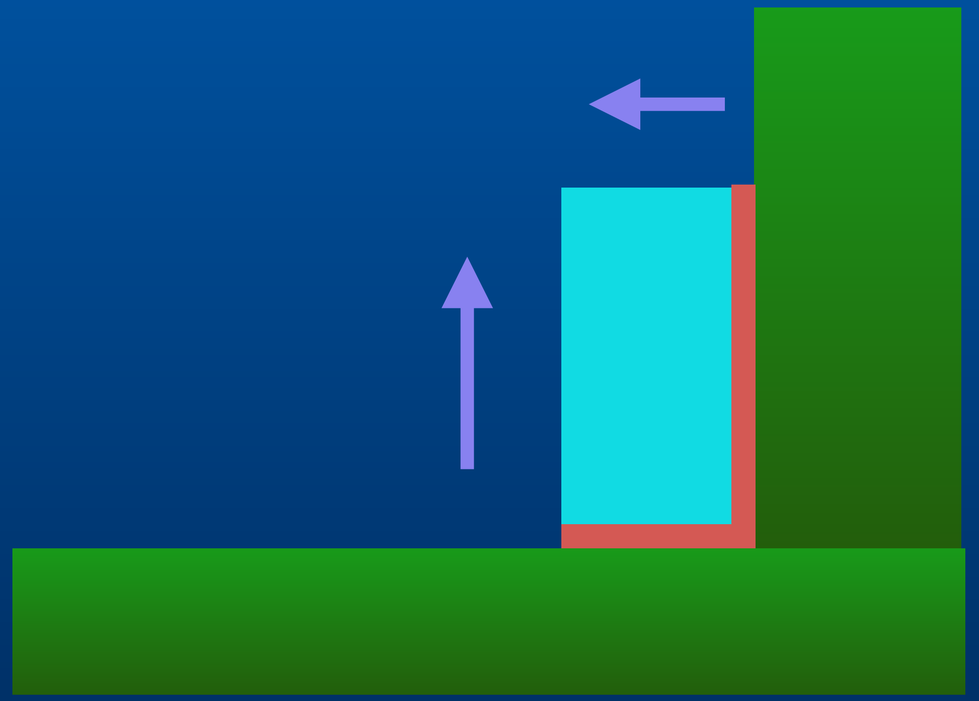
Collision sensing.

Collision flags

When we do our adjust, set collision flag for that direction to true.



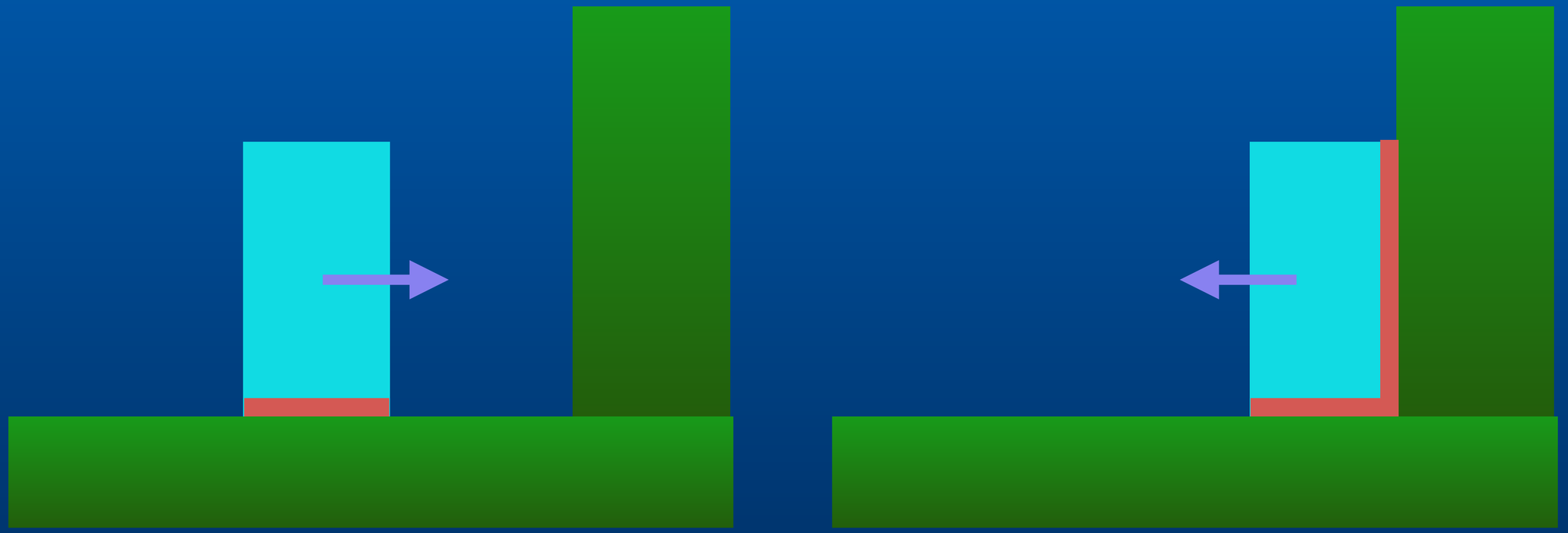
collidingBottom = true



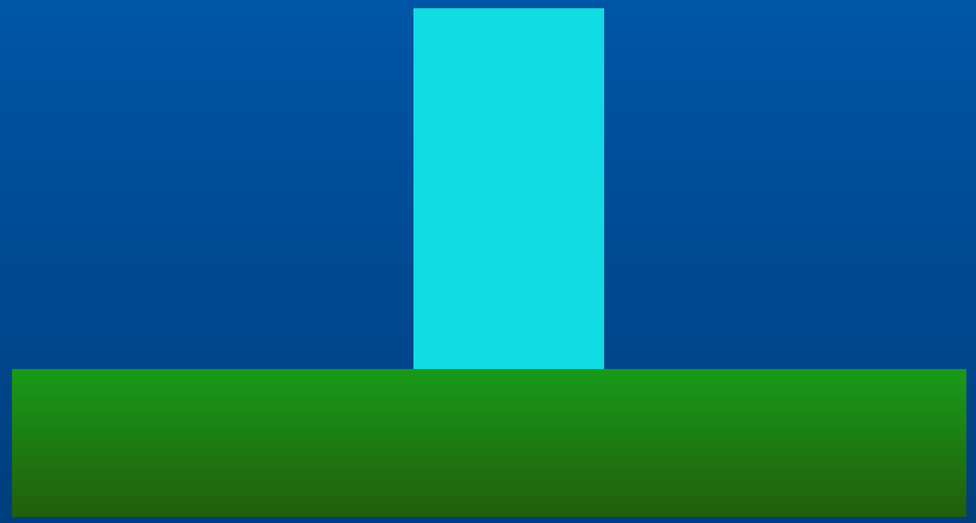
collidingBottom = true

collidingRight = true

Collision flag sensing example: turning around at a wall.

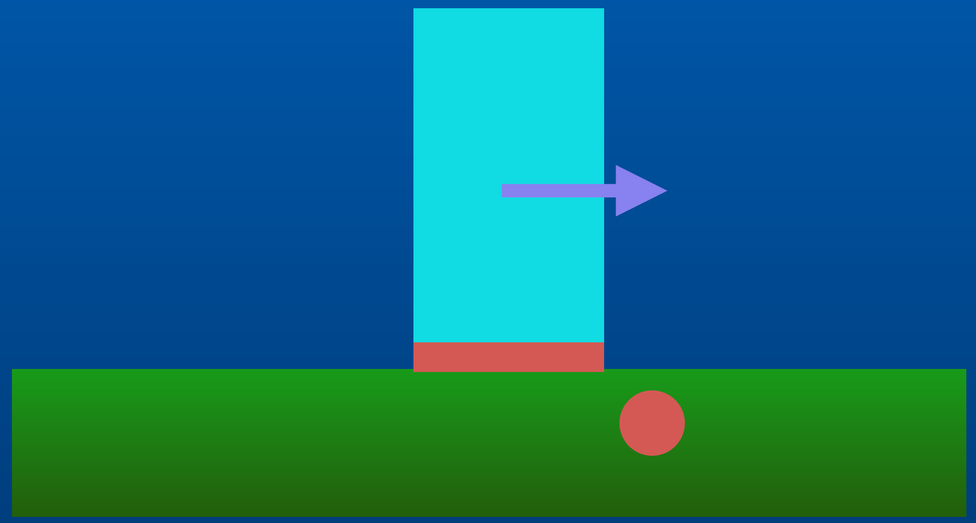


When an X-axis collision flag is set, inverse the X acceleration.



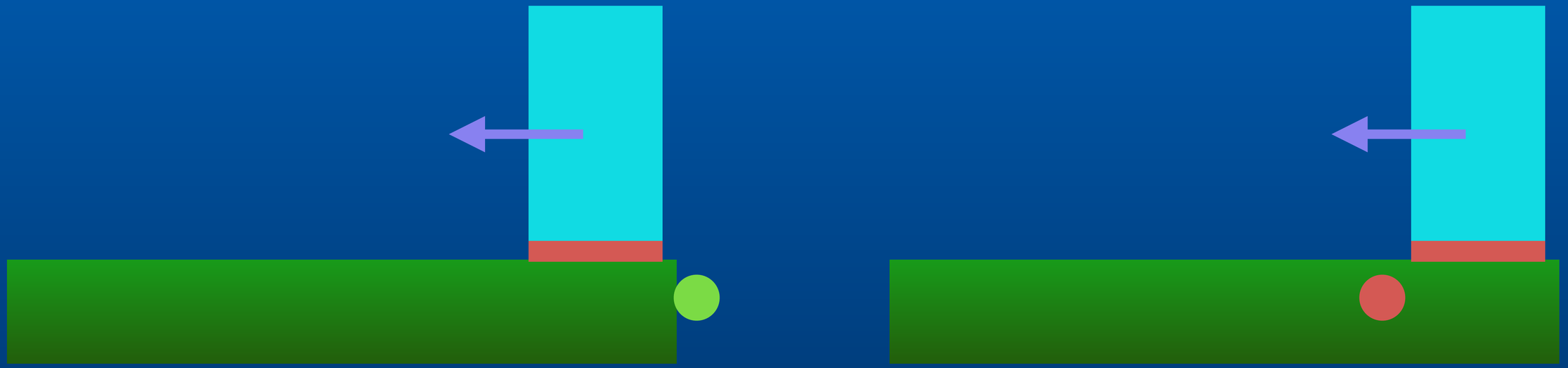
Problem: turning around at an edge.

Collision sensors.

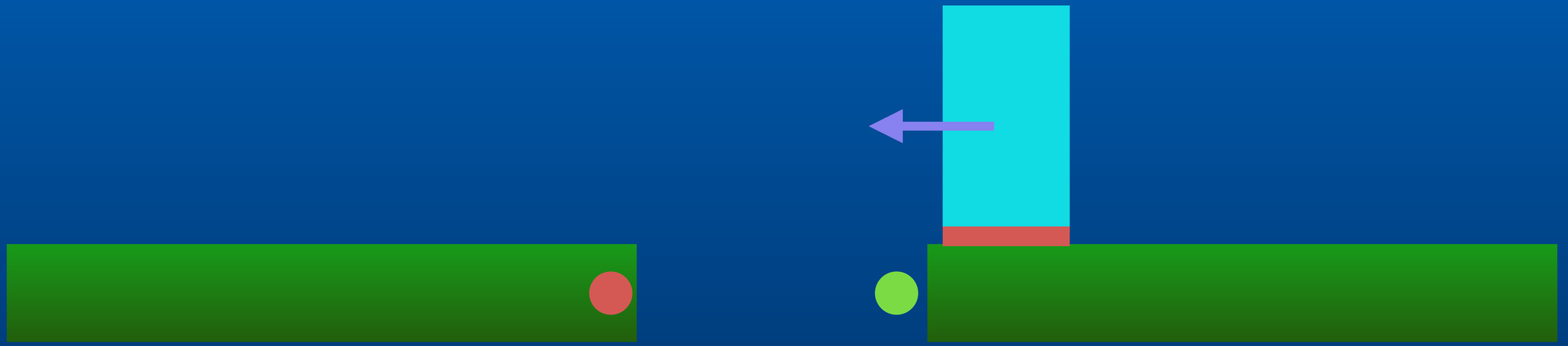


Check for collision using a detector point.

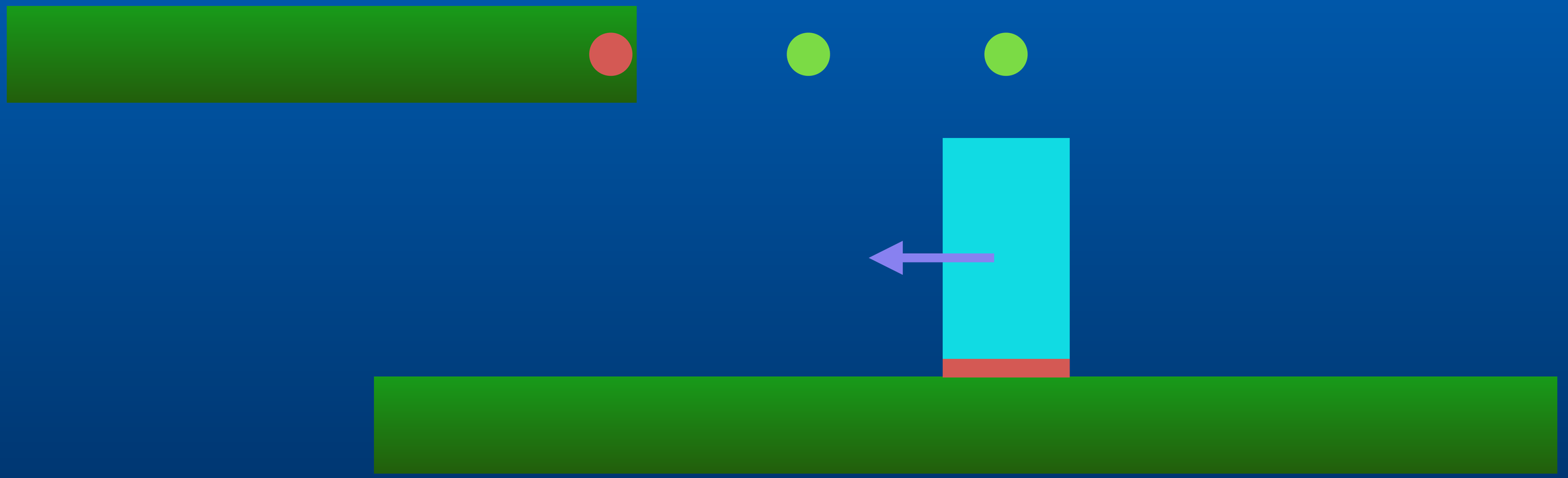
(worldToTileCoordinates)



If detector point not colliding and our bottom collision flag is set (don’t want to set it off when jumping), we’ve reached an edge!

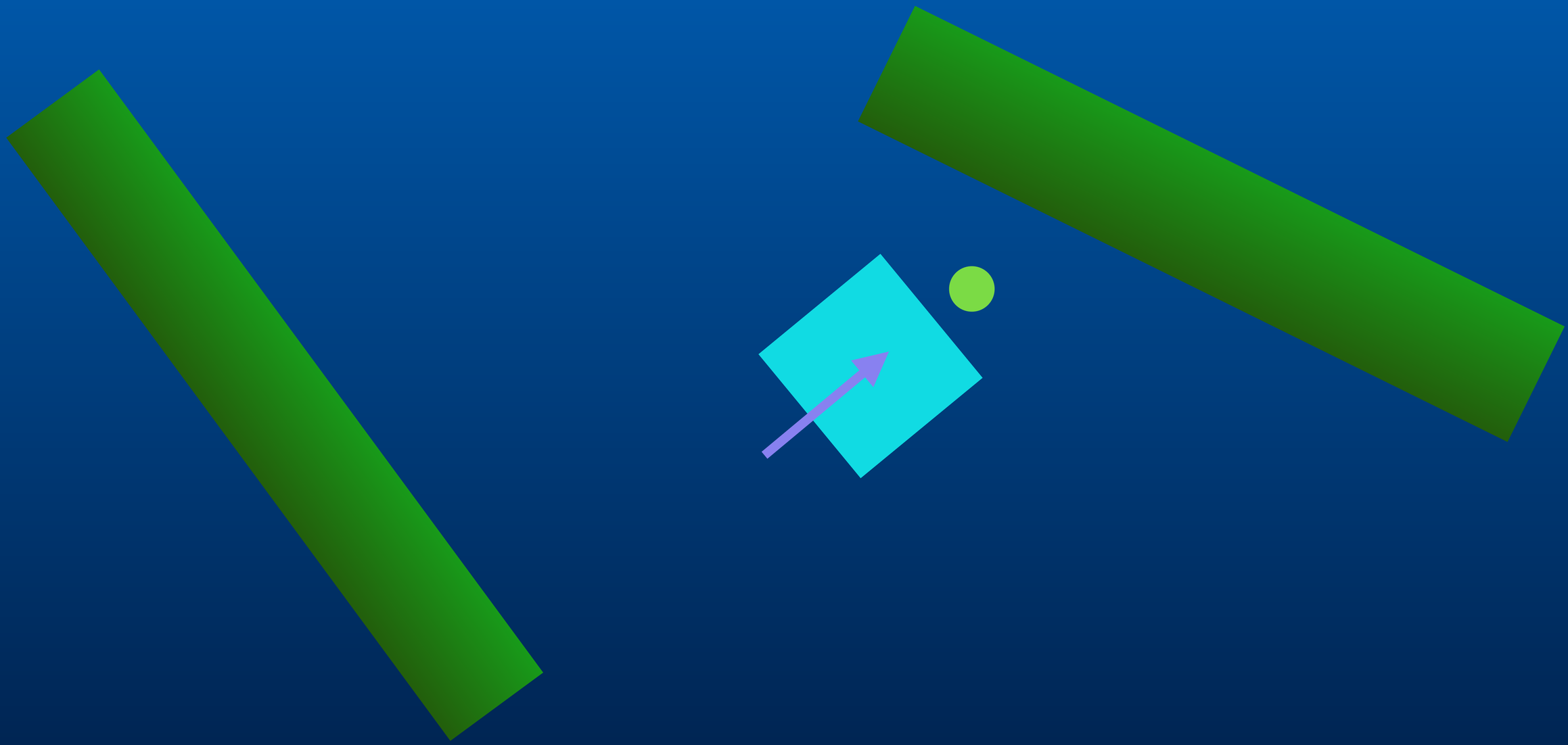


Can use a sensor point to see if we can jump across a gap.

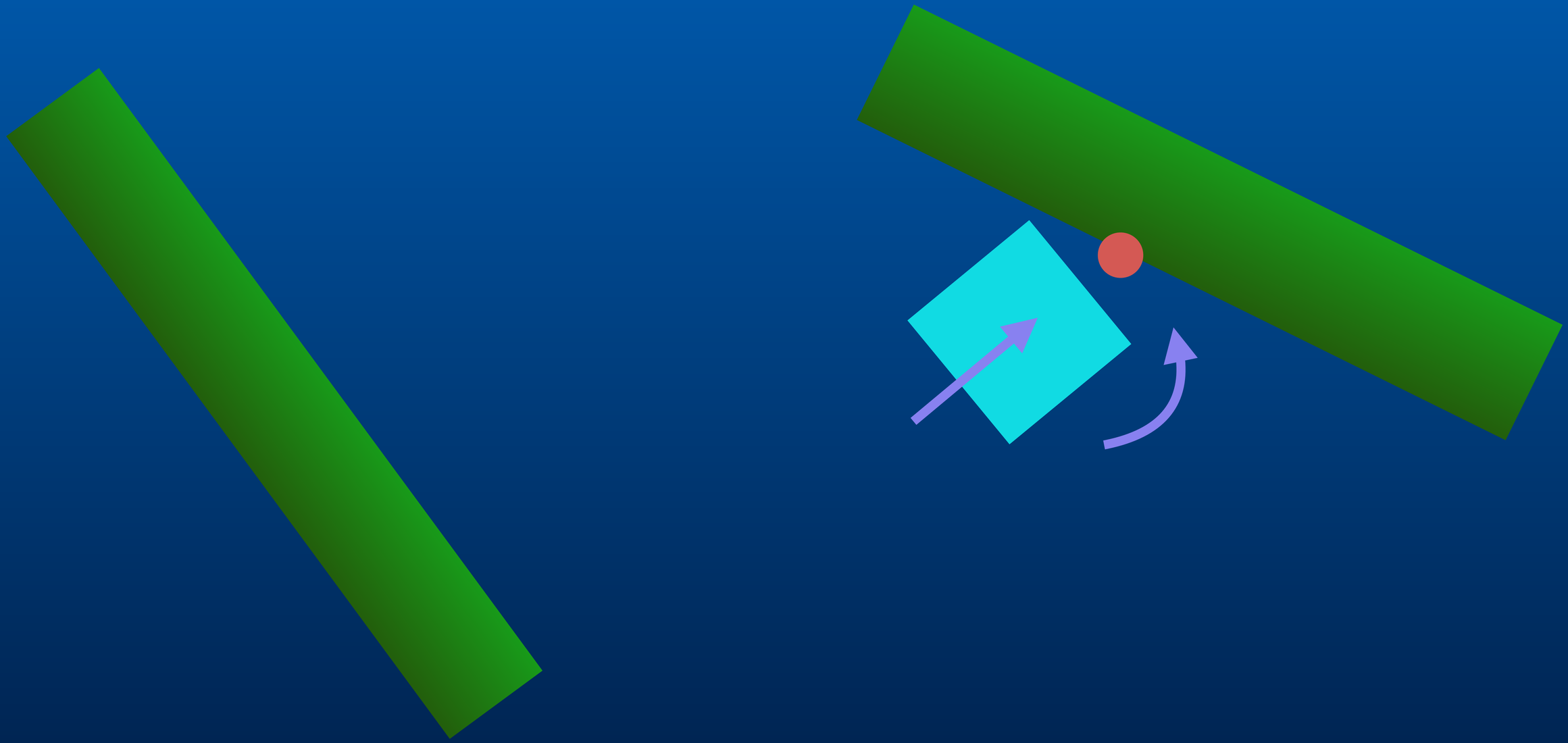


Can use 2 or 3 sensor points to see if possible to jump up to a platform.

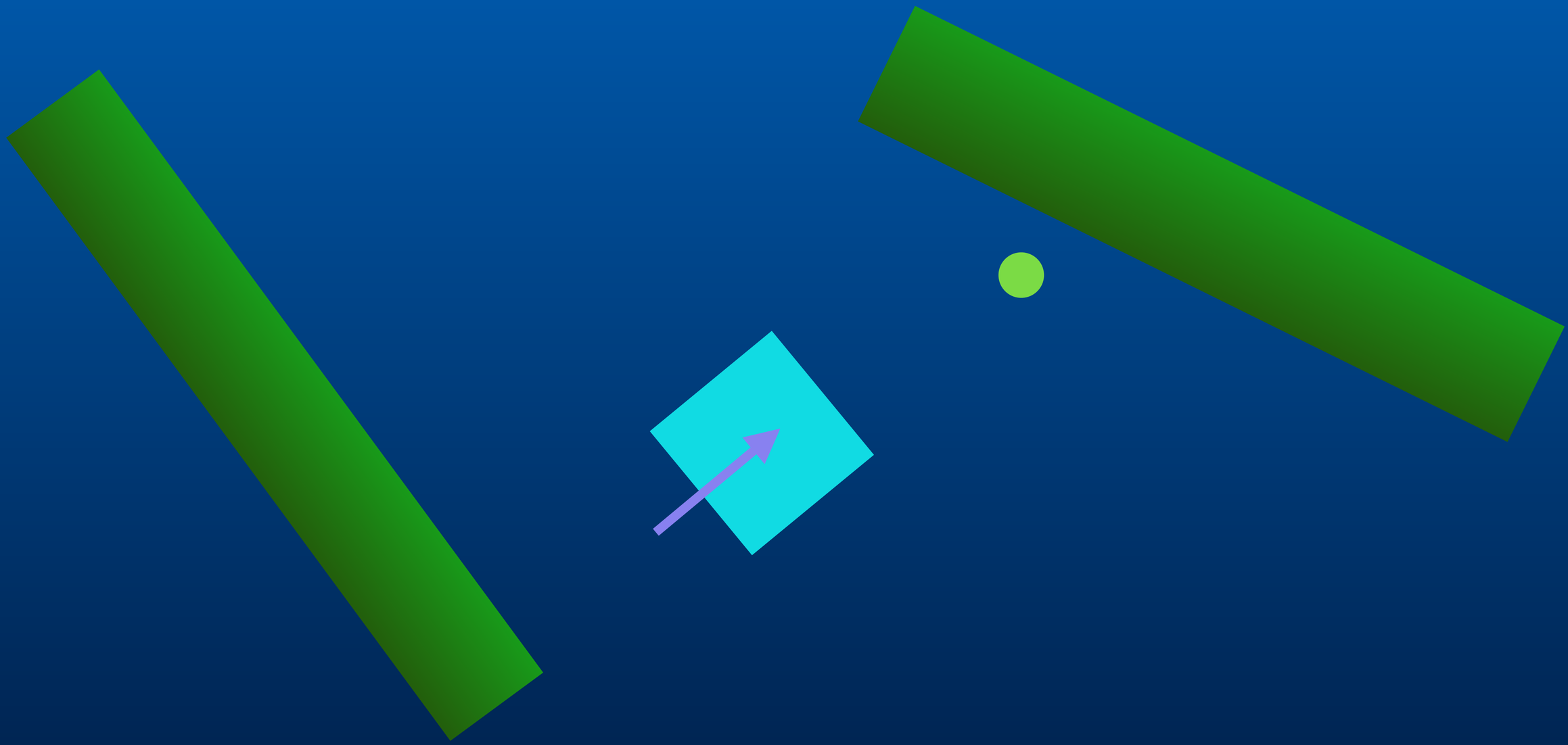
Collision sensors in non-axis aligned collisions.

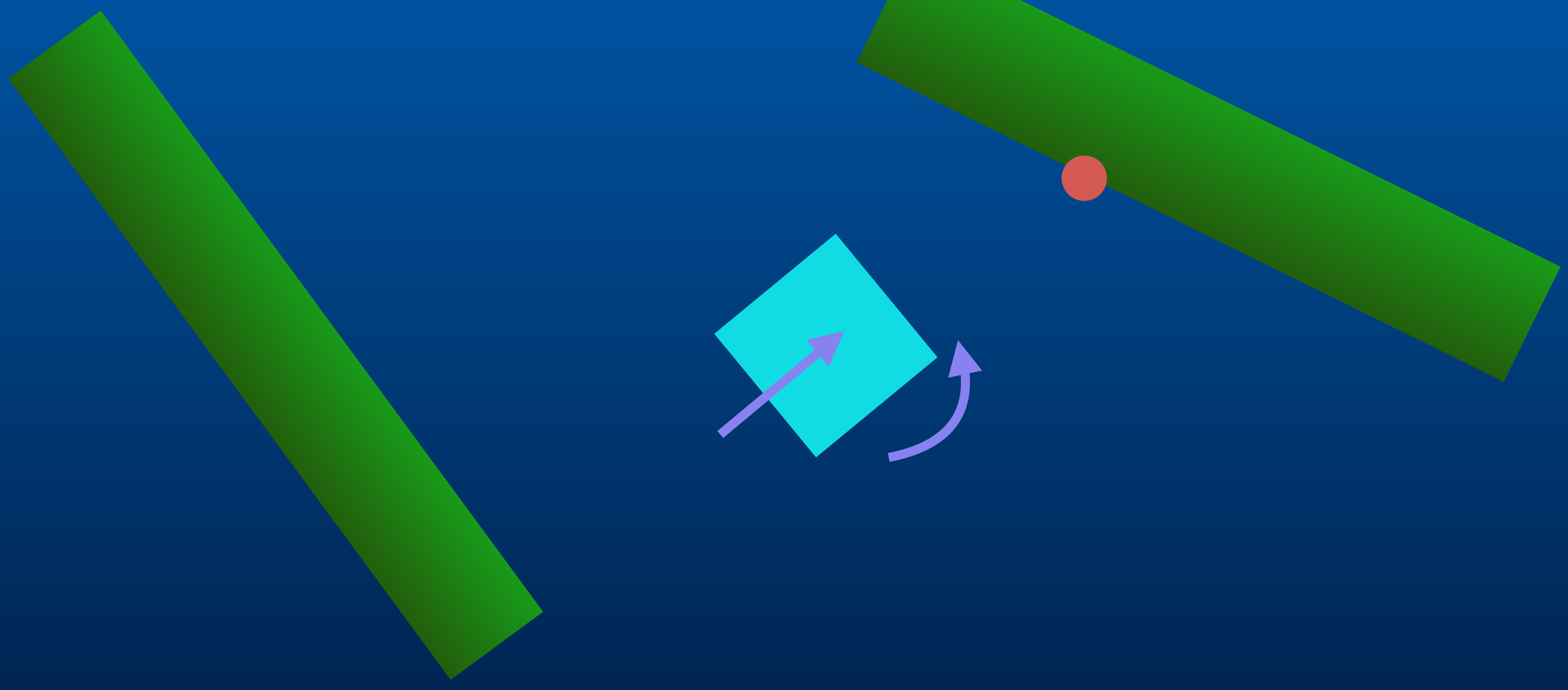


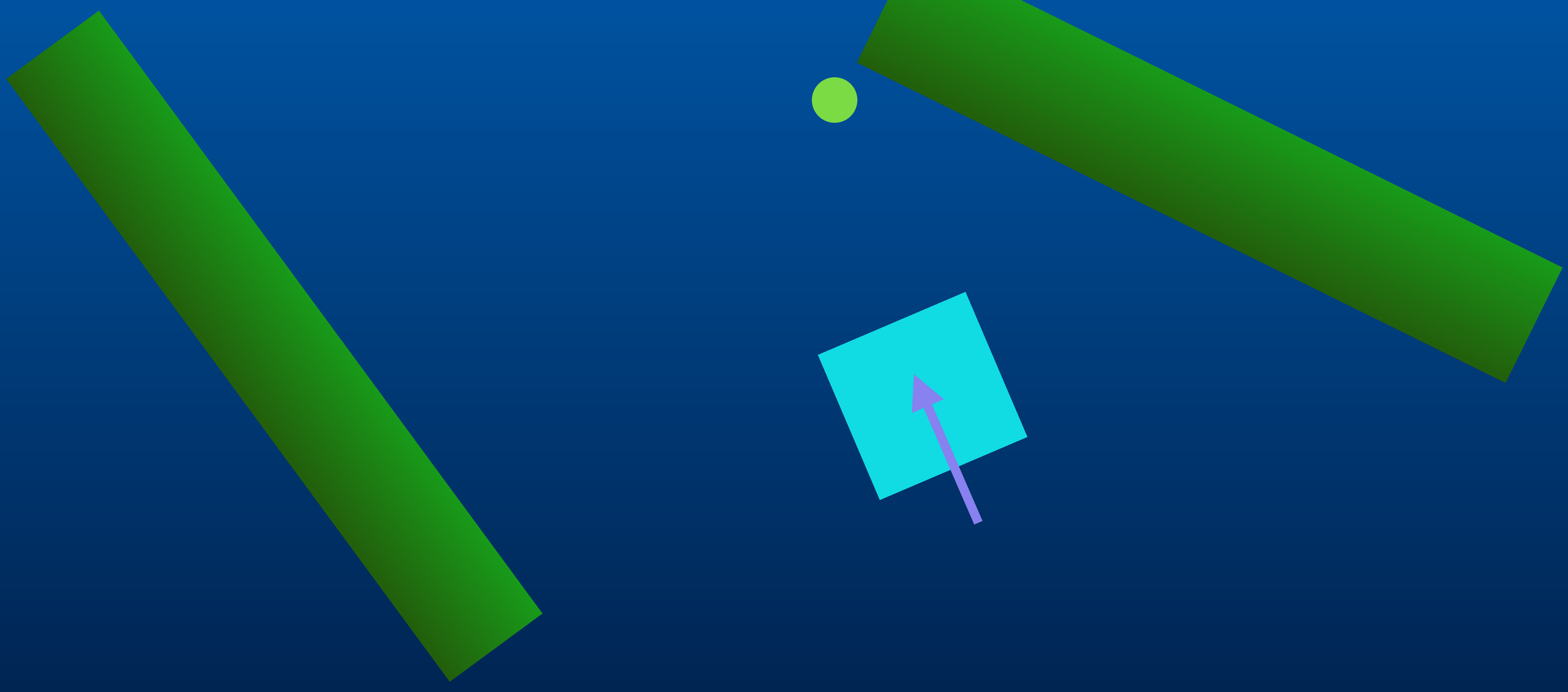
Point in rotated rectangle collision detection.



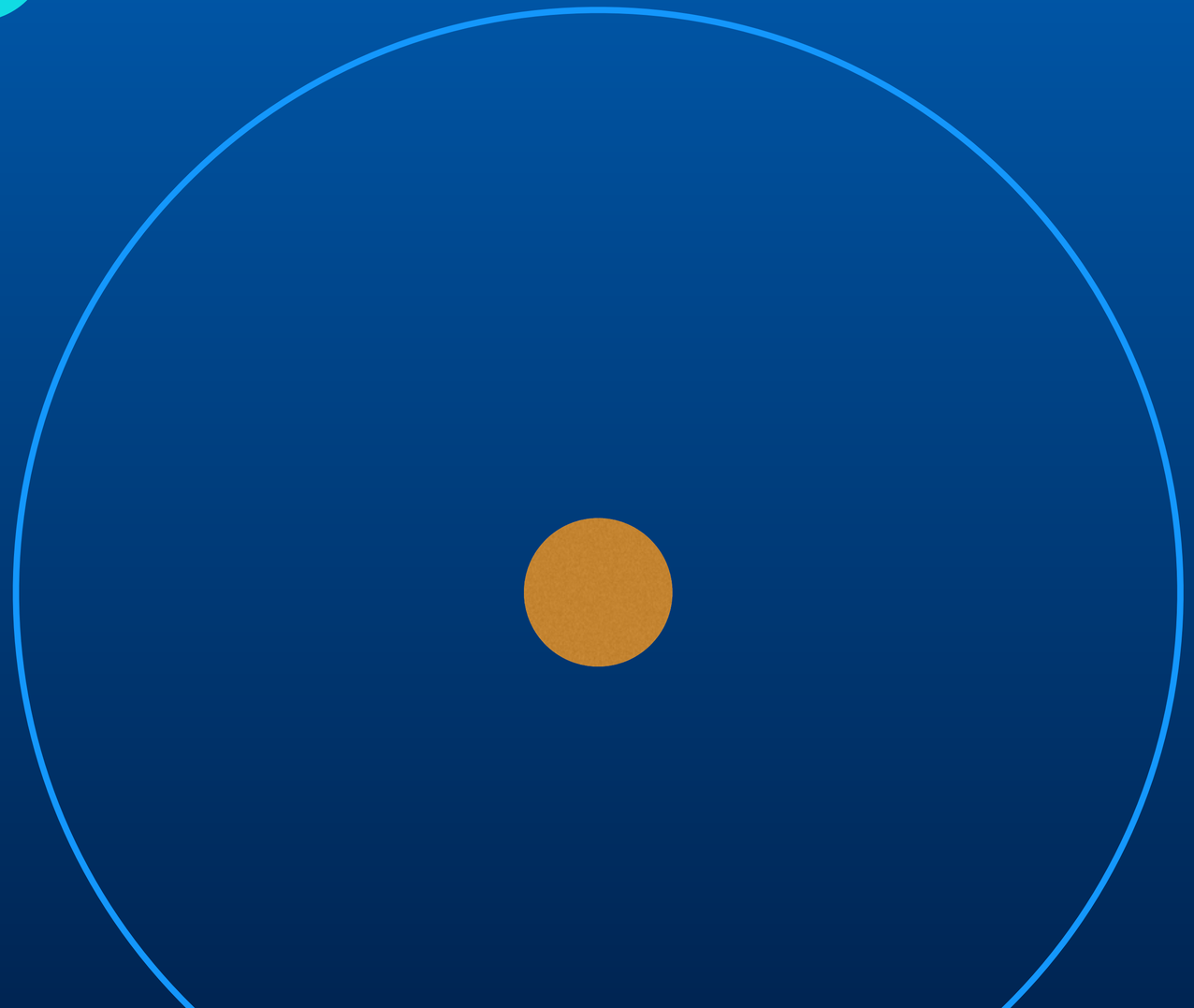
Changing sensor distance.

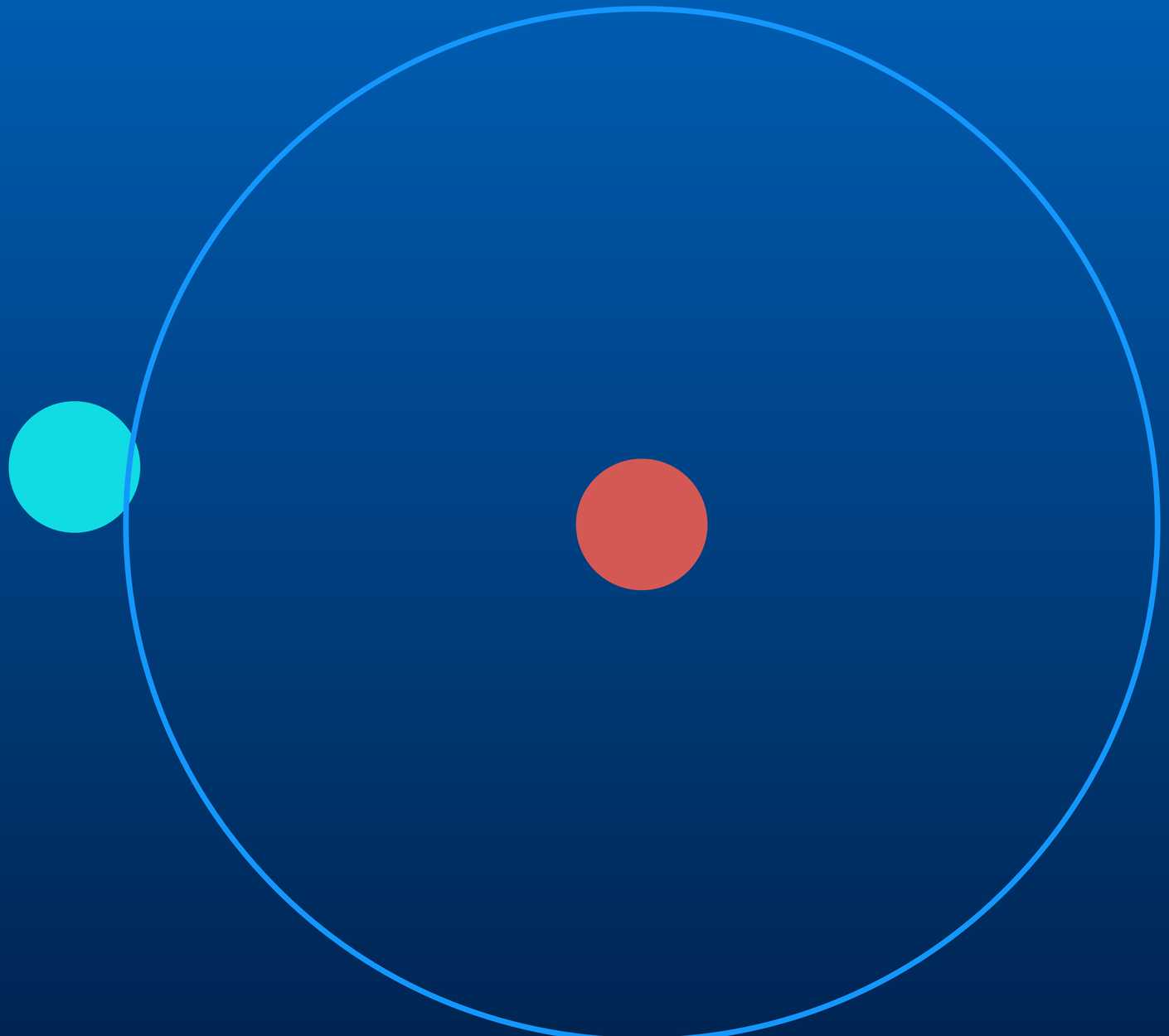




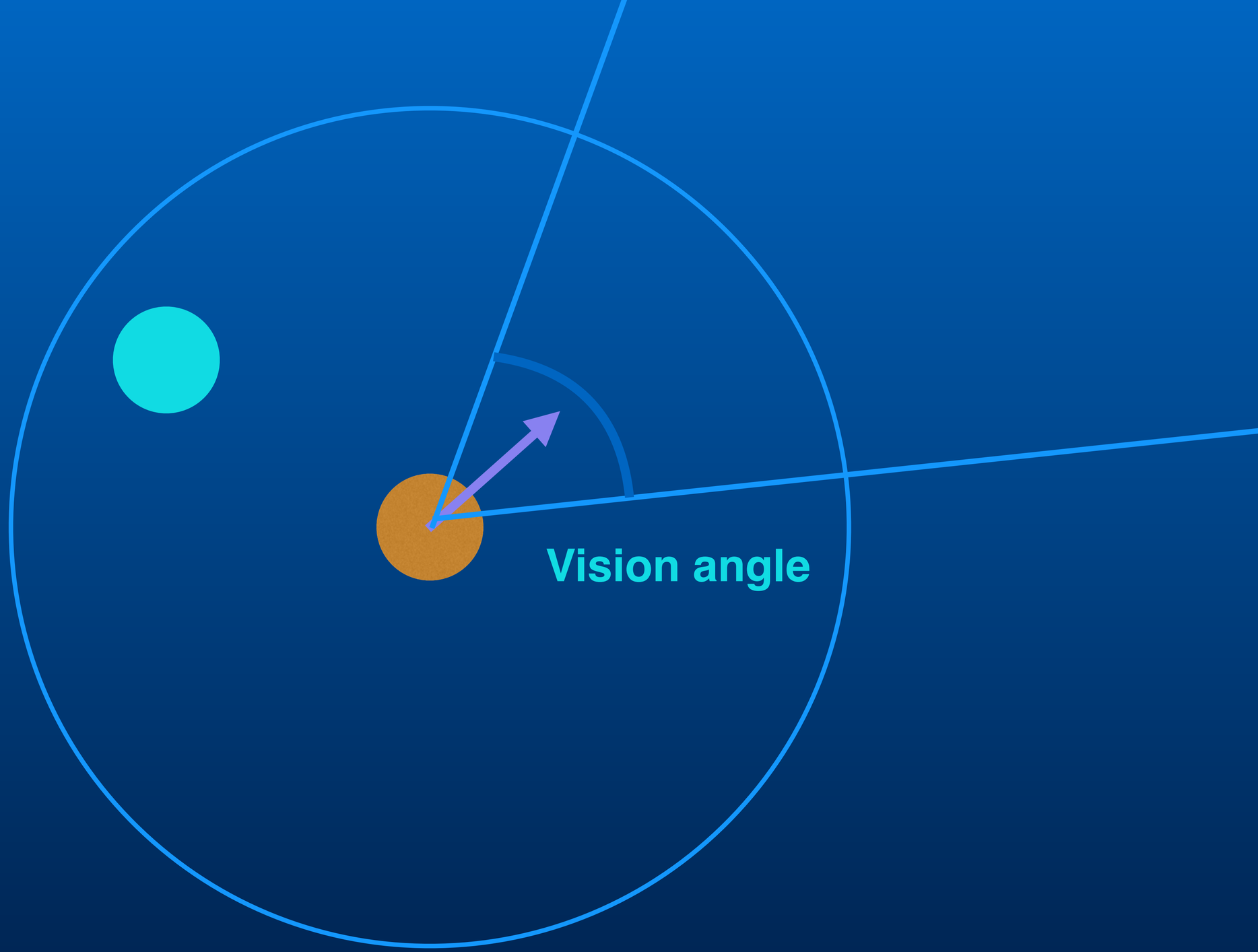


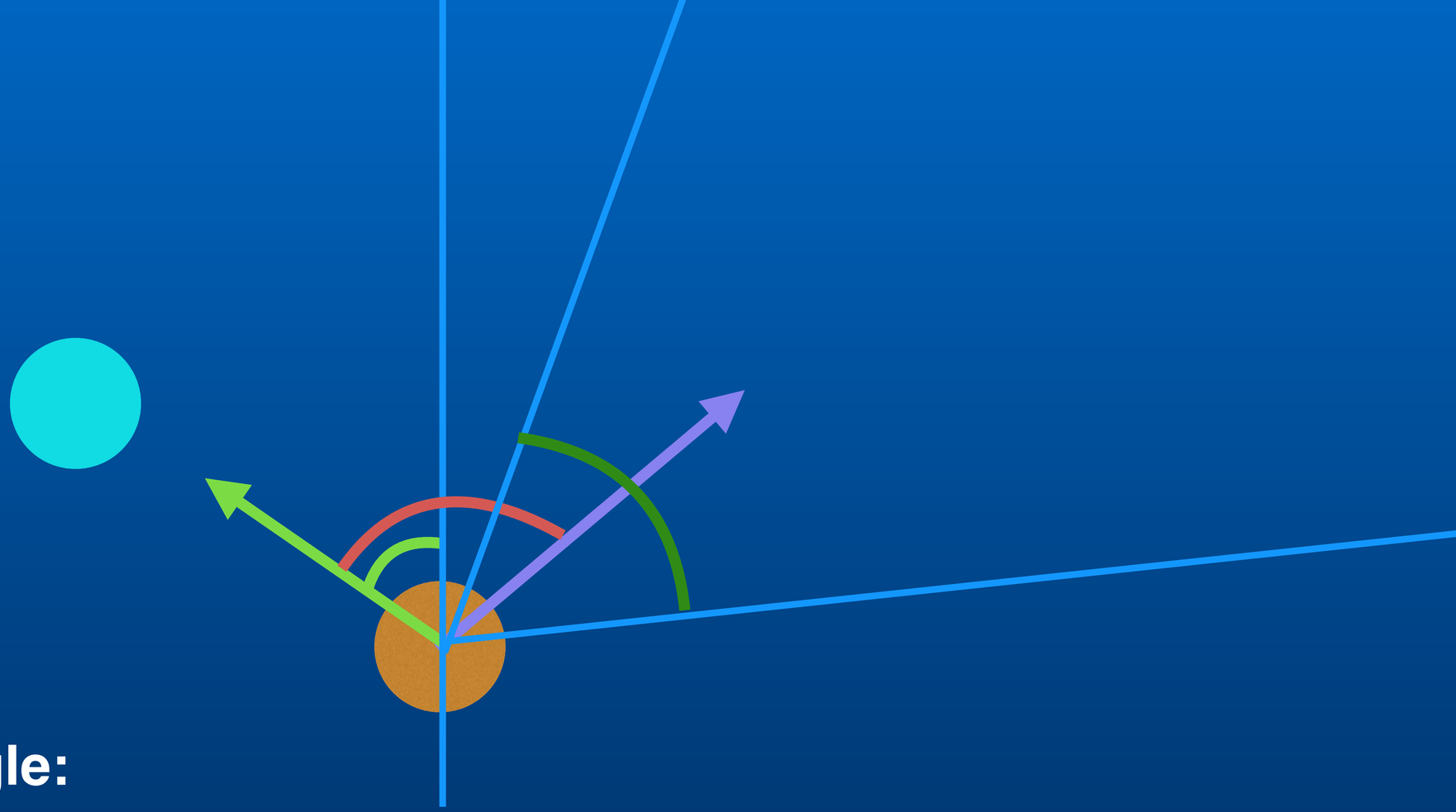
Simulating senses.





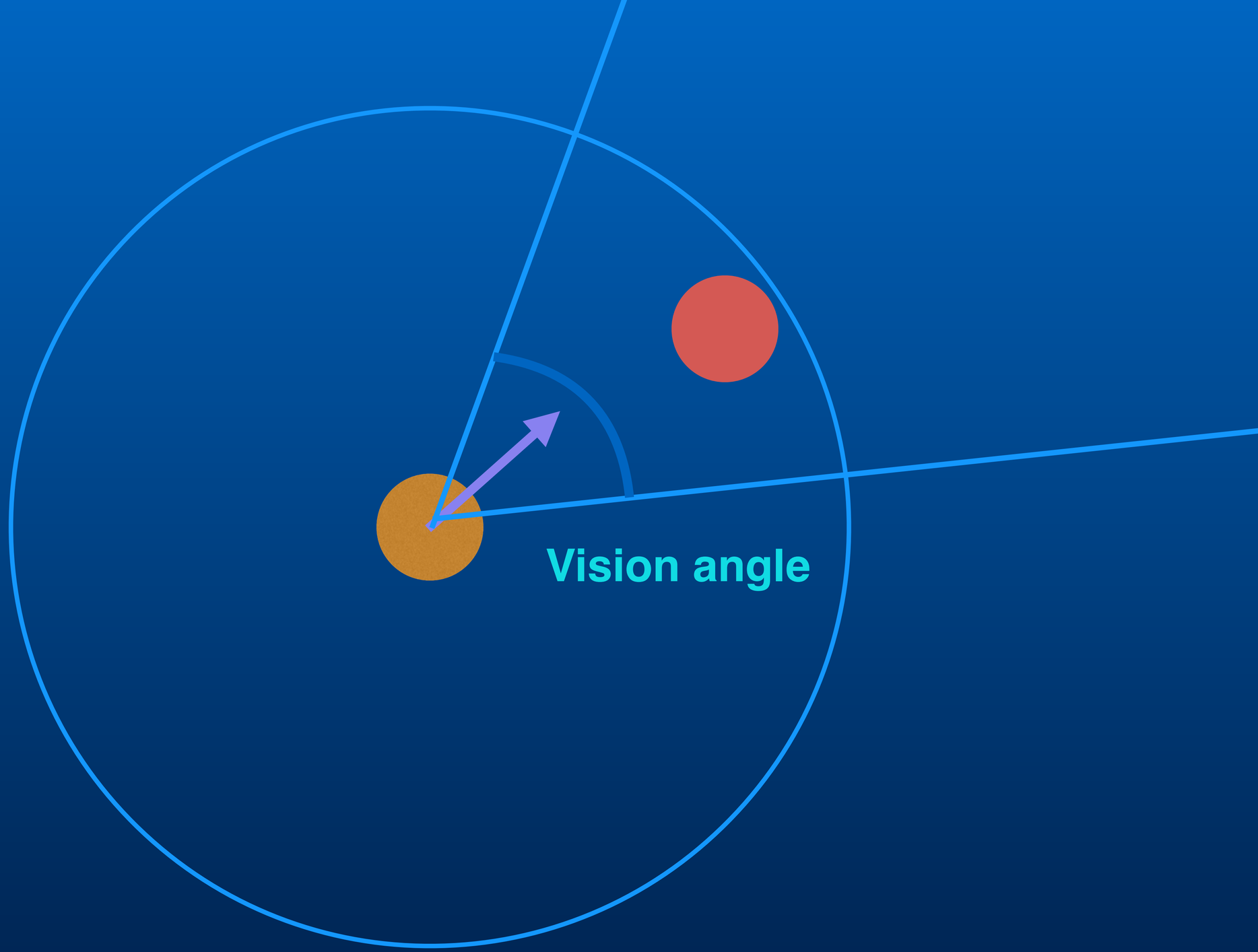
Vision cone.

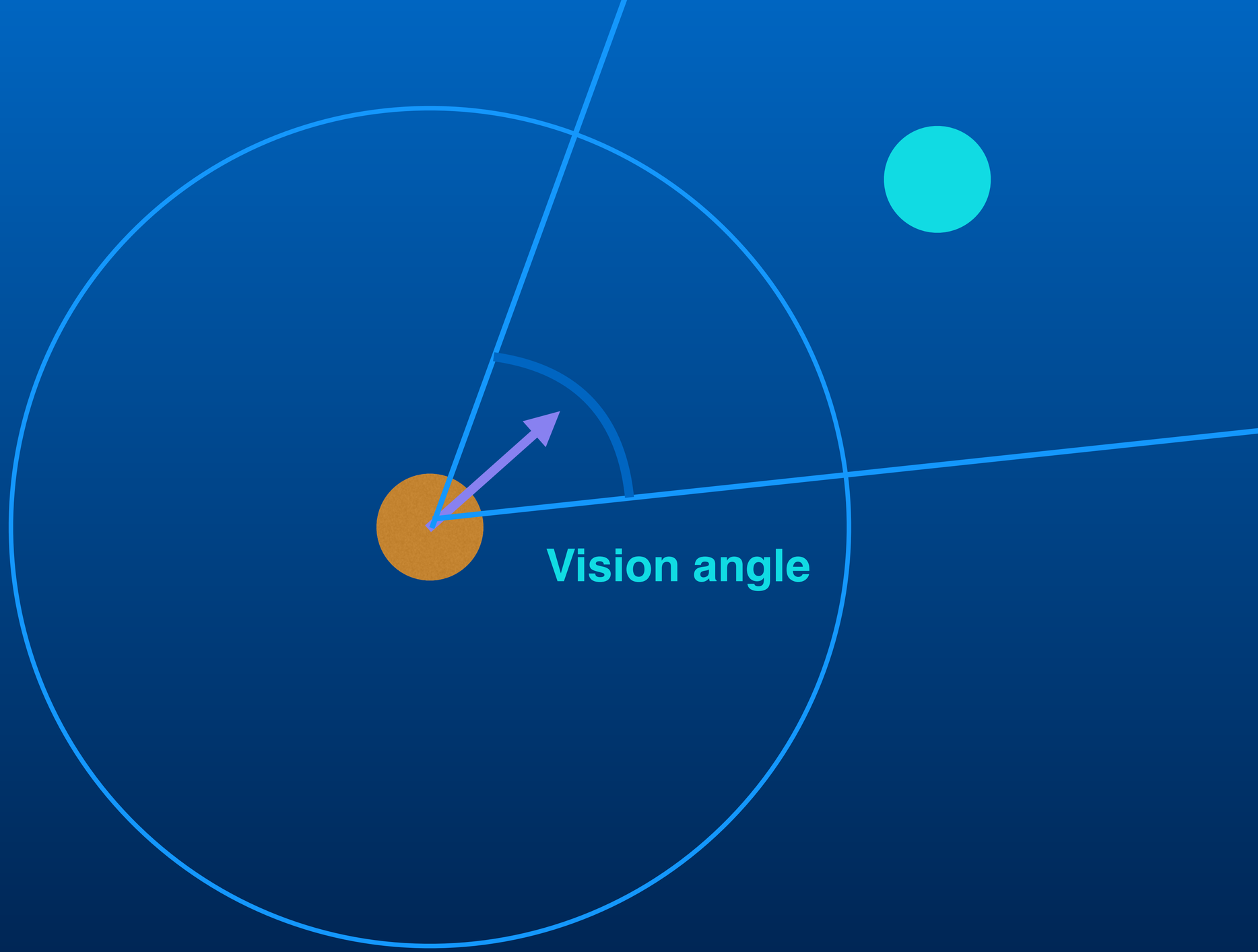

Vision angle



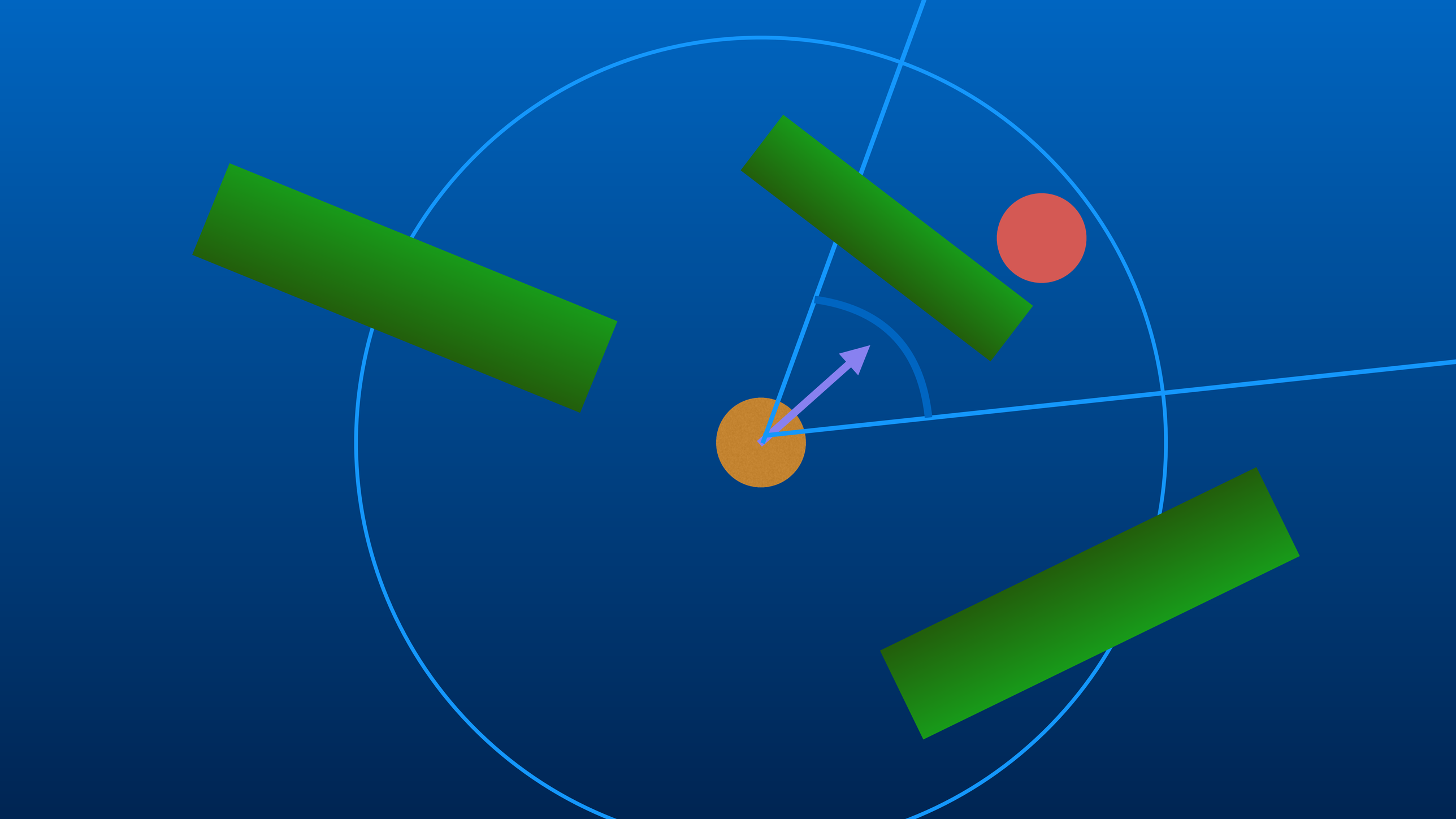
Calculating vision angle:

1. Calculate angle towards object. **atan2(normalizedDirectionToObject)**
2. Get difference between the rotation angle and angle towards object.
3. If absolute value of the difference is larger than half of our vision angle, we can see the object


Vision angle

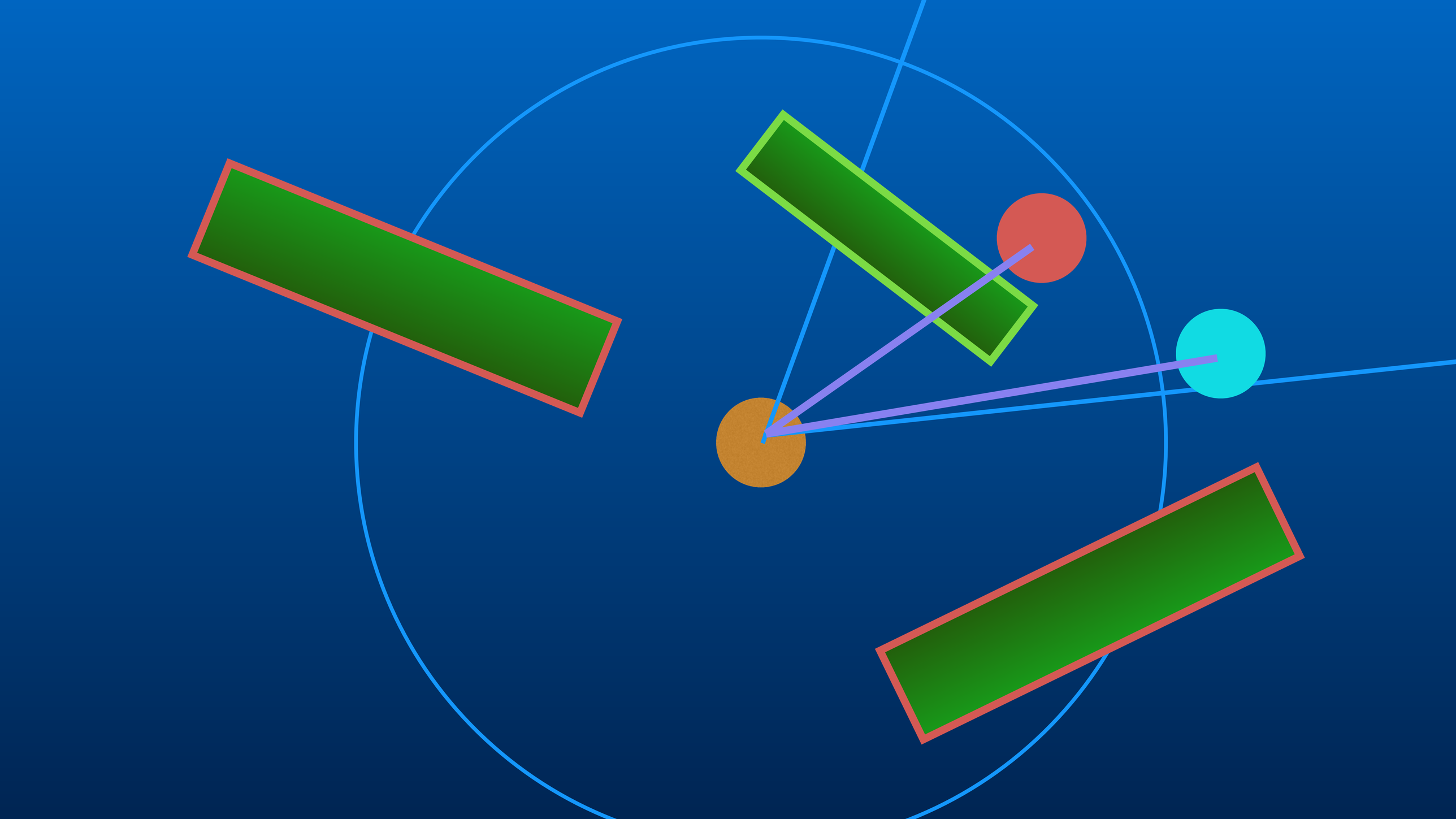

Vision angle

Vision obstacles.

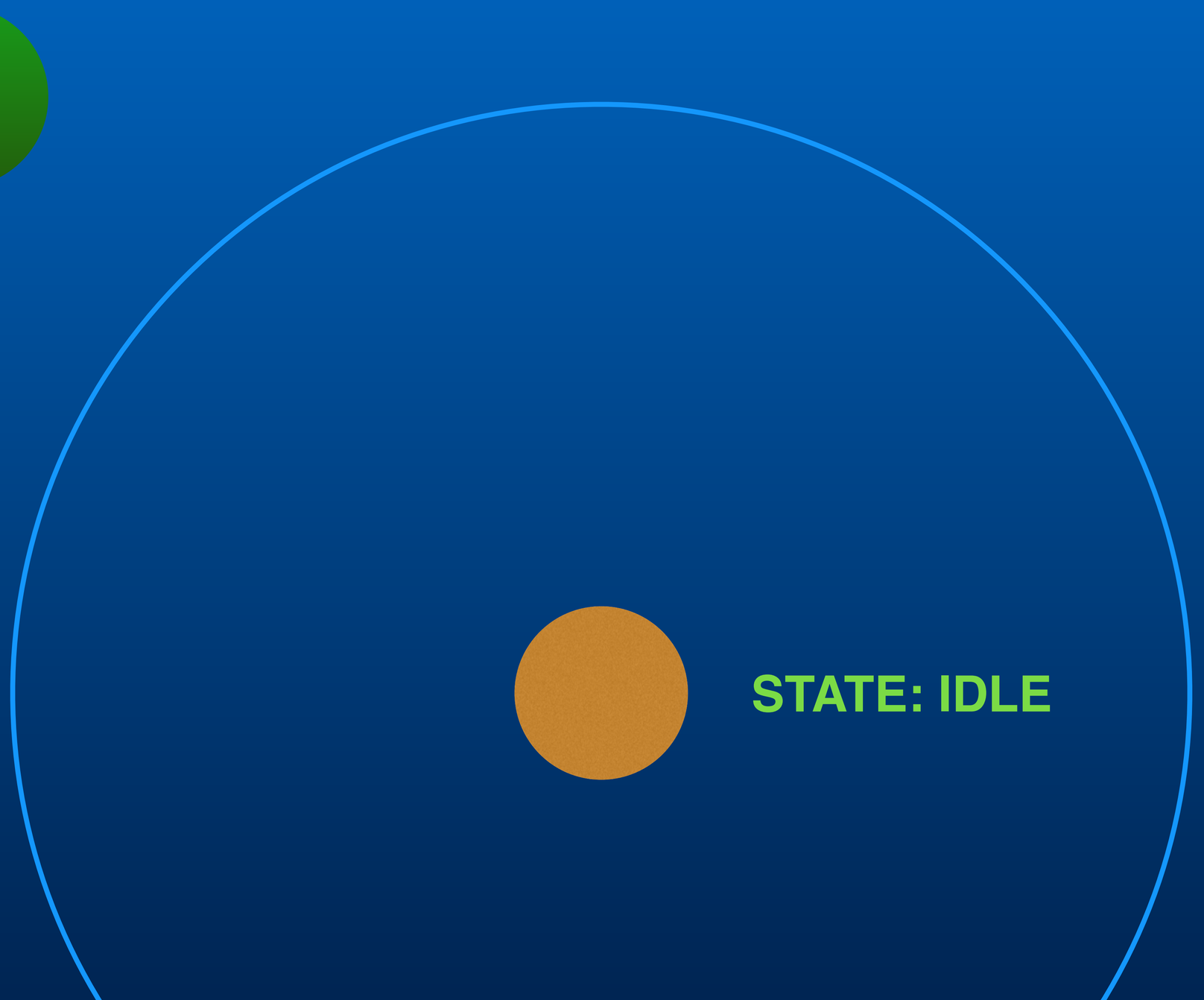


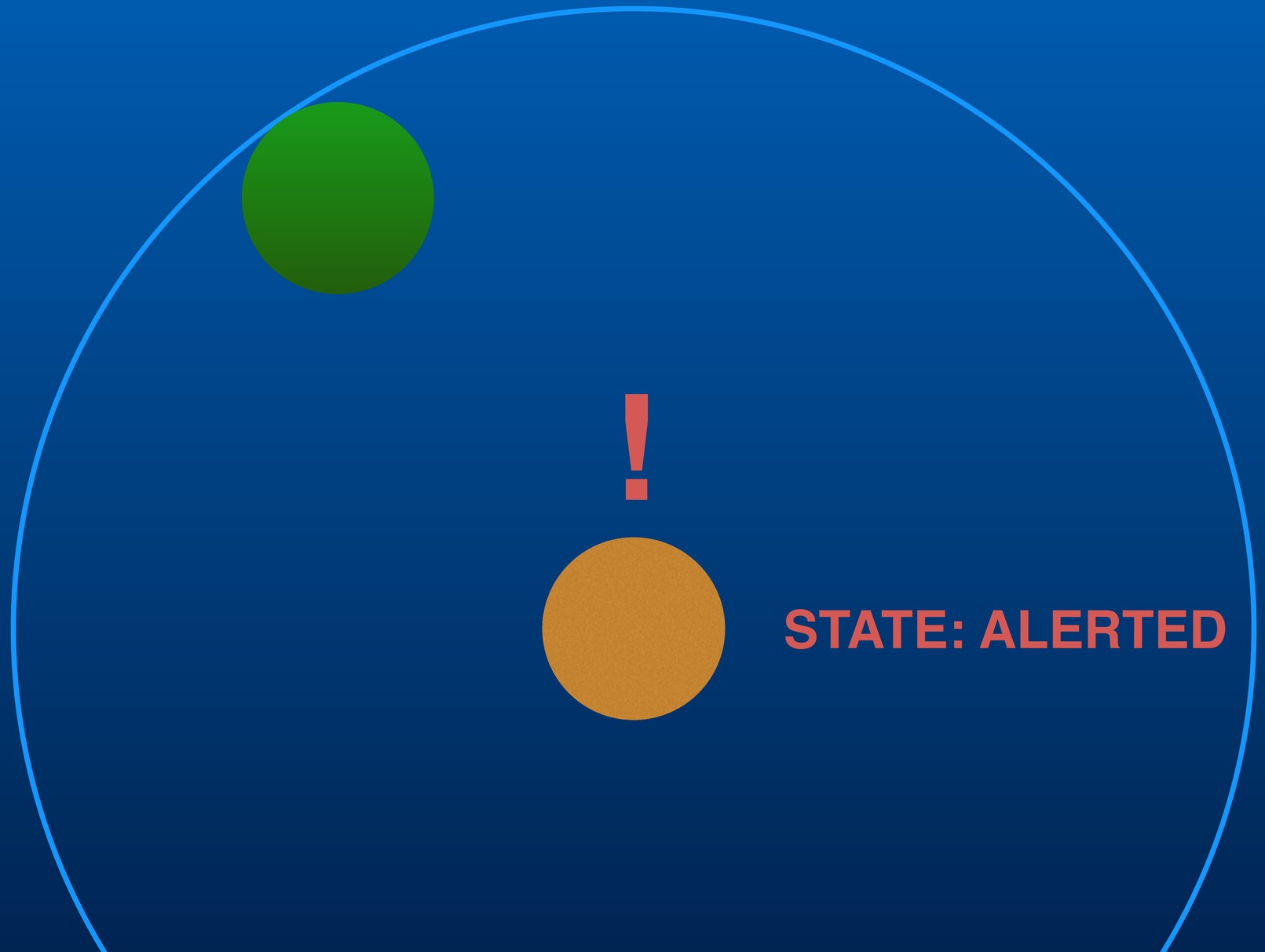
When seeing something, do a ray test from current position towars that object's position against all obstacles in the level.

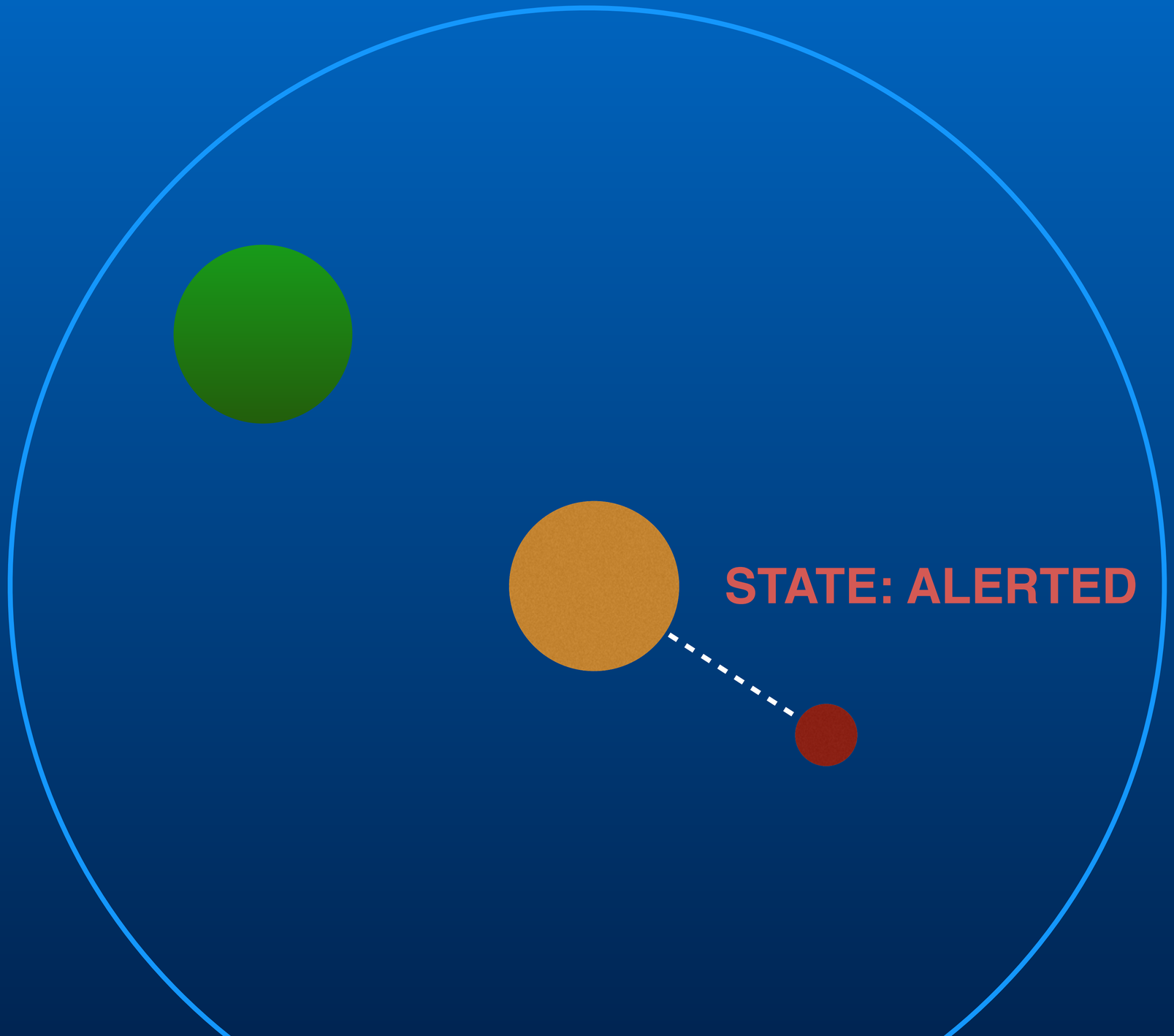
If it intersects an entity and the distance is closer than the object, then we can't see it.

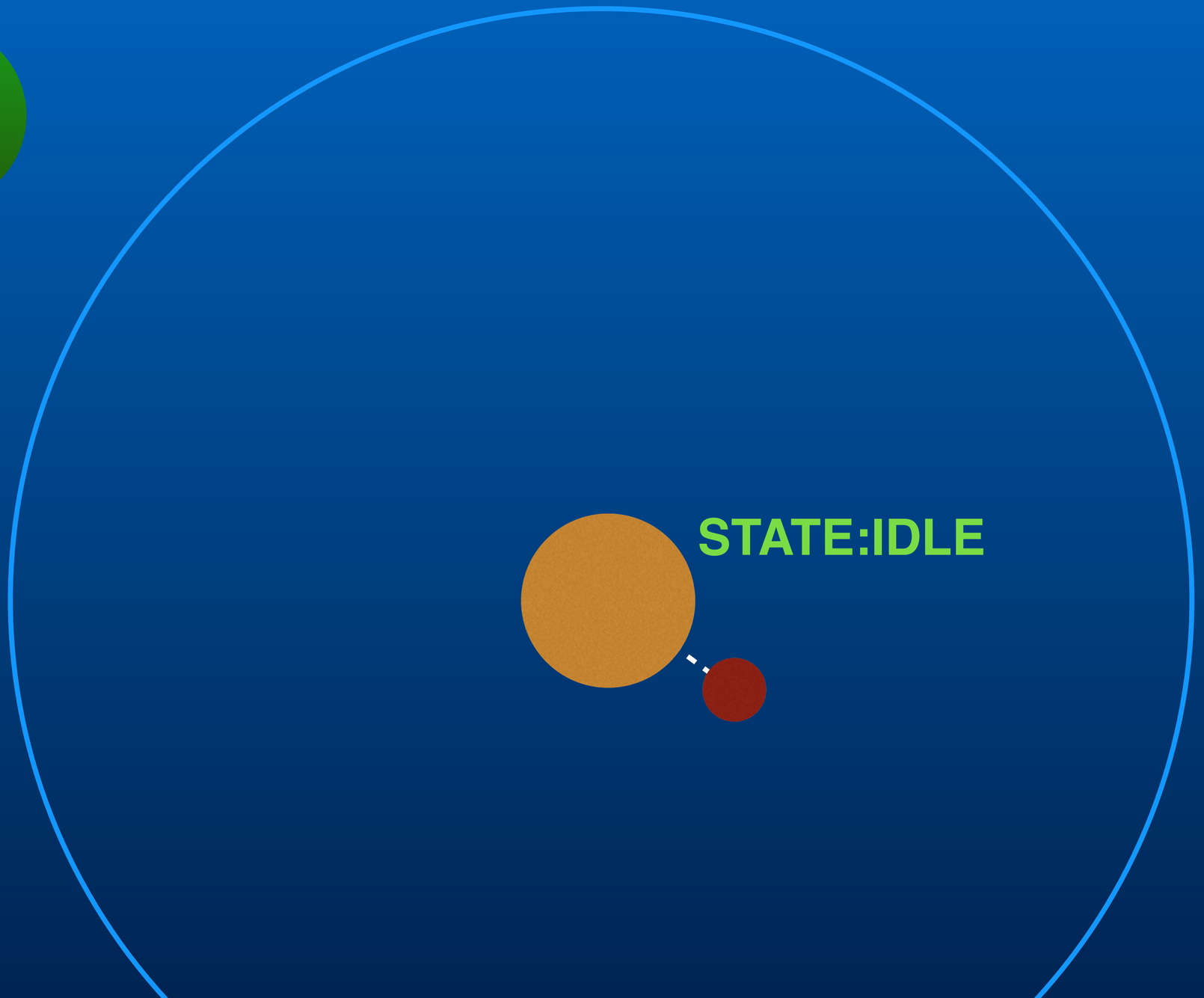


States


STATE: IDLE


!
STATE: ALERTED


STATE: ALERTED


STATE:IDLE

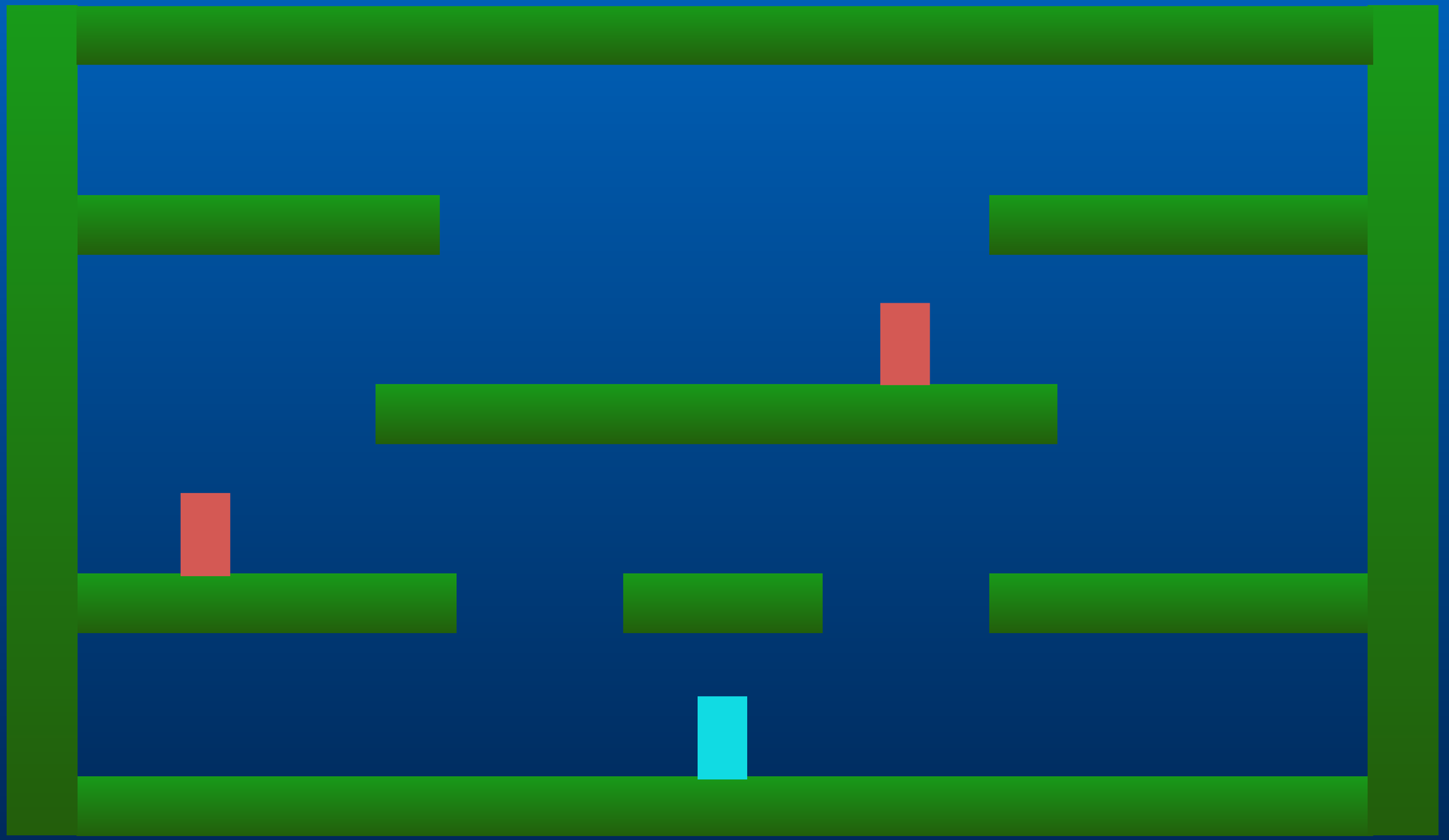
Finite state machines.

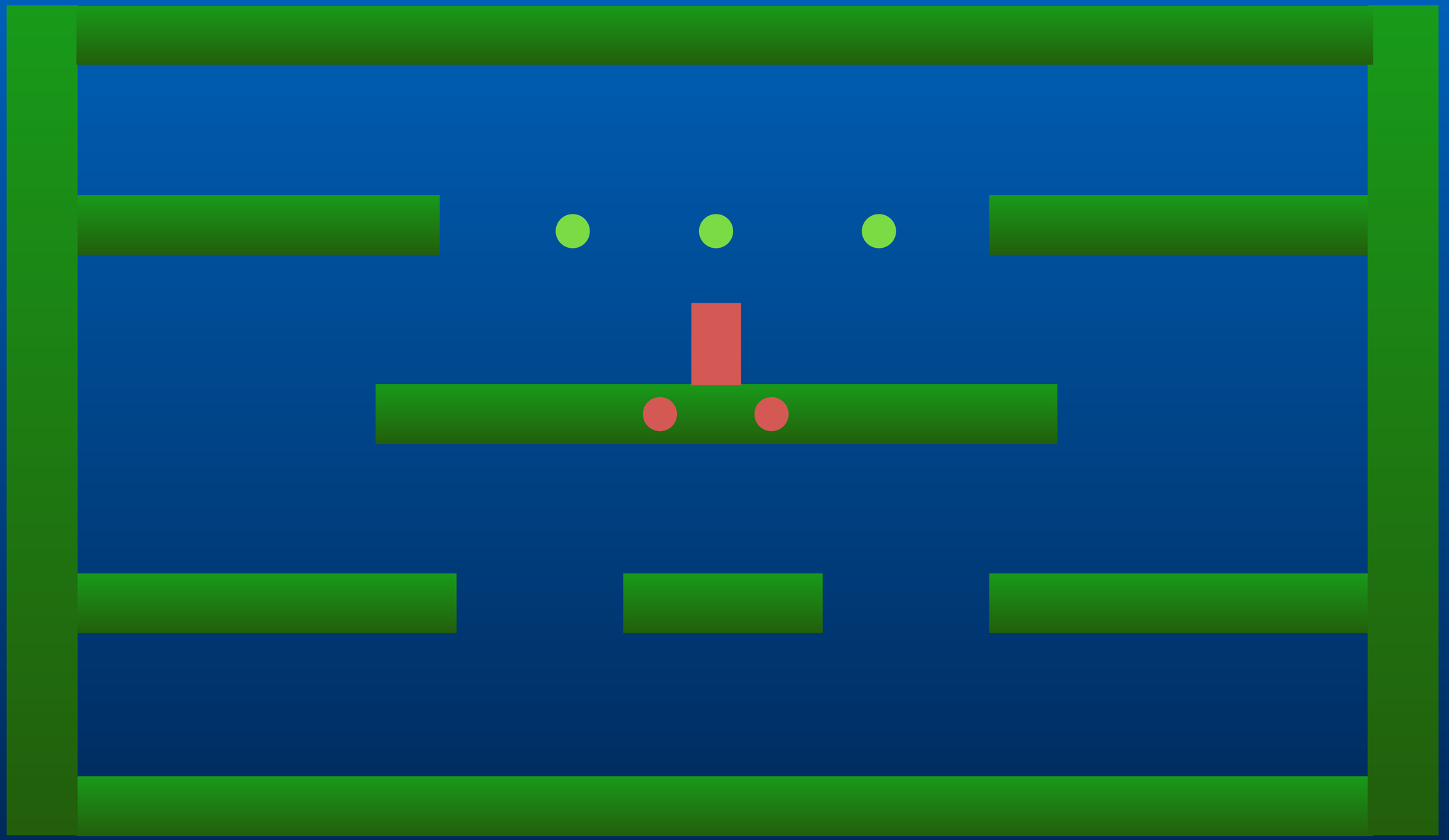
Can only be in one of some predetermined states at a time.

Update every frame based on
the current state.

React to sensing events based on
the current state.

Example:
Versus platformer.





On every frame

IDLE - Move forward, if edge sensor isn't colliding, turn around. If jump sensor is colliding, jump X% of the time.

ATTACKING - Move towards player, if edge sensor isn't colliding, stop.

FLEEING - Move away from player, if edge sensor isn't colliding, do nothing. If jump sensor is colliding, jump 100% of the time. After 3 seconds, switch state to IDLE.

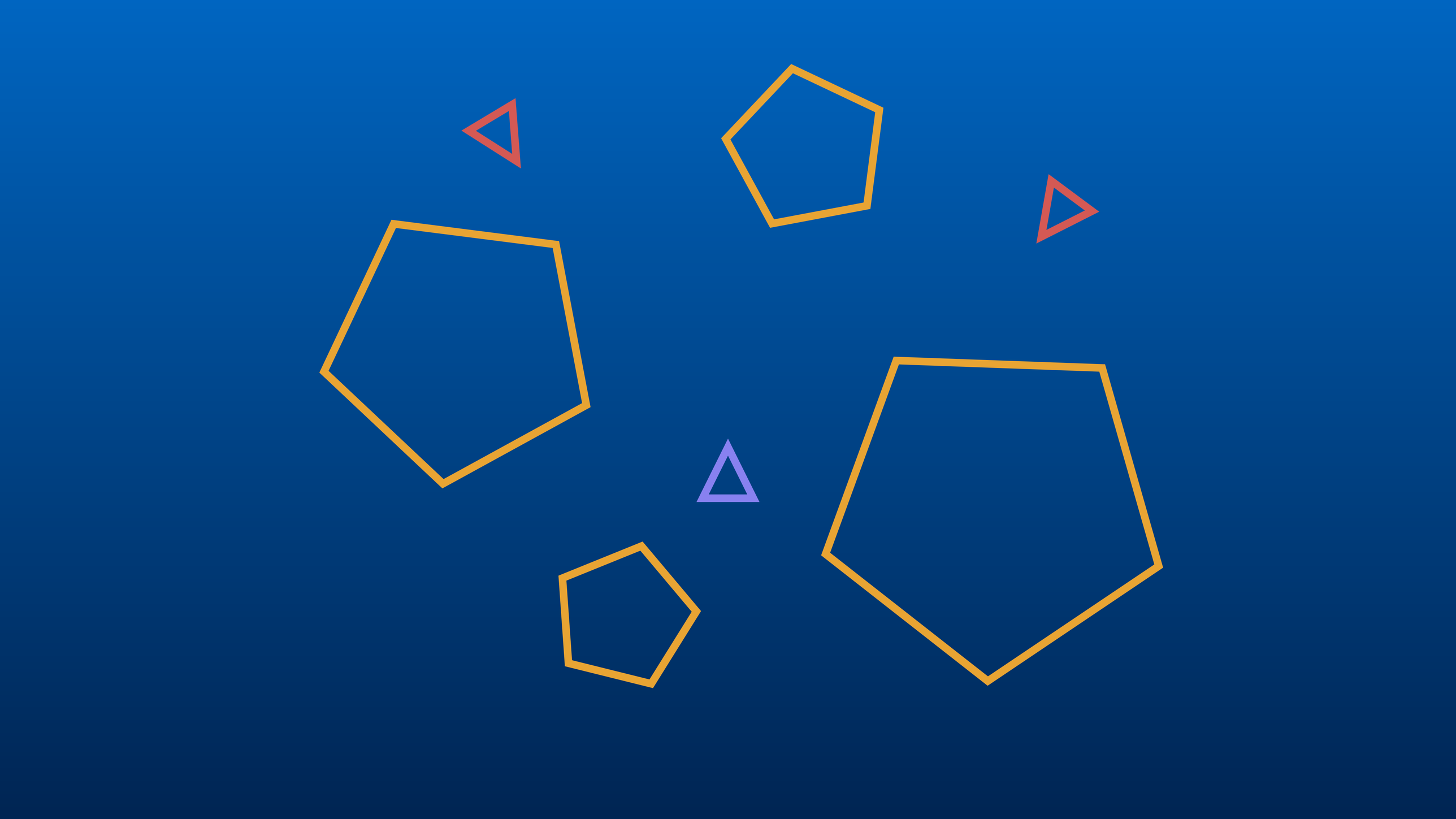
When seeing player.

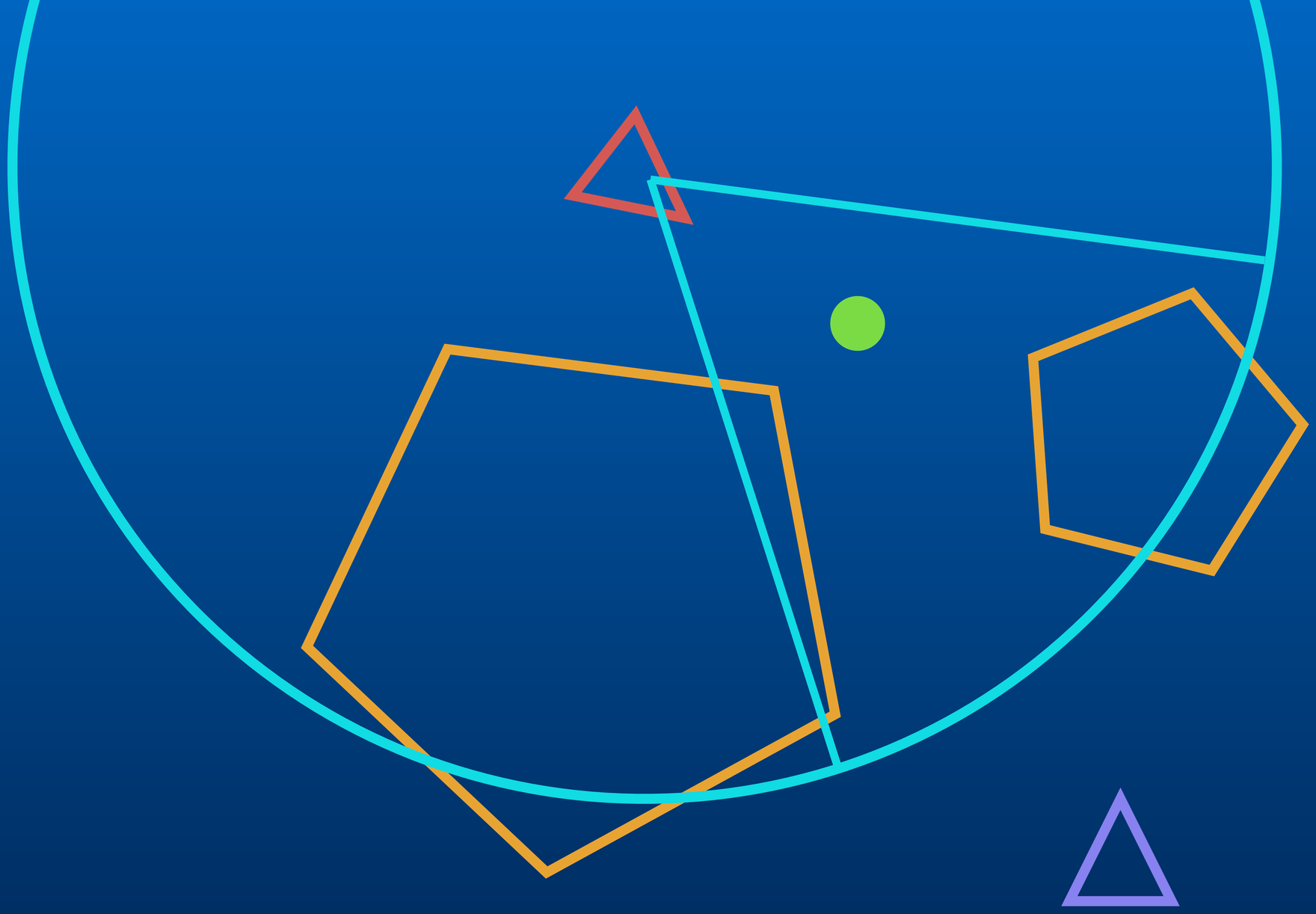
IDLE - Switch state to ATTACKING.

ATTACKING - Shoot, reverse direction, switch state to FLEEING.

FLEEING - Reverse direction.

Example:
Versus Asteroids.





On every frame

IDLE - Move forward, steer left (or right) if the asteroid sensor is triggered.

SUSPICIOUS - Spin around fast, don’t move

CHASING - Move forward, steer toward player! If haven’t seen player in X seconds, switch state to IDLE.

If player seen (can be checked on an interval).

IDLE - Switch state to SUSPICIOUS

SUSPICIOUS - Switch state to CHASING

CHASING - Shoot!