VGP332 – Artificial Intelligence

Instructor: Peter Chan

Agenda

- Assignment 3 Redux
- Autonomous Agent
- Vehicle
- Steering Behaviours
- Combining Steering Behaviours
- Smoothing
- Assignment 4 Overview

Agenda

- Assignment 3 Redux
- Autonomous Agent
- Vehicle
- Steering Behaviours
- Combining Steering Behaviours
- Smoothing
- Assignment 4 Overview

Assignment 3

- Questions?

Agenda

- Assignment 3 Redux
- Autonomous Agent
- Vehicle
- Steering Behaviours
- Combining Steering Behaviours
- Smoothing
- Assignment 4 Overview

Autonomous Agent

- System situated within an environment
- Can sense the environment
- Can react to the environment
- Pursues its own agenda

Autonomous Agent

- Movement of an autonomous agent
  - Action selection
  - Steering
  - Locomotion

Autonomous Agent

- Movement of an autonomous agent
  - Action selection
  - Steering
  - Locomotion

High-Level AI

Conditions:
My team has the ball
The opponent's goal is lightly defended
Action selected:
Make a strike for the goal

Autonomous Agent

- Movement of an autonomous agent
  - Action selection
  - Steering
  - Locomotion

Agent-Level AI

Steering behaviours:
Keep ball within agent control
Head towards opponent goal
Avoid opponent players

Autonomous Agent

- Movement of an autonomous agent
  - Action selection
  - Steering
  - Locomotion

Agent Animation

Locomotion:
Kick ball 1 foot ahead
Run to ball
Dodge 0.5 feet ahead and to left
Run to ball
Fake running left
Kick ball 1 foot ahead
…

Agenda

- Assignment 3 Redux
- Autonomous Agent
- Vehicle
- Steering Behaviours
- Combining Steering Behaviours
- Smoothing
- Assignment 4 Overview

MovingEntity

- MovingEntity derived from BaseGameEntity
- Describes a basic vehicle with a point mass
- Attributes defining a local coordinate system:
  - Heading
  - Side
- Attributes defining movement:
  - Velocity
  - Mass
  - MaxSpeed
  - MaxForce
  - MaxTurnRate

Look at code!

Vehicle

- Vehicle derived from MovingEntity
- Attributes:
  - GameWorld
  - SteeringBehaviors
- Method:
  - Update

Look at code!

Vehicle Update

- Uses Newtonian physics

```
bool Vehicle::Update( double time_elapsed )
{
    SVector2D SteeringForce = m_pSteering->Calculate();
    SVector2D acceleration = SteeringForce / m_dMass;
    m_vVelocity += acceleration * time_elapsed;
    m_vVelocity.Truncate( m_dMaxSpeed );

    m_vPos += m_vVelocity * time_elapsed;

    if ( m_vVelocity.LengthSq() > 0.00000001 )
    {
        m_vHeading = Vec2DNormalize( m_vVelocity );
        m_vSide = m_vHeading.Perp();
    }

    // Wraparound omitted
}
```

Agenda

- Assignment 3 Redux
- Autonomous Agent
- Vehicle
- Steering Behaviours
- Combining Steering Behaviours
- Smoothing
- Assignment 4 Overview

Steering Behaviours

- What types of steering behaviours do you need in a game?

Steering Behaviours

- Seek
- Flee
- Arrive
- Pursuit
- Evade
- Wander
- Interpose
- Hide
- Path following
- Offset pursuit
- Obstacle avoidance
- Wall avoidance

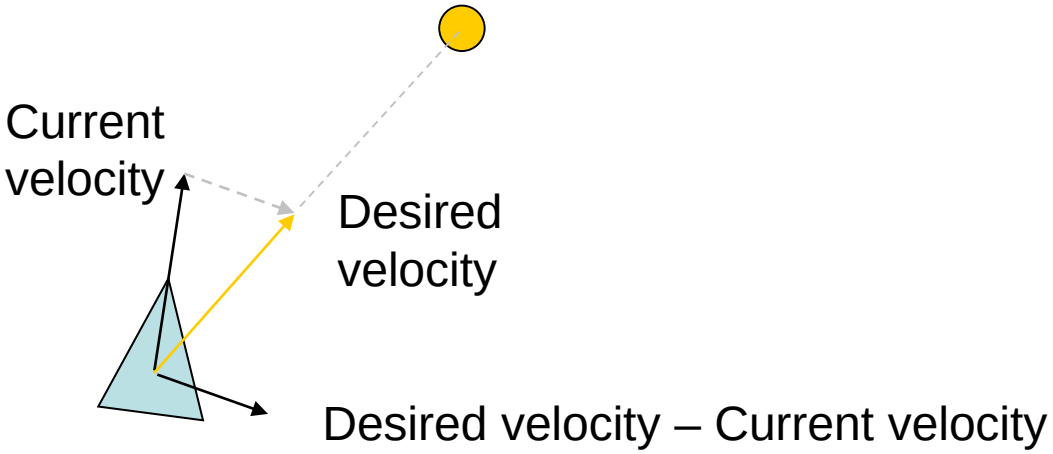
Steering Behaviours

- Let’s try out some examples:

http://red3d.com/cwr/steer/

Seek

- Force needed to direct agent toward target



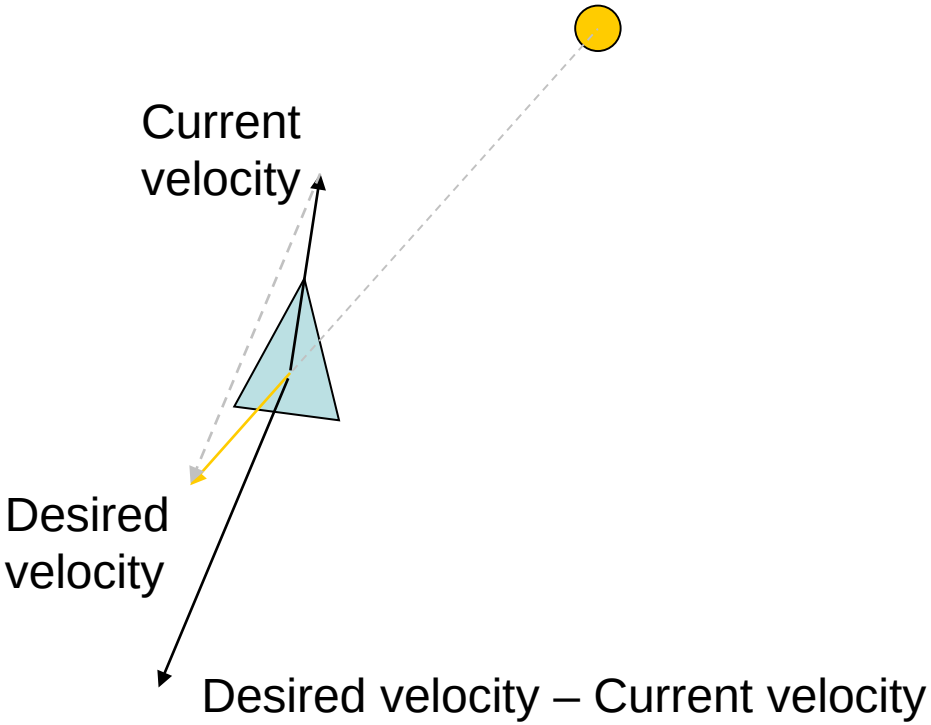
Seek

- Code:

```
Vector2D SteeringBehaviors::Seek( Vector2D TargetPos )
{
    Vector2D DesiredVelocity =
        Vec2DNormalize( TargetPos – m_pVehicle->Pos() ) *
        m_pVehicle->MaxSpeed();
    return ( DesiredVelocity – m_pVehicle->Velocity();
}
```

Flee

- Opposite of **seek**



Flee

- Code:

```
Vector2D SteeringBehaviors::Flee( Vector2D TargetPos )
{
    Vector2D DesiredVelocity =
        Vec2DNormalize( m_pVehicle->Pos() - TargetPos ) *
        m_pVehicle->MaxSpeed();
    return ( DesiredVelocity – m_pVehicle->Velocity();
}
```

Flee

- Add a “flee range”:

```
Vector2D SteeringBehaviors::Flee( Vector2D TargetPos )
{
    const double PanicDistanceSq = 100.0 * 100.0;
    if ( Vec2DDistanceSq( m_pVehicle->Pos(), target ) >
         PanicDistanceSq )
    {
       return Vector2D( 0, 0 );
    }

    Vector2D DesiredVelocity =
        Vec2DNormalize( m_pVehicle->Pos() - TargetPos ) *
        m_pVehicle->MaxSpeed();
    return ( DesiredVelocity – m_pVehicle->Velocity();
}
```

Flee

- Add a “flee range”:

```
Vector2D SteeringBehaviors::Flee( Vector2D TargetPos )
{
    const double PanicDistanceSq = 100.0 * 100.0;
    if ( Vec2DDistanceSq( m_pVehicle->Pos(), target ) >
         PanicDistanceSq )
    {
       return Vector2D( 0, 0 );
    }

    Vector2D DesiredVelocity =
        Vec2DNormalize( m_pVehicle->Pos() - TargetPos ) *
        m_pVehicle->MaxSpeed();
    return ( DesiredVelocity – m_pVehicle->Velocity();
}
```

Distance calculated in squared units to eliminate square root calculation

Arrive

- Need to slow down if agent should halt at target

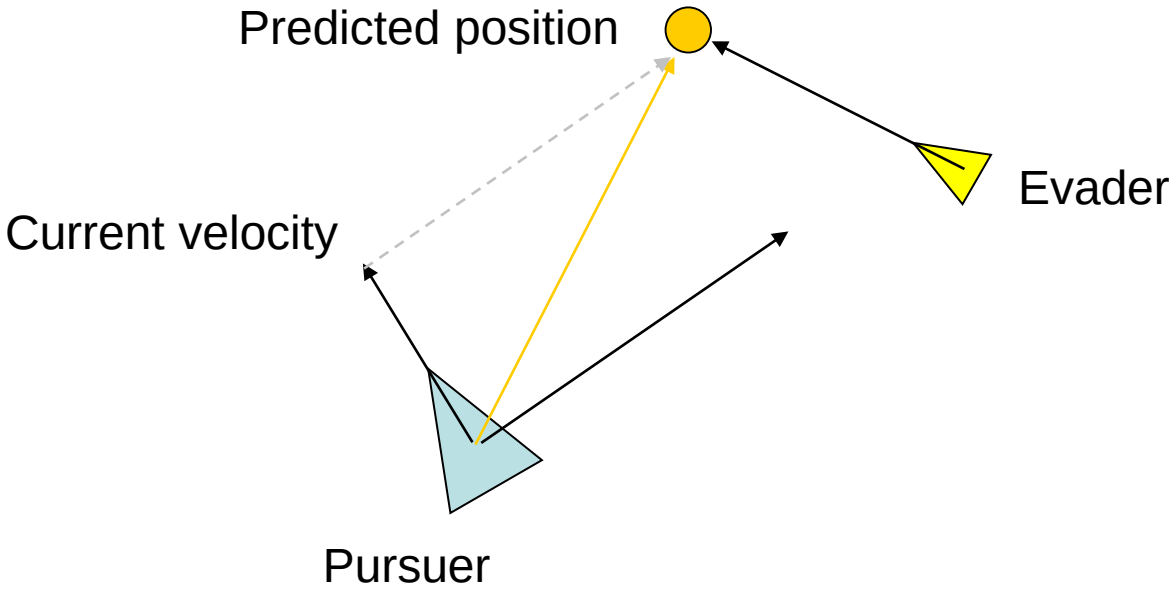
```
enum Deceleration { slow=3, normal=2, fast=1 };

Vector2D SteeringBehaviors::Arrive( Vector2D TargetPos,
                                    Deceleration decel )
{
    Vector2D ToTarget = TargetPos – m_pVehicle->Pos();
    double dist = ToTarget.Length();
    if ( dist > 0 )
    {
        const double DecelTweaker = 0.3;
        double speed = dist / ((double)decel * DecelTweaker);

        speed = min( speed, m_pVehicle->MaxSpeed() );
        Vector2D DesiredVelocity = ToTarget * speed / dist;

        return ( DesiredVelocity – m_pVehicle->Velocity() );
    }
    return Vector2D( 0,0 );
}
```

Pursuit

- **Seek**, **arrive** work for stationary targets
- What about if target is moving?

Pursuit

- **Seek**, **arrive** work for stationary targets
- If target is moving, agent needs to "**seek** to predicted position"



Pursuit

- Success of pursuit depends on accuracy of target prediction
- Q: How far to look ahead?

- Optimization: If target is facing pursuer *and* heading towards pursuer, then seek to target's current position

Pursuit

Look at code!

Evade

- Opposite of **pursuit**
- **Flee** from predicted future position of pursuer

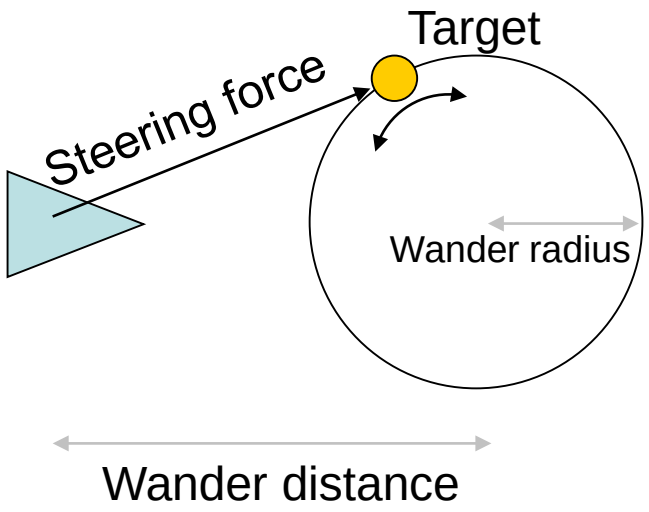
Look at code!

Wander

- How can you create random movements?

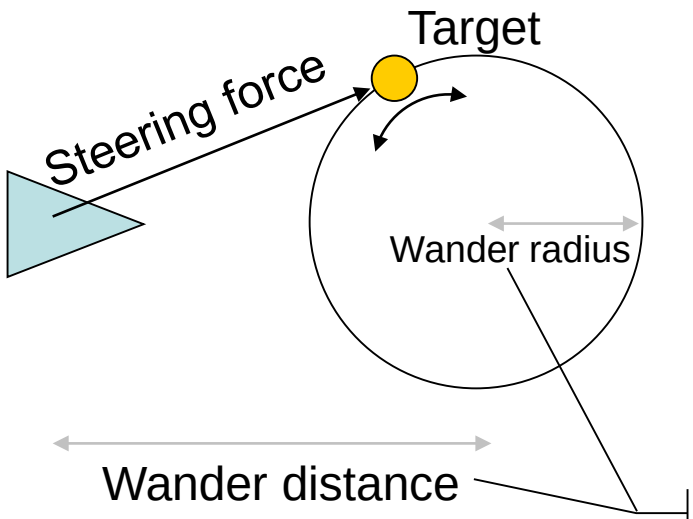
Wander

- Project circle in front of agent
- Constrain target to move randomly on circle
- Steer agent towards target



Wander

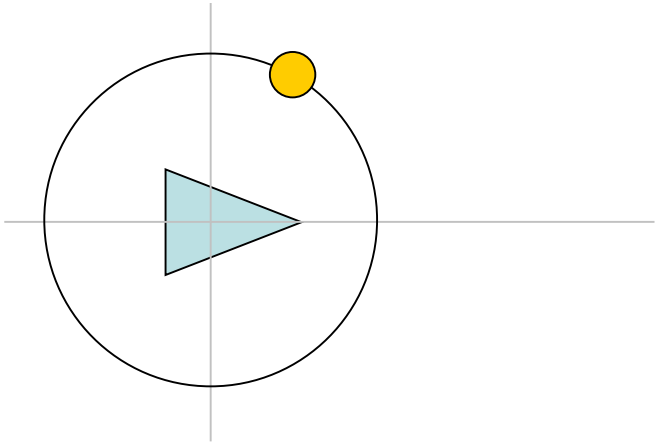
- Project circle in front of agent
- Constrain target to move randomly on circle
- Steer agent towards target



Altering wander distance and radius creates wide range of jitter-free random movement

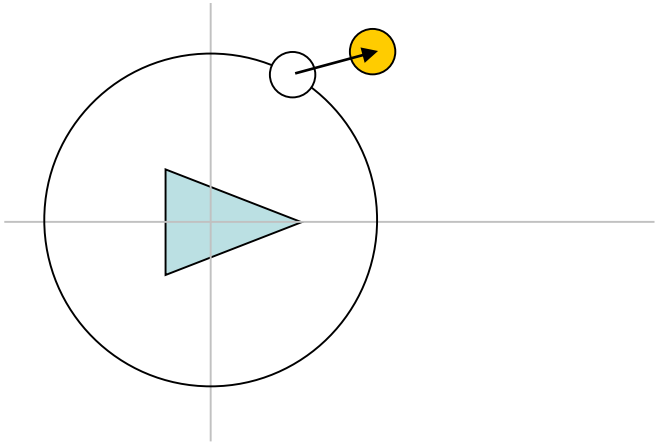
Wander

- Initialize: Start with target on wander circle



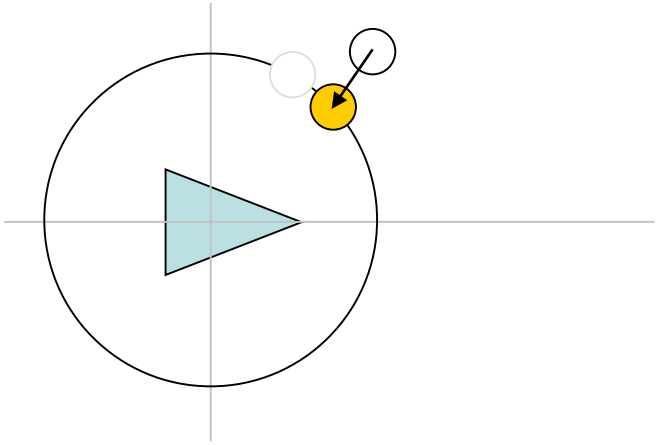
Wander

- Step 1: Add small random displacement to target



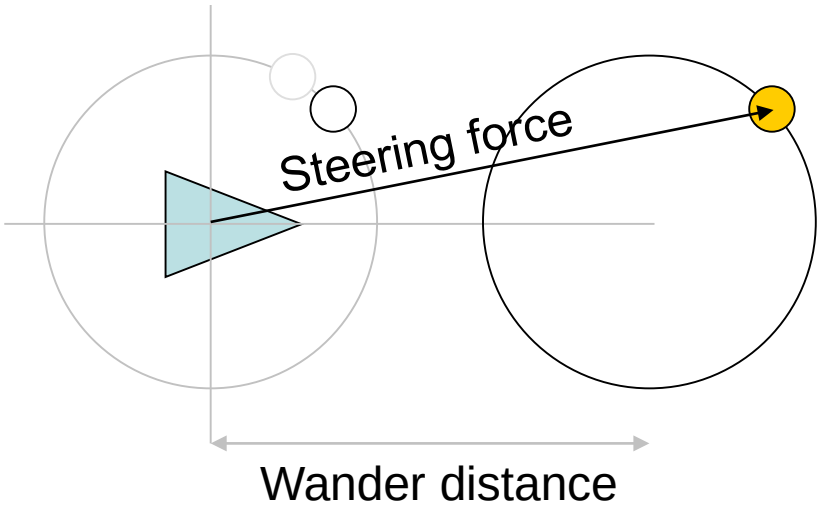
Wander

- Step 2: Project target back onto circle



Wander

- Step 3: Project circle in front of agent



Wander

Look at code!

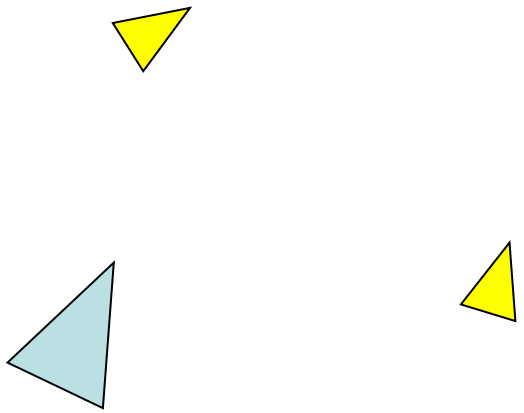
Steering Behaviours

- Seek
- Flee
- Arrive
- Pursuit
- Evade
- Wander

- Interpose
- Hide
- Path following
- Offset pursuit
- Obstacle avoidance
- Wall avoidance

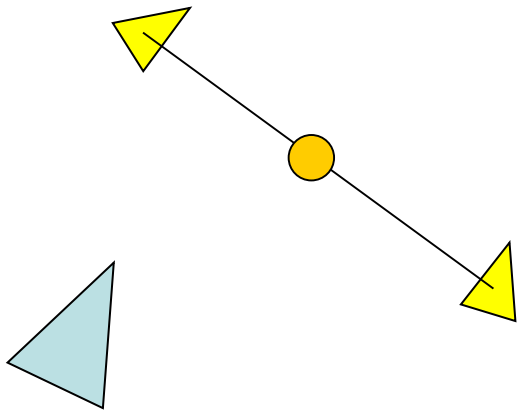
Interpose

- Move agent to midpoint of line connecting two other agents



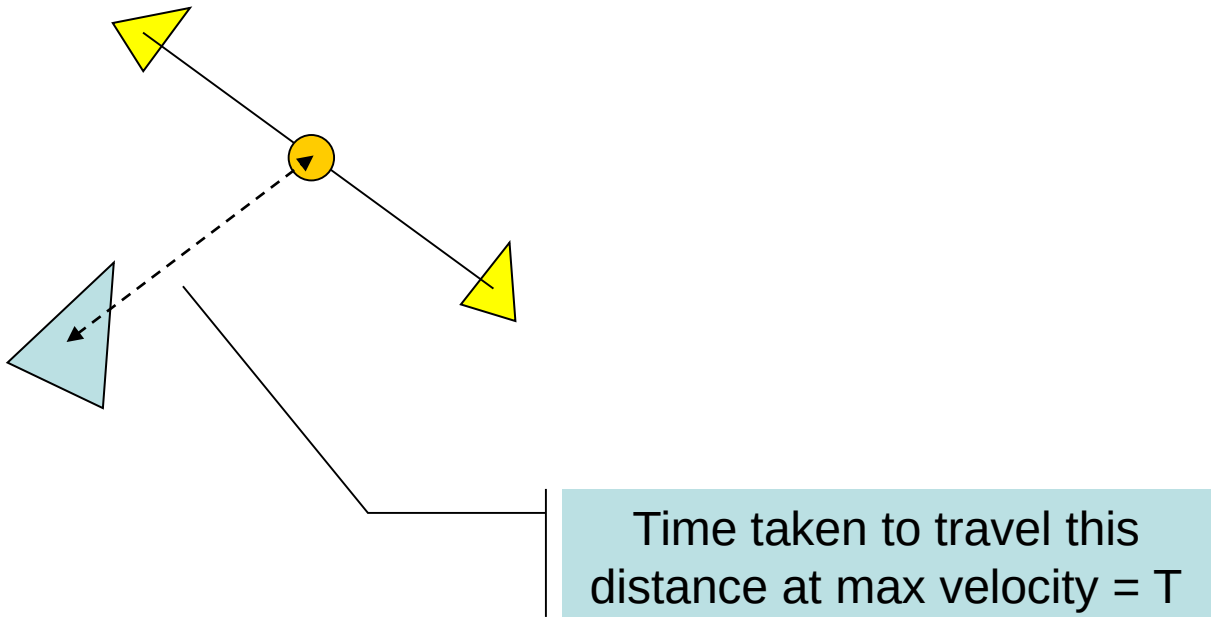
Interpose

- Step 1: Determine midpoint at current frame



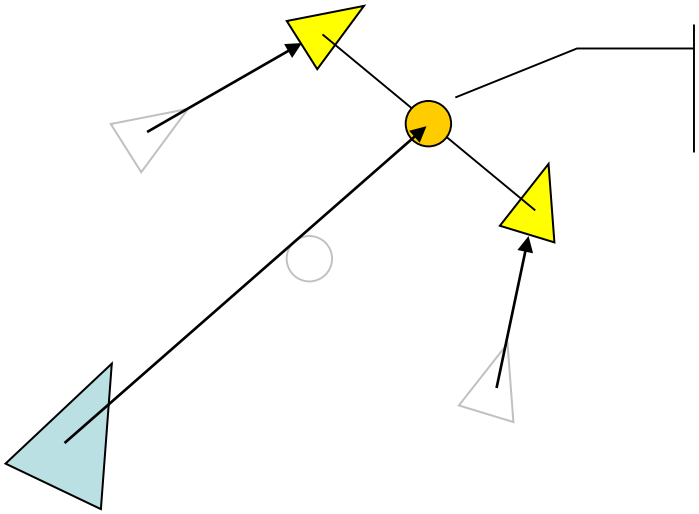
Interpose

- Step 2: If agent traveled at max velocity to midpoint, how long would it take?



Interpose

- Step 3: Extrapolate targets' positions at time T and compute new midpoint



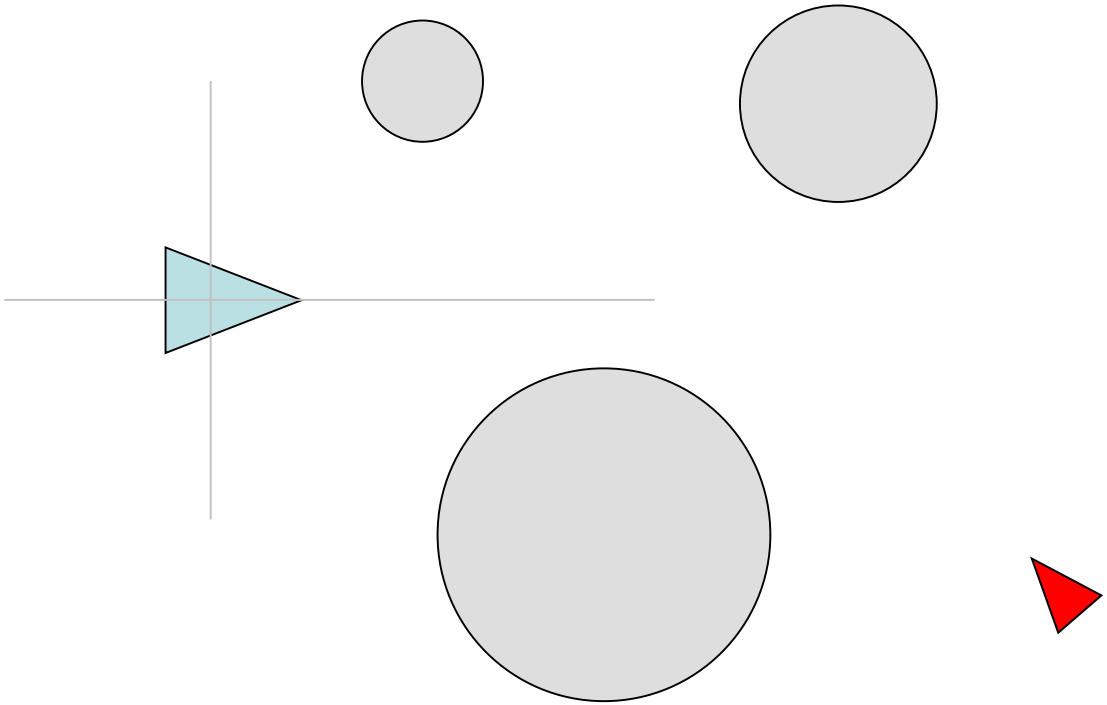
Arrive target is predicted midpoint

Interpose

Look at code!

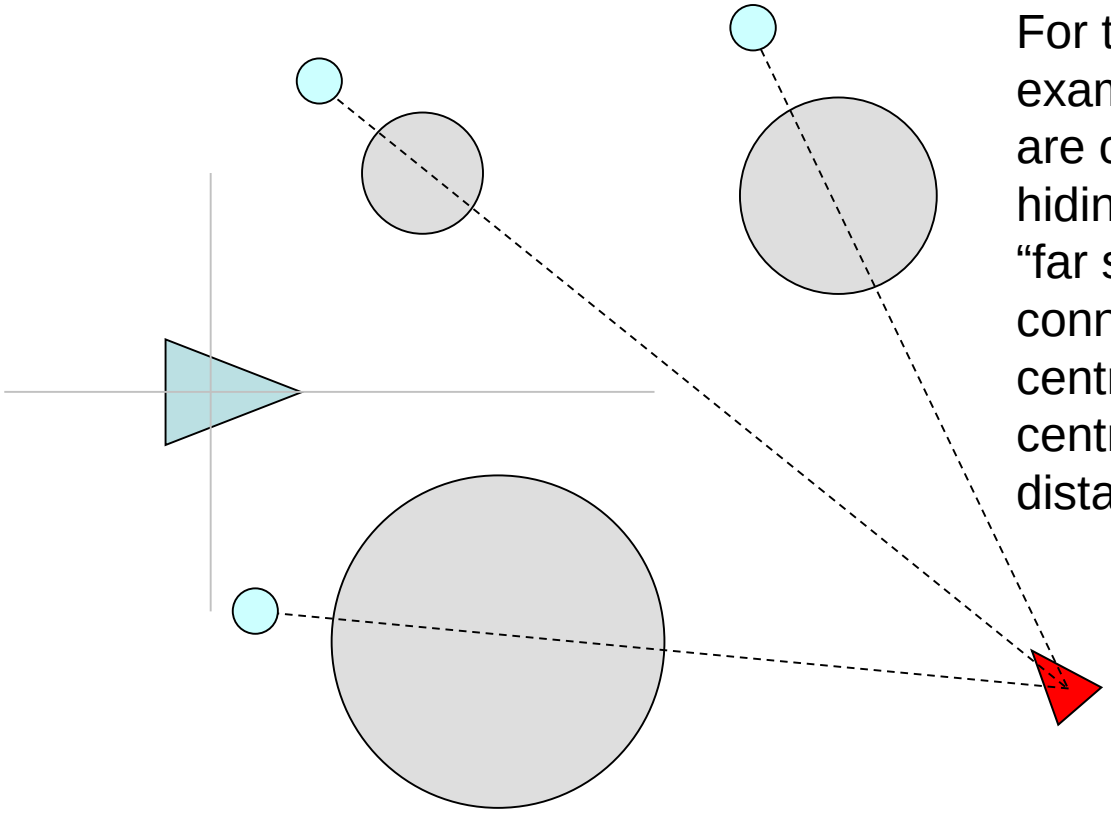
Hide

- Try to put obstacle between agent and hunter



Hide

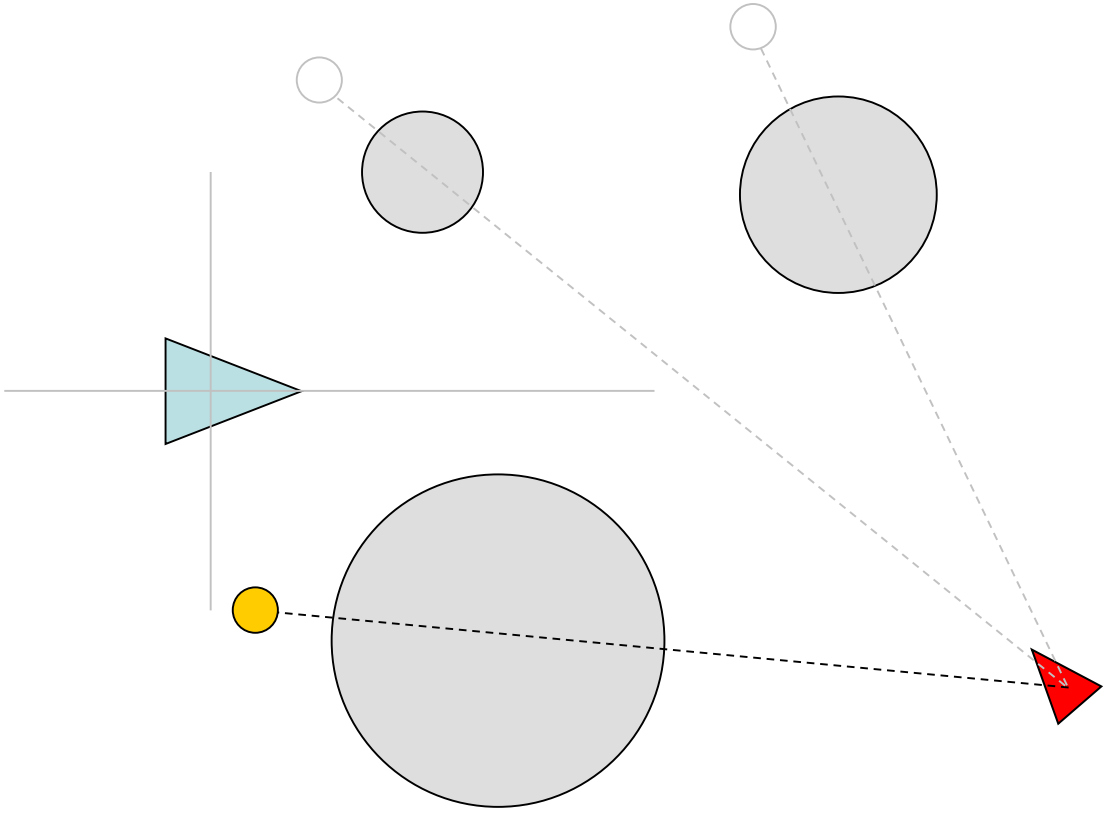
- Step 1: For each obstacle in world, determine hiding position



For the simplified 2D example, all obstacles are circles. Then, the hiding position is on the “far side” of the line connecting the obstacle centre to the hunter centre, a constant distance away.

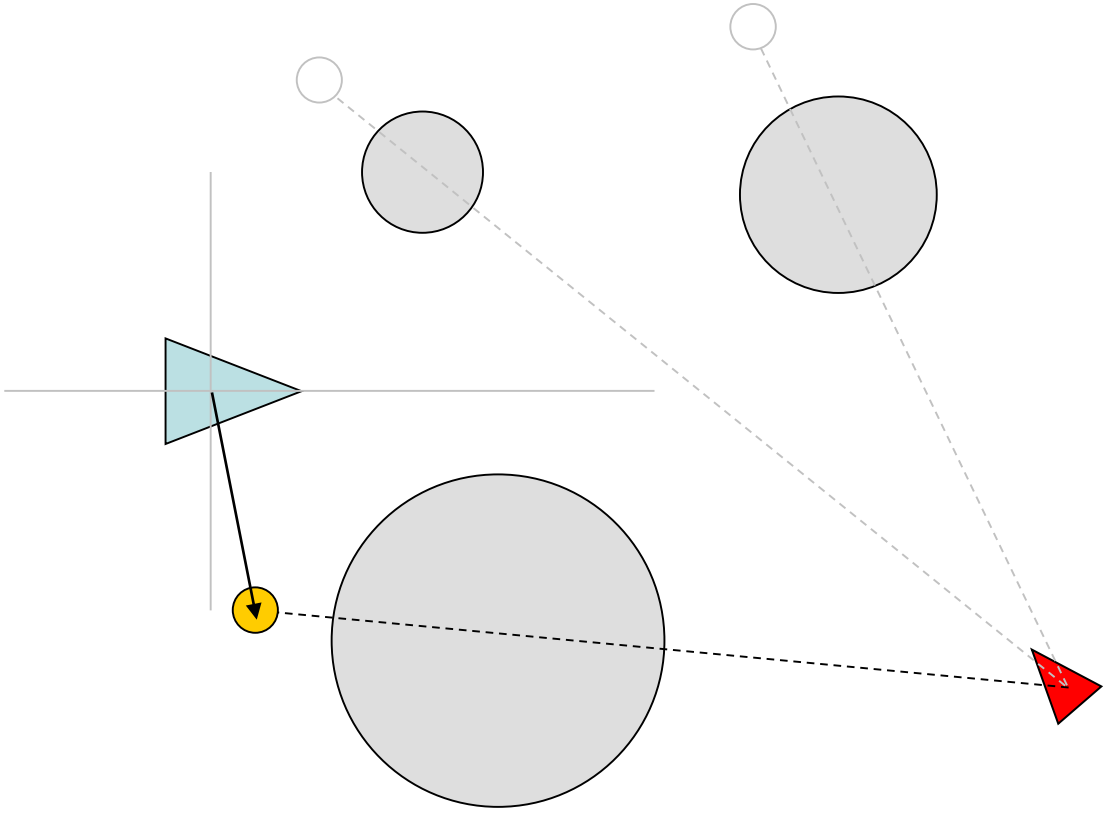
Hide

- Step 2: Keep track of closest hiding position



Hide

- Step 3: **Arrive** at closest hiding position



Hide

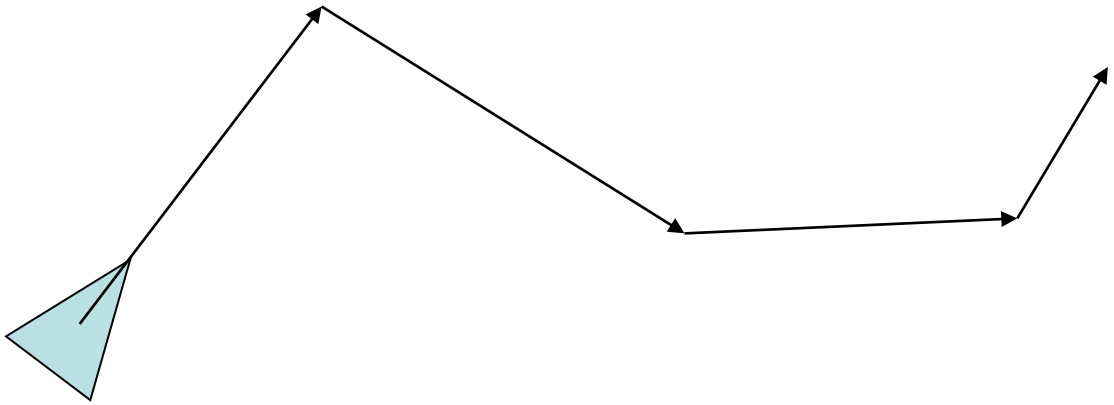
Look at code!

Hide

- Improvements?

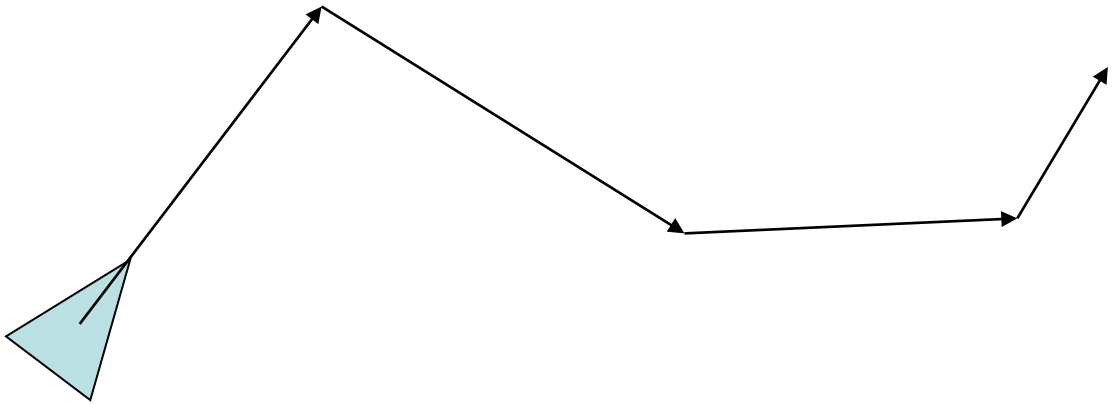
Path Following

- Get an agent to follow a path
- Discuss: when is this useful?



Path Following

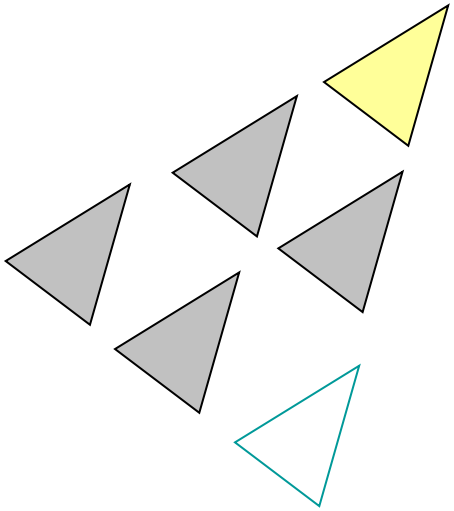
- Implementation is very game-specific



Look at code!

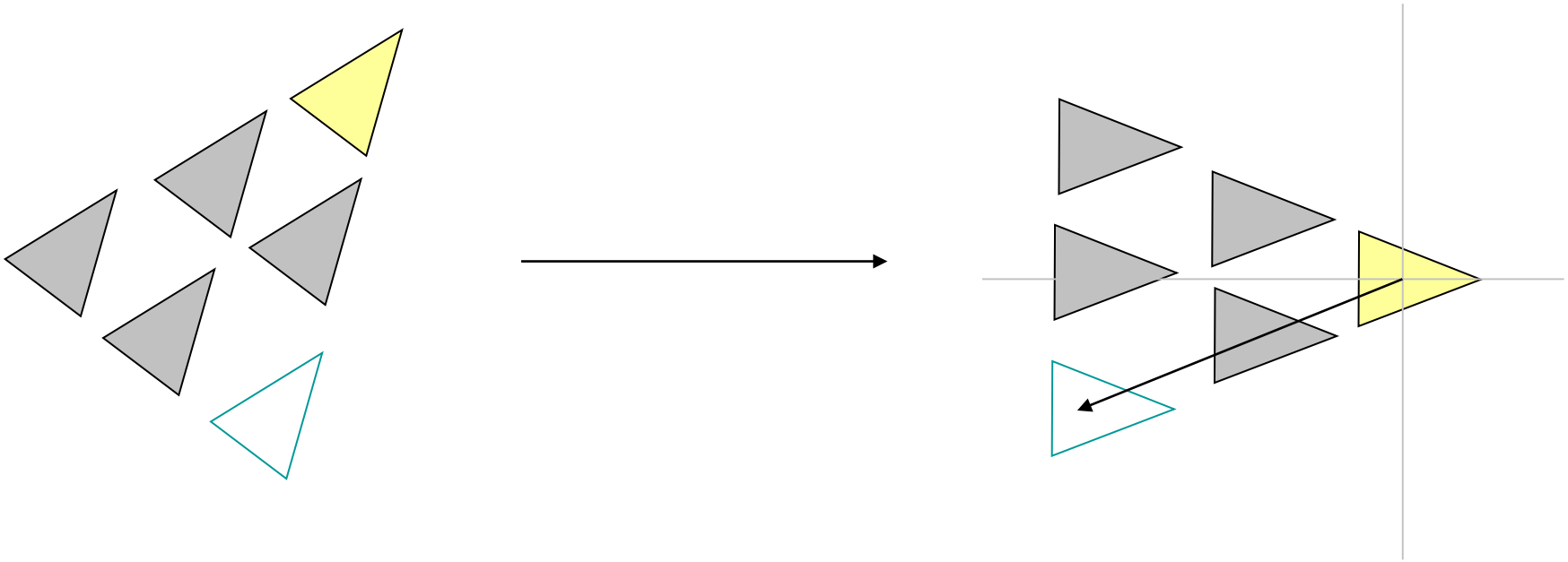
Offset Pursuit

- Useful for formation-follow-the-leader



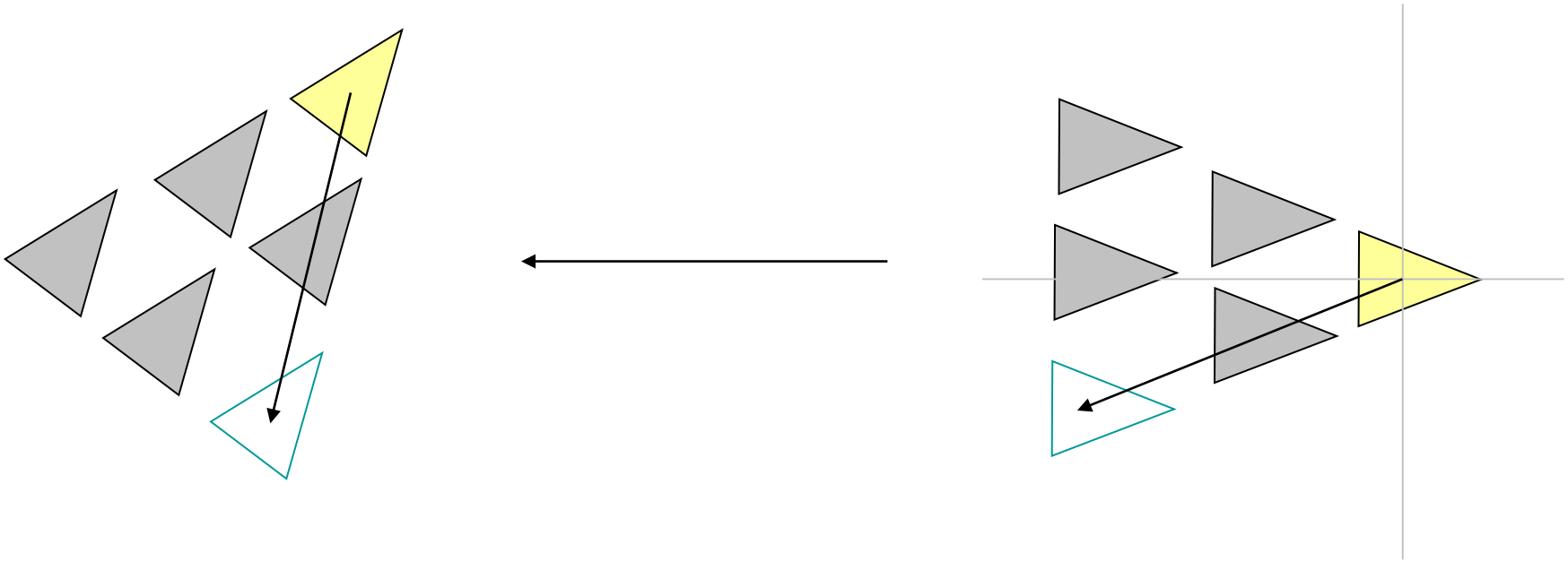
Offset Pursuit

- Initialize: Specify agent offset in “leader space”



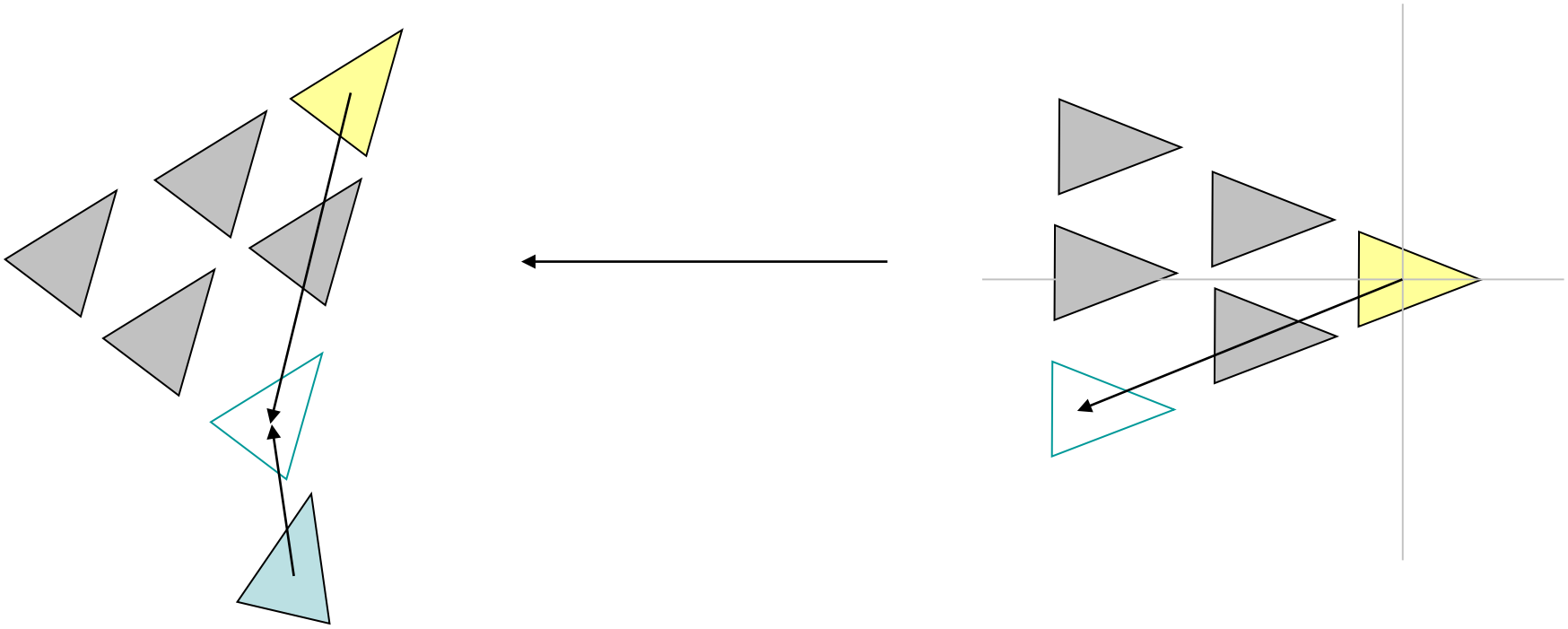
Offset Pursuit

- Step 1: Compute offset in world space



Offset Pursuit

- Step 2: **Arrive** at desired offset



Offset Pursuit

Look at code!

Steering Behaviours

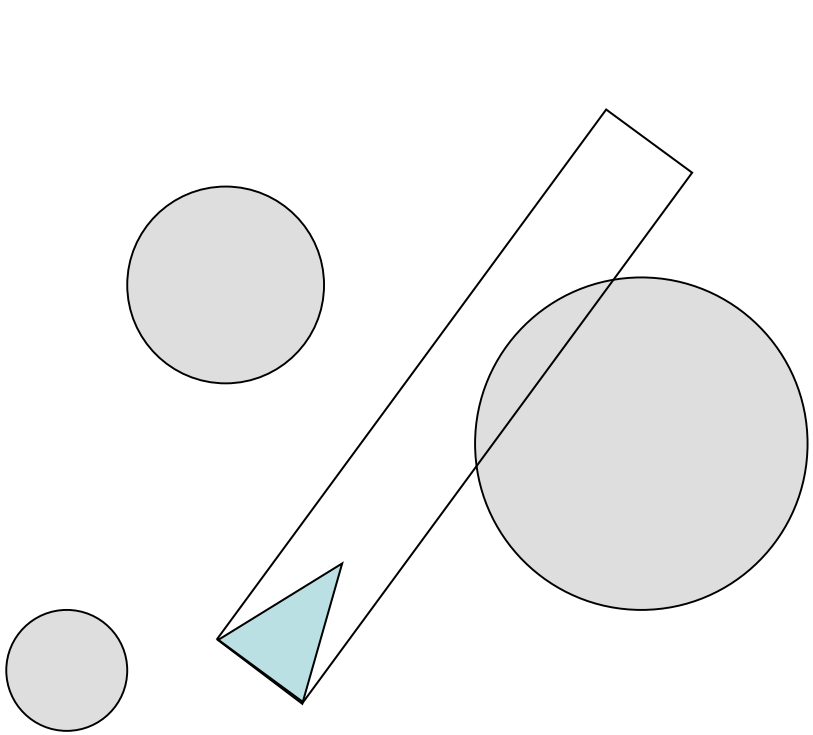
- Seek
- Flee
- Arrive
- Pursuit
- Evade
- Wander
- Interpose
- Hide
- Path following
- Offset pursuit
- Obstacle avoidance
- Wall avoidance

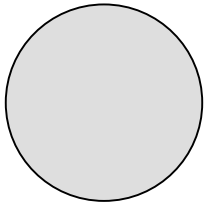
Obstacle Avoidance

- Steer agent so that it avoids obstacles in its path
- Keep space in front of agent free of collisions
- Create a bounding volume around agent
- Length of bounding volume in front of agent proportional to agent speed

Obstacle Avoidance

- Bounding space = rectangle

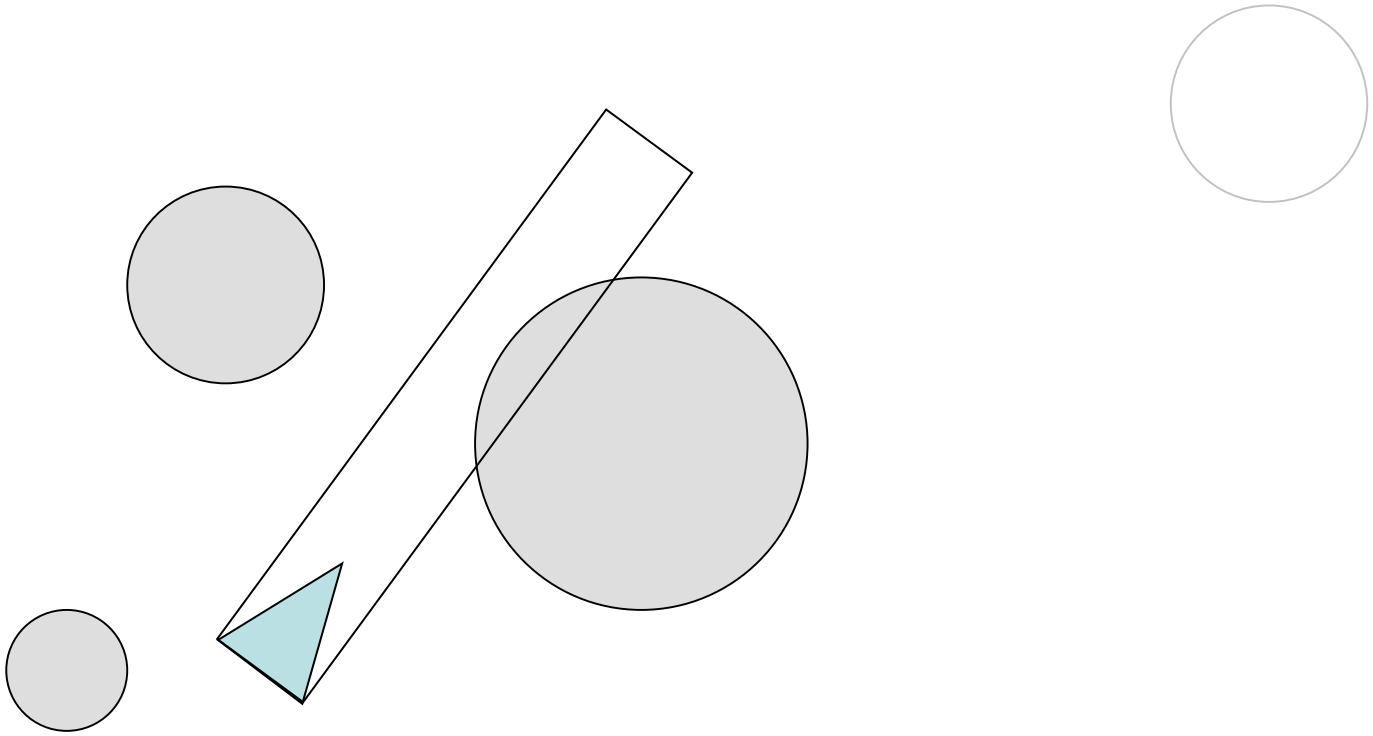




The simplified example uses 2D circles for obstacles

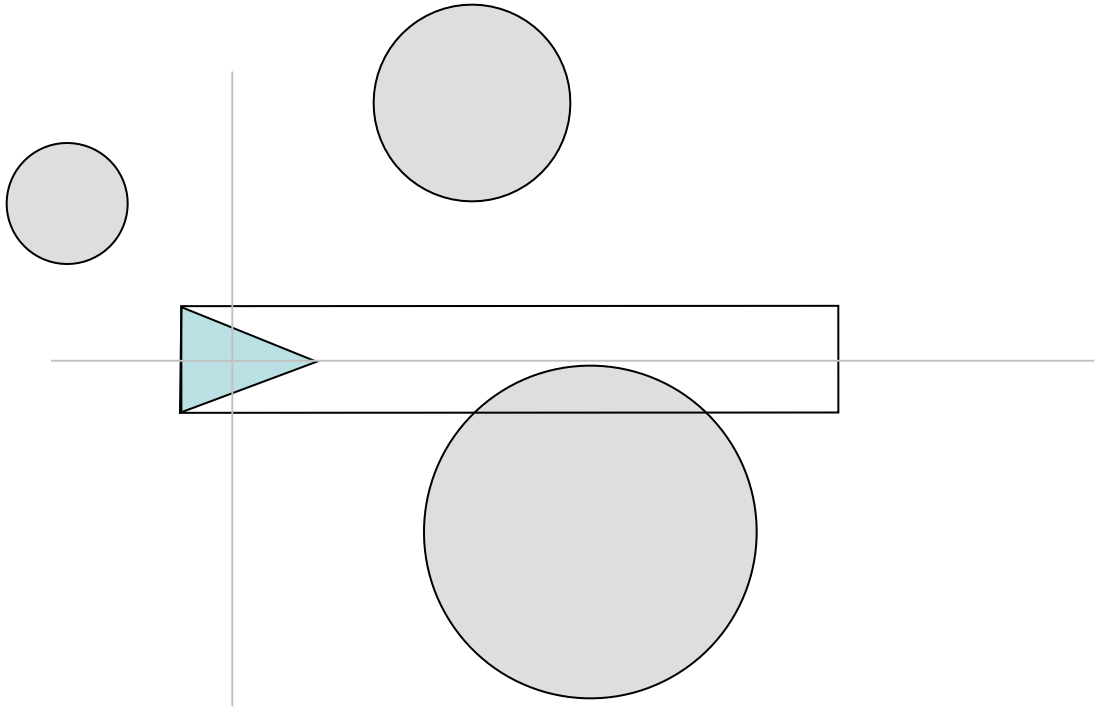
Obstacle Avoidance

- Step 1: Eliminate obstacles that are too far



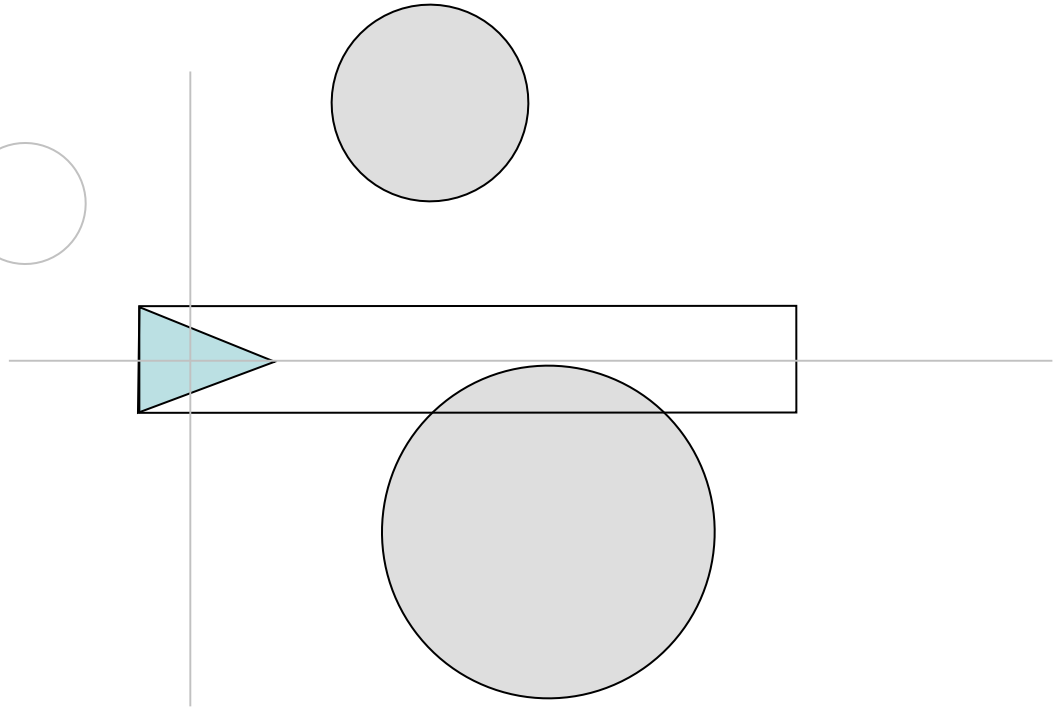
Obstacle Avoidance

- Step 2: Convert into agent space



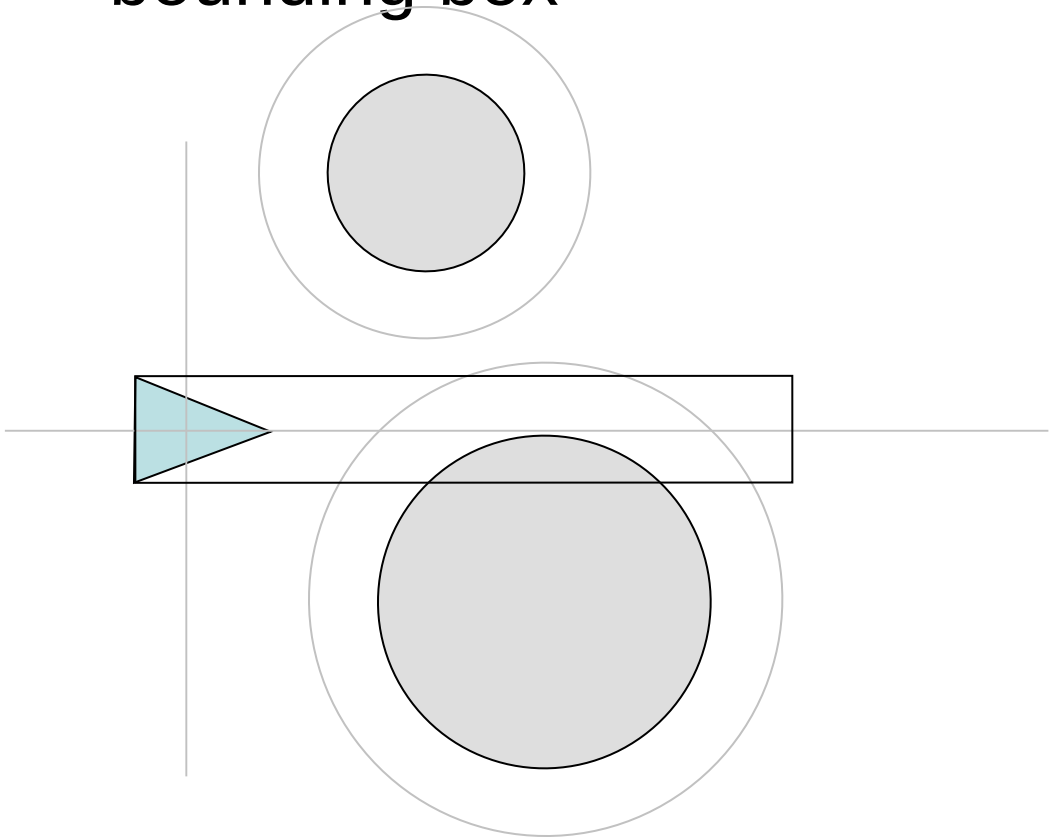
Obstacle Avoidance

- Step 3: Eliminate obstacles behind agent



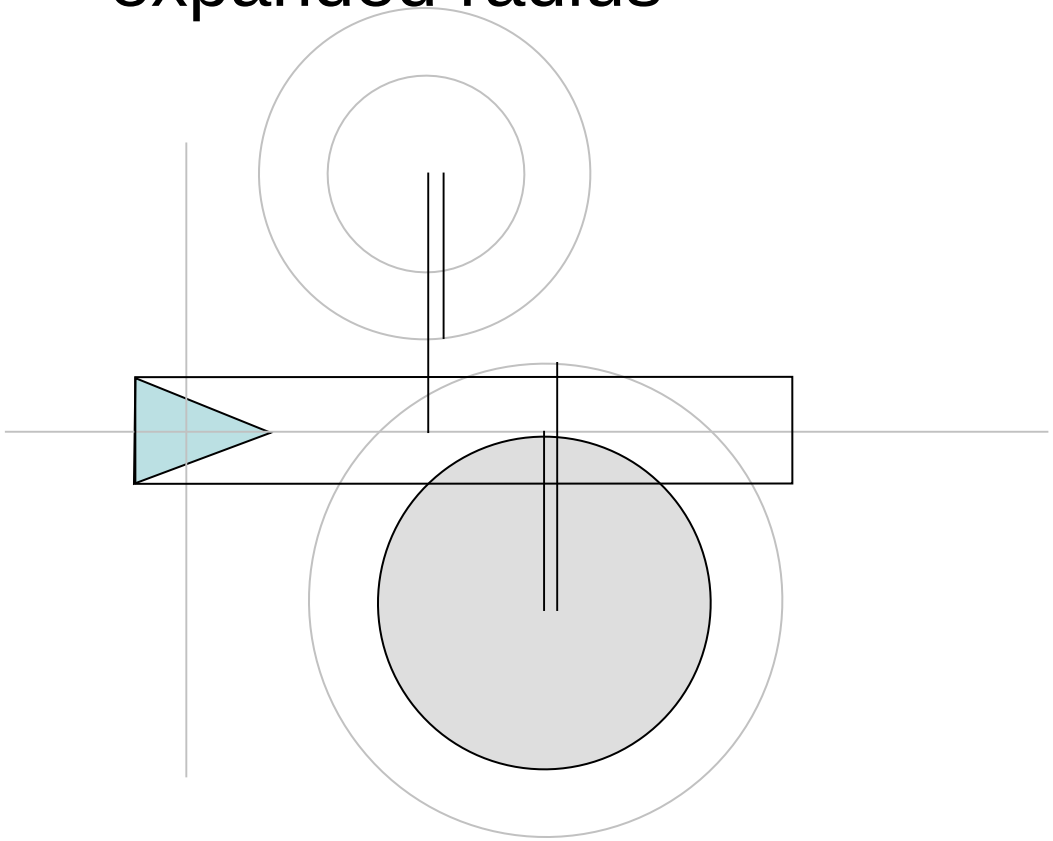
Obstacle Avoidance

- Step 4: Expand remaining obstacles by width of bounding box



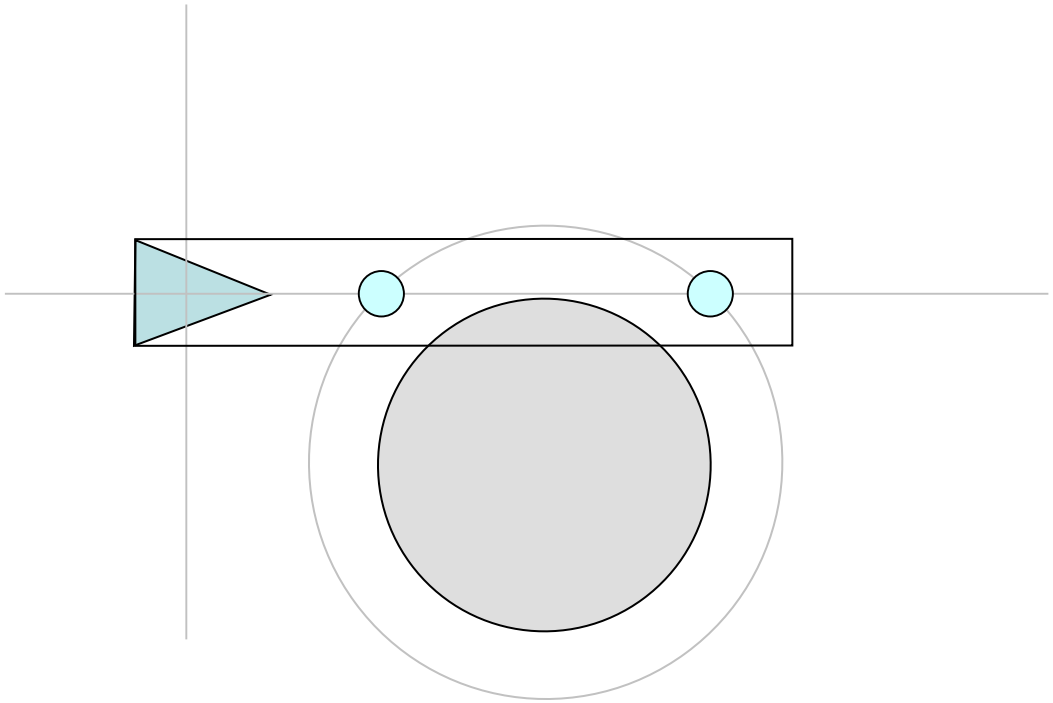
Obstacle Avoidance

- Step 5: Eliminate obstacles whose abs(local y) < expanded radius



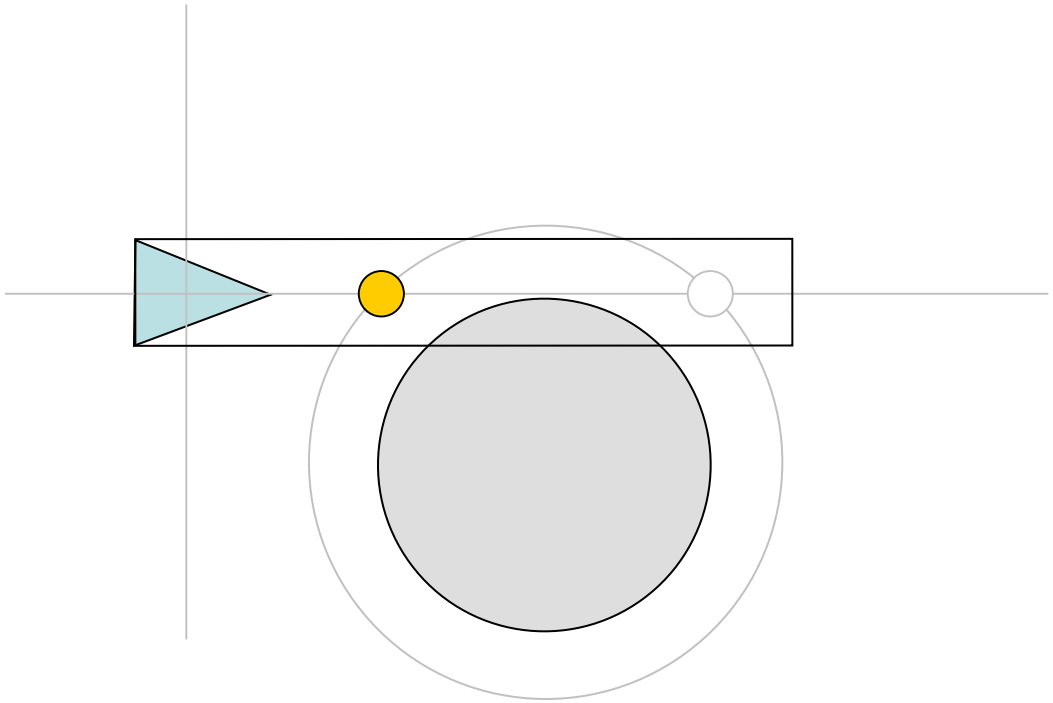
Obstacle Avoidance

- Step 6: Compute intersection points



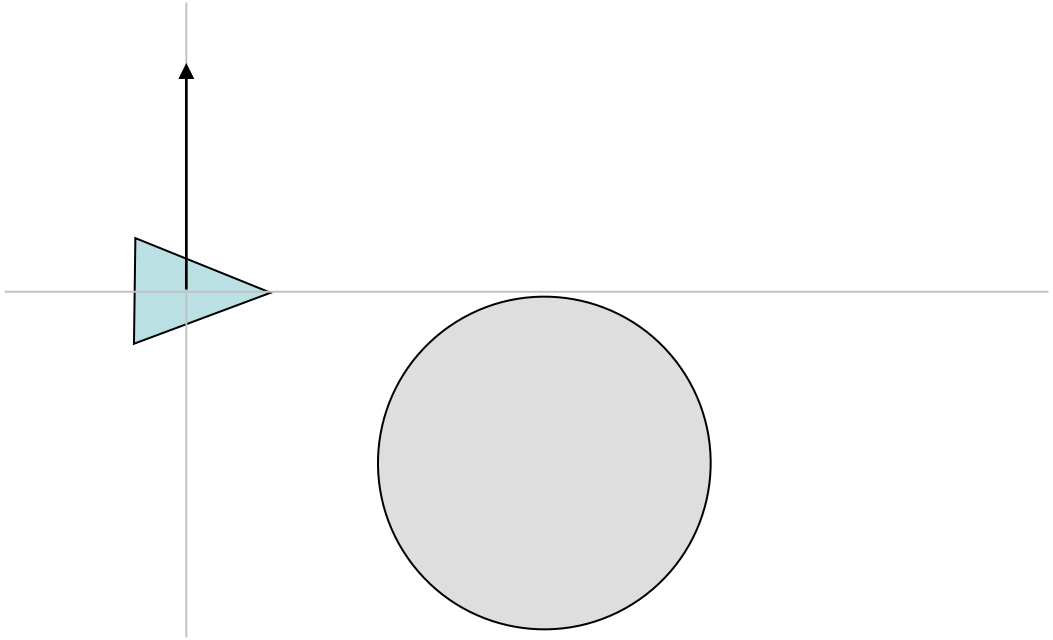
Obstacle Avoidance

- Step 7: Obstacle with closest intersection point used for computing steering forces



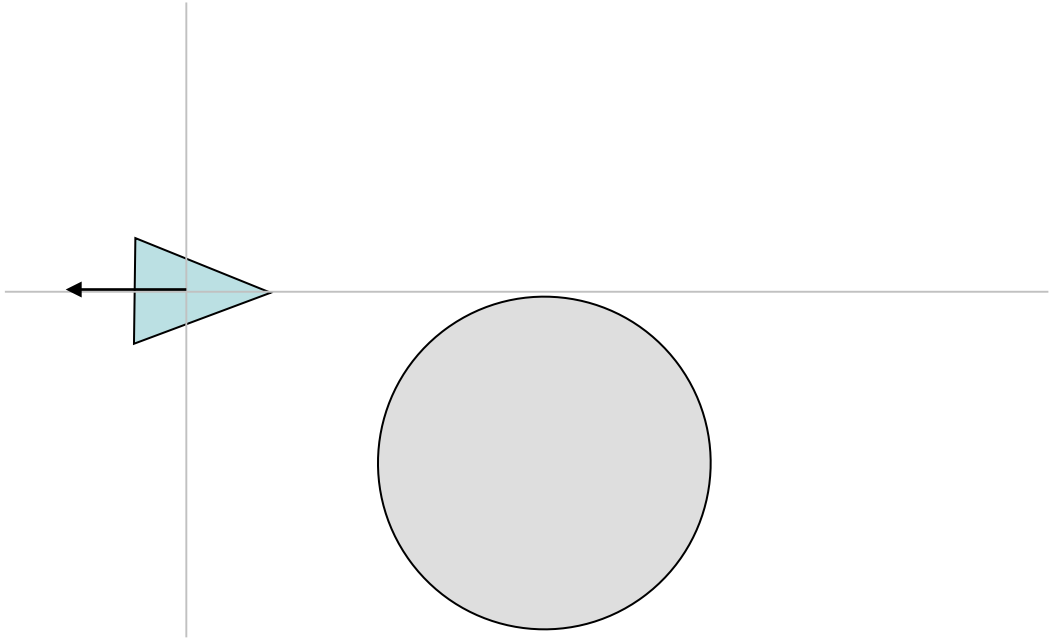
Obstacle Avoidance

- Step 8: Compute lateral force
- The closer the obstacle is to the agent's x-axis, the stronger the lateral force



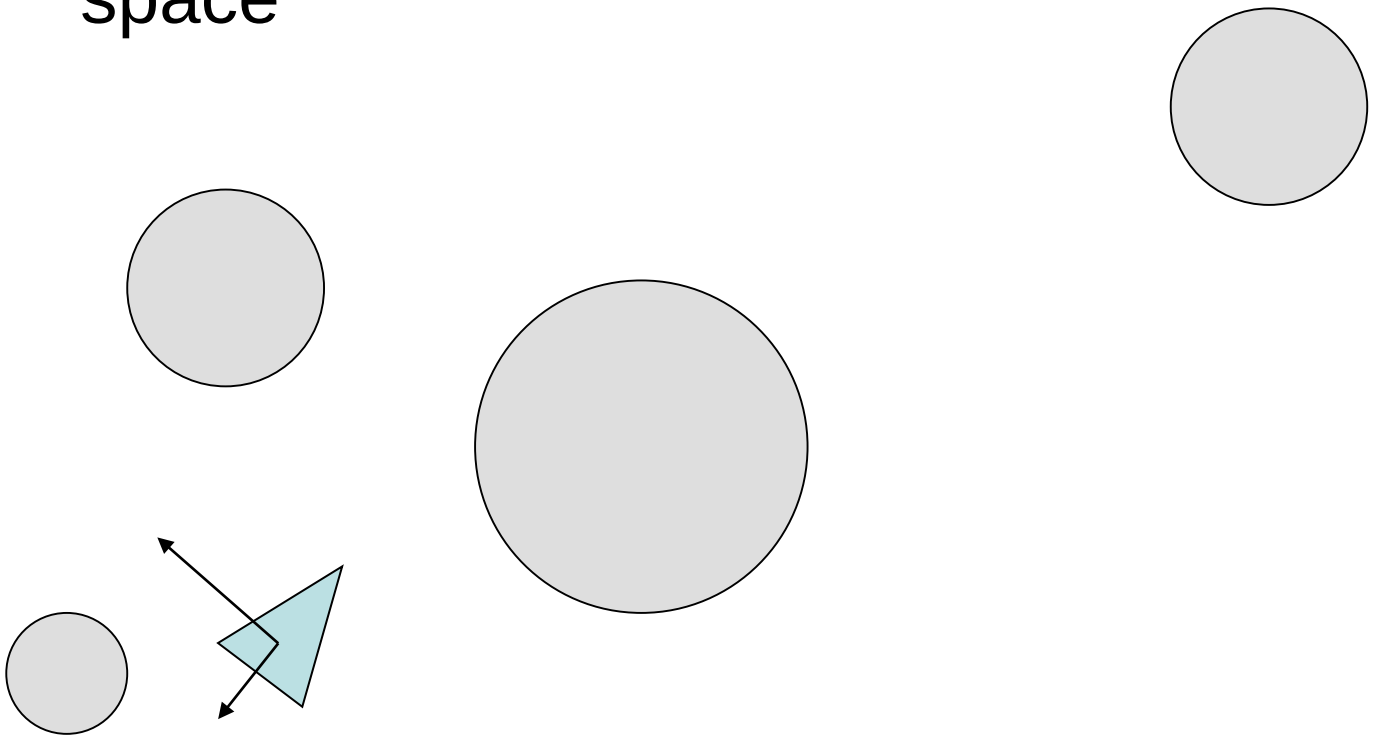
Obstacle Avoidance

- Step 9: Compute braking force
- The closer the obstacle is to the agent’s position, the stronger the braking force



Obstacle Avoidance

- Step 10: Convert lateral & braking force to world space



Obstacle Avoidance

Look at code!

Obstacle Avoidance

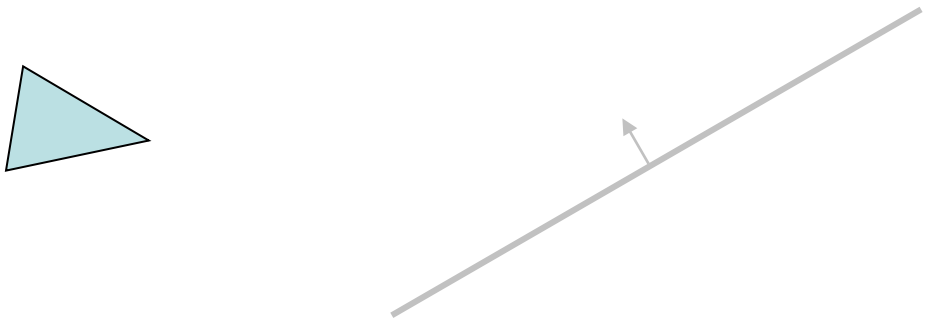
- Pros?
- Cons?
- Questions?

Steering Behaviours

- Seek
- Flee
- Arrive
- Pursuit
- Evade
- Wander
- Interpose
- Hide
- Path following
- Offset pursuit
- Obstacle avoidance
- Wall avoidance

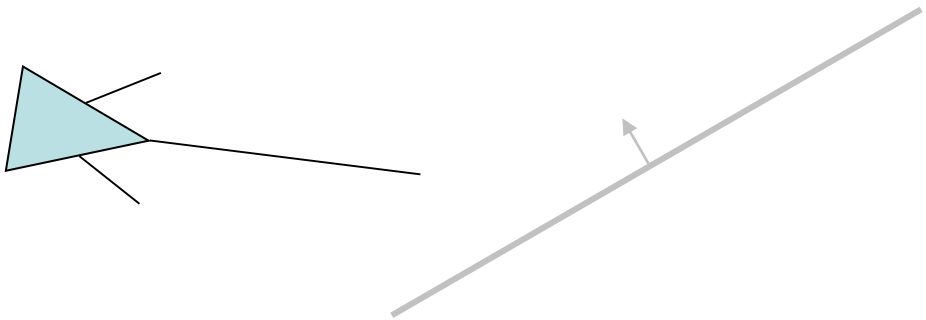
Wall Avoidance

- Steer agent to avoid collisions with walls
- A wall is a line segment (in 2D) or a polygon face (in 3D) with its normal pointing out from its face:



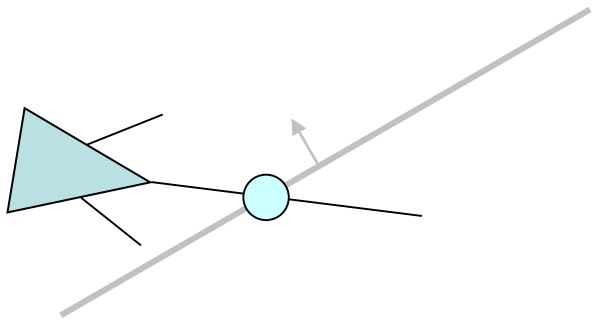
Wall Avoidance

- Initialize: Create “feelers” for the agent



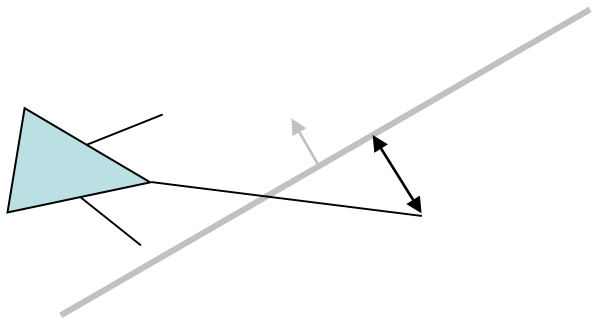
Wall Avoidance

- Step 1: Check if any feeler intersects with a wall



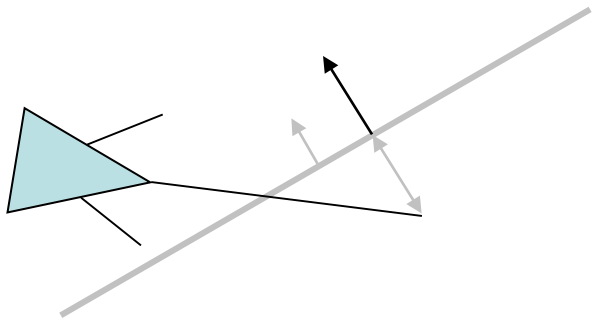
Wall Avoidance

- Step 2: Compute penetration depth



Wall Avoidance

- Step 3: Create force in direction of wall normal, of magnitude equal to penetration depth



Wall Avoidance

- Pros?
- Cons?
- Questions?

Wall Avoidance

Look at code!

Steering Behaviours

- Seek
- Flee
- Arrive
- Pursuit
- Evade
- Wander
- Interpose
- Hide
- Path following
- Offset pursuit
- Obstacle avoidance
- Wall avoidance

Using Steering Behaviours

- What types of autonomous agents can benefit from steering behaviours?

Using Steering Behaviours

- Entities under AI control:
  - Opponent characters
  - NPCs (non-player characters)
  - Vehicles
  - Cameras
  - …?
- All AI-controlled entities can be autonomous agents (to one degree or another)
- All AI-controlled entities can benefit from steering behaviours

Agenda

- Assignment 3 Redux
- Autonomous Agent
- Vehicle
- Steering Behaviours
- Combining Steering Behaviours
- Smoothing
- Assignment 4 Overview

Combining Steering Behaviours

- Vehicle owns instance of SteeringBehaviors
- Specific behaviors are turned on:

```
Vehicle *Sheep = new Vehicle();

Sheep->Steering()->SeparationOn();
Sheep->Steering()->AlignmentOn();
Sheep->Steering()->CohesionOn();
Sheep->Steering()->ObstacleAvoidanceOn();
Sheep->Steering()->WanderOn();
Sheep->Steering()->EvadeOn( Dog );
```

- Vehicle::Update() calculates steering force:

```
SVector2D SteeringForce = m_pSteering->Calculate();
```

- How can we compute total?

Combining Steering Behaviours

- Weighted truncated sum
- Weighted truncated sum with prioritization
- Prioritized dithering

Weighted Truncated Sum

- Add all the forces, truncated at end to maximum
- In pseudocode:

```
for each active behaviour:
    calculate steering force with weighting
    add this force to vehicle’s total steering force

truncate vehicle’s total steering force to maximum
```

- Pros?
- Cons?

Look at code!

Weighted Truncated Sum with Prioritization

- Some forces are more important than others:
  - Wall avoidance
  - Obstacle avoidance
- In pseudocode:

```
sort behaviours from most important to least

for each active behaviour:
    calculate steering force with weighting
    add this force to vehicle’s total steering force
    if vehicle’s total steering force has reached maximum:
        break
```

- Pros?
- Cons?

Look at code!

Prioritized Dithering

- Add probabilities to each steering behaviour
- In pseudocode:

```
add probability to each behaviour
sort behaviours from most important to least

for each active behaviour:
    if random number < behaviour probability:
        calculate steering force with weighting
        add this force to vehicle’s total steering force
        truncate vehicle’s total steering force to maximum
        break
```

- Pros?
- Cons?

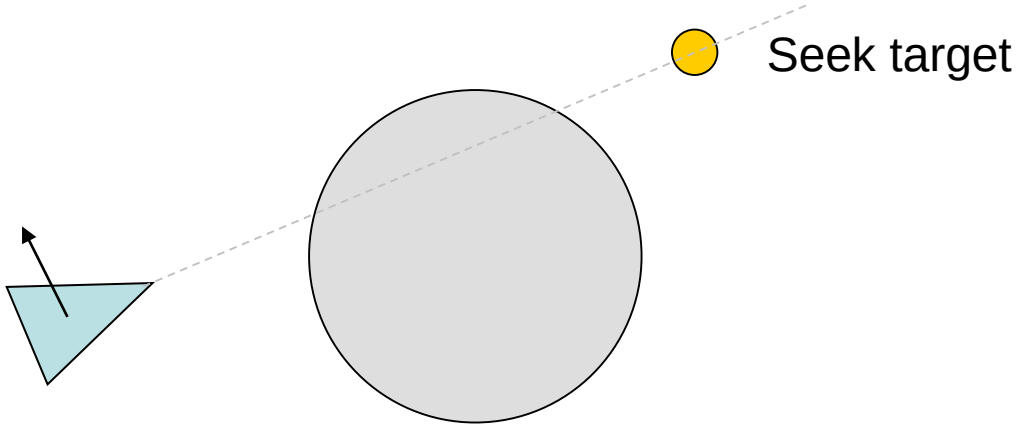
Look at code!

Agenda

- Assignment 3 Redux
- Autonomous Agent
- Vehicle
- Steering Behaviours
- Combining Steering Behaviours
- Smoothing
- Assignment 4 Overview

Smoothing

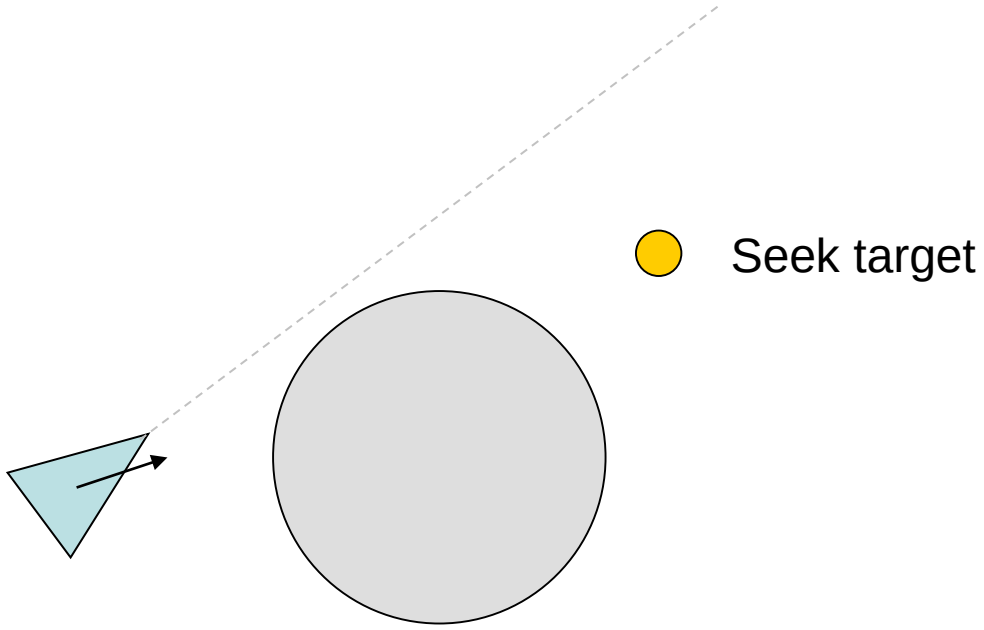
- What occurs when agent has conflicting goals?



Seek pulls agent towards target
Obstacle avoidance pushes agent away from obstacle

Smoothing

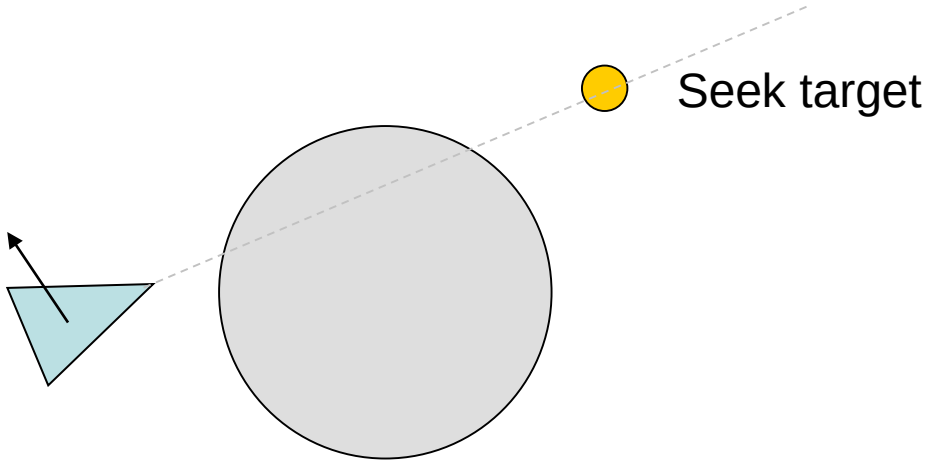
- What occurs when agent has conflicting goals?



Seek pulls agent towards target
Obstacle avoidance pushes agent away from obstacle

Smoothing

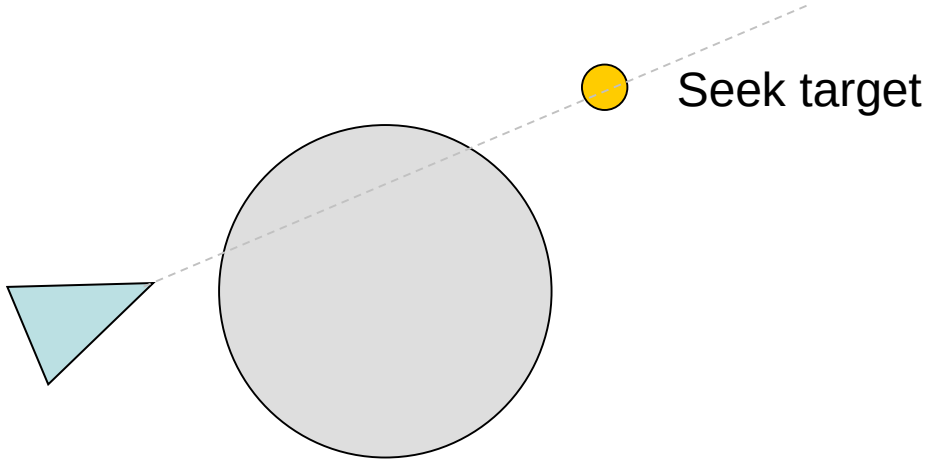
- What occurs when agent has conflicting goals?



Seek pulls agent towards target
Obstacle avoidance pushes agent away from obstacle

Smoothing

- Steering forces create oscillations as conflicting behaviours take precedence
- Agent appears to shake / jitter / judder



- How would you solve this?

Smoothing

- Predicting and avoiding conflict:
  - Yields best result
  - Computationally expensive
- Alternate approach:
  - Decouple heading from velocity vector
  - Average heading over several update steps

Look at code!

Agenda

- Assignment 3 Redux
- Autonomous Agent
- Vehicle
- Steering Behaviours
- Combining Steering Behaviours
- Smoothing
- Assignment 4 Overview