CA-MIRI
Computer Animation
2nd Project

Exercise 4 – Steering

Alejandro Beacco Porres

alejandro.beacco@upc.edu

Exercise Statement

- 1-Integrate Reynolds steering behaviours: at least **obstacle avoidance** and **seek towards waypoints**.
- 2-Improve the basic behaviour with other steering behaviors or concepts from methods studied in class: **path following, unaligned collision avoidance, pursue and evade, follow the leader…**).

Outline

- 1-Steering
- 2-Reynolds steering behaviours
- 3-Integrate steering behaviours

1-Steering
Local movement

- Now you have pathfinding implemented
    - High level navigation
    - You have waypoints set at the center of cells
        - Could be set at portals
    - No steering → straight route towards goal

- How to move locally avoiding obstacles and other agents?
- How to move between waypoints and inside cells?

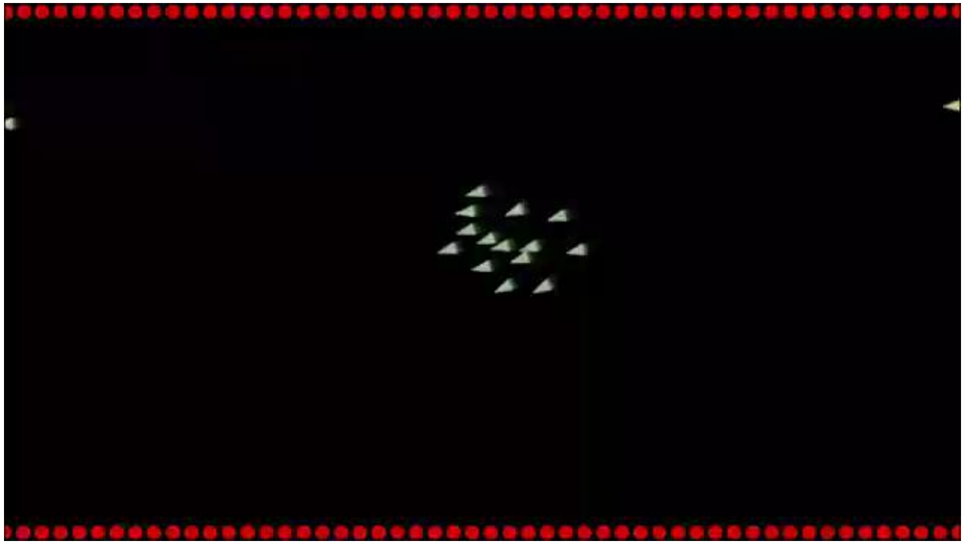
- Use a grid cell size big enough to have more than one agent fitting into cells

1-Steering
Steering force

- Without steering force
  - Straight routes and instantly changes in direction when the target moves, thus making an abrupt transition between the current route and the new one.
- Steering behaviors: influence the character's movement by adding forces (called *steering forces*). Depending on those forces, the character will move in one or another direction.
- Why? It looks more natural

- steering_force = desired_velocity - velocity https://www.youtube.com/watch?v=QbUPfMXXQIY&t=73s

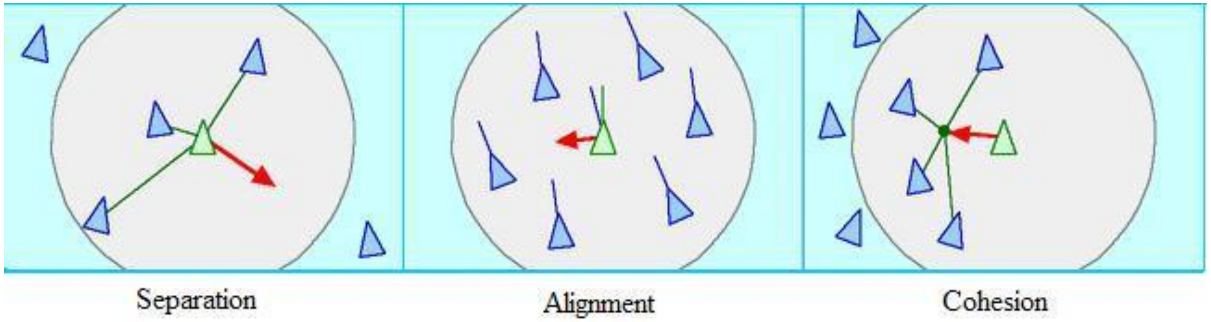
2-Reynolds steering behaviours



Reynolds C.: *Flocks, herds, and schools: A distributed behavior model.* SIGGRAPH 1987

2-Reynolds steering behaviours

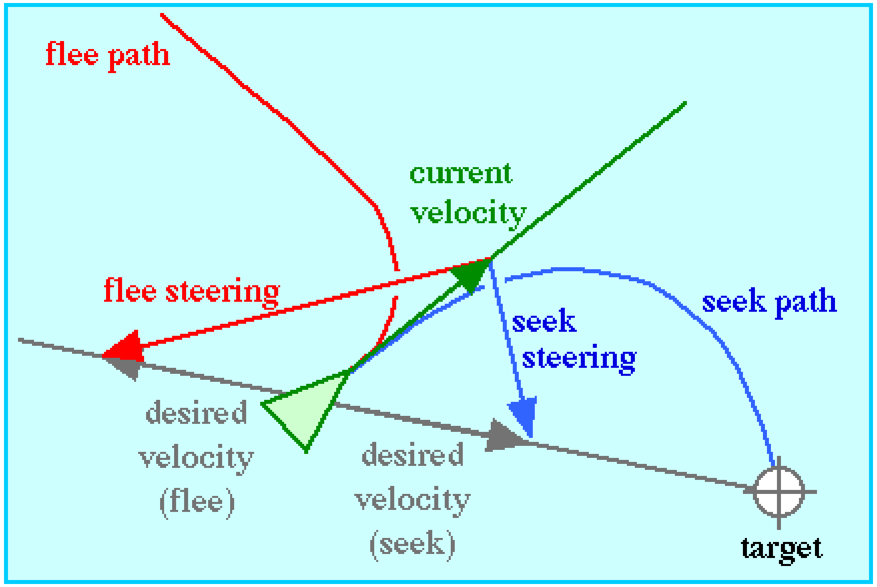
- Boids → 3 forces combined:
  - Separation: steer away from closest neighbors
  - Alignment: steer to align the velocity to the one of its neighbors
  - Cohesion: steer towards the average position of its neihbors
- Weighted sum of forces



Reynolds C.: *Flocks, herds, and schools: A distributed behavior model.* SIGGRAPH 1987

2-Reynolds steering behaviours

- Seek
  - Pursuit of a static target
  - Steer the character towards a specified position in global space
  - Velocity is radially aligned towards the target

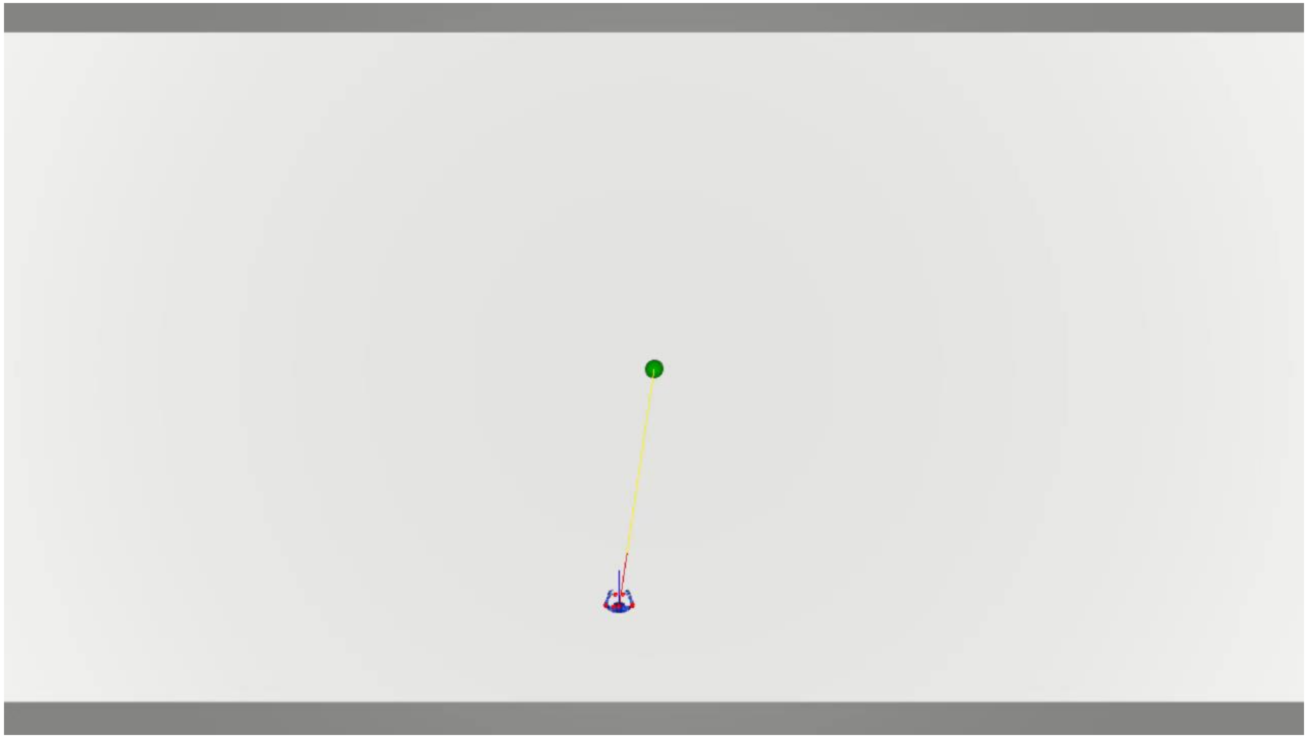


```
Vector3 desired_velocity = (target - position).normalized * maxSpeed;
```

- Flee
  - Inverse of seek

2-Reynolds steering behaviours

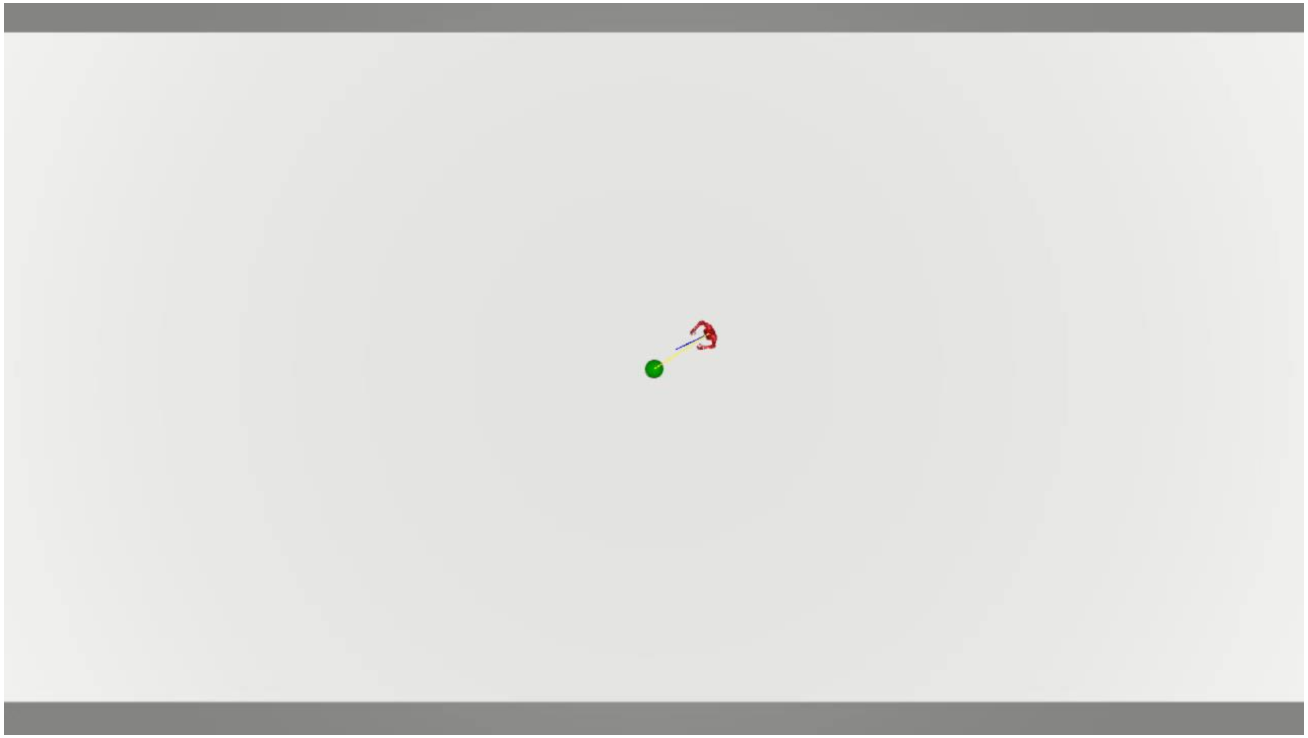
- Seek



```
Vector3 desired_velocity = (target - position).normalized * maxSpeed;
```

2-Reynolds steering behaviours

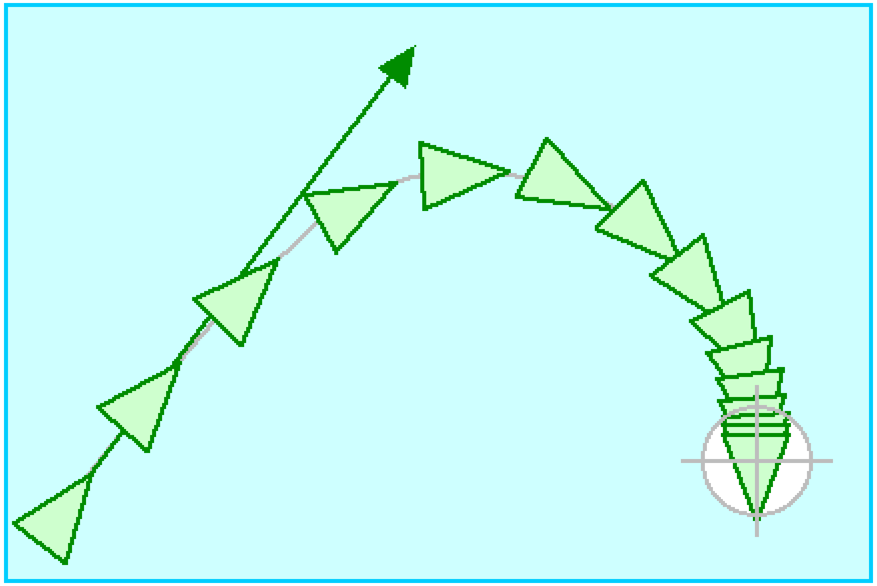
- Flee



```
Vector3 desired_velocity = (position - target).normalized * maxSpeed;
```

2-Reynolds steering behaviours

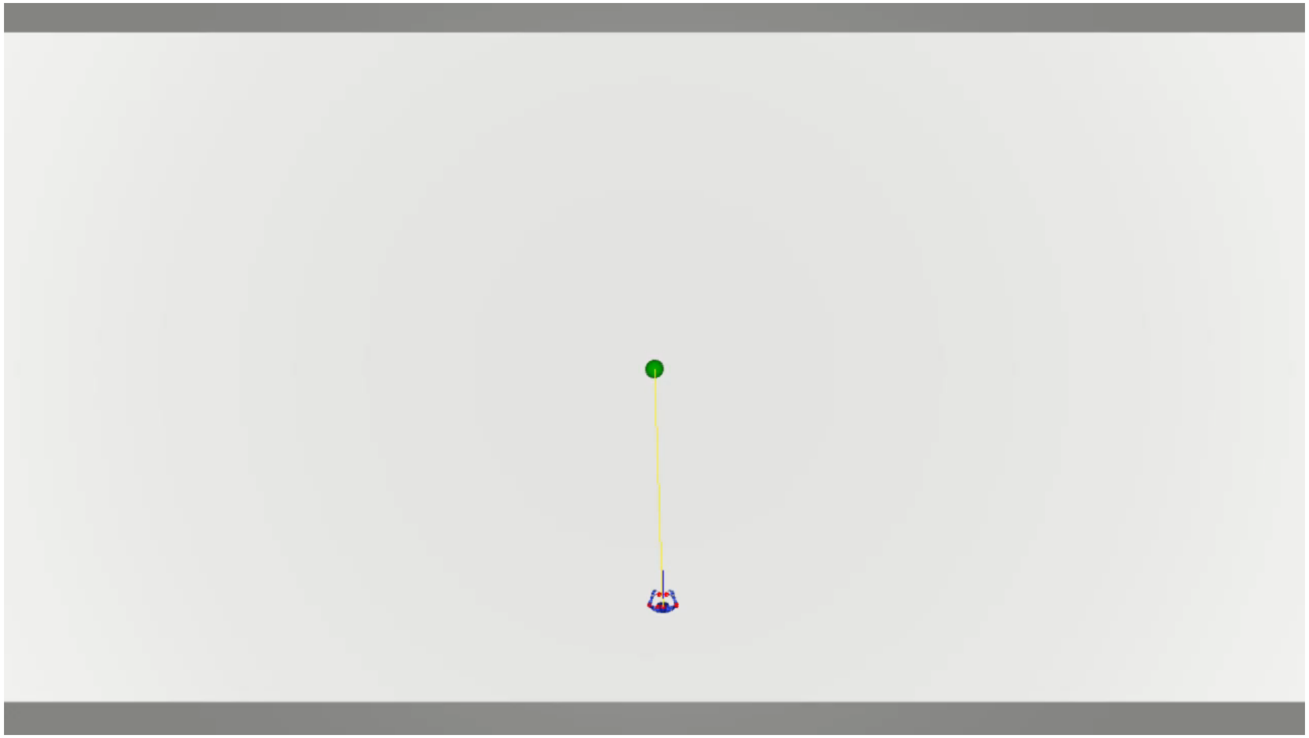
- Arrive
  - Like **seek** while the character is far from its target
  - Slow down as it approaches the target
  - Eventually stopping
  - Parameter: slowing_distance
  - Outside radius
    - desired velocity clipped to max_speed
  - Inside radius,
    - desired velocity is ramped down (e.g. linearly) to zero



```
target_offset = target - position
distance = length (target_offset)
ramped_speed = max_speed * (distance / slowing_distance)
clipped_speed = minimum (ramped_speed, max_speed)
desired_velocity = (clipped_speed / distance) * target_offset
```

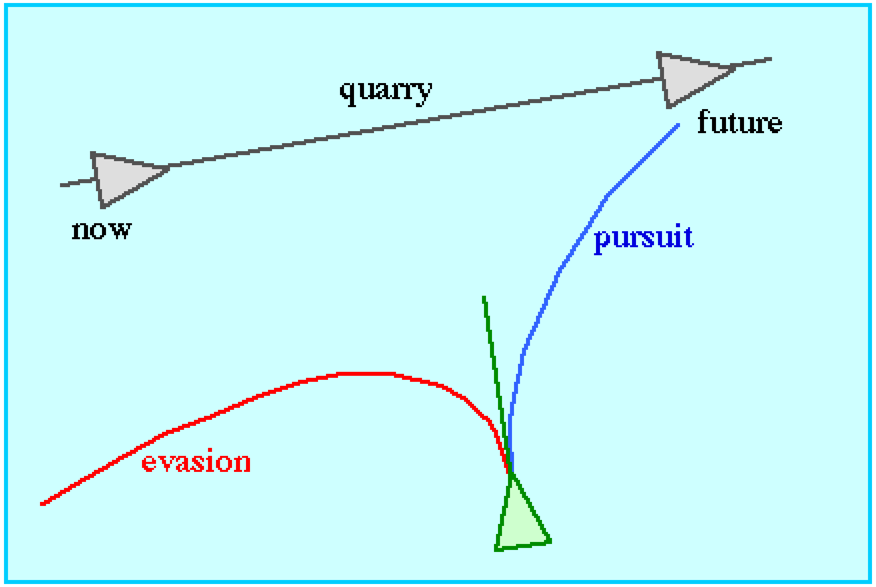
2-Reynolds steering behaviours

- Arrive



2-Reynolds steering behaviours

- Pursue
  - Similar to **seek** except that target is another moving character
  - Prediction of the target’s future position
  - Reevaluate it each simulation step



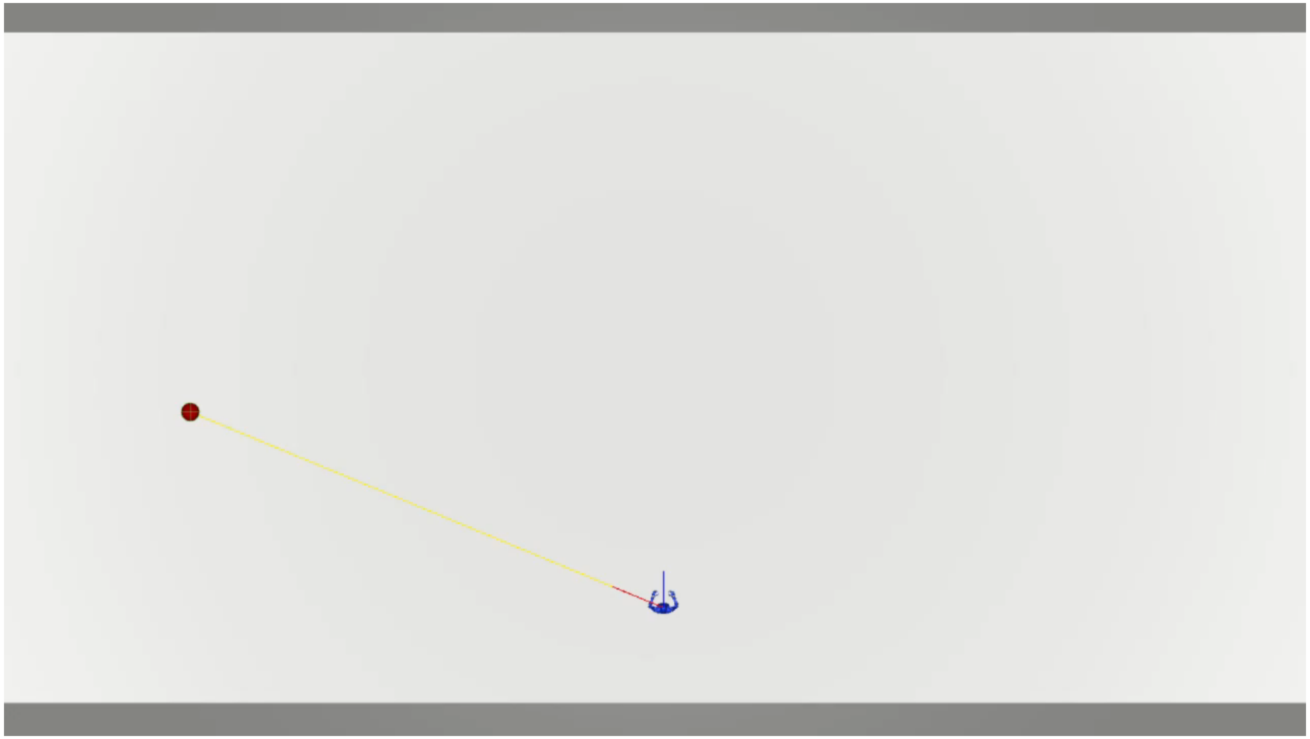
```
Vector3 desired_velocity = (predictedTargetPosition - position).normalized * maxSpeed;
```

- Evade
  - Inverse of pursue

```
Vector3 desired_velocity = (position - predictedTargetPosition).normalized * maxSpeed;
```

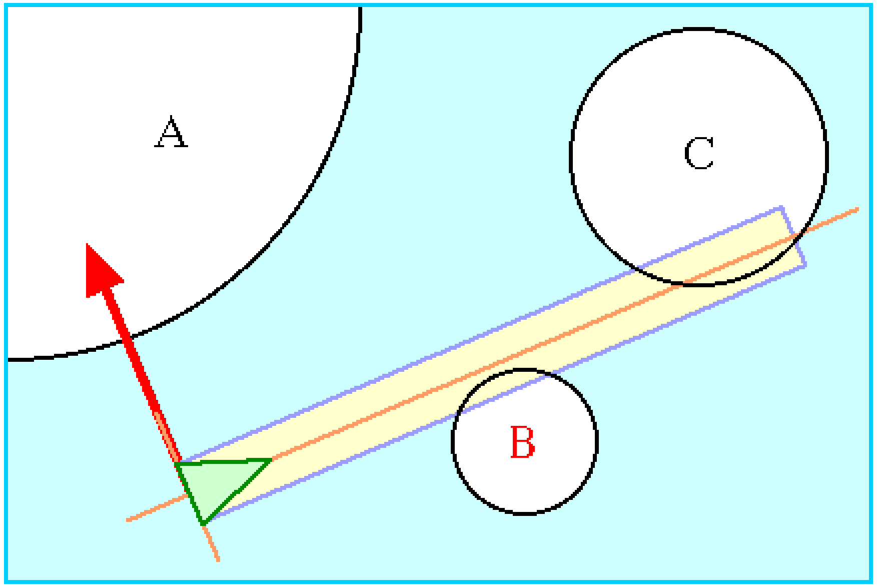
2-Reynolds steering behaviours

- Pursue



2-Reynolds steering behaviours

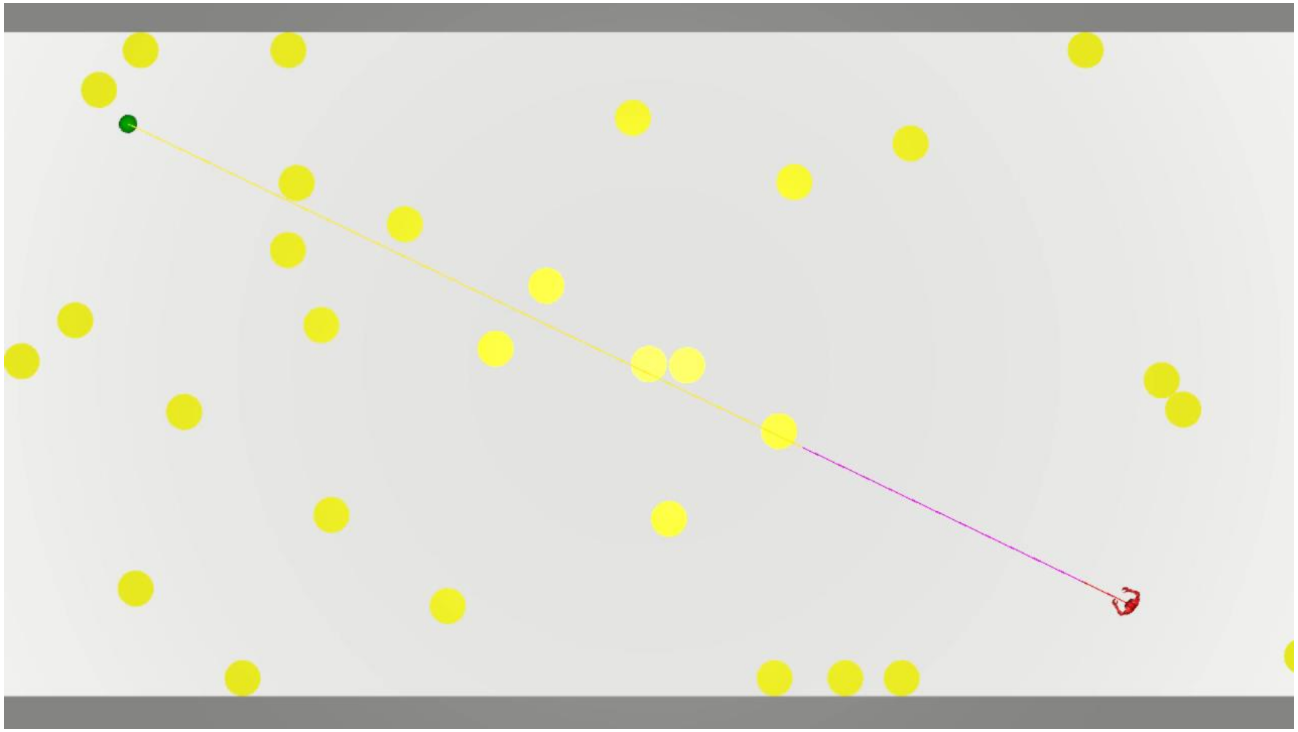
- Obstacle Avoidance
  - Takes action only when a nearby obstacle lies directly in front of the character
  - Character and obstacle can be reasonably approximated as spheres
    - easily extended to more precise shape models
  - Keep an imaginary cylinder of free space in front of the character.
    - *forward* axis
    - diameter equal to bounding sphere
    - extends for a distance based on the character's speed and agility



- Test for non-intersection with the cylinder
- The obstacle which intersects the *forward* axis nearest the character is selected as the "most threatening."
- Steering to avoid this obstacle is computed by negating the (lateral) *side-up* projection of the obstacle's center.
- Generally, we only care about obstacles which are between us and our goal

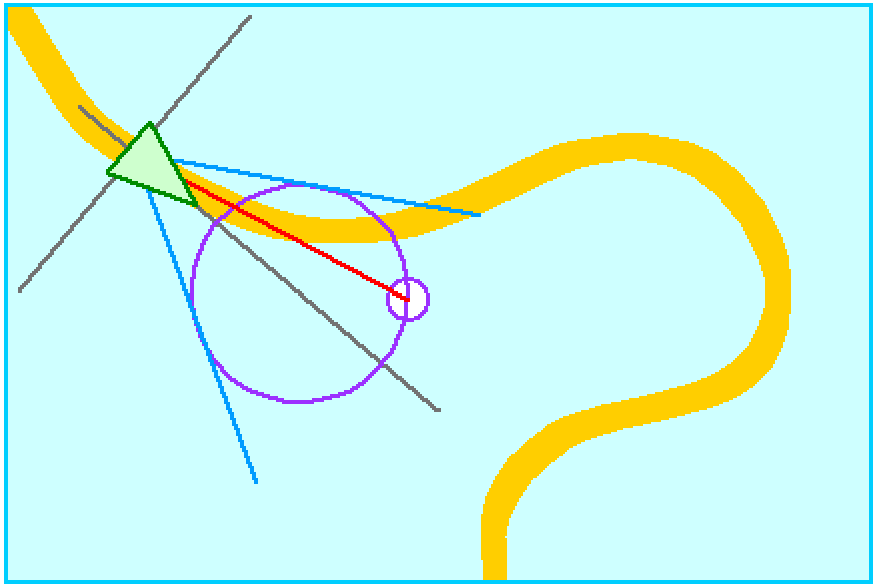
2-Reynolds steering behaviours

- Obstacle Avoidance



2-REYNOLDS STEERING BEHAVIOURS

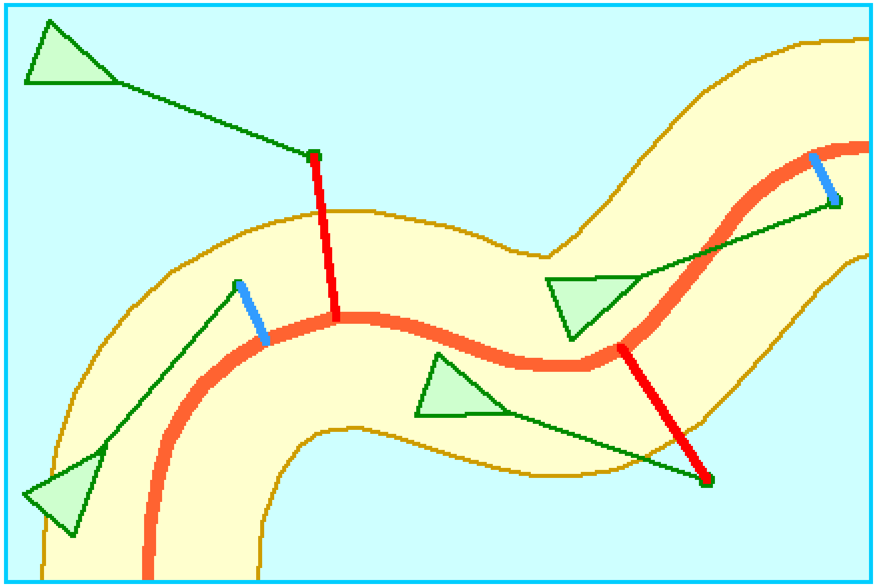
- Wander
  - Random steering
  - Retain steering direction state and make small random displacements to it each frame
  - Constrain the steering force to the surface of a sphere located slightly ahead of the character
    - A random displacement is added to the previous value, and the sum is constrained again to the sphere's surface



- The sphere's radius determines the maximum wandering "strength"
- The magnitude of the random displacement determines the wander "rate."

2-REYNOLDS STEERING BEHAVIOURS

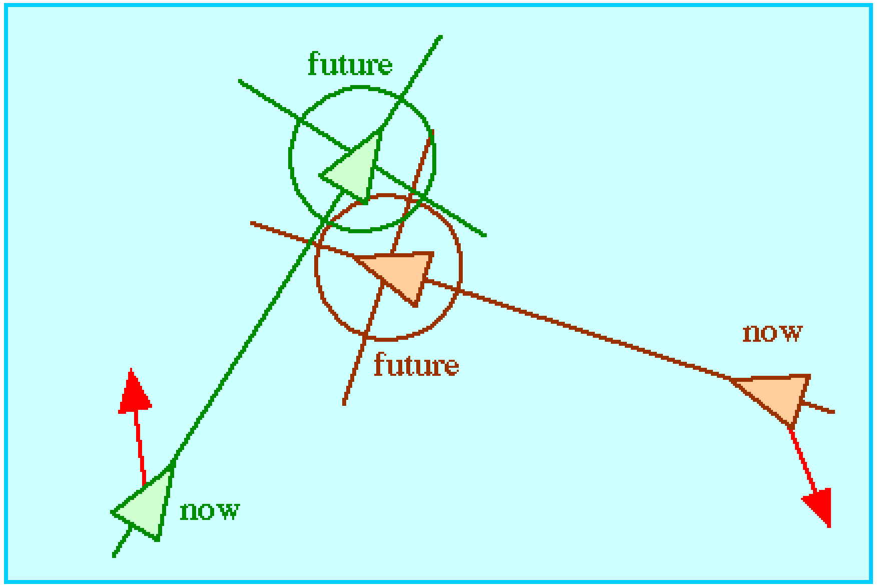
- Path Following
  - Steer along a predetermined path
  - Motion such as people moving down a corridor
    - Individual paths remain near, and often parallel to, the centerline of the corridor, but are free to deviate from it
  - Path:
    - *Spine*: spline curve or polyline
    - *Radius*
  - Move along the path while staying within the specified *Radius* of the spine



- Project predicted position onto the path
- Distance > *Radius*
  - **Seek** towards projection

2-Reynolds steering behaviours

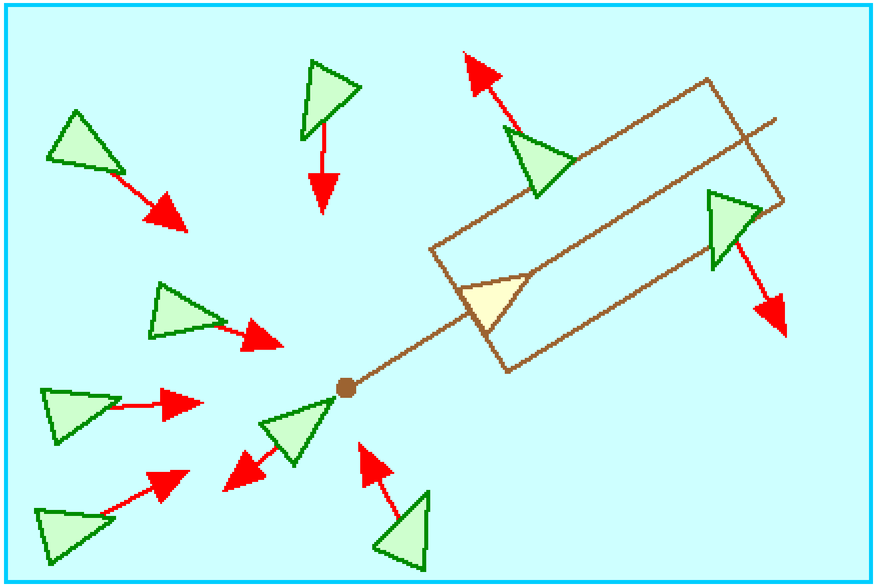
- Unaligned collision avoidance
  - Keep agents from running into each other
  - (If characters are aligned → Separation)
  - Predicts when and where the *nearest approach* will happen
    - Collision if distance small enough



- Steer to avoid the site of the predicted collision
  - Steer laterally to turn away
  - Accelerate forward, or decelerate backward

2-Reynolds steering behaviours

- Follow the leader
  - Stay near the leader, without crowding the leader, and taking care to stay out of the leader's way
  - If more than one follower
    - Avoid bumping each other
  - Arrival behaviour
    - Target: a point offset slightly behind the leader (distance increase with speed)
  - If in front of the leader (rectangular region)
    - Steer laterally away
  - Separation behaviour



2-Reynolds steering behaviours

- More in the original paper!
  - https://www.red3d.com/cwr/papers/1999/gdc99steer.pdf

- Wall following
- Flow following
- Boids
  - Separation
  - Coherence
  - Alignment
- …

3-Integrate steering behaviours

- Simulator
  - Add functions to compute steering forces
    - Vector3 seek (Agent a, Vector3 target) { … return force; }
    - Vector3 flee (Agent a, Vector3 target) { … return force; }
    - Vector3 arrive(Agent a, Vector3 target) { … return force; }
    - …
  - UpdateSimulation(float elapsedTime)
    - Loop over agents
      - Compute steering force combining forces from steering behaviors
      - Apply force to velocity
  - Individual behaviors can be turned on/off or weighted → useful for combining behaviors
    - bool doSeek;
    - bool doFlee;
    - float seekWeight;
    - float arriveWeight;
    - …

3-Integrate steering behaviours

- Apply force
  - Vector3 force = … ; // Combine steering forces
  - force = Truncate(force, maxForce); // limit the force to apply
  - Vector3 acceleration = force / rigidbody.mass; // update acceleration with Newton's 2nd law
  - velocity += acceleration * elapsedTime; // update velocity
  - velocity = Truncate(velocity, maxSpeed); // limit agent speed

```
Vector3 Truncate(Vector3 v, float max){
    float size = min(v.magnitude , max);
    return v.normalized * size;
}
```

- Agent will still update the rigid body position (Euler integration)
  - rigidbody.position += velocity * Time.deltaTime;

3-INTEGRATE STEERING BEHAVIOURS

- Now you have pathfinding implemented
  - High level navigation
  - You have always a current waypoint as goal

- Implement Seek behavior towards that waypoint
- Implement Avoidance behavior with other moving agents and obstacles
  - Simulator has access to all the agents
  - Obstacles are also accessible from your generated grid
    - Each occupied cell can be an obstacle

- Combine the two forces
  - Prioritizing/selecting
  - Blending with weights

3-Integrate steering behaviours

- Improve the basic behaviour with other steering behaviors:
  - You can implement Path following by considering the path to be the poly-line formed by the navmesh waypoints.
  - Maybe you want to have some agents following others, and the others escaping
    - Implement Pursue and Evade
    - Agents are randomly assigned to use Pursue or Evade
    - Each agent has one agent of the other type as a target
    - Paint them with different colors
  - Anything you want or find interesting…

CA-MIRI
Computer Animation
2nd Project

Exercise 4 – Steering

Alejandro Beacco Porres

alejandro.beacco@upc.edu