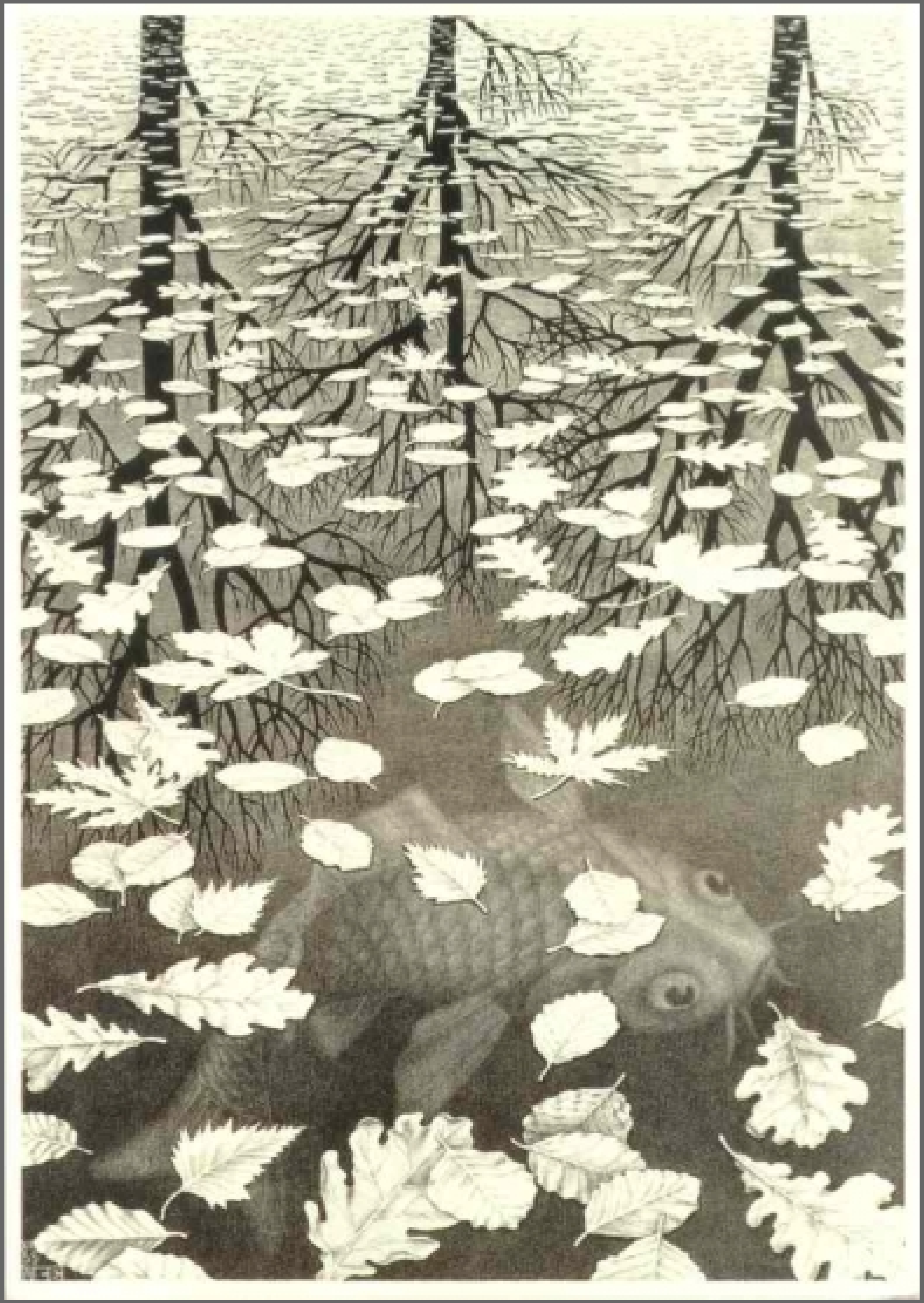
Steering Behaviors:
Autonomous Characters in Three Worlds

Craig Reynolds
Sony Computer Entertainment America
April 23, 2001

http://www.red3d.com/cwr/



Three Worlds
M. C. Escher

In this talk

- Brief review of autonomous characters
  - Definitions
  - Applications
- Steering behaviors
  - Toolkits and procedural composition
  - Evolutionary computation
  - Physical realism
    - Point−mass versus rigid−body dynamics

Autonomous characters

- Self−directing characters, operate autonomously
  - "Puppets that pull their own strings" (Ann Marion)
- Combination of:
  - Geometrical model of body
  - Animation data or procedures for body
  - Behavioral model

Autonomous characters in animation

Autonomous characters in games

Autonomous characters: groups

- Individual
  - simple local behavior
  - interaction with:
    - nearby individuals
    - local environment
- Group:
  - complex global behavior

Types of behavioral models

- Kinematic (animation)
- Dynamic (physical simulation)
- Volition
  - Reactive
    - Like instinct, off−the−cuff decision making
  - Rule based
    - Expert system: search through large knowledge base
  - Planning
    - Search through space of actions and consequences

A behavioral hierarchy

- Action selection
  - Setting goals, picking strategies
- Path selection: steering
  - Character’s motion through its world
- Pose selection: locomotion
  - Legs walking, arms reaching
  - Wheels rolling
  - etc.

Steering behaviors

- Simple, basic behaviors

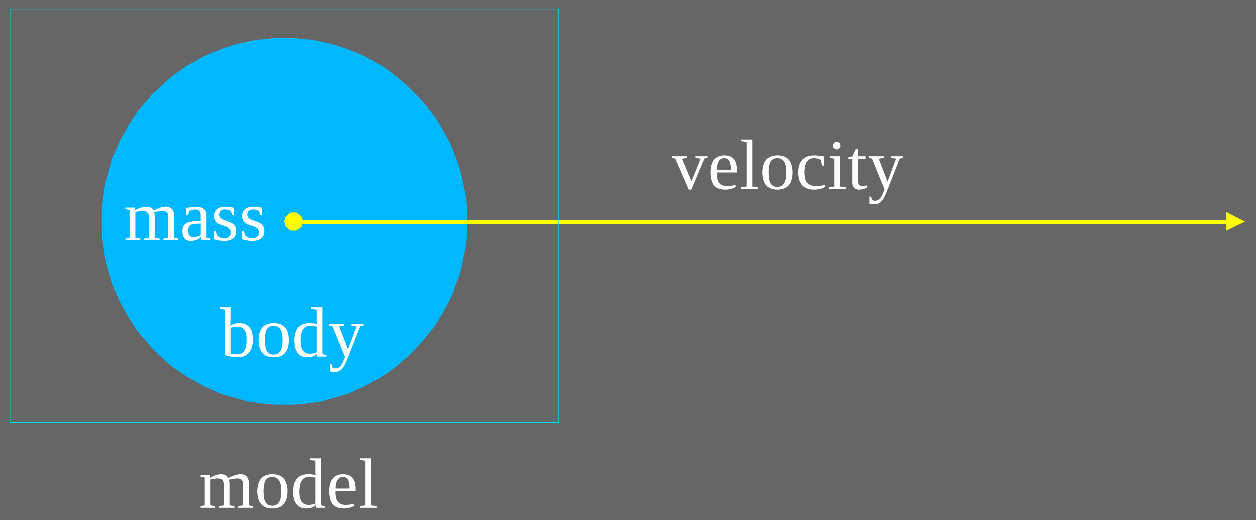
  (seek, flee, wander, ...)
- Operators to combine them

  (sum, prioritized selection, dithered decision trees)
- Toolkit of simple and combined behaviors

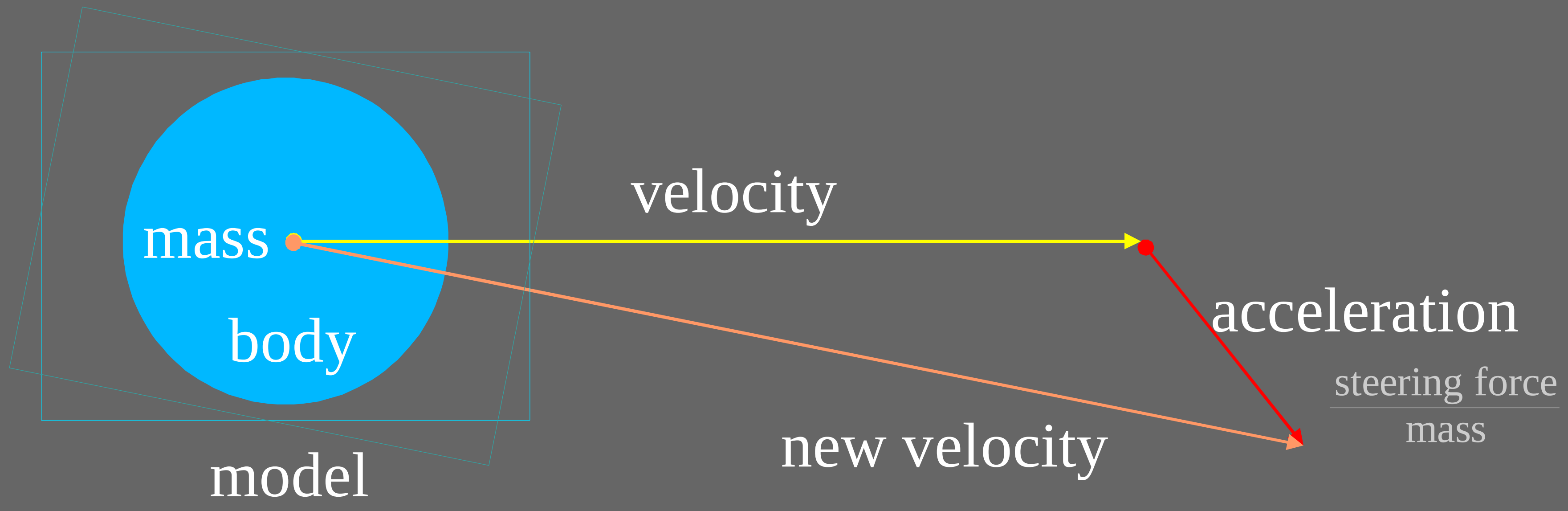
Simple physical model

- Point mass model:
  - Position, adjusted by velocity
  - Velocity, adjusted by steering forces
  - Linear momentum (zero radius: no moment of inertia)
  - Truncation of force and velocity (power limit, drag)
- Body shape: sphere (or ellipsoid)
- Velocity−aligned local coordinate system
  - Animated geometrical model can be attached

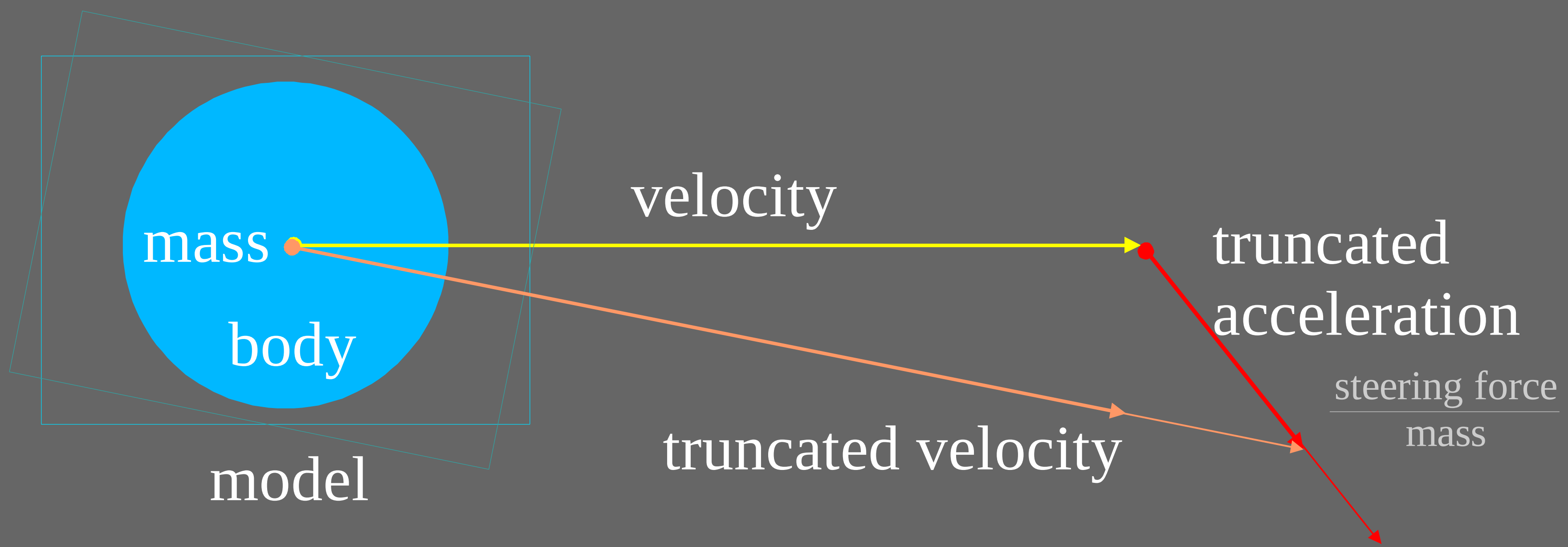
Point mass vehicle model (1)



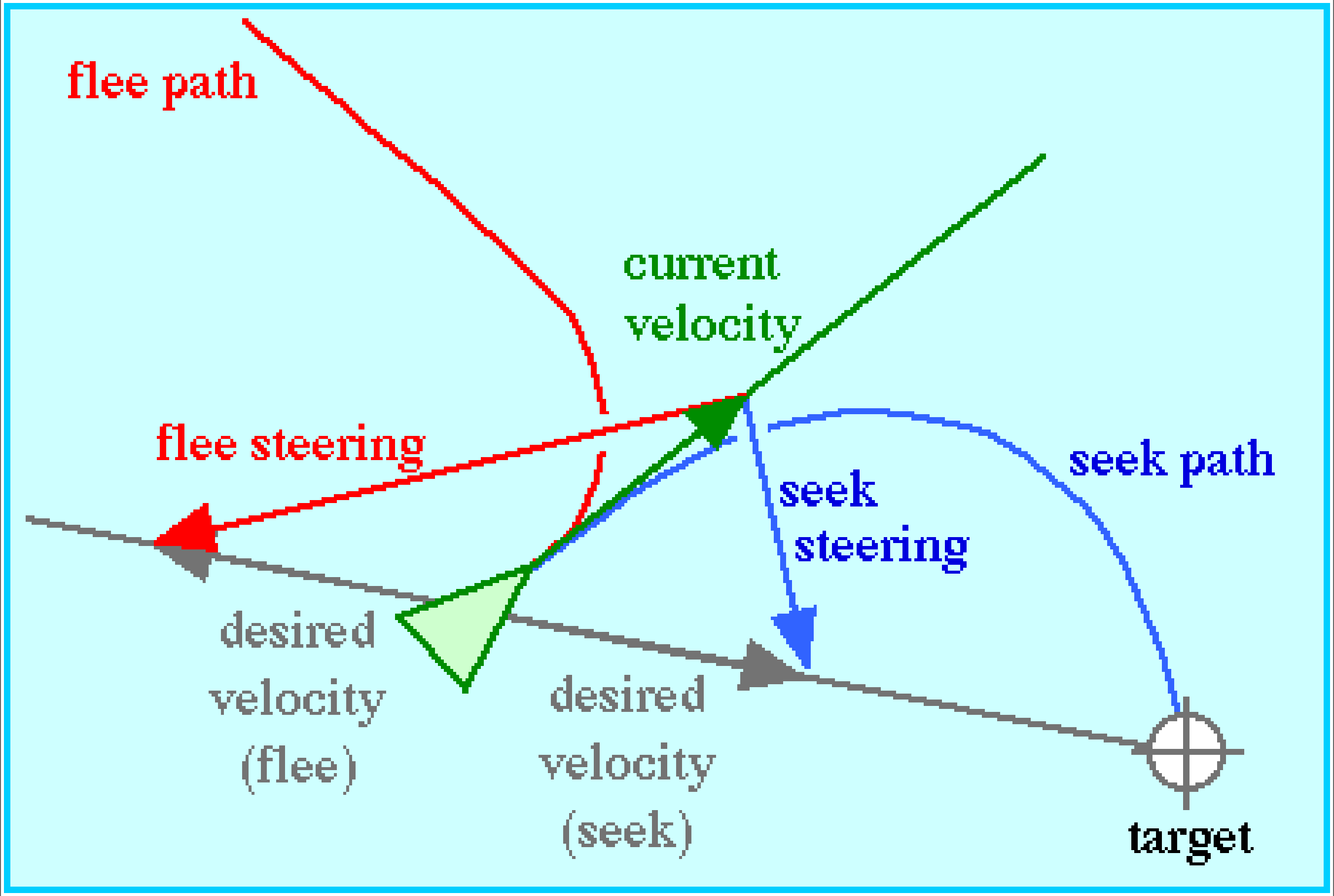
Point mass vehicle model (2)



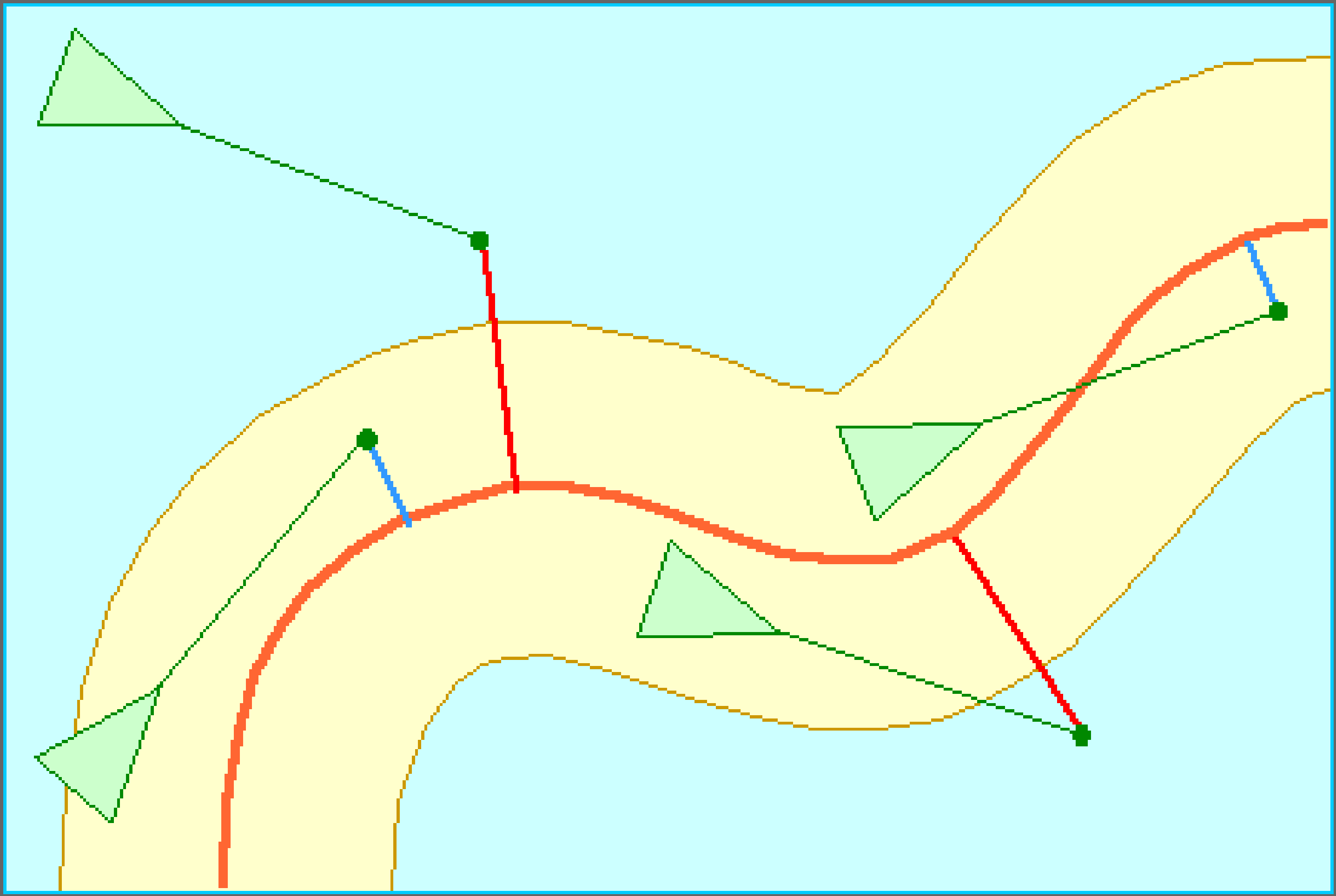
Point mass vehicle model (3)



Steering details: *seek* and *flee*



Steering behavior demos



Boids and flocking

- *Historical note: fits in better here, but actually preceded general steering behaviors (1987)*
- Natural flocks are beautiful, and a bit mysterious
  - Can they be portrayed in computer animation?
  - Perhaps gain some insight into how they work?

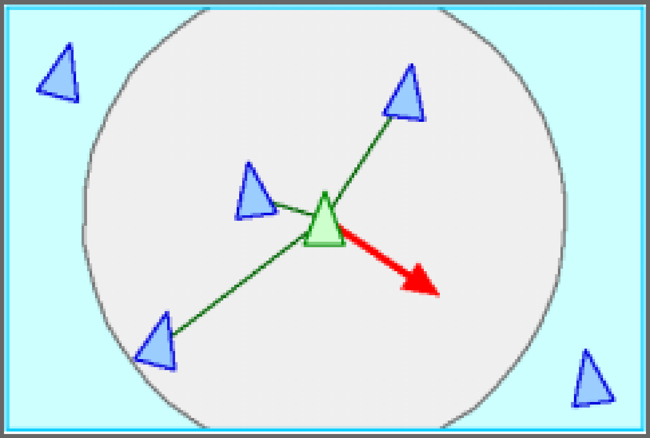
    (ALife −− artificial life)
  - Can the complex group behavior be explained in terms of simple behavior by the individuals?

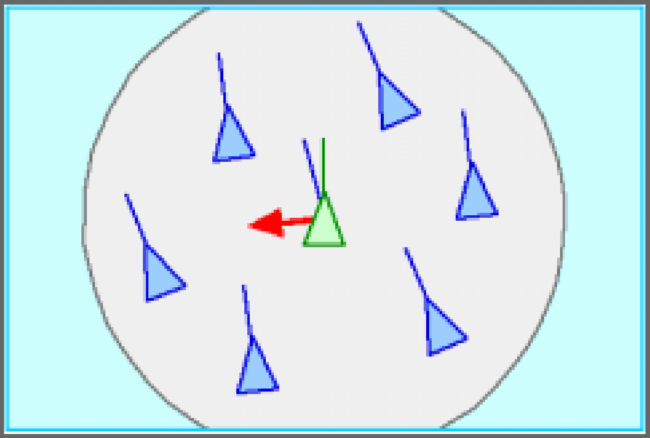
    (CAS −− complex adaptive systems)

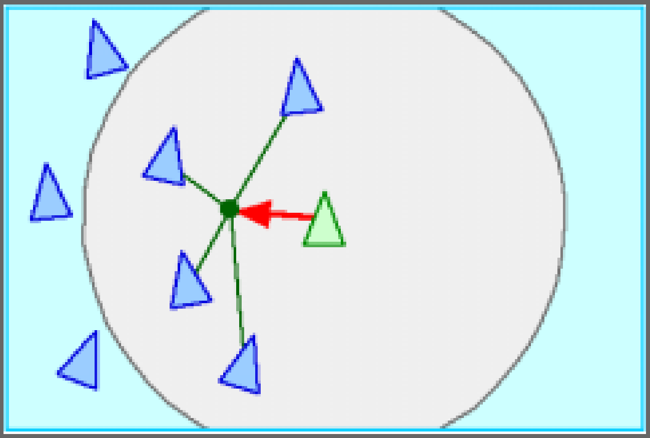
Boids: three rules

- Three rules seemed *necessary*:
  - Separation
    - Don’t get too close to nearby flockmates
  - Alignment
    - Try to move at the same speed and direction (velocity) as nearby flockmates
  - Cohesion
    - Prefer to be at the center of the local flockmates
- Early experiments verified they were *sufficient*.

Boids: three rules


Separation


Alignment


Cohesion

Boids for animation production

- Obstacle avoidance
- Flocking
  - Separation
  - Alignment
  - Cohesion
- Attraction to (or repulsion from) a moving target

Stanley and Stella in Breaking the Ice

Pigeons in the Park

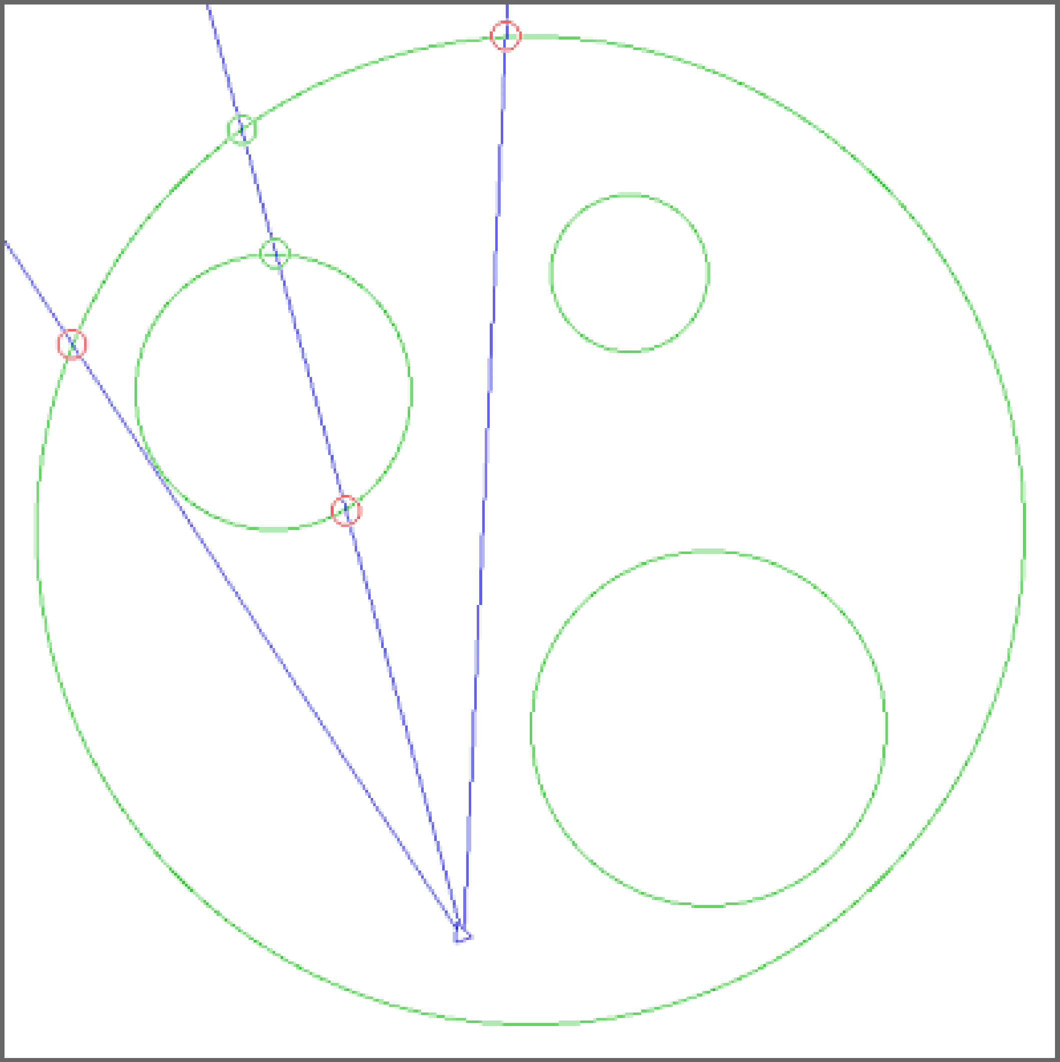
- Based on the 1987 boids model of flocks, herds and schools
- Uses fast hardware (PS2), and spatial data structures to accelerate boids: about 6000 times faster than in 1987.
- Allows real time (60 fps) interaction with a group of about 300 birds.
- Includes behavioral state transitions

Pigeons in the Park video

Coevolution of Tag Players

- The game of tag
  - symmetrical pursuit and evasion
  - role reversal
- Goal: discover steering behavior for tag
- Method: emergence of behavior
  - coevolution
  - competitive fitness
- Self−organization:
  - no expert knowledge required

Sensors and obstacles



Evolutionary computation (overview)

- Genetic programming
- Steady state population
- Coevolution
- Species and demes

Evolutionary computation (details)

- Genetic programming (versus genetic algorithm)
  - Genetic material: source code, as parse tree
- Steady state population (versus generations)
  - Pool of individuals (programs), replace one at a time
- Coevolution (versus *a priori* fitness criteria)
  - New program competes against others in population
- Species and demes (versus panmixia)
  - Crossover within species, competition within demes

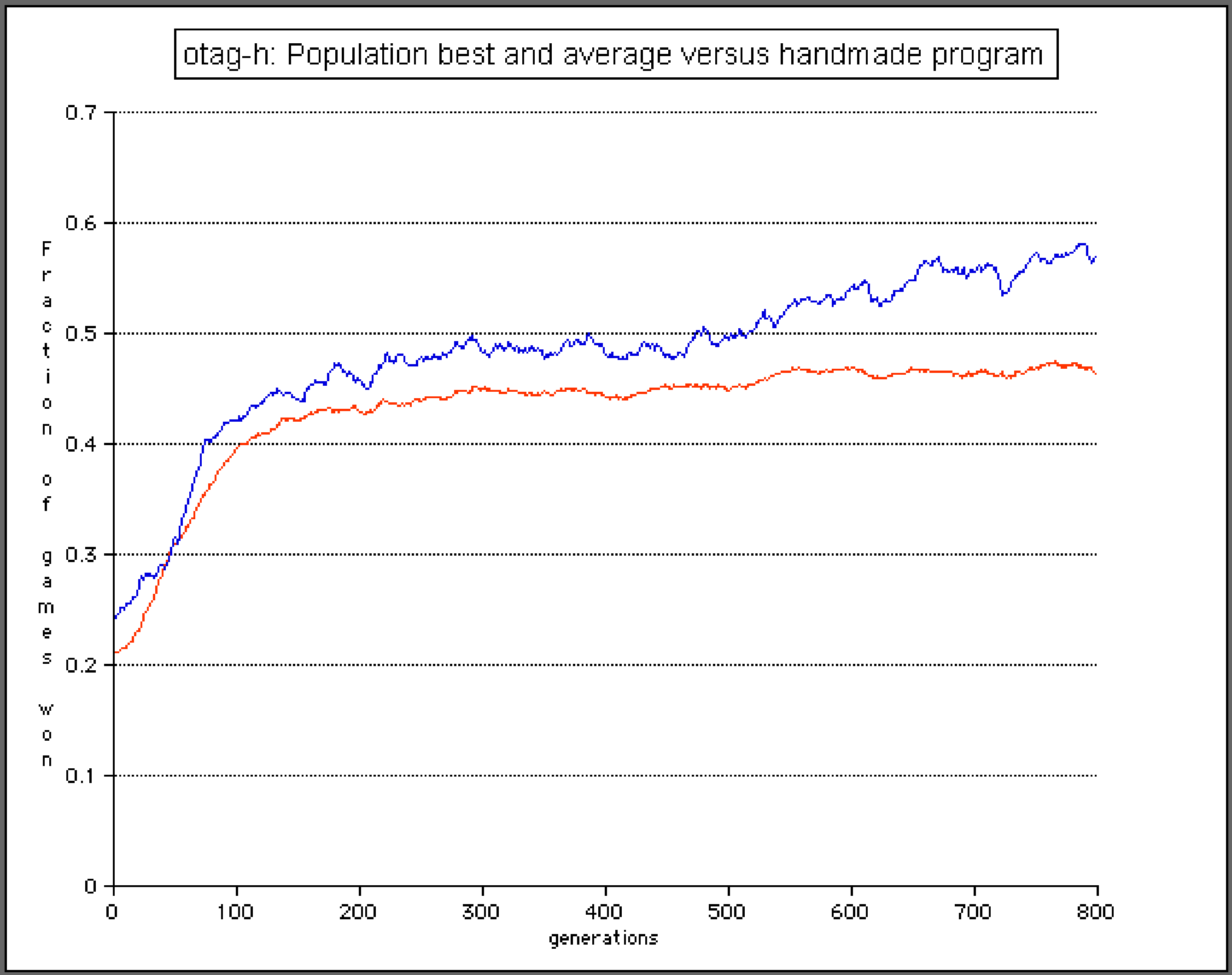
Genetic programming: crossover

```
(iflte (sensor-r2)
       (local-y)
       (sensor-r2)
       (* (sensor-r2)
          0.67))
```

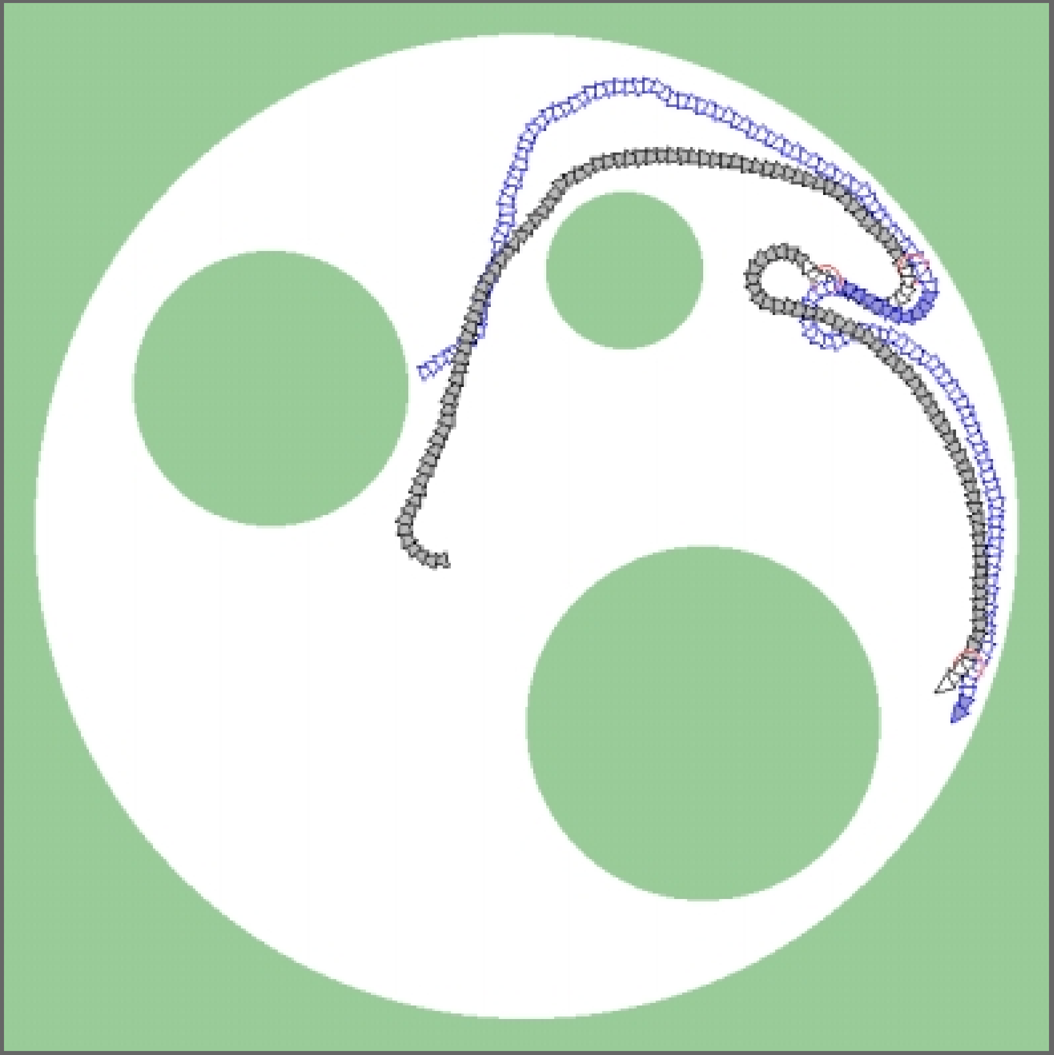
```
(* (+ (local-vy)
      (sensor-r2))
   (sensor-r2))
```

```
(iflte (sensor-r2)
       (local-y)
       (+ (local-vy)
          (sensor-r2))
       (* (sensor-r2)
          0.67))
```

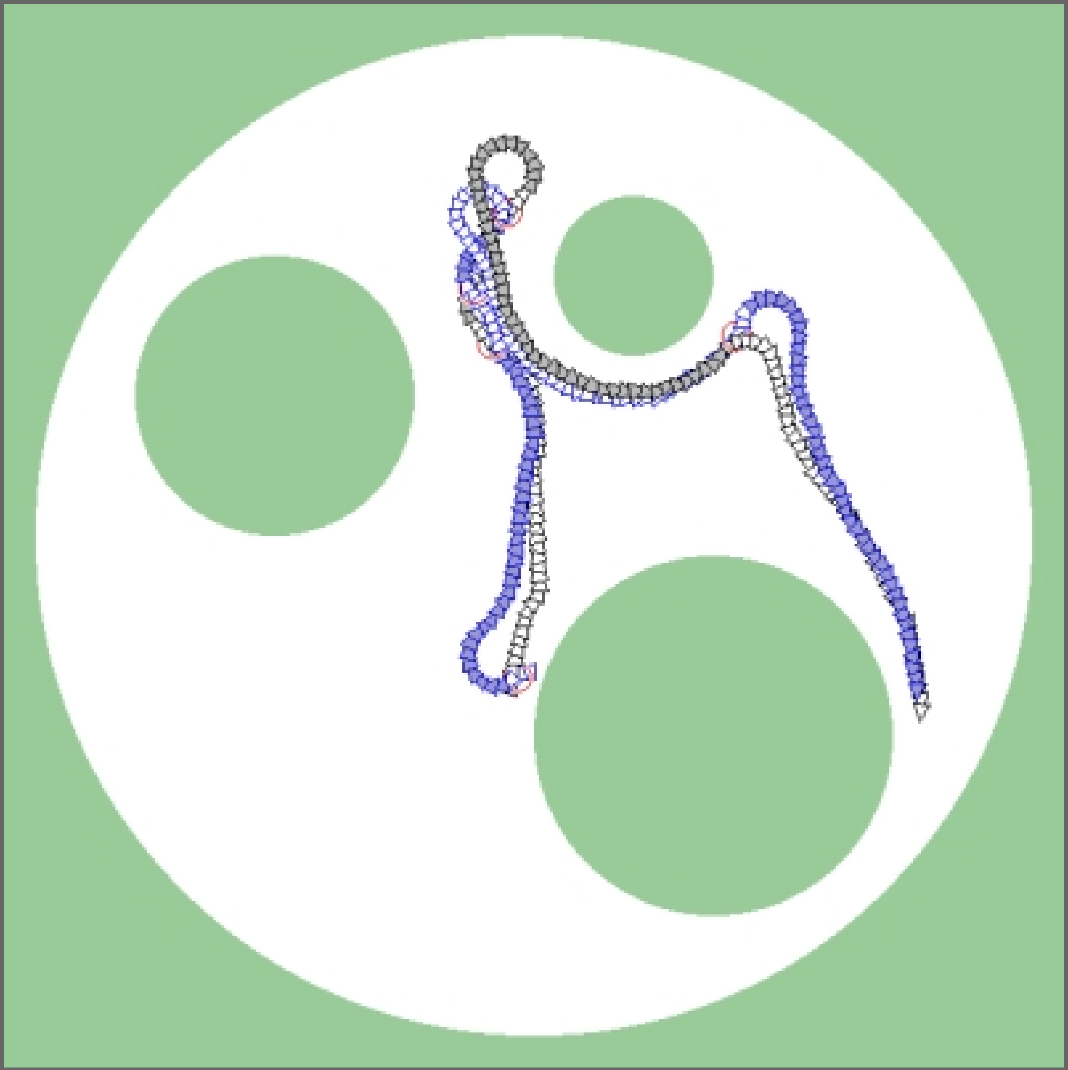
It works!


otag-h: Population best and average versus handmade program
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Fraction of games won
0
100
200
300
400
500
600
700
800
generations

Typical fitness test (1)



Typical fitness test (2)



Competitive coevolution: summary

- Pros:
  - Good results, comparable to human−designed players
  - Diversity and skill gradation from evolution history
  - Does not require knowing a winning strategy or how to implement it.
- Cons:
  - Requires very long computation time even for a very simple game.
  - Untested for games requiring complex strategy.

Steering and physical realism

Steering and physical realism

- Previous topics use simplistic models of physics
- Work in progress:
  - Real time rigid body dynamics simulator (Eric Larsen)
  - Virtual robot soccer world (Eric Larsen)
  - Autonomous steering behaviors for playing soccer
- More accurate physical model requires more sophisticated steering behaviors.

Earlier work: simplified physics

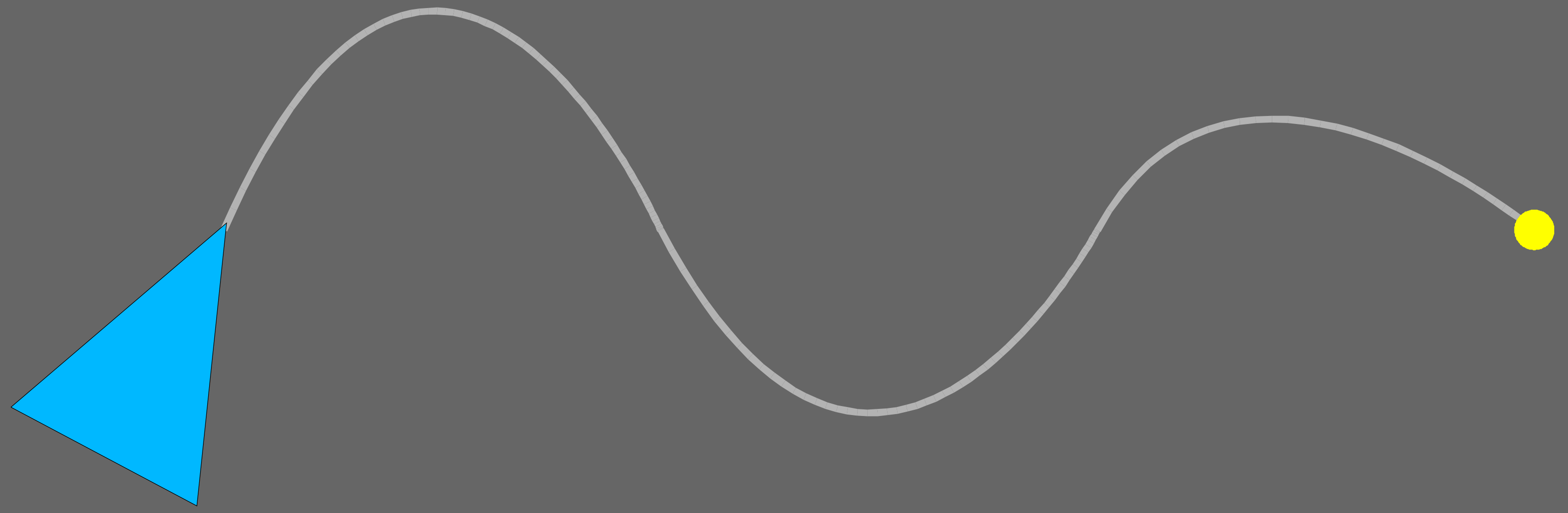
- Boids (1987), steering behavior toolkit (GDC 1999)
  - Point mass model:
    - Position
    - Velocity, so linear momentum
    - Zero radius, so no moment of inertia
  - Spherical (or ellipsoidal) body
- Evolution of steering behaviors
  - *Physically plausible* kinematic model

Steering for accurate physical models

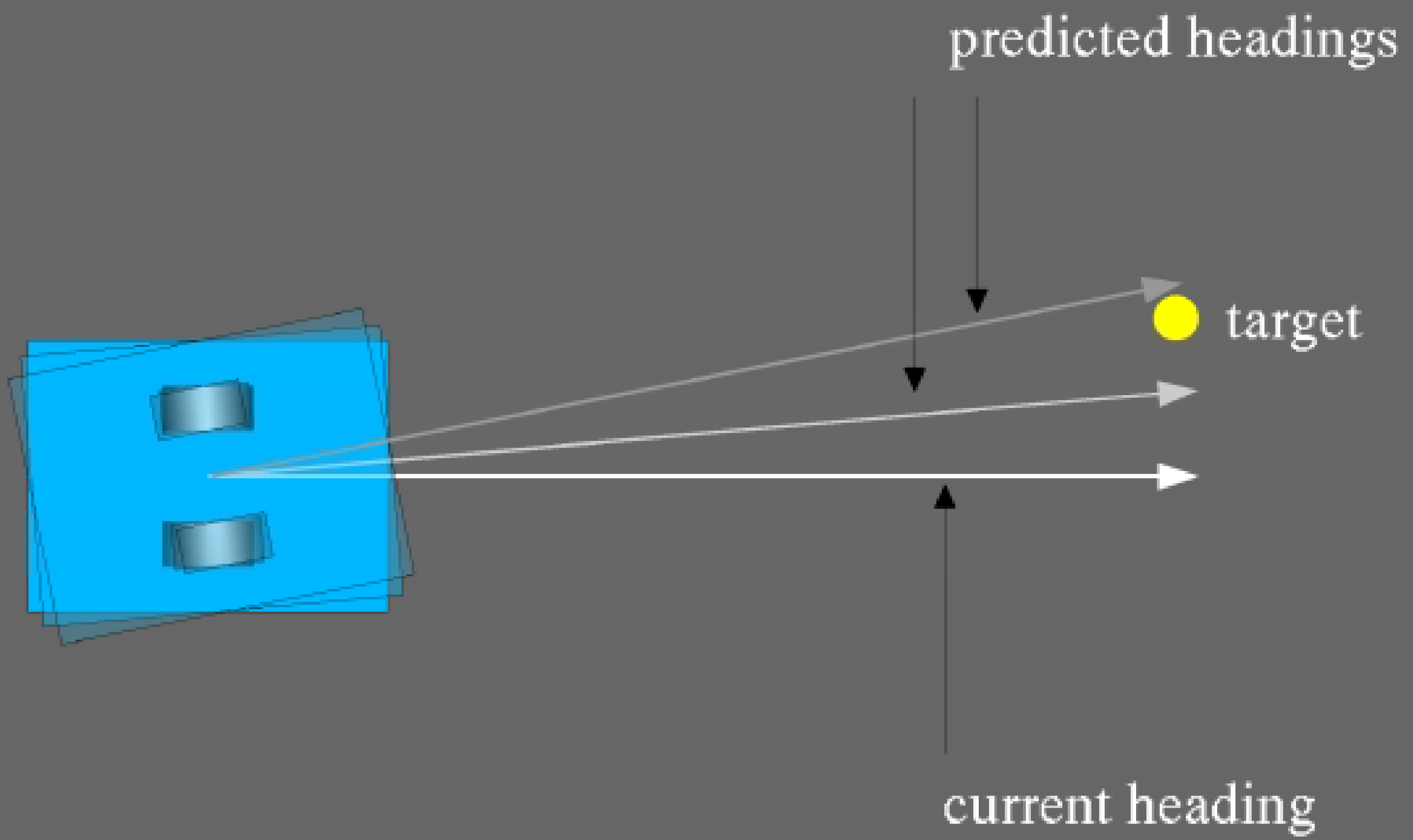
- Moment of inertia (angular momentum)
  - Must model and compensate for rotational velocity
    - Over−steering and heading oscillation
- More accurate collision modeling
  - Catching corners
    - Non−spherical body shapes
    - Friction
  - Collision avoidance more critical
  - Back up to unwedge

Simple pursuit behavior

Oversteer due to angular momentum



Pursuit with heading prediction



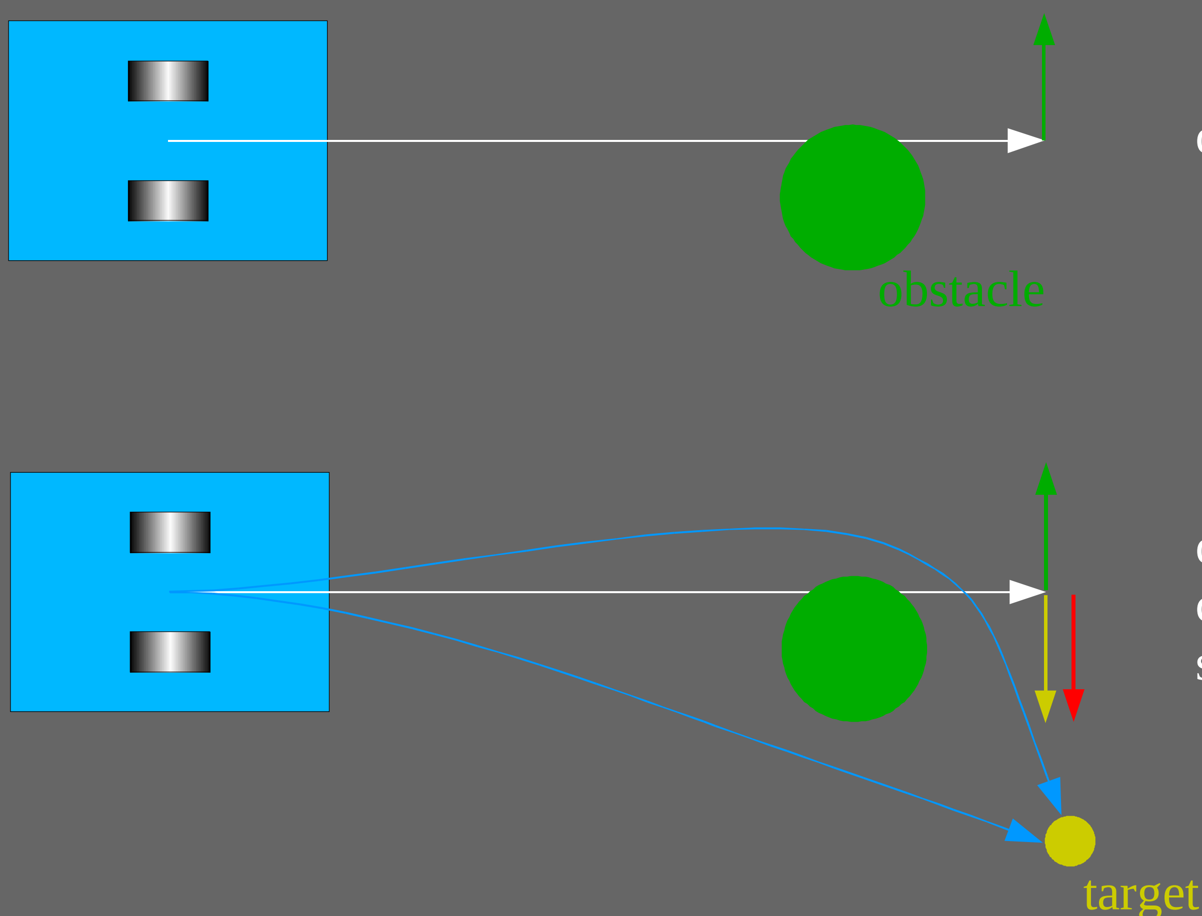
Diving into the robotsoccer code: top level robot control

```
void robotAutonomousControl (RobotState& robot)
{
    if (robot.goalie)
    {
        goalieBehavior (robot, ball);
    }
    else
    {
        if (robotMostForward (robot))
        {
            robotForwardBehavior (robot);
        }
        else
        {
            robotDefenseBehavior (robot);
        }
    }
}
```

Diving into the robotsoccer code: forward player robot control

```
void robotForwardBehavior (RobotState& robot)
{
    vec_3 steer;
    if (robotAvoidanceBehavior (robot)) return;
    if (robotCheckIfWedged (robot)) return;
    if (robotZoneContainsBall (robot))
    {
        if (robotGetBallOffWallBehavior (robot)) return;
        // if ball is closer to goal than we are...
        if (...)
            // try a shot on goal
        else
            // avoid ball while getting behind it
    }
    else
        // wait for ball
    robotBlendInNewWheelVelocities (steer, robot);
}
```

Decomposition versus big picture



obstacle avoidance

obstacle avoidance
combined with
seek target

Conclusions

- Autonomous characters
  - Definitions
  - Applications
- Steering behaviors
  - Toolkits and procedural composition
  - Evolutionary computation
  - Issues related to accurate physical models