Notes

- Assignment 2 is out
- Flocking references
  - Reynolds, “Flocks, Herds, and Schools…”, SIGGRAPH’87
  - Tu and Terzopoulos, “Artificial Fishes…”, SIGGRAPH’94

Flocking Behaviour

- Animating large groups of organisms is painful to do by scripting each member
  - Think flocks of birds, schools of fish, crowds of people, armies, …
  - Not just the complexity of each one’s motion, but making sure they all are consistent with each other
- Instead use a particle system approach
  - Each particle represents one member of the flock (“boid”)
  - Rules of motion are somewhat more complex
  - Ultimately best to replace “particle” by “agent”, and look at multi-agent systems (AI/Robotics)

State

- Need position x, velocity v
  - But we might ignore true notion of mass and forces (like gravity) for convenience
    - For enhanced realism, accurately model physical forces, but this makes it much harder to make the boids do what you want
    - See Tu and Terzopoulos, “Artificial Fishes…”, SIGGRAPH’94
- Also an orientation
  - For now, think of 3 Euler angles (e.g. pitch, heading, roll)
  - We will soon get more sophisticated
- Perhaps joint angles or more
  - To describe the external traits

Internal State

- Also want the state of the agent’s mind
  - Might be empty - purely reactive creatures
    - Reynolds’ paper
  - Simple quantities e.g. hunger
    - Tu & Terzopoulos
  - No limit on complexity

Perception

- One of the critical additions to the base particle system is perception
- Agents get a local view of the world, depending on their position and orientation (and…)
- Maybe abstracted to nearest neighbours' states plus collision geometry ahead
  - You probably don't want to do full renderings for each agent, then solve the computer vision problem
- Can also put animator controls in - e.g. agents know where goal position is

Behavioural Rules

- These could be anything from AI
- Let's look at simpler examples, from Reynolds first paper
  - Each rule produces a desired acceleration from the current state+perception
  - Collision Avoidance: steer away from others you are about to collide with
  - Velocity Matching: try to match velocity vector of nearby members
  - Flock Centering: move to middle (centroid) of nearby members
  - Collision Avoidance 2: steer away from collision geometry in scene (see it far ahead)
  - Migratory Urge: animator control

Motion Rules

- Need to somehow combine acceleration requests from subsystems into a coherent control
  - Averaging doesn't work well
  - Assigning priorities
  - Perhaps with a memory to avoid dithering
- Apply limits to final acceleration decision (e.g. a maximum, or true physics…)
- Integrate into velocity etc. maybe with limits here too (max velocity etc.)
- Secondary motion may be important
  - E.g. banking into a turn

More Details

- Let's look at behavioural rules in more detail
- Collision Avoidance - if another object dead ahead and too close, turn in some direction
  - A last ditch, very high priority, emergence procedure
- Velocity Matching
  - Accelerate current velocity towards average of neighbours' velocities, weighted by proximity
    - Inverse power laws (square, cube, …)
  - Usual mode of avoiding running into others - if you move with the flow, you won't collide - thus should be relatively high priority

More Details…

- Flock Centering
  - Accelerate so position ends up at average of neighbours' positions, weighted by proximity
  - Just like velocity matching, but for position
  - Keeps flock together, but only locally (flock may still separate)
  - Lower priority
- Avoiding scene geometry
  - Simple method: put repulsion forces around objects
    - Can get weird effects for dead-on collisions
  - Better: do ray intersection (current position + heading) with scene to find dangerous object. Find shortest direction to silhouette for avoidance.

Control

- The main problem with any system that automates lots for the animator
  - To make them do something, either intense amount of parameter tweaking and luck, or use manual override (which may violate other aspects of behaviour, requiring more overrides…)
- Can add additional acceleration requests from animation script ("migratory urges", goal directions or positions)
- Follow-the-leader approach - script one, let the others try to follow
- Add virtual geometry to guide agents

Optimized Control

- Common approach to many controlled motion problems: phrase it as an optimization problem
  - Find solution which satisfies control constraints (e.g. center of flock at position x at time t) with least violation of behavioural rules
  - Or find solution which satisfies behavioural constraints and comes closest to user's objective (note: need some variability, at least in initial conditions, probably also in "random" behaviour)
- Example reference:
  - Anderson, McDaniel, and Chenney, "Constrained Animation of Flocks", Symposium on Computer Animation 2003

Rigid Bodies

- We know how to do physics with simple point masses
- What about larger objects?
  - We talked a little about mass-spring models of deformable objects
- What about larger and basically rigid objects?
  - Most of the world around us
  - NOT a good idea to simply model with stiff springs

Rigid Bodies

- I'll introduce them from a particle perspective
  - Easy to get lost in abstract notions
  - Particles are fundamental
- Discretize an object into small point masses
  - x_i, v_i, m_i
- Assume object doesn't change shape (doesn't deform)
  - What does that mean for the motion of the particles? How do we describe it, solve for it?

World Space vs. Object Space

- World space: where the particles actually are now
  - This is where we will look at x, v, and almost every other quantity
- Object space: imaginary "reference" place for the particles
  - Label the object space position p_i
  - Does not change as the object moves - things we compute in object space stay constant
  - We can define it arbitrarily

Mapping

- The map from p_i to $x_i(t)$ cannot change the shape
  - The distance between any two particles never changes
  - Thus map has to be $x_i(t)=R(t)p_i+X(t)$
  - R(t) is an orthogonal 3x3 matrix: $RR^T=\delta$
    - The orientation (rotation) of the object
  - X(t) is a vector
    - The "location" of the object