Notes

- EA speaker series talk tomorrow:
  - Jessica Hodgins, animating humans
- Overdamped dt error
- Gravity force vs. acceleration
- mud_renderman is on the web
  - Still some bugs, but nothing too serious
- Globular Dynamics paper to read
  - Reference on the website
  - Note: related to assignment 2, but not quite the same
  - Also misses out a lot of the important details

Collision Detection

- For efficiency, in all but the simplest cases, just store geometry in object space
- Different objects have different object spaces
- We’ll need to convert between spaces a lot
- One possibility:
  - Detect collisions in world space - but need to recompute both acceleration structures, and need to transform both sets of geometry
- Often a better idea:
  - Do collision tests in object 1’s object space
  - Precompute composition of map from object 2’s object space to world space, then from world space to object 1’s object space
  - Use rotation matrix R, not quaternion
  - Only need to recompute and transform stuff for object 2

Intersecting triangle meshes

- Core operation: check if a pair of triangles intersect
  - Boils down to determinant tests
- (Also check if one vertex of object 1 is inside object 2, and vice versa)
- Naïve way:
  - Loop over faces of one, check each for intersection (accelerate each check)
  - Repeat for the faces of the other
- Better way:
  - Use acceleration structure to prune in BOTH objects simultaneously
  - Don’t check any distant faces or edges

BV Hierarchy

- Start with an empty stack
  - Holds pairs: one BV node from object 1, one BV node from object 2
- Put the root BV’s on the stack
- While stack isn’t empty
  - Remove a pair of BV’s from top, check if they intersect
    - If so, and they are at the base level, do the face/edge intersection tests
    - If so, and not at the base level, but all pairs of children on the stack
Grid

- Find all grid cells with primitives from both objects
  - Probably want separate lists for each object, or a count of # primitives from each object, to make this O(1) per grid cell
- Test those primitives for intersection

Repulsion Forces

- Repulsion forces work the same way as before
  - But now need to compute the net torque they cause as well as the net force
    \[ \tau = (x - X) \times F \]
- Simplest approach:
  - Sample rigid body geometry with points (e.g. vertices of surface mesh)
  - Check which are close to or inside another object
  - Apply force to deepest/closest, proportional to distance, in normal direction
- More complex:
  - Figure out overlap volume...
  - Not clear it's really worth it

True collisions

- Identify the points on the two objects that collided
  - E.g. deepest point of object 1 inside object 2, corresponding point of object 2
- Check that relative velocity is approaching - not already separating
  - If separating, look for another pair, or give up and apply repulsions
- We will take same Newton condition on the collision:
  - Normal relative velocity component gets reflected, reduced by \( \epsilon \)
- Friction done in a similar way

Frictionless impulse

- Object velocities at point:
  - \( v_i = \omega_i \times (x - X_i) + V_i \)
- Relative velocity \( v = v_1 - v_2 \)
  - Normal component \( v_n = v \cdot n \)
- Want post-collision relative normal velocity to be \( v_n^{\text{after}} = -\epsilon v_n \)
- Apply an impulse \( j = j_n n \) in the normal direction to achieve this
  \[
  \begin{align*}
  V_i^{\text{after}} &= V_i + M_i^{-1} j \\
  L_i^{\text{after}} &= L_i + (x - X_i) \times j \\
  \omega_i^{\text{after}} &= \omega_i + I_i(t)^{-1} (x - X_i) \times j
  \end{align*}
  \]