You will pass the hand-in with a reasonable solution on only two of the following problems, but I strongly recommend that you do them all at some point.

The problems are listed in increasing order of expected difficulty.

If you would like to exchange one of these problems for a functional programming problem, then let me know and I will come up with something.

Problem 1. (The cannon)

Cannon balls are fired from the lower left corner of the window. They are fired at constant speed (chosen by you) but varying angle. The balls are experiencing accelleration due to gravity, but no other forces. They are fired at 1 ball per second.

Implement an algorithm which finds the angle such that the ball hits the mouse cursor (if possible).

Experiment with speed and gravity so that the cannon can hit targets in (very) roughly 2/3 of the screen.

Problem 2. (The fountain)

Implement a particle system. The particles are spawned at the middle of the screen and are shooting upwards at constant speed, but random angle (-PI..0). Implement the particles so that:

a) The particles experience gravity only.

b) The particles experience gravity and wind coming in from the right.

If you have different ideas for a more involved particle system (smoke, fluids, etc) then you may do that instead, but make sure to specify the problem that you solve in the discussion.

Problem 4. (The robot)

Implement inverse kinematics using the CCD-algorithm.

The robot should have at least 3 limbs and be located in the middle of the screen. The robot should adjust its angles and move its effector as if trying to grab the mouse cursor.

There is a possible better (but a little more involved) algorithm which could be called the Jacobian method. If you are comfortable with mathematics and would like to be challenged, then let me know and I will find a reasonable online resource for you.

Problem 3. (Colliding billiard balls)

Imagine a 2d view of a billiard table (from above) filled with billiard balls. You are to set a starting velocity to one of the billiard balls and simulate what happens when the balls collide. The balls collide with each other, but also with the boundary of the screen.

You can assume that all collisions are elastic, but that the balls have varying mass and radius.

This problem is probably the easiest if:

1. you use fixed framerate (e.g 60 frames per second).

2. Keep velocity constant between collisitions and integrate position only (i.e. ignore accelleration). You can still simulate friction by damping velocities if you want, for example by interpolating to zero (i.e. vel *= (1-c*dt)*vel, where c is a small constant chosen by you).

However, I give you a hint in case you are using varying framerate: If you have detected collision, then move the balls out of a state of collision before letting them go. That is, move them away from each other in the direction of the normal of the collision. This will prevent them getting stuck to each other (or to the walls). Btw. this step can be done almost automatically if you use the Verlet integrator (not part of the curriculum).