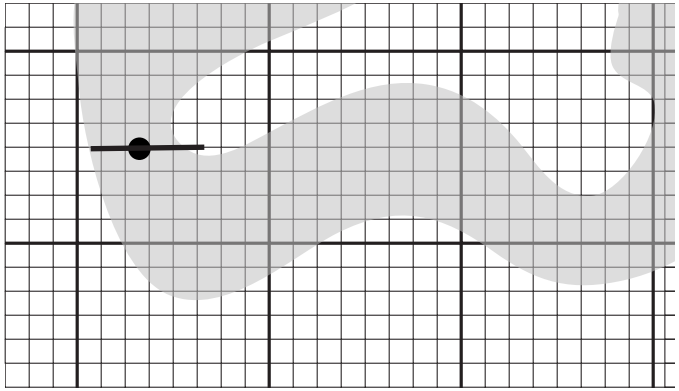
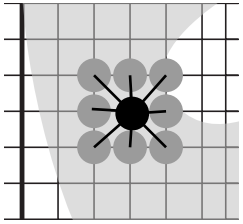
Motion!

How do things move and accelerate? How can we model this inside of a program like scratch?

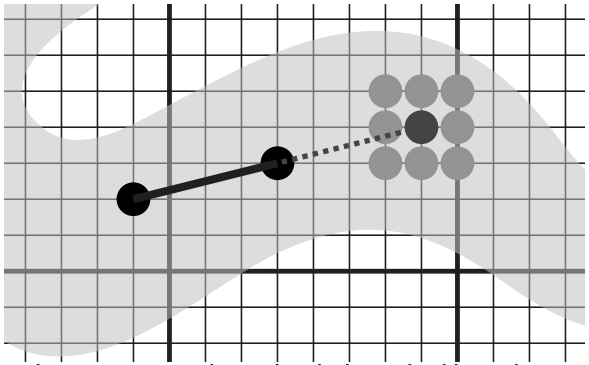
To begin, let's play a simple game, racing around a track drawn on some graph paper!



You've got some paper, so draw yourself a track and find a challenger. At first your "cars" are stopped at the start line and you can move one square in any direction. Of course if you run off the track, you crash!



From now on, it works like this. If on the last turn you moved some amount, you can move the same amount the next time, plus one additional square (if you so choose)
So for example:



If your last move was along the dark marked line, the next move has to be the same, with the option of moving one more square as shown. Of course this affects the next move too! Can this player avoid crashing?

The point of all that is that we can describe motion very nicely. At each moment, we have a position and a velocity. The velocity at each moment is the change in our position at the next.

But we can steer too! Our velocity isn't fixed from one moment to the next. The change in our velocity is our acceleration, and this is what we are choosing from moment to moment as we play the game.

So we have the following variables:

x and y (for our position)
dx and dy (for the change in our position, the velocity)

Initially we set these to
(x,y) = wherever we want to start
(dx,dy)=0

Then we repeat forever:

Ask the player what should the acceleration be? One squre N, S, E, W or NW, NE, SW, SE? In a computer game, we could use the arrow keys to learn this, so let's do it that way here:

If the left arrow key is pressed, set dx to be dx + 1
If the right arrow key is pressed, set dx to be dx – 1
If the up arrow key is pressed, set dy to be dy + 1
If the down arrow key is pressed, set dy to be dy – 1

Now we change our position:
Set x to be x + dx
Set y to be y + dy

And repeat.

This same idea is incredibly useful for modeling all kinds of motion.

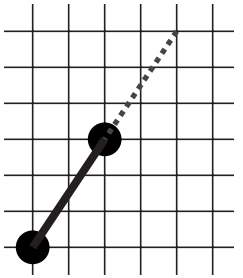
For example, how can we model a bouncing ball?

It's really pretty easy!

A bouncing ball is always being pulled down. It's vertical acceleration is always towards the floor.

Try this experiment: draw a floor and some walls, and send a ball flying through the air!

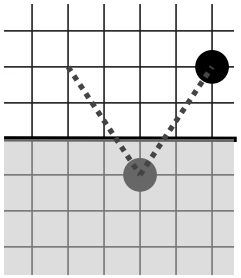
This time, it works a little differently.



Whatever your velocity was before, you drop one square lower at the next step.

Of course, if you hit the floor, or the walls, you bounce!

To figure out how your ball moves when you bounce, go ahead and move it past the wall or floor, then reflect it and the vector back. For example:



and then continue with the bounced position and velocity!

We can model this pretty simply. Let's just suppose there is a floor at y = 0. We'll drop the ball by just one unit per move (but you can experiment!)

Initially we set the position (x,y) and the velocity (dx,dy) to whatever we want.

Then we repeat forever:

(dx isn't changing)
set x to be x + dx

if y>0, then set dy to be dy – 1
otherwise, set dy to be – dy

set y to be y + dy

Does it work the way it is supposed to?