DRAFT

Unit 4 - Activity 3a

Air Hockey Table – Pyret Simulation

You've observed the motion of a puck on a physical air hockey table. But, changing its motion in specific, repeatable ways is difficult. So, we'll use Pyret to help us out.

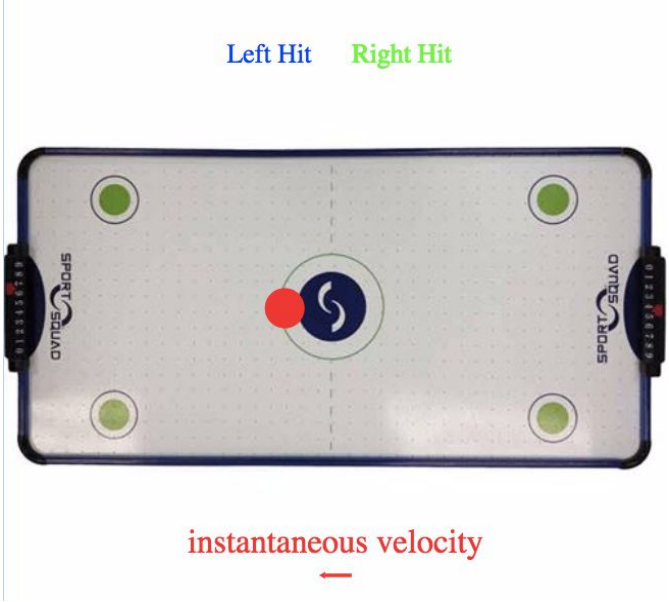
Below is a snapshot of our virtual air hockey table which is 1000 pixels wide. It is working perfectly and is a completely frictionless surface. You can find the code for the simulation at **https://goo.gl/TyG5jd**.

Press RUN. Once the simulation loads, your goal is to use the simulation to observe changes in the puck's motion each time you push it. To push the puck to the right, press the <→> key; to push the puck to the left, press the <← > key.

Now let's look at some of the following situations:

1) How do you make the puck speed up? Why?

2) How do you make the puck slow down? Why?



3) How do you make the puck move at a constant velocity?

4) Is it possible to make the puck stop? How did you (try to) do it?

Unit 4 - Activity 3b

Broken Air Hockey Table – Pyret Simulation

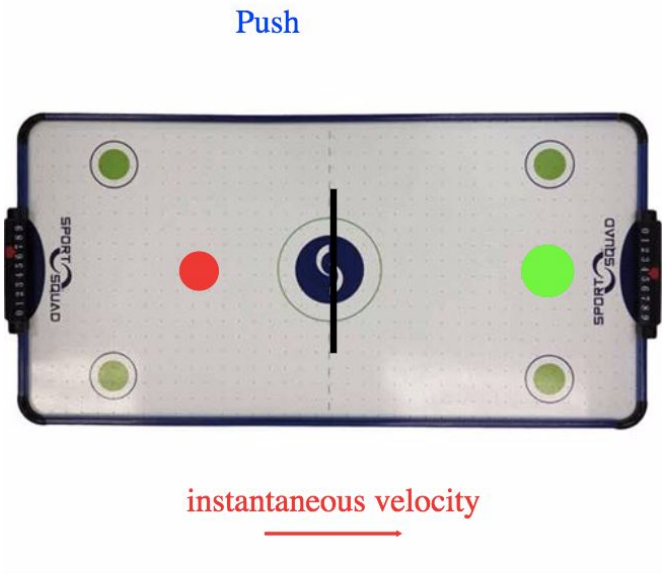
Below is the same air hockey table. The left side is working perfectly fine and is *frictionless*. You need to break the table so that the right side has broken air jets and hence there is *friction*. Your goal is to write a function, `Friction-Force`, that applies a constant force called `PUSH` to the red circle so that it stops inside the green circle. The contract is:

```
Friction-Force ::(Number) -> Number
```

Go to the code for the simulation at **https://goo.gl/QaWY1R**. When the simulation begins, the red circle has an initial velocity of 100 to the right. Again, the track width is 1000 pixels. You'll also notice an empty element:

```
direction =
```

You need to specify the direction of your frictional force with one of two strings. For left you must write `"left"`.
For right you must write `"right"`.



Unit 4 - Activity 3c

Broken Air Hockey Table – Pyret Simulation

Once again we see the broken air hockey table. Recall, the left side is working fine (*frictionless*), and the right side has friction. In the previous simulation, you provided the friction that slowed the puck down. Go to the code for the simulation at **https://goo.gl/ZDxB8a**.

This time, the friction is already present (you do not have to code it). Your goal is to write a conditional function called `My-Force` that keeps the puck moving the entire time as if the air hockey table was working fine (meaning that the air jets are on and the puck is moving at constant velocity). The contract for your function is:

```
My-Force :: (Number) -> Number
```

Even though the contract for My-Force is the same as in Activity 3b, your conditional function may be different. Nor should you assume that the magnitude of the friction force is exactly the same as in Activity 3b.

Please note that there is a blue 'goalie' paddle now blocking the path of the puck. The paddle is designed to be 'lifted' out of the way, if you:

1) Correctly write the conditions.
2) Provide the correct amount of force to balance the friction conditions.

If not, the blue 'goalie' will block your path OR crash down on top of the puck, stopping your puck.