Lab 3: TIC-80 Physics & Collisions

~~DUE September 27th at 11:59 PM~~ **October 11th at 11:59 PM**

In this lab we will practice with TIC-80 some more. We'll build upon the tutorial on driving found in **part two**, but **part one** is also useful. Things I suggest you study about the game before diving in:

1. How a lua table is used as an object (struct).
2. How animation is done using a mixture of different sprites and procedural rotation.
3. A rudimentary physics engine that simulates velocity, acceleration, and friction using vectors.

After you have acquainted yourself with the code & tutorial, add a few features to make it a two-player game. You don't have to follow the suggested changes to the letter, if you'd rather be a little creative, but your improvements must include:

1. allowing at least two players;
2. collision detection between the players, and the environment using the `map`;
3. add some point tally or win condition.

1   Suggested Changes

0. Use the `strict` mode to avoid accessing nonexistent global variables.
1. Add a second car object (I suggest adding a `Car` class).
2. Add a function or method to move the car:

```lua
function updateCar(car, amt, go)
    -- turn car by amt, and if go is true press the gas pedal (accelerate)
end
```

```lua
function Car:update(amt, go)
    -- turn car by amt, and if go is true press the gas pedal (accelerate)
end
```

3. Add support for driving the second car with another set of controls/gamepad.
4. Add the following collision function (which treats the 8x8 cars as circles), or are bounding boxes better?

```lua
function collide (c1, c2):
    -- return true if the two cars (c1,c2) are overlapping, false otw
    d = (c1.x - c2.x)^2 + (c1.y - c2.y)^2
    return d < 64
end
```

5. Add a field to the cars' tables called `spinning` that is initially `false`.

6. If cars collides, make the slower (or faster) car stop responding to the controls and just spin out.

7. Add a point system and the ability for the cars to restart after a spin-out.

---

8. Add a map and have the cars react to different levels of friction depending on which square (road, dirt, grass) the car is over. Create tiles for the different kinds of terrain and put them in a map and modify `updateCar`. **NOTE:** You will probably have to divide the car's x- and y-coordinates by 8 before passing them to `mget()` to convert screen coordinates to map coordinates.

9. Add walls which prevent the cars from passing through and stop them.
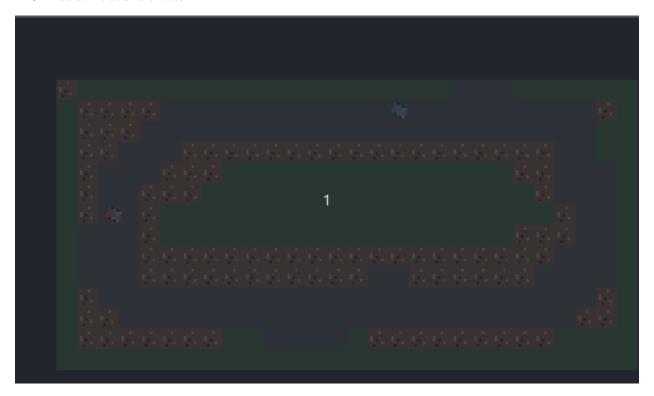
10. Add some sound effects.

Figure 1: Screencast

2   Deliverables

1. Commit the TIC-80 lua file to the repo (`save driving_model.lua`).
2. Export your game as an HTML project and include the files (html/js, not the zip) in the repo.