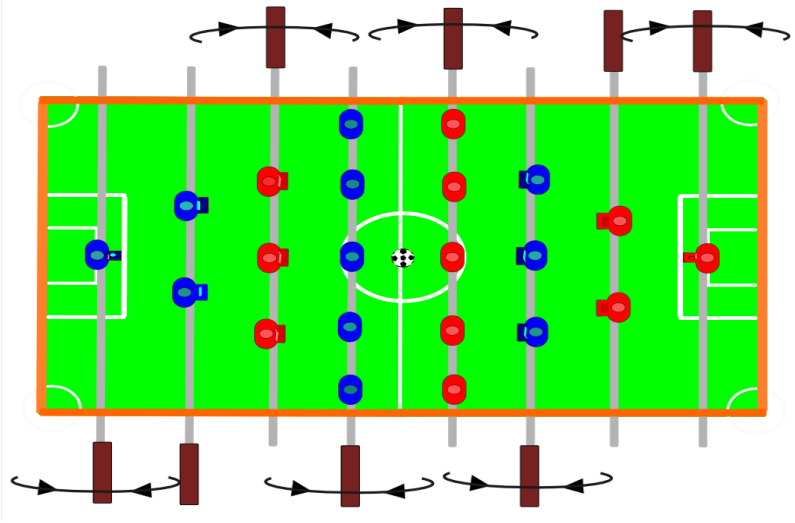
Foosball: CS296 Project, Group 27

Kiran Y - 120050049,
120050049@iitb.ac.in,
Mihir -12D020007
mihirk.1994@iitb.ac.in
Harinandan Teja-120050066
120050066@iitb.ac.in

April 8, 2014

1 Introduction

This report explains in detail the project we created- a foosball table- in Box2d. We describe in brief the bodies,fixtures and constraints we used to create this simulation.
Here is the initial model of our simulation:



2 Design

2.1 Overall

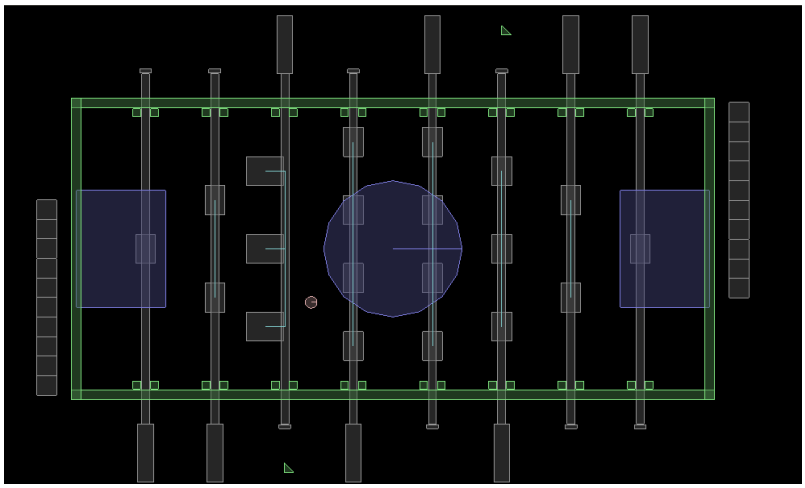
1. We simulated the game of foosball using a bounding table, rods and rigid bodies representing players attached to each rod. Whenever the ball enters one of the goals situated at either end, a score variable is updated and this is shown by moving one of the relevant score blocks present on either side of the table. The first team to ten goals is the winner.

2. The simulation consists of a ball which moves around on the frictionless table colliding with everything it comes in contact with.

3. There are (transparent) rods at relevant locations on the table, constrained to move upto a certain limit in the upward or downward direction. The players attached to each rod can be moved towards either the left or right upto a certain limit.

The user also has the option to turn the rod enough to lift all its players off the table.We used the `b2PrismaticJoint` to simulate motion of these players.

4. The simulation is heavily dependent on user input. We implemented this overloading the `keyboard()` function of the `base_sim_t` class of Box2D.



2.2 Changes from original plan

1. It was a challenge for us to implement the constraints required by the nature of the game, and we think that usage of prismatic joints and using the `filter` attributes of Box2D objects to cater to all aspects of motion of the players has made our design more efficient. We also feel that using static objects to constrain the motion of the rods (as explained later) was a simple yet effective feature.

2. We tried to use prismatic joints to restrict the motion of the rods as well. However, this made the motion considerably inefficient and we finally decided to not do so.

3 Parts of the simulation

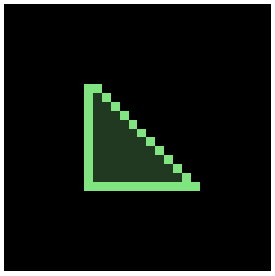
3.1 Table

The entire game is played within the confines of the table, which serves to restrict the motion of the ball and rods. This was implemented using four box shapes.

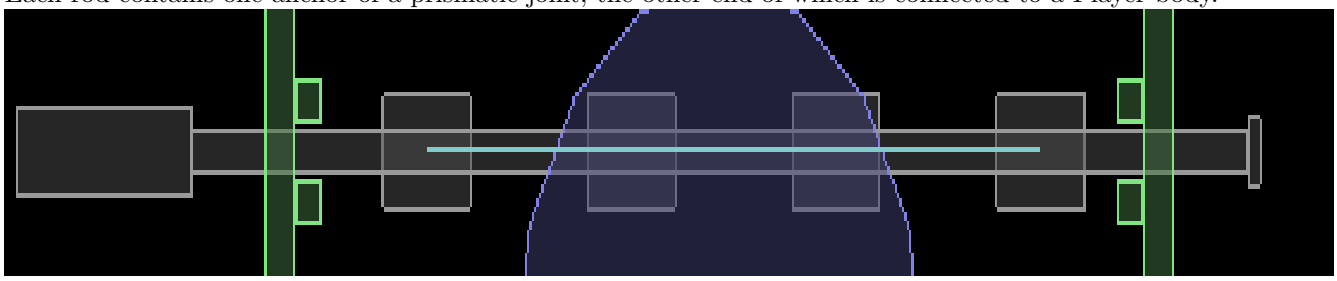
3.2 Rods

1. Rods are thin dynamic `Box` objects located at appropriate locations on the table. Two pairs of small static objects are located on either side of the rod at the point where it enters and leaves the table to constrain its motion in the y direction.

2. At any given point, only one rod per team is active. The user can change the active rod by giving a keyboard input.This is indicated by a pointer.



3. Each rod contains one anchor of a prismatic joint, the other end of which is connected to a Player body.

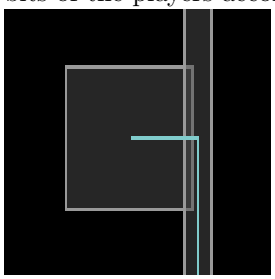


3.3 Player

1. A player is a small,rigid and dynamic body which is attached to a rod via a prismatic joint. All players on the current active rod can be move right or left(upto a certain limit) by a keyboard input.Such a rotation has greater impact if it happens from a start position far away from the final position of the player.

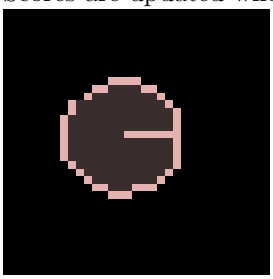
2. The user can also lift the players of a rod off the table (meaning that the ball can pass underneath them) by pressing the right motion key while the players are at the maximum right position, and similarly for the left. This can be viewed as a change in the shape(fixture) of the players.

3. These are implemented by setting appropriate attributes on the prismatic joint and by setting the fixtures and filter bits of the players accordingly.



3.4 Ball

The ball is a `b2Circle` shaped `bullet` object which keeps on moving around and being struck by the players/table walls. Scores are updated when the ball enters either of the goals.



3.5 Score boxes

1. These are ten boxes located on either side of the table that represent the score.
2. A `dominos_t::check()` function is continuously called by the `base_sim_t::step()` function that checks whether a goal has been scored. If so, the appropriate box is pushed towards the appropriate side.
3. If the score of any team reaches ten or more, the game is restarted and the boxes are moved to their appropriate location.

4 Conclusion

We have hence created a self contained two player foosball game using keyboard inputs and Box2D simulation
We have used the techniques learnt in earlier labs to make the code efficient and document it.

References

[1] Our batchmate shivam garg.

[2] `https://www.iforce2d.net/b2dtut/`.

[3] Jeff Atwood and Joel Spolsky. `https://www.stackoverflow.com`. 2009.