

MCS 320 Project Two: A simple model of billiards due Monday 29 March 2004, at 2PM.

The goal of this project is to gain experience with functions and plots in Maple to define a simple model of billiards. Despite the simplicity of our model, we will see trajectories normally not seen on a pool table. This document is made from a Maple worksheet. A good way to start this project is to download the worksheet from the course web site.

1. Defining and visualizing trajectories

Imagine a square pool table. The length of each side is one.

```
[> f0 := plottools[line]([0,0],[1,0],color=green,thickness=3):
[> f1 := plottools[line]([0,0],[0,1],color=green,thickness=3):
[> f2 := plottools[line]([1,0],[1,1],color=green,thickness=3):
[> f3 := plottools[line]([0,1],[1,1],color=green,thickness=3):
[> pool := f0,f1,f2,f3: # this stores the plots for later rendering
```

A billiard ball moving on the pool table starts at some position (x_0, y_0) and moves with constant speed (v_x, v_y) . We take random numbers for these constants.

```
[> x0,y0,vx,vy := stats[random,uniform[0,1]](4);
```

Then the ball moves along the line

```
[> xt := t -> x0 + vx*t: yt := t -> y0 + vy*t:
```

To compute the position of the ball on the pool table, we need the following functions:

```
[> r_mod_one := x -> [trunc(x),x-trunc(x)]:
[> position := nr -> piecewise(nr[1] mod 2 = 0,nr[2],1-nr[2]):
[> xp := position@r_mod_one@xt: yp := position@r_mod_one@yt:
```

Plotting the trajectory is now straightforward:

```
[> path := plot([xp,yp,0..10]):
[> plots[display](pool,path,scaling=constrained,axes=None);
```

And we can make an animation:

```
[> N := 20: # number of frames
> t := 0.0: dt := 0.05:
> path := []: # list of plots
> for i from 1 to N do
>   pt := [xp(t),yp(t)]:
>   ball := plottools[disk](pt,0.01,color=red):
>   path := [op(path),plots[display](pool,ball,scaling=constrained,axes=None)]:
>   t := t + dt:
[> end do:
```

To launch the animation, execute the following command, click on the picture and then use the toolbar.

```
[> plots[display](path,insequence=true,scaling=constrained);
```

Square pool tables are very rare. Therefore, the first assignment is to extend this model to any rectangular pool table.

Assignment One: Modify the model to work with any rectangular pool table of dimensions a and b .

Create a function `pool` which takes as arguments the dimensions a and b , and which returns the plot of a rectangular pool table with sides of length a and b . For example, `pool(3,2)` returns the plot of a 3-by-2 rectangular pool table. Give the definition of this function and show the plot command (with its output) to test the function.

Use the functions `r_mod_one` and `position` from above to define two indexed functions `rxp` and `ryp` called as `rxp[a](t)` and `ryp[b](t)`, which give the position of a billiard ball at time t on a pool table of dimensions a and b . As before, use the composition with the functions `xt` and `yt` to define `rxp` and `ryp`.

2. Computing special trajectories

Suppose we start at (x_0, y_0) and wish to reach (x_1, y_1) hitting the vertical sides n times and the horizontal sides m times.

```
[> x0,y0,x1,y1 := stats[random,uniform[0,1]](4);
[> start := plottools[disk]([x0,y0],0.01,color=blue):
[> target := plottools[disk]([x1,y1],0.01,color=red):
[> n := 4: m := 10:
[> nx1 := n+x1: my1 := m+y1:
[> vx := nx1 - x0: vy := my1 - y0:
[> nxt := t -> x0 + vx*t: myt := t -> y0 + vy*t:
[> nxp := position@r_mod_one@nxt: myp := position@r_mod_one@myt:
```

Notice that we reach the target at $t = 1$.

```
[> path := plot([nxp,myy,0..1],numpoints=1000):
[> plots[display](pool,start,target,path,scaling=constrained,axes=None);
```

This works only when n and m are both even. The purpose of the second assignment is to extend the formulas so that n and m can be any natural number.

Assignment Two: The computation of the special trajectories only works when n and m are both even. Extend the model so that n and m can be any natural number.

3. Slowing down the speed of the trajectories

In this current model, the ball moves at constant speed, without ever slowing down. This is not very realistic.

Instead of the $v_x \cdot t$ and $v_y \cdot t$ terms in the definitions of x_t and y_t , we better take an exponentially decaying function.

Assignment Three: Change the function x_t and y_t taking v_x and v_y to be 10, so that at $t = 100$, the speed drops below 0.01.

Create an animation with sufficiently many frames to see the effect.

4. The deadline is Monday 29 March 2004, at 2PM.

The solutions to the project will be collected at the beginning of our class meeting on Monday 29 March at 2PM. If you cannot come to class that day, then you must arrange to hand in your solution before the deadline. Otherwise, your solution will be discounted with 10 points if it is turned in on the same day before 5PM, and will no longer be accepted afterwards.

The solution consists of the print out of one single worksheet, organized with sections according to the two assignments. Also write proper documentation for the calculations.

To avoid any misunderstanding, this project must be solved *individually*. Under no circumstances is it allowed to copy or to collaborate. Regardless of who copied from whom, all caught in the act of plagiarism will be penalized.

If you have questions, comments, or difficulties, feel free to come to my office for help.