Report:

I think that the assignment is easy. Finding how opengl works on either mac OS or Linux are easy. But I still spend the most time on finding how to use opengl on python. I haven’t tried windows. But seems like the biggest problem for other students are finding how opengl works on windows machines.

Once you have opengl working, everything will be very easy. All you need to do is translate line by line. Then there will be no problem at all. You can even copy and paste those opengl parts(at least for python).

The GLUTDISPLAYFUNCTION and GLUTIDLEFUNCTION are the most important. It controls what is on the popped up window.

First step: get the window pop up. The main function does this. I did this first.

Then get the display function. The display function would draw all the walls.

Then get the idle function. You would clearly notice that when you move the window, it becomes less smooth. That is because the idle function will be called constantly.

The display and idle function all calls render scene function. I divided the render scene function into two parts. One does not include update, one does.

The render scene function will have call functions from other class. This is the time I implemented those classes.

I can’t think of any difficulties. If you do it step by step everything will work.

Detecting the collision is not hard. When the distance between two balls are smaller than radius \* 2, there will be collision.

But I can’t get the colliding balls to do things.

My initial function: resolve\_ball\_to\_ball(self, blist : list)

When balls collide, I need to change the velocity of two balls. However, after reading the outputs, I feel like I can only change the value of self. I can’t change the velocity of blist[i]. Eventually, 2 ball’s velocity becomes the same. They stick together.

It seems like two balls are doing different things but in fact it is not. For example. Ball 1 collides ball 2.

Ball 1 calls the function, so ball 1’s velocity would change. (I want it to change ball 2’s velocity in the same function but It doesn’t). The reason ball 2 moves too is because ball 2 will call the function after. Ball 2 will be come the self.

I spend a lot of time trying to figure this. However the only solution I can think of is return values. I suspect the reason is that python doesn’t have passing by reference, so passing a list would not change the value in the list. I tried to let the resolve function return a tuple. It contains ball 1’s index in the list, the value it should change to, ball 2’s index, the value it should change to. After getting the tuple, I tried to change the values in the update function in main function.

I failed. I don’t know the reason for now.

Then on Friday I asked you. You said passing a ball. I tried that, It works as I think by looking at the balls and output. Unfortunately, I can’t really describe what is going on in my mind when I look at the outputs. It is really hard to explain.

Then the question would be what those 2 balls would do when they collide. I didn’t make that part working. My approach is easy. They switch velocity. But it doesn’t work when you shoot balls to walls.

Problems:

When the ball finally stays steady, it should stay the same, which means that the velocity would be 0, 0, 0. But for some reason sometimes it moves up and down at a very small degree. I can’t solve the problem because the reason must be in Ball.update but it seems alright.

This problem causes my approach works badly. When 2 balls collide, one should pass a 0, 0, 0 to the other. However, since it moves up and down, it won’t pass 0, 0, 0. Even if I say if velocity’s norm < 1 velocity = 0. It doesn’t solve the moving up and down problem.

Another problem would be my approach. I tried to find some on line help but it gets complex. I get what you said. You said adding 2 velocity together and divide them. It is the conservation of momentum. Since the mass are the same, we can eliminate the mass. So the velocity’s sum would be the same before and after. But it cause a problem of how to divide them. If divide them by half, what the direction would be. I think it depends on their original direction. But the direction is 3d.

Then I need to handle acceleration. Should it remain the same? I don’t think so. It becomes complex.

The normal case is when one ball stays steady and one ball hits the ball from above at some degree. I think it is hard to solve.

There are many problems. So I didn’t finish the ball collision part. But I believe that if I were given more time I would finish that. The problem would be how to handle the physics. And it wouldn’t be a lot of changes. I hope we are given another week for the ball collision.

To run, add pyopengl using pip3 and run python3 t.py. The original translation is in master’s branch. The collision that I got so far is in ball\_collision\_modify branch.