## Part 1: Bouncing Ball on Walls

1. Create a new program and make a sphere object:

```
ball=sphere(pos=vector(-5,0,0), radius=0.5, color=color.red)
```

2. Create a "wall" by making a box object:

```
wallR = box(pos=vector(6, 0, 0), size=vector(0.2, 4, 4), color=color.green)
```

3. Animate the ball:
   You need to display "snapshots" of the position of the ball at successive times as it moves across the screen. For a short time interval we will use "dt".

```
dt=0.5
```

4. Define ball's velocity as a vector (right now we've specified that the ball moves only in the x-direction, the y and z are 0):

```
ball.velocity=vector(.2,0,0)
```

5. Run your program. Nothing happens yet! We need to update the ball's position.

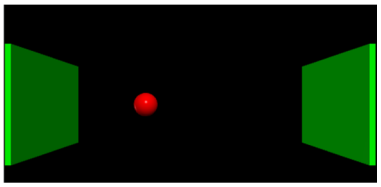
6. Add this loop:

```
while(1==1):
  rate(100)
  ball.pos=ball.pos + ball.velocity*dt
```

7. Now, run your program and see what happens.
8. You need to check for the collision with the wall and go backwards when that happens.

```
if ball.pos.x > wallR.pos.x:
  ball.velocity.x=-ball.velocity.x
```

9. Now run your program.
10. Create a new wall on the left.
11. Make the ball bounce on the left side too.



## Now turn to Part 2…

## Part 2: Ball in a Box

1. Add a top and bottom wall:
    a. Use a different color to make the 3D more visible (perhaps blue)
    b. Make sure they touch the horizontal walls.
    c. Make the ball bounce off the top and bottom walls.
    d. Modify velocity for the y position
    e. Use an appropriate “if” statement in the “while” loop.
2. Add a back wall in yet another color (perhaps magenta). And an “invisible front wall. That is, for the front wall, don’t create a wall, just include an “if” statement to prevent the ball from coming through the front. Give your ball components of velocity in the z-direction as well so it bounces off all 6 walls!