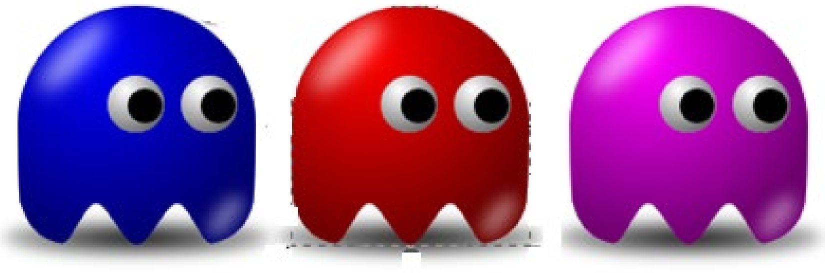
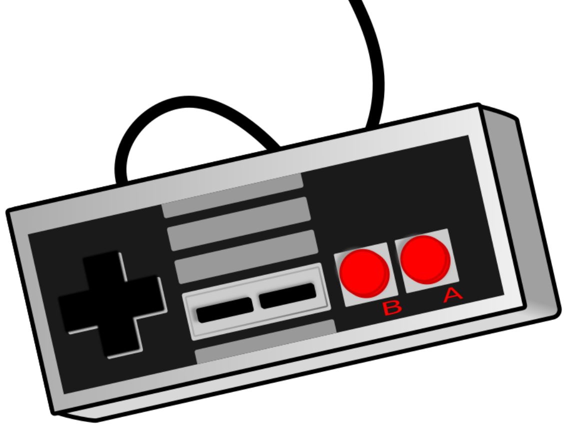




Lab_11

Greenfoot Simulation (Part 4)

420-141-VA - GAME PROGRAMMING 1 - VANIER COLLEGE

Outline

- Step 1: Greenfoot Simulation (Part4) Instructions
- Step 2: Accept the Assignment from Github Classroom to Remote Repository
- Step 3: Clone the Project from Remote Repository to Local Repository
- Step 4: Open Greenfoot Simulation from Local Repository
- Step 5: Detect collision between cannon balls in MyWorld
- Step 6: Collision response for 2 cannon balls in MyWorld
- Step 7: Commit your changes and push to github (1)
- Step 8: Keep the cannon balls within the window
- Step 9: Commit your changes and push to github (2)
- Step 10: Submission through LEA OMNIVOX

Step 4: Open Greenfoot Simulation from Local Repository

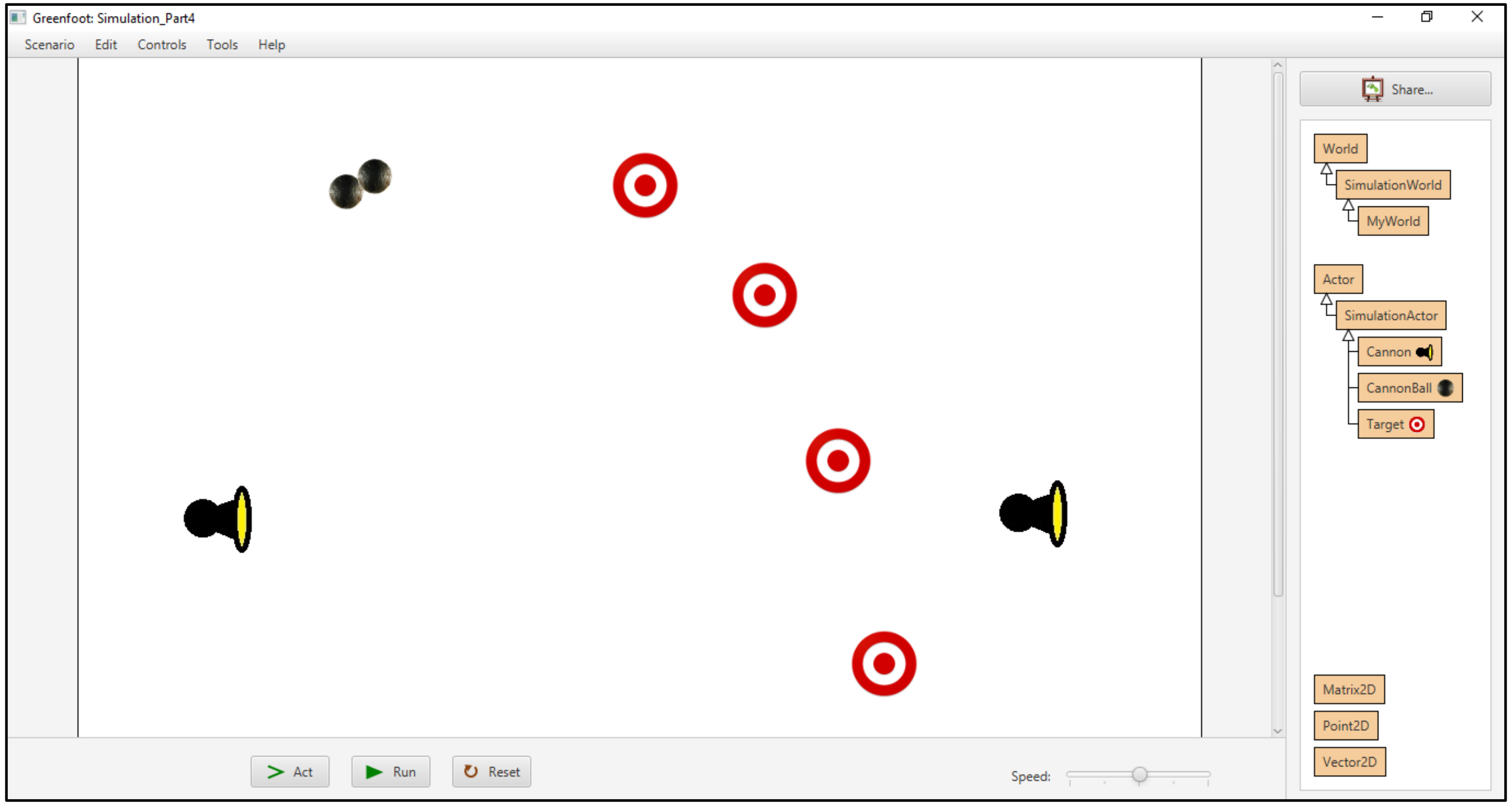
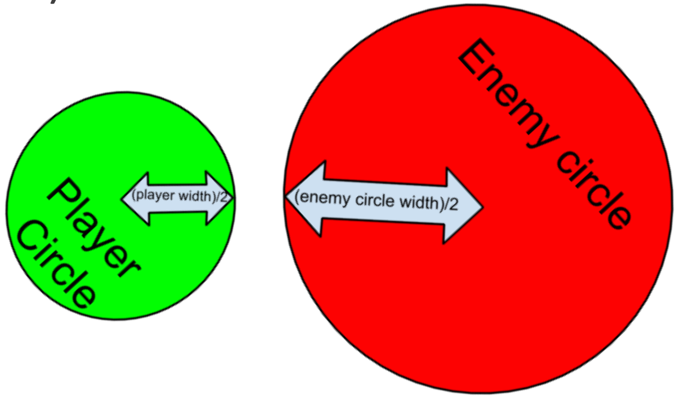


Figure 7

Projectiles and Collisions

Projectiles at constant speed

- Instantiate a projectile at an initial position
- Set the velocity vector towards a destination (v = destination – initial)
- Set the magnitude of your vector to be the speed you need (normalize then multiply by speed)



Detecting collision between 2 circles of arbitrary position (Vector2 c_1 and c_2) and radius (float r_1 and r_2)

- Calculate distance between 2 centers (c_1 and c_2)

$$d = \sqrt{(c_2.x - c_1.x)^2 + (c_2.y - c_1.y)^2}$$

- if ($d < r_1 + r_2$), then the 2 circles collide

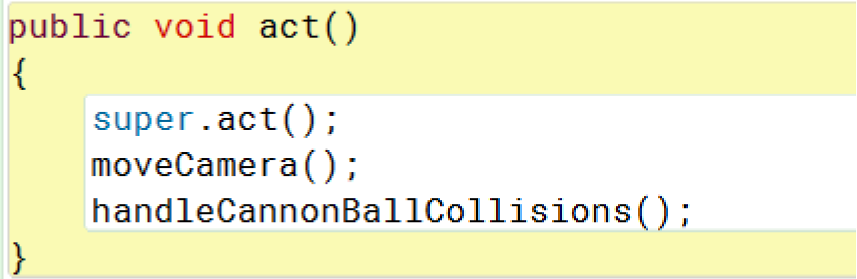
You can get projectiles collision circle approximately by setting the radius to half the size of the sprite

Every frame, detect collision between each pair of circles that can collide

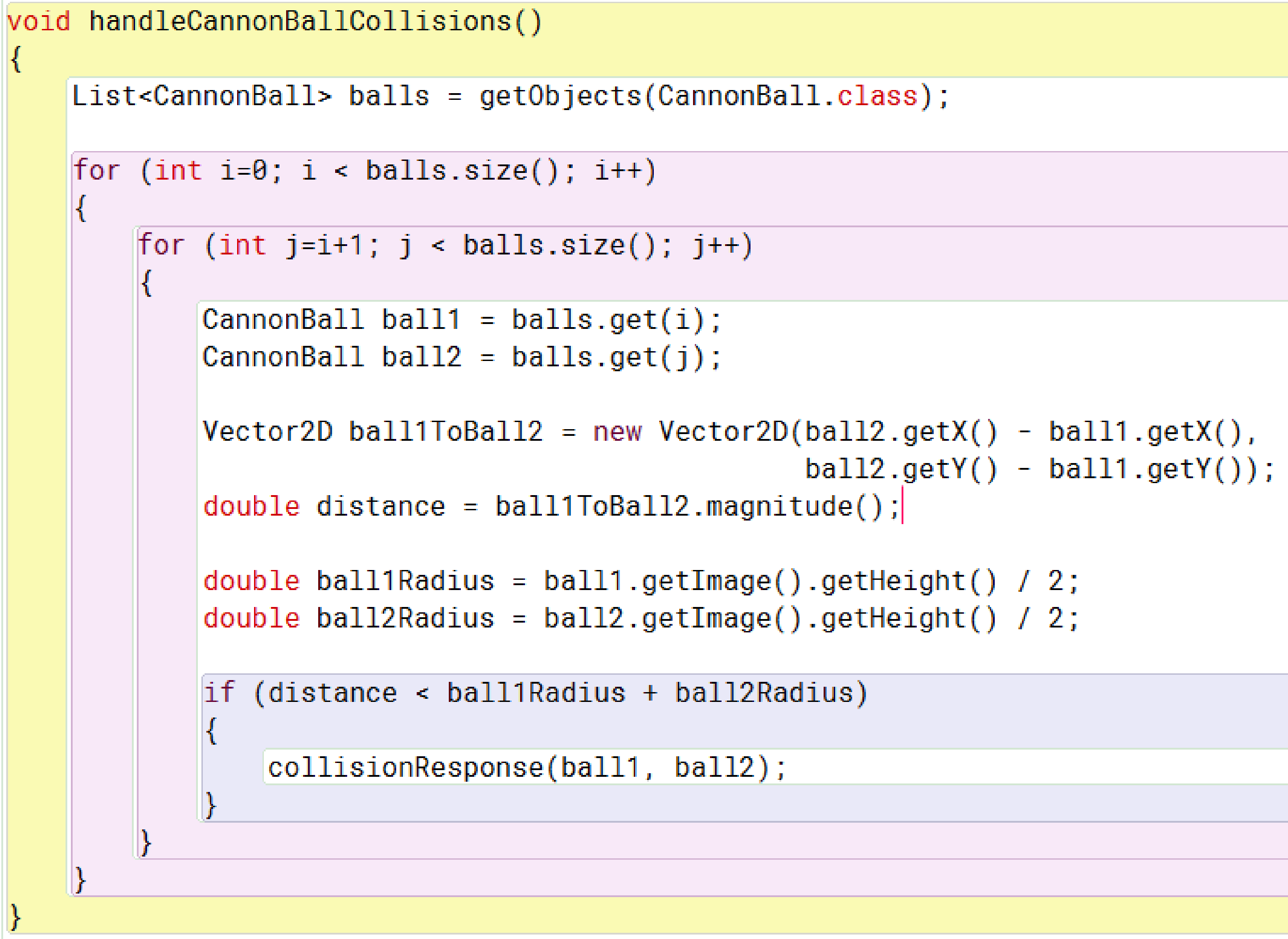
Step 5: Detect collision between cannon balls in MyWorld

For each pair of **Cannon ball**, check it distance is less than the radius of both balls.

When it is, we can compute a collision response (this one is provided, but explained next)

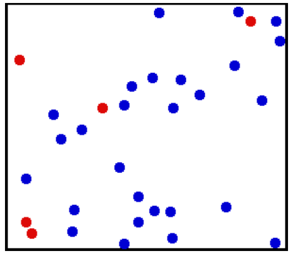


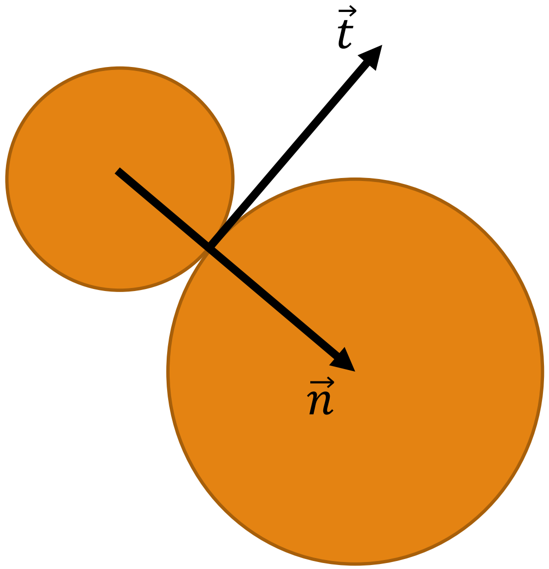
```
public void act()
{
    super.act();
    moveCamera();
    handleCannonBallCollisions();
}
```



```
void handleCannonBallCollisions()
{
    List<CannonBall> balls = getObjects(CannonBall.class);

    for (int i=0; i < balls.size(); i++)
    {
        for (int j=i+1; j < balls.size(); j++)
        {
            CannonBall ball1 = balls.get(i);
            CannonBall ball2 = balls.get(j);

            Vector2D ball1ToBall2 = new Vector2D(ball2.getX() - ball1.getX(),
                                                 ball2.getY() - ball1.getY());
            double distance = ball1ToBall2.magnitude();

            double ball1Radius = ball1.getImage().getHeight() / 2;
            double ball2Radius = ball2.getImage().getHeight() / 2;

            if (distance < ball1Radius + ball2Radius)
            {
                collisionResponse(ball1, ball2);
            }
        }
    }
}
```

2D Physics – Circle Collision Response

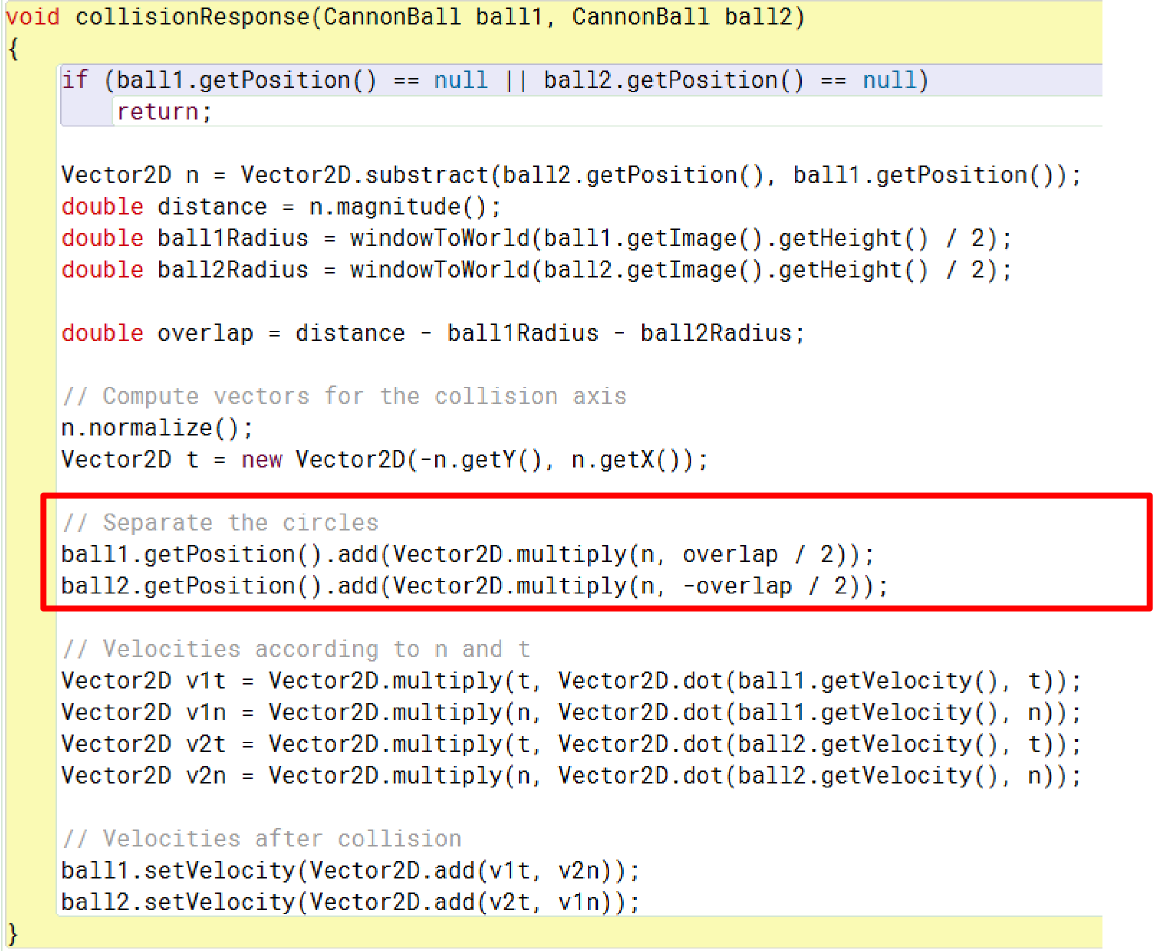




- 2 Circles
  - centered at C_1 and C_2
  - with velocities $\overrightarrow{v_1}$and $\overrightarrow{v_2}$
- Define a normal and tangent vectors
  - $\vec{n} = C_2 - C_1$ (and normalize $\vec{n}$)
  - $\vec{t}$ = $(-n_y, n_x)$ (perpendicular to $\vec{n}$)
- Decompose both velocities according to $\vec{n}$ and $\vec{t}$
  - $\overrightarrow{v_1}^t = (\overrightarrow{v_1} \cdot \vec{t})\vec{t}$ $\overrightarrow{v_1}^n = (\overrightarrow{v_1} \cdot \vec{n})\,\vec{n}$
  - $\overrightarrow{v_2}^t = (\overrightarrow{v_2} \cdot \vec{t})\vec{t}$ $\overrightarrow{v_2}^n = (\overrightarrow{v_2} \cdot \vec{n})\,\vec{n}$
- Velocities after collision $\overrightarrow{v_1}$’and $\overrightarrow{v_2}$‘
  - $\overrightarrow{v_1}' = \overrightarrow{v_1}^t + \overrightarrow{v_2}^n$
  - $\overrightarrow{v_2}' = \overrightarrow{v_2}^t + \overrightarrow{v_1}^n$

Step 6: Collision response for 2 cannon balls in MyWorld

From the previous slide, there is an extra step: we must separate circles to prevent them to get stuck with each other.



```
void collisionResponse(CannonBall ball1, CannonBall ball2)
{
    if (ball1.getPosition() == null || ball2.getPosition() == null)
        return;

    Vector2D n = Vector2D.substract(ball2.getPosition(), ball1.getPosition());
    double distance = n.magnitude();
    double ball1Radius = windowToWorld(ball1.getImage().getHeight() / 2);
    double ball2Radius = windowToWorld(ball2.getImage().getHeight() / 2);

    double overlap = distance - ball1Radius - ball2Radius;

    // Compute vectors for the collision axis
    n.normalize();
    Vector2D t = new Vector2D(-n.getY(), n.getX());

    // Separate the circles
    ball1.getPosition().add(Vector2D.multiply(n, overlap / 2));
    ball2.getPosition().add(Vector2D.multiply(n, -overlap / 2));

    // Velocities according to n and t
    Vector2D v1t = Vector2D.multiply(t, Vector2D.dot(ball1.getVelocity(), t));
    Vector2D v1n = Vector2D.multiply(n, Vector2D.dot(ball1.getVelocity(), n));
    Vector2D v2t = Vector2D.multiply(t, Vector2D.dot(ball2.getVelocity(), t));
    Vector2D v2n = Vector2D.multiply(n, Vector2D.dot(ball2.getVelocity(), n));

    // Velocities after collision
    ball1.setVelocity(Vector2D.add(v1t, v2n));
    ball2.setVelocity(Vector2D.add(v2t, v1n));
}
```

Step 7: Commit your changes and push to github (1)

Try to remember the following commands (see Figure 8a and Figure 8b)

- git add *
- git status /*check that files are there*/
- git commit -m “Detect collision between cannon balls and Collision response"

(if you need to enter your name and email, just use the setup commands and commit again)

And then push your commit to Github

- git push

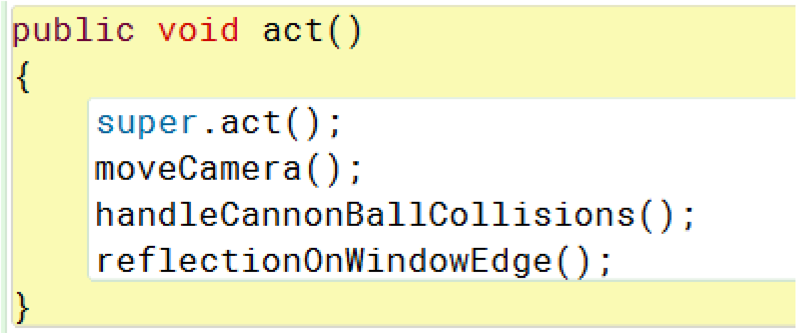
  /* Push the files to your repository, if you don’t do this step, your files will not be saved online */

Double-check Github.com for your commit

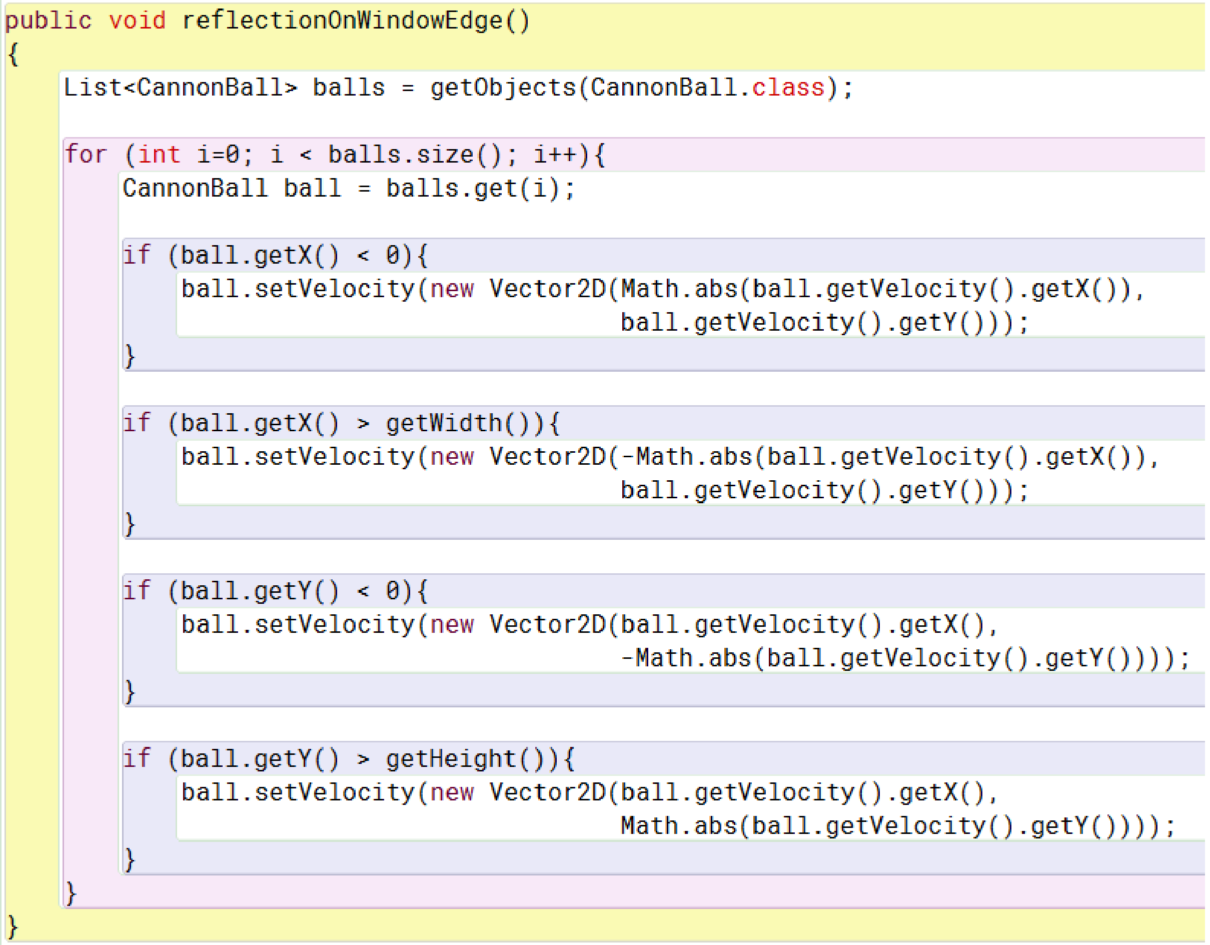
Step 8: Keep the cannon balls within the window

When the cannon ball gets out of bounds, let's set the velocity to bring it back inside the window. 4 cases:

- **Left side**: Make velocity "x" component positive
- **Right side**: Make velocity "x" component negative
- **Top side**: Make velocity "y" component negative
- **Bottom side**: Make velocity "y" component positive



```java
public void act()
{
    super.act();
    moveCamera();
    handleCannonBallCollisions();
    reflectionOnWindowEdge();
}
```



```java
public void reflectionOnWindowEdge()
{
    List<CannonBall> balls = getObjects(CannonBall.class);

    for (int i=0; i < balls.size(); i++){
        CannonBall ball = balls.get(i);

        if (ball.getX() < 0){
            ball.setVelocity(new Vector2D(Math.abs(ball.getVelocity().getX()),
                                          ball.getVelocity().getY()));
        }

        if (ball.getX() > getWidth()){
            ball.setVelocity(new Vector2D(-Math.abs(ball.getVelocity().getX()),
                                          ball.getVelocity().getY()));
        }

        if (ball.getY() < 0){
            ball.setVelocity(new Vector2D(ball.getVelocity().getX(),
                                          -Math.abs(ball.getVelocity().getY())));
        }

        if (ball.getY() > getHeight()){
            ball.setVelocity(new Vector2D(ball.getVelocity().getX(),
                                          Math.abs(ball.getVelocity().getY())));
        }
    }
}
```

Step 9: Commit your changes and push to github (2)

Try to remember the following commands

- git add *
- git status /*check that files are there*/
- git commit -m "Keep the cannon balls within the window"

(if you need to enter your name and email, just use the setup commands and commit again)

And then push your commit to Github

- git push

  /* Push the files to your repository, if you don't do this step, your files will not be saved online */

-

Double-check Github.com for your commit