2017

SIMULATION & PHYSICS – PRACTICAL 5

COLLISION 2

SJORS GIELEN
500765899

1. repeat the assignment you are implementing;

Finish the ResolveCollisionWith method to handle collisions with balls of a different mass in a generic manner.

2. explain your approach;

Implement a _Mass property and a _InverseMass property along with their respective class scope variables. The _InverseMass property is a read only property as the inverse mass is set while the mass gets set. This is to ensure that the mass and inverse mass are always respective too one another.

Then detect if there is a collision, then simply executed the vector math required for this. But now also involving the inverseMass value.

3. describe your code;

If the difference in length is smaller then the two radii added tighter there is an overlap, then get the collision normal. Fix interpenetration. Dot both ball's velocity's and place the value's in their own respective floats. Then calculate

```
float p = (2.0f * (a1 - a2)) / (this._InverseMass + ball._InverseMass);
```

From here add/minus the balls respective InverseMass and the collision Normal. Added in the inelastic value for good measure.
For fun I also added in that when you left click on a ball you grabbed said ball and can throw it around.

4. show (relevant) code snippets;

```
public void ResolveCollisionWith(Ball ball) {
    //Step 1: calculate the vector from the position of this ball to the other ball
    Vector2 dif = ball.Position - this.Position;
    //Step 2: calculate the distance between the two balls
    float dist = dif.Length() - (ball.Radius + this.Radius);
    //Step 3: if there is a collision
    if (dist < 0)
    {
        //Step 4: calculate the collision normal
        Vector2 n = Vector2.Normalize(dif);

        //Step 5: Resolve interpenetration
        Vector2 resetVector = n * (dist / 2);
        this.Position += resetVector;
        ball.Position -= resetVector;

        float a1 = Vector2.Dot(this.Velocity, n);
        float a2 = Vector2.Dot(ball.Velocity, n);

        //Step 6: calculate the velocity component parallel to normal
        float p = (2.0f * (a1 - a2)) / (this._InverseMass + ball._InverseMass);

        //Step 7: calculate the changeVelocity
        Vector2 v1 = this.Velocity - p * this._InverseMass * n * this.inelastic;
        Vector2 v2 = ball.Velocity + p * ball._InverseMass * n * ball.inelastic;

        //Step 8: change the velocities
        this.Velocity = v1;
        ball.Velocity = v2;
    }
```

The entire method.

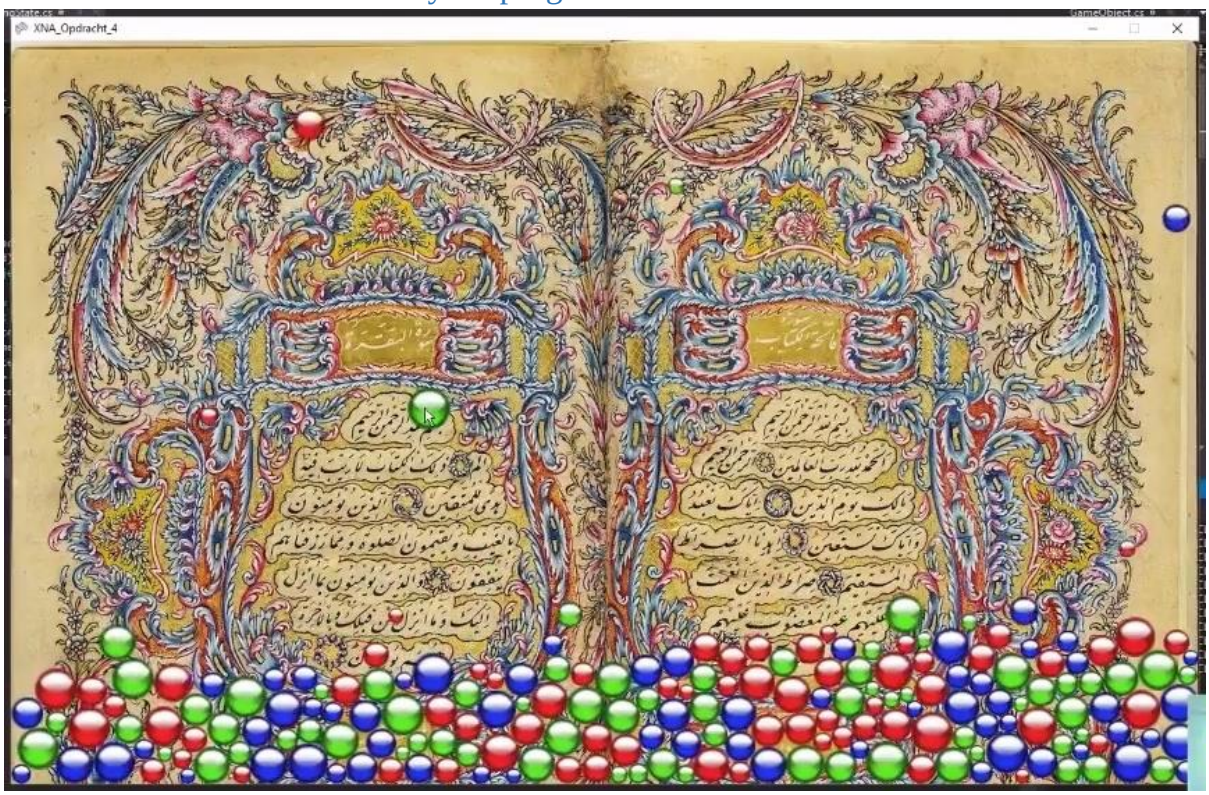
```
private float inverseMass;
public float Radius
{
    get { return radius; }
    set { radius = value; }
}

public float _Mass
{
    get { return this.mass; }
    set
    {
        this.inverseMass = 1 / (value);
        this.mass = value;
    }
}

public float _InverseMass
{
    get { return this.inverseMass; }
}
```

The properties I setup to support this

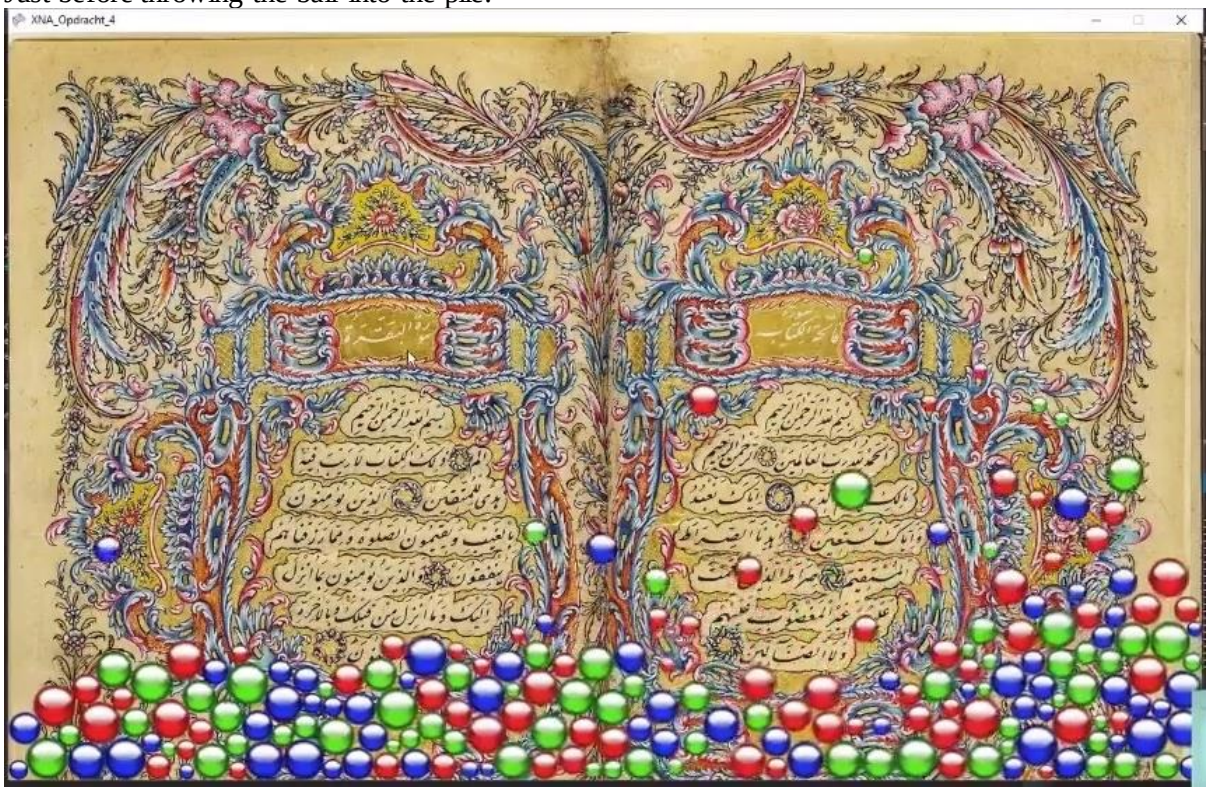
```
public override void HandleInput(InputHelper inputHelper)
{
    mouse = new Vector2(inputHelper.MousePosition.X, inputHelper.MousePosition.Y);
    if (grabbed == null)
    {
        if (inputHelper.MouseLeftButtonPressed())
        {

            for (int i = 0; i < balls.Objects.Count; i++)
            {
                Ball ball = balls.Objects[i] as Ball;
                if((ball.Position - mouse).LengthSquared() < ball.Radius * ball.Radius)
                {
                    grabbed = ball;
                    break;
                }
            }
        }
    }
    else
    {
        if (!inputHelper.MouseLeftButtonDown())
        {
            grabbed = null;
        }
        else
        {
            grabbed.Velocity = (mouse - prevMouse) / (float)this.gameTime.ElapsedGameTime.TotalSeconds;
        }
    }
    prevMouse = mouse;
    base.HandleInput(inputHelper);
}
```

Grabbing code

5. include a screenshot of your program



Just before throwing the ball into the pile.



Directly after the ball crashed into the pile.

Again I have made video's for the class as well These are found here: https://www.youtube.com/watch?v=oYf3RrOWsyI&list=PLARkMALdMekM6EMkY0gcQSKvADVAx9zK5&index=7