2017

SIMULATION & PHYSICS – PRACTICAL 1

BOUNCEWORLD

SJORS GIELEN
500765899

1. repeat the assignment you are implementing;

Create three balls that each have certain aspects of physics applied to them.

The red ball move horizontally from left to right;
The red ball starts with velocity Vector2.Zero and accelerates to the right and bounces on the side of the screen;

The pink ball move diagonally from lower left corner to upper right corner;
The pink ball starts moving diagonally from lower left corner to upper right corner but because of gravity its path will be parabolic (it keeps bouncing indefinitely);

The purple ball move vertically down from the top of the screen to the bottom.
The purple ball falls down vertically using gravity, bounces at the bottom but by use of friction it slows down and bounces less and less until it rests at the bottom.

2. explain your approach;

For the red ball I initialized a ball that starts on the desired location, then I gave it .zero vector to start out on, if a ball with the specific assetname I'd force it to take the accel vector of (10,0).

For the pink ball I've added a GravityDirection parameter to the ball, which is always a unit vector(due to the constructor code). This gravity direction controls where the gravity pulls the object towards. I also added a mass value(and parameter) and a gravity constant too the ball class. Then the gravity is added to the velocity value by doing this.velocity += mass * gravityDir * g.

Finally for the purple ball I've added in a drag coefficient that remains between 0 and 0.99999. Then I reduce the velocity with the movementDir * drag on each frame. This movementDir is calculated by normalizing the Velocity vector.

Also for testing purposes I've added 50 random balls which receive parameters on a random basis.

3. describe your code;

All the relevant code is rather simplistic in nature. I strongly believe the approach explanation above suffices together with the code snippets below. If not please review the VOD which is found at the end of this document.

4. show (relevant) code snippets;

```
private float g = 0.0981f;
```

```
public Ball(string assetName, Vector2 position, Vector2 velocity, float radius, Vector2 GravityDirection, float Mass = 0, float drag = 0)
    : base(assetName, 0, "ball") {
    this.position = position;
    this.velocity = velocity;
    origin = new Vector2(Width / 2f, Height / 2f);
    this.radius = radius;
    this._Mass = Mass;
    this._Drag = drag;
    this.gravityDir = Vector2.Normalize(GravityDirection);
    scale = radius * 2 / Width;
    if (assetName == "RedBallX")
    {
        accel = new Vector2(10, 0);
    }
}
```

```
public float _Mass
{
    get { return this.mass; }
    set
    {
        if (value < 0)
        {
            this.mass = 0;//ensure negative mass doesn't occur, in our simu we assume negative mass object do not exsist.
        }
        else
        {
            this.mass = value;
        }
    }
}
public float _Drag
{
    get { return this.drag; }
    set
    {
        if (value <= 0)
        {
            this.drag = 0;//ensure negative drag does not exsist.
        }
        else if(value >= 1)
        {
            this.drag = 0.9999999f;//ensure drag never becomes 1.
        } else
        {
            this.drag = value;
        }
    }
```

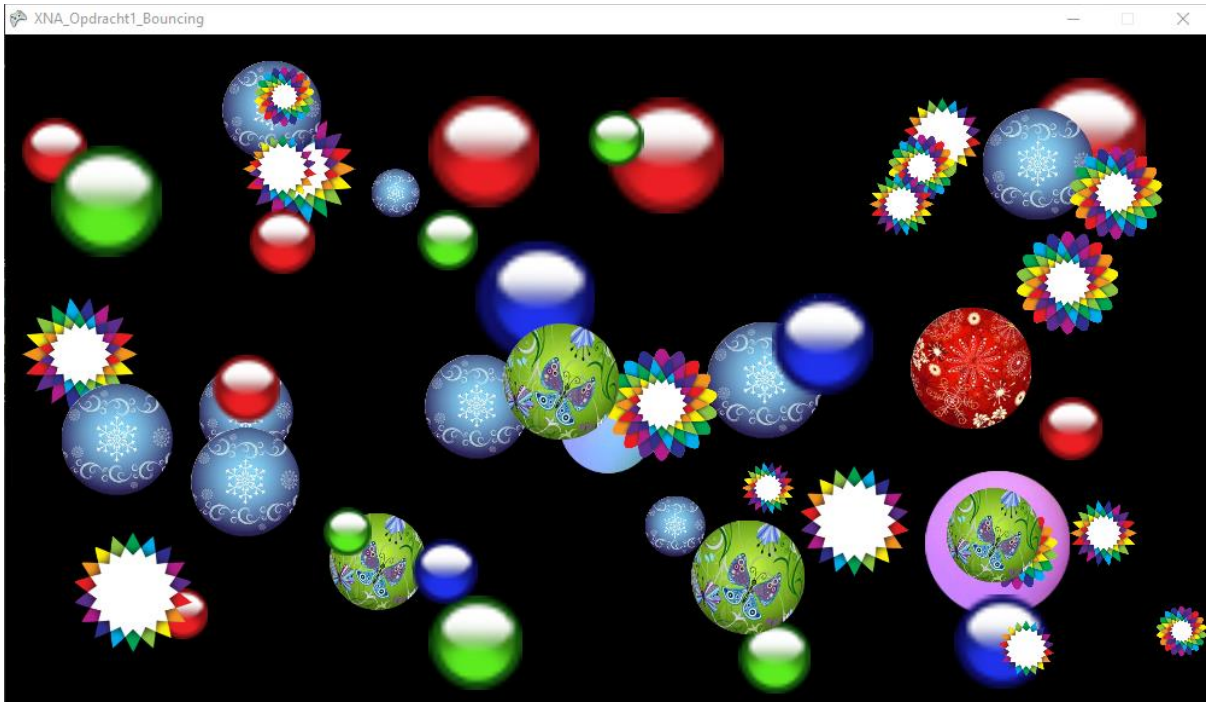
```
if(position.X - radius < 0)
{
    position.X = radius;
    velocity.X *= -1f;
}
if(position.X + radius > GameEnvironment.Screen.X)
{
    position.X = GameEnvironment.Screen.X - radius;
    velocity.X *= -1f;
}
if (position.Y - radius < 0)
{
    position.Y = radius;
    velocity.Y *= -1f;
}
if (position.Y + radius > GameEnvironment.Screen.Y)
{
    position.Y = GameEnvironment.Screen.Y - radius;
    velocity.Y *= -1f;

}
// step 5: Implement acceleration behaviour
this.velocity += accel;
// step 7: Implement gravity behaviour
this.velocity += mass * gravityDir * g;
if (drag != 0)
{
    movementDir = Vector2.Normalize(Velocity);
    Velocity = Velocity - movementDir * drag ;
}
this.velocity += mass * gravityDir * g;
```

```
// step 1: Initialize three balls on the correct position
Ball ball1 = new Ball("RedBallX", new Vector2(0, GameEnvironment.Screen.Y / 2), Vector2.Zero, 50, new Vector2(20, 12));
Add(ball1);
Ball ball2 = new Ball("PinkSoftColorBall", new Vector2(0, GameEnvironment.Screen.Y), new Vector2(200,-400), 60, new Vector2(0,10), 40);
Add(ball2);
Ball ball3 = new Ball("PurpleSoftColorBall", new Vector2(GameEnvironment.Screen.X / 2, 0), new Vector2(0,80), 40, new Vector2(0, 10), 30, 0.5f);
Add(ball3);
// step 2: Give them the correct velocity

// step 10: randomize the position and starting velocity
string[] assetnames = { /*"RedballX", "PinkSoftColorBall", "PurpleSoftColorBall",*/ "BlueBallX", "FlowerMulticolor", "GreenBallx",
    "StarLargeMulticolor", "StarSmallMulticolor", "spr_ball_blue", "spr_ball_green", "spr_ball_red" };
Random r = new Random();
for(int i = 0; i < 50; i++)
{
    string assetname = assetnames[r.Next(assetnames.Length)];
    var position = new Vector2(GameEnvironment.Random.Next(100, GameEnvironment.Screen.X - 100),
    GameEnvironment.Random.Next(100, GameEnvironment.Screen.Y - 100));
    var velocity = new Vector2(GameEnvironment.Random.Next(-150, 150),
        GameEnvironment.Random.Next(-150, 150)) * 3;
    var gravdir = new Vector2(GameEnvironment.Random.Next(-150, 150),
        GameEnvironment.Random.Next(-150, 150));
    var ballr = new Ball(assetname, position, velocity, r.Next(30) + 20, gravdir, r.Next(19) + 2, (float)r.NextDouble());
    Add(ballr);
}
```

5. include a screenshot of your program



At request of some of my class-mates I have also streamed this excersize along with explanations on how certain things work. The video on demand is available here: https://www.youtube.com/watch?v=_b_5p3FXv-M&index=1&list=PLARkMALdMekM6EMkY0gcQSKvADVAx9zK5