CE812: Main Programming Assignment Report

Registration Number: 2312468

Report

1. The Game Description

1.1. Game Story

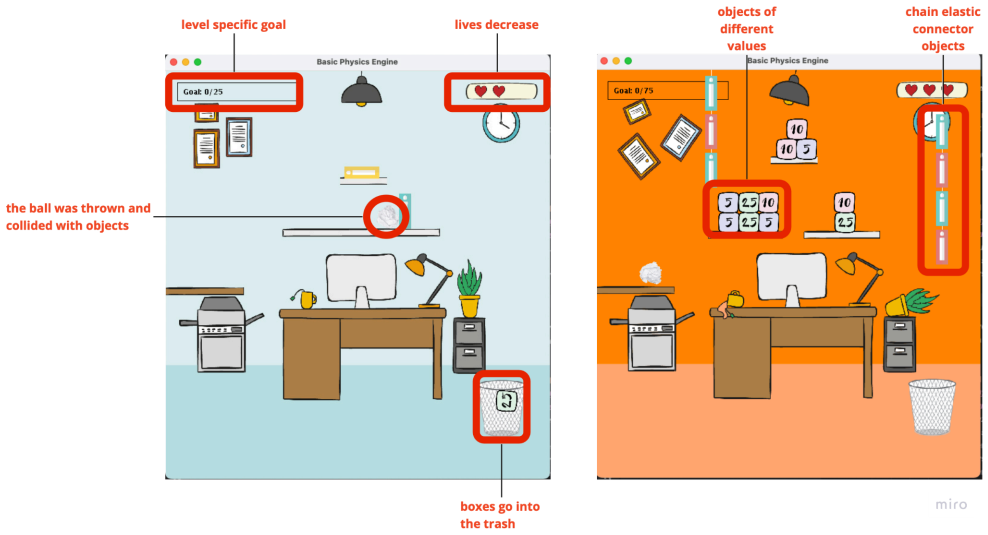
Bored Peter throws the crumpled paper in his hand into the trash. Then he tries to throw away the ornaments on the shelves with the papers on his desk. He created a game for himself out of boredom. Maybe it's a dream he had while taking a nap at his desk. Will the boss come soon?

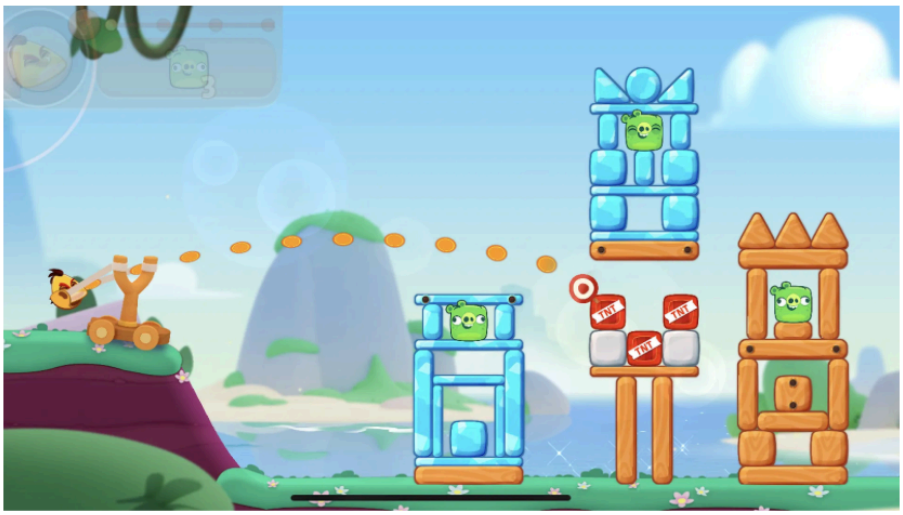
1.2. How To Play

- The player pulls the mouse and throws the paper ball - like a slingshot.
- The further back the player pulls it, the faster the ball will be thrown. It will be thrown according to the angle when you release the mouse.
- The player has 3 rights at each level.
- If the ball falls to the ground, it loses a life.
- The aim of the game is to put the squares with points written on them into the distant trash can.
- Each level has a separate target score. If the total of the squares placed in the trash can reaches the target before your life runs out, you win.

1.3. Key Features

- Throwing the paper ball with the mouse and object collision
- Score increase for score objects entering the trash can
- Lives reduction on failure
- Vertical, horizontal and chain type obstacles
- Squares with different scores

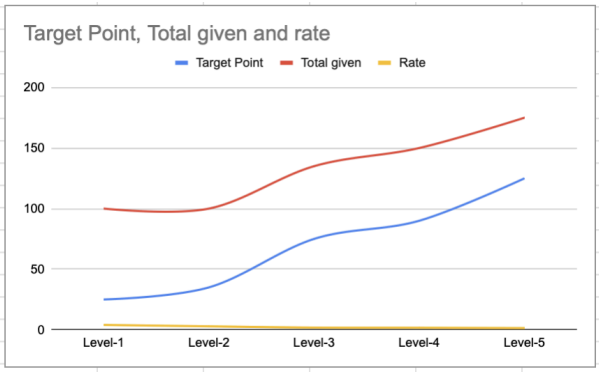




The game is inspired by the game Paper Toss on the left, Angry Birds on the right.
We throw the crumpled paper into the trash can, as in paper toss. Unlike Paper Toss, there are also obstacles in the game, it is not just about putting it in the trash can.
We try to achieve our goal by hitting obstacles, just like in the Angry Birds game. The difference from Angry Birds is that in order to get a score, we need to put the target object into the trash can, not explode it.

Hand drawn vectors were used in the graphics and general structure because it was designed as if it were a game drawn on paper. Difficulty increases in levels. There are currently 3 levels, but an economy of increasing difficulty has been prepared as the levels increase. This level difficulty design was prepared in accordance with Figure 9.6 in The Art of Game Design book.

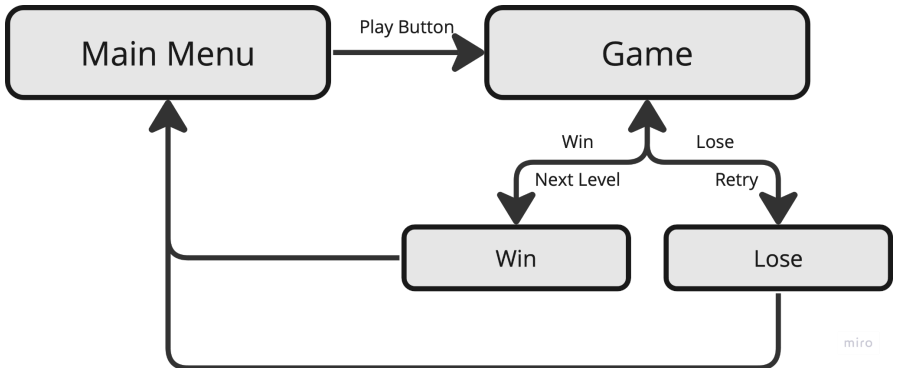
	Level-1	Level-2	Level-3	Level-4	Level-5
Target Point	25	35	75	90	125
Total given	100	100	135	150	175
Rate	4	2.857142857	1.8	1.666666667	1.4
	Given points	Given points	Given points	Given points	Given points
	25	25	25	25	25
	25	25	25	25	25
	25	25	25	25	25
	10	10	10	10	25
	10	10	10	10	10
	5	5	10	10	10
			10	10	10
			5	10	10
			5	5	10
			5	5	5
				5	5
					5



1.4. Panels

Screen Transactions

- Main Menu → Game
- Game → Win panel
- Game → Lose Panel
- Win panel → Main Menu
- Lose Panel → Main menu
- Win panel → Game
- Lose Panel → Game



2. Technical Issues

2.1. Choice of physics engine

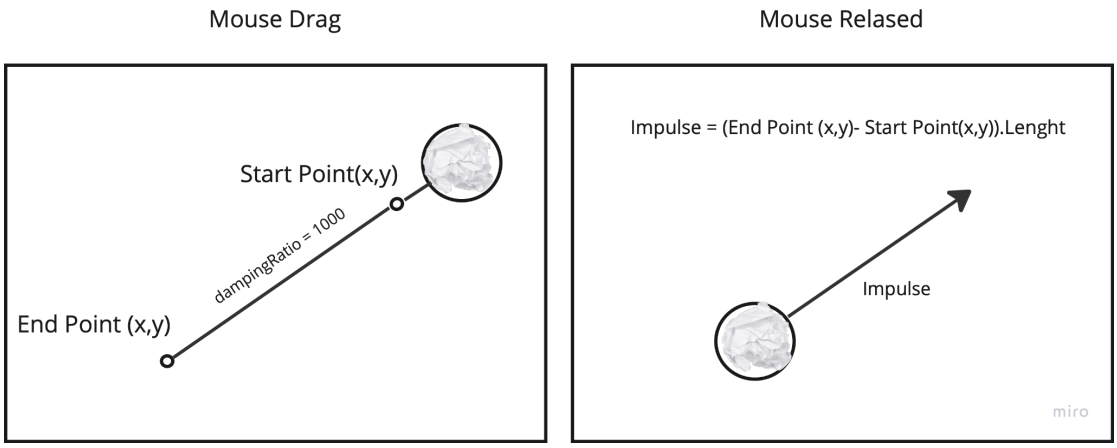
The game was designed and conceived in 2D, so JBox2D, which is optimized for this type of games, was used. JBox2D has a simple usage and is very suitable for simulating the game with simple levels such as collision of paper balls and files and creating a trash can. All methods available in JBox2D are sufficient to simulate this game.

2.2. Describe the features

2.2.1. Throwing crumpled paper

The `BasicMouseListener` class from Labs was used. On `mouseRelased`
I wrote the method and made a small change in the `mouseDragged` method.
I made a big number for the dumping ratio and the created Joint behaves like a slingshot. The pulled particle does not move.

In the Mouse Released method, the Start point is taken as the first point where the drag starts and the End point is taken as the released point. We find the lenght of the vector between them and apply it to the ball as impulse.



Code:

```
public void mouseReleased(MouseEvent e) {
   if (mouseJoint != null) {
      PlayThrowSound(); // play throwing sound when mouse released
      System.out.println("mouseReleased");
      Vec2 impulse = startPoint.sub(endPoint); // Impulse vector
      float angle = (float) Math.atan2(impulse.y, impulse.x); // finding the
angle
      float impulseMagnitude = impulse.length(); // Finding impulse length
      float scaleFactor = 0.0002f;
      impulseMagnitude *= scaleFactor;
      Vec2 impulseVector = new Vec2(MathUtils.cos(angle),
MathUtils.sin(angle)).mulLocal(impulseMagnitude);
      mouseJoint.getBodyB().applyLinearImpulse(impulseVector,
mouseJoint.getBodyB().getPosition(), true);
      linkMouseDragEventToANewMouseJoint(null);
   }
}
```

2.2.2. Trash can

The AnchoredBarrier_StraightLine class was used and the boundaries of the trash can were determined. A picture of the trash can was uploaded to the relevant borders on the screen. Additionally, if the player enters a point square within the boundaries of the trash can, the players score has been increased.

```
for (int i = 0; i < game.squareShapes.size(); i++) {
   SquareShape points = game.squareShapes.get(i);
   if (points.body.getPosition().y < 1.4f && points.body.getPosition().x > 8.3f
&& points.body.getPosition().x < 9.2f) {
```

```
            points.destroy();
            game.squareShapes.remove(points);

            int pointAmount = points.getPointAmount();
            PointSystem.increaseScore(pointAmount); // increase score
            SoundPlayer musicPlayer = new SoundPlayer("assets/collect.wav");
            musicPlayer.play();
            if (PointSystem.isTargetAchieved()) {
                showYouWinPanel(game);
            }
        }
        else if (points.body.getPosition().y < 1.4f) {
            points.destroy();
            game.squareShapes.remove(i); // Just delete the point object
        }
    }
```

2.2.3.Obstacles

2.2.3.1. Barriers and Rectangles

Shelf boundaries were determined using Barriers within the levels. The AnchoredBarrier_StraightLine class was used for this. Some changes were made to the BasicPolygon class for rectangles and for the "arch file" objects in the game. Transformations were made in the draw method within the BasicPolygon class.

```
Vec2 position = body.getPosition();
float angle = body.getAngle();
AffineTransform af = new AffineTransform();
af.translate(BasicPhysicsEngineUsingBox2D.convertWorldXtoScreenX(position.x),
BasicPhysicsEngineUsingBox2D.convertWorldYtoScreenY(position.y));
af.scale(ratioOfScreenScaleToWorldScale, -ratioOfScreenScaleToWorldScale);//
af.rotate(angle);
Path2D.Float p = new Path2D.Float (polygonPath,af);
```

mkRectangle method was written, this method creates a rectangle.

```
public static Path2D.Float mkRectangle(float width, float height) {
    Path2D.Float p = new Path2D.Float();
    p.moveTo(-width / 2f, -height / 2f);
    p.lineTo(width / 2f, -height / 2f);
    p.lineTo(width / 2f, height / 2f);
    p.lineTo(-width / 2f, height / 2f);
    p.lineTo(-width / 2f, -height / 2f);
    p.closePath();
    System.out.println(p);
    return p;
}
```

And a new constructor was added to BasicPolygon.

```
public BasicPolygon(float sx, float sy, float vx, float vy, Color col, float
mass, float rollingFriction, float width, float height, int numSides, Image
fileImage) {

   this(sx, sy, vx, vy, width/2, col, mass, rollingFriction, mkRectangle(width,
height), numSides, fileImage);
}
```

Thus, we were able to create rectangles by entering height and width values.

2.2.3.2. Chain Connector

A special obstacle has been coded for Level 3. It consists of arch files connected to each other with a chain connector. For this, I made the input parameters in the ElasticConnector class BasicPolygon. And also the Connec `ConnectorForLine` class was created to connect the line and the polygon. In this way, permanent obstacles that are connected to each other and cannot be destroyed added difficulty to the level.

`DistanceJoint` was used here. In the code below there are polygons connected to each other. These values are in section 2.5.. It is explained in.

```
connectorsForLine.add(new ConnectorForLine(lineBarriers.get(0),
chainPolygons.get(0), 0.1f, springConstant,springDampingConstant, false,
Color.WHITE, hookesLawTruncation));
connectors.add(new ElasticConnector(chainPolygons.get(0), chainPolygons.get(1),
0.1f, springConstant,springDampingConstant, false, Color.WHITE,
hookesLawTruncation));
connectors.add(new ElasticConnector(chainPolygons.get(1), chainPolygons.get(2),
0.1f, springConstant,springDampingConstant, false, Color.WHITE,
hookesLawTruncation));
```

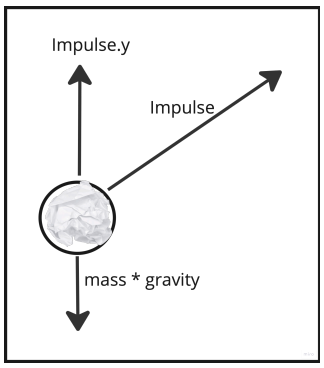
2.3. Physics principles

2.3.1. Collision

This happens when two objects collide with each other.

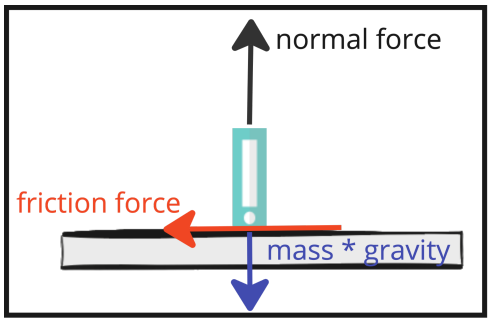
When two objects collide with each other, total Momentum is conserved. According to this principle, the sum of m*v(ball) and m*v(file) will be preserved. As a result of the collision, the motion of the two objects is created accordingly.

2.3.2. Falling



Fall of the object. We observe this when the paper ball falls after being thrown. Since there is gravity in the environment, it falls downwards as much as its own weight and gravity. If it is upward, the y component of the thrown impulse applies a force to the object in the upward direction. As a result of this balance, the object falls downwards.

2.3.3. Friction

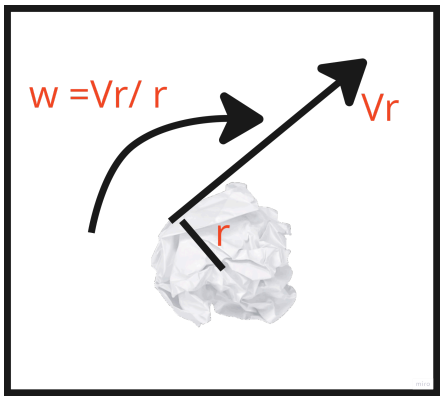


There is friction on surfaces. If objects move horizontally on the surface, they stop after a while.

```
float linearDragForce=.02f;
```

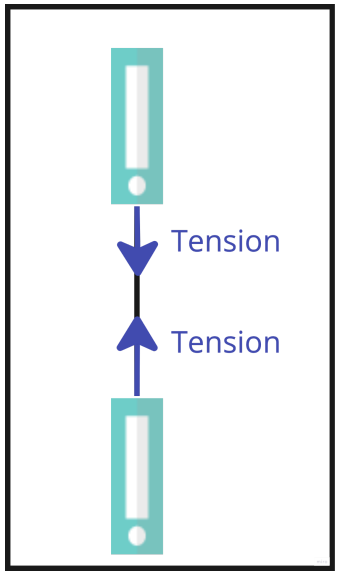
It was determined as , but 0.02 /15 was applied in basicPolygons. 0.02 was applied to the Ball object. The mass value of the Ball object was made smaller and different mass values were created in basic polygons.

2.3.4. Angular Velocity



Angular velocity (ω) is the amount of angle rotated per unit time. If the radius of the ball is divided by the Vr applied to the paper ball, the angular velocity is obtained.

2.3.5. Elasticity and tension



There are ElasticConnectors among the obstacles created at Level 3. We tie these two objects together with a rope. We wanted the rope tension here to be very high and rigid. for this $Tension = \lambda\ (extension)\ /\ natural\ lengt$ In the formula $\rightarrow T = \lambda * x\ /\ l$, natural length 0.1, spring constant = 1000000 and springDampingConstant=1000 were selected. Thus, we obtained a very rigid rope.

2.4. Bugs and Unfinished Features

2.4.1. Game features

- Longer levels can be made horizontally in the game and image scaling can be done accordingly.
- The game is currently only 640 * 680 px in size and then the paper ball can be tracked with a camera.
- 3. When we run out of rights, the game ends even if we still have squares that fall and could potentially go to waste. Here, in order to finish the game, the moving objects can be waited to stop or their speed can be expected to fall below a certain level.

2.4.2. Physics features

- The throwing mechanism in the game is currently not working properly and gives errors occasionally. In addition, we can now push all objects with the mouse, this should be arranged so that it only applies to the ball.

2.4.3. Graphical features

- The connector made by the paper ball and the mouse is not visible in the game. A drawLine method should be applied here.
- User interface improvements should be made in the game, a separate user interface class and methods should be created for this, and the scales of the images in the game should be dynamically adjusted according to the screen sizes.

2.5. Some Parameters

- Delay = 20
- DeltaT = Delay/1000f (these values offer the closest simulation to daily life in the office environment)
- The linearDragForce of the ball is 0.02 and mass = 0.005
- The linearDragForce value of arch files and scored squares is 0.02/15 and mass = 1
- Impulse scale factor = 0.05
- Impulse size is found by the difference between the start point and end point and scaleFactor is 0.002f.
- Spring constants: double springConstant=1000000, springDampingConstant=1000;

3. Reflection

Graphics were challenging without using external libraries. However, it was easier for JBox2D to handle the laws of physics.

I would like to make some changes in the general structure of the game. First of all, the level creation system can be made more dynamic. It is currently created manually. A separate class and methods can be created to spawn objects and improvements should be made on the graphics. For cleaner coding, a separate class can be created for Image embeds.

When you win the game, you just pass the level. Currency can be added and different types of paper balls can be purchased with these coins. In this way, you can create motivation to play. There are very few obstacle types in the game. Obstacles with different style mechanics can be created and animations can be added to the game, as in the Angry Birds game.Different air flows can be created with an object such as a fan.

4. Credits

Idea

- Angry Birds(https://www.angrybirds.com/)
- Paper Toss (https://en.wikipedia.org/wiki/Backflip_Studios#Paper_Toss)

Audios

- Background Music : https://www.youtube.com/watch?v=Qiz5uoPonYY
- Lose Lives Sound : (https://www.zapsplat.com/music/scrunched-up-paper-ball-drop-or-hit-surface-1/)
- Win Sound: (https://www.zapsplat.com/music/game-sound-fast-harp-glissando-win-award-4/)
- Lose Sound: (https://www.zapsplat.com/music/cartoon-orchestral-musical-pizzicato-riff-short-fail-or-lose/)
- Basket Sound: (https://www.zapsplat.com/music/game-sound-action-collect-or-pick-up-point-or-item/)
- Throwing Sound: (https://www.youtube.com/watch?v=WqpktrHj9gI)

Images

- Background *Adopted from (https://www.freepik.com/free-vector/modern-hand-drawn-office-interior_2994569.htm)*
- Crumbled paper(https://www.freepik.com/free-vector/crumple-paper-ball-white-3d-crinkle-trash-vector_48411301.htm)
- Bin(https://www.freepik.com/free-vector/set-waste-container_4298604.htm)
- Office worker in the main menu (https://www.freepik.com/free-vector/man-workers-sitting-desks-set_5529116.htm#fromView=search&page=1&position=1&uuid=b57726b5-2863-41a3-84f3-81afc50fb9c4)

Code

- Dr.Michael Fairbank Lectures Lab 5 java code source

Tools

- Adobe Illustrator : Export vector png ‘s
- Miro Board for creating documents drawings https://miro.com/
- Converting .mp3 files to .wav files https://convertio.co/

Document

- https://sangsara.net/2009/08/12/iphone-app-review-paper-toss-world-tour/
- https://www.axios.com/2022/01/28/rovio-angry-birds-comeback
- Schell, J., The Art of Game Design, 3rd ed. A K Peters/CRC Press, 2019, fig. 9.6.

GITLAB LINK :
https://cseegit.essex.ac.uk/23-24-ce812/23-24_CE812_abay_tugrul_h_f/-/tree/master/main_assignment_2312468?ref_type=heads

```
java  -classpath main_assignment_2312468/out/production/lab5
pbgLecture5lab_wrapperForJBox2D.ThreadedGuiForPhysicsEngine
```