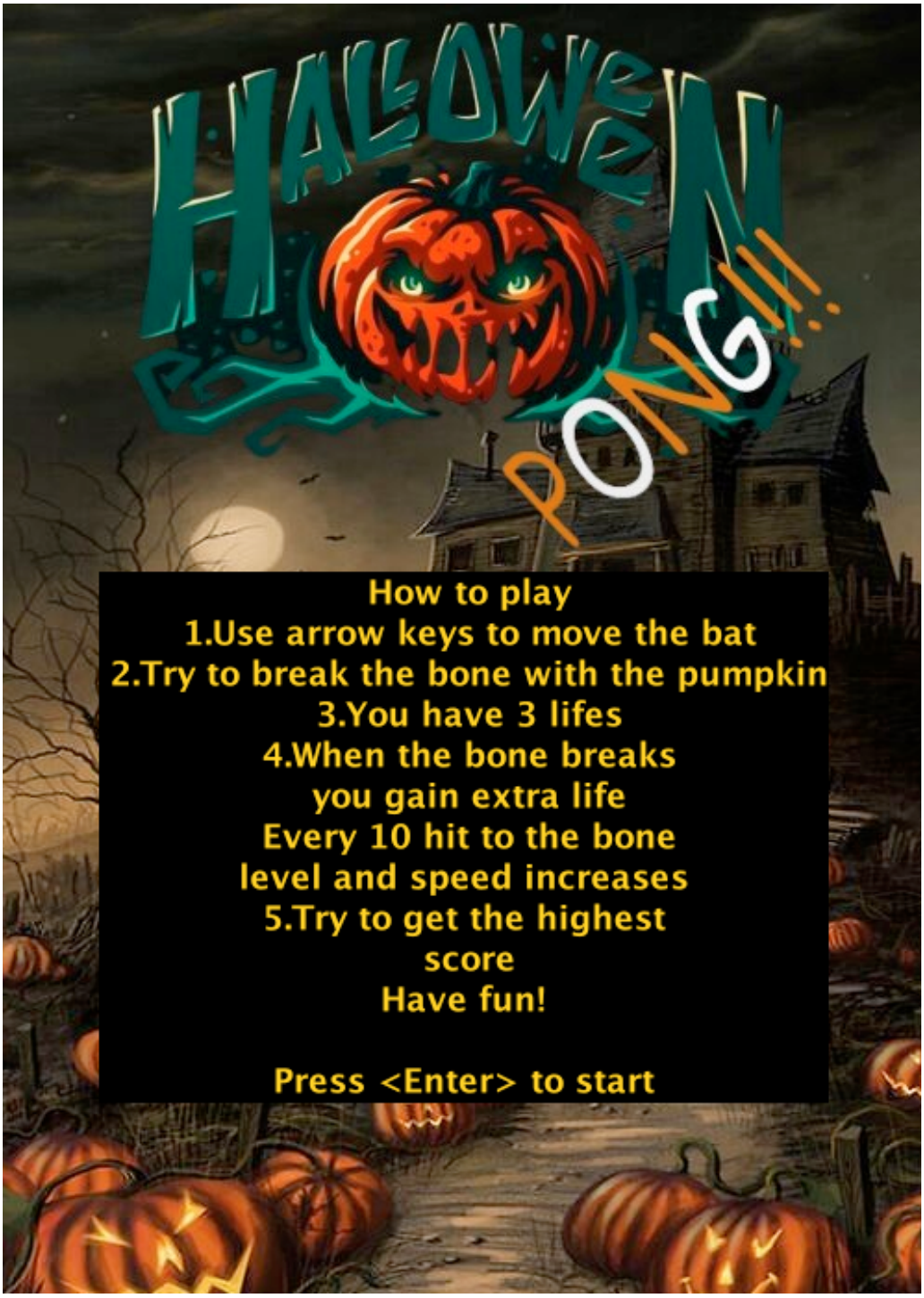
BUSINESS ACADEMY SOUTHWEST

Ping - Compulsory Assignment #1



"The greatest Halloween themed pong inspired game ever"

-Forbes "brilliant gameplay" 10/10

Handed-in at 23/09/2022 by

1. Doina Plesca
2. Tomas Simko
3. Mads Pedersen
4. Frederik Flagstad

Source code:

https://github.com/TomassSimko/PongGame

Table of Content

1. Introduction

The scariest day of the year is coming and that is why we have been working so hard on bringing Halloween themed pong. In our newly created game, you will experience how it is to be a bat that needs to defend moving bones and get the highest possible level. Our game has everything that every successful game should have such as health bar, level progression or possibly to beat your previous highest level.

Our focus regarding the code was on low coupling and high cohesion. We have used many helper classes for example for managing the sound or controlling the gif image and different Enums to control the stage of the whole game. In many cases we had to deal with the use of same classes and that is why we have tried to write the classes generic as possible to understand and reuse the code for the future.

2. Functionality

Game start with the opening screen where the player is given certain instructions before the game starts. Player can control the bat at the bottom of screen with the arrow keys represented as “left” or “right”, when the game starts a pumpkin moves towards the player part of the screen, where the player must move the paddle to bounce the pumping off against randomly moving bone in the upper part of the screen. Self-moving bone paddle was made to randomly spawn and move from left border of the playfield to the right. Paddle bone that reaches right border is afterwards spawned to another random location starting from the left.

When the bone is hit with the pumpkin, the bone takes damage and starts to break, after getting hit three times, it gets destroyed and a new one spawns which restores one extra life for the player. Level and speed of the ball is increased every tenth time that the moving bone is hit. Each time that pumping hits the bottom of the screen player loses one life and if the player life’s reaches zero game ends.
Game over screen is displayed afterwards with highest score that the player made and call to action for restarting the game is shown. Each time the game is restarted new game is created again and the player can try to beat their previous highest score.

Pong game has different variety of sounds implemented such as gameplay soundtrack, bone cracks or ending sound when the player loses the game.

Resolution class

The Resolution class extends the World and helps contribute to low coupling and high cohesion. The Resolution class; sub classes the IntroWorld and PingWorld.

```
import greenfoot.*;

public class Resolution extends World
{
    protected static final int WORLD_WIDTH = 500;
    protected static final int WORLD_HEIGHT = 700;

    /**
     * Constructor for objects of class Resolution
     */
    public Resolution()
    {
        super(WORLD_WIDTH, WORLD_HEIGHT, 1);
    }

}
```

GameState

Main game state controlled by premade Enums which allowed us to define different stages of the game in the act method.

```
public enum GameState
{
    PLAYING,
    LOST,
    NOT_PLAYING
}
```

Health bar

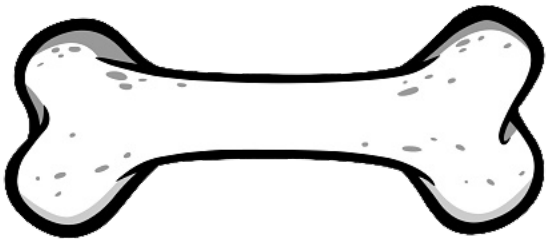
Health bar functionality gives the player indication how much life they have remaining. We solved this implementation with 3 different pictures, that updates depending on the player health.

```
private void updateImages(){
    health = getHealth();
    if (health >= 3)
        {
            setImage("heart3.png");
            getImage().scale(90,90);
        }
    if (health <= 2)
        {
            setImage("heart2.png");
            getImage().scale(90,90);
        }
    if (health <= 1)
        {
            setImage("heart1.png");
             getImage().scale(90,90);
        }
}
```

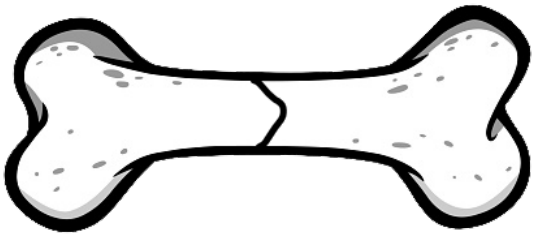
3 stages of the CPU paddle

One of the graphical enhancements we included in our game, were 3 stages cracking of the bone image that the CPU paddle uses. This brought a visual representation of the platform and indicated game progress for the player, and made it feel like the player is dealing damage to the CPU paddle.

1st stage:



2nd stage:



3rd stage:

SoundManager

Helper class that we created and called SoundManager was a helpful implementation that allowed us to add, update or manipulate sound throughout our code base.

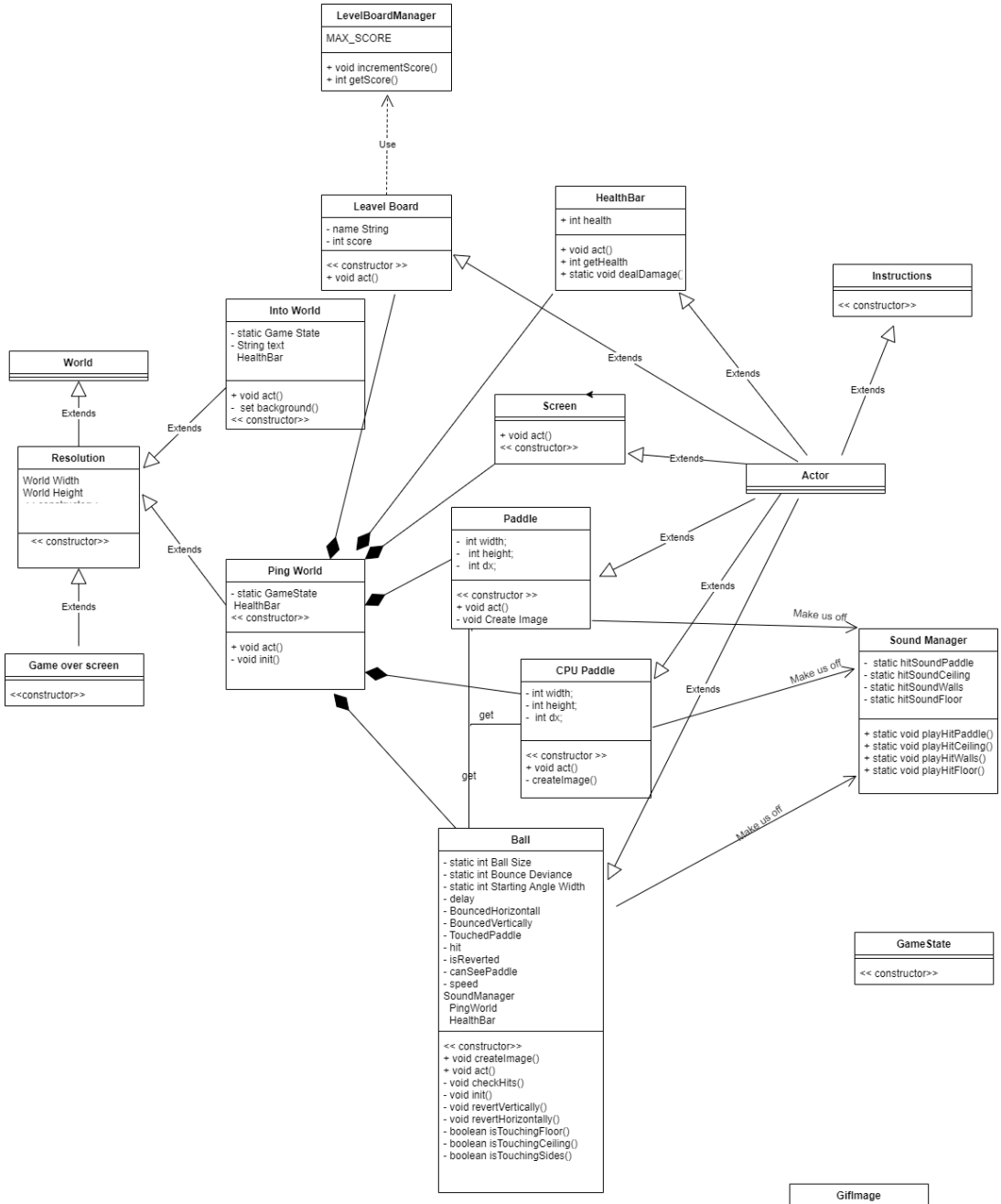
```
public class SoundManager
{
    private static GreenfootSound hitSoundPaddle;
    private static GreenfootSound hitSoundCeiling;
    private static GreenfootSound hitSoundWalls;
    private static GreenfootSound hitSoundFloor;
    private static GreenfootSound startSound;
    private static GreenfootSound boneCrack;
    private static GreenfootSound gameSound;
    private static GreenfootSound gameOverSound;


    static {
        try{
            hitSoundPaddle = new GreenfootSound("oof.mp3");
            hitSoundWalls = new GreenfootSound("bounce.mp3");
            hitSoundFloor = new GreenfootSound("wrong.mp3");
            startSound = new GreenfootSound("start_music.mp3");
            boneCrack = new GreenfootSound("bone_crack.mp3");
            gameSound = new GreenfootSound("theme_music.mp3");
            gameOverSound = new GreenfootSound("game_over.mp3");
        }catch(Exception e){
            System.out.print(e);
        }
    }
```

3. Internal design

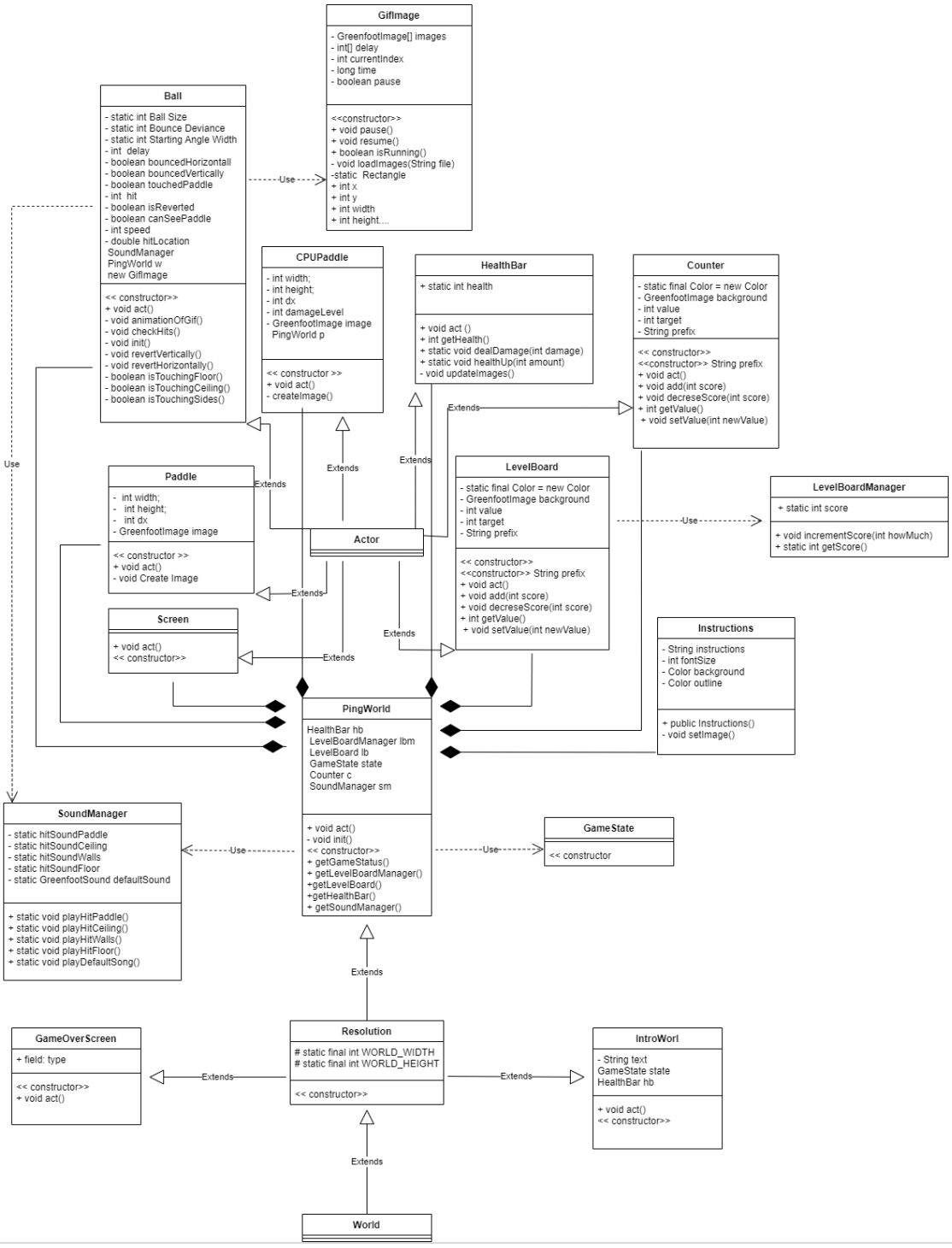
Version 1.0

First version had starting code base where we diagrammed the beginning code and classes that we have used in the program. Whole diagram seemed to be visually overwhelming which we knew in the future will create unstructured representation of the whole diagram.



Version 2.0

After reviewing the code and previous UML diagram. We decided to remake our diagram and focus properly on the fundamentals of Object-Oriented programming and the structure of the diagram itself. Throughout the work on the code base, we made changes that allowed us to lower coupling in some parts and increase cohesion in the whole project.



4. Evaluation

The main project task was to make the pong game as close as we could to the project given mandatory tasks and practise what we have learned so far at this education.

We paid a lot of attention to the project scope and time that was given, and, in that order, we were able to deliver working game with complete design.

Regarding the game functionality we would have looked more into making classes more generic, added different levels or store the highest player score that the player can see and compete against other players that will play our game.

For the next time we would use more logical mathematics to calculate different bounce angle when the ball hits the players paddle, or the SmoothMover class within the Greenfoot library, that works with more precise calculations. We have made the game completely functional and each of us have earned valuable experience, whenever it was working in the team or practising Object-Oriented programming in java in the Greenfoot environment.