Table of contents:

1. **Overview**
2. **How it works**
3. **Main Class**
4. **Player Class**
5. **Arrow Class**
6. **Redballoon Class**
7. **Yellowballoons Class**
8. **Score_Manager Class**
9. **Challenges**
10. **Team's members and their responsibilities**

1. Overview:

The Balloon Pop game is a simple interactive game where the player controls an archer to shoot arrows and pop balloons. The game consists of multiple levels, each with its own set of challenges and objectives. The player's goal is to pop all the balloons using a limited number of arrows to progress through the levels.

2. How it works

Starting Screen:

When you start, you will see a screen introducing the game. Then, the gameplay begins.

Controls:

- Move the archer by moving your mouse.
- Right-click to show the arrow.
- Left-click to fire arrows.
- Left-click to choose click the buttons.
- You can fire multiple arrows simultaneously.

Level 1:

- You are starting with this level.
- You are given an archer with 20 arrows.

- There are 15 red balloons rising up in a straight line with a constant speed.
- Your goal is to pop all the red balloons.
- When an arrow hits a balloon, it pops and shows a pop image and immediately goes, and your score increases.

Level 2:

- You are given an archer with 20 arrows.
- You face a new challenge with 15 red balloons and 3 yellow balloons.
- The red balloons positioned randomly and rising up with a constant speed.
- The yellow balloons are positioned randomly and its rising speed increases gradually.
- There are an element of surprise (apple), it may appear or not and has actions different from the ordinary arrows.
- Your goal is to pop all the balloons, both red and yellow.
- When an arrow hits a balloon, it pops and shows a red pop image for the red balloon, a yellow pop image for the yellow balloon and immediately goes, and your score increases.

Winning and Losing:

- If you run out of arrows before popping all balloons, you lose.
- If you lose at any level, you will have one option: replay Level 1.
- If you win at any level, you will have two options: replay Level 1 or proceed to Level 2.

Enjoy the game and aim for victory!

3. Main Class:

- Represents the main entry point of the bows and arrows game. It extends the “PApplet” class, which is the base class for all Processing sketches and it handles the logic of the game.
- Manages the game's main loop, setup, and drawing.
- Handles user input and game state transitions.
- Maintains instances of other game elements.
- Loads images for backgrounds and game elements.
- Updates game state based on player actions and balloon movements.

Key methods and variables:

- “setup()”: Initializes game elements and loads images.
- “draw()”: Implements the game loop, updates game state, and renders graphics.
- “mousePressed()”: Handles user input for shooting arrows and transitioning between game levels.
- “level1()” and “level2()” (in Score_Manager): Calculates the score based on remaining arrows and popped balloons for each level.

Start:

“slide4.png” (starting screen) will appear for 3 seconds.

Entering Level 1:

- The game sets lev1 to true, indicating Level 1.
- Background image slide1.png (background1) appears.
- It prints "Level 1" at position (252, 681).
- The initial arrow count is set to 20 (arrowCounter = 20), and it prints "Arrows: 20" at position (1035, 681).
- The initial score is set to 0, and it prints "Score: 0" at position (631, 681).
- start is true, so it adds 15 red balloons to the ArrayList from the Redballoon class.

Drawing Balloons:

- A for loop iterates over the ArrayList of red balloons, drawing a straight line of 15 red balloons at the bottom of the screen.
- Each iteration updates the position of the balloons using the update() method, causing them to rise up.
- If a balloon reaches the top of the screen, the edge() method makes it start again from the bottom.

Drawing the Archer:

- Initially, the archer without the arrow is printed because flag is true.

Aiming with the Right Button:

- If you click the right button, the archer with the arrow will be printed, and button will be set to true.

Firing Arrows with the Left Button:

- If you click the left button after clicking the right button:
- An arrow will be added to the ArrayList and fired.
- The arrow counter decrements by one.
- button is set to false.
- The archer with the arrow will be replaced by the archer without the arrow.
- If you clicked the left button without clicking the right button before there is not something will be happened because “button” = false

Arrow Count Update:

- It prints "Arrows: 19" after firing an arrow.

Collision Detection:

- When an arrow hits a balloon, the Checkcollision() method in the Redballoon class detects the collision.
- If a collision occurs, the balloon pops (pop() method is called), and the score increments, by this equation: Score = (Remaining Arrows + 1) * Number of Shot Balloons.

Level Completion Check:

- If all red balloons are popped (rbal ArrayList is empty), it signifies the completion of Level 1.
- The game updates the screen to display a level completion message and the current score.
- The “levelUp” flag is set to true to allow the player to proceed to the next level.

Level Transition:

- After completing Level 1, the player is presented with options to replay Level 1 or proceed to Level 2.
- Mouse input is used to handle the player's choice (mousePressed() method).
- If the player chooses to replay Level 1, the game resets Level 1 parameters.
- If the player chooses to proceed to Level 2, Level 2 parameters are initialized, and the game transitions to Level 2.

Game Over:

- If the player runs out of arrows before popping all balloons, it triggers a game over scenario.
- The game displays a "Game Over" screen with the option to replay Level 1.
- The player can choose to replay Level 1 by clicking the corresponding button.

Game Completion:

- If the player successfully completes all levels, a "You are the winner" screen is displayed.
- The player can choose to replay Level 1 or Level 2.

Scoring System:

- The Score_Manager class manages the game's scoring system.
- It calculates the score based on the number of arrows remaining and balloons popped for each level.

Surprise Element (Apple):

- Occasionally, an apple may appear during gameplay as a surprise element.

Level 2:

It works as same as level 1.

4. Player Class

This code defines a Player class for a Processing application. The Player class encapsulates the functionality and attributes of a player character in a graphical application. The class utilizes Processing's PApplet methods and properties to handle graphics rendering and interaction.

The class maintains several static fields including y which sets an initial vertical position for the player, arrowCounter which seems to track the number of arrows, and usedy which stores the vertical position used in rendering. It also includes a picture boolean flag that determines which image to use for the player's representation.

In the constructor, Player(PApplet processing), the PApplet instance is received and stored in processing, allowing the class to use Processing methods for rendering.

The draw() method handles the visual representation of the player. It uses the picture flag to decide between two images representing different states of the player ("girl_arrow.png" if picture is true, "girl_not (2).png" otherwise). The chosen image is resized to 100x100 pixels and displayed at a fixed x-position shifted by 25 pixels, and at a y-position given by usedy.

The mouseDragged() method adjusts the player's y-coordinate (y) when the mouse is dragged, increasing or decreasing it by 5 units depending on whether the mouse is below or above the current y-position. The method uses updateUsedY() to sync usedy with the modified y and forces a redraw of the screen with processing.redraw(), ensuring the player's position updates visually during the drag.

Overall, this class focuses on managing a player character's graphical representation and response to mouse interaction, using Processing's capabilities to manipulate images and respond to user input.

5. Arrow Class

This code defines a class named Arrow which is designed to be used in a Processing environment . The Arrow class manages the behavior and state of arrow objects within a game or application.

Each Arrow object include x and y coordinates that dictate its position on the screen, a Boolean Isactive to determine whether the arrow is active, and two image objects, arr and surprise, which are graphical representations of the arrow and an alternate object (an apple), respectively

The constructor of the Arrow class requires a PApplet object (which represents the Processing sketch) and sets the arrow's initial y position based on a static variable from the Player class

The update() method controls the arrow's behavior each time it's called. It moves the arrow horizontally across the screen by incrementing its x value by 10 units (controlled by dx). The arrow's image (arr or surprise) is drawn at an updated position, depending on whether a special mode (surpriseFired) is activated and a flag state. If surpriseFired is true and flag is false, the surprise image is used instead of the arr. Once the arrow reaches a horizontal position greater than 1220 pixels, it is deactivated (Isactive is set to false), reset to its initial position, and may change the flag status depending on the surpriseFired condition.

The firearrow() method allows for the activation of the arrow if there are available arrows to fire (checked by Player.arrowCounter).

Overall, this class encapsulates the functionality of an arrow in a game-like environment, managing its movement, appearance, and state based on game logic conditions and player interactions.

Redballoon Class:

Description:

The RedBalloon class orchestrates the presentation of red balloons, catering to both Level 1 and Level 2 scenarios. Each red balloon is represented as an independent object stored within an ArrayList.

Variables:

- "red": Stores the image of a red balloon.
- "pop": Holds the image representing the balloon popping.
- "yspeed1": Dictates the ascent speed of balloons in Level 1.
- "yspeed2": Governs the ascent speed of balloons in Level 2, set higher for added difficulty.
- "x2": Determines the x position of each balloon in Level 2, randomized.
- "y2": Establishes the initial y position of each balloon in Level 2, also randomized.
- "x1": Represents the x position of balloons in Level 1, calculated as tempx = i * 55.

- “y1”: Sets the initial y position of balloons in Level 1.

Methods:

- “draw()”: Renders the red balloons based on the current level, using conditional logic to display them at appropriate positions.
- “update()”: Adjusts the y position of balloons, moving them upwards by subtracting the yspeed value.
- “edge()”: Manages y value adjustments to ensure balloons reappear at the bottom of the screen when reaching the top.
- “checkCollision()”: Determines if an arrow makes contact with a balloon, toggling a boolean check accordingly.
- “pop()”: Displays the 'pop' image when an arrow hits a balloon.

6. Yellowballoons Class

Description:

- YellowBalloon Class manages the presentation of yellow balloons within Level2. And it is the same of Redballoon Class (same methods and variables) but with some changes, here they are:
  - where each yellow balloon take an initial random speed and random acceleration
  - “update()”: adds acceleration to “yspeed” to accelerate
  - “edge()”: Handles “y” adjustments to ensure yellow balloons reappear at the bottom of the screen, also return “yspeed” to random value.

7. Score_Manager Class

8. Challenges

1. At the first we made two balloon classes for level1 and level2 as array not arraylist so we face a problem to remove the balloon that arrow touch it so we change all class and make it as array list that make remove balloons easy .
2. It was really challenging to choose the condition that checks the collision between the balloons and the arrows and took long time of thinking.

3. Then, we noticed that when you try to fire several arrows at the same time only one of them can pop the balloons and the others cannot. However, we solved this problem by calling the checkcollision method in the loop that makes the arrow so that the method is called for each arrow not one arrow.

10. Team's members and their responsibilities

- **Mennatallah Khalifa**

Player class that draw the archer and manage its motion, arrow class that draws arrows and responsible for its firing and movement.

- **Shahd Ayman & Sandy Khalil:**

Level 1 which draws red balloons that move in one line upward and responsible for popping the balloons to move to level 2.

- **Nada Hesham & Manar Yousry :**

Level 2 which draws red and yellow balloons that move randomly and responsible for popping the balloons to win the game.

All the team members share in coding the main class.