Bogazici University

Introduction to Object Oriented Programming

CMPE160

Assignment 1

Author:
Ibrahim Berkay Ceylan
Student ID: 2023400327

March 20, 2024

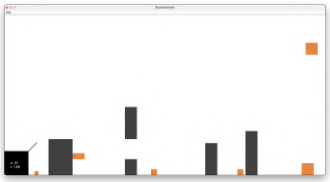
Contents

1 Introduction

This report details the implementation of the "Angry Bullet" game, a project given in the Introduction to Object Oriented Programming course, CMPE160. The primary objective of the project is to develop a simple copy of the game "Angry Bird" using Java and the StdDraw graphics library. The game's physics rely on simple projectile motion.

1.1 Game Overview

The game has a shooting platform which the player shoots a bullet from using the "SPACE" button. The user can use the arrow keys to change the direction and the velocity of the bullet. Orange blocks are targets, where if the player can hit them, he wins. The ball goes out of bounds if it hits the ground or goes to the right too much, and the bullet stops if it hits a black block. The player can then press "R" to restart the game and shoot again.



2 Implementation

The development of the Angry Bullet game was an exercise in applying the theoretical knowledge learned in classes and labs in a practical context, using Java programming language and the StdDraw graphics library. The key focus was to create a game that functions flawlessly, show accurate depictions of projectile motion, and show ability to write good practice Java code.

2.1 Development Process

The first step of the implementation was setting up the StdDraw graphics library. StdDraw is a simple graphics drawing library, perfect for a game like Angry Bullet. The game's screen, obstacles, along with the animation and the trajectory of the bullet were drawn using the StdDraw library.

The development process involved writing the code for the game's mechanics. This involved drawing the obstacles to the sensible coordinates, drawing the shooting platform, taking user inputs, animating the bullet according to simple projectile motion equations $x = x_0 + v_h t$ and $y = y_0 + v_v t - \frac{1}{2} g t^2$ where v_h and v_v are horizontal and vertical velocities respectively.

2.2 Challenges and Solutions

One of the significant challenges encountered during the implementation process was ensuring the initial bullet velocity and frame rate was set so that the player experience of the game is good. To achieve this I firstly fine-tuned the velocity by scaling it and used a decent frame rate. Then I used the gameplay video in the assignment and fine-tuned scaleVelocity coefficient and the pauseDuration by running the game over and over again, changing the coefficient and pauseDuration, saving the distance between approximately where the bullet landed on the video and on the game I ran on my computer, all autonomously. In the end, pauseDuration was set low so that the collision mechanic worked properly for long-lasting trajectories.

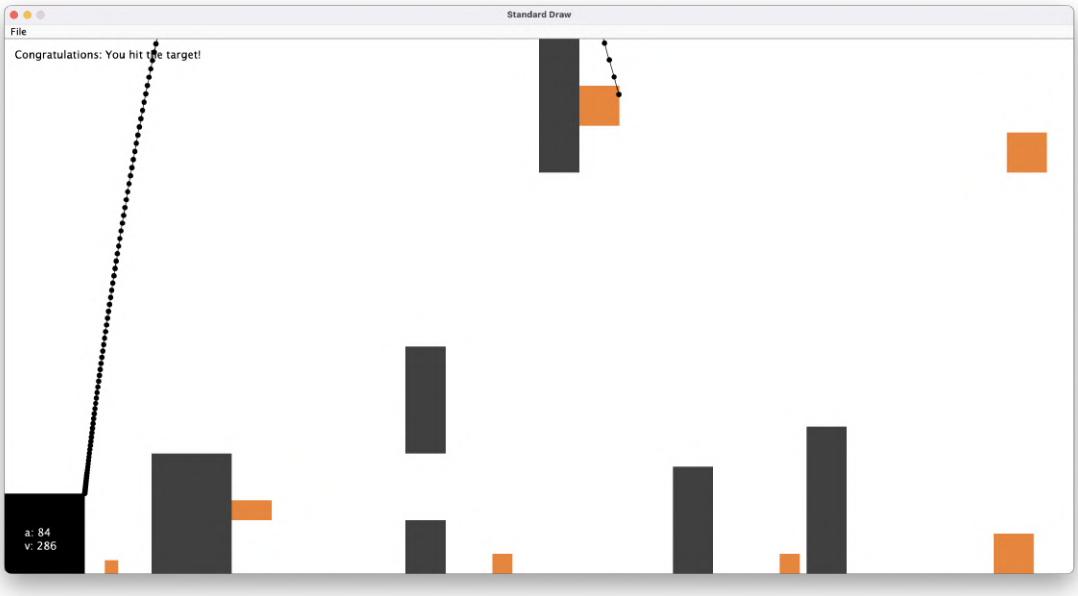


Figure 1: Long lasting trajectory for a custom target.

Another challenge was drawing the trajectory of the bullet. Since students are not allowed to use built-in Java functionalities until they are showed in class, ArrayLists were not allowed. So, I implemented an appendToArray method to simulate a dynamic array and saved the trajectory to an array.

3 Gameplay Mechanics

3.1 Objective

The objective of the game is to shoot the orange blocks placed at various places in the screen without hitting any obstacles. Player is counted as won if they can predict the trajectory of the bullet by setting the correct angle and velocity values using the game's controls to hit the targets.

3.2 Controls and Interaction

Player interacts with the game with their arrow keys. The up and down keys are used to adjust the direction which the bullet will be launched at, while the right and left keys are used to adjust the velocity which the bullet will be launched at. To start the shooting animation, the player presses the "SPACEBAR" button in their keyboard.

While the bullet is moving, its trajectory is displayed on the screen, marked by black dots connected with lines. This visual representation gives real-time feedback to the user to adjust their shooting. Figure 2 shows an example of how the trajectory of the bullet is drawn.

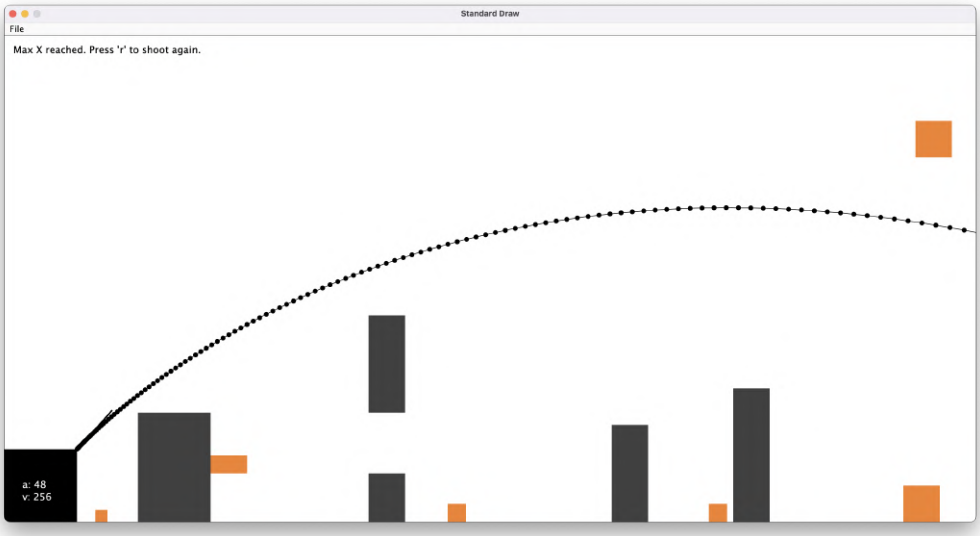


Figure 2: Trajectory of a bullet.

After each shot, player is given feedback on the outcome of their shootings at the top left of the game window. The results are categorized as, hitting the target, touching the ground, hitting and obstacle or going out of the maximum horizontal boundary of the game window. The game also allows for retrying, after each outcome, player can press the "R" key to reset and attempt a new shot.

4 Gameplay Screenshots

In this section, various scenarios encountered in the Angry Bullets game, each illustrated with a screenshot.

4.1 Hitting the Target

When the bullet successfully hits one of the targets, it gets counted as a successful shot. This scenario tests the player's ability to correctly predict the bullet's trajectory by changing the values of angle and velocity.

Figure 3: Successful hit.

4.2 Hitting an Obstacle

In cases where the bullet hits one of the grey colored blocks, the shot is considered unsuccessful. This adds a layer of complexity to the game with carefully placed obstacles to make hitting the targets harder, requiring players to plan a trajectory avoiding obstacles.

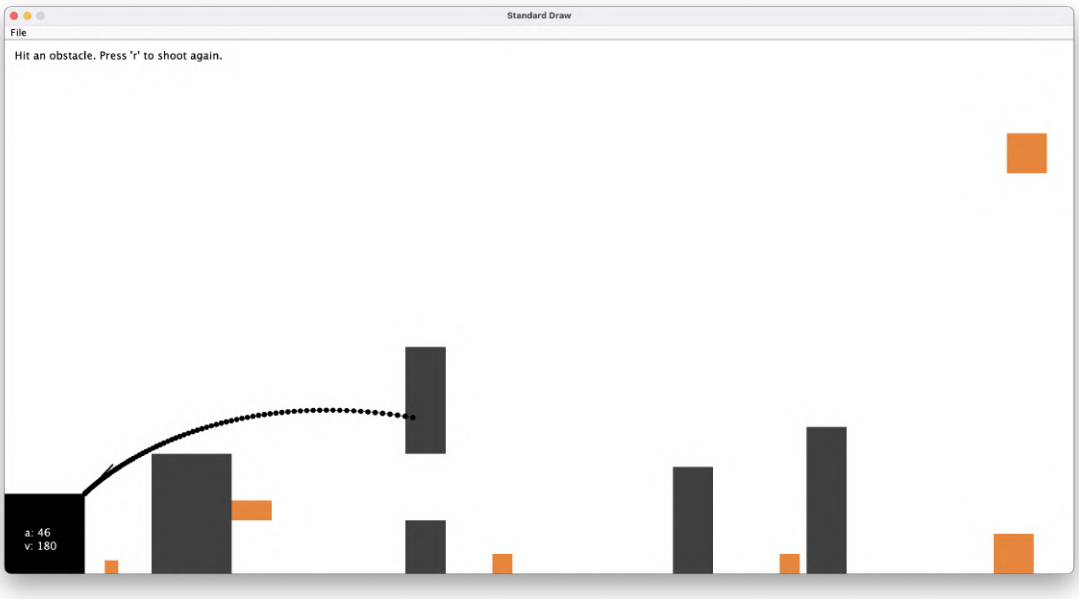


Figure 4: Hitting an obstacle.

4.3 Ground Contact

If the bullet touches the ground before hitting a target, the shot ends. This outcome is achieved when the prediction of the trajectory is incorrect, mostly encountered while trying high-shots.

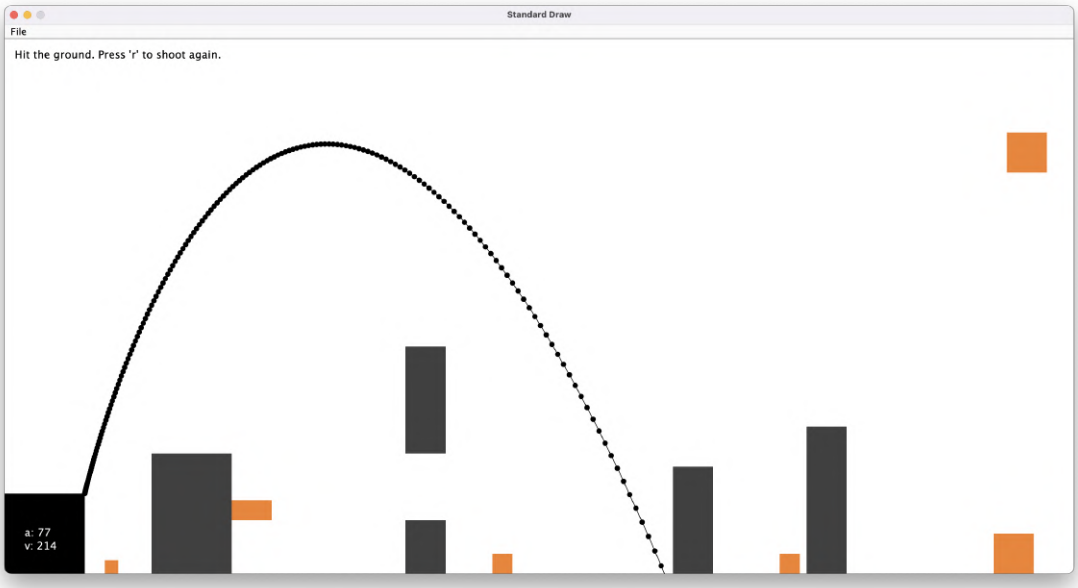


Figure 5: Ground contact.

4.4 Exceeding Horizontal Boundary

When the bullet exits from the right side of the game window, the shot ends. It shows the importance of not overshooting, and usually encountered while trying to shoot targets close to the boundary of the screen, or airborne targets.

Figure 6: Max X coordinate reached.

4.5 Bullet Through Sky

Although there is a maximum X coordinate the bullet can reach, the Y coordinate is not bounded. The bullet may go out of the screen by going through the upper sky and return back, adding another element of strategy to consider while trying to shoot targets. An example of a bullet going through upper sky, and coming back to hit a target may be seen in Figure 1 on page 3.