Angry Bullet Game

CMPE 160 Assignment 1

Salih Can ERER
2022400174

Boğaziçi University

Abstract

This report is prepared in order to explicitly detail the cases, which are provided in the PDF of the assignment on the Notion site, of the first assignment of the CMPE 160, Introduction to Object Oriented Programming Course, namely Angry Bullet Game. The game is written in Java; the StdDraw library is used. The game is about hitting the targets in a 2D environment without hitting the obstacles, by precisely adjusting the velocity and the angle of the bullet. The game involves basic physics concepts, such as gravity and projectile motion.

Example gameplay video may be accessed from here.

Keywords: Projectile Motion · Java · Physics Simulation · Collision Detection

Contents

1 General Information **1**

2 Example Gameplays **3**

2.1 Default Game Parameters 3

2.2 Custom Game Parameters 9

Chapter 1

General Information

This project involves the development of a game that is similar to the popular Angry Birds game. Java programming language and StdDraw graphics library are utilized in implementing our solution. This game, called "Angry Bullet" requires players to hit predetermined targets by shooting a bullet at them. Projectile motion, one of the fundamental principles of physics, can be considered as the central application of the gameplay.

The projectile's trajectory is determined by its initial velocity and launch angle, which are also the basis for fundamental physics equations:

$$x_t = x_0 + v_0 \cos(\theta)t$$

for the horizontal axis and

$$y_t = y_0 + v_0 \sin(\theta)t - \frac{1}{2}gt^2$$

for the vertical axis, where (x_0, y_0) represents the starting coordinates, v_0 the initial velocity, θ the launch angle, and g the gravitational constant.

Hence, when a player starts playing the game, he/she will see a setup with the bullet moving at 180 units per second and being launched at an angle of 45 degrees. This game's interface allows players to easily adjust these parameters on the go through intuitive keyboard controls: arrow keys. The "up" and "down" arrows are used for incrementing or decrementing angles while the "left" and "right" buttons help in changing the velocity of bullets. With such interactivity in mind, every shot can be carefully planned so that its path corresponds with the intended targets while avoiding any obstacles on its way.

Targets in orange rectangles and obstacles in dark grey rectangles are presented within a well-designed environment. The velocity and the angle of the bullet are shown on the launch pad box placed at the left bottom. Moreover, these variables are represented graphically by a dynamic shooting line that changes in size and direction as the player interacts with them. The bullet is projected and its movement is illustrated by a sequence of black lines and dots showing how it's being calculated.

Players aim at evading barriers and hitting targets, where every throw ends up in any of four possibilities; a target hit, obstacle collision, ground contact, or going beyond the right end of the window gaming area. Interestingly enough, bullets can go beyond the top border and appear again within the game field as provided by physical laws constituting another dimension of planning for shots. After every shot has been made players have an opportunity to try again by pressing the button 'r'.

Chapter 2

Example Gameplays

2.1 Default Game Parameters

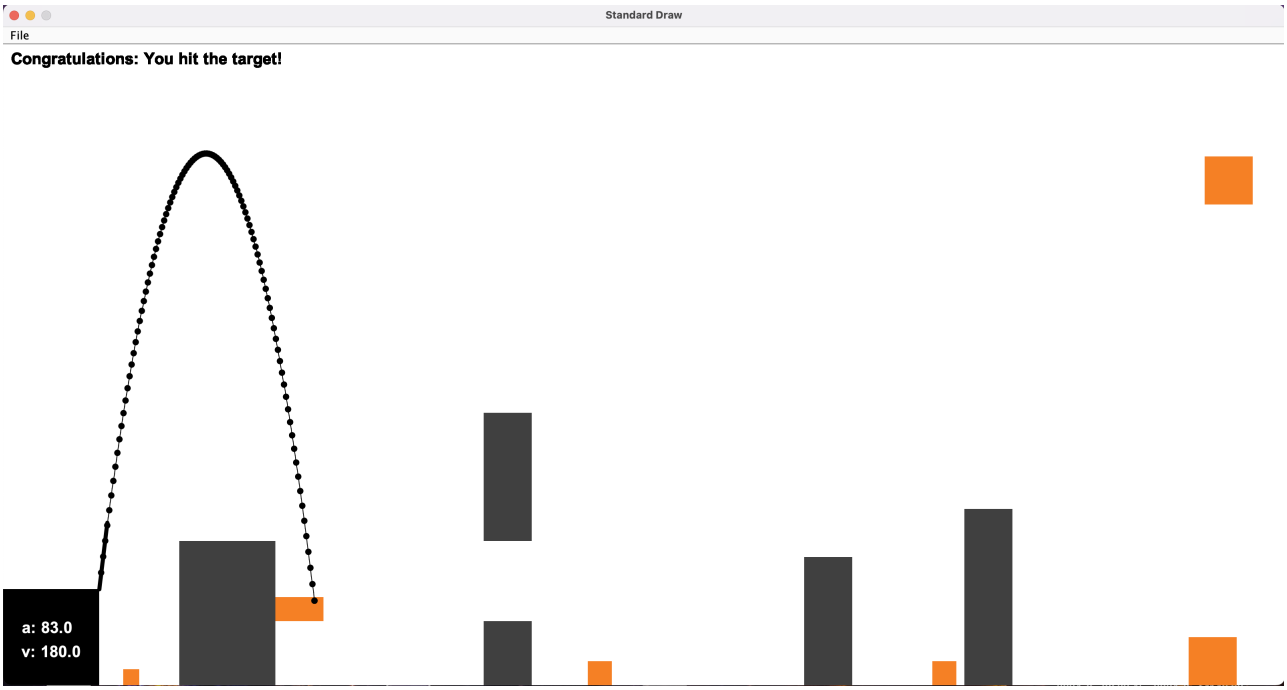
To begin with, we will examine the game for distinct scenarios using default game parameters for the obstacles and targets.

```
// Game Parameters
int width = 1600; //screen width
int height = 800; // screen height
double gravity = 9.80665; // gravity constant
double x0 = 120; // x-coordinate of the bullet's starting position
double y0 = 120; // y-coordinate of the bullet's starting position
double bulletVelocity = 180; // initial velocity
double bulletAngle = 45.0; // initial angle

// Box coordinates for obstacles and targets
// Each row stores a box containing the following information:
// x and y coordinates of the lower left rectangle corner, width, and height
double[][] obstacleArray = {
    {1200, 0, 60, 220},
    {1000, 0, 60, 160},
    {600, 0, 60, 80},
    {600, 180, 60, 160},
    {220, 0, 120, 180}
};

double[][] targetArray = {
    {1160, 0, 30, 30},
    {730, 0, 30, 30},
    {150, 0, 20, 20},
    {1480, 0, 60, 60},
    {340, 80, 60, 30},
    {1500, 600, 60, 60}
};
```

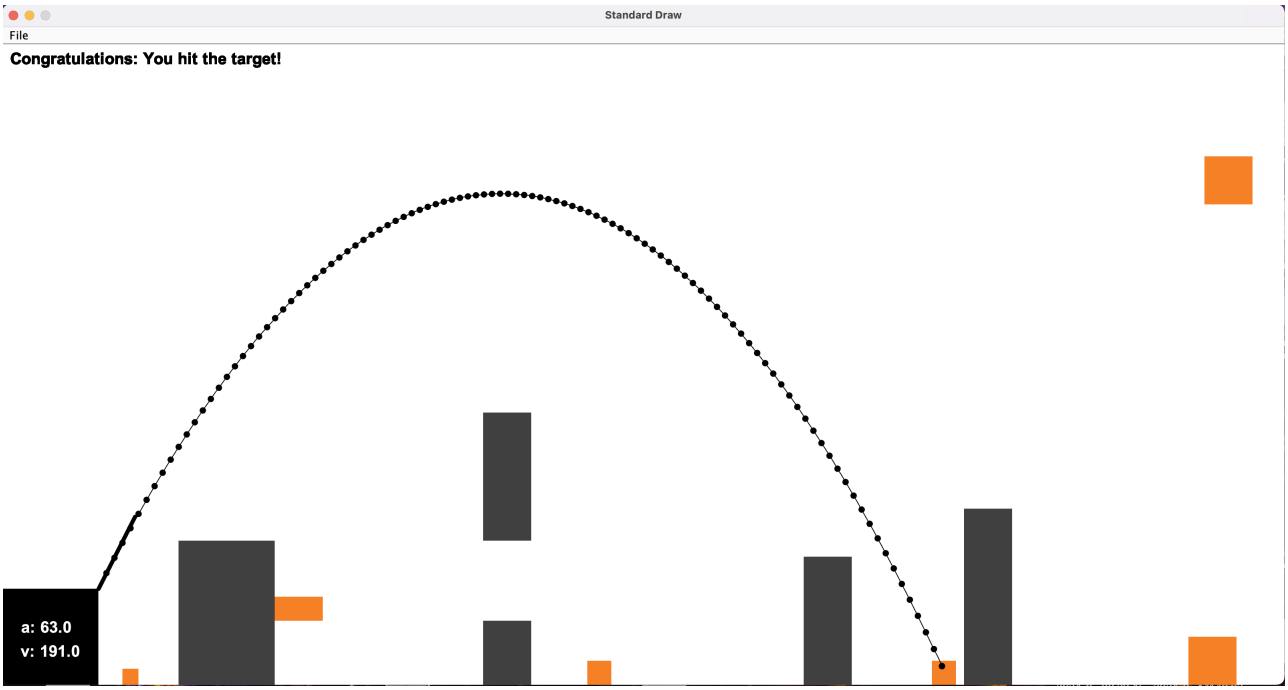
Listing 2.1: Default Game Parameters and Initialization

We will demonstrate five distinct cases, which have been mentioned on the assignment page.

- Case for hitting the target (orange box). The successful shot hits one of the orange targets.
- Case for hitting an obstacle.
- Case for ground contact. Shooting simulation should finish in this case.
- Case for Max X coordinate reach. Shooting simulation should finish in this case.
- Case for bullet going to the sky. The bullet is allowed to travel through the upper part of the sky.

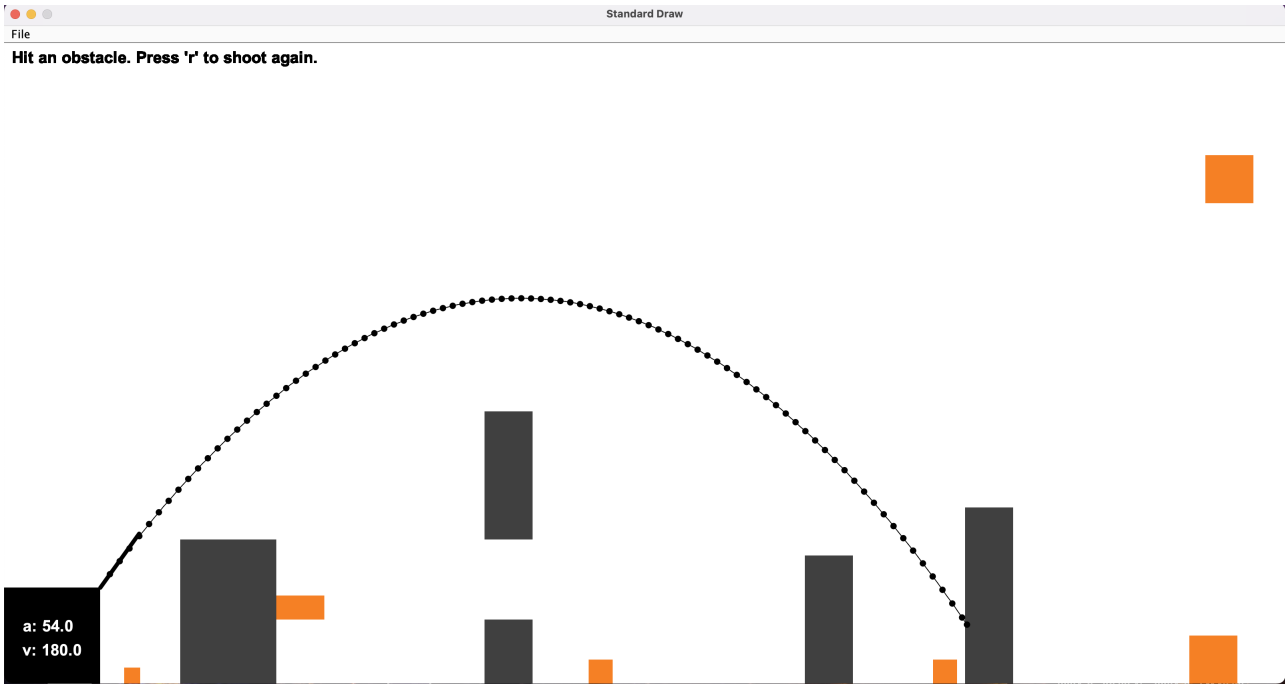


(a) Target Hit Example 1

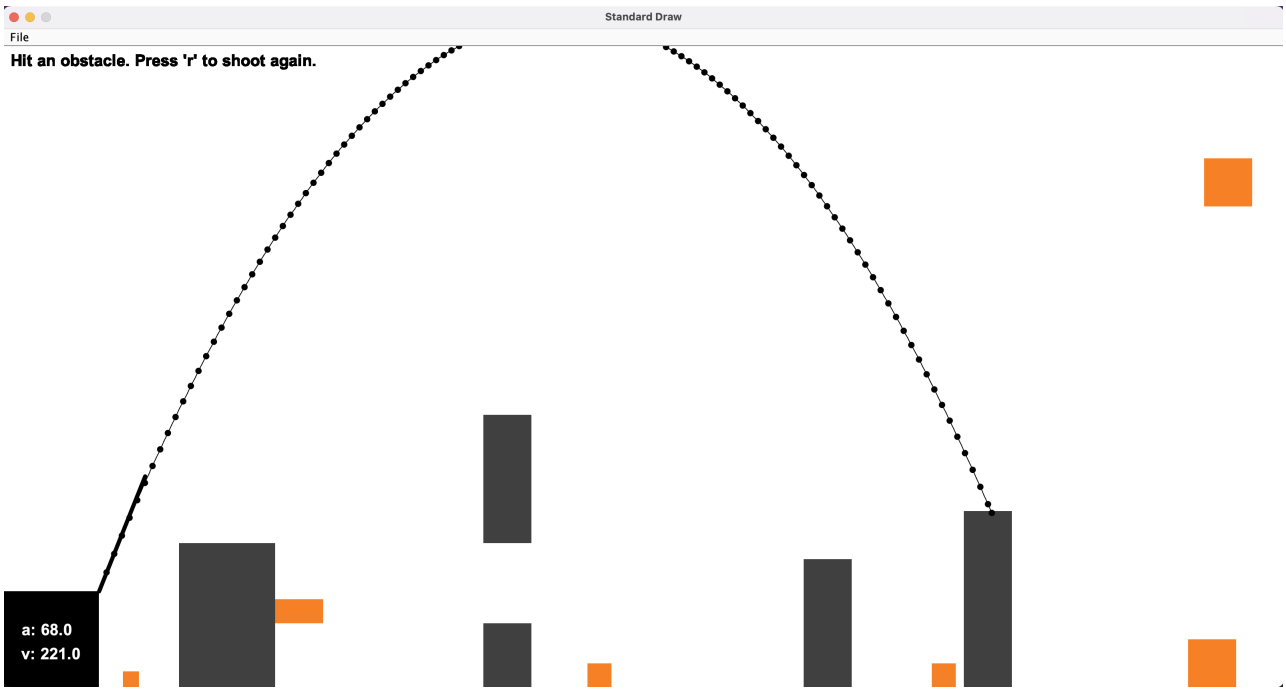


(b) Target Hit Example 2

Fig. 2.1: Illustrations of hitting targets with default parameters.

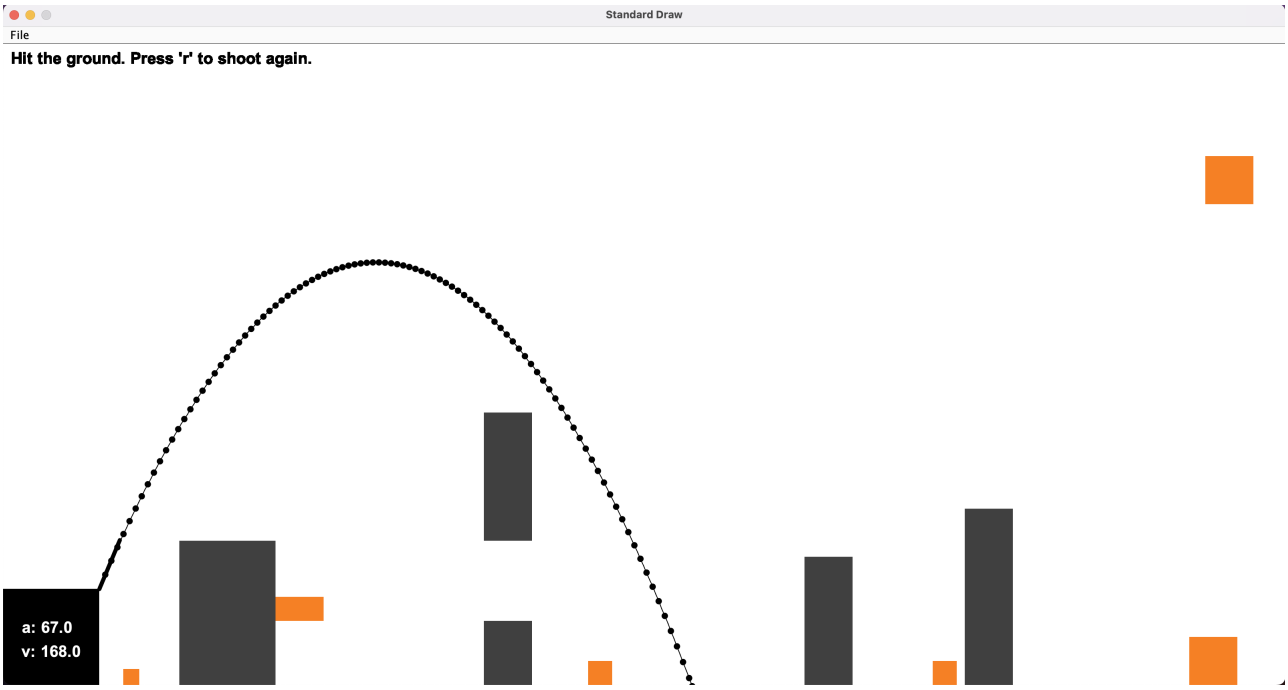


(a) Obstacle Hit Example 1

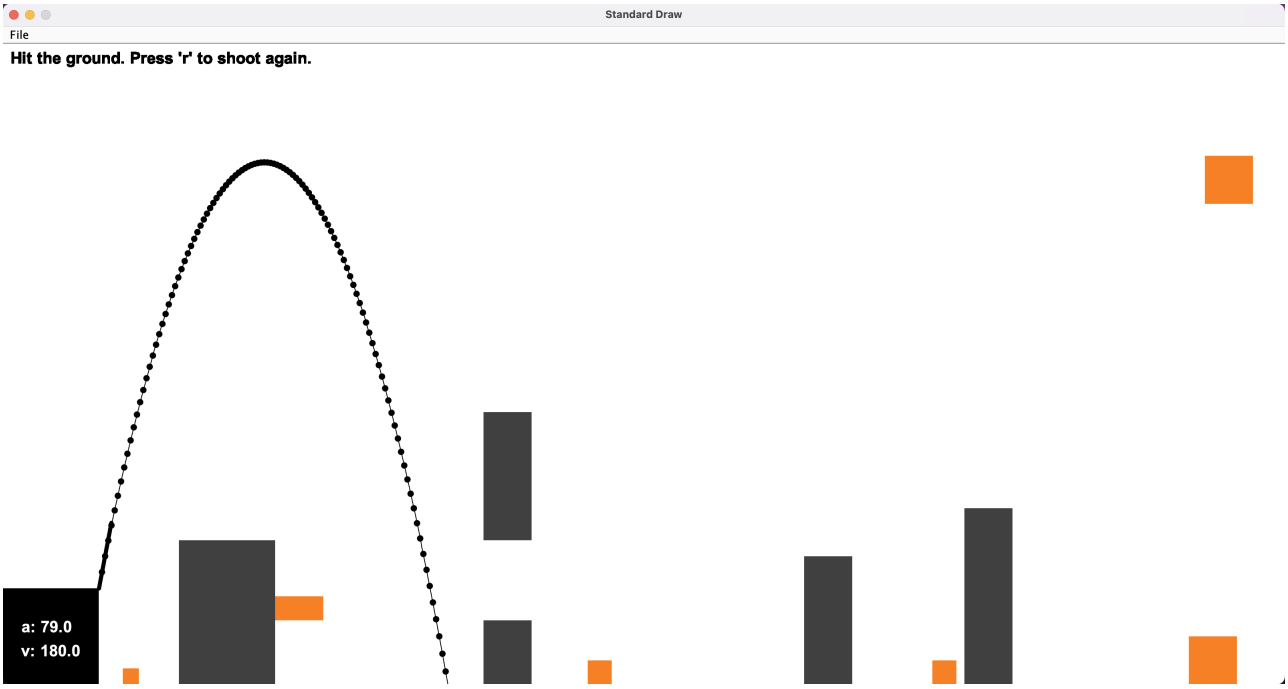


(b) Obstacle Hit Example 2

Fig. 2.2: Illustrations of hitting obstacles with default parameters.

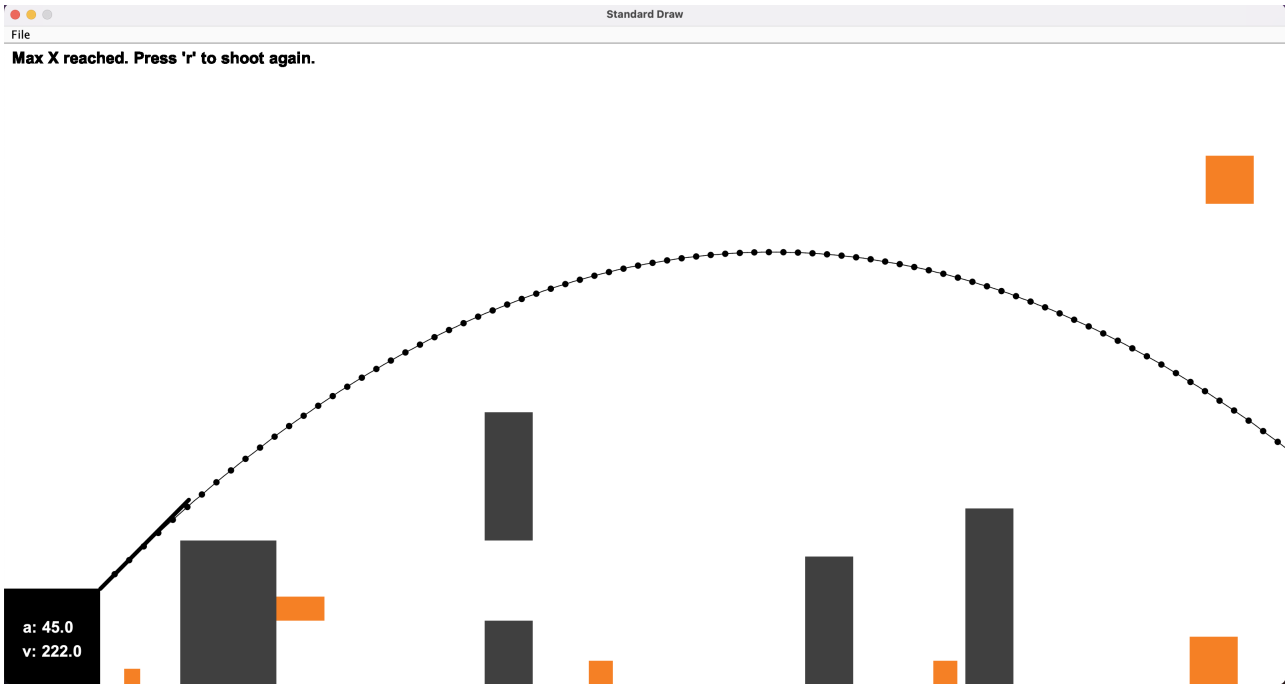


(a) Ground Contact Example 1

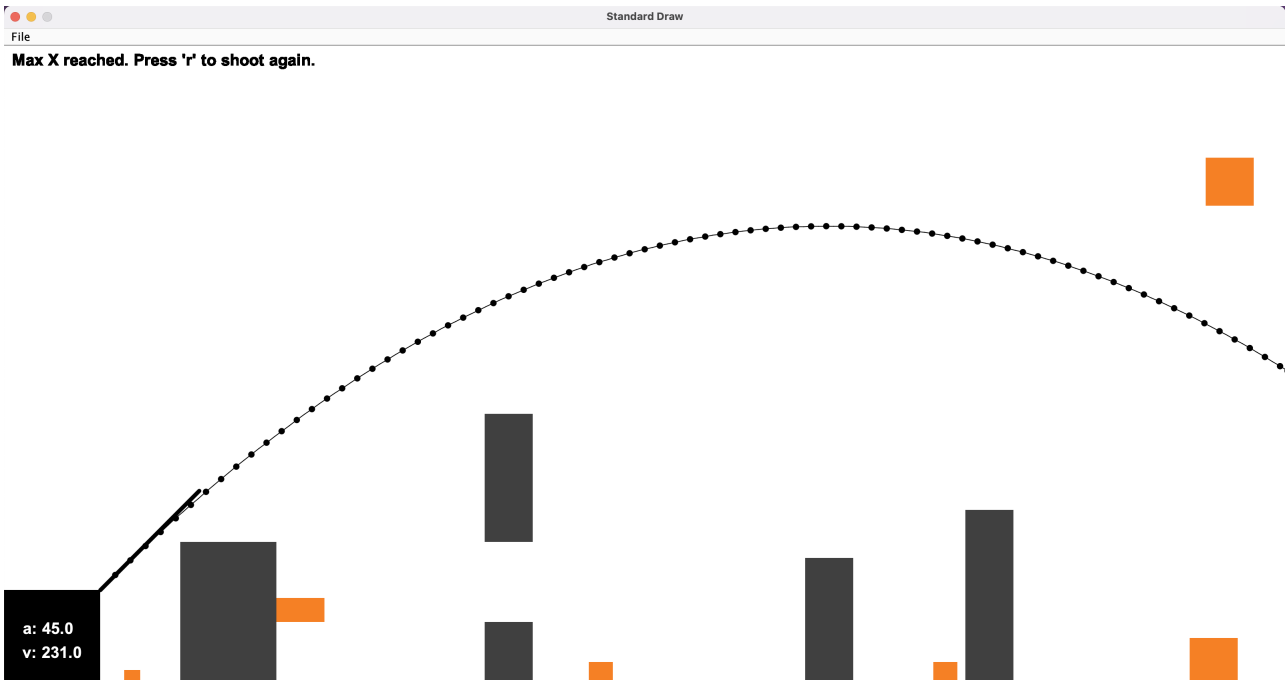


(b) Ground Contact Example 2

Fig. 2.3: Illustrations of ground contact with default parameters.

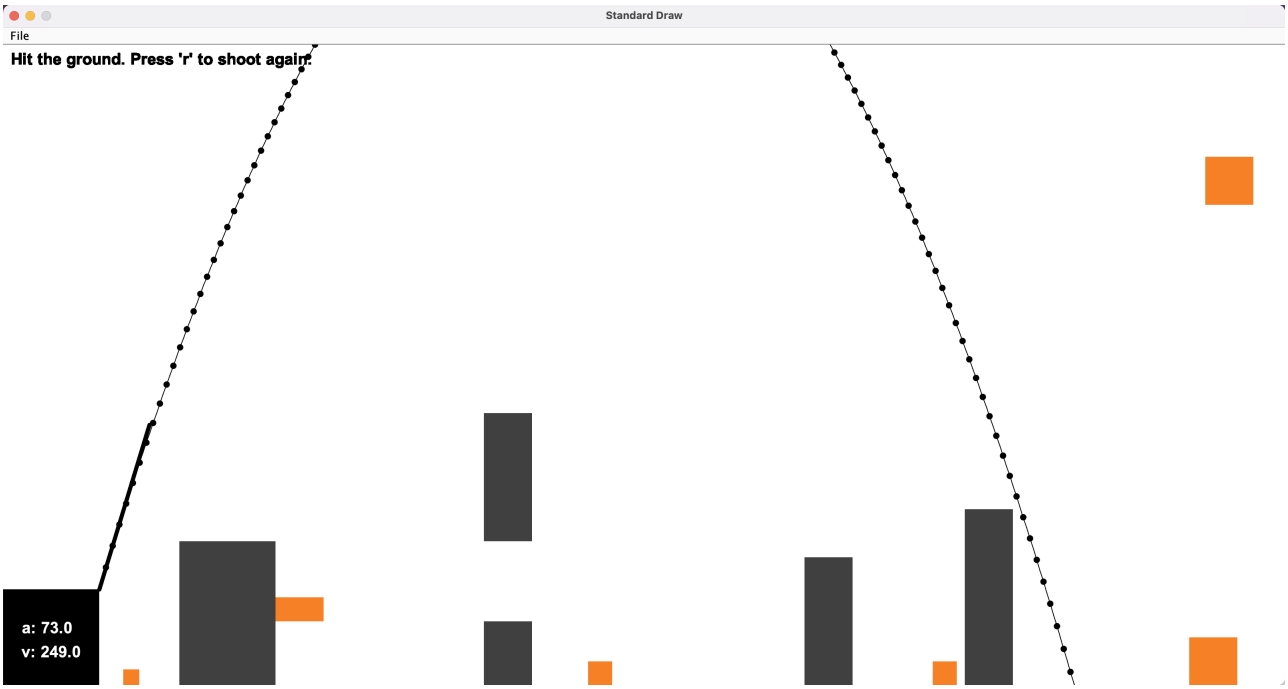


(a) Reaching Max X Coordinate Example 1

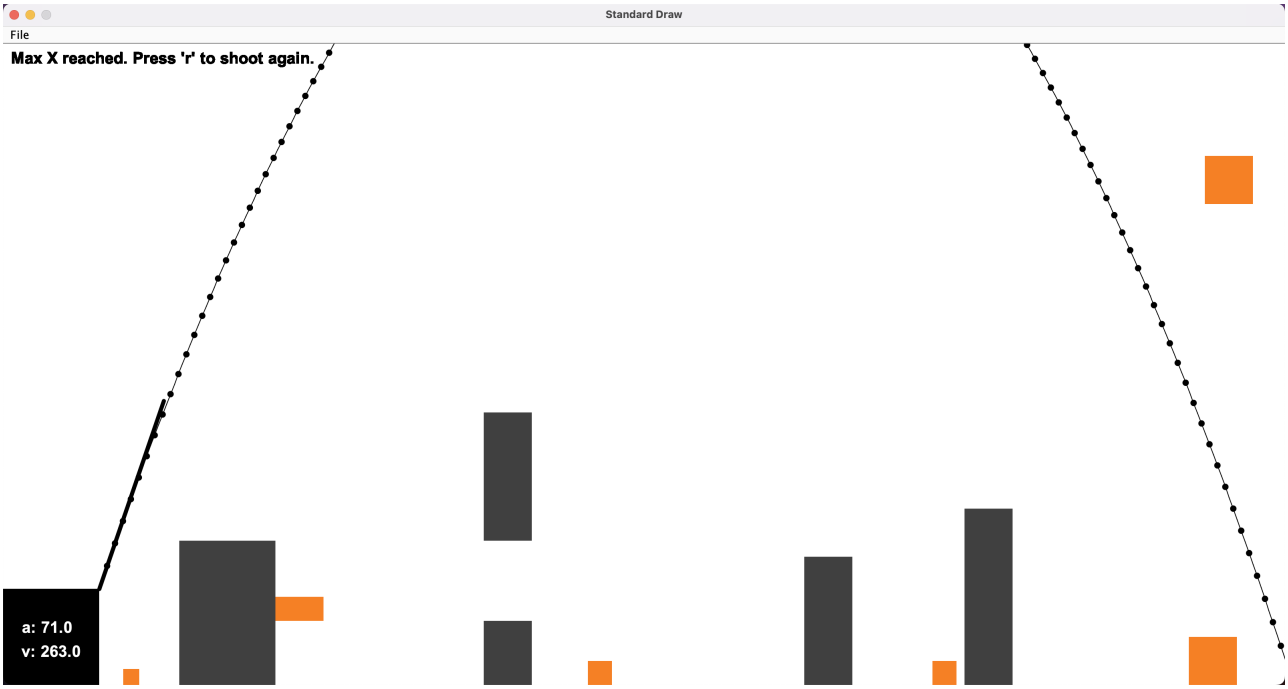


(b) Reaching Max X Coordinate Example 2

Fig. 2.4: Illustrations of reaching max X coordinate with default parameters.



(a) Bullet Going To The Sky Example 1



(b) Bullet Going To The Sky Example 2

Fig. 2.5: Illustrations of bullet going to the sky with default parameters.

2.2 Custom Game Parameters

In this section, we will examine the game for distinct scenarios using custom game parameters, which have been generated arbitrarily for the obstacles and targets.

```
int width = 1600; //screen width
int height = 800; // screen height
double gravity = 9.80665; // gravity constant
double x0 = 120; // x-coordinate of the bullet's starting position
double y0 = 120; // y-coordinate of the bullet's starting position
double bulletVelocity = 180; // initial velocity
double bulletAngle = 45.0; // initial angle

// Box coordinates for obstacles and targets
// Each row stores a box containing the following information:
// x and y coordinates of the lower left rectangle corner, width, and height
double[][] obstacleArray = {
    {1238, 388, 57, 48},
    {1297, 701, 94, 66},
    {896, 657, 31, 67},
    {1422, 297, 31, 58},
    {1172, 293, 33, 71}
};

double[][] targetArray = {
    {1360, 72, 26, 39},
    {1490, 247, 20, 15},
    {613, 690, 41, 25},
    {1526, 704, 33, 17},
    {956, 427, 47, 30}
};
```

Listing 2.2: Custom Game Parameters and Array Initialization

The game environment, which is generated arbitrarily, is given below.

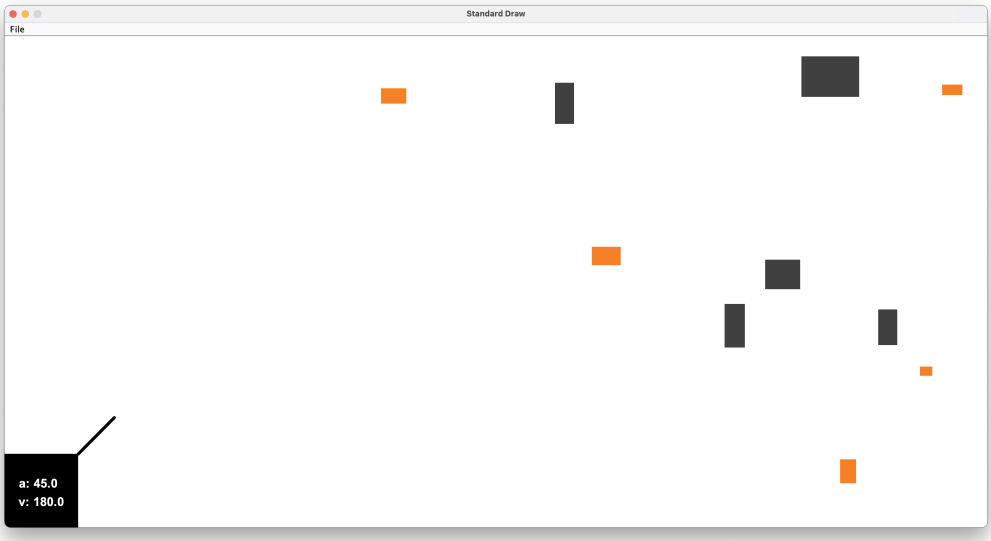
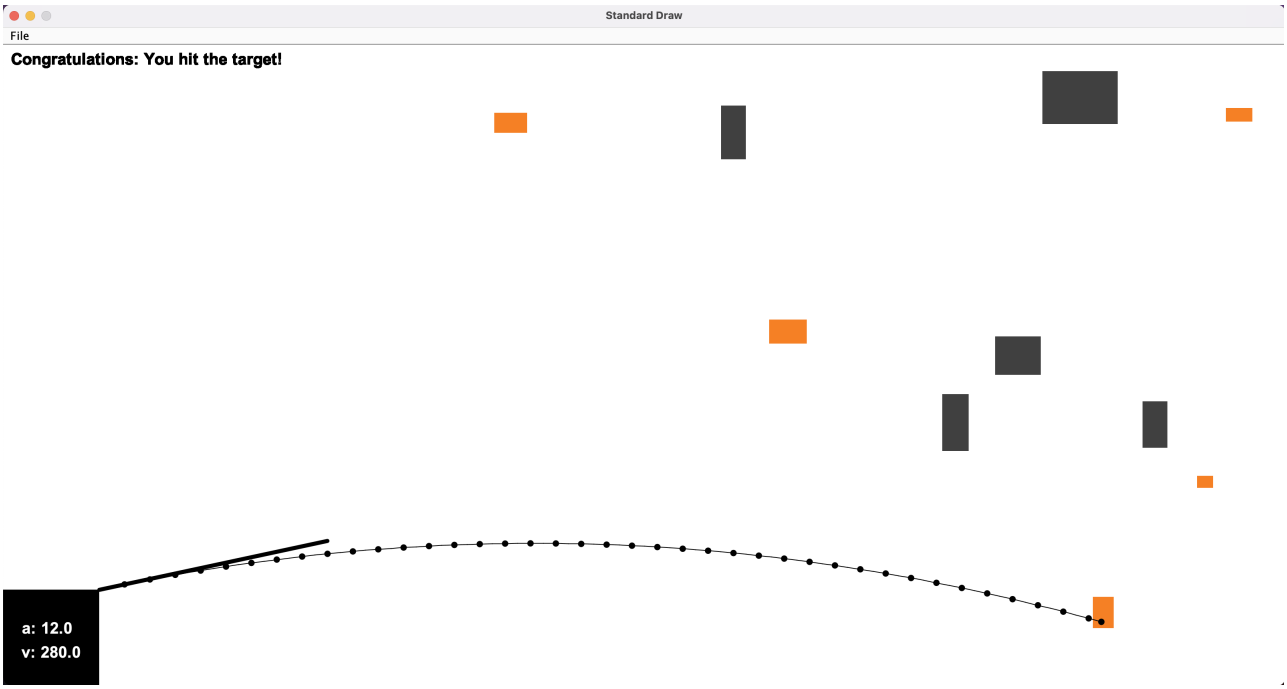
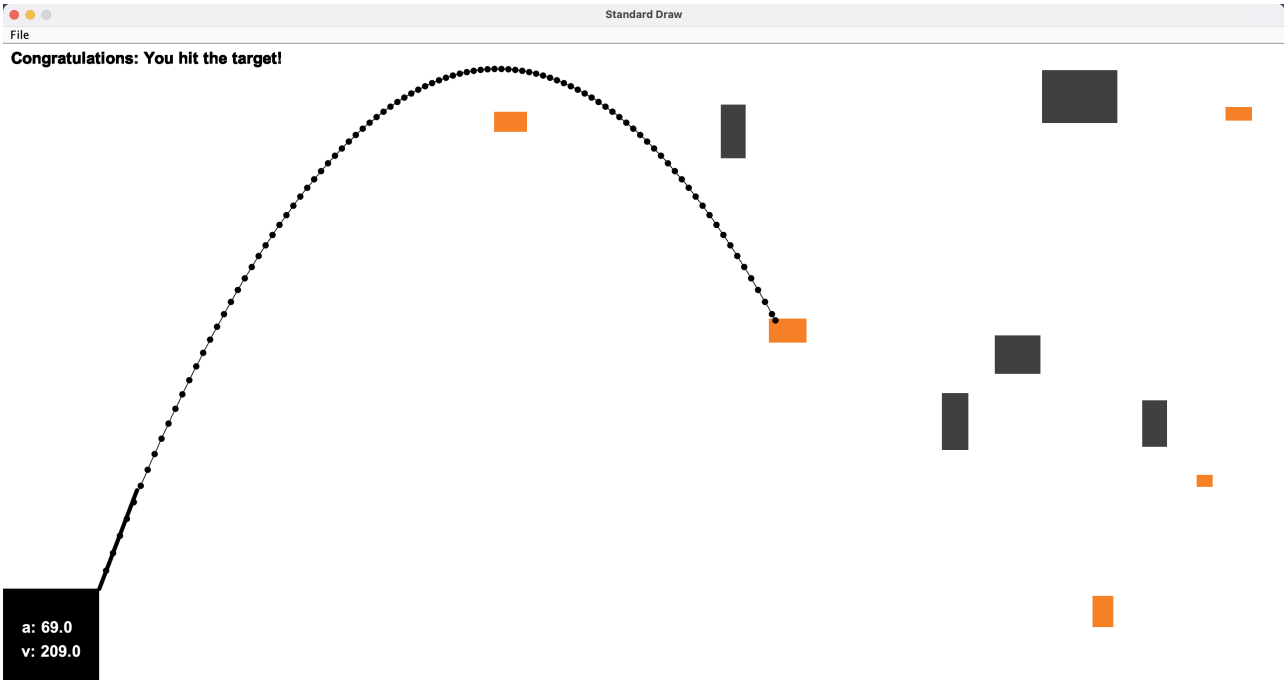


Fig. 2.6: Custom Game environment

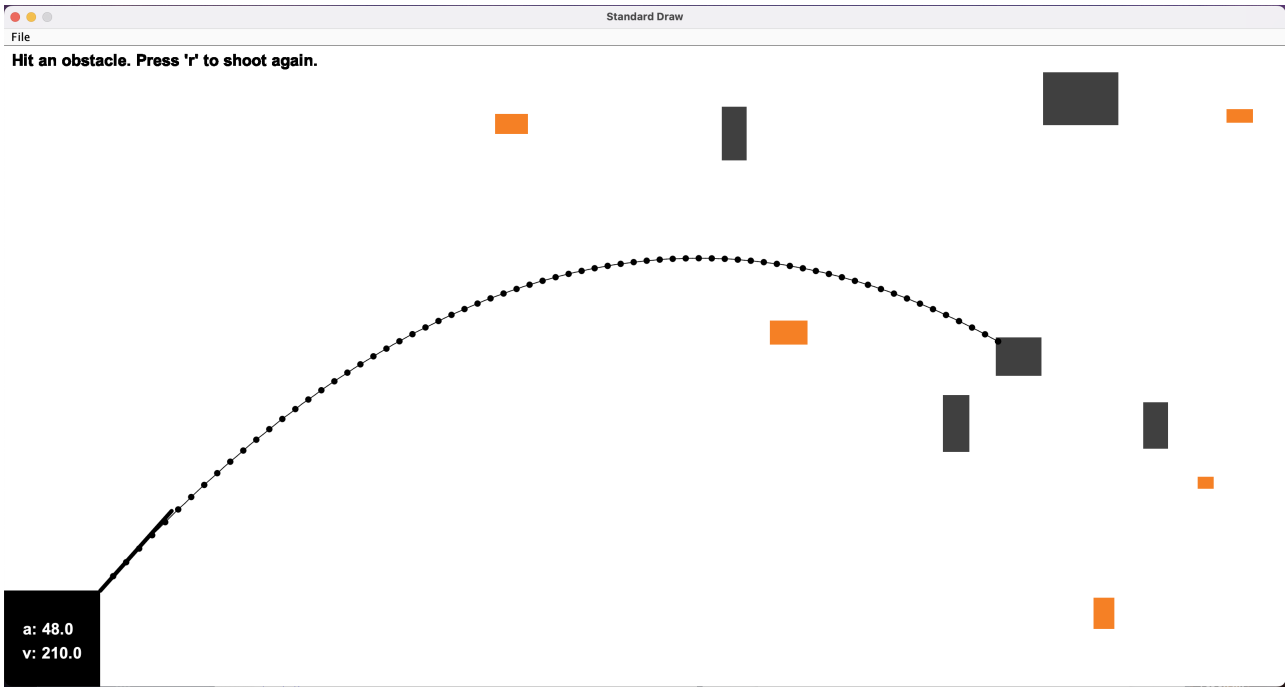


(a) Target Hit Example 1

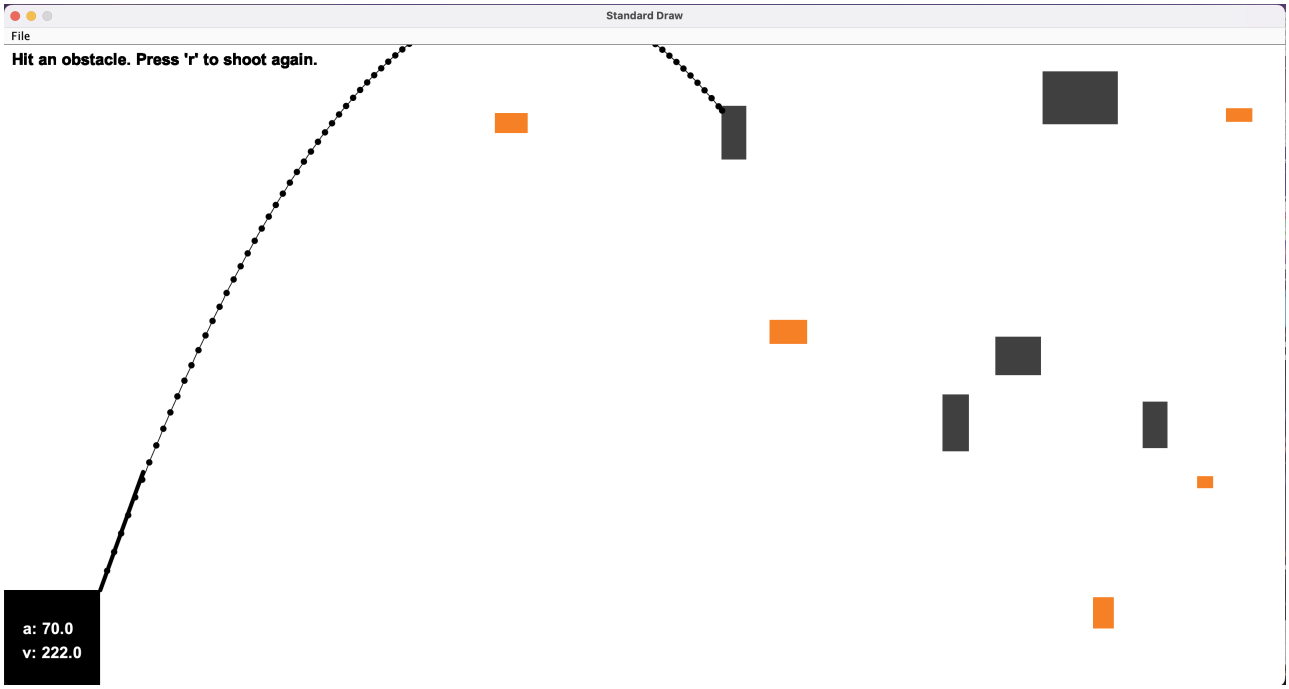


(b) Target Hit Example 2

Fig. 2.7: Illustrations of hitting targets with custom parameters.

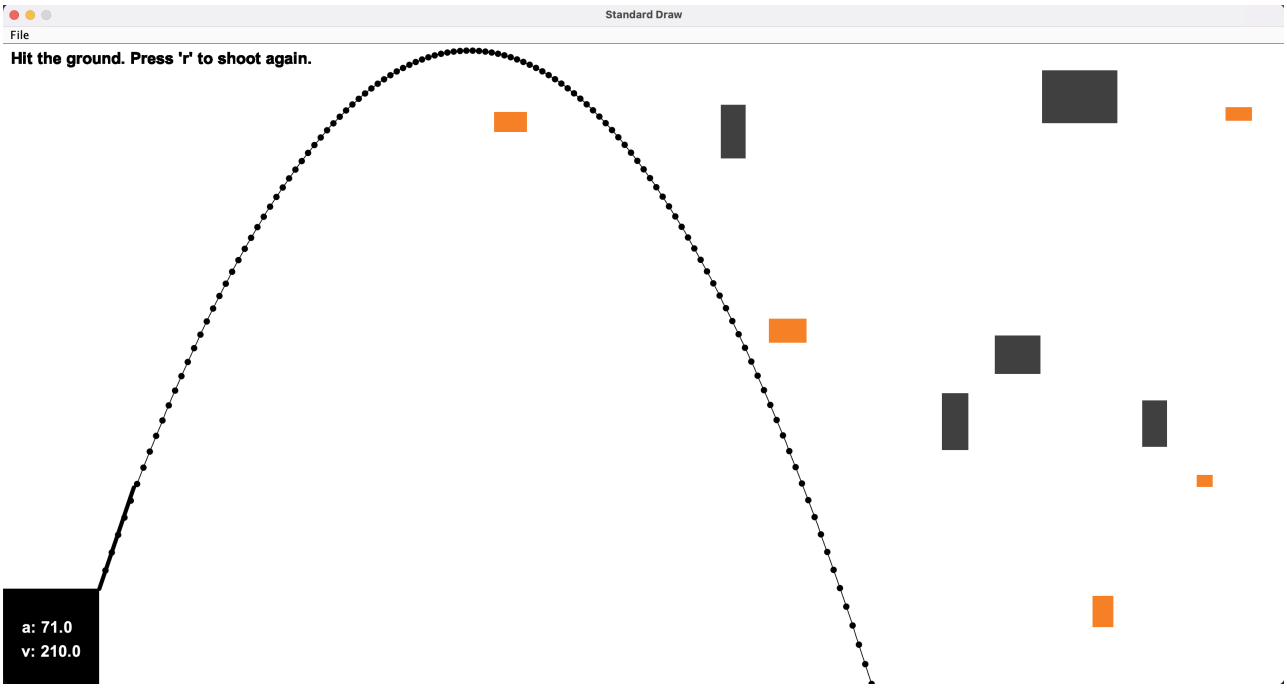


(a) Obstacle Hit Example 1

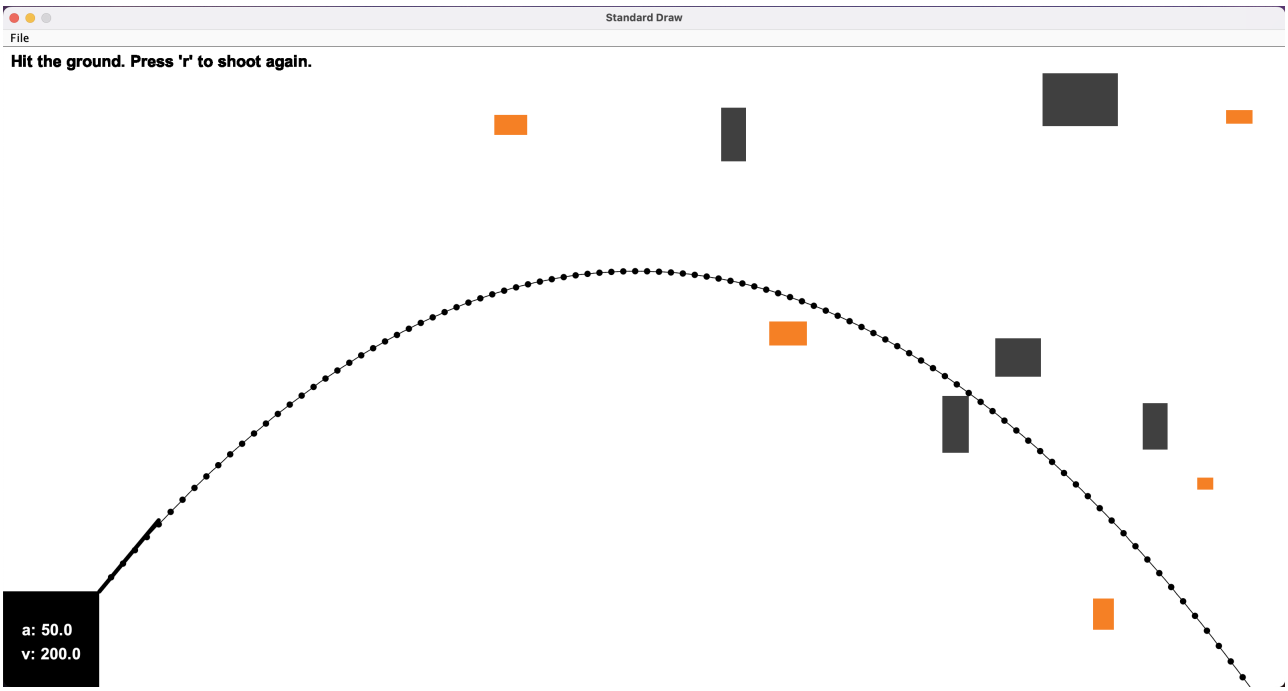


(b) Obstacle Hit Example 2

Fig. 2.8: Illustrations of hitting obstacles with custom parameters.

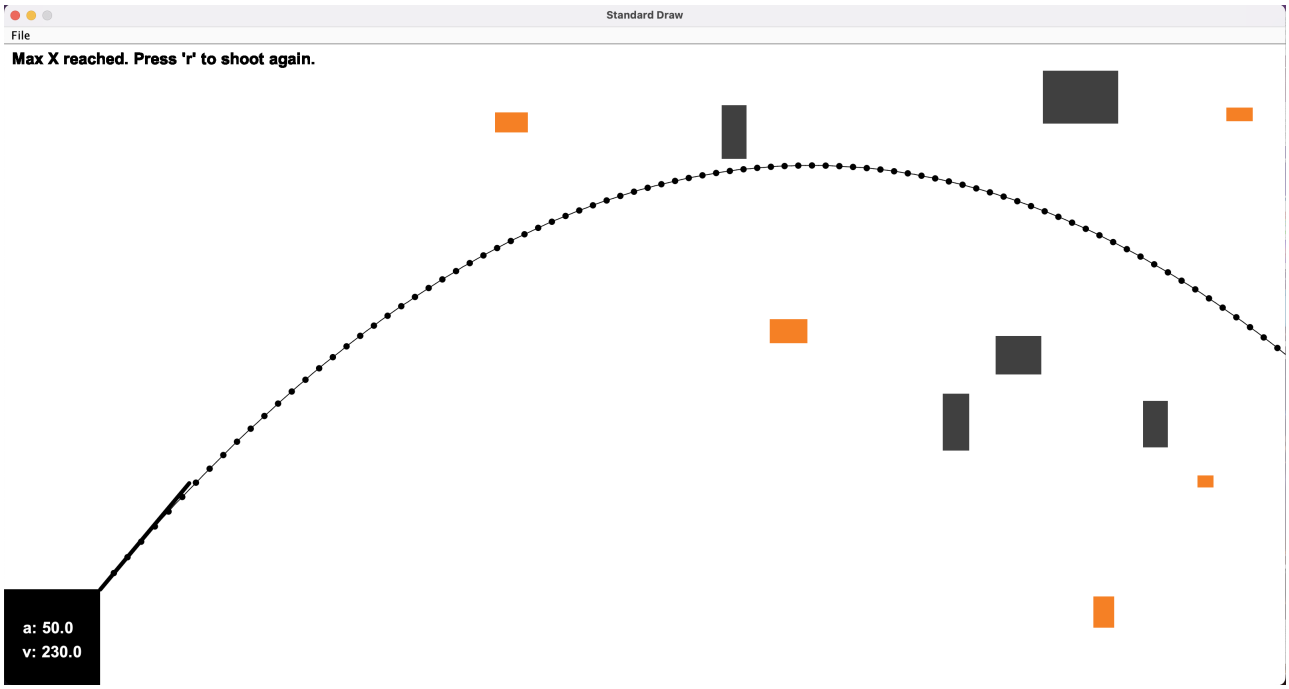


(a) Ground Contact Example 1

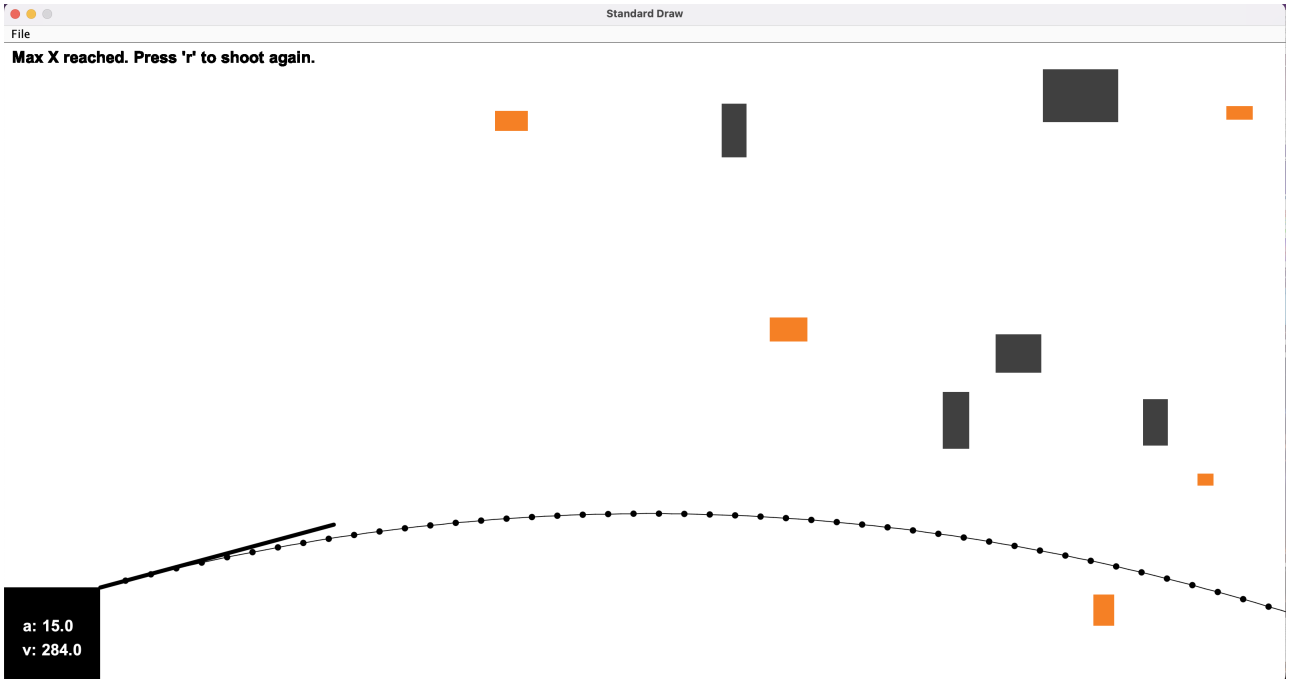


(b) Ground Contact Example 2

Fig. 2.9: Illustrations of ground contact with custom parameters.

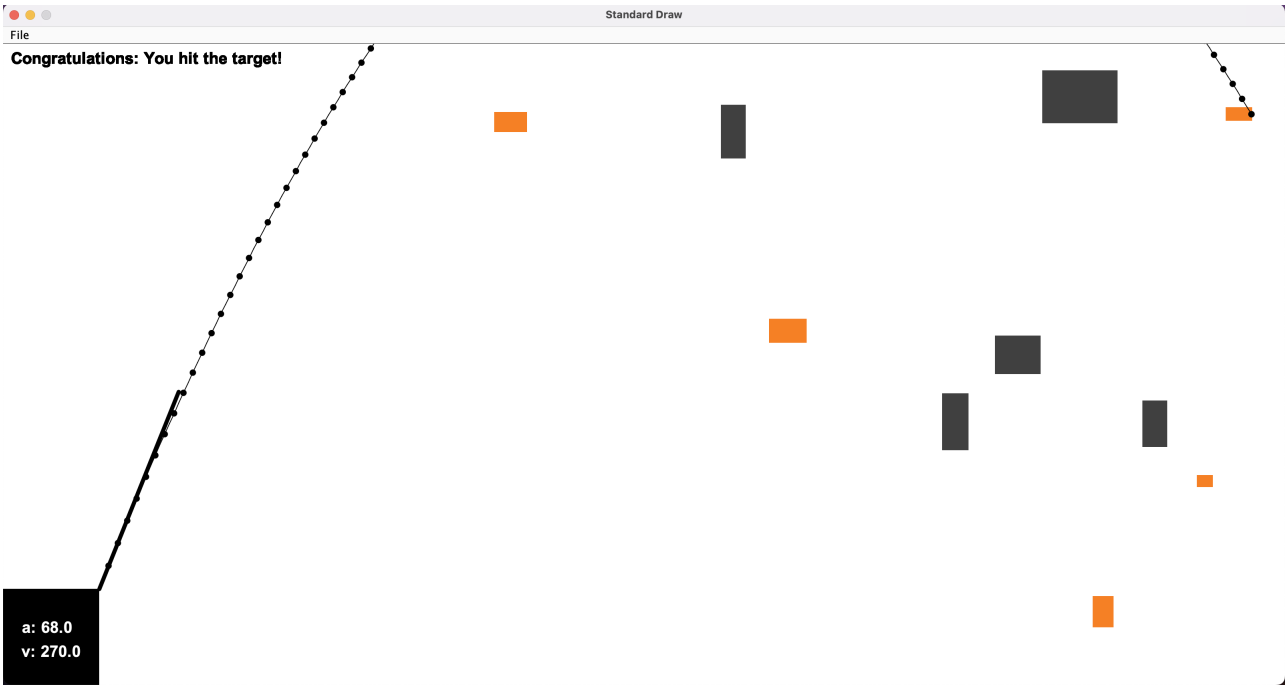


(a) Reaching Max X Coordinate Example 1

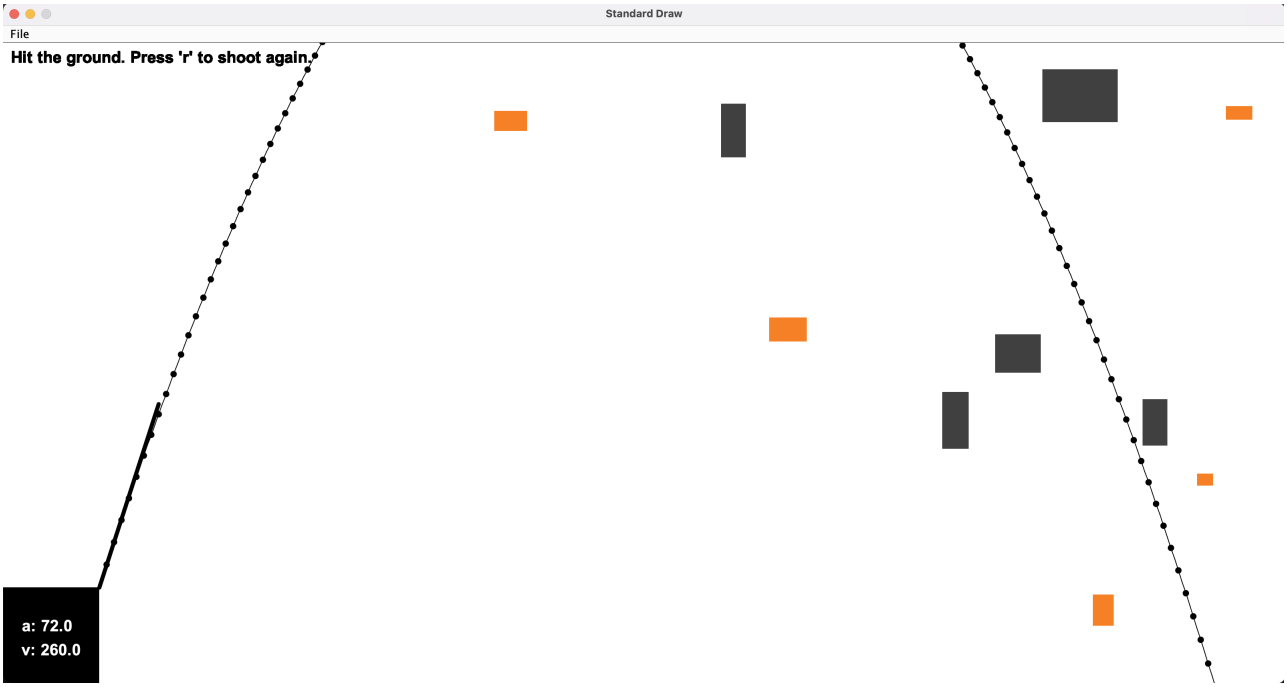


(b) Reaching Max X Coordinate Example 2

Fig. 2.10: Illustrations of reaching max X coordinate with custom parameters.



(a) Bullet Going To The Sky Example 1



(b) Bullet Going To The Sky Example 2

Fig. 2.11: Illustrations of bullet going to the sky with custom parameters.