ECE 150: Fundamentals of Programming

Sections 001 and 002

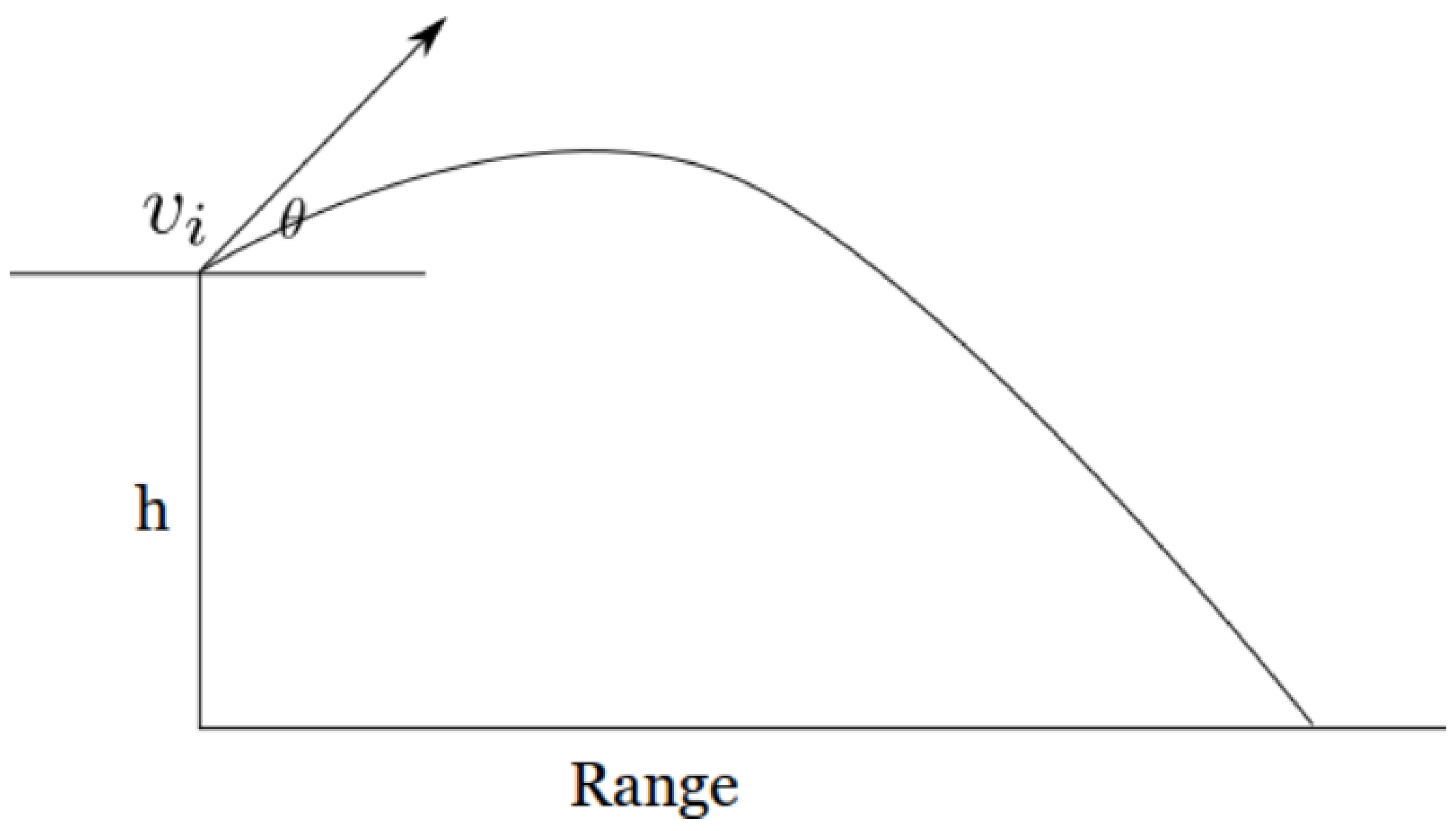
Project 0

Deadline: 11:59pm on Friday 21, 2018

Problem 1

You will implement a rudimentary computation engine for a classic QBasic game called Gorillas. This is a two-player game where each player controls a gorilla's ability to launch exploding bananas. The objective is to hit the other gorilla. If you wish to view a video of this classic game, please visit this youtube video.

The rudimentary computation engine implements projectile equations that we learn in kinematics. In order to find out if a banana following the projectile trajectory hits the target, we must compute the range of the projectile. Let v_i be the initial velocity, θ be the angle of launch, and h be the height from where the banana is launched.



Although you probably know the kinematics equations from physics, we provide you the relevant equations for this project. In particular, you would need the following equations to compute the range:

- Horizontal velocity: $v_x = v_i \cos(\theta)$
- Vertical velocity: $v_y = v_i \sin(\theta)$
- Distance: $d = \frac{v_x v_y}{g} + v_x \frac{\sqrt{v_y^2 + 2gh}}{g}$

Additionally, we need to know acceleration due to gravity, which is $g = 9.80665$ m/s^2. The computation takes four inputs: the initial velocity in m/s, the launch angle in degrees, acceleration due to gravity in m/s^2, and the height in meters.

Part A

Write a C++ program to print to the console the range of a banana when it is thrown. Your program must adhere to the following constraints:

- Use a separate function to compute each of the equations: horizontal, vertical, gravity, and range values. Note that the gravity function simply returns the value 9.80665.
- The function identifiers should be `horizontal_velocity`, `vertical_velocity`, `gravity`, and `computed_range`, respectively.
- The `horizontal_velocity` and `vertical_velocity` functions should take two input parameters (launch velocity and launch angle, in that order) and return one value.
- The range function should take three parameters (launch velocity, launch angle, and height in that order) as input and return one value.
- You can use the `cmath` standard library for sine and cosine functions. Their parameters assume the units to be in radians.
- Each function should have its function declaration, and associated function definition.
- All parameters and return values are of type `double`.

For an initial velocity of 600 m/s, a height of 10 m, compute the range for the following launch angles (in degrees): 10, 20, 25.5, 30, 40, 50, 55.5 60, 70, 75.5, 80, 90. Print to the console the following strings with their computed ranges with the characters *nnn* and *ddd* replaced with the appropriate values.

```
The range the banana reaches for the launch angle of nnn degrees is ddd
```

Repeat the printing of this string, one line per angle, to the console for each of the launch angles.

Part B

For the given initial velocity of 600 m/s, a height of 0 m, and the launch angles (in degrees) of 10, 20, 25.5, 30, 40, 50, 55.5 60, 70, 75.5, 80, and 90, state the maximum range you discovered.

Using the functions you defined in Part A, write a C++ program to compute the range if the height from where the banana was launched is at the same level as that of the target gorilla. Print to the console the range for an initial velocity of 600 m/s, and launch angles (in degrees) of 10, 20, 25.5, 30, 40, 50, 55.5 60, 70, 75.5, 80, and 90. Then, state the maximum range that your program discovered.

Marmoset Submission Requirement

In order for the Marmoset auto-grading system to work on your code, your code must meet the following requirements:

- Your submission file must be called “gorillas.cpp” (case-sensitive).
- You must wrap your main function declaration in the following pre-processor directive:

```
#ifndef MARMOSET_TESTING

int main();

#endif
```

- You must wrap your main function definition in the following pre-processor directive:

```
#ifndef MARMOSET_TESTING

int main() {
     // your code here

     return 0;
}

#endif
```