CS100B Fall 1999

Professor David I. Schwartz

Programming Assignment 5

Due: Thursday, November 18 1999

1. Goals

This assignment will help you develop skills in software development. You will:

- develop software that uses graphics and a GUI (graphical user interface)
- model physical behavior with mathematical equations
- save Ithaca from destruction by “evil” robots from the Planet CS100A.

2. Your Mission

2.1 Ithaca Defense League

Imagine that you work for the *Ithaca Defense League*, a tight-knit group of stunning programmers who protect and serve the general populace at large. One day, a local astronomer working in the Space Sciences Building discovered that a fleet of starships were zooming towards Earth.

2.2 Attack of the Evil Robots

Where did “they” come from? The “evil” robots inhabit Planet CS100A[*], a planet from far, far away, populated by mechanical *non-sentient* robots. Long ago, a race of good aliens[†] developed and programmed the brains of these robots. Unfortunately, the aliens never took CS100B on planet Earth. So, these aliens neglected to comment their code and use good, consistent style. Worse yet, the alien race used too many **`break`**s and **`goto`**s, which initiated the robots’ “evil” behaviors. The robots were programmed without structured code and, thus, became “evil.”

The “evil’ robots ultimately destroyed their creators in a rampage of compiler and runtime errors due to mismatched variable declarations, uninstantiated objects, and out-of-bound arrays. All attempts at diplomacy have failed because the robots cannot access any input – the alien designers also neglected to catch **`IOException`**s! Thankfully, since the robots lack sentience, you can destroy them, safely assured that you are really just eradicating bad, bloated code.

*. In the language of Planet CS100A, the name “CS100A” translates as “*planet whose name coincidentally is the same name as the real CS100A of Cornell University, but bears no actual resemblance.*”

†. From a secret mission to Planet CS100A, you obtained a DNA sample to study the planet’s former sentient inhabitants. Thankfully, you could examine and compare the genetic make-up by using the Smith-Waterman Algorithm, which you had learned many years ago. Even after adding gaps, you now know for certain that the human and alien DNA sequences differ vastly.

2.3 IDL to the Rescue!

The “evil” robots have inadvertently decided to attack Ithaca, New York because of the weird glitches in their programs. Fortunately, the IDL has developed a special tank that can destroy the spaceships that carry the robots. The IDL has not finished the development, however. Before building the prototype, the IDL must simulate the real device with a computer model. Fortunately, the other members of your team completed most of the work – you only need to finish a small part of the program. Once properly developed, the IDL can start constructing the real tank. May *The Schwartz* be with you!

2.4 And…

After successfully completing the program simulation, the IDL… <to be finished by you – see Section 6, last bullet)

3. Mathematics

Before modeling a device, you need the model! But, before you can even model the physics of the problem, first review some fundamental mathematical background.

3.1 Vectors

Recall that a quantity called a ***vector*** has direction and magnitude. Represent a vector with arrow notation as $\vec{V}$.

3.2 Components

Given a vector in 2 dimensional space,

$$\vec{V} = V_x\hat{i} + V_y\hat{j} \tag{1}$$

Vector $\vec{V}$ has the horizontal component V_x

$$V_x = |\vec{V}|\cos\theta \tag{2}$$

and vertical component V_y

$$V_y = |\vec{V}|\sin\theta \tag{3}$$

Vectors $\hat{i}$ and $\hat{j}$ are ***unit vectors***, vectors with magnitude of 1, along x and y axes, respectively.

3.3 Magnitude

You can find vector magnitude $|\vec{V}|$ from the Pythagorean Theorem, given values of V_x and V_y:

$$|\vec{V}| = \sqrt{V_x^2 + V_y^2} \tag{4}$$

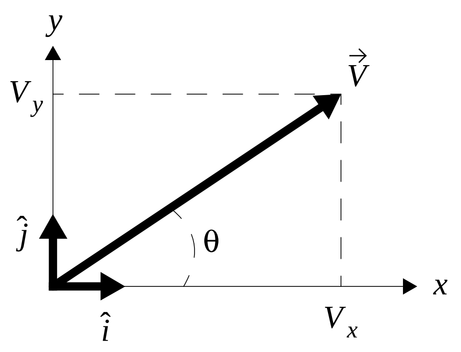

Figure 1: Vector

4. Physics

The tank you are modeling launches projectiles to blast the robot spaceships out of the sky. To simulate the projectiles, you need to model the physics of ***projectile motion*** to finish the coding of the simulation software.

4.1 Projectiles

A projected body, or ***projectile***, launches with an initial velocity and follows a path determined by gravity and air resistance. Example projectiles include balls thrown in the air and bullets fired by weapons.

4.2 Projectile Motion

The projectile follows a ***trajectory*** affected by some factors, like gravity, air resistance, and the curvature of the Earth. ***Projectile motion*** assumes only gravity with all other conditions as "ideal," like a flat Earth and no friction. Thus, a projectile is a ***freely falling body***. Since gravity acts downward, the projectile experiences only vertical acceleration due to the force of gravity. No force acts in the horizontal direction.

4.3 Coordinate System

Figure 2 illustrates an example of projectile motion. Since no other force other than gravity acts upon the projectile, the trajectory follows a planar path. Use an xy coordinate system to represent this path. For now, assume that the projectile launches where coordinate system begins. The axes x and y thus represent height and distance of the projectile, respectively. When you model the trajectory in Java, you will need to adjust the axes, as shown in Figure 3.

4.4 Vertical Acceleration

Denote the vertical acceleration as a_y. Because gravity acts downward,

$$a_y = -g, \tag{5}$$

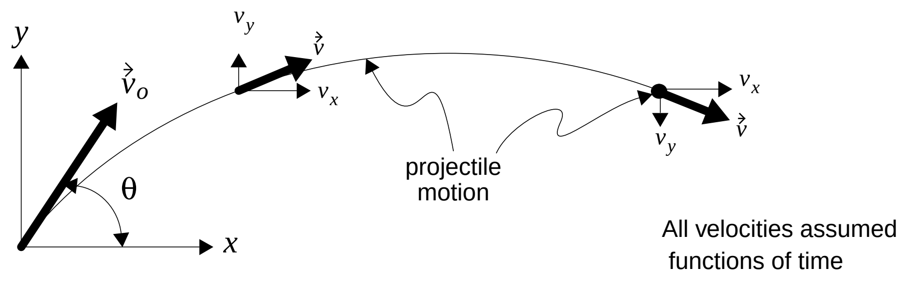

Figure 2: Projectile Motion

where g is the acceleration due to gravity. The negative sign results from gravity acting downwards, opposite to the assumed y axis orientation. You can assume that the projectile motion occurs close to the surface of the Earth where $g = 9.81 \text{ m/s}^2$ and remains constant.

4.5 Horizontal Acceleration

No force acts on the projectile in the x direction. Remember the law that a body in motion tends to stay in motion? Thus,

$$a_x = 0. \tag{6}$$

Without hitting an obstacle, such as the ground or a target, a projectile could feasibly keep moving horizontally, assuming ideal conditions.

4.6 Velocity

Given time t, note that acceleration $\vec{a}(t)$ and ***velocity*** $\vec{v}(t)$ have the following relationship:

$$\vec{a}(t) = \frac{d}{dt}\vec{v}(t). \tag{7}$$

Acceleration represents the rate of change of velocity with respect to time t. Because horizontal and vertical components of projectile motion act independently, you can consider each component separately. Using integrals to derive equations for velocity,

$$v_x(t) = \int a_x dt = \int 0 dt = C_{x1} \tag{8}$$

and

$$v_y(t) = \int a_y dt = \int (-g) dt = -gt + C_{y1}. \tag{9}$$

4.7 Displacement

Given time t, note that velocity $\vec{v}(t)$ and ***displacement*** $\vec{s}(t)$ have the following relationship:

$$\vec{v}(t) = \frac{d}{dt}\vec{s}(t). \tag{10}$$

Velocity represents the rate of change of displacement with respect to time. To find the horizontal and vertical displacement, integrate Equations 8 and 9:

$$x = s_x(t) = \int C_{x1}dt = C_{x1}t + C_{x2} \tag{11}$$

and

$$y = s_y(t) = \int(-gt + C_{y1})dt = -\frac{gt^2}{2} + C_{y1}t + C_{y2}. \tag{12}$$

The locations along the axes, x and y, represent the horizontal and vertical displacements of the trajectory, respectively.

4.8 Initial Conditions for Velocity

Denote the initial firing or launching velocity as $\vec{v}_0$. The word *initial* implies that $t = 0$ at the launch:

$$\vec{v}(t = 0) = \vec{v}(0) = \vec{v}_0. \tag{13}$$

The following information helps resolve the vector components to help plot the trajectory:

- $\vec{v}_0$ has the magnitude v_0, also called ***speed***.
- $\vec{v}_0$ has horizontal component v_{0x} and vertical component v_{0y}.
- $\vec{v}_0$ has an initial angle θ, the initial angle of the projectile.

From Figure 2, note that the initial velocity vector $\vec{v}_0$ has the components[‡]

$$v_{0x} = v_0\cos\theta \tag{14}$$

and

$$v_{0y} = v_0\sin\theta. \tag{15}$$

From Equations 8 and 9,

$$\begin{gathered} v_x(0) = v_{0x} = v_0\cos\theta = C_{x1} \\ \therefore C_{x1} = v_0\cos\theta \end{gathered} \tag{16}$$

and

‡. You can check these results with the definition of vector magnitude which uses the Pythagorean Theorem, from Equation 4: $|\vec{v}_0| = \sqrt{v_{0x}^2 + v_{0y}^2} = \sqrt{(v_0\cos\theta)^2 + (v_0\sin\theta)^2} = \sqrt{v_0^2(\cos^2\theta + \sin^2\theta)} = \sqrt{v_0^2} = v_0$.

$$\begin{gathered} v_y(0) = v_{0y} = v_0 \sin\theta = -(g)(0) + C_{y1} = C_{y1} \\ \therefore C_{y1} = v_0 \sin\theta \end{gathered} . \quad \textbf{(17)}$$

4.9 Initial Conditions for Displacement

When the projectile launches, assume that the origin of the xy axes coincides with the firing location. Usually the initial coordinates x_0 and y_0 are zero, but if the launcher changes location, then the initial locations might be nonzero, as shown in Figure 3:

$$s_x(0) = x_0 \quad \textbf{(18)}$$

and

$$s_y(0) = y_0 . \quad \textbf{(19)}$$

From Equation 11, derive the equation of horizontal displacement from

$$s_x(0) = x_0 = (v_0 \cos\theta)(0) + C_{x2} = C_{x2} , \quad \textbf{(20)}$$

where $C_{x2} = x_0$ must hold. Therefore,

$$x = x_0 + v_0 t \cos\theta . \quad \textbf{(21)}$$

From Equation 12, derive the equation of vertical displacement from

$$s_y(0) = y_0 = -\frac{g(0)^3}{2} + (v_0 \sin\theta)(0) + C_{y2} = C_{y2} , \quad \textbf{(22)}$$

where $C_{y2} = y_0$ must hold. Therefore,

$$y = y_0 - \frac{gt^2}{2} + v_0 t \sin\theta . \quad \textbf{(23)}$$

5. Problems

Simulating the trajectory of a projectile requires some knowledge of graphics. Since the IDL uses Java, you must adapt your equations of projectile motion for your program.

5.1 Projectile Motion

Each **`Graphics`** object uses the coordinate system that starts at the top, left corner, as shown in Figure 7.1 of Lewis&Loftus and Figure 3 of this assignment. You must modify Equations 21 and 23 for this altered coordinate system to model the trajectory of the projectile. Refer to class **`Projectile`** in the supplied code:

- The variable **`xInit`** corresponds to x_0.
- The variable **`yInit`** corresponds to y_0.

You must supply the appropriate code to model the correct equations of motion in methods **`calculateXPos`** and **`calculateYPos`**. Hints:

- One second of "simulated time" corresponds to one second of "real" time.
- One pixel represents one meter for the model.

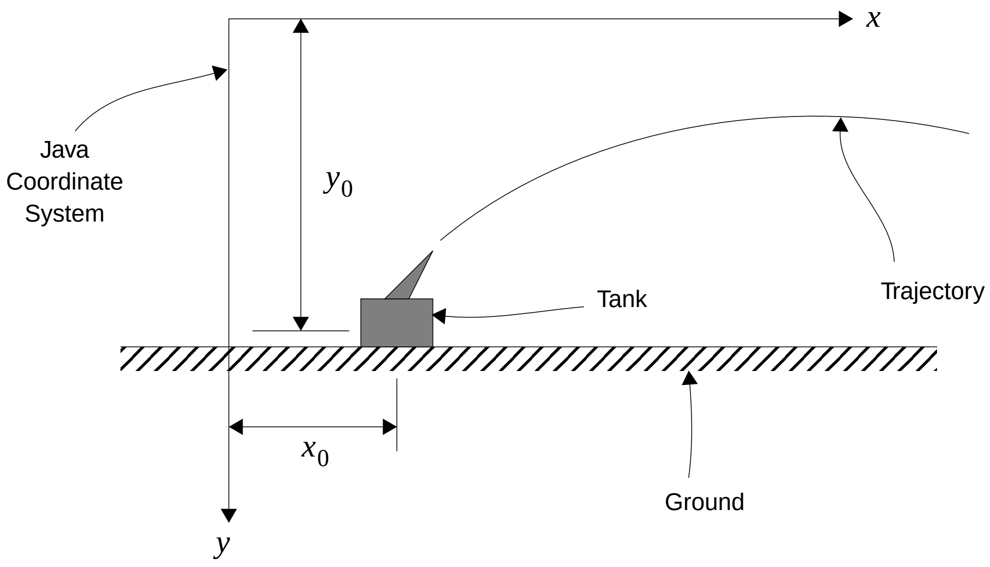

Figure 3: Java Coordinate System and Trajectory

5.2 Projectile Graphics

Your program needs to draw the trajectory of the projectile, given the equations in Section 5.1. You need to complete the **`DrawProjectile`** and **`EraseProjectile`** methods inside the **`Projectile`** class. Use a circle for the image of the projectile.

6. What to Hand In

Complete the code posted on the CS100B Website. Look for the portions that indicate where you should add code. You must use all given variable, method, and class names. What to hand in:

- full program listings – don't forget to label your work as stated in Section 7.2.
- screen shot that demonstrates the projectile being fired
- a conclusion to the story started in Sections 1 and 2. Use Section 2.4 for the first sentence. You should write about 1/4 page, typed, single-spaced, 12-point font. Your grammar and spelling must be correct – no vulgarity or lewdness, please.

7. Submitting Your Work

Failure to submit working to code might result with the "evil" robots winning. The fate of all of Ithaca rests upon your shoulders.

7.1 Due Date

This assignment is due in lecture on Thursday, November 18, 1999. You may submit your assignment to a consultant *before*, but not on, that date in the consulting room in Carpenter. Late programs will not be accepted.

7.2 Labeling Your Work

Always write your name, Cornell ID#, and the day/time/instructor for your section in the first comment of each program you hand in for credit. You must type all solutions and commentary! If you wrote the program with a partner, turn in only one printout with your partner's name and ID# in the comment, as well as your own. The comment must also include the section day/time/ instructor for the partner. The program will be returned to the first person listed. Sign your name(s) in the comment. Please staple the pages of your assignment together.

7.3 Grading

This assignment will be given two grades: the first based on correctness, the second on program organization and style. Each grade will be a 0, 1, or 2. Not only should your program work, but it should contain adequate comments to guide a reader interested in understanding it. The declaration of every significant variable should include a comment describing that variable. Include appropriate comments in the code so the reader can see the structure of the program, but not so many that the reader has trouble reading program text.

7.4 Academic Integrity

All work submitted should be your own or your partner's. The output submitted should exactly match the code submitted. Issues of academic integrity will be taken very seriously. See the Academic Integrity link from the CS100B web page for more information.

8. Acknowledgments

The CS100B staff would like to thank all those who contributed to and reviewed this assignment, especially Woong Yoon who developed most of the code provided to the students. We would also like to thank Planet CS100A.

9. References

9.1 Cornell

- Computer Graphics: *http://www.graphics.cornell.edu/*
- Physics: *http://www.physics.cornell.edu/physics/index.html*
- Mechanical and Aerospace Engineering: *http://www.mae.cornell.edu/*

9.2 Books

- Young, H. D. and R. A. Freedman (1996). *University Physics*. New York: Addison-Wesley.

9.3 On-line Java Manuals

- *http://java.sun.com/products/jdk/1.2/docs/api/index.html*