# CSC171 — Project 3

## Fireworks!

You have been hired by the Acme Fireworks Company to develop a fireworks simulator that their designers can use while developing new fireworks displays. The good news is that fireworks travel in parabolic trajectories, just like the projectiles in Project 1. The new challenge is that the company wants a graphical simulator that will show them what the fireworks will look like when they explode.

The company wants to see two things:

- The trajectory of the firework after launch; and
- A graphical rendering of the firework exploding.

Note that you do not have to *animate* these graphics (that is, draw them in real-time). You will need to render (draw) a picture showing the trajectory and the explosion, as detailed below. We will get to animation in the final project of the course.

## Project Requirements

The basics of fireworks for purposes of this project are as follows:

- Since we are launching projectiles, the user will need to specify the launch angle and speed.
- You may assume that the projectile is launched from a fixed position on the screen (e.g., bottom left), or this may be something you allow the user to specify.
- You may assume no wind resistance if you like. In that case, the coordinates at time $t$ of a projectile launched from height $0$ at speed $v_0$ and angle of elevation $\theta$ are:

$$\begin{aligned} x &= v_0 \cos(\theta) t \\ y &= v_0 \sin(\theta) t - \frac{1}{2} g t^2 \end{aligned}$$

- Most fireworks explode based on the time since they were launched (actually, an internal fuse burns and eventually causes the firework to explode). So this time delay is also a parameter of the launch.
- The user must specify the color of the firework. This is used to draw the trajectory, and perhaps also used to draw the explosion.

- The user must be able to choose among different options for type of explosion. However what the different types are is *up to you*. Each type of explosion must be drawn differently by your program. For a very simple example, you could just draw a circle of some fixed size at the right location along the trajectory using the given color. You must have at least five types of explosion and no, you won't get any points for doing the circle (but it might be a good way to start).

  You may need additional parameters for some kinds of explosions. If so, you will need to adjust your user interface accordingly. You may want to have random elements in your explosions. We will reward creativity and penalize lack of effort on this.

You will need to develop a graphical user interface (GUI) for your application. You should use the Swing toolkit for this. Your GUI will have three main items:

- An area where you can set the parameters of the fireworks launch. You should use graphical controls where possible, text fields if necessary. For some things, such as choosing colors or the type of explosion, you may need to do some work to make a good graphical control. Please do not use third-party widgets or libraries. Write your own code.

- A "canvas" in which you render (draw) the scene. Using the parameters of the launch, you must draw the trajectory and the explosion (at the correct position on the trajectory). Drawing on the screen is something you know how to do.

- A button that causes the scene to be rendered in the canvas using the current parameters. You may have other ways of triggering the rendering if that makes sense for your UI.

Your application must be coded to work properly when its window is resized. Of course, tiny and/or huge windows may look strange with controls in them. But the layout of your controls should do something appropriate in reasonable circumstances. Among other things, you may not have a fixed-size drawing canvas, but should scale your canvas to the size of the window.

For this project, you should create a graphical application by defining a class that extends `javax.swing.JFrame`. This class should setup the user interface components in its constructor. You can put the `main` method for your application in the class itself (create an instance and make it visible), or have a separate "App" ("Application") class that does that.

Your submission must include a README text file or PDF document describing your project. You should include a basic "User's Guide" for your application in this. It doesn't have to be complicated—the application isn't complicated. But your users shouldn't be confused about what to do. Also use this document to point out any special features, extra credit work, *etc.*

## Grading and Extra Credit

You can get full points for accomplishing the requirements as described above. The following are some ideas for extra credit, up to 20%. Be sure to document any claims for extra credit in your README or other documentation.

- Support firing a two projectiles towards each other. Your code would need to compute if they come close enough to each other to count as a “hit,” and then do something different with the rendering if so. [up to 10%]
- Support firing multiple projectiles at once by creating a *dynamic* UI to set the necessary properties. Typically this might have a “plus” button that adds a sub-panel for each additional projectile (and each sub-panel would have a way to delete it, typically an “X” or a “minus” button). And then, of course, you use the number of projectiles and their parameters in your rendering. This has to look nice and work well to get full points. [up to 10%]
- With some fireworks, the first explosion merely breaks the projectile into several smaller pieces, each of which shoot off in some direction and then explode separately. Add support for this type of fireworks. Your trajectory simulations should take account of the trajectory of the original projectile when determining the trajectories of the child projectiles. [up to 20%]
- Other ideas: document what you did *clearly and prominently* in your documentation. We’ll give you credit based on our assessment of how hard it was to do (but of course the total can never be more than 20%).

As noted above, since animation will be part of the next project, it is not eligible for extra credit on this one.

## Mathematical Appendix

The Project Requirements, above, give you the equations of motion for a projectile launched from height $0$ at speed $v_0$ and angle of elevation $\theta$:

$$\begin{aligned} x &= v_0 \cos(\theta) t \\ y &= v_0 \sin(\theta) t - \frac{1}{2} g t^2 \end{aligned}$$

The terms $v_0 \cos(\theta)$ and $v_0 \sin(\theta)$ are the horizontal ($x$) and vertical ($y$) components of the initial velocity $v_0$, respectively. These are sometimes referred to as $v_{0x}$ and $v_{0y}$.

In the horizontal ($x$) direction, there is no force and hence no acceleration. The horizontal speed is constant. Therefore the horizontal distance traveled in time $t$ is simply $distance = speed \times time$, or $v_0 \cos(\theta) t$.

In the vertical ($y$) direction, there is a downwards (negative $y$) acceleration due to gravity, denoted by $g \approx 9.8 \mathrm{m/s}^2$. The vertical position is therefore the distance that would be traveled in the absence of any acceleration, namely $v_0 \sin(\theta) t$ minus (because gravity is down while launch velocity is up) the distance traveled due to gravitational acceleration, $\frac{1}{2} g t^2$ (you have to integrate twice since $a = \frac{dv}{dt} = \frac{d^2x}{dt^2} = g$).

You might find it useful in your program to think of $y$ as a function of $x$. How would you do that? Think about it before reading on...

Answer: We know that $x = v_0 \cos(\theta) t$, therefore

$$t = \frac{x}{v_0 \cos(\theta)}$$

Substituting into the equation for $y$ we get

$$y = \frac{v_0 \sin(\theta) x}{v_0 \cos(\theta)} - \frac{1}{2} g \frac{x^2}{(v_0 \cos(\theta))^2}$$

Or using the shorthand for the $x$ and $y$ components of the velocity:

$$y = \frac{v_{0y} x}{v_{0x}} - \frac{1}{2} g \frac{x^2}{v_{0x}^2}$$

Remember that, as noted in class and unlike in standard mathematical notation, the Java graphics coordinate system has its origin in the top-left corner and $y$ coordinates increase *downwards*. You will need to take account of this when turning the equations of motion into code.