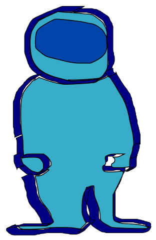
Celestial Mechanics with Scratch

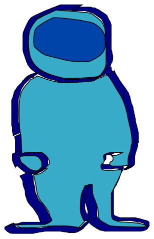
A Course for Incredibly Ambitious Kids

Sylvain Chassang[*]

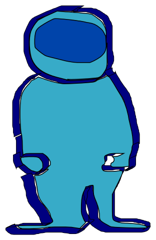
Princeton University



[*]Copyright Sylvain Chassang, 2020.



Contents



Why bother?

New York, April 1^{st}, 2020.

After three weeks confined in a New York apartment, I decided to use my new free time to formalize and expand some notes I've used to teach my niece (9) and nephew (12) simple physics *without calculus.* There is no wrong way to remain sane.

The visual programming language Scratch plays a key role in this endeavor. I think it's a great tool to run simple discrete-time simulations of dynamic systems. This avoids the abstraction of calculus without sacrificing rigor. The goal is to teach kids a bit of programming, a bit of physics, and some quantitative modeling skills using only the four basic operations and elementary logic.

In all likelihood, going through this class will require help from an adult, or an older sibling. One way to approach the course is to let kids work on each chapter on their own for a while before going through it with them again. It makes for a satisfying struggle, and hopefully there will be some meaningful rewards. Chapter 7, a revolting simulation of what goes on in the vomit comet, comes to mind.

The hope is that the course will have something to offer even to younger kids who might prefer to play around with the simulations. Scratch has remarkable graphical capabilities. Customizing the simulations to make them as pretty or weird as possible is fun.

Requirements. For this class you will need

- to add, substract, multiply, and divide like a rockstar;
- to understand mathematical symbols like $+, -, \times, /$ as well as $<$ (less than), or $>$ (greater than);
- some scrap paper and a pencil for practice exercises;
- a computer, an internet connection, and a scratch account (you can get one for free at scratch.mit.edu).

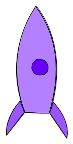
The Scratch projects corresponding to each chapter are available at

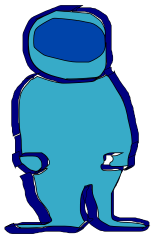
> https://scratch.mit.edu/users/prof_chassang/.

As you go through the chapters simply copy the relevant project to your own account by clicking the [`@Remix`] button in the upper-right corner of the screen. Get into the code by clicking the [`See Inside`] button and modify things as much as you want. Breaking things is a great way to learn, and there is no cleaning up here. If you mess things up, you can always go back the project's page, and create a brand new copy.

Finally, before you get started, it's probably a good idea to get acquainted with Scratch by going through a few tutorials here. Please do so now, the rest of this class will be a lot easier.

Chapter 1





Position and Speed

Position is pretty easy. The position of an object at a moment in time describes where the object is at that particular moment. For instance, right now I'm sitting at my desk. Of course, there are other ways to describe where I am. If you haven't been to my apartment, knowing that I'm at my desk will not help you find me. Instead, if you know where Grand Central Station in New York is, I could tell you

"From Grand Central, walk 3.5 blocks West and 28.5 blocks South. I'll be there."

These two numbers $(3.5, 28.5)$ describe my position on the map centered at Grand Central, with one block being the unit of distance.[1] With that information, you could go on Google Maps and find where I live.

[1]This is called a Cartesian coordinate system, named after this very smart Frenchman. There are many other coordinate systems, i.e. ways to describe position. An important one is polar coordinates.

This is the way an object is located in Scratch: the coordinate X describes the object's position on the horizontal axis (left to right), the coordinate Y describes the object's position on the vertical axis (bottom to top). The center of the screen is at the position $X = 0$, $Y = 0$ (sometimes I will write this $(0, 0)$). The top-right corner has coordinates $X = 240$, $Y = 180$, the bottom-left corner has coordinates $(X, Y) = (-240, -180)$. The unit of length is "the width of one Scratch-pixel"

Exercise 1.A Position in the Cartesian Plane

Get a piece of paper, and orient it sideways. Imagine the center has coordinates $(0, 0)$ – this is Grand Central in my example. Coordinate X corresponds to the horizontal axis, coordinate Y corresponds to the vertical axis. The unit of distance is the width of your left-thumb (right-thumb if you are a lefty).

1 Draw Grand Central, a horizontal line through the center for the X-axis, and a vertical line through the center for the Y-axis.

2 Draw the position of the United Nations: $(X, Y) = (3, 4)$.

3 Draw the position of the Empire State Building: $(-2, -8)$.

Speed (also called velocity – as in velociraptor) describes the change in position per unit of time. We get to pick our favorite unit of time. Since I don't have a watch, I propose we use the time it takes to say 'one thousand' as our unit of time. This way, we can count time: 1 'one thousand', 2 'one thousand', 3 'one thousand', and so on. Time T takes values 1, 2, 3, 4, 5 ... units of time. The X-axis position of an object at time T will be denoted by X_T. The Y-axis position of an object at time T will be denoted by Y_T.

Horizontal speed at time T, denoted by VX_T corresponds to the change of position between T and $T+1$:[2]

$$\begin{aligned} VX_T &= X_{T+1} - X_T \quad \text{or equivalently,} \\ X_{T+1} &= X_T + VX_T. \end{aligned}$$

For instance, if $X_3 = -1$ and $VX_3 = 1$, then $X_4 = 0$. If $X_3 = 2$ and $X_4 = 4$, then $V_3 = 2$.

Similarly, vertical speed at time T, denoted by VY_T, corresponds to the change of position between T and $T+1$:

$$\begin{aligned} VY_T &= Y_{T+1} - Y_T \quad \text{or equivalently,} \\ Y_{T+1} &= Y_T + VY_T. \end{aligned}$$

Exercise 1.B — Position and Speed, Speed and Position

1 The table below records the X-axis position of a tennis ball over time. Compute the horizontal speed VX of the ball over time.

T	1	2	3	4	5
X	0	10	20	30	15
VX					×

What do you think happened to the tennis ball at time $T = 4$?

2 The table below records the Y-axis speed of an elevator over time. Compute the vertical position Y of the elevator over time.

[2] Units of speed are expressed in units of distance per unit of time.

T	1	2	3	4	5	6	7
Y	0						
VY	0	1	2	2	1	0	×

What story do the numbers in this table tell?

Make sure you can solve these practice problems confidently before turning to the next part. In fact, you should imagine your own problems, and solve them.

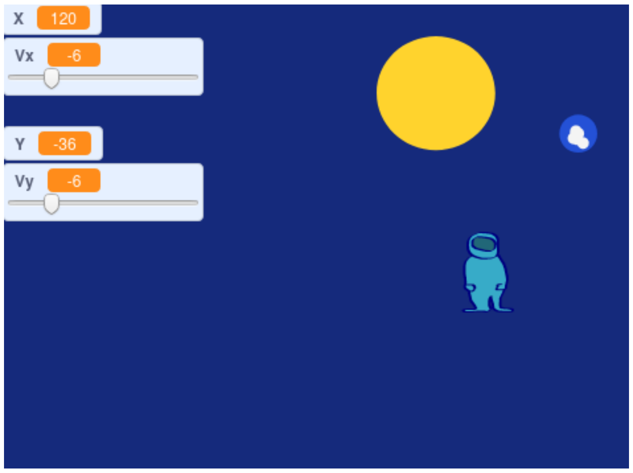


Figure 1.1: Professor Spaceman is lost in space

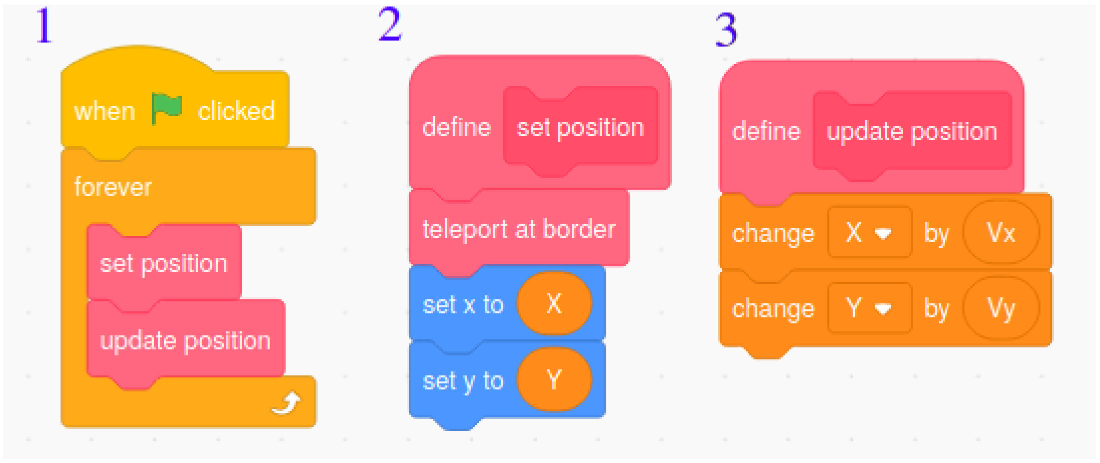
Let's Code!

We will start with something simple: Professor Spaceman (Dr Spaceman's less famous cousin) took a wrong turn near Jupiter and is lost in space. Our first project will simulate Professor Spaceman's journey as he drifts around the Solar System. We have the following use cases:

1. the user can plot the position (X, Y) of Professor Spaceman moving at a constant horizontal and vertical speed (VX, VY);
2. the user can change the speed of Professor Spaceman using sliders.

In addition we will try to keep our code neat and modular. This clarifies the logical structure of the code, and will help us recycle bits of code later on.

The key parts of the code are illustrated Figure 1.2. X and Y are variables created to represent the horizontal and vertical position of Professor Spaceman.



```
when 🏳 clicked
forever
    set position
    update position
```

```
define set position
teleport at border
set x to (X)
set y to (Y)
```

```
define update position
change [X ▾] by (Vx)
change [Y ▾] by (Vy)
```

Figure 1.2: What's going on here?

Exercise 1.C — Coding Mystery

Read the code step by step from left to right. Can you guess what it will do? What do you think the `teleport at border` module does?

Here is what the code does:

- The first block describes the logic of the program at a high level. When the green flag is clicked, a "forever" loop is started. Within each iteration of the loop, the `set position` module moves Professor Spaceman to his X, Y location, and the `update position` module updates Professor Spaceman's position using information about his speed.

- The second block describes the `set position` module. It first calls a mysterious module `teleport at border`. I'll explain this module below. Then it sets the X-axis position of Professor Spaceman equal to the variable X and the Y-axis position of Professor Spaceman equal to the variable Y.
- The third block describes the law of motion: after each unit of time variable X increases by amount Vx; variable Y increases by amount Vy.

The `teleport at border` module is used to keep Professor Spaceman inside the Scratch screen (otherwise he might drift away). Whenever Professor Spaceman hits one border of the screen, he is teleported to the opposite border.

You can check out the project code here. You can copy it to your own account using the [`@Remix`] button, and read the code using the [`See Inside`] button.

Exercise 1.D — Coding Challenge

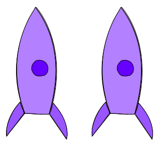
1 Launch the program by clicking the Green Flag. Did you correctly guess how the program would behave?

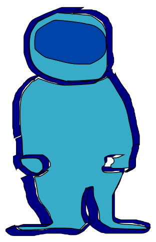
Can you bring Professor Spaceman safely back to Planet Earth by manipulating the sliders controlling his speed?

2 Can you modify the code so that Professor Spaceman spins clockwise as he travels through space?

3 Can you add the Planet Mars in the background? Can you add ambiance music to keep Professor Spaceman entertained?

Chapter 2





Position, Speed, and Acceleration

Position describes where an object is. Speed describes changes in an object's position. Acceleration describes changes in an object's speed.

Acceleration is key because in real life when we interact with an object, we do so by affecting its acceleration. Imagine that my car's battery is dead, and that I push it to get it started. At the beginning the car is not moving. Its speed is zero. When I start pushing the car, speed doesn't change instantaneously: it will still be very close to zero. However its speed will slowly increase: that's acceleration.

We will denote by AX acceleration on the horizontal X-axis, and by AY acceleration on the vertical Y-axis. At time T acceleration on the X axis is equal to the change in speed between time T and $T+1$:

$$\begin{aligned} AX_T &= VX_{T+1} - VX_T \quad \text{or equivalently,} \\ VX_{T+1} &= VX_T + AX_T. \end{aligned}$$

The same is true for the vertical acceleration of an object.

$$\begin{aligned} AY_T &= VY_{T+1} - VY_T \quad \text{or equivalently,} \\ VY_{T+1} &= VY_T + AY_T. \end{aligned}$$

Exercise 2.A — Substract and Add

1 The table below records the X-axis position of a tennis ball over time. Compute the horizontal speed VX, and the horizontal acceleration AX of the ball over time. It's easiest if you first compute all speeds, and then all accelerations.

T	1	2	3	4	5	6
X	0	10	20	30	15	0
VX						✗
AX					✗	✗

What do you think happened to the tennis ball at time $T = 3$?

2 A baseball flies in the air at a constant horizontal speed $VX = 5$. Compute its horizontal position and acceleration using the table below.

T	1	2	3	4	5	6	7
X	0						
VX	2	2	2	2	2	2	✗
AX						✗	✗

3 The baseball has a vertical speed $VY = 2$ at time $T = 1$, and a constant vertical acceleration $AY = -1$. Compute the baseball's vertical position using the table below. It's easiest if you start by computing all the speeds, and then all the positions.

T	1	2	3	4	5	6	7
Y	1						
VY	2						✕
AY	-1	-1	-1	-1	-1	✕	✕

4 On a piece of paper, plot the position (X, Y) of the baseball studied in problems 2 and 3 at times $T = 1, \cdots, 7$. You can pick your favorite unit of distance.

What is the highest point reached by the ball? What is the ball's vertical speed at its highest point? Why is that?

When does the ball hit the ground ($Y = 0$)?

The constant negative acceleration that drives the trajectory of the baseball in the previous exercise corresponds to what happens in reality when you throw an object. Its initial speed is the speed it has when it leaves your hand. After that, the object has approximately zero horizontal acceleration, and an approximately constant negative vertical acceleration corresponding to the gravitational pull of the earth.

Feeling confident? If yes, go to the next section. Otherwise, practice one more time. Give yourself a table with an object's acceleration, initial speed, and initial position, and compute its speed and position for the next 5 or 6 periods.

Let's Code!

Professor Spaceman is now back on Planet Earth. Today he is training with the US Airforce at Area 51 in the Nevada desert. We are going to simulate what happens to Professor Spaceman when he is launched into the air using a highly experimental rocket catapult. The project's specs are the following:

1. Professor Spaceman's trajectory has a horizontal acceleration AX equal to 0, and a constant negative vertical acceleration equal to AY.
2. A user can select Professor Spaceman's initial horizontal speed VX_1, initial vertical speed VY_1, and the fixed vertical acceleration AY.

As always we will try to keep the code neat and modular. Repeat after me: Neat and modular! Neat and modular!

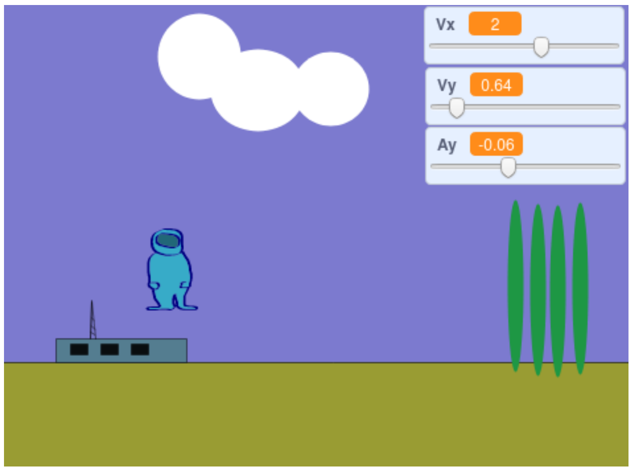
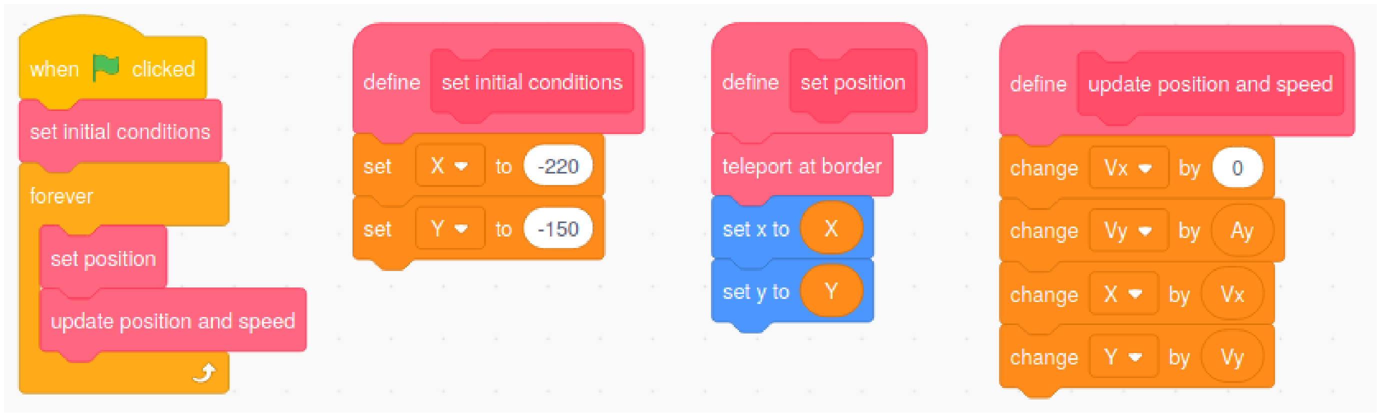


Figure 2.1: Professor Spaceman enjoying a view of Area 51 from the air

The code is described in Figure 2.2. It is similar to that of Figure 1.2.



```
when 🏴 clicked
set initial conditions
forever
    set position
    update position and speed

define set initial conditions
set X to -220
set Y to -150

define set position
teleport at border
set x to X
set y to Y

define update position and speed
change Vx by 0
change Vy by Ay
change X by Vx
change Y by Vy
```

Figure 2.2: What's going on here?

Exercise 2.B Coding Mystery

1 Read the code step by step from left to right. Can you guess what it will do? What is the horizontal acceleration of Professor Spaceman?

2 There are two main differences between the code of Figure 1.2 and the code of Figure 2.2. Can you find what they are?

In Figure 1.2, we only had an `udate position` module: the speed was given, and only the position was updated. In Figure 2.2, we have an `update position and speed` module: both the speed and the position are updated in each period. An other difference is the `set initial conditions` module: it specifies initial conditions for Professor Spaceman's trip into the sky. Here, his first position is in the lower left corner.

You can check out the project here. Copy it to your own account using the [`@Remix`] button.

Exercise 2.C — Coding Challenge

1 Play with the project (at first use initial values $VX = VY = 3$). Does it behave the way you were expecting?

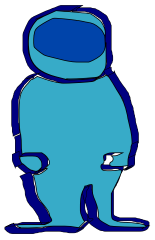
2 As time goes by, what happens to the horizontal speed VX of Professor Spaceman? What happens to the vertical speed VY of Professor Spaceman?

3 Can you modify the code so that the horizontal acceleration is equal to $-.05$? How does Professor Spaceman's trajectory change?

4 Can you modify the code so that Professor Spaceman stops moving after he hits the ground?

5 Can you add aliens having a picnic in the background?

Chapter 3



The Vomit Comet

In this chapter, we introduce force fields. I'm talking about the actual force fields studied by physicists, not the silly force fields from science fiction movies.

So far we have seen how to go from acceleration to speed, and from speed to position.[1] A force field is simply a description of the acceleration of an object depending on the position of the object as well as other characteristics of the object. The force fields we will study in the remainder of this course will depend only on the object's position, but in general things like an object's mass, electric charge and speed may affect its acceleration.

Here is a description of a force field: for every horizontal and vertical position X and Y, $AX = 0$ and $AY = -1$. This is a pretty good description of the force field generated by gravity around you.

[1] If you were a contrarian type of person, you might argue that we learned how to go from position to speed, and from speed to acceleration.

Here is a different force field that depends on the horizontal position of the object:

- if $X < 0$ (less than zero), then $AX = 0$ and $AY = 1$;
- if $X \geq 0$ (greater than or equal to zero), then $AX = 0$ and $AY = -1$.

When the object has a negative coordinate on the horizontal X-axis, the object is accelerated upwards (say by a vertical wind tunnel). When the object has a positive coordinate on the horizontal X-axis, the object is accelerated downwards (say by gravity).

Exercise 3.A — Trajectory in a Force Field

Consider the force field defined just above. We are interested in the trajectory of a ping pong ball evolving in this force field, with initial position $X = -2$, $Y = 0$, and initial speed $VX = 1$, $VY = 0$.

1 Using the table below, compute the X-axis position of the ping pong ball over time. It is easiest to compute all speeds first, and positions second.

T	1	2	3	4	5	6
X	-2					
VX	1					×
AX	0	0	0	0	×	×

2 Using the results from problem 1 and the description of the force field above, compute the vertical acceleration AY of the ping pong ball over time:

T	1	2	3	4	5	6
AY					✕	✕

3 Using the values of vertical acceleration AY obtained in problem 2, compute the vertical speed VY and the vertical position Y of the ping pong ball over time:

T	1	2	3	4	5	6
Y	0					
VY	0					✕
AY					✕	✕

In the next few chapters we are going to simulate several different force fields: the real gravitational field described by Isaac Newton, a fantasy gravitational force field that was not described by Newton, and a repulsive force field similar to the one between two magnets with the same polarity.

We end this chapter by simulating another real force field: the one felt by astronauts and scientists traveling aboard the vomit comet [1, 2].

Simulating a reduced gravity aircraft

The vomit comet – also called a reduced-gravity aircraft is a special plane that astronauts and scientists use to train or test equipment before going to space. I like to think of it as tennis ball that has learned to throw itself into the air, or as the best amusement park ride ever. The plane has a target height (say 3000 meters) and behaves like this:

1. whenever the plane is below its target height (3000 meters), the pilot runs the engines at their maximum power, and points the plane

upwards;

2. whenever the plane is above its target height (3000 meters), the pilot shuts off the engines and glides through the air under the influence of gravity alone.

Imaging how being on the vomit comet must feel! The reason the vomit comet is useful to astronauts and scientists is that it allows them to safely experience free fall. Whenever the engine is cut-off, the rocket is in free fall – like a tennis ball flying in the air. And for scientists free fall is not just fun: it allows them to simulate a zero-gravity environment. That's because when you fall, all parts of you are falling at the same time and you don't feel heavy anymore. Astronauts use the vomit rocket to learn how to move around in zero-gravity. Scientists and engineers use the vomit rocket to make sure that the machines and experiments they build for space actually work the way they expect in zero gravity.

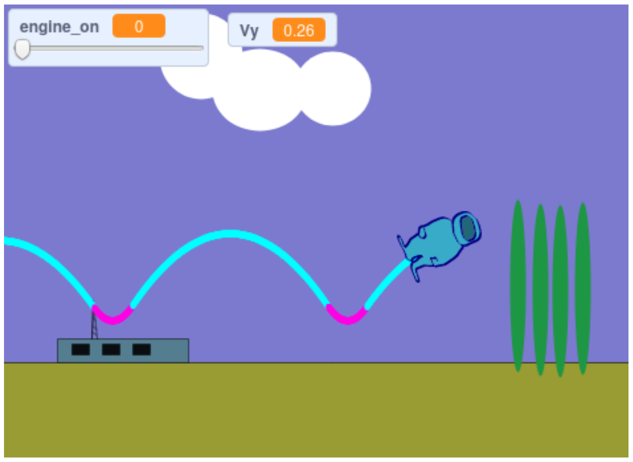


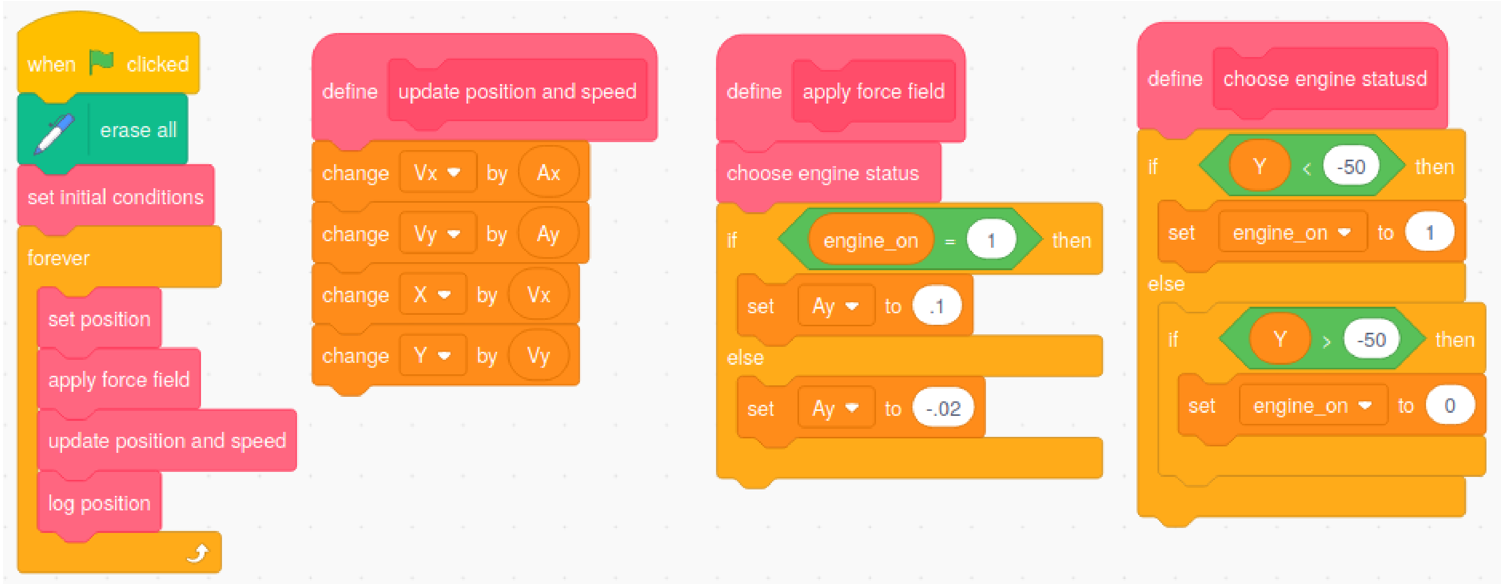
Figure 3.1: Professor Spaceman experiencing the joy of free fall

Let's Code!

Simulating the vomit rocket is pretty easy, and our specs are straightforward:

- the user watches Professor Spaceman experience the joy of free fall, with Area 51 in the background;
- a slider displays when the engine is on or off;
- the trajectory of Professor Spaceman in the air is drawn, and changes color when the engine is on or off.

As usual, we will keep our code neat and modular. The key parts of the code are illustrated by Figure 3.2.



```
when [flag] clicked
erase all
set initial conditions
forever
    set position
    apply force field
    update position and speed
    log position
```

```
define update position and speed
change Vx by Ax
change Vy by Ay
change X by Vx
change Y by Vy
```

```
define apply force field
choose engine status
if engine_on = 1 then
    set Ay to .1
else
    set Ay to -.02
```

```
define choose engine statusd
if Y < -50 then
    set engine_on to 1
else
    if Y > -50 then
        set engine_on to 0
```

Figure 3.2: What's going on here?

This looks complicated, but it reuses many parts that you have seen already.

Exercise 3.B Coding Mystery

1 Read the code step by step from left to right. Can you guess what it will do? What is the vertical acceleration of Professor Spaceman for $Y = -60$? What is the vertical acceleration of Professor Spaceman when $Y = 10$?

2 What are the differences between the code of Figure 2.2 and the code of Figure 3.2?

I want to highlight three new things about the code:

- the `apply force field` module sets vertical acceleration as a function of whether the engine is on our not;
- the `choose engine status` module determines whether the engine is on or off as a function of Professor Spaceman's vertical Y-axis position.
- the `log position` module (whose definition is not illustrated) plots the position of Professor Spaceman.

The code for the simulation is here. Copy it to your account and play around with the code for a while (say 10' or so).

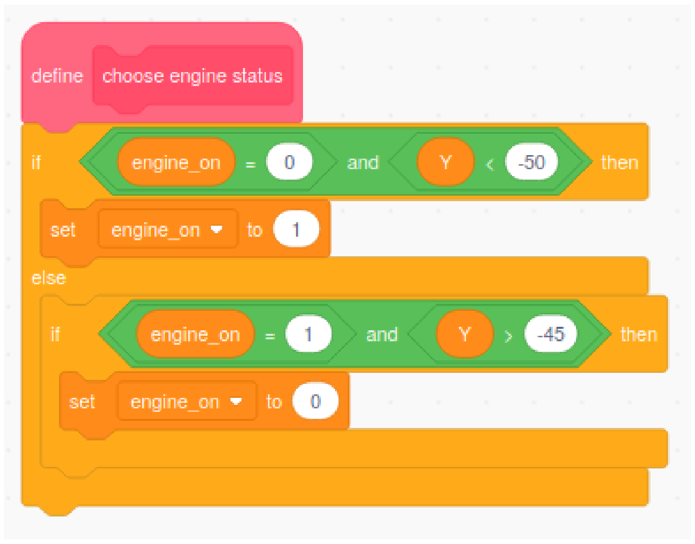
Exercise 3.C Coding Challenge

1 Does the code behave the way you expected?

2 Can you modify the code of the project so that the color is red when the engine is on, and green when the engine is off?

3 What is the altitude of Professor Spaceman at the beginning of his trajectory? What is the highest point on Professor Spaceman's path? Is the trajectory cyclical, i.e. does it repeat itself?

4 Professor Spaceman wants to reprogram the `choose engine status` module as follows:



```
define choose engine status
if engine_on = 0 and Y < -50 then
    set engine_on to 1
else
    if engine_on = 1 and Y > -45 then
        set engine_on to 0
```

Read the code through. How does this change the way the engine is started? How do you think this will change the trajectory of Professor Spaceman? Will he go higher or lower over time?

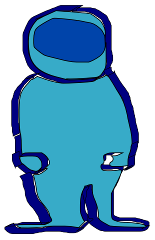
5 Modify the project's code to implement Professor Spaceman's proposal. Simulate the vomit comet. Does Professor Spaceman's trajectory behave the way you expected? Is the trajectory stable? What happens when you change the value -45 to -55? to -50?

Exercise 3.D Enigmas

1 Why do you think the reduced gravity aircraft is called the vomit comet?

2 Is it possible for Professor Spaceman to be in free fall while also having a positive vertical speed $VY > 0$?

Chapter 4



Know Your Roots

This chapter is a mathematical detour on our way to gravitation. Quite honestly, it's a difficult chapter. Do not get discouraged if you don't understand it very well at first. Try your best. Take some time off, skip to the next section if you get stuck, and try it again later. In any case, prepare to be entertained!

The reason for this detour is that to describe the gravitational force field, we need to calculate the distance between objects in a Cartesian coordinate system. To do this, we need to: (i) understand Pythagoras' Theorem; (ii) learn how to compute square roots. We will actually prove Pythagora's Theorem I'll be introducing a very powerful method to solve equations: the bisection method.

Pythagoras' Theorem

Consider the rectangle illustrated by Figure 4.1. The horizontal side has length X, while the vertical side has length Y. You can think as X and Y describing the the position of point A in relation to point O. The distance between point O and point A – called the diagonal – is denoted by D.

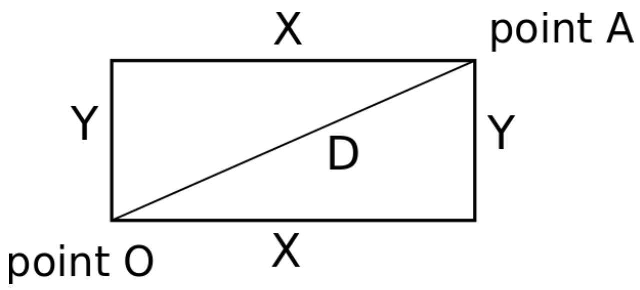


Figure 4.1: rectangles are fun

Pythagoras' Theorem states that

$$X^2 + Y^2 = D^2. \tag{4.1}$$

The notation X^2 (read it "X square") is just a shortcut that means $X \times X$. For instance, if $X = 4$, then $X^2 = 4 \times 4 = 16$.

Exercise 4.A — An Empirical Check

Draw 2 rectangles that look different. Measure their sides. Measure the diagonal. Check whether Pythagoras' Theorem holds, at least up to small errors (accurate measurement is difficult).

Now, let us prove Pythagoras' Theorem:

- First, consider the right triangle corresponding to the lower-right half of the rectangle in Figure 4.2

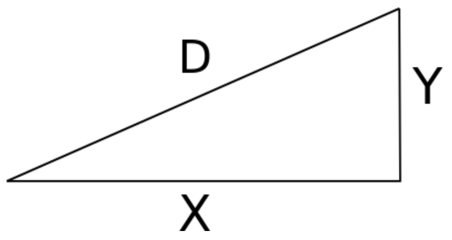


Figure 4.2: triangles are fun too

- Second, make four copies of this triangle and re-arrange them as in Figure 4.3

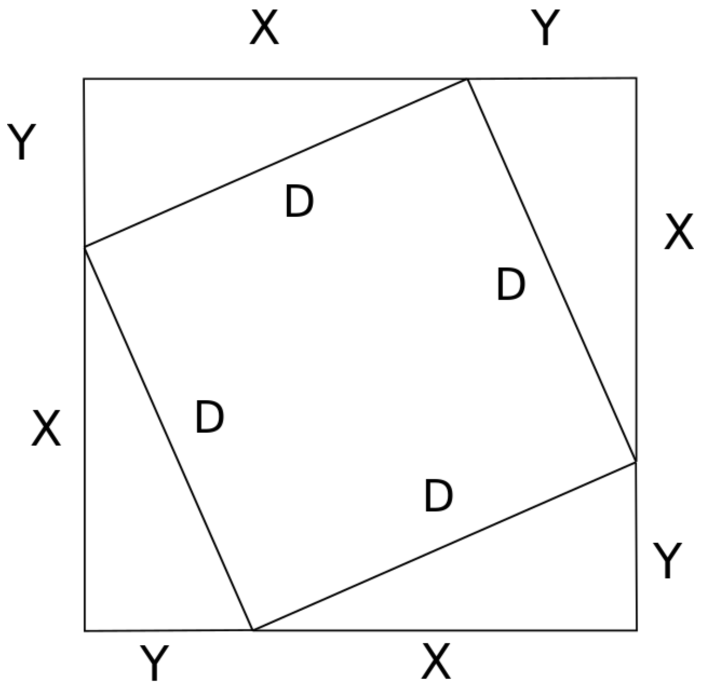


Figure 4.3: triangles turned into squares

The outside part of Figure 4.3 is 1 big square with a side of length $X + Y$. Within this big square, we can fit 1 smaller square with a side of length D, and 4 right triangles with sides of length X and Y.

- We finish with a bit of area accounting. The area of the big square can be computed in two ways.

Way #1: it's the product of the length of the two sides

$$\begin{aligned}\text{area big square} &= (X+Y)\times(X+Y)\\ &= X\times(X+Y)+Y\times(X+Y)\\ &= X^2+X\times Y+Y\times X+Y^2\\ &= X^2+Y^2+2\times X\times Y\end{aligned}$$

Way #2: it's the sum of the area of the smaller inside square with side of length D, plus the area of the 4 right triangles

$$\begin{aligned}\text{area big square} &= D^2+4\times\frac{1}{2}X\times Y\\ &= D^2+2\times X\times Y\end{aligned}$$

Since those are two equivalent ways to describe the area of the big square, it must be that $X^2+Y^2=D^2$.

That's it! We have proven Pythagoras' Theorem. Go through the proof a couple of times. Maybe draw the rectangles yourself one more time. Does every step make sense?

The reason Pythagoras' Theorem is useful is because it's the first step to compute length D.

Say that $X=5$ and $Y=7$, Pythagoras' Theorem implies that

$$D^2=X^2+Y^2=25+49=74.$$

Now we know the value of $D^2=D\times D$. We would like to know the value D such that $D^2=74$. This value D it is called the square root of 74, and is typically denoted by $\sqrt{74}$.

The bisection method will allow us to compute $D = \sqrt{74}$, as well as the solution to many other equations.

The Bisection Method

Let's first compute by hand an approximate solution to $D^2 = 74$. The reasoning is illustrated by Figure 4.4.

Step 1. We know that 8 must be less than $\sqrt{74}$ since $8^2 = 64 < 74$. 9 is greater than $\sqrt{74}$ since $9^2 = 81$. This means that $D = \sqrt{74}$ is between 8 and 9. We have just obtained a lower bound $D_{\text{min}} = 8$ and an upper bound $D_{\text{max}} = 9$ to the true value $D = \sqrt{74}$. The middle point $D_{\text{mid}} = (D_{\text{min}} + D_{\text{max}})/2 = 8.5$ is our step 1 approximation for $\sqrt{74}$.

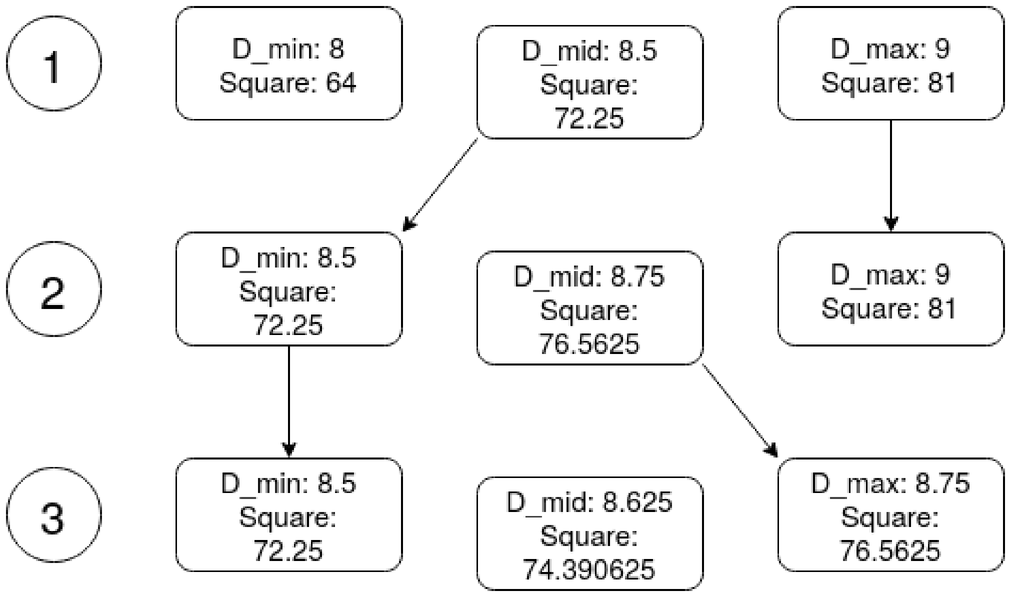


Figure 4.4: the exciting process of computing a square root

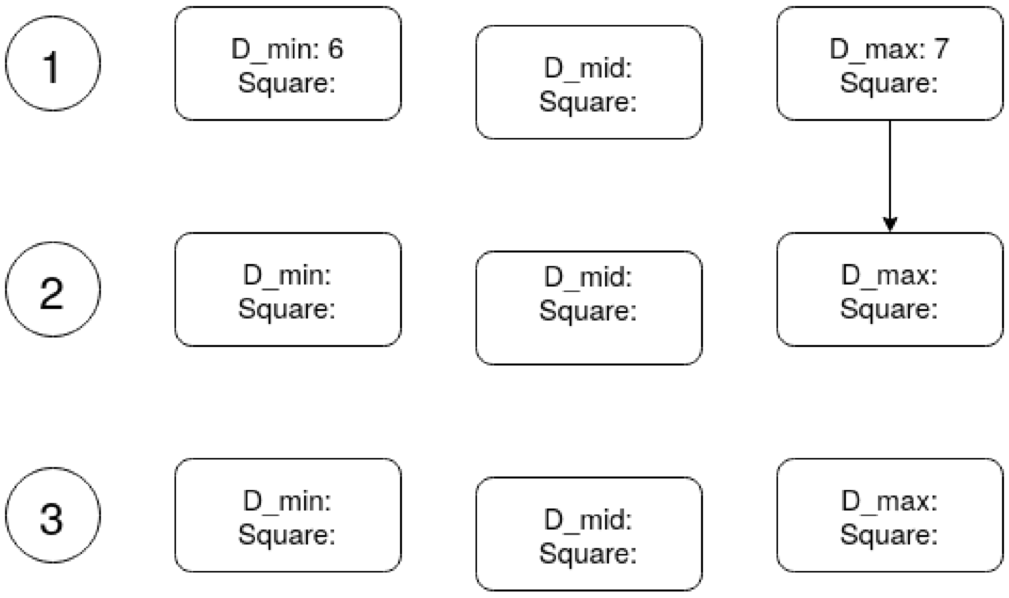
Step 2. Is the middle point 8.5 an upper or a lower bound to D? Well, $8.5 \times 8.5 = 72.25 < 74$. This means that we can use $D_{min} = 8.5$ as our new lower bound to D. $D_{max} = 9$ remains our upper bound. The new middle point, $D_{mid} = (8.5 + 9)/2 = 8.75$ is our step 2 approximation for $\sqrt{74}$.

Step 3. Is the middle point 8.75 an upper or a lower bound to D? Well, $8.75 \times 8.75 = 76.5625 > 74$. This means that we can use $D_{max} = 8.75$ as our new upper bound to D. $D_{min} = 8.5$ remains our lower bound. The new middle point, $D_{mid} = (8.5 + 8.75)/2 = 8.625$ is our step 3 approximation for $\sqrt{74}$.

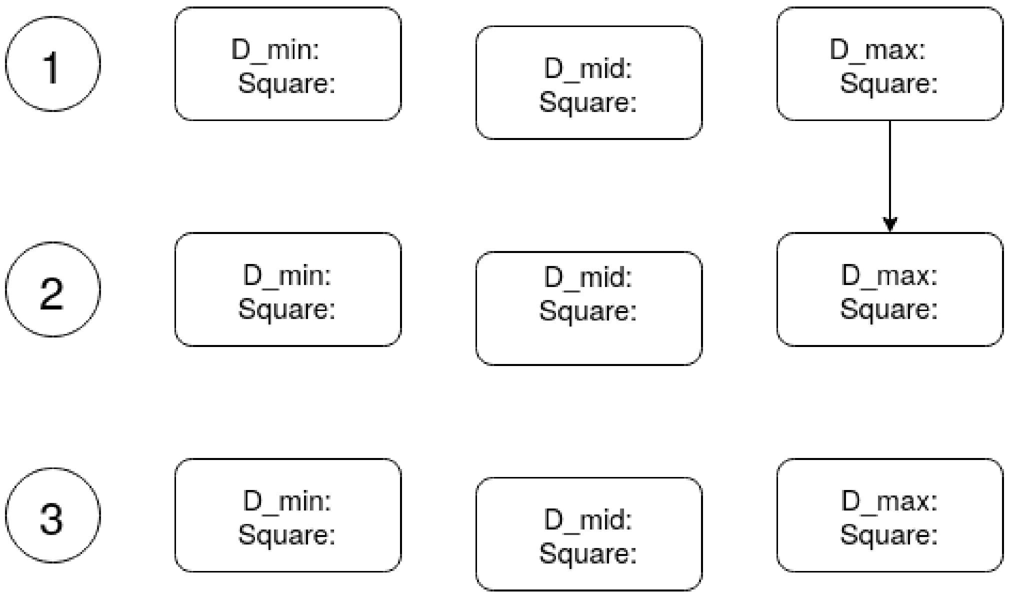
This is a pretty good approximation since $8.625^2 \simeq 74.39$. We could keep iterating and get an arbitrarily precise answer but let's stop here.

Exercise 4.B Digging for Roots

1 Compute an approximation to the square root of 45 using the diagram below

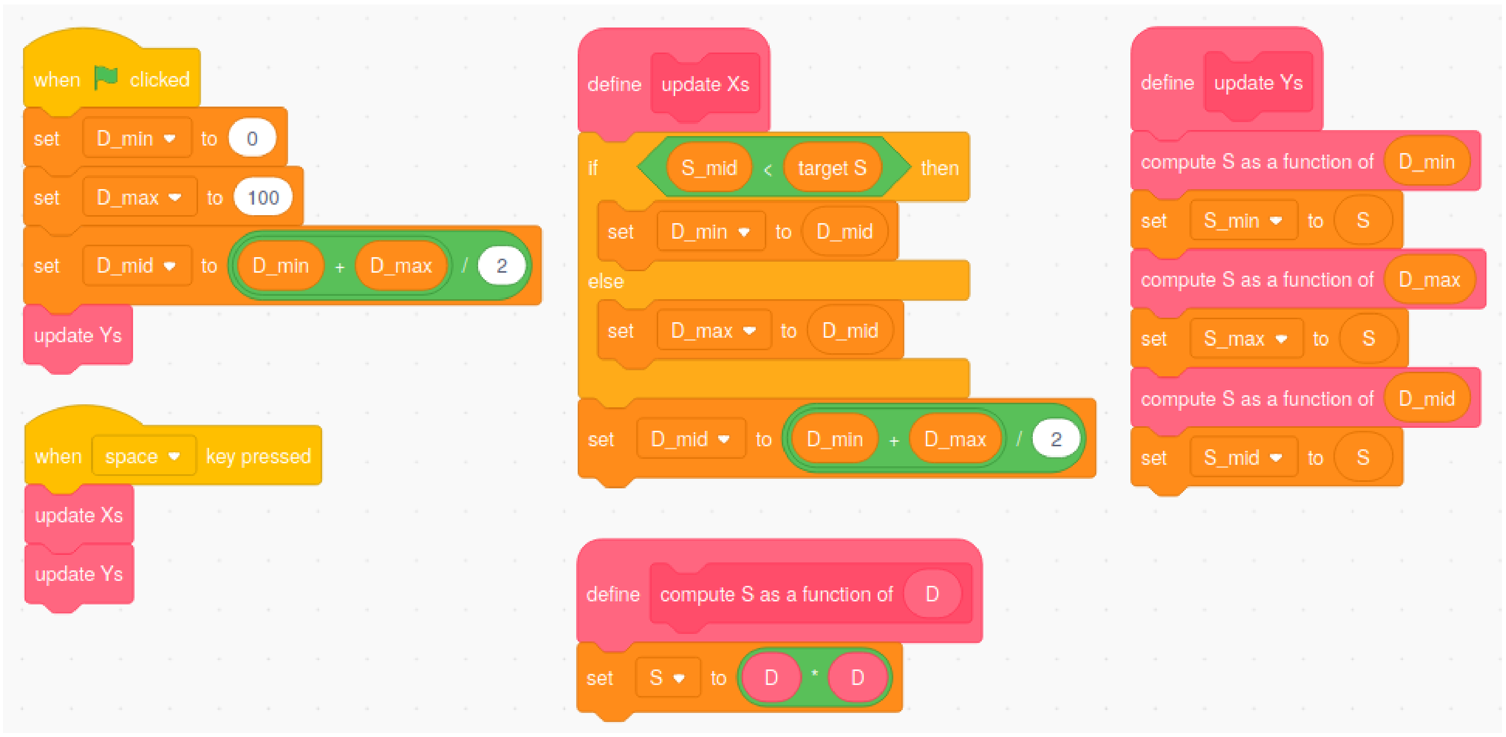


2 Choose a number between 1 and 100 and compute its square root using the diagram below



Let's Code!

We finish this long, long, long chapter by coding up a general bisection solver. The code is described Figure 4.5. You can find a working version here.



```
when 🏁 clicked
set D_min to 0
set D_max to 100
set D_mid to ((D_min + D_max) / 2)
update Ys

when space key pressed
update Xs
update Ys

define update Xs
if <(S_mid) < (target S)> then
    set D_min to D_mid
else
    set D_max to D_mid
set D_mid to ((D_min + D_max) / 2)

define compute S as a function of D
set S to ((D) * (D))

define update Ys
compute S as a function of D_min
set S_min to S
compute S as a function of D_max
set S_max to S
compute S as a function of D_mid
set S_mid to S
```

Figure 4.5: What's going on?

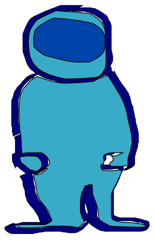
Exercise 4.C — Coding Challenge

1 Read the code. Do you understand how it works? Can you relate it to the process described Figure 4.4?

Use the project to compute the square root of 33. Verify that you got a correct answer.

2 Can you modify the code, and then use the modified program to compute a number D such that $D \times D \times D = 100$?

Chapter 5



The Gravitational Field

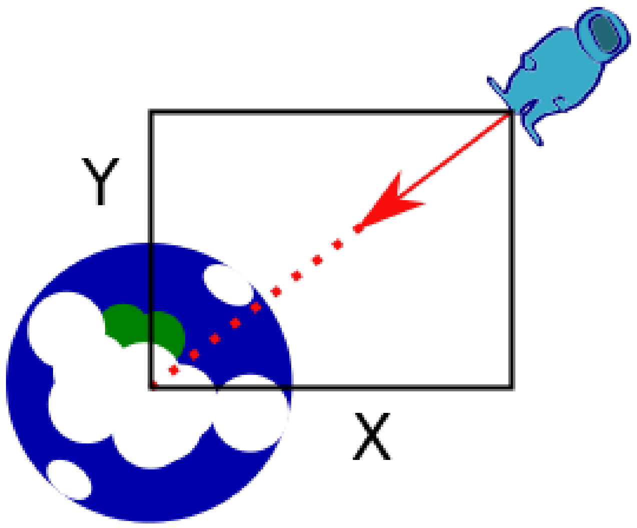


Figure 5.1: Professor Spaceman feeling powerfully attracted to Planet Earth

We are now ready to tackle Isaac Newton's theory of universal gravitation. It's one of the really amazing achievements of human knowledge and ingenuity. It explains how the same forces that make us fall on the ground when we jump also keep the moon up in the sky, following a stable orbit around the earth. It's the first great unification of different phenomena in physics.

Consider the situation of Professor Spaceman in Figure 5.1. If we center on Planet Earth, and Professor Spaceman has position X on the horizontal axis, and position Y on the vertical axis. The law of gravitation says that

> Professor Spaceman accelerates towards the Earth, with an acceleration that is equal to a constant γ (it reads gamma) times the inverse of the square distance D between Professor Spaceman and the earth. In other term the intensity of the acceleration is equal to γ/D^2.

And this is the mystery. If Professor Spaceman (or the Moon for that matter) constantly accelerates towards the earth – i.e. is falling towards the Earth – how can it be that Professor Spaceman (or the Moon) can stay in orbit around the Earth, rather than fall on the ground the way a apple does when it drops from the tree? Well, it's one of these things that is true and totally amazing, like the fact that birds are really dinosaurs. What happens is that Professor Spaceman is falling, but keeps missing the target.

Simulating Professor Spaceman's movement in the gravitational force-field will allow us to solve this mystery. Center the Earth at position $(0, 0)$. First, we need to get a formula for the force field Professor Spaceman is exposed to depending on his position (X, Y). We know from Pythagora's Theorem that the distance D between Professor Spaceman is

$$D = \sqrt{X^2 + Y^2}.$$

To go from Professor Spaceman back to Planet Earth, you need to change Professor Spaceman's position D times by an amount $(-\frac{X}{D}, -\frac{Y}{D})$. Since the distance between Professor Spaceman and the Earth is equal to D, we call $(-\frac{X}{D}, -\frac{Y}{D})$ the unit-length vector pointing from Professor Spaceman towards the Earth.

The universal law of gravitation says that Professor Spaceman acceleration is equal to $\frac{\gamma}{D^2}$ times the unit-length vector pointing from Professor Spaceman to Earth. More specifically, Professor Spaceman's horizontal acceleration AX, and his vertical acceleration AY are:

$$AX = \gamma \times \frac{1}{D^2} \times \left(\frac{-X}{D}\right) = \gamma \times \frac{-X}{D^3}$$
$$AY = \gamma \times \frac{1}{D^2} \times \left(\frac{-Y}{D}\right) = \gamma \times \frac{-Y}{D^3}.$$

D^3 (read it D cube) is just a shortcut for $D \times D \times D$. If $D = 3$, then $D^2 = 2 \times 2 \times 2 = 8$.

Exercise 5.A Compute with Gravity

Assume that $\gamma = 2$. At time $T = 1$, Professor Spaceman is at position $X_1 = 1$, $Y_1 = 2$. His initial speed is $VX_1 = -.2$, $VY_1 = .1$.

1 Compute Professor Spaceman's initial distance to Earth.

2 Compute Professor Spaceman's horizontal acceleration AX_1 at time $T = 1$.

3 Compute Professor Spaceman's vertical acceleration AY_1 at time $T = 1$.

4 Compute Professor Spaceman's speed (VX_2, VY_2) at time $T = 2$.

5 Compute Professor Spaceman's position at time $T = 2$.

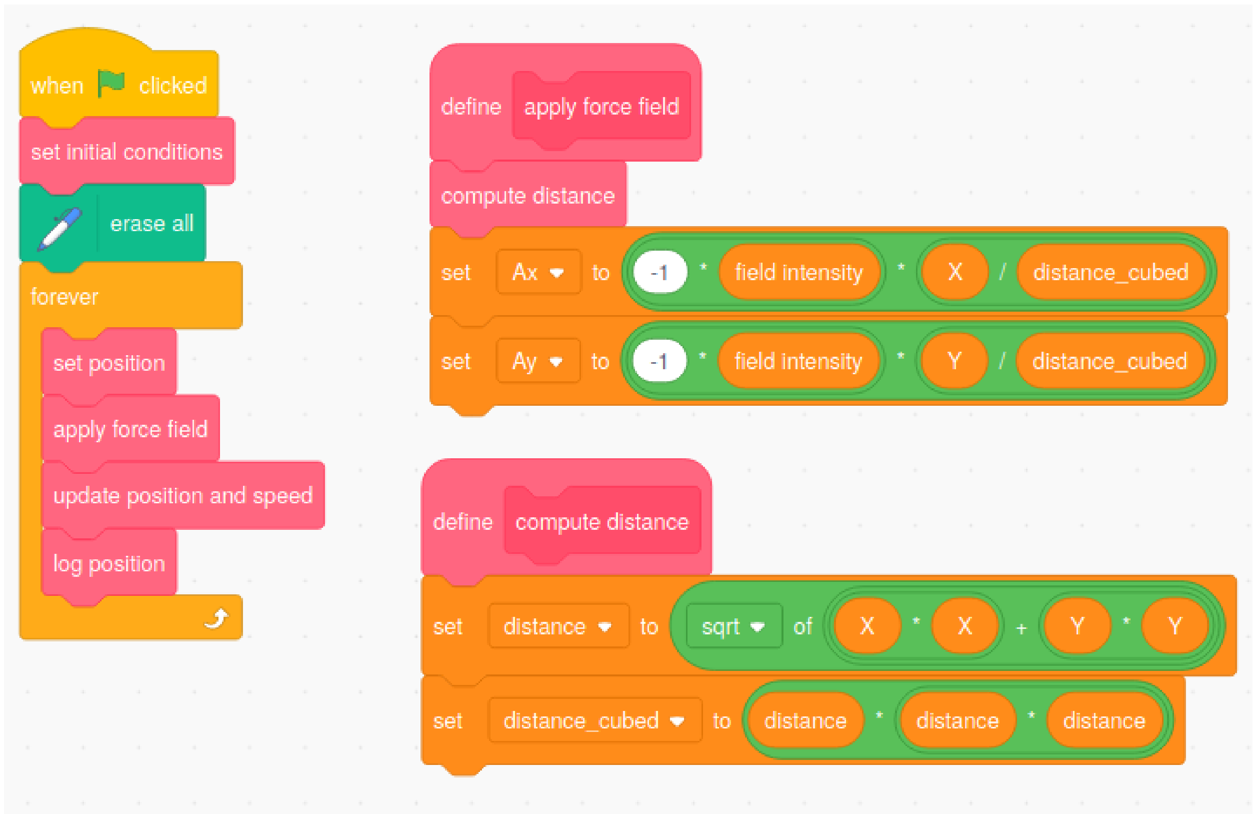
6 Compute Professor Spaceman's position at time $T = 3$.

Let's Code!

Coding up the universal law of gravitation turns out to be straightforward. The specs are as follows:

- Professor Spaceman starts at a fixed position on the horizontal axis.
- The user can specify Professor Spaceman's initial horizontal speed VX, and vertical speed VY.
- The user can specify constant γ, also called the "the field intensity."
- Professor Spaceman moves according to the law of gravitation.
- Professor Spaceman's trajectory will be drawn in a color that changes with his distance to Earth.

The important parts of the code are illustrated by Figure 5.2 (the project code is available here).



```
when [flag] clicked
set initial conditions
erase all
forever
    set position
    apply force field
    update position and speed
    log position
```

```
define apply force field
compute distance
set Ax to ((-1 * field intensity) * (X / distance_cubed))
set Ay to ((-1 * field intensity) * (Y / distance_cubed))
```

```
define compute distance
set distance to sqrt of ((X * X) + (Y * Y))
set distance_cubed to (distance * (distance * distance))
```

Figure 5.2: What's going on?

A few modules – `set initial conditions`, `set position`, `log position`, and `update position and speed` – are not described in Figure 5.2 because they are very similar to the code of Figures 2.2 and 3.2.

Exercise 5.B — Coding Mystery

1 Read the code of Figure 5.2 from beginning to end. How is it different from the code of Figure 3.2? Where are the laws of gravity described? Where are we using a square root?

2 Write down what you think the commands inside the `update position and speed` module are.

3 Write down what you think the commands of the `log position` module are. Remember that we want the color of the trajectory to correspond to Professor Spaceman's distance to the Earth.

4 Check the project code. Did you correctly guess the content of the `update position and speed` and `log position` modules.

As Figure 5.3 shows, depending on Professor Spaceman's initial speed, and on the force field you choose, Professor Spaceman's trajectory in space will fall in two categories:

- he will stick to a stable cyclical roundish orbit around the Earth;
- alternatively, he may get close to the Earth (or not), but ultimately will get far away, and never come back (concretely, he'll stay stuck on one side of the Scratch window and never come back).

In the first case, Professor Spaceman's trajectory is called an Ellipsis. In the second case, it is called a Hyperbola.

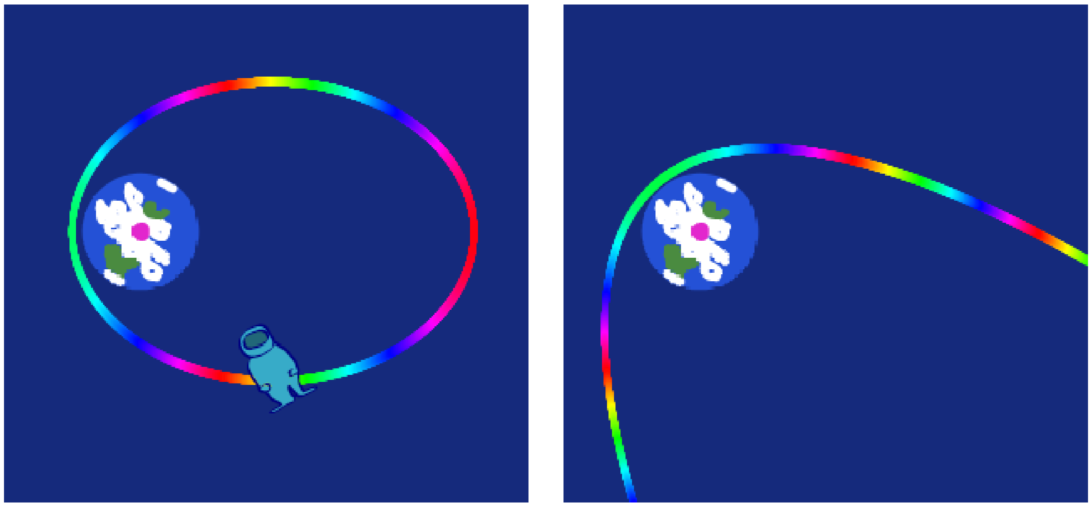

(a) An Elliptical Trajectory (b) A Hyperbolic Trajectory

Figure 5.3: Will Professor Spaceman get back home for dinner?

These trajectories are essentially the same as the trajectories of planets and comets orbiting around the Sun in our solar system. The trajectories of planets are ellipses. Comets whose trajectory is an ellipsis, like Halley's Comet are called periodic. Comets whose trajectory is a hyperbola, like the Great Comet of 1577 are called non-periodic comets. They visit the Earth once, and never come back.

The famous astronomers Tycho Brahe, and Johannes Kepler got to see the Great Comet of 1577 (Kepler was a kid, his mom took him to see it). Since it will never come back again, I think they were pretty lucky. Johannes Kepler also got to see Halley's comet in 1607, and I got to see it in 1986. William the Conqueror saw it in 1066, before invading England. I think it's pretty cool to be linked to people who have watched the same comet hundreds of years ago. Halley's comet will come back in 2061 – I hope to see it again!

Exercise 5.C Empirical Exploration

1 Set $VX = 0$. Can you find an initial value of VY such that Professor Spaceman's trajectory has the same color throughout? What is the name of this trajectory?

2 Can you find initial parameters such that Professor Spaceman's trajectory is an almond shaped ellipsis? Wait for the trajectory to complete.

Add a little pulse to Professor Spaceman's speed by moving the VX speed slider **a little bit**. What happens to Professor Spaceman's trajectory? Does it remain an ellipsis?

3 Find initial parameters such that Professor Spaceman's trajectory is nice almond shaped ellipsis. Wait for the trajectory to complete. Take a screenshot of Professor Spaceman's trajectory a save it as a file. Open the file in your favorite graphical editor.

Mark the points where Professor Spaceman in closest and further away from the Earth. The closest point is called the Apogee, the furthest point is called the Perigee.

4 Using a stopwatch, or a timer, measure the time T in seconds it takes for Professor Spaceman to go from the Apogee and come back, making a complete orbit. What is T? Does Professor Spaceman move faster at the Apogee, or the Perigee?

On the screenshot you took, mark the position of Professor Spaceman $T/4$, $T/2$ and $3T/4$ seconds after leaving the Apogee.

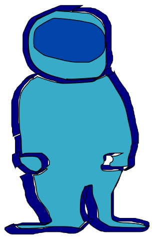
Draw the 4 line segments going from the points you've marked to the center of the Earth. This cuts the ellipsis in 4 zones. Fill each zone with a different color. Do the areas of the 4 different zones look similar or different?

Kepler's Second Law of Planetary Motion says that

> A line joining a planet and the Sun sweeps out equal areas during equal intervals of time.

How does it relate to your finding?

Chapter 6



Two Easy Pieces

OK, the previous Chapter was long. So now we take a little break and have fun with two easy projects.

Fantasy Gravitation

First, we imagine a completely different universe in which the gravitational field is not proportional to the inverse of the distance square, $1/D^2$. Instead, imagine the gravitational field is proportional to the inverse of the distance, $1/D$. What would the trajectory of the Earth around the Sun (or Professor Spaceman's trajectory around the Earth) look like?

Well, you can pretty much do it on your own.

Exercise 6.A **Coding Challenge**

1 What part of the code of Figure 5.2 do you need to change?

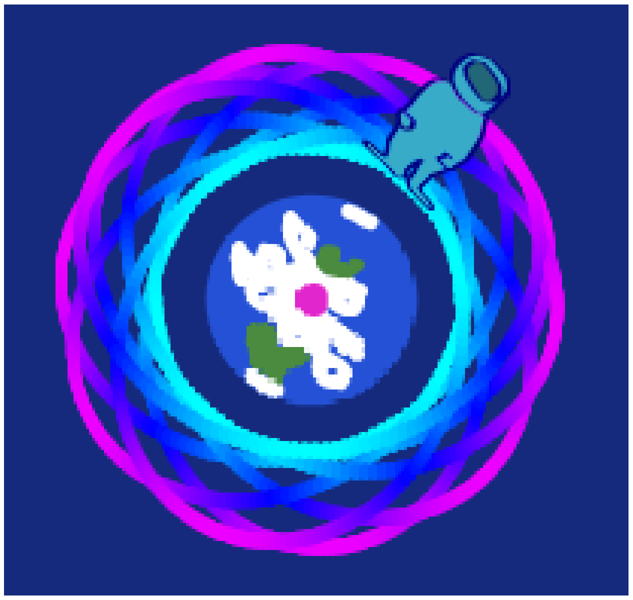


Figure 6.1: Professor Spaceman, wondering why gravity feels different today

2 Create a copy of the previous gravitation project, and change the code in the way you think is correct.

Check out the code here. Does it correspond to what you wrote?

3 Play around with parameters. How would you describe Professor Spaceman's trajectory to someone else? Write a short paragraph.

A Repulsive Force Field

Coulomb's Law describes the interaction between electrically charged particles. It turns out that like gravity: (i) the electrostatic force is parallel to the line segment between the two particles; (ii) it is inversely proportional to the square distance between the two particles.

However, while the force of gravity always pushes objects towards one another, the electrostatic force is attractive only when one object is positively charged while the other is negatively charged. If both objects are positively charged or negatively charged, then the force is repulsive: it pushes the objects away from one another.

How would you simulate the movement of a negatively charged magnet moving near another **fixed** negatively charged magnet?

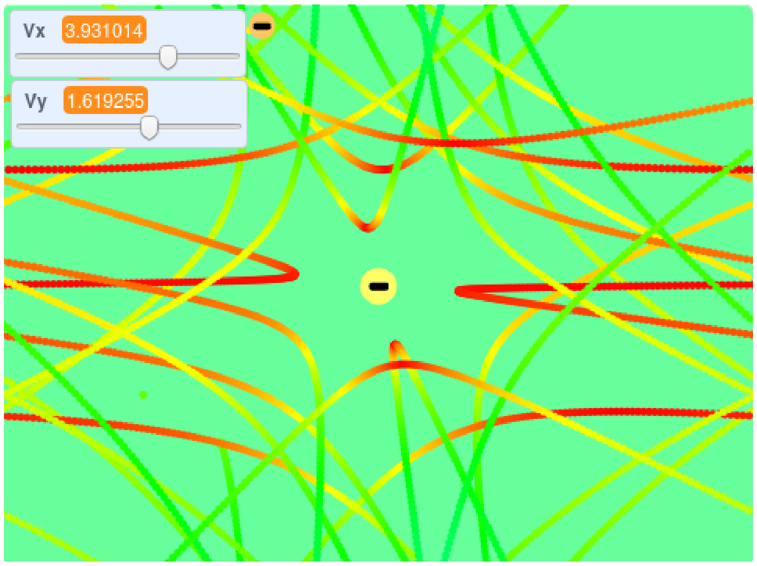


Figure 6.2: Fun with magnets

Exercise 6.B — Coding Challenge

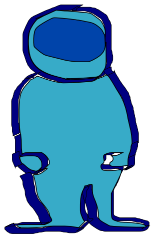
1 What part of the code of Figure 5.2 do you need to change?

2 Create a copy of the previous gravitation project, and change the code in the way you think is correct.

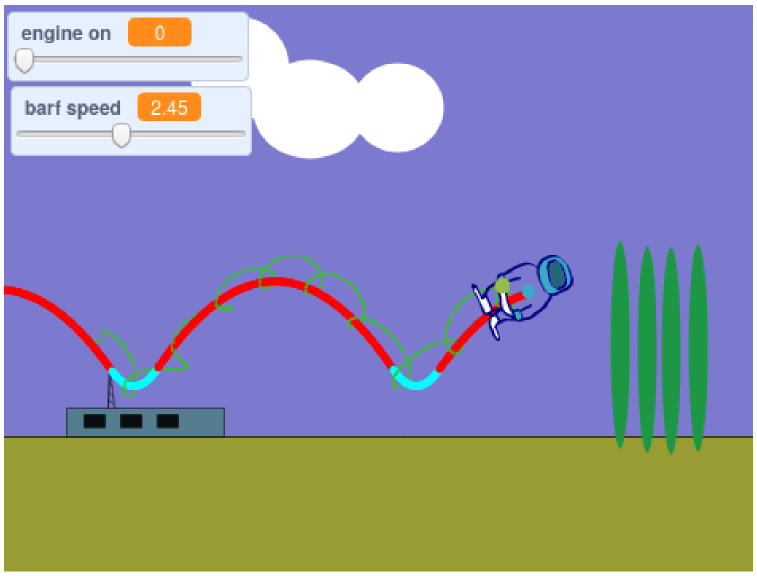
Check out the code here. Does it correspond to what you wrote?

3 How would you simulate the movement of a negatively charged magnet moving near a positively charged magnet?

Chapter 7



The Vomit Comet Earns Its Name



engine on 0

barf speed 2.45

Figure 7.1: Professor Spaceman, having an unpleasant dream

Professor Spaceman is having a nightmare about flying on the vomit comet. First, he somehow showed up to NASA in his underwear...Second, he had three extra-cheese pizzas for breakfast and the acceleration is making him feeling queasy. He...is...about...to...barf!

In this chapter, we will simulate the trajectory of Professor Spaceman's barf, as Professor Spaceman experiences the joy of free fall. For the simulation, we are going to to assume that both Professor Spaceman and his barf both accelerate upwards when the engine is on. This is what would happen if the engine was really a big fan on the ground, like in a vertical wind tunnel.

Professor Spaceman and his barf evolve in the following force fields:

1. Professor Spaceman evolves in the vomit comet force field described in Chapter 3;
2. The barf evolves in a force field combining both the vomit comet force field described Chapter 3 and the gravitational force field generated by Professor Spaceman, similar to the force field generated by Earth's gravity in Chapter 5.

Note that because Professor Spaceman is much heavier than his barf we are neglecting the impact of the barf's gravity on Professor Spaceman's trajectory. We are also ignoring the propulsion experienced by Professor Spaceman at the time he barfs.

Exercise 7.A — Relative Distance

1 Say that Professor Spaceman has coordinates $X = 50$ and $Y = 70$, while the barf has coordinates $X' = 80$ and $Y' = 30$. What is the horizontal and vertical movement ΔX and ΔY needed to go from Professor Spaceman to his barf?

2 What is the distance between Professor Spaceman and his barf?

Let's Code!

Our specs for the code are as follows:

- Professor Spaceman evolves according to the code of Chapter 3;
- Professor Spaceman barfs when the user presses the space bar;
- The barf evolves according to a force field that combines the force field of the vomit comet, and the force field generated by Professor Spaceman's gravity.

As far as Scratch code goes, Professor Spaceman and the barf will correspond to two different sprites. The code for the barf sprite is described in Figure 7.2

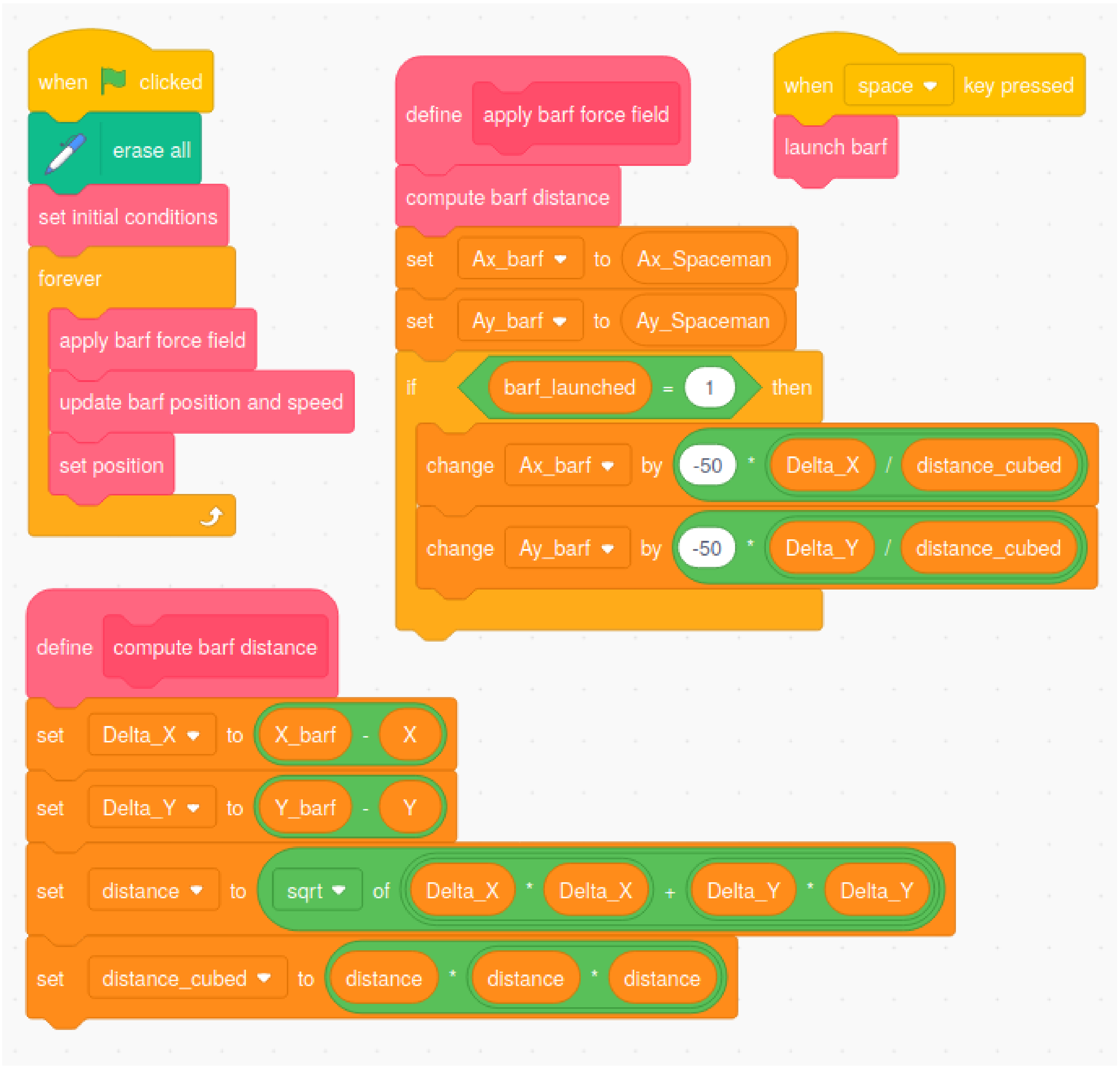
Exercise 7.B — Coding Mystery

1 Read the code of Figure 7.2 from beginning to end. How is it different from the code of Figure 5.2?

2 The role of the `launch barf` module is to increase the barf's horizontal speed by a small amount. Write what you think is inside the `launch barf` module.

3 Try and draw what you think the trajectory of Professor Spaceman and his barf will be.

4 Checkout the project online and play with it for a while. Does the code correspond to what you expected?



```
when [flag] clicked
erase all
set initial conditions
forever
    apply barf force field
    update barf position and speed
    set position
```

```
define apply barf force field
compute barf distance
set Ax_barf to Ax_Spaceman
set Ay_barf to Ay_Spaceman
if barf_launched = 1 then
    change Ax_barf by -50 * Delta_X / distance_cubed
    change Ay_barf by -50 * Delta_Y / distance_cubed
```

```
when space key pressed
launch barf
```

```
define compute barf distance
set Delta_X to X_barf - X
set Delta_Y to Y_barf - Y
set distance to sqrt of Delta_X * Delta_X + Delta_Y * Delta_Y
set distance_cubed to distance * distance * distance
```

Figure 7.2: The serious business of modeling barf

Try launching the barf at different points of Professor Spaceman's trajectory. What different types of barf trajectories do you see?

5 A dynamic system is said to be ‘chaotic’ if slightly different initial conditions lead to very different behavior. Is the barf’s trajectory chaotic? In contrast is the trajectory of Professor Spaceman in Chapter 5 chaotic?

Exercise 7.C

Coding Challenge

1 Can you simulate the trajectory of the barf under the fantasy gravity of Chapter 6?

2 Can you change the color of the barf’s trajectory so that it changes with the horizontal speed of the barf?

Chapter 8

Sun, Earth, and Moon

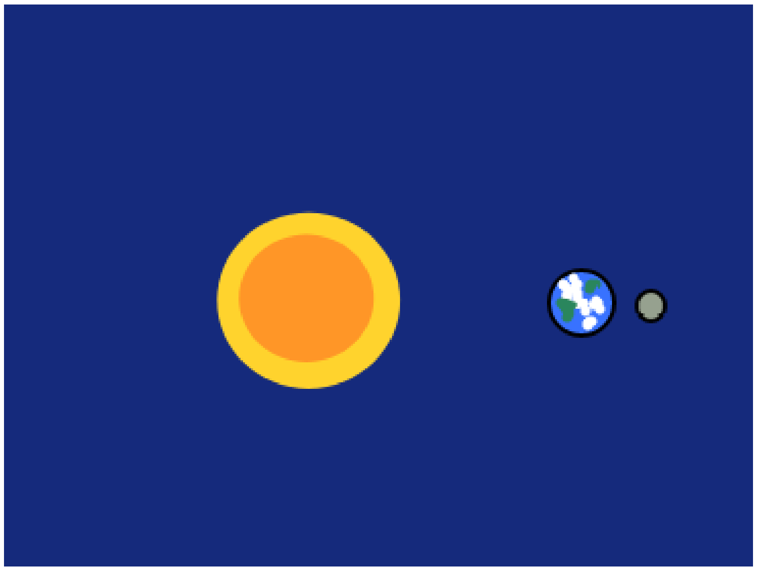


Figure 8.1: Time to shine...

In this Chapter you are on your own! The challenge is to simulate the motion of the Earth orbiting around the Sun, and of the Moon orbiting around the Earth and Sun. This can be done by reusing parts from previous projects:

- The motion of the Earth around the Sun roughly corresponds to the motion of Professor Spaceman around the Earth in Chapter 5.

- The motion of the Moon around the Earth (and the Sun) roughly corresponds to the motion of the barf around Professor Spaceman in Chapter 7

It's OK to make a few simplifications:

- First, because the Sun is much heavier than the Earth, and the Earth is much heavier than the Moon, we are ignoring the gravitational pull of the Earth on the Sun, and the pull of the Moon on the Earth (in practice, the Moon has a significant influence on the Earth: tides).
- Second, we assume that the distance between the Moon and the Sun is roughly equal to the distance between the Earth and the Sun, so that the force field of the Sun's gravity is the same for the Earth and the Moon.

The specs for the project are the following:

1. The Earth's acceleration is set by the gravitational field of the Sun;
2. The Moon's acceleration is the sum of accelerations from the Sun's gravity and the Earth's gravity;
3. The intensity of the Earth's force field and the Sun's force field can be set using sliders;
4. The Earth's trajectory is painted in blue;
5. The Moon's trajectory is painted in gold.

Exercise 8.A

Ultimate Coding Challenge

1 Reusing code from Chapters 5 and 7, simulate the trajectory of the Earth and the Moon around the Sun.

2 How would you modify the code to correctly reflect the fact that the Sun's gravitational pull on the Moon depends on the distance between the Moon and the Sun, rather than the Earth and the Sun?

Implement those changes in a new project. Does the Moon always stay close to the Earth?