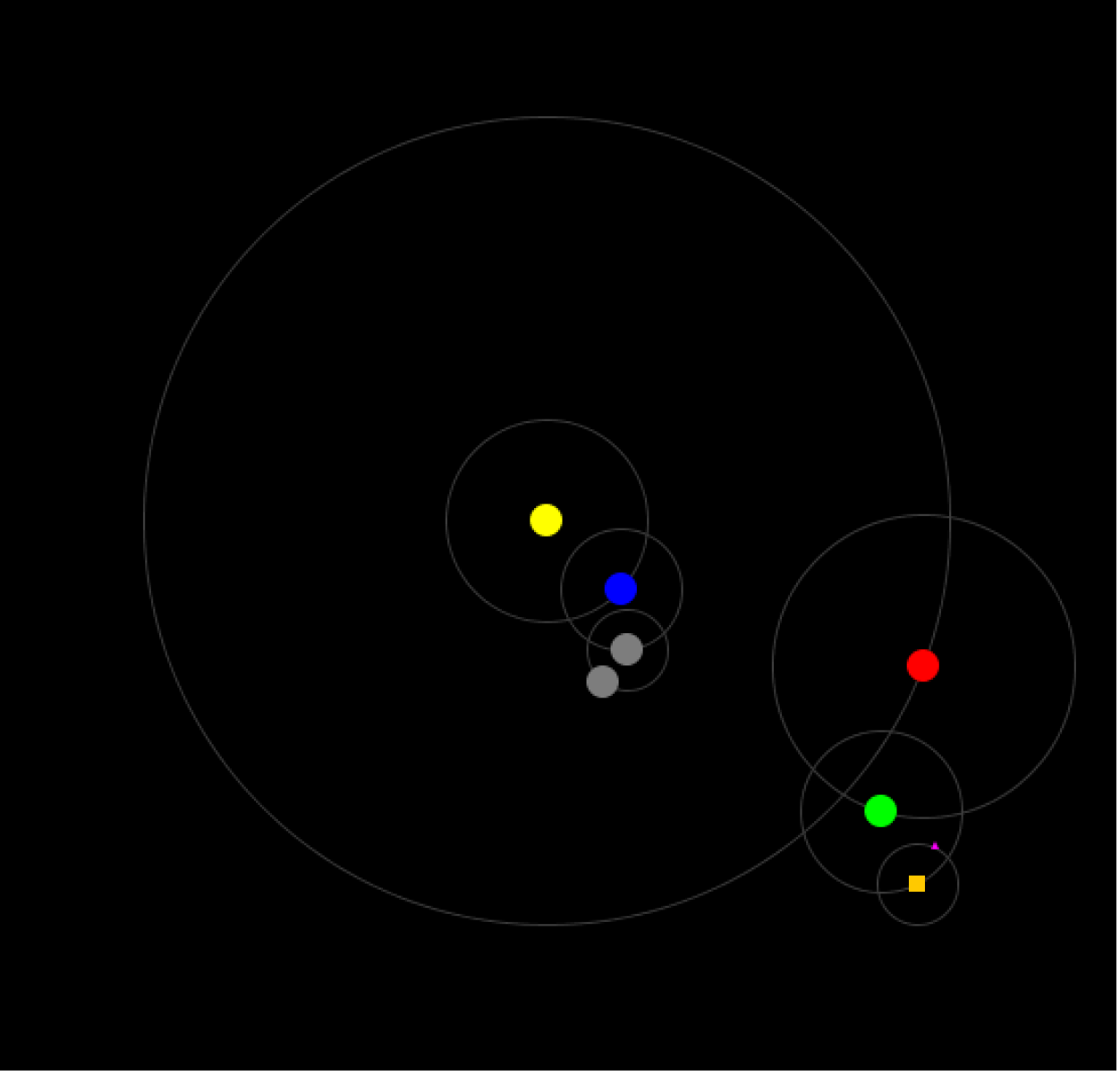


Orbiters

CS 327 Fall ’17 Prof. Bowers

You are going to code a system for simulating the orbits of planets and their moons, satellites, and exploratory space ships.

See the screenshot at the top of this document.

Learning objectives

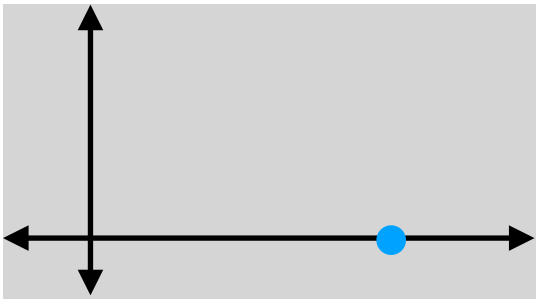
- Understand and code basic matrix operations.
- Use homogeneous coordinates to represent positions of objects.
- Use matrix operations to compute the positions of objects in a hierarchical structure.
- Understand the basics of hierarchical modeling used in graphs.

Hierarchical motions

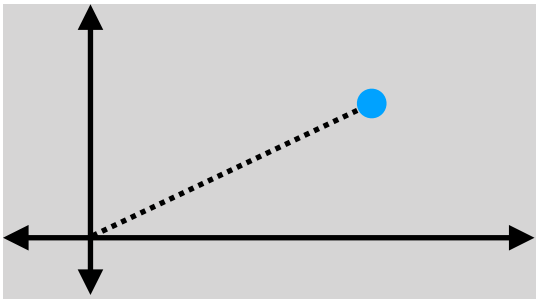
Suppose we want to represent the position of a planet orbiting the sun in space. Eventually, we'll need to obtain (x, y) coordinates for the position of the planet based on θ, the current rotation angle and r, the radius of the planets orbit.

One way to accomplish this is through linear algebra. Let's imagine that we have a little drawing "cursor" at the origin of our space $(0,0)$. We want to be able to apply both translations and rotations to this cursor, and as we noted in class, a very convenient method for accomplishing this is to use homogeneous coordinates. Thus, let our cursor be defined as $c = [0,0,1]^T$. To draw our planet, we need to move our cursor r units to the right and then rotate by an angle of θ. Something like this:

We now move the blue cursor r units to the right:



And now rotate around the origin by θ.



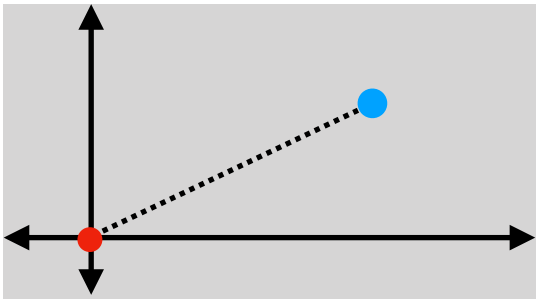
Now, let R_θ and $T_{x,y}$ be matrices of rotation by θ and translation by x units in the x-direction and y units in the y-direction. Recall that these are defined by

$$T_{x,y} = \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix} \text{ and } R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Then we can obtain the final position of of the center of the planet by the matrix multiplication (dot product):

$$R_\theta T_{r,0} c$$

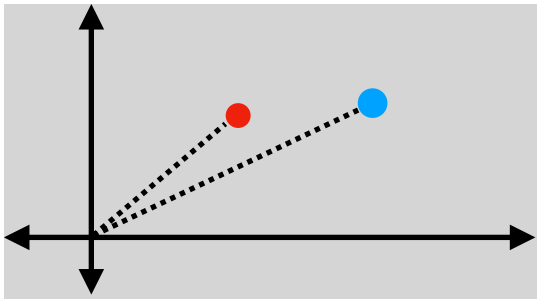
Now, what if the new origin is really a second planet (shown in red) about which the blue one is orbiting (the blue one is just a moon, it turns out)? The red planet is itself, rotating around some sun. How do we place them both?



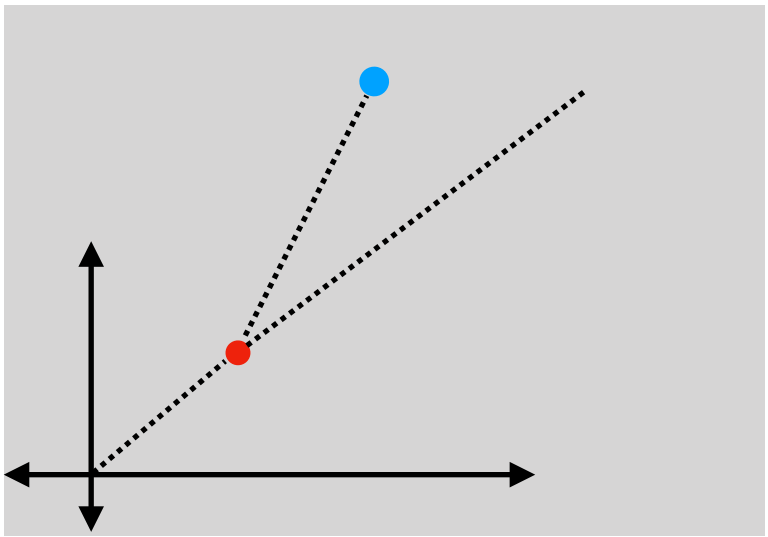
Let's assume that the red planet is at an orbital radius of r' and orbital angle of θ'. We can get the position of the red planet relative to its orbital center (the sun) using the same formula as above:

$$R_{\theta'} T_{r',0} c$$

So the result now is:



But notice the blue planet hasn't moved. We need to also apply $R_{\theta'} T_{r',0}$ to the location of the blue planet, because before we had only calculated the location of the blue planet *relative to its*



parent. Thus the location of the blue planet becomes $R_{\theta'}T_{r',0}R_{\theta}T_{r,0}c$ and we obtain the following figure.

So in general, if we want to know the location of some satellite that is on an orbit of radius r and current angle θ, and its parent is at r' and θ' relative to its parent, and that is at r'' and θ'' relative to its parent, which is at r''' and θ''', etc. we multiply the matrices alternating from rotation to translation matrix starting with the root ancestor of our orbital system and move all the way down to the planet we want to draw:

$$\cdots R_{\theta'''}T_{r''',0}R_{\theta''}T_{r'',0}R_{\theta'}T_{r',0}R_{\theta}T_{r,0}c$$

Orbiters Tree Structure

In our simplified universe, each object, like a moon, is orbiting some parent object, and is not affected by the gravity from any other object. Each orbital system has a single sun at the center of the system. Thus, the gravitational attractions of the system are modeled as a tree structure of Orbiters with the sun as the root node.

Your Task

Download the Orbit.zip starter file from Canvas. The starter code is written in Processing, an IDE and subset of the Java language that makes creating graphical programs very easy. There are lots of good tutorials on Processing available online. My own quick tutorial can be found here: https://w3.cs.jmu.edu/bowersjc/page/courses/spring17/cs480/labs/processing1/. Italso includes some computational geometry at the end (which you can ignore).

You will need to download the Processing IDE from http://processing.org to open the included pde files. The files are:

- `Orbit.pde`. This is the main file for the program, which takes the place of a main method in a typical Java program. Processing requires two methods appear in this file—`setup()` which sets up the graphical environment, and `draw()` which draws each frame of the animation to the screen.

    - You will need to code the `drawOrbiters()` and `updateOrbiters(timeDelta)` methods. These both traverse the orbiters tree structure. The `updateOrbiters(timeDelta)` traverses the Orbiters tree structure in order to call `updateRotation(timeDelta)` on each orbiter while the `drawOrbiters()` method traverses the tree structure and passes each orbiter to the `drawOrbiter(orbiter)` method defined at the end of Orbit.pde.
- `Matrix.java`. A class implementing a matrix with a set of matrix operations, like the dot (multiplication) operator. You will need to implement all of the methods based on their javadoc comments.
- `Orbiter.java`. The Orbiter tree structure. You will need to implement the `getMatrix()` method based on the TODO comment.
- `UndefinedMatrixOpException.java`. This contains a class for the exception your code should throw if any matrix operation is performed on matrices of the wrong size. (For instance, when adding two matrices, they must have the same number of rows and columns.)

Turning it in

ZIP up your code Orbit folder and submit it on Canvas. The due date is available on Canvas.