Homework 14: Animation

Project Homework

Instructions:

This homework is different than the homework assignments that you have done before. Instead of having many problems to solve with different functions, you will be graded only on one function (Problem 3, `solarSystem`), which will use `makePlanet` and `rotateOrbit` to animate a scene of a solar system in motion.

Instead, problems 1 and 2 are designed to assist you in creating separate components (e.g. planets, orbits) that will be used together in the `solarSytem` animation function. These problems are not required in order to receive full credit on this homework. However, it is *highly recommended* that you utilize them in your `solarSystem` function. They will be helpful in breaking down the final animation into manageable components, and they will also help organize your code. In addition, if you correctly complete any of problems 1 and 2, you will receive partial credit even if your `solarSystem` function does not work correctly.

Notes:

You have a lot of freedom in this project to change things up and be creative. That being said, you can change up the inputs and outputs of the helper functions provided as the first couple of drill problems to create what you need for the final animation. Further, you can create additional functions if you want to do more than what the provided function guidelines offer.

Grading Scheme:

Unlike previous homework assignments, this homework will not be graded with the auto grader. Instead, the `solarSystem` function will be hand-graded by your section TAs using the rubric provided below. Seventy percent (70%) of your grade for this homework will come from this hand-graded score.

The remaining thirty percent (30%) of your grade will be based on your ability to demonstrate and explain how your `solarSystem` function works to your section TA during a demo. This demo will be done in recitation during the week before finals (commonly referred to as Dead Week). The demo gives you a chance to boost your homework score, even if there were some errors in your code when it was submitted. If you are able to knowledgeably answer your TA's questions about how your code works or explain any errors in your code, you will receive full credit on the demo.

Make sure you look through the rubric provided to ensure you have fulfilled all the requirements for this project homework.

Extra Credit:

The animation aspect of this homework will give you an opportunity to be creative. If you go above and beyond what the problem statement specifies, there is potential to receive extra credit. Extra credit will be awarded by your TA based on how many creative additions you make to the animation. You **must** include a commented statement at the bottom of your `solarSystem` file (below all of the code) detailing the additions that you made *or that you did not make any additions*. ***The TAs are NOT responsible for helping you implement your extra credit!***

Example Extra Credit Additions:

- Create more planets than required
- Add a background
- Put stars in the animation
- Make the planetary orbits non-circular
- We have included some images of planets in a .zip file if you would like to use them in a creative way

Disclaimer:

Not all of the material on this homework will be functions we teach you in class. Most of MATLAB is like that. You have all the basic skills required to do this, but use the Internet/textbook/MATLAB help resources to figure out the specifics. This assignment gives you the opportunity to go beyond, read the documentation on certain functions and functionality, and have fun with it!

Good Luck!

Project Homework: Problem 1

Function Name: makePlanet

Inputs:

1. (*double*) A vector indicating the position of the planet is to be located
2. (*double*) A number indicating the radius

Outputs:

(*none*)

Plot Outputs:

1. A plot of the planet

Function Description:

Write a function called `makePlanet` that takes in a vector of the x, y, z coordinates for a planet's location, as well as a number indicating the specified planet's radius, and creates a plot of a planet. You will most likely want to create a spherical shape for the planet.

Notes:

- You may find the `sphere()` function very useful.
- You may find the `surf()` function useful in this problem.
- When calling multiple plotting functions, `hold on` operates in the same way for the `surf()` function as for normal plotting operations.

Project Homework: Problem 2

Function Name: rotateOrbit

Inputs:
1. (*double*) A (column) vector indicating the position of the planet to be rotated
2. (*double*) An angle (in degrees) through which to rotate the planet on its orbit

Outputs:
1. (*double*) A vector indicating the new position of the planet

Function Description:

Write a function called `rotateOrbit` that takes in a vector of the x, y, z coordinates for a planet's location, as well as an angle theta through which to rotate a planet on its orbit, and outputs a vector of the planet's new x, y, and z coordinates. You will want use rotation matrices to rotate your planet.

Rotating coordinates in 3D is very similar to rotating in two dimensions; you just have a different rotation matrix. The counterclockwise rotation matrix for three dimensions is as follows:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Notes:
- You may **not** use the `rotate()` function for this problem.
- A shape must be displaced from the origin to see the result of a rotation.
- For the sake of getting the same results as the test cases, you will want to use a counter-clockwise rotation matrix, but for your final `solarSystem` function, you can rotate your planets in either direction.

Project Homework Rubric

Function Name: solarSystem

Inputs:

You are not required to have any inputs to this function. For creativities sake, we have provided possible inputs you could implement in your code. If your code requires inputs, make sure you include commented instructions on how to use those inputs and an example input for the function below where you list your extra credit.

Outputs:

(*none*)

Plot Outputs:

1. Animated plot of solar system

Function Description:

This is the function you will be graded on overall for this homework assignment. Now that you have written functions to create planets and rotate those planets in an orbit, write a MATLAB program that will produce an animated plot of the solar system.

Because creating an animation to incorporate your other functions is challenging, skeleton code has been provided for you inside `solarSystem.m`. This skeleton code contains comments that should guide you in using your other functions from this homework to create the animation.

The file `solarSystem_sample.p` has also been provided for you to see an example of what your `solarSystem` animation may look like.

Requirements for `solarSystem`:

- You must have at least 3 rotating planets.
- You must have one object that is not a rotating planet.
- The planets must be solid objects.
- The planets cannot be the default color.
- The planets must be visible on the screen throughout the entire animation.

- Axes are turned off.
- Comments are at the bottom of the file that outline creative changes made or a statement that no creative changes were made.
- Directions are included on how to run your function if you add inputs to the function.