CS 132 – Geometric Algorithms

Homework 7

Optional; No Due Date
May be Submitted Any Time before April 30, on Gradescope

Programming Assignment. For this assignment, you will implement a set of functions in `python` to create a simple animated scene.

I have provided code in `hwk7.py` that creates the animation. Your job is to supply the propert set of matrices to implement the linear transformations that will define the motion of the objects in the animation.

The animation consists of a house (which is stationary) and a ball. Initially the house and ball are positioned for you as shown in Figure 1.

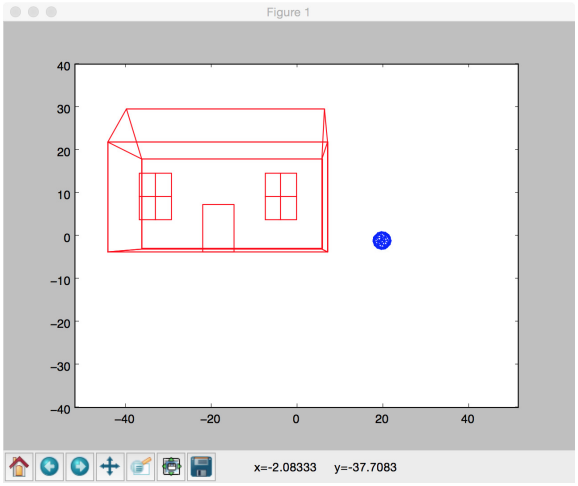

Figure 1: Initial Scene.

The sequence of movements is as follows:

- From time $i = 0$ to 49, the ball rolls toward the observer. Specifically, at each timestep the ball moves one-half foot toward the observer.
- From time $i = 50$ to 64, the ball then rolls toward the house (starting from its position at the end of timestep 50). For this, it moves faster: 2 feet per timestep.
- From time $i = 65$ to 149, the camera rotates once around the entire scene, remaining pointed toward the origin.

The coordinates of the ball and house are the columns of matrices `ball` and `house`.[1] All coordinates are given in feet. **All coordinates are homogeneous coordinates** as described in lecture and the book.

For each timestep, your job is to compute the proper matrix for transforming the columns of `ball`, and the proper matrix for transforming the columns of `house`.

[1] Actually these are *lists* of matrices, where each matrix corresponds to one face of an object. But you don't need to worry about that – the scaffolding keeps track of it.

The scaffolding that I provide will do the following for the ball, at timestep *i*: (in pseudocode)

```
M = ballTransform(i,ballLocation)
ballCurrent = M.dot(ball)
plot(ballCurrent)
```

A similar process is performed for the house.

You are to implement the routines `ballTransform()` and `houseTransform()` which each return a 4×4 matrix that captures all the movement, rotation, and perspective transformation needed for whatever the scene should look like at timestep *i*. Note that the coordinates of `ball` itself in the above pseudocode and the location of `ball` never change. In other words, all transformations are defined in terms of the original starting positions of the ball and the house.

You can see a video of what the final animation will look like by viewing 'Homework 7' on my YouTube channel (`https://youtu.be/orEH1cy1mK4`).

Walkthrough. Here is a suggested plan of attack. To start with, you need to download the files `hwk7.py`, `obj2clist.py`, `snub_icosidodecahedron.wrl`, and `basicHouse.obj`. All your edits will be to `hwk7.py`, and will consist of completing function definitions for which the templates are already provided.

1. First, implement the projection matrix which will transform the 3-D coordinates into 2-D suitable for plotting. This is the function `project()`. The camera is at location (0,0,*d*) and is looking toward the *xy* plane, just as described in the book. The book describes fairly exactly what this projection matrix should be.

2. Next, use `project()` to create a projection matrix corresponding to putting the viewer at location (0,0,100.0). Use the projection matrix to get a static view of the ball and house in their starting positions. In other words, implement your first version of `ballTransform()` and `houseTransform()` which both simply return the appropriate projection matrix. Stop and test.

3. Next, implement the function `moveTo(start, end)` which takes a starting position and an ending position and returns the matrix that translates an object from `start` to `end`. This should be fairly clear from the book. Test this function by using it to create some matrices and inspect them to make sure they are correct.

4. Next, modify `ballTransform` so that for each timestep from 0 through 49, the ball moves 1/2 foot toward the observer, i.e., in the positive *z* direction. Use `moveTo()` to create the needed matrix. Don't forget that you still need the projection as well. Test.

5. Next, modify `ballTransform` so that for timesteps 50 through 64, the ball moves 2 feet in the negative *x* direction (starting from where the ball was at time $i = 50$.) Test. At this point the scene should look like Figure 2.

6. Next, create the function `rotate(x,y,z,loc)`. This function returns the matrix corresponding to first rotating *x* radians around the *x*-axis, then rotating *y* radians around the *y*-axis, and then rotating *z* radians around the *z*-axis. The center of rotation is at point given by `loc`, which is in 3D homogeneous coordinates. Use this to cause the ball to rotate in the proper direction as it moves, so that it looks like it is rolling. Test.

7. Finally, use `rotate()` to rotate the camera during timesteps 65 to 149, completing one full circle while pointing toward the origin (and not getting closer or further from the origin). Figure 3 shows a snapshot taken during the rotation process.

Hints. At each step, you should test a small piece of code and convince yourself that it works correctly before moving on to the next step. This is crucial!

What to submit. Submit your file to gradescope, which will test your matrix creation routines and tell you if they are correct.

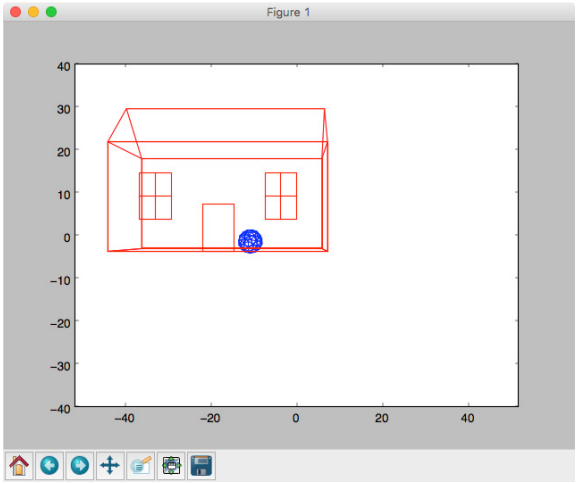

Figure 2: Final Ball Position.

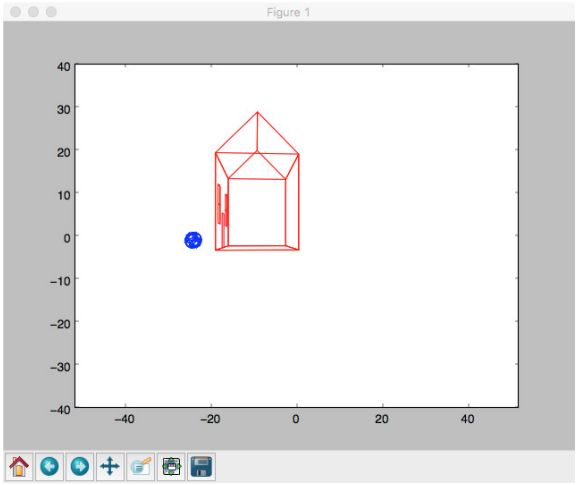

Figure 3: During Camera Rotation.

Notes. There are other wireframe objects provided. You can experiment with them, and interpret what happens when more complex objects are used. There is a big ball and a more detailed house.

I got the wireframes from the following sources:

- basicHouse.obj: I designed myself (and it shows :)
- snub_icosidodecahedron.wrl: `http://www.georgehart.com/virtual-polyhedra/archimedean-index.html`
- largeBall.obj, largeHouse.obj: `http://www.turbosquid.com/` (search for free files)