Visual Computing

Lab Exercises

Samuel Silva

November 2, 2022

Panda 3D: First Steps

5

In this hands-on we will be performing a first exploration of the Panda 3D engine. It is expected that, by completing it you should have an understanding of:

- Basic structure of a Panda3D script
- Be familiar with the concept of scene graph
- Add models to a scene
- Add light sources to a scene
- Operate the position and orientation of the camera
- Use keyboard keys manipulate the scene/camera
- Understand the concept of periodic tasks (e.g., for automatic movement)

5.1 Panda 3D Window

Open the file `CV_Aula05_HandsOn_Ex1.py` and execute it. If you still have not installed Panda3D, you can do it by typing the following line in a terminal:

```
pip3 install panda3d
```

You should see an empty window with a light grey background and the framerate on the top-right corner. You can tweak some of the properties of the window if file `config.prc` in the `/config` folder.

Whenever you have some doubt about a Panda3D instruction or want to explore its features, head to the online manual. There, you can find a thorough reference to all Panda3D's features and many examples: Panda3D Manual.

5.2 Adding a model to the scene

The first thing we will do is add a model to the scene.

While this is not the best of practises, for today's hands on we will be adding most of the code to the `__init__` method in class `MyGame`. Nevertheless, you can create additional methods to organize it your own way, if you like.

Add the following lines that load model "`skull.obj`" from disk.

```
        self.skullModel = self.loader.loadModel(self.mydir + "/
    skull.obj")
        self.skullModel.setPos(0, 0, -5)
```

If you run the script, nothing is shown. This is because for Panda3D to render a model, it **must** be added to its scene graph as a node of `render`. Therefore, add the following line:

```
        self.skullModel.reparentTo(self.render)
```

Now, if you run it, you still cannot see anything. This is because the camera is initially positioned at the origin (and so is the model). Therefore, the camera cannot see anything. We need to move it back, a little. For this, add the following lines:

```
self.disableMouse()
self.camera.setPos(0, -30, 0)
```

The first line just disables the mouse for camera control. In fact, before adding this line, you could drag the mouse over the window and use the `shift`, `ctrl` and `alt` keys to move the camera around with the mouse, but it is not very intuitive. And without disabling it, you cannot move the camera programatically.

The second line is straightforward to understand and moves the camera to -30 in the YY' axis. You should note that the coordinate system for Panda3D is different from the one for OpenGL. In here, the XX' goes from left to right, but the ZZ' axis goes from bottom to the top and the YY' axis goes from near to far. So, it is negative getting away from the screen. Therefore, the camera moves back 30 units.

By running the script, you should now see something near the top of the window, but nothing that looks like a skull. The problem is that the model is big. We need to scale it down:

```
self.skullModel.setScale(0.35,0.35,0.35)
```

You should now see a skull at the middle of the window.

5.3 Light My Bones

Panda3D starts with a standard diffuse lighting scheme, but we will now personalize it with some light sources of our own. First, we will add a point light:

```
self.plight1 = PointLight('plight')
self.plight1.setColor((1, 1, 1, 1))
self.plnp1 = self.render.attachNewNode(self.plight1)
self.plnp1.setPos(0, -10, 10)
self.render.setLight(self.plnp1)
```

This code starts by declaring a point light source, setting its color (white) and, then, attaches it as a node to the scene graph and sets its position. Finally, for it to take effect on the scene, we need to set it as a light for the renderer.

5.4 Moving the Camera

If we want to see all sides of the skull we could do one of two things: (1) move the camera around the skull; or (2) rotate the skull. We will go for the first option.

Add the following method to class MyGame:

```
def spinCameraTask(self, task):
    self.cameraRadius = 30.0
    angleDegrees = task.time * 20.0
    angleRadians = angleDegrees * (math.pi / 180.0)

    self.camera.setPos(self.cameraRadius * math.sin(
angleRadians), -self.cameraRadius * math.cos(angleRadians),
0)

    self.camera.lookAt(0.0, 0.0, 0.0)
    return Task.cont
```

This method has a lot into it. Analyse the code and try to understand what is being computed and performed. The `task.time` just provides the amount of time elapsed since starting the script. If you comment the line `self.camera.lookAt(0.0, 0.0, 0.0)`, what happens? Why is it needed? Remember, what parameters define a camera?

If you run the script, nothing happens, yet. We need to register this new method to be run every frame. For that, and inside `__init__`, add the following line:

```
self.taskMgr.add(self.spinCameraTask, "SpinCameraTask")
```

Now, you should see the camera moving around the skull. The effect is very nice, but there are some spots of the model that appear dark. Why? Can you explain it?

Do you remember what type of light is used to compensate for this effect and provide an effect similar to the natural lighting of regions not directly illuminated by a light source? Did you think of... ambient light? Let us add one to the scene. The procedure is very similar to adding a point light, but now the light type is `AmbientLight` and, since its is omnipresent, it has no position to be set:

```
self.alight = AmbientLight('alight')
self.alight.setColor((0.2, 0.2, 0.2, 1))
self.alnp = self.render.attachNewNode(self.alight)
self.render.setLight(self.alnp)
```

5.5 Adding a Panda

Now, we will add a second model to the scene from those that come bundled with Panda3D. To this effect, add the following lines to your code:

```
self.pandaModel = self.loader.loadModel("models/panda")
self.pandaModel.setPos(0, 0, 0)
self.pandaModel.setP(90)
self.pandaModel.setScale(.6,.6,.6)
self.pandaModel.reparentTo(self.render)
```

These lines are mostly similar to the ones used to load the skull model. Note the additional `self.pandaModel.setP(90)`. It sets the pitch to 90 degrees, which makes the panda lay down on its belly instead of standing up.

Do you know what Headi
Roll stand for? They are the
in Panda3D for rotations.
mean?

Well the panda is stuck in the skull. We will now make another modification to further expand our knowledge about scene graphs. We made

the parent a child node of the renderer, but we will make it a child of the skullmodel, now. For this to happen, change the reparenting line of the panda to:

```
self.pandaModel.reparentTo(self.skullModel)
```

Now, you will see that two things happened: the panda changed position and it changed size. This is due to the fact that making it a child of the skull model implies that its positioning and scale is, now defined in reference to its parent. So, any transformation we apply to the skull is also applied to all children.

Just change the position of the panda to `(0.0, 0.0, 30.0)` to make it appear on top of the skull.

Now, experiment with changing the position of the skull in the scene, for instance, to `(5, 0, -5)`. What happened to the panda?

5.6 Making the Panda Fly

Now, let us add some movement to the panda and make it go around in circles, flying over the skull. For that:

- Define a method named `spinPandaTask`, similar to the one used for moving the camera and place there the computations for the trajectory of the panda. Use a radius of 10.0 and look into `spinCameraTask` for inspiration
- Register `spindPandaTask` to be executed every frame as we did for `spinCameraTask`
- Note that you can change the heading `self.pandaModel.setH(some angle))` to make the panda always fly with its head to the front

5.7 Some Basic Interaction

Now, to demonstrate some basic interaction, we will test if any key is down, at each frame. To do this, add the following method and register it to be run every frame:

```
def moveTask(self, task):
    isDown = base.mouseWatcherNode.isButtonDown

    if isDown(KeyboardButton.asciiKey("+")):
        self.cameraRadius += 1
    if isDown(KeyboardButton.asciiKey("-")):
        self.cameraRadius -= 1
    return Task.cont
```

This method basically checks if the '+' or '-' keys are down and changes the `cameraRadius` accordingly. What changes do you need to perform, in the already existing code (in `def spinCameraTask(self, task):`), so that these changes affect the camera position?

5.8 Chaotic Experimentation

- Add a blue point light that illuminated the back of the skull model
- Add a red ambient light but attach it only to the panda model (**not** to render)