Hierarchical Modeling

Brian Curless
CSE 557
Autumn 2017

Symbols and instances

Most graphics APIs support a few geometric primitives:
- spheres
- cubes
- cylinders

These symbols are instantiated using an instance transformation.

Q: What is the matrix for the instance transformation above?

TRS

3D Example: A robot arm

Let’s build a robot arm out of a cylinder and two cuboids, with the following 3 degrees of freedom:
- Base rotates about its vertical axis by $\theta$
- Upper arm rotates in its $xy$-plane by $\phi$
- Lower arm rotates in its $xy$-plane by $\psi$

(Note that the angles are set to zero in the figures on the right; i.e., the parts are shown in their “default” positions.)

Suppose we have transformations $R_x(\phi), R_y(\psi), T(0, h, 0)$.

Q: What matrix do we use to transform the base?

Q: What matrix product for the upper arm?
3D Example: A robot arm

An alternative interpretation is that we are taking the original coordinate frames...

...and translating and rotating them into place:

Robot arm implementation

The robot arm can be displayed by keeping a global matrix and computing it at each step:

Matrix $M$, $M_{model}$, $M_{view}$:

```c
main()
{
    ...
    $M_{view} = \text{compute_view_transform}()$;
    robot_arm();
    ...
}
```

```c
robot_arm()
{
    $\text{Transform for base}$
    $M_{model}:$
    $M = M_{view} * M_{model};$
    $M_{model}:$
    $M_{model} = R_y(\theta) * T(0, h_1, 0) * R_z(\phi);$
    $M = M_{view} * M_{model};$
    $upper_{arm}:$
    $M_{model} = R_y(\theta) * R_z(\phi) * T(0, h_2, 0) * R_z(\psi);$
    $M = M_{view} * M_{model};$
    $lower_{arm}();$
    }
```
Hierarchical modeling

Hierarchical models can be composed of instances using trees or DAGs:

- edges contain geometric transformations
- nodes contain geometry (and possibly drawing attributes)

We will use trees for hierarchical models.

How might we draw the tree for the robot arm?

A complex example: human figure

Q: What's the most sensible way to traverse this tree?

Depth-first

Using canonical primitives

Consider building the robot arm again, but this time the building blocks are canonical primitives like a unit cylinder and a unit cube.

What additional transformations are needed?
What does the hierarchy look like now?

Animation

The above examples are called articulated models:
- rigid parts
- connected by joints

They can be animated by specifying the joint angles (or other display parameters) as functions of time.
Key-frame animation

The most common method for character animation in production is **key-frame animation**.
- Each joint specified at various **key frames** (not necessarily the same as other joints)
- System does interpolation or **in-betweening**

Doing this well requires:
- A way of smoothly interpolating key frames: **spline**
- A good interactive system
- A lot of skill on the part of the animator

![Key-frame animation diagram]

---

Scene graphs

The idea of hierarchical modeling can be extended to an entire scene, encompassing:
- many different objects
- lights
- camera position

This is called a **scene tree** or **scene graph**.

![Scene graph diagram]

---

Summary

Here's what you should take home from this lecture:
- All the **boldfaced terms**.
- How primitives can be instanced and composed to create hierarchical models using geometric transforms.
- How the notion of a model tree or DAG can be extended to entire scenes.
- How OpenGL transformations can be used in hierarchical modeling.
- How keyframe animation works.