Hierarchical Models

- Objects are defined in their own coordinate system
- They are then "transformed" and placed in their proper place in the scene
- Transformations:
  - Translation (moving from one spot to another)
  - Rotating (around any of the axis)
  - Scaling (making the object larger or smaller in any direction)

The Graphics Pipeline

- Model Transformation
- Viewing Transformation
- 3D Clipping
- Projection
- Window to Viewport Mapping

\[
\begin{bmatrix}
  x_o \\
  y_o \\
  z_o \\
  1
\end{bmatrix} =
\begin{bmatrix}
  \text{Transformation} & \text{Camera} & \text{World to Projection}
\end{bmatrix}
\begin{bmatrix}
  x_p \\
  y_p \\
  z_p \\
  1
\end{bmatrix}
\]

Matrix Stacks

- In OpenGL, "current" matrices are really matrix stacks
- Before adding a localized transformation, push the current transformation matrix on the stack, thus saving a copy of it.
  - `glPushMatrix()`
- When you're done, pop the stack, thus restoring the previous transformation matrix.
  - `glPopMatrix()`

Hierarchical Models

- Example: a two-armed robot

```
Robot Base
  └── Lower Arm
  └── Lower Arm
      └── Upper Arm
      └── Upper Arm
```

- Suppose we need to model an object comprised of components all placed relative to other components
  - Example:
    - A two-armed robot
      - `/usr/local/pub/ncs/graphics/OpenGL/ExamplesLab3/robot`
  - Even in a local coordinate system, it may be tedious to keep track of the transformations of the various components

Hierarchical Models

- We know how to transform of each component with respect to another component
  - Use the matrix stack in order to calculate the local coordinates of each component
Hierarchical Models - Robot

Define camera orientation
Push Matrix // Saves view matrix
Define transformation for robot as a whole
Push Matrix // Saves whole object position/orientation
Matrix
Define transformations for positioning robot base in scene
Draw robot base
Push Matrix // Saves robot base
Define transformations for left arm w/r/t the center of the robot
Draw arm
Pop Matrix // Restores robot base matrix
Push Matrix // Saves robot base
Define Transformations of right Arm with respect to robot base
Draw arm
Pop Matrix // Restores robotic base
Pop Matrix // Restores position/orientation matrix
Pop Matrix // Restores pre-robot (i.e., orig view) matrix

Hierarchical Models - drawArm

Define transformations for robot lower arm w/r/t the center of the robot
Push matrix
Define transformations needed to generate lower arm from glutCube
Draw lower arm
Pop matrix
Define transformations of upper arm w/r/t the lower arm
Push matrix
Define transformations needed to generate upper arm from glutCube
Draw upper arm
Pop matrix

Hierarchical Models - drawBase

Push Matrix
Define transformations to generate base from gluCylinder
Draw Base
Pop Matrix

Hierarchical Models

• Advantages
  - More intuitive to specify
  - Allows for individual control of each component
  - Reuse

On to Animation

• Motion and changes over time
• Create individual images (frames) to be played back at a constant rate
  - 24 frames/sec = film
  - 30 frames/sec = video and TV
  - As fast as your machine can handle = interactive animation.
• Each image reflects changes in the scene that occurs within each time frame
• Examples:
  - Early Computer Animation: Wally B
  - Artistic Breakthrough: Luxo Jr.
  - State of the Art: Bingo

Animation

• What can change in an animation:
  - Shape of objects
  - Position of objects
  - Transformation on objects and object components
  - Lighting
  - Camera
  - Colors
  - Textures
  - Etc.

  “There’s no particular mystery in animation...
  It’s really very simple and like anything that is simple,
  It’s about the hardest thing in the world to do”

  -- Bill Tytla, Walt Disney Studios, 1937
Why Animation is Difficult

- Consider a 5 minute feature
  - 5 min x 60 sec x 24 frames = 7200 frames
  - For each frame of 7200, you must determine the “state” of your scene
  - You also have to generate an image for each frame
- Now consider a 90-minute feature film
  - 90 min x 60 sec x 24 frames = 129,600 frames!

Animation Pipeline

- Storyboard
  - Series of rough cartoon-like drawings
  - Depicts staging, camera angles, pacing, division into scenes
  - One storyboard for each major piece of action

- Modeling
  - Creating the objects in your scene
    - Geometry
    - Shading
    - Texturing

- Motion control
  - Defining the motion of objects over time
  - Defining other changes in your scene over time
    - Camera motion
    - Lighting
    - Texturing, etc.

- Rendering
  - Create an image for each frame in your animation
    - Local Illumination Models
    - Radiosity
    - Ray Tracing
    - Other rendering methods
Animation Pipeline

- Post-production
  - Editing
  - Add sound, effects, narration, title
  - Assemble onto a given medium (film, video, etc.)

Motion Control

- A problem unique to animation
- How to define position and orientation for each object in scene in frame of the animation?
- Issues:
  - How much control?
  - How many degrees of freedom?

Storyboard

Modeling ➔ Motion Control ➔ Rendering ➔ Post-Production

Motion Control Techniques

- Keyframing / Tweening / Interpolation
  - Define location and orientation at a number of given keyframes
  - System interpolates to get motion for frames between the keys
- Motion is defined by animator
- Method most often used in "traditional" animation

Motion Control Techniques

- Procedural methods
- Motion is generated by some procedure or algorithm
  - Dynamic (physically based)
    - Heuristics
    - Behavioral motion

Dynamic (Physically-Based) Motion Control

- Objects have physical attributes (weight, mass, etc)
- Motion is determined by laws of physics
  - Rigid body motion
  - Shape deformation
  - Cloth modeling
- Need a physics simulator
- Animator has little control

Heuristic Motion Control

- Like dynamic motion control, except that the physics is made up
  - What looks good vs. what is physically accurate
- Take liberties with physics
- Examples:
  - Particle Systems
  - Smoke, Fog, Fire
Behavioral Motion Control

- Motion defined by some set of heuristics
- Good for modeling motion of large groups of objects
  - Ex: flocking, herding, plant growth
  - Ex: Lord of the Rings battles
- Motion of one object is based on relationship to other objects in the group
- Once again: animator has little control

Articulated Character Motion

- Defining motion for articulated figures (hierarchical models) is particularly challenging
  - Must define transformations for all joints in the hierarchy
- Forward kinematics
  - Animator defines transformation for all joints
  - End-effectors (fingers, feet, toes) end up where they may
  - Difficult to do even for the best animators
- Inverse kinematics
  - Animator defines position of end effectors
  - System determines “appropriate” transformations at each joint

Levels of control

- Low-level
  - Animator defines all joint transformations
- Procedural
  - Animator defines “motor programs”
    - Swing leg, wiggle finger
- Functional
  - Animator defines “skill”
    - Walking, grasping
- Character
  - Animator defines a “task”
    - Go to the fridge and get me a beer

Articulated Character Motion

- Levels of control
  - Low-level
    - Animator defines all joint transformations
  - Procedural
    - Animator defines “motor programs”
      - Swing leg, wiggle finger
  - Functional
    - Animator defines “skill”
      - Walking, grasping
  - Character
    - Animator defines a “task”
      - Go to the fridge and get me a beer

Real-time Animation

- Frames are generated on the fly
- Frame rate will depend upon
  - Speed of your machine
  - Complexity of your scene
  - Complexity of your rendering

Animation in OpenGL

- OpenGL (GLUT) event loop

```
main() → display()

redisplay event → no events?

idle()
```

- Perform scene updates in `idle()` callback
- OpenGL also supports `double_buffering`
  - Two frame buffers
    - Last frame shown in window
    - New frame being drawn on second buffer
    - Swap buffers
### Areas of Animation Research

- **Procedural Animation**
  - Fog, smoke, water, etc.
- **Facial Animation**
  - Facial models, lip sync
- **Sound and Animation**
  - Automated soundtracks / speech
- **Real-time animation**
  - Gaming
- **Shape Morphing**
- **Motion Capture Systems**