lecture 9

Object hierarchies

- call trees and GL_MODELVIEW stack
- fractals
- L systems
Last lecture:

- hierarchy of bounding volumes of objects and scenes
- spatial partition represented as a tree (BSP trees, octrees)

Today:

- how to model and draw object hierarchies?
Example: human body

The object has a hierarchy of parts.
ASIDE: In an object oriented design, you might define a class hierarchy:

We will **not** discuss OOD approaches today, however.
How to draw a person? (call tree)

drawBody( ){
  drawUpperBody()
  drawLowerBody()
}

drawUpperBody(){
  drawHead()
  drawTorso()
  drawRightArm()
  drawLeftArm()
}

drawLowerBody(){
  drawHips()
  drawRightLeg()
  drawLeftLeg()
  etc
}
Consider a tree whose nodes are the object parts and whose edges represent coordinate transformations between parts.

In OpenGL, you use the GL_MODELVIEW matrix stack to traverse the tree.
Of course, neither of these two types of trees exists (as data structures).

The call tree does not necessarily correspond to the coordinate system tree.

OpenGL programs may involve both of these trees. Let's sketch some examples.
drawUpperBody() {
    glPushMatrix();
    drawTorso(); // Head and arm coordinate systems
    // are relative to torso.
    glPushMatrix();
    glTranslate();
    glRotate(); // Allow head rotation.
    drawHead();
    glPopMatrix();
    
    glPushMatrix();
    glTranslate(); // Allow shoulder joint motion.
    glRotate();
    drawLeftArm();
    drawLeftArm(); // right arm too
    glPopMatrix();
    
    glPopMatrix();
}
drawLowerBody()
{
    glPushMatrix()
    drawHips()
        // Leg coordinate systems
        // are relative to hips.
    glPushMatrix()
    glTranslate
    glRotate()
        // Allow hip joint rotation.
    drawLeftLeg()
    glPopMatrix()
    glPopMatrix()
    glPopMatrix()
    glPushMatrix()
    glTranslate
    glRotate
    glRotate
    drawRightLeg()
    glPopMatrix()
    glPopMatrix()
    glPopMatrix()
}
drawLeftArm( ... ){
    glPushMatrix()
    glRotate( )
    drawLeftUpperArm()
    glTranslate( )
    glRotate( )
    drawLeftForeArm()
    glTranslate()
    glRotate()
    drawHand() // etc. draw palm, fingers
    glPopMatrix()
}
Notation (used later in lecture)

\[ D_{\text{upperbody}} = [ \, D_{\text{torso}} \begin{bmatrix} T & R \end{bmatrix} D_{\text{head}} \begin{bmatrix} T & R \end{bmatrix} D_{\text{leftarm}} \begin{bmatrix} T & R \end{bmatrix} \ldots \, ] \]

\[ D_{\text{leftarm}} = \begin{bmatrix} R \end{bmatrix} D_{\text{leftupperarm}} \begin{bmatrix} T & R \end{bmatrix} D_{\text{leftforearm}} \begin{bmatrix} T & R \end{bmatrix} D_{\text{hand}} \]

\[ D_{\text{hand}} = \ldots \]

- \( D \) is draw (including lines, triangles, etc)
- [ is glPushMatrix() and ] is glPopMatrix()
- \( T \) and \( R \) are glTranslate() and glRotate()
lecture 9

Object hierarchies

- call trees and GL_MODELVIEW stack
- fractals
- L systems
Many natural objects have complicated geometry.
http://paulbourke.net/fractals/googleearth/
"How long is the coastline of Britain? ....

Statistical Self-Similarity and Fractional Dimension", B. Mandelbrot, Science, 1967
Fractals entered computer graphics in 1980's....

See video link for Mandelbrot set
http://kottke.org/10/10/benoit-mandelbrot-rip
Here is what computer graphics can easily do now,...

But let's go back to the beginning,... the first fractals.
Koch Curve (1903)

Start with a line segment.

Replace the line segment with 4 line segments, each of length 1/3 the original.

Repeat (recursively)....
As \( n \) goes to infinity, the Koch Curve ...

- remains continuous
- has infinite length
- has no tangent \textit{anywhere}
- is \textbf{self-similar} (a key property of fractal geometry)
def koch(i):  //  0 < i < infinity
    if i == 0
        drawline()
    else if i > 0
        glPushMatrix()
            glScalef(1/3, 1/3, 1/3 )
            koch(i-1)
            glTranslatef(1.0, 0.0, 0.0)
            glRotatef(60, 0.0, 0.0, 1.0)
            koch(i-1)
            glTranslatef(1.0, 0.0, 0.0)
            glRotatef(-60, 0.0, 0.0, 1.0)
            koch(i-1)
            glTranslatef(1.0, 0.0, 0.0)
            glRotatef(60, 0.0, 0.0, 1.0)
            koch(i-1)
        glPopMatrix()

In this example, the call tree corresponds to coordinate system hierarchy tree.

The branching factor is 4.

The draw commands occur at the leaves which are all at the same depth.
Sierpinski Carpet

Start with square, partition into 9 squares of width 1/3, and delete the central square. Repeat recursively.

\[
\begin{array}{cccc}
\text{area} & \frac{8}{9} & \left(\frac{8}{9}\right)^2 & \left(\frac{8}{9}\right)^3 & \left(\frac{8}{9}\right)^4 \\
\end{array}
\]

Area goes to 0 as n goes to infinity.
Sierpinski Cube

Start with cube, partition into 27 subcubes of width 1/3, and delete the 7 cubes containing the central xyz axes. Repeat recursively.

\[ \frac{20}{27} \quad \ldots \quad \left(\frac{20}{27}\right)^4 \]
Fractal dimension

Calculus deals with objects that have integer dimension.

\[
\text{dim( line segment )} = 1
\]

\[
\text{dim( square )} = 2
\]

\[
\text{dim( cube )} = 3
\]

Fractals have a non-integer dimension.
Fractal dimension

Definition is based on "self-similarity across scale".

Assume our set (object) is in $\mathbb{R}^n$ and has the following property:

We can scale it by some $S > 1$ in each of the $n$ dimensions, such that the scaled object consists of $C$ translated and/or rotated copies of the original one.

Then the set has fractal dimension $D$ where:

$$C = S^D$$

or equivalently

$$D = \frac{\log(C)}{\log(S)}$$
<table>
<thead>
<tr>
<th>Shape</th>
<th>S</th>
<th>C</th>
<th>D = log(C) / log(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>line segment</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>square</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>cube</td>
<td>2</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>Koch curve</td>
<td>3</td>
<td>4</td>
<td>~1.26</td>
</tr>
<tr>
<td>Sierpinski carpet</td>
<td>3</td>
<td>8</td>
<td>~1.89</td>
</tr>
<tr>
<td>Sierpinski cube</td>
<td>3</td>
<td>20</td>
<td>~2.73</td>
</tr>
</tbody>
</table>
Fractals in nature are typically random.

To generate models of random fractals, we use random variables.

e.g. normal distribution ("Bell curve")

standard deviation $\sigma$

mean $\mu$
Example: Random Walks ("drunken sailor")

Step size at time t has a normal distribution.
Random Fractals

How could we compute fractal "random walks"?

These are random walk curves that continue to appear rough as we "zoom in".
Midpoint displacement method [Fournier et al 1982]

First, initialize the curve to 0 everywhere. Then, choose endpoints of the curve (somehow).

How to interpolate?
Consider midpoint. Set midpoint value to be average of values at endpoints plus a random displacement.

Repeat recursively.
def midpointDisplacement\( a, \text{std, roughness} \) \{
    \text{// a is an array}
    \text{// roughness is a scale factor between 0 and 1.}
    \text{// roughness = 1/ \sqrt{2} is called Brownian motion}

    \text{newStd = roughness* std}

    \text{size = len( a)}
    \text{if (size <= 2)}
        \text{return a}
    \text{else\{ \hspace{1cm} // *Python syntax.}
        \text{middle = size / 2}
        \text{a[middle] = (a[0] + a[size-1]) /2 + rand.normal( newStd )}
        \text{a[0:middle+1] = midpointDisplacement(}
            \hspace{2cm} \text{a[ 0:middle+1 ], newStd, roughness } \}
        \text{a[middle:size] = midpointDisplacement(}
            \hspace{4cm} \text{a[ middle:size ], newStd, roughness } \}
        \text{return a}
    \}\}

\text{// Subtle note: midpointDisplacement() only\}
\text{changes the value of the midpoint. Thus, a[middle] does not get\}
\text{changed in the second and third recursive calls.}
random walk

midpoint displacement algorithm
What should be the probability distribution of random displacements at each level of the recursion?

The math is very advanced and subtle (and not our concern in COMP 557).

The algorithm is simple and flexible, for example, just scale the standard deviation of the displacement.

[Think of drunken caffeinated sailor taken faster steps, each of smaller size.]
def midpointDisplacement( a, std, roughness) {

  // roughness is a scale factor between 0 and 1.
  // roughness = 1/ sqrt(2) is called Brownian motion

  newStd = std * roughness

  size = len( array)
  if (size <= 2)
    return array
  else{

    middle = size / 2
    a[middle] = (a[0] + a[size-1] ) /2 + rand.normal( newStd )
    a[0:middle+1] = midpointDisplacement( a[ 0:middle+1 ], newStd, roughness )
    a[middle:size] = midpointDisplacement( a[ middle:size ], newStd, roughness )

    return a
  }
}
Examples

roughness

0.6

0.7

0.8
Straightforward extension to 2D doesn't work so well. But more clever methods work very well.
Anytime you see something like this in a Hollywood movie, .... it isn't real. It was made with such fractal-based algorithms.
lecture 9

Object hierarchies

- call trees and GL_MODELVIEW stack
- fractals
- L systems
Recall notation from earlier ...

\[
D_{\text{upperbody}} = [ \ D_{\text{torso}} \ [ T \ R \ D_{\text{head}} ] \ [ T \ R \ D_{\text{leftarm}} ] \ ... \ ]
\]

drawUpperBody()
{
    glPushMatrix()
    drawTorso()
    glPushMatrix()
    glTranslate
    glRotate()
    drawHead()
    glPopMatrix()
    glPushMatrix()
    glTranslate
    glRotate()
    drawLeftArm()
    glPopMatrix
    :
    glPopMatrix()
}
def koch(i):
    if i == 0
        drawline()
    else if i > 0
        glPushMatrix()
            glScalef(1.0/3,1.0/3,1.0/3)
            koch(i-1)
            glTranslatef(1.0, 0.0, 0.0)
            glRotatef(60, 0.0, 0.0, 1.0)
            koch(i-1)
            glTranslatef(1.0, 0.0, 0.0)
            glRotatef(-60, 0.0, 0.0, 1.0)
            glRotatef(-60, 0.0, 0.0, 1.0)
            koch(i-1)
            glTranslatef(1.0, 0.0, 0.0)
            glRotatef(60, 0.0, 0.0, 1.0)
            koch(i-1)
        glPopMatrix()
We replace symbols with strings (of symbols).

Most of you are familiar with this concept from formal grammars and language theory e.g. compilers.
L systems

Introduced by theoretical biologist Astrid Lindemayer (1960's) to describe structure and growth of biological systems, especially plants.

Later adopted by computer graphics, for drawing objects by recursive substitutions (like fractals, but finite).
Example

L \rightarrow \left[ S \ D \ T \left[ R \ L \right] \ D \ T \left[ R' \ L \right] \ R \ L \right]

D is 'draw a line'
T, R, S, are translate (by 1), rotate (30), scale (by 1/3)
We have an "alphabet" of symbols. Think of OpenGL library, plus drawX().

We start with an "axiom" string or starting symbol.

We then replace symbols, using productions
Think of a function calling another function (or recursion).
"Parametric L systems"

- the symbols can have parameters
e.g. \( R(30) \) rotate CCW by 30 deg.

- can keep track of the level of recursion,
specify a base case e.g.

\[
L(n) \rightarrow [ S \ D \ T \ [ R \ L(n-1) ] \ D \ T \ [ R' \ L(n-1) ] \ R \ L(n-1) ]
\]

\[
L(0) \rightarrow D \ T
\]

As computer scientists, you should find nothing conceptually new here.
"Probabilistic L systems"

Different productions can occur with different probabilities.

Again, nothing new here. Essentially this means....

def draw() {
    ....
    if (rand() > p0) // rand() returns value in [0,1]
        draw1()
    else
        draw2()
    ....
}
"Open L systems"

Use global variables (instead of random numbers) to determine which productions get called or what the parameters passed are.

e.g. Only grow a leaf if it receives direct sunlight rather than being in shadow (we'll talk about lighting in a few weeks).

Model soil based on terrain shape and allow plants to compete for light and water...
http://www.graphics.stanford.edu/papers/ecosys/
(paper from 1998)
Announcements

next few weeks (revised schedule)

9. object hierarchies
   *OpenGL transformation stack, fractals, L-systems*

10. smooth curves and surfaces
    *cubics: Hermite curves & Catmull-Rom splines, bicubic surfaces*

11. meshes
    *level of detail, edge collapse*

**Rendering: light, material, texture, transparency**

12. lighting, material
    *ambient/point/spot, diffuse/mirror/glossy*

13. review + exercises
14. midterm exam (Thursday Feb. 19)