Assignments

Correction: Assignments

• project 1
  • out today, due 5pm sharp Fri Jan 29
  • projects will go out before we've covered all the material so you can think about it before diving in
  • template code gives you program shell and build tools
  • now out: separate packages for Linux, Mac, Windows
  • see http://www.ugrad.cs.ubc.ca/~cs314/VJan2010/Assign
    - p1_template_linux.tar
    - p1_template_mac.tar.gz
    - p1_template_win.zip
• written homework 1
  • out today, due 5pm sharp Fri Jan 29
  • theoretical side of material

Display Lists

• precompile/cache block of OpenGL code for reuse
• usually more efficient than immediate mode
• exact optimizations depend on driver
  • good for multiple instances of same object
    • but cannot change contents, not parametrically
  • good for static objects redrawn often
    • display lists persist across multiple frames
  • interactive graphics: objects redrawn every frame from new viewpoint from moving camera
  • can be nested hierarchically
  • snowman example
http://www.lighthouse3d.com/opengl/displaylists

Real Iguanas

Review: Arbitrary Rotation

- arbitrary rotation: change of basis
- given two orthonormal coordinate systems $\{X, Y, Z\}$ and $\{A, B, C\}$
- it's location in the $\{X, Y, Z\}$ coordinate system is $(x_0, y_0, z_0, 1)$ ...
- transformation from one to the other is matrix $R$ whose columns are \( a, b, c, d \)

$$
R(X) = \begin{pmatrix} a_{01} & a_{02} & a_{03} & a_4 \\ b_{01} & b_{02} & b_{03} & b_4 \\ c_{01} & c_{02} & c_{03} & c_4 \\ 0 & 0 & 0 & 1 \end{pmatrix} = (x_0, y_0, z_0, 1) = A
$$

Transformation Hierarchy Example 3

Transformation Hierarchy Example 4

Hierarchical Modelling

- advantages
  - define object once, instantiate multiple copies
  - transformation parameters often good control knobs
  - maintain structural constraints if well-designed
- limitations
  - expressively: not always the best controls
  - can't do closed kinematic chains
    - keep hand on hip
  - can't do other constraints
    - collision detection
    - self-intersection
    - walk through walls

Display Lists

Making Display Lists

One Snowman

```c
void drawSnowMan(int) {
  // Draw Eyes
  glPushMatrix();
  glTranslatef(0.0f, 0.05f, 0.0f);
  glScalef(0.05f, 0.05f, 0.05f);
  glutSolidSphere(0.25f, 16.0f, 16.0f);
  glPopMatrix();

  // Draw Body
  glPushMatrix();
  glTranslatef(0.0f, 0.05f, 1.0f);
  glutSolidSphere(0.2f, 16.0f, 16.0f);
  glPopMatrix();

  // Draw Legs
  glPushMatrix();
  glTranslatef(0.0f, -0.5f, 1.0f);
  glRotatef(90.0f, 1.0f, 0.0f, 0.0f);
  glScalef(0.5f, 0.5f, 0.5f);
  glutSolidSphere(0.1f, 16.0f, 16.0f);
  glPopMatrix();

  // Draw Gloves
  glPushMatrix();
  glTranslatef(0.0f, 0.0f, 0.0f);
  glutSolidCube(0.1f, 0.1f, 0.1f);
  glPopMatrix();

  // Draw Trees
  glPushMatrix();
  glTranslatef(0.0f, 0.0f, 0.0f);
  glutSolidCube(0.1f, 0.1f, 0.1f);
  glPopMatrix();
}
```

Instantiate Many Snowmen

```c
// Draw 36 Snowmen
for (int i = -3; i < 3; i++)
  for (int j = -3; j < 3; j++)
    glPushMatrix();
    glTranslatef(10.0f, 8.0f, 0.0f * 10.0f);
    drawSnowMan();
    glPopMatrix();
```

36K polygons, 55 FPS
Transforming Normals

- nonuniform scaling does not work
- x-y=0 plane
- line x=y
- normal: [1, -1, 0]
- direction of line x=-y
- (ignore normalization for now)

these all maintain direction

normal vector by

do this work for everything? no!

so if points transformed by matrix

transform a plane

Computing Normals

- normal
direction specifying orientation of polygon
- w=0 means direction with homogeneous coords
- vs. w=1 for points/vectors of object vertices
- used for lighting
- must be normalized to unit length
- can compute if not supplied with object

Finding Correct Normal Transform

• transform a plane

P

N

N

given M, what should Q be?

stay perpendicular

Thus the normal to any surface can be transformed by the inverse transpose of the modelling transformation