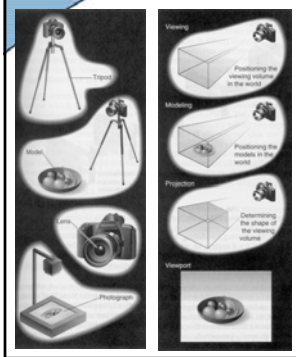
OpenGL programming:

Viewing & Trackball Assignment

Overview

- The camera analogy
- Stages of vertex transformation
- The Matrix Stacks
- A simple example: Drawing a cube ...
- Viewing & modeling transformations
- Assignment: Trackball

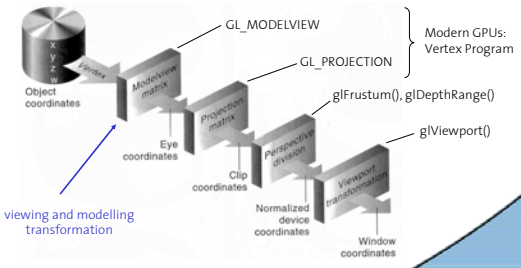
The Camera Analogy



Transformation process in computer graphics:

- Set up tripod and point camera at the scene
  Viewing transformation
- Arrange the scene
  Modeling transformation
- Choose a lens and/or adjust zoom
  Projection transformation
- Size of the final photograph
  Viewport transformation

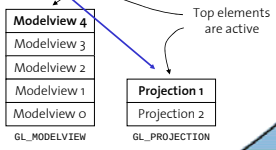
Stages of Vertex Transformation



The Matrix Stacks

order in code does not matter

- Modelview and projection defined by two matrices
- Matrices are top elements of two *matrix stacks*
- Stack selection:
  `glMatrixMode()`
- `glPushMatrix()`
  `glPopMatrix()`
- Stacks allow to undo modifications
- Functions to modify top matrix:
  `glLoadIdentity(), glLoadMatrix(), glMultMatrix(), glTranslate(), glRotate(), glScale()`



A Simple Example: Drawing A Cube ...

```
void display(void) {
  glClear (GL_COLOR_BUFFER_BIT);
  glLoadIdentity ();
  gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
  glScalef (1.0, 2.0, 1.0);
  glutWireCube (1.0);
  glFlush ();
}

void reshape (int w, int h)  {
  glViewport (0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode (GL_PROJECTION);
  glLoadIdentity ();
  glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
  glMatrixMode (GL_MODELVIEW);
}

int main(int argc, char** argv)  {
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutCreateWindow (argv[0]);
  glutDisplayFunc(display);
  glutReshapeFunc(reshape);
  glutMainLoop();
}
```

… Viewing Transformation

- Positioning and *aiming a camera*
- Specification, e.g., with `gluLookAt(…)`
- Arguments:
  - camera position (“center of projection”)
  - camera aimed to (“view reference point”, “look at”)
  - up-direction (fixes rotation around optical axis)
- Modifies current matrix
- ! Default camera position in OpenGL !

… Viewing Transformation

Default matrix mode: GL_MODELVIEW

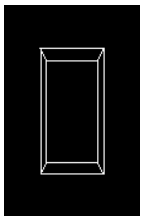
```
void display(void) {
  glClear (GL_COLOR_BUFFER_BIT);
  glLoadIdentity ();
  gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
  glScalef (1.0, 2.0, 1.0);
  glutWireCube (1.0);
  glFlush ();
}

void reshape (int w, int h)  {
  glViewport (0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode (GL_PROJECTION);
  glLoadIdentity ();
  glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
  glMatrixMode (GL_MODELVIEW);
}

int main(int argc, char** argv)  {
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutCreateWindow (argv[0]);
  glutDisplayFunc(display);
  glutReshapeFunc(reshape);
  glutMainLoop();
}
```

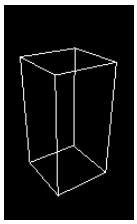

glLookAt example(1)

```
gluLookAt (0.0, 0.0, 5.0,      //camera position
           0.0, 0.0, 0.0,      //camera aimed to
           0.0, 1.0, 0.0);     //up-direction
```



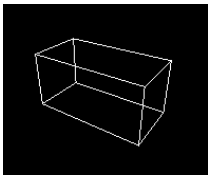
glLookAt example(2)

```
gluLookAt (3.0, 2.0, 2.0,      //camera position
           0.0, 0.0, 0.0,      //camera aimed to
           0.0, 1.0, 0.0);     //up-direction
```



glLookAt example(3)

```
gluLookAt (3.0, 2.0, 2.0,      //camera position
           0.0, 0.0, 0.0,      //camera aimed to
           0.0, 0.0, 1.0);     //up-direction
```



… Modeling Transformation

- *Placing a model* (position, orientation)
- In our example: using `glScale(…)`
- Accumulated in modelview matrix (*right multiply*)

- ! Duality of viewing & modeling transform !
  e.g. gluLookAt(…) ↔ glTranslatef(…)

… Modeling Transformation

```
void display(void) {
  glClear (GL_COLOR_BUFFER_BIT);
  glLoadIdentity ();
  gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
  glScalef (1.0, 2.0, 1.0);
  glutWireCube (1.0);
  glFlush ();
}

void reshape (int w, int h)  {
  glViewport (0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode (GL_PROJECTION);
  glLoadIdentity ();
  glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
  glMatrixMode (GL_MODELVIEW);
}

int main(int argc, char** argv)  {
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutCreateWindow (argv[0]);
  glutDisplayFunc(display);
  glutReshapeFunc(reshape);
  glutMainLoop();
}
```


… Projection Transformation

- Like *choosing a lens* for a photo camera
- Projection transformation with `glFrustum(…)`
- Determination of field of view/viewing volume
- Determination of type of projection

- Here: included in the reshape callback

… Projection Transformation

```
void display(void) {
  glClear (GL_COLOR_BUFFER_BIT);
  glLoadIdentity ();
  gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
  glScalef (1.0, 2.0, 1.0);
  glutWireCube (1.0);
  glFlush ();
}

void reshape (int w, int h)  {
  glViewport (0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode (GL_PROJECTION);
  glLoadIdentity ();
  glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
  glMatrixMode (GL_MODELVIEW);
}

int main(int argc, char** argv)  {
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutCreateWindow (argv[0]);
  glutDisplayFunc(display);
  glutReshapeFunc(reshape);
  glutMainLoop();
}
```


… Viewport Transformation

- Scene mapping to screen coordinates
  - Projection transformation: how …
  - Viewport transformation: screen area …
- Viewport transformation with `glViewport(…)`

- Here: included in the reshape callback

… Viewport Transformation

```
void display(void) {
  glClear (GL_COLOR_BUFFER_BIT);
  glLoadIdentity ();
  gluLookAt (0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
  glScalef (1.0, 2.0, 1.0);
  glutWireCube (1.0);
  glFlush ();
}

void reshape (int w, int h)  {
  glViewport (0, 0, (GLsizei) w, (GLsizei) h);
  glMatrixMode (GL_PROJECTION);
  glLoadIdentity ();
  glFrustum (-1.0, 1.0, -1.0, 1.0, 1.5, 20.0);
  glMatrixMode (GL_MODELVIEW);
}

int main(int argc, char** argv)  {
  glutInit(&argc, argv);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutCreateWindow (argv[0]);
  glutDisplayFunc(display);
  glutReshapeFunc(reshape);
  glutMainLoop();
}
```

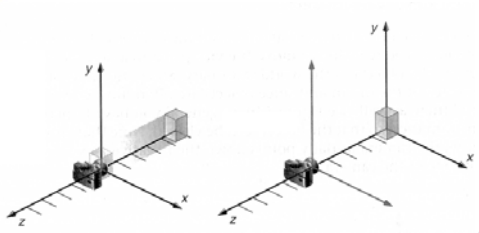

Viewing & Modeling Transformations

- Transformation matrices have to be defined *before* rendering geometry
- Choose correct matrix stack !
  e.g. `glMatrixMode(GL_MODELVIEW)`
- Order of transformations
- Thinking about transformations
  - world coordinates: fixed coordinate system
  - object coordinates: local coordinate system

Viewing & Modeling Transformations

- Moving the camera = moving all objects in opposite direction
- Viewing transformation commands prior to any modeling transformation
- Why? Order of transformations ...
- Many different roads lead to rome...

Viewing & Modeling Transformations



Leads to the same modelview matrix...

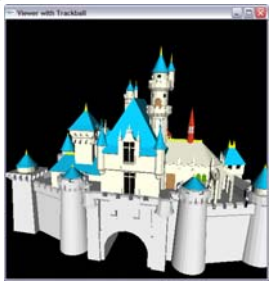
Further Readings ...

- The red book, chapter 3:

  OpenGL Programming Guide
  3rd Edition
  The Official Guide to Learning OpenGL

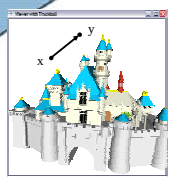
- On–line version: see link on course web page
  http://graphics.ethz.ch/

Trackball Assignment



- Interactive Viewer for OBJ triangle objects
- Intuitive interface to rotate object:
  - Trackball interface
  - Mouse motion is translated into rotation
- Requires knowledge of transformations and quaternions

Trackball Assignment



- Mouse motion from x to y in screen coordinates
- Projection onto an underlying sphere leads to x' and y'
- They define angle and axis of rotation
- Framework application allows dragging and spinning