COS426 Computer Graphics

Fisher Yu

Mar 27, 2013

Some Slides from Aleksey Boyko

Topic

OpenGL

Topics

- Getting started
- Initialization
- Drawing
- Transformations
  - Cameras
  - Animation
- Input
  - Keyboard
  - Mouse
- Textures
- Lights
- Programmable pipeline elements (shaders)

Topics

- Getting started
- Initialization
- Drawing
- Transformations
  - Cameras
  - Animation
- Input
  - Keyboard
  - Mouse
- Textures
- Lights
- Programmable pipeline elements (shaders)

Design

- OpenGL uses a big context to manage everything
- GLUT: A simple windowing API for OpenGL.
  - Callback driven event processing
  - Warning: Global variables ahead
- Resources are identified by integer ID

Input

- Keyboard
- Mouse

Keyboard

- Normal keys:
  - Anything that has ASCII code
- Register a callback with GLUT
  - void glutKeyboardFunc(void (*func) (unsigned char key, int x, int y));
  - The x and y callback parameters indicate the mouse location in window relative coordinates when the key was pressed.
- Implement the callback

Keyboard

- **Register a callback in main():**

```
   glutKeyboardFunc(processNormalKeys);
```

- **Implement the callback**

```
bool animationOn=false;
void processNormalKeys(unsigned char key, int x, int y) {
        //escape key
        switch(key)
        {
        case 27:
          exit(0);
        case 'b':
          glClearColor(0.,0.,0.,1.); break;
        case 'w':
          glClearColor(1.,1.,1.,1.); break;
        case 'a':
          animationOn = !animationOn; break;
        }
}
```

Keyboard

- Special keys:
  - F1-F12
  - Arrow keys
  - Page up/down, Home, End, Insert
- Register a callback with GLUT
  - `void glutSpecialFunc(void (*func) (int key, int x, int y));`
- Implement the callback

Keyboard

- **Register a callback in main():**

```
glutSpecialFunc(processSpecialKeys);
```

- **Implement the callback**

```
void processSpecialKeys(int key, int x, int y)
{
    switch(key) {
    case GLUT_KEY_F1 :
      ...
    case GLUT_KEY_UP:
      …
    }
}
```

Keyboard - example

Control rotation around the center point with the arrow keys

- Some additional variables and includes:
  - #include <math.h>
  - const float Pi=4*atan(1.);
  - static float phi=0, theta=Pi/2,;
  - static float phiStep=Pi/18, thetaStep=Pi/18;
  - static float camDist=5.0;

Keyaboard - example

Control rotation around the center point with the arrow keys

- Register a call back
  - glutSpecialFunc(processSpecialKeys);

Keyboard - example

Control rotation around the center point with the arrow keys

- Implement the callback

```
void processSpecialKeys(int key, int x, int y)
{
      switch(key) {
      case GLUT_KEY_UP:
         theta-=thetaStep; break;
      case GLUT_KEY_DOWN:
         theta+=thetaStep;break;
      case GLUT_KEY_LEFT:
         phi-=phiStep;break;
      case GLUT_KEY_RIGHT:
         phi+=phiStep;break;
      }
}
```

Keyboard - example

Control rotation around the center point with the arrow keys

- Update the renderScene():

```
void renderScene(void)
{
	…
	float cosTheta=cos(theta),sinTheta=sin(theta);
	gluLookAt(camDist*sin(phi)*sinTheta,
			camDist*cosTheta,
			camDist*cos(phi)*sinTheta,
			0.0,0.0,0.0,
			0.0f,sinTheta,0.0f);
	…
}
```

Keyboard – Ctrl,Alt,Shift

int glutGetModifiers(void);

returns a value that can be compared to bitmasks:

- GLUT_ACTIVE_SHIFT
- GLUT_ACTIVE_CTRL
- GLUT_ACTIVE_ALT

e.g.:

int modifier = glutGetModifiers();

- if (modifier ==GLUT_ACTIVE_CTRL) …;
- if (modifier ==(GLUT_ACTIVE_CTRL|GLUT_ACTIVE_ALT)) …;
- If (modifier & GLUT_ACTIVE_CTRL) …;

Keyboard

- Other useful keyboard functions
  - int glutSetKeyRepeat(int repeatMode);
  - int glutIgnoreKeyRepeat(int repeatMode);

- Other callbacks
  - void glutKeyboardUpFunc(void (*func)(unsigned char key,int x,int y));
  - void glutSpecialUpFunc(void (*func)(int key,int x, int y));

Mouse

- What can you do with a mouse?

Mouse

- What can you do with a mouse?
  - Click
    - Register a callback with

```
void glutMouseFunc(void (*func)(int button, int state, int x, int y));
```

    - Button:
      - GLUT_LEFT_BUTTON
      - GLUT_MIDDLE_BUTTON
      - GLUT_RIGHT_BUTTON
    - State:
      - GLUT_DOWN
      - GLUT_UP

Mouse

- What can you do with a mouse?
  - Click
  - Move
    - Register a callback with

```
void glutPassiveMotionFunc(void (*func) (int x, int y));
```

Mouse

- What can you do with a mouse?
  - Click
  - Move
  - Drag
    - Register a callback with

```
void glutMotionFunc(void (*func) (int x,int y));
```

Mouse

- What can you do with a mouse?
  - Click
  - Move
  - Drag
  - Leave/enter a window
    - Register a callback with

```
void glutEntryFunc(void (*func)(int state));
```

    - State:
      - GLUT_LEFT
      - GLUT_ENTERED

Mouse - example

Control rotation around the center point with the mouse

- Some additional variables:
  - static int width,height;
  - static bool moveCamera=false;
  - static int oldX,oldY;

Mouse - example

Control rotation around the center point with the mouse

- Register callbacks
  - glutMouseFunc(processMouse);
  - glutMotionFunc(processMouseActiveMotion);

Mouse - example

Control rotation around the center point with the mouse

- Click and drop implementation

```
void processMouse(int button, int state, int x, int y){
      if(button==GLUT_LEFT_BUTTON){
        if(state==GLUT_DOWN){
                oldX = x;
                oldY = y;
                moveCamera = true;
        }
        else //state==GLUT_UP
                moveCamera = false;
      }
}
```

Mouse - example

Control rotation around the center point with the mouse

- Drag implementation

```
void processMouseActiveMotion(int x, int y) {
      if(moveCamera)
      {
        phi += (2*Pi*(oldX-x))/width;
        theta += (2*Pi*(oldY-y))/height;

        oldX=x;
        oldY=y;
      }
}
```

Mouse - example

Control rotation around the center point with the mouse

- To keep width and height up to date:

```
void changeSize(int w, int h) {
    …

    //remember the window size
    width=(w>0?w:1);
    height=(h>0?h:1);
}
```

.

Textures

- Supports (depending on version):
  - 1D
  - 2D
    - Power of 2
    - Or not
  - 3D
  - …

Textures

- Enable/disable texturing

```
glEnable( GL_TEXTURE_2D );
glDisable( GL_TEXTURE_2D );
```

Name Texture

- Name
  - GLuint texture;
- Get a name
  - glGenTextures(N, *textures);
- Check a name
  - glIsTexture(texture)
- Delete a texture
  - glDeleteTextures(N,*textures);

Bind a texture

- Tell OpenGL that you want to use this texture
- void **glBindTexture**(
  - GLenum *target*, = GL_TEXTURE_2D
  - GLuint *texture*)

Texture Environment

- void **glTexEnv(f/x)[v]**(
  - GLenum *target*, = GL_TEXTURE_ENV
  - GLenum *pname*,
    - GL_TEXTURE_ENV_MODE
    - GL_TEXTURE_ENV_COLOR
  - GL(float/fixed) [*]*param)*
    - GL_MODULATE
    - GL_DECAL
    - GL_BLEND
    - GL_REPLACE
    - …

Textures parameters

- void **glTexParameter(f/x)**(
  - GLenum *target*, =GL_TEXTURE_2D
  - GLenum *pname*,
    - GL_TEXTURE_MIN_FILTER
    - GL_TEXTURE_MAG_FILTER
    - GL_TEXTURE_WRAP_S
    - GL_TEXTURE_WRAP_T
  - GLfloat *param*)
    - GL_NEAREST
    - GL_LINEAR
    - GL_NEAREST_MIPMAP_NEAREST
    - GL_LINEAR_MIPMAP_NEAREST
    - GL_NEAREST_MIPMAP_LINEAR
    - ..

Create texture

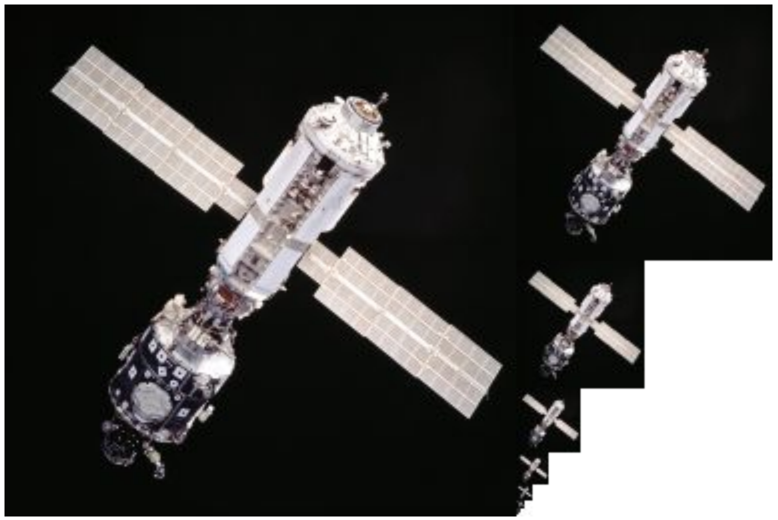
- Have an array ready - *data
- glTexImage2D()
- gluBuild2DMipmaps()
  - Mip: Latin *multum in parvo,* “many things in a small place”

glTexImage2D()

- void **glTexImage2D**(
  - GLenum *target*, = GL_TEXTURE_2D
  - GLint *level*, = 0, …
  - GLint *internalformat*, = GL_RGB,….
  - GLsizei *width*,
  - GLsizei *height*,
  - GLint *border*, = 0
  - GLenum *format*, = *internalformat*
  - GLenum *type*, = GL_UNSIGNED_BYTE ,…
  - const GLvoid * *pixels*) = data

gluBuild2DMipmaps()

- GLint **gluBuild2DMipmaps**(
  - GLenum *target*, = **GL_TEXTURE_2D**
  - GLint *internalFormat*,
  - GLsizei *width*,
  - GLsizei *height*,
  - GLenum *format*,
  - GLenum *type*,
  - const void **data*)



Assign texture coordinates

- For each glVertex that is part of textured polygon call glTexCoord().
- E.g. glTexCoord2f(
  - GLdouble *s*,
  - GLdouble *t*)

Textures

Code example

Lights

- Supports lights:
  - GL_LIGHT0
  - …
  - GL_LIGHT(GL_MAX_LIGHTS - 1)

Lights

- Enable
  - glEnable(GL_LIGHTING)
  - glEnable(GL_LIGHTX)
- Diable
  - glDisable(GL_LIGHTING)
  - glDisable(GL_LIGHTX)

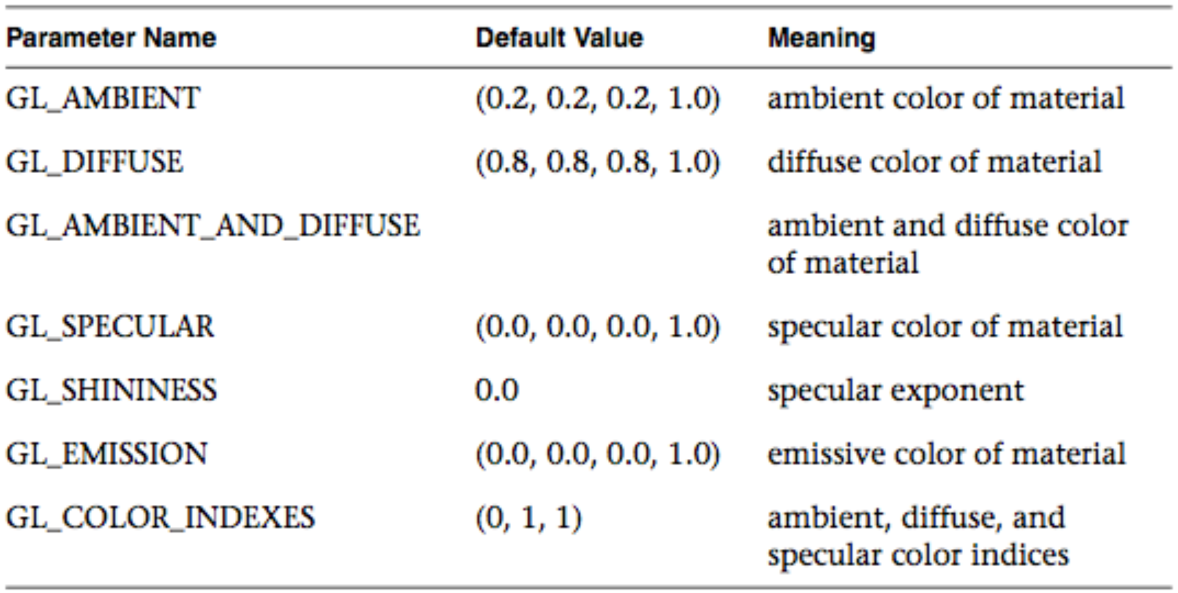
glLight

- void **glLight(f/i)[v]**(
  - GLenum *light*,
  - GLenum *pname*,
    - GL_SPOT_EXPONENT
    - GL_SPOT_CUTOFF
    - GL_CONSTANT_ATTENUATION
    - GL_LINEAR_ATTENUATION
    - GL_QUADRATIC_ATTENUATION

      if v
    - GL_AMBIENT
    - GL_DIFFUSE
    - GL_SPECULAR
    - GL_POSITION
    - GL_SPOT_CUTOFF
    - GL_SPOT_DIRECTION
    - GL_SPOT_EXPONENT
    - GL_CONSTANT_ATTENUATION
    - GL_LINEAR_ATTENUATION
    - GL_QUADRATIC_ATTENUATION
  - GL(float/int) [*]*param*);

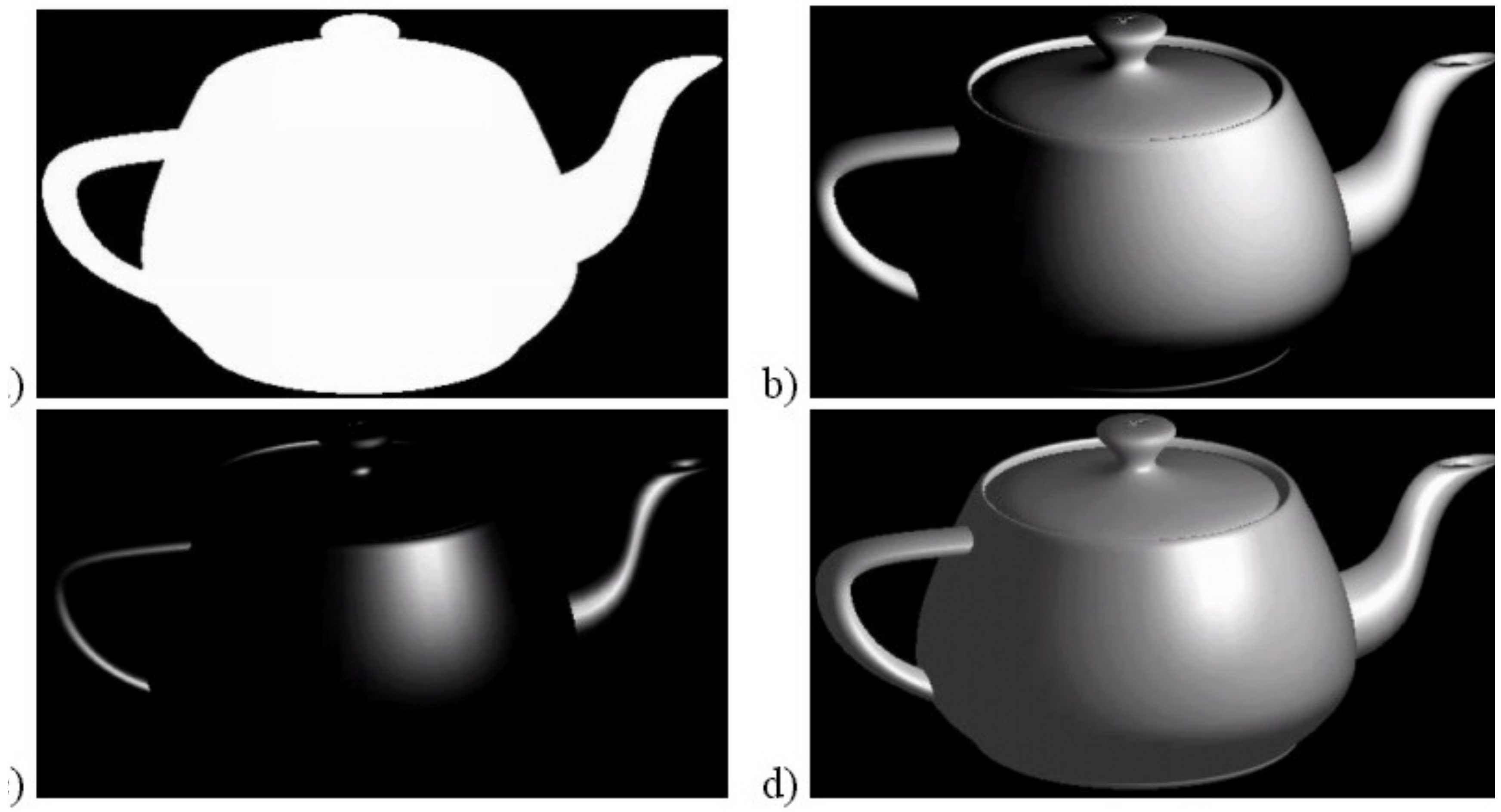
glMaterial

- void **glMaterial{if}(GLenum** *face,* GLenum *pname, TYPE param);*
  void **glMaterial{if}v(GLenum** *face,* GLenum *pname,* const *TYPE* *param);



Parameter Name	Default Value	Meaning
GL_AMBIENT	(0.2, 0.2, 0.2, 1.0)	ambient color of material
GL_DIFFUSE	(0.8, 0.8, 0.8, 1.0)	diffuse color of material
GL_AMBIENT_AND_DIFFUSE		ambient and diffuse color of material
GL_SPECULAR	(0.0, 0.0, 0.0, 1.0)	specular color of material
GL_SHININESS	0.0	specular exponent
GL_EMISSION	(0.0, 0.0, 0.0, 1.0)	emissive color of material
GL_COLOR_INDEXES	(0, 1, 1)	ambient, diffuse, and specular color indices

Light And Material



http://www.naturewizard.com/tutorial0107.html

Lights example in code

```
//light source position
float lpos[4] = {0.,0.,1.,1.};
bool lightsOn=false;
```

- In renderScene()
  - glLightfv(GL_LIGHT0, GL_POSITION, lpos);
- In processNormalKeys(..)
  - Add enabling/diabling code

Lights

Code example

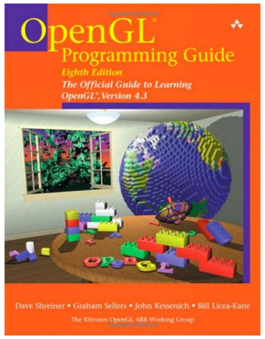
Topics

- Getting started
- Initialization
- Drawing
- Transformations
  - Cameras
  - Animation
- Input
  - Keyboard
  - Mouse
- Textures
- Lights
- Programmable pipeline elements (shaders)

References

More Detail:

THE RED BOOK


OpenGL
Programming Guide
Eighth Edition
The Official Guide to Learning OpenGL, Version 4.3
Dave Shreiner • Graham Sellers • John Kessenich • Bill Licea-Kane
The Khronos OpenGL ARB Working Group

More tutorials (partly used in the presentation):

http://www.lighthouse3d.com/opengl/glut

http://nehe.gamedev.net/

http://www.videotutorialsrock.com/

OpenGL quick reference:

http://www.khronos.org/files/opengl4-quick-reference-card.pdf