Introduction

OpenGL syntax

Modelin

Procedural modeling OpenGL primitives GLUT primitives

Conclusion



Introduction to OpenGL

Ensimag 3D Graphics, 2015

Planning

Introduction

OpenGL pipeline OpenGL syntax

Modeling

Procedural modeling OpenGI

OpenGL primitives

GLUT primitives

Conclusio

Available at

https://intranet.ensimag.fr/KIOSK/Matieres/4MMG3D/index.html

Plan

Introduction

- Graphics pipeline
 OpenGL pipeline
- Modeling Procedural modeling
- modeling
 OpenGL
 primitives
 GLUT primitives

Conclusion

- 1 Introduction
- 2 Pipeline

Graphics pipeline OpenGL pipeline OpenGL syntax

3 Modeling

Procedural modeling OpenGL primitives GLUT primitives

4 Conclusion

Introduction

Dinalina

OpenGL syntax

Modeli

modeling
OpenGL
primitives

GLUT primitives

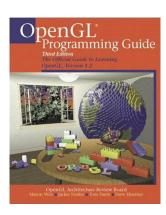
Conclusion

D. Shreiner, M. Woo, J. Neider, T. Davis OpenGL Programming Guide

aka the red book

http://opengl-redbook.com

References



Plan

- 1 Introduction
- 2 Pipeline Graphics pipeline OpenGL pipeline
- Modeling Procedural modeling OpenGL primitives GLUT primitives
- 4 Conclusion

Introduction

OpenGL pipeline OpenGL syntax

Modeli

modeling
OpenGL
primitives
GLUT primitives

Conclusio

What is it?

- API (Application Programming Interface) for graphics hardware
- Non-dependant on the architecture or programming language
- Developped in 1989 (GL) by Silicon Graphics, extended to other architectures in 1993 (OpenGL)
- About 250 commands

Introduction

Introductio

OpenGL syntax

Modelin

Procedural modeling OpenGL primitives GLUT primitives

Conclusio

Things it can **NOT** do

- Can NOT create nor manage a viewer
- Can NOT manage complex objects: only 3 types of geometric primitives (points, lines, polygons)

Introduction

Graphics pipelin

OpenGL syntax

modeling
OpenGL
primitives
GLUT primitives

Conclusion

Things it can **NOT** do

- Can NOT create nor manage a viewer
- Can NOT manage complex objects: only 3 types of geometric primitives (points, lines, polygons)

- > Additional libraries needed :
 - GLU: openGL Utility library: more complex 3D models
 - GLUT: openGL Utility Toolkit: viewer
 - QGLViewer: Qt library handling OpenGL
 - ...

Plan

- Introduction
- Pipeline
 Graphics pipeline
 OpenGL pipeline
 OpenGL syntax
- Modeling Procedural modeling OpenGL primitives GLUT primitives
- 4 Conclusion

Plan

- 1 Introduction
- 2 Pipeline Graphics pipeline OpenGL pipeline OpenGL syntax
- Modeling Procedural modeling OpenGL primitives GLUT primitives
- 4 Conclusion

Introduction

Introduction

Graphics pipeline

OpenGL pipeiii OpenGL syntax

Modelin

Procedural modeling OpenGL

primitives
GLUT primitives

Conclusion

Graphics Pipeline



① Create 3D models (modeling)

Introduction

Graphics pipeline

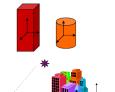
OpenGL pipeline OpenGL syntax

Modeling Procedur

OpenGL primitives GLUT primitives

Conclusion

Graphics Pipeline



- ① Create 3D models (modeling)
- 2 Build the scene from instances of models placed in a world frame (modeling transformation)

Introduction

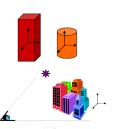
Graphics pipeline OpenGL pipeline

OpenGL pipelin

Procedural modeling

OpenGL primitives GLUT primitives

Conclusion



frustrum

Graphics Pipeline

- 1 Create 3D models (modeling)
- 2 Build the scene from instances of models placed in a world frame (modeling transformation)
- 3 Convert to camera frame (culling, frustrum)

Introduction

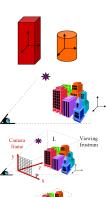
Pipeline

Graphics pipeline OpenGL pipeline OpenGL syntax

Modeli

openGL primitives GLUT primitives

Conclusion



P: Z=0

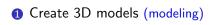
Graphics Pipeline

- 1 Create 3D models (modeling)
- ② Build the scene from instances of models placed in a world frame (modeling transformation)
- 3 Convert to camera frame (culling, frustrum)
- 4 Convert to screen frame (projection)

Graphics pipeline

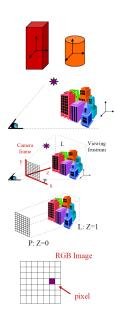
GLUT primitives

Graphics Pipeline



- Build the scene from instances of models placed in a world frame (modeling transformation)
- 3 Convert to camera frame (culling, frustrum)
- 4 Convert to screen frame (projection)

5 Compute image (rasterization)



Plan

- Introduction
- 2 Pipeline Graphics pipeline OpenGL pipeline OpenGL syntax
- Modeling
 Procedural modeling
 OpenGL primitives
 GLUT primitives
- 4 Conclusion

Introduction

Pineline

OpenGL contax

Modelii

Procedural modeling OpenGL primitives GLUT primitives

Conclusio

OpenGL Pipeline

Create OpenGL context

Introduction

Pineline

OpenGL syntax

Modeli

Procedural modeling OpenGL primitives GLUT primitives

Conclusio

OpenGL Pipeline

Introduction

Pipeline

OpenGL syntax

Modeli

Procedural modeling OpenGL primitives GLUT primitives

Conclusion

OpenGL Pipeline

Create OpenGL context Loop :

Manage mouse/keyboard events

Introduction

Pipeline

OpenGL pipeline

Modeli

Procedural modeling OpenGL primitives GLUT primitives

Conclusion

OpenGL Pipeline

- Manage mouse/keyboard events
- ② Display

Introduction

Pipeline

OpenGL pipeline
OpenGL syntax

Modeli

modeling
OpenGL
primitives
GLUT primitives

Conclusion

OpenGL Pipeline

- Manage mouse/keyboard events
- ② Display
 - Clear screen

Introduction

Craphics pinel

OpenGL pipeline OpenGL syntax

Modeli

modeling
OpenGL
primitives
GLUT primitives

Conclusion

OpenGL Pipeline

- Manage mouse/keyboard events
- 2 Display
 - Clear screen
 - 2 Viewpoint

Introduction

Pipeline

OpenGL pipeline OpenGL syntax

Modeli

modeling
OpenGL
primitives
GLUT primitives

Conclusion

OpenGL Pipeline

- Manage mouse/keyboard events
- ② Display
 - Clear screen
 - 2 Viewpoint
 - 3 For each object :

Introduction

Compliance

OpenGL pipeline OpenGL syntax

Modeli

modeling
OpenGL
primitives
GLUT primitives

Conclusion

OpenGL Pipeline

- Manage mouse/keyboard events
- ② Display
 - Clear screen
 - 2 Viewpoint
 - 3 For each object :
 - 1 Place object

Introduction

Graphics pipeli

OpenGL pipeline OpenGL syntax

Modeli

modeling
OpenGL
primitives
GLUT primitives

Conclusion

OpenGL Pipeline

- Manage mouse/keyboard events
- ② Display
 - 1 Clear screen
 - 2 Viewpoint
 - 3 For each object :
 - Place object
 - 2 Draw

Introduction

Pipeline

OpenGL pipeline

Modelii

Procedural modeling OpenGL primitives GLUT primitives

Conclusion

It's a state machine!

State machine = each parameter retains its value and is used with that value until being explicitely changed

Introduction

Graphics pipelin

OpenGL pipeline OpenGL syntax

Procedural modeling

OpenGL primitives GLUT primitives

Conclusion

It's a state machine!

State machine = each parameter retains its value and is used with that value until being explicitely changed

Parameters can be:

• modes: shading mode, matrix manipulated ...

Introduction

Graphics pipeline
OpenGL pipeline

Modeling
Procedural
modeling
OpenGL
primitives
GLUT primitives

Conclusion

It's a state machine!

State machine = each parameter retains its value and is used with that value until being explicitely changed

Parameters can be:

- modes : shading mode, matrix manipulated . . .
- **booleans**: lights on/off, blend colors, ...

Introduction

Graphics pipeline
OpenGL pipeline

Modeling
Procedural
modeling
OpenGL
primitives
GLUT primitives

Conclusio

It's a state machine!

State machine = each parameter retains its value and is used with that value until being explicitely changed

Parameters can be :

- modes : shading mode, matrix manipulated . . .
- **booleans**: lights on/off, blend colors, ...
- scalar values : colors, viewpoint, ...

Plan

- 1 Introduction
- Pipeline Graphics pipeline OpenGL pipeline OpenGL syntax
- Modeling Procedural modeling OpenGL primitives GLUT primitives
- 4 Conclusion

OpenGL Syntax - 1/2

Introduction

Pipeline

OpenGL pipeline
OpenGL syntax

Modelin

modeling
OpenGL
primitives
GLUT primitives

Conclusion

Reminder:

• modes : gl[MODE]Mode(GL_VALUE)

OpenGL Syntax - 1/2

Introduction

Pipeline Graphics pipeline

OpenGL pipelin

Procedural modeling

modeling
OpenGL
primitives
GLUT primitives

Conclusion

Reminder:

• modes : gl[MODE]Mode(GL_VALUE)

• booleans : glEnable(GL_VALUE)

Introduction

Graphics pipeline OpenGL pipeline OpenGL syntax

Modeling Procedural

modeling
OpenGL
primitives
GLUT primitives

Conclusion

OpenGL Syntax - 1/2

Reminder:

- modes : gl[MODE]Mode(GL_VALUE)
- booleans : glEnable(GL_VALUE)
 - \Rightarrow OpenGL constants start with **GL**_

Introduction

Graphics pipeline OpenGL pipeline OpenGL syntax

Modeling Procedural modeling OpenGL

OpenGL primitives GLUT primitives

Conclusion

OpenGL Syntax - 1/2

Reminder:

- modes : gl[MODE]Mode(GL_VALUE)
- booleans : glEnable(GL_VALUE)
 - \Rightarrow OpenGL constants start with \textbf{GL}_{-}
- scalar values :

• gl : OpenGL command . . .

Introduction

Graphics pipeline OpenGL pipeline OpenGL syntax

Modeling
Procedural
modeling
OpenGL
primitives

GLUT primitives
Conclusion

OpenGL Syntax - 1/2

Reminder:

- modes : gl[MODE]Mode(GL_VALUE)
- booleans : glEnable(GL_VALUE)
 - ⇒ OpenGL constants start with GL_
- scalar values :

- gl : OpenGL command ...
- 3 : ...that has 3 arguments ...

Introduction

Graphics pipeline OpenGL pipeline OpenGL syntax

Modeling
Procedural
modeling
OpenGL
primitives
GLUT primitives

Conclusion

OpenGL Syntax - 1/2

Reminder:

- modes : gl[MODE]Mode(GL_VALUE)
- booleans : glEnable(GL_VALUE)
 - ⇒ OpenGL constants start with GL_
- scalar values :

- gl : OpenGL command . . .
- 3 : ...that has 3 arguments ...
- **f**:...of type float.

Introduction

Graphics pipeline OpenGL pipeline OpenGL syntax

Modeling
Procedural
modeling
OpenGL

GLUT primitives
Conclusion

OpenGL Syntax - 1/2

Reminder:

- modes : gl[MODE]Mode(GL_VALUE)
- booleans : glEnable(GL_VALUE)
 - \Rightarrow OpenGL constants start with GL_
- scalar values :

```
glColor3f(1.0,1.0,1.0);
```

- gl : OpenGL command ...
- 3 : ...that has 3 arguments ...
- **f**: ... of type float.

```
glColor3fv(color_array);
```

⇒ The argument is a **v**ector (or array) of 3 floats (GLfloat color_array[] = { 1.0,0.0,0.0 } ;)

Introduction

Pipeline

OpenGL syntax

Model

Procedural modeling OpenGL primitives GLUT primitives

Conclusion

OpenGL Syntax - 2/2

OpenGL suffixes and types

b	integer (8 bits)
S	integer (16 bits)
i	integer (32 bits)
f	real (32 bits)
d	real (64 bits)
ub	unsigned integer (8 bits)
us	unsigned integer (16 bits)
ul	unsigned integer (32 bits)

signed char	GLbyte
short	GLshort
int ou long	GLint
float	GLfloat
double	GLdouble
unsigned char	GLubyte
unsigned long	GLushort
unsigned int ou long	GLuint

I manual di casta m

Introduction

Graphics pipelin

OpenGL syntax

Modeling

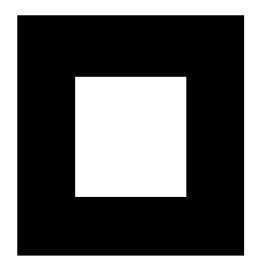
Procedural modeling

primitives

GLUT primitives

Conclusion

Basic example - the Square



Basic example - the Code

Display:

OpenGL pipeline

OpenGL syntax

Procedural

primitives GLUT primitives

Basic example - the Code

Display:

void display () {

OpenGL pipeline OpenGL syntax

Procedural primitives GLUT primitives

Basic example - the Code

OpenGL pipeline

Display: void display () {



Clear screen

OpenGL syntax

Procedural modeling primitives GLUT primitives

ntroduction

Introduction

Graphics pipeline OpenGL pipeline OpenGL syntax

Modelin

Procedural modeling OpenGL primitives GLUT primitives

Basic example - the Code

```
Display :
void display () {
```

Clear screen

glClear(GL_COLOR_BUFFER_BIT);

OpenGL syntax

Procedural primitives **GLUT** primitives

Basic example - the Code

```
Display:
```

void display () {

- Clear screen
 - glClear(GL_COLOR_BUFFER_BIT);
- 2 Viewpoint

OpenGL syntax

GLUT primitives

Basic example - the Code

```
Display:
```

```
Clear screen
      glClear(GL_COLOR_BUFFER_BIT);
```

void display () {

2 Viewpoint glMatrixMode(GL_PROJECTION); glOrtho(0.0,1.0,0.0,1.0,-1.0,1.0);

Introduction

Introduction

Graphics pipelin OpenGL pipelin OpenGL syntax

Modeli

Procedural modeling OpenGL primitives GLUT primitives

Conclusion

```
Display:
```

- void display () $\{$
 - 1 Clear screen
 glClear(GL_COLOR_BUFFER_BIT);
 - Viewpoint
 glMatrixMode(GL_PROJECTION);
 glOrtho(0.0,1.0,0.0,1.0,-1.0,1.0);
 - 3 For each object :

Basic example - the Code

Introduction

Pipeline Graphics pipeline

OpenGL pipelin OpenGL syntax

Procedural modeling

OpenGL primitives GLUT primitives

Conclusion

```
Display :
void display () {
```

- 1 Clear screen
 glClear(GL_COLOR_BUFFER_BIT);
 - Viewpoint
 glMatrixMode(GL_PROJECTION);
 glOrtho(0.0,1.0,0.0,1.0,-1.0,1.0);
- 3 For each object :
 - 1 Place object

Displ

Introduction

Graphics pipeline OpenGL pipeline OpenGL syntax

Modelii

Procedural modeling OpenGL primitives GLUT primitives

Conclusion

Basic example - the Code

```
Display:

void display () {

1 Clear screen
    glClear(GL_COLOR_BUFFER_BIT);

2 Viewpoint
    glMatrixMode(GL_PROJECTION);
    glOrtho(0.0,1.0,0.0,1.0,-1.0,1.0);

3 For each object:
    1 Place object
```

glMatrixMode(GL_MODELVIEW);

Introduction

Graphics pipeline OpenGL pipeline OpenGL syntax

Modeling

modeling
OpenGL
primitives
GLUT primitives

Conclusion

```
Display:
void display () {

1  Clear screen
    glClear(GL_COLOR_BUFFER_BIT);

2  Viewpoint
    glMatrixMode(GL_PROJECTION);
    glOrtho(0.0,1.0,0.0,1.0,-1.0,1.0);

3  For each object:
    1  Place object
    glMatrixMode(GL_MODELVIEW);
    2  Modify state machine
```

Introduction

Pipeline
Graphics pipeline
OpenGL pipeline

OpenGL syntax

Procedural modeling OpenGL primitives GLUT primitives

Conclusion

```
Display:
void display () {
  Clear screen
        glClear(GL_COLOR_BUFFER_BIT);
  2 Viewpoint
        glMatrixMode(GL_PROJECTION);
        glOrtho(0.0,1.0,0.0,1.0,-1.0,1.0);
  3 For each object :
       1 Place object
         glMatrixMode(GL_MODELVIEW);
       2 Modify state machine
         glColor3f(1.0,1.0,1.0);
```

Introduction

Graphics pipeline
OpenGL pipeline
OpenGL syntax

Modeling

Procedural modeling OpenGL primitives GLUT primitives

Conclusion

```
Display:
void display () {
  Clear screen
        glClear(GL_COLOR_BUFFER_BIT);
  2 Viewpoint
        glMatrixMode(GL_PROJECTION);
        glOrtho(0.0,1.0,0.0,1.0,-1.0,1.0);
  3 For each object :
       1 Place object
         glMatrixMode(GL_MODELVIEW);
       2 Modify state machine
         glColor3f(1.0,1.0,1.0);
       3 Draw
```

Introduction

Graphics pipeline
OpenGL pipeline
OpenGL syntax

Modelin

Procedural modeling OpenGL primitives GLUT primitives

Conclusion

```
Display:
void display () {
  Clear screen
        glClear(GL_COLOR_BUFFER_BIT);
  2 Viewpoint
        glMatrixMode(GL_PROJECTION);
        glOrtho(0.0,1.0,0.0,1.0,-1.0,1.0);
  3 For each object :
       1 Place object
         glMatrixMode(GL_MODELVIEW);
       2 Modify state machine
         glColor3f(1.0,1.0,1.0);
       O Draw
         glBegin(GL_POLYGON);
            glVertex3f(0.25,0.25,0.0);
            glVertex3f(0.75,0.25,0.0);
            glVertex3f(0.75,0.75,0.0);
            glVertex3f(0.25,0.75,0.0);
         glEnd();
```

Introduction

Graphics pipeline OpenGL pipeline OpenGL syntax

Modeling

modeling
OpenGL
primitives
GLUT primitives

Conclusion

```
Display:
void display () {
  Clear screen
        glClear(GL_COLOR_BUFFER_BIT);
  2 Viewpoint
        glMatrixMode(GL_PROJECTION);
        glOrtho(0.0,1.0,0.0,1.0,-1.0,1.0);
  G For each object :
       1 Place object
         glMatrixMode(GL_MODELVIEW);
       2 Modify state machine
         glColor3f(1.0,1.0,1.0);
       O Draw
         glBegin(GL_POLYGON);
            glVertex3f(0.25,0.25,0.0);
            glVertex3f(0.75,0.25,0.0);
            glVertex3f(0.75,0.75,0.0);
            glVertex3f(0.25,0.75,0.0);
         glEnd():
         glFlush(); // Execute OpenGL commands in hold
```

Plan

- 1 Introduction
- 2 Pipeline Graphics pipeline OpenGL pipeline OpenGL syntax
- Modeling Procedural modeling OpenGL primitives GLUT primitives
- 4 Conclusion

Plan

- 1 Introduction
- Pipeline Graphics pipeline OpenGL pipeline OpenGL syntax
- Modeling Procedural modeling OpenGL primitives GLUT primitives
- 4 Conclusion

Introduction

Graphics pipeline

OpenGL pipeline OpenGL syntax

Modeli

Procedural modeling

OpenGL primitives GLUT primitives

Conclusion

Procedural modeling

Complex object : combination of elementary elements :

- 1 Points (vertices): coordinates in a given reference frame
- 2 Lines : segments
- 3 Polygons : simple convex polygons

landari da antara

Introduction

Graphics pipeline
OpenGL pipeline
OpenGL syntax

Modelii

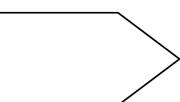
Procedural modeling

OpenGL primitives GLUT primitives

Conclusion

Example - planar pentagon

```
glBegin(GL_POLYGON);
  glVertex2f(0.0, 0.0);
  glVertex2f(0.0, 3.0);
  glVertex2f(4.0, 3.0);
  glVertex2f(6.0, 1.5);
  glVertex2f(4.0, 0.0);
glEnd();
glFlush();
```



Plan

- 1 Introduction
- 2 Pipeline Graphics pipeline OpenGL pipeline OpenGL syntax
- Modeling Procedural modeling OpenGL primitives GLUT primitives
- 4 Conclusion

Introduction

Pipeline
Graphics pipelin
OpenGL pipeline

Modelir

Procedur

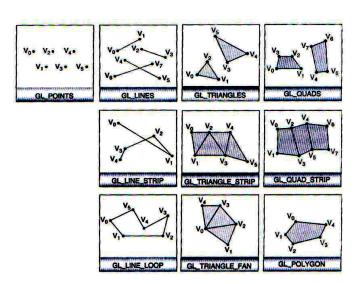
OpenGL primitives

GLUT primitives

GLU I primi

C l

OpenGL primitives



Parameters

Introduction

Graphics pipelin OpenGL pipeline OpenGL syntax

Modelii

Procedura

OpenGL

primitives

GLUT primitives

Conclusion

- Point size (in pixels): glPointSize(2.0);
- Line width (in pixels) : glLineWidth(3.0);

landar da antico

Graphics pipelin

OpenGL syntax

Modelir

Procedura modeling OpenGL

primitives GLUT primitives

Conclusion

- Point size (in pixels): glPointSize(2.0);
- Line width (in pixels) : glLineWidth(3.0);
- Line drawing: many stippling styles
- Different renderings for front and back faces :

```
glPolygonMode(GL_FRONT,GL_FILL);
glPolygonMode(GL_BACK,GL_LINE);
```

Introduction

Graphics pipeline OpenGL pipeline OpenGL syntax

Modelir

Procedural modeling OpenGL

primitives
GLUT primitives

Conclusion

- Point size (in pixels): glPointSize(2.0);
- Line width (in pixels) : glLineWidth(3.0);
- Line drawing : many stippling styles
- Different renderings for front and back faces: glPolygonMode(GL_FRONT,GL_FILL); glPolygonMode(GL_BACK,GL_LINE);
- Culling: glCullFace(GL_BACK); : back-faces non-visible

Introduction

Graphics pipeline OpenGL pipeline OpenGL syntax

Modelin

Procedural modeling OpenGL

primitives
GLUT primitives

Conclusion

- Point size (in pixels) : glPointSize(2.0);
- Line width (in pixels) : glLineWidth(3.0);
- Line drawing : many stippling styles
- Different renderings for front and back faces: glPolygonMode(GL_FRONT,GL_FILL); glPolygonMode(GL_BACK,GL_LINE);
- Culling: glCullFace(GL_BACK); : back-faces non-visible
- The color, normal, ..., at each vertex can be specified
- •

Introduction

Graphics pipelin OpenGL pipeline OpenGL syntax

Modelir

modeling OpenGL

primitives
GLUT primitives

Conclusion

- Point size (in pixels) : glPointSize(2.0);
- Line width (in pixels): glLineWidth(3.0);
- Line drawing : many stippling styles
- Different renderings for front and back faces: glPolygonMode(GL_FRONT,GL_FILL); glPolygonMode(GL_BACK,GL_LINE);
- Culling: glCullFace(GL_BACK); : back-faces non-visible
- The color, normal, ..., at each vertex can be specified
- . .
- Get current values : glGetFloatv(GL_LINE_WIDTH);

Introduction

Graphics pipeline
OpenGL syntax

Modeling Procedura

OpenGL primitives

GLUT primitives

Conclusion

Immediate definition of an objet

Each vertices and polygons are directly defined. Example of a triangle :

```
glBegin(GL_POLYGON);
  glNormal3fv(n0);
  glVertex3fv(v0);
  glNormal3fv(n1);
  glVertex3fv(v1);
  glNormal3fv(n2);
  glVertex3fv(v2);
glEnd();
```

Beware of the order: parameter (i.e. normal) before coordinates (state machine)

Introduction

Introductio

OpenGL syntax

Modeli

Procedural modeling

OpenGL

primitives
GLUT primitives

GLOT primit

Conclusion

Non immediate methods: Arrays

 Vertex-related data (coordinates, normals, colors, ...) can also be stored in arrays in the CPU memory

Introduction

Graphics pipeline
OpenGL pipeline

Modeling Procedural

OpenGL primitives

primitives GLUT primitives

Conclusion

Non immediate methods: Arrays

- Vertex-related data (coordinates, normals, colors, ...) can also be stored in arrays in the CPU memory
- Polygons refer to vertices through indices in these arrays

Introduction

Graphics pipeline
OpenGL pipeline
OpenGL syntax

Modeling

OpenGL primitives

primitives
GLUT primitives

Conclusion

Non immediate methods: Arrays

- Vertex-related data (coordinates, normals, colors, ...) can also be stored in arrays in the CPU memory
- Polygons refer to vertices through indices in these arrays
- Object are then defined with a reduced number of primitives
 - glDrawArrays(GL_QUADS, 0, 24)
 - glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_INT, vertices)

Introduction

Graphics pipeline
OpenGL pipeline
OpenGL syntax

Modeling

OpenGL primitives

GLUT primitives

Conclusion

Non immediate methods: Arrays

- Vertex-related data (coordinates, normals, colors, ...) can also be stored in arrays in the CPU memory
- Polygons refer to vertices through indices in these arrays
- Object are then defined with a reduced number of primitives
 - glDrawArrays(GL_QUADS, 0, 24)
 - glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_INT, vertices)
- Advanced methods (non studied in this course)
 - Display lists
 - Vertex Buffer Objects (VBO) : arrays are directly stored in the graphics card memory.

Plan

- 1 Introduction
- Pipeline Graphics pipeline OpenGL pipeline OpenGL syntax
- Modeling Procedural modeling OpenGL primitives GLUT primitives
- 4 Conclusion

Introduction

Introduction

OpenGL syntax

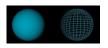
Modeling

Procedural modeling

primitives
GLUT primitives

Conclusio

GLUT primitives - 1/2



glutSolidSphere(radius, slices, stacks)
glutWireSphere(radius, slices, stacks)

Introduction

Introduction

OpenGL syntax

Modeling

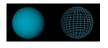
iviodeling

modeling
OpenGL
primitives
GLUT primitives

GLOT primitiv

Conclusio

GLUT primitives - 1/2



glutSolidSphere(radius, slices, stacks)
glutWireSphere(radius, slices, stacks)



glutSolidCube(size)
glutWireCube(size)

Introduction

Introduction

OpenGL syntax

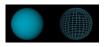
Modeli

Procedura modeling

GLUT primitives

Conclusio

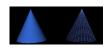
GLUT primitives - 1/2



glutSolidSphere(radius, slices, stacks)
glutWireSphere(radius, slices, stacks)



glutSolidCube(size)
glutWireCube(size)



glutSolidCone(base, height, slices, stacks)
glutWireCone(base, height, slices, stacks)

e de la companya de l

Introduction

OpenGL syntax

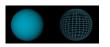
Modeli

Procedura modeling OpenGL

GLUT primitives

Conclusio

GLUT primitives - 1/2



glutSolidSphere(radius, slices, stacks)
glutWireSphere(radius, slices, stacks)



glutSolidCube(size)
glutWireCube(size)



glutSolidCone(base, height, slices, stacks)
glutWireCone(base, height, slices, stacks)



glutSolidTorus(innerRadius, outerRadius,
nsides, rings)
glutWireTorus(innerRadius, outerRadius,
nsides, rings)

Introduction

Pineline

OpenGL syntax

Modelin

Procedura

primitives
GLUT primitives

Conclusio

GLUT primitives - 2/2



glutSolidTetrahedron()
glutWireTetrahedron()

primitives **GLUT** primitives

GLUT primitives - 2/2





glutSolidTetrahedron() glutWireTetrahedron()

glutSolidOctahedron() glutWireOctahedron()

landar di castala

Introduction

Graphics pipeline

Modeli

Procedura modeling

primitives
GLUT primitives

Conclusion

GLUT primitives - 2/2







glutSolidTetrahedron()
glutWireTetrahedron()

glutSolidOctahedron()
glutWireOctahedron()

glutSolidDodecahedron()
glutWireDodecahedron()

Introduction

Introduction

Graphics pipeline

Modelii

Procedura modeling OpenGL

GLUT primitives

Conclusio

GLUT primitives - 2/2









glutSolidTetrahedron()
glutWireTetrahedron()

glutSolidOctahedron()
glutWireOctahedron()

glutSolidDodecahedron()
glutWireDodecahedron()

glutSolidIcosahedron()
glutWireIcosahedron()

Pipeline
Graphics pipeline
OpenGL pipeline

Modelin

Procedural modeling OpenGL

GLUT primitives

Conclusion

GLUT primitives - 2/2



glutSolidTetrahedron()
glutWireTetrahedron()



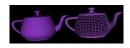
glutSolidOctahedron()
glutWireOctahedron()



glutSolidDodecahedron()
glutWireDodecahedron()



glutSolidIcosahedron()
glutWireIcosahedron()



glutSolidTeapot(size)
glutWireTeapot(size)

Remark: There is no Cylinder primitive.

Plan

- 1 Introduction
- 2 Pipeline Graphics pipeline OpenGL pipeline
- Modeling Procedural modeling OpenGL primitives GLUT primitives
- 4 Conclusion

Introduction

Graphics pipelir

OpenGL pipelin

Modeli

modeling
OpenGL
primitives
GLUT primitives

Conclusion

Conclusion:

- Done :
 - General process
 - Modeling : geometric primitives
- · Highlights:
 - State machine
 - Primitives
 - the redbook
- To do :
 - lab
 - modeling complex objects with primitives