

# COMPUTER GRAPHICS (CCG3013)

## LESSON 6

### USER INTERFACES AND INTERACTIONS: PART I



UOW  
MALAYSIA  
KDU PENANG  
UNIVERSITY COLLEGE

PART OF THE UNIVERSITY  
OF WOLLONGONG AUSTRALIA  
GLOBAL NETWORK

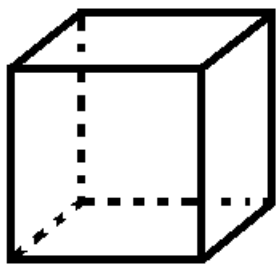


UNIVERSITY OF  
LINCOLN  
UNITED KINGDOM

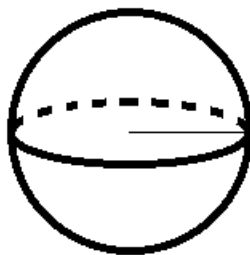
# PREVIOUS WEEK EXERCISE

Illustrate FIVE 3D primitive models with the corresponding labels.

# 3D PRIMITIVES MODELS



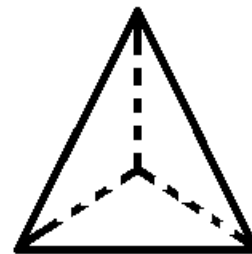
**Cuboid**



**Sphere**



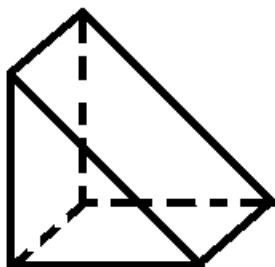
**Pyramid**



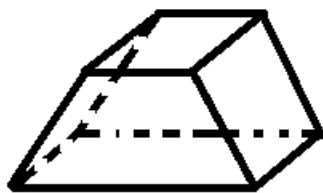
**Tetrahedron**



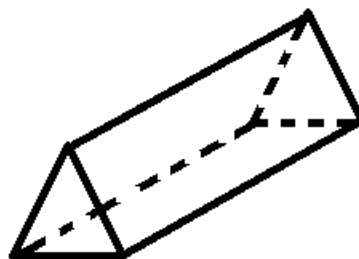
**Cone**



**Wedge**



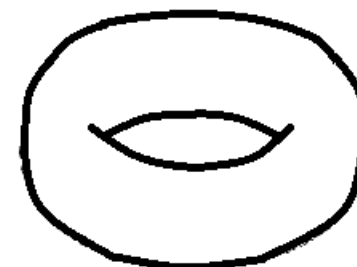
**Tripezoid**



**Prism**



**Cylinder**



**Torus**

# COURSE OUTLINE

Lesson	Topics
1	Introduction to computer graphics
2	Graphics hardware and software
3	Geometry in 2D graphics
4 & 5	Geometry in 3D graphics
6 & 7	User interfaces and interactions
8	Colour
9	Lighting and rendering
10 & 11	Motion and animation
12	Surface shadings

# LEARNING OUTCOMES

1. Explain events and callbacks for hardware inputs.
2. Manipulate 2D/3D view in the graphics window using input events and callbacks.

# ASSESSMENTS

Structure	Marks (%)	Hand-out	Hand-in
Assignment 1 (Individual)	30	Week 1(Unofficial) Week 3(Official)	Week 6
Assignment 2 (Group up to four only)	30	Week 1(Unofficial) Week 3(Official)	Week 12
Final examination	40	Exam week	

# CONTENT

No.	Topics	Duration (Minutes)
1	Mini lecturer 1: Font libraries	15
2	Exercise 1	10
3	Mini lecturer 2: OpenGL native font library	15
4	Exercise 2	10
5	Break	10
6	Mini lecturer 3: Keyboard controls	15
7	Exercise 3	10
8	Mini lecture 4: Mouse controls	15
9	Exercise 4	10

# MINI LECTURE 1

## FONT LIBRARIES

...



# TEXT RENDERING

1. The main function for a font library is to **draw text** in an application, which is in OpenGL window.
2. There are seven parameters for a text, which are **font type**, **font style**, **font size**, **font colour**, **font orientation**, **text in 2D or 3D space**, and **text kerning**.
3. **Glyph**, a textured quad for a letter.
4. **Texture atlas**, a textured quad for a set of letters.
5. Usually, the textured quad is stored in either **Targa** (.tga), **Bitmap** (.bmp) or **TrueType font** (.ttf) file format.

# FONT LIBRARIES

1. There are many font libraries available for OpenGL.
2. Four free font libraries include
  - (a) OpenGL native font library,
  - (b) OGLFT,
  - (c) OpenGLText, and
  - (d) FreeType

# OPENGL NATIVE FONT LIBRARY

1. It renders characters in a **Bitmap**.
2. It only renders two font types, which are **Times Roman** and **Helvetica**.
3. The font for each glyph or letter can be customized and rendered **pixel by pixel**.
4. The library can be extended by using the **display list**.

# OGGLFT

1. OGLFT stands for **OpenGL-FreeType Library**.
2. It renders text using **FreeType font library**.
3. It supports **Qt** graphics user interface (GUI).
4. For more information, refer to the [official website](#).

# OPENGLTEXT

1. This library is developed by **Tristan Lorach**, Nvidia developer from Santa Clara, CA.
2. It renders text using **TrueType font library**.
3. It may support **mobile application**.
4. For more information, refer to the [official website](#).

# FREETYPE

1. It is written in **C** language.
2. It renders text using **FreeType** in Bitmap.
3. Cross-platform, which included **Linux**, **iOS**, **Android**, **ChromeOS**, **ReactOS**, and **Ghostscript**.
4. For more information, refer to the [official website](#).

# EXERCISE 1

This activity will takes about 10 minutes.

1. Name two font types.
2. Name five font styles.
3. Explain text kerning.
4. State four free versions of font libraries.
5. Which font libraries using TrueType font?
6. Which font libraries using FreeType font?


LIM ZHE YUAN  in reply to LIM ZHE YUAN

a few seconds ago

1. Generic: primitive font type e.g serif,  
Family: collection of generic fonts
2. Bold, Italic, Underline, Strikethrough, Regular
3. Text kerning is the spacing between characters
4. OpenGL native font library, OGLFT, OpenGLText, FreeType
5. OpenGLText
6. OGLFT and FreeType

 Reply

 Like

More 



# MINI LECTURE 2

## OPENGL NATIVE FONT LIBRARY

...



# OPENGL NATIVE FONT LIBRARY

1. It renders characters in a **Bitmap**.
2. It only renders two font types, which are **Times Roman** and **Helvetica**.
3. The font for each glyph or letter can be customized and rendered **pixel by pixel**.
4. The library can be extended by using the **display list**.

# POSITION OF A TEXT

<b>Function name</b>	glRasterPos
<b>Purpose</b>	It specifies the raster position (top left corner of a text image) for pixel operations.
<b>Arguments or parameters</b>	(x, y) coordinates in OpenGL window.
<b>Return value</b>	None.

# RENDER A CHARACTER

<b>Function name</b>	glutBitmapCharacter
<b>Purpose</b>	It renders a bitmap character.
<b>Arguments or parameters</b>	FONT, refers to next slide; CHARACTER, ASCII code in integer type;
<b>Return value</b>	None.

# FONT

GLUT_BITMAP_8_BY_13	It fits each character in $8 \times 13$ pixels.
GLUT_BITMAP_9_BY_15	It fits each character in $9 \times 15$ pixels.
GLUT_BITMAP_TIMES_ROMAN_10	10 points Times Roman font.
GLUT_BITMAP_TIMES_ROMAN_24	24 points Times Roman font.
GLUT_BITMAP_HELVETICA_10	10 points Helvetica font.
GLUT_BITMAP_HELVETICA_12	12 points Helvetica font.
GLUT_BITMAP_HELVETICA_18	18 points Helvetica font.

# FUNCTION TO DRAW A TEXT

```
void drawText(const char *text, int length, int x, int y){  
    glMatrixMode(GL_MODELVIEW);  
    glPushMatrix();  
    glRasterPos2i(x, y);  
    for(int i=0; i<length; i++){  
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, (int) text[i]);  
    }  
    glPopMatrix();  
}
```

# FUNCTION TO CALL FOR DRAWING A TEXT

```
string text;
```

```
text = "Hello OpenGL native font";
```

```
glColor3f(1, 1, 1);
```

```
drawText(text.data(), text.size(), 400, 300);
```

# EXERCISE 2

This activity will takes about 10 minutes.

1. Native font library render characters using \_\_\_\_.
2. Name two font styles that the native font library can rendered.
3. To smoothen the edges of a text character, \_\_\_\_ is applied.



**LIM ZHE YUAN**  in reply to LIM ZHE YUAN

1. Bitmap
2. Times Roman, Helvetica
3. Antialiasing

 Reply

 Like

10 MINUTES BREAK

...



# MINI LECTURE 3

## KEYBOARD CONTROLS

...

# KEYBOARD EVENTS & CALLBACKS

1. Keyboard is one of the inputs to **interact with OpenGL applications**.
2. There are two event-handlers; **glutKeyboardFunc** and **glutSpecialFunc**, which handle **ASCII key events** and **special key events**, respectively.
3. ASCII stands for **American Standard Code for Information Interchange**.
4. Refers to **ASCII table**.

# ASCII KEY EVENTS

<b>Function name</b>	glutKeyboardFunc
<b>Purpose</b>	It callbacks for the ASCII key events on the current window.
<b>Arguments or parameters</b>	A callback function, which has key, an ASCII key code and (x, y), location of a mouse cursor.
<b>Return value</b>	None.

# SPECIAL KEY EVENTS

<b>Function name</b>	<code>glutSpecialFunc</code>
<b>Purpose</b>	It callbacks for the special key events on the current window.
<b>Arguments or parameters</b>	A callback function, which has key, such as function or arrow key code and (x, y), location of a mouse cursor.
<b>Return value</b>	None.

# CALLBACK FUNCTION

1. A conditional statement, either **if-else statement** or **switch statement** will be used to handle the keyboard events.
2. Any changes in the render will be updated using **glutPostRedisplay** function.

# EXERCISE 3

This activity will takes about 10 minutes.

Explain the following OpenGL functions in terms of its purpose, parameters involved, and return value.

- (a) `glutKeyboardFunc`,
- (b) `glutSpecialFunc`, and
- (c) `glutPostRedisplay`.

# MINI LECTURE 4

## MOUSE CONTROLS

...

# MOUSE EVENTS & CALLBACKS

1. Mouse is one of the inputs to **interact with OpenGL applications**.
2. There are three event-handlers; **glutMouseFunc**, **glutPassiveMotionFunc**, and **glutMotionFunc**, which handle **mouse button events**, **mouse move events**, and **mouse button and move events**, respectively.



# MOUSE BUTTON EVENTS

<b>Function name</b>	glutMouseFunc
<b>Purpose</b>	It callbacks whenever for mouse button events, which included mouse button pressed and released.
<b>Arguments or parameters</b>	A callback function which has button, either left button, middle button or right button; state, either button is pressed or released; (x, y), location of a mouse cursor;
<b>Return value</b>	None.

# MOUSE MOVE EVENTS

<b>Function name</b>	glutPassiveMotionFunc
<b>Purpose</b>	It callbacks whenever the mouse is moving, while no mouse button is pressed.
<b>Arguments or parameters</b>	A callback function which has (x, y), the coordinates of a mouse cursor.
<b>Return value</b>	None.

# MOUSE BUTTON AND MOVE EVENTS

<b>Function name</b>	glutMotionFunc
<b>Purpose</b>	It callbacks whenever the mouse is moving, while one or more mouse buttons are pressed.
<b>Arguments or parameters</b>	A callback function which has (x, y), the coordinates of a mouse cursor.
<b>Return value</b>	None.

# EXERCISE 4

1. Write a function in C++ OpenGL to translate a 2D shape at its current pivot point at (x, y).
2. Write a function in C++ OpenGL to rotate a 3D shape using a mouse button, with respect to the (x, y) position.

```
int x, y;
void onSpecialKey(char key, int mx, int my) {
    if (GLUT_KEY_UP) {
        y = y - 1;
    }
    if (GLUT_KEY_DOWN) {
        y = y + 1;
    }
    if (GLUT_KEY_LEFT) {
        x = x - 1;
    }
    if (GLUT_KEY_RIGHT) {
        x = x + 1;
    }
    glutPostRedisplay();
}
glutSpecialFunc(onSpecialKey);
glBegin(GL_QUADS);
glVertex2i(x, y);
glVertex2i(x + 50, y);
glVertex2i(x+50, y+50);
glVertex2i(x, y+50);
glEnd();
```

↩ Reply

Khoo Hee Kooi likes this  Like



**LIM ZHE YUAN**

1 hour ago

-condition incomplete

- follow cartesian coord origin (bottom left)

# REFERENCES

Main reference:

Hajek, D. (2019). Introduction to Computer Graphics 2019 Edition. Independently Published.

Additional reference:

Marschner, S. and Shirley, P. (2021). Fundamentals of Computer Graphics, 5th Edn. CRC Press: Taylor's & Francis.