Multimedia Technologies

Part 1

Theory and OpenGL

1. What is Multimedia?

Multimedia means combining more than one type of media to present information in a single presentation, application, lesson, advertisement, or video. A multimedia project may include text, images, graphics, audio, video, animation, and interactivity.

2. Main Types of Multimedia

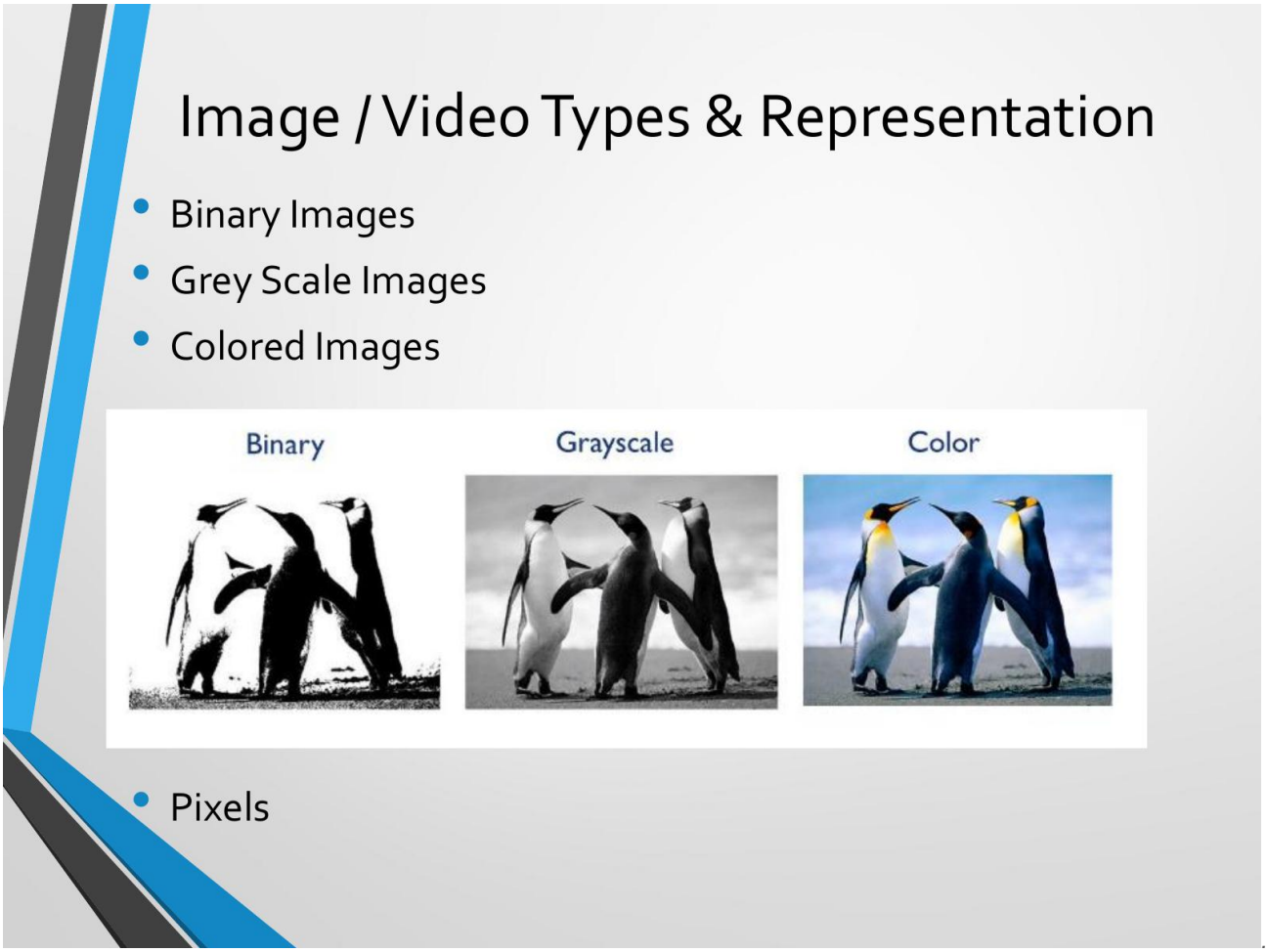
Type	Meaning	Examples
Text	Written information	Titles, subtitles, articles, PDF or DOCX files
Image / Graphics	Still visual content	Photos, posters, icons, diagrams
Audio	Sound content	Voice recording, music, sound effects
Video	Moving visual content	Tutorials, reels, films, advertisements
Animation	Moving drawings or objects	Motion graphics, cartoons, animated logos
Interactivity	User control or response	Buttons, menus, games, quizzes

3. Common Multimedia File Formats

Media Type	Common Extensions	Notes
Text	.txt, .doc, .docx, .pdf	Used for written documents and lecture notes.
Image	.jpg, .jpeg, .png, .gif, .bmp, .ico, .psd, .ai	Used for photos, graphics, icons, and design files.
Video	.mp4, .avi, .wmv	Used for moving visual content and edited videos.
Audio	.mp3, .wav	Used for music, voiceover, and sound recording.
Compressed	.rar, .zip	Used to collect or compress files into one archive.

4. Introduction to Digital Images

A digital image is made of small points called pixels. Each pixel stores a value that represents brightness or color. The number of bits used for each pixel is called bit depth. More bits usually allow more levels or more colors.



Types of Digital Images and Bits per Pixel

A. Binary Image

A binary image has only two possible pixel values: black and white. Each pixel needs only 1 bit.

- 0 can represent black.
- 1 can represent white.
- Common uses: simple masks, scanned text, simple shapes, and black-white diagrams.

B. Grayscale Image

A grayscale image contains shades between black and white. A common grayscale image uses 8 bits per pixel, which gives 256 gray levels.

- 0 = black
- 255 = white
- Values between 0 and 255 represent different gray shades.

C. Colored Image: RGB

A colored image commonly uses the RGB color model. RGB means Red, Green, and Blue. Each channel usually uses 8 bits, so one RGB pixel commonly uses 24 bits.

- Red channel = 8 bits
- Green channel = 8 bits
- Blue channel = 8 bits
- Total = 24 bits per pixel

D. Colored Image with Transparency: RGBA

RGBA adds an Alpha channel to RGB. The alpha channel controls transparency. RGBA commonly uses 32 bits per pixel.

Summary Table: Image Types

Image Type	Description	Common Bits per Pixel	Number of Values / Colors
Binary	Black and white only	1 bit	2 values
Grayscale	Shades of gray	8 bits	256 gray levels
RGB Color	Red, green, and blue channels	24 bits	About 16.7 million colors

Multimedia Technologies - Open GL

1- Setting Up (Initializing The Screen)

The header "windows.h" is needed for the Windows platform only

- GLUT header, which is guaranteed to include "glu.h" (for GL Utility) and "gl.h" (for Core OpenGL).

Core OpenGL (GL): consists of hundreds of commands, which begin with a prefix "gl" (e.g., glColor, glVertex, glTranslate, glRotate). The Core OpenGL models an object via a set of geometric primitives such as point, line and polygon.

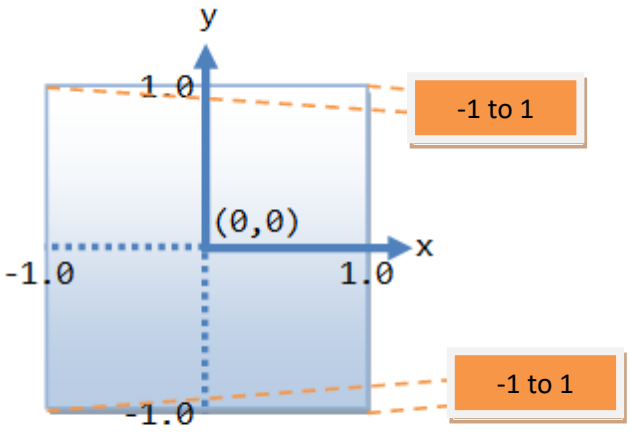
OpenGL Utility Library (GLU): built on-top of the core OpenGL to provide important utilities (such as setting camera view and projection) and more building models (such as qradric surfaces and polygon tessellation). GLU commands start with a prefix "glu" (e.g., gluLookAt, gluPerspective).

```
#include <windows.h>
#include <GL/glut.h>

void display()
 {
   glClearColor(0.0f, 0.0f, 0.3f, 0.0f);
   glClear(GL_COLOR_BUFFER_BIT);

///----- Objects code will be written here

    glFlush();
}

void main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitWindowSize(800,600);
   glutCreateWindow("My OpenGL Window");
   glutDisplayFunc(display);
   glutMainLoop();
}
```

- glClearColor(0.0f, 0.0f, 0.3f, 1.0f) : To set the background color (Red, Green , Blue , and Opacity) in the example, the background color is set to blue and opaque. The range of the color is between 0-1.
- glClear(GL_COLOR_BUFFER_BIT : Clear the color buffer and set the one used in previous command
- Objects will be created in this part. ////....
- glFlush(); Render the drawn objects.
- Void main : To call the functions that are responsible to draw the objects.
- glutInit: initializes GLUT, must be called before other GL/GLUT functions. It takes the same arguments as the main().
- glutInitWindowSize(800,600) : To set the screen dimensions (size)
- glutCreateWindow("My OpenGL Window") To create window with a title
- glutDisplayFunc(display): Display drawn objects
- glutMainLoop: enters the infinite event-processing loop, i.e., put the OpenGL graphics system to wait for events (such as re-paint), and trigger respective event handlers (such as display()).

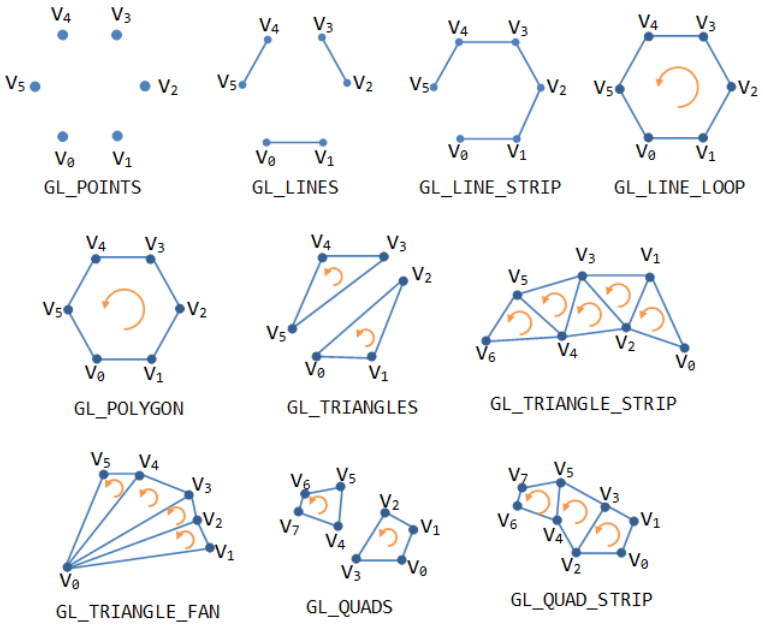
Screen Coordinates :

The default OpenGL 2D clipping-area (i.e., what is captured by the camera) is an orthographic view with x and y in the range of -1.0 and 1.0, i.e., a 2x2 square with centered at the origin. As shown in figure:



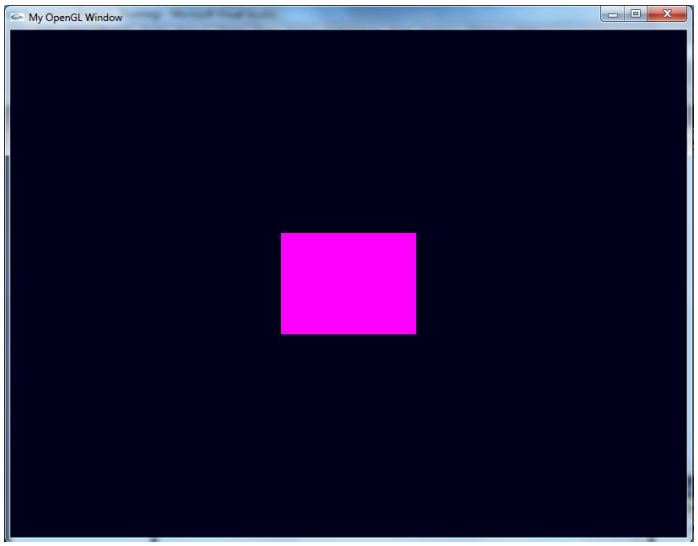
Clipping-Area
(default of 2x2 square centered at origin)

2-Different Geometric Primitives in Open GL:
As shown in figure, each primitive is consist of set of vertices, to draw a certain primitive, the name of that primitive should be written after GLBegin command , followed by the location of each pixel constructing that object.



OpenGL Primitives

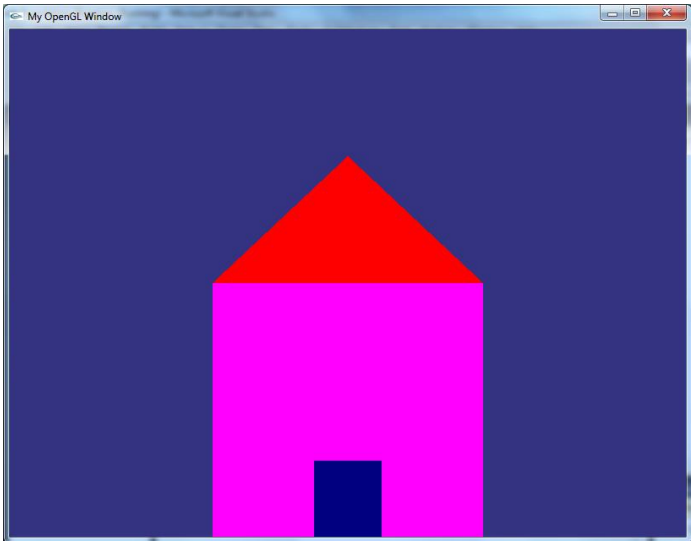
Example : An Open GL code to draw a square on the center of the screen with side length of 0.4.



Vertices are drawn in anti clockwise.

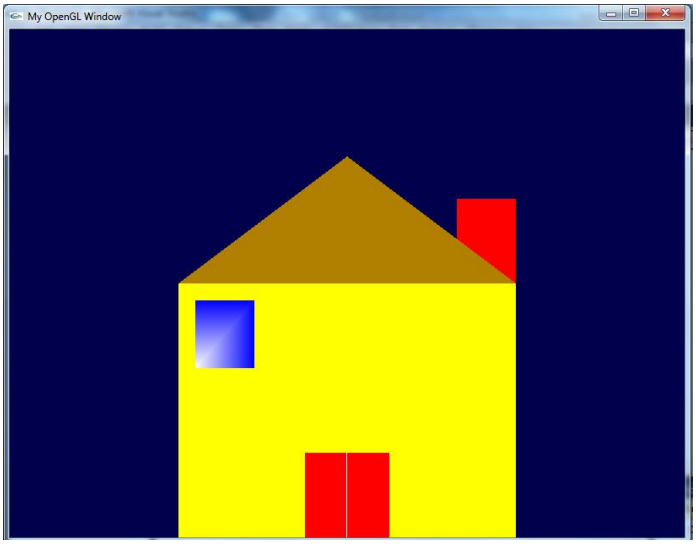
```
// Square
#include <windows.h>
#include <GL/glut.h>


void display()
 {
   glClearColor(0.0f, 0.0f, 0.1f, 0.0f);
   glClear(GL_COLOR_BUFFER_BIT);

glBegin(GL_QUADS);
   glColor3f(1.0f, 0.0f, 1.0f);

   glVertex2f(-0.2f,-0.2f);
   glVertex2f(0.2f,-0.2f);
   glVertex2f(0.2f,0.2f);
   glVertex2f(-0.2f,0.2f);


glEnd();

    glFlush();
}

void main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitWindowSize(800,600);
   glutCreateWindow("My OpenGL Window");
   glutDisplayFunc(display);
   glutMainLoop();
}
```

Ex : Write Open GL code to draw the following Object.




```
//House
glBegin(GL_QUADS);
    glColor3f(1.0f, 0.0f, 1.0f);
    glVertex2f(-0.4f,-1.0f);
    glVertex2f(0.4f,-1.0f);
    glVertex2f(0.4f,0.0f);
    glVertex2f(-0.4f,0.0f);
glEnd();
```

Home work:




```
glBegin(GL_QUADS);
    glColor3f(0.0f, 0.0f, 0.5f);
    glVertex2f(-0.1f,-1.0f);
    glVertex2f(0.1f,-1.0f);
    glVertex2f(0.1f,-0.7f);
    glVertex2f(-0.1f,-0.7f);
glEnd();

glBegin(GL_TRIANGLES);
 glColor3f(1.0f, 0.0f,0.0f);
  glVertex2f (0.4f,0.0f);
  glVertex2f (0,0.5f);
  glVertex2f (-0.4f,0);
glEnd();
   glFlush();
}
```

3-Translation in Open GL:

1-glTranslatef(X,Y,Z);
To translate(move) the object according to a certain value (0-1) towards X axis, Y axis and Z axis (which is towards the viewer)

2- glRotatef(Degree, X, Y, Z);
To rotate an object by angle specified in degrees about the axis which its value is 1.

Example beside shows the translation of a triangle towards Y axis and it is rotated by 180^o about X axis.

```
//Translate and Rotate
void display()
 {

   glClearColor(1.0f, 1.0f, 0.5f, 1.0f);
   glClear(GL_COLOR_BUFFER_BIT);

glTranslatef(0.0, 0.5f, 0.0f);
glRotatef(180,1.0, 0.0f, 0.0f);

glBegin(GL_TRIANGLES);
glColor3f(1.0f,0.0f,0.0f);
glVertex2f(-0.5f,0);
glVertex2f(0.5f,0);
glVertex2f(0.0f,0.5);
glEnd();
glFlush();

}
```

4-Animation:

Animation is the process of changing the location or the shape of a certain object during a period of time. To perform animation in open GL, idle function is needed In the idle() function, you could issue glutPostRedisplay command to post a window re-paint request, which in turn will activate display() function. We also use glPushMatrix to save the current state, perform transformations, and restore the saved state via glPopMatrix.

In following example the variable "mov" will be accumulated and used to specify the new position of the object.

H.W : Use animation to continously rotate the triangle drawn previous example around Z.

```
//Position Animation
#include <windows.h>
#include <GL/glut.h>
GLfloat mov = -1.0f;

void idle() {
   glutPostRedisplay();
}
void display()
 {
   glPushMatrix();
   glClearColor(1.0f, 1.0f, 0.5f, 1.0f);
   glClear(GL_COLOR_BUFFER_BIT);
   glTranslatef(mov, 0.0f, 0.0f);

    glBegin(GL_TRIANGLES);
       glColor3f(1.0f,0.0f,0.0f);
       glVertex2f(0,-0.5f);
       glVertex2f(0.5f,0);
       glVertex2f(0.0f,0.5);
    glEnd();
glPopMatrix();
glFlush();

mov = mov + 0.001;
if (mov>1)
       mov=-1;

}

void main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitWindowSize(800,600);
   glutCreateWindow("My OpenGL Window");
   glutDisplayFunc(display);
   glutIdleFunc(idle);
   glutMainLoop();
}
```

5-Keyboard Interaction

Two functions are used to interact with keyboard input one for special keys (like arrows) and other for traditional keys. Both are listed in the example. Here the object will be rotated clock wise while pressing the key 'a' from keyboard, while it will be rotated anti clock wise when pressing the left key.

H.W : Write OpenGL code to move a certain object to four directions according to following keyboard keys:
W : To move forward.
S : To move backward.
A : Moving to right.
D : Moving to left.

Also add the clockwise rotation by the key R.

References:

https://www3.ntu.edu.sg/home/ehchua/programming/opengl/CG_Introduction.html#zz-5.

Shreiner, D., Sellers, G., Kessenich, J., & Licea-Kane, B. (2013). OpenGL programming guide: The Official guide to learning OpenGL, version 4.3. Addison-Wesley.


```
//keyboard interaction
#include <windows.h>
#include <GL/glut.h>

GLfloat rot = -1.0f;

void idle() {
   glutPostRedisplay();
}
void display()
 {
   glPushMatrix();
   glClearColor(1.0f, 1.0f, 0.5f, 1.0f);
   glClear(GL_COLOR_BUFFER_BIT);
   glRotatef(rot, 0.0f, 0.0f,1.0f);

    glBegin(GL_TRIANGLES);
       glColor3f(1.0f,0.0f,0.0f);
       glVertex2f(0,-0.5f);
       glVertex2f(0.5f,0);
       glVertex2f(0.0f,0.5);
    glEnd();

glPopMatrix();
glFlush();
}

void specialKeys(int key, int x, int y) {
   switch (key) {
     case GLUT_KEY_LEFT:
rot = rot + 0.9;
if (rot>360)
       rot=-1;
         break;
   }
}
void keyboard(unsigned char key, int x, int
y) {
   switch (key) {
      case 'a':
         rot = rot - 0.9;
if (rot>360)
       rot=-1;
         break;
   }
}



void main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitWindowSize(800,600);
   glutCreateWindow("My OpenGL Window");
   glutDisplayFunc(display);
   glutIdleFunc(idle);
   glutSpecialFunc(specialKeys);
   glutKeyboardFunc(keyboard);
   glutMainLoop();
}
```