Ch2: OpenGL 2D Drawing

init —> create window —> display function —> loop

Display function: glClearColor, glClear, **glFlush**
glClear(GL_COLOR_BUFFER_BIT)
glClearColor: red, green, blue, alpha (0 – 1)

glBegin(); — glEnd();

Normalized Device Coordinates (NDC)
Is a display coordinate system that is screen–independent; it encompasses a 2D or 3D positions where the x, y, z components range from –1 to 1
Items are stretched and positioned proportionally.

glVertex
glVertex2f(x, y) — x, y are floats
glVertex2fv(p) — p is a pointer to an array (recognized as a vertex)

Initialization
glutInitDisplayMode: properties for the window – rendering context
– RGB color
– Single buffering
– Others
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB)

glutInitWindowSize

glutInitWindowPosition
The initial value of the initial window position is –1 and –1
If either X or Y is negative, it means that the actual window position is determined by the window system

Polygon Issues
Correct polygons display: (3)
– Simple: **edges** cannot cross
– Convex: all points on a **line segment** between two points in a polygon **are also in** the polygon
– Flat: all **vertices** are **in the same plane**

glColor3f: set the current color with 3 floats (normalized)

OpenGL state
OpenGL operates as a state machine: once the value of a property is set, the value persists **until a new value is set**.

Question

a) What is the general format of the `glVertex` function? Explain why it needs several versions and provide examples.

[5 marks]

The general format of the glVertex function is glVertex2f(x, y), x and y should be floats. There are several versions like glVertex2i(x, y) which is suitable for the case that x and y are integers and glVertex2d(x, y) which is suitable for double cases. These are created in the scenario of facing different data types. Another version is glVertex2v(p) which takes the pointer to an array as the input. Sometimes we want to set the variables for vertices at the beginning of our program, with the help of this function, we can set an array to involve all the vertices in and then use it later.

Solution: The general format of the function is: glVertex for the function name (1 mark), then a number that indicates the number of arguments (1 mark), then a letter that indicates the type of the arguments (1 mark). Examples: glVertex3f, glVertex2d (1 mark). The several versions are needed because **OpenGL is not object oriented** and so does not allow function **overloading** (1 mark).

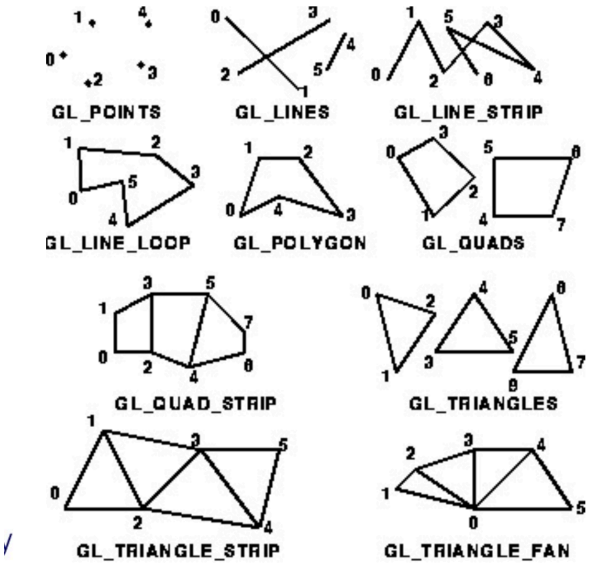
Vertices in OpenGL Primitives

GL_LINES & GL_LINE_STRIP & GL_LINE_LOOP
GL_LINES: connect vertices in pairs (two vertices in a pair)
GL_LINE_STRIP: follow the order of vertices and connect them together
GL_LINE_LOOP: follow the order of vertices and connect them together, also connect the last vertex with the first vertex

GL_POLYGON & GL_QUADS
GL_POLYGON:
GL_QUADS: 4 vertices in a group, without sharing edges
GL_QUAD_STRIP: 4 vertices in a group, sharing edges

GL_TRIANGLES & GL_TRIANGLE_STRIP & GL_TRIANGLE_FAN
GL_TRIANGLES: 3 vertices in a group, without sharing edges
GL_TRIANGLE_STRIP: 3 vertices in a group, with sharing edges
GL_TRIANGLE_FAN: draw each triangle to form a fan

OpenGL primitives



Two ways of drawing circles

1. GL_LINE_LOOP — draw a hollow circle
2. GL_TRIANGLE_FAN — draw a solid circle

Ch3: OpenGL Coordinates and Viewing

Five Coordinate Spaces

Local space — World space — View space — Clip space — Screen space

Local space: Local coordinates are the coordinates of object relative to its local origin
World space: World coordinates are the coordinates of world relative to some global origin of the world, together with many other objects also place relative to world's origin
View space: each coordinate is as seen from the camera or viewer's point of view
Clip space: projected to –1 to 1 range, determine which vertices will end up on the screen
Screen space: transform the coordinates from –1 to 1 range to the coordinate range defined by glViewport

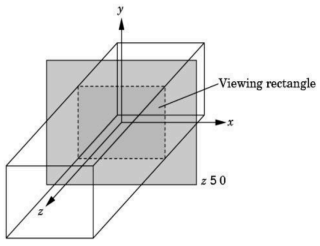
Why do we have different spaces?

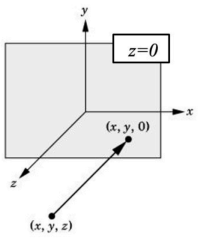
Some operations are **easier to use** in **certain coordinate systems**
(e.g. Modify an object — local space)

Viewing volume

Viewing volume: determine what appears in an image or what can be seen by users.
3D dimension — viewing volume | 2D dimension — viewing rectangle

Orthographic Viewing
In the default orthographic view, points are projected forward along the z axis onto the **plane z = 0**.





OpenGL Camera: a default camera at the origin of the space point in the negative z direction

Default viewing volume
The default viewing volume: a box **centered at the origin with a side of length 2**
glMatrixMode(GL_PROJECTION);
glLoadIdentity();

glOrtho(–1.0, 1.0, –1.0, 1.0, –1.0, 1.0); — left, right, bottom, top, near, far

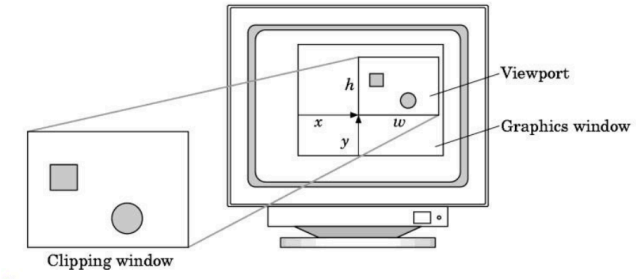
Using Traditional Coordinates
Moving projection to the right by half and to the top by half allows for a more traditional coordinate system. glOrtho2D(0, width, 0, height);

Reshape an Image
Principle: when the window size is enlarged or reduced, enlarge or reduce the viewing volume (rectangle) with the window size.

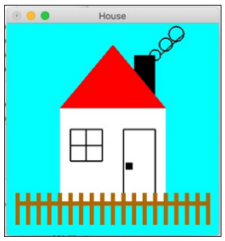
```
void reshape(GLsizei width, GLsizei height) {// GLsizei for non-negative integer

   // Compute aspect ratio of the new window
   if (height == 0) height = 1;                // To prevent divide by 0
   GLfloat aspect = (GLfloat)width / (GLfloat)height;

   // Change the aspect ratio
   glLoadIdentity();             // Reset the projection matrix
   if (width >= height) {
     // aspect >= 1, set the height from -1 to 1, with larger width
      gluOrtho2D(-1.0 * aspect, 1.0 * aspect, -1.0, 1.0);
   } else {
      // aspect < 1, set the width to -1 to 1, with larger height
     gluOrtho2D(-1.0, 1.0, -1.0 / aspect, 1.0 / aspect);
   }
}
```

Viewports

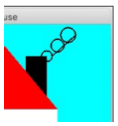
glViewport: specifies the transformation of x and y from NDC to window coordinates
glViewport(x, y, w, h)



For example:



glutInitWindowSize(300, 300)



glOrtho2D(0, 2, 0, 2)

glViewport(0, 0, 600, 600)

Question

b) Write an OpenGL orthographic projection statement which guarantees that a sphere of radius 5 can be entirely contained in the viewing volume without distortion.

[5 marks]

Answer:
Since the radius os 5, so the viewing volume should be a rectangle with side length larger than 10.
glOrtho(–15.0, 15.0, –15.0, 15.0, –15.0, 15.0);

Question

c) This question is about the OpenGL camera.

[6 marks]

i) What are the characteristics of the default OpenGL camera?

(3 marks)

ii) Which OpenGL function allows you to set the location and orientation of the camera? How does it work?

(3 marks)

Answer:
(1) The default OpenGL camera is **at the origin of the coordinate system** in the **negative z direction**. The **viewing volume** is in default a cube with a side of length 2. The default projection is **orthographic**.

Question:

c) This question is about the OpenGL camera.

[6 marks]

i) What are the properties of the default OpenGL camera?

(3 marks)

ii) Write the arguments of the glOrtho function that correspond to the default camera.

(3 marks)

(1) The camera is at the origin of the coordinate system in the negative z direction. The viewing volume is a cube with side length of 2. The default projection is orthographic.
(2) glOrtho(–1.0, 1.0, –1.0, 1.0, –1.0, 1.0);

Ch4: OpenGL Events and Animation

Callbacks and GLUT Callbacks
Define a callback function for each type of event the graphics system recognizes

glutDisplayFunc — display function callback
glutReshapeFunc
glutIdleFunc
glutMouseFunc/glutMotionFunc/glutKeyboardFunc/…

GLUT Event Loop
In each pass through the event loop
1. Looks at the events in the queue
2. For each event in the queue, GLUT executes the appropriate callback function
3. No callback is defined, the even is ignored

The Display Callback — must have in a GLUT program
The display callback execute whenever the GLUT determines that the window should be **refreshed** (first is opened, is reshaped, …)

Posting Redisplays
Motivation: can lead to multiple executions of the display callback on a single pass through the event loop.
glutPostRedisplay(): GLUT checks to see if the flag is set at the end of the event loop

Idle Callback
The idle callback is executed whenever there are **no events** in the event queue

Animation Mechanism

glClear(): start by clearing the window (redraw s.th.)

Double Buffering
Motivation: the drawing of info. in the frame buffer is decoupled from the display of its contents
• Front Buffer: one that is displayed but not written to — display buffer
• Back Buffer: one that is written to but not displayed — drawing buffer
glutInitDisplayMode(GL_RGB | GL_DOUBLE)

```
void display() {
        glClear(GL_COLOR_BUFFER_BIT);
        /* draw something */
        glutSwapBuffers();
```

}

glutSwapBuffers() — at the end of the display callback, buffers are swapped

Question:

b) Consider the code shown in **Code Box 1** on the next page.

[11 marks]

i) Briefly describe what appears on screen when executing this code.

(2 marks)

ii) Select four statements in this code that contribute to the creation of a smooth animation and explain their respective purpose.

(8 marks)

iii) Modify one line in the code to make the animation go faster.

(1 mark)

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

#define DEGREES_TO_RADIANS 3.14159/180.0

static GLfloat spin = 0.0;
GLfloat a, b;

void square()
{
    glBegin(GL_QUADS);
      glVertex2f(a , b);
      glVertex2f(-b , a);
      glVertex2f(-a , -b);
      glVertex2f(b , -a);
   glEnd();
}

void display()
{
    glClear (GL_COLOR_BUFFER_BIT);
    square();
    glutSwapBuffers ();
}

void spinDisplay (void)
{
    spin = spin + 5.0;
    if (spin > 360.0) spin = spin - 360.0;
    a = 0.5 * cos(DEGREES_TO_RADIANS * spin);
    b = 0.5 * sin(DEGREES_TO_RADIANS * spin);
    glutPostRedisplay();
}

int main(int argc, char** argv)
{
   glutInit(&argc,argv);
   glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
   glutInitWindowPosition(500,0);
   glutCreateWindow("double buffered");
   glutDisplayFunc(display);
   glutIdleFunc (spinDisplay);
   glutMainLoop();
}
```

(1) The rotation of a square with side length 1. (Rotating square figure)
(2)

GLUT_DOUBLE: double buffers avoid the issue that the drawing information of frame buffer decoupled with the current display drawings. Double buffers includes

glutPostRedisplay(): At the end of the display callback, glutSwapBuffers() function asks the GLUT to switch the buffer of the front buffer and the back buffer to change the value of the vertices positions. (It replaces the glFlush() command)

glutPostRedisplay(): the statement requests that the display callback be executed **after** the idle callback (worked as a flag). Avoid the issue that display callback **executes multiple times** on a **single pass** through the event loop.

glutIdleFunc(): it executes whenever there is no event in the event loop. The idle callback is used to **update the back buffer** (specifically the position of each vertices of the square) to create the animation.

(3) spin = spin + 10.0;

Ch5: OpenGL 3D Drawing

Rotation and Scaling
Rotation: glRotatef(theta, x, y, z) — usually rotate one axis of direction once
- glMatrixMode(GL_MODELVIEW);

Scaling: multiply the vertices with a scaling parameter

glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
glEnable(GL_DEPTH_TEST);

GLUT objects
Curved Surface:
Slices — the number of subdivisions **around** the Z axis (longitude)
Stacks — the number of subdivisions **along** the Z axis (latitude)