3D Graphics Programming Tools

OpenGL Events and Animation

Dr. Xianhui Cherry Che

x.che@qmul.ac.uk

Learning Objectives

- Get familiar with event-driven input in OpenGL
- Grasp basic understanding of double buffering for smooth animations
- Skills of programming event input with GLUT

Event Mode

- Most systems have more than one input device, each of which can be triggered at an arbitrary time by a user
- Each trigger generates an *event* whose measure is put in an *event queue* which can be examined by the user program

Event Types

- **Window:** resize, expose, iconify
- **Mouse:** click one or more buttons
- **Motion:** move mouse
- **Keyboard:** press or release a key
- **Idle:** nonevent
  - Define what should be done if no other event is in the queue

Callbacks

- Programming interface for event-driven input
- Define a *callback function* for each type of event the graphics system recognizes
- This user-supplied function is executed when the event occurs
- GLUT example:
  `glutDisplayFunc(myDisplay)`

display callback function

5

GLUT Callbacks

GLUT recognizes a subset of the events recognized by any particular window system (Windows, X, Macintosh)

- **`glutDisplayFunc`** *(Already learned so far)*
- **`glutIdleFunc`**
- **`glutMouseFunc`**
- **`glutMotionFunc, glutPassiveMotionFunc`**
- **`glutKeyboardFunc, glutKeyboardUpFunc`**
  `glutSpecialFunc, glutSpecialUpFunc`
- **`glutReshapeFunc`** *(Already learned so far)*

GLUT Event Loop

- Recall that the last line in **`main`** for a program using GLUT must be

```
glutMainLoop();
```

which puts the program in an infinite event loop

- In each pass through the event loop, GLUT
  - looks at the events in the queue
  - for each event in the queue, GLUT executes the appropriate callback function if one is defined
  - if no callback is defined for the event, the event is ignored

7

main

```
int main (int argc, char** argv) {
  glutInit (&argc, argv);
  glutInitWindowSize (ww, wh);
  glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
  glutCreateWindow ("interactive");
  myinit ();
  glutReshapeFunc (myreshape);
  glutMouseFunc (mymouse);
  glutDisplayFunc (mydisplay);
  glutMainLoop ();
}
```

The Display Callback

- The display callback is executed whenever the GLUT determines that the window should be refreshed, for example
  - When the window is first opened
  - When the window is reshaped
  - When a window is exposed
  - When the user program decides it wants to change the display (i.e. draw)
- In `main`
  - `glutDisplayFunc(mydisplay)` identifies the function to be executed
  - **Every GLUT program must have a display callback.**

Posting Redisplays

- Many events may invoke the display callback function
  - Can lead to multiple executions of the display callback on a single pass through the event loop
- We can avoid this problem by instead using

```
glutPostRedisplay();
```

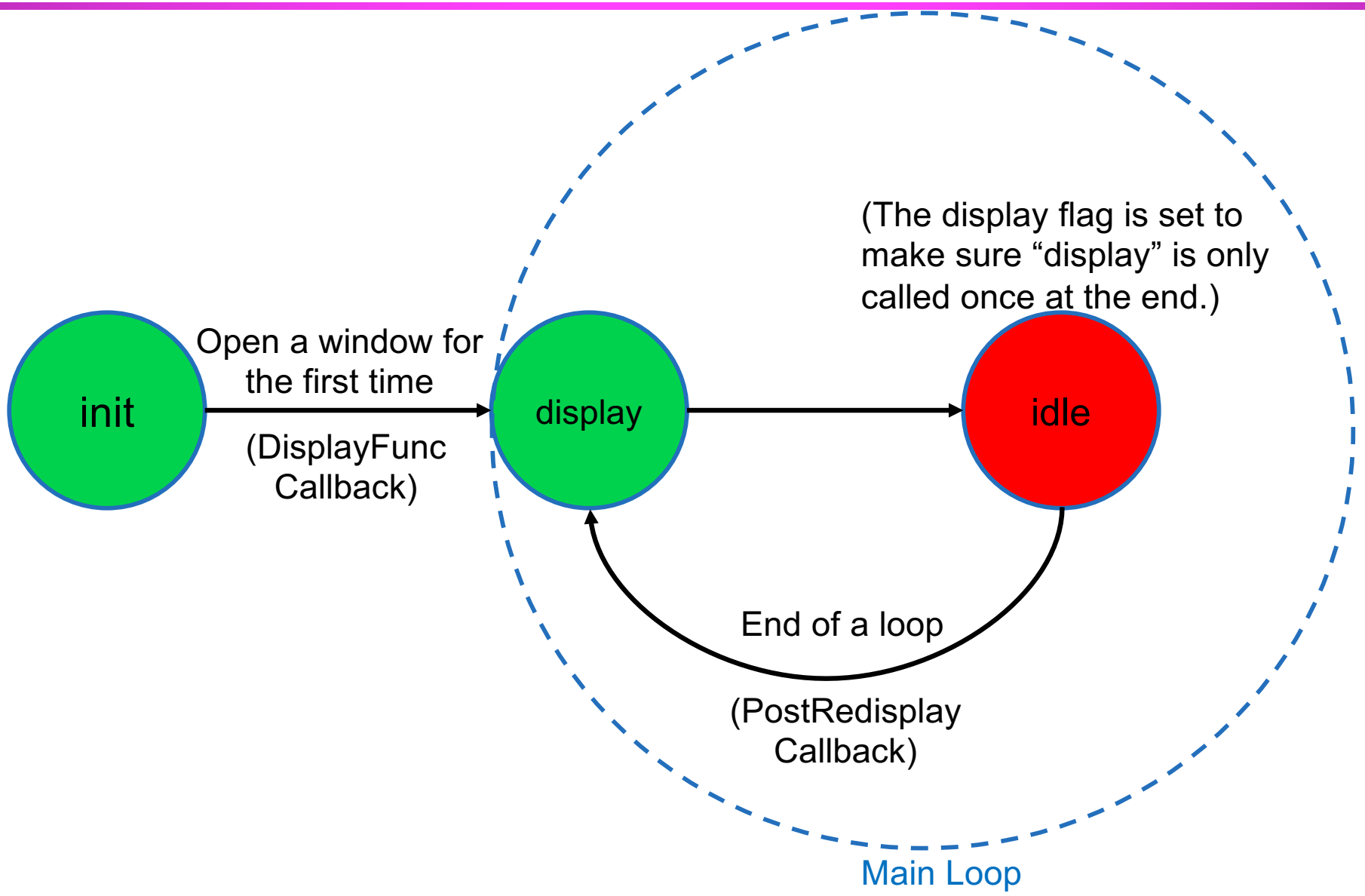
  which sets a flag.
- GLUT checks to see if the flag is set at the end of the event loop
- If set then the display callback function is executed

Using the Idle Callback

- The idle callback is executed whenever there are no events in the event queue
  - **`glutIdleFunc(myidle)`**
- Useful for animations

```
void idle() {
      glutPostRedisplay();
}
Void display() {
      /* draw something */
      }
int main(int argc, char **argv) {
      ... ...
      glutDisplayFunc(display);
      glutIdleFunc(idle);
      glutMainLoop();
}
```

State Machine of PostRedisplay



Animating a Display

- When we redraw the display through the display callback, we usually start by clearing the window
  - `glClear()`

  then draw the altered display

- Problem: the drawing of information in the frame buffer is decoupled from the display of its contents
  - Hence we can see partially drawn displays

Double Buffering

- Instead of one color buffer, we use two
  - **Front Buffer**: one that is displayed but not written to → *display buffer*
  - **Back Buffer**: one that is written to but not displayed → *drawing buffer*
- Program must request a double buffer in main
  `-glutInitDisplayMode(GL_RGB | GL_DOUBLE)`
- At the end of the display callback, buffers are swapped

```
void display()
{
        glClear(GL_COLOR_BUFFER_BIT);

       /* draw graphics here */

        glutSwapBuffers();

}
```

Using Global Variables

- The form of all GLUT callbacks is fixed
  - **`void display()`**
- Must use globals to pass information to callbacks

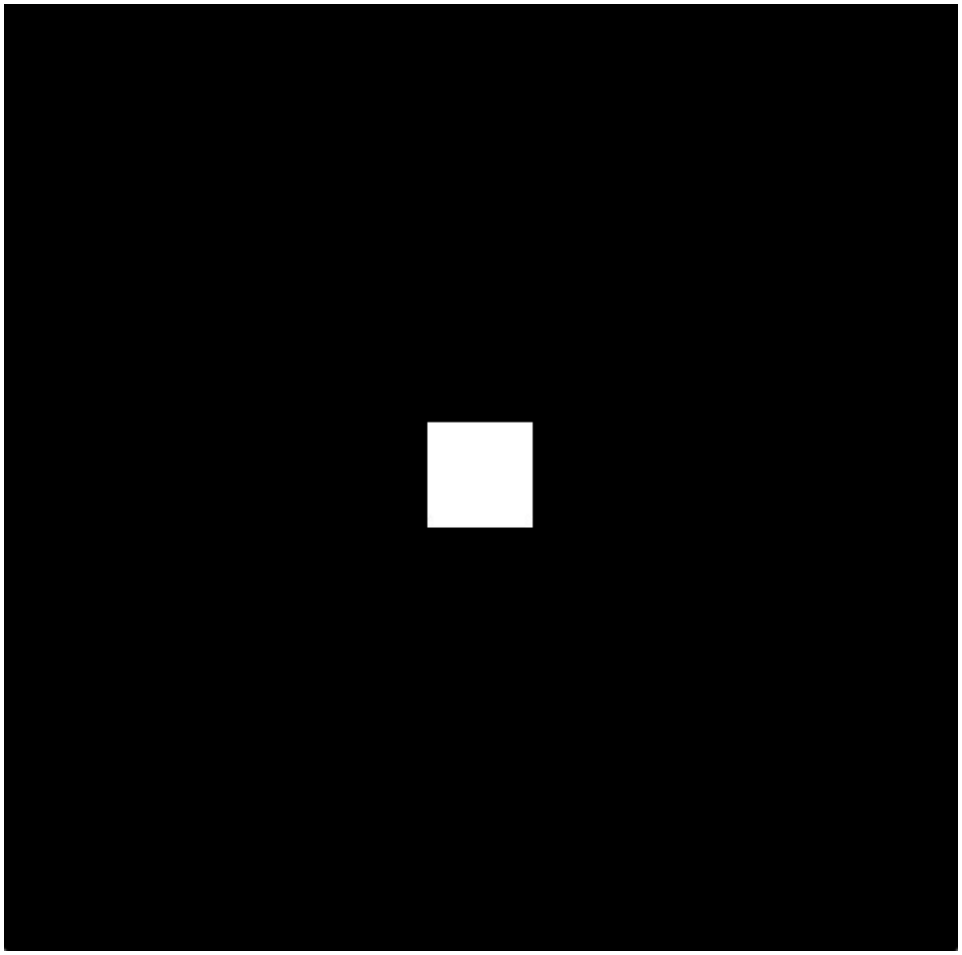
```
float t; /*global */

void display()  {
    /* draw something that depends on t */
    ... ...
    glutSwapBuffers();
}
void idle ()   {
    /* change t */
    ... ...
    glutPostRedisplay();
}
```

Full Animation Template

```
#include <GLUT/glut.h>
#include <math.h>
//Declare some global variables if needed
... ...

void idle() {
    //Change the value of global variables to change the image
    ... ...
    glutPostRedisplay();
}
void drawSomething(){
    //Code of GL drawing
    ... ...
}
void display() {
    glClear (GL_COLOR_BUFFER_BIT);
    drawSomething();
    glutSwapBuffers ();
}
```

```
void init() {
    glClearColor (0.0, 0.0, 0.0, 1.0);
    glColor3f (1.0, 1.0, 1.0);
}
int main(int argc, char **argv) {
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
  glutInitWindowSize(500, 500);
  glutCreateWindow("Animation");
  init();
  glutDisplayFunc(display);
  glutIdleFunc(idle);
  glutMainLoop();
}
```

Expand a Square



Expand a Square

```
#include <GLUT/glut.h>

GLfloat x=0.05, y=0.05;

void idle() {
    x += 0.05;
    y += 0.05;
    glutPostRedisplay();
}
void square(){
    glBegin(GL_POLYGON);
    glVertex2f(x , y);
    glVertex2f(-y , x);
    glVertex2f(-x , -y);
    glVertex2f(y , -x);
    glEnd();
}
void display() {
    glClear (GL_COLOR_BUFFER_BIT);
    square();
    glutSwapBuffers ();
}
```

When idle, make some changes and set a flag for redisplay.

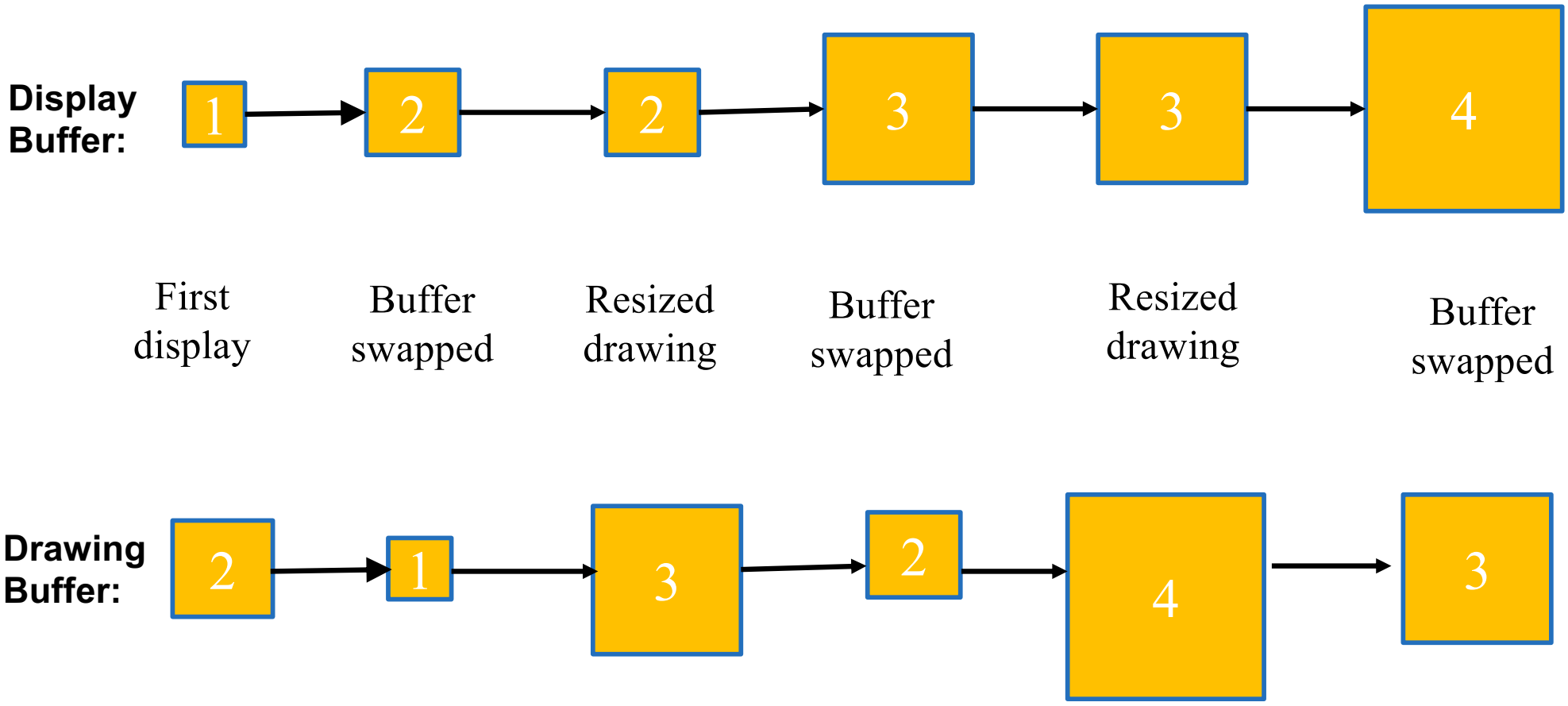
Clear the display, draw a square, then swap the buffers

```
void init()
{
    glClearColor (0.0, 0.0, 0.0, 1.0);
    glColor3f (1.0, 1.0, 1.0);
}
int main(int argc, char **argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
  glutInitWindowSize(500, 500);
  glutCreateWindow("Expand a Square");
  init();
  glutDisplayFunc(display);
  glutIdleFunc(idle);
  glutMainLoop();
}
```

Queen Mary University of London

Expand a Square

Event Transitions:



Expand a Square – Using Zooming

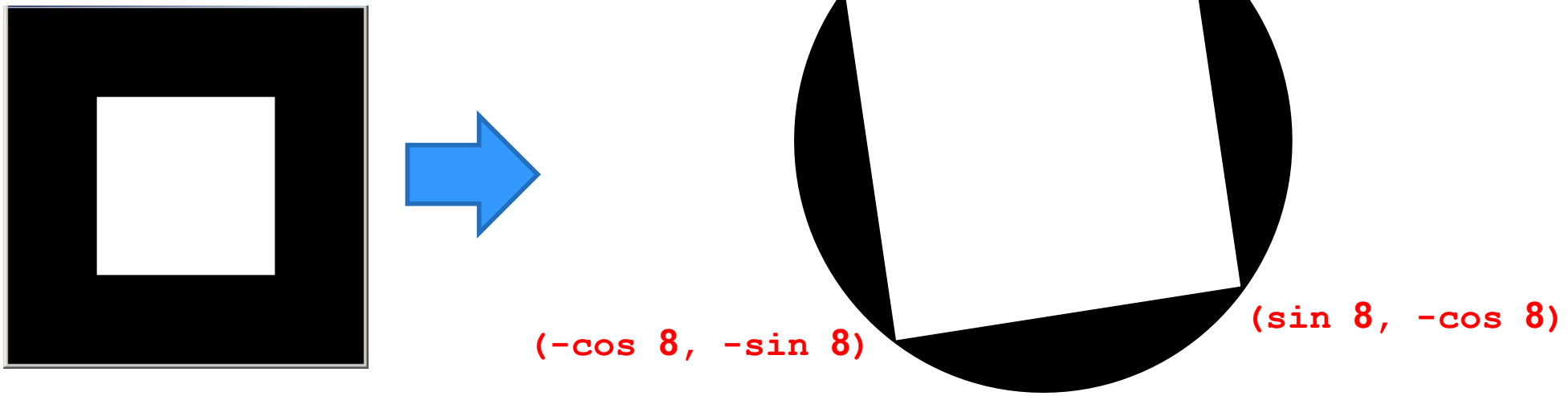
```
#include <GLUT/glut.h>

Gldouble zoom = 1.0;

void idle() {
    if (zoom >0) zoom -= 0.08;
    glutPostRedisplay();
}
void square(){
    glBegin(GL_POLYGON);
    glVertex2f(0.05, 0.05);
    glVertex2f(-0.05, 0.05);
    glVertex2f(-0.05, -0.05);
    glVertex2f(0.05, -0.05);
    glEnd();
}
void display() {
    glClear (GL_COLOR_BUFFER_BIT);
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    gluOrth2D(-zoom, zoom, -zoom, zoom);
    square();
    glutSwapBuffers ();
}
```

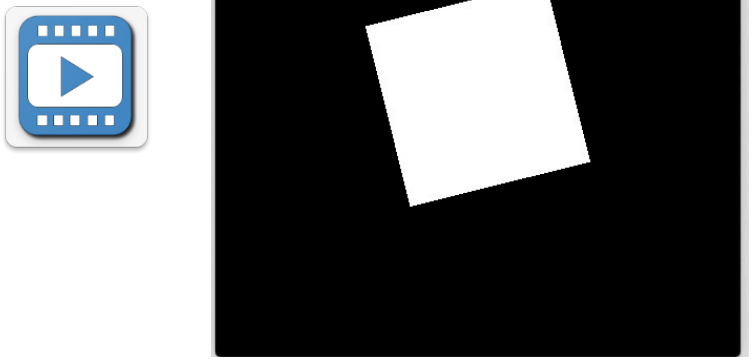
The previous approach increases the square every time.

This approach is to keep the same square and zoom in every time.

```
void init()
{
    glClearColor (0.0, 0.0, 0.0, 1.0);
    glColor3f (1.0, 1.0, 1.0);
}
int main(int argc, char **argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
  glutInitWindowSize(500, 500);
  glutCreateWindow("Expand a Square");
  init();
  glutDisplayFunc(display);
  glutIdleFunc(idle);
  glutMainLoop();
}
```

Rotate a Square





Queen Mary University of London

Rotate a Square

```
#include <GLUT/glut.h>
#include <math.h>
#define DEGREES_TO_RADIANS 3.14159/180.0
GLfloat theta = 0.0;
GLfloat x, y;

void idle() {
    theta = theta + 1.0;
    if (theta > 360.0) theta = theta - 360.0;
    x = 0.5 * cos(DEGREES_TO_RADIANS * theta);
    y = 0.5 * sin(DEGREES_TO_RADIANS * theta);
    glutPostRedisplay();
}
void square(){
    glBegin(GL_POLYGON);
    glVertex2f(x , y);
    glVertex2f(-y , x);
    glVertex2f(-x , -y);
    glVertex2f(y , -x);
    glEnd();
}
void display() {
    glClear (GL_COLOR_BUFFER_BIT);
    square();
    glutSwapBuffers ();
}
```

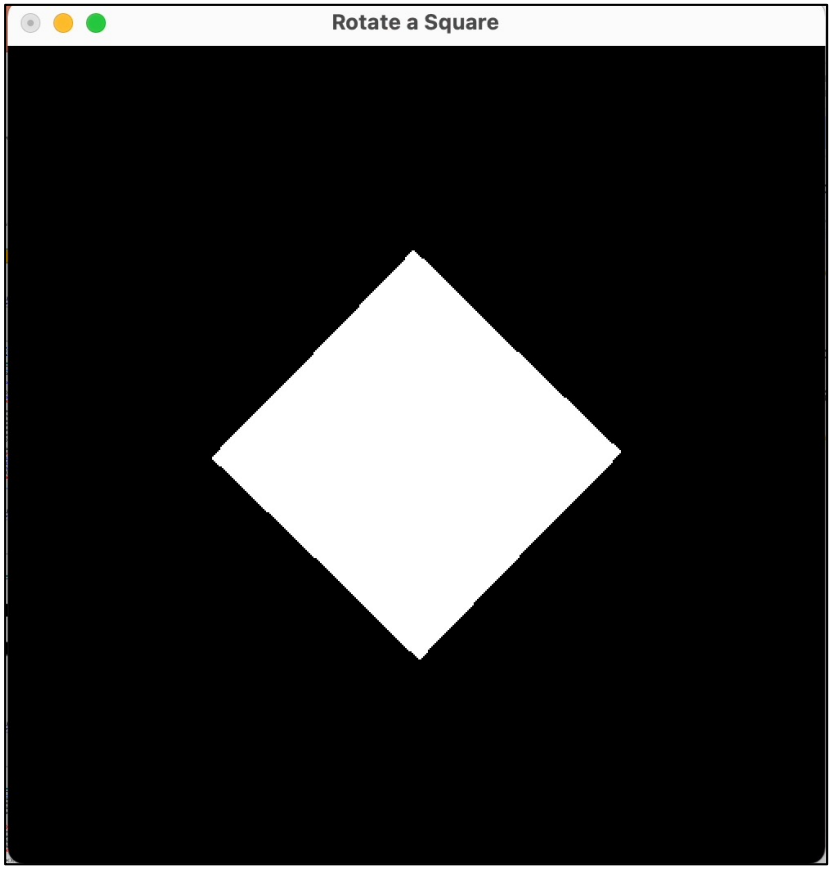
Why is it important to reset the display every time?

```
void init()
{
    glClearColor (0.0, 0.0, 0.0, 1.0);
    glColor3f (1.0, 1.0, 1.0);
}
int main(int argc, char **argv)
{
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
  glutInitWindowSize(500, 500);
  glutCreateWindow("Rotate a Square");
  init();
  glutDisplayFunc(display);
  glutIdleFunc(idle);
  glutMainLoop();
}
```

Queen Mary University of London

Why Reset Display Every Time?

If **`glClear (GL_COLOR_BUFFER_BIT)`** is placed in the **`init()`** instead of **`display()`**:



Questions?

x.che@qmul.ac.uk