

```
//~~~~~|Main.java|~~~~~\|
```

```
import com.sun.opengl.util.Animator;  
import java.awt.Cursor;  
import java.awt.Frame;  
import java.awt.GraphicsEnvironment;  
import java.awt.Image;  
import java.awt.Point;  
import java.awt.Rectangle;  
import java.awt.Toolkit;  
import java.awt.event.InputEvent;  
import java.awt.event.KeyEvent;  
import java.awt.event.KeyAdapter;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
import java.awt.image.BufferedImage;  
import javax.media.opengl.GL;  
import javax.media.opengl.GLAutoDrawable;  
import javax.media.opengl.GLCanvas;  
import javax.media.opengl.GLEventListener;  
import javax.media.opengl.glu.GLU;
```

```
/*  
 * Main sets up the window and GL context. Mouse and key events are captured here  
 * and used to set boolean values in the Game class. The Game class polls  
 * the boolean values each loop.  
 *  
 * @author Stephen Jones  
 */
```

```
public class Main implements GLEventListener{  
    private Game game;//decides what to do with events that occur here  
    private static boolean fullscreen=false;//can be toggled  
    private int w=1024;//default if not fullscreen  
    private int h=768;  
    /*  
     * We need to re-center the mouse cursor after it has moved otherwise it  
     * will run off the window and no longer throw events. center holds  
     * coordinates that define the center of the visible screen.  
     */  
    private static Point center;  
    private Point mousePos;  
    /*  
     * A class can be developed that allows the player to set their  
     * own preference for which keys do what. These variables would hold  
     * those preferences. The same is true for mouse events.  
     */  
    private int forward=KeyEvent.VK_W;  
    private int backward=KeyEvent.VK_S;  
    private int strafel=KeyEvent.VK_A;  
    private int strafer=KeyEvent.VK_D;  
    private int shoot=InputEvent.BUTTON1_MASK;  
    private int use=InputEvent.BUTTON3_MASK;  
  
    /*  
     * Setup the window, animator and hide mouse cursor  
     */  
    public static void main(String[] args) {  
        Frame frame = new Frame("JOGL Events");  
        //replace the visible cursor with an invisble image  
        Toolkit t=Toolkit.getDefaultToolkit();  
        Image img=new BufferedImage(1, 1, BufferedImage.TYPE_INT_ARGB);  
        Cursor pointer=t.createCustomCursor(img, new Point(0,0), "none");  
        //the canvas needs to have focus for the keys to work at the start
```

```

GLCanvas canvas = new GLCanvas();
canvas.addGLEventListener(new Main());
canvas.setFocusable(true);
canvas.requestFocus();
//add the canvas to the frame
frame.add(canvas);
frame.setUndecorated(true); //hide window border
frame.setSize(1024, 768);
frame.setLocationRelativeTo(null);
frame.setCursor(pointer);
frame.setVisible(true);

if(fullscreen){
    GraphicsEnvironment ge=GraphicsEnvironment.getLocalGraphicsEnvironment();
    ge.getDefaultScreenDevice().setFullScreenWindow(frame);
}

//setup animator and run as fast as possible
final Animator animator = new Animator(canvas);
animator.setRunAsFastAsPossible(true);
animator.start();

/*
 * find the coordinates that define the center of the player window.
 * Bear in mind that the window itself might be located in only part
 * of the screen
 */
Rectangle r=frame.getBounds();
center=new Point(r.x+r.width/2, r.y+r.height/2);
}

/*
 * initialize the GL context handle events
 */
public void init(GLAutoDrawable drawable) {
    GL gl = drawable.getGL();
    game=new Game(gl);
    game.setMouseCenter(center);
    mousePos=center;
    GLU glu=new GLU();

    gl.glViewport(0, 0, w, h);
    gl.glMatrixMode(gl.GL_PROJECTION);
    gl.gluPerspective(45.0f, ((float)w/((float)h), 0.1f, 100.0f);
    gl.glMatrixMode(gl.GL_MODELVIEW);
    gl.glShadeModel(gl.GL_SMOOTH);
    gl.glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
    gl.glClearDepth(1.0f);
    gl.glEnable(gl.GL_DEPTH_TEST);
    gl.glDepthFunc(gl.GL_LEQUAL);
    gl.glHint(gl.GL_PERSPECTIVE_CORRECTION_HINT, gl.GL_NICEST);

    /*
     * Events are going to chew up CPU cycles whatever you do (a key held
     * down will continuously fire events. What we
     * don't want is these events continually interrupting the game loop.
     * So what we do is use these events to continually set boolean values
     * in the Game class. The loop will poll these variables once per loop
     * to see whether a key is down, etc.
     */
    drawable.addKeyListener(new KeyAdapter(){
        @Override
        public void keyPressed(KeyEvent e){
            if(e.getKeyCode()==KeyEvent.VK_ESCAPE){

```

```

        new Thread(new Runnable(){
            public void run(){
                System.exit(0);
            }
        }).start();
    }
    /*
     * check if the key pressed value is the same as the value set
     * for forward, backward etc.
     */

    if(e.getKeyCode()==forward)game.setFord(true);
    if(e.getKeyCode()==backward)game.setback(true);
    if(e.getKeyCode()==strafel)game.setstrafel(true);
    if(e.getKeyCode()==strafer)game.setstrafer(true);
}
@Override
public void keyReleased(KeyEvent e){
    if(e.getKeyCode()==forward)game.setFord(false);
    if(e.getKeyCode()==backward)game.setback(false);
    if(e.getKeyCode()==strafel)game.setstrafel(false);
    if(e.getKeyCode()==strafer)game.setstrafer(false);
}
});
drawable.addMouseListener(new MouseAdapter(){
    @Override
    public void mouseClicked(MouseEvent e) {
        if((e.getModifiers() & shoot)!=0)
            game.setShoot();
        if((e.getModifiers() & use)!=0)
            game.setUse();
    }
});
}

/*
 * display is called by the animator object; it implements the game loop
 */
public void display(GLAutoDrawable drawable) {
    GL gl = drawable.getGL();

    gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
    gl.glLoadIdentity();

    game.tick();

    gl.glFlush();
}

public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) {
    GL gl = drawable.getGL();
    GLU glu = new GLU();

    if (height <= 0) { height = 1; }
    final float h=(float)width/(float)height;
    gl.glViewport(0, 0, width, height);
    gl.glMatrixMode(GL.GL_PROJECTION);
    gl.glLoadIdentity();
    glu.gluPerspective(45.0f, h, 1.0, 20.0);
    gl.glMatrixMode(GL.GL_MODELVIEW);
    gl.glLoadIdentity();

    center=new Point(x+width/2, y+height/2);
}

```

```

        game.setMouseCenter(center);
    }

    public void displayChanged(GLAutoDrawable drawable, boolean modeChanged, boolean
deviceChanged) {}
}

//~~~~~|Game.java|~~~~~\

package myGame;

import java.awt.AWTException;
import java.awt.MouseInfo;
import javax.media.opengl.GL;
import java.awt.Point;
import java.awt.Robot;
/**
 * As well as drawing the standard triangle and square, Main polls the events
 * set by the Main class and prints if the event has been fired.
 *
 * @author Stephen Jones
 */
public class Game {
    private GL gl;
    private Point mouseCenter;//handed from Main
    private Robot robot;//used to center the cursor at mouseCenter
    private float dx, dy;//change in mouse x/y
    /*
     * boolean values set by Main and reflect whether the key/mouse
     * is causing an event
     */
    private boolean ford=false;
    private boolean back=false;
    private boolean strafel=false;
    private boolean strafer=false;
    private int shoot=0;
    private int use=0;

    public Game(GL gl){
        this.gl=gl;
        /*
         * Robot is a Java 6 feature that can control devices such as keyboard and
mice.
         * Here we use it to reposition the cursor.
         */
        Robot r=null;
        try{
            r=new Robot();
        }catch(final AWTException e){System.out.println("Trouble stating Robot");}
        this.robot=r;
        if(robot==null)System.out.println("Error robot has not been initialized");
    }

    /*
     * tick is called from the render method in main and is the start of
     * each new game loop. The first thing we do is see what the player is up to
     * by polling the events variables. We then recenter the cursor. The shoot
     * and use variables also need to be reset. The remainder of the code is
     * the standard triangle and square
     */
    public void tick(){
        pollEvents();
        robot.mouseMove(mouseCenter.x, mouseCenter.y);
        shoot=0;
    }
}

```

```

use=0;

// Move the "drawing cursor" around
gl.glTranslatef(-1.5f, 0.0f, -6.0f);

// Drawing Using Triangles
gl.glBegin(GL.GL_TRIANGLES);
    gl.glColor3f(1.0f, 0.0f, 0.0f);
    gl.glVertex3f(0.0f, 1.0f, 0.0f);
    gl.glColor3f(0.0f, 1.0f, 0.0f);
    gl.glVertex3f(-1.0f, -1.0f, 0.0f);
    gl.glColor3f(0.0f, 0.0f, 1.0f);
    gl.glVertex3f(1.0f, -1.0f, 0.0f);
gl.glEnd();

// Move the "drawing cursor" to another position
gl.glTranslatef(3.0f, 0.0f, 0.0f);
// Draw A Quad
gl.glBegin(GL.GL_QUADS);
    gl.glColor3f(0.5f, 0.5f, 1.0f);
    gl.glVertex3f(-1.0f, 1.0f, 0.0f);
    gl.glVertex3f(1.0f, 1.0f, 0.0f);
    gl.glVertex3f(1.0f, -1.0f, 0.0f);
    gl.glVertex3f(-1.0f, -1.0f, 0.0f);
gl.glEnd();
}

/*
 * There is a season, turn, turn turn... there is a time for every something
 * under heaven, whatever... The Seekers. And there is a time within each
 * game loop when we need to see what the user wants.
 */
private void pollEvents(){
    /*
     * First we check to see if the player has moved the mouse. We get the
     * x and y positions of the mouse and check it against the mouseCenter
variables.
     * A difference in x means the player has changed direction or heading. A
     * difference in y translates into the player looking up or down, that is,
     * a change in pitch.
     */
    dx=MouseInfo.getPointerInfo().getLocation().x;
    dy=MouseInfo.getPointerInfo().getLocation().y;

    if((mouseCenter.x-dx)!=0)
        System.out.println("Changing heading " + (mouseCenter.x-dx));
    if((mouseCenter.y-dy)!=0)
        System.out.println("Changing pitch " + (mouseCenter.y-dy));
    /*
     * Check the keyboard and mouse button effected booleans to see if
     * the player had them pressed during the last operating system event cycle.
     * On keyRelease (look at the Main class) these variables are set to false.
     */
    if(ford)
        System.out.println("Moving forward");
    if(back)
        System.out.println("Moving backward");
    if(strafel)
        System.out.println("Strafing left");
    if(strafer)
        System.out.println("Strafing right");
    if(shoot>0)
        System.out.println("Pleyer has shot");
    if(use>0)

```

```

        System.out.println("Pleyer has used");
    }

    /*
     * These are the setter methods called by the event handlers in the Main class.
     * Although none of the booleans in this class are declared volatile, my
     * understanding is that Java will not allow different threads to crap all
     * over another threads attempt to write to single variables. If I am
     * wrong in this, I am not sure it matters anyway, for if a boolean is not
     * true then any other rubbish that may get planted in it will define it as false.
     */
    public void setMouseCenter(Point center){
        this.mouseCenter=center;
    }
    public void setFord(boolean flag){
        ford=flag;
    }
    public void setback(boolean flag){
        back=flag;
    }
    public void setstrafel(boolean flag){
        strafel=flag;
    }
    public void setstrafer(boolean flag){
        strafer=flag;
    }
    /*
     * Mouse button events increment the shoot and use variables. pollEvents
     * checks to see if they are >0 then 0 then ready for the next loop.
     */
    public void setShoot(){
        shoot++;
    }
    public void setUse(){
        use++;
    }
}

```