

User Interface - Events

Computer Games Development

CSCU9N6

Introduction

Event Handling

- Keyboard
- Mouse

GameCore Class

- An abstract class for a game environment

Demos

Input Events & Java

Input

- Keyboard
- Mouse
- As you should already be aware, Java manages user input via event handlers. An event handler registers for the events it is interested in and then receives notification of the event occurring.
- Joystick or Controller input would also be useful but will not be covered here. It should however work on the same principle.
- This event model is also followed in other GUI environments such as the Android or iOS platforms.

Event Listeners

We need event listeners for each type of event that could be of interest to us:

KeyListener

- Listen for events generated by the keyboard

MouseListener

- General mouse events (type of button click etc)
- Generates a *MouseEvent*

MouseMotionListener

- Mouse movement events
- Generates a *MouseEvent*

MouseWheelListener

- Mouse wheel movements
- Generates a *MouseWheelEvent*

Events : Adding Listeners

In the initialisation code, add listeners as follows:

```
Window window = screen.getFullScreenWindow();  
window.addKeyListener(this);  
window.addMouseListener(this);  
window.addMouseMotionListener(this);  
window.addMouseWheelListener(this);
```

Keyboard Event Handlers

KeyListener Interface

- ```
public void keyPressed(KeyEvent e)
{
 if (e.getKeyCode() == KeyEvent.VK_A) ...
 if (e.getKeyCode() == KeyEvent.VK_B) ...
 if (e.getKeyCode() == KeyEvent.VK_C) ...
 if (e.getKeyCode() == KeyEvent.VK_ESCAPE) ...
}
```
- ```
public void keyReleased(KeyEvent e) { ... same idea as above... }
```
- ```
public void keyTyped(KeyEvent e) { ... same idea as above... }
```

# Mouse Event Handlers

## MouseListener Interface

- `public void mousePressed(MouseEvent e) { }`
- `public void mouseReleased(MouseEvent e) { }`
- `public void mouseClicked(MouseEvent e) { x=e.getX(); y=e.getY(); }`
- `public void mouseEntered(MouseEvent e) { }`
- `public void mouseExited(MouseEvent e) { }`

## MouseMotionListener Interface

- `public void mouseDragged(MouseEvent e) { x=e.getX(); y=e.getY(); }`
- `public void mouseMoved(MouseEvent e) { }`

## MouseWheelListener interface

- `public void mouseWheelMoved(MouseWheelEvent e)`  
    {  
        wheelposition = e.getWheelRotation(); // wheel clicks since last time  
    }

# GameCore Class

You will probably notice that we keep building the same basic code, then add bits to it. This is a lot of repetitive work and therefore a candidate for using inheritance. We will therefore use a derivation of the *GameCore* class provided in DGJ.

## GameCore

- An abstract class with similar elements as before
- An `init` method which sets up the display using the `ScreenManager` class, with options for full or windowed mode
- A `gameLoop` method (similar to the animation loop)
- An `update` method which we can override
- A `draw` method, which we must override
- A `stop` method to stop the game loop
- A useful `loadImage` method to load images



# Demo

## EventTest

- Moving Sprite
- GameCore
- EventTest