CT255 NGT2
Digital Media/ 2D Games

Week 2

[2D Games in Java]

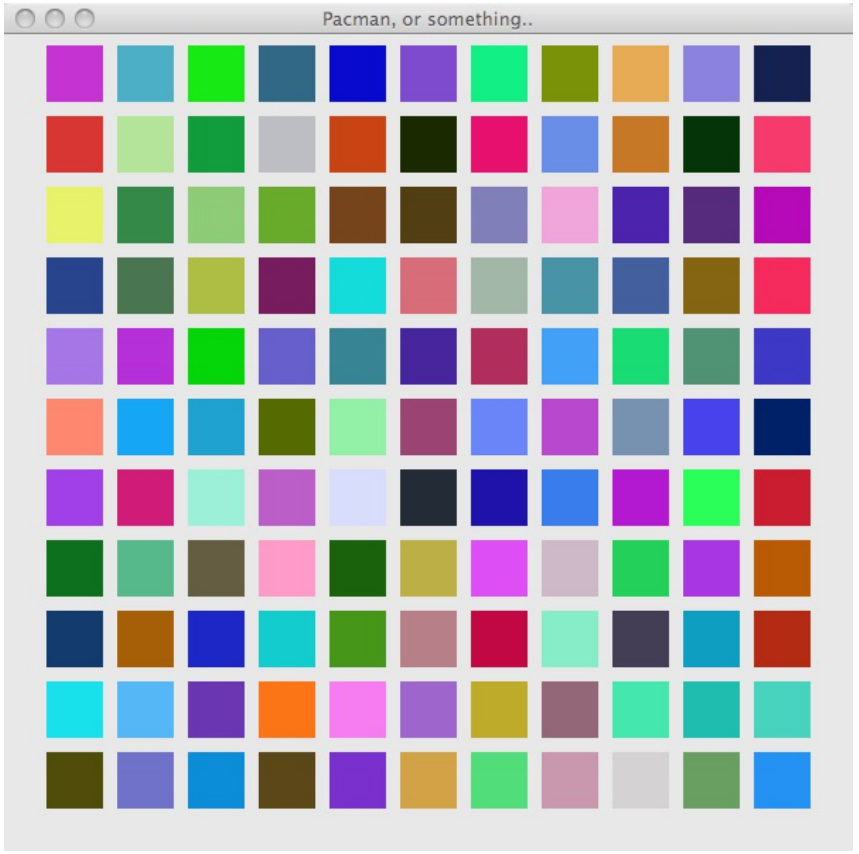
sam.redfern@universityofgalway.ie
@psychicsoftware

https://discord.gg/uWem2rQg7a



Week 1 Assignment

- Write a JFrame-based program that fills its window with randomly coloured squares



Hints:

- Use nested loops to produce the squares
- Think about your co-ordinate system (x/y)
- Investigate the `fillRect()` method of the Graphics class
- Investigate how to specify an arbitrary Color rather than using a stock Color

To get a random integer between 0 and 255: **int red = (int)(Math.*random()*255);**

Topics this week

- Animation with threads
- Creating a simple game object class
- Making a game which animates an array of game object instances

Animation with Threads

- Animation is the changing of graphics over time
- E.g. moving a spaceship across the screen, changing its position by 1 pixel every 0.02 seconds
- One of the best ways to do periodic execution of code is to use **threads**
- Threads: allow multiple tasks to run independently/concurrently within a program
- Essentially, this means we spawn a separate execution 'branch' that operates independently of our program's main flow of control
- The new Thread repeatedly sleeps for (say) 20ms, then carries out animation, and calls this.repaint() on the application

Implementing Threads in Java

- Your application class should implement the Runnable interface, i.e.:

```
public class MyApplication extends JFrame implements Runnable {
}
```

- Your class **must** now provide an implementation for the run() method, which is executed when a thread is started, and serves as its “main” function i.e.:

```
public void run() {
}
```

- To create and start a new thread running from your application class:

```
Thread t = new Thread(this);
t.start();
```

Typical Actions of an Animation Thread

1. Sleep for (say) 20ms using **Thread.sleep(20);**
   - **Note that you will be required to handle** InterruptedException
2. Carry out movement of game objects
3. Call **this.repaint();** which (indirectly) invokes our **paint(Graphics g)** method
4. Go back to step 1

Threads Test

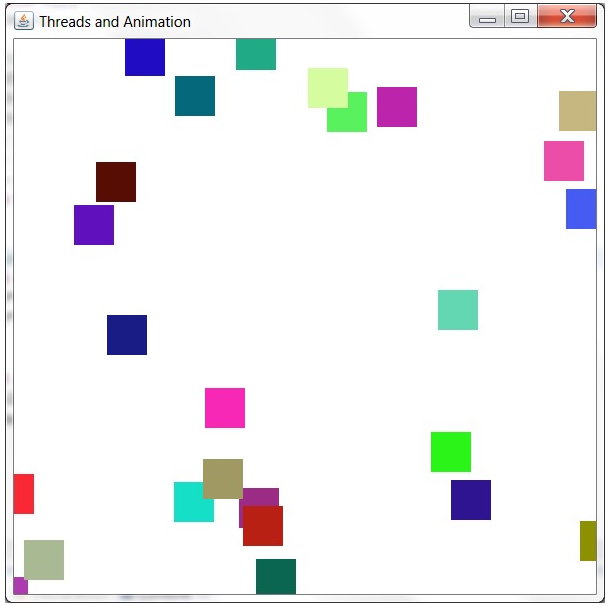
```
2  import java.awt.*;
3  import javax.swing.*;
4
5  public class ThreadsTest implements Runnable {
6
7      public ThreadsTest() {
8          Thread t = new Thread(this);
9          t.start();
10     }
11
12     public void run() {
13         System.out.println("Thread started");
14
15         for (int i=0; i<15; i++) {
16             System.out.println("Loop "+i+" start");
17             try {
18                 Thread.sleep(500);
19             } catch (InterruptedException e) {
20                 // TODO Auto-generated catch block
21                 e.printStackTrace();
22             }
23             System.out.println("Loop "+i+" end");
24         }
25
26         System.out.println("Thread ended");
27     }
28
29     public static void main(String[] args) {
30         ThreadsTest tt = new ThreadsTest();
31
32     }
33
34 }
35
```

Game object classes

- Games typically have game object classes (spaceships, aliens, cars, bullets etc.), numerous instances of each may exist at runtime
- This class encapsulates the data (position, colour etc.) and code (move, draw, die, etc.) associated with the game object
- Typically we store these instances in a data structure such as an array, so that during our animation and painting phases, we can iterate through them all and invoke the animate() and paint() methods on each instance

Week #2 Assignment

- Create a program which performs simple random animation of coloured squares
- Use two classes:
  1. MovingSquaresApplication
     - extends JFrame
     - Implements Runnable
     - has main() method
     - Member data includes an array of GameObject instances
     - Constructor method does similar setup as last week's code, and in addition instantiates the GameObjects in the array, and creates+starts a Thread
     - Uses a Thread to perform animation of the GameObjects by calling their move() methods
     - Paint() method draws the GameObjects by calling their paint(Graphics g) methods
  2. GameObject
     - Member data includes x,y,color
     - Constructor method randomises the object's position and color
     - Public move() method is used to randomly alter x,y members
     - Public paint(Graphics g) method draws the object as a square using g.fillRect()



Code should be uploaded on Blackboard (deadline: see Blackboard)

Assignment #2
Suggested Class Interfaces

*MovingSquaresApplication.java

```
import java.awt.*;
import javax.swing.*;

public class MovingSquaresApplication extends JFrame implements Runnable {

    // member data
    private static final Dimension WindowSize = new Dimension(600,600);
    private static final int NUMGAMEOBJECTS = 30;
    private GameObject[] GameObjectsArray = new GameObject[NUMGAMEOBJECTS];

    // constructor
    public MovingSquaresApplication() {

    // thread's entry point
    public void run() {

    // application's paint method
    public void paint(Graphics g) {

    // application's entry point
    public static void main(String[] args) {

}
```

GameObject.java

```
import java.awt.*;

public class GameObject {

    // member data
    private double x,y;
    private Color c;

    // constructor
    public GameObject() {

    // public interface
    public void move() {

    public void paint(Graphics g) {

}
```