CT255 NGT2
Digital Media

[2D Games in Java]

Dr Sam Redfern
sam.redfern@nuigalway.ie
@psychicsoftware

2D Games Programming in Java

- As a direct support for CT2106 and CT2109
- Problem-Based approach
- 'Just in Time' focus
- Emphasis on researching features of Java yourselves (under my direction)
- Various useful topics related to input/output, graphical display, realtime update, etc.
- Even some Artificial Intelligence! (A.I.)

Labs & Lectures etc.

- Lecture/Discussion:
  - Mondays 12-1pm, AC213
- Workshop/Practical Work:
  - Mondays 2-4pm, Lab IT101 (CS Building)
  - Some weeks we'll do less than an hour in the lecture/discussion, plus 2 hrs in the lab
- Course notes:
  - Blackboard

Discord

- I have created a Discord server for this module
- Please use this to make comments, ask questions, share resources etc.
- Using this approach, I can generally answer questions fairly quickly anytime during the semester, not just during class/lab times

- Use this Discord link to join the server:
  - https://discord.gg/uWem2rQg7a

Grading

- Weekly assignments will account for 25% of <u>my part</u> of the CT255 course (M. Schukat did the other half of CT255 in sem. 1)
- Assignments are always due at the start of the subsequent lecture (where you will be given a sample solution and we'll spend some time discussing it)
- There will also be questions in the Summer exam paper, accounting for 75% of <u>my part</u> of the CT255 course

Developing Games in Java

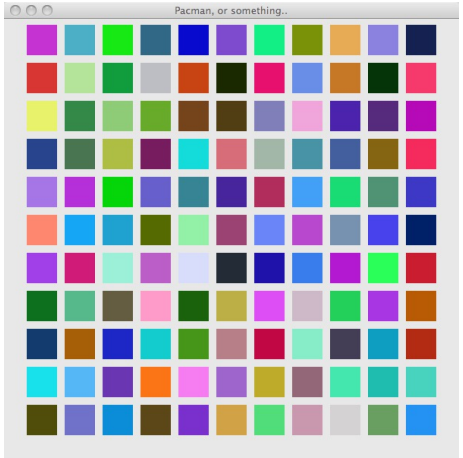
David Brackeen

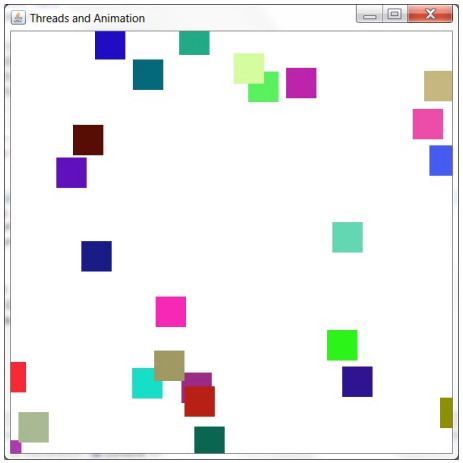
New Riders

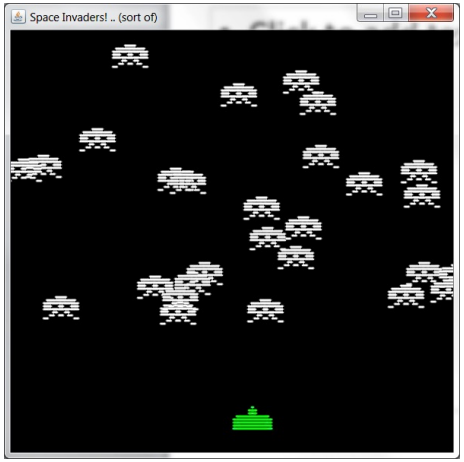
NRG

Topics will include

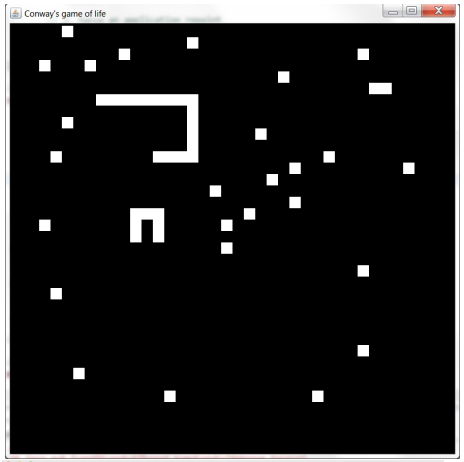
- Graphics with the JFrame class
- Raster & vector graphics methods of the Graphics class
- Graphics contexts, double buffering
- 2D ‘sprite’ animation
- Constructing game object classes
- Collision detection
- Multi-threading
- Keyboard and mouse input
- Game states using 2D arrays, hash tables and other collection classes
- Mazes and A.I. pathfinding using the A* algorithm


Pacman, or something..

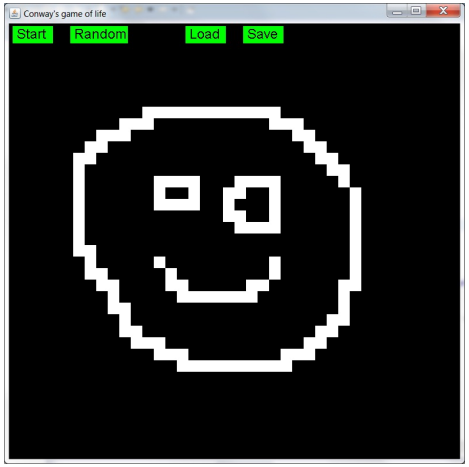

Threads and Animation

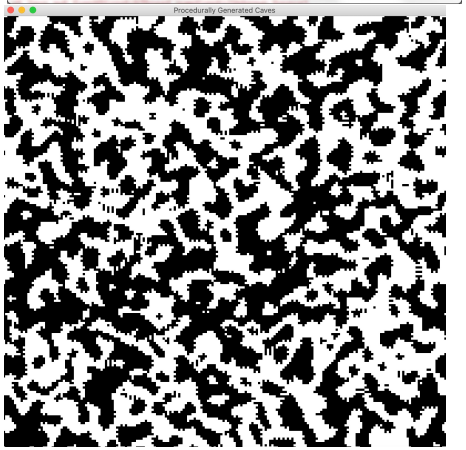

Space Invaders! .. (sort of)

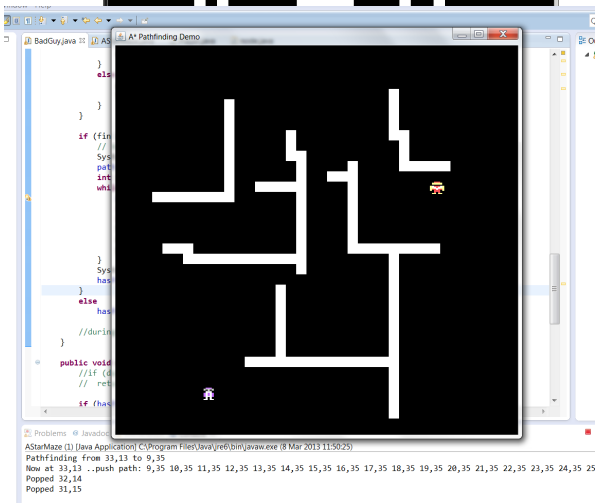

Space Invaders!
Score: 120 Best: 1040


Conway's game of life


Conway's game of life
Start
Random


Conway's game of life
Start
Random
Load
Save


Procedurally Generated Caves

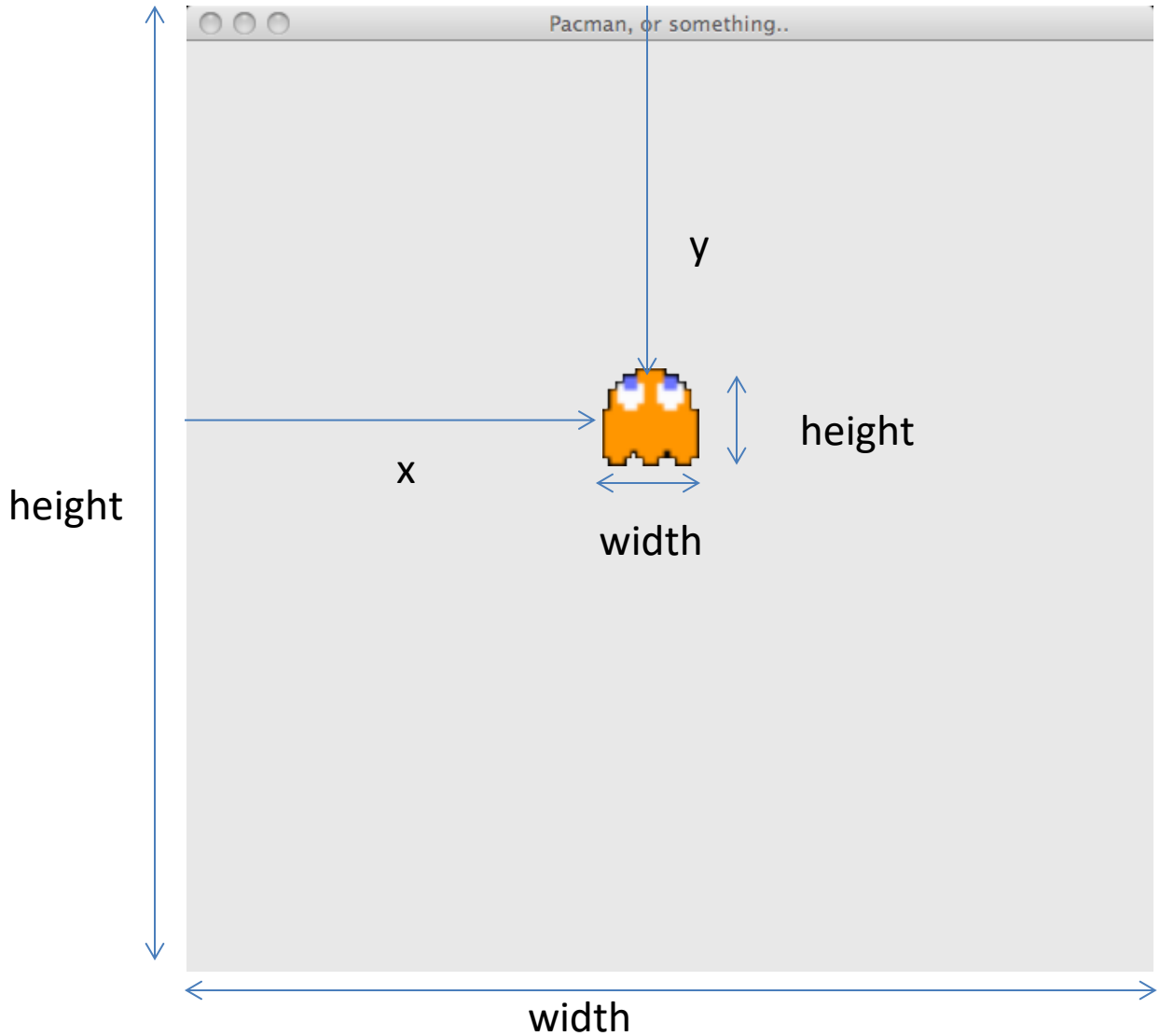

A* Pathfinding Demo
BadGuy.java
AStarMaze (1) [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (8 Mar 2013 11:50:25)
Pathfinding from 33,13 to 9,35
Now at 33,13 ..push path: 9,35 10,35 11,35 12,35 13,35 14,35 15,35 16,35 17,35 18,35 19,35 20,35 21,35 22,35 23,35 24,35 25
Popped 32,14
Popped 31,15

2D Co-ordinate System

The JFrame class will provide a Window with associated graphics canvas, and a pixel-based coordinate system

Origin (0,0) at top-left

Top 50 pixels or so are hidden by the window's title bar (depends on Operating System)



Creating a Window-based Application

- Create a new Java project in Eclipse (or other IDE)
- Right-click the project
- New > class
- Name your class, e.g. ‘MyApplication’

- In Java, you need to have a method named **main** in at least one class.

- A JFrame is a top-level window with a title and a border. To have access to JFrame and associated methods:

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

A Minimal JFrame-based app.

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

public class MyApplication extends JFrame {

    private static final Dimension WindowSize = new Dimension(600,600);

    public MyApplication() {
        //Create and set up the window.
        this.setTitle("Pacman, or something..");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //Display the window, centred on the screen
        Dimension screensize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        int x = screensize.width/2 - WindowSize.width/2;
        int y = screensize.height/2 - WindowSize.height/2;
        setBounds(x, y, WindowSize.width, WindowSize.height);
        setVisible(true);
    }


    public static void main(String [ ] args) {
      MyApplication w = new MyApplication();
    }

}
```

A note about Eclipse:

- It will attempt to suggest code fixes such as creating new methods for you
- Be careful! A method call from your base class that can't be compiled often means you have mistyped or forgotten an import, or forgotten to extend from a base class
- So don't let it trick you into creating an empty method in its place!

Basic graphics in Java

- 2D graphics can be drawn using the `Graphics` class
- This provides methods for drawing 'primitives' (lines, circles, boxes), also images (.jpg, .png, etc.)

- The `paint()` method of the `JFrame` class is automatically invoked whenever it needs to be painted (system-invoked)
- Or you can force it to happen via `repaint()` if you need to repaint when the OS doesn't think it's needed:

```
public void paint ( Graphics g ) {
        // use the 'g' object to draw graphics
}
```

Drawing text using methods of the Graphics class

setColor(Color c)

setFont(Font font)

drawString(String str, int x, int y)

```
public void paint ( Graphics g ) {
    Font f = new Font( "Times", Font.PLAIN, 24 );
    g.setFont(f);
    Color c = Color.BLACK;
    g.setColor(c);
    g.drawString("Pacman!", 20, 60);
}
```

- This should be added as a method of the MyApplication class
- Note the usefulness of the context-help in Eclipse – tells you method names, their parameters and a description of them

The graphics class has lots of useful methods!

```
clearRect(int x, int y, int width, int height) : void - Graphics
clipRect(int x, int y, int width, int height) : void - Graphics
copyArea(int x, int y, int width, int height, int dx, int dy) : void - Graphics
create() : Graphics - Graphics
create(int x, int y, int width, int height) : Graphics - Graphics
dispose() : void - Graphics
draw3DRect(int x, int y, int width, int height, boolean raised) : void - Graphics
drawArc(int x, int y, int width, int height, int startAngle, int arcAngle) : void - Graphics
drawBytes(byte[] data, int offset, int length, int x, int y) : void - Graphics
drawChars(char[] data, int offset, int length, int x, int y) : void - Graphics
drawImage(Image img, int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, ImageObserver observer) : boolean - Graphics
drawImage(Image img, int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, Color bgcolor, ImageObserver observer) : boolean - Grap
drawImage(Image img, int x, int y, ImageObserver observer) : boolean - Graphics
drawImage(Image img, int x, int y, Color bgcolor, ImageObserver observer) : boolean - Graphics
drawImage(Image img, int x, int y, int width, int height, ImageObserver observer) : boolean - Graphics
drawImage(Image img, int x, int y, int width, int height, Color bgcolor, ImageObserver observer) : boolean - Graphics
drawLine(int x1, int y1, int x2, int y2) : void - Graphics
drawOval(int x, int y, int width, int height) : void - Graphics
drawPolygon(Polygon p) : void - Graphics
drawPolygon(int[] xPoints, int[] yPoints, int nPoints) : void - Graphics
drawPolyline(int[] xPoints, int[] yPoints, int nPoints) : void - Graphics
drawRect(int x, int y, int width, int height) : void - Graphics
drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight) : void - Graphics
drawString(AttributedCharacterIterator iterator, int x, int y) : void - Graphics
drawString(String str, int x, int y) : void - Graphics
equals(Object obj) : boolean - Object
fill3DRect(int x, int y, int width, int height, boolean raised) : void - Graphics
fillArc(int x, int y, int width, int height, int startAngle, int arcAngle) : void - Graphics
fillOval(int x, int y, int width, int height) : void - Graphics
fillPolygon(Polygon p) : void - Graphics
fillPolygon(int[] xPoints, int[] yPoints, int nPoints) : void - Graphics
fillRect(int x, int y, int width, int height) : void - Graphics
fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight) : void - Graphics
finalize() : void - Graphics
getClass() : Class<? extends Object> - Object
getClip() : Shape - Graphics
getClipBounds() : Rectangle - Graphics
```

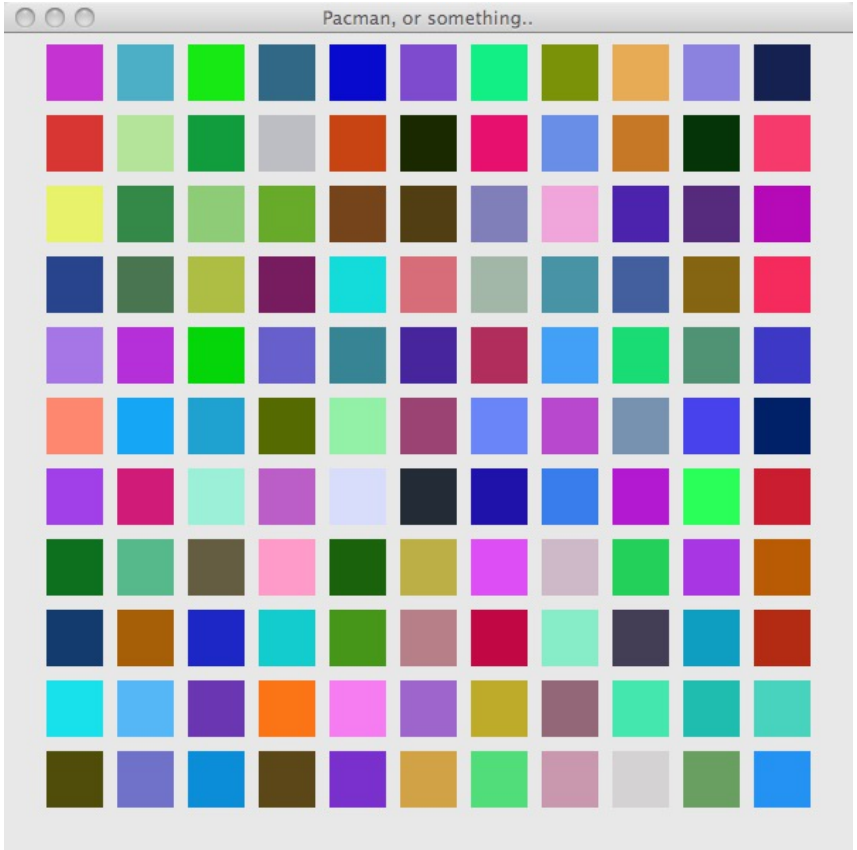
Press '^Space' to show Template Propos

Week 1 Assignment

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

<u>Hints:</u>

- Use nested loops to produce the squares
- Think about your co-ordinate system (x/y).. what are the min/max values of these you want to stay between?
- 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()*256);**

Your assignment code should be submitted via Blackboard.