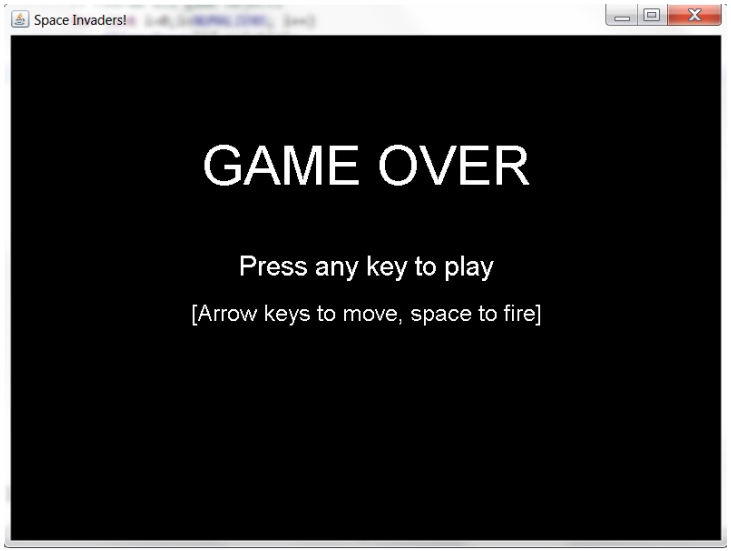
CT255 / NGT2

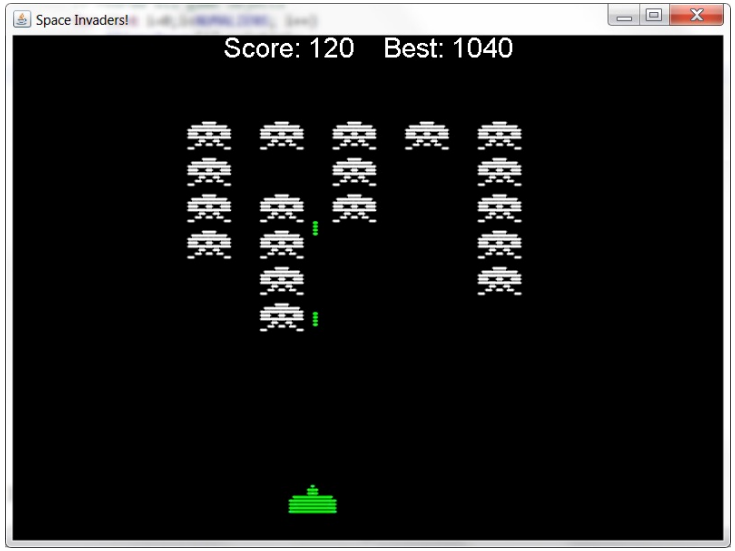
Week 6
[2D Games in Java]

Dr Sam Redfern
sam.redfern@universityofgalway.ie

Last week's assignment: Finishing off the Invaders game

- Add game states (in progress or in menus): you will need to modify various methods of the application class based on the state it's in
- Add scoring
- Add collision detection between aliens and the player spaceship (game over when this happens => switch back to menus state)
- When all aliens are killed, re-create a new, faster-moving wave of them
- I would suggest you carefully assess where certain code is running (e.g. setting the initial positions of aliens and player ship) – the constructor is no longer the best place for this – you will need to create new methods such as startNewWave() and startNewGame()





Conway's Game of Life..

- http://en.wikipedia.org/wiki/Conway's_Game_of_Life
- A famous 'cellular automata' 0-player game from 1970
- Each cell follows a simple set of rules as the game progresses, and a (relatively) complex system behaviour emerges
- Gives us the opportunity to study:
  - Handling the mouse in Java
  - 2D arrays
  - Reading/writing files
- And will also form the basis of two further projects:
  - Making procedural 'cave-like' maps with cellular automata
  - Solving mazes /pathfinding using the A* algorithm

Mouse Events

- Mouse events notify when the user uses the mouse (or similar input device) to interact with a component. Mouse events occur when the pointer enters or exits a component's onscreen area and when the user presses or releases one of the mouse buttons.
- Additional events such as mouse movement, and the mouse wheel, can be handled by implementing the MouseMotionListener and MouseWheelListener interfaces
- Step 1: have your class implement MouseListener
- Step2: In the class constructor: addMouseListener(**this);**
- Step 3: implement the methods below

```
// mouse events which must be implemented for MouseListener
    public void mousePressed(MouseEvent e) { }

     public void mouseReleased(MouseEvent e) { }

     public void mouseEntered(MouseEvent e) { }

     public void mouseExited(MouseEvent e) { }

     public void mouseClicked(MouseEvent e) { }
```

Methods of the MouseEvent class

- int getClickCount()
  - Returns the number of quick, consecutive clicks the user has made (including this event). For example, returns 2 for a double click.
- int getX()
- int getY()
- Point getPoint()
  - Returns the (x,y) position at which the event occurred, relative to the component that fired the event.
- int getButton()
  - Returns which mouse button, if any, has a changed state. One of the following constants is returned: NOBUTTON, BUTTON1, BUTTON2, or BUTTON3.

This week's assignment
Starting the Game of Life

- Create a new Java project, with a main application class that extends JFrame and implements Runnable and MouseListener
- The window should be 800x800 pixels in size
- Use double buffering to avoid flicker when we animate [next week]
- Implement periodic repainting (and later, animation) via a Thread
- Create a 2 dimensional array to store the game state: e.g. a 40x40 array of Booleans, assuming that we want each cell to be 20x20 pixels
- When the mouse clicks on the window, toggle the state of the game state cell at that position (i.e. true becomes false, and false becomes true)
- The paint method should paint, as a rectangle, each game state cell that is currently 'true'