240 Part 2

Last Year

- Random Numbers
- Computational Errors
- Monte Carlo Simulation
- 3 Lectures on Agent-based Modelling

Random Numbers

- Intro on Simulation - *"a computer program built to model a particular system so that experiments can be carried out on a system virtually, and not on the actual system."*
- Programming simulations – model first, then program
- Why we need randomness
- Generating RNGs

Random Numbers

- Linear Congruential Generator (LCG)
  - Simple

    ```
    r = (a * previous_rand + c) % m;
    ```
  - a, c and m are constant values that don't change
- LCG is bad because:
  - It wraps around quickly (small period)
  - Bits aren't equally random – low order bits are less random

Random Numbers

- Testing the quality of an RNG
  - Don't look at the RNG, look at the sequences that it makes
  - They should have properties:
    - Uniformity
    - Independence
    - Summation
    - Duplication
  - Not easy to test these – test suites like NIST and “Diehard”

Random Numbers

- Pays to know how to read ranges
  - (0,10) open interval
  - [0,10] closed interval
  - (5,12]
  - [-3,4)

Random Numbers

- Distributions
  - Given a distribution graph, aim to be able to tell which one it is
  - Exponential
  - Uniform
  - Normal (“Gaussian”)
  - Levy
- What transform do you use to generate normal random deviates from uniform random deviates?

Computational Errors

- Important because, if we don't get it right, a simulation might say that a nuclear reactor with 1 control rod will be totally safe

Computational Errors

- Rounding Errors (4/3 = 1.3333333..)
- Meaningless Significant figures
- Conversion errors (double → float)
- Human errors
- Formula errors (Can't correctly evaluate a formula with an infinite number of terms)
- Propagation errors (measurements with errors)

Computational Errors

- Subtractive Cancellation
  - Adding a small float to a huge float
  - Makes no difference... Sort them with small numbers first then add in that order

Computational Errors

- Floating point oddities
  - if (tot == 200.0)
  - infinity and not-a-number
  - float a = 1.0f / 0.0f; // gives positive infinity
  - float a = log(0); // gives negative infinity
  - float a = 0.0f / 0.0f; // gives -nan
- Testing for NaN:
  - if (myfloat != myfloat)
    // is true only when myfloat = nan or -nan
  - if (f == NAN) // does not work

Structs and stack/heap memory

```
struct Person {
    int age;
    char * name;
    char gender;
};
```

- Person guy;
  - guy.age = 40;
- Person *guy = new Person;
  - guy->age = 21;

Agent-based Modelling

- Linear Algebra with Vectors
- An agent-based model simulation has many individual agents, each of which follows a set of behavioural rules
- Time passes in turns or time steps
- Not graphics – behaviour
- Agent-based models consist of agents, an environment, and interaction rules/behaviours

Agent-based Modelling

- An agent can be a person, car, insect, robot, businesses, countries, etc.
- Agents in a model can be heterogeneous meaning they can be different (behaviour or type)
- Agent-based models don't have a central source of control – decentralised control is much more interesting (and perhaps more useful)

Agent-based Modelling

- What is an agent based model?
  - A simulation with many individual agents that are **autonomous**, **interactive** and **situated**.
- Macro-behaviour & Micro-behaviour
  - System-level & individual-level

Agent-based Modelling

- Boids
  - Cohesion, Separation, Alignment
  - Check back over slides for these rules