COMP350: Algorithms & Optimisation

1: Module Intro & The Optimisation Process

Learning outcomes

By the end of today's session, you will be able to:

- **Understand** the module aim
- **Explain** the optimisation process
- **Utilise** the profiling tools

FALMOUTH
UNIVERSITY

Module Introduction

Module Aims

- Gain in understanding of techniques used professionally in the management of computing resources.
- Acquire knowledge and experience of concepts used to predict and model resource use.
- Acquire the knowledge and experience to enable critical evaluation of trade-offs to generate optimisation and efficiency.

FALMOUTH
UNIVERSITY

Assignment Details

Assignment Overview

- Optimisation Task - 50%
- Porting Task - 30%
- Research Journal - 20%

Assignment 1 - Optimisation Task

- Take an existing project and optimise
- You have to identify the tools required for optimising
- I am more interested in your **process** during the task
- First Submission - **Friday 9th of February at 5pm**
- `https://github.com/Falmouth-Games-Academy/bsc-assignment-briefs/raw/2017-18/comp350/1/comp350_1.pdf`

Assignment 2 - Porting

- Continue on with the project from Assignment 1
- Port your project to one of the following Platforms - PS4, Android, iOS
- You will have to fulfil some of the Technical Requirement for that platform
- `https://github.com/Falmouth-Games-Academy/bsc-assignment-briefs/raw/2017-18/comp350/2/comp350_2.pdf`

Assignment 3 - Research Journal

- Write a 1200 word research journal on optimisation & porting
- Contribute to a community Wiki
- `https://github.com/Falmouth-Games-Academy/bsc-assignment-briefs/raw/2017-18/comp350/3/comp350_3.pdf`

FALMOUTH
UNIVERSITY

Optimisation

Optimiser Mantra

Optimiser Mantra

1. Benchmark

Optimiser Mantra

1. Benchmark
2. Measure

Optimiser Mantra

1. Benchmark
2. Measure
3. Detect

Optimiser Mantra

1. Benchmark
2. Measure
3. Detect
4. Solve

Optimiser Mantra

1. Benchmark
2. Measure
3. Detect
4. Solve
5. Check

Optimiser Mantra

1. Benchmark
2. Measure
3. Detect
4. Solve
5. Check
6. Repeat

Benchmark

Benchmark

- This is a point of reference for your game, serves as a standard for comparison

Benchmark

- This is a point of reference for your game, serves as a standard for comparison
- A good benchmark should:

Benchmark

- This is a point of reference for your game, serves as a standard for comparison
- A good benchmark should:
  1. Consistent between runs

Benchmark

- This is a point of reference for your game, serves as a standard for comparison
- A good benchmark should:
  1. Consistent between runs
  2. Should be quick

Benchmark

- This is a point of reference for your game, serves as a standard for comparison
- A good benchmark should:
  1. Consistent between runs
  2. Should be quick
  3. Represent an actual game situation

Benchmark

- This is a point of reference for your game, serves as a standard for comparison
- A good benchmark should:
  1. Consistent between runs
  2. Should be quick
  3. Represent an actual game situation
  4. Responsive to changes

Measure

Measure

- You should be able to measures the performance of your code

Measure

- You should be able to measures the performance of your code
- Tools like Profilers allow you to monitor the following:

Measure

- You should be able to measures the performance of your code
- Tools like Profilers allow you to monitor the following:
  - CPU Usage - Across all cores, usually % utilisation

Measure

- You should be able to measures the performance of your code
- Tools like Profilers allow you to monitor the following:
  - CPU Usage - Across all cores, usually % utilisation
  - Memory Usage - Ram, Stack, Heap etc

Measure

- You should be able to measures the performance of your code
- Tools like Profilers allow you to monitor the following:
  - CPU Usage - Across all cores, usually % utilisation
  - Memory Usage - Ram, Stack, Heap etc
  - GPU - GPU core & memory usage and shader performance

Measure

- You should be able to measures the performance of your code
- Tools like Profilers allow you to monitor the following:
  - CPU Usage - Across all cores, usually % utilisation
  - Memory Usage - Ram, Stack, Heap etc
  - GPU - GPU core & memory usage and shader performance
  - Code - Timings, function calls stats, call graphs

Detect

Detect

- Usually the result of looking at the data from the profiler

Detect

- Usually the result of looking at the data from the profiler
- With every change you are looking for the biggest possible performance increase

Detect

- Usually the result of looking at the data from the profiler
- With every change you are looking for the biggest possible performance increase
- Always start with the big picture and work your way down

Detect

- Usually the result of looking at the data from the profiler
- With every change you are looking for the biggest possible performance increase
- Always start with the big picture and work your way down
  - CPU or GPU slowing you down most?

Detect

- Usually the result of looking at the data from the profiler
- With every change you are looking for the biggest possible performance increase
- Always start with the big picture and work your way down
  - CPU or GPU slowing you down most?
  - If GPU is under utilised, perhaps shift some of work to the GPU (Compute Shaders) or perhaps GPU is waiting for CPU

Detect

- Usually the result of looking at the data from the profiler
- With every change you are looking for the biggest possible performance increase
- Always start with the big picture and work your way down
  - CPU or GPU slowing you down most?
  - If GPU is under utilised, perhaps shift some of work to the GPU (Compute Shaders) or perhaps GPU is waiting for CPU
  - If CPU is over utilised, perhaps look at profiling code in functions

Solve

Solve

- Once you have detected the problem you need to solve it

Solve

- Once you have detected the problem you need to solve it
- This could involve rewriting an algorithm or changing data structures

Solve

- Once you have detected the problem you need to solve it
- This could involve rewriting an algorithm or changing data structures
- In all cases the data captured should drive your work

Check

Check

- After a change has been made you should always run the profiler again

Check

- After a change has been made you should always run the profiler again
- Also check on different hardware!

Repeat

Repeat

- A change in your code base can cause other issues to crop

Repeat

- A change in your code base can cause other issues to crop
- Create a new benchmark and start the process again

Levels of Optimisation

Levels of Optimisation

- System Level: Utilisation, Balancing and Efficiency

Levels of Optimisation

- System Level: Utilisation, Balancing and Efficiency
- Algorithmic Level: Focus on removing work

Levels of Optimisation

- System Level: Utilisation, Balancing and Efficiency
- Algorithmic Level: Focus on removing work
- Micro-Level: Line by line optimising (data structures is a good example here)

Optimisation Pitfalls

Optimisation Pitfalls

- Assumptions: Always measure!

Optimisation Pitfalls

- Assumptions: Always measure!
- Premature Optimisation: Don't optimise with data, or too early in the development process

Optimisation Pitfalls

- Assumptions: Always measure!
- Premature Optimisation: Don't optimise with data, or too early in the development process
- Optimisation on Only One Machine: Test on the worst case system

Optimisation Pitfalls

- Assumptions: Always measure!
- Premature Optimisation: Don't optimise with data, or too early in the development process
- Optimisation on Only One Machine: Test on the worst case system
- Optimising Debug Builds

FALMOUTH
UNIVERSITY

Coffee Break

FALMOUTH
UNIVERSITY

Housekeeping and Admin

FALMOUTH
UNIVERSITY

Porting Hardware

Exercise

1. Fork the coursework repo - `https://github.com/Falmouth-Games-Academy/comp350-optimisation`
2. Identify your main development tools (Unity or Unreal, Native Code)
3. Investigate the various profiling options
4. Record in a word doc (or similar) resources for these tools
5. Answer the following questions
   - What stats can be collected?
   - Can you profile the GPU?
   - What data can you record about your own code?
   - Can you customise the Profiler, does it have an API?
6. Carry out a Pull Request for feedback