FALMOUTH
UNIVERSITY

COMP360: Research & Development: Dissertation
5: Optimisation and Refactoring

Test cases

Three questions

Three questions

► What makes a good test case?

Three questions

- ▶ What makes a good test case?
- ▶ What makes a good bug report?

Three questions

- ► What makes a good test case?
- ► What makes a good bug report?
- ► What does a unit test describe?

Three questions

- ► What makes a good test case?
- ► What makes a good bug report?
- ► What does a unit test describe?
- ► The answer to these three questions is very similar...

Test case

Test case

A good test case should include:

Test case

A good test case should include:

- ▶ Any assumptions, initial conditions, prerequisites

Test case

A good test case should include:
- Any assumptions, initial conditions, prerequisites
- A set of steps to follow

Test case

A good test case should include:

- ▶ Any assumptions, initial conditions, prerequisites
- ▶ A set of steps to follow
- ▶ The expected result

Test case

A good test case should include:

- Any assumptions, initial conditions, prerequisites
- A set of steps to follow
- The expected result
- (When the test is carried out) The actual result

Bug report

Bug report

A good bug report should include:

Bug report

A good bug report should include:

- ► Any assumptions, initial conditions, prerequisites

Bug report

A good bug report should include:

- ► Any assumptions, initial conditions, prerequisites
- ► A set of steps to follow

Bug report

A good bug report should include:

- ▶ Any assumptions, initial conditions, prerequisites
- ▶ A set of steps to follow
- ▶ The expected result

Bug report

A good bug report should include:

- ▶ Any assumptions, initial conditions, prerequisites
- ▶ A set of steps to follow
- ▶ The expected result
- ▶ The actual result (the bug being that this differs from the expected result)

Unit test

Unit test

A good unit test should include:

Unit test

A good unit test should include:

► Code to set up any assumptions, initial conditions, prerequisites

Unit test

A good unit test should include:

- ▶ Code to set up any assumptions, initial conditions, prerequisites
- ▶ A function to execute (a set of steps)

Unit test

A good unit test should include:

- ▶ Code to set up any assumptions, initial conditions, prerequisites
- ▶ A function to execute (a set of steps)
- ▶ An assertion checking the expected result matches the actual result

How they fit together

How they fit together

▶ A test case is general — can fit at unit, integration, system or acceptance testing level

How they fit together

- ► A test case is general — can fit at unit, integration, system or acceptance testing level
- ► A unit test is essentially an automated test case

How they fit together

- A test case is general — can fit at unit, integration, system or acceptance testing level
- A unit test is essentially an automated test case
- A bug report is suggestive of a new test case that should be added to the test plan

Optimisation

Optimisation

Optimisation

► **Optimisation** is the process of making a program **more efficient** in terms of speed, memory usage etc.

Optimisation

- **Optimisation** is the process of making a program **more efficient** in terms of speed, memory usage etc.
- Optimisation can **increase** or **decrease** software quality, depending on what measure of "quality" is being used

"Rules of optimization:
Rule 1: Don't do it.
Rule 2 (for experts only): Don't do it yet."

— Michael A. Jackson

"Programmers waste enormous amounts of time thinking about, or worrying about, the speed of noncritical parts of their programs, and these attempts at efficiency actually have a strong negative impact when debugging and maintenance are considered. We *should* forget about small efficiencies, say about 97% of the time: **premature optimization is the root of all evil**. Yet we should not pass up our opportunities in that critical 3%."

— Donald Knuth

"Measure twice, cut once."

— Proverb

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

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

► Trying to second-guess the compiler: in many cases, the compiler is better at micro-level optimisation than you are!

Profilers

Unity Profiler

Unity Profiler

► The Unity Profiler is built into the engine

Unity Profiler

▶ The Unity Profiler is built into the engine

▶ It can be accessed via the **Window** > **Profiler**

Unity Profiler

- ▶ The Unity Profiler is built into the engine
- ▶ It can be accessed via the **Window** >**Profiler**
- ▶ This allows you to profile the following

Unity Profiler

- ► The Unity Profiler is built into the engine
- ► It can be accessed via the **Window** >**Profiler**
- ► This allows you to profile the following
  - ► CPU Usage - Scripts, Physics, UI etc

Unity Profiler

- ▶ The Unity Profiler is built into the engine
- ▶ It can be accessed via the **Window** >**Profiler**
- ▶ This allows you to profile the following
  - ▶ CPU Usage - Scripts, Physics, UI etc
  - ▶ Rendering - Batches, Triangles, Vertices

Unity Profiler

- ▶ The Unity Profiler is built into the engine
- ▶ It can be accessed via the **Window** >**Profiler**
- ▶ This allows you to profile the following
    - ▶ CPU Usage - Scripts, Physics, UI etc
    - ▶ Rendering - Batches, Triangles, Vertices
    - ▶ Memory - Total, Texture, Mesh, Garbage Collection

Unity Profiler

- ▶ The Unity Profiler is built into the engine
- ▶ It can be accessed via the **Window** >**Profiler**
- ▶ This allows you to profile the following
    - ▶ CPU Usage - Scripts, Physics, UI etc
    - ▶ Rendering - Batches, Triangles, Vertices
    - ▶ Memory - Total, Texture, Mesh, Garbage Collection
    - ▶ Audio - Number of Sources, Audio Memory

Unity Profiler

▶ The Unity Profiler is built into the engine

▶ It can be accessed via the **Window** >**Profiler**

▶ This allows you to profile the following

  ▶ CPU Usage - Scripts, Physics, UI etc
  ▶ Rendering - Batches, Triangles, Vertices
  ▶ Memory - Total, Texture, Mesh, Garbage Collection
  ▶ Audio - Number of Sources, Audio Memory
  ▶ GPU - Deferred Lighting, Transparent, Post Processing

Unity Profiler

Unity Profiler: Hints & Tips

Unity Profiler: Hints & Tips

► You can remove items from the profiler graph by click on the colour box

Unity Profiler: Hints & Tips

► You can remove items from the profiler graph by click on the colour box
► Enabling **Deep Profile** will add a significant overhead to larger games

Unity Profiler: Hints & Tips

► You can remove items from the profiler graph by click on the colour box

► Enabling **Deep Profile** will add a significant overhead to larger games

  ► Surround you code with **Profiler.BeginSample** & **Profiler.EndSample** this will appear in the Profiler

Unity Profiler: Hints & Tips

► You can remove items from the profiler graph by click on the colour box

► Enabling **Deep Profile** will add a significant overhead to larger games

  ► Surround you code with **Profiler.BeginSample** & **Profiler.EndSample** this will appear in the Profiler

► You should consider Profiling a development build as the Editor adds significant overheard

Unreal Profiler

Unreal Profiler

▶ The Unreal Profiler is built into the engine

Unreal Profiler

- ► The Unreal Profiler is built into the engine
- ► It can accessed via **Window** >**Developer Tools** >**Session Frontend**

Unreal Profiler

► The Unreal Profiler is built into the engine
► It can accessed via **Window** >**Developer Tools** >**Session Frontend**
► Allows us to profile all major systems including CPU (code) and GPU

Unreal Profiler

Unreal Profiler: Hints & Tips

Unreal Profiler: Hints & Tips

► `https://www.unrealengine.com/en-US/blog/`
`how-to-improve-game-thread-cpu-performance`

Unreal Profiler: Hints & Tips

► `https://www.unrealengine.com/en-US/blog/how-to-improve-game-thread-cpu-performance`

► A few interesting things from this link

Unreal Profiler: Hints & Tips

- `https://www.unrealengine.com/en-US/blog/how-to-improve-game-thread-cpu-performance`
- A few interesting things from this link
  - To identify the bottleneck (GPU or CPU), run **stat unit** on a **non-debug build**

Unreal Profiler: Hints & Tips

- `https://www.unrealengine.com/en-US/blog/how-to-improve-game-thread-cpu-performance`
- A few interesting things from this link
    - To identify the bottleneck (GPU or CPU), run **stat unit** on a **non-debug build**
    - If the **Frame Time** is very close to **GPU Time**, then the GPU is the bottleneck

Unreal Profiler: Hints & Tips

- `https://www.unrealengine.com/en-US/blog/how-to-improve-game-thread-cpu-performance`
- A few interesting things from this link
  - To identify the bottleneck (GPU or CPU), run **stat unit** on a **non-debug build**
  - If the **Frame Time** is very close to **GPU Time**, then the GPU is the bottleneck
  - If the **Frame Time** is very close to **Game Time**, then your code is the bottleneck

Unreal Profiler: Hints & Tips

- `https://www.unrealengine.com/en-US/blog/ how-to-improve-game-thread-cpu-performance`
- A few interesting things from this link
  - To identify the bottleneck (GPU or CPU), run **stat unit** on a **non-debug build**
  - If the **Frame Time** is very close to **GPU Time**, then the GPU is the bottleneck
  - If the **Frame Time** is very close to **Game Time**, then your code is the bottleneck
  - In the profiler **GameThread** entry, find the **FTickFunctionTask** - this shows every actor and component that is ticking

Unreal Profiler: Hints & Tips

- `https://www.unrealengine.com/en-US/blog/how-to-improve-game-thread-cpu-performance`
- A few interesting things from this link
  - To identify the bottleneck (GPU or CPU), run **stat unit** on a **non-debug build**
  - If the **Frame Time** is very close to **GPU Time**, then the GPU is the bottleneck
  - If the **Frame Time** is very close to **Game Time**, then your code is the bottleneck
  - In the profiler **GameThread** entry, find the **FTickFunctionTask** - this shows every actor and component that is ticking
  - Another thing to track is **Blueprint Time**, switch inclusive view and locate it, then switch back to hierarchical view

Unreal Profiler: Hints & Tips

▶ `https://www.unrealengine.com/en-US/blog/`
  `how-to-improve-game-thread-cpu-performance`

▶ A few interesting things from this link

  ▶ To identify the bottleneck (GPU or CPU), run **stat unit** on a **non-debug build**

  ▶ If the **Frame Time** is very close to **GPU Time**, then the GPU is the bottleneck

  ▶ If the **Frame Time** is very close to **Game Time**, then your code is the bottleneck

  ▶ In the profiler **GameThread** entry, find the **FTickFunctionTask** - this shows every actor and component that is ticking

  ▶ Another thing to track is **Blueprint Time**, switch inclusive view and locate it, then switch back to hierarchical view

  ▶ **SkinnedMeshComp Tick** & **TickWidgets** can also be bottleneck

Visual Studio Profiler

Visual Studio Profiler

- `https://docs.microsoft.com/en-us/`
  `visualstudio/profiling/`
  `beginners-guide-to-performance-profiling`

Visual Studio Profiler

▶ `https://docs.microsoft.com/en-us/visualstudio/profiling/beginners-guide-to-performance-profiling`

▶ Switch your application to a release build

Visual Studio Profiler

- `https://docs.microsoft.com/en-us/ visualstudio/profiling/ beginners-guide-to-performance-profiling`
- Switch your application to a release build
- To run the profiler, select **Debug** >**Performance Profiler** and then click on **Performance Wizard**

Visual Studio Profiler

- ► `https://docs.microsoft.com/en-us/`
  `visualstudio/profiling/`
  `beginners-guide-to-performance-profiling`
- ► Switch your application to a release build
- ► To run the profiler, select **Debug** >**Performance Profiler** and then click on **Performance Wizard**
- ► The profiler will run and start collecting data

Visual Studio Profiler

- `https://docs.microsoft.com/en-us/`
  `visualstudio/profiling/`
  `beginners-guide-to-performance-profiling`
- Switch your application to a release build
- To run the profiler, select **Debug** >**Performance Profiler** and then click on **Performance Wizard**
- The profiler will run and start collecting data
- Close the application to start analysing the data

Visual Studio Profiler
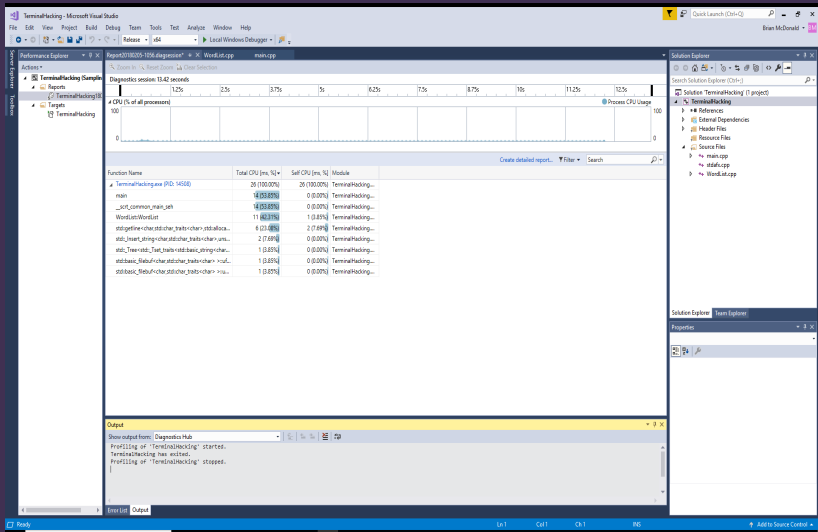
Visual Studio: Hints & Tips

Visual Studio: Hints & Tips

► Click on **Create Detailed Report** in the summary view, this will generate a report on your application

Visual Studio: Hints & Tips

► Click on **Create Detailed Report** in the summary view, this will generate a report on your application

► In this report **Show Hot Lines** will show the code paths which do the most work

Visual Studio: Hints & Tips

- ► Click on **Create Detailed Report** in the summary view, this will generate a report on your application
- ► In this report **Show Hot Lines** will show the code paths which do the most work
- ► You will not be able to do much about the *.dll calls, you should look at your own functions in here

Code smells

Code smells

Code smells

▶ "Code smells" are a useful guideline for refactoring

Code smells

- ▶ "Code smells" are a useful guideline for refactoring
- ▶ Warning signs that code needs to be refactored

Code smells

- "Code smells" are a useful guideline for refactoring
- Warning signs that code needs to be refactored
- `https://blog.codinghorror.com/code-smells/`

Code smells

- "Code smells" are a useful guideline for refactoring
- Warning signs that code needs to be refactored
- https://blog.codinghorror.com/code-smells/
- See also "How to write unmaintainable code" https://github.com/Droogans/unmaintainable-code