

# GAM250: Advanced Games Programming

## 1: Code Smells

# Learning outcomes

- ▶ **Understand** the concept of Code Smells
- ▶ **Demonstrate** the ability to detect code smells
- ▶ **Use** a Static Analyzer to detect problems in your code

# Module Intro

- ▶ This module is about becoming a better programmer
- ▶ We will examine more specialised topics such as Networking, AI, PCG
- ▶ We will look how data can inform our code and game design
- ▶ How we can build tools to assist the development pipeline

# Assessments

- ▶ Assessment 1 - Research Journal (40%)
  - ▶ This is where you detail your research on a topic of your choice
  - ▶ A **maximum** of 4000 words
  - ▶ Worth 40% of your mark for the module
- ▶ Assessment 2 - Game Project (60%)
  - ▶ This is where you detail your research on a topic of your choice
  - ▶ A **maximum** of 4000 words
  - ▶ Worth 40% of your mark for the module

# Code Smells Definition

A code smell is a surface indication that usually corresponds to a deeper problem in the system

- *Martin Fowler*

# Code Smells - Points to note

1. Something that is quick to spot or **sniffable**
2. They don't always indicate a problem, but that the code requires more investigation
3. After investigation and a deeper problem is indicated then you should **Refactor**

# Code Smells - Taxonomy

- ▶ Further work by Mäntylä and Lassenius identified a Taxonomy of smells
- ▶ These classified similar smells into categories
- ▶ Categories include The Bloaters, The Object-Orientation Abusers, Change Preventers, The Dispensables, The Couplers

# Code Smells - Research

- ▶ Form into 5 teams
- ▶ Each team will research a category
- ▶ Use the following URL
- ▶ `https://sourcemaking.com/refactoring/smells`
- ▶ Collate your research into a shareable doc such as Google doc



Coffee Time! - 30 min break

# Static Code Analysis

- ▶ Usually part of a code review
- ▶ A tool is ran on the source code and it detects numerous problems including:
  - ▶ Bugs
  - ▶ Performance issues
  - ▶ Some Code Smells
- ▶ The information provide can be used to improve your code base
- ▶ This ideally should be automated and the report analysed by the team

# Static Code Analysis Tools for Unity & C#

- ▶ Unity Engine Analyzer - <https://github.com/meng-hui/UnityEngineAnalyzer>
- ▶ Gendarme for Unity - <https://forum.unity.com/threads/gendarme-for-unity-a-code-analysis-tool.112552/>

# Static Code Analysis Demo

# Static Code Analysis Exercise

- ▶ Download one of your old Projects (GAM150, GAM160 or 1st Year Project)
- ▶ Run Gendarme or Unity Engine Analyzer on your project
- ▶ Select some of the issues identified, can you find ways to fix them?

Further reading (or watching!) -  
<https://www.youtube.com/watch?v=VxeC7WFfg3Q>