Teams, Waterfall Method and Project Brief

Workshop 1

Ruzanna Chitchyan, Jon Bird, Pete Bennett
TAs: Mitch Lui, Craig Barnfield, Kira Clements, Ollie Myers

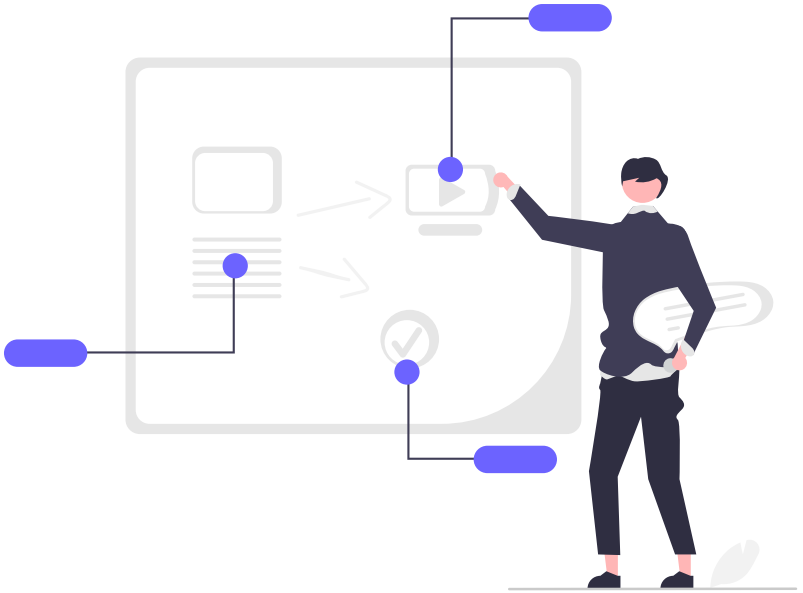
Week	Date	**Lecture** Monday 11:00-12:00 *PHYS BLDG G44 FRANK*	**Workshop** Monday 13:00-15:00 *MVB 2.11 PC*	**Groupwork**
1	23/01/22	Introduction and Process [slides] [materials]	Teams, Waterfall Method and Project Brief [slides] [case study] [project brief]	Research games, create list on team repo. Install Processing
2	30/01/22	Agile Methodology and User Centred Design	Intro to Processing, Agile Techniques	Decide on two game ideas
3	06/02/22	Requirements Engineering	Paper Prototyping, Requirements, Ideas Clinic	Collect requirements. Decide on final idea
4	13/02/22	Object Orientated Design	Classes Activity	Add requirements section to report
5	20/02/22	Implementation	Case Study, Spring Prep, Continuous Integration	Develop a working prototype over reading week!
6	28/02/22	READING WEEK	GAMES JAM	
	06/03/22	Project Management	IN CLASS TEST (assessing lectures 1-5)	Define team roles
8	13/03/22	HCI - Qualitative	HCI Qualitative Task	Add qualitative assessment (of your choice) to report
9	20/03/22	HCI - Quantitative	HCI Quantitative Task	Add quantitative assessment (of your choice) to report
	27/03/22	EASTER week 1	SPRINT 1	
	03/04/22	EASTER week 2	SPRINT 2	
	10/04/22	EASTER week 3	SPRINT 3	
10	17/04/22	Software Engineering Extended	IN CLASS TEST (assessing lectures 7-9)	Develop Game
11	24/04/22	Coursework Feedback		Finish Report
12	01/05/22	Bank Holiday Monday (no class)	Demo Day Weds/Thurs (tbc)	Submit Report

Today’s Workshop

- Meet the TAs
- Meet your team (10mins)
- Waterfall case study (60mins)
- Introduction to Project Brief (20mins)
- Set up team repo on Github (20mins)

Meet the TAs

- **Mitch Lui** - Github/CI master
- **Craig Barnfield** – old hand (TA last year) games enthusiast
- **Kira Clements** – MSc Projects + expertise from being on course last year
- **Ollie Myers** – also, games enthusiast!



Meet your Team

- You have been randomly assigned to a team of five. Look up your team:
    - You can find out which group you belong by going to https://github.com/orgs/UoB-COMSM0110/teams and typing in your username on the search box.
    - You can find your group's repository here: https://github.com/orgs/UoB-COMSM0110/repositories
- Get together in your group and introduce yourself (10mins) … *last book you read, favourite music... last game you played.*

Case Study: Insulin Pump Control System

attendance code:
7298

An insulin pump is a medical system that simulates the operation of the pancreas (an internal organ). The software controlling this system is an embedded system, which collects information from a sensor and controls a pump that delivers a controlled dose of insulin to a user.

People who suffer from diabetes use the system. Diabetes is a relatively common condition where the human pancreas is unable to produce sufficient quantities of a hormone called insulin. Insulin metabolises glucose (sugar) in the blood. The conventional treatment of diabetes involves regular injections of genetically engineered insulin. Diabetics measure their blood sugar levels using an external meter and then calculate the dose of insulin that they should inject.

The problem with this treatment is that the level of insulin required does not just depend on the blood glucose level but also on the time of the last insulin injection. This can lead to very low levels of blood glucose (if there is too much insulin) or very high levels of blood sugar (if there is too little insulin). Low blood glucose is, in the short term, a more serious condition as it can result in temporary brain malfunctioning and, ultimately, unconsciousness and death. In the long term, however, continual high levels of blood glucose can lead to eye damage, kidney damage, and heart problems.

Current advances in developing miniaturized sensors have meant that it is now possible to develop automated insulin delivery systems. These systems monitor blood sugar levels and deliver an appropriate dose of insulin when required. Insulin delivery systems like this already exist for the treatment of hospital patients. Many diabetics want to have such systems permanently attached to their bodies.

To Do: Working in Your Group

- Discuss 3 reasons on why Waterfall process could work well for this case study (15 min)
- Class discussion (5 min)
- Set up a plan of actions for your team (15 min) :
  - What needs to be done?
  - Who will do it?
  - At what time, and in what sequence?
- Class discussion (5 min)

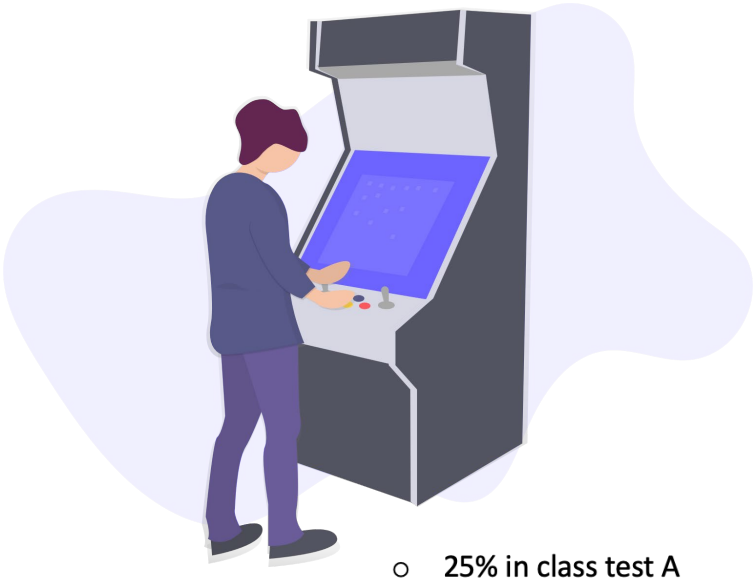
- Discuss 3 reasons on why Waterfall process could be a poor choice for this case study (15 min)
- Class discussion (5 min)

Coursework Brief – Games Project

- The aim of this group coursework is to **design and develop a game**, applying and reflecting on the software engineering methods you have learnt in the taught component of this module.

- Each team of 5-6 people (assigned randomly) will develop a novel computer game based around adding a twist to an existing game or game archetype.

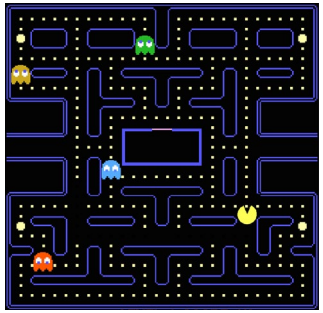
- We would like your team to identify and discuss **three major software development challenges** in developing your game.



- 25% in class test A
- 25% in class test B
- **25% project report**
- **25% project presentation**

Pac-man Example

- Pac-man, *but* the ghosts can rearrange the map
    - Challenge 1: The ghost's AI
    - Challenge 2: Data structure for storing the map
    - Challenge 3: Global leader-board



Asteroids Example

- Asteroids, *but* you play as the asteroids rather than the spaceship.
    - Challenge 1: Multiplayer implementation
    - Challenge 2: Asteroids fragmentation.
    - Challenge 3: Spaceship AI

Tetris Example

- Tetris, *but* the blocks have different properties.
    - Challenge 1: Development of physical simulation
    - Challenge 2: Documentation and onboarding for block properties
    - Challenge 3: Algorithm for collision detection

Any Game Genre? Sure!

- Action
- Platformer
- Shooter
- Beat ‘em up
- Stealth
- Survival
- Rhythm
- Puzzle
- RPG
- Simulation
- Tower Defense
- Arcade
- Sports
- Party Game
- Horror
- Social Deduction
- Serious
- Generative
- Metroidvania
- Roguelike
- Dungeon-crawler
- Deck-builder
- One-button
- Text-based

… and many many more

https://www.theguardian.com/games/2021/oct/11/modern-video-game-genres-explained-metroidvania-dungeon-crawler

https://en.wikipedia.org/wiki/List_of_video_game_genres

That sounds like a lot!

- Remember that the primary aim of this module is to **learn**, **apply** and **reflect** on the principles of **good software engineering**.
- The quality of your process is the primary focus of assessment and not your game idea, though we hope that a quality game will be the outcome of a good process!
- We aim that by following the workshops this should be a straightforward coursework to pass, but with plenty of opportunity to excel and achieve higher grades.

But, what language?

- **Processing:** https://processing.org (*not Unity!*)
- It's Java! (with some extras) You'll be learning Java this term.
- Designed for use by artists, designers, creative technologists (designed to be straightforward, excellent documentation)
- Has a simple Integrated Development Environment (IDE)
- Many useful libraries!

- Why? We want to give you the experience of working as a team in learning a new language / environment (useful for summer project)
- But also, we want it to be a simple and enjoyable experience so you can focus on the software development process.

Some tips

- We will not be teaching you Processing, your team will be expected to engage and learn from online resources available (we will link you to these).

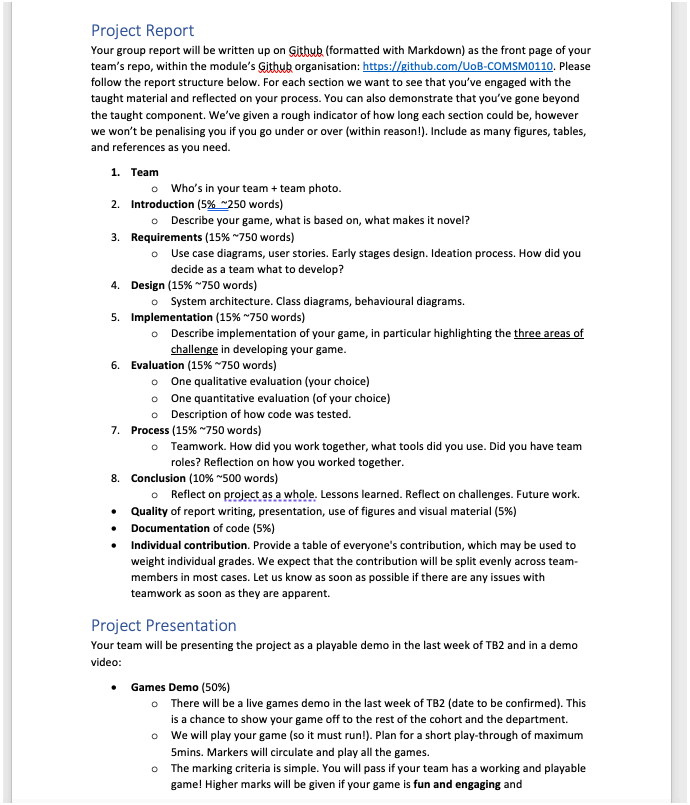
- Processing has many libraries which you are free to use, but remember that you need to identify and overcome three challenges, so if you use a library (for say physical modelling) then this may negate the challenge.

- Please note that this is a software engineering module and not a game design module, so although good narrative/artwork/levels/content will elevate a game, we are ultimately interested in your team's ability to develop software.

- We recommend sticking to 2D games (unless this is one of your challenges!)

Deliverables

- Project Report (25% module)
  - Front page of your Github project repo
- Presentation (25% module)
  - Game Demo in last week
  - Video of your game.
- Project Brief is available on BlackBoard: (link within table)



Project Report

Your group report will be written up on Github (formatted with Markdown) as the front page of your team's repo, within the module's Github organisation: https://github.com/UoB-COMSM0110. Please follow the report structure below. For each section we want to see that you've engaged with the taught material and reflected on your process. You can also demonstrate that you've gone beyond the taught component. We've given a rough indicator of how long each section could be, however we won't be penalising you if you go under or over (within reason!). Include as many figures, tables, and references as you need.

1. **Team**
   - Who's in your team + team photo.
2. **Introduction** (5% ~250 words)
   - Describe your game, what is based on, what makes it novel?
3. **Requirements** (15% ~750 words)
   - Use case diagrams, user stories. Early stages design. Ideation process. How did you decide as a team what to develop?
4. **Design** (15% ~750 words)
   - System architecture. Class diagrams, behavioural diagrams.
5. **Implementation** (15% ~750 words)
   - Describe implementation of your game, in particular highlighting the three areas of challenge in developing your game.
6. **Evaluation** (15% ~750 words)
   - One qualitative evaluation (your choice)
   - One quantitative evaluation (of your choice)
   - Description of how code was tested.
7. **Process** (15% ~750 words)
   - Teamwork. How did you work together, what tools did you use. Did you have team roles? Reflection on how you worked together.
8. **Conclusion** (10% ~500 words)
   - Reflect on project as a whole. Lessons learned. Reflect on challenges. Future work.

- **Quality** of report writing, presentation, use of figures and visual material (5%)
- **Documentation** of code (5%)
- **Individual contribution**. Provide a table of everyone's contribution, which may be used to weight individual grades. We expect that the contribution will be split evenly across team-members in most cases. Let us know as soon as possible if there are any issues with teamwork as soon as they are apparent.

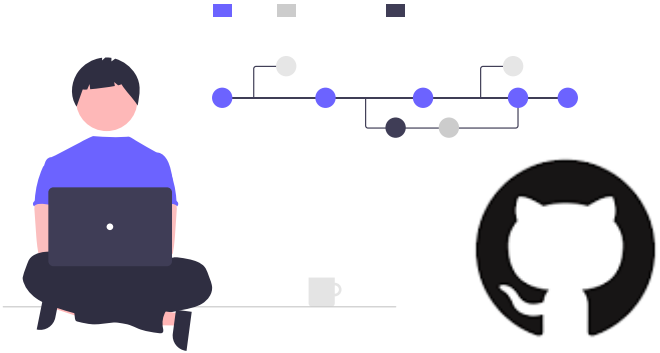
Project Presentation

Your team will be presenting the project as a playable demo in the last week of TB2 and in a demo video:

- **Games Demo** (50%)
  - There will be a live games demo in the last week of TB2 (date to be confirmed). This is a chance to show your game off to the rest of the cohort and the department.
  - We will play your game (so it must run!). Plan for a short play-through of maximum 5mins. Markers will circulate and play all the games.
  - The marking criteria is simple. You will pass if your team has a working and playable game! Higher marks will be given if your game is **fun and engaging** and

homework / groupwork

- Make sure you can make a commit. Try adding your name to your team's readme (After Software Tools on Thursday!)
- Then take and upload a team photo!
- Research games, create a list of games inspiration on your team’s repo. Make sure your team has plenty of options. Consider adding notes to each game (what makes it great?) and perhaps even ranking them by suitability and interest.



homework / groupwork

- Also… install Processing on your laptop: https://processing.org/download
    - No need to do any tutorials, we'll be starting from scratch in next week's workshop