R-Type

B5 – Advanced CPP

R-Type, the game

Space themed 1987 arcade game

Horizontal Shooter – Shoot 'em up

Simple Gameplay:
- Kill evil aliens
- Acquire power-ups
- Defeat Bosses
- Go to next level

Gameplay Demo

EPITECH. TECHNOLOGY

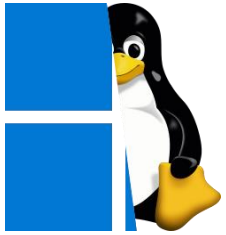
Objectives

EPITECH. TECHNOLOGY

Design your networked game-engine in C++
- Network programming
- Game Engine architecture
- Networked Multiplayer Game

Explore advanced topics
- Architecture
- Network
- Game design

Adopt proper software engineering practices
- Documentation
- Build/packaging tools
- VCS workflow
- Cross-platform programming



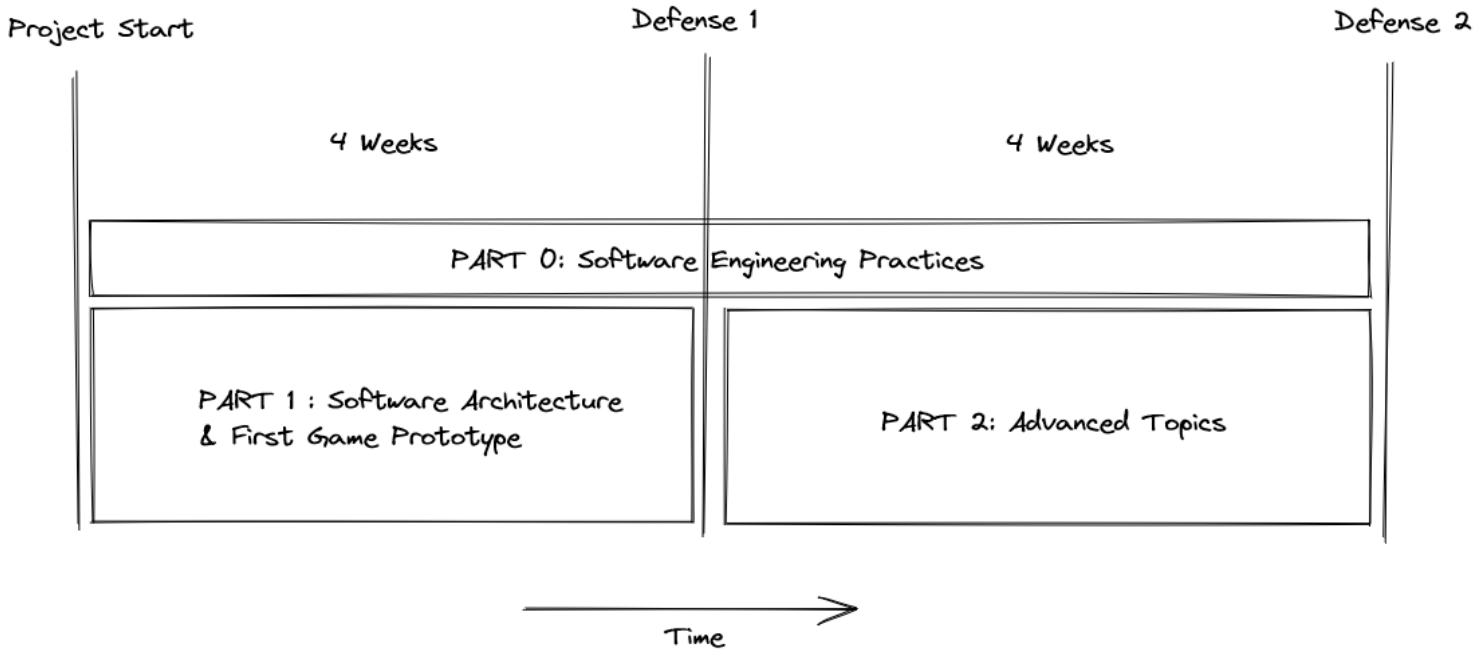
Overview

Part 0 – Software Engineering
- "Professional" product
- Both defense – 1 credit each

Part 1 – Prototype
- Core features
- Basic gameplay
- 1st defense - 2 credits

Part 2 – Advanced software
- Improve the prototype
- Pick & choose advanced topic
- 2nd defense - 2 credits

Bonus – National Jury
- A grade only
- 2 bonus credits for best projects



Part 0 – Software engineering

The difference between project and product

Project or Product ?

EPITECH. TECHNOLOGY

Project

- "Work on my computer"
- "Here are the dependencies you need"
- "Just ping me if you have questions"
- "Hop on the discord if you have idea"

Product

- Work everywhere
- Install its dependencies
- Well-made documentation
- Contributing rules & guidelines

Expectations

EPITECH. TECHNOLOGY

Self-contained project
- No need to manually download dependencies

Cross-platform – At least 2 platforms supported
- Windows/Linux (No, WSL doesn't count)
- Mac is **also** unix based (an argument can be made about M1 chips)

Proper VCS workflow
- Branches
- No "live-share"
- Explicit commits title & body

Packaged Release
- CMake´s CPack module
- Github's Release page

Documentation

Proper documentation is MANDATORY!
It **must** be up to date!

Useful Readme
- Install info
- Usage
- Examples
- etc.

Developer documentation
- Diagrams (layers, subsystems, UMLs, etc.)
- Subsystems explanation
- Tutorials/wiki/how-to's
- In-code documentation

Documentation - Protocol

EPITECH. TECHNOLOGY

What make a good protocol documentation
- language agnostic
- enough to write a client
- acknowledge possible errors
- explain how to handle them
- Follow standard formatting (RFC)

Tips:
- Integer, internal padding, etc. are compiler & language dependent
- Enums means nothing. Values do.
- Diagrams can be clearer than words

Various considerations

PDF & Words aren't proper documentation format.

In code documentation formats
- allows for IDE integration
- allows for interactive doc generation
- are a good practices

Some tools are widely used
- Markdown, RestructuredText, Sphinx, Doxygen

Github markdown support links.

Part I – Prototype

Starting a project, the correct way

What is a prototype ?

EPITECH. TECHNOLOGY

Not feature complete
- Core feature are there
- Don't care too much about details

Make careful architectural choices
- But remember that you **will** refactor later

Good questions to ask
- Do I need network in an online multiplayer game ?
- Do I need a menu ?
- How does my current choices impact the future of the project ?

Objectives

Design your networked multiplayer engine
- Have a proper software architecture

Create a basic R-Type like Gameplay
- Moving player
- Moving monster
- Basic interaction (killing enemies, dying, etc.)

Have a proper base for Part II

Part II - Improvements

Beyond the prototype

Where to go now ?

The only way is further
- Refactoring **is** progress

3 technical tracks
- Advanced Game engine
- Advanced Networking
- Advanced Game-design

Each track have several topics
- You can choose topics from each tracks
- Quality & difficulty > quantity
- Emphasis on future proofing

More information next kick-off

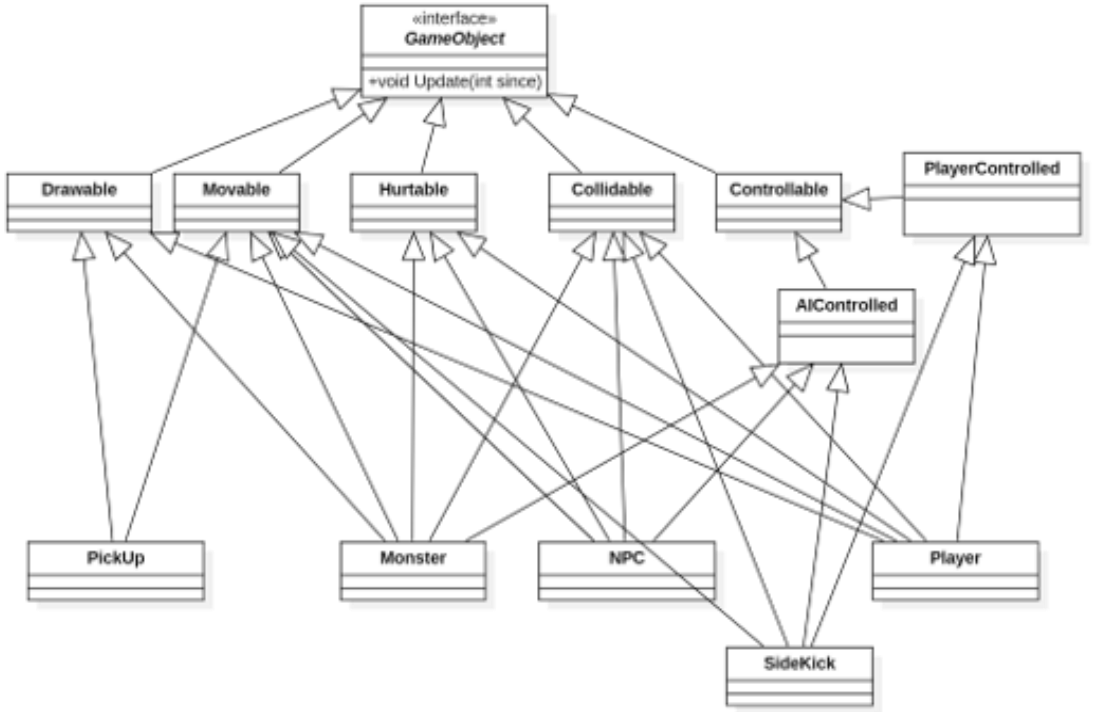
Design, Concept & Idea

Let’s make some multiplayer game engine !

Engine Design – OOP shortcomings

EPITECH. TECHNOLOGY

- Should unrelated objects have a common ancestors ?
- How many ancestors should an object have?
  - Does it make sense ?
- How to handle dynamic changes ?
- How to handle configuration ?
- Can we do better ?
  - Composition over inheritance !

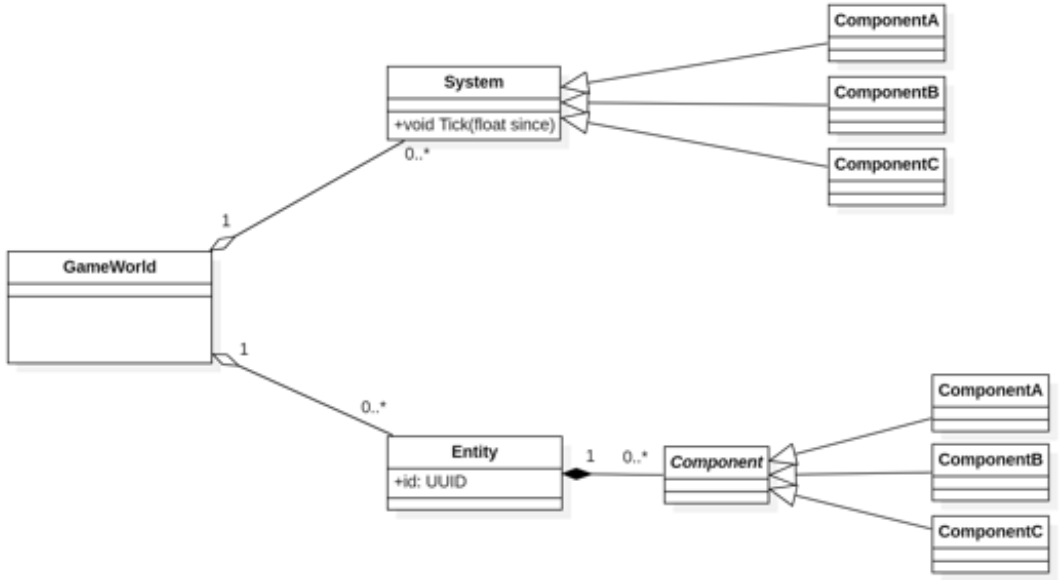


Engine Design – ECS

Entity
- Identify a "thing" inside the engine
- Can be a simple ID

Component
- Identity a trait or capability an Entity has
- Simple POCO (struct)
- Contains data

Systems
- Implement the actual logic and behaviors
- Simple functions (or script)
- Most works on only a few components
- Apply the same function to all entities with given components



Multiplayer Design – What should the server do ?

EPITECH. TECHNOLOGY

Everything ?

- Can ensure data consistency
- Usually, a bad idea
- Networking is sloooow

"Only" Room, chat & Match making ?

- Who is responsible for data consistency ?

Multiplayer Design – Where is the Engine ?

EPITECH. TECHNOLOGY

Server side only
Client is simpler
Rendering and input handling only
Client is lost without constant stream of data.

Server must handle inputs

Client side only
One client must ensure consistency
Each client render its own game

Host can cheat
Host must be powerful enough for 2 games.

Server & Client side
Server is responsible for data
Client can react "instantly" to the inputs ("tighter" controls).

Some way of reconciliation is needed (rollback ?)
More "balanced" usage that client side only.

Less processing overall (the server don't care about inputs, only actions and positions)

Network programming for games

EPITECH. TECHNOLOGY

Some question to ask
- What should I send ?
- How often ?
- Do I need speed or reliability ?
- What happen if it never arrive ? Arrive twice ?

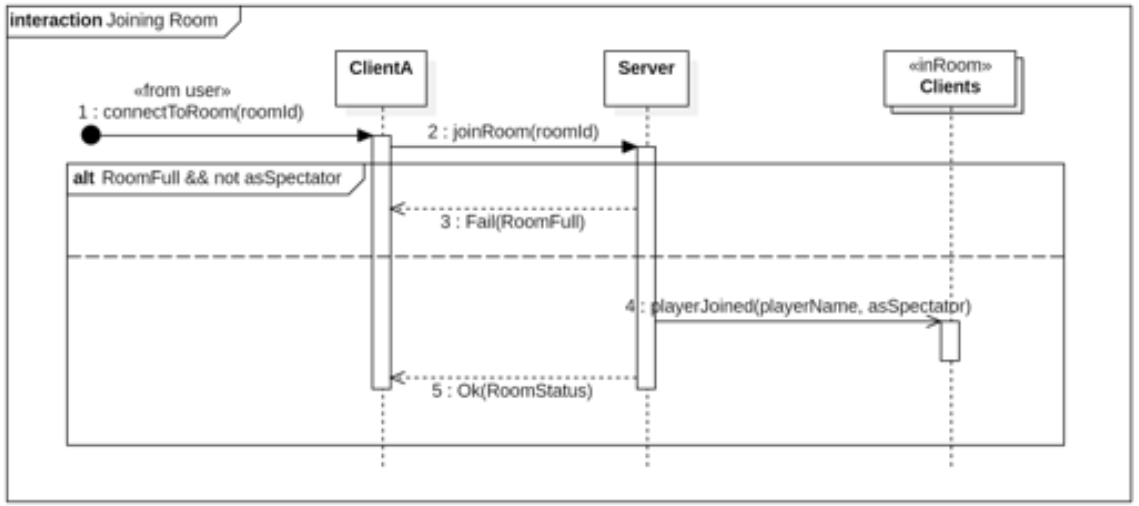
Some answer
- User input is important
- Game data isn't (you can lose part of it)

Network Programming – Protocol Design

Text or Binary ?
- Text is human readable
- Binary is shorter, faster & doesn't require parsing

Only send necessary data

Model your protocol
- Make programming easier
- Visual information is easier to understand



Resources

Where to look, and what to use ?

Research resources

EPITECH. TECHNOLOGY

List of curated articles
https://github.com/ThusSpokeNomad/GameNetworkingResources
- Gaffer on Games: What Every Programmer Needs To Know About Game Networking
- Ruoyu Sun: Game Networking Demystified

Quake3 engine writeup
https://fabiensanglard.net/quake3/index.php

GDC Conferences
- Overwatch Gameplay Architecture and Netcode (youtube)
- I Shot You First: Networking the Gameplay of Halo: Reach (youtube)

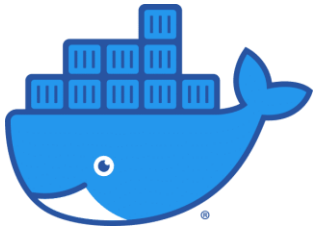
Talk about other architecture
- Bob Nystrom - Is There More to Game Architecture than ECS? (Youtube)

Tool of the trade

EPITECH. TECHNOLOGY

Dependency management
- Git submodules
- CMake (FetchPackage)
- Package managers (Conan, VCPkg)

CI/CD
- Docker (build/test environment)
- Github Actions



Good practice & sanity check
- Test Framework (Catch2, Criterion, Google tests, etc.)
- Linters/formatter (clang-tidy, etc.)

Documentation
- Dia, StarUML, Lucid Chart
- Doxygen, ReadTheDoc.io, Spinx, etc.

Questions ?

The bootstrap is all about software architecture.

Have fun on your project !