Imatge que conté text, captura de pantalla, Font, disseny

Descripció generada automàticament

**Grau en Disseny i Producció de Videojocs**

**Curs acadèmic 24-25**

**Development of a Standardized Networking System**

**for Online Games in Unity**

**TfG**

**Francisco Secchi Tarrés**

**Tutor: Ricard Perea Ros**



**Abstract**

In the abstract, the goal and the most significant aspects of the project have to be shown. It is useful to illustrate the process of development of the project proposal and the results obtained in the project. All these contents should be written in about five-seven lines, in order to fit the three abstracts in only one page. The abstract is very useful because it shows a general idea about the project at a glance. We suggest to write the abstract after the documentation of the project.

**Resum**

En el resum cal indicar l’objectiu i els punts més significatius del treball. És molt útil mostrar el procés de desenvolupament de la proposta de projecte i quins han estat els resultats obtinguts. Tot això cal fer-ho en aproximadament cinc-set línies, de forma que els tres resums càpiguen en una única pàgina. El resum és molt útil perquè mostra una idea general del treball en una lectura ràpida. S’aconsella fer el resum un cop s’hagi redactat tota la documentació.

**Resumen**

En el resumen se debe indicar el objetivo y los puntos más significativos del proyecto. Es muy útil mostrar el proceso de desarrollo y cuáles han sido los resultados obtenidos. Todo esto debe hacerse en aproximadamente cinco-siete líneas, de forma que los tres resúmenes quepan en una única página. El resumen es muy útil porque muestra una idea general del trabajo realizado en una lectura rápida. Se aconseja hacer el resumen una vez se haya redactado toda la documentación.

Contents

[List of figures 6](#_Toc191920604)

[List of tables 6](#_Toc191920605)

[1 Introduction 1](#_Toc191920606)

[2 Theoretical framework 1](#_Toc191920607)

[2.1 General concepts 1](#_Toc191920608)

[2.1.1 Object-Oriented Programming 1](#_Toc191920609)

[2.1.2 Composition 1](#_Toc191920610)

[2.2 Design Patterns 1](#_Toc191920611)

[2.2.1 Singleton 1](#_Toc191920612)

[2.2.2 Command 1](#_Toc191920613)

[2.2.3 State 1](#_Toc191920614)

[2.2.4 Observer 1](#_Toc191920615)

[2.2.5 Factory 1](#_Toc191920616)

[2.2.6 Prototype 1](#_Toc191920617)

[2.2.7 Strategy 2](#_Toc191920618)

[2.2.8 Mediator 2](#_Toc191920619)

[2.3 Unity Overview 2](#_Toc191920620)

[2.3.1 Scripting 2](#_Toc191920621)

[2.3.2 Input System? 2](#_Toc191920622)

[2.3.3 Built-in Patterns 2](#_Toc191920623)

[2.3.4 Packages and libraries 2](#_Toc191920624)

[2.4 Networking 2](#_Toc191920625)

[2.4.1 Concepts 2](#_Toc191920626)

[2.4.2 Models 2](#_Toc191920627)

[2.4.3 Synchronization techniques 2](#_Toc191920628)

[2.4.4 Networking Libraries 2](#_Toc191920629)

[3 Objectives 2](#_Toc191920630)

[3.1 Principal Objectives 2](#_Toc191920631)

[3.2 Secondary Objectives 2](#_Toc191920632)

[4 Methodological design and timeline 3](#_Toc191920633)

[4.1 Methodology 3](#_Toc191920634)

[4.1.1 MoSCoW 3](#_Toc191920635)

[4.1.2 Version Control 3](#_Toc191920636)

[4.2 Planning 3](#_Toc191920637)

[4.2.1 Definition of required features 3](#_Toc191920638)

[4.2.2 List of features using MoSCoW 3](#_Toc191920639)

[5 Development 3](#_Toc191920640)

[5.1 Project Setup 3](#_Toc191920641)

[5.2 Core Modules 3](#_Toc191920642)

[5.2.1 Networking Manager 3](#_Toc191920643)

[5.2.2 State Management 3](#_Toc191920644)

[5.2.3 Rollback System 4](#_Toc191920645)

[5.2.4 Input Buffering/Prediction? 4](#_Toc191920646)

[5.2.5 Synchronization 4](#_Toc191920647)

[5.2.6 Serialization 4](#_Toc191920648)

[5.2.7 Utilities and Debugging 4](#_Toc191920649)

[5.3 Gameplay Framework 5](#_Toc191920650)

[5.3.1 Game State 5](#_Toc191920651)

[5.3.2 Object State 5](#_Toc191920652)

[5.3.3 Object Controller 5](#_Toc191920653)

[5.3.4 HUD 5](#_Toc191920654)

[5.4 UML Design 5](#_Toc191920655)

[5.5 Implementation 5](#_Toc191920656)

[6 Conclusions 6](#_Toc191920657)

[7 Bibliografia/Referències 6](#_Toc191920658)

# List of figures

Fig. 3.1. Interlineat i espaiat del text, en el processador de text MS-Word10

Fig. 3.2. Exemple de figura, on cal remarcar les unitats en els eixos.13

# List of tables

Taula 3.1. Resum del tipus de lletra a emprar en la memòria.11

# Introduction

Unity is one of the bigger, most used, game engines

# Theoretical framework

* 1. General concepts
     1. Object-Oriented Programming

Often written as OPP, Object-Oriented Programming uses the concept of “objects” to design and organize software. Instead of working only with functions and logic, objects define entities that encapsulate data and operations performed on that data. This way of structuring software into classes that represent real-world entities, makes programs more intuitive and easier to read and write.

Each object in a program serves as a self-contained unit that holds both information (attributes) and operations (methods) that act on that information. This structure helps developers design systems that are easier to understand, test, and maintain over time.

OPP is funded in four principles.

**Encapsulation**: restricting direct access to an object’s components to avoid interference and misuse of its data. This is implemented with a privacy key set on classes, attributes and methods to define if they are private or public. Public attributes and methods work as an *Interface* that other objects can access, but still won’t see its implementation.

**Abstraction**: exposing only the essential features of an object while hiding the complex implementation details. This simplifies the interaction with objects and reduces complexity.

**Inheritance**: defines a new class as a subclass, this class inherits attributes and methods from a “superclass”. Inheritance is key to organized and reusable code establishing a hierarchical relationship between classes.

and polymorphism.

One of the main strengths of OOP lies in its four foundational principles: encapsulation, inheritance, abstraction, and polymorphism. Although these terms may sound technical, their purpose is simple: they help reduce repetition, promote code reuse, and make programs more adaptable to change. For example, encapsulation keeps the internal details of objects hidden from the rest of the program, while inheritance allows new objects to be based on existing ones, minimizing duplication.

Object-Oriented Programming is particularly valuable in large software projects where clarity, flexibility, and reusability are important. By modeling a program after real-world systems, OOP makes it easier to collaborate in teams and to extend or update the software as requirements evolve.

This approach to programming has become a standard in the industry, forming the foundation of many popular languages such as Java, Python, and C#. It continues to be a central concept in modern software development due to its natural structure and practical benefits.

* + 1. C# Language

Developed by Microsoft, C# is a modern programming language that was designed to make it easier to build software applications. Now it is one of the most popular languages for developing software, web services and mobile and game applications.

One of the key reasons for its popularity is its balance between power and simplicity. C# allows developers to write clear, organized, and efficient code without needing to manage many of the complex details that older languages often require. It also offers helpful features like automatic memory management, which means the developer does not need to manually track how a program uses the computer’s memory. Features like this make the language very accessible for new developers but still powerful for experienced programmers.

Because of its flexibility, clarity, and strong ecosystem, C# is widely used in many industries and remains an important tool for developing reliable and modern software systems.

* + 1. Composition
    2. Basic network (client, server, etc)
    3. UML
  1. Design Patterns
     1. Singleton

Network Manager accessible per a tots.

* + 1. Command

Guardar i enviar els inputs com a Commands per executar a tots els clients o per fer rollback?

* + 1. State
    2. Observer
    3. Factory

Factory de missatges per enviar a clients? Missatges de inputs, syncro...etc

* + 1. Prototype

Clonar GameStates per enviar als clients i per fer rollback

* + 1. Strategy

Per custmització del framework (diferents estrategies de sincronització etc)

* + 1. Mediator

Para hablar entre sistemas/modulos?

* 1. Unity Overview
     1. Scripting
     2. Built-in Patterns
     3. Packages and libraries
     4. Test Runner
  2. Networking
     1. Concepts

Transport Protocols

Dedicated & Listen Server

Ownership

Replication

* + 1. Models

Authoritative and Non-Authoritative

Peer-to-Peer

Hybrid Peer-to-Peer

Others

* + 1. Synchronization techniques

Rollback

State Replication

Event-based

Delay-based

* + 1. Networking Libraries

Enet-CSharp vs LiteNetLib for UDP

Telepathy vs System.Net.Sockets for TCP

# Objectives

* 1. Principal Objectives
* Develop a standardized networking system for game developers in Unity. This framework does not try to be a solution for all multiplayer games and genres, so it won’t implement all networking models. It focuses on a Host-Client solution with a State Replication system and Rollback capabilities for synchronization.
* Code a well-structured, de-coupled, and fully scalable piece of software using the best programming techniques and patterns. Letting developers implement their own network and synchronization strategies to suit their needs.
* Write a clear and useful framework documentation.
* Create simple scenes to test and visually show all the different functionalities of the framework.
  1. Secondary Objectives
* Test the framework by adapting a single player game, implementing a multiplayer mode.

# Methodological design and timeline

* 1. Methodology
     1. Waterfall
     2. MoSCoW

MoSCoW will be used in this project as the main prioritization technique. This is a categorizing method that defines the relevance of the project’s requirements, functional and non-functional included, using four different categories.

* + 1. Version Control
  1. Planning
     1. Definition of required modules

Transport Abstraction

Sistema de selección de protocolo (dar UDP i TCP + custom?)

Serialization

Data-objeto serialization

Network Management

Encargarse de P2P y listen server, con host migration?

Game Synchronization

Todas las clases para uso del desarrollador (alto nivel)

Rollback

Proporcionar sistema de rollback para P2P con historial de GameStates

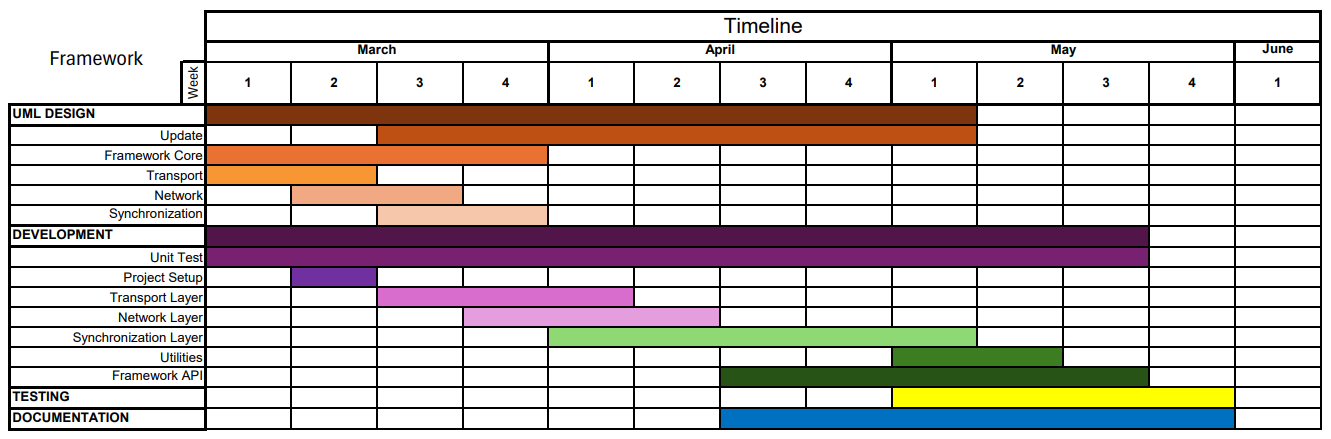
Message System

Sistema de envio de mensajes entre peers. Commands I RPCs.

Debugging and utilities

* + 1. List of features using MoSCoW

|  |  |  |  |
| --- | --- | --- | --- |
| **Priority** | **Module** | **Functionality** | **Description** |
|  | Transport | Setup Transport | Defines the network transport layer and sets up the default protocol. |
|  | Transport | Start | Starts a server/host |
|  | Transport | Connect | Initiates a connection to a server/host |
|  | Transport | Disconnect | Disconnects from the server/host |
|  | Transport | Send | Sends already serialized data to the specified client. |
|  | Transport | Receive | Receives data from the network |
|  | Network | Start Server | Starts a server as host |
|  | Network | Start Client | Starts a client |
|  | Network | Join Host | Connects a client to the host. |
|  | Network | Client Disconnection | Manages client disconnection. |
|  | Network | Host Disconnection | Manages host disconnection. |
|  | Network | Listen | Listens for incoming client connections. |
|  | Network | Send To All | Sends a message to all clients. |
|  | Network | Send To Client | Sends a message to a specified client. |
|  | Network | Send To Host | Sends a message to the host. |
|  | Network | Disconnect | Disconnects the client. |
|  | Sync | Object Identity | Assigns object identity via a unique ID |
|  | Sync | Object Ownership | Assigns an object to a client via a client ID |
|  | Sync | OnStart Network | Called when the object is first spawned in the network. |
|  | Sync | OnDestroy Network | Called when the object is removed from the network. |
|  | Sync | Synchronize variable | Synchronizes a variable across the network |
|  | Message | Call Command | Sends a command from client to host |
|  | Message | Call RPC | Sends an RPC from host to client |
|  | Rollback | Set a State | Switches between GameStates |
|  | Rollback | Clone State | Clones a State to send to all clients |
|  | Rollback | Save State | Saves a State of a specific frame for rollback |
|  | Rollback | Load State | Restores to a past State during rollback |
|  | Rollback | Handle desync | Detects and resolves desynchronizations |
|  | Rollback | Store Input | Saves Input history for rollback |
|  | Rollback | Rollback | Restores the game to a specific frame |
|  | Rollback | Reapply Inputs | Runs stored inputs after rollback |
|  | Rollback | Validate Rollback | Checks if the state is consistent on all clients |
|  | Sync | Network Transform | Synchronize Unity Transform |
|  | Sync | Network Rigidbody | Synchronizes a physic object |
|  | Message | Command Manager | Handles client-host communication |
|  | Serializer | Serialize Data | Convert data for network transportation |
|  | Serializer | Deserialize Data | Converts network data back into objects. |
|  | Transport | Custom Protocols | Allow developers to implement custom transport protocols |
|  | Network | Migrate Host | Transfers host role to another client if the host disconnects. |
|  | Network | Handle Reconnection | Allow disconnected clients to rejoin a session. |
|  | Sync | Custom GameStates | Enables developers to define custom game states. |
|  | Network | Create Lobby | Matchmaking lobby |
|  | Network | Join Lobby | Join an existing lobby |
|  | Network | Start Match | Start when all players are ready |
|  | Serializer | Compress Data | Serialize only changed data, not full states |
|  | Sync | Adaptive Interpolation | Adjust smoothing based on latency |
|  | Sync | Latency Compensation | Adjust all clients for laggy players |
|  | Network | Peer-to-Peer solution | Habilitate Peer-to-Peer |
|  | Network | NAT Traversal | Improves connectivity for peer-to-peer connections. |
|  | Util | Local Replay System | Store game states for playback |
|  | Util | Input Validation | Valid inputs verification to prevent cheats |
|  | Sync | Input Delay Buffer | Align frame execution across all players |
|  | Util | Basic cheat detections | Like speed hacks, teleportation or invalid physics |
|  | Util | Logging utilities | Log rollback, frame count, GameState, etc. |
|  | Util | Network Stats | Provide network statistics and log issues |
|  | Network | Dedicated Server Model | Only host-client model |
|  | Network | Cloud matchmaking system | Matchmaking is handled locally via lobbies |
|  | Network | Managed Game Hosting Services | No cloud-based hosting |
|  | Util | Built-in Chat voice | Developers can integrate it using third-party solutions |

* 1. Timeline

# Development

* 1. Project Setup
  2. Core Modules
     1. Networking Manager

- transporte de paquetes según protocolo

- conexión P2P híbrida.

- P2P y pasar a host-client por condición

- selección de host, si falla pasar a uno nuevo

* + 1. State Management

Crear y guardar GameStates. Proveer de clases abstractas/interficies para guardar estados de jugadores y objetos.

* + 1. Rollback System

Guardar GameStates, rollback si se desincroniza, resimulación después del rollback.

* + 1. Input Buffering/Prediction?

Cuando se pierde un paquete(frame) con inputs, predecir o replicar el ultimo input.

Guardar un buffer con los ultimos inputs?

Cuando llega el paquete 🡪 rollback

* + 1. Synchronization

Sistema de estrategias de sincronizacion.

* + 1. Serialization

Encargado de data converter para transporte.

* + 1. Utilities and Debugging

Debug varios: problemas de conexión, rollbacks...Hacer logs de frames o mostrar estadisticas de net?

* 1. Gameplay Framework
     1. Game State
     2. Object State
     3. Object Controller
     4. HUD
  2. UML Design
  3. Implementation

# Conclusions

# Bibliografia/Referències

Text.