.UBAfiuba
FACULTAD DE INGENIERÍA

TALLER DE PROGRAMACIÓN I
(75.42/95.08) VEIGA COURSE

Project Manual

Group 6 - December 9, 2024

Leticia Figueroa	Andrea Figueroa	Josué Martel	Candela Matélica
110510	110450	110696	110641

Introduction

This concise document serves as a quick reference to the team's efforts during the development of this practical assignment. It provides an overview of the objectives, the methodologies employed, and the specific roles and contributions of each team member. Through this manual, we aim to share insights into our planning process, task allocation, and the adjustments made to reconcile our actual progress with the initial goals.

The manual also features a timeline outlining key milestones, along with details about the software and tools that played a crucial role in our success.

In summary, this document reflects on the lessons learned, the skills acquired, and the overall outcomes achieved throughout the project.

Finally, we express our gratitude to those who supported us along the way. This document not only serves as a record of our accomplishments but also stands as a testament to the collaborative effort and dedication of the team.

Practical assignment

First approach

The initial approach to the project's development was centered around dividing it into key areas of work. We focused on the distinct responsibilities we could take on in relation to the various sections of the project. We identified three main categories to organize our tasks, by mutual agreement, the team members were assigned to the categories as follows:

- **Protocol Development**: Design and implementation of communication between the client and server, as well as the creation of the encoding required for sending and receiving messages across the different programs.

  Assigned to: Candela Matélica

- **Rendering (client-side)**: Creation and optimization of the graphical interface and visualization components.

  Assigned to: Josué Martel

- **Server Development**: Construction of the backend logic, resource management and Game logic design and implementation.

  Assigned to: Andrea Figueroa and Leticia Figueroa

Goals

In relation to the objectives set throughout the weeks, we can identify three key points where we synchronized our individual goals to achieve the following group objectives:

- **First week**: familiarization with the requested instruction and tools that we will use. Agreements on schedules, goals, distributions and communication platforms.
- **From the second week to the third week**: The basic functionality of a 'simple' game, where a player can join, visualize the game world, and interact with it, is implemented. It includes support for basic actions such as jumping, as well as demonstrating the collision system in action, with the results reflected in the graphics rendered by the client based on the received messages.
- **From the fourth week to the fifth week**: New features are added, such as support for multiple players in a single match, and additional functionalities are developed within the game model, including the ability to shoot and interact more complexly with other objects and entities in the game world. Furthermore, certain values are parameterized through YAML files, allowing for easy modifications to the game.
- **From the fifth week to the seventh week**: Map Editor Implementation. The complete system implementation is finalized to support multiple users, allowing them to join existing games or create new ones. A single program now supports two players in the same game. The refactoring phase begins, along with the addition of new features to enhance the user experience, including a wider variety of weapons, new entities, and secondary abilities. An orderly server shutdown is implemented with no memory leaks, and corrections made in the first delivery are incorporated. Drafting of required documentation.

Weekly progress

Under the previous distribution (Practical Assignment), we present the starting points from which we developed the foundations. From the second week onwards, a partial reallocation of responsibilities can be observed in order to optimize our progress more effectively. The progress made week by week was as follows:

First week

Member	15/10 - 22/10
Leticia Figueroa	• Implementation of the first approach of the multi-threaded architecture of the server. • Configuring the development environment • Background research and previous projects (Server side)
Josué Martel	• Dependencies and documentation management for integration into the project • Familiarization with rendering tools • Concept tests with SDL library
Andrea Figueroa	• Background research and previous projects (Server side). • Design and implementation of the first approach of the multithreaded architecture of the server: Acceptor thread, a processor thread, a pair of communication threads for each client.
Candela Matélica	• Initial implementation of inter-Process communication messages (client side) • Initial message sketching between processes (server side)

Table 1: Practical development, first week

Second week

Member	22/10 - 29/10
Leticia Figueroa	• Server-side parallelizable division of tasks. • First design of the business logic (duck game) • Diagramming of the participating understandings of the Model • Design of the interface that interacts with server-multithreading
Josué Martel	• Representation of world positions and collisions • World position to rendering view by Camera • Sprites integration
Andrea Figueroa	• Server-side parallelizable division of tasks. • Implementation of the multithreaded architecture of the server. A main Match multiplayer, server without multiple matches.
Candela Matélica	• Create a basic design of the map, including its dimensions, and main elements. • Define the structures needed to represent the map elements, such as points, polygons, or routes.

Table 2: Practical development, second week

Third week

Member	29/10 - 05/11
Leticia Figueroa	• First implementation of collision system (Basic map implementation). • Implementation of check automation for all future physical entities • Design of the interface that the map must provide to interact consistently with other entities. • Support to tolerate continuous and discrete movements (key-up and key-down).
Josué Martel	• Utilities for jump mechanic • Map blocks rendering • Duck sprite animation
Andrea Figueroa	• Design and implementation of concrete data transfer objects for the messages during a match. • Mechanisms to communicate different types of messages match to sender (Server-side). • Implementation of basic protocol assistant and the server protocol considering a single match on the server.
Candela Matélica	• Introduced a YAML configuration file for defining map elements such as terrains, objects, and metadata. • Integrated a YAML parser to dynamically load map data into the application at runtime.

Table 3: Practical development, third week

Fourth week

Member	05/11 - 12/11
Leticia Figueroa	• Use of mechanics designed for jumps. • Correction of details in the interaction with the map, both on the dynamic world side and within the map itself. • Addition of different player states • Basic implementation of a test suite to verify proper functioning when simulating commands sent by the client. • Design and implementation of the API used by the client to interact with the server, including queue and thread management.(client side) • Implementation of the snapshot capture system for player events (server side)
Josué Martel	• GUI Implementation • Initial screens implementation • Initial cache implementation
Andrea Figueroa	• Implementation of client protocol, generic receive and common interface 'NetworkMssg' for different messages. • Automation of execution of pre-commit hooks in GitHub actions.
Candela Matélica	• Create configuration file. Duck configuration and weapon configuration. • Implement the class to read these files and get the information.

Table 4: Practical development, fourth week

Fifth week

Member	12/11 - 19/11
Leticia Figueroa	• Support for the duck's receipt of damage • Support for firing a weapon • Extension of the snapshot capture system for weapon events (server side) • Setting player states with different priorities • First implementation of a new message system to receive information from the multiple available items • Support for basic weapons.
Josué Martel	• Runtime Audio playing implementation • Initial weapon and bullet rendering • Front-end 2 local players support
Andrea Figueroa	• Reassessment and rethinking of pending tasks to be carried out as a group. • Design and implementation of server support for multiple matches and main monitor for them. • Design and final implementation of the communication flow during binding with a match for both parties.
Candela Matélica	• Added edge constraints to define the boundaries of the map. • Implemented logic to validate ground elements' dimensions against the minimum size requirement before rendering. • The new configuration approach provides centralized control over game settings.

Table 5: Practical development, fifth week

Sixth week

Member	19/11 - 26/11
Leticia Figueroa	• Extension of in-game functionalities (new events related to new weapons, loking up, and weapons with bouncing projectiles). • Support for collecting and dropping collectables on the map • Support for initial collectables swpawning on the map. • Snapshot system extension (for spawn events).
Josué Martel	• Some in-game transition effects added • More weapons and bullets rendering • More screens and panels added
Andrea Figueroa	• Design and implementation of multiple local players connected through a client: restructuring a match and other mechanisms based on 1 client-1 player behavior. • Implementation of the script to install the dependencies.
Candela Matélica	• Design a class to update or create the map file (editor side). • Implement this class to write ground information, the theme, player, and collectable spawn points.

Table 6: Practical development, sixth week

Seventh week

Member	26/11 - 03/12
Leticia Figueroa	• Support for player accessories (armor and chest). • support for random spwan points in the game. • Support for the performance of surprise boxes in the game. • Support for complex weapons (like bananas and grenades).
Josué Martel	• Items and defense rendering • Front-end additional mechanics support • Code refactoring
Andrea Figueroa	• Design and writing of the user manual and technical documentation. • Implementation of the final installer of the game. • Implementation of the closure of the server: killing all the threads and cleaning-up used resources.
Candela Matélica	• Implement the editor screen, levels option screen, or create level screen, and the world to be edited. • Refactor static map, configuration files, and editor to include box spawn points.

Table 7: Practical development, seventh week

Employed software

Employed for administration

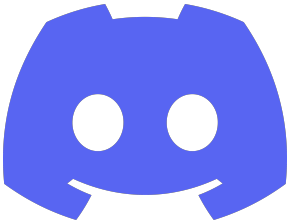
Initially, the organization agreed upon by consensus involved scheduling four weekly meetings, almost on alternating days.

These meetings were planned using Google Calendar (scheduling meetings on the Google Meet platform), ensuring efficient and clear time management. The meetings conducted through this medium were short in duration and held in the morning hours (from 7:00 to 7:40 am). For communication sessions focused on more complex and less administrative topics, the Discord platform was used.

A constant flow of communication, as implemented, facilitates team synchronization, improves task coordination, and ensures continuous progress monitoring. Furthermore, it reduces the likelihood of misunderstandings, fosters a collaborative environment, and allows obstacles to be resolved more quickly, thereby optimizing project progress.

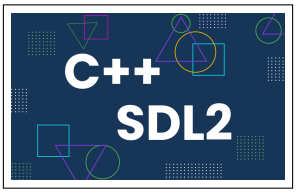
Employed for communication

Most of the communication was conducted through unscheduled meetings, which were consensually agreed upon using the Discord platform. Additionally, a Slack workspace was utilized to allow each team member to document their progress using channels specifically dedicated to this purpose. Furthermore, a system was established to officially record other relevant topics, which were documented in specific channels within the same Slack workspace.



Employed for software development

The development of the game's graphics and menu rendering was carried out using SDL2, along with the libSDL2pp C++ wrapper. This setup was utilized for both the implementation of the graphics engine and the management of the audio engine. We also used Yaml-cpp (version 0.8) to enable accessible configuration of the game's parameters through such files. Additionally, this library significantly facilitated the parsing process.



The methods used to streamline dependency installation and implement the necessary automations, which enhance the user experience when interacting with our game, are based on shell scripting. On the other hand, some patterns used as inspiration in the implementation of the video game were provided by the course (see web for patterns).

Finally, for the modular development of the project, we followed the guidelines recommended by the course regarding the use of descriptive commits.

Lessons learned and skills acquired

The project presented several challenges, especially in terms of synchronization and workflow management. Integrating various functionalities, especially with a team of different levels of experience in technologies previously unknown to us, required meticulous organization. One of the most important lessons learned was the significance of developing the project in layers, which allowed us to set clear and simple goals for each stage. These initial goals were established as fixed targets for specific deadlines, enabling us to progressively advance and expand the project's features as prior objectives were met. This modular approach not only facilitated task management but also taught us to handle complexity in a controlled manner.

Furthermore, we encountered critical points where, although certain problems were already documented within the development community, we needed to research and apply specific solutions for our context. Solving these issues required not only finding technical solutions but also a deep understanding of the systems we were building, which led us to explore more advanced technical concepts related to real-time synchronization and network architecture—fundamental aspects for game development.

Managing priorities was another crucial factor for the success of the project. As a team, we had to establish clear priorities about which tasks should be tackled first, considering deadlines and the interdependence between different game modules. This prioritization not only allowed us to optimize development time but also prevented us from getting distracted by smaller details while working toward the most important goals.

Throughout the process, the team's working dynamic had to be continuously adjusted to ensure cohesion and synchronization. Periodic meetings and constant updates were essential to keep everyone aligned and ensure that the different functionalities developed could be integrated and tested on time. This iterative and collaborative approach was key to completing the project successfully, and it left us with valuable lessons about the importance of organization, effective communication, and strategic planning in video game development.