Meal Voucher

(Middle 2022 - Now)

Android App – Front End – Java

This is an Android app for Smart Terminals. The app allows to pay via meal vouchers of any emitter.

This is the first project I started working on, and it is still ongoing. The application is already live, available to all merchants with access to smart terminals or smart POS, devices running on the Android operating system. I work independently on this project, especially when the addition of new features is requested.

New knowledge acquired:

- Java, Kotlin
- Android
- Architectural Pattern (MVVM, Observer)
- Android Navigation

Nexi Payment App

(Middle 2022 - Now)

Android App – Front and Back End – Java

Another Android app for Smart Terminals. The app allows to pay via credit cards and different acquisition methods (chip, contactless, manual, etc.)

This is another app for smart POS, widely used by merchants to facilitate credit card payments. I work on this project with a team of 5 people, responsible for developing new features and bug fixing.

New knowledge acquired:

- Team work
- Code optimization
- Dagger, Retrofit, Android Shared Preferences
- Git, Git flow
- Jira ticketing, Bitbucket
- Debug large code base

What I learned:

- Working in a team has allowed me to learn how to interact with people, organize my work, and meet deadlines.
- Writing readable and optimized code, as the app is in constant development.

Foxy Jumper

(Middle 2023) – Sole developer

Endless scroller – 2D - Web Game - Unity - C#

An endless scrolling jumper, where you control a fox and have to jump over obstacles to survive. Stay alive for as long as you can to score the highest result.

This was my first personal project, developed to start approaching the Unity engine and game mechanics. I decided to start with a very simple game, which nevertheless consists of essential components for the development of a video game.

New knowledge acquired:

- Unity, C#
- 3D math, quaternions, vector operations
- Architectural Pattern (Singleton, Observer)
- Events and Delegates
- Coroutines
- Physics and trigonometry for games
- Raycast
- Animations and Animator
- VFX Particle System
- PlayerPrefs
- Debugging, both from VisualStudio and writing debug code or drawing debug lines

What I learned:

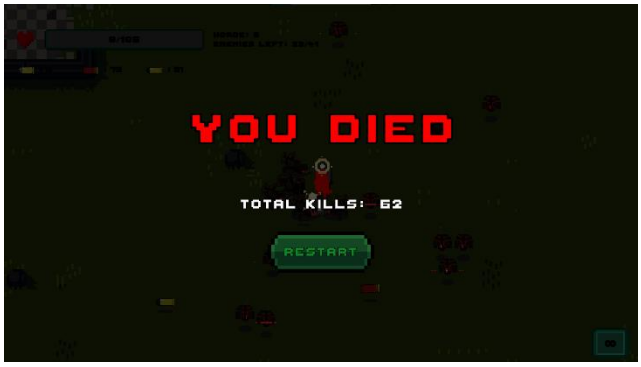
- Confidence with the Unity editor, starting from the object hierarchy in the scene, to the attributes of each gameobject and each component that can be added. Use of SerializableFields assignable by editor.
- Study of the gameobject lifecycle, and use of some of its methods: Awake, Update, FixedUpdate, Start, etc.
- Use of BoxCollider, RigidBody and raycast to detect collisions and obstacle crossings in the scene.
- Use of an EventHandler and delegates to manage communication and data exchange between the game and the UI.

Darkwave Rising

(October 2023 – July 2024)

Roguelite Horde Shooter Game – Unity

Survive as long as possible on a post-apocalyptic island, defeating enemies with different weapons and upgrading your skills.



I developed this game to continue practicing with Unity, creating something more complex and advanced compared to my first one. This project allowed me to deepen my understanding of game development principles and enhance my skills in areas such as **AI implementation**, **optimization** and **state management**.

New knowledge acquired:

- Architectural Pattern (Strategy, MVC, Object Pool)
- Scriptable Objects
- Enemy AI implementation
- Damage and damageable system
- Weapon modularity
- Pathfinding algorithm (A* Path)
- FSM implementation to control characters state
- Audio system
- Tween for animations
- Vector operations to compute correct directions

What I learned:

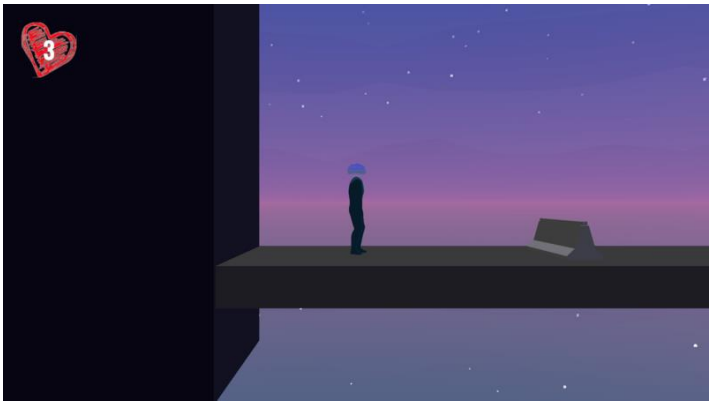
- Code optimization using **Object Pool**, to avoid instantiating game objects repeatedly and reuse them instead.
- Creation of an optimized **Damage System**: by implementing the **IDamager** and **IDamageable** interfaces, damage can be applied to damageable objects or taken by entities. These methods exhibit different behaviors depending on the character that implements them.
- Modularity and Scalability of the **Weapon System:** to add a new weapon to the game, simply create a new Scriptable object and pass it to a **Weapon Gateway**. The gateway provides the necessary information to correctly instantiate the weapon at runtime, ensuring a flexible and scalable weapon management system.
- Basic implementation of **enemy AI**: using the A* pathfinding algorithm, melee enemies follow the player and perform melee attacks; ranged enemies follow the player and shoot, ensuring there are no obstacles in the line of fire.
- Player, enemies, and weapons have an associated **FSM written in C#**, which controls their states during the game and allows specific animations to be executed based on these states.
- Use of **DoTween** engine for smooth screen and text animations implemented directly in C#

Road to Happiness

(26-28 January 2024)

Adventure Game – Unity 3D – Global Game Jam 2024

Immerse yourself in the emotional journey of battling with depression: confront your fears and challenges, symbolized by various difficult obstacles and enemies.



This videogame was developed in 2 days for the Global Game Jam 2024, in which I participated at **Event Horizon School** in Milan. The development team consisted of myself and two other colleagues, and we divided the work into various tasks.

What I worked on:

- Implementing the audio system, from sound effects to theme music
- Implementing player and enemy shooting mechanics
- Obstacle movement within the environment.

New skills acquired:

- Unity 3D
- Game creation and release pipeline.

Unreal Engine 5 C++ Game Developer Course

(Middle 2023 – In progress)

Unreal Engine 5 – 3D – RPG

This is a detailed course tailored for Game Developers, that teaches how to develop a 3D Action-RPG Open World game.

New knowledge acquired:

- Unreal Engine 5, C++
- Blueprints, Events
- Animation Blueprint, State machine, Blend space
- Meshes, Materials
- Lights: Lumen, Skylight, Directional light
- Visual Effects: Sky atmosphere, Exponential Height Fog
- Volumes: PostProcess Volume, NavMeshBounds Volume
- Foliage
- AI: Pawn Sensing, AI Perception
- Input: InputMapping Context, Input Actions