## **Overview**

In this project, we aim to build a mission-based 2D shoot-and-run game using C and C++. Our journey started with researching open-source 2D games like *Megaman*, *Contra*, and *Metal Slug*. After spending countless hours cloning, debugging, and executing various repositories, we found a promising one with well-structured character movement, screen transitions, and a solid asset collection.

To run the exe file: Please open the project and run***bin/Debug/Extinction.exe***

We have reorganized the project into three main folders:

* **data/**: Contains all assets and documents needed by the game. Currently, it includes placeholder assets from *Metal Slug* and *Call of Duty*, which will be replaced with our own artwork.
* **include/**: Stores all libraries, header files, and SDL dependencies.
* **src/**: Houses the entire source code.

## **Core Development Areas**

### **1. Screen Development**

* Splash Screen - Done
* Main Menu - Done
* Mission Announcement Screen - Done
* NPC Communication Screen - In progress
* Win/Lose Screen - Done
* Main Game Screen - Done

### **2. Hero Development**

* Hero Movement and Controls - Done
* Point Management
* Health Management - Done
* Additional Features: Super Jumping, Sliding Movements - In progress

### **3. Enemy AI Development**

We plan to implement four types of enemies:

* Normal Soldiers - Done
* Mini-Boss - In progress
* Moving Vehicles - Done
* Flying Jets - Done
* Moving Traps & Obstacles - Done
* Boss: **Dragon – 90% done**
* Bullet and Bomb Mechanics - Done

### **4. Other Functionalities**

* Clock Functionality - Not started
* Sound Effects & Background Music - Done
* Save & Load Game Progress - Done

**Core Source code description**

* **Main.cpp:** This is the entry point which will initialize the SDL and create an object for the game class.
* **Game.cpp:** This is now the file handling setting the stage for GameScreen.cpp, which I have tried to manage through 1000px per mission to get into the next mission screen and it’s crucial to find the exact pixel so that the actual background changes when the player progresses to the new mission, this has still some flaw but still waiting for the actual background for me to get the exact pixel coordinates and set the algorithm on the top it . It handles also the game logic which first swapn 12 enemies tracking number of enemies generated, bullet on the screen, collision removing the sprites and to make it efficient and less ram intensive I am trying to slow the sprite frame rendering, it seem now a bit slower but once I get the right numbers it will work fine. Additionally there is a tank, which is working flawlessly, and my own dragon(still far to go) which will come from the right side of the screen and release its attack. The next set is a helicopter with simple horizontal navigation and dropping bombs, nothing complex here and and finally there will be a big jet appearing in the middle dropping huge bombs which deduct the user life by 30%. The final enemy is the dragon boss. The object manager will handle the object creation, mapping, string comparison and decide to delete the killed sprites from the memory in real time, the event controller handles the button and the stage setting for the game.
* **Gamescreen.cpp:** This is a code that manages the horizontal movement and screen scrolling. This was the code we got from the repo and the only new thing I added here is creating a stage setter function which dynamically set the screen pixels and track that instead of managing it manually and the current screen staging is only handled by the Game logic.
* **Sprites.cpp:** This is the code that loads images of the sprites and iterates over them in a loop. The current sprite management is “mhhh” thing as you clearly see pixelation. We are trying our best to get the perfect texture loop as the current assets are just for the demo. We will try to make it pixel perfect as we get our real resources.
* **Hero.cpp:** This is one of the other files we found on the repo and we are adding what we want on the top of it. On the assets we have 3 types of sprites for the hero, standing, running, and firing and the movement and its related tracking was done. Only the keys are changed, nothing more this week.
* **Health.cpp:** This is a simple code that will check the hero health count and change the sprite of the health manager.
* **Enemies**: We got a simple but good enemy structure for the soldiers that needs to be worked on. Currently implementing a tank(name and image to be changed) it's a simple code which will move in the x direction with collusion impact and front firing. I have implemented the helicopter with horizontal movement on the x axis and keep the y distance constant with the value set by the position setter, which literally does is count the frames and pick the top frame and stay there. It was a clean way of implementing it 🙂. The other thing is obstacles which are sprites which appear between mission 1 and the user can jump and stand on it but when the tank comes and creates the collision it will blow up creating a huge deduction in the user's health.
* **Base.cpp:** This is how the dragon initially draws its position toward the hero.
* **Terrain.cpp** - This is the actual code that handles the effect of dragon attacks, which was aimed to have three, laser lease, bomb dropping and missile attacks and now only the missile is working. It uses the base tracking and launches the missile projectile.
* **Dragon.cpp(my own game physics)** - This code starts by attaching the head with the tail since it started as two sprites and the tails are also missiles to be launched. It tracks the last position of the head sprite and it multiplies the tail segment by a constant with a given space. I have also implemented a class that tracks the navigation of the dragon and the stinging effect by using angels. What it does is the head focus on following the hero sprite and the tail focus on attaching itself to the head and at attack the tail will move toward the hero sprite. Then there is the actual dragon which will try to act based on a condition deciding how to move, what type of attack to release from dropping bombs, laser attack and tail missile string, currently only the missile attack is working amd after the attack it will reset the tail movement so that it attaches itself back to the head.
* **LoadAndSave.cpp:** This is the file that will track the constants and function returns and also the screen position into a text file and use it as initiator when it is started.