# Game Design

# CONTRA


PROJECT BY:

NIRMAY NANDURKAR (23MM01016)

ANSH GUPTA (23MM02002)

ARCHIT SHARMA (23MM02003)

UTKARSH MALHOTRA (23MM01025)

# CONTENTS

# INTRODUCTION

Our team has created a game that brings back memories of playing **CONTRA** during our childhood.

From playing our favorite arcade games to developing one of our own, we have come a long way in our lives.

We've added lots of cool features to make our game stand out and look amazing.

# MOTIVATION

In today's fast-paced world, many of us lead hectic lives filled with stress from our daily routines. Taking a break to relax and unwind is essential for maintaining our well-being.

In the case of student life, it is common for us to play video games in our free time to relax. So, it gave our team an idea of creating what we like as an opportunity given to us in the form of PDS project.

Also, by doing this project we improved our understanding of C-language and developing logical reasoning which will help us learn other programming languages in the future.

Although being beginners in this field, this project gave us an idea about game development. Being students and diving into a new field simultaneously has been our main motivation for taking on this project.

# DEVLOPMENT PROCESS

At the beginning our team did some research on what concepts would be required to develop what we are aspiring to. Thats when we came to know about SDL and without wasting any time, we started learning it. Utilizing the concepts learned in C-language was another challenge we had to face. Our team faced some difficulties during the initial phases of the project as we were doing these kinds of things for the first time. But as we continued working on it, we were able to understand our mistakes and could ideate further code logic.

# ABSTRACT

***1)Sound Effect***: The provided code implements sound effects using the SDL_mixer library in a game scenario. It loads and plays sound files for various game events, such as shooting bullets and game over conditions. Sound effects enhance the gaming experience by providing auditory feedback to the player.

***2)Bullets***: The program features a bullet system where the player can shoot bullets using the spacebar key. Bullets are represented as small rectangular objects on the screen and can move horizontally across the game window. Collision detection is implemented to detect when bullets hit the enemy character or the player, affecting

their health or triggering game over conditions.

***3)Health Bar***: The game incorporates health bars to represent the health status of both the player character and the enemy character. These health bars dynamically update based on the damage inflicted by bullets. The health bars visually indicate the remaining health of each character, allowing the player to assess their status during gameplay.

***4)Keyboard Bindings***: Keyboard bindings are utilized to enable player interaction with the game. The program captures keyboard input events using SDL events, allowing the player to control the player character's movement, shooting, and

other actions. Specific keys are bound to different game actions, such as arrow keys for movement and the spacebar for shooting bullets.

***5)Start and End Screen***: The game includes start and end screens to provide an introduction and conclusion to the gaming experience. These screens display static images using SDL_image library and wait for user input to proceed. The start screen appears at the beginning of the game, prompting the player to start the game by pressing any key. The end screens are displayed upon winning or losing the game, allowing the player to exit the game or start a new game session.

***6)Enemy Character***: The program features an enemy character controlled by AI algorithms. The enemy character moves randomly within the game window and shoots bullets towards the player character at regular intervals. The enemy character's behavior is governed by game logic implemented in the updateLogic() function, providing an interactive challenge for the player to overcome during gameplay.

# FUNCTION DESCRIPTION

The code for the game involved use of many functions, each of which were responsible for a certain task which would help run the game accordingly.

***1)showStartScreen(SDL_Renderer *renderer):***

- This function displays the start screen image on the window.
- It loads the start screen image from a file using SDL_image.
- It waits for the user to press any key to continue.

***2)showWinScreen(SDL_Renderer *renderer):***

- This function displays the win screen image on the window.

- It loads the win screen image from a file using SDL_image.
- It waits for the user to press the enter key to exit the game.

***3)showLossScreen(SDL_Renderer *renderer)***:

- This function displays the loss screen image on the window.
- It loads the loss screen image from a file using SDL_image.

***4)addBullet(float x, float y, float dx):***

- This function adds a bullet to the bullets array.
- It finds an empty slot in the bullets array and initializes a new bullet with the given position and velocity.

***5)removeBullet(int i)***:

• This function removes a bullet from the bullets array.
• It deallocates memory for the bullet at the specified index in the bullets array.

***6)addEnemyBullet(float x, float y, float dx, float dy):***

• This function adds an enemy bullet to the enemyBullets array.
• It finds an empty slot in the enemyBullets array and initializes a new enemy bullet with the given position and velocity.

***7)renderEnemyBullets(SDL_Renderer *renderer):***

• This function renders the enemy bullets on the screen.
• It loads the enemy bullet image from a file using SDL_image.

• It iterates through the enemyBullets array and renders each active enemy bullet on the screen.

***8)updateEnemyBullets():***

• This function updates the position of enemy bullets.
• It iterates through the enemyBullets array and updates the position of each active enemy bullet.

***9)processEvents(SDL_Window *window, Man *man):***

• This function processes SDL events such as key presses and window events.
• It updates the state of the **man** object based on user input.

***10)doRender(SDL_Renderer *renderer, Man *man):***

• This function renders the game objects on the screen.
• It clears the renderer, renders the background, player, enemy, bullets, and enemy bullets.

***11)checkBulletPlayerCollision(Man *man):***

• This function checks for collisions between player and enemy bullets.
• It updates the player's health and removes enemy bullets upon collision.

***12)updateLogic(Man *man):***

• This function updates the game logic, including player and enemy movement, bullet movement, collision detection, and enemy behavior.
• It updates the position of the player character and checks for collisions with enemy bullets.

***13)main(int argc, char *argv[]):***

- This is the main function of the program.
- It initializes SDL, creates the game window and renderer, and loads game assets (textures, sounds).
- It contains the game loop, which processes events, updates logic, renders graphics, and manages game state.
- It cleans up resources and exits the program when the game is over.

# WHAT WE LEARNT

We had some familiarity with certain ideas before starting our project, like

pointers, structures, and functions. However, working with SDL (Simple Direct-Media Layer) was new to us, so we had to learn how to use its libraries and functions. Even though we knew the basics of concepts like pointers and structures, applying them in a larger project required us to think about them in different ways than we were used to, making it a bit challenging.

# AREAS OF IMPROVEMENT

- *2-player game*: The game design our team has created is basically a single player, that is, 1 user can play the game with the computer being the opponent. To make the game more interesting and competitive, it can be made a 2-player game. The following task will require the code to function 2 consoles at the same time.

- *Addition of multiple levels*: Our game currently only consists of a single level. To make the game more challenging, multiple levels can be added to our game just like TEKKEN(another popular videogame). This can occur by increasing the frequency of bullets and movement of enemy for each level.

# CONCLUSION

Overall, it was a gratifying experience working on this project. We learned many things while working on it. This project gave us our first experience of game development, improved our understanding in C-language and taught us how to tackle errors. This fun experience will surely help us learning new programming languages and exploring different fields in future.