University of Dhaka
Department of Computer Science and Engineering

Project Report:

Fundamentals of Programming Lab (CSE - 1211)

Project Name:

Agent Mark

Team Members:

Mohd. Jamal Uddin Mallick Tahmid

Roll - 07

Registration no - 2020615617

Abdullah Al Tanzim

Roll - 21

Registration no - 2020015631

Farhan Ibn Shahid

Roll - 56

Registration no - 2020215666

Introduction :

‘AGENT MARK’ is a SDL based running game written in C++. The code is written in a simple, modular, understandable way with the advantage of updating and customizing the code easily. The game is based on the idea that an enemy is chasing the ‘Agent’ named Mark and also the road he has to travel is not so smooth . So, he must outrun the enemy as long as possible overcoming all the obstacles along the way in order to survive.

Objectives :

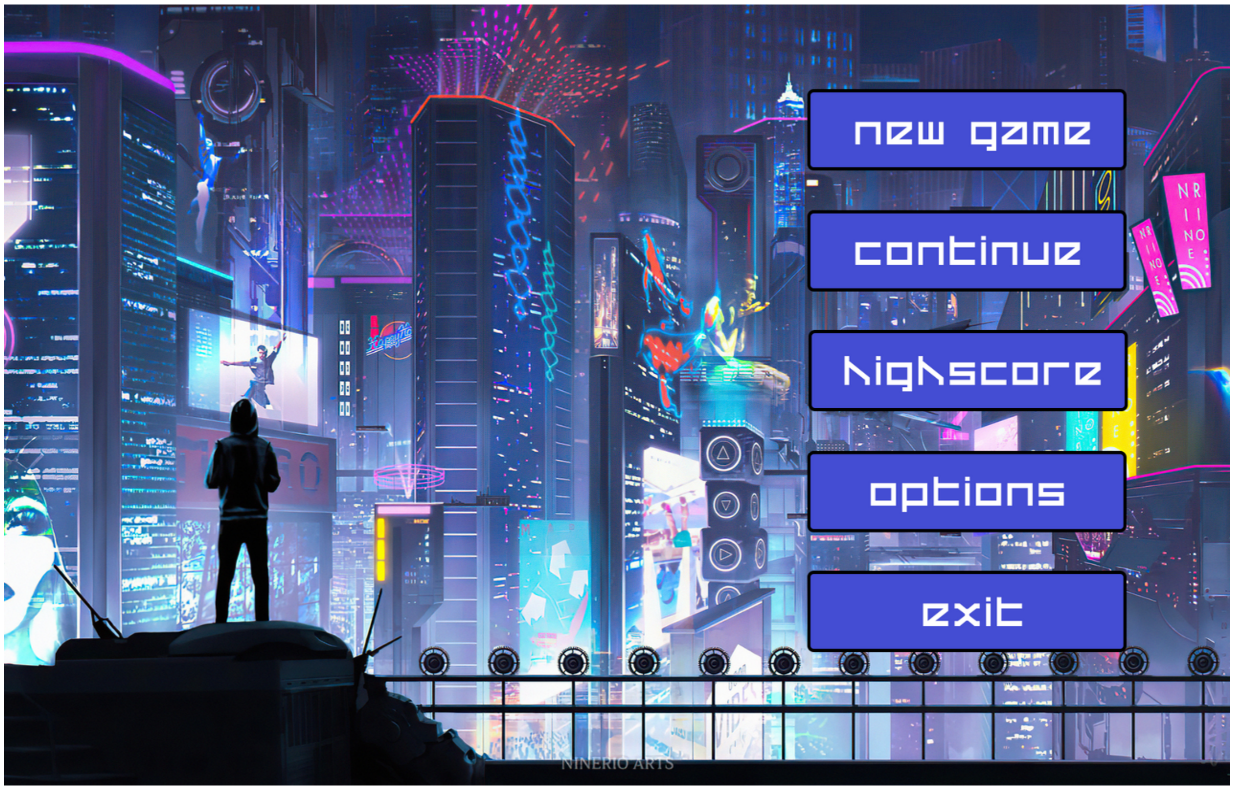
1. To apply the **structured C++ language** and make a **real life project**.
2. Make the game attractive for the gamer with **graphics and designing.**
3. **Addictive game** with extra features(**extra points, lifeline, free run**) and **increase of difficulty** with time.
4. Easily **customizable code** and room for **further development**.

Project Features :

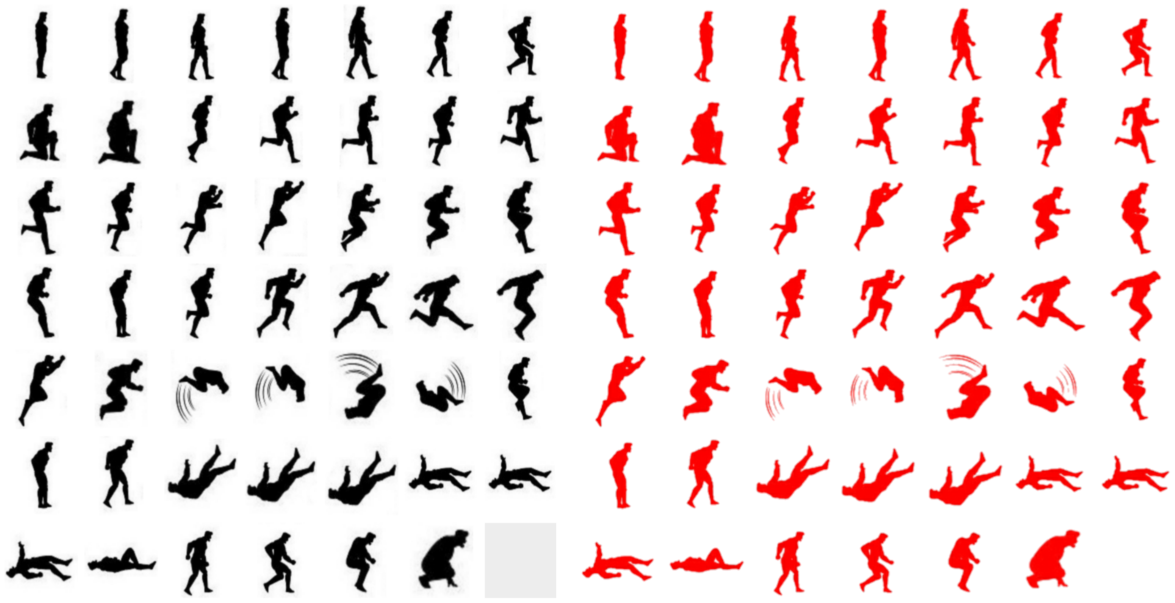
1. Simple ,easily customizable code written in a modularized way with comments wherever necessary.

2. A loading page with game name.

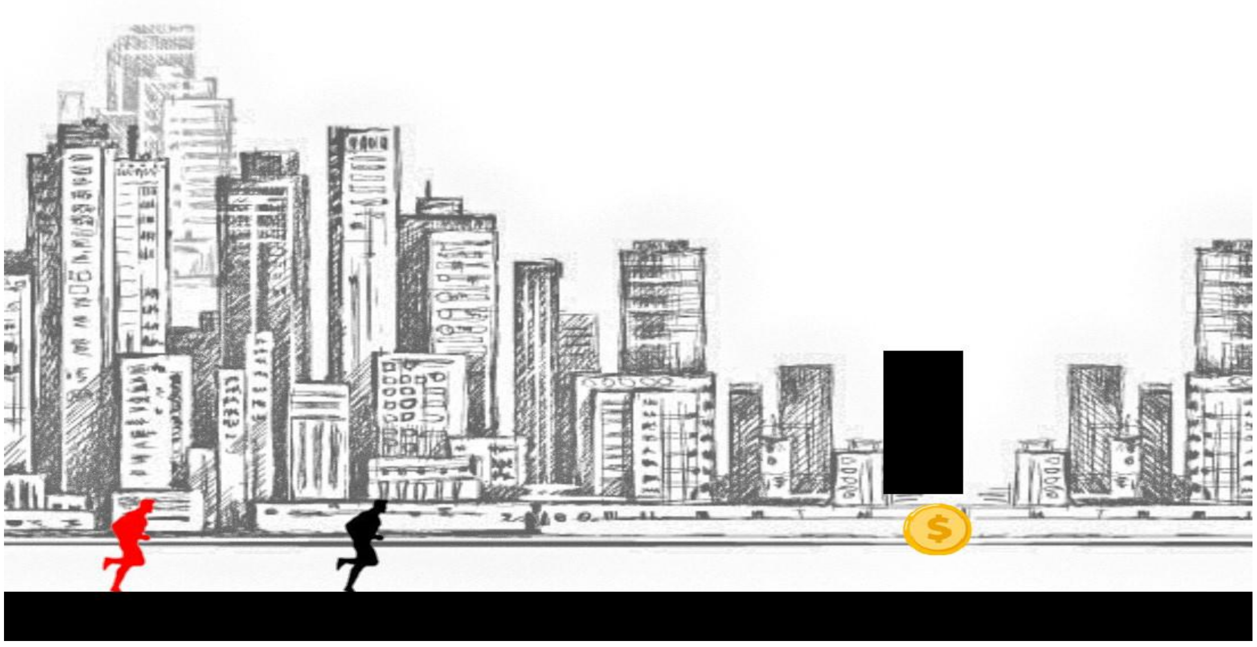
3. Attractive menu options to control the game.



4. Using sprite animation to make real life looking jump, slide, running possible.



5. Possibility of acquiring extra points by getting the coins.



6. Possibility of gaining extra life .
7. Increase of speed and frequency of obstacles with time.

Project Modules :

1. **<u>RenderWindow.hpp :</u>**

   Include the declaration of all the functions to create a window, render different objects on the screen and also clear the window. The functions declared in this header file are -

   RenderWindow() - creates a new window and renderer.
   loadTexture() - create a surface with an image, loads the surface into a texture and returns the texture pointer.
   TextLoad() - loads the text to a texture with specific font and color and returns the texture.
   cleanup() - destroys a window and renderer.
   clearScreen() - clears a renderer.
   changeRendererColor() - draws a renderer with specific colors.
   render() - renders the runner in different states.
   renderBG() - renders the background while running the game.
   random() - generates a random number in some range.
   renderObstacle() - renders the obstacles.
   renderlifeline() - renders lifeline.
   rendercoin() - renders coin.
   renderfreerun() - renders free run image
   display() - presents the render.
   lives_show() - shows the current life.
   score_show()- shows the current score.

2. **<u>Fileio.hpp :</u>**

Includes the functions for loading the states of the game .

write_history() – to write the history of a certain state of the game.
read_history() – to load the game by reading history.

3. **<u>Math.hpp :</u>**
   Consists of a structured named vector to store the x and y axis position of an object.

4. **<u>Entity.cpp :</u>**
   Handles the movement of different objects .

   Entity() – sets the position of an object.
   handleEvent() – handles keyboard operations to move object.
   checkcollision()- checks whether the runner is colliding with an obstacle or not.

5. **<u>Init.hpp :</u>**
   Includes the functions to initialize SDL features.

   initSDL() – initialize audio,video, image features of SDL.
   quit() – quits SDL operations.

6. **<u>Gamefuncs.hpp :</u>**
   Consists of all the necessary functions to score points, update enemy,agent positions, collision checker, render free run and coins.

   init_score_life() – loads the previous score.
   background_scroll() – periodically scrolls the background while running the game.
   select_agent_frame() – controls the jump,slide movement of the agent.

render_agent() - renders the agent object.
render_enemy() - renders the enemy object.
render_obstacle() - controls the position of the obstacles.
update_agent_pos() - does the animation of the running agent.
render_ground() - renders the black box ground.
reset_frame_no() - reset agent frame to initial phase.
collision_checker() - checks whether the agent is colliding with obstacles, enemy, coin, lifelines and manages life points.
render_lifeline() - shows lifeline on the screen.
void render_coin() - shows coin on the screen.
void render_freerun() - shows the free run icon on the screen.
int generate_score() - generate scores with time.

7. **<u>Gamestatus.hpp :</u>**
Contains functions for start_screen ,main_menu, also the most important 'game' function that controls all the pre-declared functions to run the game.

start_screen() - shows the first screen on the window.
main_menu() - loads the main menu window and controls the different options.
countdown() - countdown before starting the game.
game() - calls all the functions to run the game.

8. **<u>Mouse.hpp :</u>**
It has functions, enum to handle mouse events.

handleMouseEvents() - handle mouse events for each button and options of main menu.

9. **<u>Sprites.hpp :</u>**
This header file contains the implementation of the rendering of sprite animation while running, jumping and sliding, also the rendering of the different options of main menu .

10. **Music.hpp :**

Includes the loading of different sound effects at different stages of the game.

loadMedia() - loads all the sound effects and music.
music() - controls when to play the music.

11. **Gameloop.hpp :**

Contains event handling and main game loop function.

Handle_event() - handle event for SDL_quit and handle mouse event in the main menu page.
gameloop() - controls the stages of the game .

Team Member Responsibility :

Mohd. Jamal Uddin Mallick , Roll -07 (2020615617)

- Game design ideation and implementation .
- Design and implementation of the skeleton of the header files.
- Sprite animation , rendering of different states of agent, enemy , background.
- Implementation of obstacles ,enemy collision and score system.
- Handling mouse and keyboard events to control the main menu page .
- Version control (Git/Github) .
- Adding music and sound effects .
- Code debugging and testing .

Abdullah Al Tanzim , Roll - 21 (2020015631)

- Graphics designing of the starting and menu page.
- Load high score and history .
- Game design ideation and implementation.
- Code testing and bug fixing.
- Implementation of the sound option and instructions window.

Farhan Ibn Shahid , Roll - 56 (2020215666)

- Game design ideation and implementation.
- Implementation of jump, slide animation of agent .
- Adding features of Extra coin , lifeline and implementation of free run.
- Implementation of obstacle positioning and occurrence.
- Rendering of various options and states.
- Code running and debugging.

Platform, Library and Tools :

- **C/C++ :** Implementation of the basic code in C++
- **SDL2 :** SDL2 is a cross-platform development library designed to provide low - level access to audio, video, keyboard, mouse , joystick and graphics hardware.
- **VS Code :** Vs code is used to write the code which is a simple but powerful IDE
- **Git/Github :** Using git to store the code online and work in a collective manner.

Limitations :

- Jump and sliding animation can lag sometimes which is a discomfort for the user.
- Since the code is written using SDL2 and C++, no extra features outside these libraries can be used.
- The game is made to run on Ubuntu 22.04.

Future Plan :

“Agent Mark” is a running game with an enemy chasing behind and also many obstacles ahead. The plan is to add extra features and develop the graphic system and also to advance the difficulty level . The game is made to run on Ubuntu 22.04 , which can also be done to run on any OS such as windows/IOS etc.

Conclusions :

The goal of this project was to apply the learnings of C/C++ language to make a simple yet interesting game that anyone can play. We learnt to work collaboratively and also come up with solutions to different problems. The most difficult part of the project was to fix bugs at different stages and implement the ideas using SDL features.

Repositories and Youtube Video :

Github Repository : https://github.com/jamal-uddin5823/SDL2_GameDev

Youtube video : https://youtu.be/Odg1EgkEQXA

References :

- https://lazyfoo.net/ for learning how to use SDL for different purpose.
- https://www.libsdl.org/ for SDL library information.
- https://www.youtube.com/playlist?list=PL2RPjWnJduNmXHRYwdtublIPdlqocBoLS for code blueprint