NOAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY

Noakhali-3814

Computer Science and Telecommunication Engineering

Course Title: Computer Graphics and Animation Lab

Course Code: CSTE – 4102

Lab Report: Computer Graphics Lab

Submitted By:	**Submitted To:**
Name : Farman Arefin Tamim Roll : ASH2001043M Year: 04; Term: 01 Department of CSTE NSTU	Md. Kamal Uddin Assistant professor Department of CSTE NSTU

***Date of submission* : 22.12.2024**

Space Shooter OpenGL Project

The Animation Project Overview

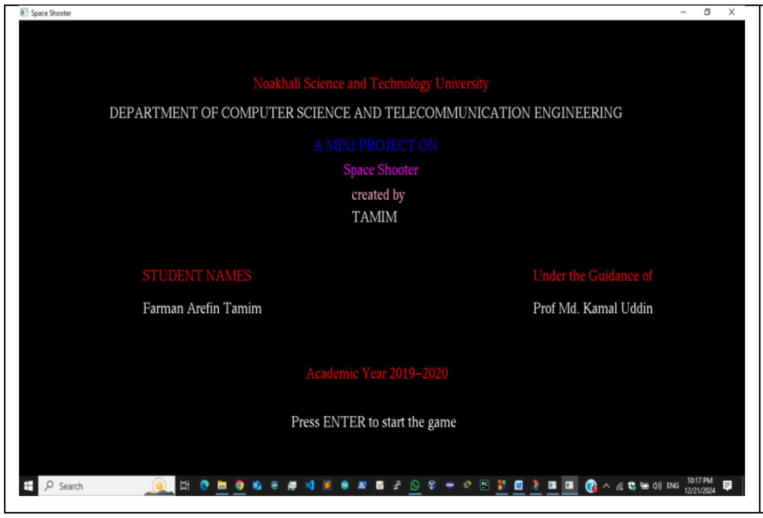
The animation project is about a **Space Shooter game**, where the player controls a spaceship to fight against enemies and avoid obstacles. The animation sequence is designed with the following steps:

- **Storyboard Layout**
- **Object Definitions**
- **Key-frame Specification**
- **Generation of In-between Frames**

Storyboard Layout

The storyboard layout determines how the game transitions between its different screens or states. These include the **Main Menu**, **Gameplay**, and **Game Over** states. The transitions and gameplay are structured as follows:

1. **Main Menu:**
   - The main menu is the starting screen of the game, containing options such as:
     - "Start Game"
     - "Instructions"
     - "Quit"
   - The player can select an option using keyboard inputs. Typing ENTER begins the gameplay.



Space Shooter

Noakhali Science and Technology University

DEPARTMENT OF COMPUTER SCIENCE AND TELECOMMUNICATION ENGINEERING

A MINI PROJECT ON

Space Shooter

created by

TAMIM

STUDENT NAMES

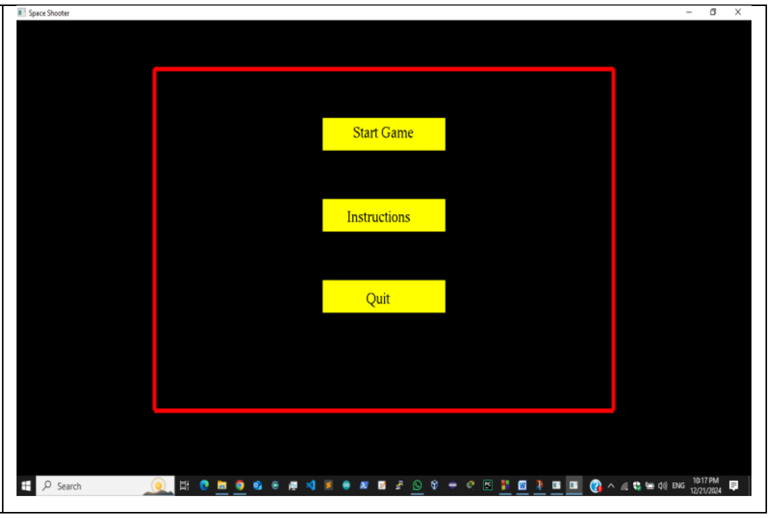
Farman Arefin Tamim

Under the Guidance of

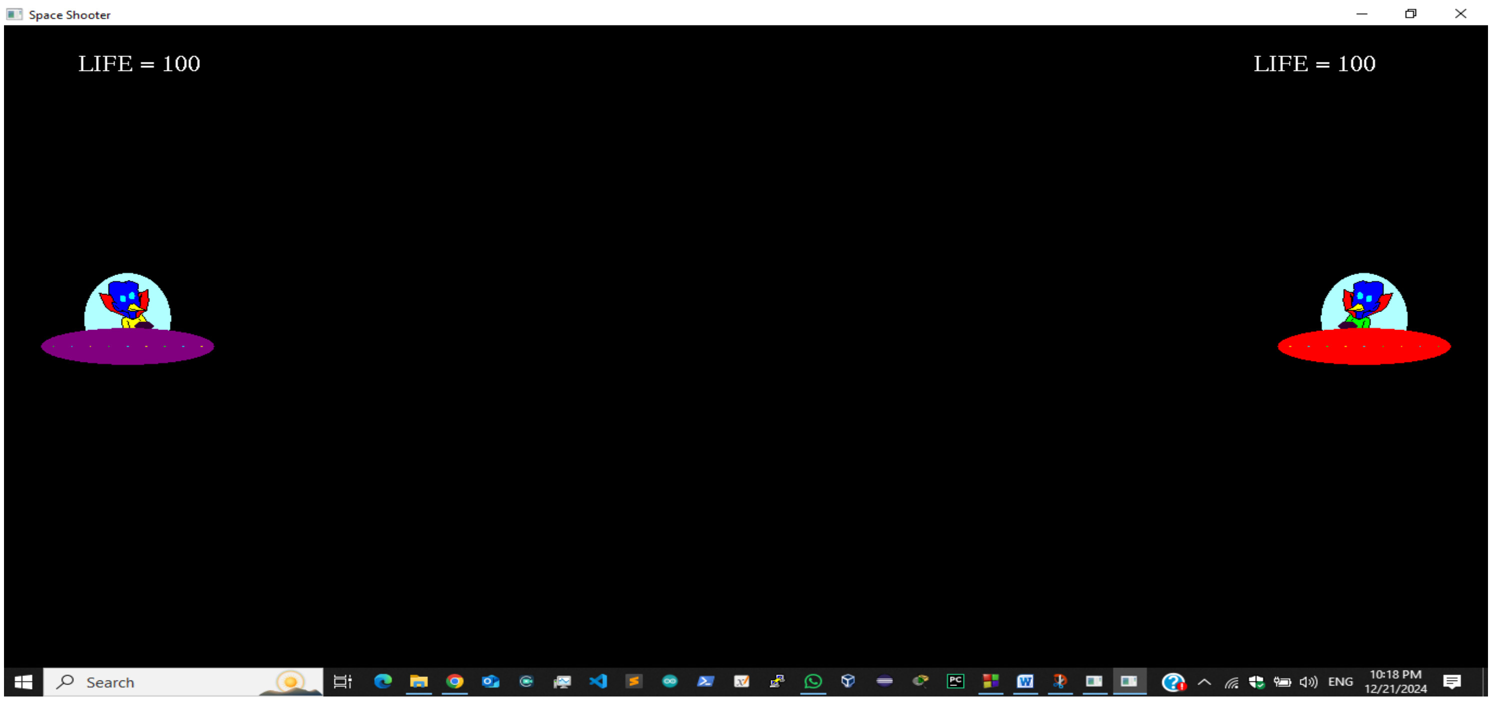
Prof Md. Kamal Uddin

Academic Year 2019–2020

Press ENTER to start the game



2. **Gameplay:**
   - In the gameplay state:
     - The player controls a spaceship that can move up, down, left, and right using keyboard controls.
     - Enemy ships spawn randomly and approach the player's ship. The player must dodge or shoot them.
     - The player earns points for destroying enemy ships.
     - The player loses health if their ship collides with enemies or incoming obstacles.
   - Gameplay ends when the player's health reaches zero or they collide with an obstacle.



3. **Game Over:**
   - A "Game Over" screen displays the final score and provides an option to restart the game or exit.
4. **Transition Logic:**
   - Transitions between these states are managed using flags or variables. For example:
     - When the player's health becomes zero, the game transitions from the gameplay state to the game over state.
     - If the player selects "Quit" from the main menu, the program exits.

Object Definitions

The objects in the game include the **spaceship**, **enemy ships**, **projectiles**, and the **game board**.

1. Spaceship:

- The spaceship is the player-controlled entity.
- **Attributes:**
  - Position: Current coordinates of the spaceship.
  - Speed: Movement speed of the spaceship.
  - Health: Remaining health points of the spaceship.
  - Bullet Positions: Tracks the positions of projectiles fired by the spaceship.
- **Functions:**
  - Move: Updates the spaceship’s position based on keyboard input.
  - Fire: Spawns a projectile at the spaceship’s position.
  - Collide: Checks for collisions with enemy ships or obstacles.

2. Enemy Ships:

- Enemy ships appear randomly on the board and move toward the player’s spaceship.
- **Attributes:**
  - Position: The current position of each enemy ship.
  - Speed: Movement speed of the enemy ship.
  - Health: Determines how many hits it takes to destroy the enemy ship.
- **Functions:**
  - Move: Updates the enemy ship’s position.
  - Collide: Detects collisions with the spaceship or projectiles.

3. Projectiles:

- Projectiles represent the bullets fired by the spaceship and enemies.
- **Attributes:**
  - Position: Current position of the projectile.
  - Speed: The rate at which the projectile moves across the screen.
- **Functions:**
  - Move: Updates the position of the projectile in each frame.

4. Game Board:

- The game board is the background area where gameplay occurs.
- **Attributes:**
    - Dimensions: Width and height of the playable area.
    - Background: Static or scrolling background image.

Key-frame Specification

Key-frames mark significant moments in the game, such as:

1. **Spaceship Movement:**
    - When the player presses a key, a keyframe is set to update the spaceship's position.
    - In-between frames ensure smooth movement across the screen.
2. **Enemy Spawning:**
    - Key-frames mark the moments when enemies spawn on the board.
3. **Projectile Firing:**
    - Key-frames occur when the player fires a projectile, marking the projectile's initial position.
4. **Collisions:**
    - Collisions between the spaceship and enemies, or between projectiles and enemies, trigger key-frames.
5. **State Transitions:**
    - Key-frames define the transitions between states, such as from **Gameplay** to **Game Over**.

Generation of In-Between Frames

Smooth animation is achieved by generating frames between key-frames using interpolation techniques:

1. **Spaceship Movement:**
    - When the player moves the spaceship, in-between frames are calculated to ensure smooth transitions between positions.

2. **Projectile Movement:**
    - Projectile positions are updated in each frame, creating continuous motion.
3. **Enemy Movement:**
    - Enemy ships' positions are interpolated between frames to give the impression of steady movement.
4. **Collision Effects:**
    - Explosion animations or other visual effects are interpolated to make collisions visually appealing.
5. **Game Board Updates:**
    - If the game board has a scrolling background, the background's position is updated in each frame for a dynamic visual effect.

Gameplay Loop

The game loop ensures a consistent frame rate and includes the following steps:

1. **Handle User Input:**
    - Process keyboard inputs for spaceship movement and projectile firing.
2. **Update Game State:**
    - Update the positions of the spaceship, enemy ships, and projectiles.
    - Check for collisions and adjust health or scores accordingly.
3. **Render Frame:**
    - Draw all objects (spaceship, enemies, projectiles) on the screen for the current frame.
4. **Transition States:**
    - If a game-over condition is met, transition to the Game Over state.

By following this structure, the **Space Shooter OpenGL Project** ensures smooth gameplay and visually appealing animations, providing an engaging experience for players. Let me know if you'd like assistance with specific parts, such as OpenGL code examples or game logic implementation!
