NOAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY

Noakhali-3814

Department: Computer Science and Telecommunication Engineering

Course Title: Computer Graphics and Animation Lab

Course Code: CSTE - 4102

Documentation on Computer Graphics Lab

Submitted by

Name : Md Mosiur Zzaman

Roll : ASH1901036M

Name : Md Kamrul Hassan Khan

Roll : ASH1901039M

Submitted To

Dr. Md. Kamal Uddin

Associate Professor

Department of CSTE

Session : 2018-19

Airplane Simulation Game OpenGL Project

The animation project is about **"Airplane Simulation Game"**

The animation sequence is designed with the following steps:

- Storyboard Layout
- Object definitions
- Key-frame specification
- Generation of in-between frames.

These are explained below:

Storyboard Layout :

- ❖ The purpose of the project is to create an airplane simulation game using OpenGL. The game will involve controlling an airplane, avoiding obstacles like clouds and buildings, and aiming to achieve the highest possible score.
- ❖ The game is a 3D simulation where the player controls an airplane flying through a virtual environment.Click and hold mouse left key to gain altitude of the plane. Release the mouse left key to reduce the altitude.
- ❖ The objective is to navigate the airplane through the obstacles while accumulating a score based on the distance covered without touching any obstacles.
- ❖ The player can control the airplane's pitch and roll using keyboard inputs or other suitable controls.Use the Right mouse key to speed up the plane.
- ❖ Obstacles like clouds and buildings are generated at random positions in the game world. The obstacles pose a threat to the airplane and need to be avoided,also avoided falling in ground or reaching outside the upper boundary line to prevent the game from ending. As we reach distance multples of 50 tour level increases as well as the speed of the plane.
- ❖ The player's score increases as the airplane covers more distance without colliding with any obstacles. The longer the player survives without touching obstacles, the higher the score they accumulate.

1.Main Menu:

The main menu is the initial screen that players see when they start the game. It typically includes options such as "Play" "Instructions" "About" and "Exit."

2. Gameplay:

Controlling:

- Click and hold mouse left key to gain altitude of the plane.
- Release the mouse left key to reduce the altitude.
- Use the Right mouse key to speed up the plane.
- Press P for paused the game.
- Press E for escape the game.

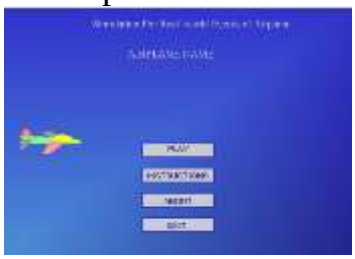
Objectives: The player's objective is to navigate the airplane as far as possible without colliding with obstacles. The longer the player survives, the higher their score.

Game Over Condition: The game ends when the airplane collides with an obstacle or falling in ground or reaching outside the upper boundary line. A game over screen is displayed, showing the player's final score.

Scoring Logic: Implement a scoring mechanism that increases the player's score as the airplane covers more distance. Update the score display on the user interface.

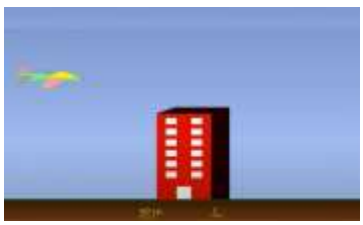
Object definitions:

❖ Airplane



❖ Cloud

❖ Building



Object definition refers to defining the attributes, behaviours, and operations of the game objects involved in the gameplay. The primary objects in airplane simulation game are the airpalne itself, the obstacles are cloud and buliding. Here's an explanation of how we might define these objects:

1.Airpalne: The player-controlled entity that represents the aircraft. Players can manipulate its flight, speed, and direction throughout the game.

2.Cloud: These are environmental obstacles in the form of clouds that the airplane must navigate through without colliding. They add a visual challenge to the game. Implement by a random obstacle generation algorithm.

3.Building: These represent urban or natural structures that pose a significant risk to the airplane if collided with. Buildings can be used to create a more complex and challenging game environment. Implement by random obstacle generation algorithm.

Keyframe generation:

Keyframe generation refers to the process of defining specific frames where significant changes occur in the game's animation or state. These frames, known as keyframes, help create smooth and visually appealing animations as the game progresses. Here's an explanation of how keyframe generation might work for a airplane simulation game:

1.Airplane Movement:

Create a function or class to control the airplane's movement.Update the airplane's position and orientation based on the current time and the interpolated values between keyframes.Ensure the airplane's movement adheres to the game's physics and constraints

2.Collision:

Implement collision detection algorithms to check if the airplane collides with obstacles (clouds or buildings) or falling in ground or reaching outside the upper boundary line. When a collision is detected, trigger appropriate actions like game over, score calculation, damage to the airplane

3.Transitions:

Keyframes are also essential for transitioning between different game states. For example, when the game transitions from the main menu to the gameplay state, we set up a keyframe that marks the exact frame when this transition occurs. This

might involve fading or sliding animations that provide a smooth shift between scenes.

4. Animations:

While airplane may not have intricate animations, we can use keyframes for other visual elements. For instance, when the game is won or lost, we can trigger an animation that displays a congratulatory message or a "game over" screen.

Generation in between frames:

Generation in between frames refers to the process of updating the game state and rendering intermediate frames between consecutive frames to achieve smooth animation and continuous gameplay. This involves calculating the positions of game objects, handling user input, and managing the game's logic in the time between two rendered frames. Here's how it works:

1. Frame Rate and Timing:

OpenGL-based games typically render frames at a specific frame rate, often measured in frames per second (FPS). For instance, if your game runs at 60 FPS, it means that approximately every 1/60th of a second, a new frame is rendered on the screen.

2. Game Loop:

The core of generating frames in between frames lies in the game loop. The game loop is a continuous loop that repeatedly performs the following steps:

Handle User Input: Check for keyboard inputs or other user interactions. For example, if the player presses a key to change the airplane’s direction, this input needs to be processed.

Update Game State: Update the positions and properties of game objects, such as the airplan’s position, cloud positions, and building position. This step ensures that the game state progresses over time.

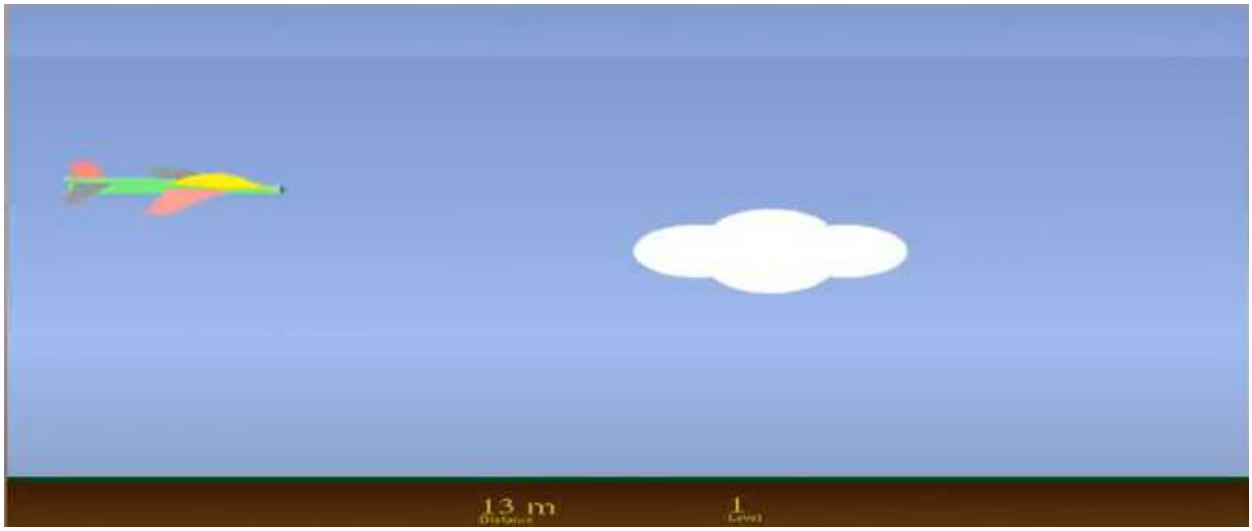
Collision Detection: Check for collisions between game objects, such as whether the airplane's head collides with obstacles. If a collision occurs, you update the game accordingly.

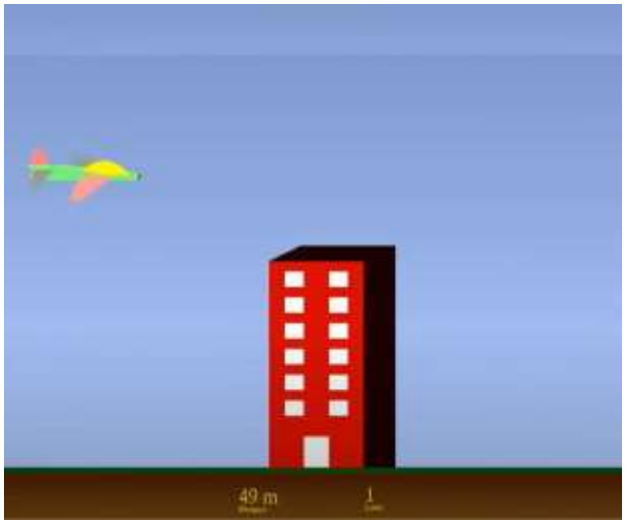
Render Frame: Render the updated game state to the screen, displaying the current positions and appearances of all game objects.

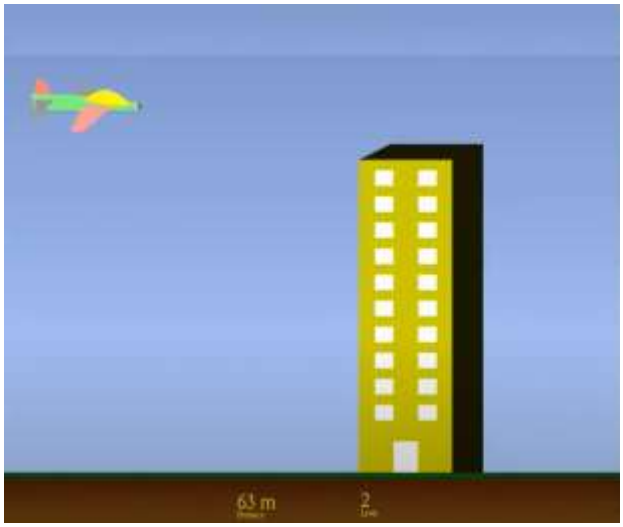
3. Frame Timing:

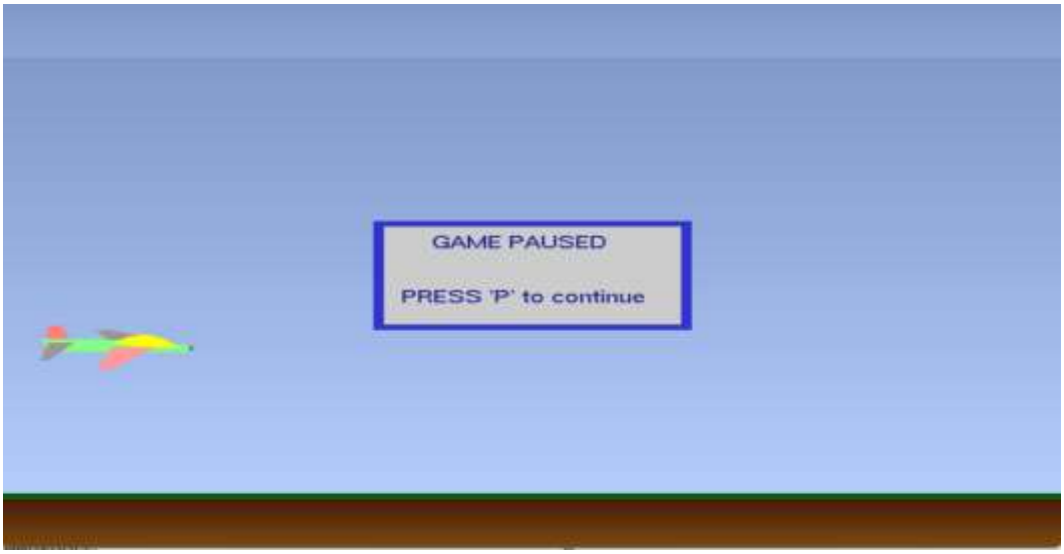
Because rendering occurs at a fixed frame rate, the time interval between frames can vary slightly due to system performance. To achieve consistent movement regardless of frame rate fluctuations, we incorporate the time elapsed between frames into our calculations. This is known as delta time and helps adjust the speed of movement and animation.

Screenshot of Airplane Simulation Game:




49 m
1


63 m
2


GAME PAUSED
PRESS 'P' to continue


GAME OVER!!!
DISTANCE : 4 m
LEVEL : 1
ENJOY PLAYING THE GAME
RESTART