Institute of Geographical Information Systems

School of Civil & Environmental Engineering

National University of Sciences and Technology, Islamabad, Pakistan

CS-212 - Object Oriented Programming

Semester: Fall 2025	**Class: SCEE-IGIS - 2024**
Name: Ali Nawaz	**CMS ID : 00000526123**
Submitted to: Ma'am Alvina Anjum	**Due Date: Dec 23, 2025**

Problem Based Learning

CLO 3 Develop modular and efficient code for real-world applications using event-driven programming techniques.

Table of Contents

Problem Scenario

Design and develop a simple **event-driven game** in **C++**. The game, **"Catch the Treasure,"** involves a player moving on a grid to collect treasures while avoiding obstacles. The game will demonstrate the principles of event-driven programming, where user actions (events) like moving up, down, left, or right will trigger responses in the game environment.

Learning Objectives (CLO3)

- Develop modular and efficient code for real-world applications using event-driven programming techniques.
- Implement user interaction through event-handling mechanisms in a game environment.
- Demonstrate the use of classes, objects, and methods for event handling in an interactive application.

Game Rules

1. The player starts at a random position on a 5x5 grid.
2. Treasures are randomly placed on the grid.
3. The player can move in four directions: **up**, **down**, **left**, and **right**.
4. If the player lands on a cell with a treasure, they earn points.
5. Some cells contain obstacles. Landing on an obstacle ends the game.
6. The player has a limited number of moves to collect treasures.

Assignment Tasks

Task 1: Class Design

1. GameObject:

a. Represents an object on the grid (player, treasure, or obstacle).
b. Attributes: `positionX` (row), `positionY` (column), `symbol` (char to display on the grid).
c. Methods:
   1) Constructor to initialize the object's position and symbol.
   2) Accessors for position and symbol.

2. Game:

1. Manages the game environment, objects, and user actions.
2. Attributes:
   a. 2D grid (vector of vectors) representing the game board.
   b. Player object, list of treasures, and list of obstacles.
   c. Score and remaining moves.
3. Methods:
   a. `void initializeGame()` – Randomly place player, treasures, and obstacles on the grid.
   b. `void renderGrid()` – Display the grid with updated positions.

c. `bool movePlayer(char)` – Move the player based on user input (`W`, `A`, `S`, `D` for up, left, down, right).

d. `void checkCell()` – Check the player's current position for treasures or obstacles.

e. `bool isGameOver()` – Check if the game has ended (all moves used or player hit an obstacle).

Task 2: Event Handling

1. User actions (`W`, `A`, `S`, `D`) will trigger the `movePlayer` method.
2. Each move will update the player's position and call `checkCell` to determine if a treasure is collected or if the game ends.
3. Display updated game state after each move.

Assignment Requirements

Game Flow

1. The grid is displayed at the start, showing the player's position (`P`), treasures (`T`), and obstacles (`O`).
2. The player enters a move direction.
3. The game processes the move, updates the grid, and displays the new state.
4. The game ends if:

a) The player runs out of moves.
b) The player hits an obstacle.
c) All treasures are collected.

Sample Grid (5x5)

```
. . . T .
P . O . .
. . . . T
. . . O .
T . . . .
```

1. `P`: Player
2. `T`: Treasure
3. `O`: Obstacle
4. `.`: Empty cell

User Interaction:

```
Enter move (W: Up, A: Left, S: Down, D: Right): W
```

Deliverables:

1. **Code Implementation:**
   - Well-structured and documented C++ code implementing the game.
2. **Code Documentation:**
   - Explanation of classes, methods, and event-handling mechanisms used.
3. **Game Demo Video/Screenshots:**
   - Demonstration of game execution with sample moves.
4. **Report (PDF/DOC):**
   - Introduction to the problem statement.
   - Explanation of event-driven programming concepts used.
   - Description of the game rules and mechanics.
   - Discussion on challenges faced and solutions implemented.
   - Conclusion and possible future improvements.

Explanation of the Code

1. **Class GameObject**:

   - This class represents the entities on the grid.

   - It stores positionX (row) and positionY (column) and the symbol (P, T, or O).

   - We used a constructor to easily set these values when creating objects.

2. **Class Game**:
   - **Grid**: I used a vector<vector<char>> to create the 5x5 board. This is easier than raw arrays because vectors handle memory automatically.

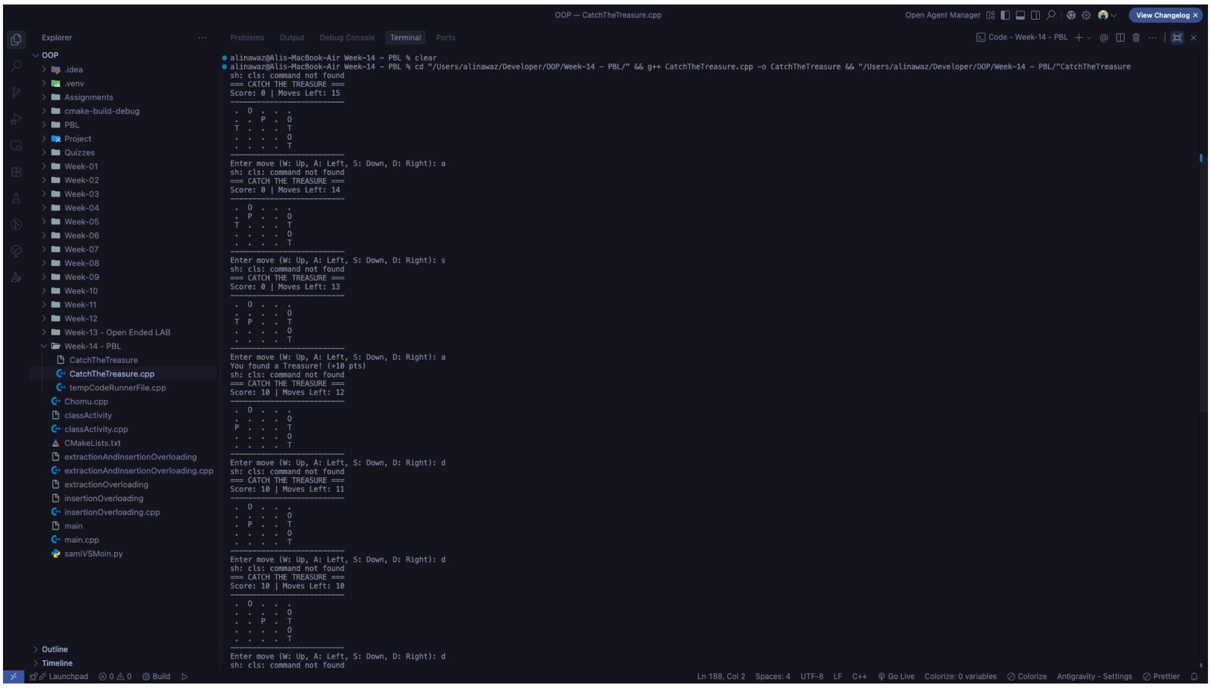
   - **initializeGame**: This function uses rand() to place the Player, Treasures, and Obstacles at random coordinates. It ensures two objects don't spawn on the same spot by checking if the grid is . (empty) before placing.

   - **movePlayer**: This is the core logic. It takes the user input (W,A,S,D), calculates the *future* coordinates, and checks what is currently in that cell.
     - If it is a 'T', we increase score.
     - If it is an 'O', we set gameOver = true.
     - If valid, we swap the player's old position with . and the new position with P.

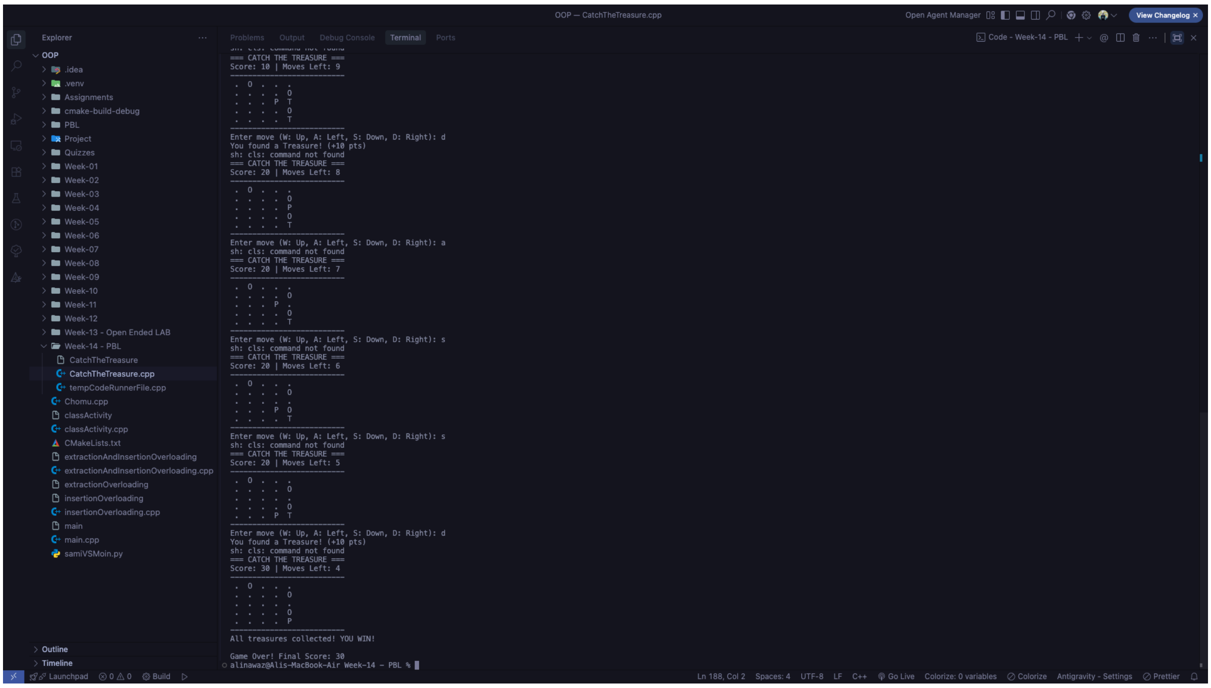
3. **Event Handling (Main Loop)**:
   - The while loop inside main() acts as the event listener. It waits for a keyboard event (cin >> input).

   - Once an event occurs, it triggers the myGame.movePlayer(input) method, which updates the internal state and re-draws the grid.

Output Screenshots



```
● alinawaz@Alis-MacBook-Air Week-14 - PBL % clear
● alinawaz@Alis-MacBook-Air Week-14 - PBL % cd "/Users/alinawaz/Developer/OOP/Week-14 - PBL/" && g++ CatchTheTreasure.cpp -o CatchTheTreasure && "/Users/alinawaz/Developer/OOP/Week-14 - PBL/"CatchTheTreasure
sh: cls: command not found
=== CATCH THE TREASURE ===
Score: 0 | Moves Left: 15
--------------------------
 .  O  .  .  .
 .  .  P  .  O
 T  .  .  .  T
 .  .  .  .  O
 .  .  .  .  T
--------------------------
Enter move (W: Up, A: Left, S: Down, D: Right): a
sh: cls: command not found
=== CATCH THE TREASURE ===
Score: 0 | Moves Left: 14
--------------------------
 .  O  .  .  .
 .  P  .  .  O
 T  .  .  .  T
 .  .  .  .  O
 .  .  .  .  T
--------------------------
Enter move (W: Up, A: Left, S: Down, D: Right): s
sh: cls: command not found
=== CATCH THE TREASURE ===
Score: 0 | Moves Left: 13
--------------------------
 .  O  .  .  .
 .  .  .  .  O
 T  P  .  .  T
 .  .  .  .  O
 .  .  .  .  T
--------------------------
Enter move (W: Up, A: Left, S: Down, D: Right): a
You found a Treasure! (+10 pts)
sh: cls: command not found
=== CATCH THE TREASURE ===
Score: 10 | Moves Left: 12
--------------------------
 .  O  .  .  .
 .  .  .  .  O
 P  .  .  .  T
 .  .  .  .  O
 .  .  .  .  T
--------------------------
Enter move (W: Up, A: Left, S: Down, D: Right): d
sh: cls: command not found
=== CATCH THE TREASURE ===
Score: 10 | Moves Left: 11
--------------------------
 .  O  .  .  .
 .  .  .  .  O
 .  P  .  .  T
 .  .  .  .  O
 .  .  .  .  T
--------------------------
Enter move (W: Up, A: Left, S: Down, D: Right): d
sh: cls: command not found
=== CATCH THE TREASURE ===
Score: 10 | Moves Left: 10
--------------------------
 .  O  .  .  .
 .  .  .  .  O
 .  .  P  .  T
 .  .  .  .  O
 .  .  .  .  T
--------------------------
Enter move (W: Up, A: Left, S: Down, D: Right): d
sh: cls: command not found
```



```
sh: cls: command not found
=== CATCH THE TREASURE ===
Score: 10 | Moves Left: 9
--------------------------
 .  O  .  .  .
 .  .  .  .  O
 .  .  .  P  T
 .  .  .  .  O
 .  .  .  .  T
--------------------------
Enter move (W: Up, A: Left, S: Down, D: Right): d
You found a Treasure! (+10 pts)
sh: cls: command not found
=== CATCH THE TREASURE ===
Score: 20 | Moves Left: 8
--------------------------
 .  O  .  .  .
 .  .  .  .  O
 .  .  .  .  P
 .  .  .  .  O
 .  .  .  .  T
--------------------------
Enter move (W: Up, A: Left, S: Down, D: Right): a
sh: cls: command not found
=== CATCH THE TREASURE ===
Score: 20 | Moves Left: 7
--------------------------
 .  O  .  .  .
 .  .  .  .  O
 .  .  .  P  .
 .  .  .  .  O
 .  .  .  .  T
--------------------------
Enter move (W: Up, A: Left, S: Down, D: Right): s
sh: cls: command not found
=== CATCH THE TREASURE ===
Score: 20 | Moves Left: 6
--------------------------
 .  O  .  .  .
 .  .  .  .  O
 .  .  .  .  .
 .  .  .  P  O
 .  .  .  .  T
--------------------------
Enter move (W: Up, A: Left, S: Down, D: Right): s
sh: cls: command not found
=== CATCH THE TREASURE ===
Score: 20 | Moves Left: 5
--------------------------
 .  O  .  .  .
 .  .  .  .  O
 .  .  .  .  .
 .  .  .  .  O
 .  .  .  P  T
--------------------------
Enter move (W: Up, A: Left, S: Down, D: Right): d
You found a Treasure! (+10 pts)
sh: cls: command not found
=== CATCH THE TREASURE ===
Score: 30 | Moves Left: 4
--------------------------
 .  O  .  .  .
 .  .  .  .  O
 .  .  .  .  .
 .  .  .  .  O
 .  .  .  .  P
--------------------------
All treasures collected! YOU WIN!

Game Over! Final Score: 30
○ alinawaz@Alis-MacBook-Air Week-14 - PBL % ▌
```

Project Report

1. Introduction

This project implements "Catch the Treasure," a C++ event-driven game where a player navigates a 5x5 grid to collect treasures while avoiding obstacles. The objective was to demonstrate Learning Objective CLO3 by developing modular code using Object-Oriented Programming (OOP) and event-handling mechanisms.

2. Event-Driven Concepts

The game relies on user interaction rather than a fixed sequence:

- Event Loop: A `while` loop runs continuously, waiting for user input.

- Event Trigger: Key presses (`W`, `A`, `S`, `D`) serve as events.

- Event Handler: The `movePlayer` function processes the input, updates coordinates, and handles collisions.

3. Game Rules & Mechanics

- **Setup:** The grid contains a Player (`P`), Treasures (`T`), and Obstacles (`O`) at random positions.

- **Action:** The player moves Up, Down, Left, or Right to collect treasures for points.

- **Game Over:** The game ends if the player hits an obstacle, runs out of moves, or collects all treasures.

4. Challenges & Solutions

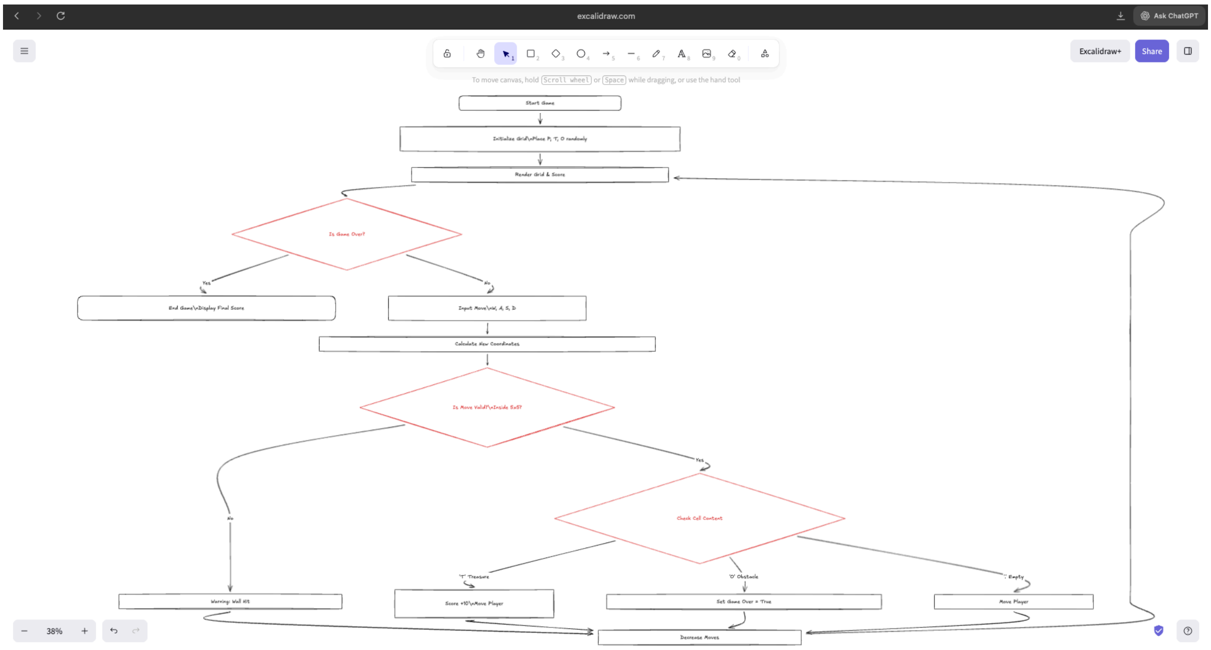
- **Overlap:** Objects could spawn on top of each other.
  Solution: Added a check to ensure a cell is empty (`.`) before placing an object.

- **Boundaries:** Moving outside the 5x5 grid caused errors.
  Solution: Implemented boundary checks; invalid moves are ignored but still cost a turn.

5. Conclusion

The project successfully created a functional, modular game. It demonstrates how classes (GameObject, Game) and event loops work together to create interactive software.

Game Logic Flowchart


excalidraw.com
Ask ChatGPT
Excalidraw+
Share
To move canvas, hold Scroll wheel or Space while dragging, or use the hand tool
Start Game
Initialize Grid\nPlace P, T, O randomly
Render Grid & Score
Is Game Over?
Yes
No
End Game\nDisplay Final Score
Input Move\nW, A, S, D
Calculate New Coordinates
Is Move Valid?\nInside 5x5?
Yes
No
Check Cell Content
'T' Treasure
'O' Obstacle
'.' Empty
Warning: Wall Hit
Score +10\nMove Player
Set Game Over = True
Move Player
Decrease Moves
38%