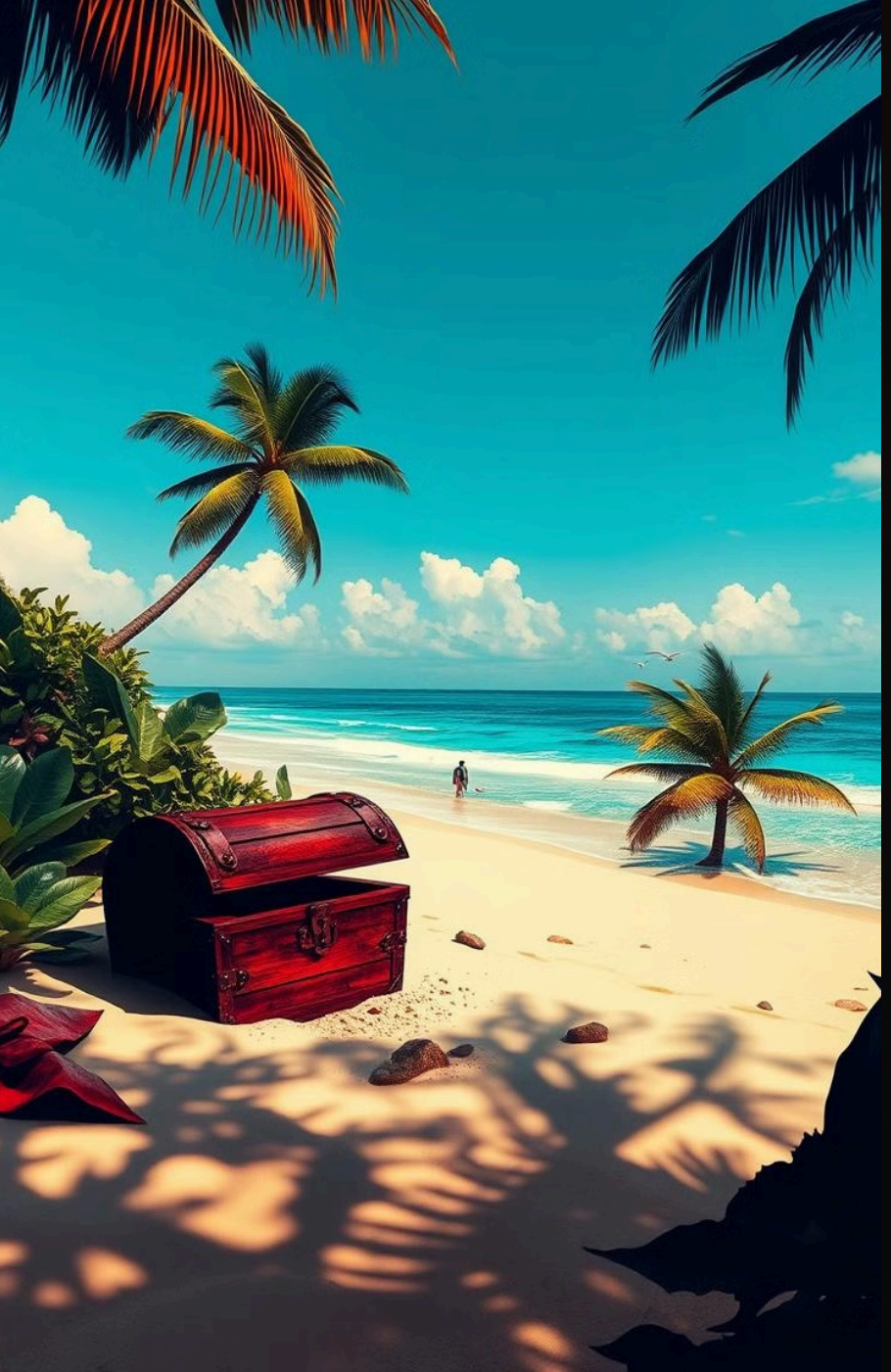


Treasure Hunt with a Twist

Welcome to a unique adventure, where the traditional treasure hunt gets a captivating twist. Join us as we delve into a puzzles, challenges,boosts,traps and unexpected discoveries.

by TEAM CODE CREW

The team members name of our group are given below :

Swachchha kanungo ROll: 24070138

Anusha Das Roll : 24070115

Afifah Mansur chowdhury Roll : 24070118

Tinni dey Roll : 24070108

FM . Muntasir tawsif tabib Roll : 24070107

Sidratul muntaha Roll : 24070112

Introduction

- **What is the Treasure Hunt Game?**

A simple grid-based game where a player navigates a 5×5 , 7×7, 10×10 grid to find a hidden treasure while avoiding traps.

Objective

- Develop a grid-based game for players to:
  - Collect treasures.
  - Avoid traps.
  - Strategically manage moves and health.
- Introduce a "hint system" to improve accessibility.

Key Features :

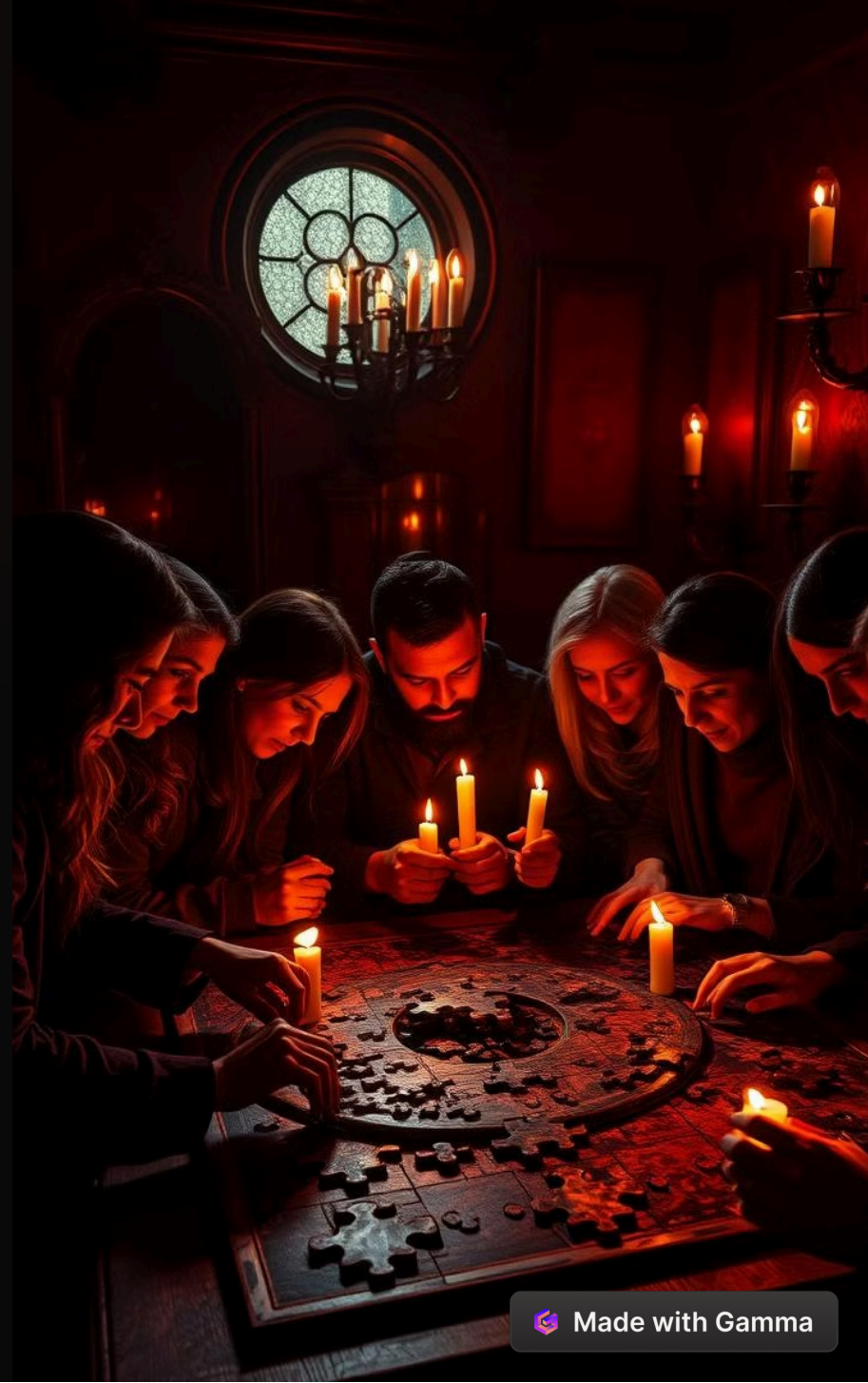
Dynamic Difficulty Levels:

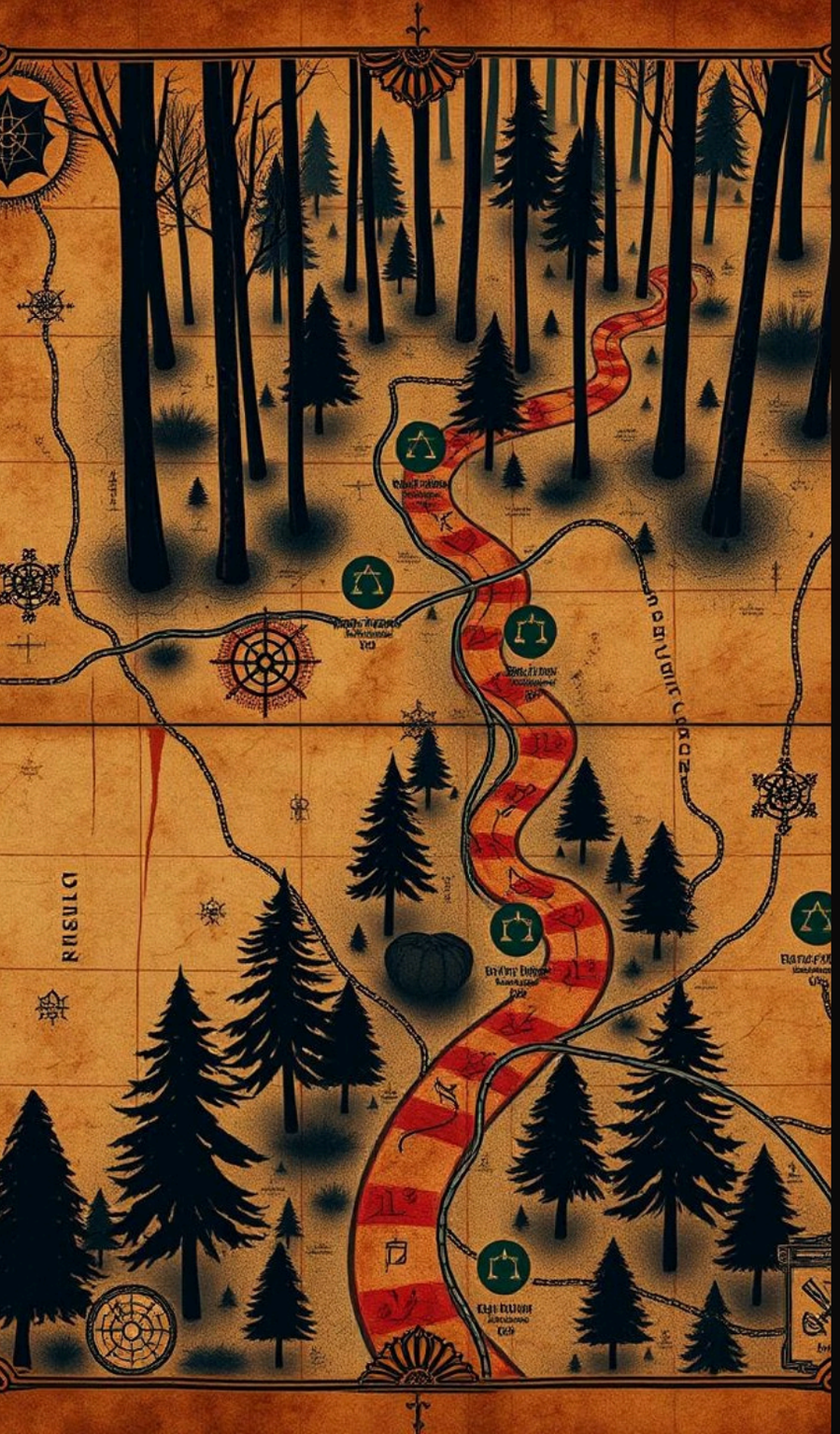
- Easy: 5x5 grid, few traps.
- Medium: 7x7 grid, more traps.
- Hard: 10x10 grid, maximum traps.

Randomized Gameplay:

- Treasures (T), traps (X), and boosts (+) are placed randomly.

1. **Player Actions**:**Winning & Losing Conditions:**
   - Move using `w` (up), `a` (left), `s` (down), `d` (right).
   - Request hints with `h`.
2. **Winning & Losing Conditions**:Lose by:Health reaching 0.Running out of moves.
   - Win by scoring 50 points.
   - Lose by:
     - Health reaching 0.
     - Running out of moves.





Gameplay Flow

1. **Game Initialization**:
   - Select difficulty.
   - Randomly populate grid with items.
2. **Main Loop**:
   - Display grid and game stats.
   - Accept player input for moves or hints.
   - Update player position and game state.
3. **Game End**:
   - Check win or lose conditions.

Hint System

1. **Purpose**:
   - Guide players towards the nearest treasure or boost.
2. **How It Works**:
   - Calculate the Manhattan distance to the closest `T` or `+`.
   - Display the coordinates of the target cell.
3. **Activation**:
   - Press `h` during the game.

CODE :

```
#include <stdio.h>

 #include <stdlib.h>

 #include <time.h>

#define GRID_SIZE 5

#define MAX_MOVES 20

typedef struct {

int x;

int y;

} Position;

char grid[GRID_SIZE][GRID_SIZE];

Position player = {0, 0};

int score = 0;

int moves = MAX_MOVES;

 int health = 3; int difficulty;

 int boardSize, trapCount, boostCount;

 int playerX, playerY, treasureX, treasureY;

 char playerName[30];

void initializeGrid() {

 srand(time(NULL));
```

```
  printf("Welcome to the Ultimate Treasure Hunt Game!\n");
  printf("Pick your level of madness:\n");
  printf("1. Easy (5x5 grid, few traps)\n");
  printf("2. Medium (7x7 grid, more traps)\n");
  printf("3. Hard (10x10 grid, max traps)\n");
  scanf("%d", &difficulty);

  if (difficulty == 1) {
     boardSize = 5;
     trapCount = 5;
     boostCount = 2;
  } else if (difficulty == 2) {
     boardSize = 7;
     trapCount = 10;
     boostCount = 3;
  } else {
     boardSize = 10;
     trapCount = 20;
     boostCount = 5;
  }

  printf("Enter your name, brave adventurer: ");
  scanf("%s", playerName);

  for (int i = 0; i < GRID_SIZE; i++) {
     for (int j = 0; j < GRID_SIZE; j++) {
        grid[i][j] = '*';
     }
  }

  for (int i = 0; i < trapCount + boostCount; i++) {
     int x = rand() % GRID_SIZE;
     int y = rand() % GRID_SIZE;
     char items[] = {'T', 'X', '+'};
     grid[x][y] = items[rand() % 3];
  }
```

```
}

void displayGrid() {

 for (int i = 0; i < GRID_SIZE; i++) {

for (int j = 0; j < GRID_SIZE; j++) {

 if (i == player.x && j == player.y) {

 printf("P ");

 } else {

printf("%c ", grid[i][j]);

  }

} printf("\n");

}

 }

void updatePosition(char move) {

int oldX = player.x; int oldY = player.y;
```

```
  if (move == 'w' && player.x > 0) player.x--;
  if (move == 's' && player.x < GRID_SIZE - 1) player.x++;
  if (move == 'a' && player.y > 0) player.y--;
  if (move == 'd' && player.y < GRID_SIZE - 1) player.y++;

  char cell = grid[player.x][player.y];
  if (cell == 'T') {
     score += 10;
  } else if (cell == 'X') {
     health--;
     if (health <= 0) {
        printf("Game Over! You hit a trap!\n");
        exit(0);
     }
  } else if (cell == '+') {
     moves += 5;
  }

  grid[oldX][oldY] = '*';
  moves--;
  if (moves <= 0) {
     printf("Game Over! You ran out of moves!\n");
     exit(0);
  }
```

```
}

void giveHint() {

int closestDistance = GRID_SIZE * GRID_SIZE; int hintX = -1, hintY = -1;
```

```
  for (int i = 0; i < GRID_SIZE; i++) {
     for (int j = 0; j < GRID_SIZE; j++) {
        if (grid[i][j] == 'T' || grid[i][j] == '+') {
           int distance = abs(player.x - i) + abs(player.y - j);
           if (distance < closestDistance) {
              closestDistance = distance;
              hintX = i;
              hintY = j;
           }
        }
     }
  }

  if (hintX != -1 && hintY != -1) {
     printf("Hint: Move closer to (%d, %d) to find a treasure or boost.\n", hintX, hintY);
  } else {
     printf("No treasures or boosts left on the grid.\n");
  }
```

```
}

int main() {

 initializeGrid();

while (1) {

displayGrid();

printf("Score: %d, Health: %d, Moves left: %d\n", score, health, moves);

printf("Enter move (w/a/s/d) or 'h' for a hint: ");

char move; scanf(" %c", &move);
```

```
     if (move == 'h') {
        giveHint();
     } else {
        updatePosition(move);
     }

     if (score >= 50) {
        printf("You win! You collected enough treasures!\n");
        break;
     }
  }
  return 0;
```

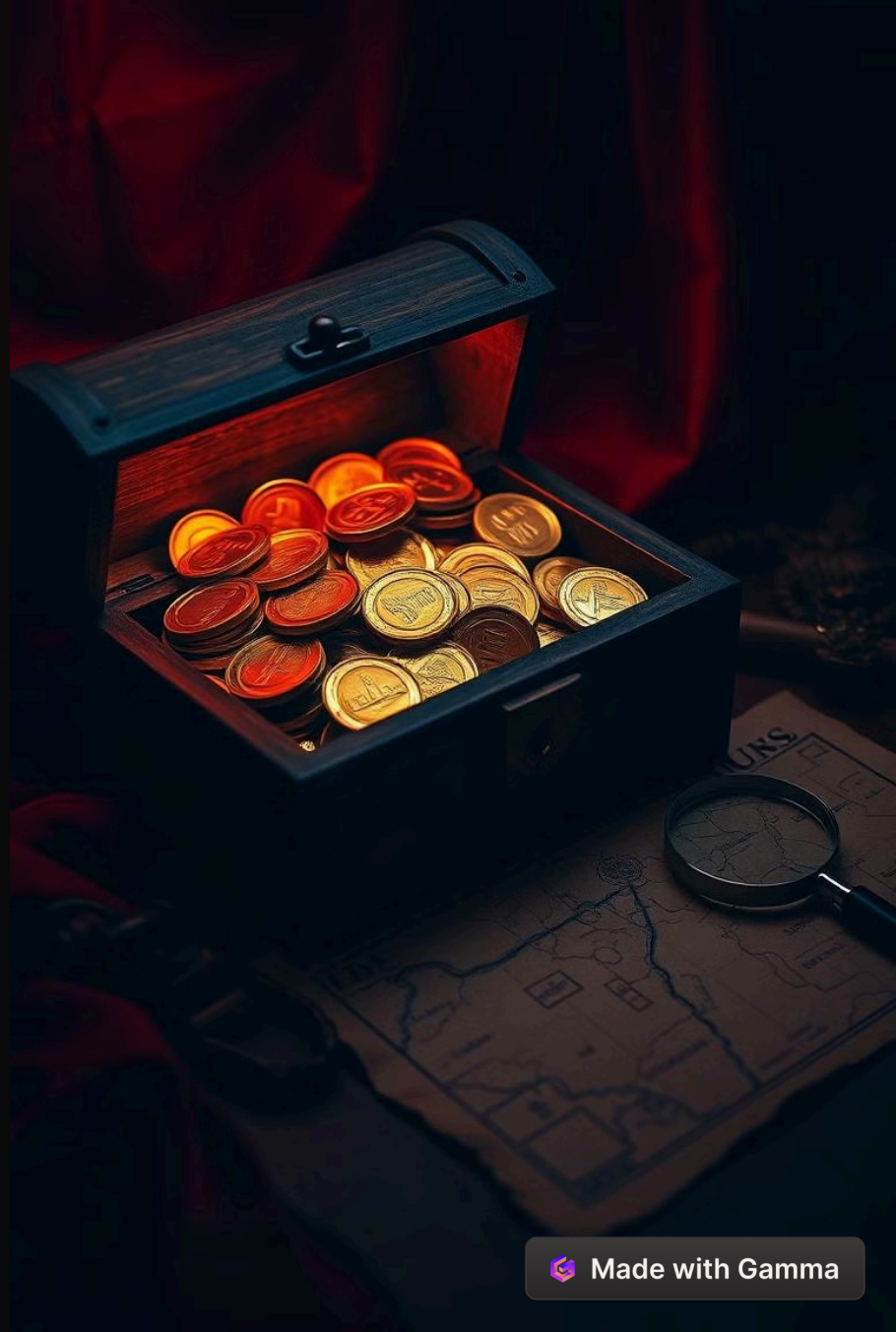
```
}
```

Key Functions

1. `initializeGrid()`:
   - Sets up the grid and places items randomly.
2. `displayGrid()`:
   - Shows the current grid with the player's position (`P`).
3. `updatePosition(move)`:
   - Updates player’s position based on input.
   - Handles interactions with items (treasures, traps, boosts).
4. `giveHint()`:
   - Provides coordinates of the nearest helpful item.

User Interaction

- **Input Options**:
  - Movement: w/a/s/d.
  - Hint: h.
- **Game Feedback**:
  - Display:
    - Current score.
    - Remaining moves.
    - Player's health.
- **End Game Alerts**:
  - Victory message when score reaches 50.
  - Game over if health or moves run out.



The Final Showdown: Solving the code:

INPUT

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define GRID_SIZE 5
#define MAX_MOVES 20

typedef struct {
    int x;
    int y;
} Position;

char grid[GRID_SIZE][GRID_SIZE];
Position player = {0, 0};
int score = 0;
int moves = MAX_MOVES;
int health = 3;
int difficulty;
int boardSize, trapCount, boostCount;
int playerX, playerY, treasureX, treasureY;
char playerName[30];

void initializeGrid() {
```

```c
void initializeGrid() {
    srand(time(NULL));

    printf("Welcome to the Ultimate Treasure Hunt Game!\n");
    printf("Pick your level of madness:\n");
    printf("1. Easy (5x5 grid, few traps)\n");
    printf("2. Medium (7x7 grid, more traps)\n");
    printf("3. Hard (10x10 grid, max traps)\n");
    scanf("%d", &difficulty);

    if (difficulty == 1) {
        boardSize = 5;
        trapCount = 5;
        boostCount = 2;
    } else if (difficulty == 2) {
        boardSize = 7;
        trapCount = 10;
        boostCount = 3;
    } else {
        boardSize = 10;
        trapCount = 20;
        boostCount = 5;
```

```c
    printf("Enter your name, brave adventurer: ");
    scanf("%s", playerName);

    for (int i = 0; i < GRID_SIZE; i++) {
        for (int j = 0; j < GRID_SIZE; j++) {
            grid[i][j] = '*';
        }
    }

    for (int i = 0; i < trapCount + boostCount; i++) {
        int x = rand() % GRID_SIZE;
        int y = rand() % GRID_SIZE;
        char items[] = {'T', 'X', '+'};
        grid[x][y] = items[rand() % 3];
    }
}

void displayGrid() {
    for (int i = 0; i < GRID_SIZE; i++) {
        for (int j = 0; j < GRID_SIZE; j++) {
            if (i == player.x && j == player.y) {
                printf("P ");
```

```c
void updatePosition(char move) {
    int oldX = player.x;
    int oldY = player.y;

    if (move == 'w' && player.x > 0) player.x--;
    if (move == 's' && player.x < GRID_SIZE - 1) player.x++;
    if (move == 'a' && player.y > 0) player.y--;
    if (move == 'd' && player.y < GRID_SIZE - 1) player.y++;

    char cell = grid[player.x][player.y];
    if (cell == 'T') {
        score += 10;
    } else if (cell == 'X') {
        health--;
        if (health <= 0) {
            printf("Game Over! You hit a trap!\n");
            exit(0);
        }
    } else if (cell == '+') {
        moves += 5;
    }
```

```c
    }

    grid[oldX][oldY] = '*';
    moves--;
    if (moves <= 0) {
        printf("Game Over! You ran out of moves!\n");
        exit(0);
    }
}

void giveHint() {
    int closestDistance = GRID_SIZE * GRID_SIZE;
    int hintX = -1, hintY = -1;

    for (int i = 0; i < GRID_SIZE; i++) {
        for (int j = 0; j < GRID_SIZE; j++) {
            if (grid[i][j] == 'T' || grid[i][j] == '+') {
                int distance = abs(player.x - i) + abs(player.y - j);
                if (distance < closestDistance) {
                    closestDistance = distance;
                    hintX = i;
                    hintY = j;
                }
```

```c
int main() {
    initializeGrid();
    while (1) {
        displayGrid();
        printf("Score: %d, Health: %d, Moves left: %d\n", score, health, moves);
        printf("Enter move (w/a/s/d) or 'h' for a hint: ");
        char move;
        scanf(" %c", &move);

        if (move == 'h') {
            giveHint();
        } else {
            updatePosition(move);
        }

        if (score >= 50) {
            printf("You win! You collected enough treasures!\n");
            break;
        }
    }
    return 0;
}
```

OUTPUT:

```
Welcome to the Ultimate Treasure Hunt Game!
Pick your level of madness:
1. Easy (5x5 grid, few traps)
2. Medium (7x7 grid, more traps)
3. Hard (10x10 grid, max traps)
1
Enter your name, brave adventurer: abir
P * * * *
+ * * * +
* T * * *
* * * * *
* X * * *
Score: 0, Health: 3, Moves left: 20
Enter move (w/a/s/d) or 'h' for a hint:
```

```
Welcome to the Ultimate Treasure Hunt Game!
Pick your level of madness:
1. Easy (5x5 grid, few traps)
2. Medium (7x7 grid, more traps)
3. Hard (10x10 grid, max traps)
```

```
Welcome to the Ultimate Treasure Hunt Game!
Pick your level of madness:
1. Easy (5x5 grid, few traps)
2. Medium (7x7 grid, more traps)
3. Hard (10x10 grid, max traps)
1
Enter your name, brave adventurer: abir
P * * * *
+ * * * +
* T * * *
* + * * *
* X * * *
Score: 0, Health: 3, Moves left: 20
Enter move (w/a/s/d) or 'h' for a hint: d
* P * * *
+ * * * +
* T * * *
* + * * *
* X * * *
Score: 0, Health: 3, Moves left: 19
Enter move (w/a/s/d) or 'h' for a hint:
```

```
Welcome to the Ultimate Treasure Hunt Game!
Pick your level of madness:
1. Easy (5x5 grid, few traps)
2. Medium (7x7 grid, more traps)
3. Hard (10x10 grid, max traps)
1
Enter your name, brave adventurer: abir
P * * * *
+ * * * +
* T * * *
* + * * *
* X * * *
Score: 0, Health: 3, Moves left: 20
Enter move (w/a/s/d) or 'h' for a hint: d
* P * * *
+ * * * +
* T * * *
* + * * *
* X * * *
Score: 0, Health: 3, Moves left: 19
Enter move (w/a/s/d) or 'h' for a hint: s
* * * * *
+ P * * +
* T * * *
* + * * *
* X * * *
Score: 0, Health: 3, Moves left: 18
Enter move (w/a/s/d) or 'h' for a hint:
```

Conclusion

- A fun and engaging grid-based game.
- Encourages strategic thinking and resource management.
- Accessible for all players with hint functionality.
- Room for future improvements like:
  - Advanced AI for traps.
  - Multiplayer mode.

THANK YOU !!