UNIVERSITY OF JOHANNESBURG

FACULTY OF SCIENCE

COMPUTER SCIENCE 1A **DESIGN**

Problem Description

Write a C++ program where You must move a player-controlled operator around a two-dimensional playing area in the game. You win the game when the operator finds the main switch. You lose the game if the operator's torch runs out of batteries or the operator steps on an exposed wire. The operator's torch can only stay on for a predefined number of moves until it runs out of battery power. Batteries are scattered around the power plant, and the operator must pick them up while looking for the switch in order to keep the torch working.

Input & Output;

Input	
Input Description	*Mechanism*
Number of rows, rows	Command-line arguments
Options for game play	Standard input
Gameplay moves	Stndard input
Output	
Output Description	*Stream (optional)*
NewPosition/newWold	Standard Output
Errors	Cerr Output

Data Format

Identifier	*Data Type*	*Description*
numRows	integer	For setting world row dimension
numCols	integer	For setting world column dimension
BatteryProbability	double	For taking in battery value
wireProbabulity	double	For taking in wire value

Pseudo Code

```
Include "mistSpace.h"
Include <iostream>

using namespace std;
using namespace MistSpace;

int main(int argc, char* argv[]) {
   if (argc != 5) {
      Print "Usage: " + argv[0] + " <numRows> <numCols> <batteryProbability>
<wireProbability>"
      Return 1
   }

   // Taking values through command line arguments
   int numRows <- atoi(argv[1])
   int numCols <- atoi(argv[2])
   double batteryProbability <- atof(argv[3])
   double wireProbability <- atof(argv[4])

   char** world
   createWorld(world, numRows, numCols, batteryProbability, wireProbability)

   int playerRow <- numRows - 1
   int playerCol <- rand() % numCols
   int movesLeft <- 15

   char move
   while (true) {
      updateMist(world, playerRow, playerCol, numRows, numCols)
      displayWorld(world, numRows, numCols, playerRow, playerCol)

      Print "Moves Left: " + movesLeft
      Print "Enter move (W/A/S/D to move, Q to quit): "
      Input move

      if (!moveOperator(world, playerRow, playerCol, movesLeft, move, numRows,
numCols)) {
         Print "Invalid move!"
      }

      if (isVictory(world, playerRow, playerCol, numRows)) {
         Print "Congratulations! You found the switch!"
```

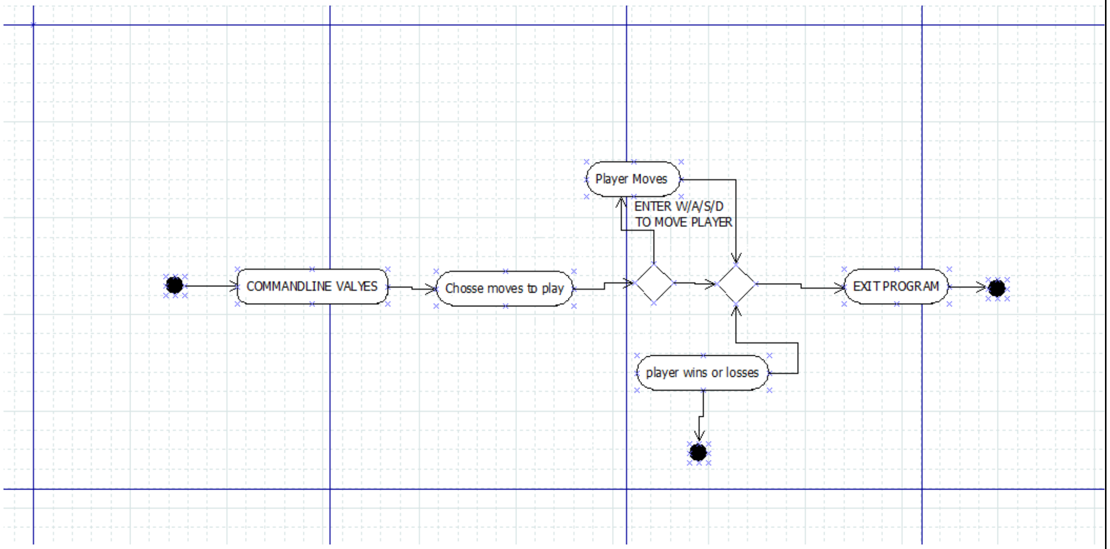
```
            Break
        }

        if (isDefeat(movesLeft, world, playerRow, playerCol)) {
            Print "Game over!"
            Break
        }
    }

    cleanWorld(world, numRows)

    Return 0
}
```

UML Activity Diagram


COMMANDLINE VALYES
Chosse moves to play
Player Moves
ENTER W/A/S/D TO MOVE PLAYER
player wins or losses
EXIT PROGRAM