:DEPARTMENT:

CS - AI

:PROJECT NAME:

WALL E

:SUBMITTED BY:

Muhammad Umer (i222365)
Ammar Ahmed (i222366)

Contents

Introduction

Efficient wall painting in 3D environments presents significant challenges, particularly under constraints such as limited paint availability, time restrictions, and adjacency rules. The problem involves optimizing the use of resources while adhering to these constraints and ensuring minimal time and effort. This project aims to address these challenges using a constraint satisfaction problem (CSP) approach.

Proposed Solution

WALL-E is a sophisticated CSP solver designed to optimize wall painting tasks in a 3D environment. The solution involves:

1. A **constraint-based algorithm** that assigns colors to walls while respecting constraints such as paint availability, adjacency rules, and time limits.
2. A **3D visualization system** for displaying results, making the solution easy to interpret.
3. An **intuitive graphical user interface (GUI)** for interactive input and visualization, allowing users to define problems, explore constraints, and view optimized solutions seamlessly.

The system integrates A* pathfinding for efficient traversal and a greedy algorithm for constraint resolution, ensuring compliance with all rules while minimizing painting time and paint wastage.

Key Features

1. GUI Features

The application leverages PyQt5 to deliver an intuitive user experience with the following screens:

Welcome Screen:

1. Manual Input: Enter inputs interactively.
2. Load JSON File: Load input data from a JSON file.
3. Test Sample JSON: Use a predefined sample for testing purposes.

Input Validation

- Ensures all fields are correctly filled.
- Displays error messages for invalid inputs or missing fields.

2. Input Structure

Input can be manually entered via the GUI or loaded from a JSON file. The required structure includes:

Surfaces: List of walls with attributes:

- *id:* Unique identifier.
- *height,* width: Wall dimensions.
- *position:* Starting point in 3D space.
- *orientation:* Wall alignment ("verticalx", "verticaly", or "horizontal").

Colors: List of available paint colors.

Constraints:

- *time_per_meter:* Time to paint 1 square meter.
- *max_time:* Total allowed painting time.
- *paint_availability:* Quantities of paint for each color.
- *adjacency_constraint:* Prevent adjacent walls from having the same color.
- *min_colors:* Minimum number of distinct colors to use.

Starting Position: Initial robot position in 3D space.

3. Core Methods

parse_surfaces

Processes wall data to calculate areas and prepare for visualization and pathfinding.

a_star_pathfinding

Implements the A algorithm to compute the shortest path for the robot's traversal, using 3D Euclidean distance as the heuristic.

solve_csp

Greedily assigns colors to walls based on constraints:

- Ensures sufficient paint for each wall.
- Enforces adjacency rules.
- Keeps painting time within the allowed limit.

Outputs include:

- Assigned colors for walls.
- Total painting and travel time.
- Paint usage.
- Robot's traversal path.

visualize_3d_environment

- Renders the 3D environment using matplotlib:
- Colors walls based on their assignments.
- Plots the robot's traversal path as a red line.

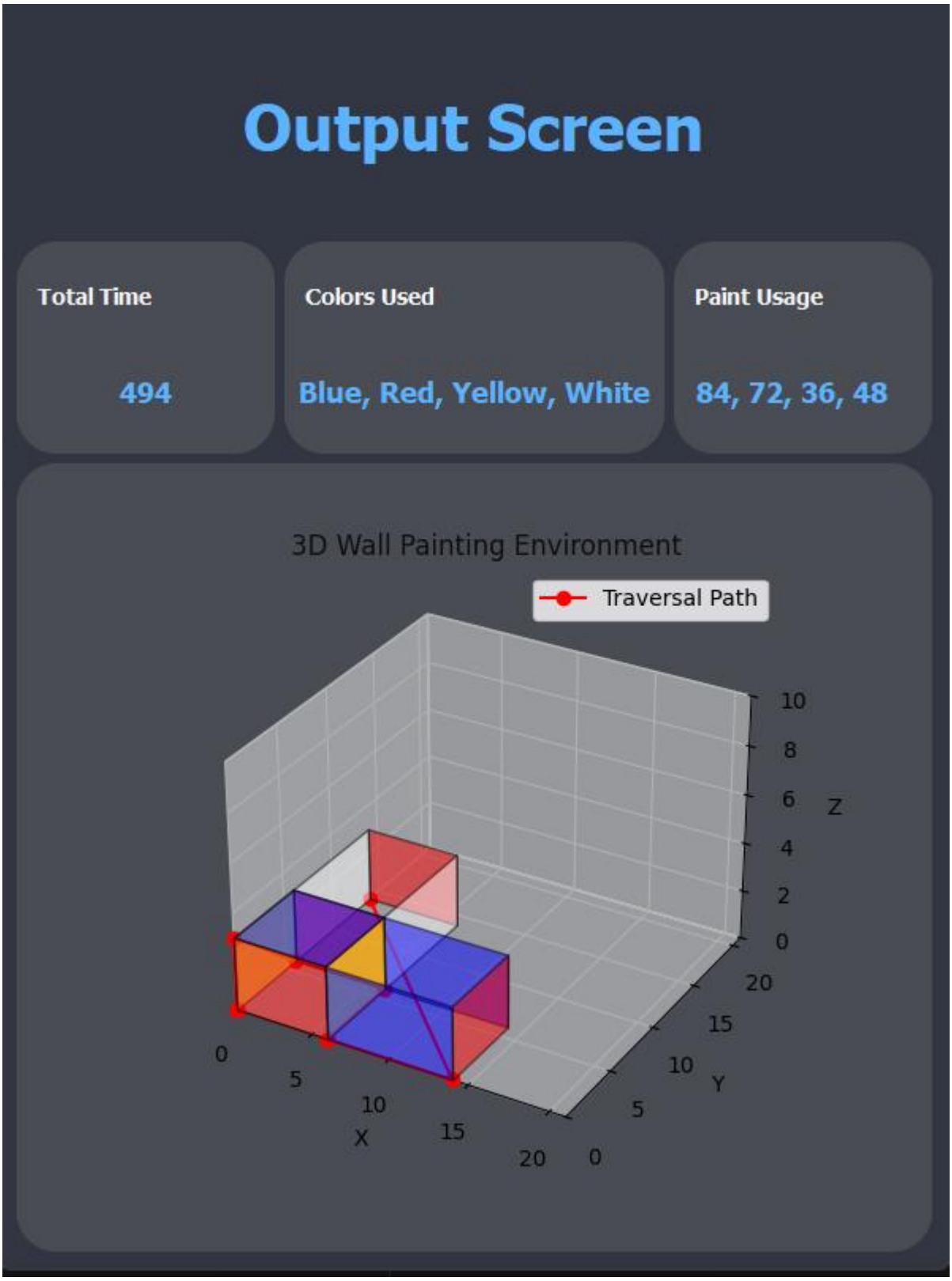
display_solutions

Outputs solutions in a readable format:

- Displays colors used, total time, paint usage, and traversal path.
- Invokes the visualization function for the best solution.

Visualization

- Walls are rendered in their assigned colors.
- The traversal path is displayed as a red line connecting the walls.

- 
Output Screen
Total Time
494
Colors Used
Blue, Red, Yellow, White
Paint Usage
84, 72, 36, 48
3D Wall Painting Environment
Traversal Path
X
Y
Z
0
5
10
15
20
0
5
10
15
20
0
2
4
6
8
10

Error Handling

- Invalid Inputs
- Validates user inputs in the GUI.
- Displays popup messages for errors (e.g., missing fields, invalid formats).

Conclusion

WALLE provides an efficient solution for optimizing wall painting tasks under strict constraints. With its robust CSP solver and user friendly GUI, the application streamlines the process of inputting data, solving the problem, and visualizing results in a 3D environment.