Spike: 06
Title: Navigation with Graphs and Agent Terrain Preferences

Author: Marco Giacoppo, 104071453

Goals / deliverables:

1. A game world that is divided into a larger number of navigation tiles, and corresponding larger navigation graph structure.
2. A path-planning system that can create paths for agents, based on the current dynamic environment, using cost-based heuristic algorithms that accounts for at least six types of 'terrain' (i.e. nodes with different costs).
3. Demonstrate multiple independent moving agent characters (at least four) that are able to each follow their own independent paths.
4. Demonstrate at least two different types of agents that navigate the world differently

Technologies, Tools, and Resources used:

- Python 3.11
- Pyglet (for 2D graphics and real-time updates)
- VSCode + Terminal for development
- Sample terrain map (map1.txt)

Tasks undertaken:

1. **Initial Setup**
   - Extracted and examined the base Pyglet navigation project.
   - Understood class responsibilities: BoxWorld, Agent, Game, graphics.py, etc.
2. **Terrain System Design**
   - Defined six terrain types: CLEAR (.), MUD (m), WATER (~), SAND (s), FOREST (f), ROCK (r).
   - Assigned symbols, colours, and movement cost matrices in box_world.py.
   - Ensured terrain type updates dynamically via mouse input.
3. **Interactive World Editing**
   - Implemented MouseModes in game.py to allow placing terrain tiles or setting start/target nodes with keys 1–8.
   - Visual updates rendered using coloured rectangles per tile type.
4. **Agent Creation & Configuration**
   - Defined 4 unique agents (Sand, Mud, Water,Forest) in game.py.

- Each agent has a name, visual colour, and terrain_costs dictionary to reflect preferences.
- Agents initially had hardcoded start/goal — later updated to use user-defined tiles.

```
# ✅ Define 4 different agents
self.agents = [
    Agent("Sand", agent_type="Sand", terrain_costs={
        "SAND": 1, "MUD": 6, "WATER": 8, "CLEAR": 5, "FOREST": 7, "ROCK": float("inf")
    }),
    Agent("Mud", agent_type="Mud", terrain_costs={
        "MUD": 1, "CLEAR": 3, "SAND": 5, "FOREST": 8, "WATER": 10, "ROCK": float("inf")
    }),
    Agent("Water", agent_type="Water", terrain_costs={
        "WATER": 1, "CLEAR": 3, "SAND": 6, "FOREST": 8, "MUD": 10, "ROCK": float("inf")
    }),
    Agent("Forest", agent_type="Forest", terrain_costs={
        "FOREST": 1, "CLEAR": 3, "MUD": 6, "WATER": 10, "SAND": 8, "ROCK": float("inf")
    }),
]
```

Figure 1: Each agent is defined with a unique name and terrain cost preferences. The values reflect how favourable each terrain is to that agent.

5. **Path Planning with A***
   - Integrated A* search from searches.py.
   - Modified agent.py to assign terrain-specific edge costs during pathfinding.
   - Ensured paths adapt dynamically to agent behaviour and terrain cost mapping.
6. **Smooth Movement System**
   - Implemented per-frame interpolation (_interpolate()) for agent movement.
   - Added move_delay and progress tracking for smooth transitions between tiles.
   - Each agent visually moves across the map according to their path.

```python
def move_along_path(self, world, dt):
    if self.done:
        return

    self.time_since_move += dt

    if self.progress < 1.0:
        # Still interpolating between previous and target
        self.progress = min(self.time_since_move / self.move_delay, 1.0)
        self.position = self._interpolate(self.prev_pos, self.target_pos, self.progress)
        return

    # Fully reached target, prepare next move
    if self.step >= len(self.path):
        self.done = True
        return

    # Begin moving to next tile
    self.current_idx = self.path[self.step]
    self.step += 1
    self.time_since_move = 0.0
    self.progress = 0.0

    tile = world.get_box_by_index(self.current_idx)
    self.prev_pos = self.position
    self.target_pos = tile.center().x, tile.center().y

def _interpolate(self, start, end, t):
    x = start[0] + (end[0] - start[0]) * t
    y = start[1] + (end[1] - start[1]) * t
    return (x, y)
```

Figure 2: Agent movement is interpolated over time between tiles to create smooth transitions, rather than jumping tile to tile.

7. Debug Logging

- Output each agent's full path with index, terrain type, and step cost.
- Calculated and printed total cost per path.
- Helped verify agent choices and troubleshoot unreachable goals.

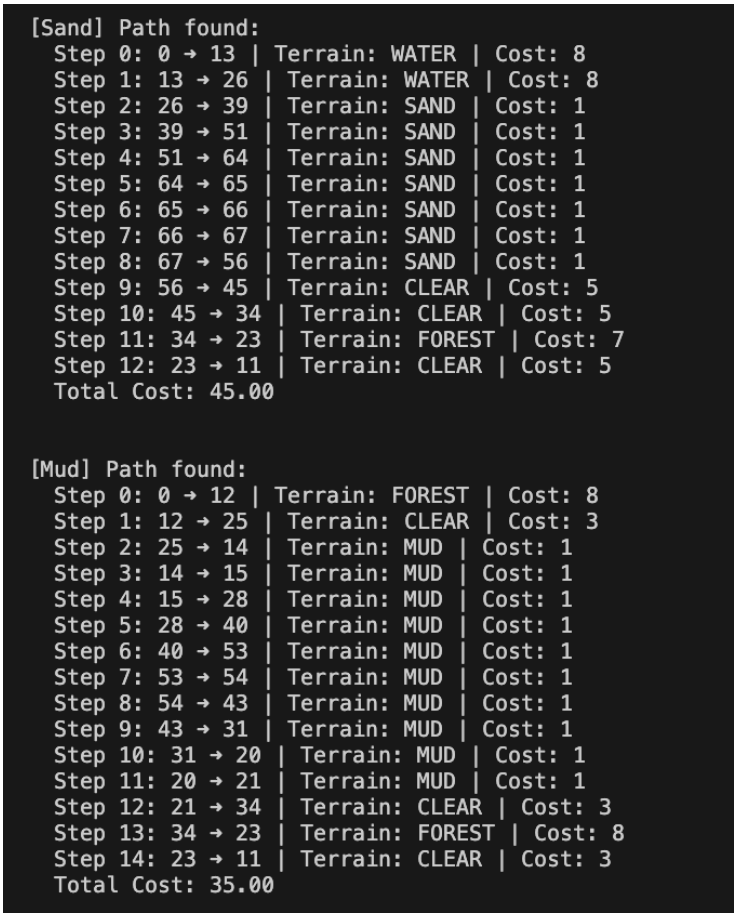

Figure 3: Logged path output shows step by step terrain, cost, and index. Useful for debugging and analysis.

8. Map Design & Testing

- Created multiple versions of map1.txt to test different agent behaviours:
  - Single terrain tunnels
  - Mixed terrain mazes
  - Rock barriers to block paths
  - Complex maps to force fallback decisions

9. Visual Output Enhancements

- Used coloured circles to represent agents.
- Rendered paths with distinct lines per agent type.
- Added on-screen labels and optional debug features (edges, paths, node numbers).

```python
def draw_agents(agents):
    type_colors = {
        "Sand": COLOUR_NAMES['LIGHT_BLUE'],
        "Mud": COLOUR_NAMES['ORANGE'],
        "Water": COLOUR_NAMES['LIGHT_GREEN'],
        "Forest": COLOUR_NAMES['PURPLE']
    }

    for agent in agents:
        # Draw the path line
        if len(agent.path) > 1:
            for i in range(len(agent.path) - 1):
                from_box = agent.world.get_box_by_index(agent.path[i])
                to_box = agent.world.get_box_by_index(agent.path[i + 1])

                x1, y1 = from_box.center().x, from_box.center().y
                x2, y2 = to_box.center().x, to_box.center().y

                line = pyglet.shapes.Line(
                    x1, y1, x2, y2,
                    thickness=2,
                    color=type_colors.get(agent.agent_type, (150, 150, 150))
                )
                line.draw()

        # Draw the agent circle
        if agent.position:
            circle = pyglet.shapes.Circle(
                x=agent.position[0],
                y=agent.position[1],
                radius=6,
                color=type_colors.get(agent.agent_type, (255, 0, 0))
            )
            circle.draw()

            label = pyglet.text.Label(
                agent.name,
                x=agent.position[0] + 8,
                y=agent.position[1] + 8,
                font_size=10,
                color=(0, 0, 0, 255)
            )
            label.draw()
```

Figure 4: Each agent is drawn as a coloured circle based on its type, allowing visual distinction during movement.

10. Report Writing

- Compiled summaries, screenshots, and code explanations for documentation.
- Clarified agent behaviour via annotated output logs.

What we found out:

- For the first deliverables, I've implemented a flexible tile-based world system using BoxWorld, supporting dynamic map sizes (12 x 10). Each tile became a node in a corresponding sparse navigation graph. The system automatically updates the graph structure as terrain is edited by the user.
- For the second one, I've integrated A* search as the main path-planning system. Terrain types (CLEAR, MUD, WATER, SAND, FOREST, ROCK) were defined with adjustable costs. Agents calculate the best path using both heuristic distance and the cost of the terrain, resulting in adaptive, intelligent pathfinding that reflects the current state of the environment.

- Third one, I've created four agents (Sand, Mud, Water, Forest), each initialized with different preferences. They independently calculate and follow their paths based on terrain cost and current map layout. Their movements are rendered smoothly and updated in real time using frame-by-frame interpolation.
- And lastly, the agents have completely unique behaviours: for example, the Water agent prefers WATER, while the Mud agent favors MUD. These differences cause them to select different paths even when given the same start and goal. This diversity in navigation showcases how terrain-aware agents can simulate unique roles or strategies.

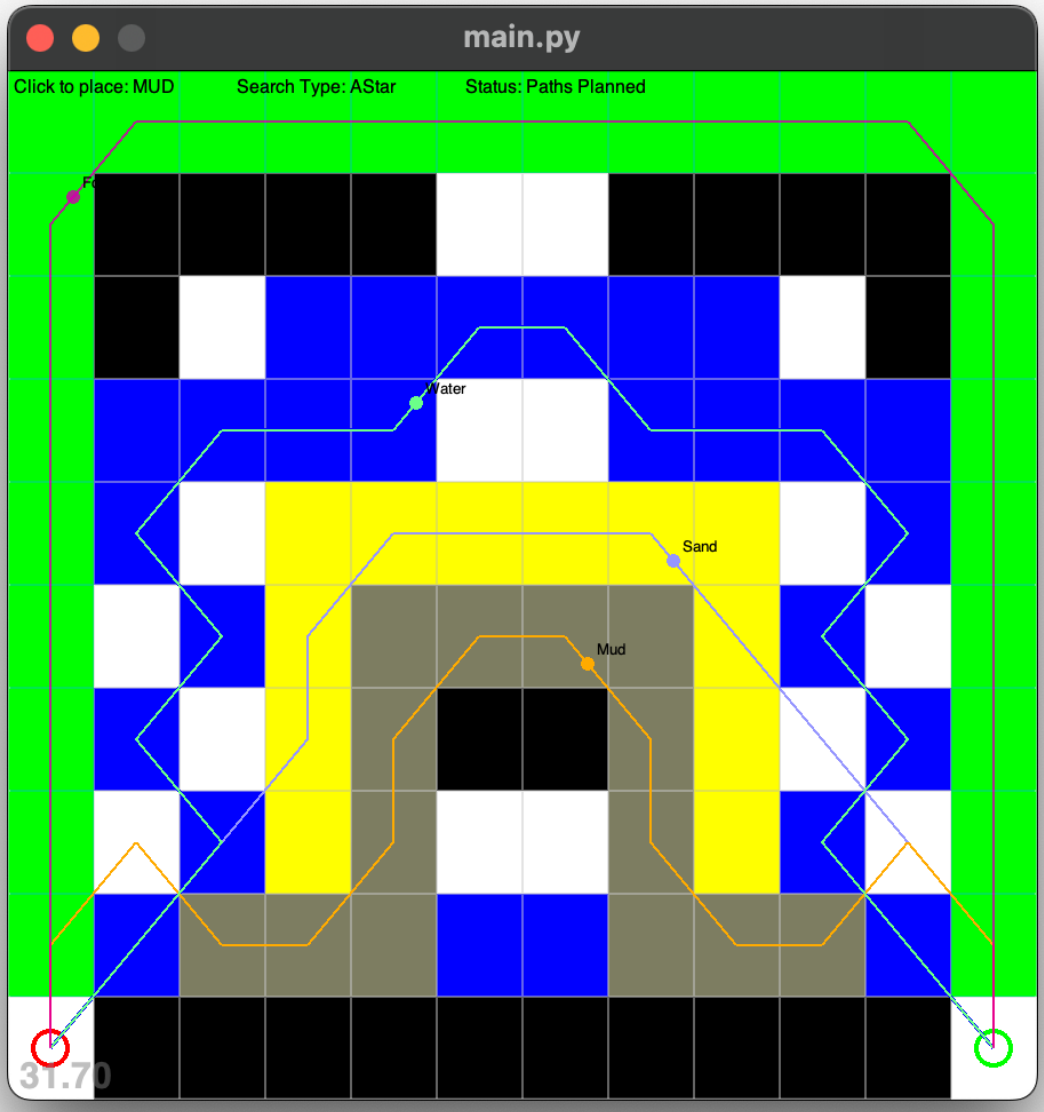

Figure 5: Fully running system.