CATERPILAR CONTROL SYSTEM IMPLEMENTATION

Presented by:	Dancan Mwanthi
Total time:	75.5 hours

CONTENTS

I. OBJECTVE

- To create a control system in C# for managing the movement and actions of a giant caterpillar on a distant planet. The system should allow the rider to navigate the caterpillar, collect spice and boosters, avoid obstacles, and control the caterpillar's growth.

II. IMPLEMENTATION PLAN

1. **Understanding the Requirements:** Analyze the provided mission brief thoroughly to understand the functionalities and constraints of the caterpillar control system.

2. **Designing the System Architecture:** Plan the structure of the control system, including classes, methods, and data structures required to implement the functionalities.

   - Manually added the map from the test to the source

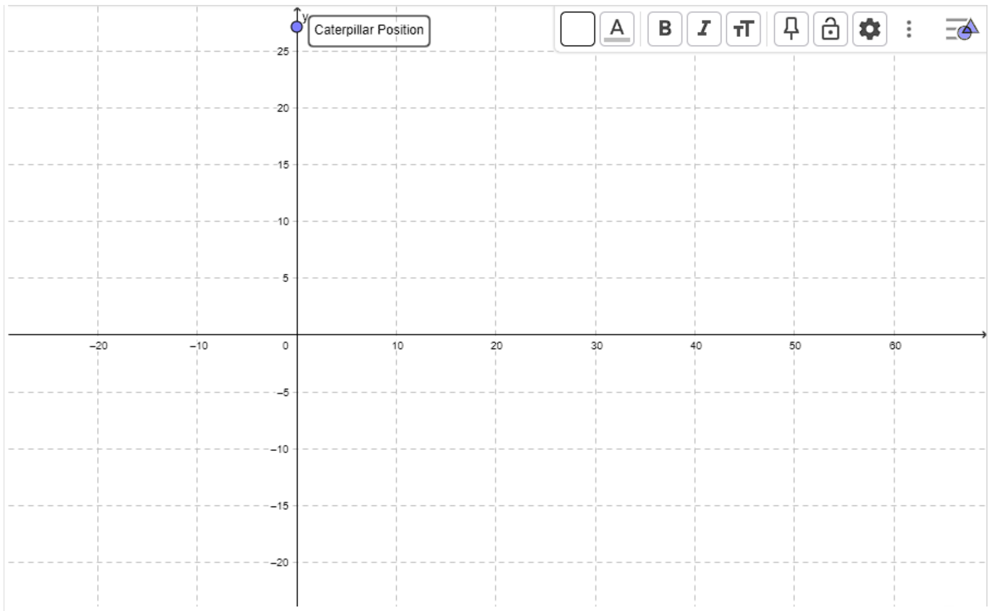
   Total time: 4 hours

3. **Implementing Caterpillar Movement:** Develop algorithms to handle the movement of the caterpillar based on rider commands while ensuring that a mirrored map is created whenever the caterpillar is at the edge of the map.

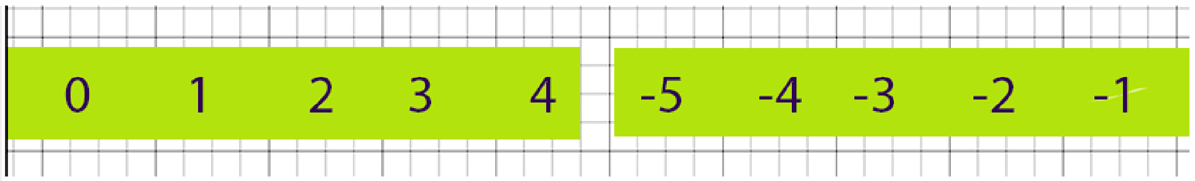
   - **Issue:** How to track the caterpillar on the planet's 2D surface?
     - Used a 3D array to track both the x-axis and y-axis positions from the map and also the x-axis and y-axis of the cartesian co-ordinate system.

       ```
       int[,,] Radar_3d = new int[_scope_Diameter, 2, _scope_Diameter];
       ```

   - **Issue:** How to mirror the map if the caterpillar is at the edge?
     - Developed an algorithm to utilize the cartesian co-ordinate system and mapped the caterpillar based on its axis position as shown below.



- Whenever the caterpillar is detected to be on the edge the model it collects the negative axis position and adds it to the max width and height which is 30m; this gives us the mirrored map by 30 m^2 .



Total time: 2 days

4. **Incorporating Spice and Booster Collection:** Implement mechanisms to detect and handle interactions with spice ($) and boosters (B) on the planet's surface.

   - **Issue:** How to remove a Spice or Booster from the planets surface if consumed by the caterpillar?

     - Modified the model to update the map's 3D array whenever the caterpillar's head is detected on a Spice or Booster; modified it with an empty field.

   Total time: 3 hours

5. **Handling Obstacles:** Develop logic to prevent the caterpillar from moving through obstacles and to handle collisions appropriately.

   - Resolved this by terminating the application whenever the caterpillar is detected to be on an obstacle (#); this signifies the caterpillar has disintegrated.

   Total time: 1 hour

6. **Managing Caterpillar Growth:** Implement functionality to control the growth of the caterpillar based on the consumption of boosters, ensuring the limit of five segments is not exceeded.

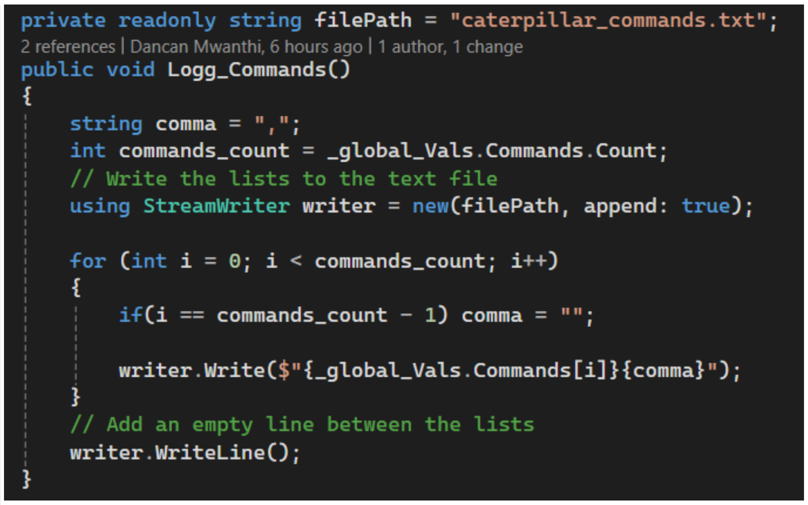
   - **Issue:** How to move the segments relative to the caterpillar's head?

     - Since the head's position is always known I kept track of the commands the rider gave and updated the segments respective to the direction and steps provided.

     Total time: 5 hours

7. **Logging Commands:** Create a logging system to record all rider commands for later analysis, including the ability to undo or redo commands.

     - Every successful command is updated into memory. On the applications termination a document labeled caterpillar_commands.txt is updated with the commands.



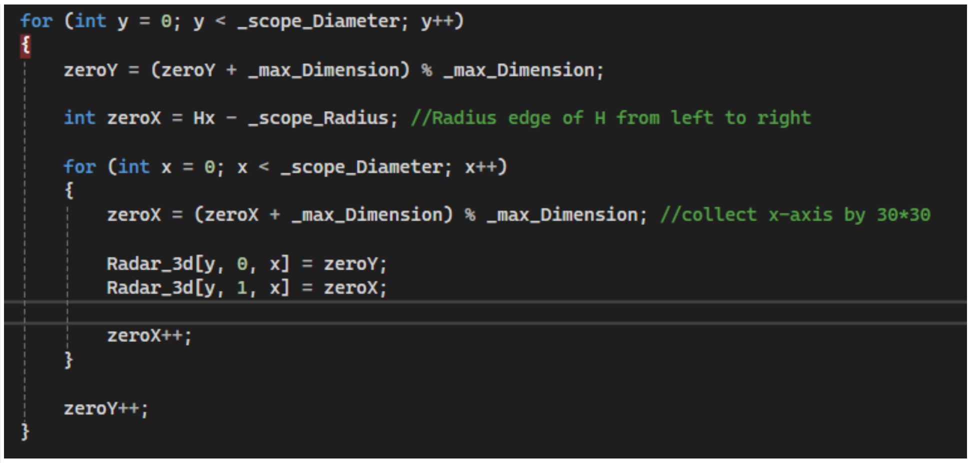
```
private readonly string filePath = "caterpillar_commands.txt";
2 references | Dancan Mwanthi, 6 hours ago | 1 author, 1 change
public void Logg_Commands()
{
    string comma = ",";
    int commands_count = _global_Vals.Commands.Count;
    // Write the lists to the text file
    using StreamWriter writer = new(filePath, append: true);

    for (int i = 0; i < commands_count; i++)
    {
        if(i == commands_count - 1) comma = "";

        writer.Write($"{_global_Vals.Commands[i]}{comma}");
    }
    // Add an empty line between the lists
    writer.WriteLine();
}
```

     - The rider is offered the option to undo a command or commands. This removes the undone command or commands from the collection memory.

     Total time: 1.5 hours

8. **Implementing Radar Display:** Develop a display system to visualize the radar image around the caterpillar's head, showing obstacles, spice, boosters, and caterpillar segments.

     - Developed a model that only collect sections of the map within the 11 meters diameter and the caterpillars head as the center.

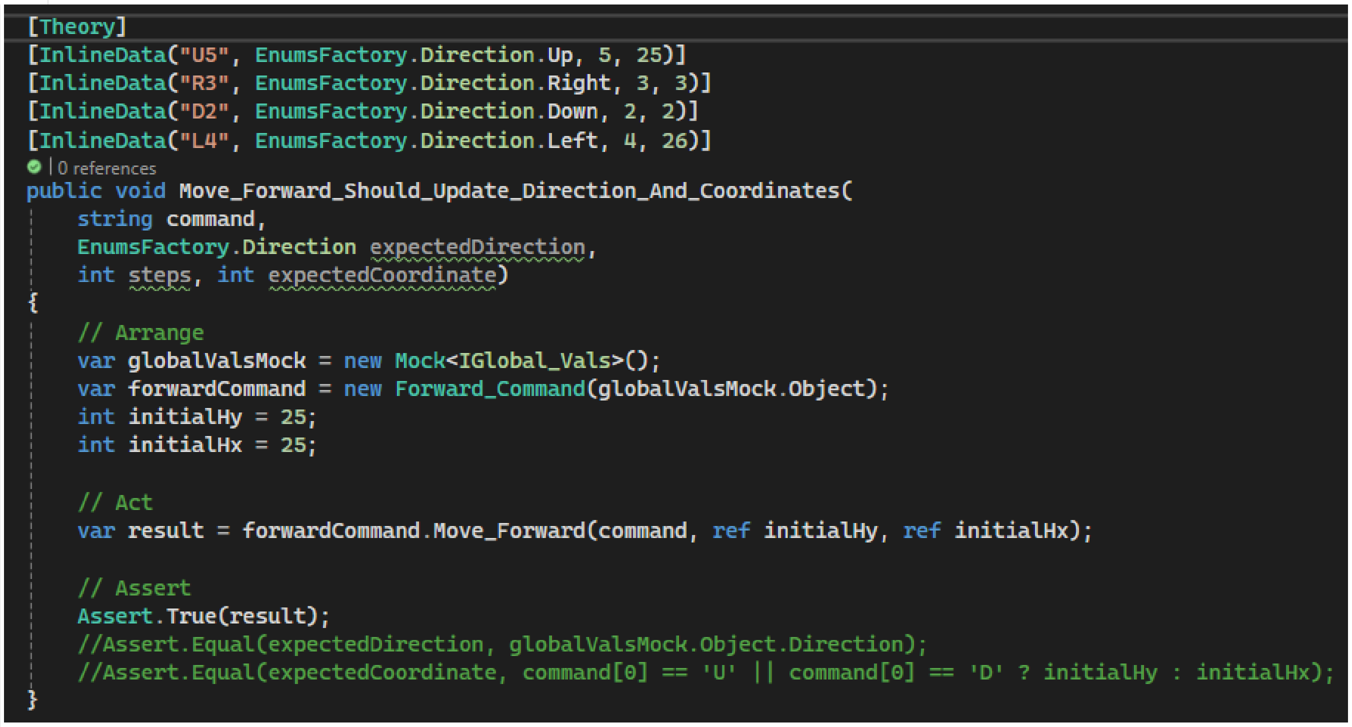


```
for (int y = 0; y < _scope_Diameter; y++)
{
    zeroY = (zeroY + _max_Dimension) % _max_Dimension;

    int zeroX = Hx - _scope_Radius; //Radius edge of H from left to right

    for (int x = 0; x < _scope_Diameter; x++)
    {
        zeroX = (zeroX + _max_Dimension) % _max_Dimension; //collect x-axis by 30*30

        Radar_3d[y, 0, x] = zeroY;
        Radar_3d[y, 1, x] = zeroX;

        zeroX++;
    }

    zeroY++;
}
```

Total time: 7 hours

9. **Testing and Debugging:** Write automated tests to validate each component of the system and debug any issues that arise.

   - Created an automated test to test moving forward with the caterpillar



```
[Theory]
[InlineData("U5", EnumsFactory.Direction.Up, 5, 25)]
[InlineData("R3", EnumsFactory.Direction.Right, 3, 3)]
[InlineData("D2", EnumsFactory.Direction.Down, 2, 2)]
[InlineData("L4", EnumsFactory.Direction.Left, 4, 26)]
| 0 references
public void Move_Forward_Should_Update_Direction_And_Coordinates(
    string command,
    EnumsFactory.Direction expectedDirection,
    int steps, int expectedCoordinate)
{
    // Arrange
    var globalValsMock = new Mock<IGlobal_Vals>();
    var forwardCommand = new Forward_Command(globalValsMock.Object);
    int initialHy = 25;
    int initialHx = 25;

    // Act
    var result = forwardCommand.Move_Forward(command, ref initialHy, ref initialHx);

    // Assert
    Assert.True(result);
    //Assert.Equal(expectedDirection, globalValsMock.Object.Direction);
    //Assert.Equal(expectedCoordinate, command[0] == 'U' || command[0] == 'D' ? initialHy : initialHx);
}
```

Total time: 6hours