Lost in the Woods - User Manual

Welcome to Lost in the Woods, an interactive text adventure game where you find yourself trapped in a mysterious forest with no memory of how you got there. Your objective is to navigate through the forest, overcome obstacles, and ultimately find your way out to safety.

Link to Video: https://youtu.be/aqrMbVKmQo0

Gameplay Overview:

Upon launching the game, you'll be presented with a text-based interface that provides information about your surroundings, your health status, and available options to progress through the game. Here's a brief overview of how to play:

1. Initial Message:

- When you start the game, you'll see an initial message welcoming you to Lost in the Woods. Read through the message to understand your situation and the objective of the game.

2. Exploration:

- You'll be prompted to make a choice between different actions, such as exploring the forest or resting. Use the provided options (usually labeled A and B) to select your desired action.

3. Decision Making:

- Based on your choice, the game will update your location, display any relevant descriptions or events, and present you with new choices. Pay attention to the narration and descriptions to understand the consequences of your decisions. Press A or B to make Decisions.

4. Inventory:

- Throughout the game, you may collect items that could aid you in your journey. Your inventory will be displayed on the screen, showing the items you've acquired so far.

5. Health Status:

- Your health status will be displayed to indicate your current condition. Be cautious, as certain actions may affect your health positively or negatively.

6. Progression:

- Continue making choices and exploring different paths to progress through the game. Remember, your goal is to find a way out of the forest.

7. End of Game:

- The game will conclude when you reach the end of your journey, either by successfully escaping the forest or encountering a scenario that ends your adventure.

8. Replay:

- After reaching the end, you'll have the option to replay the game and embark on a new adventure. When prompted, input Y/N to either replay the game or close the game.

Game Mechanics:

- **Command Input:** Enter your choice using the provided text input field and click the "Submit" button to proceed.
- **Narration:** The game will provide descriptive text to narrate events, locations, and outcomes based on your decisions.
- **Choice Selection:** Choose between available options (labeled A and B) to determine your actions and progress through the game.
- **Inventory Management:** Keep track of the items you've collected in your inventory, as they may be useful later in the game.
- **Health Management:** Pay attention to your health status, as it can impact your ability to overcome challenges and make progress.

Enjoy your adventure in Lost in the Woods, and good luck finding your way out!

Form Design and Layout:

1. Set background image: **BackgroundImage**, **BackgroundImageLayout**
2. Set form size: **Size**
3. Adjust control positions and sizes:
    - **lblNarration**: Location, Size
    - **lblHealth**: Location
    - **lblLocation**: Location
    - **lblDescription**: Location, Size
    - **lstInventory**: Location, Size
    - **txtCommand**: Location, Size
    - **btnSubmit**: Location
    - **txtOutput**: Location, Size

Initialization and Display:

1. **ShowInitialMessage()**: Display initial welcome message and options.
2. **DisplayGameOutput()**: Display current game state, including narration, health, location, description, inventory, and choices.
3. **GetNarration()**: Get narration text based on current game state.
4. **DisplayInventory()**: Display inventory items.
5. **DisplayChoices()**: Display available choices based on current game state.
6. **AppendResponse()**: Append response text to the output textbox.

Command Processing:

1. **ProcessCommand()**: Main method for processing user commands.
2. Command processing methods for each game state:
    - **ProcessIntroductionCommand()**
    - **ProcessForestCommand()**
    - **ProcessRiverCommand()**
    - **ProcessCaveCommand()**
    - **ProcessCaveAdditionalCommand()**
    - **ProcessMountainCommand()**
    - **ProcessEndCommand()**
    - **ProcessMountainTurnBackCommand()**
3. **ResetGame()**: Reset the game state and display initial message.

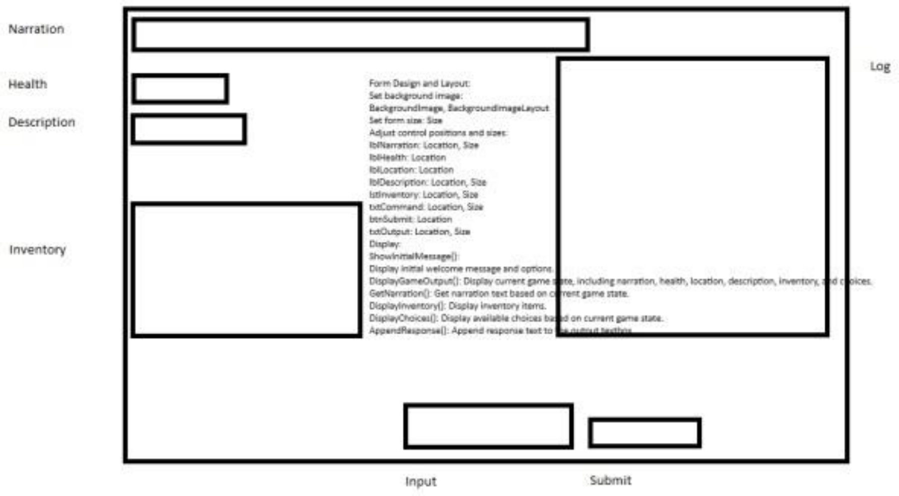
Button Click Event:

1. **btnSubmit_Click()**: Event handler for the submit button click, retrieves the command from the textbox and processes it.

Miscellaneous:

1. Enum **GameState**: Defines the different states of the game.
2. Class variables: **health**, **location**, **description**, **inventory**, **gameState**: Maintain game state and player information.
3. Form constructor: Initializes form properties and controls, sets background image.
4. Image loading: Load background image from file.

Mockup for GUI:



Form Design and Layout:
Set background image:
BackgroundImage, BackgroundImageLayout
Set form size: Size
Adjust control positions and sizes:
lblNarration: Location, Size
lblHealth: Location
lblLocation: Location
lblDescription: Location, Size
lstInventory: Location, Size
txtCommand: Location, Size
btnSubmit: Location
txtOutput: Location, Size
Display:
ShowInitialMessage():
Display initial welcome message and options.
DisplayGameOutput(): Display current game state, including narration, health, location, description, inventory, and choices.
GetNarration(): Get narration text based on current game state.
DisplayInventory(): Display inventory items.
DisplayChoices(): Display available choices based on current game state.
AppendResponse(): Append response text to the output textbox.

Project Code:

Form1.cs:

```
using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;


namespace TextAdventureGame

{

    public partial class Form1 : Form

    {

        private int health = 100;

        private string location = "Forest";

        private string description = "You find yourself in a dense forest. Trees surround you, a path
leads deeper into the woods.";

        private List<string> inventory = new List<string>();


        private enum GameState

        {

            Introduction,

            Forest,

            River,

            Cave,

            CaveAdditional,

            Mountain,

            End,

            MountainTurnBack

        }


        private GameState gameState = GameState.Introduction;
```

```
public Form1()
{
    InitializeComponent();
    this.BackgroundImage = Image.FromFile(@"C:\Users\imdri\Desktop\R.jpg");
    this.BackgroundImageLayout = ImageLayout.Stretch;
    this.Size = new Size(1000, 600);

    // Adjust control positions and sizes
    lblNarration.Location = new Point(20, 20);
    lblNarration.Size = new Size(700, 50);
    lblHealth.Location = new Point(20, 80);
    lblLocation.Location = new Point(20, 110);
    lblDescription.Location = new Point(20, 140);
    lblDescription.Size = new Size(300, 100);
    lstInventory.Location = new Point(20, 270);
    lstInventory.Size = new Size(150, 100);
    txtCommand.Location = new Point(250, 450);
    txtCommand.Size = new Size(150, 20);
    btnSubmit.Location = new Point(420, 448);
    txtOutput.Location = new Point(600, 80);
    txtOutput.Size = new Size(350, 470);

    ShowInitialMessage();
    DisplayGameOutput();
}

private void ShowInitialMessage()
{
    AppendResponse("Welcome to Lost in the Woods!");
```

```
            AppendResponse("You wake up in the midst of a dense forest, disoriented and confused. You have no memory of how you got here or why. All you know is that you must find a way out of this unfamiliar place.");
            AppendResponse("What will you do?");
            AppendResponse("A) Explore the forest");
            AppendResponse("B) Rest for a while");
        }

        private void DisplayGameOutput()
        {
            lblNarration.Text = GetNarration();
            lblHealth.Text = $"Health: {health}";
            lblLocation.Text = $"Location: {location}";
            lblDescription.Text = $"Description: {description}";
            DisplayInventory();
            DisplayChoices();
            txtCommand.Focus();
        }

        private string GetNarration()
        {
            switch (gameState)
            {
                case GameState.Forest:
                    return "How did this happen?";
                case GameState.River:
                    return "You come across a swift river blocking your path. What's your next move?";
                case GameState.Cave:
                    return "From the bridge you look up, seeing the entrance to a small cave.";
                case GameState.CaveAdditional:
                    return "You light a nearby torch and enter the cave.\nWhat will you do?";
                case GameState.Mountain:
                    return "Is this heaven?";
                case GameState.End:
                    return "You have reached the end of your journey.\nWould you like to play again?
(Y/N)";
                case GameState.MountainTurnBack:
                    return "You decide to turn back and explore other paths.\nWould you like to turn
back and explore other paths? (Y/N)";
                default:
                    return "";
            }
        }

        private void DisplayInventory()
        {
            lstInventory.Items.Clear();
            foreach (var item in inventory)
            {
                lstInventory.Items.Add($"· {item}");
            }
        }

        private void DisplayChoices()
        {
            switch (gameState)
            {
                case GameState.Forest:
                    DisplayForestChoices();
                    break;
                case GameState.River:
                    DisplayRiverChoices();
```

```
}

            break;
        case GameState.Cave:
            DisplayCaveChoices();
            break;
        case GameState.CaveAdditional:
            DisplayCaveAdditionalChoices();
            break;
        case GameState.Mountain:
            DisplayMountainChoices();
            break;
        case GameState.End:
            DisplayEndChoices();
            break;
        case GameState.MountainTurnBack:
            DisplayMountainTurnBackChoices();
            break;
    }
}

private void DisplayForestChoices()
{
    AppendResponse("A) Follow the path deeper into the woods");
    AppendResponse("B) Climb a tree to get a better view");
}

private void DisplayRiverChoices()
{
    AppendResponse("A) Try to swim across the river");
    AppendResponse("B) Look for a bridge upstream");
}
```

```
        private void DisplayCaveChoices()

        {

            AppendResponse("A) Light a torch and enter the cave");

            AppendResponse("B) Stay outside and find another path");

        }


        private void DisplayCaveAdditionalChoices()

        {

            AppendResponse("A) Continue going deeper, climbing up through the seemingly
expanding cave system");

            AppendResponse("B) Turn around and exit the cave");

        }


        private void DisplayMountainChoices()

        {

            AppendResponse("A) Keep climbing to reach the top");

            AppendResponse("B) Descend and explore the foothills");

        }


        private void DisplayEndChoices()

        {

            AppendResponse("Would you like to play again? (Y/N)");

        }


        private void DisplayMountainTurnBackChoices()

        {

            AppendResponse("Would you like to turn back and explore other paths? (Y/N)");

        }
```

```
private void AppendResponse(string response)
{
    txtOutput.AppendText(response + Environment.NewLine + Environment.NewLine);
}

private void ProcessCommand(string command)
{
    command = command.ToLower();
    switch (gameState)
    {
        case GameState.Introduction:
            ProcessIntroductionCommand(command);
            break;
        case GameState.Forest:
            ProcessForestCommand(command);
            break;
        case GameState.River:
            ProcessRiverCommand(command);
            break;
        case GameState.Cave:
            ProcessCaveCommand(command);
            break;
        case GameState.CaveAdditional:
            ProcessCaveAdditionalCommand(command);
            break;
        case GameState.Mountain:
            ProcessMountainCommand(command);
            break;
        case GameState.End:
            ProcessEndCommand(command);
```

```
                break;
            case GameState.MountainTurnBack:
                ProcessMountainTurnBackCommand(command);
                break;
        }
    }


    private void ProcessIntroductionCommand(string command)
    {
        if (command == "a")
        {
            gameState = GameState.Forest;
            description = "You find yourself in a dense forest. Trees surround you, a path leads
deeper into the woods.";
        }
        else if (command == "b")
        {
            health += 10; // Assuming resting increases health
            AppendResponse("You decide to rest for a while, recuperating your strength.");
            gameState = GameState.Forest;
            description = "You find yourself in a dense forest. Trees surround you, a path leads
deeper into the woods.";
        }
        else
        {
            AppendResponse("Please choose between A and B.");
            return;
        }


        DisplayGameOutput();
    }
```

```
        private void ProcessForestCommand(string command)
        {
            if (command.StartsWith("a"))
            {
                gameState = GameState.River;
                description = "You follow the path deeper into the woods, hoping to find a way out.";
            }
            else if (command.StartsWith("b"))
            {
                inventory.Add("Map");
                AppendResponse("You climb a tree, scanning the surroundings you mentally map out
the area.");
            }
            else
            {
                AppendResponse("Please choose between A and B.");
                return;
            }

            DisplayGameOutput();
        }

        private void ProcessRiverCommand(string command)
        {
            if (command == "a")
            {
                health = 0;
                gameState = GameState.End;
                description = "You try to swim across the river but get caught in the strong current.
You drown.";
```

```
            }
            else if (command == "b")
            {
                inventory.Add("Rope");
                AppendResponse("You look for a bridge upstream and find a sturdy rope tied to the
railing of the bridge.");
                gameState = GameState.Cave;
                description = "You decide to look for a bridge upstream.";
            }
            else
            {
                AppendResponse("Please choose between A and B.");
                return;
            }

            DisplayGameOutput();
        }

        private void ProcessCaveCommand(string command)
        {
            if (command == "a")
            {
                inventory.Add("Torch");
                AppendResponse("You light a nearby torch and enter the cave.");
                gameState = GameState.CaveAdditional;
                description = "You light a nearby torch and enter the cave.";
            }
            else if (command == "b")
            {
                AppendResponse("You decide to stay outside and find another path.");
                gameState = GameState.Forest;
                description = "You find yourself in a dense forest. Trees surround you, a path leads
deeper into the woods.";
            }
            else
            {
                AppendResponse("Please choose between A and B.");
                return;
            }

            DisplayGameOutput();
        }

        private void ProcessCaveAdditionalCommand(string command)
        {
            if (command.StartsWith("a"))
            {
                gameState = GameState.Mountain;
                description = "You climb toward a faint light. You come out on a mountainside.";
            }
            else if (command.StartsWith("b"))
            {
                health = 0;
                gameState = GameState.End;
                description = "While exiting the cave, a snake bites you. You perish.";
            }
            else
            {
                AppendResponse("Please choose between A and B.");
                return;
            }

            DisplayGameOutput();
        }

        private void ProcessMountainCommand(string command)
        {
            if (!inventory.Contains("Rope") || !inventory.Contains("Map") ||
!inventory.Contains("Torch"))
            {
                AppendResponse("You don't have all the necessary items to proceed.");

                // Offer the player the option to turn back and explore other paths
                AppendResponse("Would you like to turn back and explore other paths? (Y/N)");
                gameState = GameState.MountainTurnBack;
                return;
            }

            if (command.StartsWith("a"))
            {
                // Proceed with climbing to reach the top
                health -= 70;
                AppendResponse("You keep climbing but slip and fall down the mountain, severely
injuring yourself.");
                gameState = GameState.End;
            }
            else if (command.StartsWith("b"))
            {
                AppendResponse("You descend and explore the foothills, eventually finding a way
out of the forest. Congratulations! You have successfully escaped!");
                gameState = GameState.End;
            }
            else
            {
                AppendResponse("Please choose between A and B.");
                return;
            }

            DisplayGameOutput();
        }

        private void ProcessEndCommand(string command)
        {
            command = command.ToLower();
            if (command == "y")
            {
                ResetGame();
            }
            else if (command == "n")
            {
                Close();
            }
            else
            {
                AppendResponse("Please choose between Y and N.");
            }
        }

        private void ProcessMountainTurnBackCommand(string command)
        {
            command = command.ToLower();
```

```
            if (command == "y")
            {
                ResetGame();
            }
            else if (command == "n")
            {
                gameState = GameState.Mountain;
                DisplayMountainChoices();
            }
            else
            {
                AppendResponse("Please choose between Y and N.");
            }
        }

        private void ResetGame()
        {
            health = 100;
            location = "Forest";
            description = "You find yourself in a dense forest. Trees surround you, a path leads
deeper into the woods.";
            inventory.Clear(); // Clear the inventory
            gameState = GameState.Introduction;
            ShowInitialMessage();
            DisplayGameOutput();
        }

        private void btnSubmit_Click(object sender, EventArgs e)
        {
            string command = txtCommand.Text.Trim();
            txtCommand.Clear();

            ProcessCommand(command);

        }

    }

}
```

Form1.Designer.cs

```
namespace TextAdventureGame
{
    partial class Form1
    {
        private System.ComponentModel.IContainer components = null;

        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        private void InitializeComponent()
        {
            this.lblNarration = new System.Windows.Forms.Label();
            this.lblHealth = new System.Windows.Forms.Label();
            this.lblLocation = new System.Windows.Forms.Label();
            this.lblDescription = new System.Windows.Forms.Label();
            this.lstInventory = new System.Windows.Forms.ListBox();
            this.txtCommand = new System.Windows.Forms.TextBox();
            this.btnSubmit = new System.Windows.Forms.Button();
            this.txtOutput = new System.Windows.Forms.TextBox();
            this.SuspendLayout();
            //
            // lblNarration
            //
            this.lblNarration.AutoSize = true;
            this.lblNarration.Location = new System.Drawing.Point(20, 20);
            this.lblNarration.Name = "lblNarration";
            this.lblNarration.Size = new System.Drawing.Size(0, 13);
            this.lblNarration.TabIndex = 0;
            //
            // lblHealth
            //
            this.lblHealth.AutoSize = true;
            this.lblHealth.Location = new System.Drawing.Point(20, 80);
            this.lblHealth.Name = "lblHealth";
            this.lblHealth.Size = new System.Drawing.Size(0, 13);
            this.lblHealth.TabIndex = 1;
            //
            // lblLocation
            //
            this.lblLocation.AutoSize = true;
            this.lblLocation.Location = new System.Drawing.Point(20, 110);
            this.lblLocation.Name = "lblLocation";
            this.lblLocation.Size = new System.Drawing.Size(0, 13);
            this.lblLocation.TabIndex = 2;
            //
            // lblDescription
            //
            this.lblDescription.AutoSize = true;
            this.lblDescription.Location = new System.Drawing.Point(20, 140);
            this.lblDescription.Name = "lblDescription";
            this.lblDescription.Size = new System.Drawing.Size(0, 13);
            this.lblDescription.TabIndex = 3;
            //
            // lstInventory
            //
            this.lstInventory.FormattingEnabled = true;
            this.lstInventory.Location = new System.Drawing.Point(20, 270);
            this.lstInventory.Name = "lstInventory";
            this.lstInventory.Size = new System.Drawing.Size(150, 95);
            this.lstInventory.TabIndex = 4;
            //
            // txtCommand
            //
            this.txtCommand.Location = new System.Drawing.Point(250, 450);
            this.txtCommand.Name = "txtCommand";
            this.txtCommand.Size = new System.Drawing.Size(150, 20);
            this.txtCommand.TabIndex = 5;
            //
            // btnSubmit
            //
            this.btnSubmit.Location = new System.Drawing.Point(420, 448);
            this.btnSubmit.Name = "btnSubmit";
            this.btnSubmit.Size = new System.Drawing.Size(75, 23);
            this.btnSubmit.TabIndex = 6;
            this.btnSubmit.Text = "Submit";
            this.btnSubmit.UseVisualStyleBackColor = true;
            this.btnSubmit.Click += new System.EventHandler(this.btnSubmit_Click);
            //
            // txtOutput
            //
            this.txtOutput.Location = new System.Drawing.Point(600, 80);
            this.txtOutput.Multiline = true;
            this.txtOutput.Name = "txtOutput";
            this.txtOutput.ReadOnly = true;
            this.txtOutput.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
            this.txtOutput.Size = new System.Drawing.Size(350, 470);
            this.txtOutput.TabIndex = 7;
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(984, 562);
            this.Controls.Add(this.txtOutput);
            this.Controls.Add(this.btnSubmit);
            this.Controls.Add(this.txtCommand);
            this.Controls.Add(this.lstInventory);
            this.Controls.Add(this.lblDescription);
```

```
            this.Controls.Add(this.lblLocation);
            this.Controls.Add(this.lblHealth);
            this.Controls.Add(this.lblNarration);
            this.Name = "Form1";
            this.Text = "Lost in the Woods";
            this.ResumeLayout(false);
            this.PerformLayout();
        }

        private System.Windows.Forms.Label lblNarration;
        private System.Windows.Forms.Label lblHealth;
        private System.Windows.Forms.Label lblLocation;
        private System.Windows.Forms.Label lblDescription;
        private System.Windows.Forms.ListBox lstInventory;
        private System.Windows.Forms.TextBox txtCommand;
        private System.Windows.Forms.Button btnSubmit;
        private System.Windows.Forms.TextBox txtOutput;
    }
}
```

Program.cs

```
using System;
using System.Windows.Forms;

namespace TextAdventureGame
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```