Week 11: TEXT-BASED GAMES IN C# (PART 1) - PREPARING YOUR WORKSPACE

Welcome to Your Journey into Text-Based Game Development Using C#

In this assignment, you will begin your journey into creating a **text-based adventure game** using the C# programming language. The goal of this first assignment is to prepare your development environment and organize your project for success. As with any creative project, especially in game development, proper preparation is the foundation of a smooth and efficient workflow.

This assignment is designed to help you understand the importance of workspace organization, project structuring, and essential setup steps before you dive into the actual game development process. By the end of this assignment, you will have a fully structured C# project ready to start building your text-based game in the next phase.

Understanding C# and Console Applications

C# is a powerful, object-oriented programming language used extensively in game development, desktop applications, and more. In this project, you'll leverage C# to build a text-based adventure game using a **Console Application**. A console application operates in a simple text window (the console), making it perfect for text-driven games where players interact with the game through written input.

Why Pre-Project Organization is Crucial

1. **Efficiency and Speed**: Organizing your project from the start will help you navigate your files and resources quickly as your project grows. Having a clear structure also makes adding new features and modifying existing ones much easier.
2. **Clarity and Collaboration**: Whether working individually or in a team, keeping your project organized ensures clarity. Anyone who accesses your project should be able to understand the structure and find files easily. This is especially important when collaborating or using version control systems like Git.
3. **Error Reduction**: A clean and organized workspace makes debugging and troubleshooting easier. When errors arise (and they will), a well-organized project allows you to isolate problems faster, making development more efficient.

Assignment Objectives

By the end of this assignment, you will have:

- Set up your development environment for C# text-based game development.
- Created a new **Console Application** project in Visual Studio or another IDE.
- Organized your project folders for easy file management.
- Documented your initial setup and project structure.

This preparatory assignment sets the stage for the next phase of development, which involves building your actual game mechanics, story structure, and player interaction.

What This Assignment Entails

Follow the steps below to prepare your workspace and development environment for your C# text-based game:

Step 1: Set Up Your Development Environment

1. **Install Visual Studio (or Visual Studio Code)**:
    - If you haven't already, download and install **Visual Studio Community Edition** or **Visual Studio Code**. Ensure that the **.NET Core SDK** is installed, as this will allow you to create C# projects.
    - **Download Links**:
        - Visual Studio: https://visualstudio.microsoft.com/
        - Visual Studio Code: https://code.visualstudio.com/
2. **Create a New Console Application**:
    - Once Visual Studio is set up, create a new project by selecting **"Console App (.NET Core)"** as the template.
    - Name your project something appropriate, like `TextAdventureGame`.
    - This console app will be the foundation of your game, where you'll code game logic, story progression, and handle player input/output.

Step 2: Organize Your Project Workspace

Now that you have created your console project, it's time to organize it for efficient development.

1. **Folder Structure**:
    - In your project, create the following folders to keep different components of your game separate and well-structured:
        - `/Scripts`: This is where all your C# scripts will go. Each script should serve a clear purpose, such as handling game states, story paths, or player choices.

        - `/Assets`: If you plan to include any external text files, data files (JSON or XML), or other assets, store them here.
        - `/Resources`: Place any additional resources, libraries, or documentation here that may be useful throughout the project.
        - `/Saves`: If you implement game saving/loading, use this folder to store saved game data.
2. **Naming Conventions**:
    - Adopt clear and consistent naming conventions for your files. For example:
        - Scripts: `GameState.cs`, `PlayerChoices.cs`, `StoryLoader.cs`
        - Assets: `story.json`, `game-over.txt`
    - This will help you and your collaborators quickly identify the purpose of each file.

Step 3: Document Your Project

1. **README File**:
    - Create a `README.txt` or `README.md` file in your project's root folder. In this file, document the purpose of your project, a brief description of its structure, and any important notes on how the project is set up.
2. **Version Control** (Optional, but Recommended):
    - Set up a Git repository for your project (e.g., using GitHub). This will allow you to track changes to your code, collaborate with others, and roll back to previous versions if needed.
    - **GitHub**: https://github.com/

Step 4: Research Story Assets and Data

1. **Choose a Story Database**:
    - Refer to the list of **text-based story databases** provided in class (e.g., Interactive Fiction Database, TwineHub, etc.). Choose a source where you can find stories or text-based adventure games in formats like JSON or plain text.
    - **Tip**: Download a simple text-based game story to use as a starting point for your project.
2. **Prepare to Load Story Data**:
    - If you are using a text-based file (e.g., `story.txt`), ensure it is formatted properly for easy reading in your console application.
    - In later assignments, you will write code to load this data and use it to guide the gameplay.

Step 5: Verify Your Setup

1. **Run Your Console App**:
    - Run your console app to verify that everything is working correctly. You should see a blank console window open up.
2. **Basic Interaction** (Optional):
    - As an extra challenge, try adding a simple `Console.WriteLine()` statement to output a welcome message or instructions to the player.

code:

```
Console.WriteLine("Welcome to the Text Adventure Game!");
```

3.

Next Steps and What's Coming Next

Congratulations! You've completed the foundational steps for your C# text-based game. In the next assignment, we'll dive deeper into **loading story data, building the game flow**, and handling player input.

Resources and Study Materials

- **Microsoft C# Documentation**: https://learn.microsoft.com/en-us/dotnet/csharp/
- **C# Console Application Tutorial**: C# Console App Guide
- **Text Adventure Game Tutorial** (YouTube): Build a Text-Based Game in C#

Let's get started! Be sure to follow the steps carefully, and reach out if you need help along the way.