DifficultyManager

Purpose

Controls the difficulty setting for the Firefly minigame. Handles both manual user selection and programmatic assignment of an advised difficulty.

Key Responsibilities

- Stores and exposes the currently selected difficulty (`SelectedDifficulty`).
- Optionally stores an advised difficulty (`AdvisedDifficulty`) for external reference.
- Connects to a UI dropdown for user interaction.

Core Elements

Singleton Access

```
public static DifficultyManager Instance;
```

- Allows global access via `DifficultyManager.Instance`.

Difficulty Tracking

- `SelectedDifficulty`: current active difficulty for gameplay.
- `AdvisedDifficulty`: an externally suggested difficulty (e.g. based on user performance).
- `Difficulty` is likely an enum with values: `Easy`, `Medium`, `Hard`, `None`.

UI Integration

```
ShowDropdown(bool show)
```

- Shows or hides the dropdown menu UI.

```
OnDropdownChanged()
```

- Called when dropdown value changes.
- Maps dropdown index (0–2) to the corresponding enum value.

```
SetAdvisedDifficulty(int index, Difficulty diff)
```

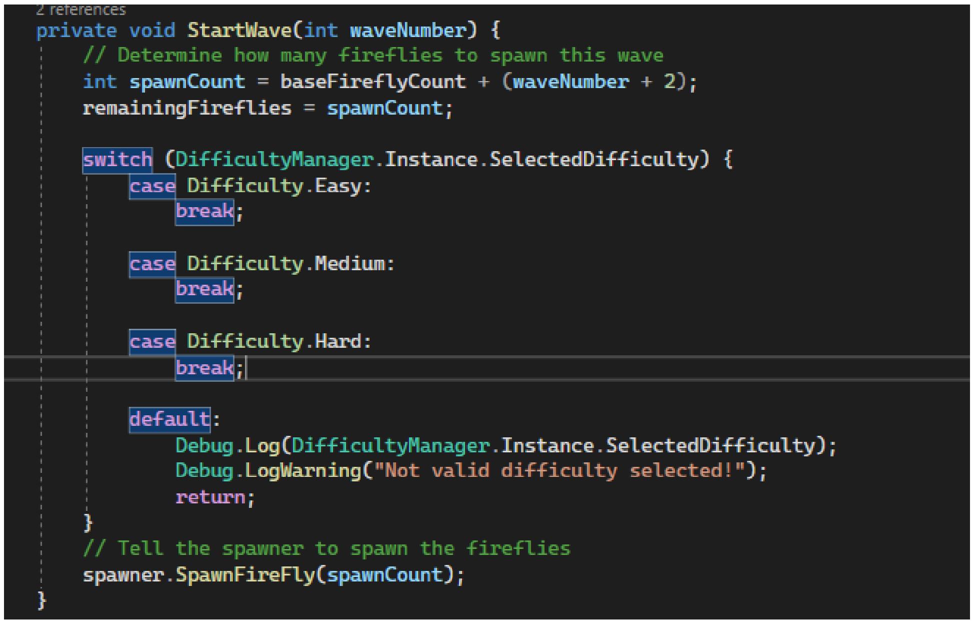
- Used to programmatically set both the dropdown UI and difficulty values.
- Syncs `AdvisedDifficulty` and `SelectedDifficulty` with the chosen index.

Dependencies

- Requires a `TMP_Dropdown` for difficulty selection.
- Expects a matching enum-to-index setup:
  - 0 = Easy
  - 1 = Medium
  - 2 = Hard

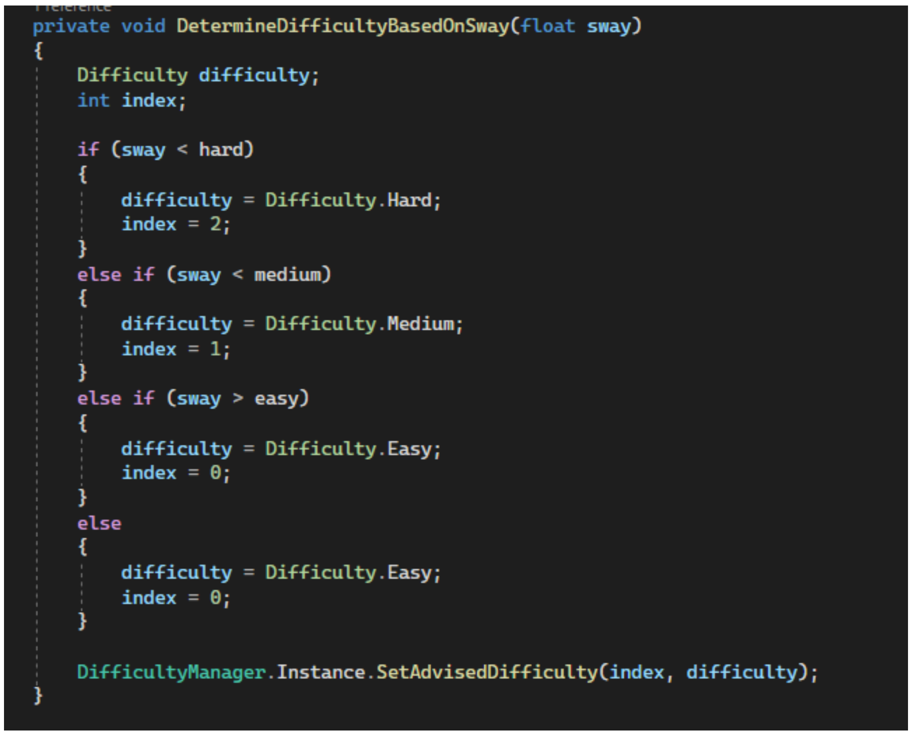
Usage Notes

- Call `SetAdvisedDifficulty(...)` when difficulty needs to be set based on prior performance.
- Use `SelectedDifficulty` in any system that needs to react to the current challenge level (e.g., wave spawner).



```
2 references
private void StartWave(int waveNumber) {
    // Determine how many fireflies to spawn this wave
    int spawnCount = baseFireflyCount + (waveNumber + 2);
    remainingFireflies = spawnCount;

    switch (DifficultyManager.Instance.SelectedDifficulty) {
        case Difficulty.Easy:
            break;

        case Difficulty.Medium:
            break;

        case Difficulty.Hard:
            break;

        default:
            Debug.Log(DifficultyManager.Instance.SelectedDifficulty);
            Debug.LogWarning("Not valid difficulty selected!");
            return;
    }
    // Tell the spawner to spawn the fireflies
    spawner.SpawnFireFly(spawnCount);
}
```

Wave spawner looks through the selected difficulties



```
1 reference
private void DetermineDifficultyBasedOnSway(float sway)
{
    Difficulty difficulty;
    int index;

    if (sway < hard)
    {
        difficulty = Difficulty.Hard;
        index = 2;
    }
    else if (sway < medium)
    {
        difficulty = Difficulty.Medium;
        index = 1;
    }
    else if (sway > easy)
    {
        difficulty = Difficulty.Easy;
        index = 0;
    }
    else
    {
        difficulty = Difficulty.Easy;
        index = 0;
    }

    DifficultyManager.Instance.SetAdvisedDifficulty(index, difficulty);
}
```

Set an advised difficulty based on your head sway