Document: Understanding the Marching Band Grid Position Exporter Script

Purpose of the Script

This script is designed to generate grid position data for a marching band performance in Unity, formatted into a CSV file. The script is tailored to help marching band teams, especially drill coordinators, visualize and map each performer's position on the field, with accurate descriptions based on position, side, and distance from the yard lines.

Script Overview

The script:

1. Utilizes Unity's editor window to create an export tool accessible from the Unity menu.
2. Processes position data from a grid component (`SnapToGridLines`) and exports detailed descriptions of each position to a CSV file.
3. Calculates the position descriptions based on the performer's proximity to specific yard lines, sideline boundaries, hashes, and end zones.

Script Breakdown

1. Setup and Initialization

Unity Editor Window Setup:

```
1. Define class `GridPositionExporter` that inherits from
`EditorWindow`.
2. Set up variables:
   - `snapToGrid`: holds a reference to the `SnapToGridLines`
component, which contains grid position data.
   - `exportPath`: the path where the CSV file will be saved.
   - `FIELD_LENGTH`, `FIELD_WIDTH`, and `STEP_SIZE`: constants
defining the field's dimensions and 8-to-5 step size.

3. Define `ShowWindow()` method:
   - Adds an entry in Unity’s menu under "Marching Band/Export Grid
Positions".
   - Opens the editor window titled "Grid Position Exporter".
```

Explanation: This section initializes the editor window and sets up basic configuration for the export, including where the output file will be saved and the field dimensions.

---

2. GUI Setup for Export Button

OnGUI() Method:

```
1. Display a label "Marching Band Grid Position Exporter".
2. Allow the user to assign the `SnapToGridLines` component to
`snapToGrid`.
3. Set an export path for the CSV file.
4. Add a button labeled "Export 8-to-5 Grid":
   - If `snapToGrid` is assigned, calls `ExportGridPositions()`.
   - If not assigned, shows a dialog error.
```

Explanation: The GUI section provides a user interface for selecting the grid component, choosing the export location, and starting the export. This is the interactive section of the tool where users initiate the process.

---

3. Exporting Grid Positions

ExportGridPositions() Method:

```
1. Check if the export directory exists:
   - If not, create it.
2. Call `ExportGridType()` with `xPositions8_5` and `zPositions8_5`
data from `snapToGrid`.
3. Refresh the AssetDatabase to ensure the file appears in Unity's
project view.
4. Display a dialog indicating the export was successful.
```

Explanation: This method controls the export process, ensuring the necessary directory exists and calling the appropriate function to process and save the data.

---

4. Exporting Individual Grid Data

ExportGridType() Method:

```
1. Define the CSV file path based on `exportPath`.
2. Initialize a StringBuilder to build CSV content.
3. Loop through each `x` and `z` position:
   - Generate a line of data for each grid intersection by calling
`GeneratePositionData(x, z)`.
   - Append each generated line to the CSV.
4. Write the CSV content to the specified file.
```

Explanation: This function generates each line of position data, effectively creating the CSV file by looping through `x` and `z` positions and appending the formatted descriptions.

---

5. Generating Position Data

GeneratePositionData() Method:

```
1. Determine the `yardLine` based on the `z` position using
`CalculateYardLine()`.
2. Determine the `side` (1 or 2) based on whether `z` is less than 60.
3. Generate descriptions for `x` and `z` using
`DetermineXPositionDescription(x)` and
`DetermineZPositionDescription(z, yardLine, side)`.
4. Combine the descriptions and side indicator into a single position
description string.
5. Return the complete CSV line with x, z, yardLine, side, and
position description.
```

Explanation: This function calculates the full description of each position, detailing proximity to yard lines, hashes, and sidelines, along with a mirrored description on each side of the field.

6. Calculating the Yard Line Based on `z`

CalculateYardLine() Method:

```
1. If `z` is approximately 60 (middle of the field), return 50 (center
yard line).
2. For Side 2 (z < 60), return the corresponding yard line for ranges
based on intervals from the 0-yard line.
3. For Side 1 (z > 60), return the mirrored yard line from the end
zone towards the center of the field.
4. Return -1 if no yard line match is found.
```

Explanation: This method determines the closest yard line based on the `z` position, taking into account each side of the field and ensuring symmetry.

7. Generating X-Axis Position Descriptions

DetermineXPositionDescription() Method:

```
1. Check `x` position against front sideline, back sideline, and other
reference points.
2. Based on the range:
   - Provide descriptive phrases like "On front sideline", "X steps in
front of front hash", etc.
3. Return the appropriate position description for `x`.
```

Explanation: This method describes the position on the field along the x-axis, identifying the performer's location relative to the sideline, front/back hashes, and within the 8-to-5 step grid system.

8. Generating Z-Axis Position Descriptions with Mirroring Logic

DetermineZPositionDescription() Method:

```
1. For end zones (z between 0-10 for Side 2 or 110-120 for Side 1):
   - Calculate steps outside the 0-yard line.
   - Return descriptions for steps outside or "On the 0 yard line" if
directly on it.

2. For field positions on other yard lines:
   - Use mirrored logic across the 50-yard line to reflect Side 2
positions onto Side 1.
   - Calculate `reflectedZ` as 120 - z for Side 1.
   - Identify the mirrored yard line using
`CalculateYardLine(reflectedZ)`.
   - Calculate steps inside or outside the mirrored yard line.

3. Return appropriate descriptions, adjusting for inside/outside based
on mirrored reflection.
```

Explanation: This function calculates the description along the z-axis, ensuring each position description on Side 1 mirrors Side 2 exactly. It adjusts for yard line references and inside/outside descriptions based on the mirrored position, which is reflected across the 50-yard line.

Key Considerations and Limitations

1. **Field Mirroring:** The script mirrors Side 2 onto Side 1 at the 50-yard line, ensuring descriptions are consistent on both sides of the field.
2. **Step Calculations:** Ensure that `STEP_SIZE` aligns with your marching band's step size (8-to-5 in this case) for accurate position descriptions.
3. **Yard Line Ranges:** Adjustments to the ranges in `CalculateYardLine()` might be necessary if field layout or grid intervals change.

Final Notes

This script is designed to output detailed marching positions for drill and performance alignment. Understanding the mirroring logic at the 50-yard line is essential, as it allows for accurate descriptions on both sides of the field, simplifying adjustments during practice.

With this guide, your team should have a solid foundation for understanding, adjusting, and utilizing the script for performance planning and drill design. Feel free to reach out for any clarifications or further customization.