Heatmap Shader Documentation

Overview

The "HeatmapShader" is an unlit shader designed for visualizing heatmap data in Unity. It takes input from a texture and dynamically updates the visualization based on an array of heatmap points with associated properties such as position, radius, and intensity.

Properties

- **_HeatTex (Texture2D)**: Texture property used for the heatmap visualization.
- **_QuadTransform (Vector)**: Property to specify quad transformation.

SubShader

- **Tags**: Specifies rendering order and transparency.
- **Blend**: Alpha blend mode for transparency.

Vertex Shader

- **Inputs**: Vertex position.
- **Outputs**: Transformed vertex position and world position.
- **Functionality**: Transforms vertex position to clip space and calculates world position of the vertex, adjusting it based on quad transformation.

Fragment Shader

- **Inputs**: Output from vertex shader.
- **Outputs**: Color of the fragment.
- **Functionality**: Calculates heatmap intensity by looping through all heatmap points, computing distance and influence (intensity) based on distance and radius. Accumulates intensity and converts it to a color using the heatmap texture.

Uniforms (Shader Variables)

- **_Points_Length (int)**: Length of the heatmap point arrays.
- **_Points (float4[100])**: Array storing heatmap point positions (x, y, z).
- **_Properties (float4[100])**: Array storing heatmap properties (x = radius, y = intensity).
- **_HeatTex (sampler2D)**: Heatmap texture sampler.

Fallback

- If the shader is not supported, it falls back to the "Diffuse" shader.

Heatmap Script Documentation

Overview

The "Heatmap" script is responsible for updating and managing heatmap data in Unity. It utilizes raycasting to detect hit points and updates the heatmap material accordingly.

Public Variables

- **count (int)**: Maximum number of points in the heatmap.
- **material (Material)**: Reference to the material used for rendering the heatmap.
- **positions (Vector4[])**: Array storing the positions of the heatmap points.

Private Variables

- **properties (Vector4[])**: Array storing properties (e.g., intensity) of the heatmap points.
- **currentIndex (int)**: Index to keep track of the current position in the circular buffer.

Start Method

- **Functionality**: Initializes arrays for positions and properties. Starts a coroutine to update positions periodically.

UpdatePositionsPeriodically Coroutine

- **Parameters**: Interval (float) - time interval between updates.
- **Functionality**: Periodically updates heatmap positions based on raycast hits. Calls **UpdatePositionsFromRaycast()** and **UpdateMaterial()**.

UpdatePositionsFromRaycast Method

- **Functionality**: Performs a raycast from the center of the screen. Transforms the hit position from world space to local space. Checks if the hit point matches any of the previously hit points. If it's a new hit point, finds the next available index in the arrays using a circular buffer approach.

UpdateMaterial Method

- **Functionality**: Updates the material properties to reflect changes in heatmap data. Sets the length of the point arrays in the shader using **material.SetInt("_Points_Length", count)**. Sets the arrays of positions and properties in the shader using **material.SetVectorArray("_Points", positions)** and **material.SetVectorArray("_Properties", properties)** respectively.

Additional Notes

- The script utilizes raycasting to detect hit points on objects in the scene.
- It employs a circular buffer approach to efficiently manage the storage of heatmap points.
- Each heatmap point consists of a position and associated properties such as radius and intensity.

- Intensity of a heatmap point is incremented when a hit point matches a previously hit point, indicating increased activity at that location.
- The script continuously updates the heatmap visualization by updating the material properties based on the latest heatmap data.

Example Usage

1. Attach the "Heatmap" script to a GameObject in the scene.
2. Assign a material with the "HeatmapShader" to the **material** variable in the script inspector.
3. Adjust the **count** variable to control the maximum number of heatmap points.
4. Run the scene and observe the heatmap visualization updating in real-time based on raycast hits.

Considerations

- Performance: Depending on the number of heatmap points and the frequency of updates, performance may be impacted. Optimize the script and shader as needed.
- Visualization: Experiment with different heatmap textures and shader properties to achieve desired visualization effects.
- Interaction: Extend the functionality to support user interaction, such as adjusting heatmap parameters or displaying additional information on mouse hover.