SeenMeshManager

Overview

The SeenMeshManager is a component that manages and visualizes triangles that have been scanned or "seen" during a point cloud scanning process. It tracks scanned triangles, updates a mesh representation of the seen areas, and provides functionality for toggling visibility and updating mesh colliders. This system is essential for ensuring that previously scanned areas are not redundantly processed, improving performance and accuracy in AR or simulation applications. Creating a completely new mesh using only the scanned point cloud data would be difficult to implement and require heavy calculations which would substantially slow down the simulation. Instead of creating a completely new mesh from the point cloud data, we use the existing meshes of the scene to create a mesh that will be updated with new triangles in real-time as they are hit by ray casts. This way we can easily build a new mesh using existing meshes and without any calculations. The performance of SeenMeshManager is optimized by updating colliders only once every second and caching large lists of vertices and triangles.

How It Works

1. Core Components

The SeenMeshManager consists of the following key components:

- Mesh (_seenMesh): A Unity Mesh object that stores the vertices and triangles of the seen areas.
- Triangle Tracker (_seenTriangles): A HashSet<int> that ensures each triangle is only processed once.
- Vertex and Index Lists (_seenVertices, _seenIndices): Lists that store the vertices and indices of the seen triangles.
- MeshCollider and MeshRenderer: Components for collision detection and visibility control.
- Dictionaries for the seen mesh triangles (_meshTrianglesCache) vertices (_meshVerticesCache): Caches to avoid repeated access to large lists

---

2. Initialization

The SeenMeshManager is initialized with the following steps:

- A new Mesh object is created with 32-bit indices enabled to support large meshes.
- Internal data structures (_seenTriangles, _seenVertices, _seenIndices) are initialized to track the scanned data.
- A new GameObject is created to hold the mesh, with MeshFilter, MeshCollider, and MeshRenderer components attached.
- The MeshFilter and MeshCollider are assigned the shared mesh (_seenMesh).

- The GameObject is tagged as "ARMesh" and assigned to the "ARMesh" layer for filtering purposes.

---

3. Adding Triangles

The AddTriangleToSeenList method processes triangles hit by a raycast:

1. **Triangle Validation:**
    - The triangle index from the RaycastHit is checked to ensure it is valid and not already processed.
    - Bounds are validated to prevent out-of-range errors.
2. **Vertex Transformation:**
    - The triangle's vertices are transformed from local to world space using the MeshFilter's transform.
3. **Data Storage:**
    - Unique vertices are added to the _seenVertices list.
    - Indices are updated to reference the newly added vertices.
4. **Mesh Update:**
    - The UpdateSeenMesh method is called to update the mesh with the new vertices and triangles.

---

4. Updating the Mesh

The UpdateSeenMesh method ensures the SeenMesh is up-to-date.

- Clears the existing mesh data.
- Assigns the updated vertices and triangles to the mesh.
- Recalculates normals for proper lighting and shading.
- Calls UpdateMeshCollider in intervals of 1 second to synchronize the MeshCollider with the updated mesh.

---

5. Visibility Control

The ToggleVisibility method allows toggling the visibility of the SeenMesh:

- Uses the MeshRenderer to enable or disable rendering of the mesh.
- Updates the mesh to ensure it is vizualized correctly

Code Flow Overview

Initialization

- The SeenMeshManager is instantiated and initialized with empty data structures.
- A GameObject is created to hold the mesh, and its components are configured.

Scanning and Adding Triangles

- When a raycast hits a triangle, the AddTriangleToSeenList method is called.
- The mesh's triangles and vertices are fetched from the cache if they have been cached, otherwise fetch them directly and cache them for later use
- The triangle's vertices are transformed, stored, and added to the mesh.

Mesh Updates

- The UpdateSeenMesh method is called whenever new triangles are added.
- The MeshCollider is updated once every second to reflect the changes.

Visibility Toggling

- The ToggleVisibility method is used to show or hide the SeenMesh.

Key Components

Mesh (_seenMesh)

- Stores the vertices and triangles of the seen areas.
- Updated dynamically as new triangles are added.

Triangle Tracker (_seenTriangles)

- Ensures that each triangle is only processed once.
- Prevents redundant calculations and improves performance.

MeshCollider and MeshRenderer

- MeshCollider: Synchronizes with the SeenMesh for collision detection.
- MeshRenderer: Controls the visibility of the SeenMesh.

Cache for mesh triangles and vertices

- _meshTrianglesCache: Stores triangle lists in a dictionary where the mesh is the key
- _meshTrianglesCache: Stores vertex lists in a dictionary where the mesh is the key

---

Error Handling

- Invalid Triangle Index: Logs a warning if the triangle index is out of bounds.
- Missing Components: Logs a warning if the renderer is missing when trying to set a material to the renderer and if the collider is missing when trying to update the collider.

Integration with PointCloudManager

The SeenMeshManager is tightly integrated with the PointCloudManager:

- The PointCloudManager uses the SeenMeshManager to track and highlight triangles hit during the scanning process.
- The AddTriangleToSeenList method is called whenever a valid triangle is detected by a raycast.
- The ToggleVisibility method is mapped to a key press (KeyCode.Y) to allow toggling the visibility of the seen mesh. (This will soon be delegated to a separate Input class)

Summary

The SeenMeshManager is a robust system for managing and visualizing scanned triangles in Unity. It efficiently tracks seen areas, updates a mesh representation, and provides tools for collision detection and visibility control. This makes it an essential component for AR applications, simulations, and real-time 3D scanning projects. Its modular design ensures flexibility and ease of integration into larger systems.