Appendix A

3D Audio-Visual Scene Reproduction

A.1 Component Hierarchy

Order	Component	Description
1	**Model Importer (Game Object)**	An object holding the scripts to perform scenery import.
1.1	Import Scenery (Script)	`ImportScenery.cs`: A script to import scenery meshes from a .obj file generated by the pipeline.
2	**Directional Light (Game Object)**	Provides lighting to the scenery. Placed at coordinates $(x : 0, y : 5, z : 0)$.
3	**Audio Source (Game Object)**	The Game Object defining the source of the audio.
3.1, 3.2	Sphere (Mesh); Mesh Renderer	Renders the object as a sphere so that the user can see the audio source during runtime; the Mesh Renderer changes the material during interaction (red by default, blue whilst being interacted with).
3.3	Sphere Collider (Component)	Gives the Audio Source a collider hitbox, allowing for interactivity through collision detection (i.e.: grabbing and moving the source).
3.4	Audio Source (Component)	Sets the audio file to play from the Audio Source (controlled by `ChangeAudio.cs` and `ImportSound.cs`); determines whether sound play behaviour is manual or automatic (controlled by `PlaySoundOnClick.cs` and `SoundClick.cs` scripts); determines attenuation settings (logarithmic, as previously described in the Acoustic Room Modelling section under "Steam Audio Source").
3.5	Steam Audio Source (Component)	Determines Steam Audio settings for producing the sound and its acoustics, as defined previously in the Acoustic Room Modelling section under "Steam Audio Source".

3.6	Rigidbody (Component)	Allows for the source to be physically moved throughout the environment.
3.7	XR Grab Interactable (Component)	Provided by OpenXR; Allows for the source to be physically moved throughout the environment through VR interaction events.
3.8	Change Audio; Play Sound On Click (Scripts)	Scripts, as defined in the description for the Audio Source Component, that allow for the audio source file to be changed and the audio to be played on button press, respectively.
4	**Steam Audio Static Mesh (Game Object)**	Game Object dynamically created by Steam Audio at runtime to handle its audio pipeline. Has a reference to the serialised mesh of all acoustic geometry of the scene, to which it is exported to.
5	**LocomotionArea (Game Object)**	A Game Object defining the area in which the user can locomote.
5.1	Quad Instance (Mesh)	A mesh defined by four corner vertices, allowing for simple resizing of its area on scene import. Located at $(x : 0, y : 0.163, z : 0)$, to account for the thickness of the floor when the scene mesh is imported at $(x : 0, y : 0, z : 0)$.
5.2	Resize Locomotion Area (Script)	`ResizeLocomotionArea.cs`: Calculates the combined bounds of all sub-meshes upon scene import from a new .obj file; repositions the four defining vertices of the Game Object's quad mesh; then re-renders the mesh.
5.3, 5.4	Teleportation Area; Continuous Move Provider (Component)	Components provided by OpenXR that allow for teleport locomotion (bounded to the Mesh's vertex coordinates) and continuous movement (upon the Mesh's surface).
6	**XR Origin (Game Object)**	Game Object asset provided by OpenXR, serves as the "origin" for the VR environment (tracking, interaction, etc.). Located at $(x : 0, y : 0.163, z : 0)$, to account for the thickness of the floor when the scene mesh is imported at $(x : 0, y : 0, z : 0)$.
6.0.1	XR Origin; Input Action Manager (Components)	Provided by OpenXR, manages all inputs sent from the VR environment.
6.0.2	Ray Hider (Script)	`RayHider.cs`: Enables the line visual for teleportation and interaction only during such events.
6.1	XR Interaction Manager (Game Object)	Provided by OpenXR, manages all interactions sent from the VR environment.
6.1.1	XR Interaction Manager, Input Action Manager (Components)	Allows for the binding and mapping of customisable interaction events (such as controller buttons for activating teleportation).

6.2	Camera Offset (Game Object)	Sets the offset from the origin for the camera (headset) and controller tracking; with the Tracking Origin Mode set to Floor, this allows the scene to be rendered at the height of the user registered by the headset.
6.2.1	Locomotion System (Game Object)	Provided with OpenXR's XR Origin Asset, this maps the locomotion system to the XR Origin.
6.2.1.1, 6.2.1.2	Teleportation Provider, Snap Turn Provider (Component)	Allows for the mapping of the Teleportation locomotion and Snap Turn to controller/action inputs.
6.2.2	Main Camera (Game Object)	The main camera from which the visual output is rendered.
6.2.2.1	Camera (Component)	Renders visual output to the target headset (the displays of both eyes).
6.2.2.2, 6.2.2.3	Audio Listener; Steam Audio Listener (Component)	Enables Steam Audio-calculated audio to be heard by the user based on camera position, as defined previously in the Acoustic Room Modelling section under "Steam Audio Listener".
6.2.2.4	Tracked Pose Driver (Input System)	Provided by OpenXR, changes the camera's orientation and position from the tracked inputs from the OpenXR Default Input Actions (corresponding to the headset).
6.2.2.5	Keyboard Move (Script)	`KeyboardMove.cs`: Disabled in the final build, this script allowed for moving the camera with keyboard and mouse input to evaluate audio spatialisation.
6.2.3	Left Controller (Game Object)	Game Object corresponding to the representation and functions of the Left Controller.
6.2.3.1, 6.2.3.2	Action-Based Controller Manager; XR Controller (Components)	Provided by OpenXR, allows mapping of all controller input actions to specified functions such as locomotion.
6.2.3.3, 6.2.3.4	XR Ray Interactor, Line Renderer (Components)	Provided by OpenXR and customised to a projectile curve, terminating in a 2D circular visual on the ground to allow for selecting a location to teleport to during Teleport Locomotion.
6.2.3.5, 6.2.3.6	XR Interactor Line Visual; Sorting Group (Components)	Further customisation for the ray interactor's line visual, with a high sorting layer to ensure no occlusion of visuals.
6.2.4	Right Controller (Game Object)	
6.2.4.1 - 6.2.4.6	[Ditto Left Controller]	Same as with left controller, though with differing mapped bindings for enabling function such as snap-turn with the right analogue stick.
7	**KitchenDemoScene (Game Object)**	Holds the mesh of a LIDAR scan of a Kitchen scene, provided by our supervisor, Dr. Hansung Kim, for use when running the demo scene (overlaid on top of pipeline output mesh).

7.1	Load Demo (Script)	`LoadDemo.cs`: Loads a specific pipeline-generated .obj file corresponding to the kitchen scene from the project's Resources folder, and enables the LIDAR mesh overlay.
8	**ModelWrapper (Game Object)**	Wrapper Game Object to hold the meshes obtained from any input .obj file.
8.0.1	Assign Materials (Script)	`AssignMaterials.cs`: Iterates through all sub-meshes of the contained Game Object corresponding to the .obj file input, assigning Steam Audio Geometry accordingly, then calls Steam Audio's Export function the current scene mesh.
8.1	[.obj filename] (Game Object)	Game Object holding all sub-meshes, split by material, of the pipeline output.
8.1.n	[.obj material name] (Game Object)	Sub-mesh extracted from .obj file; a contiguous mesh of a certain material.
8.1.n.1	Mesh Renderer (Component)	Renders the mesh within the scene, with the material colour for each mesh corresponding to each material recognition output.
8.1.n.2	Steam Audio Geometry (Component)	As defined previously in the Acoustic Room Modelling section under "Steam Audio Geometry", sets the Steam Audio Material for the sub-mesh with the corresponding parameters.

Table A.1: Breakdown of Unity scene's hierarchical components.