Research Project

Doorduck Player Position Tracking and Spatial Audio – An Interactive Experience

Author: Sriraj Rajkumar

Co-Authors: Tejaswi Gowda, Associate Professor, ASU.

Members: Nickolas Pilarski, Director of RVCo Lab, and Associate Professor, ASU; Sven Ortel, Professor of Practice, ASU; Danyal Khorami, Ph.D Student, ASU.

Integration Workflow

Step 1: Player Tracking

- Use a tracking system (e.g., LillyGo UWBs Web Socket Trackers) to capture the player's position and rotation data in real time.
- Stream this data to the game engine via Web Sockets (e.g., Web Serial Communication).
- Collaboration with Prof. Tejaswi Gowda to develop, test and trial tracking data over web sockets.

Step 2: Dynamic LED Volume Rendering

- In the game engine, create a virtual environment that transports the audience in the physical space of the LED volume.
- Use the player's position and rotation coordinates to adjust the spatial audio perspective in real time, ensuring the visuals on the LED wall align with the player's viewpoint.
- Implement parallax effects and depth cues to enhance immersion as the player moves. (Desired to achieve)

Step 3: Spatial Audio Rendering

- Map the virtual environment's audio sources to the player's position and orientation.
- Use an unreal audio engine to dynamically adjust sound cues, space, shape, reflections, and attenuation effects based on the player's movement.
- Sync the audio system with the LED volume's visuals to ensure audio-visual coherence.

4. Tools and Technologies

- Game Engine: Unreal Engine (with nDisplay for LED volumes).
- Spatial Audio: Binaural Audio, and Ambisonic Recordings.
- Tracking System: Web Socket - Web Serial Communication with UWBs trackers.
- Middleware: OSC, NDI, or Syphon for synchronisation.

5. Benefits

- Immersive Experience: Players feel completely immersed in the virtual environment.
- Scalability: The system can be adapted for VR, AR, or mixed-reality applications. Once standardised, could use it as a medium for creative directors or storytellers an immersive platform to present their stories in a new way.
- Interactivity: Real-time tracking enables dynamic and responsive environments. This enables each person to have a unique experience within a shared experience. Each player will have their own personal experiences, based on where you stand in the space (in real life) your experience differs because you have position-based audio localisation.

This approach pushes the boundaries of virtual production by tightly integrating visuals, audio, and player movement into a cohesive, immersive experience.

WebSocket —

Purchased UE5 blueprint for web socket. *Websocket created by WanWanHa*

Link: UnrealEnginePluginsDemo/WebSocketDemo/WebSocketClientDemo/README.md at master · Tangha-Technologies/UnrealEnginePluginsDemo

Importing Web socket into UE5

Goto Epic games launcher > my library > WebSockets plugin (install this for all the compatible UE versions)

Open Unreal Engine 5.4 (version I have used)

Goto Edit > Plugins, and search for websockets. Enable the plugin 'WebSockets' and restart Unreal Engine 5.

Creating Unreal Engine 5 Client - Server connection

Blueprint data

Open the Unreal engine 5 project file you desire to create the websocket Client. Once you have the websocket plugin installed and restarted UE5.

STEP 1: Create a server. (If you have a server already skip this step).

We created an [AWS Server](AWS server), Prof. Gowda already had them purchased.

(write about creating a server)

STEP 2: Create Websocket:

Open Level Blueprint

Open Level blueprint. Under ‘variables’ tab on the left side. Click on the ‘+’ icon to add a variable. Label it as ‘websocket’ and assign the custom single object reference. This should create the required websocket variable. Drag and drop it in the level blueprint in event graph editor.

Create Variables websocket, Headers, and Protocols.

Creating Custom Variables.

Now, Right-click and create a node ‘Create Custom Web Socket’.

It requires two new variables “Headers” and “Protocols”. To create I followed the same steps as how I have created the web sockets variables. Although both are strings, we need to assign Headers as maps of strings and Protocols as array of strings.

You can set token parameter here. Add a token and fill the token value.

Headers

Variable Type - String - Map

Protocols

Variable Type - String - Array

Connect them to their respective nodes and you can use the parameter as your web server required.

Drag a link from ‘Return Value’ and search for ‘Websocket set’ in the directory.

Bind event for each delegate you care about. The connected event will be called when connected to the server.

Similarly, create for connection error event, this connected event will be called for server connection failed. When connection received from the server,

Add a print string with an empty message input. This will print the received message from the server on the play window. Then, message event sent will be called when message send to server successfully.

STEP 3: Connect to Server:

Use Connect to Server function to connect to the server.

Create a Key 2 Pressed node and link to Connect to Server (Target is Web Socket Client). Link the target to Web Socket and create a Get Server URL node. Return this value to Print String and the execution node from Connect to Server to Print String.

STEP 4: Send Message to Server:

Use Send message to Server to send messages.

Create Key 3 Pressed node and link it to Send Message to Server (Target is Web Socket Client). Link the target to Web Socket and enter 'Send Message!' in the message field. Link the execute node to Print String.

STEP 5: Compile and save.

Transform, Location, and Rotation - Player Start

Place the player start in the scene to set the transform, location, and rotation at 0,0,0.

How to create variables for Transform, Location, and Rotation?

Screenshot of testing position and rotation data into Unreal Engine 5.

Parse and Map IMU Data to the Mannequin (Manny) in Unreal Engine 5.4

Parse IMU data and map it to Manny's follow camera using Blueprints in Unreal Engine 5.4.

I received IMU data (Quaternion or Euler angles) from LILYGO Mini D1 Plus via web socket into Unreal Engine and map that data to the Manny mannequin's follow camera.

Using getfield node to snoop only the specific tracker information to parse JSON.

Separating yaw, pitch, and roll value to dedicatedly combine and send into rotation node.

Link to the target component 'Follow Camera' and its rotation.

Web Serial Port Communication —

UWB trackers - Ultra Wideband is a short range, wireless communication technology that uses broad spectrum of frequencies for precise ranging and location tracking, offering high accuracy and low power consumption.

UWB - Dongle (1)

UWB - Active Anchors are charging

UWB - Anchors (2)

UWB - Trackers (2)

Map for Dongle + Anchors + Trackers installation in SS3

Tracker = UE5 Run test

(document UE5 run test)

Testing trackers with Prof. Tejaswi Gowda and his team.

DW1000 UWBs EP32 board

Full stacked tracker with position and rotation micro chip and a battery circuit installed with a on/off button.

ON/OFF button for the tracker.

Thanks to Julian, he helped me create a tracker mount on the headphone.

Dedicated trackers for the headphones 1 & 2.

Mounted trackers on the headphones.

Installing anchors for the motion capture in Sound Stage 3.

Testing and Calibrating trackers.

Trackers sending position and rotation data.

-

Bibliography

Credits and Citations —

source: https://www.fab.com/listings/a2c43038-5711-442c-9a98-9cd2ade9f89f by WanWanHa a.k.a TangHa, unreal engine fab market, Github, https://github.com/Tangha-Technologies/UnrealEnginePluginsDemo