

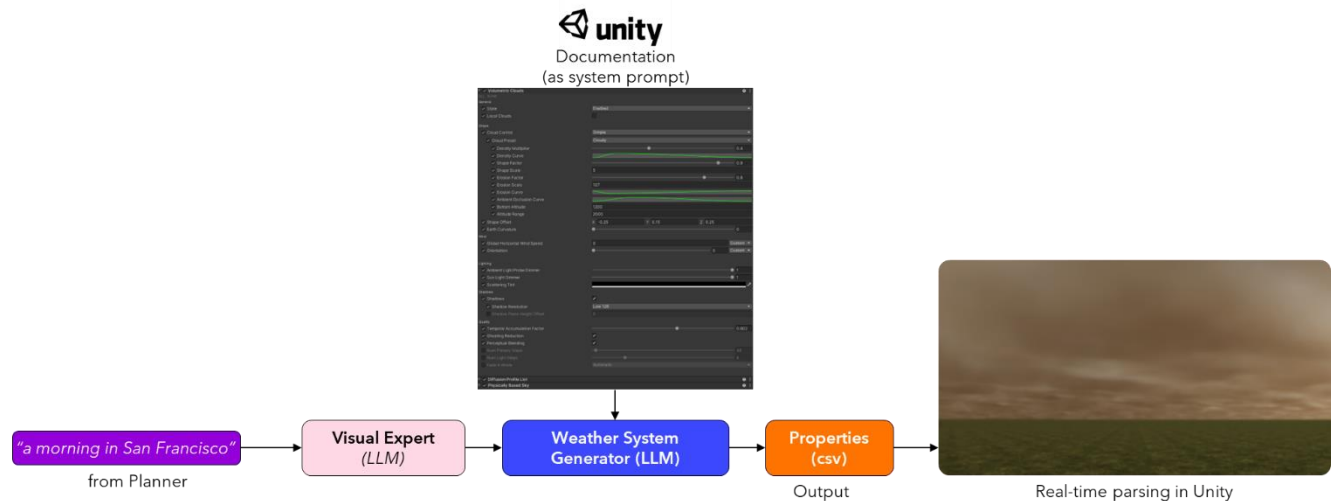
GIVE: A Multi-Agent Framework for Generating Immersive Multi-Modal Virtual Environments for 3D Games

Supplementary Materials

1. Details of Expert Team Agents:

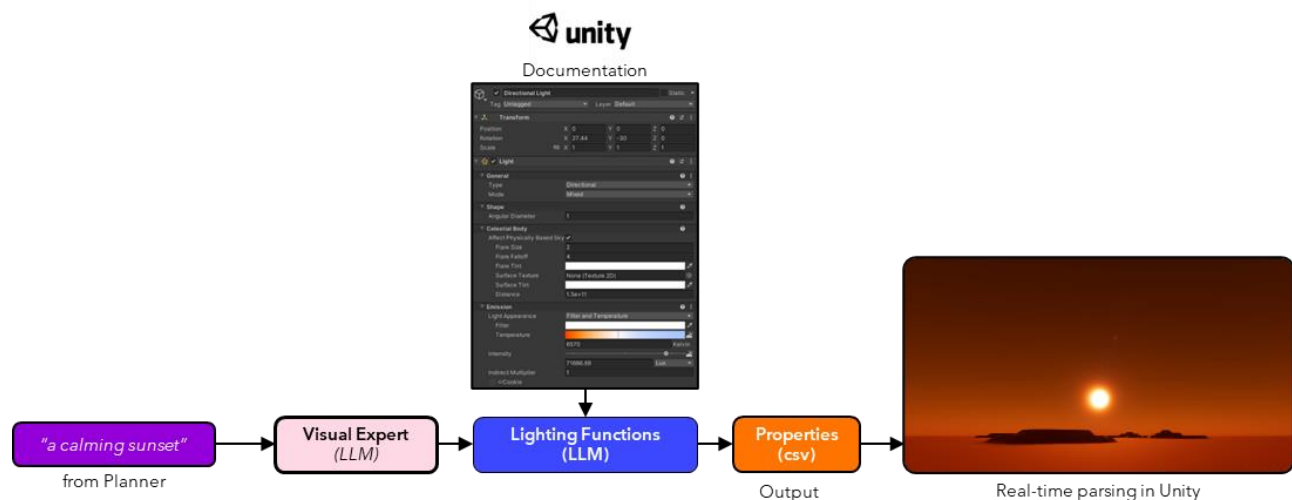
a. Details of the Visual Expert Team:

The team includes the following agents:



Working of the weather system generator agent (blue box).

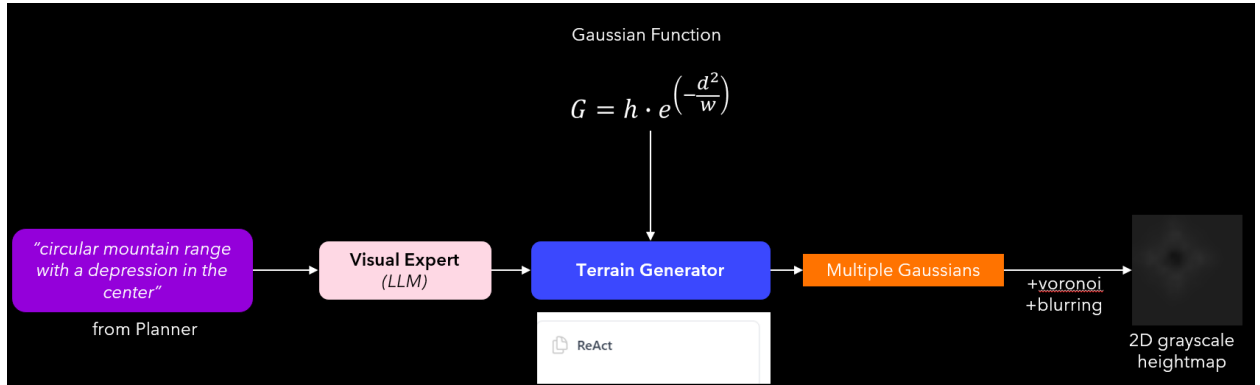
- Weather System Generator: An LLM-powered agent that manipulates global volume parameters in the Unity Game Engine (specifically the high-definition rendering pipeline) to create weather conditions matching the scene description. It adjusts properties like cloud appearance, wind speed, fog conditions, and precipitation.



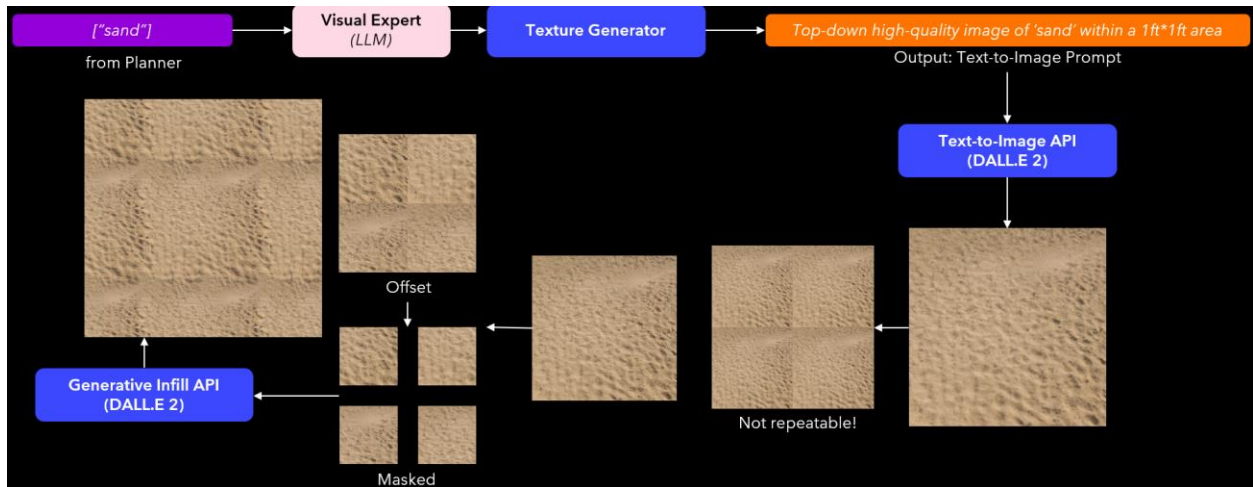
Working of the weather system generator agent (blue box).

- Lighting Function: Similar to the Weather System Generator, this LLM-powered agent manipulates existing lighting assets to match user requirements. It converts high-level

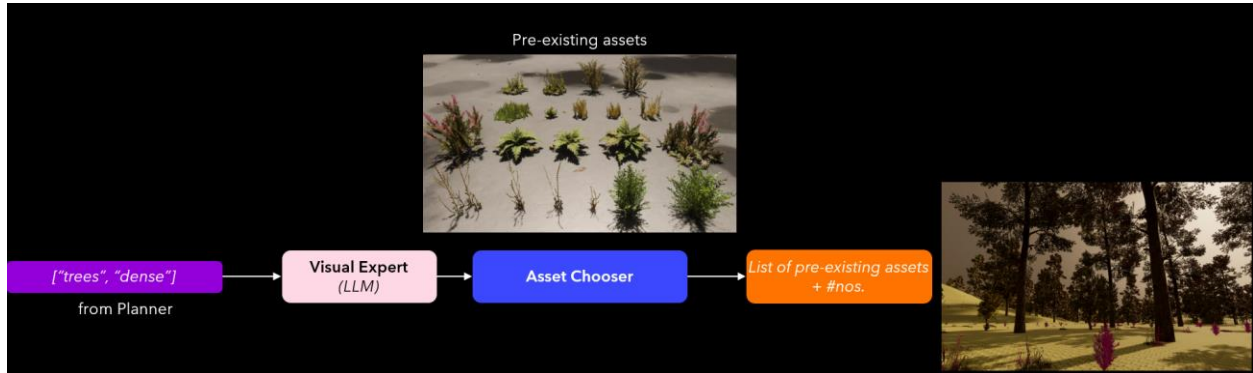
semantic statements about lighting conditions into specific property values, such as light position and intensity during different times of the day or mythical lighting conditions.



- **Terrain Generator:** Inspired by Zhang et al.'s work [1], this agent uses an LLM to generate Gaussian-Voronoi maps of terrains, which are converted to heightmaps. Terrain features like mountains and valleys are represented through overlapping Gaussians, which are then used to paint Voronoi cells, adding noise to the terrain. The output is smoothed using Gaussian blur and converted into a heightmap, which the game engine uses to alter terrain height.



- **Texture Generator:** This agent generates prompts for a text-to-image model to create texture images based on the required scene textures. The agent then calls the text-to-image model and the generated texture images are then used as texture maps in the game engine. A number of post-processing steps are completed in order to make the generated image tile-able. This includes offsetting the generated image by 50% in the horizontal and vertical directions, creating a mask to hide the seams in the middle, and then passing the masked image to an image model to generatively infill the sections. This post-processing step creates repeatable texture maps.

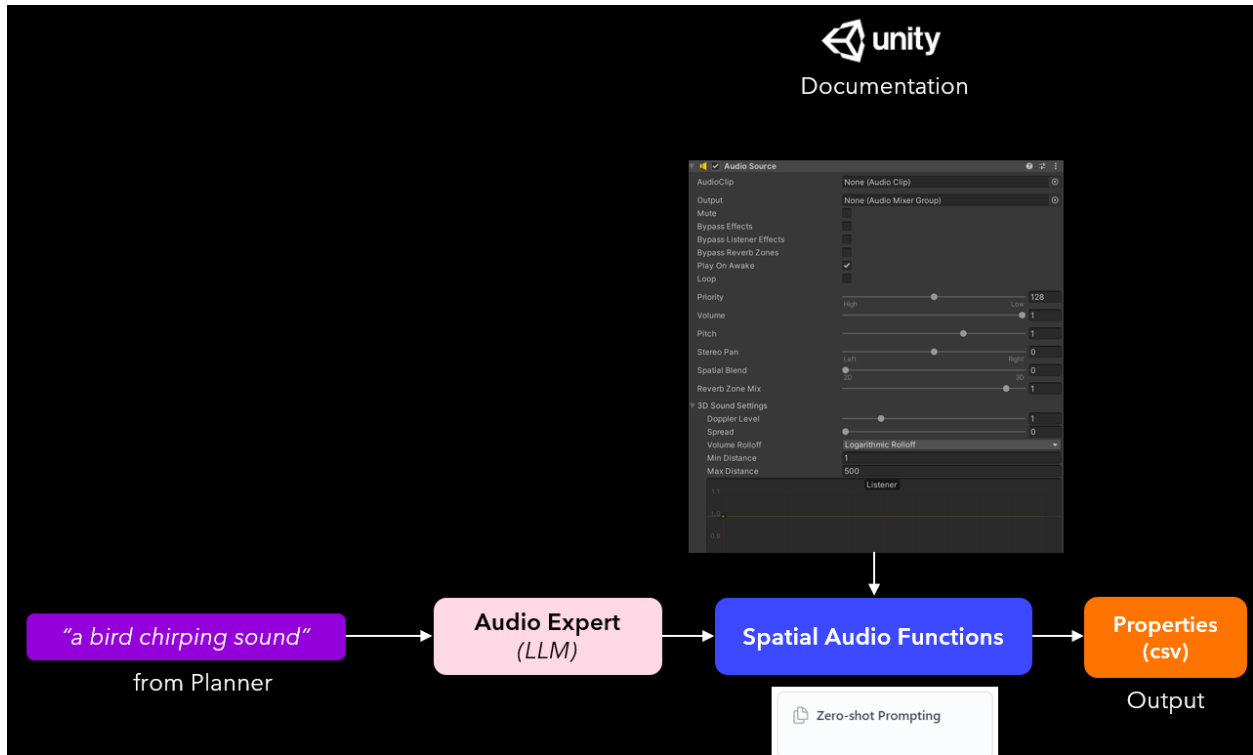


- Asset Choosing Agent: Given a list of pre-existing assets available with the user (e.g., trees, bushes, rocks, etc.), this agent decides what assets match the scene description and should be added to the virtual environment. Additionally, it decides on the density of the selected asset to be instantiated in the game engine (for e.g., a forest might have dense trees)

b. Details of the Audio Expert Team:

The team comprises three main agents:

- Audio Generator: This tool processes a list of audio descriptions by converting them into textual prompts for a text-to-audio model. Unlike an agent with an integrated language model, this tool solely focuses on calling the text-to-audio model to generate corresponding audio files.



- *Spatial Audio Agent*: This is an LLM-powered agent designed to adjust the audio properties of generated audio source files within the Unity game engine. Similar to the weather system generator and lighting function agent, this tool operates based on a system prompt that specifies the properties it can modify, the definitions of these properties, and their respective value ranges. It generates values for various audio properties for each audio source file, including parameters such as the audible range, spatial blend (3D/2D audio), Doppler effect level, and more.
- *Audio Placement Agent*: While the placement of objects is typically managed by the spatial expert, determining the optimal placement of audio elements is best handled by the audio expert. This LLM-powered agent collaborates with the spatial expert tool to understand the scene, specifically the locations of various objects in relation to the camera. Using this information, it decides whether an audio source should be attached to multiple game objects (e.g., bird chirping sounds on multiple trees), placed near specific objects (e.g., insect sounds above the ground and near rocks), or positioned freely with respect to the camera (e.g., wind noises above the user). The agent generates a list detailing the number and locations of audio sources to be instantiated. This information is then passed on to the spatial expert to place the audio elements accordingly.

c. Details of the Spatial Expert Team:

The spatial expert team aims to leverage 3D environments provided by game engines and has two main roles: analyzing and understanding the virtual scene as it is manipulated, and

collaborating with other expert teams to place assets semantically in the 3D environment. Currently, the spatial expert team uses two main tools:

- *Scene Analysis & Understanding Tool*: Inspired by the LLMR paper [2], this tool is implemented directly in the game engine. It searches the scene for objects with mesh renderers (visible 3D objects) and returns a detailed document containing the object's name, distance from the main camera (the character's 'eyes'), x, y, z coordinates, and bounding box dimensions. This document is shared with relevant agents in other expert teams.
- *Object Placement in 3D Tool*: Also implemented in the Unity game engine, this tool parses outputs from other agents/tools and runs one of three functions: assigning as a child of existing game objects with local x, y, z offsets; placing near a target object with global x, y, z offsets; or placing freely with global x, y, z offsets relative to the main camera. This tool is called whenever other tools need to place objects in 3D.

2. Implementation Details

In this section, we will highlight the implementation of our complete pipeline, which consists of two main components: the game engine workflow and the multi-agent framework.

For our game engine, we selected Unity because of its popularity among novice game developers and its beginner-friendly interface. We specifically opted for the high-definition rendering pipeline (HDRP) in Unity, as it allows for more immersive game assets, such as global volume assets (enabling clouds, fog, etc.). Additionally, HDRP provides more complexity, which poses challenges for code generation workflows due to the more complex documentation and the lack of extensive publicly available examples.

Here's the snapshot of the tools and libraries used in our implementation:

Game Engine-specific:

- Unity Game Engine – v2022.3.30f1
- High-Definition Rendering Pipeline (HDRP) – 14.0.11
- Game Assets (e.g., trees, bushes, etc.) – Unity Sample Terrain Scene ([Link](#))
- Scripting Language – C#

Multi-Agent Framework-specific:

- Scripting Language – Python 3.11.9
- Multi-Agent Libraries
 - LangGraph – 0.0.66
 - CrewAI – 0.30.11
 - CrewAI Tools – 0.2.6
- Large Vision and Language Models (through OpenAI API)
 - GPT 4.1
 - o4-mini
- Text-to-Image Models (through OpenAI API)
 - Dall.E 2

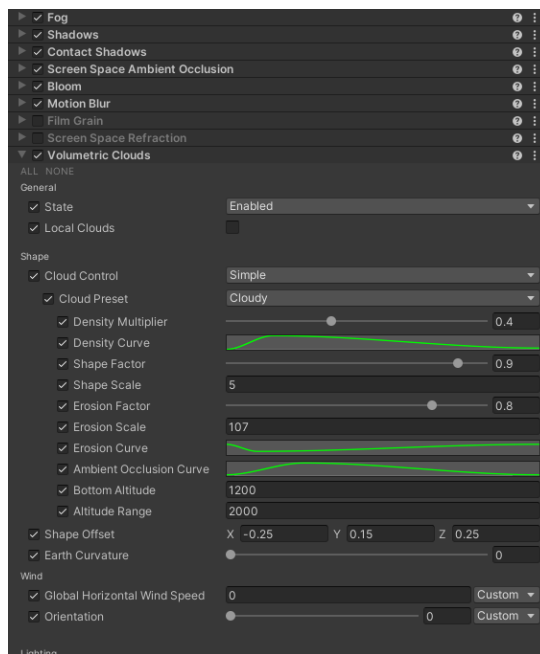
- Text-to-Audio (through ElevenLabs API)
 - Sound Generation – v1

Below, we will first define the game engine component of the workflow, followed by the multi-agent framework.

a. Game Engine Workflow in Unity –

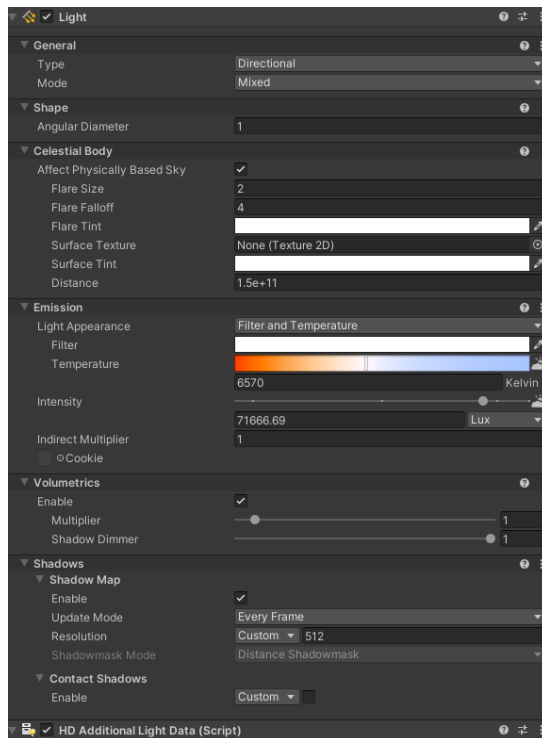
Given our goal of creating immersive virtual environments, we picked specific components of the game engine scene that would contribute most towards making the scene more immersive. As a result, we selected the following components:

- *Global Volume:* The global volume component can be used to apply post-processing effects or other scene-wide settings uniformly. Some of these settings include volumetric clouds, fog, physically-based skies, etc. ([Link](#))



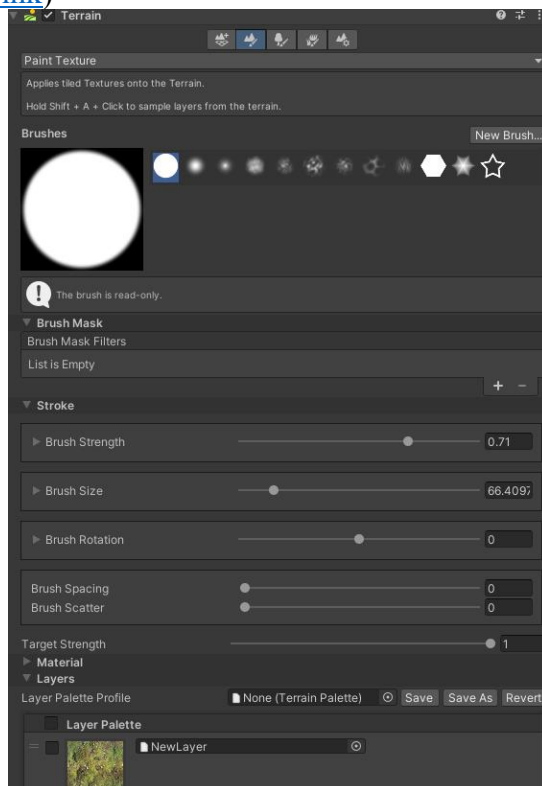
Example of the settings/properties associated with Global Volume.

- *Lighting:* The lighting component in Unity is essential for illuminating scenes and creating realistic or stylized visual effects. It includes various types of lights such as Point Lights, which emit light in all directions from a single point, Spot Lights, which emit light in a cone shape, and Directional Lights, which simulate sunlight by emitting light in a single direction. ([Link](#))



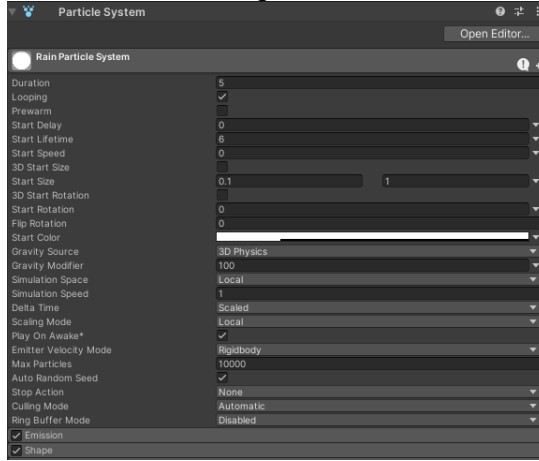
Example of the settings/properties associated with Lighting.

- *Terrain*: The Terrain Component is used to create expansive, detailed landscapes. It enables the sculpting and painting of terrain features such as mountains, valleys, and plains. Additionally, it allows us to add components like textures, trees, grass, etc. to the scene. ([Link](#))



Example of the settings/properties associated with Terrain.

- *Particle Systems:* In Unity, the Particle System Component is used to simulate and render fluid entities like smoke, fire, rain, and explosions by generating and animating large numbers of small 2D images or meshes called particles. ([Link](#))
 - o **Note:** In our work, we create Particle System prefabs for rain and snow, and the multi-agent framework simply decides to enable or disable the prefab based on the scene description.



Example of the settings/properties associated with the Particle System.

Using these components, we create a template scene in Unity. As these components can be altered and updated through the scripting API, we create C# scripts for each of the component, which consist of methods that: search and read the relevant '.csv' outputs of the multi-agent framework; and update the properties of the component every few hundred frames. These scripts are then attached to a 'Scene Manager' game object, from where they can easily be activated/deactivated if necessary. In addition to these scripts, we also have methods that capture the scene through different cameras every few hundred frames. The captured scenes are saved as images (.png) and are inputs to the Reflective Partner (R) when required.

To make the scenes more realistic, we also have the multi-agent framework decide what assets need to be added to the scene, for instance different types of trees, bushes, etc. To make this possible, we collect HDRP assets created by Unity for the demo scenes and store them in the folder. Depending on the multi-agent output, we create a script that searches for the relevant assets and randomly instantiates n copies of them in the scene on non-steep surfaces of the terrain.

The above components and methods form most of our Game Engine workflow. In the section below we will discuss the implementation of the multi-agent framework in detail.

b. Multi-Agent Framework:

While the agentic framework could be built from scratch using sequential calls to different instances of LLMs, recent open source developments have introduced powerful libraries to create intricate networks of agents with customizable communication channels. In our work, we specifically use two such libraries: LangGraph and CrewAI.

- LangGraph uses the concept of graph networks to define multi-agent workflows. The agents are initialized as nodes and communication between different agents is initialized as edges. It is a low-level framework providing fine-grained control over the flow of information, state of the application and cyclic execution of tasks. ([Link](#))
- CrewAI, on the other hand, is a library that offers high-level framework for building agentic workflows. It focuses on role-playing as its central premise and allows easy creation of agents and tasks using this concept. It also provides options to allow agents to delegate tasks to one another and create hierarchical workflows. ([Link](#))

In this work, we utilize both libraries for their respective benefits. LangGraph is used to build the overall framework, as it allows greater control over the flow of information and ease of integrating human-in-the-loop. CrewAI is used to build the expert teams, where once they receive tasks, the crews, with the respective roles, can potentially communicate with one another and complete tasks.

The Manager, Planner, and Reflective partner serve as individual agent nodes in the LangGraph framework, the Visual Expert node is a CrewAI crew, the Audio Expert is a set of LLM-powered tools and the Spatial Expert is a set of Unity Methods interacting with the multi-agent framework through simple file reading/writing. Below, I detail the implementation of each of these nodes, along with their system prompts, type of LLMs, and the tools they have access to.

1. **Manager (M):** The manager is the first node in the LangGraph framework, where it behaves as a chat-bot and directly interacts with the user in a turn-based interaction. The details of the manager are given below:
 - *LLM Model:* GPT 4.1 (temperature – 0.3) [low temperature to avoid deviating from the topic of conversation]
 - *Maintains Memory:* True
 - *Flow of Information:*
 - i. User (*U*) → Manager (*M*)
 - ii. Manager (*M*) → Planner (*P*)
 - *System Prompt:*

You are an expert interviewer hired by a game development company. Your task is to extract detailed information from a user about the type of scene they want to create and how they envision making it immersive. Ensure that the conversation remains concise, ideally around 6 to 7 messages in total. If any information is unclear, ask for clarification. If you still can't discern their needs, make reasonable predictions based on the conversation. After gathering the information, you need to create a detailed and creative description of the immersive scene that covers the weather, lighting, terrain, types of vegetation, audio elements and their general placements in the scene. Ask the user if the scene description is suitable and then move on to the next node.

Interview Objectives:

Immersive Visual Elements:

Ask the user what visual elements are most important to them.

Probe for details about colors, lighting, textures, and visual effects.

Audio Elements:

Ask the user about the necessary audio elements.

Inquire about background music, sound effects, voice-overs, and ambient sounds.

Location, Setting, and Environment:

Ask the user about the desired location, setting, and environment.

Clarify whether they want a realistic or fantastical setting, and gather details about specific features or atmospheres they envision.

Example Interview Flow:

Introduction:

"Hello! I'm here to help you design an immersive scene for your game. Could you start by telling me what kind of visual elements are most important to you?"

Follow-up on Visual Elements:

"Great! Can you describe the colors, lighting, and textures you envision for this scene?"

Immersive Audio Elements:

"Now, let's talk about audio. What kind of background music or sound effects do you think will make this scene more immersive?"

Clarification on Audio Details:

"Can you provide more details about any specific ambient sounds or voice-overs you want to include?"

Setting and Environment:

"Let's discuss the setting and environment. Where do you want this scene to take place? Is it a realistic location or something more fantastical?"

Clarification on Environment:

"Can you describe any specific features or atmospheres you want in this environment?"

Final Clarifications and Predictions:

"Is there anything else you want to add or clarify about your scene? If I understand correctly, you want [summarize key points]. Is that right?"

Note: Adapt your questions based on the user's responses to ensure all necessary details are captured effectively.

2. Planner (P): The planner is the intermediary node between the manager (M), the experts (V_E, A_E, S_E) and the reflective partner (R). For the planner, we also utilize the [function calling](#) option enabled by the OpenAI API. Function calling, similar to tools, allows the LLM to output structured data, call external APIs and so on. In the case of the Planner, we specifically use function calling to get a structured list of tasks for the expert teams. The details of the planner are given below:

- LLM Model: GPT 4.1 (temperature – 0.8) [higher temperature to make logical predictions in the event the user's requirements are vague]
- Maintains Memory: True
- Flow of information:
 - i. Manager (M) \rightarrow Planner (P)
 - ii. Planner (P) \rightarrow Visual Expert (V_E)
 - iii. Planner (P) \rightarrow Audio Expert (A_E)
 - iv. Planner(P) \leftrightarrow Reflect Partner (R)
- System Prompt:

You are an expert planner.

You have to lay out specific plans for expert agents to complete from the description of the scenario. The specific plan for these agents should entail a detailed description of what their tasks should be.

Here's the description: {detailed_description}.

- Function Specifications:

```

detailed_weather: str = "Detailed description of the weather conditions (e.g., 'sunny with scattered clouds'
detailed_lighting: str = "Detailed description of the lighting conditions (e.g., 'bright sunlight with the sun at noon')
detailed_geographic_terrain: str "Detailed description of the geographic terrain (e.g., 'rocky mountains with a steep incline')
terrain_texture: List[str]= "List of textures used for the terrain ground (e.g., ['rock', 'sand', 'mud', 'grass'])
flora_and_details: List[str] = "List of flora and details present in the scene (e.g., ['pine trees', 'bushes', 'twigs', 'small rocks', 'large rocks']).
flora_density: List[str] = "List of density values corresponding to the flora and details (e.g., ['dense', 'dense', 'few', 'medium', 'sparse'])
detailed_audio_sounds_description: List[str]= "List of very detailed descriptions of audio sounds used in the scene (e.g., ['woodpecker pecking on the tree', 'small leaves rustling in the wind'])

```

3. Visual Expert (V_E) Crew: For the visual expert team, we use the CrewAI library to create a crew of expert agents. The expert agents include weather systems generator, lighting functions, terrain generator, texture generator, and asset adding expert. Each of the agents and the CrewAI specifications are mentioned below:

3.1 Weather Systems Generator: This agent specializes in generating values to manipulate the Global Volume properties in the game engine.

- LLM Model: GPT-4.1 (temperature = 0.8)
- Maintains Memory: False
- Flow of information:
 - i. Planner (P) \rightarrow Visual Expert Agent ($v_{weather}$)
- System Prompt:

You need to predict values that will satisfy the users requirement. Assume that the user will not provide any additional information.

The values you can change are as follows:

```

densityMultipliervalue - The global density of the volumetric clouds. (range 0 and 1)
shapeScaleValue - Controls the size of the noise HDRP uses in the shaping stage to generate the general cloud shapes. (range 0.1 to 30)
shapeFactorvalue- Controls the amount of shaping to apply to the cloud volume. A higher value produces less cloud coverage and smaller clouds (range 0 to 1)
erosionFactorValue- Controls the amount of erosion to apply on the edge of the clouds. A higher value erodes clouds more significantly. (range 0 to 1)
erosionScalevalue- Controls the size of the noise HDRP uses in the erosion stage to add local details to the cloud edges. (range 0.1 to 30)
earthCurvatureValue- The curvature of the cloud volume. This defines the distance at which the clouds intersect with the horizon. (range 0 to 1)
bottomAltitudeValue- Controls the altitude of the bottom of the volumetric clouds volume in meters. (range 1000 to 4000)
altitudeRangeValue- Controls the size of the volumetric clouds volume in meters (range 1000 to 4000)
windSpeedvalue- Sets the global horizontal wind speed in kilometers per hour. (range 4 to 200)
windOrientationValue- Controls the orientation of the wind relative to the world-space direction x-axis. (range 0 to 360)
ambientLightProbeDimmerValue- Controls the influence of light probes on the cloud volume. A lower value suppresses the ambient light and produces darker clouds overall. (range 0 to 1)

```

sunLightDimmerValue- Controls the influence of the sun light on the cloud volume. A lower value suppresses the sun light and produces darker clouds overall (range 0 to 1)
 scatteringTintValueColorR- The color to tint the clouds. Red component out of 255
 scatteringTintValueColorG- The color to tint the clouds. Green component out of 255
 scatteringTintValueColorB- The color to tint the clouds. Blue component out of 255
 densityCurveKeyFrameX1 = Controls the density (Y axis) of the volumetric clouds as a function of the height (X Axis) inside the cloud volume. This value specifically corresponds to the X value of the first point in the curve. (range 0 to 1)
 densityCurveKeyFrameY1 = Controls the density (Y axis) of the volumetric clouds as a function of the height (X Axis) inside the cloud volume. This value specifically corresponds to the Y value of the first point in the curve. (range 0 to 2)
 densityCurveKeyFrameX2 = Controls the density (Y axis) of the volumetric clouds as a function of the height (X Axis) inside the cloud volume. This value specifically corresponds to the X value of the second point in the curve. (range 1 to 3)
 densityCurveKeyFrameY2 = Controls the density (Y axis) of the volumetric clouds as a function of the height (X Axis) inside the cloud volume. This value specifically corresponds to the Y value of the second point in the curve. (range 0 to 2)
 densityCurveKeyFrameX3 = Controls the density (Y axis) of the volumetric clouds as a function of the height (X Axis) inside the cloud volume. This value specifically corresponds to the X value of the third point in the curve. (range 3 to 5)
 densityCurveKeyFrameY3 = Controls the density (Y axis) of the volumetric clouds as a function of the height (X Axis) inside the cloud volume. This value specifically corresponds to the Y value of the third point in the curve. (range 0 to 2)
 inputProvidedbyAgent - this is the input that was provided to you

- **Expected Output:** csv file containing relevant properties.

3.2 Lighting Function: This agent specializes in generating values to manipulate the Lighting properties in the game engine.

- **LLM Model:** GPT-4.1 (temperature = 0.8)
- **Maintains Memory:** False
- **Flow of information:**
 - i. Planner (P) \rightarrow Visual Expert Agent (v_{light})
- **System Prompt:**

You need to predict values that will satisfy the users requirement. Assume that the user will not provide any additional information.

The values you can change are as follows:

angDia = Angular diameter of the emissive celestial body represented by the light as seen from the camera (in degrees). Used to render the sun/moon disk. (range 1.0 to 4.0);
 flareSize = Size the flare around the celestial body (in degrees). (range 0 to 10.0);
 flareFalloff = The falloff rate of flare intensity as the angle from the light increases. (range from 0 to 90.0);
 flareTintColorR = Tints the flare of the celestial body. Red component out of 255;
 flareTintColorG = Tints the flare of the celestial body. Green component out of 255;
 flareTintColorB = Tints the flare of the celestial body. Blue component out of 255;
 lightIntensity = the intensity of the light in lux. (range is from 70 to 13000);
 emissionColorR = color of the sun light. Red component out of 255;
 emissionColorG = color of the sun light. Green component out of 255;
 emissionColorB = color of the sun light. Blue component out of 255;
 colorTemp = Temperature of the sun light in kelvin (range is from 1500 to 20000);

multiplier = the light dimmer / multiplier on volumetric effects, between 0 and 16. ;
shadowDimmer = the volumetric shadow dimmer value, between 0 and 1. ;
x_rotation = the euler angle rotation along the x-axis for the sun light (range 0 to 180);
y_rotation = the euler angle rotation along the y-axis for the sun light (range 0 to 360);
z_rotation = the euler angle rotation along the z-axis for the sun light (range 0 to 360);
inputProvidedByAgent = this is the input that was provided to you

- Expected Output: csv file containing relevant properties.

3.3 Terrain Generator: This agent has access to a tool that is powered by a reasoning LLM. The role of this agent is to look at the task provided by the planner, specific to the generation of the terrain, and based on that create a detailed description of the different terrain features that maybe present. The detailed description is then sent to the LLM-powered tool, which is prompted through a ReAct system, where it outputs the height and influence of different Gaussians that would represent the terrain features.

- LLM Model: o4-mini (temperature = 0.8)
- Maintains Memory: False
- Flow of information:
 - Planner (P) \rightarrow Visual Expert Agent (v_{Terrain})
 - Visual Expert Agent ($v_{\text{TerrainDesc}}$) \rightarrow Terrain Generation Tool ($v_{\text{TerrainGen}}$)
- Terrain Description System Prompt:

You need to create an accurate description of a terrain based on the following topic: {terrain}
Provide the accurate description to the terrain generation tool;

- Terrain Generation System Prompt:

Using a user' s description, you can create terrains by employing Gaussian functions to express the height of specific areas. This Gaussian function is defined by the equation: $G=h \cdot \exp(-(d^2)/w)$, where:

h is the peak height,

d is the distance from the center of the Gaussian to any point in the terrain,

w is the weight of influence of that Gaussian.

Since you cannot scale or rotate the Gaussians directly, you need to create multiple Gaussians to mimic scaling and rotation. For example:

A continuous valley is defined by multiple overlapping Gaussians with negative heights along a curve.

A sharp peak on top of a cliff consists of multiple Gaussians with higher weights of influence for the cliff and smaller influences but taller heights for the peak.

You need to output four parameters for each Gaussian: the x and y coordinates of their centroid, h (peak height), and w (weight of influence).

Solve this complex problem step-by-step by answering these questions and finally providing your parameter outputs for each Gaussian. The grid size of the map is 257x257.

Does the user' s description have enough information about the terrain? If not, make logical predictions to create the terrain. No additional details will be provided by the user.

From the description/prediction, how many terrain features are present?

For each of these terrain features, what is their general location on the map?

For each of these terrain features, how many Gaussians will be needed to represent the feature?

Remember, the user expects high-quality terrains, so do not limit the use of Gaussians for simplicity.

Focusing on one feature at a time, identify the peak heights of all Gaussians representing that feature.

Subsequently, thinking about their centroid locations and their corresponding weight of influence.

Repeat this for each feature.

Typically, a large mountain centered on the map would have a weight of around 600.

Finally, provide a combined output of all Gaussian parameters like this: (x1, y1, h1, w1), (x2, y2, h2, w2), ...

Only provide the values. For example, your output for the Gaussian parameters would be something like:

FINAL OUTPUT: (100, 100, 20, 150), (50, 200, -10, 300)

- Expected Output: grayscale heightmap of the terrain.

3.4 Texture Generator: This agent has access to a tool that is powered by a text-to-image model. The role of this agent is to look at the task provided by the planner, specific to the generation of the textures pertaining to the terrain and based on that create a single sentence prompt for the text-to-image model to generate a 2D texture map. The agent then calls the text-to-image API, which produces the necessary 2D texture. Further post-processing steps are followed to create a repeatable, tile-able 2D texture.

- LLM Model: GPT-4.1 (temperature = 0.8)
- Text-to-Image Model: Dall.E 2
- Maintains Memory: False
- Flow of information:
 - i. Planner (P) → Visual Expert Agent ($v_{\text{TexturePrompt}}$)
 - ii. Visual Expert Agent ($v_{\text{TexturePrompt}}$) → Texture Generation Tool ($v_{\text{TextureGen}}$)
- Texture Prompt Generation System Prompt:

You need to create a simple one sentence prompt to generate a terrain texture. Provide the one sentence prompt to the texture generation tool. Here is the topic: {terrain_texture}

Examples of one sentence prompt: Top-down high quality photo of desert sand, avoiding shadows, within a 1ft x 1ft area; Top down high quality photo of snow, avoiding shadows, within a 1ft x 1ft area

- Expected Output: 2D texture image.

3.5 Asset Choosing Agent: This agent has access to a list of pre-existing assets in the game engine. When given a task, the agent receives a list of potential assets relevant to the scene. The agent then determines whether any of these potential assets resemble those in the pre-existing list. If a match is found, the agent selects the appropriate asset and decides how many instances of that asset should be instantiated in the scene.

- LLM Model: GPT-4.1 (temperature = 0.8)
- Maintains Memory: False
- Flow of information: Planner (P) → Visual Expert Agent ($v_{\text{TexturePrompt}}$)
- Asset Choosing System Prompt:

"Here is the flora and flora density information: {flora}, {flora_density}"

"You have access to the following assets: Conifer, Cypress, Bush_A, Bush_B, BushDry_A, BushDry_B, Grass_A, Grass_B, Grass_C, Grass_D."

"Based on the flora and flora density provided, identify if any of the available assets can be made of use and their corresponding numbers. Your output should be a csv, where the first row is names of the assets that can be used and second row is the number of that asset that needs to be added."

"For density, if it is high density, add more than 1000 if its a tree and 5000 for grass, bush, fern. Adjust accordingly for other densities."

"If there are classes of dense options such as dense trees, try using different types of trees that match that description. We want to create a diverse and consistent scene whenever possible."

"Example Output: Conifer, Grass_A, \n1000, 5000"

- Expected Output: csv containing the name of pre-existing assets and their corresponding instances.

4. Audio Expert (A_E) Team: For the audio expert team, we develop a host of LLM-powered tools that are tasked towards generating audio, identifying its spatial properties, and deciding where in the scene the sounds will be placed. As a result, the expert tools include the audio generator, spatial audio functions, and audio placement tools. Each of the tools' specifications are mentioned below:

4.1 Audio Generator: This tool receives the list of potential audio sources from the planner agent. It then iterates through each of the audio descriptions and calls a text-to-audio API (ElevenLabs API). Currently, the duration of the generated audio is set to 10 seconds.

- Text-to-Audio Model: ElevenLabs Sound Generation v1.0 (prompt influence – 0.8) [prompt influence is similar to temperature but an inverse. Higher the value, the less the model's output will deviate from the prompt, i.e., prompt will have a higher influence.]
- Flow of information:
 - i. Planner (P) \rightarrow Audio Expert Tool (a_{audio_gen})
- Prompt: Each item in a list of audio descriptions.
- Expected Output: mp3 files corresponding to each audio description.

4.2 Spatial Audio Function: This agent specializes in generating values to manipulate the spatial audio properties of each audio source in the game engine. It receives the list of audio descriptions from the planner and decides on the spatial audio properties based solely on the description.

- LLM Model: GPT-4.1 (temperature = 0.9)
- Maintains Memory: False
- Flow of information:
 - i. Planner (P) \rightarrow Audio Expert Tool ($a_{spatial_audio}$)
- System Prompt:

You need to predict values of spatial audio properties that will satisfy the users requirement. Assume that the user will not provide any additional information.

The properties you can change are as follows:

loop = is the audio source going to loop in the scene? (should be a boolean – only choose from 'true' or 'false');

maxDistance = (Logarithmic rolloff) MaxDistance is the distance a sound stops attenuating at. (float range 0 to 100.0);

minDistance = Within the Min distance the AudioSource will cease to grow louder in volume. (float range 0 to 2);

priority = sets the priority of the audio source (int range from 0 to 255, where 0 = high priority and 255 = low priority);

volume = The volume of the audio source (0.0 to 1.0);

spatialBlend = Sets how much this AudioSource is affected by 3D spatialisation calculations (attenuation, doppler etc). 0.0 makes the sound full 2D, 1.0 makes it full 3D;
pitch = The pitch of the audio source. (float range from -3 to 3)
dopplerLevel = Sets the Doppler scale for this AudioSource. (float range 0 to 5)

Your output should strictly be only values for the properties above.

An example output would look like this:

true, 100, 0, 128, 1.0, 0.8, 1, 2

- Expected Output: csv file containing relevant properties.

4.3 Audio Placement Tool: This tool specializes in identifying where in the scene the audio source needs to be instantiated. It receives the list of audio descriptions from the planner and a list of objects, with their 3D spatial properties, from the scene analyzer tool belonging to the Spatial Expert. It then decides whether the audio source should be a child of an existing game object (thereby inheriting the parent's transformation matrix), or whether it should be placed next to an existing game object, or whether it should be placed freely in 3D space with the location based on the main camera.

- LLM Model: GPT-4.1 (temperature = 0.7)
- Maintains Memory: False
- Flow of information:
 - i. Planner (P) \rightarrow Audio Expert Tool ($a_{audio_placement}$)
- System Prompt:

You will be provided a list of audio sources and a list of existing game objects in a scene. You will also be provided with an image of the scene from the perspective of the user. Your goal will be to place the audio sources in the 3D scene. You can choose one of three ways of doing this: the audio sources can be made children to existing objects from the list of objects presented to you, the audio sources can be placed next to existing objects, and the audio sources can be placed freely in the scene with respect to the location of the user. One audio source can be used multiple times. Depending on the type of process you choose for each audio source, use the following output format:

For making children of existing object:

example output format: (audio source index, "assign_child", parent object, localXoffset, localYoffset, localZoffset)

example output: (0, "assign_child", "tree_0", 0, 0.5, 0)

For placing next to existing object:

example output format: (audio source index, "place_next_to", target object, Xoffset, Yoffset, Zoffset)

example output: (4, "place_next_to", "bush_0", 2, 0, 1)

For placing freely in the scene:

example output format: (audio source index, "free", "free", Xoffset_wrt_user, Yoffset_wrt_user, Zoffset_wrt_user)

example output: (2, "free", 25, 4, 15)

The input will be provided in the following format: object name, distance to user, x_size, y_size, z_size
It is exceptionally important that you follow the format provided in the example output below. Do not add any other preceding or succeeding text as it will break my system.

Example Input:

list of audio: "bird chirping sounds", "leaves rustling", "wind blowing", "insects chirping"]
list of objects:
tree_1, 12, 2, 4, 3
tree_0, 5, 2, 4, 3
stone_0, 6, 0.5, 1, 1

Example Output:

```
(0, "assign_child", "tree_0", 0, 0.5, 0)
(0, "assign_child", "tree_1", 0, 0.5, 0)
(1, "assign_child", "tree_0", 0, 0.75, 0)
(2, "free", "free", 4, 5, 0)
(2, "free", "free", 1, 3, 1)
(3, "place_next_to", "stone_0", 1, 1, 0)
(3, "place_next_to", "tree_0", 0.5, 0, 0)
(3, "free", "free", 4, 0.25, 3)
```

- Expected Output: csv file containing relevant properties.

5. Spatial Expert (A_E) Team: The spatial expert team, as explained in the methodology section, consists of the scene analysis and object placement tools implemented in the Unity and interacts with the other agents through simple file reading and writing.

6. Reflective Partner (R): The reflective partner is the agent that evaluates the performance of the experts (V_E, A_E, S_E) by visually comparing their outputs to the plan (T) generated by the Planner (P). It conducts this evaluation by receiving screenshots of the scene as captured by a camera in the Unity game engine and being powered by a vision language model. The details of the reflective partner are given below:

- VLM Model: GPT 4.1 (temperature – 0.5)
- Maintains Memory: True
- Flow of information:
 - i. Planner(P) \leftrightarrow Reflect Partner (R)
 - ii. Camera Shots (C) \rightarrow Reflect Partner (P)
 - iii. Planner (P) \rightarrow Experts (V_E, A_E, S_E)
- System Prompt:

You are an expert reflective partner. Your goal is to think deeply about the plan that is provided to you, understand the image and provide details on whether the image follows the plan, and if not how can it be improved. Use the tool to output structured data

- Function Specifications:

detailed_reflection: str = "Detailed reflection on whether the visual scene matches the plan. Why/Why not."

detailed_weather: str = "Detailed description of the weather conditions (e.g., 'sunny with scattered clouds')

detailed_lighting: str = "Detailed description of the lighting conditions (e.g., 'bright sunlight with the sun at noon')

detailed_geographic_terrain: str "Detailed description of the geographic terrain (e.g., 'rocky mountains with a steep incline')

terrain_texture: List[str]= "List of textures used for the terrain ground (e.g., ['rock', 'sand', 'mud', 'grass'])

```

    flora_and_details: List[str] = "List of flora and details present in the scene (e.g., ['pine trees',
'bushes', 'twigs', 'small rocks', 'large rocks']).
    flora_density: List[str] = "List of density values corresponding to the flora and details (e.g.,
['dense', 'dense', 'few', 'medium', 'sparse']
    detailed_audio_sounds_description: List[str]= "List of very detailed descriptions of audio sounds
used in the scene (e.g., ['woodpecker pecking on the tree', 'small leaves rustling in the wind'])

```

3. Results

In this section, we will highlight more results produced by the individual agents, expert teams, and tools in isolation of the multi-agent workflow.

a. Manager & Planner:

Below is **Example 1** of the conversation between a user and a manager and the subsequent steps:

Please provide a short description of the scene you want to create.

User (q/Q to quit): i want to create a scene of a lush green forest

===== Ai Message =====

Great! Let's dive into the details to make your lush green forest scene truly immersive.

Visual Elements

What specific visual elements are most important to you for this forest scene? For instance, do you have preferences for the colors, lighting, textures, or any visual effects?

Audio Elements

To make the scene more immersive, what kind of audio elements are you thinking about? Do you want background music, specific sound effects like rustling leaves or chirping birds, or even voice-overs?

Setting and Environment

Can you provide some details about the setting and environment? Is this forest realistic or more fantastical? Are there any specific features like rivers, cliffs, or magical elements you envision?

Feel free to elaborate on anything else you think is crucial for your scene!

User (q/Q to quit): can you make predictions?

===== Ai Message =====

Absolutely, I'll make some detailed predictions to create an immersive lush green forest scene for you.

Visual Elements

- **Colors**: A palette of vibrant greens for the foliage, mixed with earthy browns for tree trunks and forest floor. Splashes of colorful flowers (reds, yellows, purples) to add variety.
- **Lighting**: Soft, dappled sunlight filtering through the tree canopy, creating a mix of light and shadow on the forest floor.
- **Textures**: Rich, detailed textures for leaves, moss-covered rocks, tree bark, and the forest floor, which might be covered with fallen leaves and twigs.
- **Visual Effects**: Gentle sway of the trees and plants in the breeze, occasional shafts of sunlight breaking through the canopy, and perhaps some light mist or fog to add depth.

Audio Elements

- ****Background Music****: Gentle, ambient music with natural instruments like flutes and strings to enhance the serene atmosphere.
- ****Sound Effects****:
 - Rustling leaves as the wind moves through the trees.
 - Birdsong from various species to indicate a thriving ecosystem.
 - Occasional animal sounds like distant deer calls or rustling from small critters.
 - A gentle stream or river flowing in the background.
- ****Ambient Sounds****: The sound of leaves crunching underfoot and the occasional snap of twigs.

Setting and Environment

- ****Location****: A realistic, dense forest with towering trees, a mix of deciduous and evergreen species.
- ****Specific Features****:
 - A clear, meandering stream or small river running through the scene.
 - Moss-covered rocks and fallen logs adding visual interest.
 - Possibly a small clearing with wildflowers providing a focal point.
 - Occasional beams of sunlight breaking through the canopy, highlighting certain areas.

Does this detailed description align with your vision for the scene? If so, I'll proceed to create an immersive scene description. If you need any adjustments, let me know!

User (q/Q to quit): this looks good!

===== Ai Message =====

Tool Calls:

ImmersiveSceneDescription (call_fAr6E6KSklo0gC8XtAi9082M)

Call ID: call_fAr6E6KSklo0gC8XtAi9082M

Args:

detailed_description: The scene is set in a lush green forest, alive with natural beauty and tranquility. The forest is dense with towering trees, a mix of deciduous and evergreen species, creating a vibrant palette of greens. The foliage is rich and varied, with splashes of colorful flowers in reds, yellows, and purples adding visual interest. The forest floor is covered with a thick carpet of fallen leaves, twigs, and moss-covered rocks, all depicted with rich, detailed textures.

Soft, dappled sunlight filters through the tree canopy, casting a mix of light and shadow on the forest floor. Occasional beams of sunlight break through, highlighting specific areas and adding depth to the scene. The gentle sway of the trees and plants in the breeze, along with light mist or fog, enhances the immersive experience.

unlight break through, highlighting specific areas and adding depth to the scene. The gentle sway of the trees and plants in the breeze, along with light mist or fog, enhances the immersive experience.

breeze, along with light mist or fog, enhances the immersive experience.

The audio elements are carefully designed to complement the visual beauty. Gentle, ambient music with natural instruments like flutes and strings plays in the background, creating a serene atmosphere. The forest is alive with the sounds of nature: rustling leaves as the wind moves through the trees, diverse birdsong, distant deer calls, and the rustling of small critters. A clear,

meandering stream or small river flows through the scene, its gentle babbling adding to the sense of tranquility. Additional ambient sounds of nature: rustling leaves, calls, and the rustling of small critters. A clear, meandering stream or small river flows through the scene, its gentle babbling adding to the sense of tranquility. Additional ambient sounds include leaves crunching underfoot and the occasional snap of twigs.

The setting is realistic, with specific features that make the forest come alive. A clear stream or river runs through the scene, surrounded by moss-covered rocks and fallen logs. A small clearing with wildflowers provides a focal point, and the occasional shafts of sunlight breaking through the canopy highlight certain areas, adding to the overall magic of the forest environment.

===== Tool Message =====

Plan generated! [PLANNER OUTPUT]

===== Ai Message =====

Tool Calls:

ImmersiveInstructionPlan (call_i2a2eZSR7LN7KxadW1VR24IJ)

Call ID: call_i2a2eZSR7LN7KxadW1VR24IJ

Args:

detailed_weather: Clear weather with a gentle breeze, enhancing the overall tranquility of the forest.

detailed_lighting: Soft, dappled sunlight filters through the tree canopy, casting a mix of light and shadow on the forest floor. Occasional beams of sunlight break through, adding depth and highlighting specific areas.

detailed_geographic_terrain: The terrain is typical of a dense forest, with a thick carpet of fallen leaves, twigs, and moss-covered rocks. There is a clear, meandering stream or small river running through the scene, adding visual interest.

terrain_texture: ['leaves', 'twigs', 'moss', 'rocks']

flora_and_details: ['towering trees', 'deciduous trees', 'evergreen trees', 'colorful flowers', 'fallen logs', 'wildflowers']

flora_density: ['dense', 'dense', 'dense', 'sparse', 'medium', 'sparse']

detailed_audio_sounds_description: ['rustling leaves as the wind moves through the trees', 'diverse birdsong', 'distant deer calls', 'rustling of small critters', 'gentle babbling of a stream or river', 'leaves crunching underfoot', 'occasional snap of twigs']

Example 2:

Please provide a short description of the scene you want to create.

User (q/Q to quit): i want to create a desert scene

===== Ai Message =====

Great! Let's start by diving into some specifics.

What kind of visual elements are most important to you in this desert scene? Could you describe the colors, lighting, and textures you envision?

User (q/Q to quit): the first thing i think of when i think of deserts is heat, bright sunlight and sand everywhere

===== Ai Message =====

Got it. Bright sunlight, heat, and lots of sand.

Now, let's talk about audio. What kind of background music or sound effects do you think will make this desert scene more immersive?

User (q/Q to quit): hmm.. i think it would be windy there. maybe something related to that. what else would make sense?

===== Ai Message =====

Windy sounds are a great choice. In addition to that, we could include occasional distant animal sounds like birds or insects, and perhaps some faint rustling of sand.

Does that sound good to you? And would you like any background music or voice-overs, or should the focus be on ambient sounds only?

User (q/Q to quit): that sounds good. Maybe some subtle background music taking inspiration from the movie 'The Mummy'

===== Ai Message =====

Excellent choice with the music inspiration. Now, let's discuss the setting and environment in more detail.

Where do you want this scene to take place? Is this a realistic desert, or something more fantastical? Are there any specific features or atmospheres you want in this environment?

User (q/Q to quit): it is a realistic scene. maybe sand dunes, a few desert rocks. Probably desert plants

===== Ai Message =====

Perfect! To summarize:

- **Visual Elements**: Bright sunlight, heat, sand everywhere, sand dunes, desert rocks, and some desert plants.
- **Audio Elements**: Windy sounds, occasional distant animal sounds (birds, insects), rustling of sand, and subtle background music inspired by "The Mummy."
- **Environment**: Realistic desert setting with features such as sand dunes, a few desert rocks, and desert plants.

Let's proceed to create a detailed and creative description of this immersive desert scene.

Tool Calls:

ImmersiveSceneDescription (call_oXUL2IY9ERPtoMTWgABGlyod)

Call ID: call_oXUL2IY9ERPtoMTWgABGlyod

Args:

detailed_description: The scene is set in a vast, realistic desert. The sun blazes brightly in a cloudless sky, casting an intense, golden light over the landscape, creating an aura of blistering heat. Sand stretches endlessly, with towering sand dunes shaped by the relentless wind, their surfaces marked by intricate, rippling textures. Scattered around are a few rugged desert rocks, weathered by time and the elements. Sparse vegetation breaks the monotony: hardy desert plants like cacti and small shrubs dot the terrain, their shadows providing minimal contrast to the sun-drenched surroundings.

The audio landscape complements the visual elements perfectly. The sound of a constant, inescapable wind sweeps across the scene, carrying with it the faint rustling of sand. Occasionally, distant chirps of desert birds and the intermittent buzz of insects add a touch of life to the arid environment. Subtle background music, reminiscent of the evocative score from 'The Mummy,' plays softly, enhancing the sense of mystery and ancient allure without overpowering the ambient sounds. The overall atmosphere is one of isolation, heat, and the rugged beauty of the desert.

===== Tool Message =====

Plan generated! **[PLANNER OUTPUT]**

===== Ai Message =====

Tool Calls:

```
ImmersiveInstructionPlan (call_VQmEag8nNS1lkFWmFpmU2TEQ)
Call ID: call_VQmEag8nNS1lkFWmFpmU2TEQ
Args:
  detailed_weather: Blazing sun in a cloudless sky, creating an intense heat
  detailed_lighting: Bright sunlight, casting a golden light over the landscape
  detailed_geographic_terrain: Vast desert with towering sand dunes and scattered desert rocks
  terrain_texture: ['sand']
  flora_and_details: ['cacti', 'small shrubs', 'desert rocks']
  flora_density: ['sparse', 'sparse', 'few']
  detailed_audio_sounds_description: ['constant wind sweeping across the scene', 'faint rustling of
sand', 'distant chirps of desert birds', 'intermittent buzz of insects', "subtle background music
reminiscent of 'The Mummy'"]
```

b. Weather System Generator:

The weather system generator oversees adjusting the volumetric properties of the scene in the game engine. It specifically outputs the following values, along with a Boolean for rain/snow:

densityMultiplervalue, shapeScaleValue, shapeFactorvalue, erosionFactorValue, erosionScalevalue, earthCurvatureValue, bottomAltitudeValue, altitudeRangeValue, windSpeedvalue, windOrientationValue, ambientLightProbeDimmerValue, sunLightDimmerValue, scatteringTintValueColorR, scatteringTintValueColorG, scatteringTintValueColorB, densityCurveKeyFrameX1, densityCurveKeyFrameY1, densityCurveKeyFrameX2, densityCurveKeyFrameY2, densityCurveKeyFrameX3, densityCurveKeyFrameY3



Prompt: “Blue skies with a few clouds”

Output: 0.05, 10.5, 0.75, 0.2, 15.0, 0.2, 1200, 2000, 15, 180, 0.8, 0.9, 70, 130, 255, 0.5, 0.5, 1.5, 1.0, 3.0, 0.5

Precipitation: False



Prompt: “a scene from movie Home Alone”

Output: 0.66, 25.00, 0.55, 0.40, 15.00, 0.50, 1000.00, 4000.00, 50.00, 180.00, 0.30, 0.20, 200, 180, 255, 0.80, 0.90, 1.50, 1.20, 3.00, 1.00

Precipitation: True (Snow)



Prompt: “Intense and scary climate”

Output: 0.9, 25.0, 0.7, 0.5, 20.0, 0.2, 1500.0, 3500.0, 100.0, 180.0, 0.1, 0.2, 128, 0, 0, 0.5, 1.5, 2.0, 0.1, 2.0, 0.9

Precipitation: True (Rain)



Prompt: “morning in San Francisco”

Output: 0.15, 10.00, 0.50, 0.20, 15.00, 0.30, 1200.00, 3000.00, 10.00, 90.00, 0.50, 0.80, 255, 223, 186, 0.20, 0.50, 1.00, 0.70, 3.00, 1.50

Precipitation: False



Prompt: “A scene set in Sunnyvale, CA”

Output: 0.02, 8.75, 0.45, 0.12, 15.00, 0.18, 1200.00, 3000.00, 20.00, 180.00, 0.70, 0.80, 255, 255, 255, 0.5, 1.0, 1.5, 0.5, 3.0, 2.0

Precipitation: False



Prompt: “the calm before the storm”

Output: 0.25, 15.34, 0.70, 0.20, 18.40, 0.30, 1500.00, 3000.00, 80.50, 120.00, 0.60, 0.50, 200, 180, 220, 0.50, 0.80, 1.50, 0.70, 2.00, 0.90

Precipitation: False

c. Lighting Functions:

The lighting functions agent oversees adjusting the lighting properties present in the scene. It specifically outputs the following values, along with a Boolean for rain/snow:

angDia, flareSize, flareFalloff, flareTintColorR, flareTintColorG, flareTintColorB, lightIntensity, emissionColorR, emissionColorG, emissionColorB, colorTemp, multiplier, shadowDimmer, x_rotation, y_rotation, z_rotation



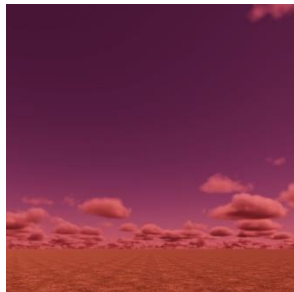
Prompt: “Sunny afternoon”

Output: 2.0, 3.0, 10.0, 255, 165, 0, 13000, 255, 223, 191, 6000, 1.0, 0.5, 45, 180, 0



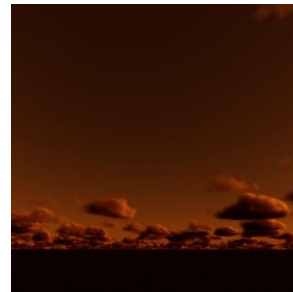
Prompt: “a beautiful sunrise”

Output: 3.5, 8.0, 45.0, 255, 200, 150, 6500, 255, 150, 100, 3500, 2.5, 0.6, 15, 90, 0



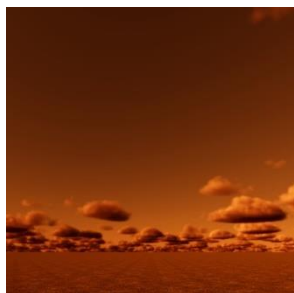
Prompt: “a scene from the cyberpunk game”

Output: 3.0, 6.0, 45.0, 200, 50, 50, 5000, 255, 100, 100, 9000, 8.0, 0.7, 60, 180, 0



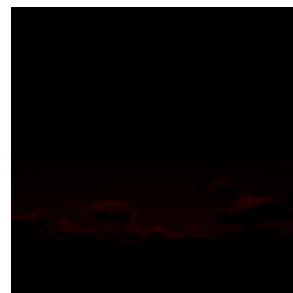
Prompt: “a scene at dusk”

Output: 3.0, 6.0, 45.0, 255, 100, 100, 500, 255, 180, 120, 3000, 0.5, 0.5, 10.0, 180.0, 0.0



Prompt: “10pm in Norway during summer”

Output: 1.5, 3.0, 45.0, 255, 100, 50, 2000, 255, 140, 60, 4000, 0.5, 0.3, 15, 330, 0

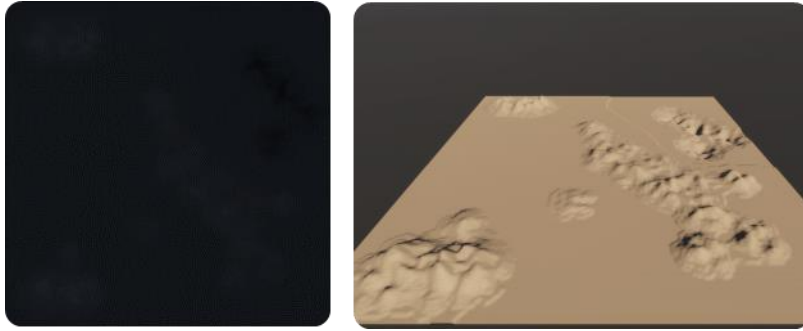


Prompt: “10pm in Norway during winter”

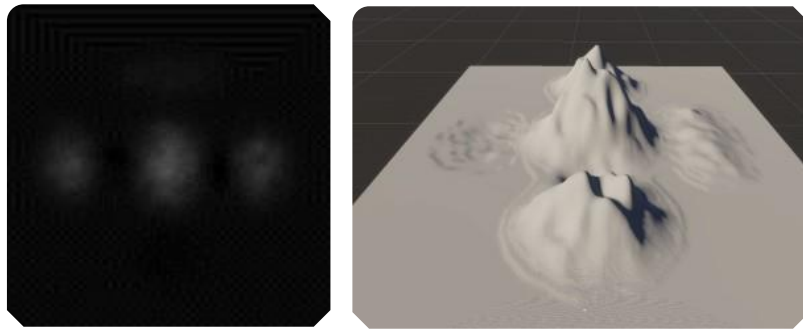
Output: 1.0, 2.0, 45.0, 255, 196, 122, 70, 255, 196, 122, 2000, 0.5, 0.8, 0, 270, 0

d. Terrain Generator:

The terrain generator agent outputs the gaussian-voronoi 2D arrays which are passed through 2 to 3 iterations of gaussian blur function and then converted to a grayscale heightmap. The heightmap is parsed by the Unity game engine to generate the 3D terrains. In the results below, the heightmap is shown on the left and the generated Unity scene on the right



Prompt: *“a scene in the desert”*



Prompt: *“snowy mountain ranges”*

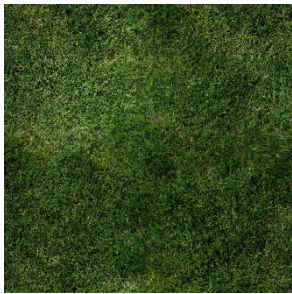
e. Texture Generator



Prompt (from planner): “*desert sand*”



Prompt (from planner): “*snow*”



Prompt (from planner): “*green grass*”

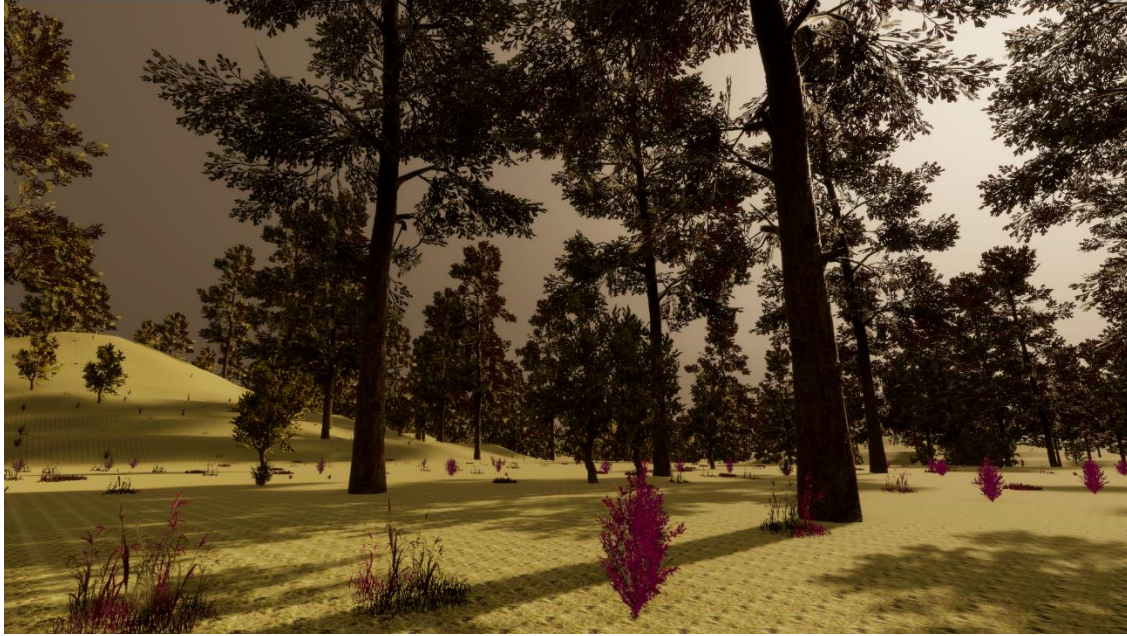


Prompt (from planner): “*rocky mountain*”

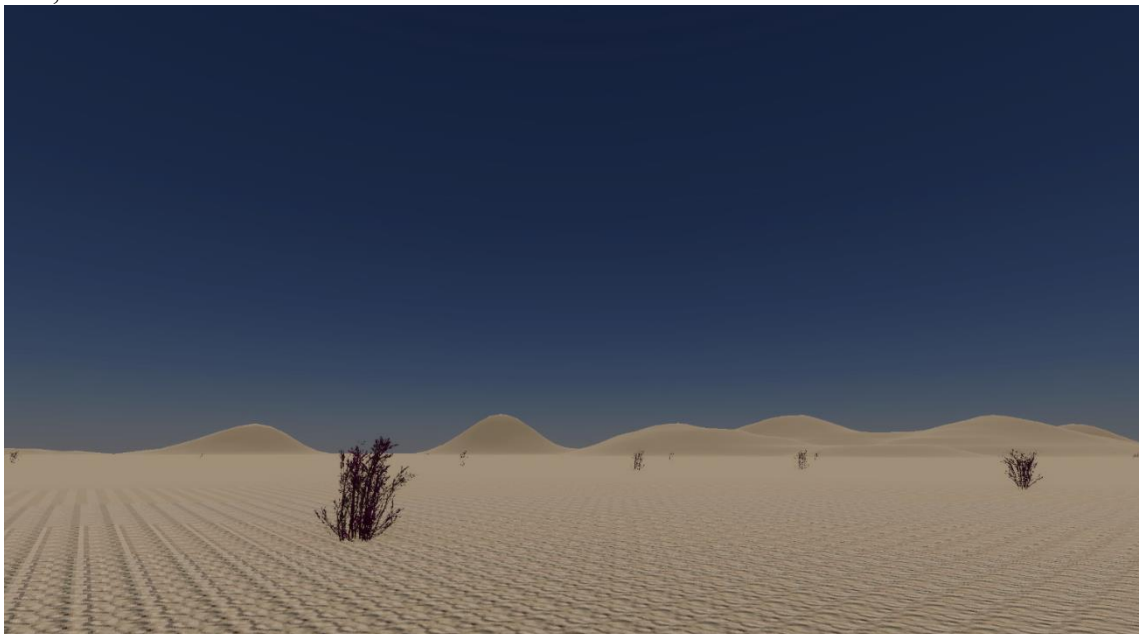
f. Asset Chooser

The asset chooser agent understands the planner's output regarding the assets that need to be added to the scene and then looks at all the pre-defined assets available to it. Based on that it chooses the assets that are relevant to the planner's output and lists them out along with their respective numbers/instances that need to be instantiated.

Conifer, Cypress, Bush_A, Bush_B, Grass_A, Grass_B, Grass_C, Grass_D
1000, 1000, 5000, 5000, 5000, 5000, 5000, 5000



BushDry_A, BushDry_B
500, 500



g. Audio Expert Team

Please refer to the results ppt file for the results generated by the audio generation agent. Some of the outputs of the spatial audio functions and audio placements functions are provided below, however without the video, they would lack context.

Spatial Audio Function:

It outputs the following audio source properties:

loop = is the audio source going to loop in the scene? (should be a boolean - only choose from 'true' or 'false');
maxDistance = (Logarithmic rolloff) MaxDistance is the distance a sound stops attenuating at. (float range 0 to 100.0);
minDistance = Within the Min distance the AudioSource will cease to grow louder in volume. (float range 0 to 2);
priority = sets the priority of the audio source (int range from 0 to 255, where 0 = high priority and 255 = low priority);
volume = The volume of the audio source (0.0 to 1.0);
spatialBlend = Sets how much this AudioSource is affected by 3D spatialisation calculations (attenuation, doppler etc). 0.0 makes the sound full 2D, 1.0 makes it full 3D;
pitch = The pitch of the audio source. (float range from -3 to 3)
dopplerLevel = Sets the Doppler scale for this AudioSource. (float range 0 to 5)

Audio description: “wind blowing in the desert”

Output: false, 50, 0.5, 128, 0.5, 1.0, 1, 1

Audio description: “rustling sounds of the sand”

Output: false, 50, 0.5, 128, 0.5, 1.0, 1, 1

Audio description: “background music”

Output (multiple outputs for multiple instantiations): false, 50, 0.5, 128, 0.5, 1.0, 1, 1

false, 50, 1, 100, 0.5, 1.0, 0, 1

false, 20, 0.5, 200, 0.6, 0.5, 0, 1

false, 50, 1, 100, 0.5, 0.7, 0, 1

Audio Placement Function:

It outputs the following properties of the spatial expert function:

(audio source index, "assign_child", parent object, localXoffset, localYoffset, localZoffset)

(audio source index, "place_next_to", target object, Xoffset, Yoffset, Zoffset)

(audio source index, "free", "free", Xoffset_wrt_user, Yoffset_wrt_user, Zoffset_wrt_user)

0,assign_child,Rock_A_02_LOD04,0,0.5,0

1,assign_child,Rock_A_02_LOD02,0,0.5,0

2,place_next_to,BushDry_B_176,1,0,0

3,free,free,5,0.5,2

h. Spatial Expert Team

Given that the team was primarily implemented in Unity as functions, the outputs are related to the 3D placements of objects in the virtual environment. As such,

only the Scene Analysis & Understanding function generated output used by other agents. It outputs the following values: object name, distance to user, x_size, y_size, z_size

The following output is for a 'desert scene'



View of the scene

```
Rock_A_02_LOD04,3.15568,0.2386215,0.3285345,0.1666722
Rock_A_02_LOD03,3.15568,0.2386215,0.3288367,0.1666722
Rock_A_02_LOD02,3.15568,0.2404813,0.3288367,0.1688944
Rock_A_02_LOD01,3.15568,0.2409847,0.3294596,0.1696592
Rock_A_02_LOD00,3.15568,0.2409847,0.3294596,0.1696754
Rock_A_02_LOD04,3.801896,0.2386215,0.3285345,0.1666722
Rock_A_02_LOD03,3.801896,0.2386215,0.3288367,0.1666722
Rock_A_02_LOD02,3.801896,0.2404813,0.3288367,0.1688944
Rock_A_02_LOD01,3.801896,0.2409847,0.3294596,0.1696592
Rock_A_02_LOD00,3.801896,0.2409847,0.3294596,0.1696754
Rock_A_02_LOD04,4.602651,0.2386215,0.3285345,0.1666722
Rock_A_02_LOD03,4.602651,0.2386215,0.3288367,0.1666722
Rock_A_02_LOD02,4.602651,0.2404813,0.3288367,0.1688944
Rock_A_02_LOD01,4.602651,0.2409847,0.3294596,0.1696592
Rock_A_02_LOD00,4.602651,0.2409847,0.3294596,0.1696754
Rock_A_02_LOD04,4.935704,0.2386215,0.3285345,0.1666722
Rock_A_02_LOD03,4.935704,0.2386215,0.3288367,0.1666722
Rock_A_02_LOD02,4.935704,0.2404813,0.3288367,0.1688944
Rock_A_02_LOD01,4.935704,0.2409847,0.3294596,0.1696592
Rock_A_02_LOD00,4.935704,0.2409847,0.3294596,0.1696754
Rock_A_02_LOD04,6.2509,0.2386215,0.3285345,0.1666722
Rock_A_02_LOD03,6.2509,0.2386215,0.3288367,0.1666722
Rock_A_02_LOD02,6.2509,0.2404813,0.3288367,0.1688944
Rock_A_02_LOD01,6.2509,0.2409847,0.3294596,0.1696592
Rock_A_02_LOD00,6.2509,0.2409847,0.3294596,0.1696754
BushDry_A_123,7.21372,0.2745735,0.3036128,0.2198444
BushDry_A_275,7.352441,0.2745735,0.3036128,0.2198444
BushDry_B_216,9.327816,0.3596541,0.332011,0.3218468
```

4. Limitations

This study presents several limitations that should be considered when interpreting the results. First, the current approach primarily focuses on the generation of natural environments, which restricts the diversity of settings that can be produced: such as urban, futuristic, or fantasy landscapes. One of the primary complexities in generating more dense urban or man-made environments is the necessity of creating high-fidelity 3D models and assets, which remains an ongoing research challenge. Additionally, as more objects are introduced into these complex scenes, ensuring semantically consistent object placement becomes increasingly difficult, as many objects require highly specific locations within the environment. Although these challenges are the subject of active research, the multi-agent framework has the potential to leverage new models and methods as they are developed, integrating them as part of agent-specific tasks to improve both the diversity and realism of generated environments. This limitation suggests a need for integrating expert agents specialized in various types of environment generation to further expand the range of environments possible.

Another limitation is the specificity of the human-in-the-loop mechanism for invoking specific individual agents, particularly in tasks where human judgment is crucial (e.g., assessing audio output quality). Allowing human feedback to invoke specific expert agents to guide and refine their actions without affecting other agents outputs would improve the system's overall accuracy and effectiveness. Furthermore, as task complexity increases, the multi-agent workflow may experience delays, compromising the system's ability to generate results in real time. This calls for workflow optimization and more efficient task distribution among agents, as well as the development of expert agents to prioritize and streamline complex operations in parallel.

The generalizability of the proposed system also presents a limitation, as it is currently tailored for the Unity game engine, which restricts its applicability to other platforms, such as Unreal Engine. Future work could focus on developing agents that can extract and interpret documentation from various game engines, enabling the generation of engine-relevant outputs across platforms. Finally, while the aim is to create immersive environments, the evaluation of immersiveness relies on user studies that capture quality of experience and immersion metrics. As such, future work should include comprehensive user studies to assess and validate the immersive qualities of the generated environments.

References

- [1] “Applied Sciences | Free Full-Text | Translating Words to Worlds: Zero-Shot Synthesis of 3D Terrain from Textual Descriptions Using Large Language Models.” Accessed: Aug. 08, 2024. [Online]. Available: <https://www.mdpi.com/2076-3417/14/8/3257>
- [2] F. De La Torre, C. M. Fang, H. Huang, A. Banburski-Fahey, J. Amores Fernandez, and J. Lanier, “LLMR: Real-time Prompting of Interactive Worlds using Large Language Models,” in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, in CHI '24. New York, NY, USA: Association for Computing Machinery, May 2024, pp. 1–22. doi: 10.1145/3613904.3642579.