# [GitHub] Genies IRL Documentation

# 1. Description

## 1.1 Overview

**Genies IRL** is an XR project for Apple Vision Pro that demonstrates how virtual characters can perceive and interact with the physical world in a believable manner. Built upon Genies' modular avatar tech stack, Genies IRL leverages Apple's and Unity's spatial computing capabilities to showcase next-generation, character-driven XR experiences.

**This open-source project mirrors the official Genies IRL application, available on the Vision Pro App Store.** It serves as an example for how, using the advanced capabilities of Apple Vision Pro, a virtual character can be brought to life in your real-world space.

## 1.2 Why Genies IRL Matters

The leap from conventional AR/VR avatars to context-aware, fully interactive characters is essential for the next generation of XR experiences. Genies IRL explores solutions to core challenges such as spatial awareness, real-world interaction, and intelligent behavior orchestration. These advancements are crucial not just for gaming, but for future applications in education, productivity, social presence, and more.

Genies IRL demonstrates:

- How avatars can **navigate real spaces**, avoiding obstacles and understanding context.
- How avatars can **interact with surfaces and objects** in the user's environment.
- How avatars can **respond intelligently** to both the world and the user.
- How to utilize **Unity and industry-standard packages** (such as Aron Granberg's A* Pathfinding Project) for robust XR character logic.
- The value of **interoperability** using Genies' modular avatar technology.

## 2. Use Cases

The Genies IRL codebase serves as a foundation for a wide range of XR applications involving intelligent virtual characters. Some example use cases include:

### 2.1 AR Games with Spatially-Aware NPCs

Build augmented reality games where non-player characters (NPCs) act as opponents, allies, or guides—moving around your actual room, hiding behind real furniture, and engaging with players in shared physical space.

### 2.2 Interactive AR/VR Experiences with Embodied Agents

Create AR or VR applications where avatars or agents can perceive and respond to users' gestures, body movements, and positions – enabling new forms of embodied interaction, training, fitness, and social presence.

Embodiment within physical and virtual spaces creates rich opportunities for smart avatars to express themselves by using furniture, painting on walls, decorating rooms, etc.

## 3. Getting Started

This section outlines the system requirements and installation steps necessary to set up your project for development.

### 3.1 System Requirements

To build and run Genies IRL, you will need:

- **Compatible Mac computer**
- **Apple Vision Pro** running visionOS 2.0 or above
- **Unity Editor** version 6000.0.42f1 or later

  *(Note: Higher versions may require a project upgrade)*
- **Unity Pro License** (required for Apple Vision Pro development)
- **Xcode** 1.2 or later

  *(Note: Higher versions may require a Unity Editor upgrade for compatibility)*

### 3.2 Project Installation Steps

1. **Open the Project in Vision OS Build Mode**

   Open the project in Unity Editor with Vision OS build mode enabled.

   *When you first open the project, compiler errors will appear due to missing dependencies.*
2. **Install Dependencies: A* Pathfinding Project**

   Genies IRL uses Aron Granberg's A* Pathfinding Project for spatial navigation. For best performance, the **Pro** version is recommended (supports asynchronous scanning), however, the **Free** version is also compatible.

   - **Free Version Installation (Tested with version 4.2.17):**
     - Download here: A* Pathfinding Project Free Download
     - Import into Unity:
       1. In the Unity Editor, go to:

          Assets > Import Package > Custom Package

          *(You may need to Exit Safe Mode to access this.)*
       2. Select the downloaded package and import all files.
   - **Pro Version Installation (Tested with version 5.3.7):**
     - Purchase: A* Pathfinding Project Pro (Unity Asset Store)

- Follow the package installation guide.

  *(You may need to Exit Safe Mode in the Editor*

- In Player Settings, under each Platform tab, add an additional Scripting Define Symbol: "ASTAR_PRO".

3. **Resolve Compiler Errors**

   After importing the A* Pathfinding Project, compiler errors should be resolved.

4. **Open and Run the Main Scene**

   The main scene is located at:

   Assets/Project/Scenes/Main

## 4. End-User Genies IRL Experience

Whether you're running the App Store version or your own build of Genies IRL, here's a breakdown of the end-user experience and application capabilities.

*NOTE: From here on, you may see the word "Genie" to describe the character or avatar. We at Genies colloquially use the term "Genie" to describe such a character, so you'll see this used frequently in the codebase as well.*

### 4.1 Setup Phase

1. The user will be taken through an initial setup phase. You can gaze and pinch at the slides to skip through them.
2. The purpose of the scanning phase is to give the app a baseline understanding of the area. The app will continue to scan throughout the life of the experience.
3. At the end of the setup phase, the user will be instructed to give a "thumbs up" to spawn the Menu, which includes a button to spawn the Genie.
4. When the Menu launches, it will include a button to teleport the Genie in front of the user. Use gaze/pinch or simply poke the button with you finger to select it.

### 4.2 Menu

The user will be instructed to access the Menu via a “thumbs up” gesture. The Menu can be summoned at any time with this gesture.

The Menu contains the following UIs and functionalities:

- **Teleport Here Button**: Use this to Teleport the Genie directly in front of the user’s look direction. You can use this to move the Genie in case they get into a situation where they cannot move (for example, if they originally spawned in-between two desks and are trapped).
- **Brain Inspector To-Do List:** Displays each of the Genies’ tasks, including whether she has already completed it, or it is currently in progress. Note that if the Genie is interrupted, that task will not register as completed.
- **Celebration Button: When all tasks have been completed, the user will be presented with an option to spawn balloons in a huge AR party.**
- **About:** Displays attributions for sound files, etc.
- **Debug Submenu:** Provides certain debug functionality such as plane detection visualization and more.
- **Quit Button: Triple-Tap to completely close the application.**

## 4.3 Spatial Scanning Tips

Simply looking at a chair may not be enough for the device to register it, and the same goes for walls, windows, and any other trackable real-world object. You’ll want to really get up-close-and personal. For ceilings, make sure to look up to get a proper scan!

Keep in mind that the scan range is actually quite low. Moving side-to-side can help with scanning as well.

## 4.4 Genie Capabilities

### Spatial Pathfinding

The application uses the spatial mesh to construct a navigable grid for pathfinding, allowing the Genie to navigate around objects in the environment. This grid updates with the spatial mesh, and it takes into account the user’s position, allowing her to navigate around the user.

While making decisions, the Genie can determine whether a path is possible before even trying to take it.

**Tasks**

The Genie has a dozen tasks she can accomplish. Each task is broken up into steps called Actions.

*NOTE: For our purposes in the experience, we use the word **'task'**, which actually maps closely to the character's **'Plans'** in its goal-oriented brain structure. We'll delve into this in the later technical sections of this document.*

Before selecting a task, the Genie will check if the task is possible to accomplish (for example, if the task requires walking over to a location, the Genie will first check if that location is unblocked and pathable).

Once the Genie selects a task, they will attempt to accomplish it by following various steps, often including pathfinding. If a step fails, the task will fail and the Genie will abandon it for a new Goal.

| **Goal** | **Description** | **Requirements** |
|---|---|---|
| Wave At User | Genie stands and looks at the user, waving. | None |
| Give High-Five | Genie sees the user soliciting a high-five and reciprocates. | Genie must recognize the user's high-five gesture and be able to path close enough to the user. |
| Receive High-Five | Genie solicits a high-five by raising her hand in the air. | Genie must be able to navigate close enough to the user. |
| Offer Item | Genie holds out a sheep plushie to the user. If there are no plushies available, she will spawn one. | Genie must be able to navigate close enough to the user. |
| Zone Out | The Genie will occasionally take a "break" by chewing gum, looking at her nails, or texting on her smartphone. | None |
| Put Item on Table | The Genie will demonstrate her understanding of table surfaces by placing sheep plushies on them. | There must be at least one plushie in the world that is pathable. There must also be at least one recognized table free |

| | | |
|---|---|---|
| | | from obstructions, and that table must be pathable. |
| React to Projectile | The Genie will react to being hit by a sheep plushie. If struck from behind, she will perform a “what was that” reaction and look all around. If she was hit from the front, she’ll turn to the user and react playfully. | The user must toss a plushie at the Genie, and it must make contact. |
| Draw on Wall | The Genie will draw hearts on walls. | There must be a scanned wall with enough flat, unobstructed space, and it must be pathable by the Genie. |
| Sit on Seat | The Genie will sit on seats. | There must be at least one recognized seat that is unobstructed and pathable. |
| Admire Window | The Genie will place her hand on a window pane and look out. | There must be at least one recognized window that is pathable. |
| Throw Pencil at Ceiling | The Genie will throw pencils that stick on the ceiling. | There must be some recognized ceiling space, and the area underneath must be pathable. |
| Maintain Personal Space | The Genie will avoid the user when they step into their personal space. | The user must get within the Genie’s personal space. |

### 4.5 Celebration

When the Genie has completed every task in the To-Do list, a “Celebrate” button appears on the Menu. Press this button to spawn a ton of balloons for your enjoyment. Note that the frame-rate may worsen, especially if you have scanned a large area.

## 5. Development Workflow

We’ve found the best development workflow involves switching between three modes: Editor-Centric, Device-Centric, and Play-To-Device. We’ve added some tooling to make this easier, and we’ve also identified some “gotchas” and workarounds for various limitations.

## 5.1 Editor-Centric Development

When we say, Editor-Centric, we mean developing, testing, and iterating inside the Unity Editor. This is likely where you'll spend the most of your time due to ease of development and rapid iteration time.

**Simulating XR Input**

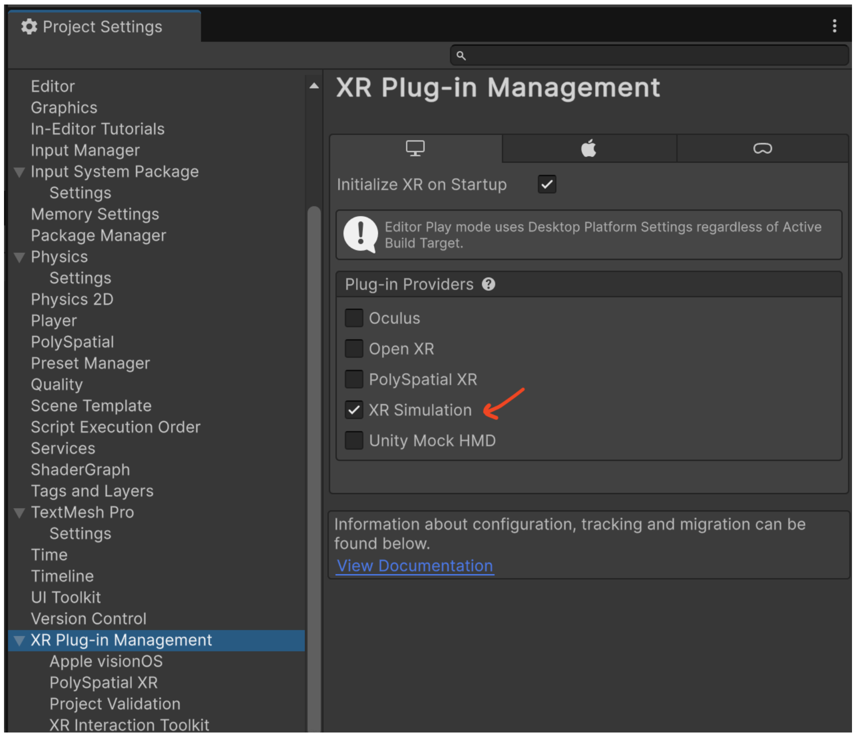
When in Play Mode, in the Editor, you can simulate user interactions:

- While holding the right mouse button:
  - Mouse look
  - WASD to "walk"
  - Q and E to raise and lower
- Click the left mouse button to simulate gaze-and-pinch
  - Works on UIs
  - Works on grabbable items like the sheep plushie
- The space bar summons the Main Menu

Note that you do *not* have access to skeletal hand data, so you may have to implement some creative Editor-only ways to simulate this. For example, in the GenieHighFiver, we spawn a dummy object that represents the user's hand position.

**XR Simulation Environment**

Unity offers an easy way to test things that rely on spatial meshes and planes right in the Editor. This should be on by default, but here's how you can activate it in Project Settings > XR Plug-In Management: in the Standalone tab, make sure 'XR Simulation' is ticked and 'PolySpatial XR' is off.



XR Plug-in Management

Initialize XR on Startup ✓

Editor Play mode uses Desktop Platform Settings regardless of Active Build Target.

Plug-in Providers

- [ ] Oculus
- [ ] Open XR
- [ ] PolySpatial XR
- [x] XR Simulation
- [ ] Unity Mock HMD

Information about configuration, tracking and migration can be found below.
View Documentation

These environments are actually prefabs that get automatically loaded into your scene at runtime. You can make your own simulation environments as well, which, combined with our Spatial Mesh Capturer tool, can be very useful.

For more information on XR Simulation Environments, refer to the Unity documentation.

When using XR Simulation Environments, there are some limitations, however:

- AR Plane classification doesn't work, so in Editor, we might just assume any flat plane above a certain height is a table, etc.
  - Also, remember that AR Planes only spawn when you're looking at objects pretty close and moving around.
- Spatial meshes are extremely clean, which is not very similar to what you would see in real-world conditions.
- Spatial meshes only update once, right at the start.

## 5.2 Device-Centric Development

It's easy to get too comfortable developing in the Editor, which is why we employ device-centric development frequently. This is critical in any development process, but particularly so in XR, where the user interface and experience is so wildly different from what's happening on your workstation. And it's *especially* important for development with PolySpatial, whose support for RealityKit is quite incomplete. This list of supported and unsupported Unity features illustrates this.

Building to device works similarly to how it does for Unity iOS development. Building from Unity creates an XCode project, which you can then use to deploy to a device or the simulator. You can rely on XCode's log output to catch logs from Unity.

## 5.3 Play-to-Device

When Play-to-Device is enabled, pressing 'Play' in the Editor will trigger Unity to make a connection with your Apple Vision Pro on the same network. This allows for rapid development in the XR environment, giving you access in the Editor to live spatial meshes from your environment and skeletal hand tracking. What's more, the images you see in the device will reflect what you would see in a build, which is perfect for testing shaders and other visual things that might look different on the device vs. in the Editor.

**We find Play-to-Device is especially useful when:**

- We're developing features that are heavily reliant on hand-tracking or gesture recognition
- We're developing features that are heavily reliant on live spatial meshes or AR planes
- We're making adjustments to size, layout, etc. to optimize the user experience
- We're debugging a problem that only seems to happen on-device

**As helpful as Play-to-Device is, we've found it to have some limitations**:

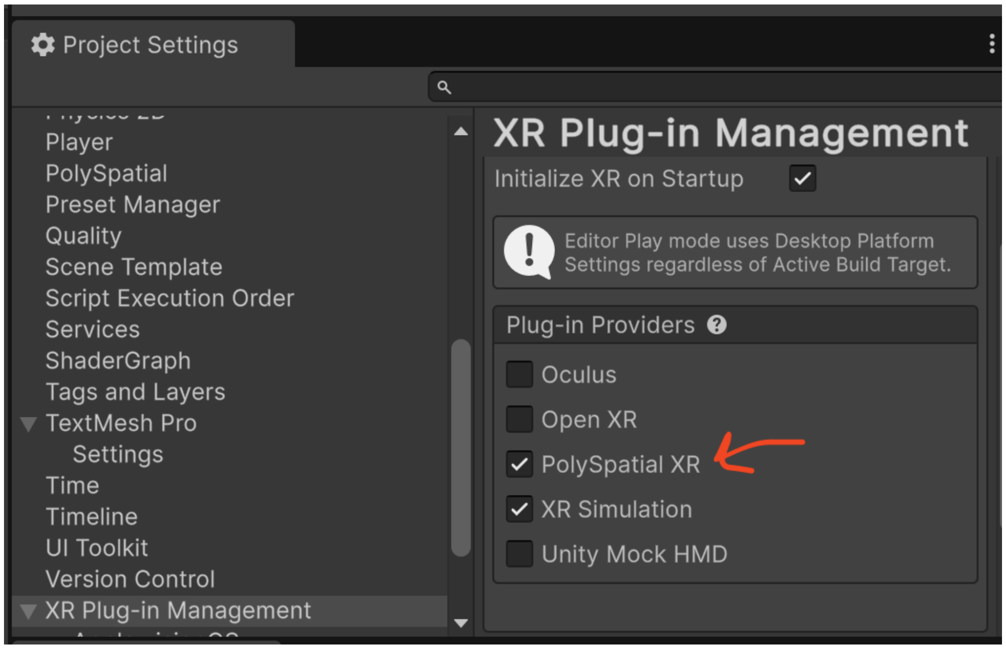
- There seems to be a hard limit of data that is allowed to be sent over, and if you go above this, it will error out right away. To mitigate this problem, we'll often test things in an isolated scene with minimal asset references.
- After a few moments in Play Mode, spatial meshes may stop updating.
- You often must force-quit the Play-to-Device Host app running on your Apple Vision Pro.

- The overall experience can be buggy, especially if your network connection isn't solid.

We're hoping that over time, these problems will be fixed. But for now, it can still be quite a useful feature.

**Enabling Play-to-Device**

To enable Play-to-Device, navigate to Window > PolySpatial > Play to Device. In the Play-to-Device window, enter your device's information and enable 'Connect on Play.' This will trigger a Genies IRL popup notifying you that PolySpatial XR has automatically been enabled in Standalone in the XR Plug-in Management settings.



XR Plug-in Management

Initialize XR on Startup ✓

Editor Play mode uses Desktop Platform Settings regardless of Active Build Target.

Plug-in Providers

- [ ] Oculus
- [ ] Open XR
- [✓] PolySpatial XR
- [✓] XR Simulation
- [ ] Unity Mock HMD

**It's automatic – you don't have to tick this box yourself!** It's just good to know what it's doing.

When you hit Play, the Editor will try to connect your device, **ignoring XR Simulation functionality**. So when you're done remember to disable "Connect on Play." PolySpatial XR will be unticked again and XR Simulation will function normally.

NOTE: If you're working in version control, avoid pushing a change with PolySpatial XR turned on, otherwise your team members might become confused next time they pull and try to use XR Simulation!

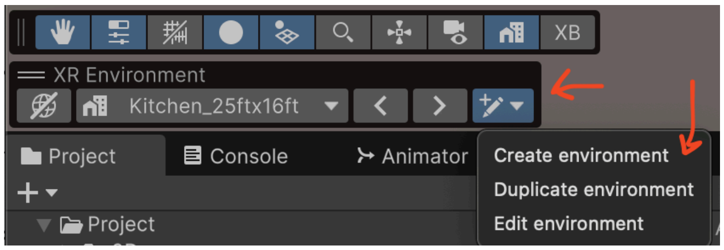
**Spatial Mesh Capturer**

Sometimes, it's helpful to capture spatial meshes from your real-world environment to rapidly iterate and debug. For example, let's say there's an area of your home that your character has a difficult time navigating. By capturing the spatial mesh and importing it into your project, you can reproduce and debug the problem straight from the Editor and without Play-to-Device.

Here's how:

1. Open the EmptyXRScene in Assets/Project/Scenes. This scene contains only our XRNode and is small enough to work with Play-to-Device.
2. Locate the ARMeshManager component in your scene and ensure there is a SpatialMeshCapturer component attached to the same GameObject. (In Genies IRL, our XRNode prefab already has this component.)
3. Note the output path of the SpatialMeshCapturer component and modify it if necessary.
4. Enable Play-to-Device and scan your environment. Remember that in Play-to-Device, spatial meshes may inexplicably stop capturing, so don't take too long before capturing what you need.
5. When you're ready to capture, navigate to the SpatialMeshCapturer component and press the "Capture" button.
6. Look inside your Project View and you'll see your mesh.

You can then use the XR Simulation environment tools to create a new XR Simulation environment out of this mesh.



Once created, you can set it as the current environment and test in the Editor, without Play-to-Device enabled at all!

NOTE: Make sure that the floor of your custom environment is aligned to the y=0 position of your prefab. The way the Floor Manager is written, floor detection is disabled in the Editor, because it relies on floor mesh classifications (which only works on-Device).

NOTE: Simulating with captured spatial meshes is, geometrically, nearly identical as it would be on device, but there is one important difference to keep in mind: On device, there may be many spatial meshes, but the Spatial Mesh Capturer combines them all into one. So beware of coding with the assumption there is only one spatial mesh at a time!

# 6. Project Architecture

This project is intended as an example that you can freely use, modify and build off from as you please. You don't have to use all our designs or methodologies, but they worked for us, and we hope you'll find them inspiring in your own work!

## 6.1 Scene Structure

In the Main scene, you'll see a Directional Light, an Event System, a GeniesIrlBootstrapper, and an object called "Test Objects" which can make it easier to test certain things in the Editor (for example, fake seats and windows).

The Genies IRL Bootstrapper is responsible for kicking everything off, spawning many of the managers and systems needed to run the experience. By controlling the initialization of our managers in this way we're less reliant on Awake(), Start, and OnEnable() and therefore less prone to initialization race conditions. It also provides a way for objects to get access to each other without having to find components by name or type, or by having to manage too many serialized references.

## 6.2 Core Managers

1. **Genies IRL Bootstrapper**: Spawns all other core managers. Also makes sure that the user accepts hand-tracking and world-sensing permissions, and if they fail to, it spawns a UI to guide them on how to remedy it.
    a. Parented to this prefab are two GameObjects that, for some reason, must be in the scene at launch. One is the high-five particle system, and the other is the sheep plushie. Otherwise, the particle system will fail to play

the first time when spawned, and the sheep plushie's mesh collider will crash the application. We think this is caused by an issue with PolySpatial spawning objects in RealityKit.

2. **XRNode:** Not to be confused with a Unity XR enumeration by the same name, our XRNode contains core XR systems relating to the camera, head and hand tracking, AR Plane and spatial mesh detection, floor height detection and management, gesture recognition, spatial pinch interactors, and a ton more. The idea here is that you can simply add this to a scene and have everything you'd need to make it a fully-fledged XR scene.
3. **GenieManager: Responsible for finding an optimal place to spawn a Genie, and then spawning them there.**
4. **UI Manager: High-level management of things like the Splash Screen, Tutorial, MainMenu, and more.**
5. **AR Navigation: Core system that uses the live spatial mesh to create and update a navigation grid. Spatial meshes tend to be extremely noisy and difficult to work with, and we weren't able to get Unity's mesh-based system to give us reliably good results. So far, we've gotten our best results assuming a flat ground, and by using a grid-based pathfinding approach. We do a bit of extra processing too, such as throwing out points that are too high off the floor. Genies can then access this grid to process navigation.**
6. **Launch UX: Handles UX flow from the Splash Screen through to spawning the Main Menu.**
7. **AR Surface Understanding: Uses AR Planes and spatial mesh data to provide information to the Genie about important surfaces, such as seats, walls, windows, ceilings, and tables.**
8. **Auto Item Spawner: A simple script that populates the world with sheep plushie items, which the user and the Genie can grab, place, and toss.**
9. **XR Image Tracking Object Manager: Used to track "fake" windows, in case the user doesn't have access to a real window in their development environment, or if the device just has a hard time detecting any of their windows. These targets are located in Assets/Projects/Textures/AR Image Targets and can be printed out on an 8×11 paper or displayed on your**

**monitor to simulate a window at runtime. Just make sure you get really close to it; if you see a small grey arrow pointing out of it, that means it's locked on and tracked. You can only have one of each target in your environment.**

10. **PermissionsRequiredWarning: GeniesIRLBootstrapper spawns this if the user denies hand-tracking or world-sensing permissions on launch.**

## 6.3 Genie

At Genies, we use the word "Genie" to colloquially refer to a Genies avatar, but the word has invaded much of our work, including our codebase in IRL!

The root Genie prefab in Genies IRL is the "EmptyGenieBase", which includes most of the core functionality of a Genie, but is missing a character model and rig. Sticker_Gen13 is a prefab that derives from EmptyGenieBase, but contains a model and rig along with some additional configurations.

1. **Components on Genie Root GameObject**
    a. **Genie**: Contains Core Genie Functionalities
        i. **GenieLookAndYaw**: Allows the Genie to turn and look at objects.
        ii. **GenieNavigation**: Uses the pathfinding system to navigate around the world.
        iii. **GenieAnimation**: Interfaces with the Animation Controller on the model and rig to perform various functions.
        iv. **GenieSense**: Senses occurrences in the environment, like object impacts and user gestures, to inform the GenieBrain.
        v. **GenieBrain**: Uses information about the environment to prioritize goals and execute actions.
        vi. **GenieSitAndStand**: Leverages the character animation and pathfinding systems to sit and stand up from seats.
        vii. **GenieHighFiver**: Leverages the character animation and IK systems to exchange high-fives with the user.
        viii. **GenieDraw**: Uses the character animation system and ARSurfaceUnderstanding to draw hearts on walls.

    ix. **GenieOfferItem**: Uses the animation, IK, and interaction systems to offer sheep plushies to the user.

    x. **GenieAudio**: Plays sounds to accompany animations.

   b. **Seeker, AIPath, and Simple Smooth Modifier**: Critical pathfinding Components. We subclass AIPath into our own custom subclass to ensure support for "partial" paths – that is, paths that allow for the Seeker to get as close as they can to the target, even if the target itself is not accessible.

   c. **Capsule Collider**: Detects collisions with thrown items.

   d. **Behavior Agent: Accesses the Behavior Graph used by GenieBrain.**

2. **Components on Genie Rig GameObject**

   a. **Animator:** Contains all character Animations. The current setup is not very scalable, but it's simple enough to manage for now.

   b. **BlendShapeAnimatorBehaviour**: Maps certain AnimationController parameters to facial blendshapes, to support blendshape animation. Note that in its current implementation, Genie facial animations will only play in Play Mode.

   c. **GenieAnimEventDispatcher: Detects Animation Events from the Animation Controller and dispatches them so they can be used by the Genie component.**

   d. **Genies IK Component: Leverages the Unity Animation Rigging package to facilitate arm and body reaching.**

   e. **EyeballAimer: Allows the Genie to train its eyeballs on specific targets using facial blendshapes.**

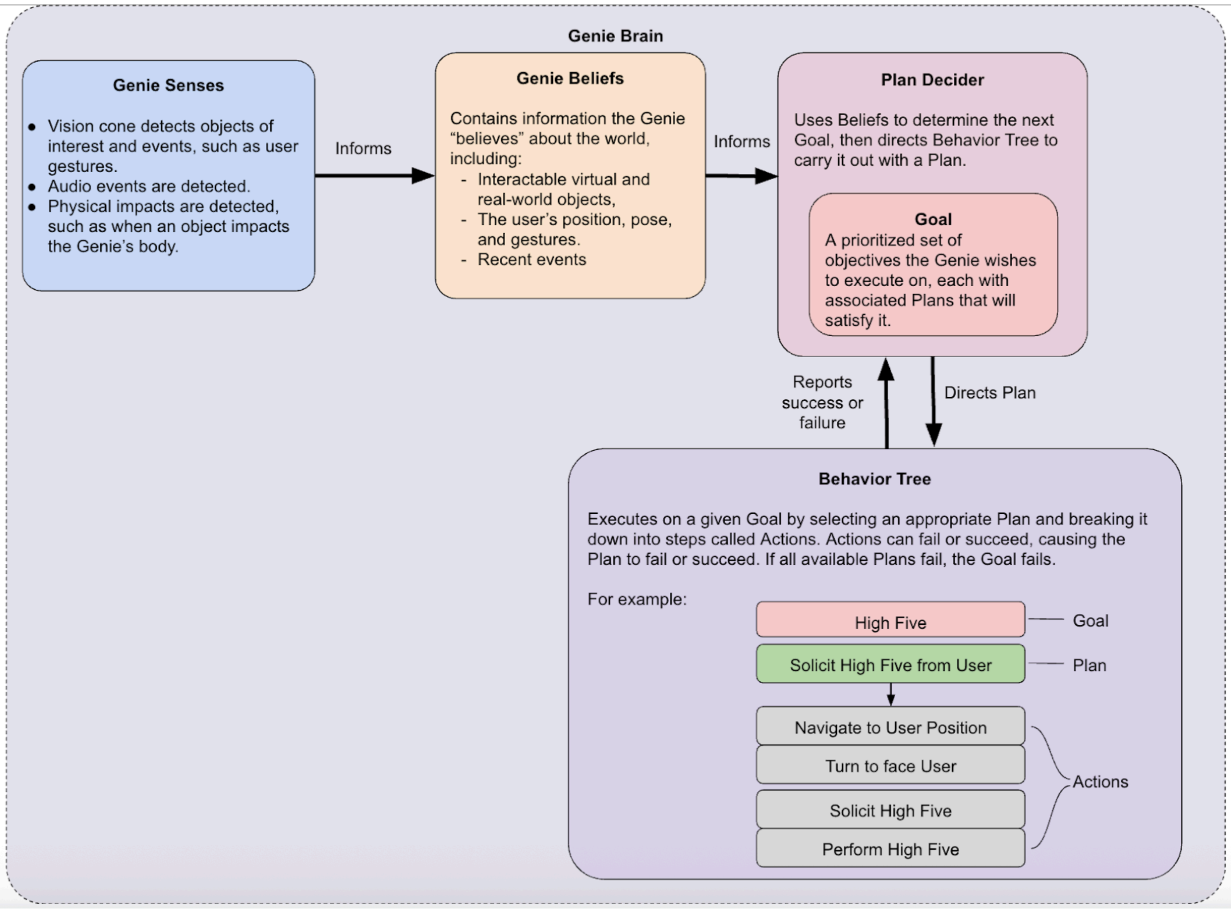
### Genie Brain

The Genie's brain is inspired by Goal Oriented Action Planning, or GOAP, which is an approach for NPC behavior design in games. While it doesn't strictly follow every aspect of the technique, our version borrows key concepts while not over-engineering things for our needs here. Your project might use pure GOAP, or be completely behavior-tree driven, or perhaps the whole thing will be governed by a large language model instead!

Our design for the Genie Brain aims to:

1. Make the Genie feel spontaneous, yet intelligent.
2. Gracefully deal with failure caused by unexpected changes or corrections in the spatial environment.
3. Gracefully deal with unexpected interruptions, such as if the player tosses an item at them.
4. Provide developers a way to easily add new behaviors.

The way it generally works is that a Genie's **Senses** inform their **Beliefs**, which in turn inform the **Plan Decider**. The **Plan Decider** manages a set of **Goals** and picks the highest-priority Goal, along with a **Plan** that will accomplish this Goal. The **Behavior Tree** (powered by Unity's Behavior Graph) allows us to break the Plan down into reusable steps called **Actions**. Plan failure and success is reported back to the Plan Decider, who uses that result to decide the next course of action.


Genie Brain
Genie Senses
Vision cone detects objects of interest and events, such as user gestures.
Audio events are detected.
Physical impacts are detected, such as when an object impacts the Genie's body.
Informs
Genie Beliefs
Contains information the Genie "believes" about the world, including:
- Interactable virtual and real-world objects,
- The user's position, pose, and gestures.
- Recent events
Informs
Plan Decider
Uses Beliefs to determine the next Goal, then directs Behavior Tree to carry it out with a Plan.
Goal
A prioritized set of objectives the Genie wishes to execute on, each with associated Plans that will satisfy it.
Reports success or failure
Directs Plan
Behavior Tree
Executes on a given Goal by selecting an appropriate Plan and breaking it down into steps called Actions. Actions can fail or succeed, causing the Plan to fail or succeed. If all available Plans fail, the Goal fails.
For example:
High Five
Goal
Solicit High Five from User
Plan
Navigate to User Position
Turn to face User
Solicit High Five
Perform High Five
Actions

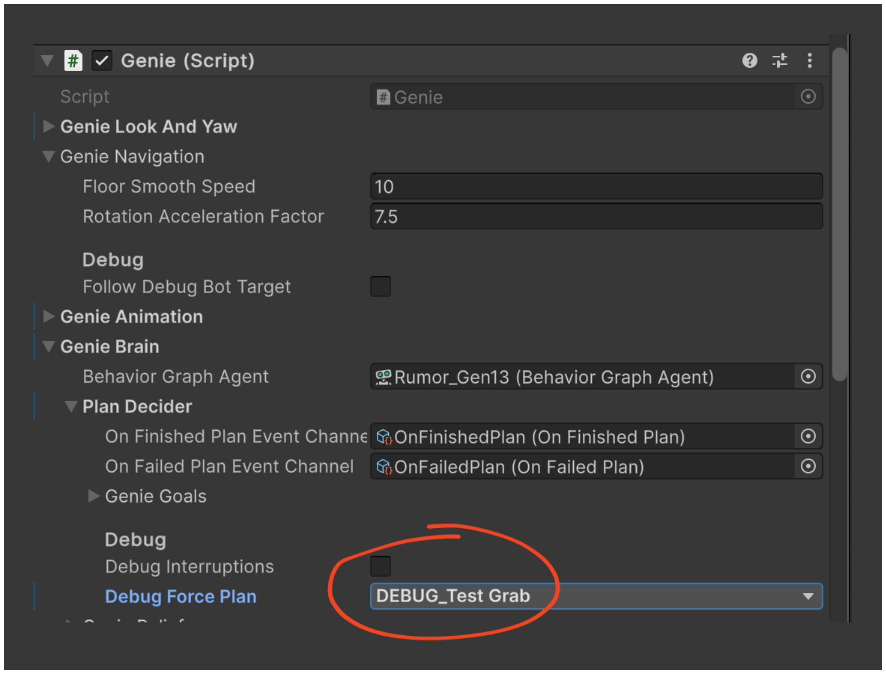
Tip: To test specific Goal, the easiest way is often to go into the Plan Decider and mute all of the Goals except the one you want to test.

## 7 Inverse Kinematics

The avatar in Genies IRL uses a component called GeniesIKComponent, which is attached to the rig/model in the prefab. This component uses Unity's Animation Rigging package to engage the character's arms, shoulders, and spine to reach objects. Additionally, we utilize the Animator Controller to blend between different "crouching" heights, allowing the avatar to grab objects on low surfaces like coffee tables, or on the floor.

### 7.1 Testing Reach IK

The easiest way to test the IK system is to engage a special "Debug" Plan in the Genie's brain. To do this, locate the Genie Brain section in the character prefab, and set DebugForcePlan to Debug_TestGrab:



Genie (Script)
Script: Genie
Genie Look And Yaw
Genie Navigation
Floor Smooth Speed: 10
Rotation Acceleration Factor: 7.5
Debug
Follow Debug Bot Target
Genie Animation
Genie Brain
Behavior Graph Agent: Rumor_Gen13 (Behavior Graph Agent)
Plan Decider
On Finished Plan Event Channe: OnFinishedPlan (On Finished Plan)
On Failed Plan Event Channel: OnFailedPlan (On Failed Plan)
Genie Goals
Debug
Debug Interruptions
Debug Force Plan: DEBUG_Test Grab

This corresponds to an Plan and Action in the Behavior Tree. If you set this, the Genie will do nothing aside from what this plan dictates. The code for the Action is located in DebugTestGrabAction.

Next, ensure you have an object in your scene called “TestReachTarget” (the name must be exactly that).

In play mode, the Genie will reach towards the object when you press the “G” key. You can use this to test your character’s ability to reach towards targets at various positions and heights.

# 8 Known Issues

## 8.1 Player framerate performance degrades when Spatial and Plane Meshes are updated

This is an issue we discovered brought to Unity’s attention, which was confirmed by their team is now on the Issue Tracker: https://issuetracker.unity3d.com/issues/player-framerate-performance-degrades-when-spatial-and-plane-meshes-are-updated

Each time spatial meshes or AR planes are updated, the framerate sharply drops. In Genies IRL, this happens every couple of seconds, so it can be very noticeable, especially when you are trying to toss a virtual item, as it seems to interrupt hand tracking.

If you disable Spatial Mesh and AR Plane updates, the problem seems to go away, but then the character can’t get vital environment updates or travel with the user.

We expect Unity to fix the issue in the next PolySpatial update.

## 8.2 Apparent Z-Fighting Between Character Submeshes

Sometimes our character’s eyebrows, hair, or clothes might seem to flicker as though Z-fighting with the skin. We suspect it actually might have something to do with the way RealityKit processes skinning of multiple meshes by one skeleton and Animator – sometimes it seems to perform the skinning in a manner that is not “in sync” with the other meshes, causing meshes to clash with each other a little. Hopefully this will be fixed in future versions of PolySpatial, but if you can find a workaround, we’d love to hear it!

## 8.3 Sometimes Gesture Tracking Fails

We’ve noticed a rare bug where the device just silently stops tracking hand gestures with no errors. Typically the first thing that happens is that we’ll try to summon the Main Menu with a thumb’s up gesture, but nothing happens. To fix this, the app must be restarted.

We think it’s due to some internal tracking mechanism, of XRI or visionOS, but we’re not sure because it happens so rarely that we haven’t been able to investigate it!

### 8.4 Unexpected High-Five IK Solving

Everyone approaches high-fives differently, and some tuning is required to make this action more robust so that strange-looking IK situations don’t happen as often.

In Genies IRL, the best way to perform a high five with the Genie is to place your hand out in front and mostly let her hand come to you.

### 8.5 Unexpected Teleporting

If the application thinks the Genie is separated from the user (i.e. they are on an “island”), the Genie will teleport in front of the user. This can happen at unexpected times, however, and could use some tuning to be less jarring.

### 8.6 Stuck on “Searching for Walkable Ground”

We’ve found that taking the headset off, walking to a totally new location, and putting it back on can sometimes result an infinite “Searching for Walkable Ground” UI to pop up. The app must be restarted to fix this.

### 8.7 Celebratory Balloons Fall Through Ground

If you start on one floor of your house, then walk upstairs, then start the celebration, the balloons will fall through the second floor and onto the first floor of the house.