Adjacency

Overview

Adjacency is a multiplayer social Virtual Reality (VR) game, constructed entirely from User-Generated Content (UGC). The primary goal of Adjacency is to provide players with a powerful yet approachable set of creation tools for players to utilize in a completely shared social space. It is designed to operate on both tethered (PC) VR as well as on standalone. Adjacency is designed as an entirely standalone program – no external tooling or "studio" application should ever be required.

Challenges

Adjacency has several major technical hurdles that must be overcome:

- Multiplayer is fundamentally difficult to implement for any game.
- All tooling must be usable and performant on standalone platforms.
- Any object in Adjacency could become dynamic at any time.

Multiplayer

Adjacency is, by design, a multiplayer game with singleplayer as a convenient side effect. It is based around a hybrid client-server architecture – secure operations like use of the creation tools and spawning objects are mediated by the server, but players retain complete authority over their own position and rotation. While this is a compromise, it enables smooth, low-latency movement for players while still ensuring security for important operations.

Adjacency uses a shared code model for networking, supported by Unity Multiplayer. This means that any given instance of Adjacency can serve as either a client, a server, or both (referred to as a "host"). This means that any player, including those on standalone, can host their own private instance.

First-class Standalone VR

Because many VR games are built first for tethered VR and subsequently ported to standalone headsets, many games have functionality on tethered VR that is not available or not performant on standalone VR. Examples include creating Unity's Asset Bundles, the ProBuilder package, or the use of blend shapes & rigging. Adjacency is built from the ground up to support both tethered and standalone VR equally, with no platform-specific features or external SDKs.

UGC Optimization

Because the entire world of Adjacency is 100% player-created and inherently dynamic, traditional methods of optimization often do not apply. Hand-optimized Level of Detail (LOD) meshes, pre-baked occlusion culling, static lighting, and other precomputation techniques are all luxuries afforded to conventional games that are not available to Adjacency.

One part of Adjacency's solution to this issue is that of chunking, which also serves as a core gameplay mechanic. In Adjacency, the entire world is split up into a practically infinite grid of 10-meter by 10-meter, infinitely tall "chunks." These chunks are then loaded and unloaded on the server as they are needed, and are then further culled by the server to only transmit data for chunks that clients can actively see. The gameplay aspect of this is also quite useful – chunks can be "claimed" or purchased by players, granting them rights to modify the geometry of those chunks. In addition, chunks can be reserved by the "system" user (representing the server itself) to allow for content authored by the server owner rather than by players.

Design

The game design of Adjacency has been shaped by many requirements – technical, social, or simply due to time constraints.

Distance

As stated, Adjacency takes place on an infinite grid of player-owned and player-created chunks. As a result, the entire world effectively serves as one large social space. For this reason, players are not provided with a way to rapidly move between distant chunks instantly (e.g teleportation or portals). While this can feel like an inconvenience to players, it forces them to traverse the distance naturally or by in-game transit systems, which encourages interaction and exploration by players on the way to their destination.

User Interface

Adjacency inherits much of its User Interface (UI) guidelines from previous work. Tried, tested, and efficient, 3D buttons and tactile feedback form core components of Adjacency's UI stylebook. Large font sizes and a high-contrast color palette permit both accessibility and simplicity of design.

A planned addition to the UGC system in Adjacency is a radial menu activated by the Secondary Button (A / X on Meta Quest, for example). This will permit fast and intuitive switching between various editing contexts or tools. It also allows for the introduction of shortcuts for common actions (undo, redo, stop editing, etc) in the future.

Server Configuration

All server configuration in Adjacency can be modified at runtime in-game. This permits users on standalone platforms to easily host local instances without the necessity of an external configuration editor. It also simplifies administration of remote instances, particularly when only a small change to configuration is needed. To clarify this process to administrators, we use various spatial metaphors, such as turning a key to unlock a given feature.

Development

Adjacency is currently under active development. The initial prototype of Adjacency was created in 17 days (between 13 April 2025 and 30 April 2025) entirely by myself, with over 230 commits being made in that timeframe.

Timeline

Below is an abridged timeline of Adjacency's development thus far.

14 April 2025:
Initial commit to Git version control (project start).

15 April 2025:
Multiplayer first becomes functional.

18 April 2025:
Authentication and authorization system implemented.

20 April 2025:
Chunking system prototype implemented.

22 April 2025:
Grabbable objects implemented.

23 April 2025:
Chunk saving implemented, begin work on UGC.

24 April 2025:
MeshBuilder mesh generation system implemented.

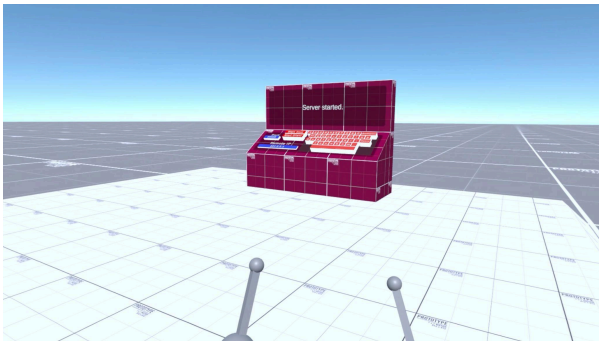
28 April 2025:
Drawing of primitive shapes implemented.

30 April 2025:
Textures, colors, and colliders implemented.
Minimum Viable Prototype complete.

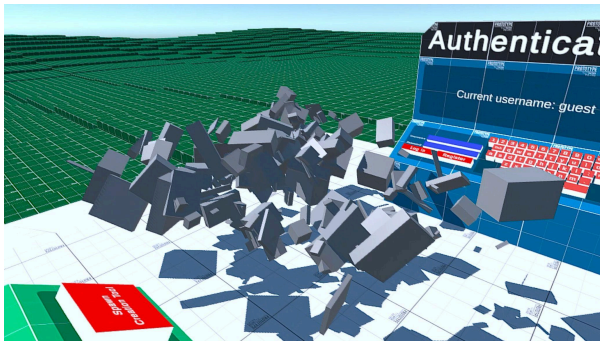
1 May 2025:
Flight locomotion implemented.
In-depth "dogfooding" test begins.

Gallery

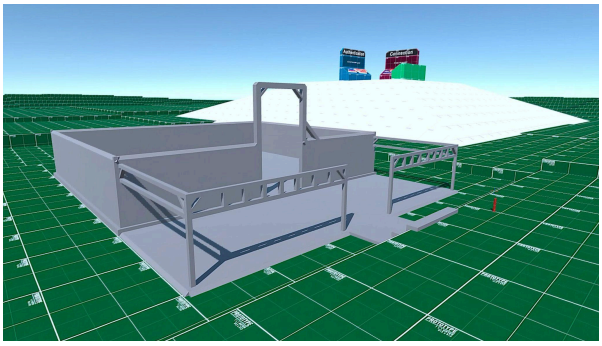
Attached are screenshots from various points in time during the development of Adjacency. All screenshots were taken over Meta Quest Link on a Meta Quest 3S while Adjacency was running in the Unity editor. All content in screenshots is under active development and subject to change.



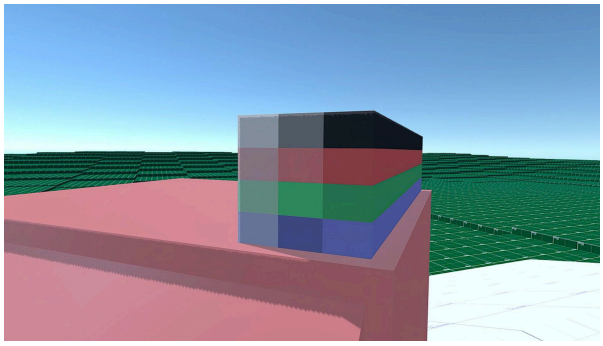
First-ever screenshot - UI test



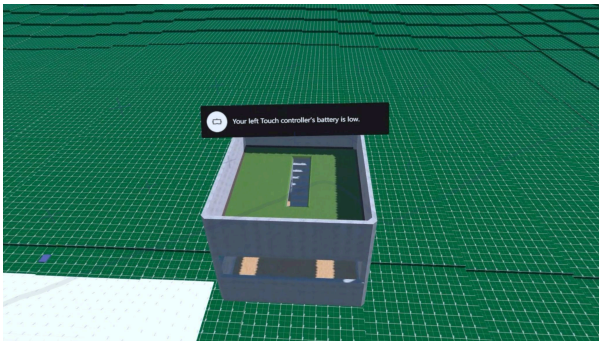
Stress-testing the MeshBuilder API



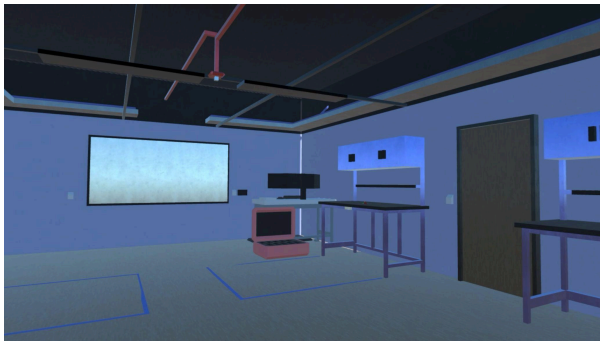
Position & angle snapping demo



First test of tinting shapes



Flight locomotion & collider testing



Work-in-progress VR Lab recreation