3D Navigation Multi-User Template

AKA *LikeLike 3D Template User Guide*, *v1*
by Hand Eye Society (Brendan Lehman)

for Mackenzie Art Gallery – Digital Exhibition Toolkit and Art Installation Launcher

Table of Contents

Acknowledgements

Hand Eye Society

- Brendan Lehman – Guide, Template (Client), Super FESTival
- Victoria Kucher – Super FESTival
- Rokashi – Super FESTival
- Jordan Sparks – Super FESTival

MacKenzie Art Gallery

- Cat Bluemke – Digital Exhibitions Consultant
- Jonathan Carroll – Digital Exhibitions Consultant, Code Contributions (Client)

LIKELIKE

- Paolo Pedercini – LIKELIKE Online, Code Author (Client, Server)

Introduction

Hello! Welcome to the LikeLike 3D User Guide! This document will help you through some of the many ways to use the LikeLike 3D platform to host live multiplayer events and shows online. It is written to get your event up, running, and ready for showtime.

This guide assumes some knowledge of the game engine Unity and working with files and websites. We'd give it an intermediate difficulty rating. It's not totally ridiculous to get through the basics, and would make a nice challenge for a beginner to these kinds of things. The only real difficulty emerges in using Unity to customize it to your liking. You can do anything you can imagine within the game engine. Check out the project files to gauge the complexity for yourself at `likelike-3d-template-client`.

As we mentioned, the event space is built using the game engine Unity. This guide is not written as an introduction to the game engine. The guide will get you through to the event space running, even if you just follow along blindly, but it is beyond the scope of this guide to teach you how to use Unity for further customization and development. There are so many resources across the web for solving problems in and with Unity. Search your ideas and you'll be surprised!

The platform backend relies on a website called Glitch. The guide references the Glitch project `likelike-3d-template-server`. The server essentially keeps track of the data shared between players (clients), like player locations and objects everyone can see.

The networking bits we're using in our Unity project, and the server code, is written by Paolo Pedercini. Paolo calls this the Distributed Authority Framework. We'll reference these later too but the GitHub repositories for the client and server are located at these links. The client template includes features and interactions that we added in. Jonathan Carroll helped with these. Jordan Sparks, Aaron Demeter, and Len Predko contributed 3D art.

I Setup

1 Overview

In this section we'll go over getting set up to work on the project.

We'll get the server going on Glitch and the Unity project set up to explore.

2 Glitch

Head over to Glitch and make an account or login using your sign–on method of choice.

Once you're logged in, go to the template server project.

Glitch's term for duplicating or cloning a project is 'Remix'. Click the Remix button on the template page to make your own to work with.

Once loaded, the first thing to do is to change the project name. Go to Settings > Edit project details and enter a new name under PROJECT NAME. Click Save.

The Glitch edit interface allows everything you need to do to work on the project, including a code editor, terminal, version control, and asset handling. Lucky for us, we don't need to touch this part other than to make sure it's running.

3 Unity

Installing Unity

For this project, there are specific Unity versions we can use because of our use of WebGL. Building WebGL projects is only officially supported on recent versions 2021.3 and 2022.3.

We recommend working with the Unity Hub. It makes things a lot easier to manage. You can get it here if you don't have it. Unity's documentation is here if you'd like more detail about how to use it. Work your way through making a Unity account if you don't have one and the Hub will help you to get a version installed.

Install the latest 2022.3. When installing, make sure to select 'WebGL Build Support' in the 'Platforms' list of the 'Add Modules' section.

We made this guide with 2022.3.49f1 (September 2024) and it'll be best if you'll be working on this into the future as 2022.3 is the current long–term–support (LTS) release. Into the far future, the new Unity 6 will continue to support WebGL builds but we'll probably need to rewrite this guide for it, so stick to the "classics" for now.

Getting The Template

To get our template client project, head over to our Bitbucket repository page.

Git

From here, and if you're familiar with Git, you can get the command to clone the project at the button at the top right. Clone it into your development folder of choice. After cloning you can add the project to the Unity Hub in the 'Projects' section by clicking "Add" in the top right and selecting the folder you just cloned.

Not Git

If you're not familiar with Git you can navigate to the 'Downloads' page at the bottom of the menu on the left side of the page. Click "Download repository' in the table and save the .zip to a development folder of choice on your computer. Extract the contents of the .zip. It should extract into a top–level folder with the same name as the repository, but if it doesn't make sure it ends up that way.

After extracting you can add the project to the Unity Hub in the 'Projects' section by clicking "Add" in the top right and selecting the folder you just extracted.

Open the project up and have a look around. We'll be importing most of these files into a new project for your event but click around the files for a minute and get a sense of what we're dealing with.

Creating The Project

There are two ways to proceed from here. First is the hack-y way where you can change the name of the template project and continue your own work off of it. But for the sake of learning and the freedom of a blank canvas, we recommend creating a new project and importing the important parts from the template project.

In the Unity Hub, start a new Project. Choose the '3D (Built-in Render Pipeline Option). This will keep building for the web nice and simple. Give the project a name and choose your preferred location on your computer. We kept the name of our project consistent with the name of the server project. For example, the server is `likelike-3d-template-server` so we called ours `likelike-3d-template-client`. Call yours something appropriate for your event.

Setting Up

Once our project loads up, there are a few changes we need to make before we should start working.

First, we need to change the build platform. Click 'File > Build Settings...' to open the Build Settings menu. There, select 'WebGL' on the left menu.

A ⚠ warning appears that we can quickly deal with. Select the 'Player Settings...' button at the bottom left of the Build Settings menu. In the Player Settings, is another ⚠. There, unselect 'Auto Graphics API'. Everything should be happy now. Close the Player Settings menu.

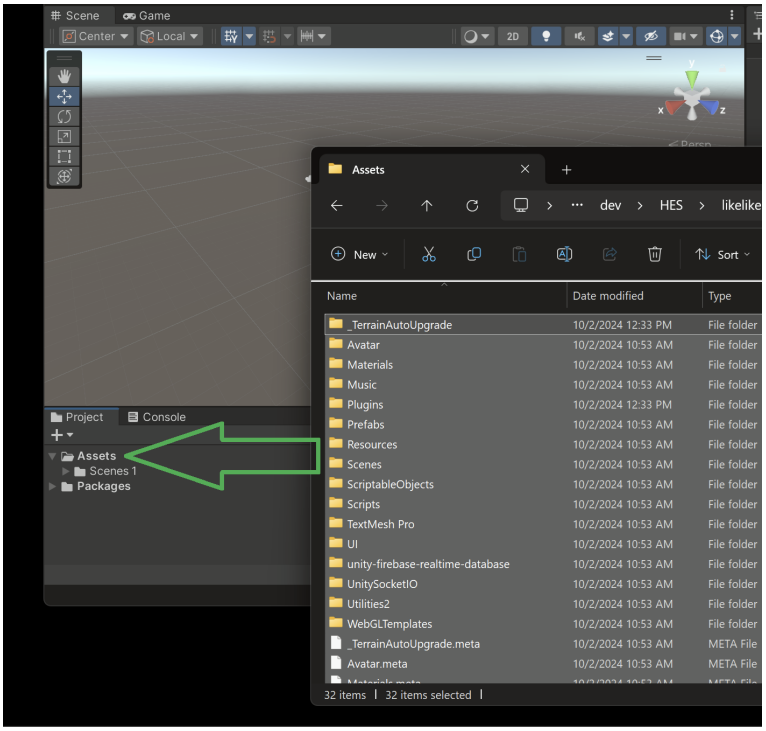
Back in the Build Settings menu, select the 'Switch Platform' button. Wait for it to do things and close the menu.

Importing Assets

Now we're ready to get the important parts in. The files are located in the template project we set up earlier. Open it up in an Explorer/Finder window. We want everything in the 'Assets' folder.

Back in Unity, we'll make sure our Project Folder tab is open to the empty Assets folder. Rename the 'Scenes' folder that's there to something else like 'Scenes 1'.

In our Explorer/Finder window, select everything in the 'Assets' folder and drag into the Assets folder in Unity.



After Unity takes its time to think, there may be some warnings in the console we can clear out with no problem.

In the 'Scenes' folder, open the 'Client' scene. Delete the 'Scenes 1' folder that we renamed just before importing.

Finally, we need to hook it up to our Glitch server. In the Scene Hierarchy, there will be a GameObject called 'Main'. This contains all the networking stuff. In the Inspector, the top Component called 'Socket IO Controller' has a 'Settings' dropdown. In there we can add the URL for our Glitch project **without the "https://"**. We can set the port to 0 and select 'SSL Enabled'.

There it is! Basic setup complete :) There are still some important things to do before we can build our project for going live but we'll get to that later.

Now you're ready to customize! Check out Section II for an overview of what all the files written for us are, Section III to learn about what the files do, or skip straight to Section IV to start working.

4 Website

Despite having the infinite possibilities of Unity now available to us, an important thing we should set up for ourselves before we get too into it is a place to put the game when it's ready to share.

The best way to do this is to set up a page on a website. On your website, you can upload the built project to a place that's either public, or hidden so you can embed it in other things. We recommend the latter. This way you don't have to fiddle with customizing this particular page, but can easily embed it with `<iframe>` into other pages already styled on your website.

For example, we might set up the project at www.handeyesociety.com/projects/3d–game (not a real site) that we won't share directly, and then embed that into the page www.handeyesociety.com/festival that is already dressed up.

We'll go over the details of embedding with an `<iframe>` in Section V.

FTP Access

If you have a website, you can set these pages up by getting FTP access to your hosting server. If you have a cPanel you can log into, you can find the FTP information there. We recommend logging in to your hosting server with FileZilla.

II File Structure

1 Overview

This project is set up in the standard client-server method. Server files are in the cloud on our Glitch project and client files are in the Unity project.

There is a more-than-sufficient description of the inner workings of this project on Paolo's Github so we won't over-complicate things here too much. We recommend you read it through even if it doesn't make much sense.

2 Server Files

Server files contain the code that run on Glitch and are shared among each user of the app. This code keeps track of where everyone is, what and where objects created are, and what folks say in the chat. It uses a simple database in JSON format to pass the data back and forth with the clients and keep itself up to date.

server.js

This file is the core of the project. It determines the behaviour of the space and the shared features of the platform. It loads all the code packages the project uses, in particular the socket.io package that handles communication with clients/users. Big changes to this file are generally not needed as it works with whatever JSON data the clients send at it.

3 Client Files

Client files contain the code that runs in the browser of each player, communicates with the server, and determines what appears on–screen. They work inside Unity to update the world based on the data that comes in from the server.

Describing the client networking files (in 'Scripts > Networking') in detail is outside the scope of this guide. Please read Paolo's documentation if you are curious about the concepts of networking and how they are implemented here.

The 'Scripts' folder contains a few other folders with the workings of our other features. We'll describe the features in the next section, but for now you can get oriented to them and see what's driving them.

The 'Art' folder contains the code for the user art sharing feature. These files handle getting an image (by opening a dialog box to ask for a file on a key press), processing the image, putting the processed image data into a database, and retrieving the image data from the database to display. These scripts are called in the avatar prefab (AvatarPrefab, located in the 'Resources' folder) that gets created when a player logs in.

The 'Character' folder contains code for the player character, notably movement and animation. It also contains code for actions the player can take to manipulate objects in the environment. They are also called in AvatarPrefab.

`FirstPersonController.cs` handles the movement of the player.

`FirstPersonInteraction.cs` handles the interactions of the player with objects in the space other than the art–share feature.

`UIController.cs` handles the visibility of the chat and the info panel.

The 'Objects' folder contains code for certain objects in the environment to have special behaviour. They're placed on the prefabs of specific objects (the Ball in the template for example).

The 'UI' folder contains code to manage the character select menu and we won't have to touch this for the purposes of this guide. You can play with it if you're interested in adding to the menu that appears before connecting to the shared event.

4 Assets

Working with media assets is a critical part of customizing the experience to your own design. Fortunately for us there is an unreasonable amount of content on the internet and IRL to teach you how to make game–ready digital assets and how to use them in Unity. We'll leave the creativity up to you!

For this project in particular however, we recommend searching for and following best practices for web–based graphics. Browsers can only handle so much and there are ways to choose and make assets that help keep your computing ask nice and low.

The template contains some assets we made for Super FESTival 2023, other stuff made by HES staff, and the characters made by Paolo. This kind of stuff can mostly be found in the 'Resources' folder, but also the 'Avatar', 'Materials', 'Music', 'Prefabs', and 'UI' folders.

III Current Features

1 Overview

This section will outline the current features of the platform, and give some examples of how they are or could be used. Since we talked about the back-end in Section II, this section is more about the front–end and the user facing features.

The platform experience is centred around the shared content of the Unity scene; the environment loading locally the same way for everyone, and the players and certain objects sharing data with the server to update for everyone within that environment.

The platform supports live chat, the ability to upload an image to share art with others, spawning/removing/grabbing/moving objects, and a ball you can hit around.

We kept this part of the guide, and the features in the template in general, light because there's so much you can do with a fully-fledged game engine and we had to draw a line somewhere. Here we go into ultra-basic mechanics like spawning, grabbing, and uploading images. Some mechanics we've used in past online shared spaces that worked really well were basic NPCs, timers to trigger events, and pickup objects. Some more advanced ideas could include an inventory, progress flags, and questing. There's so much to do that really just depends on how much time you want to put into it.

2 Player-Level Features

Chat

The platform features a live chat where text input by the player is "spoken" in a speech bubble and appears to all others in the vicinity.

`SendMessage.cs` sends the message from the speaker to the server with `NetManager.cs`. Everyone else through `NetManager.cs` receiving the server message and calling `ReceiveMessage.cs` to display the message over the sender by being attached to the AvatarPrefab and updating a text box.

`UIController.cs` handles the behaviour of the chat panel by controlling the UI canvas attached to the AvatarPrefab.

3 Environment-Level Features

Interactivity Overview

Various interactivity mechanics are supported to interact with the environment beyond just shouting at it.

The back-end of calling custom functions, and how to get started making your own, is covered in Section IV: 4 Interactivity Customization.

Sharing Art

This very cute feature, written by Jonathan Carroll, allows a player to share art with others in the space by asking for an image file from their computer and displaying the image on a little canvas. The canvas can be grabbed, moved, and displayed by the player who spawned it or others.

The caveat with this feature is that there is no way to moderate the images players upload other than to delete them after they're created, so be careful with the crowd you're expecting at your event.

Custom Objects

The template has the ability to spawn particular objects into the world with a key press. The template also supports clicking and holding on the spawned object to move it around, and the ability to right–click on the object to remove it from the shared space.

We mapped the spawn action to the number keys, but you could change this to your liking. We spawned some random objects like gems and a ball.

The ball also demonstrates additional functionality that can be added to objects, in this case right-clicking on the ball sends it flying with an applied force.

IV Build Flow

1 Overview

This section will guide you through some additional advanced topics for customizing your event space, and how to get your event ready for your website.

We'll go over how to make any of the things you may add to your world sync up for every player, how to set up the website side of your event's web player, and how to create said web player for the website setup you completed in Section I: 4 Website.

2 WebGL Template

In Unity, WebGL projects are built into an HTML page to put online. You can customize the page that UNity builds the project to, using something called WebGL Templates. Make sure you complete this section to keep your project running smoothly!

MinimalSocket

In our template, we included a template we made called MinimalSocket. You can find it in the WebGL Templates folder in the project assets.

To set your WebGL Template, click through 'Edit > Project Settings...' and in Project Settings, go to the 'Player' menu on the left. In the Player menu, you can customize a lot about the way your project will look on the other side. We'll talk more about this in Section V. In the 'Resolution and Presentation' drop–down menu, you'll find 'WebGL Templates'. Here, make sure MinimalSocket is selected. Keep customizing, leaving the advanced stuff in 'Other Settings' menu alone unless you know what's up, or close the Project Settings window.

The MinimalSocket template's `index.html` file can and needs to be customized a little bit if you want to modify the look to suit your needs or add any metadata for social media and search engines. Open the file in your code editor of choice. It can even be Notepad.

Where it says `<!-- UPDATE ME --->`, update the URL to match that of your Glitch server. This will link this page with some data from your server, and make sure things don't break later when the random other server stops.

3 Venue & Landscape

This section covers how we set up the assets in our play space, and how you can customize your own. We set ours up as an indoor venue space with an outdoor area, but yours could be anything

Venue

In Unity's Hierarchy, the 'Venue' object contains our yellow venue building. It has a mesh for the building, the contents of each room sorted, and some other parts to dress it up and make the ramps work better.

We were inspired to create our space as a 3D version of our 2022 2D game. Your event could be this or anything else you want to put the time into making in Unity.

Landscape

The 'Landscape' object in the hierarchy similarly organizes the assets of the outdoor area of the template space. We made it into an enclosed area with the tall rocks, and thought it would be fun if everything was in the water. Other objects around dress it up a bit.

Just like the venue, the way you set this up is totally up to you. Our effort had a few layers of polish on it but you could put more or less. Happy worldbuilding!

4 Making Objects Shared

If you bring any new assets into Unity and add them to your scene, they'll only be visible to each local player and not show the same position, etc for everyone. TO make something sync up for all players there are some bits we can attach to share the object's data with the server, and as a result, everyone.

Net.Instantiate()

`Net.Instantiate()` is the call to an instantiate function of the `Net.cs` class. There are a number of functions with different arguments that can be seen in the code window's popup or read in the `Net.cs` code.

For most uses, it will take the name of a prefab, the type of object it will be on the server, and a position.

The server types are `TEMPORARY`, `PRIVATE`, `SHARED`, or `PERSISTENT`. `TEMPORARY` sets a timer on the server and removes it automatically. `PRIVATE` prevents others from seeing it even though it's on the server, `SHARED` is the default way to make something visible to everyone, and `PERSISTENT` shares it with everyone but doesn't allow it to be removed across sessions.

You can call one of the versions of `Net.Instantiate()` in your own code to create things that the server will know about. Examples of its use can be found in `FirstPersonInteraction.cs:53`. Here a green gem gets instantiated as a shared object.

NetInstantiate

`NetInstantiate.cs` is the component class version we can attach in the Editor to create an object linked to the server. `NetInstantiate.cs` just provides an Editor interface for calling the `Net.Instantiate()` function. Depending on your style or the use case it is useful to have this option.

An example of it in use can be found in the AvatarPrefab of the template, in the object 'CreateArt'. It is used in conjunction with other scripts to create and share the Art prefab, which is then updated with the user's image data.

5 Build Process

Once we've got everything we need to do in Unity mostly finished, it's time to make a build.

Builds are what you'll put up on the website for the masses to consume, but you'll also want to work through a phase of making builds to test your experience and looping back to make adjustments based on how it behaves in the build.

To do this, we'll click to 'File > Build Settings...' or press Ctrl/Cmd+Shift+B.

In the Build Settings menu, we'll first click the 'Add Open Scenes' button. If we're just trying to test some stuff we can choose 'Development Build' which enables some helpful things, and keep the selection next to 'Code Optimization' as 'Shorter Build Time'. If we're making a build to put up on the website we should uncheck 'Development Build' and choose check 'Runtime Speed' under 'Code Optimization' instead.

Click 'Build'. A dialog will pop up asking you where to put it on your computer. In the project folder, add a new folder called 'Builds' and save it there. Let Unity do its thinking,

which might take a few minutes depending on how much stuff you've got in the space and even more time if it's your first build.

Unfortunately, browser security will not allow you to run the build from the file locally so you'll have to move it to the spot on your website we set up at the beginning to test it out.

V Going Live

1 Overview

Congratulations on getting the content complete stage! There are still a couple things we should go over before showtime. We need to update our webpage headers so everything shows up nice in search engines and social media previews, and set up Glitch for the extra public traffic.

2 Installing On Your Website

There are two good ways to install and direct people to the experience on your website. First is to use the page that Unity builds, customizing the template to suit your needs. Second, is to keep the template minimal and to embed it on a different page. We'll chat about each.

Using Build Page

If your event is super simple, it might be easier to make some modifications to the template and direct attendees straight to this page.

If you'd like, you can make a copy of MinimalTemplate, name it something appropriate, and work from here. We edited this page in the last section so we'll use the same setup again. Modify this page with any bells and whistles you'd like. All the scripts, which should be left as-is, feed into the `<canvas>` element, so you're free to design a web page around that line.

Once you've got this page to your liking, you can put the build in a folder on the root of your website that will host the event (ie. https://handeyesociety.com/event). Then the template with the game in it will load when users go to this site.

Embedding Build Page

Alternatively, if you already have some other kind of website that goes along with your event, you can install the build to a folder on your website that will not be directly accessed

(like an assets folder, or for example, something like https://handeyesociety.com/event/build). Then you can embed this hidden site in something else with `<iframe>`. Using our URL example, it'll look something like this:

HTML

```
<div class="likelike_container">
    <iframe src="https://www.handeyesociety.com/event/build"
style="border:none;" height="700" width="1300" title="Event Title"></iframe>
</div>
```

CSS

```
.likelike_container {
    width: 100%;
    height: 100%;
    display: flex;
    align-items: center;
    align-content: center;
    justify-content: center;
}
```

Place this HTML into your existing web page at the spot you'd like. We included a wrapper and some CSS to keep the player centred on the page and taking up the full space of the game viewport.

3 Header Info & Metadata

Title & Favicon

The first thing to do to dress up the page is update the Product Name of the Unity project in the Project Settings. Click through 'Edit > Project Settings...' and in Project Settings, go to the 'Player' menu on the left. Here we can set 'Product Name'. This will update the text in the browser tab and on the taskbar/dock.

To update the icon that appears on the tab, you can make your own. It's a 16x16 pixel PNG you can create pixel art-style, or make a slightly bigger image and crunch it down with a good algorithm (Photoshop, GIMP, etc). Add it to your Assets folder, and clicking on the new image, select 'Sprite (2D and UI)' in the Texture Type dropdown in the Inspector. Select the icon in the 'Default Icon' box in the Player settings.

If you've made your own page and you're going to embed the game, you can update the favicon on that page too if you'd like.

SEO & Social Media

To get your project properly indexed by search engines and display the right thing in social media previews we need to set some <meta> tags. Other social media sites not shown here may have their own formats. Quickly Googling for your social media's preview meta tags will find what you need. Below is an example of what you're looking for. Add these to the top <head> section of the WebGL Template or the page you've embedded the <iframe> into.

Below are some examples of what to look for in your research.

```
<!-- SEO / social media-->
<meta name="title" content="LikeLike Template" />
<meta name="description" content="A tiny MMO exhibition template" />
<meta name="author" content="Hand Eye Society, Biome Collective, Molleindustria, MacKenzie Art Gallery" />
<meta name="copyright" content="LGPLv2.1" />
<meta name="robots" content="index, follow" />
<meta name="description" content="A tiny MMO exhibition template" />
<meta name="keywords" content="Game,Art,gallery,videogame,digtal,exhibition,hand eye" />

<meta property="og:image" content="https://cdn.glitch.global/028048c3-4b00-495b-8199-069e60bba66c/preview_image.png?v=1683819990210" />
<meta property="og:title" content="LikeLike Template" />
<meta property="og:description" content="A tiny MMO exhibition template" />

<meta name="twitter:card" content="summary" />
<meta name="twitter:site" content="@handeyesociety" />
<meta name="twitter:title" content="LikeLike Template" />
<meta name="twitter:description" content="A tiny MMO exhibition template" />
<meta name="twitter:image" content="https://cdn.glitch.global/028048c3-4b00-495b-8199-069e60bba66c/preview_image.png?v=1683819990210" />
```

4 Glitch

Depending on how many people you're expecting to attend your event, we recommend making some changes to the project on Glitch to make it a nicer time.

If you're expecting more than 20 concurrent active users, we suggest paying for Glitch Pro for the months your show is up or the months you have events focusing on it to boost your server CPU and memory capacity. Server lag can build up quickly on the free plan, especially if you've done some customizing and have added a lot of rooms and assets and a bunch of people are trying to load in at once. It's only $8 USD per month.

Glitch usually supports being pointed at by custom domains, but currently has a backend issue preventing this. Keep an eye on this linked page if you find yourself here in the future.

Outro

You made it! You're now more than ready to put up a little thing and have a show. Good luck with your event! We'd love to hear from you should you have any questions, comments, shows to promote, or nerdy stuff to link. Get at us!

See you out there :D

HES

Editor note: this document can be viewed in its original format at the link available here.