D O C U M E N T A T I O N

networked-locationbased-AR by saaribus (March 2024)

Intro

This documentation provides a detailed insight into the 'networked-locationbased-AR' GitHub repository. It offers an overview of its use case and currently implemented features.You'll learn how to set up a Unity project, import necessary packages, configure project settings, and build for different platforms. Using the interactive theater piece „INBETWEEN - die andere Stadt“, developed in 2022, I demonstrate the interaction between the different platforms and what a theatrical play system can look like in concrete terms. The code is extensively commented and this documentation describes the functionalities of each script. Please note that this repo uses and builds upon the publicly available code from Google ARCore aka ARFoundation and mirror by vis2k. Additionally, openstreetmap data is utilized for 3D city models, processed with the program osm2world to generate .obj files. Special thanks to all contributors who share their code openly.

Overview / Use case / Features

Since 2017 I have created several interactive and site specific theater pieces. All of them use the Game Engine Unity and Openstreetmap data to guide the audience through the physical environment and provide location based information in Augmented Reality. Over the years several features have been added to the system to meet project requirements. Generally the locationbased functionality relies on the parallelism of the virtual and the physical world. Using openstreetmap data, the relations of the physical world are translated accordingly to Unity coordinates. The view into the virtual world (the Unity camera) aligns with the mobile device’s gyroscope rotation. The position in the virtual world is calculated with the GPS coordinates of the player, converted to Unity coordinates. In order to move in the virtual world (on Android) the player has to move in the physical world.

The 'networked-locationbased-AR' repo provides access to a shared virtual world from three distinct entry points: the desktop computer application (Windows), the web application (WebGL) and mobile application (Android) accessed from the sight-specific outside world. Independent of the entry points all players have a 3D representation in the virtual world and can therefore see each other in the networked environment. The player character is customizable for each platform. You can change the whole 3D model or just the color of the mesh.

The implemented **features** are:

- Platform dependent character customization
- Networked Radio (AudioSource) that on click plays/pauses the audio on all clients simultaneously.
- Note generator, with which the players can leave notes in the virtual world, visible for everyone else, only deletable by the original poster.
- Specific areas, points of interest (POI), that trigger customizable events, handled differently for mobile and windows/WebGL.
- For Android: a switch between two different modes of anchoring the Augmented Reality content (changing from the gps based localisation to the SLAM method by Google ARCore)

Setup Unity Project / package dependencies

BEFORE you can download and successfully use the 'networked-locationbased-AR' package you have to do the following:

- Create a new Unity project (I used the Unity Editor Version 2021.3.ff LTS), with the 3D (URP) template

Import the following External Packages (via Unity Package Manager):

- ARFoundation 4.2.9
- ARCore XR Plugin 4.2.9

and via Github:

- ARFoundation Extensions v. 1.40.0: https://github.com/google-ar/arcore-unity-extensions/releases
- mirror Networking library v. 83.2.0: https://github.com/MirrorNetworking/Mirror/releases

For the ARFoundation Settings, follow the quick guide on Google ARCore: https://developers.google.com/ar/develop/unity-arf/getting-started-ar-foundation

Once you have all of this installed, download and import the 'networked-locationbased-AR' package from my github page.

Open the DemoScene_GPSARMultiplayer in Unity to see how the Scene is set up and how the elements correlate.

NOTE: If the materials of the demo scene appear pink, it most likely has to do with the Universal Render Pipeline Asset. In your assets folder create a new Rendering > URPAsset (with Universal Renderer). In the newly created Renderer you need to add the RenderFeature „ARBackground“. In the project settings > Graphics > set the reference to the newly created Universal Render Pipeline Asset. Now all the pink should be gone.

To test, press play and „host“ - the editor now works as a server and a client at the same time connected via localhost. With the keyboard („w“, „a“, „s“, „d“) and the mouse you can now move around in the virtual space.

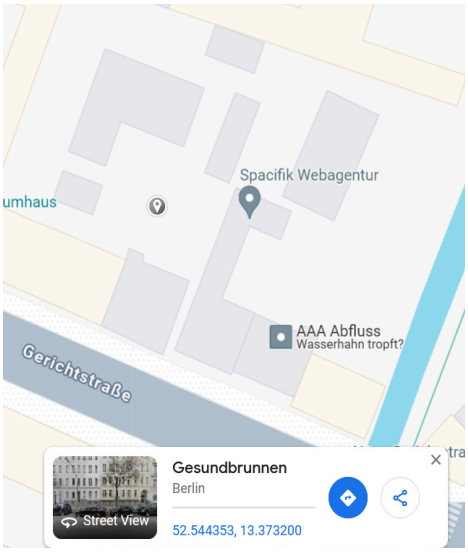
In Game View press „Host“

Setup area map and GPS for Android

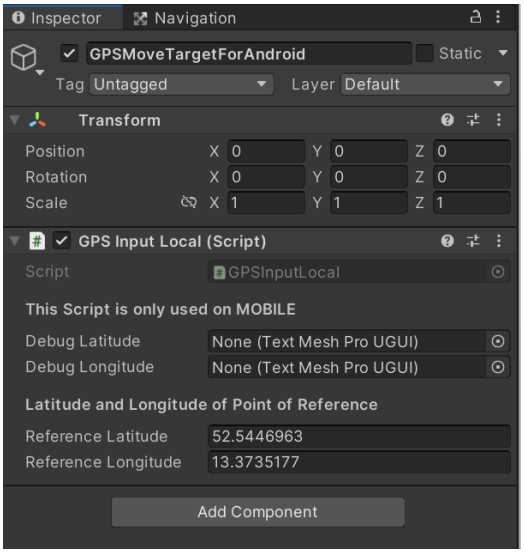
If you want to use the location based part on Android, you can set up the Unity Scene with a model of the area you want to walk around in. You can do this using Open Street Maps. Export an area from https://openstreetmaps.org and convert your .osm file into a .obj with the program you find on https://osm2world.org/. You possibly need an old Java Installation for this, if you don't have this already on your computer. Download the latest build and run the OSM2world.jar file. Drag and drop the .osm file into the OSM2World Viewer. Once loaded and visible, go to File > Export OBJ file. Then import it to your Unity project.

For more detailed instructions visit the wiki: https://wiki.openstreetmap.org/wiki/OSM2World

In Unity open the model in your Scene. Don't scale it, don't rotate it. In the DemoScene_GPSARMultiplayer you find a PointOfReference GameObject, this has to be at the world position 0,0,0. Drag and align the OBJ Game Object in the Editor, so that the PointOfReference points to a specific point in the real world, that you can easily identify on a map. Then go and find the exact latitude and longitude of this point on your map website (you can also use google maps, if you want).

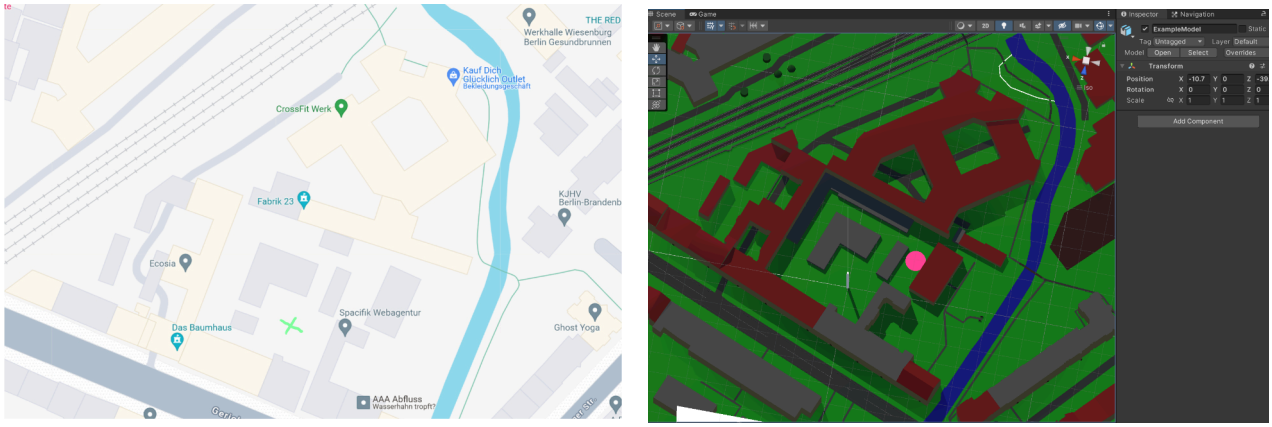


Find out the latitude and longitude of your chosen point in the world.



Set „Reference Latitude“ and „Reference Longitude“ in the Inspector.

On the GPSMoveTargetForAndroid Game Object find the GPSInputLocal.cs Component and set the „Reference Latitude“ and „Reference Longitude“ to the coordinates of the specific point in the physical world. This way, the PointOfReference Game Object and the latitude and longitude coordinates refer to the identical point in the world and on the 3D map object.



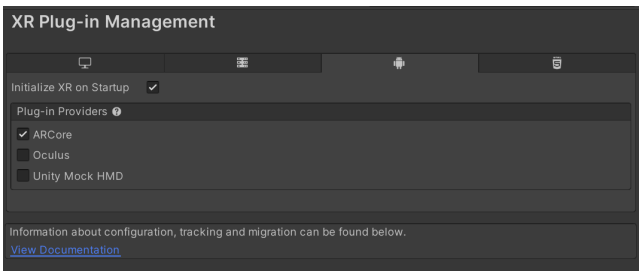
The chosen point in the real world aligns with the Transform Position of the PointOfReference Object in Unity

NOTE: The PointofReference Object needs to be at unity world coordinates 0,0,0. To align the virtual and the real world, the latitude and longitude as well as the OBJ in Unity have to align with the world coordinate 0,0,0. Don’t move the PointOfReference Object, move the map instead.

Project Settings

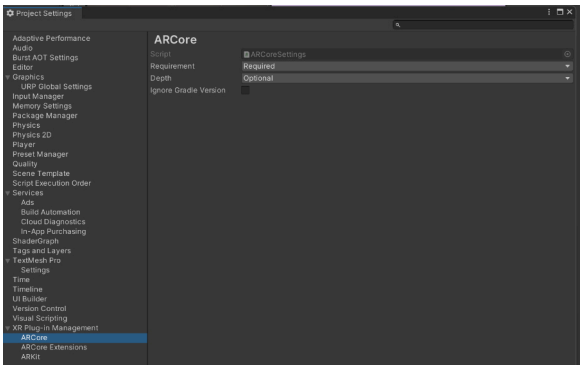
Before we look at the build settings for the different platforms, go to the project settings and make sure that under Graphics > Scriptable Render Pipeline Settings, you have the right URPAsset selected.

Under XR Plug-in Management > Android > check ARCore
(no other checks required)



Settings for XR Plug-in Management

Under XR Plug-in Management > ARCore have „Requirement“→ required, „Depth“ → optional



ARCore specific settings

Interaction of the three different platforms – Example INBETWEEN – die Andere Stadt

In the following I want to describe how the different platform builds (all originated in the same Unity project) interacted with each other in the participatory theater piece „INBETWEEN – die Andere Stadt“ we realized in autumn 2022 at Schaubude Berlin.

For „INBETWEEN“ we had the audience divided in two groups. One group stayed at the theater venue and could operate five computers. On all of them the Windows Build of the Unity project was running. In the virtual world each player was represented with a character. The audience members of this group were invited to investigate this world. They met different characters: they could see each other, they met non-player-Characters (NPCs), who resembled them but acted autonomously and the third type of characters they met where players of the other group.

While the first group stayed at the theater venue and was operating the computers, the other group was given tablets with the .apk installed (the Android build of the Unity Project) and was sent outside the theater space. Via the display and the gps location based positioning they saw into an alternative virtual world, that looked different from the physical world they were actually standing in, but resembled the space they were standing in in structure and dimensions. An actor then guided them to a nearby park. The locationbased players could see the players of the other tablet holders in the group through networking. In order to move in the virtual world they literally had to walk in the physical world. In the park next to the theater they would also encounter the NPCs mentioned before. In addition to this they would also meet the players of the group that stayed in the theater (on the Windows machines). On designated areas they could talk to each other or leave notes with the implemented Note-Generator feature. Despite the different contexts of the two groups, both were part of the same networked virtual world and could interact with each other.

Watch the Trailer of „INBETWEEN“ on youtube: https://youtu.be/rFbjrBI4ZHE

Build Settings depending on Platform

Unity allows us to build a project for different platforms, and the project allows us to integrate different networked clients from different platforms into the same virtual world, here are the specific requirements for different build targets:

Android
Other Settings > Rendering > uncheck «Auto Graphic API» > delete «Vulkan»
Other Settings > Identification > Minimum API Level: Android 9.0
Other Settings > Identification > Target API Level: Automatic (highest installed)
Other Settings > Configuration > Scripting Backend: IL2CPP
Other Settings > Configuration > Target Architecture: uncheck ARMv7, check ARMv64
Leave everything else on default.

Windows
Resolution and Presentation > Choose your desired „Fullscreenmode“ and „Default Screen Width/Height“
Other Settings > Rendering > Color Space: Linear
Leave everything else on default.

WebGL
there is no specific settings for the project to work, you can set your own preferences.

If you want to have a **Dedicated Server Build** to run on a remote server (e.g. Linode) switch platform to «Dedicated Server» and for the target platform choose „Target Platform“ „Linux“. With e.g. MobaXterm and WinSCP start the dedicated Server build on a remote computer.

Scripts and Functionality

Name	Function	Particularities
GPSInputLocal	• receiving GPS data • converting gps coordinates to Unity coordinates	• only used on Android
GyroRotation	• receiving gyroscope data • converting gyroscope data to unity rotation, applying to Unity Camera (GPSCam)	• only used on Android
TriggerChecker	• detects collision between GPSCam and the POI	• uses the tag «POI», collision detections can be expanded for different objects with new tags.
Poi	• describes what happens if player walks into POI (atm: turns on/off example cubes) • Distance Check for Android (desired exit distance can be set in the inspector)	• interacts with CamSwitcher.cs (on Android) • on windows only turns off/on objects
RadioHandlerHook	• Handles a networked, interactive radio, that can be clicked on to play/ pause an AudioSource	• on Radio Game Object • Sync Direction: Server to Client

Name	Function	Particularities
CamSwitcher	• Switches between GPS- and AR Camera. Handles the necessary steps in between: aligning view and position and turns on/off Camera Components	• only used on Android
PlatformIdentifier	• identifies the platform the application is running on and writes it to a debug UI TextMeshPro	• Only for debugging. Game Object can be set inactive
NoteSpawnManager	• processes the string of the input field and the position of the player to instantiate a Note across the network on every client. • Remembers the Original Poster.	• interacts with InteractableNote. cs • sync Direction: Server to Client
InteractableNote	• saves the string from the Input Field to the specific note Game Object • handles deleting itself	• Sync Direction: Server to Client
PseudoButtonOnClick	• detects clicks on the PseudoButton GameObject • checks if the clicker has authority over the note clicked on	• part of the Note-Prefab
MoreThanOnePlayerNetwork Manager (customized from: NetworkManager)	• handles all the mirror networking stuff • differentiates between the players' platform and instantiates a platform specific Player Game Object	• there are three different Player Objects: Android, Windows, WebGL • the Player-Prefabs need to be referenced in the Inspector and registered under „Registered Spawnable Prefabs"
ClientInstance	• all the Player-Stuff • is a Singleton that determines if the player is the local player • here you could setup additional OnServerStart things.	• exists on all the Player-Prefabs (platform unspecific)
BasicMovement	• handles keyboard and mouse movement • childing GPS Cam to the Player	• only on Windows / WebGL
AndroidRepresentationSetup	• NavMesh Agent and MoveTarget Setup • makes the GPSCam follow the player Game Object without childing it, to avoid jitter • applies only the xz-rotation to the Player Game Object.	• only on Android

For further details and explanations concerning the mirror specific or ARFoundation specific scripts, please consult the respective documentations:

Mirror library doccumentation: https://mirror-networking.gitbook.io/docs/manual/components

ARFoundation documentation: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.2/manual/index.html

Acknowledgments

This package is the third of three packages. The previous packages (GPS-AR-Switch and Locationbased AR) you can find as separate repositories on my Github page. Make sure you only have one of these packages installed at the time. They don't build up on each other.

This publication is made possible by the Rechercheförderung of Fonds Darstellende Künste under the name of "Vernetzte Fiktionen": Gefördert vom Fonds Darstellende Künste aus Mitteln der Beauftragten der Bundesregierung für Kultur und Medien Deutschland.

I want to thank Friedrich Kirschner, with whom I worked on several location based Augmented Reality projects and developed all of the code during the course of the last years. Also many thanks to the open source community on the internet. Without your sharing of code snippets and custom plugins, none of these projects would have been possible.

The people I worked with over the years and that contributed directly or indirectly to the theater projects and this code: Annina Polivka, Ralf Harder, Michael Anklin, Leoni Voegelin, Nicole Frei, Patrick Slanzi, Benjamin Drexler, Friedrich Kirschner, Nicolas Heitz, Hannes Kapsch, Alejandra Jenni, Sarah Freiermuth, Caspar Bankert, Tomás Montes Massa, Judith Hanke, Anna Vera Kelle, Annalena Steiner, Philipp Seifert. Thank you.