Creative Citizen Augmented Reality Application

Human-Computer Interaction

On 11. November 2020

Information Systems and New Media / Siegen University, Siegen

Author: Hassan Hussein
Hassan.hussein@student.uni-siegen.de

Table of Contents

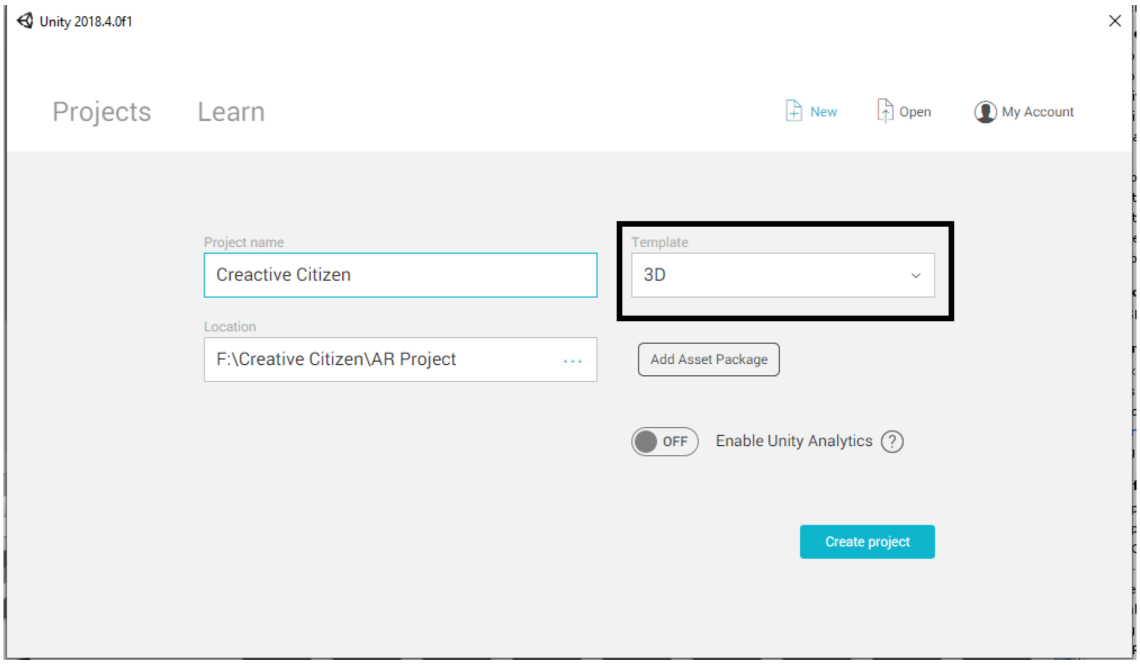
Introduction

The application is mainly built on an open source project called Mapbox. Mapbox itself is a ready made SDK for unity, that was built to help the developers to develop interactive augmented reality applications. The SDK helps the developers to create game objects via a C#-based API and graphical user interface. Mapbox supports both android and iOS builds, therefore it also supports Google AR core. This project was built also to support AR core.

Installation Guide

In this section we will learn how to install the augmented reality application on Android device. But before we do that, we should install the prober software, to make it work the best as expected. So it is very important at the first place to check the System Requirements section before you go ahead in this document.

1. Download the project from our GitHub repository here
2. Open unity and select create a new 3D project, give your project a name, select the downloaded file location.



Unity 2018.4.0f1

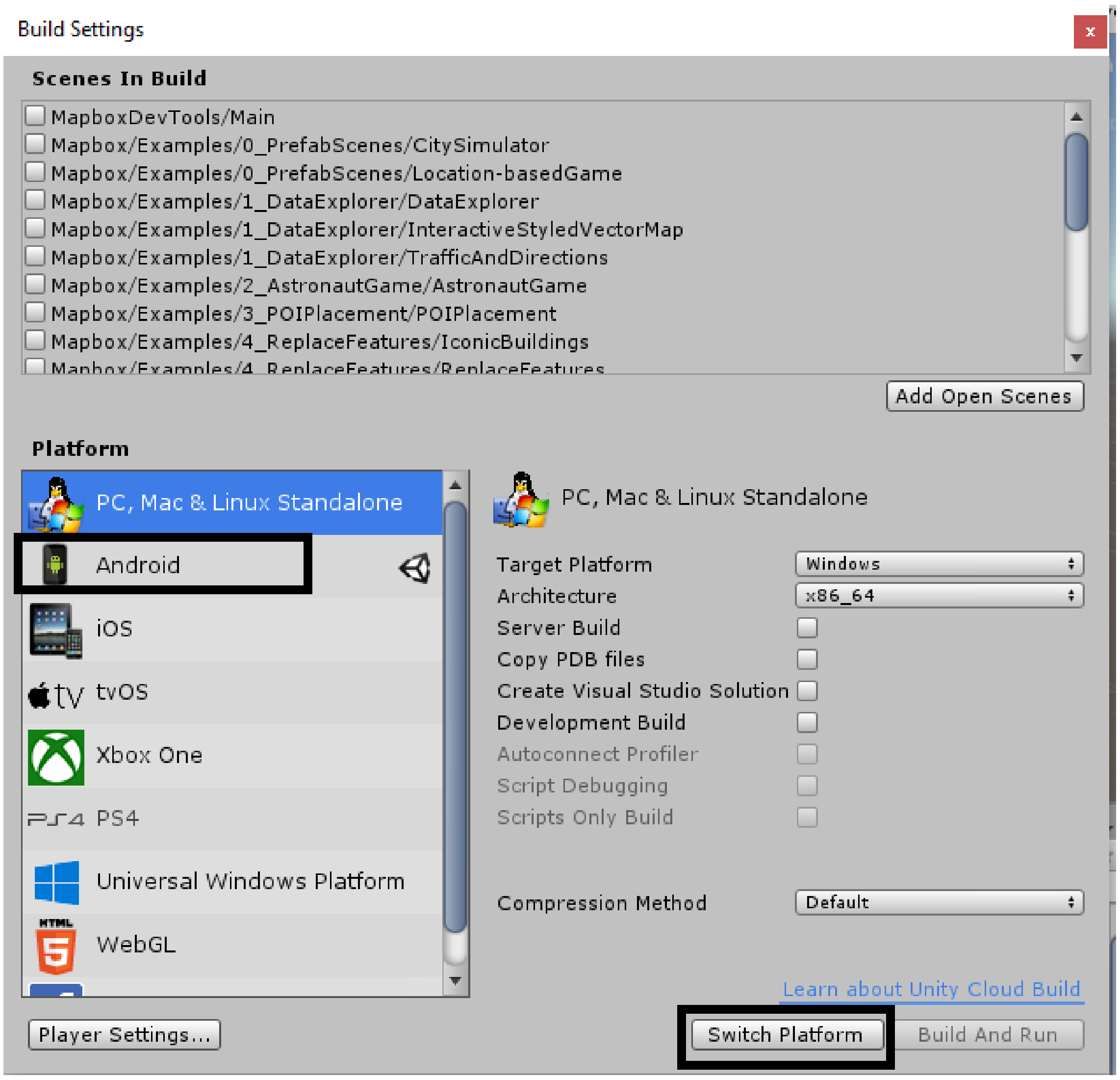
Project name: Creactive Citizen

Template: 3D

Location: F:\Creative Citizen\AR Project

OFF Enable Unity Analytics

3. In unity select File -> Build Settings and then select Android, and then switch platform.


Build Settings

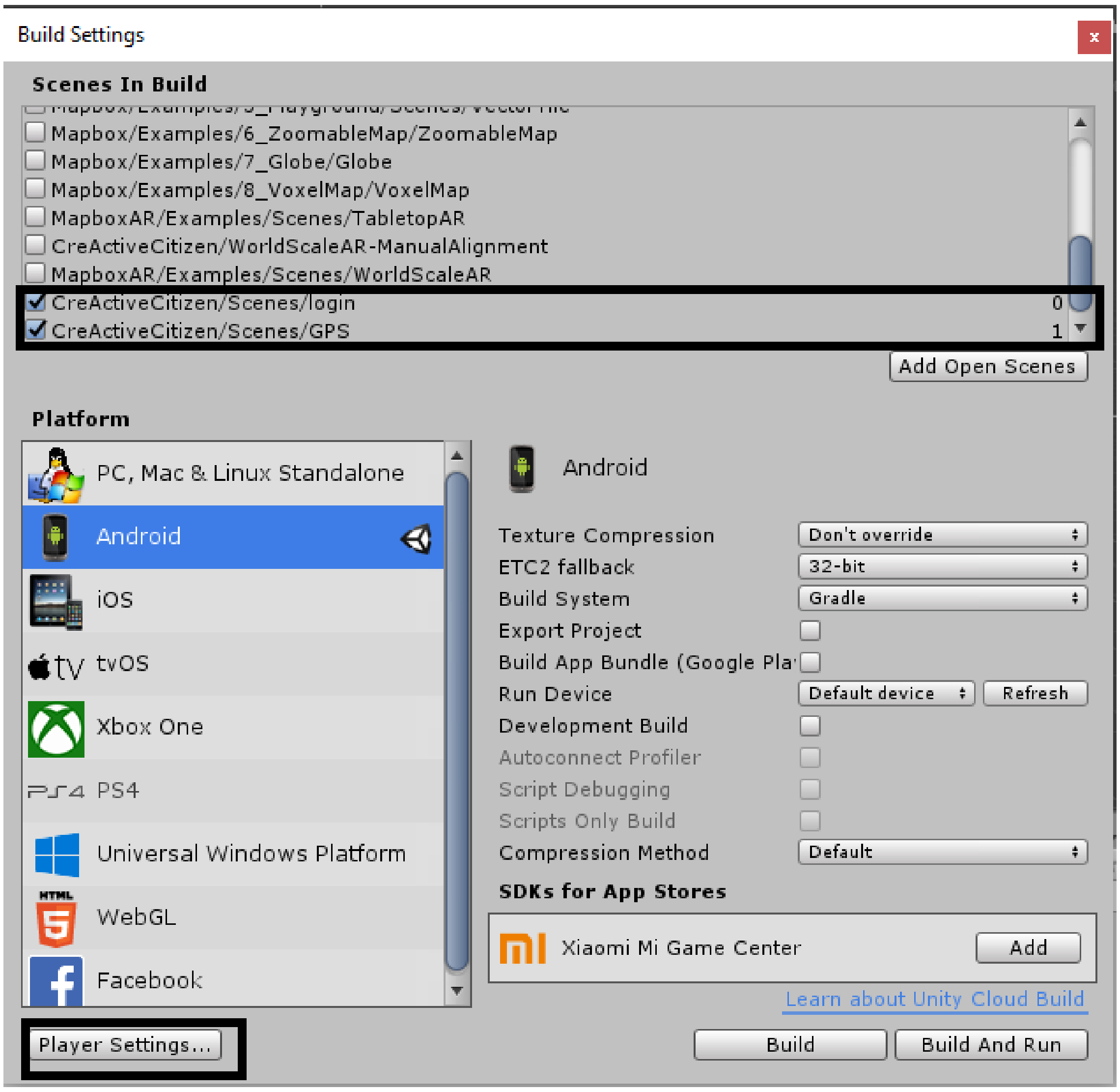
Scenes In Build

- MapboxDevTools/Main
- Mapbox/Examples/0_PrefabScenes/CitySimulator
- Mapbox/Examples/0_PrefabScenes/Location-basedGame
- Mapbox/Examples/1_DataExplorer/DataExplorer
- Mapbox/Examples/1_DataExplorer/InteractiveStyledVectorMap
- Mapbox/Examples/1_DataExplorer/TrafficAndDirections
- Mapbox/Examples/2_AstronautGame/AstronautGame
- Mapbox/Examples/3_POIPlacement/POIPlacement
- Mapbox/Examples/4_ReplaceFeatures/IconicBuildings
- Mapbox/Examples/4_ReplaceFeatures/ReplaceFeatures

Add Open Scenes
Platform
PC, Mac & Linux Standalone
Android
iOS
tvOS
Xbox One
PS4
Universal Windows Platform
WebGL
PC, Mac & Linux Standalone

Target Platform: Windows
Architecture: x86_64
Server Build
Copy PDB files
Create Visual Studio Solution
Development Build
Autoconnect Profiler
Script Debugging
Scripts Only Build
Compression Method: Default

4. Make sure that at least GPS, and Login scenes are checked and next to login is 0, while next to GPS is 1 in the below list, if not click on the checkbox next to login first and then the checkbox next to GPS.


Build Settings

Scenes In Build

- [ ] Mapbox/Examples/6_ZoomableMap/ZoomableMap
- [ ] Mapbox/Examples/7_Globe/Globe
- [ ] Mapbox/Examples/8_VoxelMap/VoxelMap
- [ ] MapboxAR/Examples/Scenes/TabletopAR
- [ ] CreActiveCitizen/WorldScaleAR-ManualAlignment
- [ ] MapboxAR/Examples/Scenes/WorldScaleAR
- [✓] CreActiveCitizen/Scenes/login 0
- [✓] CreActiveCitizen/Scenes/GPS 1

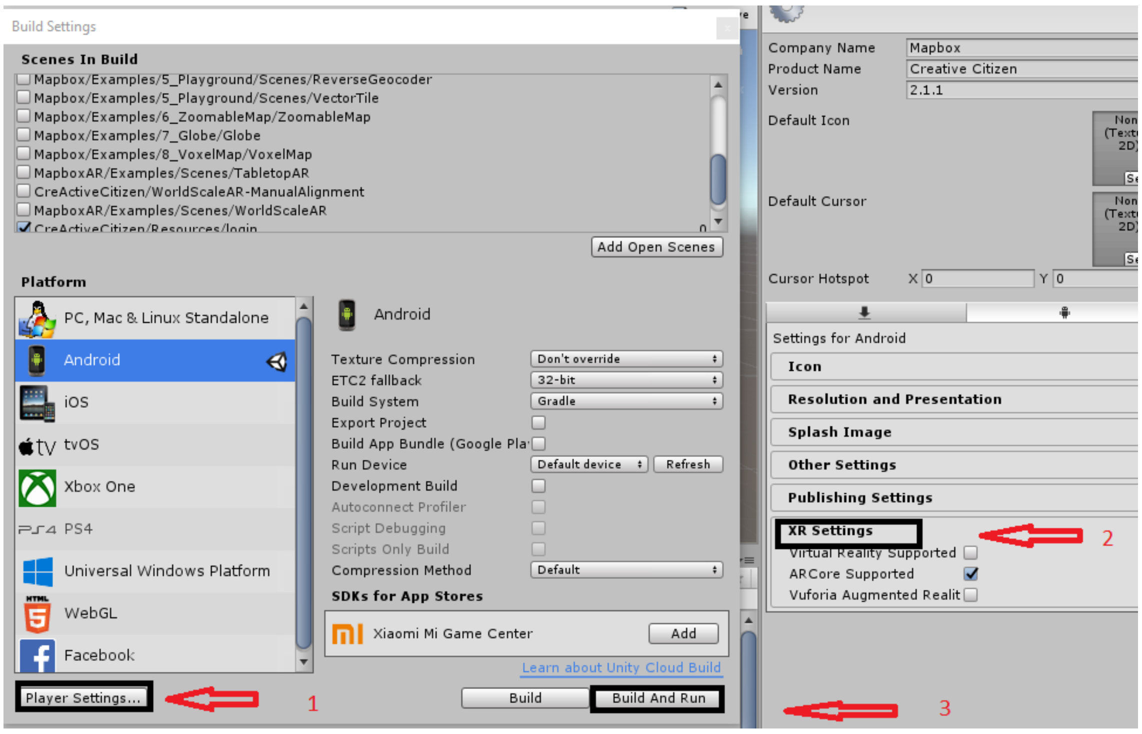
Platform

Android

Texture Compression	Don't override
ETC2 fallback	32-bit
Build System	Gradle
Export Project	☐
Build App Bundle (Google Pla	☐
Run Device	Default device
Development Build	☐
Autoconnect Profiler	☐
Script Debugging	☐
Scripts Only Build	☐
Compression Method	Default

SDKs for App Stores

5. Click on player settings, and then other settings. Make sure that the minimum API level is Android 7 or higher (it depends on the Android version on your device)
6. Scroll down a little bit and click XR settings and then click on ARCore Supported.
7. Leave every thins the same in the player settings. Now, your build is ready, and you can click on build and run



Build Settings

Scenes In Build

- [ ] Mapbox/Examples/5_Playground/Scenes/ReverseGeocoder
- [ ] Mapbox/Examples/5_Playground/Scenes/VectorTile
- [ ] Mapbox/Examples/6_ZoomableMap/ZoomableMap
- [ ] Mapbox/Examples/7_Globe/Globe
- [ ] Mapbox/Examples/8_VoxelMap/VoxelMap
- [ ] MapboxAR/Examples/Scenes/TabletopAR
- [ ] CreActiveCitizen/WorldScaleAR-ManualAlignment
- [ ] MapboxAR/Examples/Scenes/WorldScaleAR
- [✓] CreActiveCitizen/Resources/login 0

Add Open Scenes

Platform

PC, Mac & Linux Standalone
Android
iOS
tvOS
Xbox One
PS4
Universal Windows Platform
WebGL
Facebook

Android

Texture Compression	Don't override
ETC2 fallback	32-bit
Build System	Gradle
Export Project	☐
Build App Bundle (Google Pla	☐
Run Device	Default device / Refresh
Development Build	☐
Autoconnect Profiler	☐
Script Debugging	☐
Scripts Only Build	☐
Compression Method	Default

SDKs for App Stores

Xiaomi Mi Game Center — Add

Learn about Unity Cloud Build

Player Settings... 1

Build — Build And Run

Company Name	Mapbox
Product Name	Creative Citizen
Version	2.1.1
Default Icon	
Default Cursor	
Cursor Hotspot	X 0 Y 0

Settings for Android

Icon

Resolution and Presentation

Splash Image

Other Settings

Publishing Settings

XR Settings 2

Virtual Reality Supported	☐
ARCore Supported	✓
Vuforia Augmented Realit	☐

3

8. Before you click on build and run. You need to connect your android device to your computer (laptop/PC) using the USB cable.
9. In addition, you should prepare your android device for the build by enabling developer options and USB debugging by following these steps
10. After the application is installed on your device, open the application and then the application will ask you for some permissions please accept it.
11. Log in using your username and password, if you see a black screen you might need to give some additional permissions to the app. From your android device go to settings->apps and then click on your app name (if you did not change it before the build in the player settings section the name should be “creative citi-zen”) and then -> authorization and then give permissions to the camera.
12. Close the application and make sure to remove it from your task manager and then reopen it again. Make sure that you have internet connection either using a Wi-Fi or using your mobile data. In addition, you will need to enable the GPS in your android device.

System Requirements

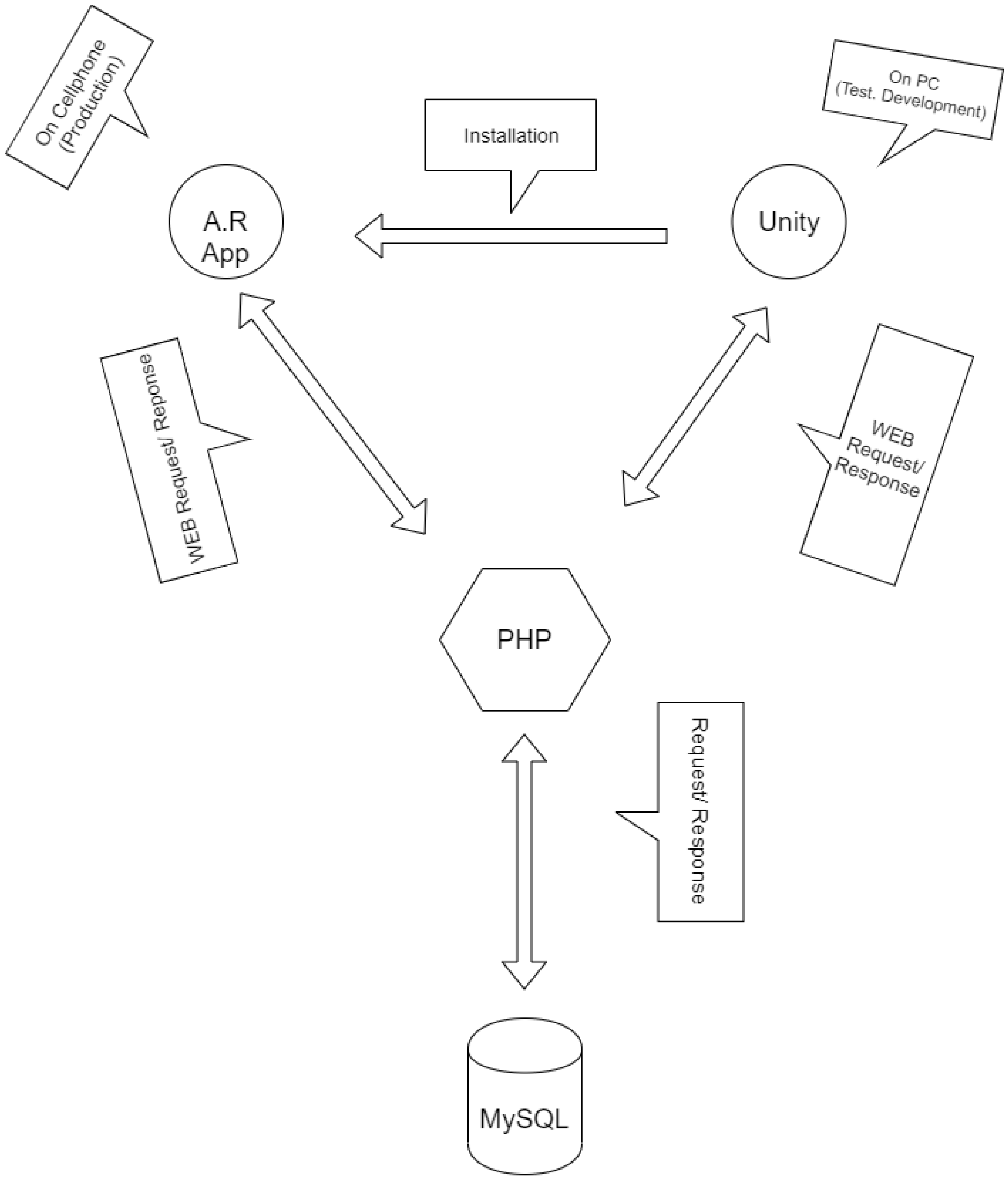
	System Name	Version	Type
1	Unity	2018.4.0f1 or higher from here	Software
2	Android device supports AR Core	7.0 or newer. Please check this page	Hardware
3	USB Cable	Any	Hardware

System Design

The application is designed as shown in the below diagram. The system is developed mainly in 3 layers as following:

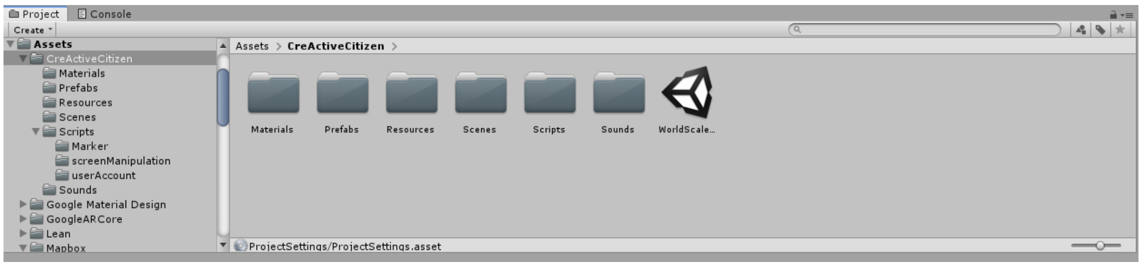
1. **Unity**: is used as the main design interface for the application. It is mainly used to design and integrate the C# scripts with the game objects. It is also used as a testing tool to test the application before deploying it to the android device.
2. **PHP& MySQL:** Both are built on a web server to save the user information and markers' location (longitude& latitude). MySQL was selected because it is an open source database, and it has a high integrate with PHP. Furthermore, PHP is a scalable scripting language that is widely used in the web, and it has a good compatibility to integrate with the C# scripts. Both PHP, and MySQL are acting as the application backend where it send/ request the user data between the android device and the end user.
3. **Android Device:** The device serves as the application frontend, where the user can interact with application and set his/ her marker and design the scene on his/ her own.

The crucial question here is why saving the data on the database and not on the user device? The reason behind that, that the philosophy behind the application itself requires that the user should be able to see the other users' markers. In that case it will be not helpful to save the users markers location on their own devices because in that case it will be impossible to show their markers to the other users.


On Cellphone (Production)
Installation
On PC (Test. Development)
A.R App
Unity
WEB Request/ Reponse
WEB Request/ Response
PHP
Request/ Response
MySQL

Application Design

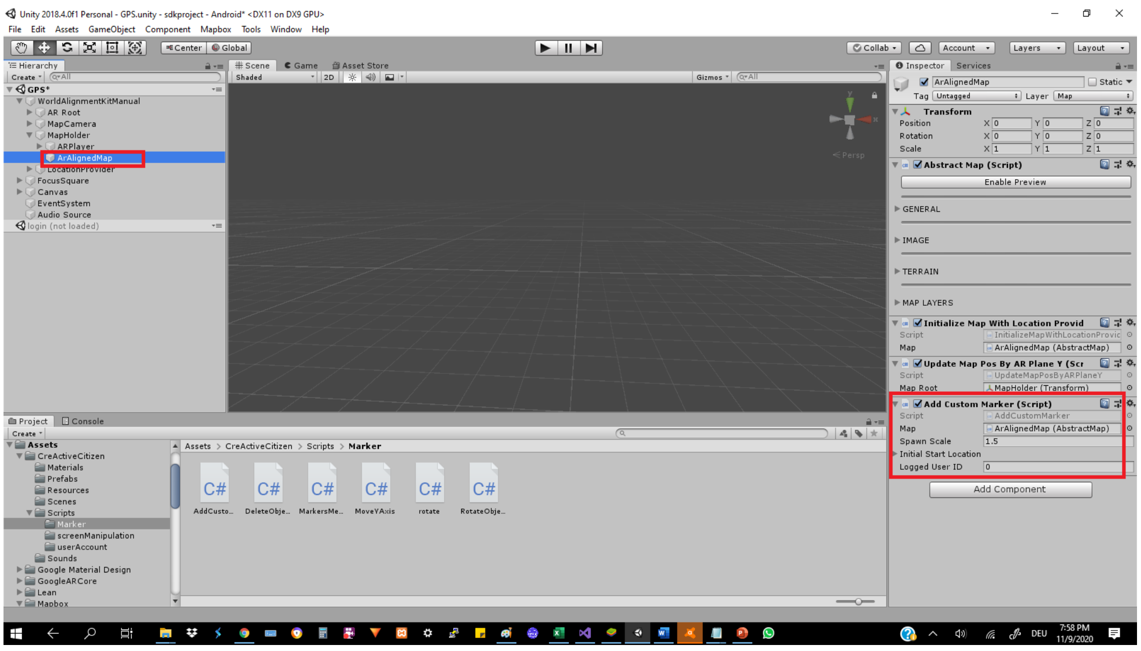
After importing the application to unity you can find it on the path Assets/ CreActive-Citizen.



As it is shown in the previous photo, there are mainly the following folders:

1. **Materials**: In this folder are mainly all icons, that are used in the application. Also, it should contain all the materials, that will be used in the future (no materials were developed for this project so far).
2. **Prefabs**: In this folder there is all the prefabs that are used in the project. It was mainly used to back up the designs. Just in case I needed to revert some changes. The prefab of any design could be easily created be dragging it from the hierarchy and drop it to the prefab folder.
3. **Resources:** It is very important to state at the beginning, that this folder name should not be changed because the markers menu is searching for the markers objects in this folder. In this folder all the 3D models of the markers are located.
4. **Scenes:** In this folder all the scenes are loaded, where I redirect the user from a scene to another depending on their actions. For example, when the application is loaded, at the beginning I load the login scene, and when the user can login successfully the application redirect him/her automatically to the GPS scene, where the user can experience the augmented reality.
5. **Scripts:** In this folder all the C# scripts, that are related to the application are located. In this folder is mainly the DB Manager script located. This script is responsible to handle all the database requests between the front and back end. The folder also has some subfolders as following:
    a. **Marker:** In this folder, you will find all the scripts that deals with the object. Below is the list of these scripts:

i. **Add Custom Marker:** In this script all the magic happens, as here the user can add a marker of his/her choice. Also, this script is responsible for displaying all the markers that are in a circle of 100 meters based on the user current location(the list of markers is populated by a php script and passed as a JSON format to the script). Furthermore, all the functionalities that are related to the markers itself are written in this script. This is script is attached to the map, to see that go to scenes folder and find GPS scene double click it, and then at the hierarchy expand it. You will find the script under GPS-> WorldAlignmentKitManual-> MapHolder-> ArAlignedMap



ArAlignedMap
Tag Untagged Layer Map

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Abstract Map (Script)
Enable Preview
GENERAL
IMAGE
TERRAIN
MAP LAYERS

Initialize Map With Location Provid
Script InitializeMapWithLocationProvic
Map ArAlignedMap (AbstractMap)

Update Map Pos By AR Plane Y (Scr
Script UpdateMapPosByARPlaneY
Map Root MapHolder (Transform)

Add Custom Marker (Script)
Script AddCustomMarker
Map ArAlignedMap (AbstractMap)
Spawn Scale 1.5
Initial Start Location
Logged User ID 0

Add Component

ii. **Delete Object:** This script is attached to prefab closeMarker under folder Assets/ CreActiveCitizen/Prefabs. This prefab is attached to each marker object manually under Assets/ CreActiveCitizen/Resources and it is shown and hidden using C# script according to the marker case for example it is shown on the top of marker, if it was set by the logged in user. (the close object interaction code e.g., show, hide is written in script add custom marker)

        iii. **Marker Menu:** This script is responsible for handling all actions related to the menu. The script is attached to object GPS ->Canvas -> Objects List

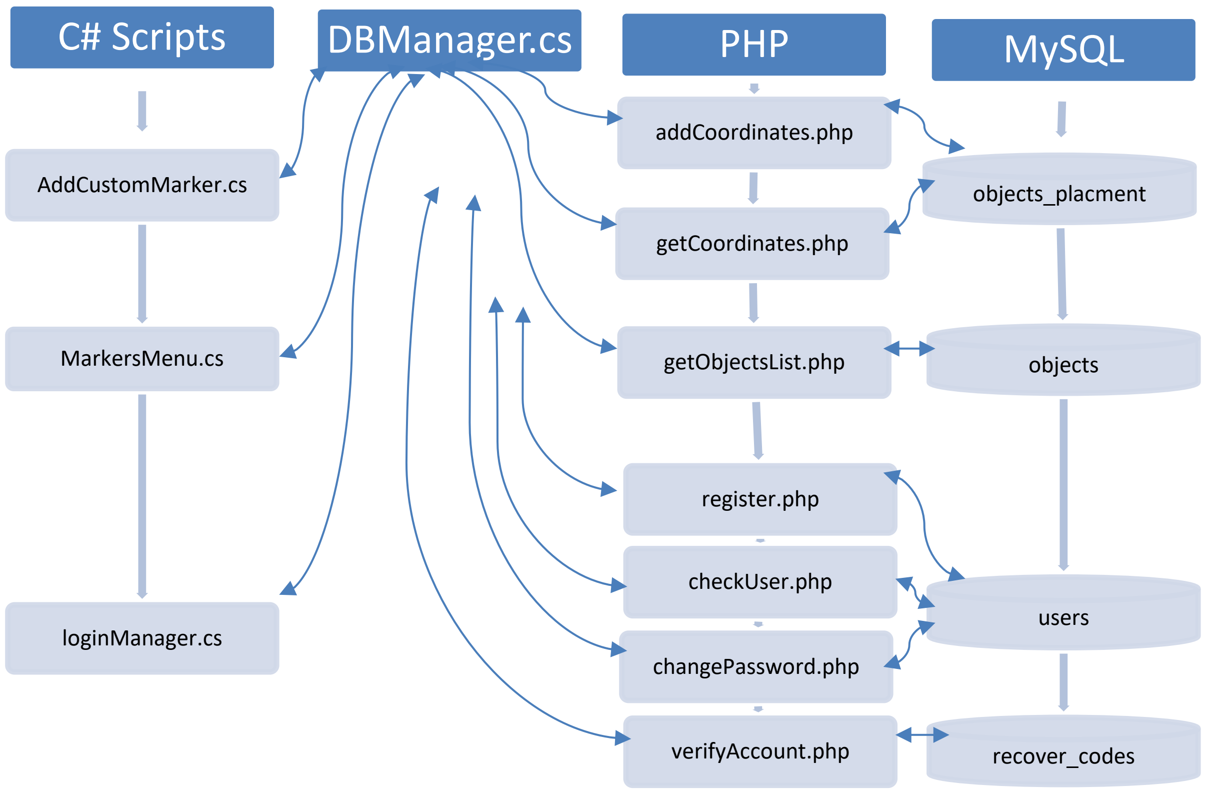
    b. **Screen Manipulation:** There is only two scripts here, one is responsible for sharing the screen and one is responsible for taking a screenshot.
    c. **User Account:** In this folder, there is only one script. This script is very important and responsible for managing all functionalities related to the user account, such as login, register, forget password. All the functionalities are existed but not working fully, as there is a problem when moving between the scenes, the user information is lost. But the user can login normally and after login his/her user ID is registered to the system memory as an identification ID, that is used all time to interact with the database.

6. **Sounds:** According to the android guideline, it is highly recommended to play sound along side with the user actions to give him/her the feeling that something is happening in the system. This functionality is newly implemented to the application, and it is only used with the screenshot function. It is highly recommended to save all the future sounds in this folder to make the system more organized.

Code Design

The code was designed in a way to make the system very generic and give the ability for more scalability in the future. There are some functionalities were added recently in a short time. So, it is highly recommended to refactor the code as soon as possible. So that, in the future changes, would it be easier for tracking and changing. There are almost comments in every line inside the folder Assets/ CreActiveCitizen/Scripts, and any other C# script in the project is a part of the Mapbox project, and it is highly recommended to refer back to the project official documents if needed.

In this section we will learn more about how the relation between the application parts (C#,PHP, MySQL) are linked and how the data flow works in the application. We can see the application from the code level as three tires as following:

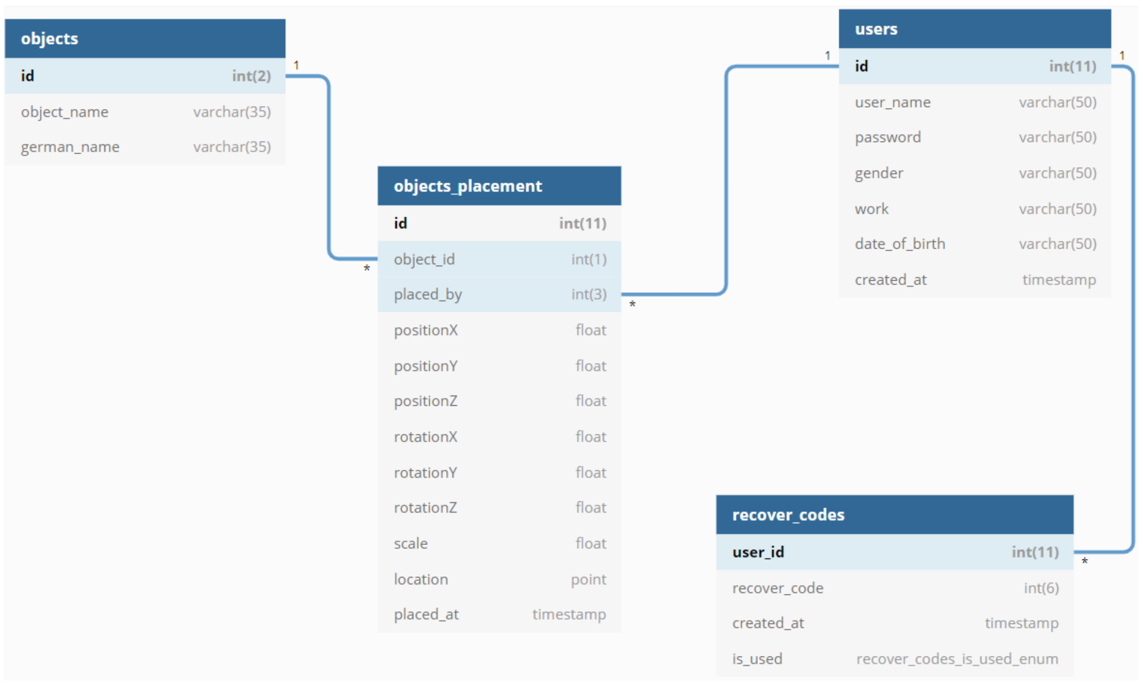


The script DBManager acts as the bridge between the C# scripts and the PHP scripts. The data is passed between the C# and the PHP is in JSON format, while the data that is passed between the PHP and MySQL, is in SQL query format. The below table shows which functions are called in each script.

C# Script	C# Method		D.B.M Method	PHP Script	Table
Add Custom Marker .cs	Save Marker Info ()	DB Management Script (D.B.M)	Post Coordinates ()	Add Coordinates .php	Objects _ placement
	Spawn Markers on Start ()		Get Coordinates ()	Get Coordinates .php	
Markers Menu .cs	Start ()		Get Objects Lists ()	Get Objects List .php	objects
Login Manager .cs	Check Login Info ()		Check Login Data ()	Check User. php	Users
	Register ()		Register ()	Register .php	
	Forget Password ()		Recover Password ()	Forget Password .php	
	Change Password ()		Change Password ()	Change Password .php	
	Verify E Mail ()		After Registration ()	Verify Account .php	Recover _codes

Database Design

The below database diagram shows the relations between the different tables:



objects	
id	int(2)
object_name	varchar(35)
german_name	varchar(35)

objects_placement	
id	int(11)
object_id	int(1)
placed_by	int(3)
positionX	float
positionY	float
positionZ	float
rotationX	float
rotationY	float
rotationZ	float
scale	float
location	point
placed_at	timestamp

users	
id	int(11)
user_name	varchar(50)
password	varchar(50)
gender	varchar(50)
work	varchar(50)
date_of_birth	varchar(50)
created_at	timestamp

recover_codes	
user_id	int(11)
recover_code	int(6)
created_at	timestamp
is_used	recover_codes_is_used_enum

The below table explains each table, and data type and which column is needed for:

Table Name	Column Name	Datatype	Notes
Objects	ID	Integer	Primary key
	Object_ name	Varchar	Should be the same name as the object name in folder /Assets/Cre Active Citizen/Resources
	German_ name	Varchar	This is what is showed in the app menu
xes	xes	xes	xes
	Object _id	Integer	Foreign Key of table objects
	Placed _by	Integer	Foreign Key of table users
	Position X	Float	Object position in A.R environment on the axes x
	Position Y	Float	Object position in A.R environment on the axes y

	Position Z	Float	Object position in A.R environment on the axes z
	Rotation X	Float	The object rotation in the AR environment using Euler Angles calculation on the axes x
	Rotation Y	Float	The object rotation in the AR environment using Euler Angles calculation on the axes y
	Rotation Z	Float	The object rotation in the AR environment using Euler Angles calculation on the axes z
	Scale	Float	The object scale according to unity measures
	Location	Point	The object location based on the GPS system in two coordinates (longitude, latitude)
	Placed _at	Timestamp	The time when the object was set in the AR environment
Users	Id	Integer	Primary Key
	User _name	Varchar	Usually it is email address
	Password	Varchar	One-way encrypted password using method MD5
	Gender	Varchar	It is better to be changed later to Enum type, to be standard
	Work	Varchar	
	Date _of _birth	Varchar	
	Created _at	Timestamp	When was the account created for the first time
Recover _ codes	User _id	Integer	Primary key
	Recover _code	Integer	Randomly generated code to allow the user to recover the account

	Created _at	Timestamp	When was the code was created (it is recommended for security reasons to make the code expire after some time e.g. 24 hours)
	Is _used	Enum	A flag to identify whether the code was activated or not

Added Assets

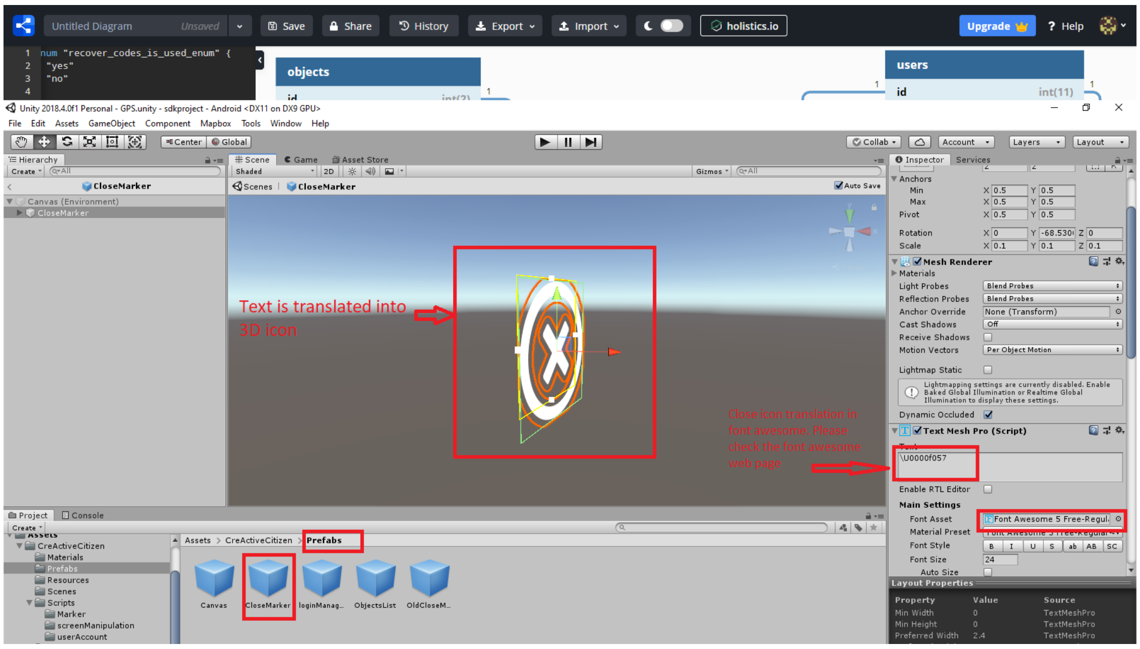
There are some assets that were imported from unity assets to enrich the experience. The assets are listed below:

1. **Lean**: This asset is used to give the object the elasticity and scalability. So, the user can move the marker in all directions and can also resize it to the desired size. This asset is imported from this link. The asset components could be added as a component manually to any object but in our case it is added using scripts, as the markers are added dynamically and components also should be added like this. Below is an example form script add custom marker:

```
obj.AddComponent<LeanTouch>();
var Scale = obj.AddComponent<LeanPinchScale>();
Scale.Sensitivity = 1f;
Scale.Dampening = -1;
var Twist = obj.AddComponent<LeanTwistRotateAxis>();
Space World = default;
Twist.Space = World;

var Move = obj.AddComponent<LeanDragTranslate>();
```

2. **Text Mesh Pro**: This asset is used to add mainly used to add some icons using the font awesome. It is mainly imported to create the close icon, as the photo icons in the AR environment looks like a 2D image, while the Text Mesh Pro looks like a 3D object. Please look at the below screen.


Text is translated into 3D icon
Close icon translation in font awesome. Please check the font awesome web page
CloseMarker
Prefabs
Canvas
CloseMarker
loginManag..
ObjectsList
OldCloseM..

Text: \U0000f057

Font Asset: Font Awesome 5 Free-Regul...

Recommendations

Testing AR apps on unity is a nightmare, as some functionalities cannot be tested from unity like touching the screen for example. So, with every little change you must deploy the app on your android device and then test it. The deployment time usually takes around 7-10 minutes depending on your hardware efficiency. But this is not now a problem since unity has offered a great tool called Unity Remote. This tool must be installed to your android device, and when you connect your device with the USB cable to your computer, so you can test the running application on unity from your device. More information about the tool and how you can use it from here