Sitsim AR Editor

User's guide

Version 1.4 - 18 April 2019

Sitsim AR Editor

Preface

The Sitsim (situated simulation) platform emerged immediately after Unity launched its support for iPhone OS in October 2008. Since the early experiments the platform has evolved into a complex system for creating advanced Indirect Augmented Reality (AR) applications (for use in both past and future settings). Since our early projects and productions we have often been asked by developers and designers alike if it was possible to get access to the platform and its code. Since the overall development of the platform is based on research funding we have not yet had the resources to support an open public version. Plans for making a free basic editor as an add on to Unity have slowly developed over the years. Finally, with our participation in the CINE–project, it was possible to proceed with these plans. With this first release of the Sitsim AR Editor we have included some of the most basic features of the Sitsim platform: import of environments and objects, including scaling, orientation and positioning, as well as adding of spatially distributed hypertext links ('balloon links') with access to different types of information, documents and events. We hope that the Sitsim AR Editor will benefit digital designers and curators whenever they wish to augment cultural heritage sites or any location where alternative versions and simulations of the current environment is wanted.

More information about the CINE–project can be found at <http://cine.interreg-npa.eu>
More information about the Sitsim–project can be found at <http://www.sitsim.org>

Tomas Stenarson/Gunnar Liestøl
Stockholm/Oslo, December 2018

Contents

Preface 2

Introduction 4

Limitations 4

Prerequisites 4

- Mac running macOS 10.13 4
- Unity 2018.2 4
- Xcode 10.x 4
- Apple developer account 4
- iOS device 5
- Basic knowledge 5
- Reference system 5

Preparations 5

Creating a Sitsim AR application 6

- Application Settings 6
- Core Sitsim AR Settings 7
- Terrain Settings 8
- Player Settings 10
- Balloon Editor 10
- Build Xcode project 11

Tips and Tricks 11

- Add fences 11
- Multiple Scenes 12
- Drop the crutches 12

Common Issues 12

- Editor window out of sync 12
- Sitsim application crashes at start 12
- Xcode Provisioning profiles 13

Introduction

The Sitsim AR Editor is a proof of concept application used to simplify the process of creating sitsim applications. The editor guides the user through a step-by-step process of configuring a sitsim application and adding content. After creating a basic sitsim application, the user is free to add more content or build the application as it is to a device.

Limitations

The Sitsim Application Framework that is used by the Sitsim AR Editor does currently only support iOS devices, i.e iPhone and iPad. As such, the development work has to be done on a system running macOS.

Prerequisites

In order to use the Sitsim AR Editor to build sitsim application there are a couple of prerequisites regarding hardware, software and online services.

Mac running macOS 10.13

A Mac running macOS 10.13 or later is required to build the final sitsim application. The Sitsim AR Editor has been developed and tested on a MacBook Pro running macOS 10.13.5.

Unity 2018.2

The Sitsim AR Editor is based on the Unity editor (https://unity3d.com), the editor is not an application itself but is a plugin to Unity that extends its functionality and adds in the Sitsim Application Framework. The editor by itself works with Unity 2017.3, but Unity 2018.2 is required in order to build the final application.

Xcode 10.x

Xcode is Apple's tool (toolkit) for developing applications for iOS devices and is required to build the final sitsim application. The Sitsim Application Framework requires Xcode 10.x and Swift 4.2 (https://itunes.apple.com/se/app/xcode/id497799835).

Apple developer account

The Sitsim AR Editor can be used without an Apple developer account (see the developer portal at https://developer.apple.com), but in order to build the final application an account is necessary.

iOS device

In order to build and run the final sitsim application, an iOS device running iOS 9.1 or later is required. The sitsim applications has recently been tested on multiple device generations back to iPhone 6, but depending on the content a device with better performance may be needed for optimum user experience. The recommended devices are iPhone 7 (2016), iPad (2017 or later) and iPad Pro.

Basic knowledge

This guide doesn't go into details in using Unity or how to develop application for the iOS platform. The user is expected to have basic knowledge in using Unity and in developing iOS applications.

Reference system

The following system (hardware and installed software) is the reference system

- MacBook Pro (15-inch, late 2016) with macOS 10.13.6
- Xcode 10.1 with Swift 4.2
- Unity 2018.2
- iPhone X with iOS 12.1.x and iPad Pro with iOS 12.1 (9.7 inch)

Preparations

By following these steps, you will have a new Sitsim AR Editor project, ready to start building a sitsim application!

1. Start Unity
2. Create a new Unity project
    1. Use Template "3D"
    2. Unity Analytics is optional
3. Optional: Configure your SCM system (Software Configuration Management)[1]
4. Configure the project for iOS
    1. In Unity, navigate to the menu "File -> Build Settings"
    2. In the dialog that appears, select Platform: "iOS" and press "Switch Platform"
    3. Close the dialog
5. Import *Sitsim AR Editor.unitypackage*
    1. In Unity, navigate to the menu "Assets -> Import Package -> Custom Package…"
    2. In the open dialog, select the file *Sitsim AR Editor.unitypackage* and press "Open"
    3. In the dialog that appears, make sure all items are selected and press "Import"

[1] It is recommended to use some kind of SCM, e.g Git (https://git-scm.com) or Mercurial (https://www.mercurial-scm.org) but it's not necessary. See Unity documentation for help on setting up version control for Unity projects: https://docs.unity3d.com/Manual/ExternalVersionControlSystemSupport.html

Creating a Sitsim AR application

In Unity, navigate to the menu “Sitsim AR -> Editor”; this will open a new window from which you will continue configuring your sitsim application. The window is named “Sitsim AR Editor” but will from hereon be referred to as **the editor window**.

If you see the “Get started” button in the editor window, press it to continue.

Before configuring your sitsim application, the editor needs to perform some basic configuration and setup, press the “Prepare Sitsim AR project” button and continue.

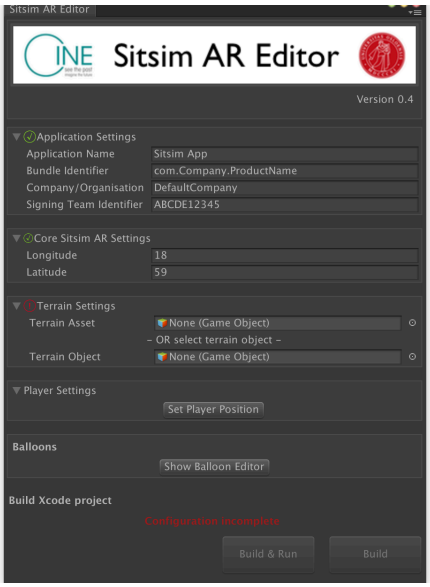
Application Settings

The editor window will now show a section called “Application Settings”, configure this as follows:

- Application Name
  - This is the name the installed application will display on the device.
- Bundle Identifier
  - This must be registered with Apple and must be unique to your application, see the Apple developer portal for more information on how to get your own bundle identifier.
- Signing Team Identifier
  - The Signing Team Identifier can be found on the Apple developer portal under “Membership” in your account. It is necessary to configure this in order to build to a device.

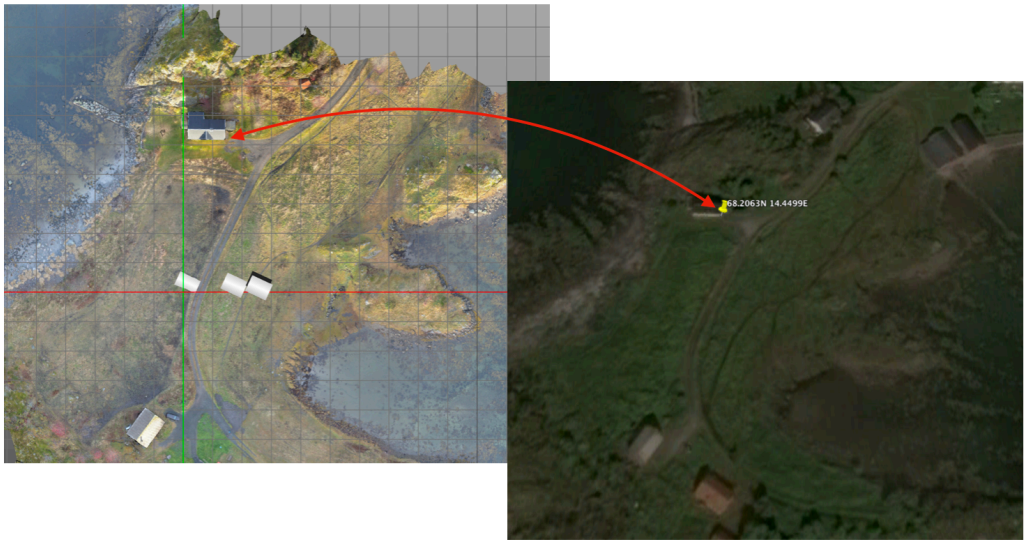
Press “Add Core Sitsim AR Objects” and continue (you can go back and change your application settings at any time).

The editor window now show the rest of the configuration settings, we’ll go through them one by one in the following sections.



Core Sitsim AR Settings

In this section, you configure the sitsim applications real world coordinates in latitude and longitude (decimal). You should enter a coordinate that matches a specific point in your terrain model (configured in the next section). The easiest way to find a good coordinate is to look at your terrain model (with or without textures) and use a map tool, e.g Google Maps, to find a spot that's easy to identify in both.
Things to bear in mind when selecting a coordinate:



- Use a coordinate with as high precision as possible
- Find a coordinate close to the center of the area of interest of your sitsim application (usually near the center of the terrain model)

Note. Remember that east/north are positive longitude/latitude values and west/south are negative.

Terrain Settings

This section is about adding and configuring the basic surroundings in the sitsim application, the terrain. Adding terrain can be done in two ways; either add the terrain to the scene manually or let the editor do it. Either way, the first step is to import the necessary assets into the project (terrain model and texture(s)). With the terrain assets in the project you follow one of the procedures below (A or B).

A. Add terrain manually
First, make sure that your terrain model has a collider component on all surfaces on which the end user is supposed to be able to "walk", without a collider the user will fall through the terrain).

1. Drag your terrain model into the scene or hierarchy window (make sure to place it under the object "Visual Root" in the hierarchy window)
2. In the editor window, select your terrain object in the scene as "Terrain Object"

B. Add terrain using the Sitsim AR Editor

1. In the editor window, select your terrain model asset as "Terrain Asset"
2. If the editor can't detect a collider component on the terrain model asset, it will show the "Add collider" button. Pressing this button will add a collider to the asset
3. Press the "Add terrain to scene" button, this will add an object called "Terrain" under the object "Visual Root" in the hierarchy window

Note. The rest of the terrain configuration assumes that the terrain now lies "flat" in the XZ-plane, if it's not, then you will have to modify your terrain model asset and re-import it.

Now, with a terrain object in the scene, the next step is to make sure that the terrain has the correct size and orientation. There are some basic pieces of information you need to know as a user of Sitsim AR Editor:

- One distance unit in Unity corresponds to one meter
- Unity's positive Y-axis is "up" (i.e reverse gravitational vector)
- Unity's positive Z-axis is aligned with "north"

The editor has some basic tools to help you to scale and rotate the terrain to match the real world. Press the"Add reference object to resize or rotate" button, and a window with some instructions will appear. Read the instructions, press "Ok" and continue. The perspective of the scene will now change to top-down (along negative Y-axis) and use an isometric projection, you may have to move and/or change the zoom level in order to get a good overview of the scene (see Unity documentation, https://docs.unity3d.com/Manual/SceneViewNavigation.html), but keep the isometric top-down view.

You can now see an arrow with an 'N' in the scene, this is called the "Comparison object" and you can see its properties in the editor window and change them using the sliders or the value fields:

- Length (m)
  - The length (real world corresponding) of the arrow from tip to base. The tip of the arrow has a fixed position when changing length

- Orientation
  - Compass orientation of the arrow (0-360 degrees), i.e a value of 90 means that the arrow points eastward. The arrow rotates around the tip and when pointing to the north (0 degrees), the 'N' is visible in the scene view
- Altitude (m)
  - Use this to place the arrow so that it is visible on or above your terrain.

The "Select" button is used to select the "Comparison object" so that you are able to move it around in the scene. The "Remove" button will remove the "Comparison object" and leave the "Add reference object to resize or rotate" mode (keeping the terrain properties configured below).

The "Terrain Settings" section also contains setting for "Terrain orientation and scale":
- Scale
  - This is the scale relative to the terrain asset, i.e the object is scaled up or down before being rendered
- Orientation
  - This is actually the orientation of the object "Visual Root" (which contains the terrain), i.e when using the slider (or value field) to change the orientation, all objects under "Visual Root" will follow the rotation.

The "Confirm scale" button will apply the new scale to the terrain asset. Doing so will change the property back to one, but the terrain scale is still changed (remember, the scale value is relative to the asset). It is recommended (for performance reasons) to use "Confirm scale", but it is not completely necessary.

The recommended way of configuring the terrain is by the following process:

1. Make sure the terrain has the correct orientation, depending on the source this is usually already correct or 180 degrees off
2. Scale
   1. Find two points in the model and two corresponding points in the real world (separated as much as possible).
   2. Measure the distance in the real world (or using a map)
   3. Place the tip of the arrow at one point and its base at the other (use the "Length" and "Orientation" of the "Comparison Object" together with "Select" and moving the arrow in the scene)
   4. If everything is correct, the length of the arrow is the same as the real-world length measured in step 2 above (an error of 5-10% is usually acceptable). If it's too far off, calculate the factor Length:Measured distance and use as scale and repeat from step 3.
   5. When satisfied, press "Confirm scale" and then "Remove" on "Comparison object"

The last step in "Terrain Settings" is to place the origin. The origin is a position in the model that corresponds to the coordinate defined in "Core Sitsim AR Settings" above. Press the "Place origin" button and a window with some instructions will appear. Read the instructions, press "Ok" and continue.

The scene will now be presented top-down in isometric projection, move the origin object (you can always select it by pressing “Select and Focus” in the editor window) to the desired position. When you are satisfied with the position, press “Confirm” to align the terrain (and all other objects under “Visual Root”) to the new origin. You can also press “Abort” to leave “Place origin”, this will remove the origin object and leave the terrain and visual objects as they were.

Player Settings

In this section you set the starting position of the player/user. Even though the app will follow the users movements in the real world, the application needs a starting position. Pressing the “Set Player Position” button will select the player object and present the scene top-down in isometric projection. Move the player to the desired position and continue.

Balloon Editor

Information balloons (spatially positioned hypertext links), conveying geo referenced information in an easy to use and natural manner are an important part of a sitsim application. A balloon is visually placed in the terrain and links to a web site, PDF, a 3D object or even a complex flow of events, and the user can easily access the link by tapping on the balloon. The Balloon Editor currently supports four kinds of links:

- Detail Object
  - Drag a 3D object from the scene or the project asset catalog. When the user taps the balloon, the 3D object will be presented on a black background. The user can rotate and zoom in on the object.
- Detail Object (Clean)
  - Drag a 3D object from the scene or the project asset catalog. When the user taps the balloon, the 3D object will be presented on a black background. The user can rotate and zoom in on the object.
- Detail Audio
  - Drag an audio clip from the project asset catalog. When the user taps the balloon, the audio clip will start playing.
- Detail Web
  - Enter a valid URL (e.g https://www.google.com). When the user taps the balloon, the sitsim application will open a built in browser window and navigate to the URL provided.
- UnityEvent “On Activate()”
  - Connect an object instance and event method to be called when the balloon is activated. Multiple event receivers can be added.

Follow these steps to create a balloon:

1. In the Sitsim AR Editor window, tap the “Show Balloon Editor” button. This will open/show the Balloon Editor window.
2. In the Balloon Editor window, tap the “Add balloon” button. This will add a balloon with default configuration to the scene.
3. Give the balloon a name (not visible to the end user)
4. Give the balloon a label (short text/title on the balloon, visible to the end user)

5. Provide a detail type
    1. Drag a 3D object from the scene or project assets to “Detail Object”
    2. Drag a 3D object from the scene or project assets to “Detail Object (Clean)”
    3. Enter URL into “Detail Web”
    4. Provide event receivers
6. Pressing the “Select” button will select the ballon object, and present the scene top-down in isometric projection. Move the balloon to the desired position and continue.

Note. By providing both a 3D detail object (“Detail Object” **OR** “Detail Object (Clean)”) and detail web URL, the user will be presented with the 3D object and have the option to press a “read more” button which will open the browser at the URL provided. Event receivers will always be activated, independent of other details. Also, detail audio can be used in combination with 3D detail object (audio will start playing at the same time as the 3D object is presented) and detail web URL (optional to show URL by pressing “read more” button).

Build Xcode project

With all the previous steps completed, we’re ready to build the Xcode project. There are two ways to build the project

- “Build & Run”
  This option will build the Xcode project, open Xcode, build the project in Xcode, install the application onto an attached iOS device and run the sitsim application
- “Build”
  This option will only build the Xcode project and show the result in a Finder window.

Tips and Tricks

Add fences

Sitsim applications currently rely entirely on the device’s GPS to 100% when it comes to positioning the user. The GPS is not very accurate, especially when the application is started. The GPS may provide locations that are off the terrain that has been added to

the Sitsim, this will send the player object over the edge of the terrain and into the abyss, from which the player object is not recoverable.

In order to avoid this, we recommend adding invisible “fences” around the area of interest. This is easily done by adding simple 3D plane objects, facing inwards towards the terrain and making sure they’re inside the edges of the terrain. Make them high enough so that there is no risk of the player “jumping” over them. Disable the fence object’s mesh renderer to make them invisible.

Multiple Scenes

When building the application from the editor window, only the current scene is included in the build. It’s also possible to perform the build from Unity’s Build Settings window, and there configure it to include multiple scenes, both Sitsim scenes and others.

Drop the crutches

The Sitsim AR Editor provides a simplified and streamlined process for creating a sitsim application. If you feel bold, don’t hesitate to go beyond the editor and dig into the project and what can be done with Unity in general and the Sitsim framework specifically. One note though, depending on the changes made outside the Sitsim AR Editor, it might stop functioning as expected, you will be on your own and have to use Unity directly to configure your project if this happens. We advise using a SCM system to allow you to reverse any changes.

Common Issues

Editor window out of sync

In certain circumstances, the Sitsim AR Editor window (and Balloon Editor window) may loose synchronisation with the project and the scene. If you suspect that this has happened, close the windows and reopen them again.

Sitsim application crashes at start

This can happen for many reasons, here are some common issues and how to solve them.

- If the Xcode console outputs “Symbol not found” and references to “libswiftCore.dylib”, “SpatialManagerCore.framework” or “TrackerNGCore.framework”, the most probable cause is that wrong version of Xcode and/or Swift being used. Make sure that Xcode 10.x and Swift 4.2 is used.
- If the Xcode console outputs “NSInternalInconsistencyException” and “_rootView should be inited at this point” or “UIApplicationInvalidInterfaceOrientation”, the application is configured with illegal device configuration. In Unity Player Settings, make sure that “Default Orientation” is set to “Landscape Left.

Xcode Provisioning profiles

If Xcode has problems signing and/or installing your sitsim application, make sure that you have entered the correct “Signing Team Identifier” (TeamID in the Apple developer portal) and “Bundle Identifier” in the Sitsim AR Editor window. The editor relies on automatic code signing, if the two parameters are correct, it should just work.

If you still experience problems, try creating a simple application directly in Xcode and solve any issues there. It is also possible to make adjustments to the Sitsim project in Xcode, but any changes made in Xcode are lost with next build from Unity.