# AR Navigation

## Version 3.0.0

# Contents

# 1. Introduction

AR Navigation allows creating apps that use AR to navigate within a given target area. It's being developed in two variants: cross-platform (iOS/ Android) and native (at this time, iOS only).

## 1.1 Features

AR Navigation features: automatic calculation of the optimal route using two types of algorithms; dynamic 3D minimap that shows user current location and rotation; intuitive UI that shows user destination, remained distance, and has a list of quick buttons for predefined destinations; floating icons to highlight possible points of interest; custom markers that can snap to a detected surface; QRCode detection.

## 1.2 Algorithms

AR Navigation uses two types of algorithms for pathfinding:

**1. Regular** (Dijkstra algorithm) is the default algorithm used to find the optimal path between a starting vertex and a target vertex. The algorithm will "visit" all existing vertices on the given graph to find the shortest possible path.

*The regular algorithm should be used when your given target area is a graph (there are always multiple possible ways to reach a vertex) and/or it contains non-unique vertices (vertices with the same ID, for example, you want to allow users to search for a nearest exit, and there are multiple exits in the building).*

**2. Fast** is the supplementary algorithm used to find the optimal path between a starting vertex and a target vertex. The algorithm will start "visiting" existing vertices on the given graph to find the shortest possible path, but instead of "visiting" every possible vertices, it will terminate immediately after a vertex with the given ID is found.

*The fast algorithm should be used only when your given target area is a tree (there is always no more than one possible way to reach a vertex) and the target area doesn't contain any non-unique vertices. Using the fast algorithm on the graph target area and/or on the area with non-unique vertices may lead to unpredictable and non-optimal pathfinding.*

# 2. Requirements

The scale of each map segment must match its real-world scale with the smallest possible error. Even very small mismatches in scale will lead to a big cascading increase in the error level depending on the distance traveled.

## 2.1 Requirements for iOS

The device must support ARKit. Lowest possible deployment target is iOS 12.0, but latest iOS 14.0 is strongly recommended. For more information on ARKit requirements refer to Apple's official documentation.

## 2.2 Requirements for Android

The device must support ARCore. For more information on ARCore supported devices and requirements refer to Google's official documentation.

# 3. How it works

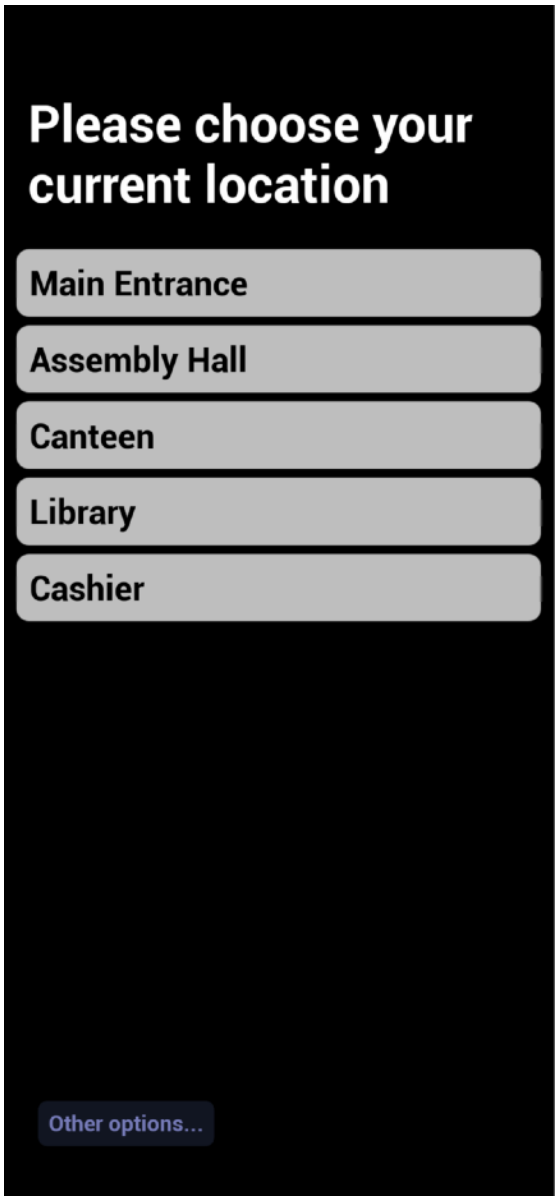
## 3.1 App Launch

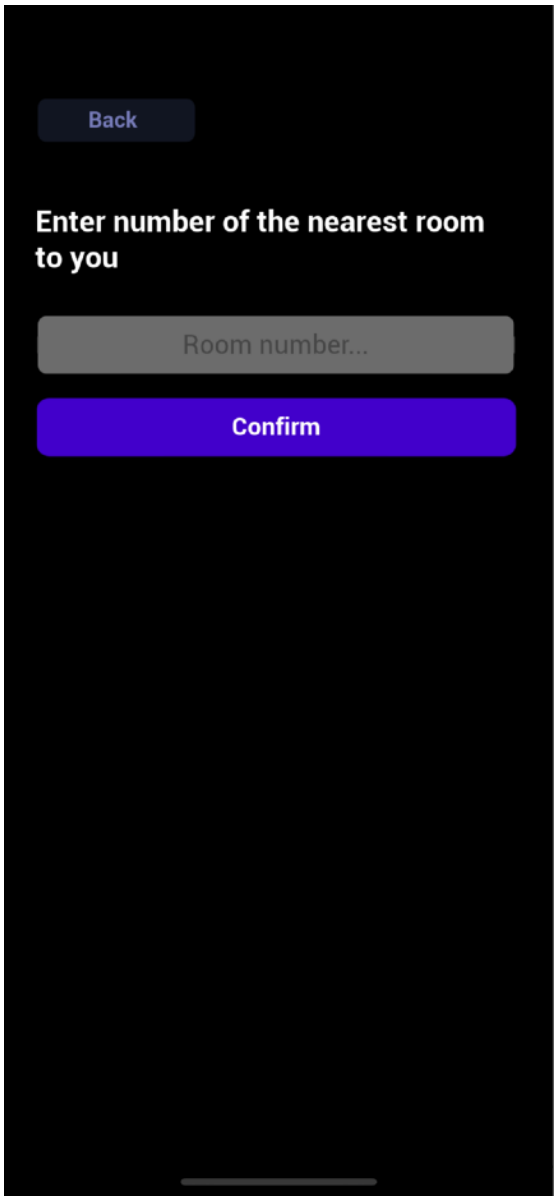
When the user launches the app, they will be prompted to scan the QR code used to download the app. This is needed to immediately set their position in the virtual world. Using such QR code is the most convenient way to ensure the accuracy of the user position.

*Also, it's possible to use any image which can fit on an A4 sheet instead of a QR code. For more information on requirements for the reference images refer to Apple's official documentation.*

## 3.2 Predefined Locations

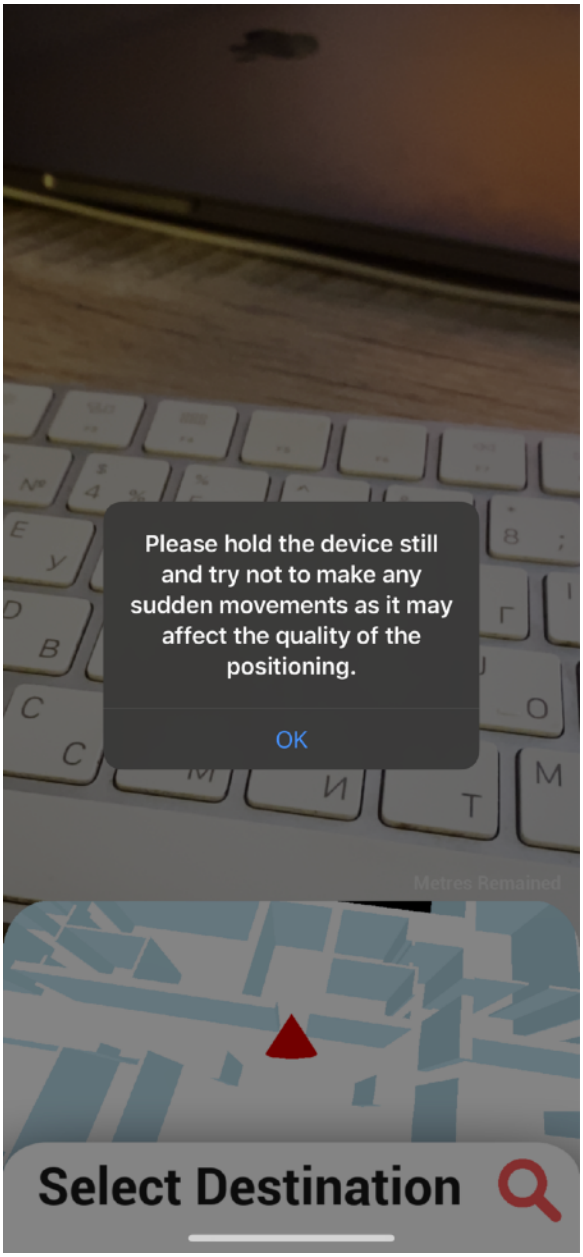
The user can select their position from a predefined list. In the provided demo level, this list contains of 5 most popular positions for the chosen target area. Buttons are embedded into a scroll box, so it’s possible to add as many predefined positions as you want. Also, the user can select their position using manual input. In this scenario, the user position will be set to a vertex with an ID that they have entered (the user will be virtually teleported). To make this process easy and convenient for the user, set vertices’ IDs to some friendly name (for example "Canteen" and "Assembly Hall”).



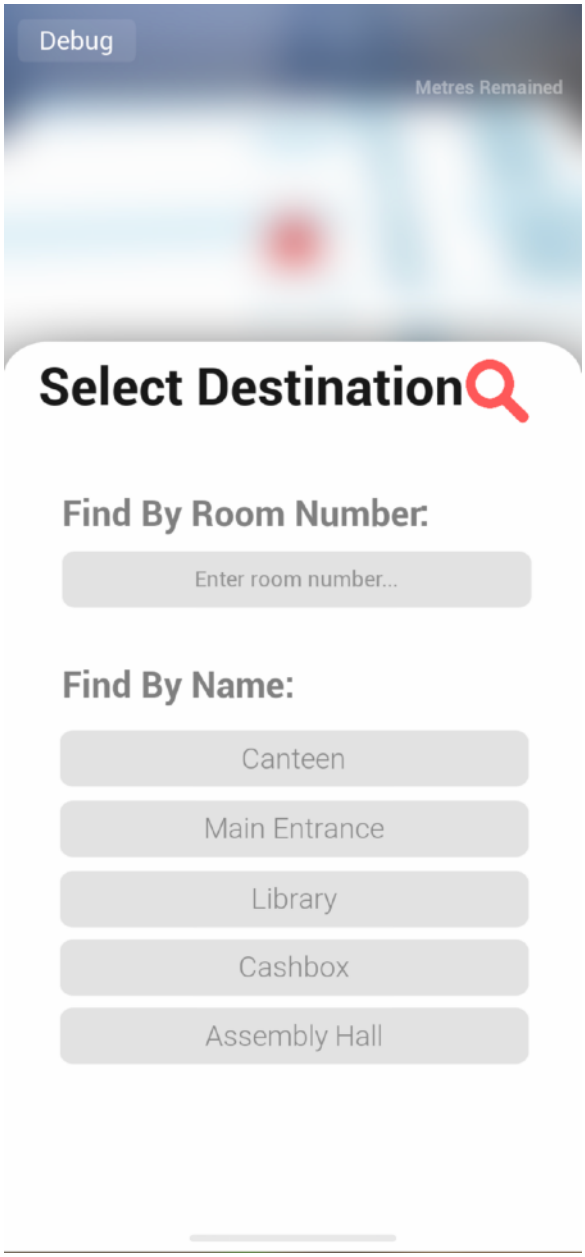
## 3.3 Start

After the initial position is set, the app starts the AR session and prompts the user with a small message regarding how to use the app. Now the app and its minimap will react to user movement seamlessly using visual-inertial odometry provided by ARKit and ARCore runtimes.



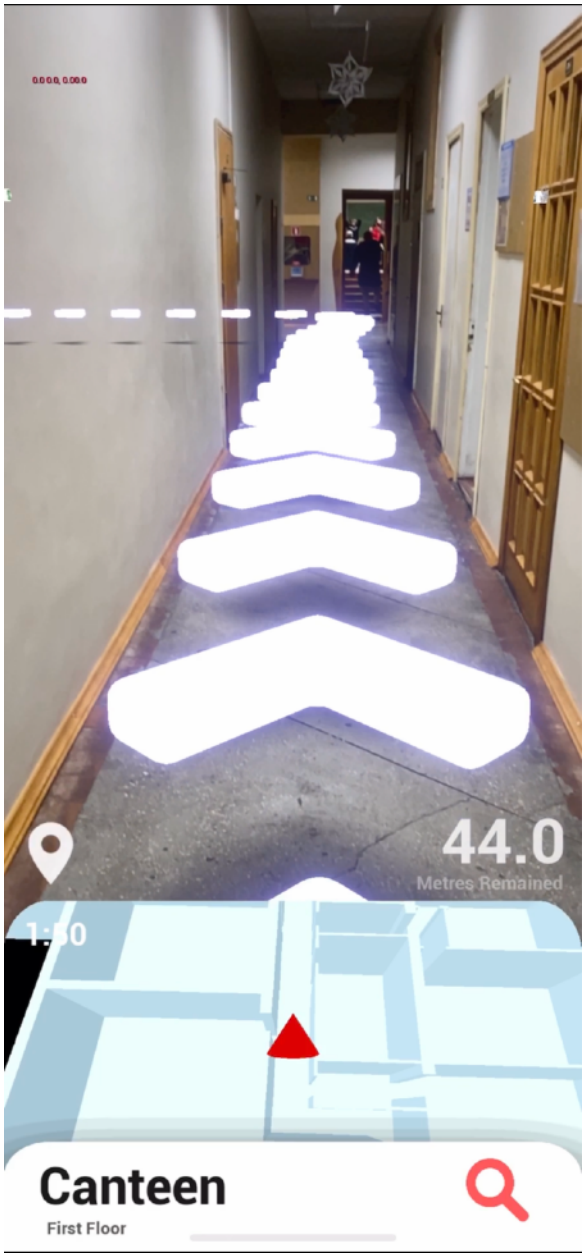
## 3.4 Selecting a Destination

The user can choose a destination point from a predefined menu or enter it manually. The app searches for a vertex with the given ID and marks it as a destination point.



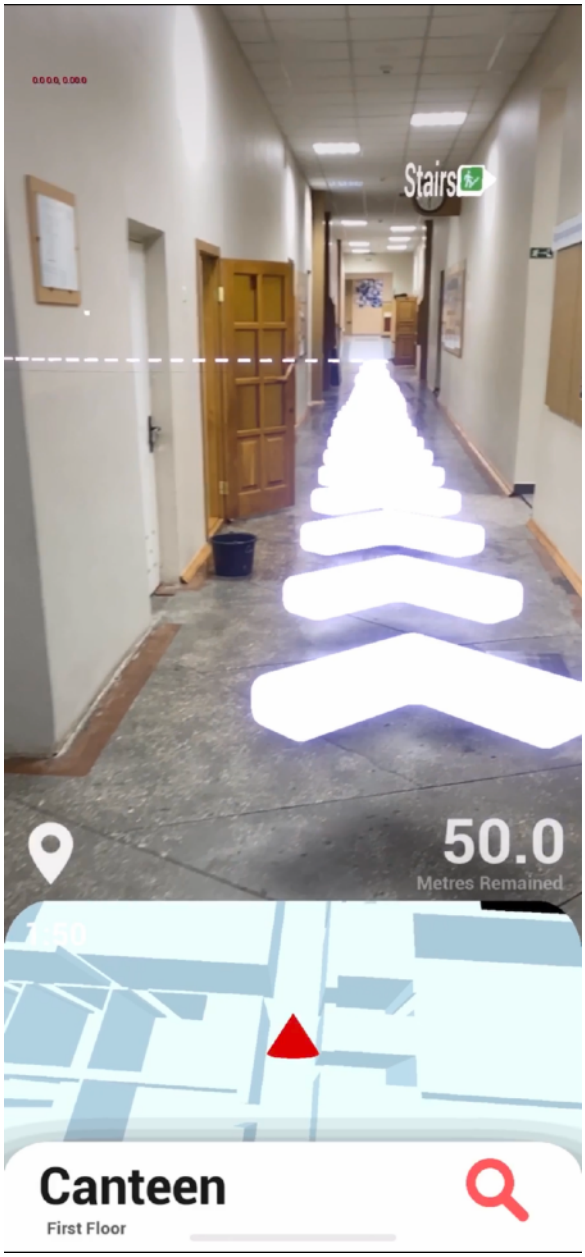
## 3.5 Pathfinding

The navigation path from the user current location to the destination point will be calculated in real-time (using regular or fast algorithm) and displayed as a spline with animated arrows. Also, the app will display and update remained distance in real-time.



## 3.6 Minimap

During the whole period of navigation, the app will show user location and rotation relative to true north on the dynamic 3D minimap.



## 3.7 Markers

The user can tap on any detected surface to place a custom marker. The marker will snap to the surface, and will show its given name and distance to the user.

## 4. Setting up

1. Place an ARWorldBase instance anywhere on the map, it will serve as an origin for the target area.

2. In ARWorldBase properties enter rotation of the target area relative to the world true north.

3. Place one or multiple instances of MapSegment, they will serve as containers that represent the properties of each segment. Multiple segments are used to divide the map by floors or by some small sub-areas. Attach each MapSegment to the ARWorldBase.

4. For each MapSegment assign its floor map (used only for debugging purposes, will be hidden in the app) and a static mesh (3D model of your building) that represents the segment's map in the real world.

5. Adjust the scale, position, rotation and height of the segment to match its real-world values (it's very important!). The scale and position of each segment must match its real-world scale and position with the smallest possible error. Even very small mismatches in scale will lead to a big cascading increase in the error level.

6. Build a navigation graph by placing Vertex instances on your map segments and connecting them using BilateralConnections or LateralConnections arrays.

7. Give an ID to every Vertex that represents a destination point (user will be able to perform a search using this ID). Vertices that are used only to connect other vertices should have their ID field empty. If a vertex is non-unique (there are multiple vertices on the map with the same ID) set its “IsUnique” property to false.

8. Attach all placed vertices to the map segment they belong to.

9. *[Optional]* Place a FloatingIcon instance where there is a possible point of interest. Enter its name and a material containing its icon. Attach all placed FloatingIcon instances to the map segment they belong to.

10. *[Optional]* Place all Room instances that represent each room. Set their size or leave it zero for automatic size detection. Rooms are not required and are not used by default. The main reason why you can use them is to get user location in real-time by using the room as a trigger. Attach all placed Room instances to the map segment they belong to.

11. Place Configuration, 2DImageProvider and NavigationSpline classes anywhere on the map.

12. In the Configuration class properties fill the InitialPositionPage and MainUIText arrays with your possible user's initial positions and available destinations for the main UI. Select the appropriate pathfinding algorithm in the “Path Building Type” property.

13. Replace the default QR code texture that comes with the project with your own. Put a Vertex instance where your QR code is located in the real world and set its ID to “QRCode".

# 5. World mapping API

## ARWorldBase

The supplementary class that is used as a root to all MapSegment classes and also stores references to them. Every new MapSegment should be attached to the ARWorldBase class.

*Usage: Place it anywhere on the map.*

### Public properties

- WorldTrueNorthAngel: Float - Stores rotation shift of the target area relative to the true north. You can get the rotation shift of your target area using Google Maps and calculating the angle of rotation relative to a latitude(any vertical line). The ARWorldBase class will rotate to the given angle with a short delay when the application starts. The delay is needed to allow the attached MapSegments and their Rooms to calculate their sizes and positions using the provided model of the target area (Static Mesh). The default value is zero.

- MapSegments: Array of MapSegment - Supplementary array that contains references to its segments in case you need it.

## MapSegment

The class that is used to define segments of the target area. All FloatingIcon, Room, and Vertex instances that are located in this segment should be attached to it.

*Usage: Place it anywhere on the map and attach it to the ARWorldBase class.*

**Public properties**

- Floor: Integer - Stores floor number of the segment. The default value is zero.

- FloorDisplayedText: Text(Localized String) - Stores a localizable text that describes the segment's floor. Keep in mind that segments can be used not only to differentiate floors but also to divide a complex map into more small pieces in case the target area is big enough. The default value is empty.

- SegmentMesh: StaticMesh - Fundamental mesh (3D model) that defines the area of the segment. The default value is null.

- SegmentOffset: Vector2D - Stores the X,Y offset of the segment's mesh relative to the segment's origin. The default value is (0.0, 0.0).

- SegmentHeight: Float - Stores the height of the segment's mesh in meters. For this to work, the segment's mesh should be 1 meter in height. Alternatively, you can create a mesh with any height and set SegmentHeight to 1.0. The SegmentHeight will also be used to determine if the user is inside the segment to show or hide it from the minimap. The default value is 1.0.

- SegmentScale: Float - Stores all axes scale multiplier of the segment’s mesh. The default value is 1.0.

- SegmentMapMaterial: Material - A supplementary material that is used to show the floor plan. This is only needed to simplify the vertex and room arrangement on the segment. The material is only shown in the editor and is hidden in the app. The default value is null.

- SegmentMapScale: Float - Stores the X,Y scale of the SegmentMapMaterial. Use this to adjust the floor plan to match its real-world scale. The default value is 200.0.

## Vertex

The fundamental class that represents a single vertex used for building navigation paths in the target area.

*Usage: Place it anywhere on the MapSegment and attach it to this segment.*

**Public properties**

- BilateralConnections: Array of Vertex - An array of Vertex instances connected to this vertex bilaterally (Stair, Corridor, etc). The navigation path will be drawn in both directions.

- LateralConnections: Array of Vertex - An array of Vertex instances connected to this vertex laterally (Turnstile, Escalator, etc). The navigation path will be drawn only in one direction.

- ID: String - A unique identifier for this Vertex used for search algorithms. The default value is empty.

- IsUnique: Bool - Indicates if the vertex has unique ID on the entire map. The default value is true.

## FloatingIcon

The class represents a floating icon that can be used to highlight points of interest.

*Usage: Place it anywhere on the MapSegment and attach it to the segment.*

**Public properties**

- IconMaterial: Material - A material that contains the image for the floating icon. It can include a static picture (like in the demo) or anything else. The default value is null.

- Text: Text(Localized String) - A localizable text that describes this point of interest. Will be displayed on both sides of the floating icon.

## Room

The class that represents a single room in the target area. It can be literally a room or just a small sub-area.

*Usage: Place it anywhere on the map segment and attach it to the segment.*

**Public properties**

- Size: Vector2D - Stores size of the room. Adjustable in design-time. If set to zero the room will set its size automatically depending on its surroundings. The default value is zero.

- Height: Float - Defines the height of the room. The default value is 3.0.

- Title: Text(Localized String) - Stores a localizable text that describes the room. The default value is "No Title”.

- Floor: Text(Localized String) - Stores a localizable text that describes the floor of the room. The default value is "First Floor”.

- DebugMaterial: Material - Stores a material that will be applied to the room's mesh. For debugging purposes only, it will be hidden in app. The default value is null.

- Type: Enum - Stores a type of the room. Can be "Room", "Corridor", "Staircase" or "Other". The default value is “Room".

## Configuration

The supplementary class that stores system configuration for this map.

*Usage: Place it anywhere on the map. Use only one instance.*

**Public properties**

- Path Building Type: Enum - Contains the type of the pathfinding algorithm that should be used to find the optimal path between a starting vertex and a target vertex. The algorithm types are described in section 1.2. The default value is Regular.

- InitialPositionPage: Struct - Contains a struct "Initial Positions" that stores a list of user's possible initial positions. Each InitialPosition stores a "Text"(Localizable friendly name that will be displayed in the list) and "TeleportToVertexTag" (Vertex ID which coordinates will be used to virtually teleport the user to that vertex). The default values are missing and are implemented directly in the demo level.

- MainUIText: Struct - Contains a struct "QuickButtons" that stores a list of user's possible destinations which can be selected by the user from the main UI. Each QuickButton contains Text"(Localizable friendly name that will be displayed in the list), "Floor" (Localizable friendly name of the floor that will be displayed in the main UI under the destination) and "DestinationTag" (destination Vertex ID).

## 6. Supplementary API

### 2DImageProvider

The class is responsible for rendering the 2D minimap image.

*Usage: Place it anywhere on the map. Use only one instance. Do not edit or modify it somehow.*

### NavigationPathSpline

The class which is responsible for building a navigation path from the user location to the destination point using arrows provided by the Arrow class.

*Usage: Place it anywhere on the map. Use only one instance. Do not edit or modify it somehow.*

**Public properties**

- Spacing: Float – Distance between arrows on the navigation path.

### Marker

The class which represents a single custom marker.

*Usage: Not intended for direct usage. Will be instantiated automatically by the character controller.*

### Arrow

The class which represents a single arrow used in NavigationPathSpline .

*Usage: Not intended for direct usage. Will be instantiated automatically by the NavigationPathSpline class.*

## 6.1 Structs

**CustomMarker**
The struct is storing name and location data about one single custom marker placed on the map.

**InitialPosition**
The struct is storing displayed text and a vertex ID for one single predefined initial position.

**InitialPositionPage**
The struct is storing an array of InitialPosition structs.

**QuickDestinationButton**
The struct is storing displayed name, displayed floor and a vertex ID for one single predefined destination.

**MainUIButtons**
The struct is storing an array of QuickDestinationButton structs.

## 6.2 Data Assets

### QR Codes

**InitialPositionQRCode**
Contains ARCandidateImage(reference image) data, such as: texture, friendly name, width, height and orientation.

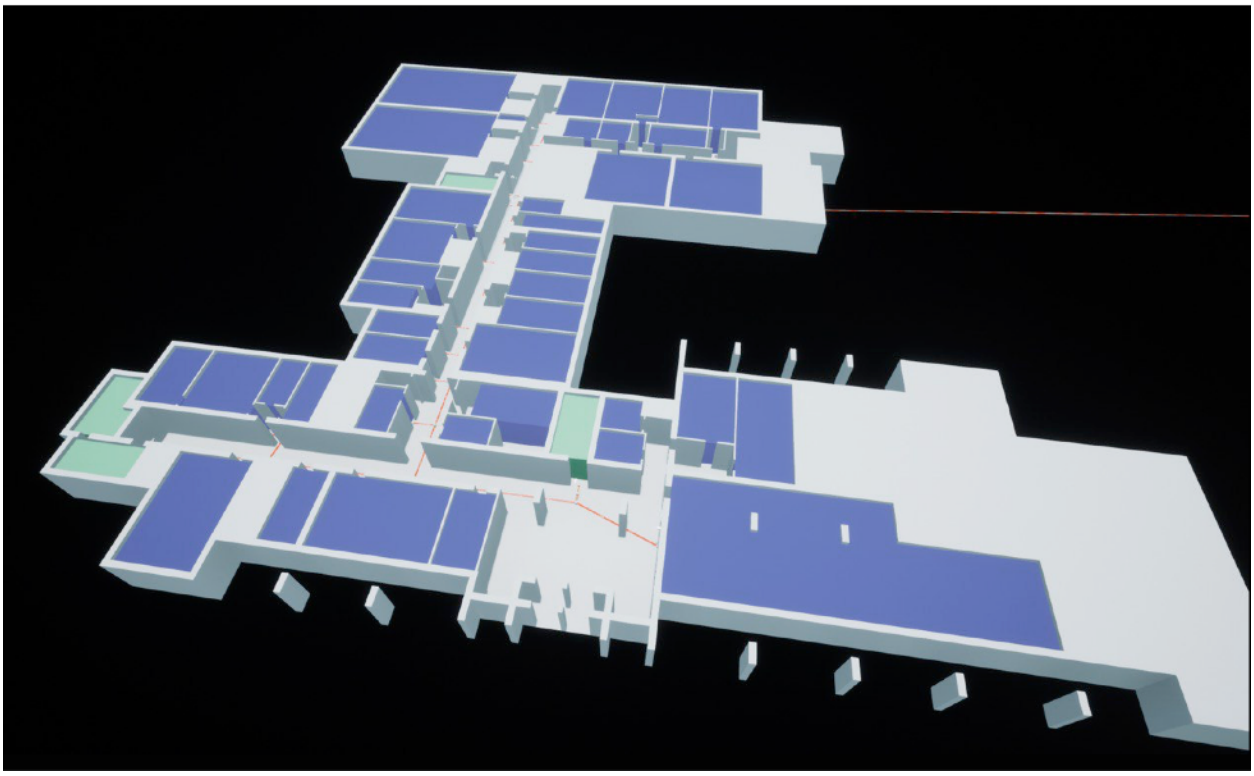
### AR Sessions

**ARSessionInitial**
Contains AR session configuration used for initial positioning. World alignment is Gravity and Heading, plane detection is disabled, one reference image (QR code) is set, everything else is at its default values.

**ARSessionMain**
Contains AR session configuration used for runtime positioning. World alignment is Gravity and Heading, plane detection is enabled for horizontal and vertical planes, no reference images are set, everything else is at its default values.

## 7. Demo

AR Navigation comes with a demo level that represents one floor with an area of 80mx60m. It contains one map segment that represents one floor, a graph built using multiple vertices, several points of interest icons, and all rooms set up.



The demo was tested on multiple iOS devices by multiple users, the feedback showed a high level of interest in the product and its potential.

## 8. Debug

To toggle debug mode tap on the "Debug" button located in the top left corner. The debug mode will display all vertices and their axes. This will allow to test how the virtual world matches the real one.