# General

Archi-VR is an application that that enables to prepare a presentation/preview of architectural projects in for presentation in virtual Reality (VR) on the Oculus Quest device. The application can be built with a set of Archi-VR projects.

The application has the following functionality:

* Browse through a series of ArchiVR projects.
* Open an ArchiVR project as the active one.
* Navigate the active project
* Create/Edit/Delete content in the active project:
  + Prop
    - FromModel
      * Terras
      * Kitchen
        + Cooking Plates (Make procedural (option)?)
        + Dampkap (Make procedural (option)?)
        + MessenSet
        + Toaster
        + Herbs
        + Koffiezet
        + Fruitmand
        + Tablet
      * Living
        + TV
      * Toilet
        + WC-Pot
        + WC-Rol
      * Laundry
        + DroogRek
        + DroogMand
      * ...
    - Procedural
      * Kast
        + Gewoon

Pick bottom Plane

Define back plane by picking a point

Use this point as upper right point back plane

Define the other (right/left) side on the back plane by picking another point on the back plane.

Define front face using picked point on bottom plane

* + - * + InbouwKast

Pick planes by picking points in the in the following order:

Bottom plane

Top plane

Left plane

Right plane

Back plane

Front plane

* + - * TV-Meubel
      * Tablet
      * Deur?
        + Gewoon
        + Shuifdeur
      * ...
  + Lighting
    - LightGroup
    - Light
      * Pendant
        + Parameters

CordLength

CordMaterial

BodyType

Cylinder

Cube

* + - * CubeSpot
        + Parameters

Size

Color

LightColor

LightRange

* + - * + 2 Spots: Up, Down
      * Cylinder
        + Parameter

Radius

* + - * + 1 Point Light
      * Cube
        + Parameter

Size

* + - * + 1 Point Light
      * TL
        + Parameters

Length

Point **Color**

* + - * + 1 Point Light
      * RectangularSpot
        + Size2D
        + BorderRadius
        + Height
        + LightColor
      * CircularSpot
        + SpotRadius
        + BorderRadius
        + Height
        + LightColor

## Startup

The application starts up as follows:

* Active project: First project
* Activted POI: First POI in first project
* [User mode](#_User_mode): [Single user](#_Single-user_mode)
* Initial State
  + State: Default
    - [Immersion mode](#_Immersion_mode): Maquette

## Operation mode

Either ‘Present’ or ‘Edit’.

Present: The active project is presented to the users.

Edit: The active project is edited as preparation of its presentation to the users.

## Network mode

In operation mode Edit, the application is always in single -user mode (NetworkMode Standalone).

In operation mode Present, the application can be run in different network modes:

* [Single-user mode](#_Single-user_mode)
  + NetworkMode Standalone
* [Multi-user mode](#_Multi-user_mode)
  + NetworkMode Client
    - Running a client
  + NetworkMode Server
    - Running a client and a server

### Single-user mode

The presentation is experienced by a single user. The application runs on the local devices without network connection.

### Multi-user mode

In this mode, multiple users can take part of the presentation simultaneously.

The application runs either as a server or a client in a client-server network relation with other devices running Archi-VR.

In order to start a multi-user session:

* All devices must be in the same IP subnet.
* The first device must start the multi-user session by starting as a server
* Any arbitrary number of devices can then join the multi-user session by starting a s a client.

To start as server, open the ‘Network’ menu, then click ‘Start multi-user session’.

To start as server, open the ‘Network’ menu, then click ‘Join multi-user session’.

For now, only the server can execute commands:

* activate project
* teleport to POI
* toggle immersion mode
* Manipulate model in Maquette mode

Server and clients can all move around in the tracking space, and point at things using the laser pointer.

## Immersion mode

The application allows to present a preview of architectural designs in 2 immersion modes:

* Walkthrough mode
* Maquette mode

### Walkthrough mode

The model is represented in Virtual Reality at full scale. The user can teleport to Points-Of-Interest included in the active project.

#### Commands

In Walkthrough immersion mode, the following commands are available:

* **Toggle immersion mode:** Transitions into Maquette immersion mode.
* **Next project:** Activate next project. *Only available if multiple projects included in build.*
* **Next project:** Activate previous project. *Only available if multiple projects included in build.*
* **Next POI:** Teleport to the next POI. *Only available if multiple POI in active project.*
* **Prev POI**: Teleport to the previous POI. *Only available if multiple POI in active project.*
* **Toggle menu**: Opens/Closes the HUD menu.
* **Move**: Move forward/backward, strafe left/right on the horizontal plane.
* **Toggle On/Off vertical translation:** Toggle On/Off vertical translation:
* **Vertical translation:** Translate up/down along vertical axis.
* **Rotate tracking space:** Rotate tracking space around vertical axis.
* **Enable laser pointer:** Point at things with a ‘laser pointer’ representation.

|  |  |
| --- | --- |
| **Command** | **Touch** |
| Toggle immersion mode | Left index trigger |
| Next POI | X Button |
| Prev POI | Y Button |
| Next project | A Button |
| Activate laser pointer | Hold right index trigger pressed |

#### User comfort while teleporting

A naïve teleportation implementation would be to relocate the user instantaneously to exactly the new POI location.

This implementation would be problematic because of the following issues:

* Disorientation of the user due to abrupt relocation.
* Misalignment between ‘intended and effective freedom of movement volume around the POI

In order to mitigate the above issues, the teleportation system has been enhanced as follows:

##### Smooth teleportation

When teleporting, the screen first fades out to black while still in the old POI location, then fades in again at the new POI location.

This smooth transitioning feels much more natural to the user, and thereby mitigates the **user disorientation issue**.

##### Teleportation area

When teleporting in single-user mode, we want the following to be met:

* **The user should be relocated as close to the new POI location as possible.** This prevents ‘awkward’ relocations, ends up in an uncomfortable location after a teleportation (eg inside a wall, above a staircase, in mid-air,...)
* **After teleportation, the new POI should be aligned with the middle of the physical tracking space.** At design time, POI’s are placed to be in the center of the intended ‘freedom of movement volume’. IE we do not set POI’s close to a corner of a room, but in the middle of the room. This ensures that the user can freely walk through the entire extents of the room.

In order to meet the above requirements, the concept of ‘TeleportArea’ has been devised. The teleport area is a vertically aligned cylindrical volume around the Shared Reference Frame. The user can only teleport if his center eye position is inside the teleport area.

The application has notion of 3 reference frames:

**TRF:**

Headset and controllers report positions defined in TRF. TRF is the location where the headset got its tracking up.

**SRF**

Position of a shared reference frame that was defined by all users measuring the same markers (2 points in consistent order) in the physical playspace. Stored expressed in Tracking space

**POI**

The location of the currently active Point-Of-Interest.

Initially, SRF is aligned with TRF.

When teleporting, the SRF is aligned with the newly activated POI.

When the user tries to teleport to a new POI, he is instructed to return to the teleport area first. The teleport area is represented by 2 semiTransparant vertical cylinders around 1m diameter:

* representing the teleportarea floor area (flat disk)
* Representing the teleportvolume (long cylinder)

TODO: Implement textured/animated version?

TODO:  
The direction to the teleportation area (user groundlevel position-> teleportation area center) is represented just above the info text, by   
 - an ArrowLinerenderer?  
 - Quad with arrow texture

This representation always points to teleportArea as a sort of compass.

When in multi-user user mode, the application can have Colocation enabled or not. Colocation means that multiple VR players share the same play area. In order to achieve collision-free colocation, players need to know their mutual positions.

In order to achieve the therefor needed common Shared Reference System ‘SRF’ can be defined by all players measuring the same 2 points, in a consistent order.

When in multi-user mode and having colocation enabled, the user cannot manipulate the TRF to’Fly() around.

### Maquette mode

In maquette mode, the model is rescaled to 1/25 scale in Virtual reality. This is the usual scale for physical architectural maquettes. When entering this mode, the tracking space is located around the model anchor by default. By default, the model resides at a height of 1m 20 cm, with no rotation.

#### Preview context

In maquette mode, the ‘maquette preview context’ scene is visible. It is a real-scale context in which the maquette is located, in order to make the process of previewing a model feel more natura lto the user(It does not feel natural to float in thin air without something to stand on). On top of that, some info is displayed in the preview context. The preview context consists of the following:

* A platform/room on which the model is centered.
* A 3D text couting ‘KS-architect’
* A 3D tekst couting the name of the project being previewed.

#### Toggle model layer visibility

The user can toggle different parts of the model visible/invisible as follows:

* Touch the part with the left controller, so that it (or it’s outline/bbox) becomes highlighted.
* ‘Click’ the part with the left controller index button.

#### Manipulate model location

The model location can be manipulated using the as follows:

* Left Thumb horizontal: Rotate model around up vector.
* Left Thumb vertical: Translate model along up vector.
* Left Thumb Click: Reset model to its default location (no rotation, y=1m 20cm)

# Movement

In any immersion mode, the position of the tracking space (and hence viewpoint) can be manipulated while colocation is disabled. Movement is defined as follows:

* Move forward: Translate along ‘forward’ viewing direction.
* Move backward: Translate along ‘backward’ viewing direction.
* Move left: Translate along ‘left’ viewing vector
* Move right: Translate along ‘right’ viewing vector
* Move up: Translate along ‘up’ viewing vector
* Move up: Translate along ‘down’ viewing vector

In VR, the controls are mapped to the Right controller as follows:

* Thumb Up -> Forward
* Thumb Down ->Backward
* Thumb Left -> Left
* Thumb Right ->Right
* Index Trigger -> Up
* Hand Trigger ->Down

# Archi-VR project

An Archi-VR project represents a snapshot of an architectural design for a physical construction project.

## Examples

Some examples of ArchiVR projects are:

* ProjectKS046
* ProjectKS003Before
* ProjectKS003 After
* ProjectKS069OptionA
* ProjectKS069OptionB

## Project scene structure

Each Archi-VR project is defined in its own Unity scene.

The structure of an Archi-VR project scene looks as follows:

* Project
  + Model
    - Model Anchor
  + POI
    - ...
    - Terras
    - Living
    - ...
  + Lighting
    - ...
    - KitchenPlafond01Point
    - LivingPlafond02Point
    - ...

### Project GameObject

The root object for the project scene.

### Model GameObject

The game object that contains the project-specifig geometry.

### Model Anchor GameObject

This game object represents the conceptual ‘anchor point’ around which the model will rotate in maquette mode.

### POI GameObject

All point-of interest GameObjects need to be placed directly under this.

### Lighting GameObject

All lights need to be placed under this. Mode needs to be set to ‘Baked’ for all lights when targeting the Ocuslus Quest, in order to achieve acceptable runtime performance.

# Archi-VR build types

## Single-project build

For a single construction project, several project can be prepared and included in a single Archi-VR build for that construction project.

Original/Before  
Cleanup/Teardown  
Construction Phase RuwBouw  
Construction Phase Technieken  
…  
Construction Phase Final

This is called a ‘single project build’

## Portfolio build

Likewise, several Archi-VR projects for different construction projects can be added to a single Archi-VR build.

Project 001  
Project 002

This is called a ‘portfolio build’

Archi-VR Trainstation

* Projects
* Original State
* Proposal A
* Proposal B
* Proposal C

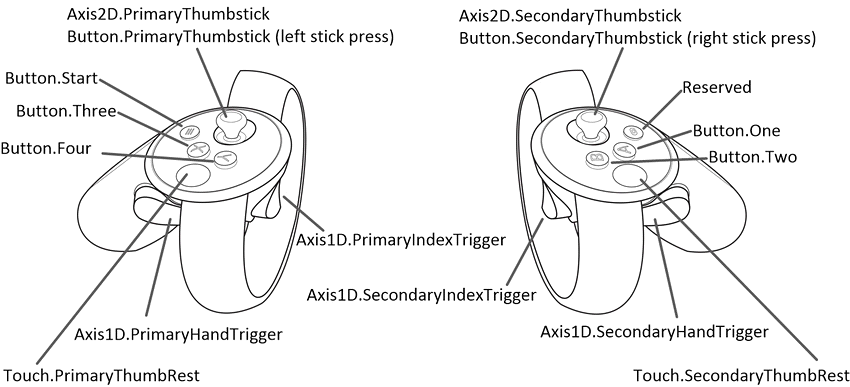
The Archi-VR application has the concept of an ‘active project’ this is the project currently being previewed. The user can toggle between the active project by cycling the projects included in the build.

When switching to a new project, and the project contains a POI with the same name as the active POI, then that POI is automatically activated.

# Input

## Input when running on Oculus quest

When run on an Oculus Quest, all input is perfomed via the Oculus Quest’s Touch Controllers:



Some UI is anchored to the controller representations:

* Controller menu.
* Button mapping labels
  + For all buttons that have afunctionality, a label is shown next to the button on the controller, containing a descriptive text of the command bound to that button.

Startup

Application starts up in Walkthrough in the following way:

* immersion mode ‘Walkthrough’ activated
* first Project (if available) activated
* first POI (if available) activated

Immersion mode

Immersion mode can be toggled using the left index trigger.  
  
Immersion modes:

* Walkthrough mode
* Maquette mode

The application is said to ‘run in the active immersion mode’. The active immersion mode can be toggled between ‘walkthrough’ and ‘maquette’ mode.

Walkthrough mode

The project is represented in real scale in Virtual Reality. The user can toggle between Points-Of-Interest included in the project.

When switching to a new POI, the tracking space is ofsetted to counter-act the offset from the tracking center position. This assures that the user always lands on the intended POI position, thereby avoiding ‘awkward’ teleportations into walls, thin air, or above staircases. Downside is that the volume in which the userr can walk freely, can be malaligned tot he POI – leaving too much room for movement at one side, too little room of movement on the opposing side. When teleporting, the screen first fades to black, then fades in again. This improves the comfort level for the user.

Controls

|  |  |  |
| --- | --- | --- |
| Command | Touch |  |
| Toggle immersion mode | Left index trigger |  |
| Next POI | X Button | Left arrow |
| Prev POI | Y Button | Right arrow |
| Next project | A Button |  |

## Input when running in Unity editor

While running in the Unity editor, the touch controls are not available. Therefore, the inputs are mapped on the keyboard an mouse as follows:

Left Controller:

|  |  |  |
| --- | --- | --- |
| X Button | F1 |  |
| Y Button | F2 |  |
| Index Trigger | R |  |
| Hand Trigger | F |  |
| Thumstick click | A |  |
| Thumbstick left | Q |  |
| Thumbstick Right | D |  |
| Thumbstick Up | Z |  |
| Thumbstick Down | S |  |
| Start Button | F11 |  |

Right Controller:

|  |  |  |
| --- | --- | --- |
| A Button | F3 |  |
| B Button | F4 |  |
| Index Trigger | LMB |  |
| Hand Trigger | ? |  |
| Thumstick click | ? |  |
| Thumbstick left | Left Arrow |  |
| Thumbstick Right | Right Arrow |  |
| Thumbstick Up | Up Arrow |  |
| Thumbstick Down | Down Arrow |  |

The view can be rotated by pressing LShift while moving the mouse.

When running in Unity editor, a visualisation of the Oculus Quest controllers is present at a fixed offset in front of the user viewpoint, in order to aid in debugging controller-attached UI (eg button labels)

# How to prepare a Sketchup model for import in Unity

In order to be suited for import in Unity, a Sketchup model must comply to the rules below:

## Materials with textures

* The material name MUST match the texture file name exactly.
* Material/Texture names MUST NOT contain spaces
* ~~Use Pascal Casing? (eg. VloerTegel512x512, Boom2D256x512) for material/texture names~~
* ~~Use only JPG? (or is png also fine?)~~

## Face orientation

Face orientation must be consistent and correct, ie. there should NOT be any reversed faces in the model.

Having inconsistent face orientation in the model will result in the following problems when using the model in Unity:

* Bad/Incorrect lightmapper results.
* When importing a model with ‘generate back faces’ turned off in the imprt settings, will result in ‘holes’ in the model where reverse faces are present. (because faces will be invisible from their back side).Face orientation

### Tip

In Sketchup, use **Edit>Face Style>Monochrome** to check face orientation.

# How to prepare a Sketchup model for usage in Archi-VR

* Model
  + Surroundings (Street, garden,…)
  + Roof
  + Floor2
    - Walls topside
  + Floor 1
    - Walls topside
  + Groundfloor
    - Walls topside
  + Basement
    - Walls topside
  + …

### Tip

Use outliner to divide model into the entities that should be visiblility-togglable in Maquette mode.

# How to import a Sketchup model into Unity for use in Archi-VR

* Import SKP file as asset.
* Make sure a folder with the needed textures is present next to the model asset. Note: This is necessary for now because of a bug in Unity SKP imprter, that fails to find the textures embedded in the SKP file itself.
* Select the newly imported SKP model asset, and in Inspector pane setup its import settings as follows:
* Disable 'Generate Back Faces'. This will enable backface culling on the model, and result in improved rendering performance.
* Disable 'Import cameras'. (unless you want them imported).
* Enable 'Generate Lightmapping UVs'. This is necessary for storing the UV’s of prebaked lightmaps.

# How to create an ArchiVR project

* Create a Unity project
* Generate a build with the Unity Project and edit it in VR

## Create a Unity project

* Copy-paste-rename an existing ArchiVR unity project
* Open it
* Remove the content of the ‘Model’ node
* (create a new folder under assets/Projects/ProjectName to store the necessary assets fort he ArchiVR project in.
* Import the assets
* Drag in model under the ‘Model’ node.
* Add colliders for layers.
* Select a mesh (or multiple meshes) for each layer, and add collider to it (or them).
* Select imported models root entity, and set 'staic' flag to 'contribute GI' recursively.
* Select all entities that need not contribute to GI and unset static flag on them. Also disable cast/receive shadow.
* Adjust the POI.
* Add/Edit/Remove POI to match the project.
* Tip: Create additional POI by copy/paste/rename existing POI. (start from an original on the same level)
* Relocate the a POI by selecting it (not its children!), and dragging it to the correct position/orientation.
* Adjust the lighting
* Make sure thate scene global lighting mode is set to 'baked'
* Add/Edit/Remove lights to match the project.
* Tip: Create additional Lights by copy/paste/rename existing lights. (start from an original on the same height level)
* Make sure there is only one directional light (name it 'Sun'), and that it has a sensible orientation (usually 60,30,0)
* Make sure all lights are set to 'Mixed' mode.

## Edit ArchiVR project in VR

* Generate a build with only the project(s) to be edited.
* Push the build to a VR headset.
* Launch the application on the headset, enable ‘edit mode’
* Edit the project in VR.
  + Place POI
  + Place lighting
  + Place Props
* Copy over the ProjectData.XML from headset to development pc
* Run the project in editor, and generate baked GI.