# Description

The Archi-VR application offers the functionality of previewing architectural projects on a wide range of technologies, with different levels of immersion/hardware needs.

The application is targeted for the following platforms:

* Desktop
  + Windows
* Mobile
  + 2D
  + GearVR(TODO)
  + Google Cardboard (TODO)
* VR Headset
  + Oculus
  + Google DayDream (TODO)
  + Windows Mixed Reality (TODO)

The application has notion of the concept ‘Project’.

# Project

A project represents all data related to a specific architectural project. This includes all data during the entire lifetime of the project. Data from different discrete snapshots in time are subdivided into project Phases.

The application has an ‘active’ Project. The active Project references the construction project (from the list of included available Projects) to be currently viewed.

# Phase

* Before
* During
  + Tear-down
  + Rebuild
    - Step 1
    - Step 2
* After

# Modes

The Archi-VR application will have the following application states:

* **Home** application state
* **Play** application state

# Startup

The application always starts up in the **Home Menu** application state.

# **Home menu** application state

When entering this mode, the **Home** menu is loaded and shown.

The **Home** menu exists in 2 flavours:

* **non-VR** mode
* **VR** mode

## Non-VR mode Home menu

In this mode, the Home menu is represented using a screen-space full-screen overlay menu. The application has notion of a ‘selected’ project. There is always exactly one selected project. The user can browse the available projects, thereby changing the selected project. The user can also transition into **View Project** mode, for viewing the selected project, using the ‘Go’ button.

* A preview image of the selected Project is shown using a full-screen 2D preview image.
* The selected project’s name is shown by a Text UI control, located at the bottom-center of the screen.
* 2 Project selection browse buttons are shown to the lef, respectively right side of the selected project name Text at the bottom of the menu:
  + ‘Previous project selection’ button:
    - Caption ‘<’
    - Tot he left of the selected project name
  + ‘Next project selection’ button:
    - Caption ‘>’
    - To the left of the selected project name
* The portfolio name is shown by a Text UI control at the top-center of the menu.
* The credit ‘Powered by Archi-VR’ is shown by a Text UI control at the top-right of the menu.
* A ‘Go’ button
  + Caption ‘Go’
* An ‘Exit’ button
  + Caption ‘X’

## VR mode Home menu

In this mode, the Home menu is represented using world-space UI controls. Because mouse/keyboard/touchscreen/gamepad input is not always present in VR systems, the UI control scan be interacted with by using timed gaze. When the user starts gazing at a control, it is highlighted to visualize this. (For now the control is scaled a bit bigger while gazing at it.)

The **VR mode Home** menu consists of the following components:

* Project selection menu
* Floor menu

### Project selection menu

The project selection menu consists of **project select** buttons. For each available project, a project select button is shown as a world-space button, that shows a preview image of its corresponding project. Project select buttons are layed out in a circular order around the user’s view position. Project select buttons are ofsetted at a fixed distance from the user view position, and face the user view position. The user can transition into **View Project** mode for a specific project, by gazing at the corresponding **project select** button.

### Floor menu

The floor menu consists of the following controls:

* The portfolio name is shown by a Text UI control at the top-center of the menu.
* The credit ‘Powered by Archi-VR’ is shown by a Text UI control at the top-right of the menu.
* ‘Settings’ button
  + Image: Cog wheel
* ‘View Mode’ button
  + Image: Icon corresponding to current view mode.
* ‘Exit’ button
  + Caption: ‘X’

The user can:

* Open a specific project for viewing.
  + By gazing at the corresponding **Project Select** button.
* Open the Settings menu.
  + By gazing the **Settings** button.
* Exit the app.
  + By gazing the **Exit** button.

When opening a project:

1. The application leaves the Main Menu.
2. The application enters ‘View Project’ mode, for the active architectural project.

# **Play** application state

When entering this application state:

1. The application opens the 3D model of the active project.
2. The application loads and shows the UI controls for the **View Project** menu.

## View Project menu

The View Project menu is shown when ‘Show UI’ option is on. It is hidden when ‘Show UI’ option is off.

### View Project menu controls

The View Project menu contains the following controls:

* **Home** button
* **Show/Hide UI** button
* **Rotate Mode** button
* **View Mode** button
* **Time** button
* **Construction Lighting Mode** button

### VR and Non-VR flavours

The ‘View Project’ menu exists in 2 flavours:

* Non-VR
* VR

The **View Project** menu is always represented by a single canvas that contains all of the UI controls that represent the menu.

In the **non-VR** flavour, the ‘View Project’ menu is represented by a screen-space overlay canvas. The controls are standard UI controls that can be interacted with by mouse clicks and taps.

In the **VR** flavour, the ‘View Project’ menu is represented by a world-space canvas. The controls are special UI controls targeted at VR, that can be interacted with by gazing. The canvas is attached to the player position, and is located on an horizontal ground plane a bit below the player position. When the user is looking forward or up, the menu is oriented automatically to the user viewing direction. This ensures that the menu is always located in front of the player when he starts looking down. When the player looks down, and the menu cones in view, the canvas orientation becomes fixed, in order to enable the player to gaze at controls in order to interact with them.

## Keyboard key functions

|  |  |
| --- | --- |
| M | Show/hide Menu |
| C | Toggle Construction light mode |
| U | Move Up |
| D | Move Down |
| Arrow up | Move Forward |
| Arrow down | Move backward |
| Arrow Left | Move left |
| Arrow Right | Move right |
| Shift | Fast movement |
| L | Show/hide debug Logging |
| Fx | Show debug logging menu x |

TODO?

* F: Show/Hide FPS counter
* Home: Open main menu

## Show/Hide UI

While in the View Project mode, all UI can by shown/hidden by:

* On devices with keyboard: pressing the ‘M’ key. (from UI)
* On touch-enabled devices: tapping anywhere on the window, where there is no UI control.

## Time animation

The animation of time can be controlled with the following keyboard keys:

* F: Increase animation speed Forward.
* B: Increase animation speed Backward.
* S: Stop animation. (Set animation speed to 0)

The user can also tap or click the corresponding on-screen overlay controls. (TODO)

The time animation controls the celestial lighting in the scene.

## Construction lighting mode

The **Construction Lighting Mode** can be set to one of the following modes:

* AUTO: Construction lights are automatically turned ON or OFF, dependant on the current time.
* ON: Construction lights are always ON, irrespective of the current time.
* OFF: Construction lights are always OFF, irrespective of the current time.

When entering the ‘View Project’ state, the **Construction Lighting Mode** is initially set to AUTO.

In the menu, a ‘Construction Light Mode’ button is present, that represents the current ‘Construction Light Mode’ by means of a dynamic icon:

* ON: lightbulb with rays.
  + 
* OFF: lightbulb without rays.
  + 
* AUTO: lightbulb with caption ‘A’. (from Auto)
  + 

While in the View Project mode, the construction light mode can be changed by:

* On devices with keyboard:
  + Pressing the ‘C’ key. (from Construction light Mode)
* On devices with mouse input:
  + Clicking the ‘Construction Light Mode’ button.
* On devices with touch input:
  + Tapping the ‘Construction Light Mode’ button.

## FPS counter

The application keeps track of the Frames-Per-Second. The FPS counter is shown be default initially at the first launch of the application, and its last show/hide setting is stored when closing the application, and reapplied at the next startup of the application. In non-VR mode, the FPS counter is located in the screen-space overlay canvas. In VR mode, the FPS counter is shown on a world-space Text UI control that is attached at a fixed offset in front of the player head position.

Show/hide FPS counter can be done as follows:

* On all devices: by enabling/disabling **Settings > Graphics Settings > Show FPS** menu option
* On devices with keyboard: pressing the ‘F’ key. (from FPS)

Note: The FPS counter is subject to the **Show UI** option. If **Show UI** option is off, the FPS counter is hidden, along wit all other UI, irrespective of the **Show FPS counter** option.

The FPS counter also computes statistics, like the minimum/maximum and mean FPS. These statistics are saved in a file ‘FPS stats.txt’, located under the Application persistent data path.

## Supported platforms

The application supports the following platforms:

* Desktop
* Mobile handheld
* Mobile HMD
  + Cardboard
  + GearVR

## Navigating through the model

The user can navigate through the model using different input devices and methods:

* Touch
  + Swipe
  + Virtual D-pad
  + Screen Buttons
* Gamepad
  + Physical DPad
  + Physical Button
* Gyro
* Mouse
  + Drag
  + Click
* Keyboard
  + Key presses
* GearVR trackpad
  + Swipe
  + Click
* 6DOF tracking
  + Kudan SLAM
  + WMTracker

Scene navigation can be devided in 2 operations:

* Camera translation
* Camera rotation

### Camera rotation

Camera rotation is unconstrained (softwarematically speaking) when controlled by gyro or 6DOF tracking. Camera rotation by Gamepad, Virtual D-Pad, Mouse, Keyboard is softwarematically constrained in the vertical direction to [-90,90] degrees (IE User can not flip view upside-down.)

Camera rotation can be controlled using:

* Gyro
* Mouse
  + Drag
  + ?Mouse-lock mouse move?
* Swipe
  + Only on touch-enabled devices
  + ?Looks suspiciously like MouseDrag?
* ?Virtual D-Pad?
* ?Keyboard keys?

### Camera translation

Camera translation can be done in one of the following modes:

* Fly mode
  + User can translate in all directions (including up/down) at will
  + Unconstrained: No collision with model: User can move freely through model
* FPS Mode
  + User can translate in horizontal directions, and jump/crouch
  + Constrained: Collision with model: User can not move through model
* Teleport mode
  + Timed gaze
  + Click
  + ? Constrained: Collision with model: User can not move through model

Camera translation can be controlled depending on the mode as follows:

### Fly Mode

* Physical GamePad
  + Default controls:
    - Left D-Pad: Up/down
    - Right D-Pad: Horizontal strafe
* GearVR Trackpad
  + Default controls:
    - Swipe to any direction and hold pressed at the end of the swipe to strafe
      * Horizontal swipe = fwd/backwd
      * Vertical swipe = up/down
* Virtual D-Pad
  + For both touch and mouse.
  + Default controls:
    - Y-only D-Pad on left side of the screen: Up/down
    - XY D-Pad on right side of the screen: horizontal strafe
* Touch Swipe
  + Only touch enabled devices.
  + Default controls:
    - Same as GearVR Trackpad?
* Keyboard
  + Default controls:
    - Arrows: horizontal strafe
    - u/d: up/down
    - Shift: fast movement

The default Camera Navigation mode depends on the System:

* Desktop
  + Rotation :
    - Gamepad
    - Mouse
    - Touch
    - Keyboard
  + Translation
    - Gamepad
    - Touch
    - Keyboard
* Mobile
  + Rotation
    - Gyro
    - Virtual D-Pad
  + Translation
    - Gamepad
    - Virtual D-Pad
* GearVR
  + Rotation:
    - Gyro
  + Translation
    - Gamepad
    - TrackPad
    - Teleport
      * Which method? TimedGaze/Picking/UIButtonClick/UIListBoxSelection

When quitting the application, the current Navigation mode is stored. This previously stored Navigation Mode is then reapplied upon the next Application launch.

### Touch + Gyro

Look around by rotating the device in real world (like a camera).

Translate using the virtual joysticks:

* Left joystick: Translate in vertical directions (up/down).
* Left joystick: Translate in horizontal directions (forward/backward/left/right)

### Touch Only

Look around using mouse swipe.

Translate using the virtual joysticks:

* Left joystick: Translate in vertical directions (up/down).
* Left joystick: Translate in horizontal directions (forward/backward/left/right)

### Mouse & KB

Desktop mode.

Look around using mouse drag.

Translate using following keyboard keys:

* Arrow Up: Move forward.
* Arrow Down: Move backward
* Arrow Left: Move left
* Arrow Right: Move right
* U: Move Up
* D: Move Down
* Right Shift: Fast translation speed

# Settings menu

The settings menu consists of the following controls:

* An ‘Exit’ button
* A checkbox to enable/disable dynamic vegetation
* A checkbox to show/hide the FPS counter
* An option button to set the graphics Quality level:
  + Low
  + Medium
  + High
  + Very High
  + Ultimate

# Procedures

## Procedure: Add a construction project

1. Import assets
   1. model of building (Sketchup v2015), into Assets/KS/Model/ProjectXXX folder.
   2. Project preview image, as 2D/Sprite, into Assets/Resources/ProjectPreview/ folder.
2. Create project scene
   1. Named ‘ProjectXXX’, in folder ‘Assets/Scenes’
3. Add Lighting to project scene
   1. World/Lighting
      1. LightgroupXXX
      2. Light01
      3. Light02
      4. …
4. Add POI to project scene
   1. POI.Default
      1. L0Leefruimte
      2. …
5. Add project selection button to the Home state’s Project Selection menu
   1. This could be automated, so project selectionn scene buttons are dynamically creadted and arranged according tot he project scenes available.

## Procedure: Remove a construction project

Remove the project scene

Remove the project preview Sprite asset.

Remove the project selection button from Home state’s Project Selection menu.

# Development tools

## Monitoring application performance

The FPS can be investigated using the FPS counter widget and exported file.

## Developing/debugging a project scene

When making a lot of successive changes to a project scene, the easiest way to review and debug those changes is to set the ‘InitialProjectName’ setting on the PlayApplicationState. This makes the Play state start up with the designated scene loaded right away.

# TODO:

* Make it so that the ‘selected project’ in Home menu is the project that was last played.
  + Add ‘active project’ to application settings.
* Implement a key to generate screenshot:
  + While in ‘Play’ mode
  + Shortcut key = ?
  + Of specified resolution (1920\*1080)
  + Filename = Project name + datetime stamp.
  + (optionally) adjust rendering temporarily:
    - Hide all UI elements
    - Activate best available graphics quality setting.
  + Generated images will be usable, for example, as project preview images.
* Properly implement and document a way to generate POI.
  + Shortcut key = ?
  + To File (path:???)
* Add proper lighting to P006
* Align P006 With World Axes. This will enable using terrains more easily
  + Fix terrain for Tuin Achter
  + Add Terrain(s) in tuin voor
  + Add terrain(s) for tuin links/rechts)
    - Zet beplanting als ‘mesh’ details
* Add proper lighting to P008
* Add proper lighting to P011
* Add proper lighting to P024
* Add proper lighting to P025
* Add proper furniture to P011
* Review and finalize reusable lighting components as prefabs
* Initialize lighting in project 008 and 001 properly
* Load reusable ‘furniture’ components as prefabs.
  + Set material settings.
* Create ‘Furniture-less’ version of model for each project (skp v2015)
* Use furnitureless model in project scenes
* Prepare furniture in projects, using tweaked unity furniture
* Sky: Implement Clouds
  + Based on perlin noise?
  + Add heightmap for extra visual ‘depth’
* SkyDome : Implement Fog
* SkyDome : Implement Finegrained control over Sky gradient.
  + Shader: add property \_SkyLight1InfluenceRangeAngle
* SkyDome Shader: implement properly support for arbitrary number of celestial objects.
* Sky behavior: use SkyDome Shader support for arbitrary number of celestial objects, to properly represent both sun and moon.
* SkyDome : Implement making rendering ground optional.
* SkyDome : Implement timed ‘Ground Colors’