Project RelaXR - Technical document

This document serves to describe the interface classification of the project’s software, how feedback from peers and project owner was considered and turned into reality, how the software was tested and how it can be further developed and deployed.

# 1. Software interface classification

The software’s interface can be described by three categories – graphical & multimedia interface, gestures & multimodal interface, and augmented reality. Under each section we dive deeper into what parts of the software correspond to each of the categories.

## 1.1 Graphical & multimedia interface

Our project relies heavily on clicking buttons to get into the exercise and configuring all the settings necessary to these exercises in the best way possible. We split these up into smaller segments to avoid having all windows active at once.

* Main menu
* Muscle relaxation
  + Settings for muscle relaxation
* Breathing exercise
  + Settings for breathing exercise

The main menu is designed to contain all the fundamental settings that alter the game and the environment itself such as toggling the various sounds available and toggling the terrain on and off. All of the sounds available are meant to calm the user and give them feelings of comfort. It is important that you can turn the terrain on and off to avoid risking users bumping into something if they want to move around while in the application. In the main menu, there are also two exercises to choose from. When selecting an exercise, you are faced with an instruction text to the corresponding activity as well as the option to configure the amount of time to spend on the different segments. When you’ve configured the settings, press the play button to start the exercise. While performing it, there is an option to quit earlier than specified. This is useful if any settings are wrong or you simply had to quit because something happened and you need to quit the exercise urgently.

The icons are designed to be intuitive and are therefore the standard in terms of what we are used to. Some examples include, but are not limited to, settings are displayed as a cogwheel, sound with a speaker and terrain with some mountains. Some text is needed to explain and describe the exercises so everyone can understand them. It is of great importance to design the application so everyone can use it with ease.

## 1.2 Gestures & multimodal interface

When navigating through the menus some gestures are needed. If the user is situated close to the menus, they can tap with their finger on the desired icon to do what they want. If they can’t reach the desired icons, the user can select it via the visible pointer in the HoloLens 2 and pinch their fingers to access it. The pinching isn’t only necessary if the player is far away from the menu though, it is also imperative if they want to change the duration on any of the timers. It is not enough to point at the sliders, one has to pinch and drag to the desired time and thereafter release the pinching.

In future development of this application, tracking of the body to make sure that the user is doing what they are supposed to should be considered. Thereby, the chance of the user cheating is mitigated as much as possible.

## 1.3 Augmented reality

In augmented reality, virtual representations are overlaid on physical objects and devices, meaning they become a part of the application. The HoloLens 2 is developed with such technology.

# 2. Feedback & further development

During the development process a recurring point of feedback was to make the exercises more intuitive and explain them inside the application. Rather than relying on external explanations. This feedback is what led to the intermediate menu with exercise explanations and settings. As well as visualizing the different steps of the exercises (lungs filling/draining the blue color etc.).This is an important facet of making the application as useful as possible and should be kept in mind during further development.

Another point of feedback is the lack of interactivity during the exercises. In the application’s current state the exercises are not much more than simply watching an animation play out. Using the possibilities provided by the HoloLens 2 headset, such as hand tracking, could be useful and enhance the utility of exercises in the future.

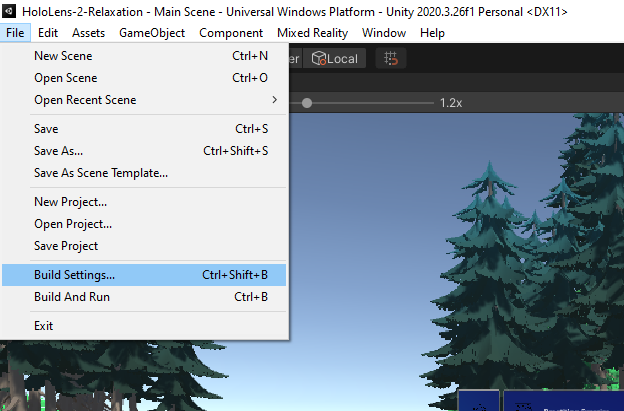
As it stands now, the application menu and the position of the exercises is entirely dependent on the location where the application is started. It would be useful to implement a button that follows the camera that can make the menu relocate in front of the camera.

One important note for the future developer is that the project on GitHub requires you to manually add the MRTK library to your Unity project once you have opened it. This is because those files are too large to upload.

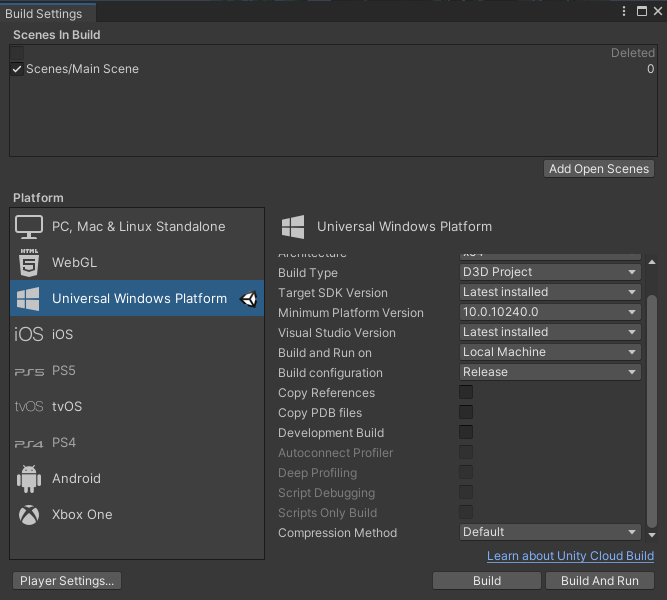
# 3. Packaging & deployment

There are a fair amount of steps required to get the application up and running on the HoloLens 2. In this guide we show two ways to deploy the application onto the HoloLens 2.

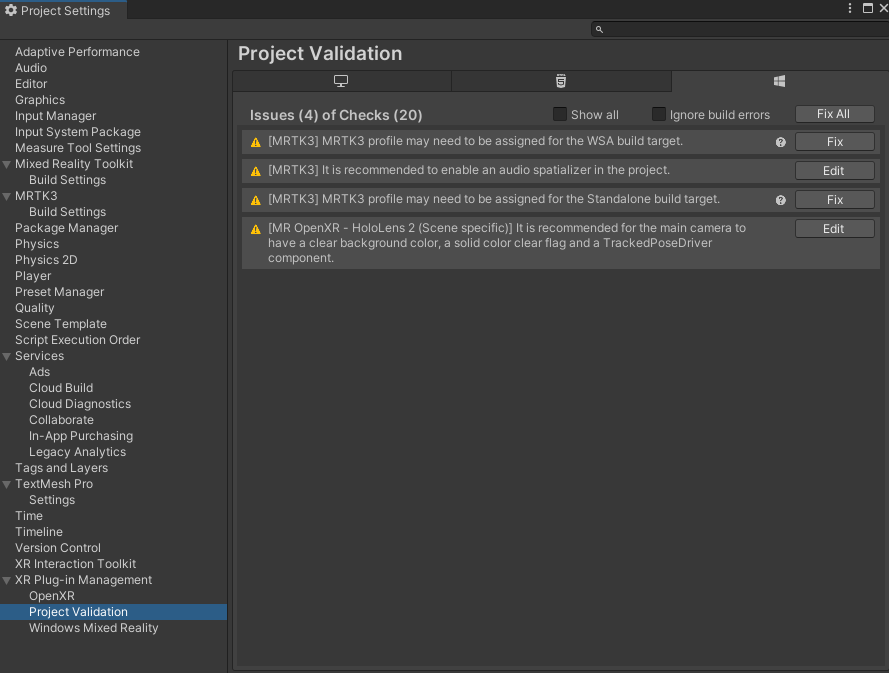
## 3.1 Building

Inside the Unity project, enter “File->Build Settings…”

From here ensure that the project is set to build for the Universal Windows Platform and that the Main Scene is part of the “Scenes In Build”.



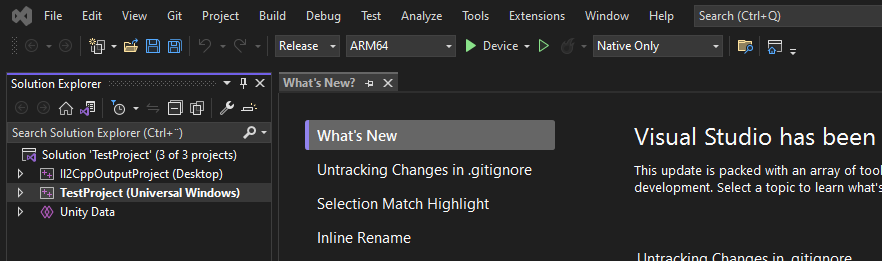
Also ensure that the project validation is complete by pressing the “Player Settings…” button located at the bottom left of the Build Settings window. This will open the project settings which can also be found from “Edit->Project Settings…” from the main menu bar.

From the project settings menu, open the project validation tab of the “XR Plug-in Management” tab.  


If any issues have arisen. Press the “Fix All” button located at the top right of the Project Validation window.

After this, return to the Build Settings window and press the Build bottom located at the bottom right. Specify a build folder and wait for the process to finish.

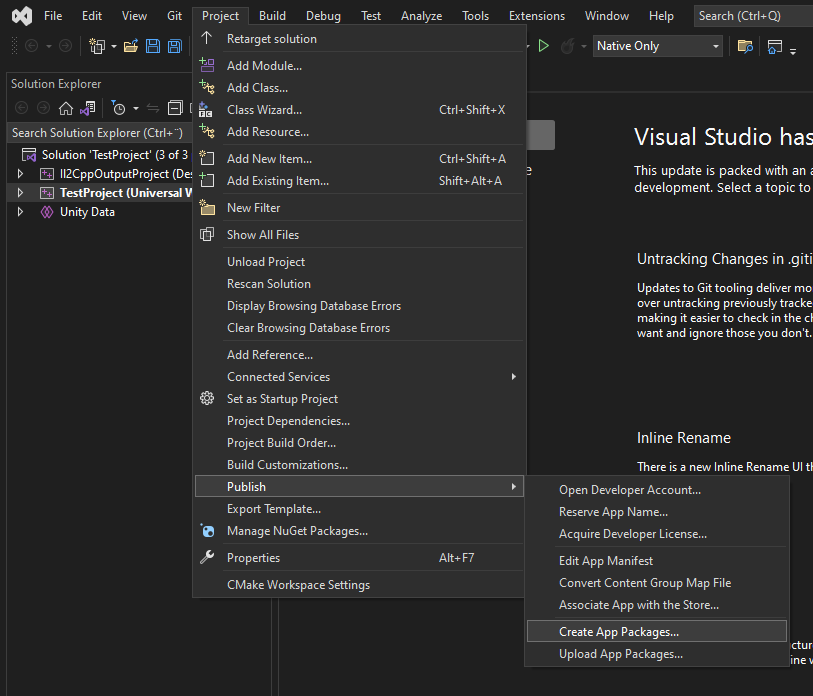
## 3.2 Deploying over USB

When finished, use Visual Studio to open the created solution file [projectname].sln located in the build folder. Inside Visual Studio, ensure that the correct project is selected in the solution explorer.  
  
Plug in the Hololens using USB. To deploy the application. Change the deployment settings to “Release” for “ARM64” targeting “Device”. It should look like this.  


Press the button labeled Device and wait for the application to deploy. If deployment error arises and you are asked to continue. Select no and then, without unplugging the Hololens or changing any settings, press the button labeled Device again. When the deployment has succeeded the application should start running on the Hololens.

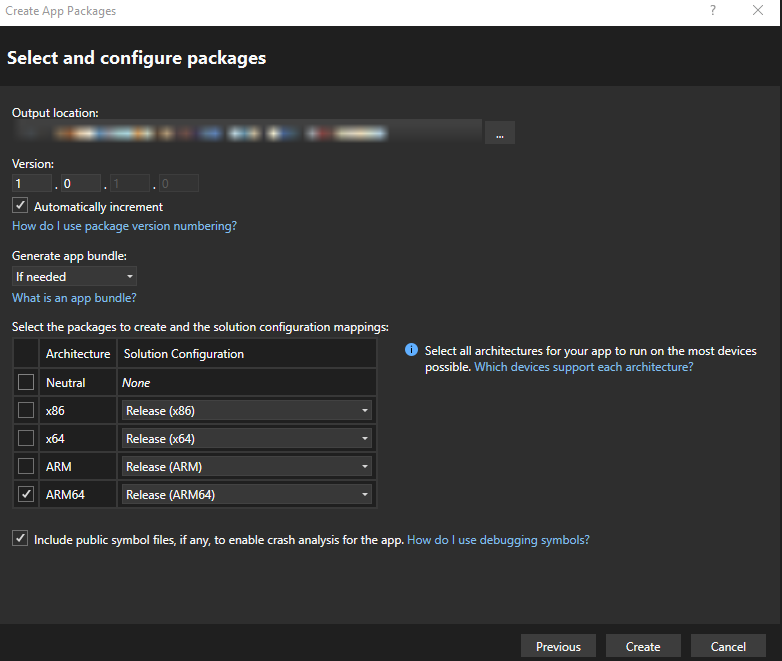
After successful deployment the application will remain on the Hololens, being accessible through the application menu of the headset. Even when not plugged into the computer.

## 3.3 Deploy as app package

After building your solution in Unity, open it using Visual Studio.  
When browsing your solution, open the “Project->Publish->Create App Packages…” option.

When asked to select a distribution method, select Sideloading. As for the signing method, select “Yes, use the current certificate”. As it stands now no company information etc has been set in the unity project.

When configuring the package select the ARM64 architecture with a solution configuration set to “Release (ARM64)”.

  
Press create and wait for the app package to finish building.

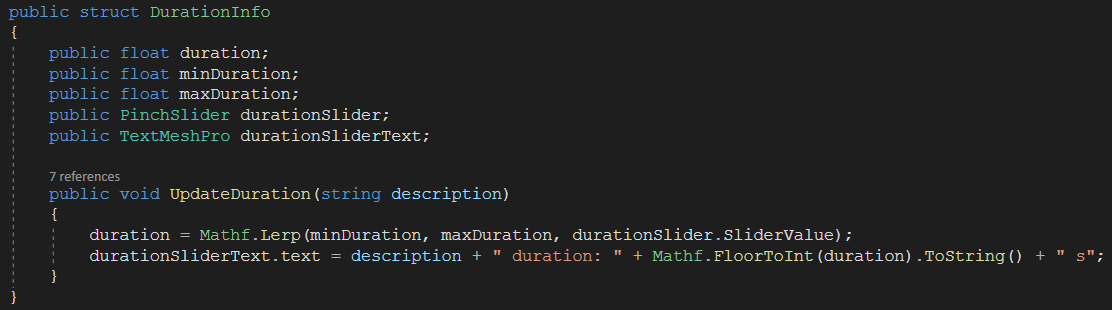
When it has finished you will find a [projectname].appx file at the output location.  
Move this file into the Hololens’ internal storage and boot it using the file explorer inside the Hololens.   
For detailed instructions on how to sideload applications on Hololens. See: <https://learn.microsoft.com/en-us/hololens/app-deploy-app-installer>

# 4. System testing

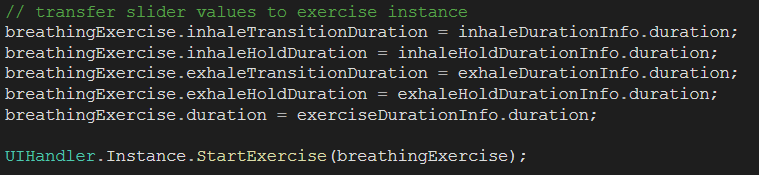
The system testing was divided into two parts – validating the software by checking the functional requirements, and testing the communication between components. Checking the functional requirements was quite trivial for our small system and it could be performed both inside Unity’s play mode and on the HoloLens 2 simply by going through the list of requirements and testing each functionality. The following list of functional requirements was used:

1. The application shall present the user with VR experiences focused on breathing exercises and muscle relaxation
2. While in a breathing exercise, instructions for the breathwork shall be presented visually using 3D models and animation and/or coloring
3. While in a muscle relaxation exercise, instructions for the tensening and relaxation of the muscles shall be presented visually using 3D models and animation and/or coloring
4. When starting the application, the player shall be able to choose which exercise to perform via a menu
5. While performing an exercise, the player shall be able to go back to the menu by interacting with the interface
6. At the end of an exercise, the program returns to the main menu

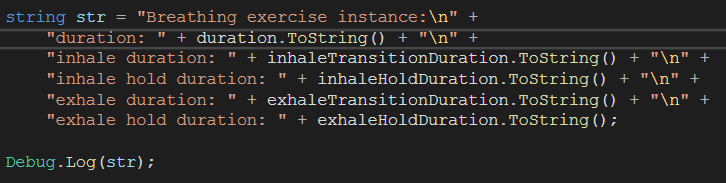
Although most communications between components were already implicitly tested by the requirements validation, there was one type of communication that was not covered – the transfer of data from the instruction menu sliders to the exercise instances. Each instruction menu utilizes a class called *DurationInfo*, that collects the value of a *PinchSlider*, to get the data from the settings page.

**

When the *play* button is pressed in the instructions menu, the values are then transferred from the *DurationInfo* objects to the public variables of the corresponding exercise instance that the instructions menu has been assigned with.



To test if the expected values were used, the values displayed in each instructions menu were compared to the corresponding values used in the exercise instance by printing them out when the exercise was started.



The following images show how values were entered on sliders in the instructions menus and how the corresponding values were printed out from the exercise instances once the exercises started (note that the values displayed above the sliders are floored values).

| Breathing settings: | Breathing exercise: |
| --- | --- |
| Muscle relaxation settings: | Muscle relaxation exercise: |

As seen in the images, the values that the user sees in the sliders of the settings pages are what the exercise instances end up using.