Unity Engine Crash course

The main technology used to develop this project is Unity a cross platform game engine and integrated development environment (IDE) which also will be used for many other purposes like this application too. First, we dive into the story of unity and how it was founded followed from a quick but efficient crash course to understand the structure of the Unity editor today.

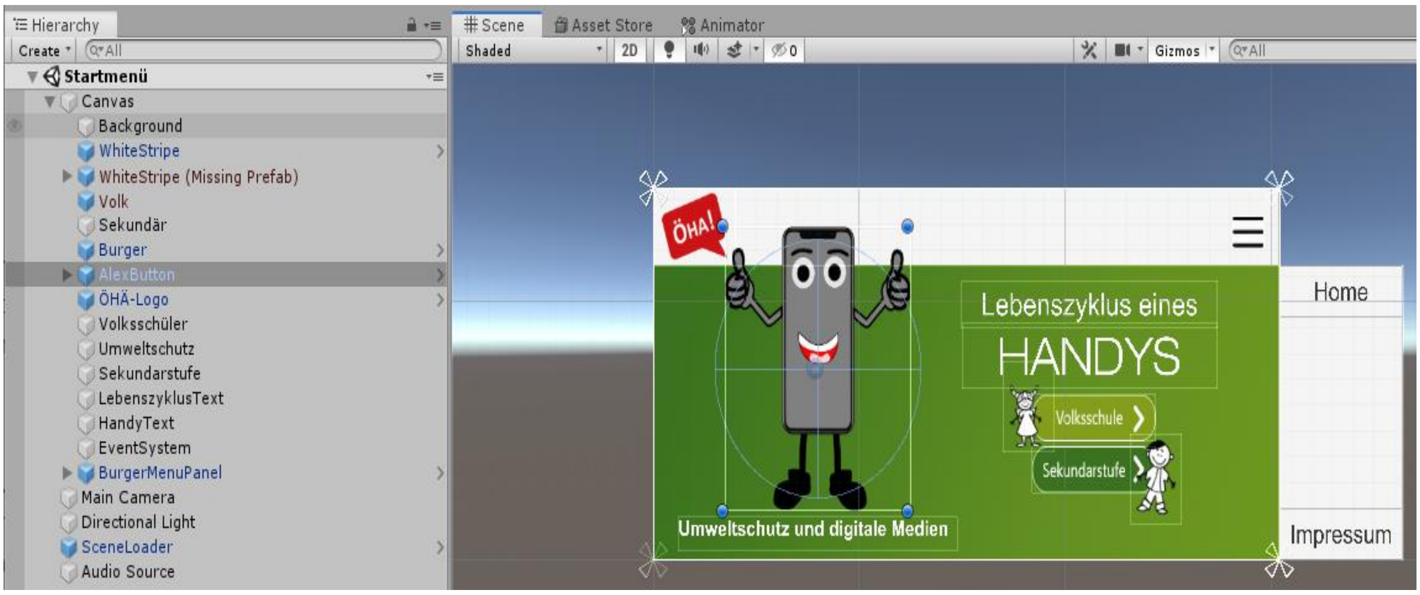
Origins of unity

The first version of unity was created in Denmark by the founder of Over the Edge Entertainment (OTEE) which are also the founder of Unity Technologies. The three founders are David Helgason, Joachim Ante and Nicholas Francis. The initial product launched on June 6, 20005. The first version of unity was only compatible with Mac OS X. But with the time Unity supported more and more platforms till today with over 25 platforms on their list. Also the technologies evolved with the time further and further especially the augmented reality section has high standards in Unity.

The Unity Editor today

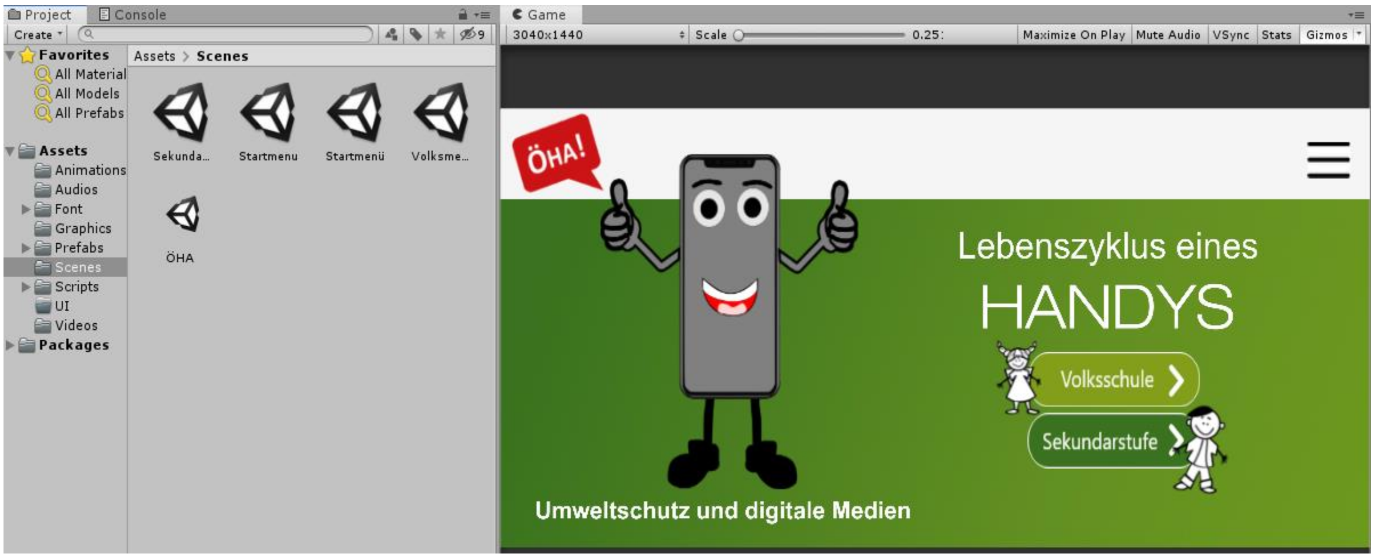
The Unity editor used in this project has several sections important for the daily developing in Unity. The most important areas are: The Hierarchy window, the Scene, the inspector, the game window and the project section. There still many others like the asset store but it will be focused on the necessary ones.

The Hierarchy and the Scene



Unity works with big elements called scenes every scene is like an own level or area. The Hierarchy section on the left manages all the Object which will be used in a specific scene. Every Scene has its own hierarchy and the Objects in it are called “GameObjects”. GameObjects can be anything from the camera element, to graphical elements, audio or video specific elements. In the scene the objects can be placed, moved around, selected or manipulated in many other ways. Also the management of inheritance will be managed in the Hierarchy window. All the elements little offset to the right are children from the first more left object above in the hierarchy.

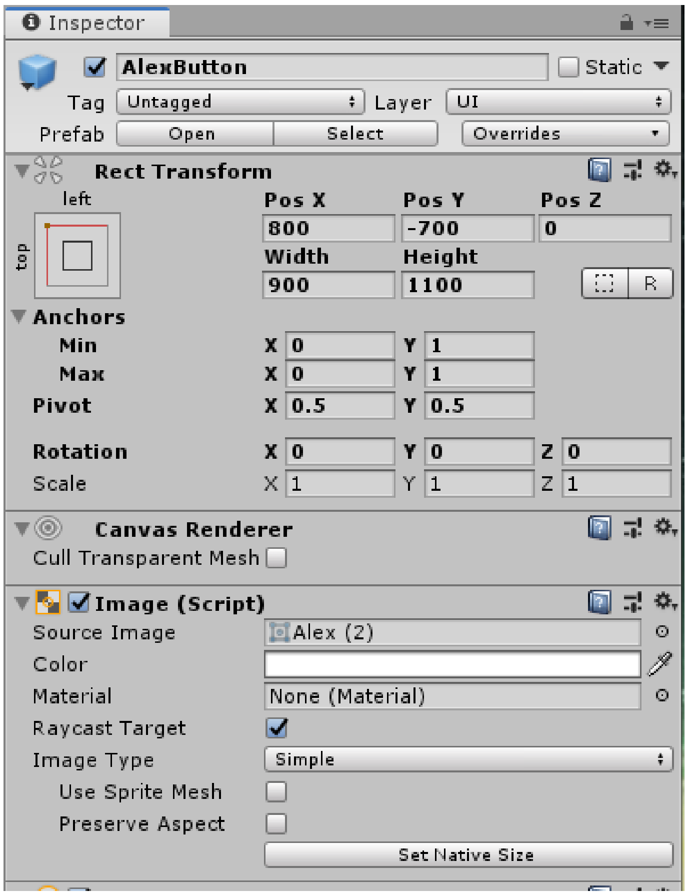
Project and game view



The project area is more or less the storage area of the editor. The file system here is nearly equal to the filesystem in which the project is saved. With the help of folders can be here made an efficient structure for all the files which will stored in the project. For examples script files, audio files, images, video files and many more. But also Prefabs will be stored in the project area. The Game view is just how the whole project would look like in lifetime right now. Also live tests can be made by simulating the behaviour of the project in the game mode.

The Inspector

The inspector holds most of the important attributes and information about specific objects, in this case a button object. Every GameObject has several components and every component has a specific topic for example here the Rect Transform which contains all the important data about the position, the anchors, will be discussed in a later chapter, the rotation or the size of the object. With the inspector the developer is able to attach any kind of object or scripts from the project view to this GameObject and also other smaller GameObjects. The inspector connects the different logics and the information from the windows and is definitely one of the key elements in Unity. There are opportunities to adapt the inspector for specific situations, but this requires much advanced knowledge.


Inspector
AlexButton — Static
Tag: Untagged — Layer: UI
Prefab: Open | Select | Overrides

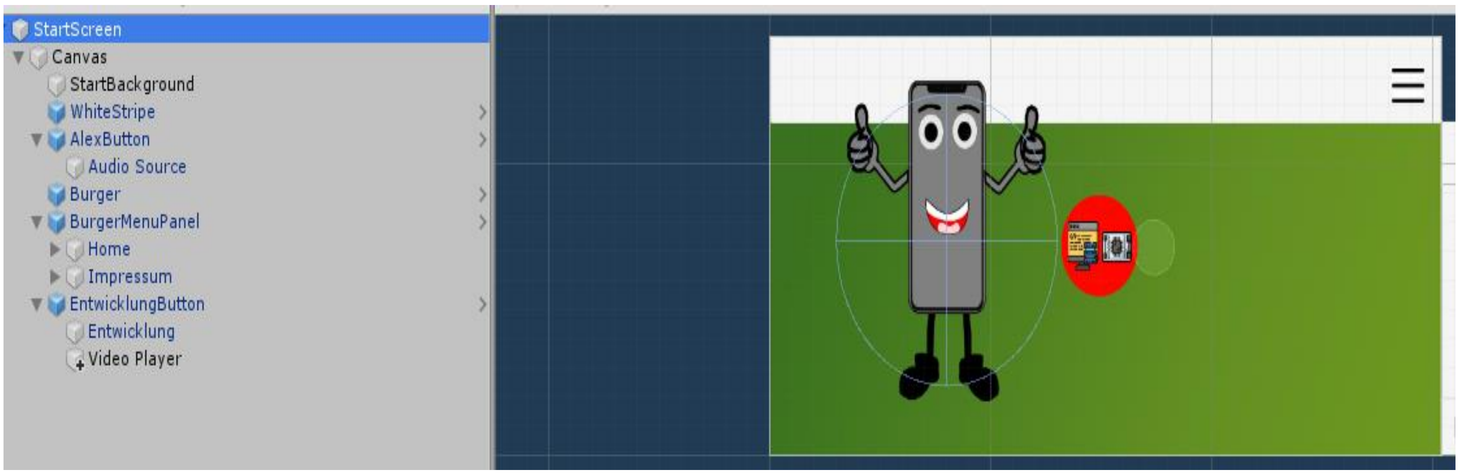
Rect Transform
left / top
Pos X: 800 — Pos Y: -700 — Pos Z: 0
Width: 900 — Height: 1100
Anchors
Min: X 0, Y 1
Max: X 0, Y 1
Pivot: X 0.5, Y 0.5
Rotation: X 0, Y 0, Z 0
Scale: X 1, Y 1, Z 1

Canvas Renderer
Cull Transparent Mesh: ☐

Image (Script)
Source Image: Alex (2)
Color:
Material: None (Material)
Raycast Target: ✓
Image Type: Simple
Use Sprite Mesh: ☐
Preserve Aspect: ☐
Set Native Size

Prefabs and the prefab view

Prefabs or the prefab asset type are an important mechanism in Unity. Prefabs are like blueprints of GameObjects used to make many copies of an object. A prefab stores a GameObject completely with all the components and properties. Another advantage is that any changes made to the prefab asset will immediately be changed on all copies as well, which saves a lot of time and effort. The prefab view works like the hierarchy and the scene mode, but it manages one prefab and every component inside of the prefab. Furthermore offers prefabs the functionality to be generated dynamically, during runtime, via a script. The amount of new opportunities with prefabs and the combination with other Unity tools is endless.

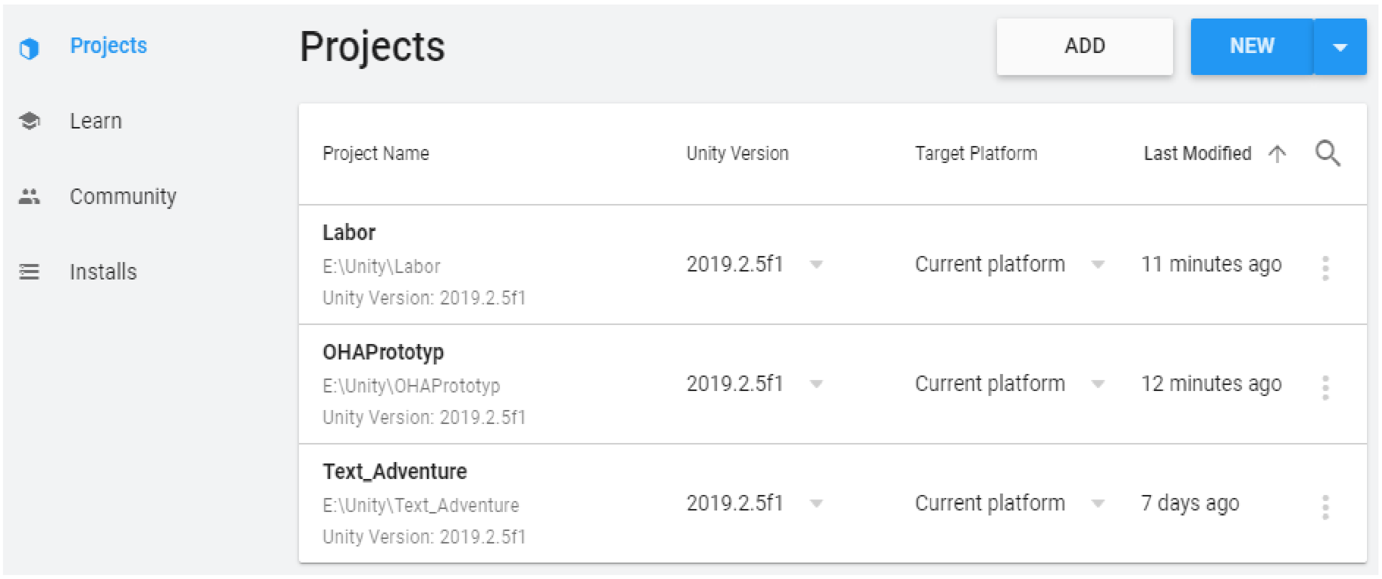


The Unity Hub

This standalone application is a great and new management tool of Unity to

- create new projects
- manage your Unity installations and versions.
- Observe your Unity account and editor licenses.

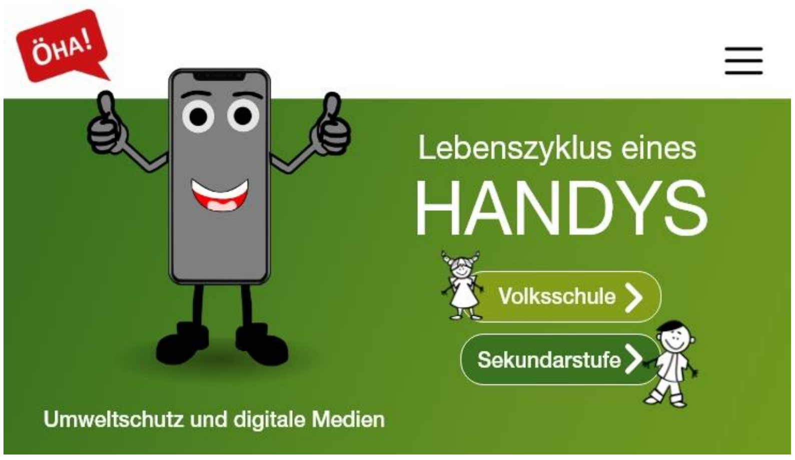
Also to add modules like the support for different platforms, get the access to the documentation and other resources of Unity.



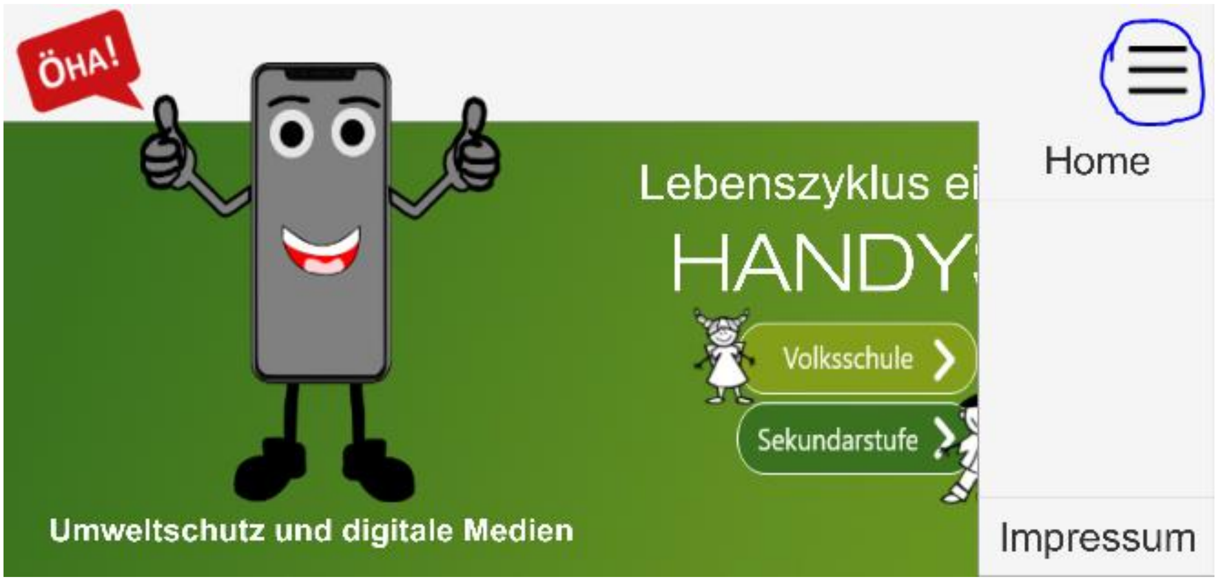
Project Name	Unity Version	Target Platform	Last Modified
Labor E:\Unity\Labor Unity Version: 2019.2.5f1	2019.2.5f1	Current platform	11 minutes ago
OHAPrototyp E:\Unity\OHAPrototyp Unity Version: 2019.2.5f1	2019.2.5f1	Current platform	12 minutes ago
Text_Adventure E:\Unity\Text_Adventure Unity Version: 2019.2.5f1	2019.2.5f1	Current platform	7 days ago

Starting Screen

The first screen of the application should look like the Image under this text. By clicking on the mobile phone avatar named Alex, an audio in which important information will be provided, starts. The technology for this functionality and my approach will be explained in the chapter “How I worked with audios in Unity”. The next interactive element is the Burger Menu in the upper right corner of the screen. This element makes use of the Animation Technology of Unity and will be explained in the following chapter with the name “Burger Menu and the Animation System”. The last clickable object on this screen are the two Buttons “Volksschule” and “Sekundarstufe”. With these buttons the user singles out what level of information should be provided in the following screens.



Burger Menu and the Animation System



After a click on the Burger Menu button an animation will be started and this menu slides into the screen offering new selection opportunities for the user. But how does this mechanism work? Let’s make a crash course about animations in unity.

Animations in Unity

The current animation system in Unity often is referred as "Mecanim". There has been an older animation system called Legacy. Due to the fact that Mecanim is recommend in almost every situation today, is the following description about Legacy kept very short.

Legacy-Animations before Unity 4

This system was simpler but also less efficient. For Humanoid characters an avatar was needed and for non-Humanoid characters it was important to define a root-bone which was the approximation to the centre of mass in the body. Another important element was the rig tab, but I will not go on this any further.

Mecanim-Animation system

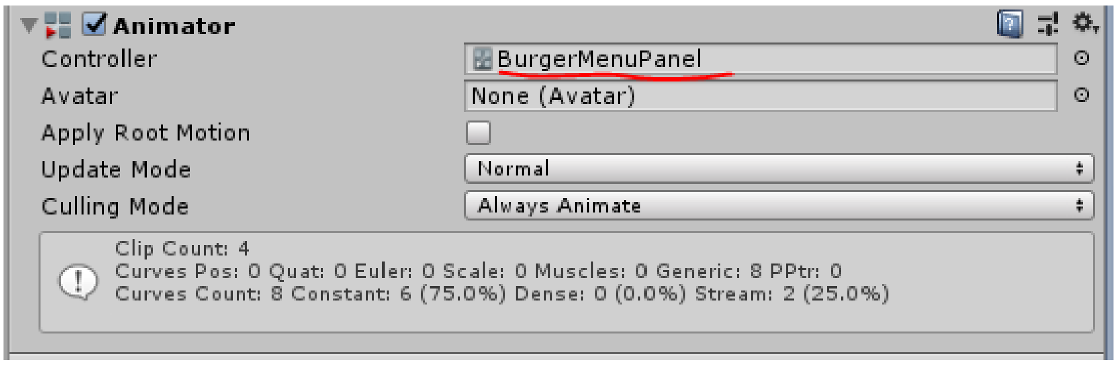
The whole animation system is based on so called animation clips. These can be thought as linear recordings, containing the information about movements, changing of the position, rotation and other aspect over a specified amount of time. The next important element is the animator controller. This flowchart like system provides the organisation of all used animation clips for a specific area. These two components are essential to provide animations in Unity. The third powerful component is the avatar system. This application does not need an avatar but there will still be a short explanation for interested persons, followed from the description how the animation in this application are created.

Avatar system

This system provides special features for humanoid character. It is able to retarget humanoid animations from any source, like for example motion capture, to your own character model. Another great functionality is the ability to use muscle definitions instead of bone. This makes the animations more realistic due to the fact that the character now is able to deform in a convincing way, without any self-overlaps.

The animator component and how the Burger Menu is made

At first the UI elements were grouped together. In that case, one Image having several buttons as children, but the whole component will be referred as Burger Menu in this text. At next it is necessary to implement an Animator component in the Burger Menu. This component is the connection between the GameObjects and the animation system in Unity. The next step is to assign an animator controller to the animator component. The following picture shows an animator component with the animator controller underlined in red. Also the avatar would be assigned here if there is one needed.



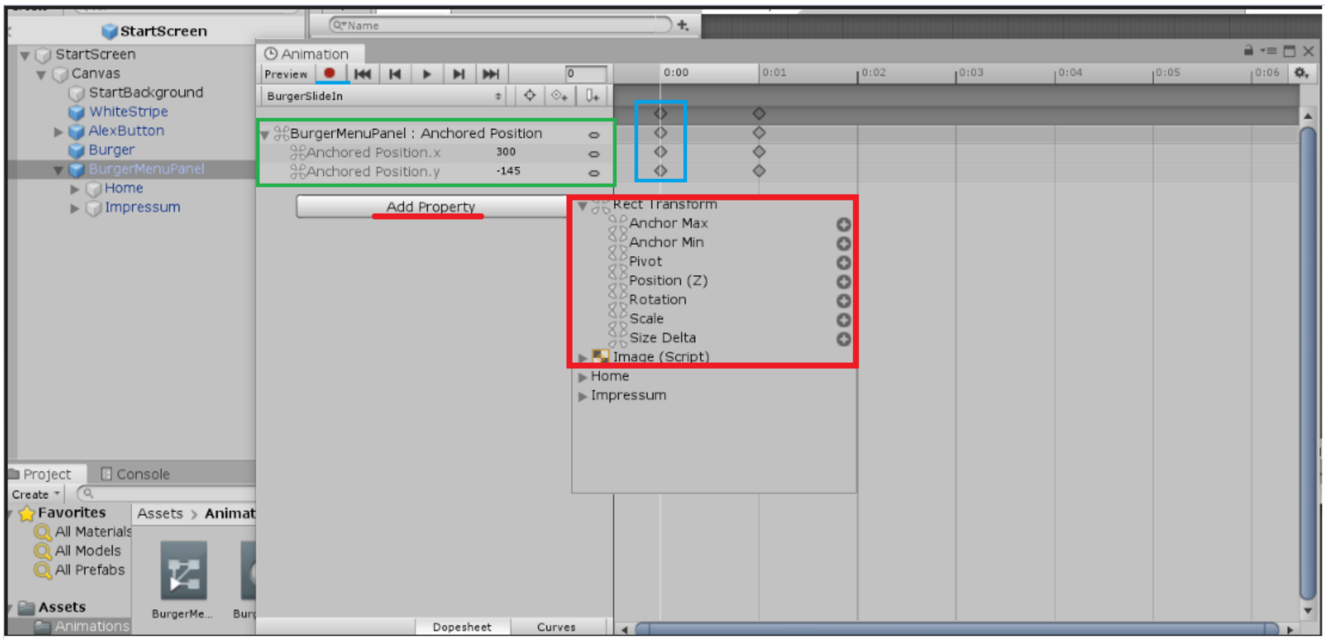
Animator

Controller	BurgerMenuPanel
Avatar	None (Avatar)
Apply Root Motion	☐
Update Mode	Normal
Culling Mode	Always Animate

```
Clip Count: 4
Curves Pos: 0 Quat: 0 Euler: 0 Scale: 0 Muscles: 0 Generic: 8 PPtr: 0
Curves Count: 8 Constant: 6 (75.0%) Dense: 0 (0.0%) Stream: 2 (25.0%)
```

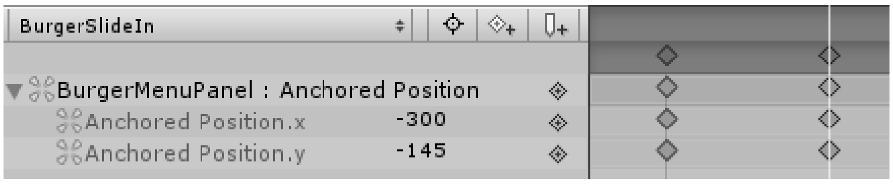
Animation clips

The theory of animation clips has already been discussed but now it is time to point out how does the practical part looks like. Animation clips will be created in the project area in Unity. A simple right-click →Create (the menu on the top → Animation generates an animation and opens the animation editor. At next the GameObject in the Hierarchy need to be chosen that the animation knows which element will be manipulated. It is only possible to manage one GameObject per animation. Clicking on a different GameObject in the Hierarchy will display another animation which already made use of the selected Object or if there is no animation using the object a new animation window appears. Now its time to explain what is included in this animation window.



BurgerMenuPanel : Anchored Position
Anchored Position.x 300
Anchored Position.y -145

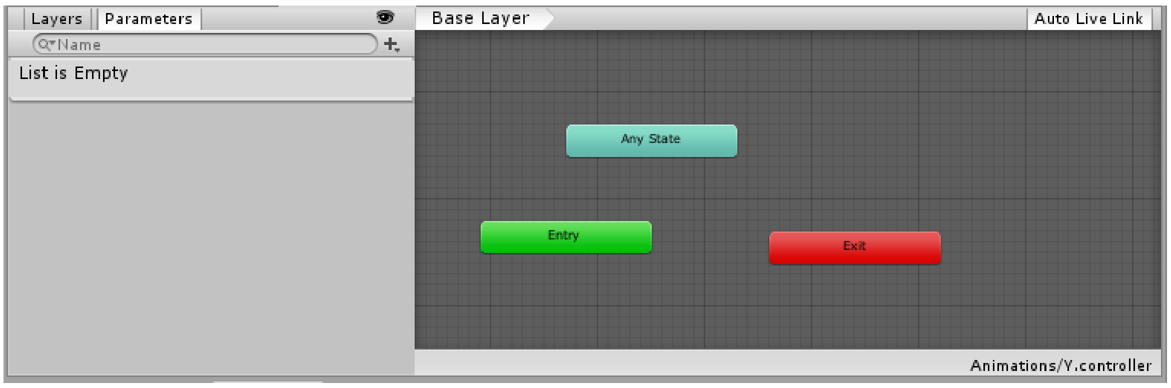
On the left side is the selected object our Burger menu called “BurgerMenuPanel”. After clicking the “Add Property”-button the little window in the middle of the screen appears. All components of the Burger menu and are arranged in the red rectangle. In this scenario the components are only the “Rect Transform” and the Image identifiable by the small arrow and a symbol left of the name. The parameters of the components are slightly offset and marked with a plus symbol on the far right. Underneath are the child object, in this case the buttons inside of the menu. The difference between child objects and components attached to the object is just the missing image after the arrow. The blue underlined button is the recording button, which enables the functionality of defining the change of the selected properties over the different timeline on the right. The position of the Burger menu at the beginning of the animation is selected in the blue box and the values are displayed in the green box, in form of the anchored position of the Burger menu. The following picture shows the second position of the Burger menu. The white stripe always shows which moment is handled in the timeline right now. Last but not least this animation clips are saved.



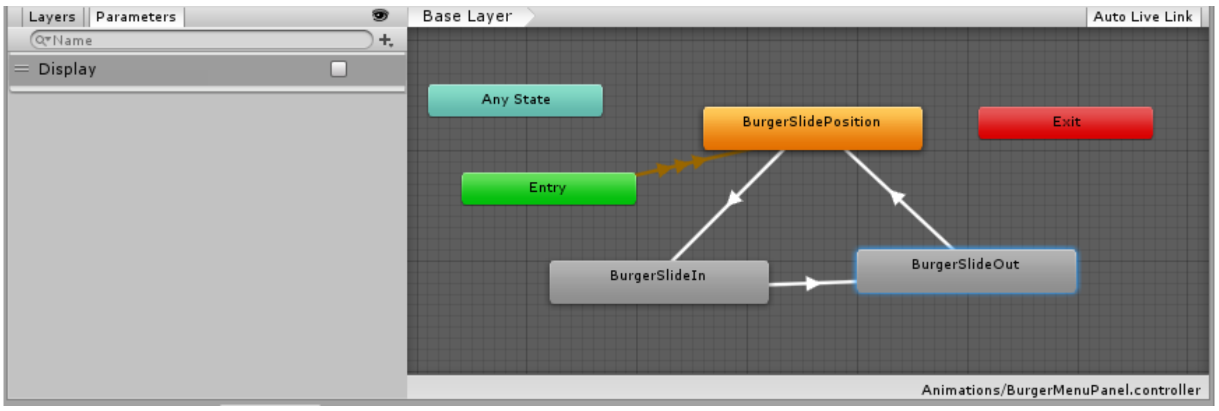
BurgerMenuPanel : Anchored Position
Anchored Position.x -300
Anchored Position.y -145

Managing the animation clips with the animator controller

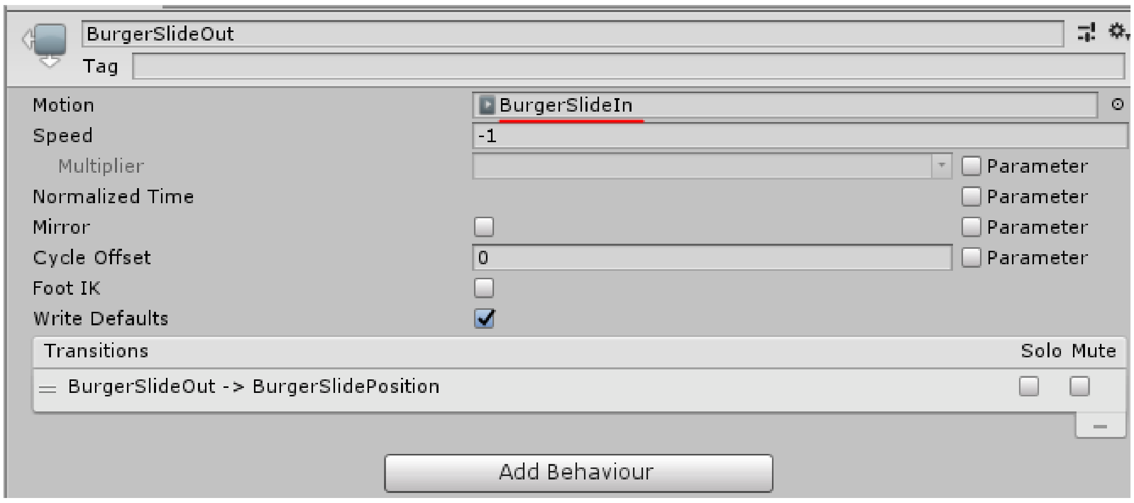
After being able to make single animation clips it is time to combine all required clips into one animation. The animator controller will be created on the same way as the clips with a right-click followed from a click on the animator-controller field in the create menu. The animator controller window will first look like this. Maybe at the beginning there is the layers window on the left side enabled but it has no impact on this animation therefore this is now the start setup.



Any state is a special state which would always be present. There are situations where you might want to go to a specific state regardless of which state you are currently having, but in this scenario, there is no need for the any state. The simplest way to integrate animation clips into this system is just to drag and drop it into the base layer window. Afterwards a bool parameter for this animation is needed. Boolean means just that this parameter can have two states, true or false. With the plus button on the upper right corner auf the parameter window, parameters will be created. Now it is time to connect the animation clips and make a sequence how the clips will be executed. A right click on the clip followed by a click on the "set as default state"-button defines this button as the state at the beginning of the animation. Then always select the first clip of the connection by right-clicking and selecting make transition. A left click on defines the endpoint of the connection. In this case the result would look like this:



The arrows are the transitions, which leads in this scenario to an endless loop because the animation should be able to be executed more than one times. Display is the bool parameter needed for the programming logic. "BurgerSlideIn" is the animation form the former chapter. "BurgerSlideOut" is just a copy of "BurgerSlideIn" with just the speed attribute set to -1. The attributes of animation clips are displayed in the inspector. The following image shows the attributes of (BurgerSlideIn) with the animation clip underlined red and the speed attribute below.


BurgerSlideOut
Tag

Motion	BurgerSlideIn	
Speed	-1	
Multiplier		☐ Parameter
Normalized Time		☐ Parameter
Mirror	☐	☐ Parameter
Cycle Offset	0	☐ Parameter
Foot IK	☐	
Write Defaults	✓	

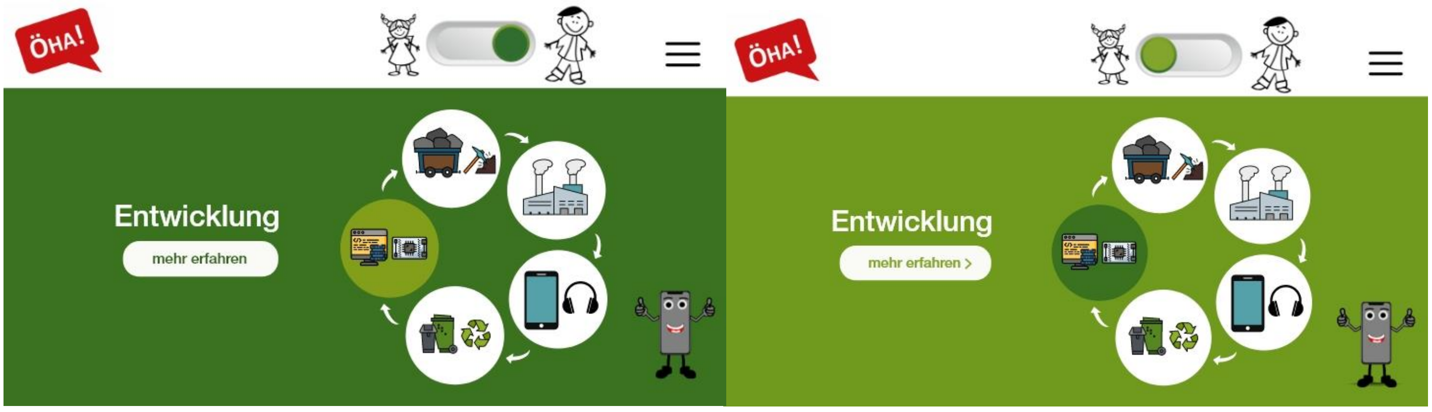
Transitions	Solo	Mute
BurgerSlideOut -> BurgerSlidePosition	☐	☐

Add Behaviour

That is everything on the animation side of this task. The last step is to implement a C#-script in the Burger button at the beginning of the chapter and assign the Burger Menu as parameter to the function which is executed by the "OnClick"-event of the button.

Second screen: Topic selection with radial menu

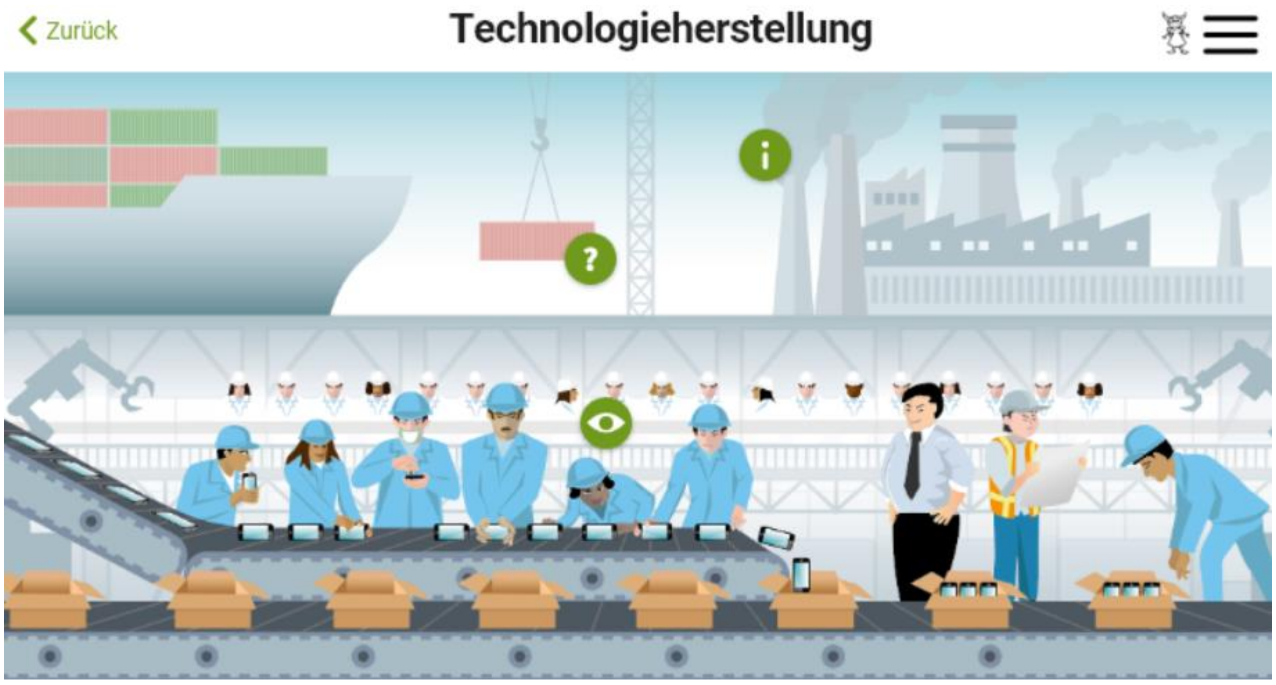
After the selection between elementary school or secondary school, one of the following screens appears:



The secondary school screen has the dark green background while the elementary school has the lighter green design. On this screen is a topic selection from one out of five exciting topics, by clicking onto one of the five buttons. After the click on one topic button an animation starts. This animation let the whole radial menu rotate till the selected button is on the left position. Moreover is the text on the left changing to the name of the chosen topic and the colour of the buttons is changing so that only the selected button has the green background. The Burger Menu and the functionality with Alex is nearly the same despite from the fact that the script managing the audios is here more complicated, because the audio is depending on the chosen topic.

Last but not least the last element to mention is the toggle button on the top of the screen with the two pupils. With this button is it possible by just one click to change from the elementary to the secondary mode. The green button will change the colour and move to the other side and the whole screen will change into the other mode. The "mehr erfahren"-button leads to the next and most important layer of the application.

Third layer: The topics themselves



This screen provides all the learn material for the pupils. Every green button with a symbol in it offers a different information. The form of the information can be text elements but also other kinds of media like videos, quizzes and 3D-Models. Again this layer also offers a BurgerMenu and a toggle button but this time it is inside of the BurgerMenu to save space.

Navigation between the screens

First Attempt with many scenes

After defining the structure of the different layers and their functionalities it is time to solve the problem how to switch between the different layers. The first attempt which is described here shows how many people begin with Unity before the different logic and way of thinking of Unity will be discovered. Every layer will be used as scenes because scenes would be easy to manage and there is separation between them. The problem is that scenes are used for such small layer. Scenes are more meant to split big levels in videogames. Loading too many scenes can also cause a lack of performance due to loading times if there is too many switching between the scenes. A second disadvantage is the circumstance that the topic selection has two scenes, but they are nearly the same. With a simple prefab there would not be this double work.

Second Attempt via Prefabs

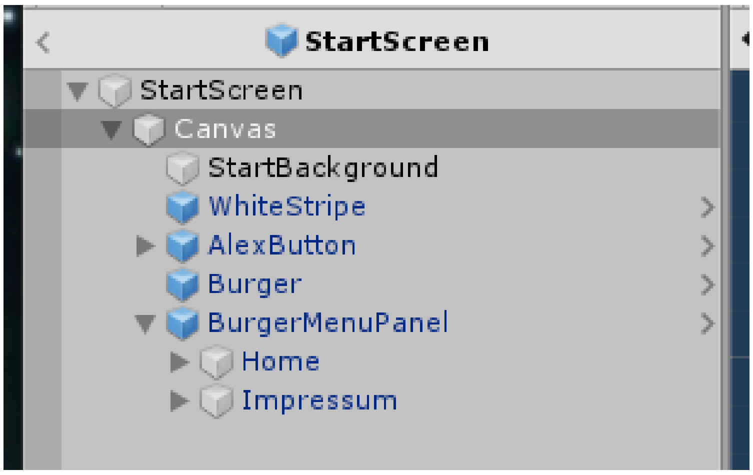
With the time the decision was made to change the former approach of the navigation to make a much better system. One of the biggest changes is the implementation of behaviour scripts. In other words C#-Scripts and GameObjects can be connected much more efficient by assigning more smaller scripts direct as components to the GameObjects instead of using less bigger scripts. In Unity the way how to connect scripts with Objects can be implemented with a different mindset in comparison to many other systems. With this strategy it is much easier to develop flexible systems.

UI-Design for multiple resolutions

Making a graphical design is the one thing but how to implement a design in Unity which supports a variety of different resolutions and is able to adapt the graphical layout? Fortunately are there tools in Unity for such purposes.

Canvas and the UI Scale Mode

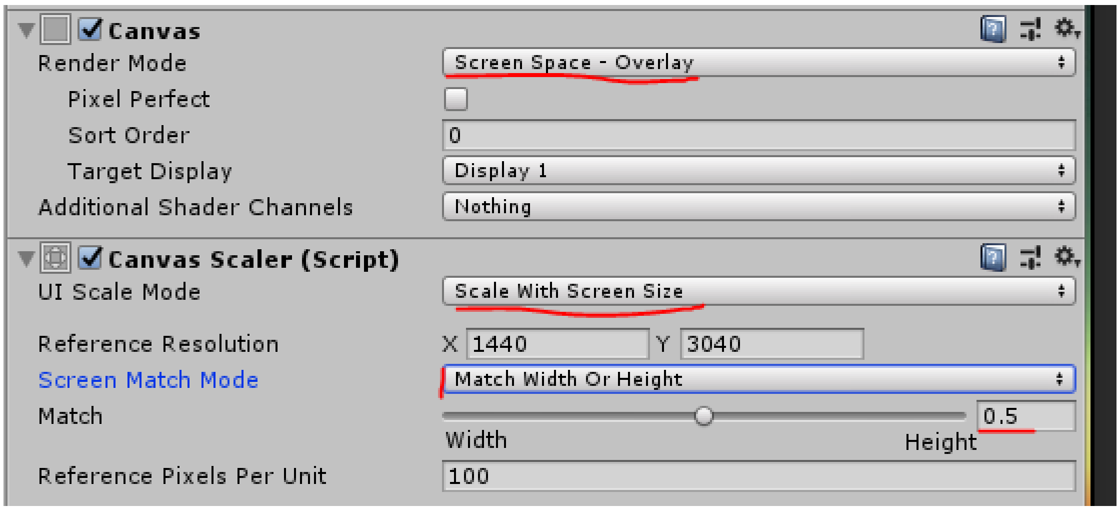
The most important Object in the hierarchy for managing all the UI in Unity is the Canvas. Normally every UI Object will be made to a child of the canvas to benefit from its functionalities. As can be seen on the pictures on the left every UI-Component is a child.



StartScreen

- StartScreen
  - Canvas
    - StartBackground
    - WhiteStripe
    - AlexButton
    - Burger
    - BurgerMenuPanel
      - Home
      - Impressum

The canvas scaler, a component of the canvas, includes many important features needed for the scaling of the UI-elements. First of all it is important to use the UI Scale Mode “Scale with Screen Size” this allows the canvas to stretch the UI elements in it. Afterwards it was for this case the best to set the screen match mode to “Match Width or Height”. The last parameter in this component needed to change is the match attribute to 0.5 this ensures an equally change of the width and height.



Canvas

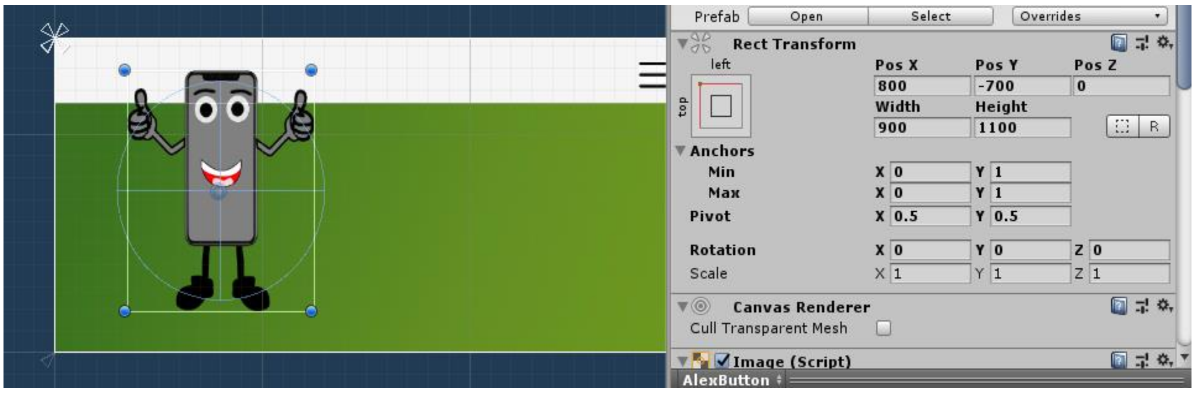
Render Mode	Screen Space - Overlay
Pixel Perfect	☐
Sort Order	0
Target Display	Display 1
Additional Shader Channels	Nothing

Canvas Scaler (Script)

UI Scale Mode	Scale With Screen Size
Reference Resolution	X 1440 Y 3040
Screen Match Mode	Match Width Or Height
Match	0.5 (Width – Height)
Reference Pixels Per Unit	100

Anchors and how the place them

To ensure that the scaling works correctly in the same way as expected it is recommended to make use of anchors in every of the UI-elements used in the canvas. This is a layout concept which uses four little triangular handles to manage the anchors and ensure a working scaling and layout.



Prefab | Open | Select | Overrides

Rect Transform

left / top

Pos X 800 | Pos Y -700 | Pos Z 0

Width 900 | Height 1100

Anchors
Min X 0 Y 1
Max X 0 Y 1
Pivot X 0.5 Y 0.5

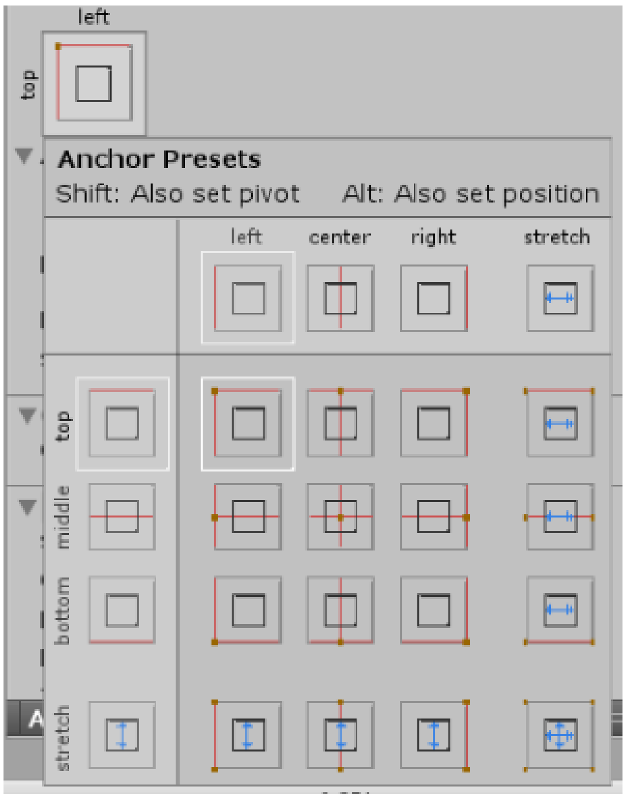
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Canvas Renderer
Cull Transparent Mesh

Image (Script)

AlexButton

With the window on the right the anchors will be placed and managed. By clicking on the square symbol in the right upper corner a new menu emerges



In this menu are twenty options how to place the anchors. But there is no need to despair with some simple rules it is easy to find out which option is the most suitable.

- When you use an object which should cover the entire screen like an image being the background than you split the anchor over all four corners of the screen.
- If the object is touching two corners and the width or the height are equal to the screen width or height than split the corners over these two.
- Normally every other object can be assigned to the closest corner.
- There can be exceptions for special circumstances like for the five buttons in the radial menu. To ensure that the constellation of the buttons is not destroyed because of the scaling they all got assigned to one corner also if it was not the nearest.

How I worked with audios in Unity

To work with audio sources in Unity you make use of following elements:

Audio sources

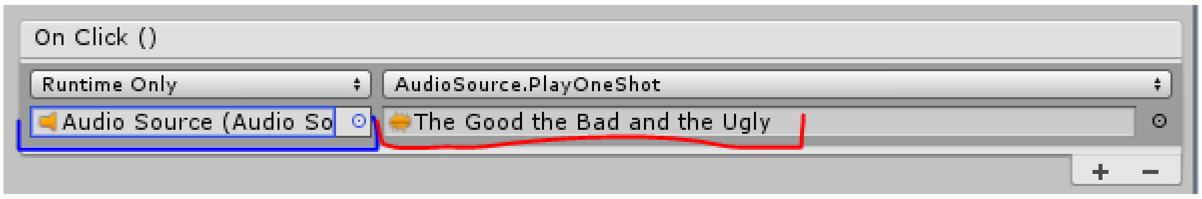
Can be generate in the hierarchy panel after a right-click and choosing in the audio menu the Audio Source field.

Audio Clips

With a simple drag and drop of an audio file from your explorer into the Project-section of Unity will be an audio clip generated. The audio clip which will be showed in the pictures is just a dummy audio and not the audio-file used in the application.

How to play audio-clips by clicking a button

Use the button onclick event and drag and drop the audio-source element into the left placeholder (blue on the image). At next you choose from the function dropdown menu audio-source→playOneShot (Audio clip). The last step is to drag and drop the audio clip which should be played into the right placeholder (red space on the image).



On Click ()

Runtime Only | AudioSource.PlayOneShot

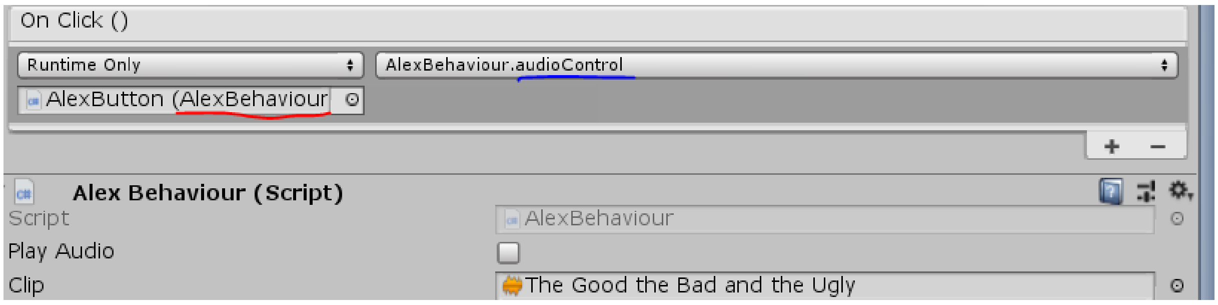
Audio Source (Audio So | The Good the Bad and the Ugly

Going further with a more advanced logic

But this logic is to simple and not flexible enough. There is no opportunity to stop an already running audio clips and every following clip starts another audio clip. This leads to canon of audios but that is not a useful solution for our task.

My workaround for a small but efficient audio system

At the beginning there is the decision to make the audio-source to a child of the Button with the Alex symbol. The second step consists of the implementation of a script that defines the behaviour and the logic of the following operations. This script is component of the button and stores the audio-clip which should be played. In the script it is possible to make use of the function “PlayOneShot” of the audio-source. We will use another function called “Stop” to stop the sound. The following picture shows that this time the function (underlined blue) of the script called “AlexBehaviour”, which is underlined red, will be executed.



On Click ()

Runtime Only | AlexBehaviour.audioControl

AlexButton (AlexBehaviour

Alex Behaviour (Script)

Script: AlexBehaviour

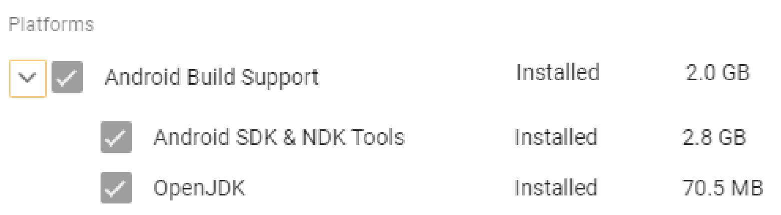
Play Audio: ☐

Clip: The Good the Bad and the Ugly

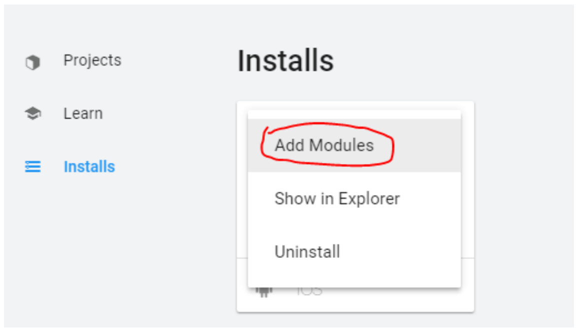
Build to Android Device

Why is it important that you build your application to a device? If a developer wants to test the functionalities of his application in a live situation and not just in the Unity editor, it is necessary to go through the following Steps. Furthermore, it checks the compatibility of the device and the technologies used in the application.

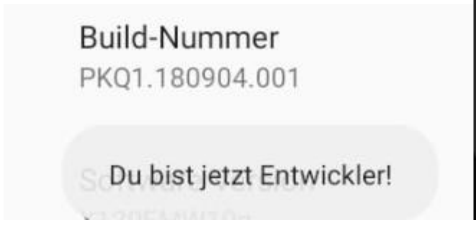
1. Installation of the Android Build Support and the Android SDK and NDK tools using the Unity Hub under installs and choosing Add Modules in the menu.

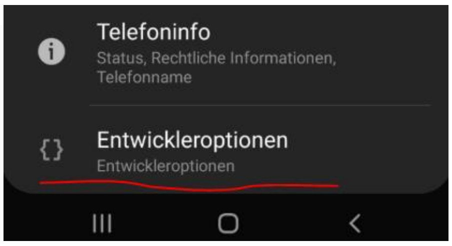

Platforms

Android Build Support	Installed	2.0 GB
Android SDK & NDK Tools	Installed	2.8 GB
OpenJDK	Installed	70.5 MB



2. Enable USB debugging on the android device
   This requires the developer options on the Android device. If they are not yet unlocked, the build number will be searched in the settings of the smartphone. This location varies from device to device but can usually be found in the phone details. After a click on the build number a text appears saying that after a certain number of steps the user becomes a developer. After tapping the build number often enough the developer options are unlocked. Finally you activate USB debugging under these settings.


Build-Nummer
PKQ1.180904.001
Du bist jetzt Entwickler!


Telefoninfo
Status, Rechtliche Informationen, Telefonname
Entwickleroptionen
Entwickleroptionen

3. Changing the platform in the build settings
   Back in Unity after clicking on the "Switch Platform" button, the device on which the application should later run is identified in this window under the " Run Device" item.
4. customizing of the player settings
   Under the item "Player Settings" in the build window you can access the most important settings concerning your target device. The following settings must be made before the build.
   - Under "Other Settings" in the Identification area the Minimum API Level must be at least 24.
   - Because AR-Core is not yet compatible with Vulkan Graphics an error message may occur. This problem can be solved by removing those Graphic API under "other Settings" Graphic API.

Warning possible incompatibility may occur if the SDK version does not meet the requirements of the AR core application.

AR functionalities

In this project, augmented reality is only used based on image recognition. In Unity there is an own subsystem just for this technology and this subsystem is the foundation of the augmented reality functionalities used during this project.

Image tracking with the XR image subsystem

The XR Image Subsystem attempts to detect two-dimensional images in the environment, previously stored in a library of reference images.

Reference Image

These are the images searched for in the real world. When the camera detects an image that is identical to a reference image, a mechanism will be triggered. Each detected image has a position in the real world.

Reference image library

A lot of reference images. This library is needed from the beginning so that the system recognizes which images to search for.

Steps to implement image tracking

1. download the necessary packages from the Package Manager.

The Package Manager is a powerful tool which helps to download packages. Now it is important to explain what is meant with packages in Unity. A package is a container which holds any combination of Assets, Shaders, Textures, plug-ins, icons and scripts enhancing various parts of a project. The package manager can be accessed via the "Window"→Package Manager over the menu panel.

The following packages have been downloaded to ensure the AR-functionalities

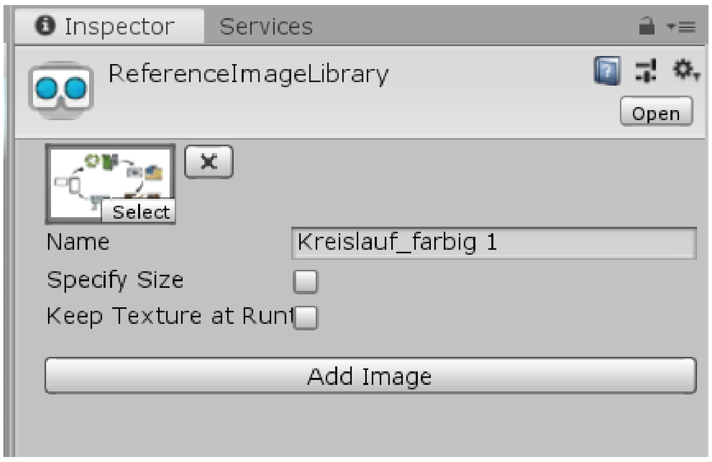
- AR Foundation
- AR Subsystems
- AR Core XR Plugins (for Android)
- ARKIT XR Plugin (for IOs)

2. Create an AR Session Object and an AR Session Origin Object in the Scene

The most important GameObjects in our case are the AR Session Origin and the ARSession object. The ARSession manages the lifecycle of an AR experience by dis- or enabling augmented reality on the target platform. Disabling the ARSession leads to a stop on the tracking of the environment through the system. By enabling it at a later time the system will try to reload and maintain previously detected features of the environment. The AR Session Origin has the function of transform the trackable features, like feature points and planar surfaces, into their final position and orientation in the Unity scene. AR devices tend to provide their data in so called "session space". This unscaled relative to the beginning of the AR session needs to be converted into Unity space by the AR Session Origin.

3. In the Project view, create a library with Create -> XR-> Reference Image Library to store the reference images.

   After creating the library the reference images, for which the image manager will look for should be added to the library. Use just the add image button to add all the reference images.



Inspector Services

ReferenceImageLibrary

Open

Select

Name Kreislauf_farbig 1

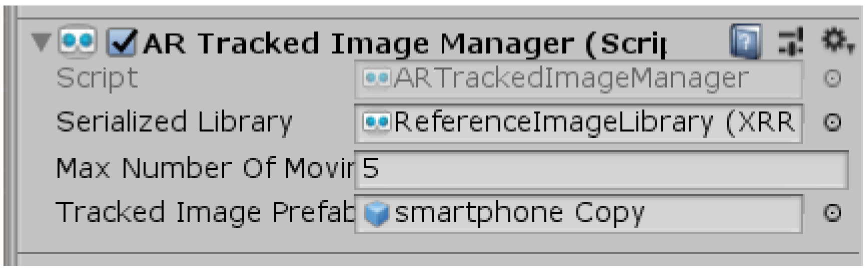
Specify Size

Keep Texture at Runt

Add Image

4. Add an AR Tracked Image Manager to the AR Session Origin object using Add Component.

   This component is required to use world tracking. The tracked image manager creates for every detected image a GameObject. Therefore a reference image library will be attached to the image manager. Only reference images in this library will be detected. The second important attached object is the Tracked image prefab. This prefab will always be instantiated when a desired image will be detected and should be the 3D-model which will then be displayed.



AR Tracked Image Manager (Scrip

Script ARTrackedImageManager

Serialized Library ReferenceImageLibrary (XRR

Max Number Of Movir 5

Tracked Image Prefab smartphone Copy

5. The AR-Camera

The last thing for the minimal setup to make use of AR is to create a AR-Camera in the hierarchy and make it to a child of the AR Session Origin.

Self-reflection in relation of the individual topic and its tasks

This was my first really big project in Unity and therefore an interesting and challenging experience. The historical background of Unity, how it has developed over the years and in the course of the project, were really interesting and educational as well. There have been several moments during this projects where I realized how impressive the increase in knowledge can be if you set yourself a goal and how many different creative workarounds are possible due to the amazing amount of opportunities and different technologies in our modern world today. The creative aspect of having the chance to create an own design showed me how hard it can be to find designs which fits also the target audience. But the biggest development was for me the many hours of research, experimenting, finding workarounds and writing an own small tutorial like documentation to remember and to consolidate the knowledge I gained over the time. All the hours are a big help to find out a lot about myself. What working strategies fits to me and enhance my performance. Which different documentational approaches helps me the best to learn fast completely unknown topics. Moreover it could also be possible that the research skills will develop much faster with every project. This diploma project is just the start to what will wait in the future working world out there. But it can be a great lesson about what should be avoided and what works better in the organisation, the development, design and also the implementation of software these times. Another important lesson which have been learned by me thanks to the dynamical expansion and evolving of Unit is the fact that in this sector is a standstill the biggest mistake what can be made. Every day are modern systems changing, being developed, rebuild and enhanced. In addition, every day offers the opportunity to contribute to this growth in order to create opportunities from which others can later profit sustainably. I am excited of all the hours of developing and being able to change something too as well and this project was just the beginning of this journey.