System porting to mobile devices at the example of the SEE project

Master Thesis

Roman Gressler

Matriculation number: 3217822

July 1, 2022

Faculty 3 — Mathematics and Computer Science
Computer Science

1st Supervisor: Prof. Dr. Rainer Koschke
2nd Supervisor: Dr. Robert Porzel

ERKLÄRUNG

Ich versichere, diese Arbeit — sofern dies nicht explizit anders gekennzeichnet wurde — ohne fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, sind als solche kenntlich gemacht.

Bremen, 01.07.2022

Roman Gressler

ACKNOWLEDGEMENTS

First of all, I want to thank Rainer Koschke and his chair for providing quick help whenever it was needed. I would also like to thank all 20 participants of the user study for taking the time to help me. Most of all, I want to thank my family for always supporting me.

CONTENTS

1 INTRODUCTION

For the last two decades smartphones have been on the rise and caused many things to change in software development. Forster Research predicted one billion smartphone consumers in 2016 (Schadler and McCarthy (2012)). It actually were 3.668 billion and in 2021 it were already 6.259 billion[1]. With the rapid increase in smartphone usage, approaches like mobile-first came up where applications get developed for mobile devices first and other devices like desktop computers second (Wroblewski (2012)).

This thesis however will implement a mobile version of *Software Engineering Experience* (SEE), which already has an adaption for desktop devices. The SEE project aims to connect people regardless of space in a virtual room to analyze code structure. In the current state of the SEE project the virtual room can be accessed with *Virtual Reality* (VR) glasses or a desktop device.

The motivation of this master thesis shall be to integrate mobile devices into the SEE project. Therefore, solutions need to be found to transfer the given operations to mobile devices considering the constraints and benefits of rather small devices. A concept shall be found that integrates the given operations in an innovative way. The main challenges will be finding solutions to the constraints of having a smaller screen and the control operations, that use different input methods than the other systems. After implementing the concept as a prototype, the new ways of operating in the virtual room shall be evaluated in comparison with the desktop version. In conclusion this work shall give insight whether the mobile support is a valuable extension to the system or not. In addition to that, it shall show an approach of how to convert a rather complex system to mobile devices.

SEE: *An interactive software visualization that uses the* code-city *metaphor and enables collaborative multiplayer interactions via multiple platforms like desktop, virtual reality and soon Android devices.*

VR: *A by a computer generated environment. The user sees through a head mounted display and can interact in the Virtual Reality with some kind of controllers.*

1.1 RESEARCH QUESTION

The central research question of this thesis is: *Are Android smartphones suitable of working with SEE?* To answer this question, part of this thesis will be to implement an *Android* version of SEE and evaluate it in a user study. The user study will compare the desktop version with the mobile version of SEE.

Android: *An operating system for portable devices such as smartphones and tablets.*

1 https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/ (last visited: 25.06.22, 22:26)

1.2 THESIS STRUCTURE

This thesis will begin with discussing the fundamentals of this work in chapter 2. This will include the SEE project itself with its use of *Code-Cities* as well as a short introduction to *Unity* and its key concepts needed for the implementation.

Afterward, the concept for the mobile version will be explained in chapter 3. Part of this chapter will be the planned interface and interactions for the implementation as well as a list of requirements that need to be fulfilled for a working prototype.

The concept will be followed by an implementation, which will be unfolded in chapter 4. That chapter will discuss how certain parts are implemented and why they are implemented in that way.

With the given implementation from chapter 4, the earlier announced evaluation will be executed in chapter 5. Therefore, 20 participants took part in a user study and their feedback will be the foundation for the answer of the research question.

Finally, a conclusion of this thesis as well as further ideas for future projects will be given in chapter 6.

Code-City: *In the Code-City metaphor, software components are represented by buildings in a city, and the properties of these buildings can express different metrics of the software. For example, the height of a "Node" could represent the lines of code in that class.*

Unity: *A cross-platform game engine that forms the foundation of SEE.*

2 FUNDAMENTALS

This chapter will cover the basics of this thesis beginning with a short introduction in SEE in section 2.1. Section 2.2 will continue with giving insight on Code-Cities. In the end of this chapter, section 2.3 will be about Unity and its core functionalities that will be used in this thesis.

2.1 SEE

The SEE project aims to connect people regardless of space in a virtual room to analyze code structure. The Unity based project uses various types of code visualization (Koschke (2020), Koschke and Steinbeck (2021)). One scenario uses the metaphor of cities to illustrate the code structure of an application. An example of such an Code-City can be seen in figure 2.1. Such city graphs can be imported from *Graph eXchange Language* (GLX) files. Elements that represent building will be called *Nodes* and an underlying platform will be called *Plane*.

GLX: *A file format for graphs. Is is used for example in SEE to import and export Code-Cities.*

Node: *A point in a diagram where lines intersect. In SEE it usually displays a software class.*

Plane: *An area that bundles Nodes. It could for example represent a namespace.*

In the current state of the SEE project the virtual room can be accessed with VR glasses or a desktop computer. This thesis aims to add an implementation of SEE for Android devices. This shall help developers to collaborate on software projects as freely as possible.

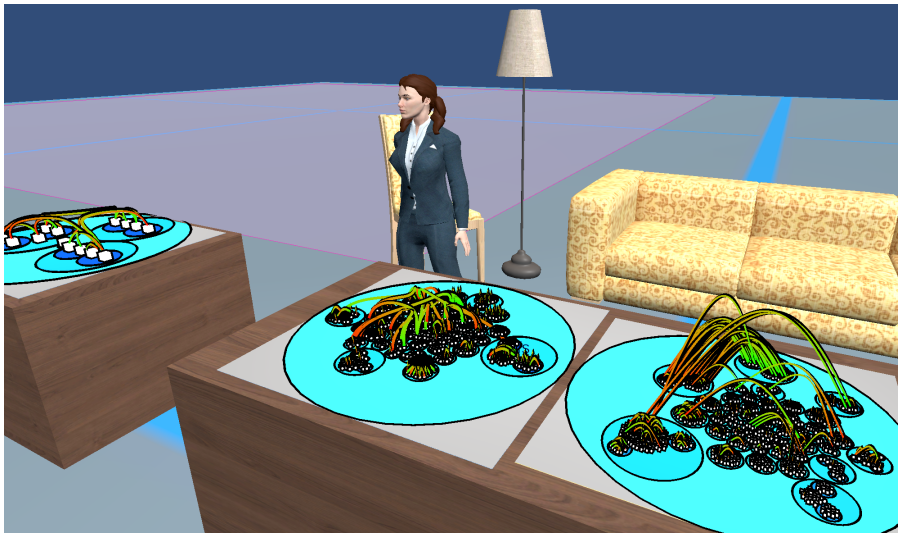

Figure 2.1: An example of SEE used for the user study in chapter 5

2.2 CODE-CITIES

A Code-City is an approach of visualizing software projects. An example of a Code-City by Wettel and Lanza (2007) can be seen in figure 2.2. Since the representation is three-dimensional, many metrics can be visualized in a single Code-City. Code-Cities could use, as an example, the metrics number of classes as building height, number of attributes as base size and the nesting level of a package as the building color (Wettel and Lanza (2008a)). The Code-City provides an overview of the represented system and by walking around it in SEE the user can get an idea of the structure of the represented system. One challenge of representing large code bases is the overwhelming amount of information. Therefore, using metaphors as the Code-City aims to avoid overwhelming the user with too much abstract information (Wettel and Lanza (2008b)).

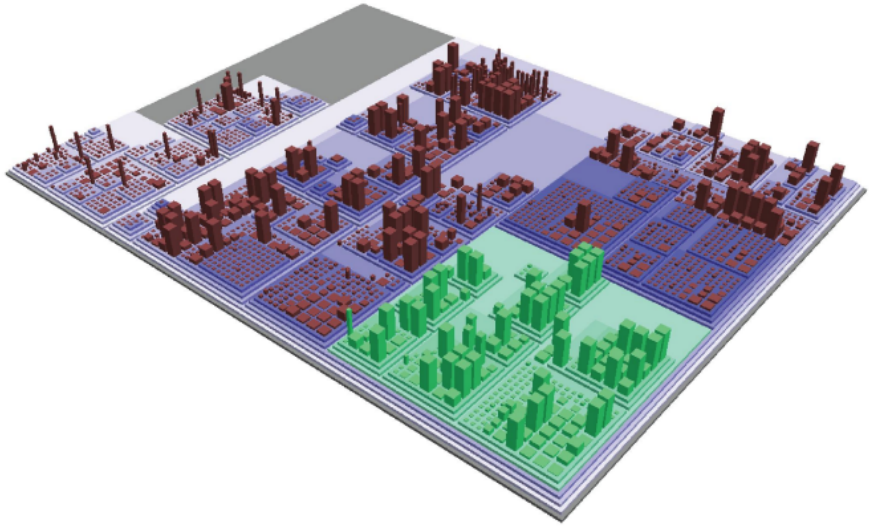

Figure 2.2: An example for a Code-City by Wettel and Lanza (2007)

2.3 UNITY

Unity is a cross-platform game engine that has been released in June 2005 as *Unity 1.0*. The engine is not only used for games but also for other industries regarding architecture, automotive or film[1]. Unity started to become successful with its launch in the Apple App Store. This is due to the growing popularity of mobile video games and Unity being one of only few game engines that are optimized for Apple. The game engine grew ever since and has grown its support for multiplatform development (Nicoll and Keogh (2019)).

1 https://unity.com/ (last visited: 22.06.22, 2:39)

2.3.1 *Ray Casting*

Ray casting is a basic computer graphics rendering algorithm. Rays are cast to determine on their path what can be seen by the camera perspective. In the example of figure 2.3 a local camera coordinate system can be seen. The lines from the origin on the right are rays that determine the view plane of $z = 0$.

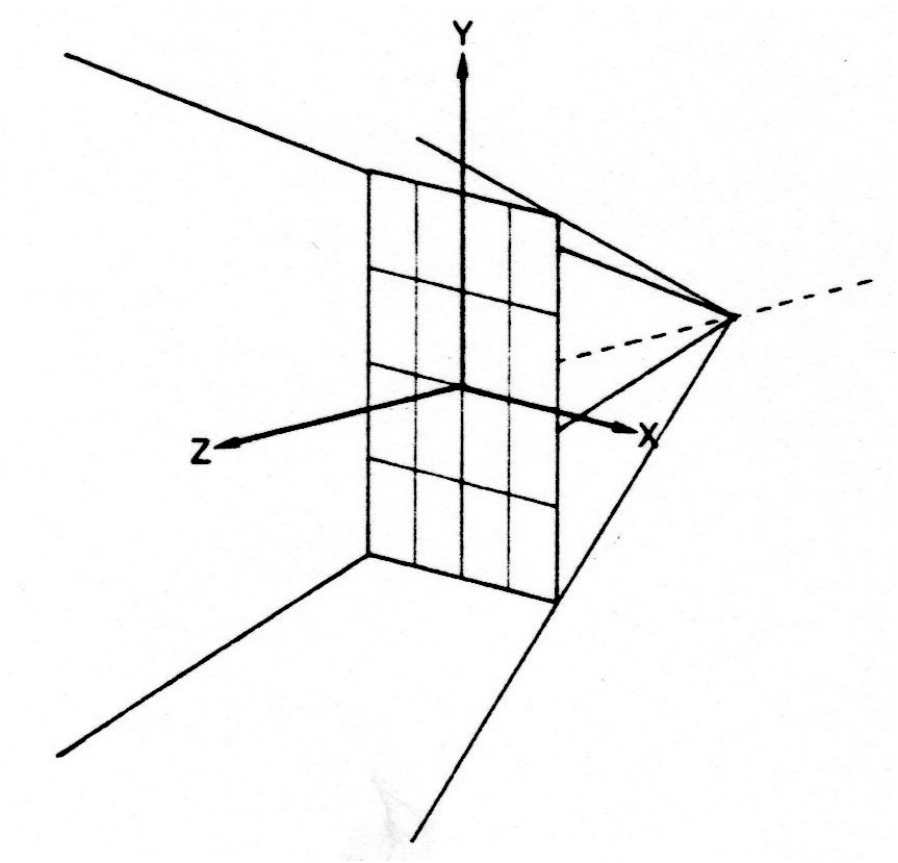

Figure 2.3: A local camera coordinate system by Roth (1982)

Ray casting will also be used to detect the closest object from a certain point in the camera's perspective. For example objects will often be selected by touch and to determine which object is touched ray casting is needed. Therefore, a ray will be cast from the center of the touch input and call back the closest object. This could also be used to determine if a touch input is on a Code-City to allow rotating or moving interactions.

2.3.2 *Debugging and Testing*

Building another version of a consisting application can take a lot of testing. Testing in Unity for Android applications can be escalated in three steps. Debugging always works in combination with the *Microsoft*

IDE: *An application that provides software developers with tools like a source code editor, build automation and a debugger.*

Visual Studio[2] debugger, an *Integrated development environment* (IDE) for multiple coding languages including *C#*.

The first and easiest method can be done directly in the Unity Editor by using the core feature *Play Mode*. In *Play Mode* the project starts and builds as it would in a final build. Any changes made in *Play Mode* will not be saved and will be reset after leaving *Play Mode*. However, the building time in the *Play Mode* is much faster than the actual Android build. Unfortunately touch input can not be tested on a desktop computer, but the *Play Mode* suffices for testing UIs or other basic changes.

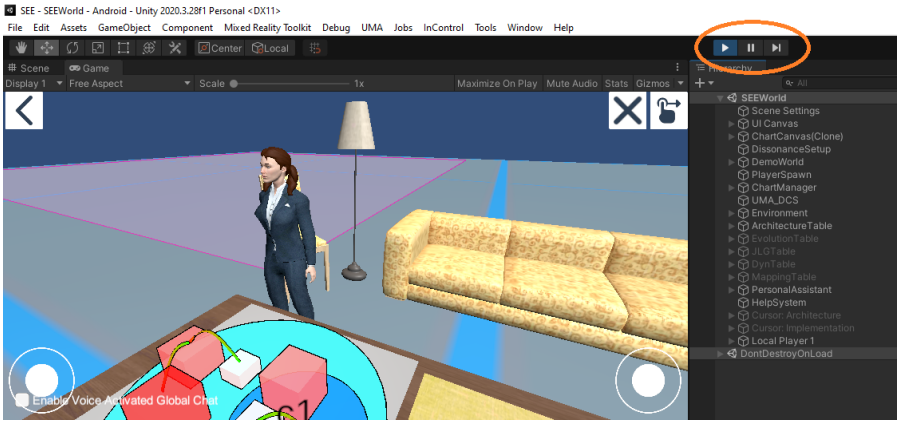

Figure 2.4: The *Play Mode* can be entered by pressing play at the marked spot

Scene: *A collection of components that form an overall picture. A scene could contain for example environmental or UI components.*

However, if actions with touch input need to be tested, the second method is suitable. Therefore, Unity provides the app *Unity Remote*[3]. The app can be connected via USB with the Unity Editor. After hitting the play button the *Scene* will also be displayed on the connected device and also take input. Besides touch input other factors for a mobile application like the accelerometer, gyroscope or the camera can be tested. The application however will be build on the desktop computer and not on the Android device.

To test the application, for example for bugs that cause a crash on some Android devices, the application has to be build and then installed on the device. The device can then be connected with the IDE via Wi-Fi or USB. Unity also provides an option to let the built application wait for a connection with the IDE, since the connection can take some time. This however is more time-consuming than the other two methods. A Unity Android build can take, depending on the project size, more than five minutes. It should therefore only be a last option and only be used to debug concerns regarding the devices hardware constrains.

2 visualstudio.microsoft.com/ (last visited: 24.06.22, 0:12)
3 https://docs.unity3d.com/Manual/UnityRemote5.html (last visited: 24.06.22, 0:14)

3 CONCEPT

In this section a concept of a mobile SEE version will be presented. Therefore, a prototype will be created to point out the features that a mobile version of SEE requires.

Prototypes are a common way to express the needs of a system. It is a low-cost way of planning an implementation, that can highlight challenges regarding constraints of a system early on.

Even though a prototype will never be able to show every aspect and need of a complex system, it should still help to answer questions like: How should the system feel? How should it be implemented, and what are the key features? (Houde and Hill (1997))

SEE is meant to be used by multiple platforms such as desktop devices, mobile devices and virtual reality devices. Each device has different interaction constrains. While a desktop user will control the player with mouse and keyboard, a mobile user will interact with virtual joysticks on a touchscreen. Selecting Nodes of a Code-City will be done by clicking on it with a mouse on desktop devices, while a mobile device will require a touch input.

3.1 INTERFACE

In the following a paper prototype will be presented that marks out a concept for the mobile interface. Since the field of mobile development is quite young there are only few guidelines regarding the design of mobile device interfaces. A guideline that is widely accepted is problematic to find (Renaud and Van Biljon (2017); Punchoojit and Hongwarittorrn (2017)).

Major differences to desktop environments are the screen size, forms of input and input feedback. To ensure that as much space as possible is used for the actual interaction with the app, the menu should only take up as much space as necessary. As a study has shown, a size of at least 8*8 mm is needed to reduce error rates selecting the right button (Conradi et al. (2015); Parhi et al. (2006)).

Moving the player will be handled with virtual joysticks as seen in figure 3.1. The left joystick will move the player through the virtual room and the right will move the camera angle or in other words the direction the player looks at. The joysticks are placed in the left and right corner and should just take as much space as needed to be handled comfortably. This way the players are able to navigate through the

virtual room with their thumps while still having enough space to work on the Code-City.

Figure 3.1: Joysticks for moving in SEE

The menu on the top left side seen in figure 3.2 will be called "quickbar" further on. The *quickbar* can be minimized to save screen space when not needed. The *quickbar* is designed to offer more general functions that are needed in various situations. Because there are no shortcuts on mobile devices each function has to have a button to be activated.

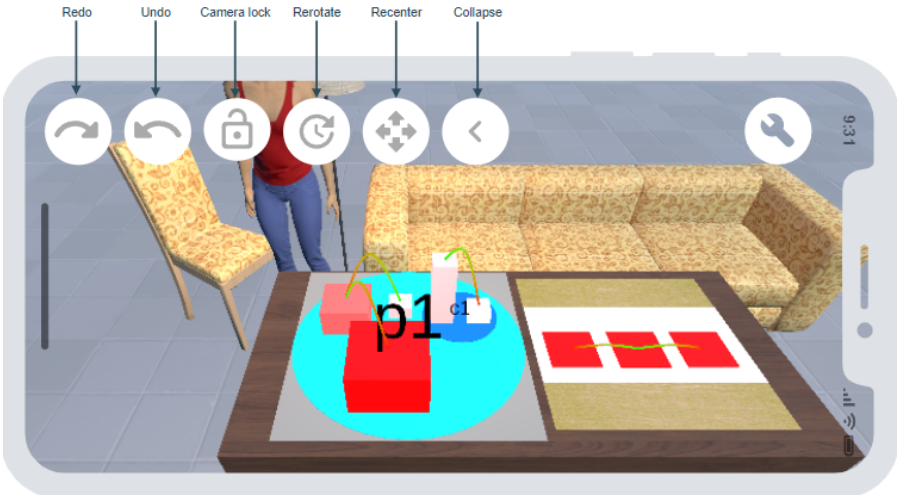

Figure 3.2: The *quickbar* for various interactions in SEE

The functions are *undo* and *redo*, which will revert an action or do a reverted action again. Then there is a camera lock that will lock the player's perspective to a certain Code-City so that the player can only move around the selected city and move closer or further away from it. The next function is to rerotate a Code-City. That means the Code-City that was last rotated will be set back to its initial state of rotation. Last but not least, there will be a button for recentering the city, which will work quite similar to the rerotate button and center the last moved

Code-City. The button on the right can be used to collapse or minimize the *quickbar*.

On the top right side another menu will be placed that contains different interaction modes. By clicking a button an interaction mode will be selected and moved to the top right corner. Also, the menu will be collapsed and only the buttons regarding the selected interaction mode shall be shown. By clicking the button on the top right again the menu shall expand, and the other interaction modes shall be selectable. The other buttons shall be kept in the same order to reduce confusion of the user.

The first interaction mode, seen in figure 3.3, is for selecting Nodes. Nodes can be selected by being touched and deselected by being touched again. There can be multiple Nodes selected at once. The hole selection can be deselected by clicking the deselect button next to the select interaction mode button. Selected Nodes shall be highlighted with a different Node color and also display their name.

Figure 3.3: Selection mode in SEE

The second interaction mode, seen in figure 3.4, is for deleting Nodes. It does not need additional buttons. Nodes will be deleted by being touched. Unlike in the desktop version, there will not be a group deletion interaction because it would require an additional menu panel. The added functionality would be minimal and selecting a group of Nodes, confirming and finally deleting would require a handful more steps and would therefore most likely not be used.

The following interaction mode, seen in figure 3.5, is dedicated to the Nodes and *Edges* of a Code-City. The Node interactions start with the "add Node" button on the right. When the button is activated the user can create new Node by clicking on a certain spot on the Code-City Plane. The following button on the left is for adding Edges. By selecting two Nodes a new Edge will be created between them. Then, the button one further on the right is for editing Nodes. By touching a Node a window will pop up that allows the user to edit the Node by changing

Edge: *A line that connects two Nodes. Could for example represent a class call.*

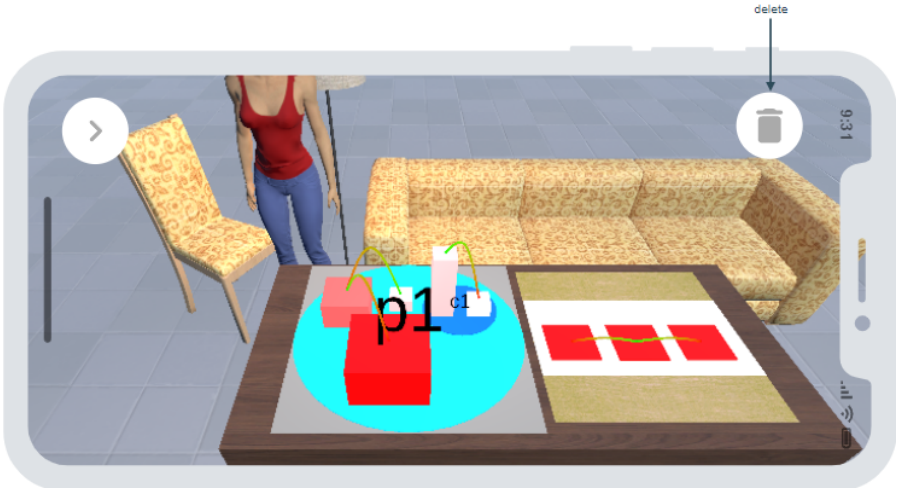

Figure 3.4: Delete mode in SEE

its name and its type. Last but not least, the button on the left-hand side will be used to scale Nodes. That means the Node height and width can be adjusted by first selecting it via touch and then hold a corner and drag it further away from the Node center to increase the size or drag it towards the center to decrease the size of the Node. Each button of the Node interactions will be marked green after being pressed to indicate that it is active.

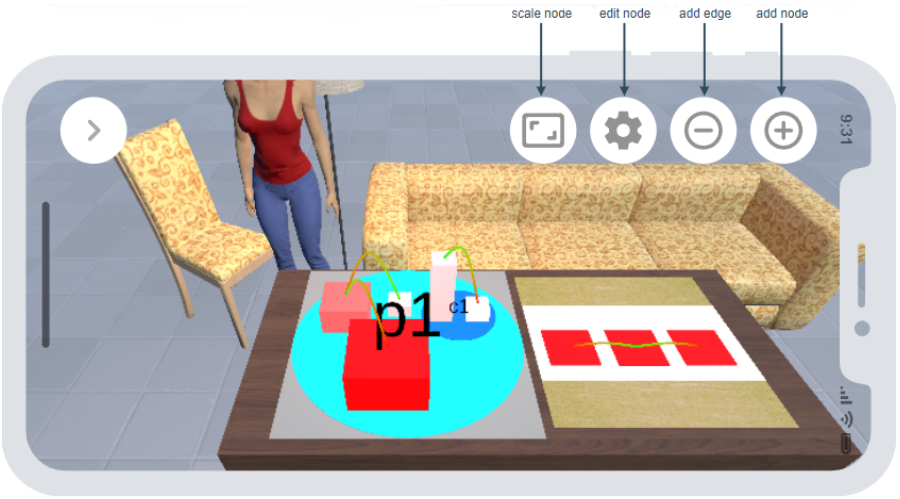

Figure 3.5: Node interactions in SEE

Then, there will be a button panel for rotation interactions that can be seen in figure 3.6. The rotate interactions start with the first activatable button "n", that lets the user rotate the whole Code-City by touching any point on it and then drag it away from that point. Similar to that, there will be a button that lets the user rotate just a single Node on the Code-City. Additionally, there will be a button that activates the so-called "locked-rotation" mode. While in "locked-rotation" mode, the rotation of a Node or Code-City will be done in eight predefined

steps to a full rotation. Each step will have the same 45° range. The last button of this group will be for changing the center of the rotations. There are two options: the first option is a center of rotation in the middle of the Code-City and the second is in the middle of a Node selection made with the interactions seen in figure 3.3. The second option can be activated by pressing the last button.

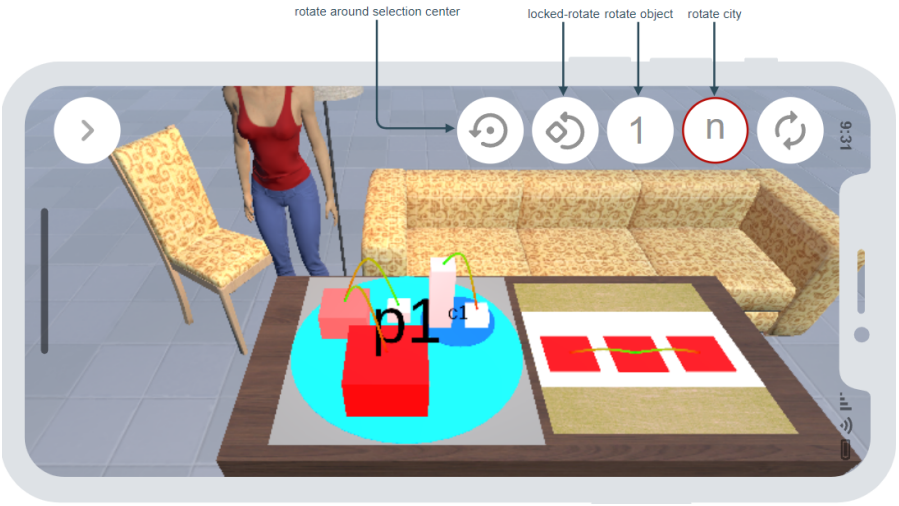

Figure 3.6: Rotation mode in SEE

The last interaction group, seen in figure 3.7, is for moving the Code-City or a single Node. The move interactions are quite similar to the rotation interactions. There will be a button to move a whole Code-City as well as a button to move only single Nodes. In addition to that there will be a button that restricts the movement of the Code-City or node to a predefined direction. The directions will be again in 45° angles and objects can be moved on a straight line on that angle. Moving a Node or a Code-City can be achieved by touching and holding it and then moving it to the desired position.

3.2 INTERACTION

Smartphones are quite limited in space and there are few input possibilities. Unlike a desktop computer there is no mouse and there is no physical keyboard. Smartphones use virtual keyboards, but due to the restriction of screen space the keyboard is hidden most of the time. This would make keyboard shortcuts uncomfortable because the user has to open the keyboard first. Therefore, smartphones need different ways of interaction such as touch gestures.

Zooming in to a Code-City happens by scrolling on a desktop environment. There is no option to scroll on mobile devices, but there are at least two popular alternatives. The first option would be to double tap on the Code-City. The double tap would zoom in, in predefined steps and after reaching a certain level of closeness it would trigger

Figure 3.7: Movement mode in SEE

to zoom out again. In SEE, zooming in at predefined steps might not be precise enough because there could be a fairly large Code-City or a rather small one. Finding predefined steps that would fit every situation is rather hard. Therefore, a second option by zooming in with a two finger gesture might be better. In this option the user uses two fingers and slides them towards each other to zoom in or slides the two fingers away from each other to zoom out. This way there are no predefined steps necessary and zooming interactions can be done precisely.

3.3 REQUIREMENTS

In the following a list of requirements will be given, which will specify in detail what the implementation of a mobile version has to take care of. The list will be referred to multiple times in the upcoming realization part in chapter 4. Requirements are essential for the planning phase as they give a good fundamental structure for the developer to rely on (Robertson and Robertson (2012); Stevens and Pooley (2005)).

[R1] The application shall run on Android devices

[R2] The application shall be controlled via touchscreen

- [R2.1] The player and camera shall be moved with virtual joysticks
- [R2.2] Needed shortcuts of the desktop version shall be handled with buttons
- [R2.3] Zooming shall be handled with a two finger gesture

[R3] The user shall be able to select a Node of a Code-City

- [R3.1] After the selection the name of the Node shall be shown
- [R3.2] The user shall be able to deselect single Nodes or a group of Nodes

[R4] The user shall be able to delete Nodes

[R5] The user shall be able to interact with Nodes

- [R5.1] The user shall be able to add Nodes
- [R5.2] The user shall be able to add Edges
- [R5.3] The user shall be able to edit Nodes
- [R5.4] The user shall be able to scale Nodes

[R6] The user shall be able to rotate a Code-City

- [R6.1] The user shall be able to rotate a Code-City in 45° steps
- [R6.2] The user shall be able to rotate single objects
- [R6.3] The user shall be able to rotate around a center of selected Nodes
- [R6.4] The user shall be able to undo the rotation

[R7] The user shall be able to move a Code-City

- [R7.1] The user shall be able to move single object of a Code-City
- [R7.2] The user shall be able to restore the Code-City initial position
- [R7.3] The user shall be able to move a Code-City or single node in predefined directions

[R8] The user shall be able to undo and redo actions

[R9] The user shall be able to lock the camera to a selected Code-City

4 IMPLEMENTATION

This chapter will discuss the implementation of the mobile version of SEE, starting with the mobile player implementation in section 4.1. Section 4.2 will give insight on the player movement before section 4.3 will discuss the mobile menu. Next, the implemented player action will be introduced in section 4.4. This chapter will end with pointing out the requirements for an Android build and how they have been handled.

4.1 MOBILE PLAYER

In this section the mobile player *Prefab* will be discussed. SEE is supported on multiple platforms and with each platform having different requirements, each platform needs to be divided. Therefore, each platform gets its own player Prefab. The mobile player prefab for the Android version of SEE can be seen in figure 4.1.

Prefab: *Predefined GameObject that can be loaded into a Scene.*

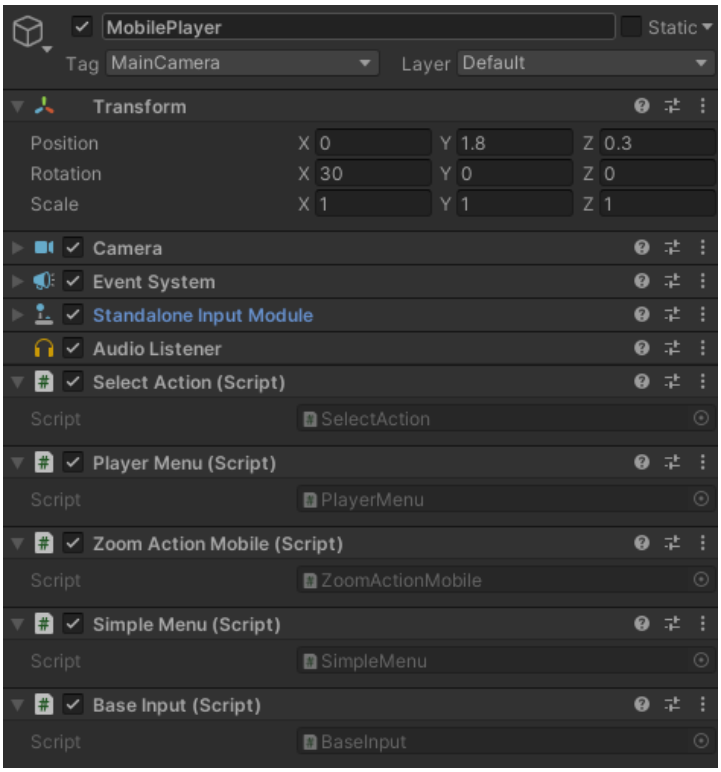

Figure 4.1: The mobile player Prefab

The Prefab consists of the basic parts provided by Unity *Camera*, *Event System*, *Standalone Input Module* and *Audio Listener*. In addition to that the following custom scripts were added. There is the *SelectAction* and the *ZoomActionMobile* script, which will both be discussed in section 4.4, and then there is the *PlayerMenu* script, which creates the right menu based on the player type. In this case the player type will be "mobile player" and the menu created (here *SimpleMenu*) will be discussed in section 4.3.

4.2 PLAYER MOVEMENT

After having created a player, the player also has to be able to move. Therefore, the *MobilePlayerMovement* script got added. The script handles the input by the joysticks seen in figure 4.2. To fulfill [R2.1] the player will be moved by using the left joystick and the player's perspective will be handled by the right joystick.

Figure 4.2: The joysticks are for moving in the virtual room. The left joystick is for moving the player and the right one is for moving the player's perspective.

Asset: *Can be any type of media that can be used in a Unity project.*

For the joystick Prefabs the *Joystick Pack*[1] *Asset* is used. It contains the design and basic logic for the joysticks. A joystick returns a horizontal and a vertical value depending on how far the joystick is dragged into a direction. The left joystick data can then be transformed into player movement as follows:

1. Get the horizontal and vertical values from the joysticks
2. Transform the values into a 3D vector
3. Combine the values into a velocity vector
4. Normalize the velocity vector for a smooth transition

1 https://assetstore.unity.com/packages/tools/input-management/joystick-pack-107631#description (last visited: 17.06.22, 15:21)

5. Transmit the velocity vector to the player position value

The data of the right joystick shall move the player's perspective, which is implemented as a Unity camera. The camera has a pitch angle and a yaw angle as illustrated in figure 4.3. The angles of the camera will be adjusted with the input from the right joystick. For the angles of the player's perspective there is a range from 0° to 360°, after reaching an end of this range the value will be transformed to the other end of the range. In other words, if the angle becomes higher than 360° it starts at 0° again and if it gets lower than 0°, it can go further down from 360°. In that way the player's perspective can move a full rotation.

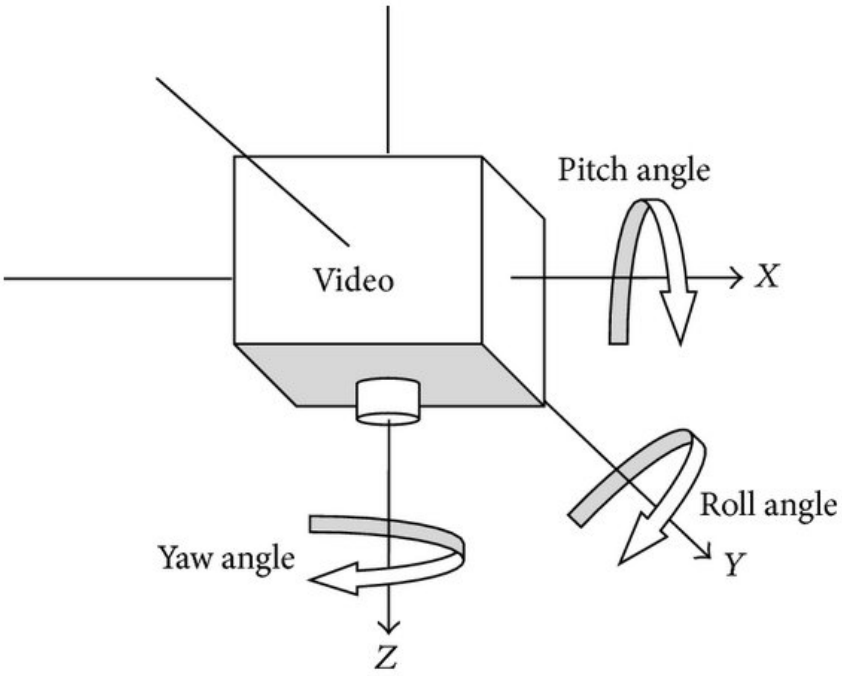

Figure 4.3: The angles of a camera by Zhang et al. (2014)

4.3 MOBILE MENU

The mobile menu is essential for the implementation since the input methods of a smartphone in an everyday usage are limited. The desktop version uses many *Shortcuts*, which cannot be used in the mobile version. These Shortcuts will be replaced with buttons in the menu to fulfill requirement [R2.2].

Shortcut: *A combination of key that will call an action like for example "ctrl" + "c" for copying a text.*

The menu will be divided into two parts. The first part, which was named *quickbar* in section 3.1, will be responsible for all interactions that need to be available at all times. The implemented *quickbar* can be seen in figure 4.4. By pressing the button on the right end of the *quickbar*,

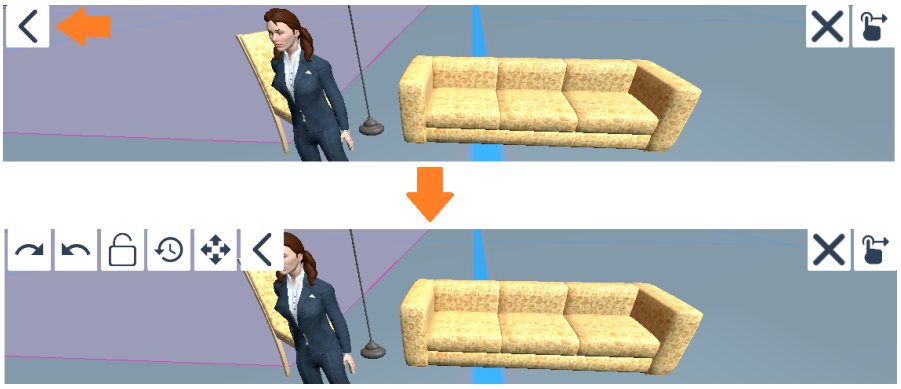

Figure 4.4: The *quickbar* on the left top side of the mobile device. Pressing the button with an orange marked arrow will expand the menu.

the menu can be expanded and minimized to safe screen space when not needed. The menu also contains buttons for redoing and undoing actions to fulfill requirement [R8]. In addition, there is a button for a "locked-mode". Unfortunately, the button is just a placeholder since the functionality to lock the players view to a city is not available in the desktop version at this time. Therefore, requirement [R9] can not be completed at the time of this thesis.

The other part of the menu can be seen in figure 4.5. The menu is placed on the right side of the screen, and it contains all the player interactions. This part of the menu is slightly more complex since the selected button moves to the top, while the other buttons shall remain their initial order.

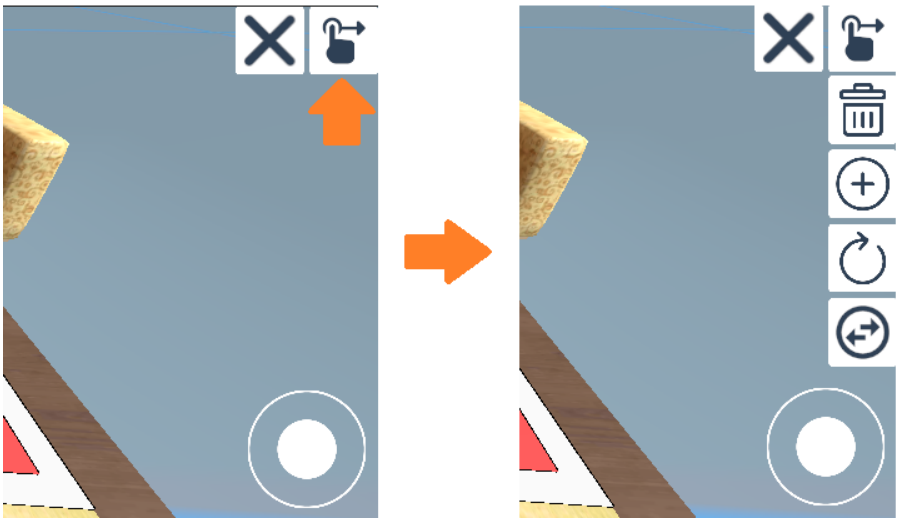

Figure 4.5: The player interaction menu on the right top side of the mobile device. The button on the top right side indicates the active interaction mode. Pressing the same button also expands the menu.

The implementation of the menu on the right is done in the following steps:

1. All main buttons get an index
2. When a button on the far right side gets clicked, the index gets saved
3. The button group with the selected index get shown on the top right side
4. All other buttons get disabled
5. By clicking the button on the top right again, the other main buttons get reactivated
6. The order is as follows: selected index → index in ascending order

This way the selected interaction mode will always be in the top right corner, while the other buttons remain the same order.

4.4 PLAYER ACTIONS

One essential part of the implementation are the player actions. Almost all interactions a user can make in SEE differ from the interactions in the desktop version. In the following, the implementation of all mobile player actions will be discussed briefly.

4.4.1 *Zooming*

To fulfill requirement [R2.3], zooming needs to be implemented. The interaction of zooming in or out of a Code-City can be fundamental when working with a large Code-City because Nodes become small and especially on a small mobile screen it becomes impossible to interact with those Nodes via touch input. Therefore, the user has to zoom in to properly interact with the desired Node.

Luckily the general zooming function is already implemented in SEE. The zooming method requires a center point and a scale of how far the Code-City shall be zoomed in or out. Zooming on a touchscreen will be done by dragging two fingers either towards or away from each other as visualized in figure 4.6.

The implementation of the zooming interaction in done in the following steps:

1. Check if there are two touch inputs on the same Code-City. If both inputs are not on the same city, zooming is not wanted and will not be activated.
2. Compute the center of the two touch inputs
3. Pass the center and dragging range of the two inputs to the zooming function

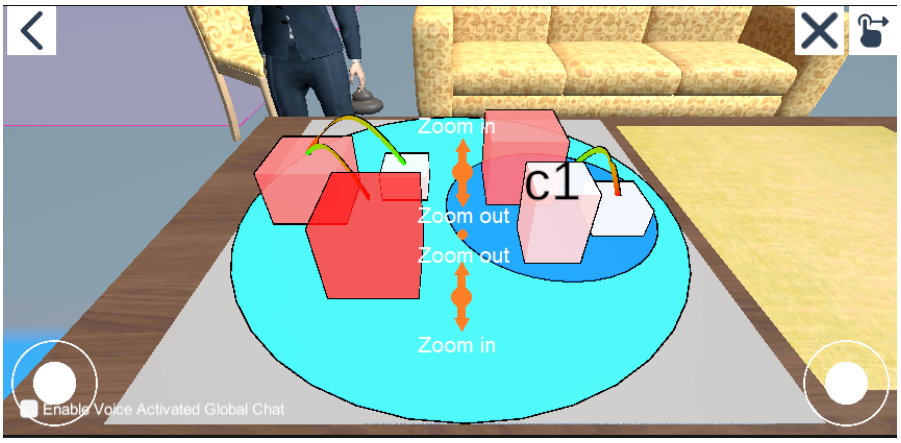

Figure 4.6: Dragging the two touch points towards each other will zoom out and dragging them away from each other will zoom in.

Note that the direction of zooming has been inverted in the version used for the evaluation in chapter 5. This is because of the heavy demand by the subjects to invert the direction of zooming since it felt odd.

4.4.2 *Selecting and Deleting*

The interactions of selecting and deleting a Node or Plane are quite similar because the user selects or deletes an object by touching it. The object get determined by a ray cast from the center of the touch position as discussed in section 2.3.1.

Since the selecting mode is always enabled except when the delete mode is active, the select button really does nothing apart from ensuring that no other interaction mode is active. Keeping the selected objects active makes sense for the mobile version because the user cannot hover with a mouse to see the object names. Therefore, the selection will be kept and not discarded by selecting a different object as in the desktop version. The deselect button calls the already implemented *UnselectAll* method that empties a list of selected items.

The delete-interaction works as already mentioned just like in the desktop version. The only difference is the type of input. Other than in the desktop version, the mobile version only supports single deletions and not the deletion of multiple objects at once. This is due to the limited space for the interactions menu and the results from a multi deletion can also be achieved with the single delete interaction.

The discussed implementation fulfills requirements [R3] and [R4].

4.4.3 *Node Interactions*

The Node interactions consist of four types. The active interaction is always highlighted in green as seen in figure 4.7

Figure 4.7: The Node interactions menu. The active Node interaction has a green button

The circled plus button stands for adding a Node. Adding Nodes is implemented in the following steps:

1. Check if the device is an Android with a preprocessor tag
2. Check if there is exactly one touch input
3. Cast a ray and check if the collided object is of the type "Node" (SEE does not distinguish between Plane and Node as a type)
4. Create a Node at that position with the already implemented *AddChild* method of the class *GameNodeAdder*

The circled minus button is for adding an Edge. The implementation is quite similar to the one for adding Nodes. In this case however the first and second *GameObject* of type "Node" will be saved. After there are two GameObjects, an Edge between those objects will be created.

GameObject: *A fundamental object in Unity Scenes.*

Next in line is the gear button, which can be used for editing Nodes. The implementation here is almost the same as for the desktop version. A Node or a Plane can be selected by touch and a window will open. In that window attributes like the name can be changed. Different from the desktop version is that a virtual keyboard will pop up to let the user type in a new name for example.

Last but not least, there is the interaction of scaling a Node left. The implementation here was largely adopted from the desktop version, except again the input type is via touch. In addition to that, the dots that can be selected and dragged are larger in the mobile version (see figure 4.8), because otherwise it would be hard to hit them with a touch input.

The listed implementations of the Node interactions fulfill the requirements of [R5].

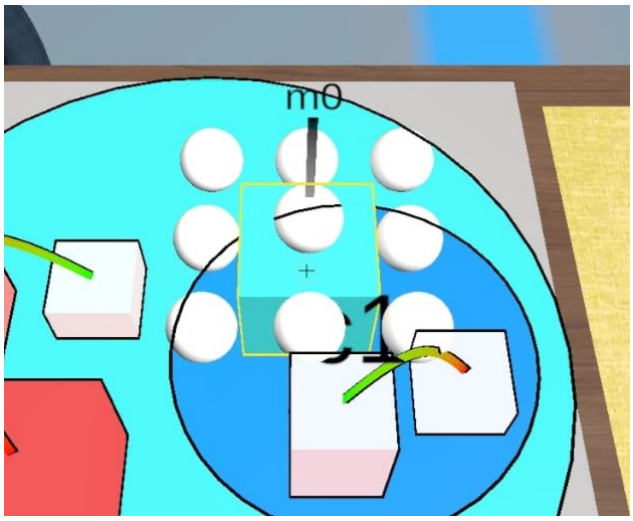

Figure 4.8: The Node interactions menu. The active Node interaction has a green button

4.4.4 *Rotating*

To fulfil the requirements of [R6], rotating has to be adapted to the mobile version of SEE. The user can rotate the city either freely or in eight predefined steps. Figure 4.9 shows a rotation in those eight predefined steps. As it can be seen the "8" button is green which means that it is active. The user can drag from a certain point on the Code-City towards a direction the Code-City shall be rotated to.

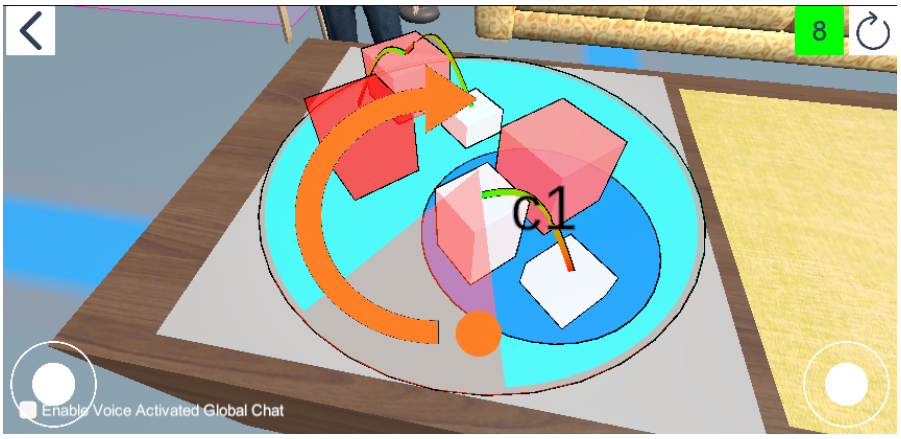

Figure 4.9: The Code-City can be rotated by for example touching the screen on the orange dot and dragging from there like the arrow indicates.

The implementation for the interaction of rotating is done in the following steps:

1. Check if there is exactly one touch input on an object of the type "Node".
2. Check if the "8" button is active

3. Calculate the new position of the Code-City just like in the desktop version of SEE

4.4.5 *Moving*

Last but not least, the player action of moving a Code-City or object has to be implemented to fulfill the requirements of [R7]. The user can touch a certain point on a Code-City and drag it to move it as visualized in figure 3.7. The same can be done with any Node or Plane by activating the "1" button. Same as for the rotate-interaction, the user can also activate the "8" to move an object or city in one of eight predefined directions. Also, by selecting the "1" or "8" button, it will turn green to indicate that it is active.

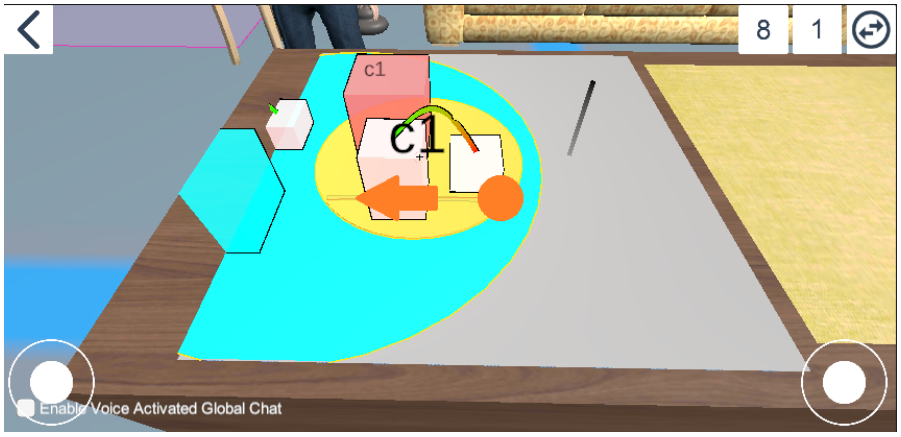

Figure 4.10: The Code-City can be moved by touching and dragging it to a desired direction.

The implementation is done similar to the one for the rotate-interaction. The only thing added here is the check whether the "1" button is active or not.

Note that for the rotate and move-interactions the "n" buttons have been removed (see figure 3.6 and figure 3.7). This is because these buttons are not needed, since the initial mode, when entering the interaction, is already for the whole Code-City. Also, the rotate object button and the rotate around selection center button have been removed. Rotating around a selection center will always be activated and can be deactivated by removing the selection of objects. Rotating single objects is not intended as an interaction in SEE.

4.5 ANDROID BUILD REQUIREMENTS

The last part of this chapter will discuss the requirements to build a version of SEE on Android to fulfill the last requirement [R1]. In the

following, there will be a deeper look at the challenges porting SEE to Android devices brings.

4.5.1 *Conditional Compilation*

SEE uses, as it is usual, many Assets for a medium to large Unity project. Unfortunately, not all Assets support an Android implementation. Therefore, the unsupported assets have to be exchanged or excluded from the implementation. Some unsupported Assets are *Valve* and *UnityEngine.Windows.Speech*. The *Valve* Asset is required for the VR implementation of SEE and *UnityEngine.Windows.Speech* is fundamental for the speech assistant "See". Since the Assets are not supported but still needed in the project, they need to be excluded.

Unity offers a way to compile only partial code. Therefore, it uses the *directives*[2] of the C# language. These use the hash character (#) followed by *if* or *endif* to mark code areas with a condition. Unity has platform scripting symbols like *UNITY_ANDROID* to set conditions for code areas to be compiled for a certain device or not. The following code example shows how the technique is used in this project.

```
using SupportedPackage
#if not UNITY_ANDROID
using NotSupportedForAndroidPackage
#endif

    SomeClass
    {
    ...
    SomeSupportedPackageMethod()
    ...
    #if not UNITY_ANDROID
    SomeNotAvailableMethodForAndroid()
    #endif
    ...
    }
```

Examples like this can be found many times in the SEE project. In addition to that, there are some cases where the main code is exchanged, whether the device is an Android or not. For example the player actions discussed earlier in this chapter require touch input and can not use the hover effect of a mouse cursor on a mobile device. Therefore, on mobile devices the touch input code will be used and on desktop devices the mouse hover code, but both code parts use the same methods and variables. Having two separated classes would make up redundant code. In opposition, excessive use of conditional compilation makes it

2 https://docs.unity3d.com/Manual/PlatformDependentCompilation.html (last visited: 21.06.2022, 1:27)

harder to maintain code because it might have to be adjusted at multiple points for a single change. This way part of the code could easily be forgotten and lead to bugs.

4.5.2 *File Loading*

One last challenge for the mobile implementation of SEE was how files are handled in the project. SEE uses specify file space for Unity called *StreamingAssets*[3] to store for example the various states of an Evolution-Code-City. An Evolution-Code-City visualizes multiple states of development. Only the first state is preloaded, the other states are stored in the *StreamingAssets* and are loaded at runtime. The problem however is that an Android application is packed into a *Java ARchieve* (JAR). Files inside a JAR cannot be obtained without further ado. To achieve access to files that are not initially loaded, *WebRequests*[4] have to be used for an Android implementation. Therefore, once again the code needs to be divided and handled with conditional compilation. This has to be done at every point in the code where files are obtained the common way with *SystemIO* methods.

JAR: *A package file format that is typically used to aggregate multiple Java class files and appendix into one file.*

4.5.3 *Restructuring*

The disadvantage of conditional compilation is that it makes code harder to maintain as discussed earlier. To minimize this disadvantage, it was thought of restructuring the entire project to only have conditional compilation at a core component of SEE.

The concept idea was to restructure the project in a way that from a core component all the different versions get initialized and only in that core component conditional compilation gets used. All components would have abstract super classes that could be easily supplemented with assets that are not available for all platforms. The platform specific classes would then only be compiled for a certain platform build, since they are divided in the core class.

Unfortunately, SEE is already entangled and a restructuring in the described way above would take a high effort. SEE already has many components that depend on another part of SEE. These parts however also depend on other parts of SEE. Restructuring and setting a new order for the parts of SEE that support the idea of a core part of SEE would take too much effort for this thesis and was therefore not implemented.

3 https://docs.unity3d.com/Manual/StreamingAssets.html (last visited: 21.06.22, 2:47)
4 https://docs.unity3d.com/ScriptReference/Networking.UnityWebRequest.html (last visited: 21.06.22, 3:13)

5 EVALUATION

In the following chapter the mobile implication of SEE will be evaluated in a user study. Therefore, the mobile application will be compared with the desktop version.

This chapter will start with a description of the desktop version and its main differences in section 5.1. Further on, a defined aim and precise hypotheses for the user study will be explained in section 5.2. After sketching the first experiment set up in section 5.3, the actual experiment set up will be discussed in detail in section 5.4 including the used survey tool, questionnaires and the pilot study. This chapter will then be closed by discussing the results of the user study in section 5.5 and talking about the threats of validity in section 5.6.

5.1 SEE DESKTOP

In this section the desktop version of SEE will be explained. In this evaluation the mobile version of SEE will be compared with the desktop version. Therefore, it is necessary to take a deeper look at the differences between those two versions. Especially at how the interactions differ and what impact it could have on the user experience.

One outstanding difference from the desktop version to the mobile version is the selection of the interaction modes. While in the mobile version the menu for the interaction modes is always visible, in the desktop version a menu screen opens by pressing space as seen in figure 5.1. Alternatively interaction modes can be changed by pressing one of the "1-9" keys, which however requires the user to memorize which number belongs to which mode.

Another difference is the type of user input. The desktop version uses mouse hovering to display the name of a hovered Node or Plane. This is a faster method than touching the object first in the mobile version. In addition to that, in the mobile version the object also has to be deselected otherwise there will be a lot of Node and Plane names displayed, and it will soon get quite messy. Also, the precision of object selection differs because touch input can never be as precise as selecting with a mouse cursor. This could force the mobile user to zoom further in because with a touch input it will not be possible to select small objects like it might be with a cursor. This, of course, would require more time.

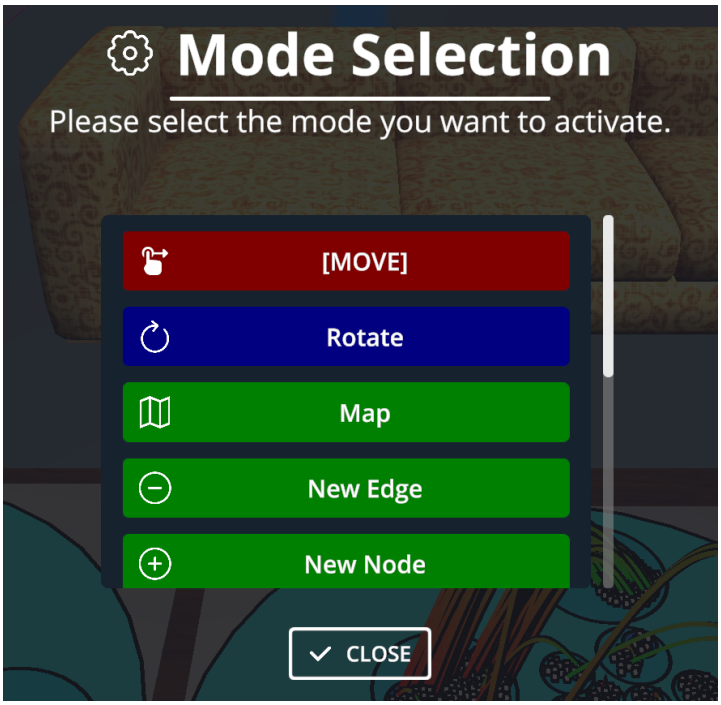

Figure 5.1: The desktop menu for selecting interaction modes.

One more key difference is the available keyboard for desktop users. It allows using Shortcuts, which make some menu items unnecessary but also requires the user to memorize those Shortcuts. The desktop version for example uses the "R" key in the move and rotation mode to recenter or rerotate a Code-City. In the mobile version in opposition, the user will find a button for both actions. With the right amount of training, both actions should probably equal in the amount of time they need but the mobile version sacrifices screen space for those buttons. If however the user has to type more text like in renaming objects, the common desktop keyboard should come in handy, as a study from Kim et al. (2014) shows that even at a same keyboard size, a virtual one will lack in productivity.

5.2 AIM AND HYPOTHESIS

The aim of this user study is to answer the research question discussed in section 1.1. In order of answering the research question, the finished prototype of the mobile extension shall be evaluated. Therefore, the system shall be compared on Android smartphones as well as desktop computers. Comparing these two use cases shall give insight on how much impact the constraints of mobile devices have on the *Usability* and overall user experience. To measure the difference between the desktop and the mobile version, the following hypotheses will be used. The two aspects *Performance* and *Usability* will be measured in the following

Usability: *A term that describes how well a (software) system can be used.*

study and each aspect will have a null hypothesis and an alternative hypothesis.

a) **Performance:** The time required for a task in SEE desktop will be called t_D and for mobile t_M.
   - *Null Hypothesis* H_{a0}: The time required in SEE desktop is higher or the same as the time required in SEE mobile: $t_M \geq t_D$
   - *Alternative Hypothesis* H_{a1}: The time required in SEE desktop is lower than the time required in SEE mobile: $t_D < t_M$

b) **Usability:** Two aspects are measured for Usability. First the *After-Scenario Questionnaire* (ASQ)-Score as a *Post-Task* result and second the *System Usability Scale* (SUS)-Score as a *Post-Study* result.

   i) **ASQ:** Once again the aspect has to be split into three sub-aspects, because the three questions of the ASQ are independent:

      1) The ASQ-Score for *complexity* for SEE desktop is called A_{cD} and for SEE mobile is called A_{cM}
         - *Null Hypothesis* H_{b0}: The ASQ-Score for *complexity* is higher or even for SEE mobile than on SEE desktop: $A_{cM} \geq A_{cD}$
         - *Alternative Hypothesis* H_{b1}: The ASQ-Score for *complexity* is higher for SEE desktop than on SEE mobile: $A_{cD} > A_{cM}$
      2) The ASQ-Score for *effort* for SEE desktop is called A_{eD} and for SEE mobile is called A_{eM}
         - *Null Hypothesis* H_{b0}: The ASQ-Score for *effort* is higher or even for SEE mobile than on SEE desktop: $A_{eM} \geq A_{eD}$
         - *Alternative Hypothesis* H_{b1}: The ASQ-Score for *effort* is higher for SEE desktop than on SEE mobile: $A_{eD} > A_{eM}$
      3) The ASQ-Score for *information* for SEE desktop is called A_{iD} and for SEE mobile is called A_{iM}
         - *Null Hypothesis* H_{b0}: The ASQ-Score for *information* is higher or even for SEE mobile than on SEE desktop: $A_{iM} \geq A_{iD}$
         - *Alternative Hypothesis* H_{b1}: The ASQ-Score for *information* is higher for SEE desktop than on SEE mobile: $A_{iD} > A_{iM}$

   ii) **SUS:** The SUS-Score is called S_D for SEE desktop and S_M for SEE mobile.

ASQ: *A post-task questionnaire consisting of three questions that is used to access how difficult a user perceived a task (See Post-Task).*

Post-Task: *A questionnaire that is taken after every task of an experiment.*

SUS: *The Sytem Usability Scale consists of ten questions that measure Usability.*

Post-Study: *A questionnaire that is taken after every block of an experiment.*

- *Null Hypothesis* H_{e0}: The SUS-Score is higher or even for SEE mobile as for SEE desktop: $S_M \geq S_D$
- *Alternative Hypothesis* H_{e1}: The SUS-Score is lower for SEE mobile than for SEE desktop: $S_M > S_D$

The experiment will be participated by different groups:

1. **SEE-developer:** They are already experienced with SEE-desktop. They are also experienced with software development and with first person games because they tried at least SEE itself, which counts as first person game experience.
2. **Non-SEE-developer:** This group has to be divided into four subgroups as follows:
   - **Software development and third person game experience:** They are more likely to understand the Code-City metaphor and are also more likely to be comfortable with the controls in SEE.
   - **Software development experience:** They are more likely to understand the Code-City metaphor and therefore might be able to find Nodes faster.
   - **First person game experience:** They are also more likely to be comfortable with the controls in SEE.
   - **No experience:** They do not benefit from experience and therefore have to learn the most to interact in SEE

After the experiment the two versions of SEE can be compared as well as all above listed groups. This should give a detailed answer to the research question of this thesis.

5.3 EXPERIMENT SET UP

The system shall be tested in two groups, each starting with a different device. Each group does the test on both devices, but one group will start with the mobile application and the other one with the desktop application. The participants will be randomly assigned to the groups. The subjects will get various tasks to test the Usability of the two applications. Afterward, the users will get a survey in English to document their impressions.

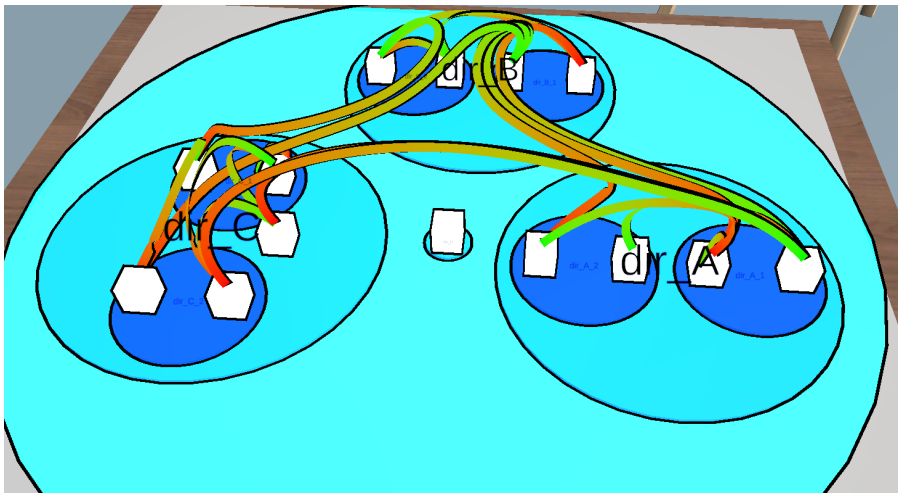

Figure 5.2: The first Code-City for the user study

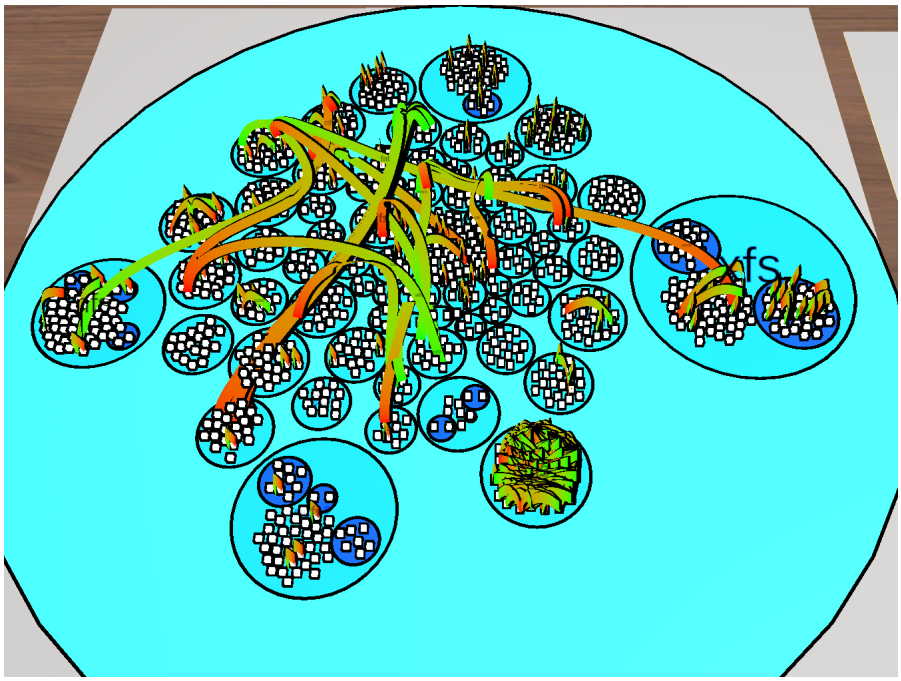

Figure 5.3: The second Code-City for the user study

In this survey the subjects will be asked various demographic questions as well as what Android device and version they will be using. In addition to that, the subjects will be asked if they are experienced with SEE and if they are experienced with software development. Before the subjects will be asked to solve various tasks, they will be asked to watch a short tutorial video on each application. After the video, they will get a training task where every subject can get used to the system and ask questions if they have trouble solving the training task or using SEE in general. The overseer will also make sure that every essential action will be practiced such as zooming and moving the Code-City.

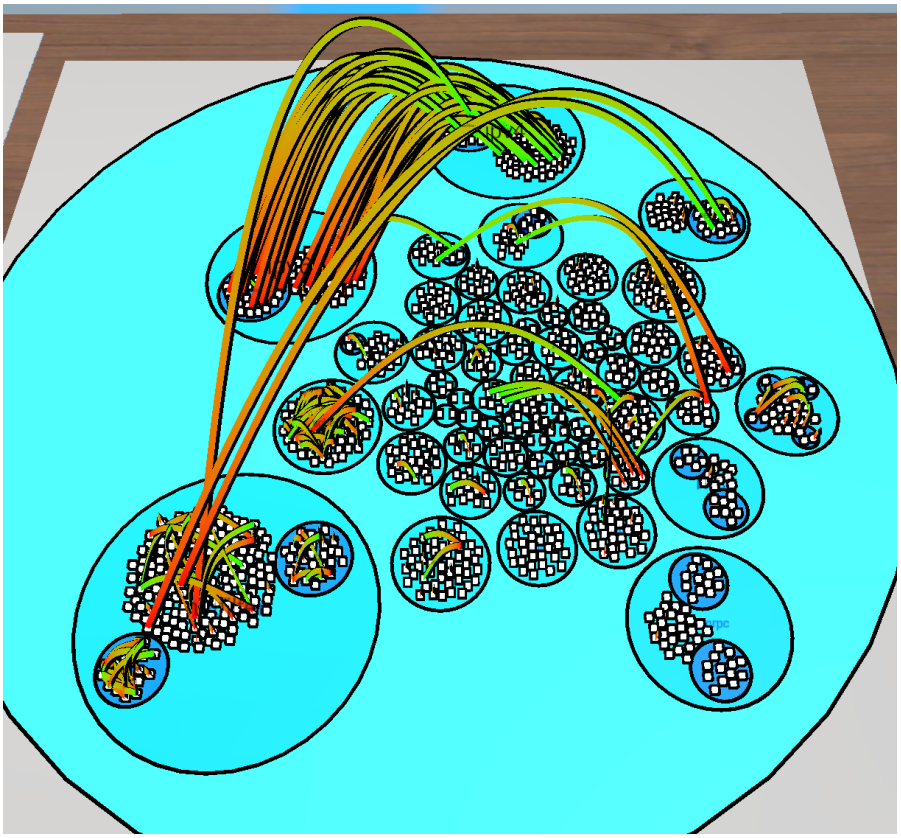

Figure 5.4: The third Code-City for the user study

Figure 5.2 shows a small arranged Code-City that shall be used for the training tasks. The structure of the training Code-City is generic and follows a simple pattern. This shall ensure that the user can focus on the training and that the user does not get overwhelmed.

Following the first questions and the training, the subjects can start with the main tasks. For each application there will be two tasks and after each task the subjects will be handed a Post-Task questionnaire. Last but not least, there will be another questionnaire that aims to scale the Usability of the two applications. For the Post-Task questions the ASQ will be used and for the Usability questions SUS will be used. Both questionnaires will be discussed later on in section 5.4.2. For each main task the overseer will also take the completion time of every main task. The first and second task on the first device will be performed on the Code-City that can be seen in figure 5.3 and the third and forth task on device two will be performed on the Code-City that can be seen on figure 5.4. These examples are much larger than the training Code-City and represent real life code. The second Code-City shows the file system of Linux and the third one shows the network component of Linux. That way the tasks might reflect better on real world uses for SEE.

To not exhaust the testers too much the experiment shall not take longer than one hour. This also ensures that there is no to little variance

due to exhaustion. Each participant might have a different concentration span, but this shall not be the focus of this experiment.

5.4 REALIZATION

The following sections will cover the realization of the previously planned study. The choice of the used survey tool and questionnaires will be explained in section 5.4.1 and section 5.4.2. Afterward, a pilot study will be executed to test the study and possibly find missing aspects in section 5.4.3. Finally, the final experiment set up will be discussed in section 5.4.4.

5.4.1 *Survey Tool*

As a survey tool Google Forms[1] will be used. The survey tool has to fulfill the following requirements:

- The study will be online because an overseer has to attend every experiment, and therefore it comes in handy to be flexible in terms of location. For this reason the survey shall be fully in a browser.
- The survey tool should be free to use.
- Subjects shall be anonymous.
- The results shall be exportable in a data format like *Comma-Separated Values* (CSV).
- Subjects should have the option to presave their answers. In addition to that, answers should not be lost on reload.
- There should be an option to embed the introduction videos in the survey.

CSV: *A file format used for example to store table data. Each line represents a data record and each value is seperated with a comma.*

Google Forms fulfills all these requirements and will therefore be used. The final form can be seen in figure 5.5, which shows the intro of the survey and in figure 5.6, which shows the embedded intro video for SEE mobile.

1 https://www.google.com/forms/about/ (last visit: 05.06.2022)

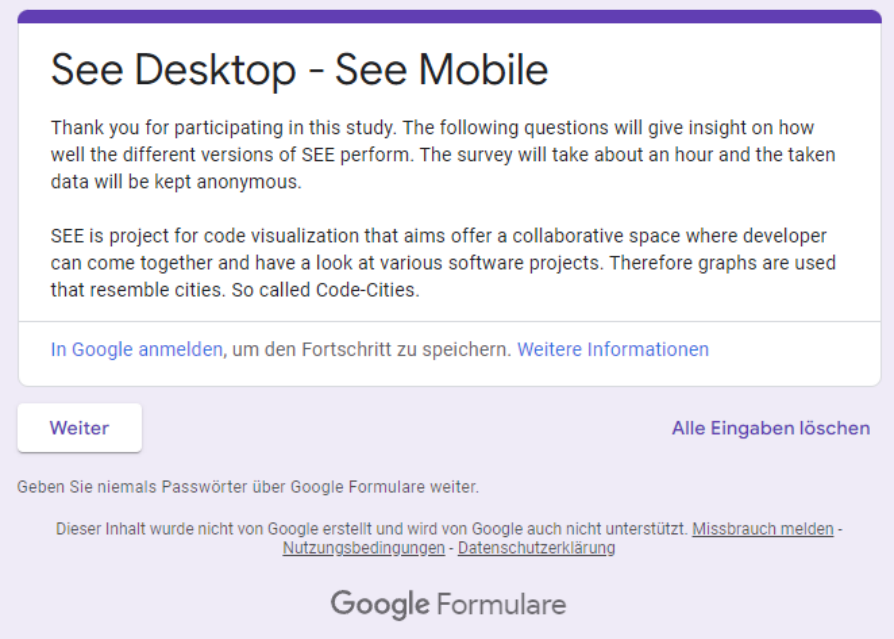

Figure 5.5: The intro of the survey

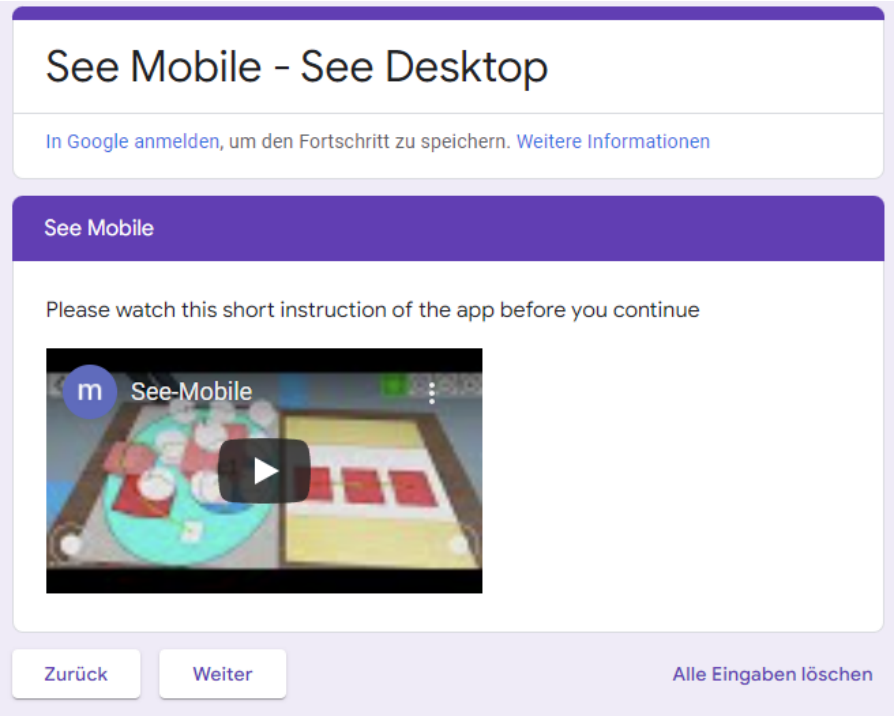

Figure 5.6: The introduction video of the survey

5.4.2 *Questionnaires*

There will be three questionnaires used for the study that will be discussed in detail in the following. The study will start with a demographic questionnaire that covers general information about

the subject. After every task there will be an ASQ and after every block of tasks for each of the two covered devices there will be a SUS questionnaire.

DEMOGRAPHIC QUESTIONNAIRE

The subjects shall start the survey with a demographic questionnaire. In that section they will be asked for their age, gender, highest degree, experience with SEE, experience with first person video games, their Android device name, their Android version and their experience with software development. These specifications will be used to form different groups to see if there is any impact on the result of the following measurements. Mclellan et al. (2011) has shown that the user experience can have a significant impact on the SUS-Score. It is therefore important to view the measured results in context of the paired demographic data.

The mobile version was only tested on a single Android device. It is likely that the performance of the application varies on different Android versions or devices.

POST-TASK QUESTIONNAIRE

The Post-Task questionnaire will supplement the Post-Study questionnaire on a micro level. The main focus here will be on single tasks, which allows to have a look at different aspects like effort, complexity and information provided by the system.

As a Post-Task questionnaire the ASQ will be used. The ASQ was first introduced in 1991 by Lewis (1991). It is designed for task based surveys and contains three questions. The ASQ will be used because it brings the following advantages:

- The questionnaire has been used many times over the years and has proven its validity (Hajesmaeel-Gohari et al. (2022); Lewis (1991); Lewis (1995)).
- With its three questions it is short and does not exhaust the subjects. This is especially important because it will be required to finish this questionnaire a total of four times.
- It fits well for this study because it is a questionnaire designed for task based evaluations.

The ASQ consists of three questions that scale from one to seven, where one means “strongly disagree” and seven means “strongly agree”.

POST-STUDY QUESTIONNAIRE

The Post-Study questionnaire is mainly to obtain as much information about the Usability of the two systems as possible. The questionnaire can be longer than the Post-Task but still should not be too long to keep the processing time of the survey at around an hour.

The SUS questionnaire was first published in 1986 by John Brooke (Brooke (1996)) and is therefore widely used and proven as citations in more than 1200 publications up until 2013 show (Brooke (2013)). The SUS is used in this study for the following advantages:

- It consists of ten questions and has to be done twice. Twenty questions in total fit well into the planned one hour total time span of this experiment.
- It has been made publicly available and is free to use (Brooke (1996)).
- It is widely used and therefore already proven to give useful results (Brooke (2013); Lewis (2018); Grier et al. (2013)).
- According to Peres et al. (2013) and Bangor et al. (2008) it is suited to compare two systems.

The SUS questionnaire will contain, as already mentioned, ten questions. Each question will give a five-level rating from "strongly disagree" to "strongly agree". The results will then be combined into a Usability-Score from zero to 100 where a high value represents a good result and a low value a bad one.

5.4.3 *Pilot Study*

In a first test, the pilot study was executed with one subject. Afterward, the study was discussed and checked for errors. It stood out that the example Code-City of task one was too different to the one in the second task. Therefore, the Code-City of the first task was exchanged with a larger and better comparable one. Further on, a Code-City with 1288 nodes (see figure 5.3) as well as one with 1464 nodes (see figure 5.4) will be used.

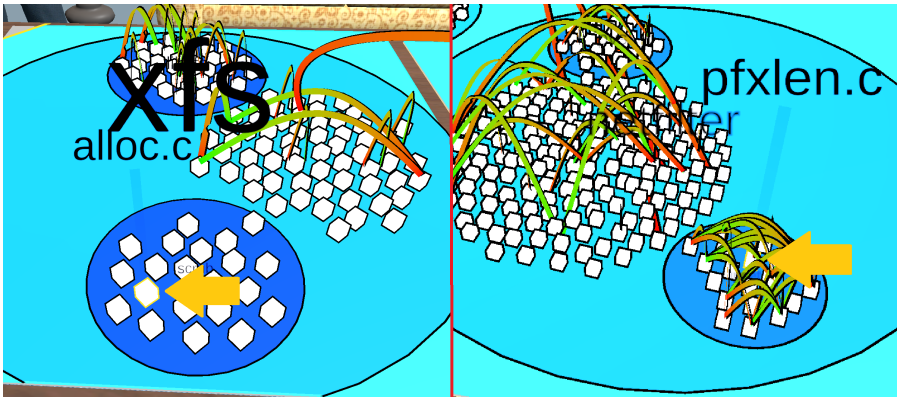

Figure 5.7: The two key nodes are marked with a yellow arrow

Also, the tasks were not comparable because they differed in the types of interactions they used. In one task the user was asked to

rename a node and in the other one the user should have added four nodes. For renaming a node, the user had to use a keyboard, which does not make it comparable to just click and add nodes in the other task.

To also ensure that task one and task three (first task of each version) are comparable, both tasks are similar in their set-up as seen in figure 5.7. As one part of the task is to find the right node, both targeted Nodes are on a Plane with a similar amount of nodes.

The same goes for task three and four (second task of each device) as the target Planes that have to be found have a similar size. Both tasks also include a hint where to find the desired Planes.

5.4.4 *Final Experiment Set Up*

After the pilot study the final set-up can be discussed. The demographic questionnaires will be designed as described in section 5.3. The final questions will be as listed below.

DEMOGRAPHIC QUESTIONS:

- Age
  - 0-15 years old
  - 16-30 years old
  - 31-45 years old
  - 46+ years old
- What gender do you identify as?
  - Male
  - Female
  - Other ...
  - Prefer not to say
- What is the highest degree or level of education you have completed?
  - Some High School (Hauptschule/Realschule...)
  - High School (Abitur)
  - Bachelor's Degree
  - Master's Degree
  - Ph.D. or higher
  - Prefer not to say
  - Other ...
- Questions regarding used hardware and experience

- Are you already experienced with See?
- Do or did you play first person video games?
- Do or did you develop software?
- On which Android device will you attend?
- Which Android version are you using?*

After filling out the demographic questionnaires the subjects will be asked to watch a short tutorial video. Which video they will watch first will depend on with group they are in. If they start with the mobile version of SEE, they will be asked to watch SeeMobile.mp4, otherwise they will be asked to watch SeeDesktop.mp4. After completing their first training and their first two tasks, as described in table 5.1, they will be asked to watch the video they have not started with.

Nr.	Task	Expected time
Training	Navigate through the Planes “dir_root” >“dir_B” >“dir_B_2”. On that Plane select “b2_b.cpp” and rename it “b42”.	1 - 5 mins
1	Detect the largest Plane “xfs”. On that Plane find plane “scrub”. Then find and delete node “alloc.c”.	0.5 - 5 mins
2	Find the Plane with one blue child Plane (“btrfs”). On the blue child Plane “tests” add four new nodes.	1 - 5 mins
Training	Navigate through the Planes “dir_root” >“dir_C” >“dir_C_2”. On that Plane select “c2_b.cpp” and rename it “c42”.	1 - 5 mins
3	Detect the largest Plane “netfilter”. On that Plane find Plane “ipset”. Then find and delete node “pfxlen.c”.	0.5 - 5 mins
4	On the Plane with the most edges (“ipv6”) find the smallest Plane “ila” and connect all four nodes on it.	1 - 5 mins

Table 5.1: The tasks used for the experiment. The device will be switched after task 2.

After each task, the subjects will be asked to fill out an ASQ and after each block of task, as described in table 5.2, they will be asked to fill out a SUS questionnaire. Therefore, each participant has to fill out four ASQs and two SUS questionnaires. The full study should not take longer than an hour.

Phase		Description	
Pre-Experiment		Demographic questionnaire	
	City	Group 1	Group 2
Training	Figure 5.2	Desktop	Mobile
Task 1	Figure 5.3		
ASQ			
Task 2	Figure 5.3		
ASQ			
SUS			
Training	Figure 5.2	Mobile	Desktop
Task 3	Figure 5.4		
ASQ			
Task 4	Figure 5.4		
ASQ			
SUS			

Table 5.2: Experimental procedure per subject. The procedure is swapped per group.

5.4.5 *Execution*

Before the study, the subjects got an installation instruction as well as the two applications. For the execution of the user study, every subject got one of two links to a Google form. Each form starts with a different device. The forms were handed out in alternating order and the subjects should therefore be randomly assigned to the two groups.

The study was executed within a week and a total of 20 subjects participated. From the 20 subjects, two could not finish the tasks on their mobile device, which leaves n = 18 participants that finished the survey.

The instance of SEE was hosted from the home network of the overseer. The overseer also watched the subjects doing their task. The subjects got instructions for the training tasks, and it was ensured that all essential interactions were trained such as zooming and moving the Code-City. Every subject got the same base training, but was allowed to ask further questions or try out other interactions.

5.5 RESULTS

The calc_data.ipynb script can be used to reproduce the result data for the following section. Also, all shown diagrams in this section can be reproduced with the script.

In the following, the group that started the study with the desktop version of SEE will be called *Group 1* and the group that started with the

mobile version will be called *Group 2*. The task order of the two groups remains the same. Both groups start with the same task. That way not only both groups but also the desktop and the mobile SEE version will be compared.

Mann-Whitney-U-Test: *A test that compares two independent samples with ordinal scales to find out whether there is a difference between the two groups. The Mann-Whitney-U-Test does not require a normal distribution.*

In the coming up section the *Mann-Whitney-U-Test* will be used multiple times to see if there are significant differences between two data sets. The Mann-Whitney-U-Test brings the advantage that the tested data sets do not need to fulfill the requirement of a normal distribution (Gibbons and Chakraborti (1991)). This is important because the sample size of n = 18 is too small to assume a normal distribution. The used level of significance will be $\alpha = 0.05$.

5.5.1 *Demographic Data*

In this subsection it will be verified if the two groups differ significantly in their demographic data. Therefore, a two-tailed Mann-Whitney-U-Test that checks whether a set of demographic values from *Group 1* $\neq$ a set of demographic values from *Group 2*, will be used. Both groups have a sample size of $n = m = 9$ with $\alpha = 0.05$ which leads to a critical value of $U = 17$ for a two-tailed Mann-Whitney-U-Test (Zar (2010)).

GENDER

Of 18 participants, 17 reported being male. One participant preferred not to say their gender and can therefore not be grouped. Due to this fact, a grouping does not make sense here in general.

AGE

The subjects were asked to choose their age in grouped options. Most of the participants were aged between 16 and 30 years. Only two participants were aged between 31 and 45 years. Therefore, the age of the participants should not fall into account of the results of the study.

HIGHEST DEGREE

Nine of the participants answered with "Bachelor's Degree" for their highest achieved degree. It is therefore the most common option as illustrated in figure 5.8.

To measure the Mann-Whitney-U-Test, the options were put into an ordinal scale which follows the same order as in the survey. The test shows that there is no significant difference between the two groups ($U = 33.0; p \approx 0.51$).

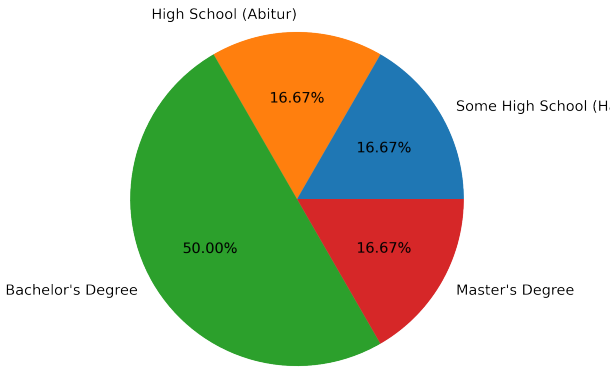

Figure 5.8: The distribution of the highest completed degrees of all 18 subjects

EXPERIENCE

The subjects were asked if they had experience regarding SEE, software development and first person video games. All but one subject had experience with first person video games and the single subject without experience had at least experience with SEE which could arguably count as a first person video game. Therefore, experience with first person video games should not have impact on this study.

Five of 18 subjects had experience with SEE. The Mann-Whitney-U-Test shows there is no significant difference between the two groups ($U = 27.0; p \approx 0.23$). Even though there is no significant difference, four of the subjects with SEE experience are in *Group 2* and only one in *Group 1*.

The majority of ten subjects had experience with software development. The other eight subjects had no experience. The Mann-Whitney-U-Test shows there is no significant difference between the two groups ($U = 49.5; p \approx 0.38$).

ANDROID DEVICE AND VERSION

A large variety of 17 different Android devices were used. Therefore, the distribution of Android devices should not have impact on the study results. Nonetheless, it has to be considered that the devices "Huawei P10 lite", "Huawei P20 Pro" "Samsung A52S", "Samsung S9" and the "Poco X3 Pro" had performance issues. These performance issues showed in bumpy movement and player interactions. Unfortunately, it could not be measured how strong the impact was on the performance on the single phones. The threats of validity this could cause will be further discussed in section 5.6. Three of the devices with bad performance belong to *Group 1* and the other two to *Group 2*.

There was not as much variety in the used Android version as it was with used Android devices. The most commonly used one was Android 12, which is also the most current one to the time of this writing. Android 12 was used by seven subjects. The second most used version was Android 11. One subject used Android 7 on the "Huawei

P10 lite". This is remarkable because Android 7 was released in 2016[2] and the "Huawei P10 lite" was released a little later in March 2017[3] but SEE still managed to work, even if with lower performance. The distribution of the used Android devices can be seen in figure 5.9. The Mann-Whitney-U-Test shows there is no significant difference between the two groups ($U = 39.0; p \approx 0.93$).

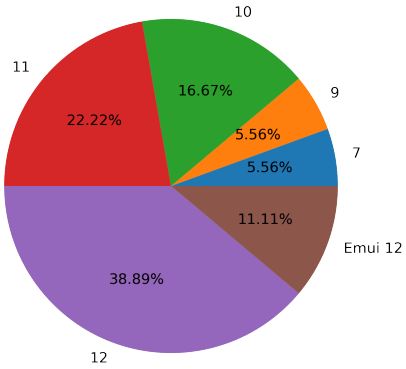

Figure 5.9: The distribution of the Android versions the subjects were using

Surprisingly, there has been no significant difference with the demographic data between the two groups. In the following section the more exciting data will be analyzed.

5.5.2 *Performance*

For the *performance* aspect, the time needed for each task was measured. The time for the trainings task however was not measured because it would not fit the purpose of a training to be measured by time. Therefore, the overseer asked the subjects to tell when they begin the task. The time was taken after the subject has read and understood the task, and it was stopped when the subject successfully finished. This will help to prove or discard hypothesis H_a.

In the following, the two groups in general will be in focus. Is there any significant difference? The total time needed for all subjects in the two groups can be seen in figure 5.10. The average time *Group 1* needed to complete all tasks was $\overline{t_1} = 487s$ and for *Group 2* it was $\overline{t_2} = \approx 388s$. The median time for *Group 1* was $\widetilde{t1} = 315s$ and for *Group 2* it was $\widetilde{t_2} = 384s$. As figure 5.10 shows and the values mentioned above suggest the Mann-Whitney-U-Test shows there is no significant difference between the two groups with a $U = 43.0$ and $p \approx 0.86$.

2 https://android-developers.googleblog.com/2016/08/taking-final-wrapper-off-of-nougat.html (10.06.2022, 12:37)

3 https://www.gsmarena.com/huawei_p10_lite-8598.php (10.06.2022, 12:37)

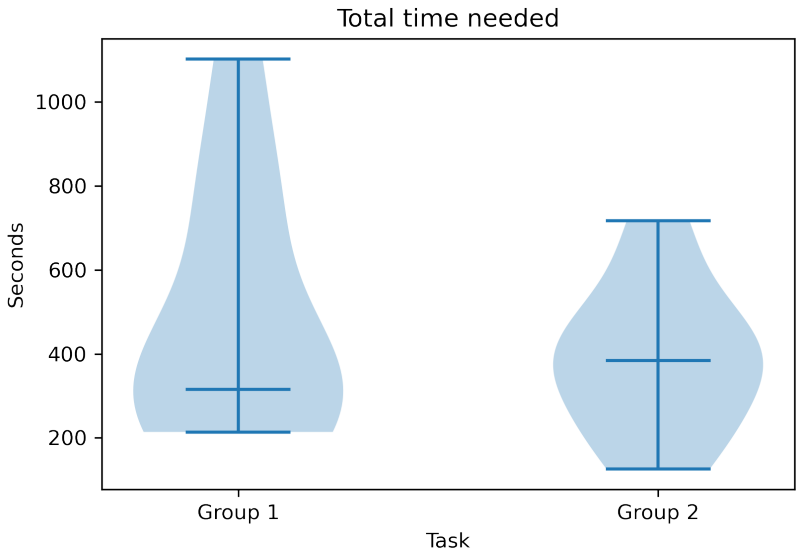

Figure 5.10: Total time each group needed as violin plot

Furthermore, a look at the single tasks will be taken. Has there maybe been a learning effect or difference in general between the groups? To find out task 1 from *Group 1* will be compared with task 3 from *Group 2* and so on. Every pair of tasks is comparable because they have been done on the same device and use the same set of user interactions as described in section 5.4.4. The Mann-Whitney-U-Test shows that there is no significant difference in any combination as listed below:

- *Group 1*, task 1 & *Group 2*, task 3: $U = 26.5; p \approx 0.23$
- *Group 1*, task 2 & *Group 2*, task 4: $U = 35.5; p \approx 0.69$
- *Group 1*, task 3 & *Group 2*, task 1: $U = 49.0; p \approx 0.48$
- *Group 1*, task 4 & *Group 2*, task 2: $U = 57.0; p \approx 0.16$

Finally, the question, if there is a significant difference between the two used versions of SEE in the single tasks, will be looked at. The time needed for every single task on both versions can be seen in figure 5.11. There are some results that stand out and took way longer than the medians would suggest. That could be caused by various reasons as for example:

- The mobile version of SEE did not run smoothly on the subjects device and therefore the tasks became harder.
- The subject did not understand the task correctly and had to reread it.
- The subject overlooked a key object and continued the search at wrong areas.

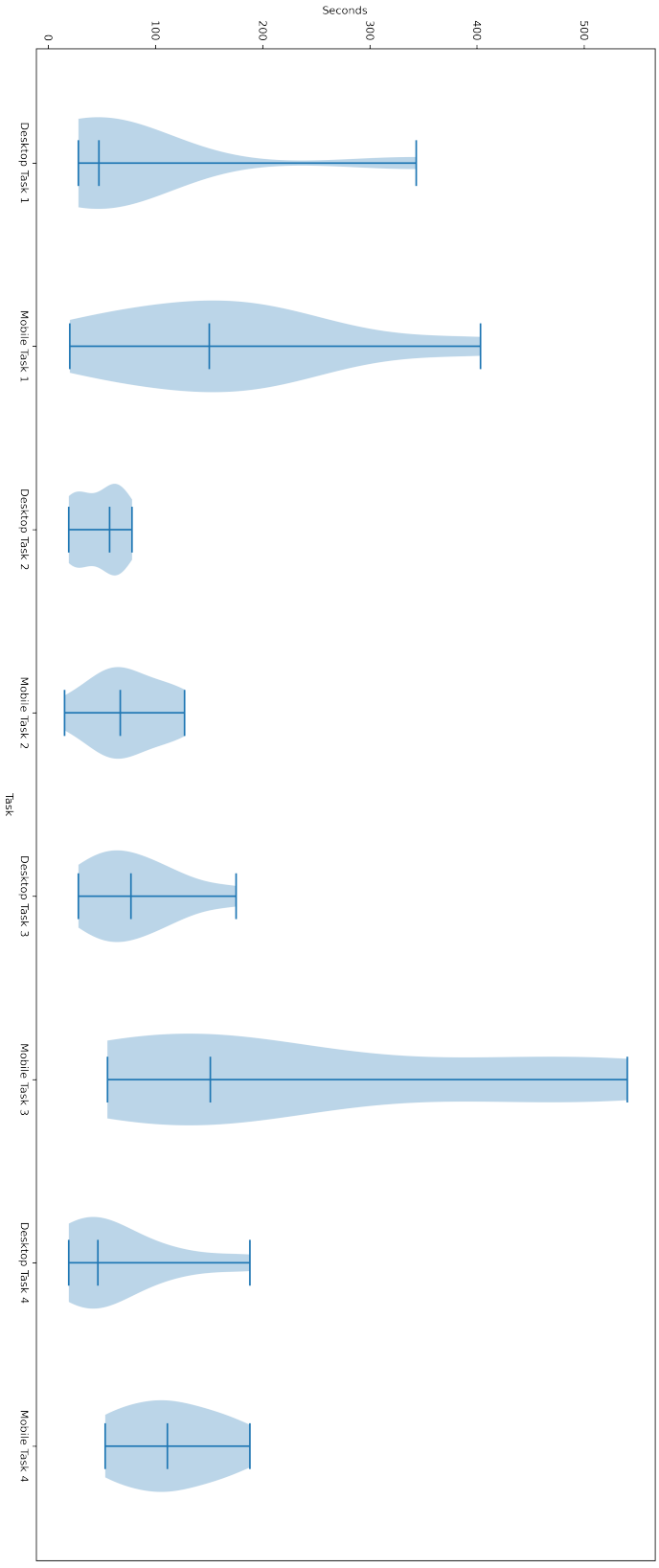

Figure 5.11: Violin plots of all tasks by device

To prove the hypothesis H_a, every task will be checked if the subjects needed significantly less time with the desktop version of SEE than with the mobile version. Therefore, a one-sided Mann-Whitney-U-Test will be used. The critical value here is $U = 21$.

TASK I The subjects needed an average time of $\approx 80s$ with the desktop version and $\approx 166s$ with the mobile version. The median here was 47s with the desktop version and 150s with the mobile version. The medians show that there are some outliers on both sides that took longer than the usual subject needed. The one-sided Mann-Whitney-U-Test shows, with $U = 19; p \approx 0.03$, that the subjects needed significantly less time with the desktop version than with the mobile version.

TASK II This task shows the lowest notable difference in figure 5.11. The Mann-Whitney-U-Test shall confirm this assumption with $U = 24; p \approx 0.08$. This results shows that the subjects did not need significantly less time to solve task 2 with the desktop version than with the mobile version. The average time of the desktop version here is $\approx 47s$ and of the mobile version ≈ 74, while the desktop median lays at $57s$ and the mobile median at $67s$.

TASK III In contrast to task 2, this task shows the biggest difference in figure 5.11 and once again the one-sided Mann-Whitney-U-Test approves this clearly with a result of $U = 12.0; p \approx 0.007$. This shows that the desktop version needed significantly less time than the mobile version once more. Also, the average time needed spreads quite far with an average of $\approx 82s$ for the desktop version and $\approx 247s$ for the mobile version. The median values are 77s for the desktop version and 151s for the mobile version. The desktop median is almost the same as the desktop average while the mobile median is again quite far from the average, which shows that there are again some outliers that took much longer than the usual subject. This could, among other things, be caused by the reported bad performance of the devices "Huawei P20 Pro" and "Huawei P10 Lite", which were used for task 3.

TASK IV The average time needed to complete the task was $\approx 67s$ for the desktop version and $\approx 113s$ for the mobile version. The medians here were 46s for the desktop version and 111s for the mobile version. The values differ quite much and the Mann-Whitney-U-Test validates one last time that the desktop version needed significantly less time than the mobile version to complete the tasks ($U = 17.5; p \approx 0.023$).

To close this section, figure 5.12 shows the total time needed for all tasks by used device. The one-sided Mann-Whitney-U-Test for the alternative hypothesis has a clear result of $U = 11.0$ and $p \approx 0.996$.

Therefore, the alternative hypothesis H_a0 : "The time required in SEE desktop is lower than the time required in SEE mobile" cannot be rejected, and the null hypothesis can be assumed false.

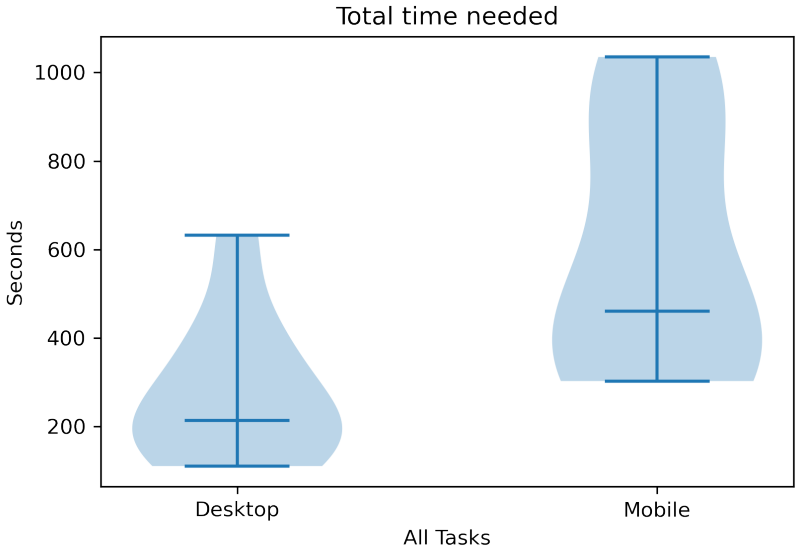

Figure 5.12: Total time the subjects needed on the different devices as violin plot

Summarized it can be said that the subjects were faster on the desktop version of SEE. All data sets regarding the required time to solve the four tasks are comparable, even crossed over like for example task 1 and task 3. Only for one task the subjects did not need significantly less time on the desktop version, but the border value to a significant result was quite close. All in all, the desktop version did perform better than the mobile version regarding the required time.

5.5.3 *Usability*

This section will discuss the results of the Usability questionnaires presented in section 5.4.2. First the ASQ results will be analyzed and second the SUS results to test the hypotheses that were presented in section 5.2.

ASQ

As a start, the Post-Task questionnaire will be looked at. Figure 5.13 shows the ASQ results by group, independent of the used device. Unfortunately, the results differ significantly as the Mann-Whitney-U-Test shows with $U = 392.5$ and $p \approx 0.0037$. Therefore, a deeper look is needed. The results of *Group 1* do not differ significantly from *Group 2*, looking only at the tasks completed with the desktop device, as the Mann-Whitney-U-Test shows with $U = 122.5$ and $p \approx 0.21$. In opposition, the results only caused by completing tasks with a

mobile device differ significantly as the Mann-Whitney-U-Test shows ($U = 73.0; p \approx 0.0048$). Having another look at figure 5.13, the result of *Group 1* with the mobile device stands out. Comparing the two groups without the described results of *Group 1* results in a Mann-Whitney-U-Test of $U = 266.5; p \approx 0.29$, which means without the subset of *Group 1* the two groups show no significant difference.

To ensure a good comparability further on all tasks will be looked at separately. The ASQ consists of three questions, each representing a different aspect. In the following, all tasks will be compared as shown in table 5.3.

ASQ Question	Complexity			
Group 1, Task:	1	2	3	4
Group 2, Task:	3	4	1	2
Mann-Whitney-U	$U = 47; p \approx 0.58$	$U = 37.5; p \approx 0.81$	$U = 7.0; p \approx 0.003$	$U = 24.5; p \approx 0.16$
ASQ Question	Effort			
Group 1, Task:	1	2	3	4
Group 2, Task:	3	4	1	2
Mann-Whitney-U	$U = 44; p \approx 0.78$	$U = 37; p \approx 0.78$	$U = 13.5; p \approx 0.02$	$U = 23; p \approx 0.11$
ASQ Question	Information			
Group 1, Task:	1	2	3	4
Group 2, Task:	3	4	1	2
Mann-Whitney-U	$U = 39.5; p \approx 0.96$	$U = 36; p \approx 0.70$	$U = 20.5; p \approx 0.07$	$U = 20.5; p \approx 0.07$

Table 5.3: Two-sided Mann-Whitney-U-Test for each comparable question on the same device.

The two marked Mann-Whitney-U-Tests in table 5.3 show that the pair *Group 1, Task 3 - Group 2, Task 1 - ASQ Question 1* as well as *Group 1, Task 3 - Group 2, Task 1 - ASQ Question 2* have significant differences. As those two pairs are not comparable, they will not be observed further on. All remaining results can be seen in figure 5.14 for the ASQ aspect *effort*, in figure 5.15 for ASQ aspect *complexity* as well as in figure 5.16, which covers the ASQ aspect *information*.

The Mann-Whitney-U-Test only shows significant findings for two tasks:

TASK III The desktop version of SEE has significant better results than the mobile version in the aspect of information. The subjects rated the desktop version with an average score of 6.22 and a median score of 7, while the mobile version was rated an average of 6.44 and a median of 5.22. The Mann-Whitney-U-Test results here are $U = 20.5$ and $p \approx 0.037$.

TASK IV The desktop version also showed better results for the aspect of information in this task. The subjects rated the desktop version with an average score of 6.44 and a median of 7, while the mobile version was rated with an average of 5.22 and a median of 6. The Mann-Whitney-U-Test resulted in $U = 20.5$ and $p \approx 0.035$ in a one-sided test, which means the desktop version got a significantly higher rating than the mobile version.

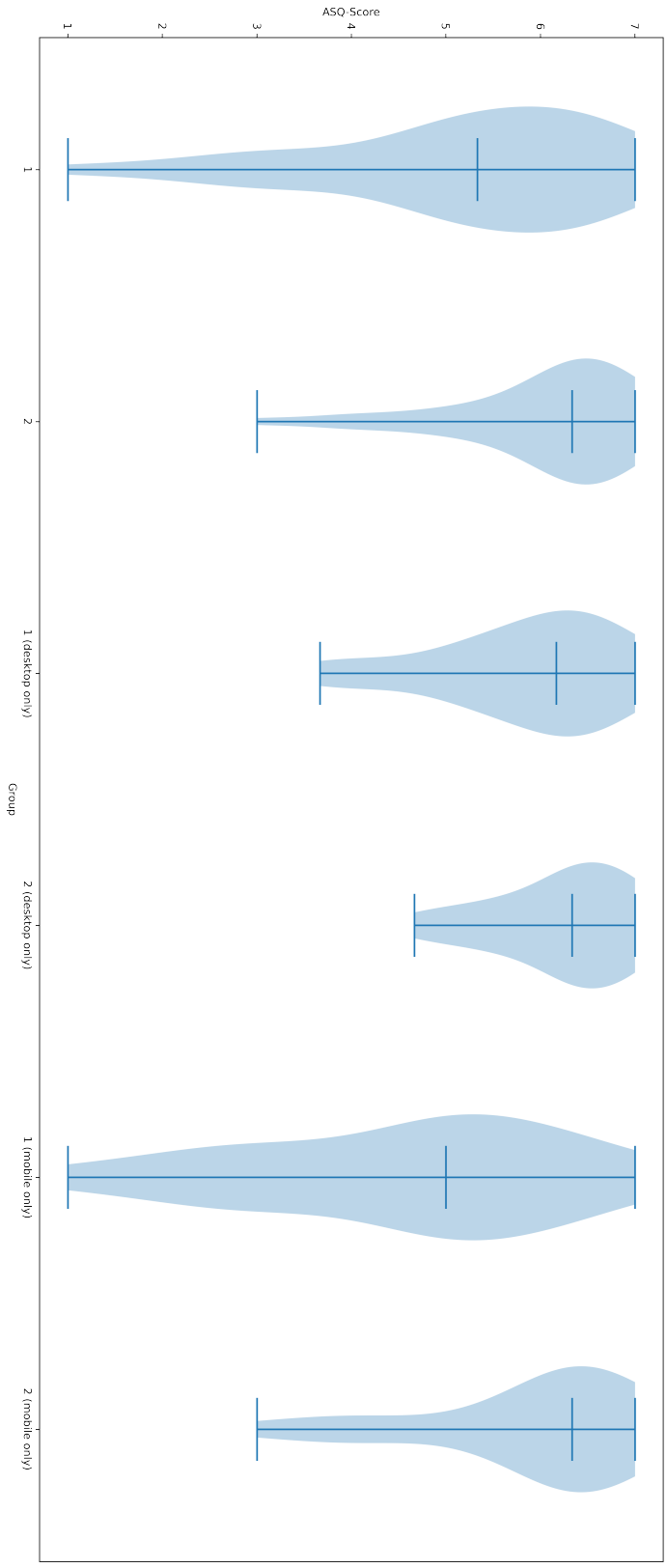

Figure 5.13: Violin plots of all tasks by group

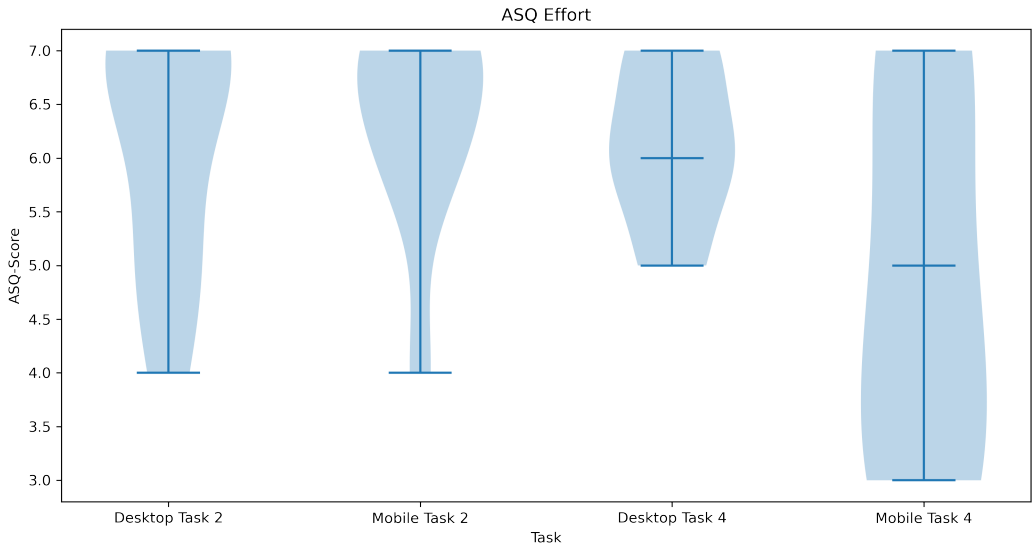

Figure 5.14: ASQ effort results violin plots of all tasks by device

All other tasks show no significant higher or lower results. It is also to mention that the desktop version only scored a significant higher value at the aspect of information. That was to be expected because the mobile versions offer less information due to its constrains regarding the screen size. The mobile version uses for example only icon buttons and not text buttons like the desktop version, which saves screen space but sacrifices information value, since the user has to remember the meaning of each icon.

Finally, the hypotheses H_b, H_c, H_d can be rated:

H_b The null hypothesis H_{b0} can not be rejected, since the Mann-Whitney-U-Test results in $U = 362.0; p \approx 0.23$. This means that the desktop score for the ASQ aspect *complexity* is not significantly higher than the score for the mobile version.

H_c The same goes for null hypothesis H_{c0}. It can not be rejected, since the Mann-Whitney-U-Test results in $U = 348.5; p \approx 0.32$. That means that the desktop ASQ score for the aspect *effort* is also not significantly higher than the mobile ASQ score.

H_d Two of the four tasks showed a significantly higher score for the desktop version in the ASQ aspect *information*, but for all four tasks combined the null hypothesis H_{d0} can not be rejected, since the Mann-Whitney-U-Test resulted $U = 776.5; p \approx 0.06$. The results are notably close to the significance level of $\alpha = 0.05$ in this example.

The final ASQ scores, which are an average of all three questions, can be found in table 5.4. However, the final ASQ-Scores still differ

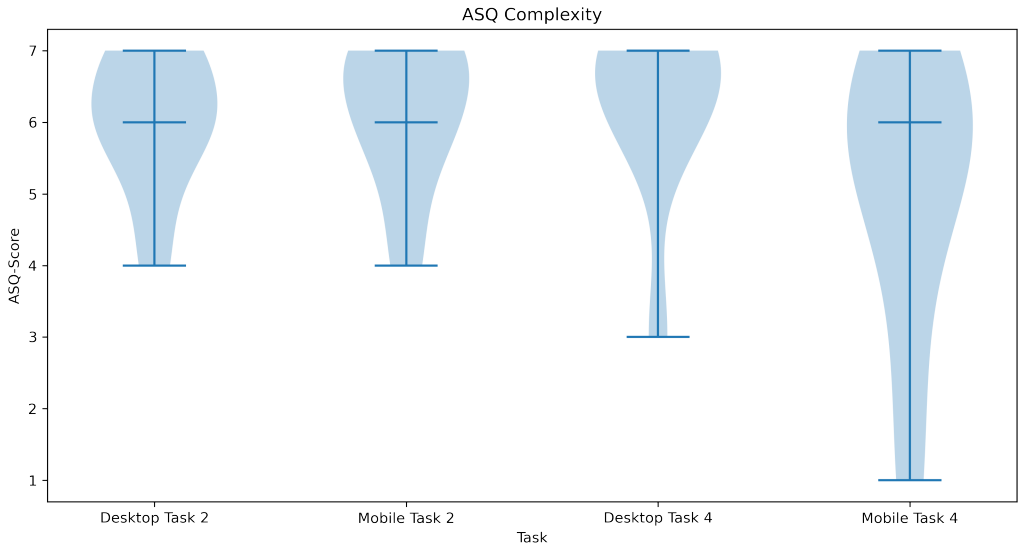

Figure 5.15: ASQ complexity results violin plots of all tasks by device

significantly as the Mann-Whitney-U-Test of $U = 92.0; p \approx 0.03$ shows. Since, even after not taking into account the previously discussed pairs of tasks, the ASQ scores of the two groups differ significantly, which means the final scores cannot be compared.

D	5.8	4.7	6.3	5.2	6.8	5.8	6.7	6.3	4.8	6.7	6.7	5.8	5.3	6.8	5.0	6.5	6.8	6.3
M	6.8	6.8	6.0	6.8	6.3	4.0	6.0	7.0	6.3	6.3	5.0	6.0	5.3	5.3	2.3	7.0	5.3	3.0

Table 5.4: The ASQ-scores from all 18 subjects. The first row contains the ASQ-scores for the desktop application and the second row for the mobile application. The figures have been rounded to whole numbers.

In summary, it can be said that the ASQ results show that the two versions of SEE show little significant difference. In total only task 3 and task 4 showed a significant difference in favor of the desktop version for the ASQ aspect *information*.

SUS

Last but not least, the results of the SUS questionnaire will be discussed. The SUS questionnaire results in a SUS-Score which was calculated by the formula of Lewis (2018):

$$SUS = 2.5 \cdot (20 + \sum_{i=1}^{1} (-1)^{i+1} \cdot S_i) \tag{5.1}$$

In this formula S_i is the score of one of the ten questions in the SUS. The question scores are alternating positive and negative for the SUS score. Therefore, the factor $(-1)^{i+1}$ is needed to either add or subtract from the final SUS score.

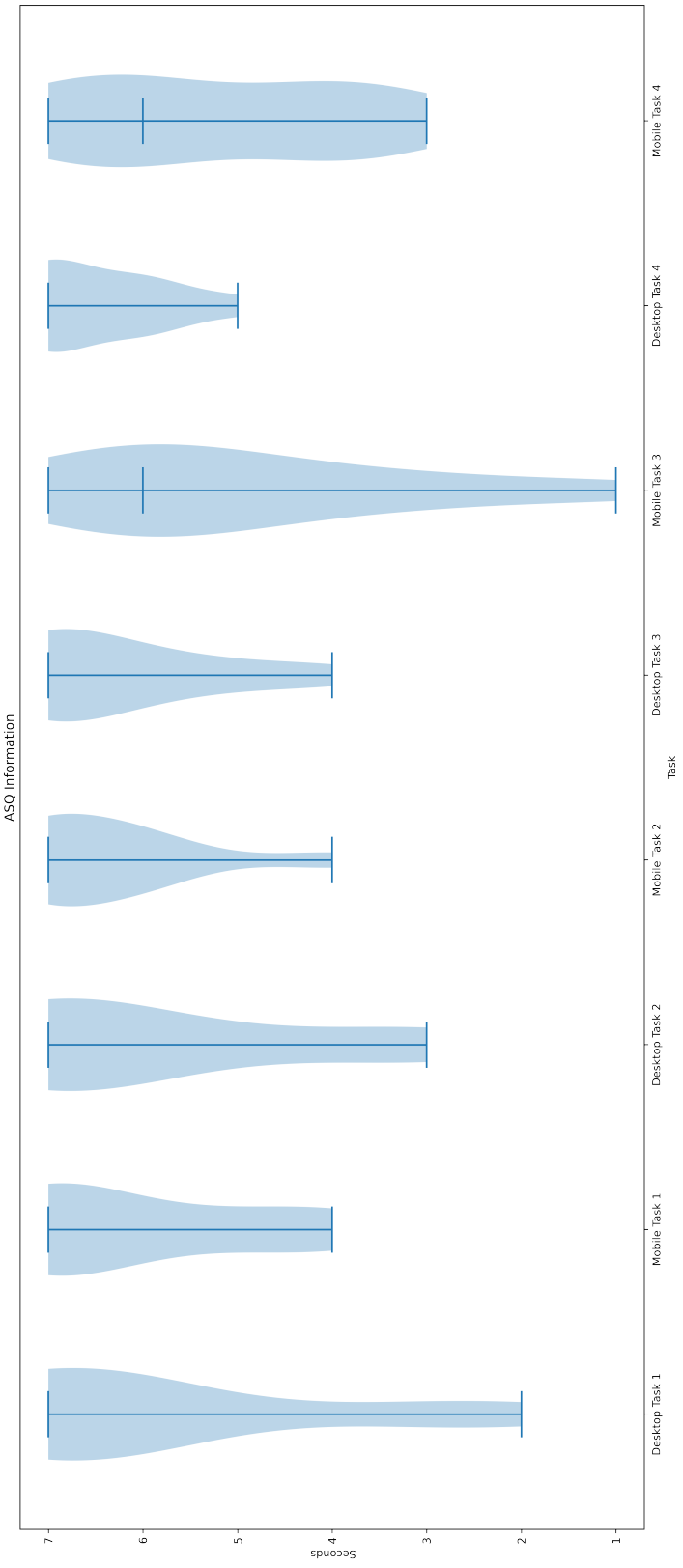

Figure 5.16: ASQ information results violin plots of all tasks by device

The SUS scores from both groups can be seen in table 5.5. Both groups can be compared, as the Mann-Whitney-U-Test shows with a result of $U = 106.0; p \approx 0.08$. The group results can also be seen visualized as violin plots in figure 5.17. The average score of *Group 1* was ≈ 60.83 and the score of *Group 2* was ≈ 72.77. The median of *Group 1* was 58.75 and of *Group 2* it was 82.5. The median of *Group 2* differs quite much from the average which indicates an uneven distribution and outliers that are lower than normal. This can also be seen in figure 5.17.

D	60	58	80	60	78	70	85	53	80	88	80	50	63	85	50	93	100	85
M	38	53	83	53	53	30	88	35	43	86	48	48	55	90	50	88	98	58

Table 5.5: The SUS-scores from all 18 subjects. The first row contains the SUS-scores for the desktop application and the second row for the mobile application. The figures have been rounded to whole numbers.

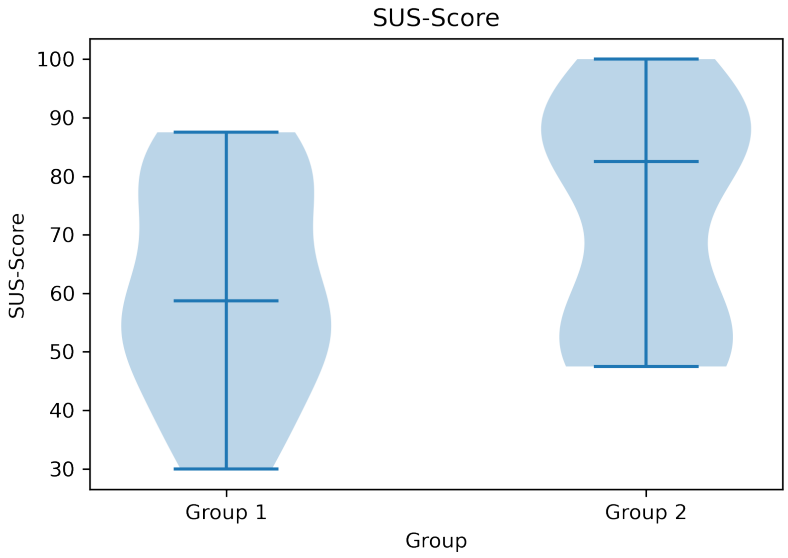

Figure 5.17: The SUS-Scores of the two SEE versions by group

Finally, comparing the SUS-Scores (figure 5.18) of the two different versions shows that the null hypothesis H_{e0} can be rejected, since the Mann-Whitney-U-Test resulted in $U = 219.5; p \approx 0.035$. The desktop version had an average SUS-Score of ≈ 73.06 and a median of 78.75, while the mobile version had an average SUS-Score of ≈ 60.56 and a median of 52.5. After seeing the results, it can be assumed that the Usability of the desktop version is significantly higher than the Usability of the mobile version.

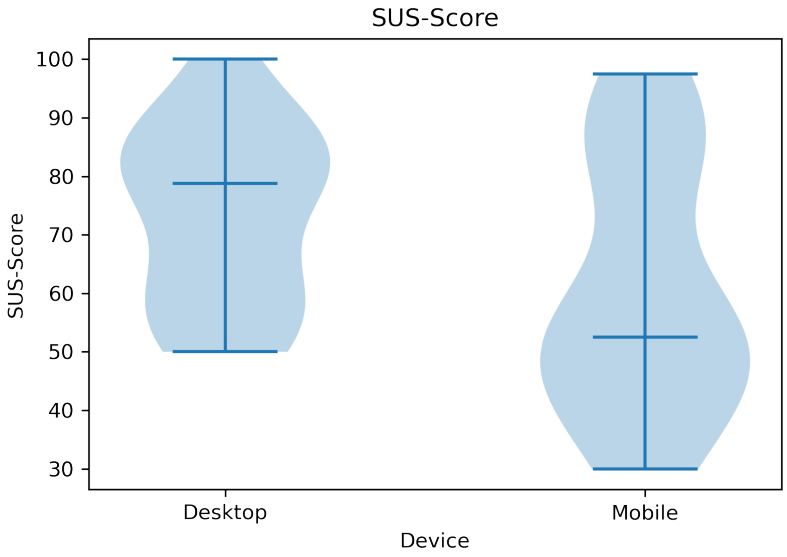

Figure 5.18: The SUS-Scores of the two SEE versions by device

5.5.4 *Impact of Experience*

As already described in section 5.2, the subjects will be grouped by their level of experience and their results will be analyzed. Since only one subject had no experience with first person video games but is still experienced with SEE, the subjects can be divided into three groups:

- SEE-developer
- Non-SEE-developer with software development experience
- Non-SEE-developer without software development experience

In addition to that, there will also be a look at the impact the highest achieved degree of the subjects has. In order to measure the impact the *Kendall rank correlation coefficient* will be used. Another option to calculate the correlation would have been Pearson's correlation coefficient, which has been more popular, but its distribution has slightly worse statistical properties (Hauke and Kossowski (2011)). In addition to that, Khamis (2008) also suggests the use of the Kendall rank correlation coefficient if there is a small ordinal scale of less than five or six levels, which is the case for our results.

Kendall rank correlation coefficient: *The rank based correlation coefficient is a statistic measured with two ordinal or interval-scaled variables. The result is a range between [-1; 1], where a positive value indicate a positive correlation and a negative value a negative correlation. The value 0 indicates no correlation at all.*

EXPERIENCE WITH SEE

Table 5.6 contains the correlation between the experience with SEE of the subjects and the dependent variable of the certain version of SEE. The significance level will be $\alpha = 0.05$ again. The only significant correlation was found for the SUS-Score with $\tau \approx 0.47$ and $p \approx 0.02$. That means that there is a weak positive correlation between the experience with

	SEE-Desktop		SEE-Mobile	
Performance	$\tau \approx -0.32$;	$p \approx 0.12$	$\tau \approx -0.37$;	$p \approx 0.07$
ASQ - Complexity	$\tau \approx 0.29$;	$p \approx 0.18$	$\tau \approx 0.30$;	$p \approx 0.15$
ASQ - Effort	$\tau \approx 0.19$;	$p \approx 0.37$	$\tau \approx 0.16$;	$p \approx 0.46$
ASQ - Information	$\tau \approx 0.04$;	$p \approx 0.87$	$\tau \approx -0.03$;	$p \approx 0.88$
SUS	$\tau \approx 0.29$;	$p \approx 0.17$	$\tau \approx 0.47$;	$p \approx 0.02$

Table 5.6: Correlation between experience with SEE and the dependent variables calculated with the Kendall rank correlation coefficient

SEE and the SUS-Score. This could be because subjects that worked with SEE might be happy to see their work on a different device.

EXPERIENCE WITH SOFTWARE DEVELOPMENT

The correlations between the experience with software development and the corresponding dependent variable from one of the two versions are listed in table 5.7. For this comparison only the subjects without experience with SEE were looked at. Not a single significant correlation has been found in that context, which means that there has been no impact on the dependent variables by having experience with software development.

	SEE-Desktop		SEE-Mobile	
Performance	$\tau \approx -0.14$;	$p \approx 0.56$	$\tau \approx -0.07$;	$p \approx 0.77$
ASQ - Complexity	$\tau \approx -0.08$;	$p \approx 0.76$	$\tau \approx 0.06$;	$p \approx 0.82$
ASQ - Effort	$\tau \approx -0.21$;	$p \approx 0.41$	$\tau \approx -0.05$;	$p \approx 0.83$
ASQ - Information	$\tau \approx -0.39$;	$p \approx 0.15$	$\tau \approx 0.21$;	$p \approx 0.41$
SUS	$\tau \approx -0.07$;	$p \approx 0.77$	$\tau \approx 0.04$;	$p \approx 0.88$

Table 5.7: Correlation between experience with software development and the dependent variables calculated with the Kendall rank correlation coefficient without the subjects that had experience with SEE.

HIGHEST ACHIEVED DEGREE

The correlations between the highest achieved degree and the corresponding dependent variable from one of the two versions can be found in table 5.8. Also in this context there were no correlation even close to significance. This concludes that there is also no correlation between the highest achieved degree and the dependent variable.

ANDROID VERSION AND DEVICE

The correlations between the Android version or device and the corresponding dependent variable from one of the two versions can be found in table 5.9. The devices were divided into the ones reported with a bad performance and the devices with and a mention of a bad performance.

	SEE-Desktop		SEE-Mobile	
Performance	$\tau \approx 0.06$;	$p \approx 0.77$	$\tau \approx -0.36$;	$p \approx 0.06$
ASQ - Complexity	$\tau \approx 0.06$;	$p \approx 0.76$	$\tau \approx 0.10$;	$p \approx 0.62$
ASQ - Effort	$\tau \approx -0.11$;	$p \approx 0.59$	$\tau \approx 0.19$;	$p \approx 0.32$
ASQ - Information	$\tau \approx 0.16$;	$p \approx 0.44$	$\tau \approx 0.28$;	$p \approx 0.17$
SUS	$\tau \approx 0.02$;	$p \approx 0.90$	$\tau \approx 0.14$;	$p \approx 0.46$

Table 5.8: Correlation between highest achieved degree and the dependent variables calculated with the Kendall rank correlation coefficient.

Unfortunately, it is not safe to say if the division is correct because it is possible that subjects did not mention the bad performance on their device. There is still a significant correlation between the ASQ *effort* aspect and the Android device. A higher performing device means a weakly higher ASQ rating in the *effort* aspect, which is a positive effect since a high ASQ score represents a low effort. It is also to mention that the correlation between the Android device and the ASQ *complexity* aspect is right on the edge of the significance level of $\alpha = 0.05$. It is quite possible that on another day it would fall below, which would signify that there is also a positive correlation between a better performing device and a higher ASQ *complexity* score.

	Android Version		Android device	
Performance	$\tau \approx -0.23$;	$p \approx 0.22$	$\tau \approx -0.35$;	$p \approx 0.09$
ASQ - Complexity	$\tau \approx 0.13$;	$p > 0.05$	$\tau \approx 0.40$;	$p > 0.05$
ASQ - Effort	$\tau \approx -0.02$;	$p \approx 0.91$	$\tau \approx 0.44$;	$p \approx 0.04$
ASQ - Information	$\tau \approx 0.13$;	$p \approx 0.50$	$\tau \approx 0.33$;	$p \approx 0.11$
SUS	$\tau \approx 0.07$;	$p \approx 0.72$	$\tau \approx 0.31$;	$p \approx 0.14$

Table 5.9: Correlation between the Android version or device and the dependent variables calculated with the Kendall rank correlation coefficient.

5.5.5 *Comments of the Subjects*

Last but not least, the subjects were asked if they had suggestions for improvements or additional features. The original answers can be found in See Desktop - See Mobile.csv and See Mobile - See Desktop.csv. All the result have been grouped, commented and listed below:

DESKTOP VERSION

- Information on key assignments **x3**
  - Since this was mentioned multiple times, it might be considered to add some kind of information on the Shortcuts. An option would be to implement a tutorial mode, which offers

this kind of information but can later be turned off after the user memorized the Shortcuts.

- Better contrast or font for Nodes and Planes **x5**
  - This was mentioned the most for the desktop version. This problem is already known and solutions are in progress.
- Easier switch between interaction modes
  - This is already implemented with the key 1-9 but was not mentioned in the tutorial video, since it is not beginner-friendly and the subjects should not be confronted with too much information to memorize.
- Hard to use without a mouse
  - This is not optimized yet. Maybe it is worth considering an option to work only with a touchpad or just to clarify that a mouse is needed for an optimal use of SEE desktop.
- Visible UI elements and key combinations are hard to remember
  - Eventually pressing Shortcuts like "alt" in the rotate mode could be indicated as well.

MOBILE VERSION

- Different colors for menu buttons and sub menu buttons
  - The UI design seemed to be confusing for some subjects. Maybe a division of the main button that also reopens the menu again and the additional buttons should be considered. Different colors or different icons for the main buttons, that also indicate that the menu can be opened, could be an option.
- Less battery usage
  - This is probably due to the high need of computing power of SEE. SEE is in ongoing development and on optimizing the efficiency is always a topic.
- Better icons **x2**
  - One subject mentioned that the minus icon for adding an edge is confusing since a minus rather indicates to delete something. Most of the icons are the same in both versions, but the desktop version also contains a button title. Designing a set of homogeneous and innovative icons might be worth considering.
- Deselect button should always be available
  - This can easily be fixed by moving the button into the *quickbar*, which is available from every interaction mode.

- The camera notches of some devices overlap some buttons **x2**
  - A general approach for this would be to not allow screen space with a notch on it. This would sacrifice screen space but provide consistency over different devices. Having a design for every possible device seems to be too much effort.
- Joysticks too far in the corner
  - Since this did not apply for the testing device, there might need to be found a more dynamic approach which considers the actual screen size and adapts the UI size to it.
- Joystick handling too sensitive
  - This seemed to be an issue for more than one subject even if only one mentioned it. The sensitivity should probably be reduced. That way the user can work more precisely.
- Zooming inverted **x9**
  - Since this was explicitly mentioned by half the subjects, this should definitely be changed.
- Better performance on older devices
  - This is probably hard to implement. Maybe optimizing features of SEE that require heavy computing will help in the future.
- App layout should be turned by 180° **x2**
  - This can be adjusted in the Unity editor and is an easy fix.
- Add a search function
  - This could be considered for the future since the desktop version already has a search option.
- Sometimes Planes were deleted after the mode was changed from delete
  - The menu and joysticks need to be safe to touch. Right now objects below buttons or joysticks get selected.

5.6 THREATS TO VALIDITY

This chapter will be closed with a discussion on the threats of validity to the executed study. For this purpose, there will be a distinction into *internal validity* and *external validity* as by Campbell and Stanley (1963).

5.6.1 *Internal Validity*

The internal validity is about the correlation between the independent variable, which is the version of SEE in this case, and the dependent variable, which is *performance*, *ASQ-Score* and *SUS-Score*. Some of Campbell and Stanley (1963) factors will be discussed in the following.

HISTORY
The subjects did the study asynchronously and there is no significant difference between the time it took place for the two groups as seen in figure 5.19. The study took place within five days. It is very unlikely that one of the subjects could have gotten an advantage by for example trying out SEE because every subject got the applications just shortly before their appointment.

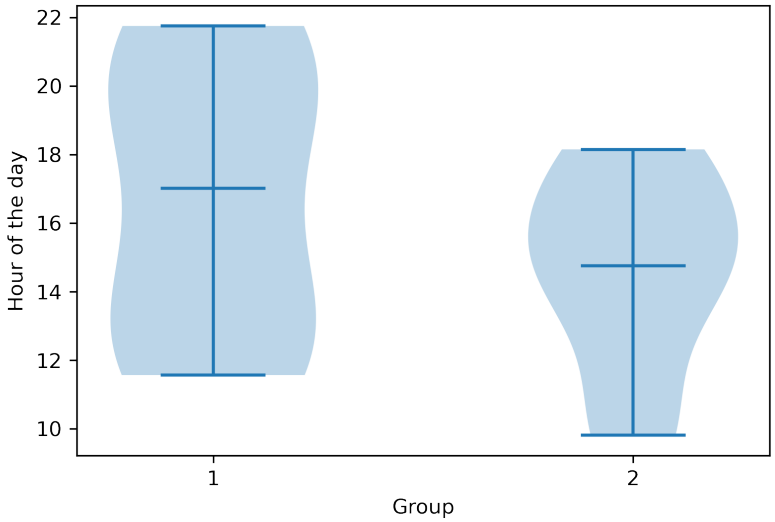

Figure 5.19: The time the subjects started the experiment

TESTING
There could be a learning effect since the tasks were comparable to each other. To prevent a learning effect, the subjects got a training before the time was measured in the actual tasks. The training contained all necessary interactions to solve the tasks and since the interactions are executed in different ways in the two versions, the learning effect should not be too high if even existing.

INSTRUMENTATION
Unfortunately, the performance on the subjects Android devices fluctuated quite much and there was also a significant correlation between the performance of the Android device and the ASQ aspect of *effort* as described in section 5.5.4. Furthermore, it can also not be ruled out that

the performance of the desktop device had an impact on the results even if no subjects mentioned any troubles.

BIASES

Many of the subjects know me personally and most of them also knew that I did the mobile implementation of SEE. It is therefore possible that they gave a higher rating. There was also found a correlation between subjects experienced with SEE and the SUS score in section 5.5.4, which could be caused because SEE developers are most likely emotionally involved in the project. Since there is no significant difference of the amount of SEE experienced subjects this should not have impact on the study results but could still be a threat to validity.

SELECTION

The selection of the subjects was random, however it is still possible that the groups would have been formed uneven. In section 5.5.1, no significant difference between the two groups was found. Still, in section 5.5.3 significant differences between the ASQ ratings the two groups gave were found. Therefore, this threat to validity can not be excluded.

OTHER THREATS

The Code-Cities of the two task blocks were not the same. This is to reduce the learning effect, but it is possible that they differed in complexity and had impact on the difficulty of the tasks. However, the Code-Cities are at least similar as discussed in 5.4.3

5.6.2 *External Validity*

External validity looks at the generalizability of a study. Are the findings appliable in general or just for this specific case? However, the question of generalizability can never be fully answered and there can only be approximations. Threats to the external validity could be:

NONREPRESENTATIVE SAMPLE

Unfortunately, the subjects do not fully represent the population of the software development industry as all the subjects are male and 16 of 18 subjects are in the age of 16-30 years. In addition to that, most of the subjects are not even software developers.

REACTIVE EFFECTS OF EXPERIMENTAL ARRANGEMENT

Since the study was supervised, the subjects might felt observed, which could have an impact on their performance. The environmental conditions of the real use of SEE cannot be reproduced in an experimental arrangement. The external validity is therefore in doubt.

6 CONCLUSION

Further on, this thesis will be closed with a conclusion. Therefore, the research question: *"Are Android smartphones suitable of working with SEE?"* will be in focus one last time. The research question can be interpreted into several sub-questions.

- Is it possible to use SEE on Android devices?
  - Yes, it is possible with the given implementation from chapter 4. The given implementation however is only a first approach and does not cover every functionality from the desktop version. Functions like reviewing source code are not implemented yet and need further solutions.
- Is it possible to run SEE on any given Android device?
  - No it is not, as the results of chapter 5 show. Two of the 20 participants could not attend the user study because the mobile version of SEE crashed on their devices. 18 of 20 participants however were able to use Android. Five of these 18 participants reported performance issues. The bad performance of a device could have caused a worse result as discussed in section 5.5.4. In summary, that means in the current state of SEE, an Android device with solid hardware is needed for a carefree use.
- Does it make sense to use the mobile version or is the Usability too low?
  - The results discussed in section 5.5.3 show that the SUS-score of the mobile version is significantly lower than the one of the desktop version. However, this does not have to mean that the mobile version is not useful. Bangor et al. (2008) provided a scale to rate the value of a SUS-score (see figure 6.1). According to Banger at al., a SUS-Score below 50 means almost certainly that the product will have issues regarding Usability. On contrary, a score between 70 and 90 is promising that the product will have a high acceptability. The SUS-Score of the mobile version is therefore in the acceptable range and yet it is significantly lower than the desktop versions score. It can probably be said that the mobile version is useful to have, even though there is potential for improvement.

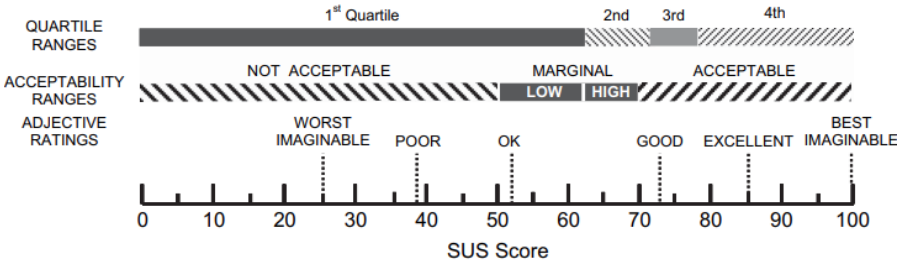

Figure 6.1: A comparison of mean System Usability Scale (SUS) scores by quartile, adjective ratings, and the acceptability of the overall SUS score by Bangor et al. (2008)

All in all, the research question can be answered with "yes". Regardless of the facts that the subjects needed significantly less time to solve the tasks in the user study with the desktop, the mobile version still seems to be acceptable and therefore suitable to be used. This can be additionally substantiated with the results from the ASQ questionnaires, which just slightly differed between the two versions. Also, as mentioned earlier, the SUS-Score, even if significantly lower, seems to be in an acceptable range. It was also to expect that the outcome of a mobile version can hardly be better than the desktop version because of all the constraints a small device brings along. The named answer of the research question is, of course, only a suggestion, since questions like these cannot be answered completely objectively.

6.1 FURTHER STUDY AND IDEAS

The presented version of SEE for Android devices is just a first approach and far from perfect. In the following ideas of improvement or further studies will be discussed.

- The first thing that comes to mind is the performance of the application on certain devices. Working on a small device often requires a zooming interaction. Zooming gets harder if the application does not run smoothly. Future work could focus on improving this performance for a better user experience or at least find out what the minimum requirement to an Android device is.
- Some subjects mentioned concerns regarding the interface. A few devices have camera notches on their display that are in the way of the menu and for other devices the joysticks were too far in the corners, so that they could not be handled properly. It is also thinkable that the menu could be improved in general with better icons and color. Improvements like these would require further user studies.

- As mentioned in section 4.5.3, it was intended to restructure SEE in order to minimize the need of conditional compilation. This process is far from trivial and would require further work.
- The current version has only been tested for Android smartphones. It would be interesting to see how SEE performs on tabled devices and on other operating systems like iOS.
- Another aspect to look at could be *Augmented reality* (AR). Santos et al. (2016) provided guidelines for a mobile AR interface, which could be adapted to SEE. Adding AR to SEE could have great potential and offer new innovative ways of cooperating. It would also be interesting to compare the existing VR version of SEE with an AR approach.

AR: *An interactive experinece in a real world environment where virtual objects are added. In the example of SEE this could be a Code-City projected on a real world table.*

6.2 CLOSING WORDS

This thesis has started with describing the motivation and research question in chapter 1. To answer this question, fundamental information was given in chapter 2 before a concept for the mobile version of SEE was sketched in chapter 3. The concept was then realized almost exactly as planned in chapter 4. It was also discussed were the implementation faced problems and how they were solved. Then, in chapter 5, the evaluation was planned, executed and analyzed. Therefore, a significance level of $\alpha = 0.05$ was used. 20 subjects participated even though only 18 could finish the user study. The user study gave insight on aspects like *performance*, *usability*, *complexity* and many more. Finally, in chapter 6, the research question was answered with "yes", even though the answer to this question can never be completely objective and the desktop version of SEE did perform better in general.

A

GLOSSARY

Android An operating system for portable devices such as smartphones and tablets. 1, 3, 5, 6, 12, 15, 21, 23–25, 28, 31, 35, 38, 41, 54, 58, 61–63, 73

Asset Can be any type of media that can be used in a Unity project. 16, 24

Code-City In the Code-City metaphor, software components are represented by buildings in a city, and the properties of these buildings can express different metrics of the software. For example, the height of a "Node" could represent the lines of code in that class. 2–5, 7–13, 19, 22, 23, 25, 28, 30–32, 36, 39, 59, 63, 67

Edge A line that connects two Nodes. Could for example represent a class call. 9, 13, 21

GameObject A fundamental object in Unity Scenes. 15, 21

Kendall rank correlation coefficient The rank based correlation coefficient is a statistic measured with two ordinal or interval-scaled variables. The result is a range between [-1; 1], where a positive value indicate a positive correlation and a negative value a negative correlation. The value 0 indicates no correlation at all. 53–55

Mann-Whitney-U-Test A test that compares two independent samples with ordinal scales to find out whether there is a difference between the two groups. The Mann-Whitney-U-Test does not require a normal distribution. 39–43, 45, 46, 48, 49, 51, 52

Node A point in a diagram where lines intersect. In SEE it usually displays a software class. 2, 3, 7, 9–13, 19–23, 27, 30, 37, 55

Plane An area that bundles Nodes. It could for example represent a namespace. 3, 9, 20, 21, 23, 27, 37, 55, 57

Post-Study A questionnaire that is taken after every block of an experiment. 29, 35

Post-Task A questionnaire that is taken after every task of an experiment. 29, 32, 35, 46, 67

Prefab Predefined GameObject that can be loaded into a Scene. 15, 16

Scene A collection of components that form an overall picture. A scene could contain for example environmental or UI components. 6, 15, 21

Shortcut A combination of key that will call an action like for example "ctrl" + "c" for copying a text. 17, 28, 55, 56

Unity A cross-platform game engine that forms the foundation of SEE. 2–6, 16, 17, 21, 24, 25

Usability A term that describes how well a (software) system can be used. 28–30, 32, 35, 36, 45, 52, 61, 67

B

ACRONYMS

AR An interactive experinece in a real world environment where virtual objects are added. In the example of SEE this could be a Code-City projected on a real world table. 63

ASQ A post-task questionnaire consisting of three questions that is used to access how difficult a user perceived a task (See Post-Task). 29, 32, 35, 38, 45, 46, 48, 49, 51, 54, 55, 57, 58, 62

CSV A file format used for example to store table data. Each line represents a data record and each value is seperated with a comma. 33

GLX A file format for graphs. Is is used for example in SEE to import and export Code-Cities. 3

IDE An application that provides software developers with tools like a source code editor, build automation and a debugger. 6

JAR A package file format that is typically used to aggregate multiple Java class files and appendix into one file. 25

SEE An interactive software visualization that uses the *code-city* metaphor and enables collaborative multiplayer interactions via multiple platforms like desktop, virtual reality and soon Android devices. 1–4, 7–12, 15, 19, 21–25, 27, 29–33, 35, 38, 39, 41–43, 45, 48, 51–59, 61–63, 67, 73

SUS The Sytem Usability Scale consists of ten questions that measure Usability. 29, 30, 32, 35, 36, 38, 46, 51–53, 57, 58, 61, 62

VR A by a computer generated environment. The user sees through a head mounted display and can interact in the Virtual Reality with some kind of controllers. 1, 24, 63

LIST OF FIGURES

LIST OF TABLES

E

RELATED FILES

The related files to this thesis are attached on a USB stick.

SEE_DESKTOP.ZIP The desktop version of SEE used for the user study. Unzip and start by executing the *See.exe*

SEE_MOBILE.APK The Android version of SEE that is used for the user study. The file can be installed on an Android device.

SEE_FINAL.APK A revised Android version of SEE. Zooming is inverted, and the application is flipped 180° as it was heavily requested in the user study.

CALC_DATA.IPYNB A Python script that calculates all results of the study.

SEE DESKTOP - SEE MOBILE.CSV The survey answers from *Group 1*. The subjects started with the desktop version of SEE.

SEE MOBILE - SEE DESKTOP.CSV The survey answers from *Group 2*. The subjects started with the mobile version of SEE.

SEE_MOBILE.MP4 The instruction video for the mobile version of SEE used in the user study.

SEE_DESKTOP.MP4 The instruction video for the desktop version of SEE used in the user study.

USER_MANUAL_SEE.PDF A short overview for the interactions in the mobile version of SEE.

SEE_MOBILE_REPOSITORY.ZIP The repository for SEE containing all the changes made for the mobile version. Note that SEE is an ongoing project and only the changes discussed in chapter 4 are part of this thesis.

MASTER_THESIS_ROMAN_GRESSLER.PDF A digital version of this thesis.

BIBLIOGRAPHY

Ted Schadler and John C McCarthy. Mobile is the new face of engagement. *Forrester Research*, 13:1–30, 2012.

Luke Wroblewski. *Mobile first*. Number 6. Editions Eyrolles, 2012.

Rainer Koschke. Auge in auge mit ihrer softwarearchitektur. Universität Bremen, 2020.

Rainer Koschke and Marcel Steinbeck. SEE your clones with your teammates. In *15th IEEE International Workshop on Software Clones, IWSC 2021, Luxembourg, October 2, 2021*, pages 15–21. IEEE, 2021. doi: 10.1109/IWSC53727.2021.00009. URL https://doi.org/10.1109/IWSC53727.2021.00009.

Richard Wettel and Michele Lanza. Visualizing software systems as cities. In *2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, pages 92–99. IEEE, 2007.

Richard Wettel and Michele Lanza. Visual exploration of large-scale system evolution. In *2008 15th Working Conference on Reverse Engineering*, pages 219–228. IEEE, 2008a.

Richard Wettel and Michele Lanza. Codecity: 3d visualization of large-scale software. pages 921–922, 01 2008b. doi: 10.1145/1370175.1370188.

Benjamin Nicoll and Brendan Keogh. The unity game engine and the circuits of cultural software. In *The Unity game engine and the circuits of cultural software*, pages 1–21. Springer, 2019.

Scott D Roth. Ray casting for modeling solids. *Computer graphics and image processing*, 18(2):109–144, 1982.

Stephanie Houde and Charles Hill. What do prototypes prototype? In *Handbook of human-computer interaction*, pages 367–381. Elsevier, 1997.

Karen Renaud and Judy Van Biljon. Demarcating mobile phone interface design guidelines to expedite selection. *South African Computer Journal*, 29(3):127–144, 2017.

Lumpapun Punchoojit and Nuttanont Hongwarittorrn. Usability studies on mobile user interface design patterns: a systematic literature review. *Advances in Human-Computer Interaction*, 2017, 2017.

Jessica Conradi, Olivia Busch, and Thomas Alexander. Optimal touch button size for the use of mobile devices while walking. *Procedia Manufacturing*, 3:387–394, 2015.

Pekka Parhi, Amy K Karlson, and Benjamin B Bederson. Target size study for one-handed thumb use on small touchscreen devices. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pages 203–210, 2006.

Suzanne Robertson and James Robertson. *Mastering the Requirements Process: Getting Requirements Righ*. Addison-Wesley Professional, 2012. ISBN 978-0-13-294285-0.

P. Stevens and R. Pooley. *Software Engineering with Objects and Components*. Springer, 2005.

Lifeng Zhang, Qian Fan, Yujie Li, Yousuke Uchimura, and Seiichi Serikawa. An implementation of document image reconstruction system on a smart device using a 1d histogram calibration algorithm. *Mathematical Problems in Engineering*, 2014:1–10, 06 2014. doi: 10.1155/2014/313452.

Jeong Ho Kim, Lovenoor Aulck, Michael C Bartha, Christy A Harper, and Peter W Johnson. Differences in typing forces, muscle activity, comfort, and typing performance among virtual, notebook, and desktop keyboards. *Applied ergonomics*, 45(6):1406–1413, 2014.

Sam Mclellan, Andrew Muddimer, and S. Peres. The effect of experience on system usability scale ratings. *Journal of Usability Studies*, 7, 11 2011.

James R Lewis. Psychometric evaluation of an after-scenario questionnaire for computer usability studies: the asq. *ACM Sigchi Bulletin*, 23 (1):78–81, 1991.

Sadrieh Hajesmaeel-Gohari, Firoozeh Khordastan, Farhad Fatehi, Hamidreza Samzadeh, and Kambiz Bahaadinbeigy. The most used questionnaires for evaluating satisfaction, usability, acceptance, and quality outcomes of mobile health. *BMC Medical Informatics and Decision Making*, 22(1):1–9, 2022.

James R Lewis. Ibm computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction*, 7(1):57–78, 1995.

John Brooke. Sus: a "quick and dirty'usability. *Usability evaluation in industry*, 189(3), 1996.

John Brooke. Sus: a retrospective. *Journal of Usability Studies*, 8:29–40, 01 2013.

James R Lewis. The system usability scale: past, present, and future. *International Journal of Human–Computer Interaction*, 34(7):577–590, 2018.

Rebecca A Grier, Aaron Bangor, Philip Kortum, and S Camille Peres. The system usability scale: Beyond standard usability testing. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 57, pages 187–191. SAGE Publications Sage CA: Los Angeles, CA, 2013.

S. Camille Peres, Tri Pham, and Ronald Phillips. Validation of the system usability scale (sus): Sus in the wild. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 57(1):192–196, 2013. doi: 10.1177/1541931213571043. URL https://doi.org/10.1177/1541931213571043.

Aaron Bangor, Philip T. Kortum, and James T. Miller. An empirical evaluation of the system usability scale. *International Journal of Human–Computer Interaction*, 24(6):574–594, 2008. doi: 10.1080/10447310802205776. URL https://doi.org/10.1080/10447310802205776.

Jean D Gibbons and S Chakraborti. Comparisons of the mann-whitney, student'st, and alternate t tests for means of normal distributions. *The Journal of Experimental Education*, 59(3):258–267, 1991.

J.H. Zar. *Biostatistical Analysis*. Prentice Hall, 2010. ISBN 9780131008465.

Jan Hauke and Tomasz Kossowski. Comparison of values of pearson's and spearman's correlation coefficient on the same sets of data. *QUAESTIONES GEOGRAPHICAE 30(2)*, 2011.

Harry Khamis. Measures of association: how to choose? *Journal of Diagnostic Medical Sonography*, 24(3):155–162, 2008.

Donald T Campbell and Julian C Stanley. *Experimental and quasi-experimental designs for research*. Ravenio books, 1963.

Carlos Santos, Brunelli Miranda, Tiago Araujo, Nikolas Carneiro, Anderson Marques, Marcelle Mota, Jefferson Morais, and Bianchi Meiguins. Guidelines for graphical user interface design in mobile augmented reality applications. In *International Conference on Virtual, Augmented and Mixed Reality*, pages 71–80. Springer, 2016.