GIS and Geoinformatics Lab

UrbanX

Urban planning in mixed reality

Hasret Gümgümcü & Daniel Laumer

Institute of Cartography and Geoinformatics
ETH Zurich

Professorship
Prof. Dr. Martin Raubal

Supervisors
Dominik Bucher
David Rudi
Christian Sailer
Fabian Göbel

Abstract

The goal of this project was to develop a Hololens application to do urban planning in a mixed reality environment and provide a framework which makes the process more efficient and immersive. The Microsoft Hololens is a recently developed pair of headmounted mixed reality smartglasses, which lets the user place virtual holograms in the real world and interact with them using specific gestures. The application should display the buildings in the region of interest and let the user access the attributes of each building in an easy and intuitive way. Also there should be editing functionalities so that the model can be changed to the users content.

So we introduce *UrbanX*, our hololens application for efficient and interactive urban planning. We used the open data from Stadt Zürich to get 3D models of the buildings. To remove buildings we developed a gesture called *TapAndHold* which removes the building and only leaves a transparent ground plane indicating where the building was before. This process can easily be reversed with the same gesture. For adding new building we used markers printed on cardboard which the hololens can recognize and display virtual new buildings on top of it. This way, the new buildings can be positioned and oriented manually by just moving the cardboard markers, which make the whole process more intuitive and easier to handle since drag and drop on virtual holograms is not that easy. Also people like to still have things they can touch and move. With this solution, there we created an interesting mix between the virtual and real world and are thus blurring the border between the two realities.
Information about each building like for example size, year of construction or number of people is stored and can be displayed either detailed for a single building or for the whole model with the help of rule based coloring. The overall state of attributes is shown in absolute and relative bar charts to keep track of the change. Since planning projects often have to achieve a certain goal of number of new residents or similar, we added this element integrating it into the statistical bar charts and telling the user, how far to the goal he progressed. This adds a game-like component and allows a more playful and incentive experience.

Contents

1 Introduction

In the recent years mixed-reality gained importance in various fields like for example the entertainment sector, construction business or for designing tasks. Another possible use for AR in general is urban planning. When developing strategies for a new region, usually 3D models are built. Those models are used to collaboratively discuss about pros and cons to reach the final result after many iterative steps. This is a time consuming process.

We use the Microsoft Hololens for a mixed-reality urban planning experience. The goal is to design the idea generation in the very beginning of a new urban planning process in an efficient way. A virtual 3D city model of the surrounding target area can be placed on a flat surface in the real world. A toolbox of newly designed virtual 3D buildings for an urban planning project allows to develop several strategies for the target region. Since this process is highly collaborative, each building is attached to a physical target which can be moved around the real world to insert the new 3D building into the city model. On top of each buildings are some attributes which help for the planning process. Additionally, an information panel helps to have an overview all the time when trying out different strategies how it affects the current state. Those can be values like how many new residents or work spaces can be provided with the current strategy or the change of other attributes. The city model can also be colored depending on those attributes which is helpful to have a quick impression how well the new designed region fits to its surroundings.
Generally speaking, this application makes urban planning into a more playful and intuitive process.

2 Functionality of the App

This section is a short instruction manual which introduces the app and helps a user to quickly start working with our application.

2.1 Objective

Our app provides a tool for efficient urban planning. To get it working you have to select the area you want to work on and gather the necessary data. The data acquisition is a very important step since our app heavily relies on a correct and sufficient display of attributes. The data you need includes:

- Digital 3D model of the already existing buildings and the coordinates of those buildings
- Digital 3D models of possible new buildings which may be built there. It can also be a collection of any kind of building to have a wide range of possibilities.
- Attributes that you find important for each building. This can include information like number of residents, workplaces, height, number of floors, gross floor area, etc.
- Printed out screen shot of each new building, which will act as physical markers to place new buildings.

When all this data is added to the application, you are ready to start working and planning your chosen area. As an example we chose a real project currently going on, which will be explained in section 3.2 and 4.3.

2.2 Interface

The interface is simple. The buildings are in plain white and there is an info panel in the back of the model where all the buttons are and where we can see the changes we applied to the model in a graphical way. The info panel is explained in more detail in section 4.4.

Figure 1: Interface of the application

When you start up the app the first thing to do is to place the model somewhere in the room. We used the spatial mapping from the hololens to scan the room and automatically fit the model to the surface you are looking at. Confirm the location by clicking on the model and anchoring it in the real world.

There are four main functionalities we implemented:

- Getting information
- Visualize
- Removing old buildings
- Adding new buildings

The first two are about displaying the attributes that we added to each building and the other two are about making actual changes to the model.

2.3 Getting Information

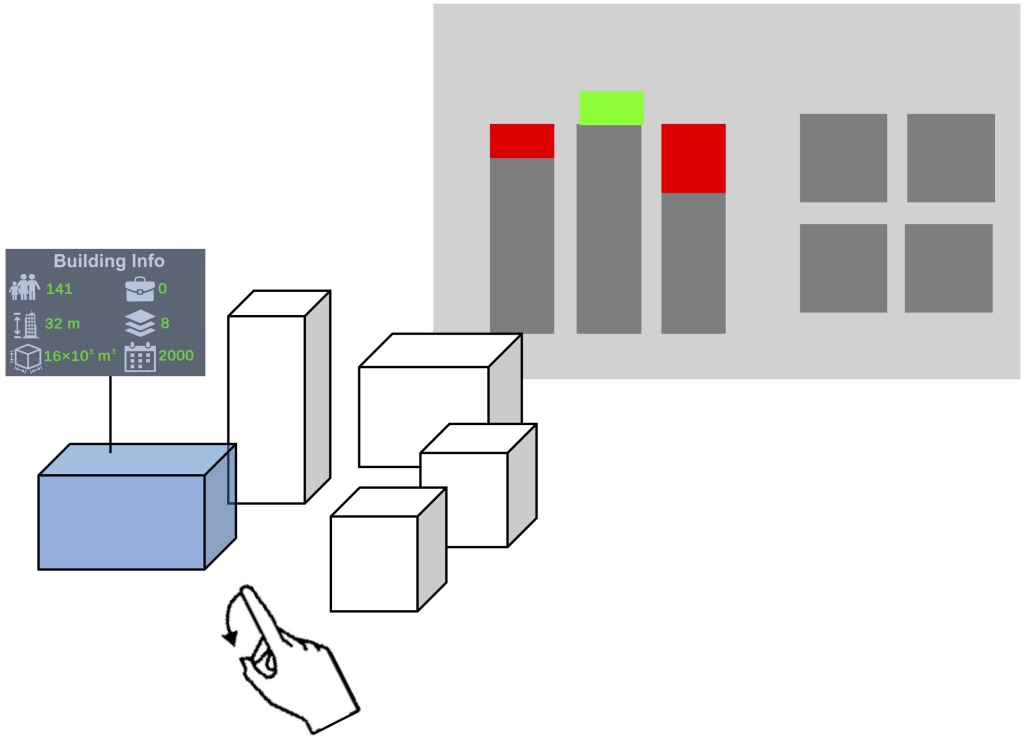

Figure 2: Functionality: Getting Information by clicking on an object

If you want some information about one building, you just click on it and a popup window appears, which shows all the relevant info about this specific building. Additionally the clicked building also gets highlighted in a blueish color. The content of the popup that we chose is explained in more detail in section 4.4.3. If you click again, the popup and highlight disappears.

2.4 Removing old buildings

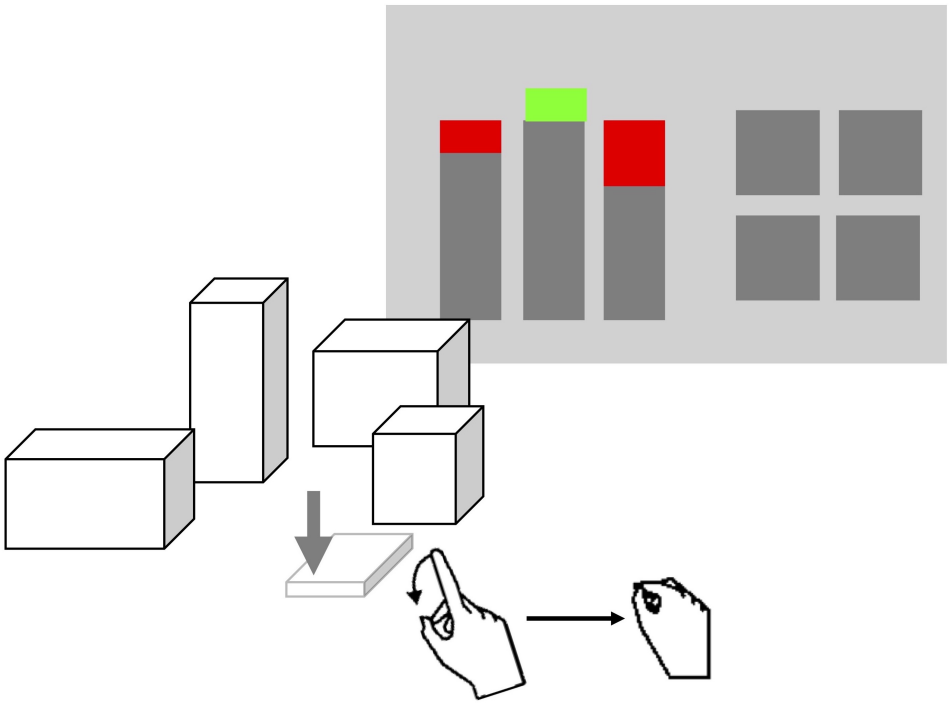

Figure 3: Functionality: Removing old buildings by tap and hold

To remove existing buildings we implemented a new gesture, called *TapAndHold*. This means you just have to click on the building you want removed and hold that click for about half a second. Then the buildings slowly scales down until we just see a transparent ground plane. This process can also be reversed by applying the same gesture again.

2.5 Adding new Buildings

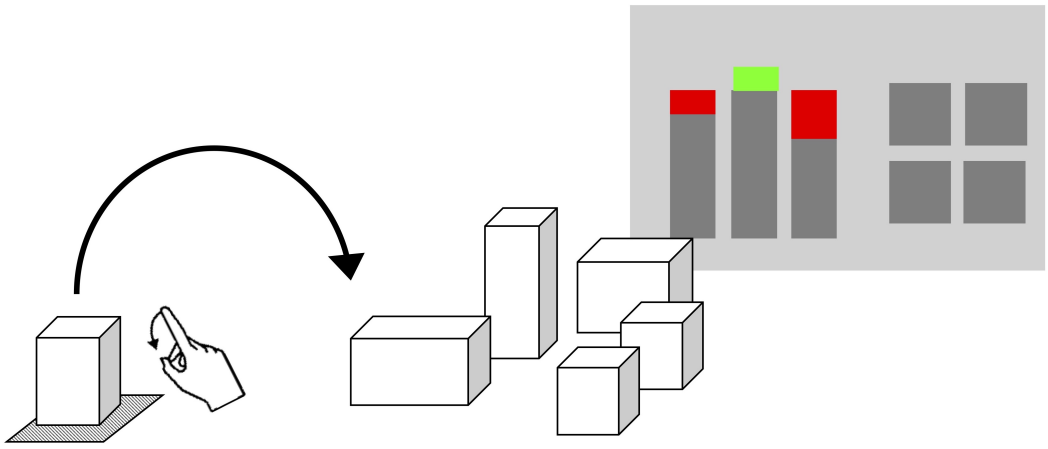

Figure 4: Functionality: Adding new buildings by positioning the physical marker

To add new buildings we use physical markers. We chose this so that the process gets more natural and easier to use for people who are not used to mixed and virtual reality. You just have to grab the marker, look at it and the virtual model of the new building is displayed on top of the marker. Then you can position it where ever you want and if you click on it, it gets integrated into the model and acts like the other buildings which are already existing. To reverse the process you can just click on the marker itself and take the building out of the model and start the tracking mode again. Also if you have a tracked building somewhere and it you want to remove it, you can either look at the marker and position it there or use the tap and hold gesture explained in the previous section to just disable it for a moment. It will popup again as soon as you look at the marker for the next time.

2.6 Visualize

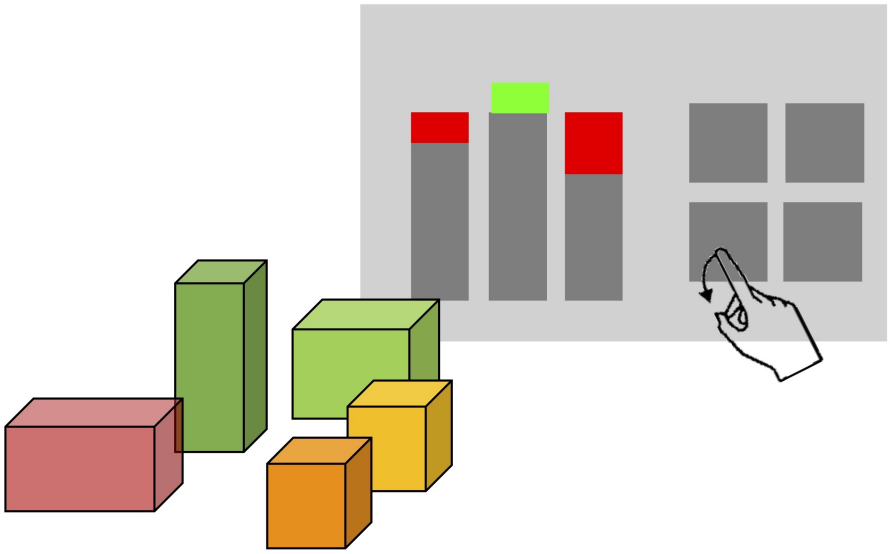

Figure 5: Functionality: Visualize by coloring the buildings according to the attributes

If you want to see the attributes as a whole, there are buttons on the info panel which color each building rule based by their attributes. The colors go from green over yellow to red and there is a legend which described the color range for each attribute.

2.7 Additional functionalities

There also additional functions like a button to move the panel, one to place the model somewhere else in the room and one to reset the scene and start planning anew. Those are all also on the info panel on the right upper corner.

3 Use Case

Urban planning is a complex task in which many parties are usually involved. Depending on the project, the process until realization can take many years. Especially when the area is big in its dimension or the location is of big importance, many different ideas and scenarios needs to be developed. Usually it starts by sketching and ends with a sophisticated plan. Before reaching the final result, many try and error can happen.

3.1 Target Audience

From laymen until experienced urban planners need to rethink all possible variations when beginning with a new planning process. To make this time consuming and extensive process more efficient for all kind of people who are involved in this complex urban planning process, we developed a mixed-reality application which serves more as a framework in the beginning of this process to play around and try different possible scenarios. This can reduce the amount of time for finding proper scenarios dramatically.

3.2 Example Project

To demonstrate the usefulness of our application, we chose a current planning project in the city of Zurich called Neugasse. The location can be seen in Fig 6.

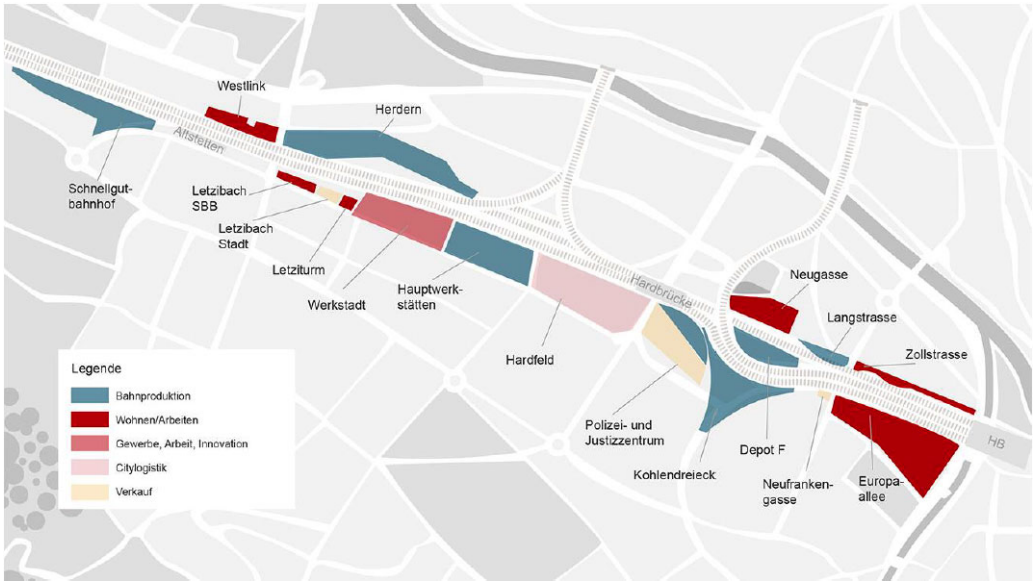

Figure 6: All the land owned by SBB along the rails in city of Zurich.[1]

SBB is releasing three areas for urban development. One of them is Neugasse, which is currently used as a repair station for trains. This construction will be torn down and the parcel will be rezoned("umzonen") to a residential zone. In Fig 8 we can see the old SBB repair station and the extend of the Neugasse project in red.

Figure 7: Image of the current use of Neugasse area with the SBB repair station for trains.[2]

The requirements for this area is to provide new living space for around 900 people and new working space for 250 people. To reach this goal many workshops are organized with the public together to develop new ideas in smaller groups. Fig **??** shows an example how this workshops are taking place in groups and the results look like.

Figure 8: Insight to the workshops organized for Neugasse project with the public. A 3D model is build for the chosen idea[3]

4 Methodology

4.1 Mixed Reality Toolkit (MRTK)

The Mixed Reality Toolkit is a collection of scripts and components to support developing applications for Microsoft Hololens[4]. Ones downloaded from Unity, we can import it as a custom package, which will add all the scripts to the current project. Another benefit of this Toolkit is, that we can set project settings suitable for mixed-reality applications only with a single click. The scripts and components can cover a broad range of applications. However, we used the following scripts of the MRTK for our application:

- InputManager: To handle all the inputs from the user (e.g. gesture) for the following tasks.
- SpatialMappingManager: Required for the 3d spatial understanding of the environment. In combination with the SpatialMappingObserver and the ObjectSurfaceObserver, it is used to find flat surfaces like tables.
- GazeManager: It handles everything related to a gaze ray that can interact with other objects.

4.1.1 Gaze

The GazeManager has all the control over the gazing functionality. In this application, we use gazing for four tasks:

- We need to gaze to the physical space for placing the model to an appropriate location.
- Our GUI contains many clickable buttons. A prerequisite for interacting with them is to let the device know which button is currently gazed.
- For a better user experience we change the color of each building in our model when hovering over it with the cursor. The system needs to know in each frame, if a building is currently gazed and which one it is to update its texture.
- MRTK class Billboard is used to keep the popup window oriented towards the user all the time.

[4]Microsoft 2018b.

4.1.2 Gesture

Similar to the previous section, InputManager handles all the gesture inputs when running the application. Fig 9 shows how clicking virtual objects works with the in-build tapping gesture called *Air Tap* of the Hololens.

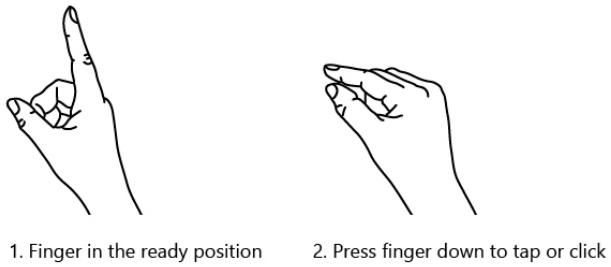

Figure 9: After targeting a clickable object with gazing, the tapping gesture called Air Tap can be used to "click" the object. Once the index finger is in a ready position by raising it up, the finger can be pressed down to select or tap and then back up to release the tap.[5]

The following functionalities are added to the script:

- TapToPlace: This class is used to place the city model in the beginning of the application to a flat surface in the physical space. Additional, the spatialMapping (see section 4.1.3 component is used to scan the room and find flat surfaces.
- Interpolator: interpolates position, rotation or scale when moving the object around the environment for placing it (during TapToPlace process).
- HandDraggable: Allows to drag and drop the info panel for placing it to another position in space.
- OnInputClicked: Needed for several tasks like clicking on building for getting more information in a popup window, selecting different buttons on the information panel (coloring model, replacing or reset model) or for placing models tracked with markers.
- OnInputUp/OnInputDown: Those functions are used to implement our own input gesture called *TapAndHold*. It is activated when OnInputDown is triggered and for the following 50 update steps, OnInputUp is not invoked.

We can summarize that gaze and gesture together are core elements for the interactivity.

4.1.3 Spatial Mapping

To make the mixed reality experience more realistic, the application need to interact with the physical space around the user. SpatialMappingManager is needed to get the information about the physical space. It manages the scannin
If the room and generating a meshed representation of the space and also for finding flat surfaces through another script to forward this data to other components of the script. The room is being scanned all the time and when changes occur (dynamic space) it will update the current mesh. Fig 10 shows a clip how the mesh is visualized during the placing mode of the city model. Once the model is placed, the mesh visualized disappears and the mode changes, so that all the functionality of the application can now be used.

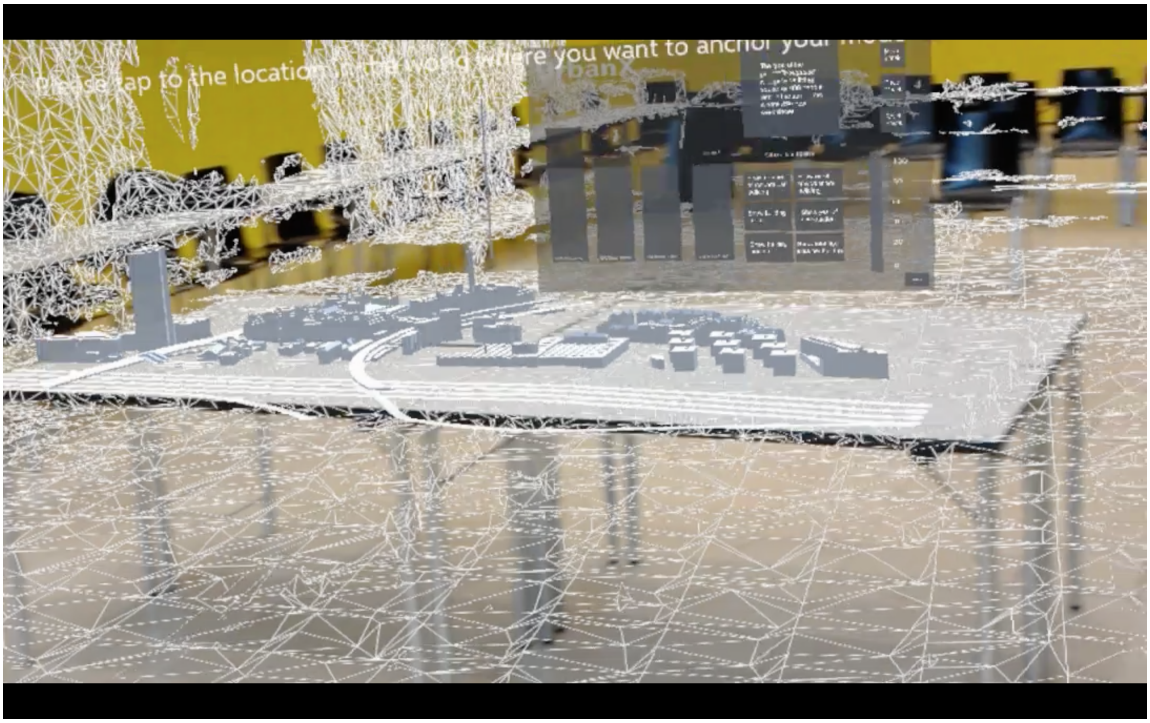

Figure 10: Physical space represented with meshes when trying to place the model to a flat surface.

4.1.4 Sound Output

Since the Hololens has integrated speaker to output sound, we use this capability to inform the user when an activity is successfully triggered. We have the following two cases where the hololens should output a sound:

1. Selecting a building in the city model to enable a popup window with information

about the selected building.

2. Placing a new tracked building with physical marker to the city model.

In each of the two cases a unique sound output indicates that eather a building is successfully selected or a tracked building is inserted to the building. Even when this process could be seen by changing the color of the building when one of the two cases happens, it is still an intuitive tool similar to clicking sound on traditional personal computer.

4.2 Vuforia

Vuforia is an augmented reality software development kit (SDK) for mobile devices[6]. It is the market-leading solution for integrating AR projects with Unity[7].

4.2.1 Integration

For using Vuforia SDK in our Unity project we can process similar to the MRTK. There is a Vuforia package especially developed for Hololens applications in the Asset store of Unity[8]. After imported the package to our project, we can use several scripts. For our task, we need to design personalized markers which will be used to track buildings. First, we need to create a database with all the markers inserted. This database must be connected to the unity project, so that when starting the tracking through the Hololens, the camera knows which marker in the physical world is connected to a virtual object.

4.2.2 Marker detection

Onces the app starts, Vuforia is activated. It is constantly searching for those predefined markers. The recognition quality of each marker can be evaluated at the Vuforia website. Once the marker is included in the database it will show all points of interest which the algorithm is searching in space. The more distributed over the entire marker the better is the tracking performance. An indicator with stars (from 1 star to 5 star) shows how good the marker is for tracking.

4.3 Data acquisition

Since we used a real world project example Neugasse for our application, the 3D city model data needs to be also from this area. The data from for the city model was taken fro Stadt

[6]Vuforia 2018b.
[7]Unity 2018.
[8]Vuforia 2018a.

Z/"urich and all the

4.3.1 3D building data

Stadt Zürich recently published all the 3D buildings of the city as open data. We chose the 3D-Dachmodell (LoD 2) dataset which is a detailled model of buildings including the roof structure[9]. To chose only the area of interest from the entire city model we load the geodatabase into CityEngine. This is a 3D modeling software application developed by Esri R&D Center Zurich[10].

4.3.2 Attributes

Unfortunately, the open data from Stadt Zürich provides only spatial data and not textual data (attributes) because of privacy reason. For that reason, the attributes had to be collected manually in the official online map-portal called *Maps of Switzerland* from the swiss federal government[11]. Under the section Register of Buildings and Dwellings (Wohnungs- und Gebuderegister) we can download a PDF with some attributes for each building.

Figure 11: Chosen area for the city model. Yellow dots can be clicked to show some information about the corresponding building.

In Fig 11 we can see the buildings used for our model. For each of the building there is a yellow dot which can be clicked to show some object information. To get more detail information of the clicked object, there is a button to download a PDF with detailed

[9] Zürich 2018.
[10] ESRI 2018.
[11] Government 2018.

information (in red). We could extract the following attributes from this PDF (see Fig 12):

- Year of construction
- Number of floors
- ground area
- type of usage (resident, work or mixed)

For our city model we had 53 separate buildings. For each of them this process had to be done manually.

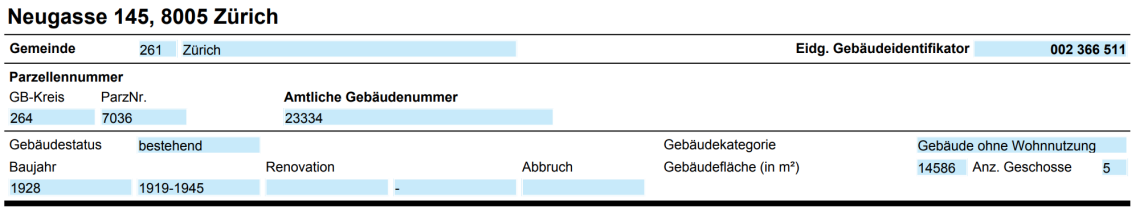

Figure 12: Extract of the detailed object information from the geoadmin portal.

4.4 Graphical User Interface(GUI)

We divided the building models into static and tracked buildings.

4.4.1 Static Model

The static buildings are the ones which are already existing. That's why they are called static, they can't be moved around, only deleted with the *TapAndHold* gesture. They are visible from the beginning and define the area we are looking at. Initially they are colored in white and have full opacity (see fig. 13).

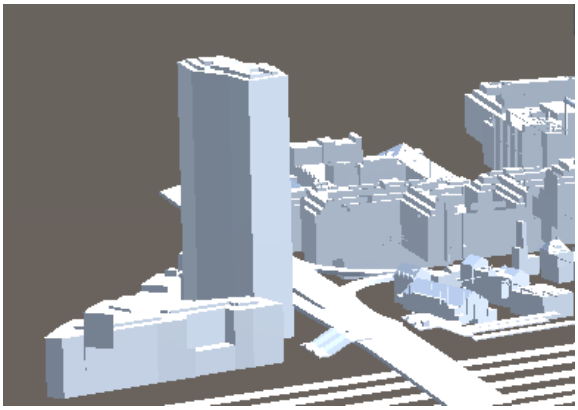

Figure 13: Example of the static buildings

4.4.2 Tracked Buildings

The tracked buildings are the ones which are not existing yet. They can be actual existing buildings or new ones, it doesn't matter, you just need to have a digital 3D model. The buildings appear as soon as you look onto the marker. They are also white, but have transparent to show that they are not part of the model yet.

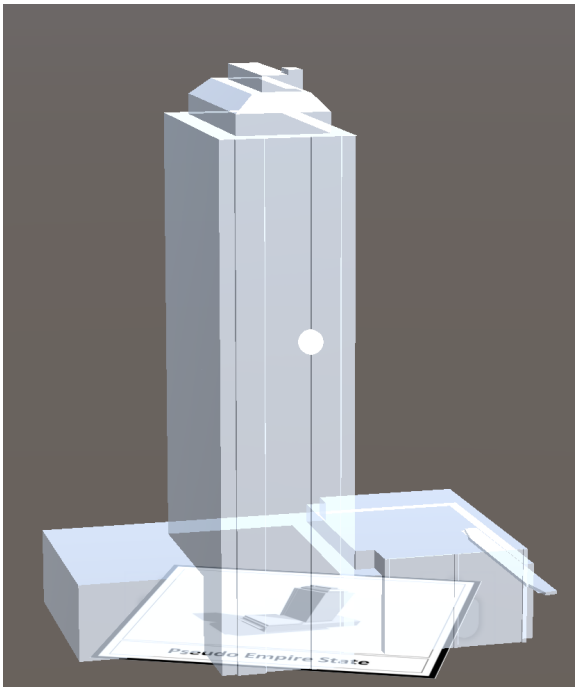

Figure 14: Marker with the virtual tracked building on top

4.4.3 Popup window

When you click on an static building, a popup window appears. There is also a smaller version of the same popup on every tracked building, so you can immediately see which tracked building to choose next. This is how the interface of the popup looks like:

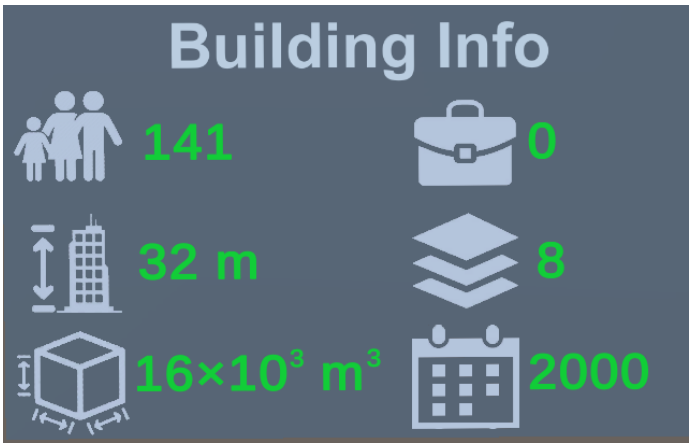

Figure 15: Popup Window: Showing information about one specific building

From the data acquisition described in section 4.3 we chose those six attributes to show on the popup window (but those can easily be adapted to different demands):

- Number of residents: We derived this number from the gross floor area if the buildings with type "resident" by dividing it by a factor of 35. The gross floor area we calculated by multiplying the ground area with the number of floors. This is of course a very unsophisticated approach which could be refined for real applications.
- Number of workplaces: This is calculated in the same way but with the buildings labeled as "work" and the factor is 40 instead of 35 because offices often take up more space because of storage rooms etc.
- Height: We could not get this data in the data acquisition step, so we took the number of floors and multiplied them with an average floor height of 4m.
- Number of floors
- Volume: Here we use the ground floor area from the data acquisition step and multiply by the height calculated as explained above.
- Year of construction

4.4.4 Information Panel

The info panel holds one section for total information about the whole area and one for all the buttons and functionalities. The main ones are explained in this picture:

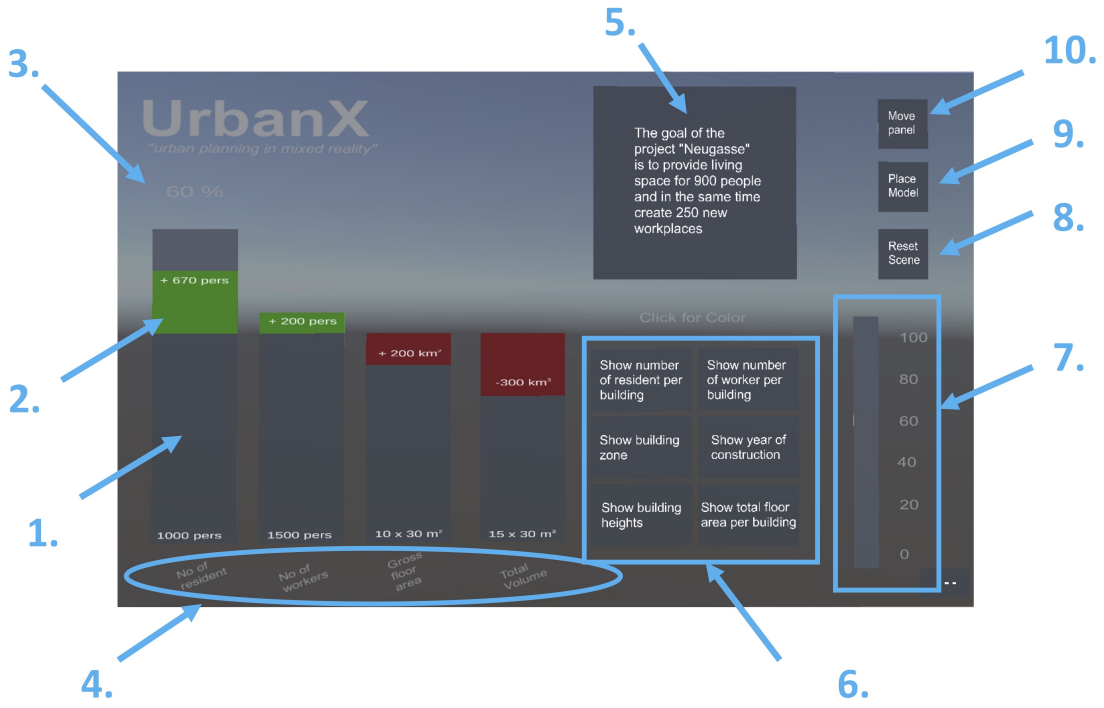

Figure 16: Info Panel: Different functionalities

Information part:

1. Bar chart displaying the total change of attributes over the whole model. We chose four of the attributes: Number of residents, number of workplaces, gross floor area and volume. In the begin each bar starts at the same height and the absolute value is displayed the bottom of the bar.

2. If there are changes applied to the model they are displayed in red or green above the already existing bar. The absolute number is displayed in top of the new bar and always shifts its position accordingly.

3. For some bars we can also visualize the goal of the project, in this case the 900 additional residents and 250 workplaces. You can see how close you are to that goal in absolute numbers at the change in green, but to have a relative value this

percentage is added at the top. Also there is a bar in a slightly different color to show where the goal is.

4. The type of the attribute summarized by each bar is displayed at the bottom of each bar.

5. There is a small info text at the top to inform the user about the project and the goal that should be reached.

Functions part:

6. There are six buttons which enable the user to color the whole model rule based according to the attributes they possess. They are more or less the same as the ones on the popup window explained in the previous chapter.

7. The legend contains a color bar with the specific attribute values for the corresponding color, so that we can verify and read each color value from the legend.

8. Button to reset the scene. It removes all the added buildings and also reestablishes the buildings which were removed with the tap and hold gesture. The buildings and the legend are colored white. It sets the scene to the beginning so that the planning can start anew.

9. Button to place the model again. When the application starts, this mode is automatically enabled, but with this button the user can change the position of the model anytime without having to restart the app.

10. Button to move the panel. It is clicked, the button gets blue and the whole panel become hand draggable. The other buttons are disabled until the move button is clicked again and the panel can't be moved anymore.

4.4.5 Color Design

For an optimal user interface (UI), we tried to use colors which first suites to mixed-reality usage (some colors are more suitable than others) and secondly are chosen in a meaningful way. For defining an overall color scheme, we used Colorbrewer 2.0[12], which is an online platform for color advice for a Map. This helped to discover which colors fits together.

Attribute coloring

As we could see in section 4.4.4 some buttons are available on the information panel for rule-based coloring based on the chosen attribute. For each of the attributes a color range

[12]ColorBrewer 2018.

starting at green (for lowest values) over yellow to red (highest values) is used. One easy solution would be to change the color linear to the attribute values. The interpolation could be done by defining a ration with the following formula

$$ratio = \frac{val_{curr}}{val_{max} - val_{min}} \tag{1}$$

Where val_{curr} is the attribute value of a building and val_{max} and val_{min} are the maximum and minimum values containing in the city model.
Since most of the attribute values are not evenly distributed, this solution would lead to ineligible coloring. Most of the buildings are colored either in red or green. To overcome this issue and provide a more suitable color gradient we use tangens hyperbolicus function $tanh(x)$ which is more sensitive around the middle of the range

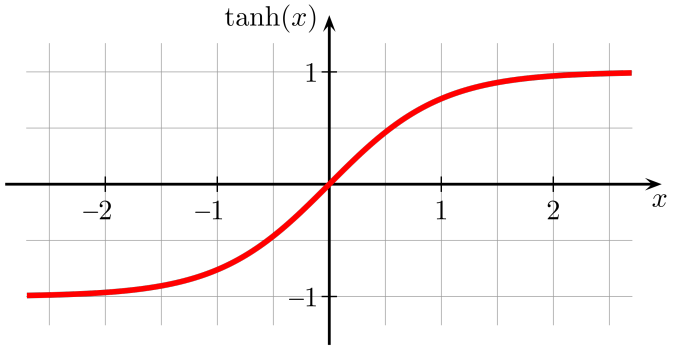

Figure 17: Tangent hypebolicus function

Additionally, we scale the gradient with λ which depends on the distribution of an attribute in the model. Lambda is defined as followed

$$\lambda = \frac{val_{max} - val_{min}}{Q_{0,75} - Q_{0,25}} \tag{2}$$

where $Q_{0,75}$ and $Q_{0,25}$ are 75%-Quantil and 25%-Quantil, to fit the color gradient properly to the value distribution. When single extreme values contains in the attributes, this lambda scalar will make those less affecting the overall color gradient. Summing up all those values together we get the following formula to get the ratio (from 0 to 1 where red is 0 and green is 1)

$$ratio_{final} = tanh(\lambda \cdot ratio_{curr}) \tag{3}$$

Only for the attribute zone, a predefined coloring is used because a color gradient is not possible with unique zone definition. The coloring based on the zone is the same as in the *Zonenplan* of Zurich.

Hovered/Clicked color

As we already mentioned, that wen hovering or clicking a building in the model, the color should change to let the user know that a building is hovered or clicked. For the hovering, there are two possible color changes, depending on the current color of the building. When the sum of the RGB color channels are higher than 2.4 (each color is in range between 0 and 1), each channel will be scaled by a value 0.8 to make the hovering a bit darker. However, if the sum of the RGB channels is less than 2.4, each channel will be scaled by a value of 1.3 which makes the hovering color brighter as its default color. The reason for differentiating this is because highly bright or dark colors are visually not appealing in a mixed-reality application. For the clicking, we use a predefined blueish color. The reason for that is because our model coloring based on attributes have a color gradient from green over yellow to red. In this case, the blue color for clicking is unique and will not be confusing with all the other colors in the city model.

4.5 UrbanX Classes

We wrote several different classes ourselves to manage the different parts of our application. We used the basic principles of object oriented programming and subdivided the different functionalities into reasonable classes which interact with each other. The classes have a loose hierarchy order so that the main information flow is ordered and centralized. This figure () shows this hierarchy of the classes:

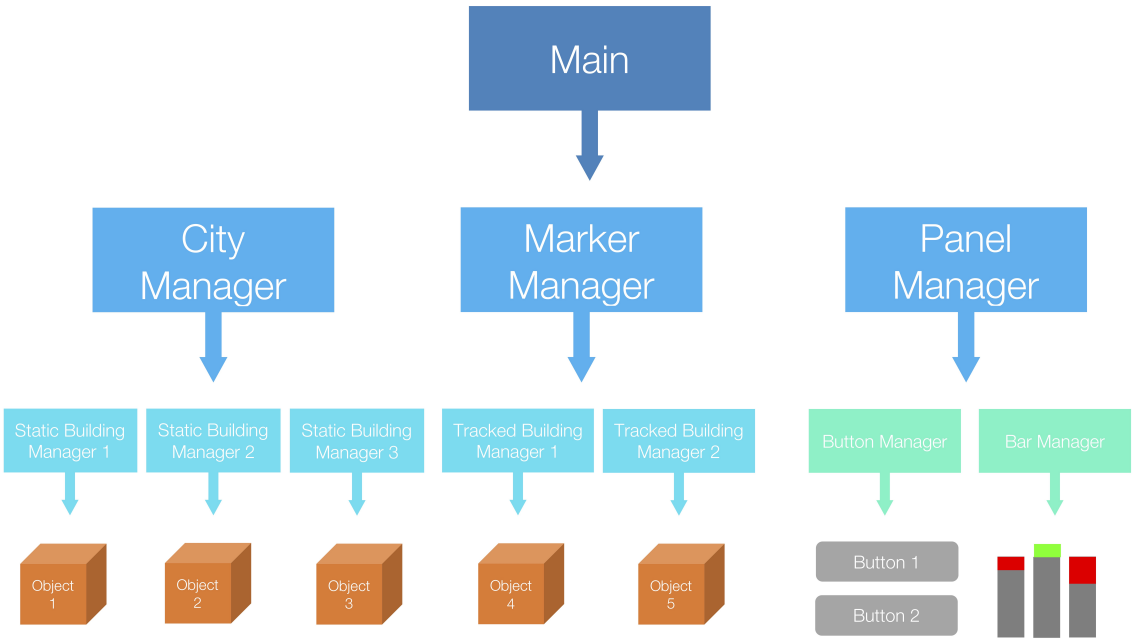

Figure 18: Graphical representation of the different classes and their hierarchy

4.5.1 Main

The main class holds all strings together and sends the important data to three subordinate classes. It contains most of the information which needs to be accessed from several classes. It decided in which status of the app we're in and stores the data for the main colors and goals of the project. There is also a subclass called *CityInfo* which is used to store all the attributed of each building. It contains several function for example for calculating total information or extracting the active buildings. To integrate it into Unity we attached it to an empty GameObject.

4.5.2 CityManager

This class manages everything which has to do with all the geometrical buildings. So this is where the hover feature is defined and it stores which building is clicked or looked at. It also stores the height at which the model was placed so that the additional buildings can set to the exact same height later on.

4.5.3 MarkerManager

The main purpose of this class is to deal with the recognized physical markers. It stores the information if each building is in view or not. Each marker has building defined it belongs to and when the building is in view, a smaller version of the popup window explained in section 4.4.3 appears, which is also done in this class. The marker can be clicked, which makes the tracked building reappear on the marker and it will be tracked again.

4.5.4 StaticBuildingManager

This class is attached to each of the buildings-objects in Unity. It contains functionality to delete the building and let it reappear with the *TapAndHold* function. It also managed what happens when you click on the building, which in this case lets the popup window apprear and disappear.

4.5.5 TrackedBuildingManager

This class is similar to the *StaticBuildingManager* class, it also handles the click functionalities, but here they have another result. When you click on the tracked building it gets integrated into the model and is not tracked anymore. When you do the *TapAndHold* gesture it disappears until the marker reappears in the field of view. // Additionally to these implementations this class holds the instance of the Kalman Filter which is explained in section 4.5.7.

4.5.6 PanelManager

This is the largest class. It handles all the things which have to do with the attributes, the textual data. It s responsible to read the data from a list of strings into the instance of *CityInfo* which is stored in the *main* class.
When the app is initialized on the beginning, this class handles all the preparation like adding scripts to the builings, putting them to the right position, orientation and layer. It therefore also holds the function to reset the model to the initial state, which enables to start from scratch without having to start the application anew.

One of the main functions is called *updateBarChart*. It is called at every update step and checks through the model if anything has changed, meaning if a building was added or if a building was deleted. The deleting step of the building does not go instantly, we distributed that over several update steps so that there is a nice animation of the house going down. And because we call the *updateBarChart* function at every update step, the

numbers and the bar chart also just gradually go down instead of jumping to the new value, which creates a nice visual animation.

4.5.7 Kalman Filter

The tracking is provided by Vuforia and explained in the section 4.2. The recognition of the targets works very well and is quite accurate. However when you move the marker, the tracking is very jittery. It lacks smoothness and the shaky image impairs the user experience. With help of the framework proposed by Daniel Laumer, Hasret Gümgümcü, et al. (2017), a filtering method for the position of the tracked buildings was introduced. There a Kalman Filter was used to smooth and predict the position of tracked hand joints using an Intel RealSense camera. The filter assumes a movement model and for each step predicts the next position. This position is then corrected with the actual measurements by taking a weighted mean between the two positions. This leads to a smoothed trajectory. This tool can be easily applied here too. We just take the position of the marker and feed it into the filter algorithm for every update step. We were able to make the movement of the tracked buildings noticeably more steady and consistent.

4.5.8 Connecting the different classes

This classes are of course all connected with each other. As mentioned before the main function holds most of the globally important data and sends the necessary information to the three main classes *CityManager*, *MarkerManager* and *PanelManager*. The tracked buildings have a special kind of relation: When they ae not yet added to the model they are controlled by the *MarkerManager*, which makes them appear and move with the marker, but then they are integrated into the model they act as a static building which is managed by the *CityManager*. When they are in the tracked state, they interact with the Kalman Filter to correct their position and ensure a smooth tracking experience.
The static buildings send their information to the *BarManager* , for example when one building is deleted, so that the total bar chart can be updated. The *ButtonManager* on the other hand send information to the buildings in order to color them by their attributes.

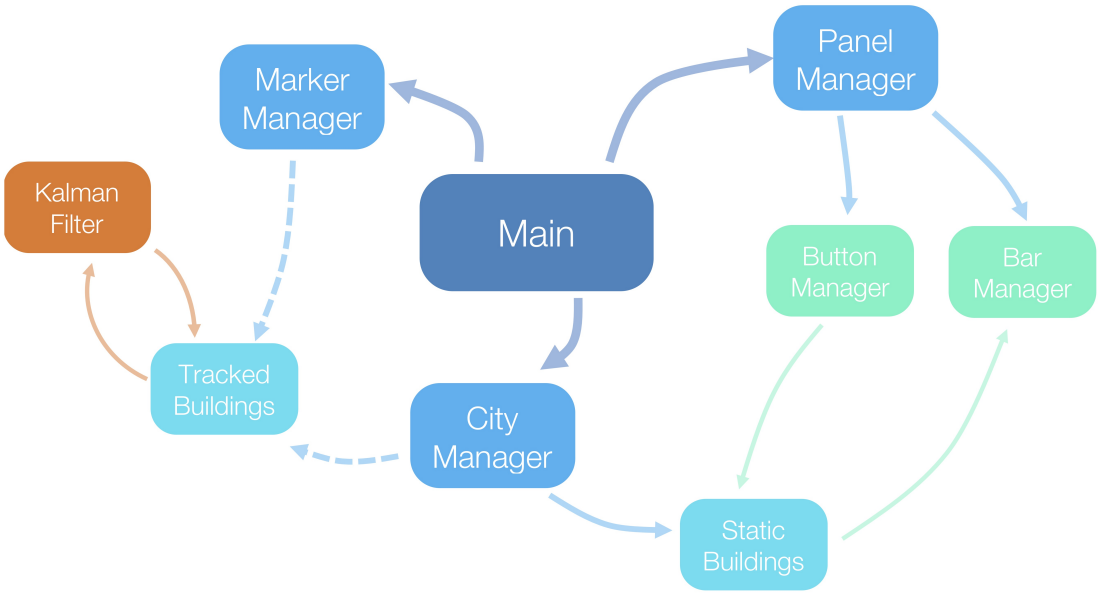

Figure 19: Graphical representation of the different classes and how they interact with each other

5 Problems and Challenges

During the development, from the very beginning to the very end, we have faced many problems and challenges.

5.1 Idea/Use case

Some could be solved easily whereas other took much more time. A general problem in the beginning of this project was to find an idea with a real use. There was no restriction at all to choose a topic. After a lot of research what is already in the market and which needs still are not fulfilled with mixed-reality applications, we decide to develop an application useful for urban planning. Because of lack of knowledge in urban planning we had to change our main goal several times during the development process. The final direction was set after we organized a meeting with an urban planner at the ETH Zurich and brainstormed some ideas to make the application more valuable and useful for real cases. The idea of adding attributes to the model was also an advice from this meeting.

5.2 Unity Packages

To start with Hololens development in Unity we need to import the mixed-reality toolkit (MRTK) package. It includes many useful classes and functions. It required some time and effort to understand what those classes are really doing before we can add it to our project. In general, several project settings needed to be defined before we could start working with it.

For our tracking part, we found two possible tools (ARToolKit or Vuforia) which could work with Hololens. We went for the first one in the beginning. Unfortunately, it was the wrong choice in our case. There was almost no support/community in the web for this toolkit. Because of unclear description what each part of the toolkit is doing, it was almost impossible for us to make big changes to the code. After many attempts to understand the code, we gave up and decided for using Vuforia. It was much better to work with because it has a toolkit especially for Hololens development in Unity. Since the Hololens development in Unity and Vuforia toolkit for Hololens are new and still developing a lot, we had some issues with the Unity version used. First, Vuforia and MRTK packages did not work well together and made it impossible to develop. After some research in forums, we found out that the Unity version used (2017.2.1) was the problem. We need to update to another version to solve this issue. When we solved this problem, another version problem

started. It was important to use specific version of the universal windows platform SDK when compiling the app, instead of using the latest available.

When adding external packages (for coloring the outline of the building and MathNet for the Kalman Filter) the scene did not compile anymore. We could not fix this issue totally. The package for coloring the outline had to be removed and could not be used as planned. For MathNet we found a solution to still use it in our code. We had to build the project without this package and once it was successfully done, we add the MathNet package in Visual Studio to the project before sending the application to the Hololens.

5.3 Data

As already mentioned in section 4.3.2 we had to gather attributes for each building separately by downloading a PDF file for each of the building. This data acquisition process was more time consuming as expected. We had to insert those attributes from the PDF files manually in an excel file to extract it in the application once it starts. One annoying issue in CityEngine is that the normal vectors of the planes (a building is defined with many separate planes) are not consistent and are randomly set. Because of that, depending on the normal, some planes of the buildings are transparent so that the inner structure can be seen. The only function in CityEngine that we found was to invert the normal but not to set it all towards outside of the body. Maybe it is just a single click in the software which we could not find.

5.4 Coding

The scripts could be chaotic and not easy to understand. Even when we started well organized before coding, the problem was that the main idea of the application changed during the development process. For that reason, the classes and functions needed for the App has also changed many times. Anyway, the main starting point should be the main class in which all the other classes should be controlled. The separation of the classes is not fully satisfied because the same class needs to be referenced in many other classes as well, which can be confusing.

5.5 Technical

Many technical issues occurred during the development. One big problem until the very end was that the holograms were unstable. The result was that, when moving the Hololens

to fast, the colors were separated (the same object could be seen with a small shift in red, green and blue). In the end, this could be solved by enabling a check-box called buffer enabled. We found out that the combination of Vuforia and MRTK is still not fully satisfying and some internal problems were the reason for that. Compiling is a time-consuming process. The code needs to be built first in Unity and in the next step sent to Hololens trough Microsoft Visual Studio. The entire process takes around 5 minutes. Since the testing of the tracking part could not be done in an emulator, we were forced to do the debugging of the tracking part always on the Hololens which was a huge time issue. Even when using the Unity emulator for debugging and testing, one issue could still not be solved. The visual appearance (coloring, transparency, overall impression, etc.) is distinct between how it looks on the Hololens and how it looks on the emulator. In general, we can see that Unity is still in development to design it for Hololens applications. The community reports different technical issues for individual versions. This makes the development even harder, since some advises for error fixing are valid for some Unity versions.

5.6 Sharing

The single big issue that could not be solved is to implement sharing functionalities between multiple Hololenses. The idea was that multiple user can simultaneously see the same model in the same physical space. This would allow a more collaborative use of the App. We were able to use the sharing between two Hololenses. They could both see the same model. The problem was that a shared world anchor could not be set, so that each of the user saw the model at a different physical location. Without having the mixed-reality model in the same physical space for all the user, it is not possible to use the model shared because issues will occur when adding tracked objects to the model. Because of limited time, we had to give up this idea and focus on other parts of the application.

6 Discussion and Conclusion

We provide a framework which is implemented so that it can be easily extended. Although we used a specific urban planning project to demonstrate this framework, it can be adapted and replaced with any other urban planning project. Only a 3D building model with the attributes of the buildings are needed. New buildings as tracked buildings can be modelled in any 3D modelling software and included to the framework. The interface is designed visually appealing so that even laymen can understand easily the functionalities, so that human-computer interaction (HCI) works smoothly. Since mixed-reality (AR in general) usage is still in its very beginning, many limitations will be solved in future. The processing power is still limited. With technical process the devices can be designed simpler so that ordinary usage is possible. With more real usage and public awareness, the price of those devices will be reduced dramatically. This on the other hand will affect the progress in AR applications.
For the Hololens especially, another problem is still the limiting field of view. A new version of the Hololens will be released in near future which may have a bigger screen.

7 Outlook

As already mentioned, a collaborative urban planning in mixed-reality can be realized in future when multiple users can collaborate with the same virtual model. The app could be also extended with collision control, so that a new building cannot be inserted to a location where already another building is placed. Rule based conditions can be defined, to make the usage more realistic, e.g. allowing a maximal height depending on the current zone. Another more exotic idea is to use markers where the user can draw a 2D shape (edges of the polygon) and with image processing tools the computer can extract it. The height of this extracted 2D footprint can be manually adjusted. This would be a more interactive and fun way to sketch generalized virtual city models. As a conclusion, the variations and extensions of this framework is limitless, since mixed-reality applications for urban planning just started rising.

References

Bibliography

ColorBrewer (2018). *Color advice for cartography.* http://colorbrewer2.org. visited on 22-12-2018.

Daniel Laumer, Hasret Gümgümcü, et al. (2017). "Hand Tracking using Kalman Filter for Safe Human-Robot Interaction". In: *38. Wissenschaftlich-Technische Jahrestagung der DGPF und PFGK18 Tagung in München - Publikationen der DGPF*, pp. 743–748.

ESRI (2018). *Esri CityEngine.* https://www.esri.ch/de/produkte/cityengine. visited on 22-12-2018.

Government, Federal (2018). *Maps of Switzerland.* https://map.geo.admin.ch. visited on 22-12-2018.

ImmobilienBusiness (2018). *Zürich: SBB macht Areale für Stadtentwicklung frei.* https://www.immobilienbusiness.ch/zuerich-sbb-macht-areale-fuer-stadtentwicklung-frei. visited on 22-12-2018.

Microsoft (2018a). *Hololens Air Tap Gesture.* https://docs.microsoft.com/en-us/windows/mixed-reality/gestures. visited on 19-12-2018.

– (2018b). *MixedRealityToolKit Github.* https://github.com/Microsoft/MixedRealityToolkit-Unity. visited on 19-12-2018.

Unity (2018). *Vuforia as official Unity partner for AR.* https://unity3d.com/de/partners/vuforia. visited on 19-12-2018.

Vuforia (2018a). *Developing Vuforia Apps for HoloLens.* https://library.vuforia.com/articles/Training/Developing-Vuforia-Apps-for-HoloLens. visited on 19-12-2018.

– (2018b). *Vuforia Website.* https://www.vuforia.com. visited on 19-12-2018.

Zeitung, Neue Zürcher (2018). *Zürcher Neugasse-Areal.* https://www.nzz.ch/zuerich/zuercher-neugasse-areal-vorschlag-fuer-mehr-guenstige-wohnungen-geht-initianten-zu-wenig-weit-ld.1402975. visited on 22-12-2018.

Zürich, Stadt (2018). *3D-Stadtmodell.* https://www.stadt-zuerich.ch/ted/de/index/geoz/geodaten_u_plaene/3d_stadtmodell.html. visited on 22-12-2018.

List of Figures

Appendix

ReadMe for the code

```
UrbanX Version 1.00 06/01/2019
==================================================

Authors: Hasret Gümgümcü and Daniel Laumer
GIS and Geoinformatics Lab, Fall 2018
Institute of Cartography and Geoinformatics at ETH Zurich
==================================================

Abstract
————————
The goal of this project was to develop a Hololens application to do
urban planning in a mixed reality environment and provide a
framework which makes the process more efficient and immersive. The
Microsoft Hololens is a recently developed pair of headmounted mixed
reality smartglasses, which lets the user place virtual holograms in
the real world and interact with them using specific gestures. The
application should display the buildings in the region of interest
and let the user access the attributes of each building in an easy
and intuitive way. Also there should be editing functionalities so
that the model can be changed to the users content.

So we introduce UrbanX, our hololens application for efficient and
interactive urban planning. We used the open data from Stadt Zürich
to get 3D models of the buildings. To remove buildings we developed
a gesture called TapAndHold which removes the building and only
leaves a transparent ground plane indicating where the building was
before. This process can easily be reversed with the same gesture.
For adding new building we used markers printed on cardboard which
the hololens can recognize and display virtual new buildings on top
of it. This way, the new buildings can be positioned and oriented
manually by just moving the cardboard markers, which make the whole
process more intuitive and easier to handle since drag and drop on
virtual holograms is not that easy. Also people like to still have
things they can touch and move. With this solution, there we created
an interesting mix between the virtual and real world and are thus
blurring the border between the two realities.
Information about each building like for example size, year of
construction or number of people is stored and can be displayed
either detailed for a single building or for the whole model with
the help of rule based coloring. The overall state of attributes is
shown in absolute and relative bar charts to keep track of the
change. Since planning projects often have to achieve a certain goal
of number of new residents or similar, we added this element
integrating it into the statistical bar charts and telling the user,
how far to the goal he progressed. This adds a game-like component
and allows a more playful and incentive experience.
Further information is available at http://gis-lab.ethz.ch

For more information or the report feel free to contact us.

Requirements
————————
- Computer with Windows 10  (macOS or Linux are not supported)
- Microsoft Hololens
```

- Unity Game Engine (Version 2017.2.1)
- Mixed Reality Toolkit SDK (Version 10.0.16299)
- Vuforia SDK (from Asset Store)
- Microsoft Visual Studio

Installation/Deployment Guide

1) Make sure you have the „UrbanX“ software or download it from the GIT repository (contact us for access permission)
2) Install Microsoft Visual Studios Community at https://www.visualstudio.com/de/downloads/?rr=https%3A%2F%2Fwww.google.ch%2F
3) Open the project solution from the build at UrbanX/AppFINAL/UrbanX.sln
4) Using the top toolbar in Visual Studio, change the target from Debug to Release and from ARM to X86.
5) Click on the arrow next to the Local Machine button, and change the deployment target to Device
6) Click Debug > Start without debugging. (If this is the first time deploying to your device, you will need to pair using Visual Studio)

For more information see https://docs.microsoft.com/en-us/windows/mixed-reality/using-visual-studio

Edit and Build

To edit the Unity Project and code:
1) Make sure you have the „UrbanX“ software or download it from the GIT repository (contact us for access permission)
2) Install Unity Game Engine (the free version, called „Personal“ is sufficient). We had some problems with the version and ended up using the version 2017.2.1. Maybe other versions work too. You can download older version at the following link: https://unity3d.com/get-unity/download/archive?_ga=2.118178830.6609551.1546722431-76478782.1499594340
3) Open Unity, create a new account online and sign in (you will need to participate in a small survey)
4) Open the unity project by File -> Open -> and then select the Folder with the name
5) Download the Vuforia Package, the easiest way is via the Unity Asset Store at https://assetstore.unity.com/packages/templates/packs/vuforia-core-samples-99026
6) Install the Mixed Reality Toolkit (MRTK) SDK. Important: Download the version 10.0.16299.91 at the following link: https://developer.microsoft.com/en-us/windows/downloads/sdk-archive
There you can also download the emulator to test the app on the computer
7) Via the MRTK tab at the top of the window, apply the project and scene settings.
8) Install Microsoft Visual Studios Community at https://www.visualstudio.com/de/downloads/?rr=https%3A%2F%2Fwww.google.ch%2F
9) Open the code files by clicking the .cs files under UrbanX/

```
Assets/Scripts in Unity
10) Edit the Unity project and files to your hearts content

To build a new app after the changes:
11) Build the app as explained in https://docs.microsoft.com/en-us/
windows/mixed-reality/holograms-100
12) Open the created solution (.sln) file
13) Under References find the MathNet reference, right click and
remove it.
14) Right click on the reference and install MathNet.Numerics
directly from NuGet. Important: we had success with the version
3.20.2, not sure if the others work
15) Deploy as described above

Useful links
————————
We faced several problems when developing and this forums and links
helped us a lot:

- https://docs.microsoft.com/en-us/windows/mixed-reality/
holograms-100 (General Info)
- https://github.com/Microsoft/MixedRealityToolkit-Unity/issues/1461
(Vuforia and MRTK problem)
- https://github.com/mathnet/mathnet-numerics/issues/575 (MathNet
problem)
- https://github.com/Microsoft/MixedRealityToolkit-Unity/issues/1699
(Wobbly holograms)
- https://developer.vuforia.com/forum/hololens/hololens-app-vuforia-
live-stream (Streaming really not possible)


==================================================

Contact Information
———————
```

Daniel Laumer
ETH Zurich
daniel.laumer@gmail.com

Hasret Gümgümcü
ETH Zurich
hasret.g46@gmail.com