FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Masterarbeit in Informatik

3D virtuality sketching: a freehand sketch tool for conceptual urban design in architecture

Violin Yanev

FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Masterarbeit in Informatik

3D virtuality sketching: a freehand sketch tool for conceptual urban design in architecture

Virtuelles 3D-Skizzieren: ein Freihand-Skizzierwerkzeug für die städtebaulichen Entwurfsphasen in der Architektur

Author:	Violin Yanev
Supervisor:	Prof. Dr. Gudrun Klinker, Prof. Dr. Frank Petzold
Advisor:	M. Sc. Eva Artinger, Dipl. Ing. Gerhard Schubert
Date:	July 15, 2012

Ich versichere, dass ich diese Masterarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 15. Juli 2012 Violin Yanev

I hereby declare that this master thesis is entirely the result of my own work. Where I have consulted the published work of others, this is always clearly attributed.

Munich, July 15th, 2012 Violin Yanev

Acknowledgments

Thanks to my advisors Gerhard Schubert and Eva Artinger for their long-lasting support and advice which was invaluable for the successful completion of this work.

My sincere thanks go to many friends and colleagues for scientific discussion, advice and continuous support always so greatly appreciated. Thanks to everyone from the chair of Augmented Reality in the TUM as well. Last but at no means least, I like to thank my family for general education and never-ending encouragement and ongoing support!

Summary (English)

Conceptual design is an important stage of urban development in architecture. To integrate computer tools into the workflow of architects, the Collaborative Design Platform (CDP), which employs tangible interaction methods on a large-scale multi-touch table top, has recently been developed. The platform supports finger gestures, fiduciary markers, and also scans arbitrary objects converting them to digital 3D shapes.

The goal of the present project was to extend the CDP by 3D capabilities and add a new kind of interaction with the system, namely through sketching. A large-scale vertical touchscreen was connected to the CDP to visualize a virtual equivalent of the multi-touch table top and its objects in a 3D perspective. The additional touchscreen was also used as an input device for sketching. Both large-scale displays (horizontal and vertical) essentially represent a different view of the same urban context. Thus, moving a building model on the table also moves its virtual representation on the vertical display accordingly.

Four different sketching modes were developed in this work – standard, extended surface, sketch paper and 3D mode. The sketching methods make use of the underlying virtual environment to understand and embrace 3D ideas. The standard mode maps sketches directly to the underlying urban 3D scene as if they were drawn with a laser pointer. The sketch remains related to its underlying object even after changing the viewpoint, or if the object itself moves. The second mode – extended surface – allows the user to select a surface on which he wishes to sketch, and applies all strokes to the plane of this surface. The third mode – sketch paper – employs a sheet of semi-transparent sketch paper commonly used by architects. It is possible to sketch on different paper layers and switch from one to another.

A special attention was devoted to the 3D sketching mode which performs 3D reconstruction out of raw, inaccurate, and possibly incomplete sketches. It has the advantage of speed compared to similar 3D sketching approaches and retains the roughness of the initial sketch and its visual appearance as a collection of raw strokes.

The new features of the CDP were tested in an evaluation carried out with ten students in Architecture. Participants were asked to perform a simple conceptual design task – to design a living area in an urban scenario. The evaluation criteria included frequency and time of use of each sketching mode as well as the user ability to conceive the dynamic interaction between both displays.

The evaluation demonstrated that architects are willing to use the CDP and its newly implemented presentation mode as supplementing tools for conceptual design. Nevertheless, they would hardly abstain from using traditional tools such as pen and paper. The evaluation also showed a demand for more functionality and customization of the system, which will be a task for a future research.

Zusammenfassung (Deutsch)

Der Entwurfsprozess ist eine wichtige Arbeitsstufe im Rahmen des Städtebaus in der Architektur. Um digitale Werkzeuge in den Entwurfsprozess einzubinden, wurde die Collaborative Design Platform (CDP) entwickelt – ein multi-touch Tisch, welcher Methoden physischer Interaktionen mit einschließt. Der Tisch unterstützt Fingergesten, Fiducial Marker und kann beliebige Objekte erkennen und in digitale 3D Formen umwaldeln.

Das Ziel des gegenwärtigen Projekts war es, die CDP um 3D Fähigkeiten zu erweitern und eine neue Art von Interaktion einzufügen – nämlich das Skizzieren. Ein 65″ senkrechtes Touchscreen-Display wurde an die CDP angeschlossen, um das virtuelle Äquivalent des multi-touch Tisch und der darauf liegenden Objekte aus einer 3D Perspektive zu visualisieren. Das zusätzliche Display wurde auch als Eingabegerät für das Skizzieren verwendet. Beide Anzeigen (horizontal und senkrecht) bieten eine unterschiedliche Sichtweise auf denselben städtischen Kontext. Das Verschieben eines Gebäudemodells auf dem Tisch bewirkt daher die gleichzeitige Verschiebung seiner virtuellen Darstellung auf dem senkrechten Display.

Vier verschiedene Zeichenmodi wurden in dieser Arbeit entwickelt – Standardmodus, Erweiterte Flächen, Skizzenpapier, und 3D Modus. Die Zeichenmodi verwenden die darunterliegende virtuelle Umgebung der Stadt, um 3D Ideen zu verstehen und in die Szene einzubetten. Der Standardmodus bildet Zeichnungen direkt auf das darunterliegende 3D Modell der Stadt ab, als ob man mit einem Laser-Pointer zeichnet. Die Zeichnung ist am darunterliegenden Objekt fixiert, auch wenn sich der Standpunkt des Betrachters ändert, oder wenn sich das Objekt selbst verschiebt. Der zweite Modus – erweiterte Flächen – lässt den Benutzer eine Fläche selektieren, und bildet alle Zeichenstriche auf die Ebene dieser Fläche ab. Der dritte Modus – der Skizzenpapier-Modus – simuliert eine durchsichtige Skizzenpapierrolle, welche oft von Architekten verwendet wird. Es ist außerdem möglich, auf unterschiedlichen Papier-Blättern zu skizzieren und von Blatt zu Blatt zu wechseln.

Eine besonders Aufmerksamkeit wurde dem 3D Zeichenmodus gewidmet, welcher die 3D Form eines Objekts aus einer groben, ungenauen, und möglicherweise unvollständigen Skizze rekonstruiert. Der 3D Modus hat einen Geschwindigkeitsvorteil im Vergleich zu ähnlichen Ansätzen und bewahrt die Rauheit der ursprünglichen Skizze und ihre visuelle Darstellung.

Die neue Funktionalität der CDP wurde im Rahmen einer Benutzerstudie mit zehn ArchitekturstudentInnen getestet. Die Teilnehmer an der Studie wurden mit einer einfachen Entwurfsaufgabe beauftragt – ein städtisches Mischwohngebiet zu entwerfen. Die Untersuchungskategorien der Benutzerstudie schlossen die Häufigkeit und die Benutzungsdauer der vier Zeichenmodi ein, sowie das Vermögen der Teilnehmer, den Bezug zwischen beiden Anzeigen (Tisch und Touchscreen-Display) herzustellen.

Die Benutzerstudie zeigte, dass Architekten die CDP und den neu implementierten Präsentationsmodus als Ergänzungswerkzeugen im Entwurfsprozess gern verwenden würden. Dennoch würden sie kaum auf die etablierte Werkzeuge wie Stift und Papier verzichten. Die Benutzerstudie hat auch Nachfragen nach neuen Funktionalitäten und nach grösserer individueller Anpassungsmöglichkeit des Systems aufgedeckt, welches ein Gegenstand der zukünftigen Forschung sein wird.

Die Benutzerstudie zeigte, dass Architekten die CDP und den neu implementierten Präsentationsmodus als Ergänzungswerkzeugen im Entwurfsprozess gern verwenden würden. Dennoch würden sie kaum auf die etablierte Werkzeuge wie Stift und Papier verzichten. Die Benutzerstudie hat ergeben, dass sich Nutzer neue Funktionalitäten und grössere individuelle Anpassungsmöglichkeiten des Systems wünschen, welches ein Gegenstand der zukünftigen Forschung sein wird.

Contents

1. Introduction

The present work originates from a project on development of the Collaborative Design Platform (CDP), a large-scale interactive table top which main application field is conceptual design in urban architecture. The challenges of urban conceptual design are tackled in the CDP in order to improve the interaction between architects and computer in the early stages of design.

Consequently, various requirements and properties of the CDP apply to the work presented in this thesis. An overview of the conceptual background of the CDP and its requirements towards design and functionality will be given In this chapter; furthermore, the need for an extension of the CDP for the purposes of conceptual design in architecture as well as the approaches used for development of the CDP extension in this work will be presented.

1.1. The CDP background

1.1.1. Conceptual design in architecture

Conceptual design in urban architecture refers to the process of conducting an initial shape of a building(s), revising and improving existing concepts about the building(s), and exploring new and novel ideas. Conceptual design is considered an essential activity of the workflow of an architect [1]. Many theories exist about what designing actually is. Moreover, the designing process is unique for an architect, thus it is hard to give a general and universal definition of the process. But it seems easier to understand conceptual design by taking a look at the topics it addresses. Figure 1.1 presents some of these topics. In the early design stages of a building, architects are usually interested in its rough position, orientation, and dimensions. Cost and efficient utilization of energy as well as legal regulations of the local authorities and topological connection to surrounding buildings are also important factors.

Rittel et al. claim that architecture problems are usually open-ended [2]. The design of an architectural concept is a non-linear, iterative process that involves prototyping and decision-making. It aims to "devise a form for an object, without having that actual object in front of you." [3].

It is difficult to answer the question "What is design?" directly. It is much easier to discuss the alternative formulation "What is the archetypal activity of design?". The almost universal and most readily understandable answer to this question is: "sketching" [4].

In terms of architecture, sketching refers to more than drawing with a pen on a sheet of paper. "Sketches represent a draft or design idea: they are tentative, not fully thought-through [...] ideas, thoughts and visions that need further development and elaboration" [5]. A sketch represents a rough idea, which can be refined, remodeled, or discarded.

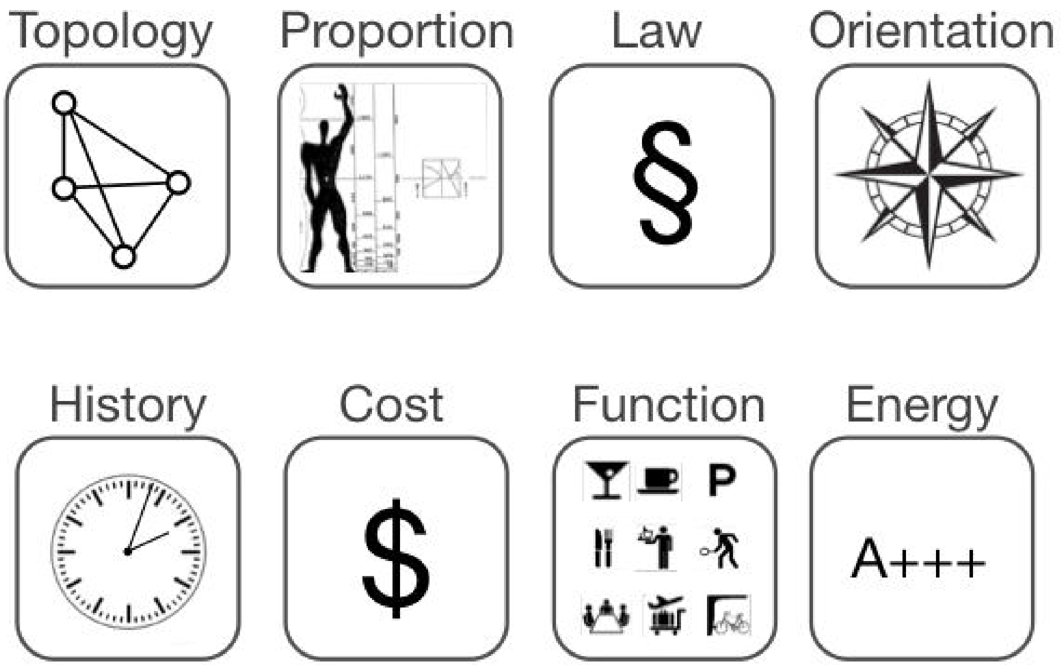


Figure 1.1.: Topics of conceptual design in architecture. (image property of Christoph Langenhan, unpublished)

Thus, precision is not an important quality of a sketch. In fact, it is the roughness and incompleteness of presentation which makes sketching such an attractive tool for designing.

Although sketching is convenient and fast, it does not allow for digital analysis, which often requires a 3D shape. It is known that high-level discoveries may induce changes in the basic shape of the model; for example, an architect could discover that his building occludes an important landmark, using occlusion simulation. This observation would be impossible to make by a hand-drawn sketch which lacks the surrounding buildings. But the observation would force the architect to change the very basic shape of his building.

In order to overcome the limitations of commonly "established" tools such as sketching on paper, architects have to repeatedly switch back and forth to digital tools. However, this transition is not fluent, as the two worlds have a completely different interface. Hence, switching between native, traditional tools, and Computer Aided Architectural Design (CAAD) tools, interrupts the flow of ideas and generates additional overhead. The goal of the CDP was to resolve this difficulty by fusing digital and traditional tools for the aim of prototyping.

1.1.2. The CDP

The CDP introduces a multi-touch table top which combines traditional design tools, such as sketching, with the computational power of the computer. To accomplish this, a new kind of interaction is introduced. The architect can use the tools of his choice to design a prototype, and then let the multi-touch table scan it. Once in the system, the model becomes a part of a mixed reality world and can undergo digital processing. The CDP was initially developed for the purpose of conceptual design in urban development, so the virtual world consists of a city model. Other fields of interest, such as interior design, or public transportation, might be also used in the future.

The CDP provides a real-time visual feedback directly on the table surface. It supports several types of simulation programs in the form of plug-ins, which can be extended to fit the users' needs. Some of these simulations include [1]:

- Light and shadow
- Access and distance
- Legal regulations

The simulations are performed in real time on the augmented urban environment, which consists of a 3D virtual city model and the scanned physical objects on the table surface. It is possible to combine different simulations together to keep track of multiple features of the physical model. The processing reacts to changes on the table surface. When objects are added or removed, the running simulation is updated to include the new contextual information.

This kind of human-computer interaction is seamless and intuitive. The designer does not have to recreate his model using digital input methods such as mouse and keyboard – it is sufficient to put the model on the table surface.

Furthermore, the large size of the table (1.45 x 0.95 meter) allows for better collaborative work. In this regard, collaboration will gain even more attention in the future development of the CDP, as different parties such as architects, employers, investors, or customers can be involved in conceptual designing.

The CDP attempts to combine digital and (traditional) analogous tools into a homogeneous, uninterrupted workflow. It needs a simple and self-revealing user interface, which requires as little attention from the designer as possible and allows him to concentrate on the actual task. The user interface should be intuitive, direct, and easy to learn. These requirements are consistently incorporated into the presentation mode newly introduced and presented in this work.

1.1.3. The presentation mode

Interactive table tops are not new in the field of human-computer interaction. There is a lot of research on this topic (see section 2.1.1), including also some commercial products in development [6, 7]. None of these products, however, has given the designer the complete freedom of designing like the CDP does. The CDP project inspired a whole lot of new ideas for applications for urban development, despite its prototypic stage of evolvement. Some of the applications suggested during the design of the CDP were:

A. Version branching

B. Tangible sun simulation

C. Orientation-correct user interface

D. Remote collaboration

E. Augmented visualization of legal specifications

F. Augmented visibility simulation

Figure 1.2 visualizes these ideas and gives a short description to each of them. For some of these applications (B, E, F, and possibly D), an additional opportunity for an interactive 3D visualization may be substantial. The multi-touch table surface itself could not enable this opportunity – its horizontal alignment suggests 2D visualization – and thus representing 3D information with the CDP would require a new alternative solution.

The benefits of a 3D view are undisputable, especially in the context of architecture. The information of interest for the CDP is often of three or more dimensions, including:

- 3D geometry
- Volume information
- 3D particle flow and/or distribution
- Energy distribution and/or propagation (4-D information)
- Spatial relationships such as distances, space, surface area etc.

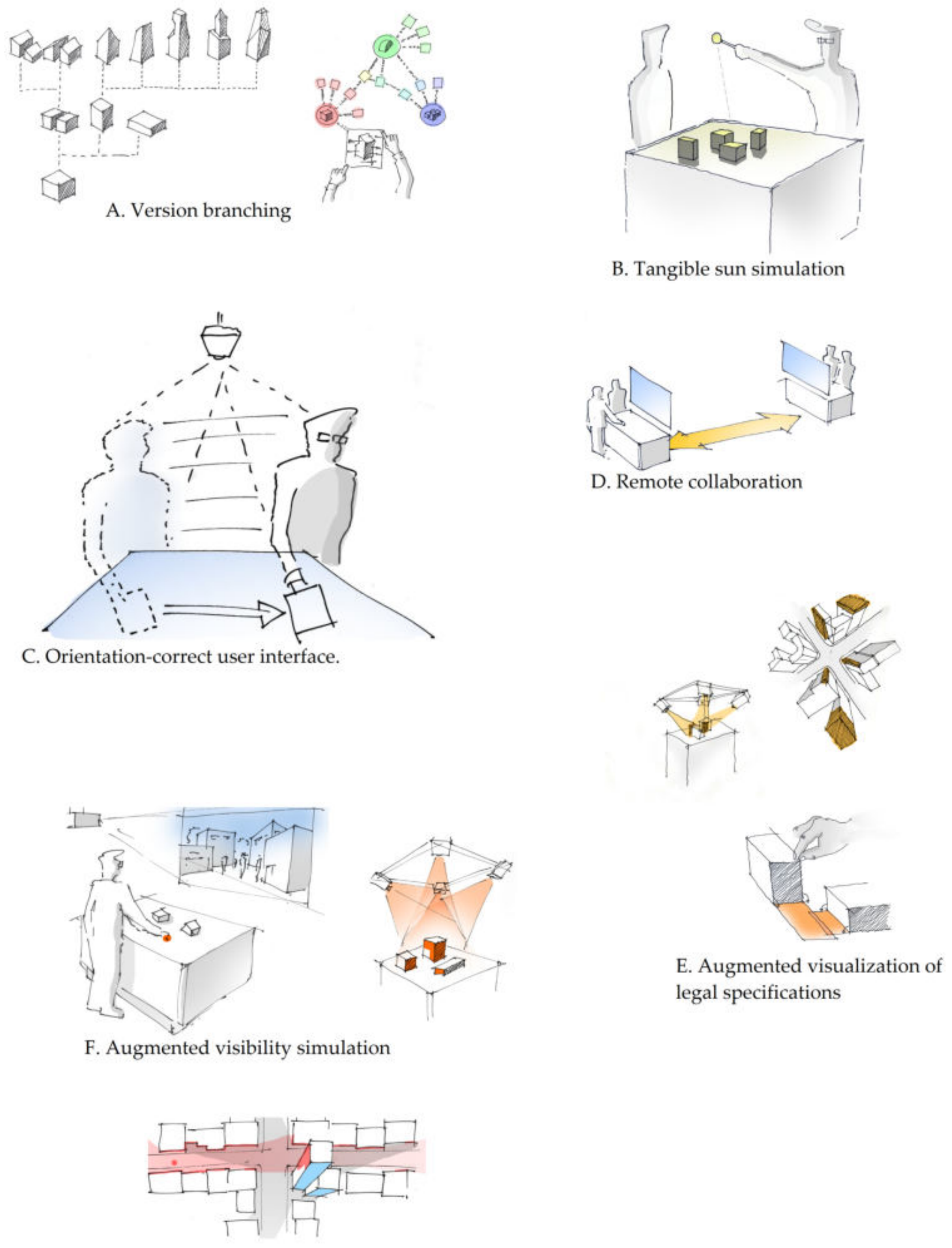

A. Version branching
B. Tangible sun simulation
C. Orientation-correct user interface.
D. Remote collaboration
E. Augmented visualization of legal specifications
F. Augmented visibility simulation

Figure 1.2.: CDP application ideas; (A) Storing and managing a version history of prototypes; (B) Simulating sun light using a tangible object, e.g. augmented reality marker; (C) User interface elements follow the designer and are correctly aligned; (D) Designing on multiple remote table tops; (E) Visualizing legal limitations, such as permitted height and shape; (F) Simulation of point-to-point or point-to-everywhere visibility. (property of Michael Mühlhaus, Nils Seifert, Violin Yanev, and Evi Andergassen-Sölva, unpublished)

The ability to display such information would substantially enhance the capabilities of the CDP for the purposes of conceptual design in architecture. Furthermore, being able to interact with the virtual 3D environment is an additional invaluable feature, which allows exchanging ideas in a familiar, but more powerful way.

The main aim of the present work was namely to develop an extension of the CDP which provides a smart implementation of the desired additional features mentioned above. The newly developed product of the current project was called and consistently referred as "the presentation mode" throughout the thesis. The term refers to the target usage of the new features, namely to present information to an audience, or as an assisting tool for the architect to visualize his thoughts.

The following two approaches were considered suitable for visualization of a 3D scene and enabling interaction with it:

A. Using one or several beamers to superimpose visual information horizontally on top of the table surface (as in figure 1.2 - F).

B. Using a large vertical display (or a beamer) to present a mixed reality 3D scene, containing physical objects on the table and virtual 3D data.

A direct comparison of these techniques is hardly possible due to their fundamental difference. The first one has a closer connection to the real world [1], but introduces some technical difficulties. For example, a reverse-mapping has to be performed on the beamer image in order to display the 3D information on top of the facades correctly.

For the purposes of the "presentation mode" developed in this work, the second approach was chosen: a 65" screen supporting touch input was used as a basis for the new working mode. While the table surface serves as a workbench for sketching and prototyping, the presentation mode provides possibilities for 3D visualization, discussion, and comparison between different versions of the urban model. The touch capability of the screen was also important for the main feature of the presentation mode developed here – the 3D sketching.

Sketching is a fundamental tool for conceptual design in architecture. In fact all of the established tools for conceptual design can be understood as a form of sketching [1]. In this sense, the automatic object recognition of the CDP is also a form of sketching, or more precisely, a way to accept and augment objects, or "sketches", as analogous input.

In this work, a specific approach to sketching – drawing on a digital touch-screen – has been explored. Artists often reject digital input devices with the argument of inconvenience, slow responsiveness, and lack of haptic feedback. In this case, a reasonable question is "Why should be a digital input used provided that the goal of the CDP is to retain analogous input (e.g. on paper)". Several methods exist to receive sketching input from analogous media. For example, in [8] a camera is used to scan sketches drawn on paper. The main reason to use a digital medium for sketching in CDP is the immediate response and processing provided by the computer. This is especially important when trying to assign a stereoscopic meaning to a sketch (i.e. to sketch in 3D). Computer drawings have further advantages – they are easier to edit, share, replicate, analyze, and provide convenient features such as "undo", "rotate", "zoom", etc.

[1]The data is displayed directly on top of the real objects

The presentation mode presented here explores possibilities to intuitively express 3D spatial ideas by sketching using computer assistance. The mixed reality world (containing the virtual model of a city and the physical objects scanned by the table) is displayed in a 3D view on the vertical screen. The architect can draw directly on the screen using his finger or a pen.

The following chapter 2 summarizes related work in the field of interactive table tops, as well as literature concerning sketching in conceptual design. Chapter 3 presents the schematic concept for development of the presentation mode, while chapter 4 describes its implementation. Chapter 5 presents and discusses the evaluation of the presentation mode and its results. Chapter 6 summarizes the results and proposes future work. Additional information about projective mathematics, which is used in this thesis, can be found in appendix A.

2. Related work

The development of the presentation mode outlined in section 1.1.3 was preceded by a literature search on two topics – Tangible User Interaction (TUI) and 3D sketching – to select the best practices used and avoid repeating existing research. A vast amount of literature is available on both topics. In the present chapter only reports related to conceptual design and architecture will be reviewed.

2.1. Tangible User Interaction

Tangible User Interaction is a general term describing user interfaces, in which digital information is manipulated using physical objects. TUI is a subject to research since more than 40 years. An early attempt to combine physical and digital interaction is the DigitalDESK [9]. This system lets the user write and sketch on real paper and uses a camera mounted on top of the desk to track gestures and drawings, and a beamer to project digital information back to the desk (figure 2.1). Despite the limitations of technology in the late 60's, DigitalDESK was able to recognize alphanumeric signs, scanned and copied drawings, and supported collaborative work on multiple remote desks.

Since then, TUI and augmented reality as a means for human-computer interaction have evolved substantially. However, several questions/problems remained open for researchers: for example, how can computers be utilized to assist creative thinking and prototyping? TUI is still the most feasible supplement to accomplish such a task. Three main categories of TUIs have been suggested: interactive surfaces, constructive assemblies, and token + constraint systems [10].

The CDP resembles a mixture of the first and the second TUI categories – it is an interactive surface which allows the designer to build his own model. The third TUI cathegory limits the interaction, allowing only certain combinations of the tokens. This behavior is not applicable in conceptual design where freedom of action plays a prominent role; hence, token + constraint TUIs are not revealed in detail here.

2.1.1. Interactive surfaces

The majority of TUI-based projects could be grouped in the category of interactive surfaces. These systems usually employ an augmented planar surface, where physical objects are electronically tracked, computationally interpreted, and graphically mediated [10]. Early examples of such projects can be found in the literature [11, 12, 13, 14, 15, 16]. The other way around, not every interactive surface uses tangible objects for interaction. Some platforms position the surface vertically and receive input from fingers or a pen only.

Interactive surfaces have gained higher research interest in the last decade. Platforms such as reacTIVision [7], Pictionaire [17] and Luminous table [18] extend the well-known

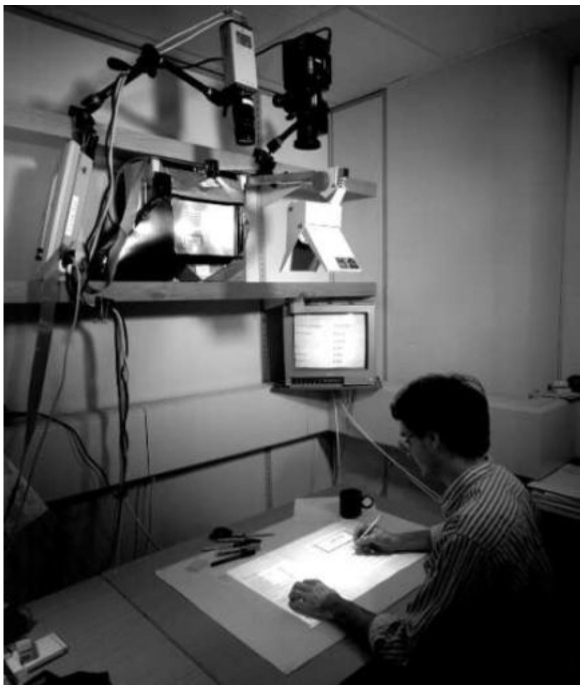


Figure 2.1.: DigitalDesk [9]

“multi-touch” concept beyond the simple finger-gesture interaction. Despite being a great technology, the “multi-touch” input and gestures are by far not the only form of human-computer interaction available on a luminous surface. The ReacTable, developed within the reacTIVision project, uses fiduciary markers in order to recognize the position and orientation of objects placed on the table (figure 2.2, left). Fiduciary markers are often used in mixed reality applications to mark physical objects, which are then embraced by the digital system. Usually, the markers resemble a black and white pattern, which is easy to recognize from a computer (similar to bar codes). The markers used in ReacTable (see figure 4.32) are optimized for robustness and consistency on a planar interaction surface [19, 20] and are used to mark tangible objects on the table surface. The sound synthesizer application developed for the ReacTable composes musical tracks by tracking these tangible objects and combining them in accordance with their topological ordering on the table. Moving one of the objects changes the topology, which results in a different audio melody. ReacTIVision is a great example how tangible objects can be used to support collaborative work and improve the creativity process.

Another project, Pictionaire, emphasizes on the fusion between digital and physical information in collaborative scenarios (figure 2.2, right). The platform distinguishes different users and their input devices (mice and keyboards) and gives them the opportunity to create, lookup on the internet, or scan imagery on the table surface. Digital data and the user interface are projected on the table top. A key feature of Pictionaire is that multiple users can “join” with their own input device and collaborate with other users (e.g. to type simultaneously in a document). A similar project was developed at the Upper Austria University of Applied Sciences [21]. Interesting about their platform is that it replaces

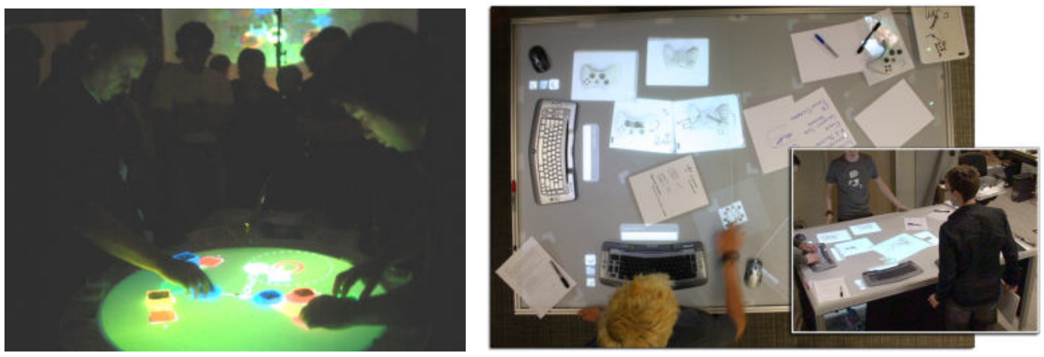


Figure 2.2.: Left: ReacTable and its sound synthesizer [7]; Right: Pictionaire [17]

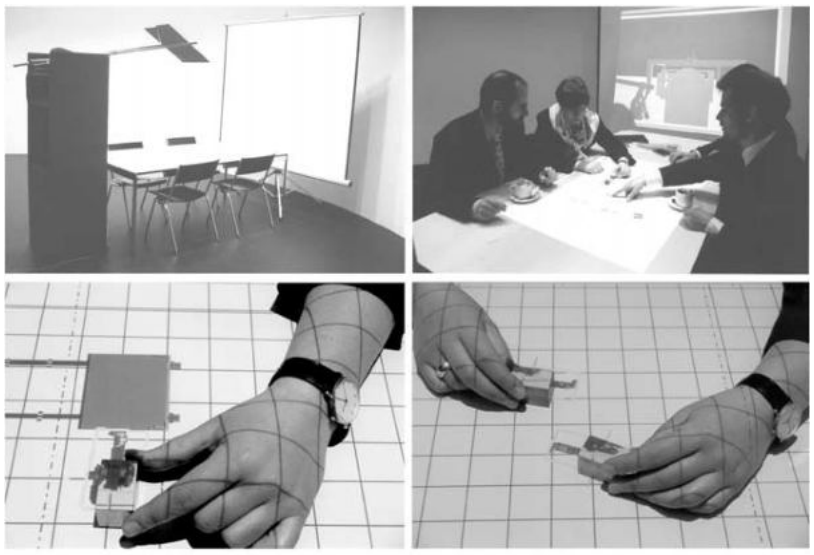


Figure 2.3.: The BUILD-IT system by Fjield et al.[22]

physical palettes with digital projections. Designers can change the color of their pen by tipping it in the projected palette.

An early attempt towards graspable interaction which is close to the CDP in terms of design and construction has been introduced by Fjeld et al. [22] and Sharlin et al. [23]. The BUILD-IT platform presented by Fjeld et al. features seamless 2D and 3D views of the design space. Graspable objects on the interactive surface correspond to pieces of information in the 3D virtual world (figure 2.3).

Ishii and Ullmer present another TUI-based platform called Metadesk [24]. They coin the terms "phicon" and "phandle" for physical interaction elements replacing conventional icons and handles, respectively (figure 2.4, left). Phicons are used to substitute digital information in the physical world, while phandles are used to interact with this information. In addition, the system provides augmented views of the surface through two types of lenses – "active" and "passive". The active lens provides an arm-mounted augmented view of the table top. The passive lens resembles a transparent frame, which is

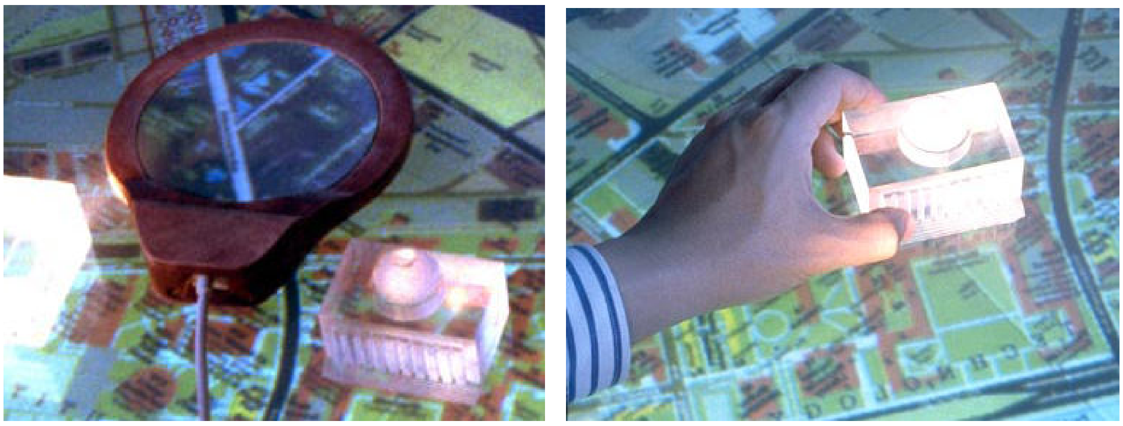


Figure 2.4.: MetaDESK [24]; Left: Great Dome phicon in tangible geospace; Right: a passive lens.

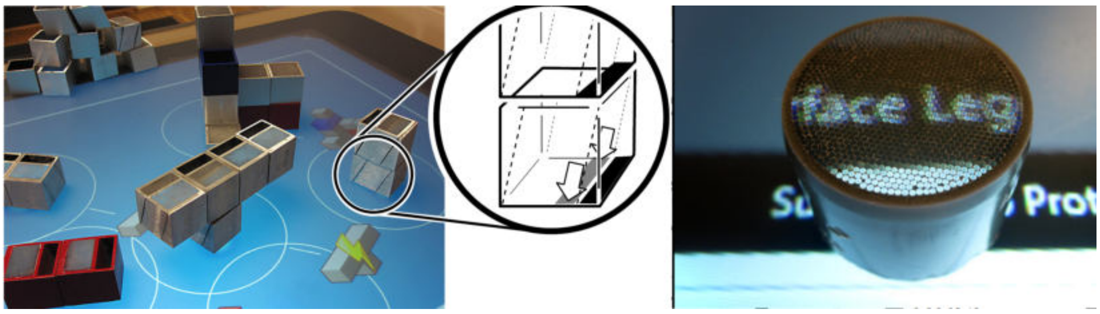


Figure 2.5.: Lumino and its building blocks filled with glass fibers [25]

tracked and augmented by the back-projected display of the table to "see-through" different layers of the map. The passive lens serves as a personal magnifying glass (figure 2.4, right).

Baudisch et al. illustrate another mixture of interactive tabletops and constructive assemblies (figure 2.5) [25]. The so-called "luminos" introduced in the paper are small building blocks with a filling of glass fibers. The fibers have different colors and form a traceable pattern. Moreover, the transparent fibers transmit light, enabling tracking several luminos stacked on top of each other. The platform can even recognize tunnel-like constructs and overhangs. Although the luminos can be combined arbitrarily, they don't offer complete freedom in modeling.

The Luminous planning table [18, 26] and its predecessor Urp [27] ("Urp" stays for "urban planning") pursue the same objectives like TUI – connection between physical modeling and digital analysis via TUI (figure 2.6, left). The Luminous table tracks objects based on a paradigm called I/O Bulbs [28] – entities which receive and emit visual information simultaneously, in contrast to normal light bulbs, which emit light only. A camera is used to capture the scene and present it on a large 3D view, projected by a beamer (figure 2.6, center and right). The Luminous table is capable of handling arbitrary models, but it requires the models to be constructed from wireframe materials. This limitation impedes the perception of mass and density. Furthermore, object geometry is not recognized by the

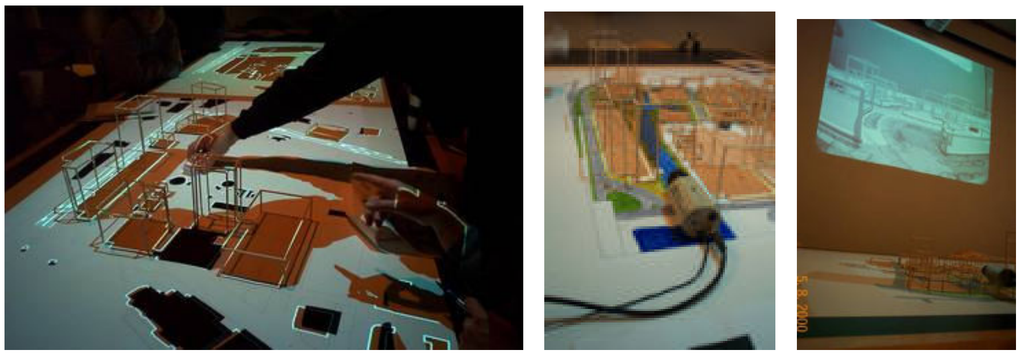


Figure 2.6.: The luminous planning table. Left: the interactive table surface; Center: a camera placed on the table; Right: a projection of the video stream of the camera. Images according to [26]

system, but is rather loaded from a "shape file" and linked to its physical counterpart via optical marker. This kind of interaction slows down the modeling process and impedes rapid prototyping. Despite their limitations, the two platforms received a wide interest from both academic and commercial sides.

Knecht from the Bauhaus-Universitaet Weimar developed an interactive table with a setup very similar to the CDP [29]. The table employs a depth camera (Kinect) to track arbitrary objects placed on its surface. The platform performs real-time simulation of light and uses fiduciary markers to control the daytime or time interval of the simulation.

Following the demand for multi-touch devices, several commercial solutions have recently emerged on the market. These platforms are not bound to a specific field of application, but are intended for more general use and thus provide their own Application Programming Interface (API). Among them is the Microsoft PixelSense [6]. The ReacTable-community has also given birth to two commercial products – the "ReacTable live!" and "ReacTable Experience" [30].

2.1.2. Constructive assemblies

TUIs based on constructive assemblies can be divided in two types – discrete and continuous. Discrete assemblies employ a set of building blocks to construct physical models, which are then digitalized. Continuous assemblies use a soft, deformable material to create models.

There are plenty of systems based on building blocks, which are chained together electrically or mechanically. Most of them are irrelevant for architectural applications due to their level of restriction. The most noteworthy one is Frazer's Universal Constructor described in his book "An evolutionary architecture" [31, 32]. Small electronic cubes are stacked and organized into cellular automata, which represent physical models of buildings as illustrated in figure 2.7).

The work of Schäfer, et al. embodies a concept called "real reality", where the user interacts with real objects using a special sensor glove [33]. The system tracks the gestures of the glove and carries them out in the virtual world. The system provides feedback in the form of acoustic signals, replacing classical GUI input/output paradigms. The system

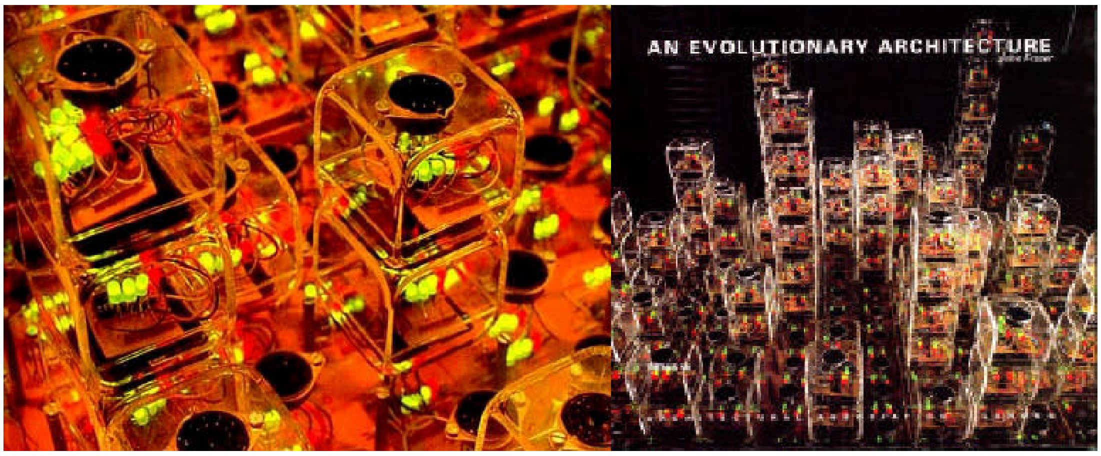


Figure 2.7.: Frazer's universal constructor (images taken from [31, 32])

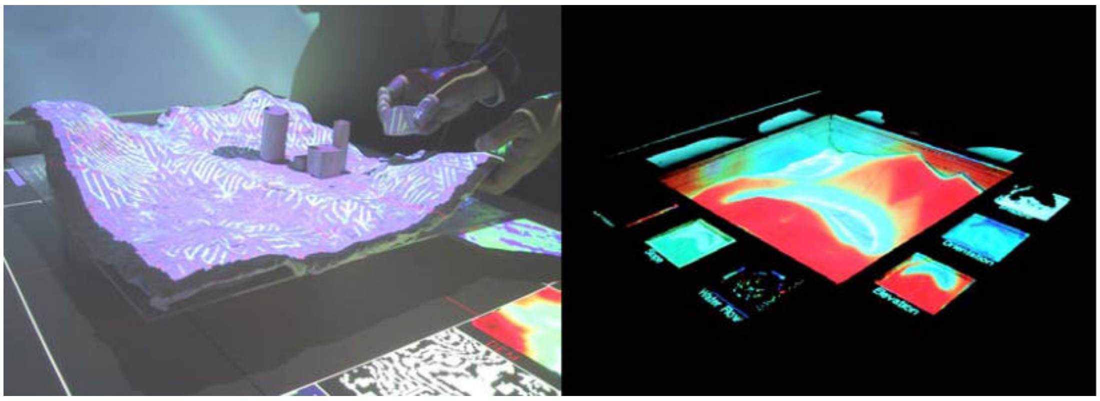


Figure 2.8.: Left: Illuminating clay [34]; Right: SandScape [35].

was exemplified in industrial scenarios, where construction lines were assembled.

A more appealing method, at least from architectural point of view, is part of the research of Ishii et al. at the MIT media lab, Massachusetts. They propose two continuous TUIs to model, augment and visualize terrain data – Illuminating Clay [34] and SandScape [35]. The Illuminating Clay uses a deformable material to model the terrain and a laser scanner to record its structure. The SandScape is a cheaper method, which uses glass beads and a source of infrared light under the platform. The height is measured by the amount of light passing through the pool of glass beads. Figure 2.8 depicts both systems.

2.2. 3D sketching

The sketching methods we considered applicable for the presentation mode developed in the present thesis needed to fulfill the following requirements:

- A 2D touch-sensitive surface should be used as input

- What You See Is What You Get (WYSIWYG) – the sketching input should not be beatified or modified significantly
- Digital sketching should behave like sketching on paper (e.g. no line straightening)
- As little additional user interaction as possible, ideally only the sketch strokes
- Sketching directly on top of the augmented virtual world
- Ability to visualize 3D concepts

Thus, the literature search on 3D sketching and geometrical reconstruction was performed with an emphasis on research topics conforming to the requirements listed above. The following sections give an overview on, the taxonomy of existing sketching methods and provide details on regularity-based reconstruction methods, which are considered to be the most suitable methods for the requirements of the CDP.

2.2.1. Taxonomy of methods for 3D sketching

3D sketching can be generally described as the act of expressing 3D ideas in an informal, imprecise way. In the present thesis, the term 3D sketching refers to the process of drawing stereoscopic objects on a planar medium. Certainly, real 3D input devices for sketching are also evident [36, 37, 38, 39], but have several limitations, such as:

- High complexity of the equipment
- Require training, or adaptation to their use
- Often are not perceived as natural and intuitive

These limitations dissociate methods using 3D input devices from classical sketching and make them less suited for rapid prototyping. Hence, there is an almost unanimous agreement among scientists that the term 3D sketching should be interpreted as "understanding 3D information from planar sketches". Only research dedicated to this paradigm was therefore considered for the review of related literature in this thesis.

Describing the geometry of 3D objects on a 2D surface has been of interest for scientists for more than 2000 years [40]. The reverse process – deriving 3D information from a 2D drawing – is considerably more difficult. This process is usually called "geometrical reconstruction" as it comprises the assembly of intended 3D shape by interpreting a respective 2D drawing. Diverse methods for geometrical reconstruction have been reported over the last 50 years. Their applicability depends to a great extent on the complexity of sketched objects and the amount of required user intervention. Three main approaches to 3D modeling by sketching are presently well established [40]:

- Gestural, comprising systems which provide predefined gesture alphabets that encode some geometric modeling operations; basically these systems substitute the selection of icons and menus by graphic gestures.
- Reconstructional, comprising systems which apply geometric reconstruction techniques to build the object's geometry from a sketch.

- Hybrid, comprising systems that combine the aforementioned two approaches.

The reconstructional systems were mainly considered for the development of the presentation mode described in this thesis. Gesture interaction requires the designer to keep in mind the gesture alphabet of the software. Such complications distract the designer from his primary intent, namely the design itself.

Sketch reconstruction methods can be further categorized into "single-view" and "multi-view", depending on how many different views of the object are captured. There is a clear distinction between these two methods. The single-view method concentrates on interpreting the sketched object and constructing a psychologically plausible approximation of its real shape. Multi-view methods, on the other hand, aim for precise reconstruction based on correspondence of points in different views, which is frequently the goal in engineering blueprints. Informally, a single-view reconstruction is closer to prototyping and conceptual design than a multi-view. This thesis implements only on single-view methods for sketching. The CDP targets early stages of conceptual design in architecture where the shape of the designed object is not known yet. In order to sketch two or more views of an object, the designer has to know its shape aforehead.

Single-view reconstructions can be classified according to:

- Method of reconstruction
- Support for curved surfaces or polytopes only [footnote: shapes consisting only of planar polygons]
- Drawing of hidden lines or only visible lines
- One solution or several solutions are produced

There are different reconstruction methods organized into six categories according to Company et al. [40]:

- Labeling (assigning a semantic label to each entity in the drawing – "outer edge", "orthogonal corner", etc.)
- Gradient space (based on general coherence rules that the orientations of the surfaces and edges must satisfy)
- Linear programming (solving a linear system to find the optimal solution)
- Progressive (performing the reconstruction continuously, after every few strokes or a faces in the sketch)
- Primitive identification
- Regularities (detection of regularities in the image, such as parallel lines, symmetry, orthogonality of faces and corners, etc.)

Hybrid methods exist as well, and often produce more plausible results than the individual components they are built upon (an example is reported by Beom-Soo and Chang-Hun [41]).

The regularity-based reconstruction is considered to be the most plausible approach for interpretation of general sketches with included hidden lines. This type of reconstruction has been subject to extensive research over the past decades due to its practical feasibility and proximity to human psychology and cognition. In the next section, several regularity-based algorithms are briefly introduced and characterized.

2.2.2. Regularity-based reconstruction of 3D sketches

One of the most influential studies on the topic of 3D reconstruction from sketches is the Ph.D. thesis of Hod Lipson [42], which introduces geometrical regularities as fundamental source of information for the reconstruction. For instance, parallel lines in the image space are with high probability also parallel in 3D space. This assumption is based on a statistical analysis of both computer generated sketches and human drawings [42]. A typical regularity-based reconstruction method analyzes the sketch geometry and assembles a compliance function. The compliance function maps the set of z-coordinates of the sketch vertices to a so-called "compliance value", which corresponds to the "psychological" plausibility of the reconstructed shape. Hence, the global minimum of the compliance function corresponds to the set of z-values yielding the 3D shape expected by a human observer when looking at the sketch.

Geometrical regularities have become the de-facto-standard for reconstruction from sketches with hidden lines included. The various proposed algorithms differ with regard to:

- type of investigated regularities
- dominance of the regularities (by importance)
- construction of the compliance function
- optimization method (Gauss-Newton, conjugated gradients, steepest descent, etc.)
- approximation of the initial solution

Table 2.1 chronologically encloses six of the regularity-based reconstruction applications from the past decade. Only the most plausible for the requirements of this thesis applications were included. A more exhaustive list can be found in [40].

Kang et al. [46] refined the method proposed by Lipson et al., 1996 [42], by analyzing dominant axis alignment of strokes in the sketch. Most technical drawings contain mutually aligned lines (architecture drawings make no exception), so that it is possible to partition the lines in several groups based on their alignment. This partitioning enforces implicit constraints on the sketch, which speed up the reconstruction and improve the quality of the recognized 3D shape.

Oh and Kim introduce a progressive solver, which performs the reconstruction continuously after each drawn face [41]. By doing so, the algorithm achieves an almost run-time execution. A significant advantage of this approach is that it allows the user to change the viewing perspective while sketching.

Year	**Ref.**	**Authors**	**Reconstruction method**	**Curves**	**Hidden lines**	**One solution**
1996	[42]	Lipson/Shpitalni	Regularities	Yes	Yes	Yes
1997	[43]	Eggli	Regularities	Yes	Yes	Yes
2001	[44]	Varley/Martin	Labeling	No	No	Yes
2003	[41]	Oh/Kim	Regularities/Progressive	Yes	Yes	Yes
2003	[45]	Company et al.	Regularities	No	Yes	Yes
2004	[46]	Kang et al.	Regularities	No	Yes	Yes

Table 2.1.: Six sketch reconstruction methods suitable for drafting in architecture [40]

Eggli performs online interpretation directly on the input sketch and lets the user select the desired recognition method and its accuracy [43]. 3D shapes can be created by extrusion or interpolation between curves. The system further includes an automated constraint solver.

Company et al. refine the reconstruction by introducing the so-called “tentative models” [45]. Tentative models are a fast and simple inflation of the sketch vertices, which speeds up the optimization process. Moreover, their method increases the chance to obtain the global minimum of the optimization function instead of running into a local minimum.

An important topic in the context of digital sketching is the pre-processing and analysis of sketch strokes. A reliable reconstruction requires precise partitioning of the sketch into lines, arcs, and face circuits. Wolin et al. present a robust algorithm for segmentation of sketch strokes, which analyzes pen speed and curvature of stroke segments to qualify them as lines and arcs [47].

In general, regularity-based reconstruction is a computationally intensive process, because it ultimately solves a highly non-linear optimization problem of dimension $\mathbf{N}$, where $\mathbf{N}$ is the number of edges in the sketch. Proposed methods attempt to speed-up the computation by increasing the user control, making implicit assumptions about the sketch, or performing the reconstruction progressively during sketching. For example, the method proposed for development of the presentation mode in this thesis employs the contextual information provided by the sketching environment (the buildings and the plane of the underlying virtual city). However, one should consider that increasing the speed and precision of a reconstruction generally reduces its diversity, meaning that fewer object types can be reliably reconstructed.

3. Concept

In this chapter, the initial concept of the present project including desired features and functions of the presentation mode will be presented. As basis, the functionality of the CDP is described with some of the specific activities during conceptual design in its classical form. In addition, a typical use case of the original multi-touch table is presented as well as a use case describing the desired functionality of the presentation mode. Not all features described in this chapter were finally implemented in the presentation mode. For details on the implementation of the presentation mode, refer to chapter 4.

3.1. Case definition and setup

3.1.1. Activities of classical conceptual design in urban architecture

There is no generally approved way to design. It is a unique creative process which may be easier understood by observing activities and tools utilized while designing. Frequently employed techniques by architects at the TU Munich, without pretension of completeness, include:

- Sketching on a piece of transparent sketch paper
- Cutting simple models out of rigid styrodur foam
- Cutting and folding objects from cardboard materials
- Shaping figures from deformable materials, such as clay
- Using all kinds of tangible objects to visualize thoughts (e.g. an old ticket to represent a subway station)

Furthermore, architects reach out to digital tools to perform advanced analysis on top of the conceptual model. But their use for rapid prototyping appears to be limited by complicated and cumbersome user interface of available CAAD tools. For this reason, digital analysis is rarely performed in early stages of conceptual design and is usually conducted at later stages when the design has reached a level of maturity which is worth the efforts to construct a precise digital model.

3.1.2. Use-case of the CDP

The CDP project has been developed to enhance conceptual design in urban architecture by incorporating digital analysis directly into the design flow. The CDP surface – a multi-touch table – displays a digital model of a city loaded from a database. The CDP surface operates on 3D data of the city objects, but displays a 2D view of the data, as if the user is

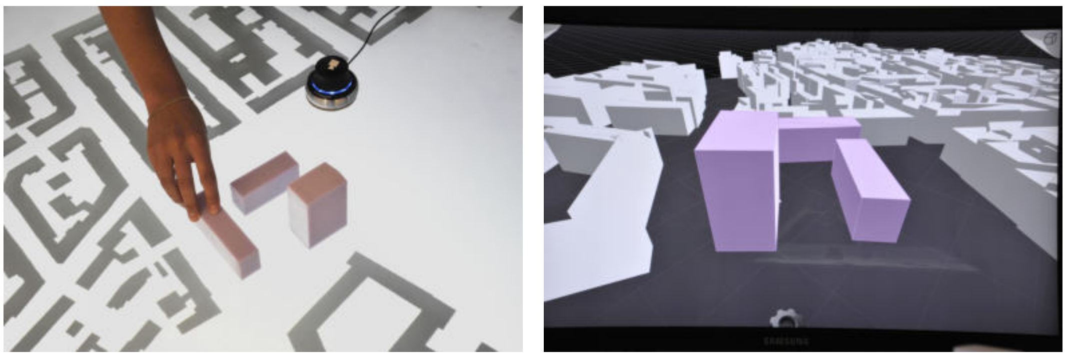


Figure 3.1.: Two views of the same world: the CDP surface (left) and the vertical presentation screen (right)

working with a map. The user can also scroll, rotate and zoom into and out of the map on the surface.

The designer can place his physical model (e.g. a block cut of styrodur rigid foam, or an arbitrary object) on the CDP surface where he wants to position his new building. The CDP scans its 3D shape and position on the surface (cf. figure 4.2b). Using this data, numerical simulations can be performed in real time. Simulations take into account the virtual buildings loaded from the database as well as the physical model(s) placed on the CDP surface. The CDP can perform various simulations, such as distribution of sun light, spatial relations between buildings, visibility, etc.

An additional feature initially included in the concept was that the designer should be able to paint on the multi-touch table using a special pen or to interact with the virtual objects and the simulation using touch gestures and tangible objects (e.g. remove a building, or change the direction of the sun). However, this feature was not implemented in the presentation mode presented in this thesis (see section 4.6).

3.1.3. Use-case of the presentation mode

The presentation mode extends the functionality of the CDP surface by an additional screen (here called presentation screen) placed vertically in proximity to the multi-touch table. Furthermore, it offers a 3D view of the scene shown in 2D on the multi-touch table. Physical objects are also displayed on the presentation screen, so that the viewer gets the impression that they are part of a virtual city (figure 3.1 right). The designer(s) can navigate a virtual camera in order to change the 3D view and explore the augmented city. They can sketch directly on the buildings in the presentation screen in order to share conceptual ideas visually. The presentation screen is linked to the CDP surface, which means that moving a physical object on the table would cause a respective change of the object on the presentation screen; also, sketches drawn by users on the presentation screen would be displayed back on the CDP surface.

The presentation mode transforms the mixed reality of the multi-touch table (i.e. the 2D city map and the 3D model placed on the table top) into a full virtual reality setting. The architect can navigate in the virtual world and sketch ideas directly on the objects contained there. He can also customize the presentation mode to fit his personal preferences.

In order to realize this use-case, the concept of the presentation mode should implement means for 3D visualization, space navigation, user interaction, and sketching. The concept for each of these features is described in the following sections 3.2, 3.3, 3.4, and 3.5.

3.2. Visualization

Given that the presentation mode provides the 3D view on the scene using a large-scale vertical display, essentially it works like a virtual camera view, as commonly seen in CAAD programs. The presentation screen represents a single view to the scene and is controlled by a tangible object, representing a camera (the camera concept is explained in detail in section 3.3). It is possible to partition the screen in multiple views, each corresponding to an instance of a camera. However, this would reduce the usability of the system to novice users (with no experience with CAD software) and impair their ability to recognize the correspondence between the 3D and 2D views. The CDP is designed to provide one consistent view for multiple users, not multiple views for one user. Moreover, the presentation mode employs a tangible camera for navigation, which is much easier to use than traditional Windows Icons Menus Pointer (WIMP) camera controls.

Nevertheless, it is not excluded that future refinements may add multi-view functionality to the presentation mode. At present, one view is sufficient to explore the applicability of the additional CDP display. Having one view is also more efficient in terms of space – objects are bigger and better to select and sketch on. It is also easier to work on a bigger view if there is more than one user working on the CDP.

One of the most important requirements, namely that the designer should be distracted as little as possible from the actual design process, is also relevant for the visualization of the city. The presentation mode has to be simple and clean. Having these considerations in mind, the following concept for the visualization was made:

- The 2D visual information, which is present on the table surface, should be also present in the 3D view mapped to the city terrain. This includes sketches, annotations, and the output of digital simulations (e.g. displaying shadows). Moreover, simulations can generate 3D information. In this case the display should also display this information.
- The background (the sky) should be very simple and light, but still feel real.
- The used colors should be neutral. The buildings should be emergent.
- The GUI elements (menus, buttons, palettes and similar) should be collapsed, unless they are needed and explicitly invoked by the user.

Not all ideas could be finally implemented in the presentation mode. For details in this regard, refer to section 4.6.

3.3. Camera control and behavior

The 3D view of the urban model is controlled by a single camera. The camera is controlled by a combination of a 3D mouse [48] and a fiduciary marker glued on its bottom (figure 3.1, left). The fiduciary marker is used to track the 2D position and orientation of the camera by the infrared cameras below the table surface. The 3D mouse controls the height and forward tilting of the camera.

In general, the 3D pose of a camera has 6 Degrees of Freedom (DOF) – three DOF for the position and three DOF for its orientation in space. In addition, the camera has a view frustum consisting of six planes – up, bottom, left, right, near and far. The view frustum is usually fixed unless advanced zooming is needed. Controlling all these parameters together can be cumbersome, especially for a user with little experience in CAAD. Hence, the design of the camera in the presentation mode has been simplified and constrained.

The camera model of the presentation mode is reduced to five DOF. Three DOF are used to describe the position of the camera, one DOF for its orientation in the horizontal plane (corresponds to a rotation around the UP-axis of the camera), and one DOF for tilting the camera up and down (corresponds to a rotation around the SIDE-axis of the camera) [1]. There are some further constraints on the camera movement – it cannot sink below the ground level and can be tilted from -90 to +90 degrees which corresponds to looking at the ground and at the sky, respectively.

These restrictions are imposed on the camera model to simulate human behavior. The camera moves exactly like a casual person, but able to fly. People rarely tilt their head to the side, or look at a building upside-down, so there is no need for the camera to do so. Furthermore, the camera is positioned by default at 1.7 meter above the ground, corresponding to the average human height. Sinking below the ground level is also useless, because none of the objects in the city is underground. Restricting the camera movement makes it easier to control and helps the architect feel as being inside the virtual world.

The reason why the 3D mouse is used to control only two DOF (and not all of them) is that the mouse sensor is extremely sensitive, so that only experienced user can operate with it with all DOF active. Even after dampening its sensitivity, the sensor still feels stiff and uncontrollable to a novice. Thus, it is necessary to lock out four DOF and use only two.

In addition to tracking the camera position, the fiduciary marker has one more function. It is used to switch the presentation mode on and off. As soon as the marker is placed on the table surface, the presentation mode is switched on and starts the 3D view. Likewise, if the marker is removed from the table surface, the presentation mode is shut down.

It is possible to use more than one camera, but this is not the primary goal of the presentation mode which aims to present and share ideas. Having more than one physical camera representation would disrupt the one-to-one relation between the tangible camera object and the 3D view, and would probably cause more confusion than it would help. Instead, several “views” can be stored and referenced back later using the so-called “sketch paper mode” (see section 3.5.3).

3.4. GUI and interaction

In order to interact with the system in presentation mode, a user interface is required. The CDP system provides two types of interface – graphical (on two multi-touch displays) and tangible (using the interactive table surface). For the purposes of the presentation

[1]Essentially, the camera can not “roll”, i.e. it cannot rotate around the “VIEW”-vector.

mode, a special GUI was developed. Both multi-touch displays are capable of receiving input and producing visual output; hence, both displays are suitable for GUI interaction. The advantages of the table surface in this regard are that it can track shapes, fiduciary markers and an unlimited number of fingers, which finally can provide rich user interface. The presentation screen is limited to only two fingers, but provides better precision at recognizing the fingers. Thus, the presentation screen is used to implement the GUI, as quality of interaction is more important than quantity at this stage of development of the CDP.

The GUI's main requirement is to draw as little attention as possible. Therefore, it is hidden and its elements can be "pulled" into the screen. Single touches are normally interpreted as sketch strokes and are projected directly on the object they collide with.

The GUI has one general-purpose menu to control common settings and customize the user interface. In addition, there are two more interaction elements which are used to activate two of the drawing modes – the sketch paper mode and the 3D mode (see section 3.5). In addition, the user can pull a palette from the top of the presentation screen to change the thickness and the color of the pen.

Thus, the options menu on the presentation screen is supposed to provide the following functions:

- adjust the camera sensitivity
- unlock the camera (so that all degrees of freedom of the 3D mouse can be used)
- switch between drawing modes
- activate the eraser tool

The eraser tool is used to wipe sketches from a surface.

The menu for the sketch paper mode looks, when closed, like a paper edge; when the user pulls on the edge, a semi-transparent paper scroll is unrolled over the entire screen which serves as a temporary sketching surface. The user can sketch in 2D over the 3D scene and discard the drawing or save it. It is possible to "jump back" to the drawing later, using the same perspective. Since this involves the camera being moved without the usage of the fiduciary marker, the virtual camera is used to display the position of the current view.

3.5. Sketching

Conceptual design is a creative process – a function of the personal experience, preferences and character of the designer. Architects joke that fifty architects would yield fifty different design approaches. A digital tool for sketching should be either extremely customizable (so that everyone can adjust it to his own personal needs) or highly flexible (so that it meets everyone's needs). Both alternatives are not technically achieved yet. Therefore, architects and other engineers still prefer pen and paper to advanced CAD tools.

In order to create an attractive tool for sketching in urban conceptual design, it was necessary to study how architects work individually and in groups. Important requirement for this study was that participants are constrained as little as possible. In order to maximize the learning effect, four different sketching modes were implemented in the presentation mode, including:

- Standard mode (figure 4.8a).
- Extended surface (figure 4.8b).
- Sketch paper (figure 4.17a).
- 3D mode (see figure 4.8c and d).

3.5.1. Standard mode

The standard mode is the least constrained one. It can be used to draw 2D drawings on top of the 3D objects in the scene. The standard mode works as if the designer would use a laser pointer to sketch his drawing in the virtual world. When the pen glides on the presentation screen, the system identifies the underlying object (i.e. the object on which the user is currently drawing) and performs perspective correction on the stroke to match the underlying object. It is not a real 3D process, but rather an extension to the common 2D drawing architects are used to in their daily work (figure 4.12).

There are several problems raising from this concept. First, if the designer accidentally draws a line over two adjacent surfaces, which are not coplanar, then the resulting line must not necessarily be a straight line in 3D (figure 4.13). This might be intended or unintended but the system has no way of recognizing the intent of the user.

Sometimes a sketch may be desired not directly on a building but “in the air” or close to a building. For example, if the intent is to draw an extension of an existing building, one would like to be able to draw on the extension of a wall in space (figure 4.8b). In this case, the concept proposed above is useless, as it will project the line somewhere else and not on the “invisible plane” of the extended wall. This problem is tackled by the second drawing mode – the extended surface mode.

3.5.2. Extended surface mode

The “extended surface mode” is activated by a click on a surface (e.g. a facade) while using the standard mode. In the extended mode, user drawings are mapped on the plane of the particularly selected surface. This way, the architect can extend buildings and objects easily, by selecting the facade wanted to be extended.

In order to enhance the visual perception and the feeling for space and orientation, the active plane is blended using a semi-transparent color and a regular grid with square cells (see figure 4.8b). A second click deactivates this mode and returns to the standard mode.

3.5.3. Sketch paper mode

The sketch paper mode is activated upon pulling on a paper edge at the top-left corner of the presentation screen, which unfolds a roll of semi-transparent paper. Any further drawing takes place on the paper, as if the architect is holding it in front of him. Sketches from the paper sheet can be “applied” to the image at any given moment, as if they were drawn using the standard mode. The advantage of this mode is that it can discard drawings easily by folding the paper roll, or by changing the sheet, just like real sketch paper.

Changing the sheet does not delete the previous sheet, but rather “saves” it in a special thumbnail bar, where all sketches drawn in the sketch paper mode are stored. From there, the user can switch from one sketch to another, discard sketches, or “apply” a sketch to the scene. Applying a sketch projects it to the underlying urban environment, as if it was drawn using the standard mode.

The user can draw everywhere on the presentation screen, not only on surfaces, although drawings which were not on a surface cannot be “applied” to the world later. The disadvantage of this mode is that the user cannot move the camera while using it. Every paper-sketch is bound to the particular camera position and orientation and by switching to that sketch the camera “flies” back to the position it was when the sketch was drawn. This behavior ensures that sketches remain related to the background on top of which they were drawn.

3.5.4. 3D sketching mode

The 3D sketching mode is considered the most advanced one of all developed in this thesis. It is activated similarly to the sketch paper mode – by pulling an edge of paper at the top-right corner of the screen. In contrast to the sketch paper mode, underlying objects appear with enhanced edges in the 3D mode. The edge enhancement serves as a hint that edges can be used as starting points for sketching. Linking sketch strokes to existing edges “connects” the sketch to its context. For example, drawing an extension of a building would bind the imaginary extended part to the existing one (figure 4.18).

The user has the freedom to sketch a 3D object as he would do it on a piece of paper. The advantage of the 3D mode is that the sketch, in contrast to the paper drawing, does not remain planar in shape, but is “inflated” to create a 3D structure.

After pressing a designated button, the system attempts to reconstruct the real 3D shape of the object and place it at the most plausible position in space. The resulting shape resembles a wireframe model, which projection coincides exactly with the drawn planar sketch when taken from the viewpoint it was drawn. The source sketch should not be resampled and no beautification of the final shape should be approached.

The reconstruction approach applied in this thesis has the following specifications:

- No support for curved lines and surfaces
- The camera cannot be displaced while sketching (off-line sketching)
- Imperfections in the sketch are tolerated
- Processes hidden lines
- A single solution is generated

In addition to sketching from edges of an existing object, the user can sketch a completely new object on the screen. In this case, the system assumes the object as “lying” on the virtual city plane. In urban design, this assumption is reasonable since new buildings reside on the ground or extend existing constructions.

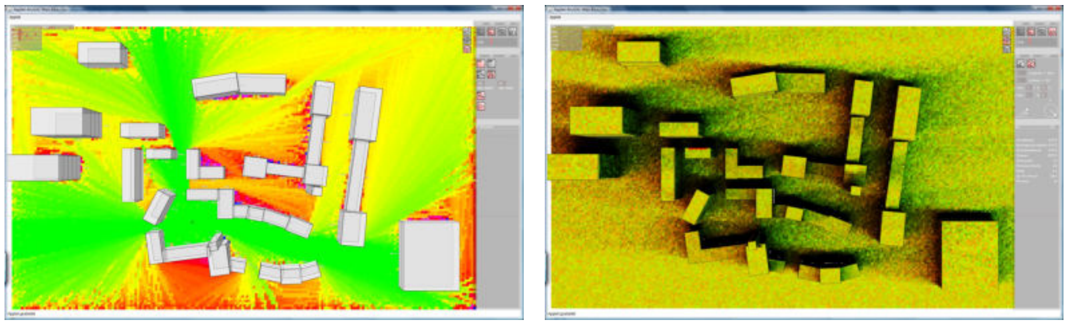


Figure 3.2.: Digital simulation within urban architecture. Left: visibility simulation; Right: simulation of sunlight (image property of Michael Mühlhaus and Niels Seifert, unpublished)

3.6. Integration and run-time behavior

This section describes two very important characteristics of the presentation mode – its integration into the CDP project and its behavior with respect to the highly dynamic and multi-user environment of the multi-touch table.

The multi-touch table is a dynamic system, which tracks the outline of any object placed on its surface and scans its 3D shape. The objects' position and orientation are synchronized with the presentation mode, resulting in a visual correspondence between the table and the presentation screen. Hence, moving a physical object (or the camera) on the table surface causes a respective movement of its virtual representation on the presentation screen. However, sketching on the presentation screen necessitates that all objects remain static.

The presentation mode developed here overcomes this difficulty by detaching the virtual world from the physical table for the duration of sketching activities. This mechanic ensures that sketches are done in a static scene and prevents buildings, on which the user is currently drawing, to be moved away or deleted. Changes in the physical world occurring during sketching are cached and applied after completion of the drawing. It is therefore possible that a building "slips" under the pen after the stroke is finished. The resulting sketch will be still applied to the moved building, unless the building was removed from the table surface.

Sketches are also attached to the building they have been drawn on. Thus moving a building will also move any sketches drawn to it, and deleting a building will delete its sketches too.

An additional feature discussed was that the user can sketch on the table surface directly using a special pen, which can be distinguished from a finger. The sketches done using this pen should be also displayed on the presentation screen. Similarly, sketches done in the presentation mode should be also displayed on the table-top, so far they can be visualized on a planar surface. This way the designer can choose which surface to use for his sketch, but the result will be visualized simultaneously on both displays. This holds only for sketches made on the city plane and not on the buildings, because the table can display only a top-view of the city plane on its 2D display. However, both ideas were finally not implemented (see section 4.6).

Another feature of the table is to execute numeric calculations on the city objects, such as sun light distribution, wind flow, or visibility analysis (figure 3.2). These simulations are usually computationally expensive and therefore must be calculated iteratively (they refine the result iteratively until the data does not change significantly). The 2D visual data provided by them is displayed on the table surface continuously. It was also desired that the presentation mode should display that data on its city plane whenever a simulation is running. It is also possible to extend the output of a simulation to generate 3D data and display it in the presentation mode. For example, to visualize how buildings throw shadows on each other, not only on the city plane.

4. Implementation

4.1. Big picture

A general overview of the CDP together with the presentation is illustrated in figure 4.1.

4.2. Used hardware and software

4.2.1. Hardware

The full list of hardware used originally for the CDP table top environment includes the following devices:

A Glass plate with projection screening.

B OPTOMA full high-definition beamer [49].

C A clear mirror with high reflectance.

D Four infrared projectors.

E Two Firefly MV2 infrared cameras, each with an infrared filter (which filters other wavelengths out).

F A computer with two firewire ports and two High-Definition Multimedia Interface (HDMI) ports (one for the table beamer, and one for the presentation screen).

G Microsoft Kinect [50]

The left side of figure 4.2a presents the arrangement of the above components. In turn, the presentation mode requires the following hardware:

H Samsung TS650-2 65″ touch display [51] (capable of recognizing two touches at a time)

F A powerful graphics card for parallel computing with Compute Unified Device Architecture (CUDA) – nVidia GeForce 580 GTX [52]

Z A 3Dconnexion 3D mouse [48] (not shown in the figure)

The hardware setup used in the CDP does not differ considerably from other multi-touch table top configurations. The technology used to track gestures is "diffuse illumination", or DI [53]. Four projectors are used to create a smoothly lighted environment under the table top.

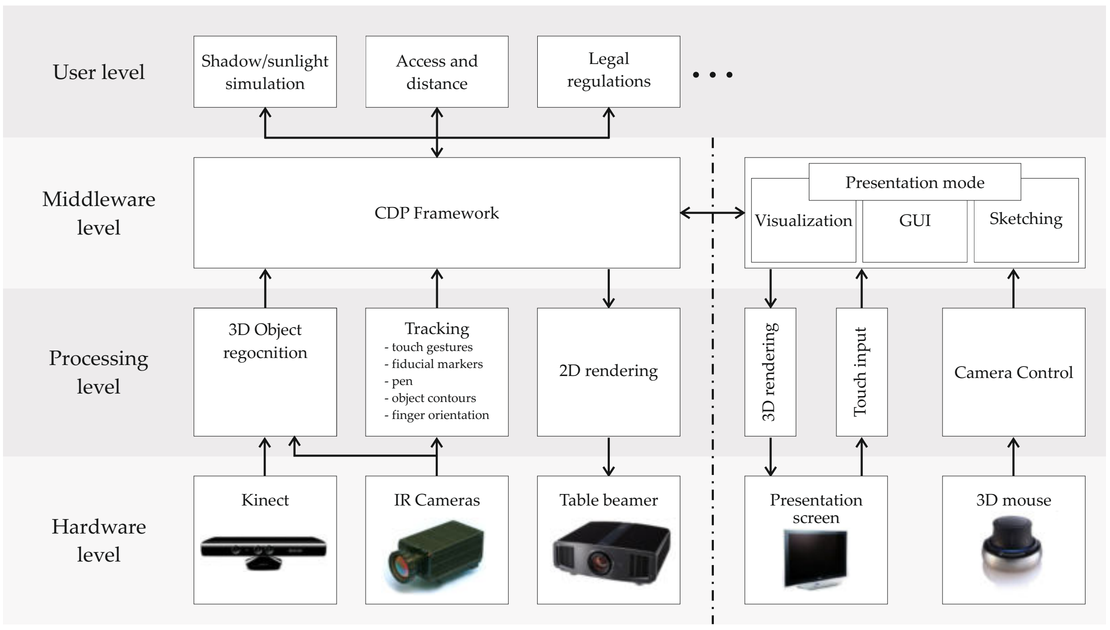


Figure 4.1.: Component overview of the CDP arranged in layers. The dashed line separates the components of the presentation mode from the rest of the system.

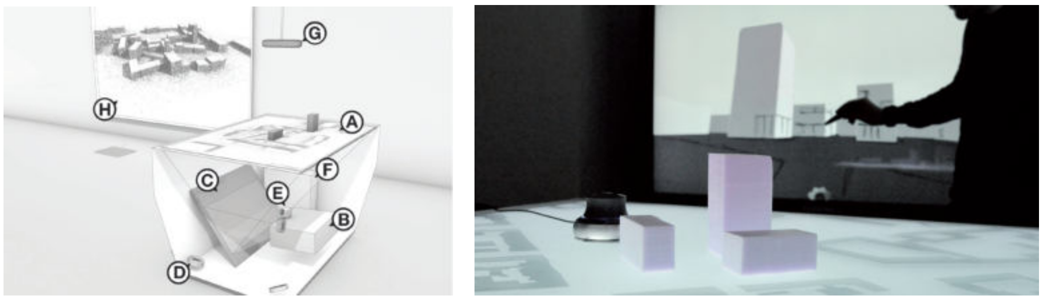


Figure 4.2.: The CDP and the presentation mode; (a) hardware setup (for explanation, see 4.2.1); (b) a photograph of the system.

The table's dimensions (145 x 95 cm) are rather big for using a single camera to capture the entire table surface, hence there is a need for two cameras. The images of the two monochrome cameras are stitched together and adjusted in a pre-configuration step. Each frame is processed with a set of filters in order to ensure smooth user interaction and lower noise and shadow effects.

The table can detects finger movements, touch gestures, fiduciary markers, and objects placed on the table. In addition, a special pen has recently been developed by Saburo Okita within a separate interdisciplinary project. The pen can be distinguished from fingers and used for sketching directly on the table surface. For more details regarding the implementation of the CDP table, please refer to [1].

4.2.2. Software

Several libraries, drivers, and software tools were included in the development of the CDP. They are listed here for the sake of completeness. The interested reader can follow the references for more information accordingly:

- OpenCV (for the image manipulations and tracking of object/gestures) [54]
- CUDA (to enhance the performance of OpenCV) [55]
- 1Dollar [56] – for the gesture recognition
- OpenTK (an OpenGL wrapper) for rendering on the table surface (with C#) [57]
- PointClouds library (for the manipulation of depth images, generated by Kinect) [58]
- OpenNI and the drivers for Kinect [59]
- Boost (for smart pointers, file operations, and others) [60]
- FlyCapture (the Software Development Kit (SDK) of the IR cameras)
- Drivers for the IR cameras
- Autocad MAP [61] and ORACLE (to manipulate and store geospatial data sets) [62]

The presentation mode requires additionally the following software:

- Visualization Library, an OpenGL based rendering framework (to render the 3D scene) [63]
- QT (to manage the GUI, component interaction, event handling, and more) [64]
- Assimp, a library for loading and saving 3D scenes [65]

The CDP project is developed in a 64-bit Microsoft Windows environment, using the Visual Studio 2010 64 bit compiler set and CMake [66]. The main reason for this choice is that architects are used to work on Windows computers and their skills in programming are mostly in .NET languages such as Visual Basic and C#. The given configuration (Windows + CMake + Visual Studio) seemed to be the best choice for these requirements.

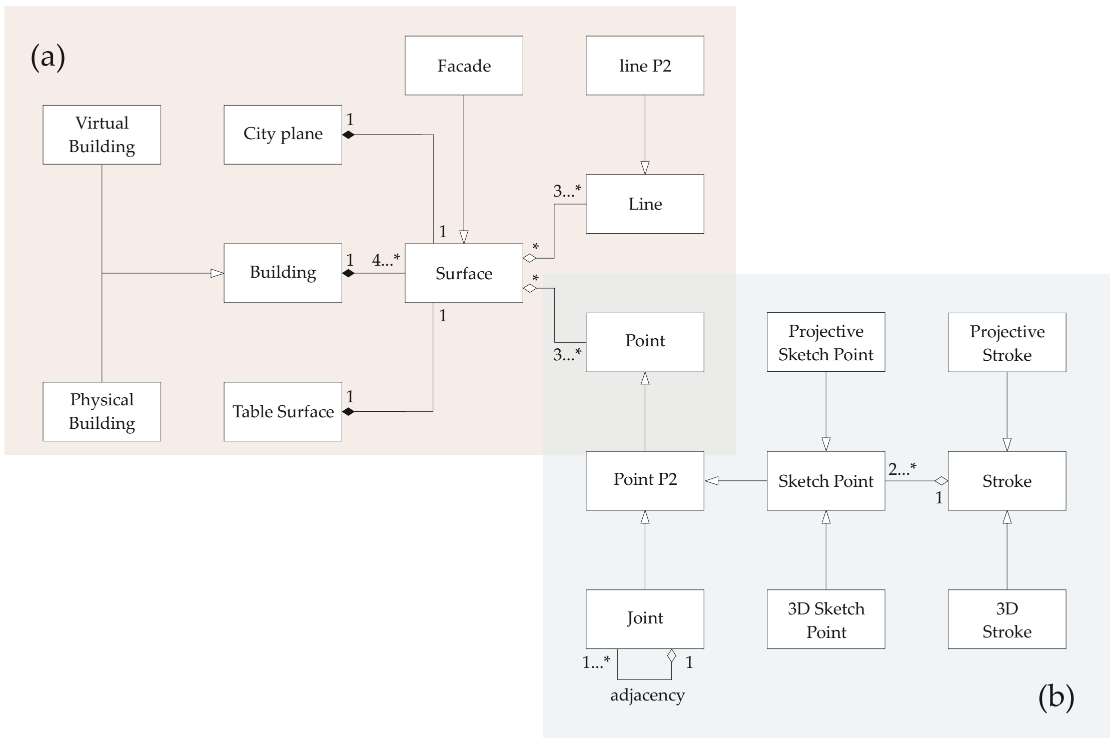


Figure 4.3.: An UML diagram explaining the data model of the presentation mode. (a) object classes; (b) fundamental geometry classes .

4.3. Datamodel

A consistent and unified data model is critical for data transport and processing. The requirements of the data model are as follows:

- minimum redundancy
- good structure in terms of hierarchy, semantics, aggregation, and generalization
- easily extensible through further data types and objects

These three principles were upheld when developing the data model concept (see figure 4.3). The data model classes can be divided mainly into two categories: scene objects and sketching geometry [footnote: in future refinements of the CDP, a third category may be included - simulation data]. Even though they have some common classes, these categories generally describe two distinct concepts.

Scene objects refer to parts of the physical or virtual world of the CDP. These include buildings (both physical and virtual), streets, subway stations, etc. Currently, there are four kinds of scene objects – virtual and physical buildings respectively, the infinitely large city plane, and the representation of the table surface. Figure 4.3a depicts the classes which describe these objects.

Sketching geometry includes points and lines generated by user input – 2D and 3D sketches, annotations, and labels. These data structures contain more than just the 2D or 3D coordinates of sketch points (see section 4.4.3). For example, they store the camera position at the time of sketching, an intermediate representation of the sketching geometry, and can contain other contextual information. Figure 4.3b presents the classes describing this kind of objects.

One of the major concerns of the data model is the metric system: which units (meters, centimeters, decimeters...) should be used to describe the scene geometry? How many of these units correspond to one internal unit in the presentation mode? This question gains even more importance when considering the following problems:

- Too small scales (1 unit = mm) cause numerical inaccuracies (see figure 4.36a)
- Too large scales (1 unit = 10m) cause artifacts in the depth buffer (see figure 4.36b)
- Metrics are also relevant for data interchange with other components, so all values should follow the same metric system.

The first two problems arise from technical limitations. Using small coordinate values cause numerical calculations to be imprecise and produce visible offsets. On the other hand, using large coordinate values increases the absolute distance between vertices, which essentially causes depth-fighting. The bigger the scale, the more visible are the depth artifacts. Hence, reducing one error increases the other one and vice versa. Therefore, a balance between these two errors had to be found. The technical background of these problems is explained in more detail in section 4.5.2.

The unit metric was empirically chosen so that both numerical and visual errors are minimized. The best results are achieved when using the following metric:

```
1 virtual unit = 1cm
```

This metric produces no visible numerical oscillation, and negligible z-bleeding artifacts. However, centimeters are too small units when talking about rough conceptual design in urban development where errors in the range of meters are common. Thus, meters were used to exchange data within the CDP framework, and centimeters were used internally in the presentation mode. Whenever data is received or transmitted by the presentation mode, an extra division/multiplication by 100 is needed in order to keep data representation consistent.

4.4. Software architecture of the presentation mode

4.4.1. Overview

Figure 4.4 presents the three main components of the presentation mode – “visualization”, “GUI”, and “sketching”. Each component is isolated from the others as much as possible, so that interfaces and data transfer remain consistent and cyclic dependencies are eliminated. The following sections present each component of the presentation mode in details.

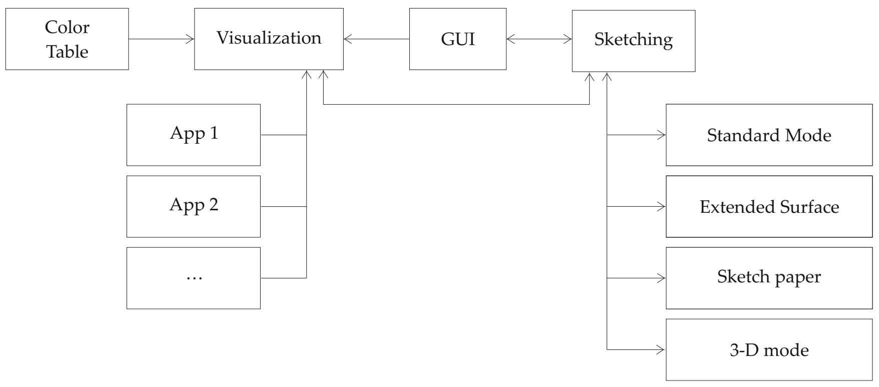


Figure 4.4.: Components of the visualization mode and data transfer between them.

4.4.2. Visualization

The CDP project is essentially a novel attempt at human-computer interaction, in particular the interaction between architects using a digital medium. This kind of interaction typically involves digital visualization – in this case, it is performed on two surfaces. Visualization is an important topic within the CDP and especially within the presentation mode. Visualization is particularly important with regard to the expected visual output, which is a large volumetric data set comprising an urban environment.

4.4.2.1. Review on existing visualization engines

There is a wide variety of methods used in modern software for data visualization and manipulation of geometric structures. Some of them are listed here, starting with the low-level frameworks and ending with the most-powerful and least performant ones:

- native graphics APIs such as OpenGL and DirectX
- managed wrappers and libraries such as OpenTK and SlimDX
- scene-graph and game libraries like Ogre, XNA, VTK, OpenInventor and Coin3D
- data visualization frameworks such as ParaView and OpenCASCADE

Many of these frameworks are optimized for a specific purpose, such as games in the case of Ogre and scientific data and vector fields in the case of ParaView. Generally, they represent a trade-off between performance and convenience. The more convenient and automatized a library becomes, the less performant it is and therefore offers a lower frame rate compared to other libraries for the same task.

In order to choose a library, three criteria should be considered:

- Magnitude and complexity of rendered data
- Effects and features needed or useful for the future

- How critical is performance

These criteria are hard to estimate at the very beginning of a project. Then again, it is very hard to change the visualization library at a later point in the development since it also offers the fundamental mathematical structures (vector and matrix formats and operations, geometry primitives, camera model etc.). These structures are used by other system components, such as sketching. Changing the visualization library would most probably require the entire project to be rebuilt from scratch. Therefore, the visualization engine should be chosen carefully still at the beginning of product development process.

The following subsections attempt to give an approximation of the three criteria given above, considering not only the current requirements but also future extensions of the CDP project. These subsections also attempt to give a justification for the choice of software tools for the presentation mode. The software tools themselves are covered in the subsequent sections.

What is the magnitude and complexity of the rendered data?

Both displays – the table surface and the presentation screen – are synchronized and basically display the same objects (buildings, streets, parks). However, the table presents the city from a 2D isometric orthogonal view, whereas the presentation mode has a potentially unconstrained 3D view from any point in space. Thus, the view frustum of the table is limited by the position and scale of the city map, while the presentation mode can possibly display the entire city with all its structures (see figure 3.1).

The upper bound of the objects which need to be displayed in the 3D view is therefore the entire city. However, the designer is mostly interested in his own conceptual design. The most common and frequently encountered working case is shown on figure 4.2, right and can be described as follows:

- The designed model is placed near the middle of the table
- The city map is centered around the model
- The camera is positioned close to the model, looking towards it
- The camera is either close to the sea level, or
- The camera is slightly above the average building height, slightly tilted downward

Taking the above information and the camera viewport width of 60 degree into account, it is reasonable to assume that about 25-50% of the city buildings are visible in the 3D view at a time. This assumption was confirmed during the user evaluation phase (chapter 5).

The 3D scene has to display a vast number of objects. In addition, sketching results also have to be displayed on top of the urban environment. In future releases of the presentation mode, even more data will have to be displayed such as subway stations, infrastructure, terrain data etc. Therefore, an important requirement to the visualization engine is performance.

The buildings themselves are displayed using simple polygons. As of the current implementation, all buildings have prismatic shape – an extruded polygon without holes. This will probably remain so in the future, since most urban data is simplified and optimized to save space.

Even though they are very vague and imprecise, the estimations given above served as a rule-of-thumb. In conclusion, the 3D view has to display a lot of information, which should be taken into account when choosing the visualization library.

Which effects and features are needed or might be useful in the future?

The most basic use case for the visualization module is to display the city as a collection of solid buildings within an urban environment. This is a very basic scenario and one can expect that the CDP framework will require more functionality in its future development. Still, the main purpose of CDP is to aid conceptual design, so the features offered by the visualization library do not have to exceed the requirements of conceptual design.

Figure 1.1 in chapter 1 presents the most important topics of conceptual design in architecture. A topic could have several aspects to consider; for example "proportion" may include dimensionality, distance, volume, etc. Furthermore, one aspect may be visualized in different ways. This explosion of features makes it difficult to anticipate all possible requirements of visualization in conceptual design in architecture. In practice, however, the following visual effects are often applied:

- Phong Lighting
- Gouroud shading
- Transparency and alpha blending
- Texturing (in order to display simulation data such as light distribution)
- Connection to CUDA/ComputeShader to speed up computation of simulated results
- Geometry manipulation and interpolation
- Text rendering and manipulation (in order to visualize numeric or textual data)

Some other useful features, which are not directly related to conceptual design, include:

- GUI creation/event management
- Image input/output and manipulation
- Scene management (spatial and semantic organization of objects)

The above features slightly exceed the capabilities of current native graphics APIs such as DirectX and OpenGL. Even though it is possible to use one of these in favor of performance, a more high-level library would be a better solution.

How critical is performance?

As discovered later in the implementation, performance is indeed an important factor for sketching quality. The presentation mode performs real-time perspective correction on the drawn sketches, which is a computationally intensive task. Therefore, the higher the frame rate of the visualization library, the more time remains to perform sketching calculations and thus the better the sketching quality is. Hence, having a visualization library which operates fast is beneficial.

Having the above considerations in mind, the final choice fell on the OpenGL-based "Visualization Library" [63].

4.4.2.2. Visualization Library features

The best quality of Visualization Library (VL) is that it has a rich set of features while maintaining a native profile with great performance. The complete set of features can be found at [67]. Table 4.1 summarizes the features important for the presentation mode development.

A very convenient feature of VL are the bindings to GUI frameworks, in particular the QT bindings (see section 4.4.2.4).

4.4.2.3. Alternative rendering libraries

There are alternatives to VL, which also fulfill the discussed requirements.

A more performant library is OpenGL (or the Windows alternative DirectX). The disadvantage of OpenGL is that more advanced features have to be implemented manually. One such feature is transparency, which needs correct polygon z-ordering in order to produce plausible results.

Scene-graph based renderers, such as Ogre, Coin3D, and OpenInventor, operate on a similar level as VL, but lack some of its more sophisticated features, such as volume rendering and terrain rendering. These features are commonly used in scientific applications, while scene-graph renderers are used primarily in game development. Therefore, this kind of renderers is not the optimal choice for the visualization of architectural content.

A very powerful visualization framework is OpenCASCADE. It can do everything that VL can do and more, but at a performance cost. Moreover, OpenCASCADE is a large and heavyweight framework, in contrast to VL, which is small, fast, and easy to learn. Therefore, VL was selected as a rendering framework.

4.4.2.4. QT

Even though the presentation mode primarily displays 3D data, a pure 3D visualization library is not sufficient for the requirement of the presentation mode, which also requires a graphical user interface. For instance, the "sketch paper" drawing mode requires a scrollbar with thumbnail images (see section 4.4.3.4). The thumbnails have buttons on their part and can be selected, moved, or deleted. This kind of interaction requires a true 2D GUI framework.

Feature	**Description**
Lightweight	Low-level, fast, fine-grained C++ middleware on top of OpenGL.
Simplification	Size of the library is kept to a minimum and only the most efficient rendering paths are used.
Scene-graph support	Topological organization of the world into scene graphs is supported. Scene-graphs are lightweight and easy to maintain.
GUI bindings	Bindings to popular GUI frameworks, such as QT, MFC, GLUT, etc.
High performance	Very close to OpenGL; buffer optimization; render state sorting; occlusion culling; hierarchical frustum culling; Level-of-Detail geometry; (...)
Transparency management	Automatic ordering of polygons for correct alpha blending results.
Image I/O	Loading and saving images and textures, automatic mip-map generation, palette management.
Terrain rendering	Rendering of textured terrain data.
Automatic memory management	Smart pointers, which make memory management easier, without decreasing performance too much.
Volume rendering	Rendering of volumetric data, such as the inside of a building, or heat distribution.
Geometry manipulation	Simple primitives, tessellation of polygons, generating geometry by extrusion, automatic calculation of normal vectors, and much more.

Table 4.1.: Features of VL important for the presentation mode.

Furthermore, future extensions of CDP, and in particular the presentation mode, will probably require more advanced 2D drawing techniques such as brushes, gradients, different kind of pens, palettes, etc. VL is capable of manipulating 2D images, but this feature is not a main application field of VL and is therefore highly limited.

Fortunately, VL has bindings to many GUI frameworks. The one which was used in the presentation mode is QT [64]. QT is a large framework and is not limited only to 2D visualization and GUI creation. Some of the QT features used in the presentation mode include:

- GUI management
- Threading
- Synchronization (mutexes, locks and semaphores)

QT also uses a very convenient communication method called "signals and slots". The latter enable asynchronous event-handling in the communication between multiple components. This is a very useful feature as the CDP framework consists of several modules which have to communicate and exchange data very frequently and asynchronously. Using the QT slots and signals, one does not need to care about synchronization and event triggering/intercepting.

VL uses QT to create the OpenGL rendering context (see section 4.5.1 for more detail concerning context creation), which is not a trivial task, and also to manage the windows and the event chain. This way the presentation mode could almost abstain from communication with the operating system, as this is taken care of by QT. The only exception is the communication with the 3D mouse used to control the camera, as it is not automatically handled by the Windows event queue.

Another benefit of QT is that it is open source and platform independent. The latter is not a requirement for the CDP project, as it is designed to run specifically on a 64-bit Microsoft Windows machine. But platform independence is an advantage in case of sharing the framework with a third party, exporting the presentation mode to another project, or publishing it as an open-source project.

4.4.2.5. Implementation

The visualization module is the foundation for everything else, as it also defines the data structures used to represent the various objects in the scene. The visualization module accomplishes the following tasks:

- Geometry and transform management
- Shader organization and effect switching
- Lighting
- Emulation of sky

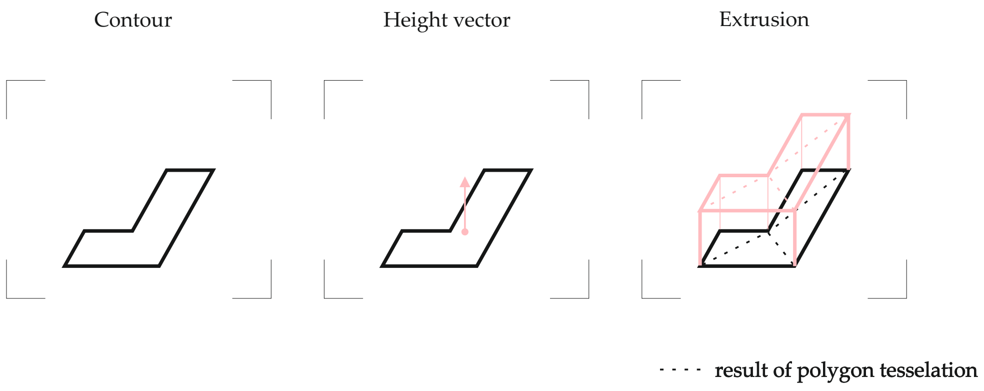


Figure 4.5.: Polygon tessellation and extrusion.

Geometry management comprises the conversion of geometrical data into the OpenGL display format, consisting of vertex arrays, index buffers, draw calls and/or display lists. Geometry input may come from a file, a database, or from the program itself. It can be also a physical object scanned on the table surface.

The current implementation renders the following types of geometry:

- Virtual buildings (coming from a database, a file, or an online service)
- Physical buildings (tracked on the table surface)
- The city plane
- Sketches which the user draws

The visualization module has to load (or generate) the geometry of these objects. This is not always a trivial task. For example, the object recognition of CDP (at the time of writing) scans only the bottom contour of an object, then measures its height. Thus, an object is composed out of its base contour and a height value. This representation is sufficient for numerical computations and analysis, but in order to render an object, the exact mesh is required. In order to create a volumetric object, the base contour is lifted in the "up"-direction with magnitude equal to the height. This process is known as "extrusion" (see figure 4.5).

Extrusion, as well as many other tools for managing geometry, is a part of VL. VL can tessellate complex polygons into lists of triangles, simplify meshes, interpolate shapes, and even combine complex polygons and remove holes. Although these features are currently not utilized in the presentation mode, they will certainly be needed once the object recognition module delivers more precise meshes of the physical models.

The city plane is displayed as a large regular grid, which creates the illusion of infinity. The grid cells are supposed to assist the perception of spatial dependencies, such as distance between buildings and space quadrature. The physical table surface is represented in the virtual world as a rectangle on top of the city plane. This way the designer can have

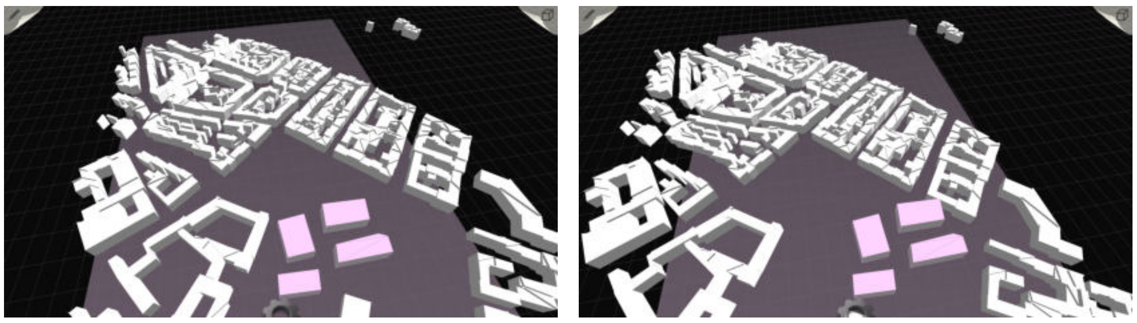


Figure 4.6.: Navigation of the table top view and its effect on the presentation screen. The light quadric area corresponds to the table top; the four pink buildings correspond to physical objects on the table top. Left: the city before moving the table; Right: the city after moving the table.

a feeling of what part of the city is also visible on the table and how the physical models relate to the virtual context. The designer can also navigate the view of the table top, which changes the content of the presentation screen too as it is directly related to the table top (see figure 4.6).

Sketches are displayed as line strips (a set of interconnected line segments). See section 4.4.3 for more details about sketching and the computation of sketch geometry.

No terrain data, such as streets and subway stations, and no semantic data are presently displayed on the table. The urban environment is loaded from a simple COLLADA file [1], which contains only building geometry for the purpose of testing. Loading more complex and detailed data is the subject of a future project.

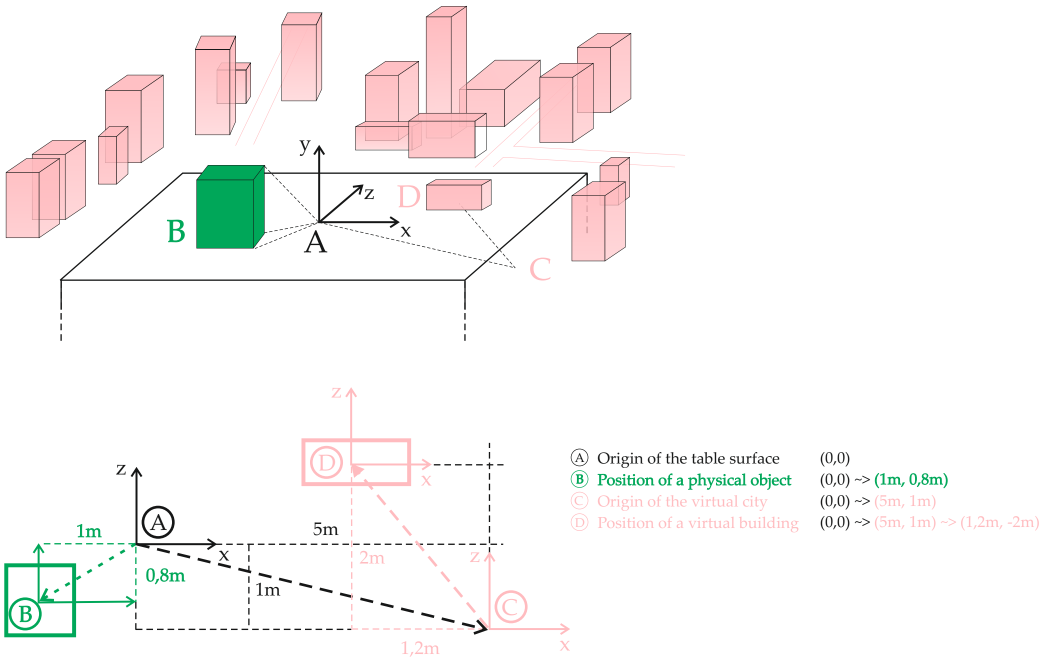
Transform management refers to the process of assigning a proper transformation matrix to each object. Objects are usually positioned relatively to some other object or to the world coordinate system. This kind of relation between objects yields a so-called scene-graph in which the root node is the world coordinate system. The state of an object (position, material, geometry...) is defined by its position in the graph.

A good implementation of a scene graph should try to assign as few matrices as possible and to cache them whenever possible in order to save time. Objects which are completely static should have only one matrix, which positions them in world space. An even better solution is to directly transform the geometry of static objects as soon as they are loaded. Frequently moving objects should have a dynamic matrix which is actualized before each frame. Objects which move occasionally should have their matrix cached and update it only upon movement. Having a well optimized transform tree greatly enhances the overall performance.

VL is better than common scene graph libraries in that sense that it does not update the entire transform tree automatically and gives a lot of freedom to the developer when it comes to customization of the rendering process.

Having a good transform management is important mainly for the sketching. The sketching module captures the position of the pen continuously and computes its inter-

[1]Collaborative Design Activity (COLLADA) is an open format for storing 3D geometry and physics data. It is designed for exchange between content providers, such as CAAD software.



A	Origin of the table surface	(0,0)
B	Position of a physical object	(0,0) ~> (1m, 0,8m)
C	Origin of the virtual city	(0,0) ~> (5m, 1m)
D	Position of a virtual building	(0,0) ~> (5m, 1m) ~> (1,2m, -2m)

Figure 4.7.: Transform distribution within the virtual world. The following color codes are used: black: the coordinate system of the virtual table (the root coordinate system); green: the physical objects and their transforms; red: virtual objects and their transforms. Upper: the coordinate system of the table surface; Lower: a close-up, showing the connection between the physical and the virtual coordinate systems. A, B, C, and D: examples of transformation chains for various objects; only 2D translation is considered here for simplicity.

section with the city objects simultaneously. To do so, the sketching needs access to the exact vertex positions of scene objects as well as the lines and faces connecting the vertices. This naturally involves computing the transformation matrix for each point and for each face normal. In case of a large transform chain (e.g. a point is attached to three matrices) this would considerably slow down the computation of intersection, resulting in a worse quality of the drawn sketches due to fewer computed intersections.

In order to minimize the amount of transforms for scene objects, the following transform distribution was implemented (see figure 4.7):

- **The table surface** defines the world coordinate system. The middle of the table is the origin (0, 0, 0), the direction of the long table edge is the 'x'-axis and the shorter edge corresponds to the 'z'-axis, respectively. The 'up'-axis is the positive 'y'-axis.
- **Physical objects** (the ones scanned by the table) are placed relatively to the table surface and their transform is cached (objects on the table are not supposed to move constantly, so this is a reasonable optimization)
- **Virtual objects** (the city model loaded from an external source) are also placed relatively to the table surface. Their transform is also cached, since the table is not supposed to be permanently navigated.
- **The camera** is positioned relatively to the table surface, similarly to the positioning of physical objects.

This transform distribution is not the most intuitive one: one would expect that the table surface is defined relatively to the city model, as it is the table surface which "moves around the city" and not the other way around. However, this kind of positioning would mean that physical objects on the table have one additional transform and would also need to be updated every time the table is navigated.

Shader management refers to the process of assigning effects to rendered objects (e.g. color, transparency, light, etc.). The presentation mode organizes visual data into "apps", each app having its own set of effects. This allows for quick switching between different visualization modes without having to recreate the entire scene. The most basic one is the solid app, which displays buildings as solid blocks lit by the sun and the city plane as a grid.

A *skybox* is also shown at the background. A skybox is a simulation of sky, which uses the shape of a box and maps a texture of celestial bodies to it. The skybox is always rendered first and moves together with the camera, creating the illusion that the scene is surrounded by a sky. The skybox of the presentation mode is very bright and simplified in order to avoid distracting the designer with effects. Still, the scene looks much deeper with a sky than with plain color.

4.4.3. Sketching

The most-important component of the presentation mode is sketching. It transforms the 3D viewer into interactive whiteboard for 3D drawing. As described in section 3.5, the purpose is to simulate drawing in 3D, thus enabling sharing of stereoscopic ideas without

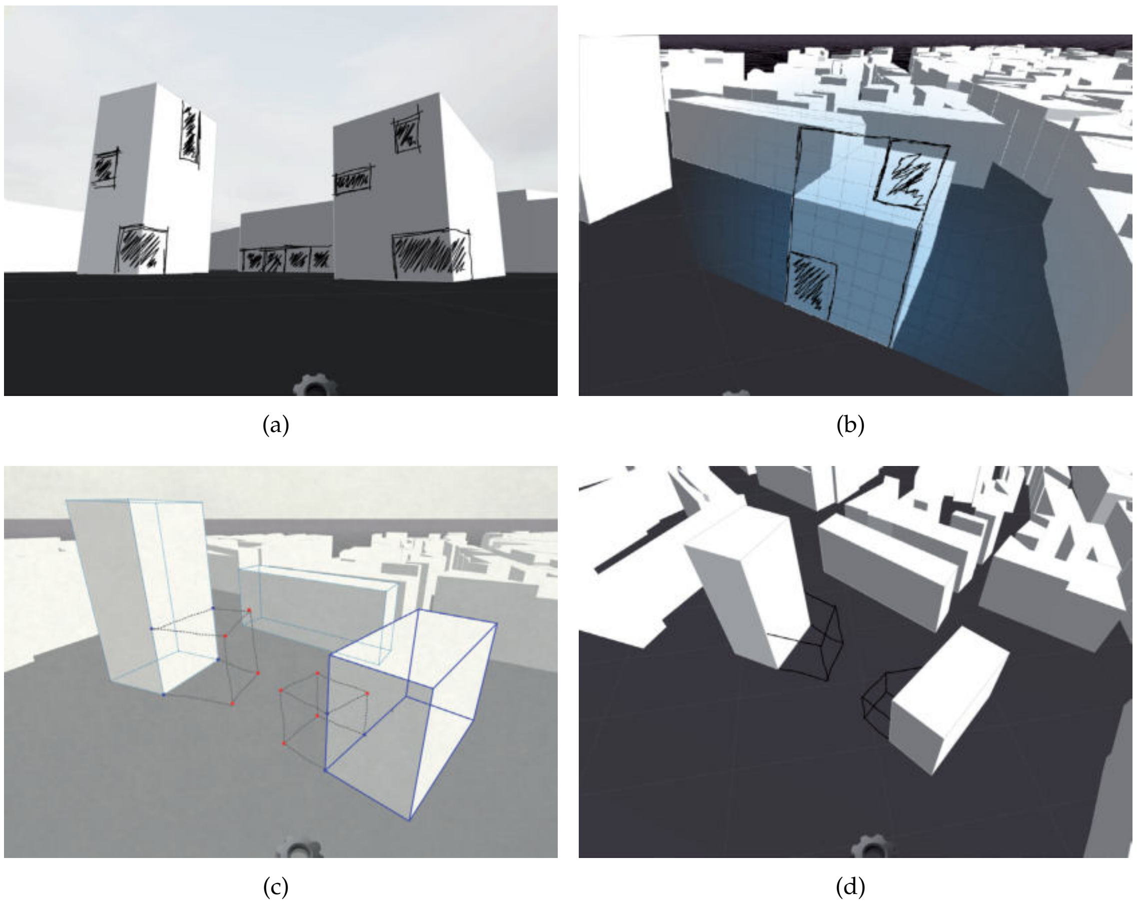


Figure 4.8.: Sketching in the presentation mode. (a) standard mode; (b) extended mode; (c) 3D mode; (d) resulting sketch from the 3D mode.

having to recreate the entire environment on paper. Figure 4.8 illustrates examples of several sketching modes.

It is difficult to create true 3D sketches out of 2D input without involving additional user interaction. Conventional CAAD software tackles this problem by defining multiple 2D views on the same 3D environment, and lock out one of the camera axes in each view. The user can draw a shape in 2D space which is placed in its correct 3D position depending on which view was used to draw the shape. This kind of interaction is more complex than sketching in plain 2D. It requires a deeper understanding of conventional CAAD and 3D modeling tools. The presentation mode attempts to simplify this approach by introducing new ways of sketching.

Even though humans are able to recognize the shape of an object by looking at a 2D drawing, this ability is enhanced by the knowledge of how objects look or should look like in the real world. For example, a drawing of a circle can be seen as a circle or as an ellipse which is being drawn exactly from the angle which makes its projection a circle. But if the drawing is the wheel of a car, the observer will naturally decide that its shape is indeed a circle.

From mathematical point of view, due to the transition from 3D space to 2D space one

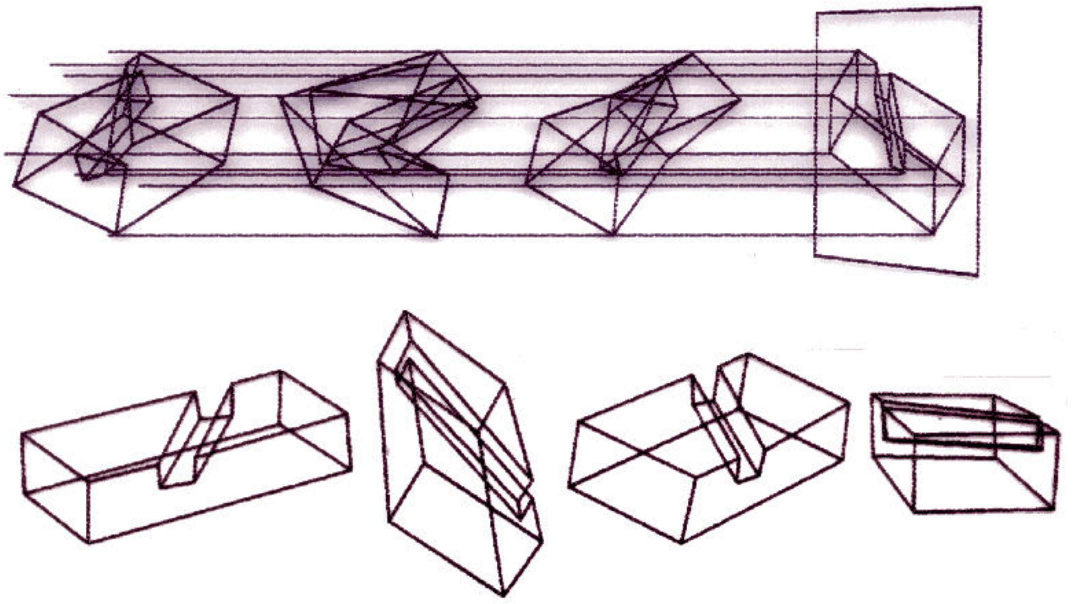


Figure 4.9.: Ambiguity of representation of a 2D sketch (figure according to [42]); Upper: possible 3D interpretations of a 2D sketch; Lower: common interpretation of the sketch by most humans.

dimension is lost and cannot be reconstructed deterministically (see figure 4.9). There are many empirical and theoretical approaches which try to solve this problem (reconstructing 3D objects from 2D drawings), most of which are based on constraints – assumptions that the shape of the drawn object must obey specific laws [68, 40, 42]. Technically, this is an attempt to simulate human behavior. Humans tend to recognize shapes based on experience and knowledge of similar shapes.

Many of the 3D sketching algorithms found in literature achieve very good results in terms of similarity of the reconstructed object to the real prototype. However, they require several tenths of seconds up to few minutes in order to reconstruct the object, because the algorithm optimizes a potentially large set of constraints over a large set of points/lines/-surfaces.

Long waiting intervals violate the requirements of the presentation mode where a delay in the calculation could distract the designer from his intent. Therefore, a simple and fast approach to the problem was desired and implemented in the presentation mode developed here. This approach uses the drawing environment (the virtual city) and tries to estimate which plane the user is drawing on. This concept is implemented in different variations within the presentation mode of CDP, resulting in four different sketching modes.

The next two sections describe the basic mathematical model used to map each stroke to its corresponding surface (section 4.4.3.1) and the individual sketching modes which use this model (sections 4.4.3.2, 4.4.3.3, 4.4.3.4, and 4.4.3.5).

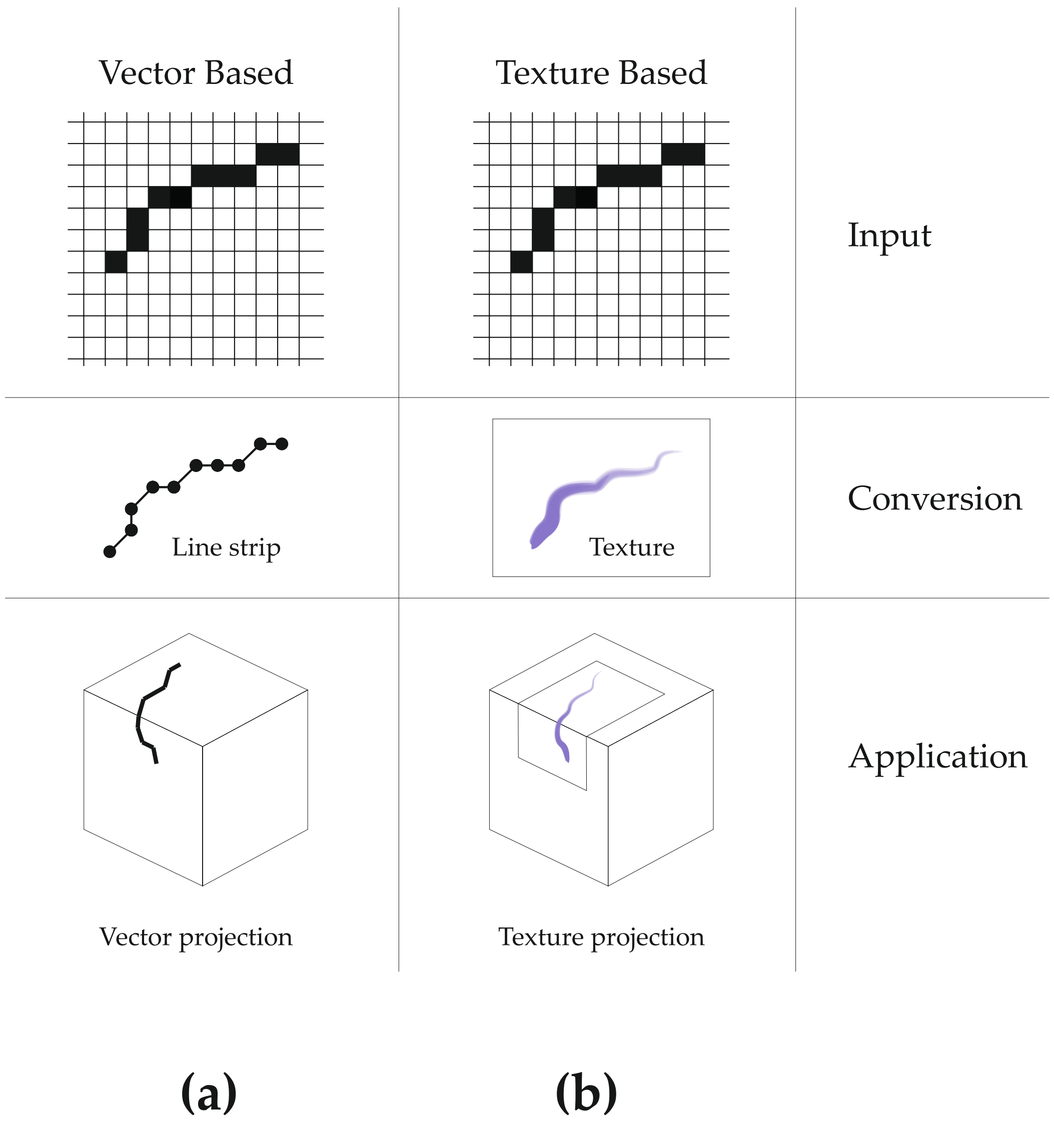


Figure 4.10.: Two possible sketching approaches – (a) vector based and (b) texture based.

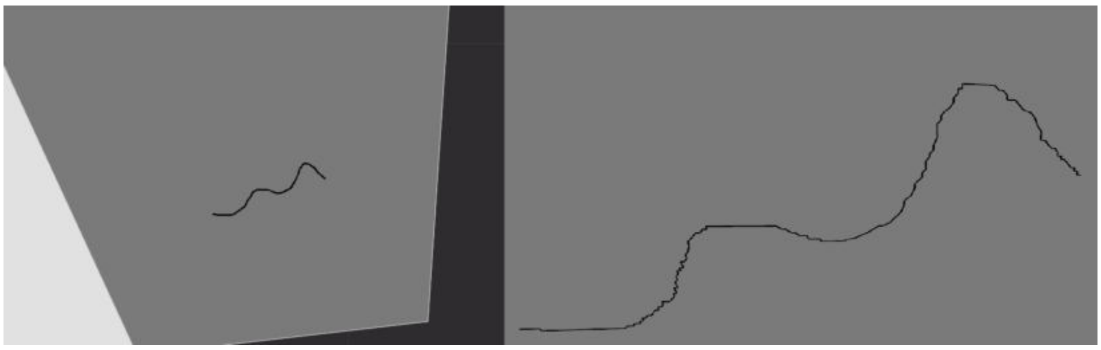


Figure 4.11.: A sketch stroke drawn on a surface (left) and the same stroke when zoomed in (right)

4.4.3.1. Mathematical model

How does the presentation mode know where and what the user is trying to draw? The basic method used in this thesis is to find the surface lying directly under the pen and project the sketch onto that surface. This method resembles the standard sketching mode (see section 4.4.3.2).

There are two approaches which were considered for the implementation of this behavior – vector based and texture based (see figure 4.10). The **vector based** approach represents the drawn stroke as 2D coordinates in the screen space and as 3D coordinates in the virtual city space. For each screen coordinate, a P-2 ray (see Appendix A) is computed, which starts at the position of the camera and goes "into" the scene. Its intersection with an object defines the 3D position of the sketch point. Connecting all points from a sequence forms the 3D curve (figure 4.10a). The **texture based** approach captures the drawing in screen space as a sequence of 2D points and renders it on a texture. The texture is then "projected" to whatever geometry lies behind it (figure 4.10b).

The **vector based** approach was chosen for the implementation of sketching in the presentation mode. It is simpler in terms of implementation, portability (it is easier to transport and map a set of points than a texture) and program logic. It has the only disadvantage to produce jagged artifacts sometimes which become visible when the view is zoomed closer to the drawing (see figure 4.11 right). The reason for these artifacts is that the 2D screen coordinates are pixels. Therefore, lines drawn on the screen are jagged and their projection is also jagged.

A single sketch stroke starts when the pen touches the presentation screen and ends when the pen is removed from the screen surface. Each pixel touched by the pen between these two events is considered part of the stroke (a **stroke point**). The set of these points defines a stroke. A set of strokes resembles a stroke sequence.

A stroke point has the following components:

(A) The 2D screen coordinates

(B) The P-2 projective coordinates

(C) The 3D coordinates

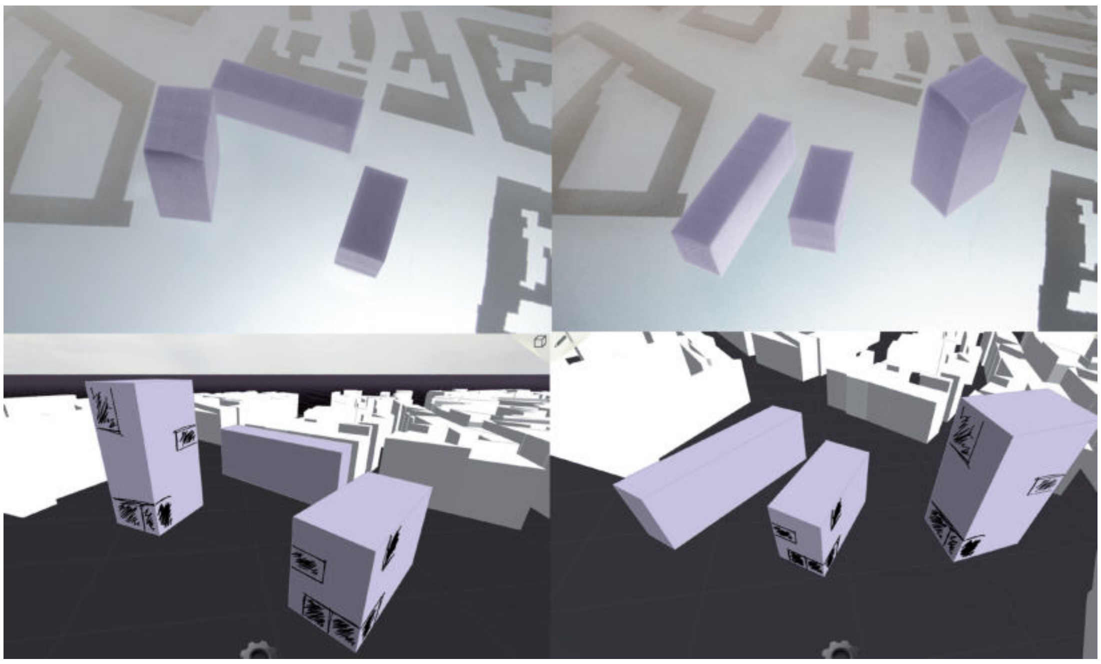


Figure 4.12.: Corresponding views of the table top and the presentation screen, demonstrating the stable relation of sketches to the scene. The sketches remain “glued” to their surface, whenever the camera or the building moves.

‘A’ measures the point as a pixel in screen coordinates. ‘B’ is computed based on the current camera position. A ray passing through the middle of the pixel defined by ‘A’ is constructed, which corresponds to the visual ray through that pixel of the camera viewport. This P-2 ray represents all points in 3D space whose projection falls on that viewport pixel. The third component, ‘C’, is the actual 3D position of the sketch point. This component is unknown at the time of drawing and needs to be computed. The way how it is computed is different for the various sketching modes. Once the 3D positions of the points of the sketch are determined, the sketch will appear as if it was “glued” to its surface whenever the camera moves (see figure 4.12).

The following sections illustrate the implementation details about the developed sketching modes (standard, extended, sketch paper and 3D). For the conceptual description of each mode, please refer to chapter 3.

4.4.3.2. Standard mode

The standard sketching mode is active by default until the user activates a different mode. It is not the simplest one in terms of program logic, though. It maps each sketch stroke to the surface lying directly underneath the pen (see figure 4.13).

In order to assign each point of the sketch to its correct 3D location, the algorithm has to accomplish three steps:

1. Compute the P-2 ray of the point (using the 2D screen coordinates and the camera position);

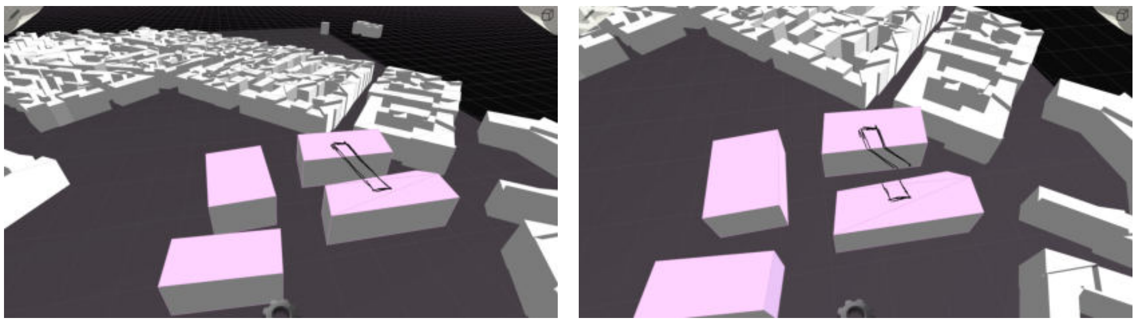


Figure 4.13.: The standard sketching mode. Left: while sketching; Right: after moving the camera slightly (it indicates that the mode is not well suited to "sketch in the air")

2. Find the underlying surface;

3. Find the intersection of the P-2 ray with the surface.

After performing these steps, one can determine the 3D position of a point uniquely and assign it to a surface. Unfortunately, having many objects in the scene consumes a lot of computation time at step 2, causing a significant delay which diminishes the amount of stroke points that can be handled and results in a bad sketch quality. Moreover, if implemented without optimization, the calculation of step 2 blocks the CPU for a short duration which also causes the GUI to freeze and results in bad user experience.

A relatively simple solution to this problem is found in threading. Assigning each point to a different thread caused the computation to speed up significantly. Tested on a system with 8-core CPU, the standard drawing mode performed well even when the table surface was densely populated with building models.

Threading causes some complications, though. The following conditions need to be maintained:

- Each point is assigned to exactly one thread.
- Access to the point data is synchronized.
- Points are processed in the order they were drawn.

All these tasks are solvable with standard multithreading techniques – a priority queue to hold the points in a sequence, semaphores to count the thread access to the queue, and mutexes to synchronize the access to the queue.

The intersection of each point with its underlying surface yields a 3D point. Connecting these 3D points yields a 3D stroke. Moving the corresponding surface however would leave the stroke "floating" in space, which is not the desired behavior of the sketching tool (see figure 4.14); a stroke should move together with its surface. Thus, the world coordinates of the stroke points are replaced with coordinates relative to the corresponding surface. Also, the stroke points are attached to the transform of the surface so they follow whenever the surface moves. The transform is split using the following scheme:

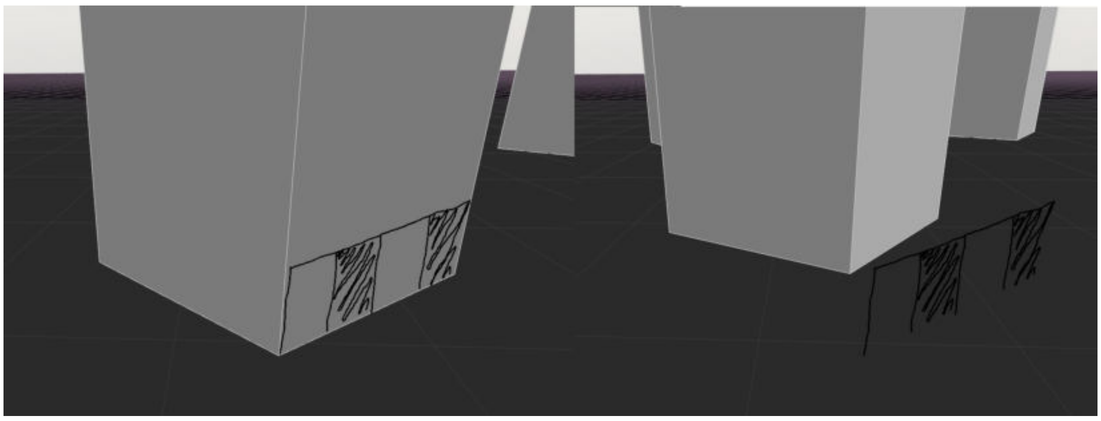


Figure 4.14.: A sketch without connection to its underlying building. Left: sketch drawn on a building; Right: the same sketch after moving the underlying building away.

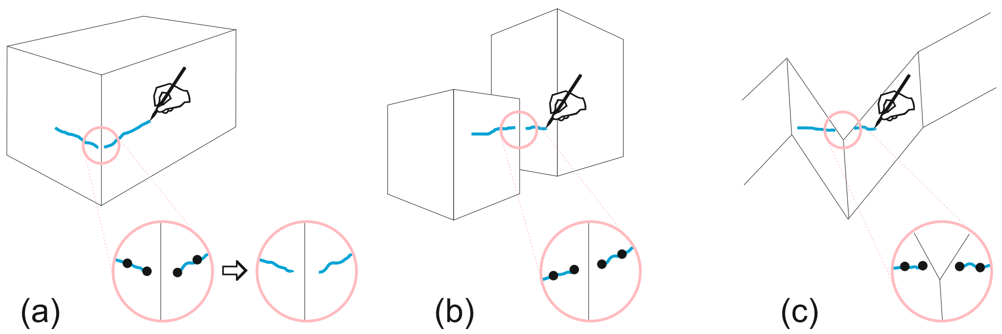


Figure 4.15.: Flaws in the sketch strokes in the standard mode and their respective corrections; (a) a stroke segment crosses a surface boundary; (b) a stroke leaves a building; (c) a stroke segment crosses two surface boundaries.

Let P be the 3D position of a point in world space, and S – the surface it is attached to. Let T_s be the transform of S, T_p the transform of P and P_{new} the new 3D position of P. Then the point P is transformed using the formula:

$P_{new} \leftarrow (T_s)^{-1} * P$
$T_p \leftarrow T_s$

The new position of P is $T_p * P_{new}$, which equals P (the initial position of the point) if the surface remains static. If the surface moves, then P will move in world space together with the surface, but will stay static relative to the surface.

Another difficulty regarding this (vector-based) sketching method comes from the fact that the presentation screen has a finite precision (1920 x 1080 pixels). The sketch is essentially a set of pixels which are converted to rays and intersected with the surfaces in the scene (see Appendix A). Very often, two consequent P^2 rays P_1 and P_2 from a stroke will intersect two adjacent surfaces S_1 and S_2 each and none of the pixels will lay on the

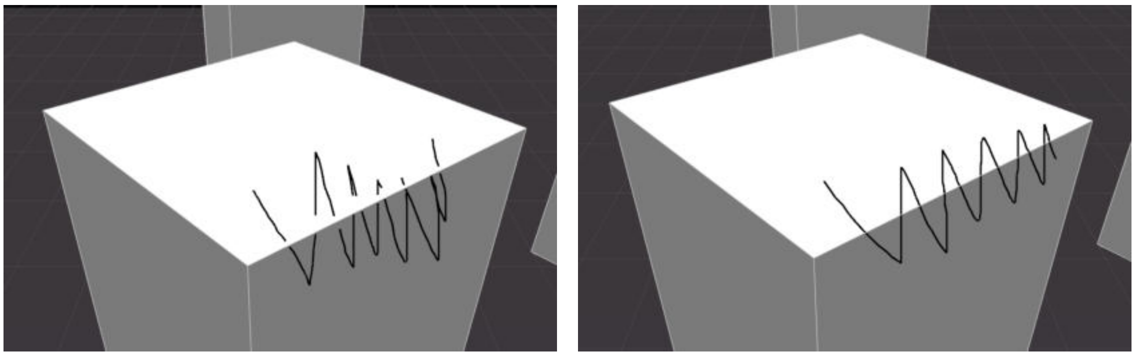


Figure 4.16.: Adding an additional point on boundaries improves the quality of the sketch. Left: without the additional points; Right: after adding the additional points on the boundary.

common edge E_c between the surfaces [2]. Since the algorithm connects the two consecutive points with a line, that line lies inside of the object defined by the surfaces and will not be visible from outside (see figure 4.15a). In the case of a concave edge, the line segment would "hang in the air", which is also not desired.

In order to enhance the sketch strokes on surface boundaries, a new point P' is created with has the following properties:

- It lies on the edge E_c
- It lies on the plane π defined by P_1, P_2, and the camera position

This point is unique, because E_c cannot be parallel to π (otherwise, S_1 and S_2 would be rendered as a line on the screen and the user would not be able to draw on them). Thus, E_c intersects π in exactly one point. However, sometimes the sketch stroke crosses several edges at once, not just one (see figure 4.15c). This case was not handled as it may occur very rarely and could be therefore neglected.

Intuitively, the forged point P' is the point where the interpolated curve of the stroke crosses the edge E_c between S_1 and S_2. Figure 4.16 shows the result of adding this new point.

The last difficulty causes the transition of the pen from one surface to another when the two surfaces have no common edge. This happens, for example, when the designer is drawing on two building models, one occluding the other, and the pen jumps from a surface of the front building to a surface of the back building (see figure 4.15b). In this case, two new points have to be added – on the boundary of each surface. These points split the stroke into two parts – one finishing at the boundary of the "old" surface, and one starting at the boundary of the "new" surface.

[2]This case occurs even more often if the computer performance is not sufficient, or if the scene has many polygons.

4.4.3.3. Extended surface mode

The extended surface sketching mode is activated whenever the user clicks on a surface. Every next stroke is projected to the plane of this surface. This mode is much simpler than the standard mode as it does not have to care about surface boundaries and transition between surfaces.

Whenever the extended mode is active, the selected surface is highlighted with a blue elliptic semi-transparent overlay gradient, which is approximately three times bigger than the surface itself and fades towards the outer borders of the surface. In addition, a transparent grid is displayed over the gradient in order to enhance the perception of space and proportion.

If a different surface is clicked, that surface gains focus and strokes are projected to its plane. If the same surface is clicked twice, the extended mode is deactivated and the standard mode is restored.

Technically, the extended mode works just like the standard mode, using the same intersection method. The only difference is that in the extended mode only one plane is used to project the sketch points.

4.4.3.4. Sketch paper mode

The sketch paper mode is activated by dragging a paper edge from the top left corner of the screen. The paper edge unfolds into a sheet which covers the entire screen except for a scrollable thumbnail area at the bottom. The sheet is semi-transparent can be dragged away with the same edge which was used to pull it into the screen area.

The paper mode mimics real sketch paper. The user can pull a new sheet at any given moment, without losing the contents of the old sheet. All sheets are visible at the bottom of the screen as thumbnail images. One can switch to a sheet by clicking on its thumbnail, or pull a new sheet by dragging a button at the left of the thumbnail bar. It is also possible to scroll the thumbnail bar if the number of sheets is too large to fit in the bar. Selecting a sheet also moves the camera to the position where it was at the time the sheet was created.

Each thumbnail has two buttons – a “discard” button and a “bake” button. The function of the discard button is to delete the thumbnail and its corresponding paper sheet. The bake button applies the sketch drawn on the current paper sheet to the scene objects lying underneath. Figure 4.17 depicts an example usage of the sketch paper mode, where two layers are used to visualize the conceptual infrastructure of an urban area. Projection works similarly to a beamer, which projects an image to whatever geometry lies in front of the beamer frustum.

In order to project the sketch, the sketch paper mode uses the same algorithm as the standard mode. Thus, sketching in the sketch paper mode and using the “bake” button has the same effect like drawing when using the standard mode. The advantage of using the paper mode though is that the sketch is not permanently applied, but can be discarded and redrawn.

There are other features of the sketch paper mode that can be offered over the standard mode but not yet implemented. For example, it is possible to stack multiple sketch paper layers on top of each other, just like architects usually do when using real sketch paper (see figure 4.31). As the sheet is light and transparent, one can overlay several sheets without

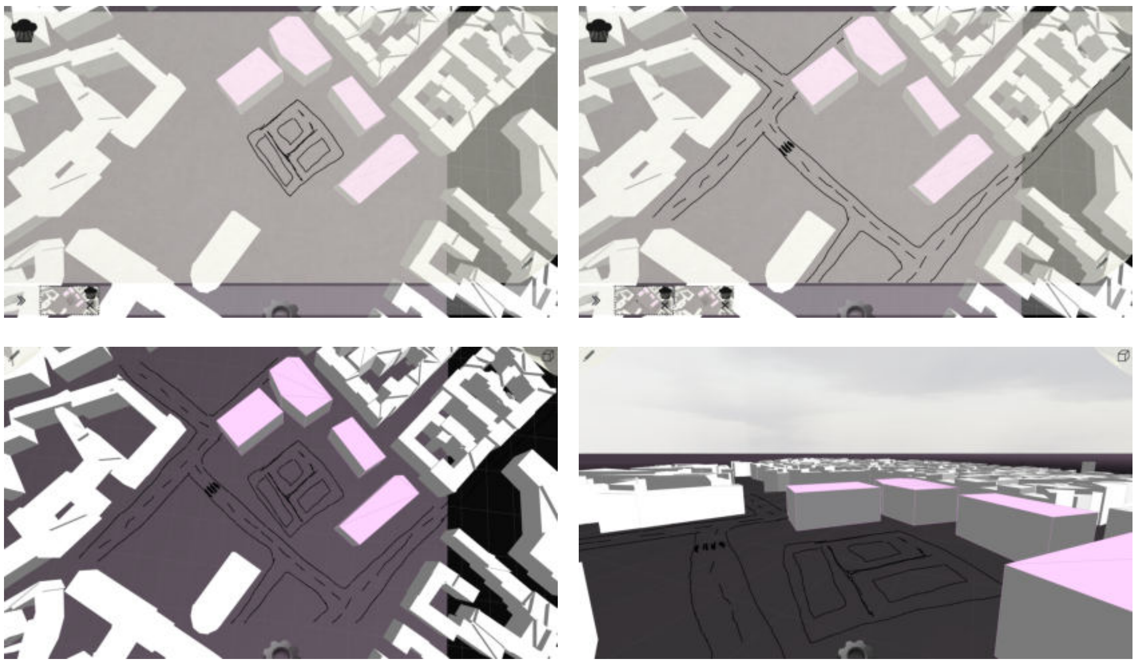


Figure 4.17.: Usage of the sketch paper mode. Top left: drawing an inner yard on the paper; Top right: drawing streets in a separate paper layer; Bottom left: Projecting both layers; Bottom right: Projection remains intact even after changing the view.

completely obscuring the bottom image (in this case, the virtual city environment).

Another advantage of the sketch paper mode is its planarity, which brings it closer to traditional tools in architecture. An imaginable feature of this mode is an eraser tool, which can wipe parts of the sketch before being applied to the virtual city. It is much easier and intuitive to implement this feature in the 2D sketch paper mode than in 3D space.

4.4.3.5. 3D sketching mode

The most advanced drawing mode – the 3D mode – estimates the real shape of a volumetric object drawn by the user. This mode is considerably more complicated than the three modes described so far, but it is also more powerful. In the following, a simplified overview of the 3D sketching algorithm is presented. More details regarding the individual stages of the algorithm will be given in paragraphs 4.4.3.5.1, 4.4.3.5.2, and 4.4.3.5.3.

After the user activates the 3D mode with the sliding paper edge at the top right corner of the screen a semi-transparent paper sheet slides over the screen. The paper sheet looks like in the sketch paper mode, but here the edges of the buildings in the scene are enhanced with overlaid lines (see figures 4.18a and 4.18b). Edges and corners of buildings in the virtual city play a major role in the reconstruction phase. The reconstruction phase is explained in more detail in paragraph 4.4.3.5.2.

The designer must select a virtual building which serves as a reference for the sketch, as shown in figure 4.18b. This selection is necessary, otherwise the algorithm will have to check every virtual building in the city for collision with the sketch strokes. The selected building is used to assign the sketch real dimensions in the virtual world. This process is explained in detail in paragraph 4.4.3.5.2.

The designer can sketch on the sheet like when using the sketch paper mode. The strokes are drawn with a dashed line to indicate that it is not complete yet. Once the sketch is projected to 3D space, the dashed lines are converted to solid lines.

While the designer draws, the following actions are performed in the background:

- Each stroke is subdivided into straight lines (see figure 4.18c and 4.18d)
- At both ends of each line, "connecting points" are added, so-called "joints", and displayed (see figure 4.18d and 4.18e)).
- Each stroke segment is connected to other stroke segments or building edges, if their ends are touching or almost touching. Their respective joints are also merged.

The whole process is visualized in figure 4.18. It produces an undirected graph, which consists of the lines in the sketch (the **edges**) and the joints connecting them (the **nodes**). The graph is an intermediate representation of the sketch. After the designer clicks the "apply"-button, an attempt is made to estimate the shape of the sketch and project it into 3D space, as depicted in figure 4.18f.

The following limitations are imposed:

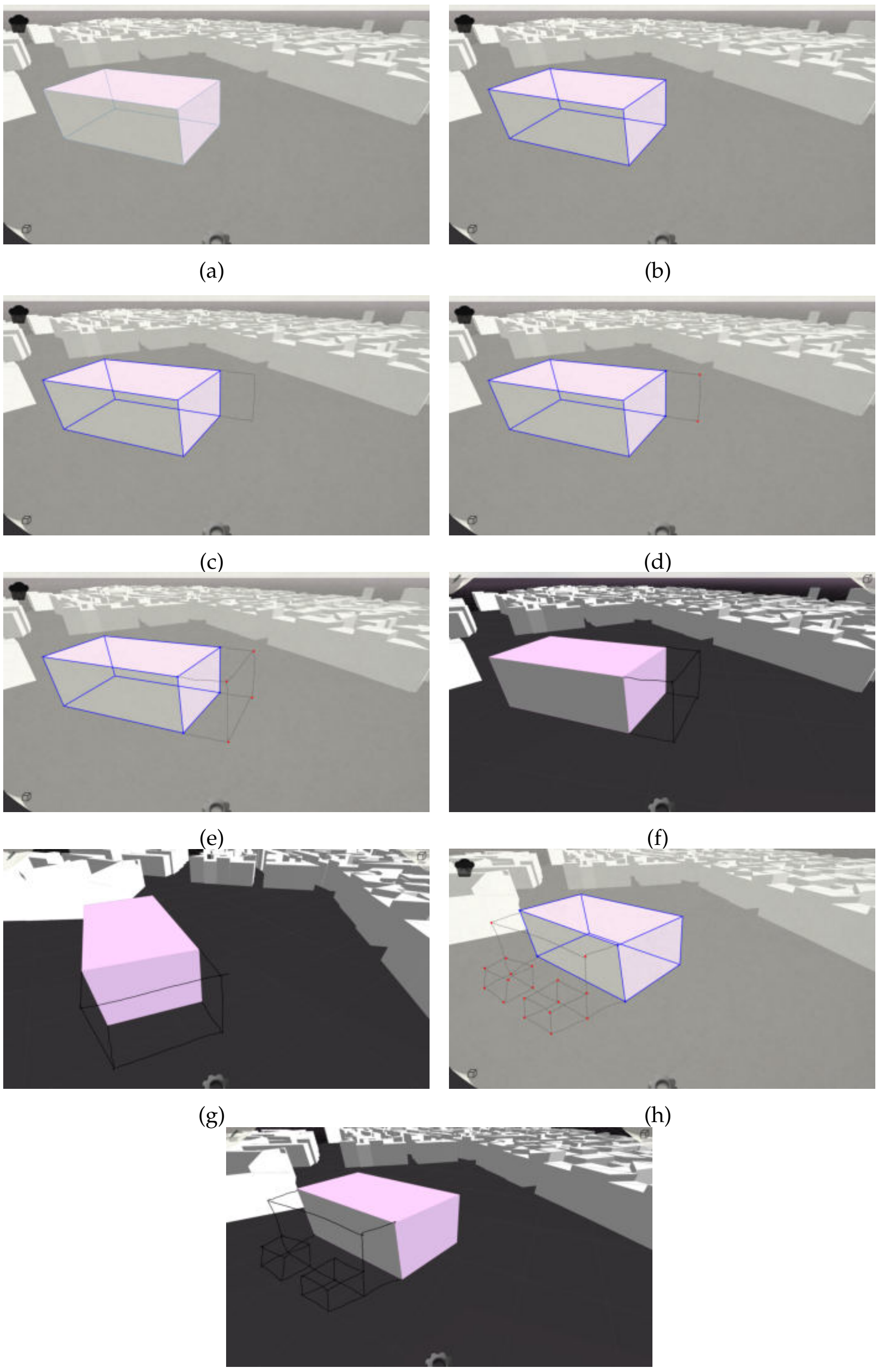
- The sketch is reconstructed to a shape only upon user's request. No intermediate results are visible.
- The camera cannot be dislocated while drawing.

The algorithm attempts to gather as much information as possible before computing the final 3D shape. An alternative approach is to refine the 3D shape iteratively with every further stroke. Iterative methods are much more complicated though, and will be a subject of further research.

The reconstruction algorithm can be subdivided into three logical steps:

- Constructing the sketch graph
- Reconstruction of the 3D shape
- Projecting the sketch to 3D space

The first step constructs a graph out of the raw sketch. The reconstruction step converts the graph into a 3D shape consisting of 3D vertices and topological connection between them. The last step – projection, transforms the 2D strokes of the raw sketch into 3D curves so that the final result looks similar to the original sketch. The following sections provide a more detailed insight into each single step.



(a) (b) (c) (d) (e) (f) (g) (h) (i)

Figure 4.18.: The 3D sketching mode in use. (a) A scene with one object; (b) Selecting the object; (c) Drawing a wall with one stroke; (d) After releasing the pen, the stroke is segmented, and the lines - connected with joints; (e) Drawing the rest of the sketch; (f) Reconstruction result; (g) Rotating the camera; (h-i) A more complex sketch.

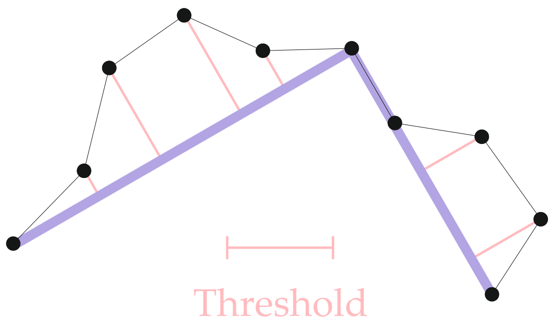


Figure 4.19.: The threshold parameter of the Douglas-Peucker algorithm. Black dots correspond to the original curve. The blue lines correspond to the approximated curve. All distances, marked with red, have to be smaller than the given threshold.

4.4.3.5.1. Constructing the sketch graph

The construction of the sketch graph comprises *stroke subdivision* and *stroke connection*. The first handles the subdivision of curved strokes into line segments, while the second handles the topological connection of strokes within the graph. Both actions are performed for each stroke after the designer lifts the pen.

Stroke subdivision refers to the segmentation of strokes into a set of lines. It is necessary whenever the designer draws two or more lines without lifting the pen, or when drawing curves.

Stroke subdivision is performed using the OpenCV [54] function approxPolyDP, which takes a contour and returns a set of lines along that contour to achieve best approximation. The function is based on the Ramer-Douglas-Peucker algorithm [69]. The algorithm accepts a precision value for the approximation. The value corresponds to the maximal tolerated distance between any of the curve points to the approximated line sequence (see figure 4.19). For the purposes of the 3D sketching mode, a relatively large tolerance is chosen, so that fewer line segments are produced. The more line segments, the more calculations are needed later on. Besides, the line segmentation has almost no effect on the final visual appearance of the sketch. It influences only the topology of the produced graph, which is used for the 3D reconstruction.

Stroke connection is performed immediately after stroke subdivision. It ensures that every stroke is linked to the graph based on its topological relation to other strokes. Each stroke end is tested for collision with any of the existing joints in the graph and with all structures within the urban environment. There are four possible entities which can lie under a stroke end. Hence, the following collision outcomes are possible depending on the underlying structure (see also figure 4.20):

- An already existing joint. In this case, the ending is linked to the joint.
- A corner of a building. In this case, a new static joint is created at the corner and linked to the stroke end. This means the new joint has a fixed 3D position – the position of the building corner.

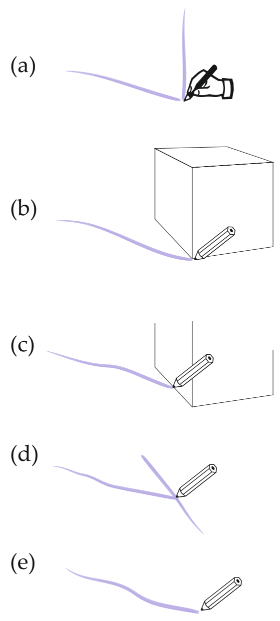


Figure 4.20.: Five cases encountered when connecting a stroke to the existing graph depicted in the order they are checked: (a)another joint, (b) a building corner, (c) a building edge, (d) another stroke, and (e) none of the above.

- An edge of a building. This case is processed like (2) and yields again a fixed joint
- An existing stroke. In this case, a new joint is created at the intersection. This joint splits the colliding stroke and has three incident edges accordingly.
- None of the above. In this case, a new joint is created and linked to the stroke.

These five cases are tested in the same order as they appear in the list. Hence, a building corner has precedence over a building edge. This way if the designer draws a stroke end at the corner of a building, the stroke will snap to the corner and not to an edge starting from that corner, even when the end point of the stroke does not coincide exactly with the corner. An image of the graph produced in this way is shown in figure 4.21.

4.4.3.5.2. Reconstruction of the 3D shape

The actual reconstruction starts as soon as the user activates the "apply"-button at the top left corner of the screen. It takes as input the graph generated in the previous step and estimates the 3D positions of the joints within the graph. The result of the estimation is a mesh with 3D vertices and 2D stroke segments connecting them. Strokes are projected in the next step of the algorithm described in 4.4.3.5.3.

Figure 4.22 shows the taxonomy of algorithms approximating a 3D shape out of a sketch. The algorithm implemented here uses graph propagation. It traverses the sketch graph and updates the position of each joint according to a set of rules (constraints). Each joint is assigned a type defining also its processing priority. Table 4.2 presents a list of all joint

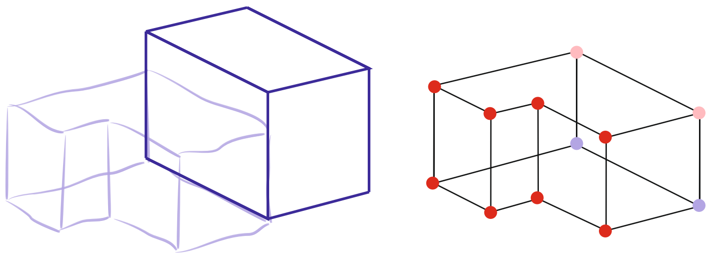


Figure 4.21.: Conversion of a sketch into a graph. Left: the source sketch (dashed line) and an auxiliary building (solid blue line); Right: the resulting graph. Red nodes are "floating", dark blue nodes belong to the base contour of the physical building, and light blue nodes belong to side edges.

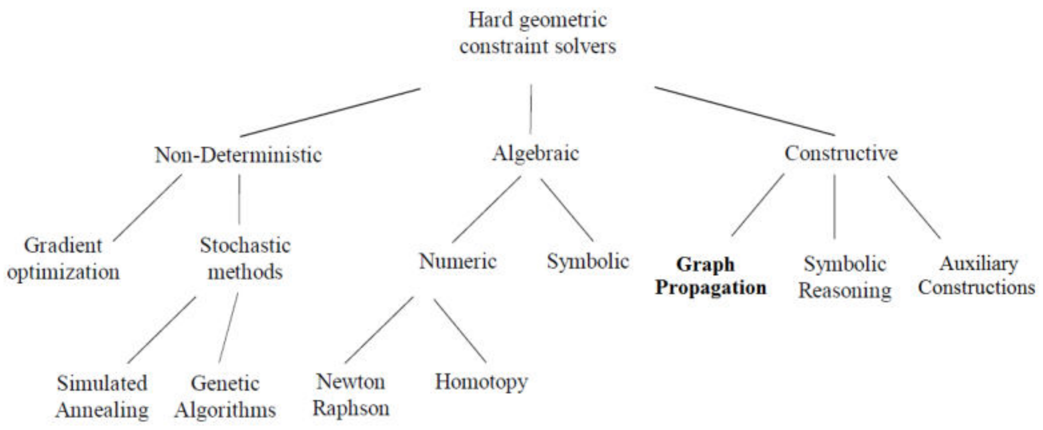


Figure 4.22.: Approaches to 3D reconstruction from sketches. (image according to Hod Lipson [42])

FLOATING	100	Not yet computed joint
BUILDING_FOUNDATION_CORNER	0	Coincides with the corner of a building foundation
BUILDING_FOUNDATION_EDGE	1	Lies on the edge of a building foundation
FOUNDATION_PROLONGED	2	Lies on the city plane
RAISED_VERTICALLY	3	This joint was "raised" vertically by another joint. The line between the two is orthogonal to the city plane
PROLONGED_HORIZONTALLY	4	
BUILDING_ROOF_CORNER	5	Coincides with the corner of a roof
BUILDING_ROOF_EDGE	6	Lies on the edge of a roof
BUILDING_SIDE_EDGE	7	Lies on a vertical edge of a building

Table 4.2.: Types of joints and their processing priority. The underlined joints are with fixed 3D positions.

types and their description. A graphical explaination of joint types is illustrated in figure 4.23.

The lower the priority number of a joint, the earlier it is processed. The reconstruction starts from the joints with a fixed position lying on the base contour of a building. Strokes starting from these joints most probably describe a base contour. After all joints of this type have been processed, the execution proceeds to joints with the next priority class, until there are no more joints to process.

The main characteristics of the reconstruction algorithm (Input, initialization, iteration step) are presented below, followed by a discussion about the functionality, side effects and limitations of this approach.

Input

The algorithm receives an undirected graph with vertices (joints, V) and edges (strokes, E). Each joint has an initial type (see table 4.2) which might be either a fixed type or type FLOATING. Fixed joints are these which are linked to a structure in the scene. Joints of type FLOATING are without a fixed 3D position yet; they need to get one assigned.

Pseudo code:
$G = (V \subset R^2 \times R \times T, E \subset V \times V)$

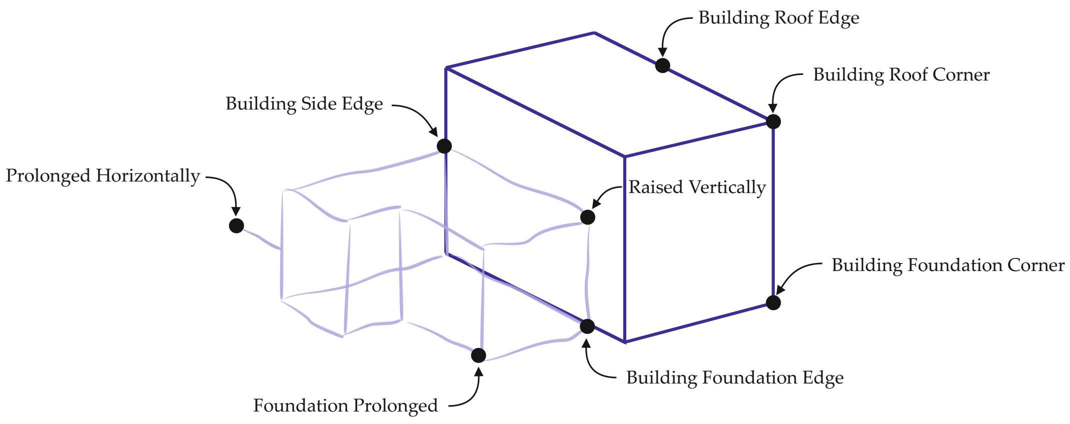


Figure 4.23.: Different types of joints.

where G is the graph, V are the joints, T are the joint types, and E is the set of strokes. R^2 is the 2D screen space domain, and R is the depth value of a joint. We introduce the following notation to access the individual components of a joint:

$$\mathrm{coord2d}(v) := \{\, c_{2d} \,|\, v = (c_{2d}, d, t) \wedge v \in V \,\}$$

$$\mathrm{depth}(v) := \{\, d \,|\, v = (c_{2d}, d, t) \wedge v \in V \,\}$$

$$\mathrm{type}(v) := \{\, t \,|\, v = (c_{2d}, d, t) \wedge v \in V \,\}$$

Initialization
All joints with a fixed position are organized in a priority queue. The priority is derived from the joint type, meaning that joints of specific type are processed earlier than others. *Note: fixed joints are used to initialize the processing queue, but they are not necessarily processed before other joints. A newly inserted joint displaces all joints with lower priority than its own. This is exactly the behavior expected from a priority queue.*

Pseudo code:

```
$V_i \leftarrow \{\, \emptyset \,\}$
**for all** $v \in V$ **do**
    **if** $\mathrm{type}(v) \in T_f$ **then**
        $V_i \leftarrow V_i \cup v$
    **end if**
**end for**
```

where T_f is the set containing all fixed joints (see table 4.2) and V_i is the set containing the initialization joints.

Iteration step
At each step, the joint with the highest priority is extracted from the priority queue and processed. Let this joint be v_p. Within one iteration of the algorithm, all adjacent to v_p joints

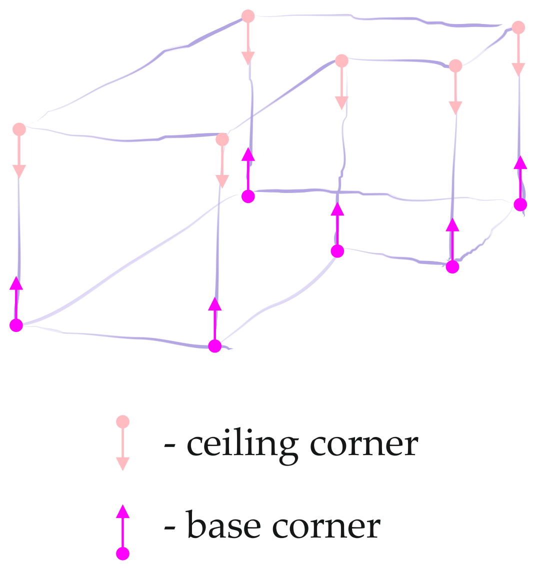


Figure 4.24.: Estimation of the type of corner based on the direction of the orthogonal edge starting from the corner. A direction pointing "up" indicates a base corner (here green), whereas "down" indicates a ceiling corner (here red).

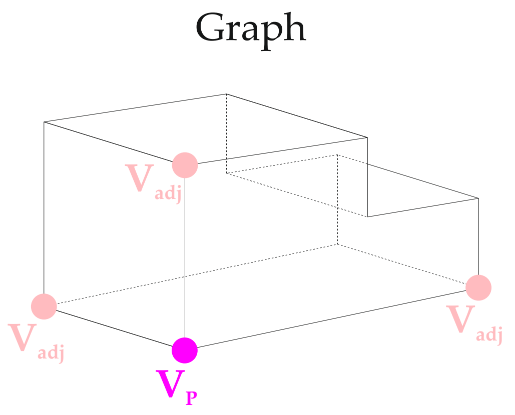


Figure 4.25.: The neighborhood V_{adj} of a joint v_p.

are collected and their 3D position calculated according to a given rule (constraint). Let the set of these joints be V_{adj} (see figure 4.25). If an adjacent joint has a 3D position already, then it is skipped. At the end of the iteration, all joints from the set V_{adj} are inserted into the processing queue, unless they were already processed. This guarantees that the algorithm terminates, as no joint can be processed twice, and each step processes exactly one joint. To avoid confusion, the joint v_p is *already positioned* in 3D space. Only the *adjacent joints* V_{adj} are calculated. All joints in the priority queue have a 3D position already. The queue is used merely to traverse the graph G, not to hold the joints which are not calculated yet. Thus, a joint is inserted into the queue only after its 3D position has been calculated.

A *constraint* in the context of this algorithm is a rule applied locally on v_p, which influences the position of the adjacent joints V_{adj}. The type of the applied constraint depends on the arrangement of the processed joint and its neighborhood. Currently, only two constraints are implemented, which correspond semantically to a *base corner* and a *ceiling corner* of a building, respectively (see figure 4.24).

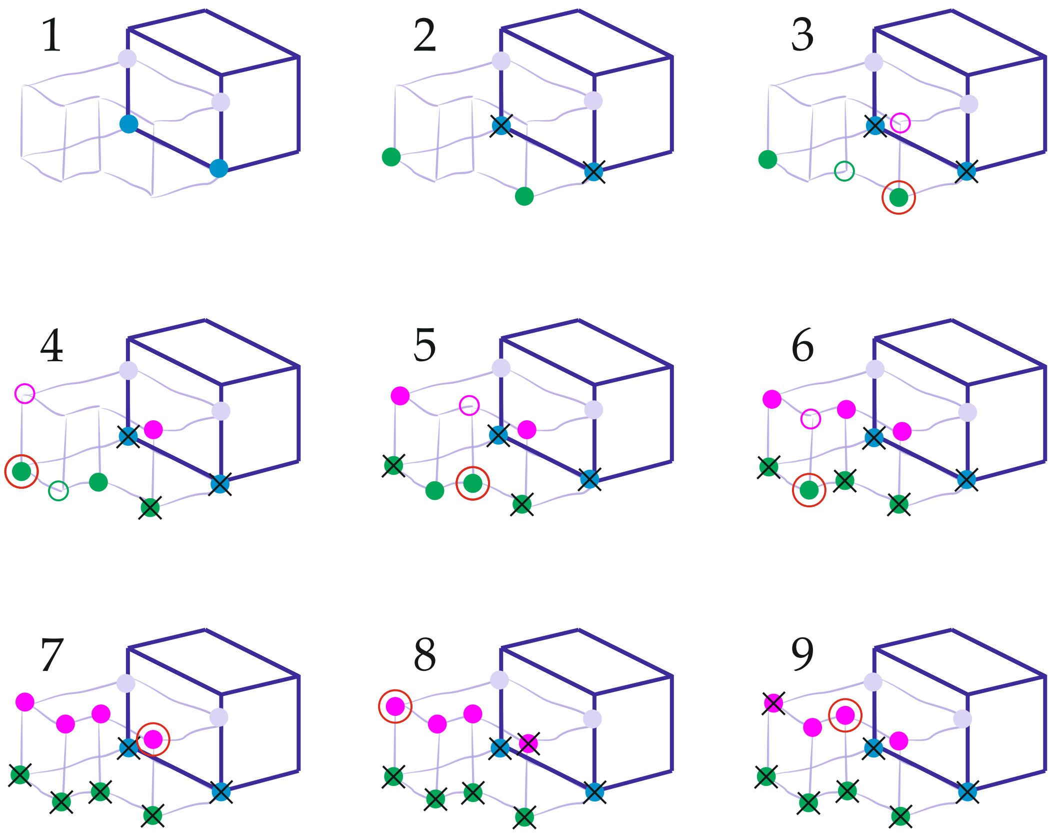
A corner is a joint which is incident to at least three strokes from which exactly one stroke is upright (orthogonal to the city plane). If the orthogonal stroke points "upwards", then the corner is assumed to belong to the base of a building and the rest of the strokes are assumed to describe the base contour. Similarly, if the orthogonal stroke points "downwards", it is part of a ceiling corner and the rest of the strokes belong to a ceiling contour. More information about measuring angles of strokes in the projective space is given in appendix A.

Let's label the joints from V_{adj} as v_0, v_1, ..., v_n, and the strokes which connect them to v_p, as e_0, e_1, ..., e_n accordingly. Let the strokes e_i be ordered after their proximity to the 90 degree axis in a typical polar coordinate system [3] as illustrated in figure 4.27a , so that e_0 is closest to 90 degree and e_n is closest to 270 degree (see figure 4.27). Table 4.3 presents the cases which can occur depending on the angular arrangement of the strokes.

Pseudo code:

```
$Q \leftarrow \{\,\text{empty\_queue}\,\}$
$Q_p \leftarrow \{\,\emptyset\,\}$
for all $v \in V_i$ do
    insert($Q$, $v$)
end for
while $Q \neq \{\emptyset\}$ do
    $v_p \leftarrow \max(Q)$
    $V_{adj} \leftarrow \text{neighborhood}(v_p)$
                                                        ▷ Apply a constraint
    $\text{ctype} \leftarrow \text{detect\_constraint}(v_p, V_{adj})$
    if $ctype == \text{BASE\_CORNER}$ then
        $v_{max} \leftarrow \text{max\_edge}(v_p, V_{adj})$
        $\text{raise\_vertically}(v_p, v_{max})$
        $type(v_{max}) \leftarrow \text{RAISED\_VERTICALLY}$
        $V_{adj} \leftarrow V_{adj} \cup v_{max}$
    end if
    for all $v \in V_{adj}$ do
```

[3] a stroke pointing to the right is of 0 degree, with angles increasing in a counter-clockwise direction.



●	=> in the priority queue	🟦	- BUILDING_FOUNDATION_CORNER
⊗	=> not in the priority queue	⬜	- BUILDING_SIDE_EDGE
○	=> computed position	🟩	- FOUNDATION_PROLONGED
◯	=> joint currently in process	🟪	- RAISED_VERTICALLY

Figure 4.26.: Several iteration steps of the reconstruction algorithm (1 to 9). Different colors indicate the type of a joint as specified in the legend. Unmarked joints are of type “floating”

Condition	**Applied constraint**	**Procedure**
$n \geq 3$ and $\lvert e_0 - 90\rvert < 7$	Base corner	v_0 is placed above v_p in the 3D space and is assigned type RAISED_VERTICALLY. v_i , (i from 1 to n) are placed so that the line (v_p, v_i) is parallel to the city plane and are assigned type FOUNDATION_PROLONGED.
$n \geq 3$ and $\lvert e_n + 90\rvert < 7$	Ceiling corner	v_i , (i from 1 to $n-1$) are placed so that the line (v_p, v_i) is parallel to the city plane and are assigned type PROLONGED_HORIZONTALLY.
None of the above is true	No constraint identified	v_i , (i from 0 to n) are placed so that the line (v_p, v_i) is parallel to the city plane and are assigned type PROLONGED_HORIZONTALLY.

Table 4.3.: Application of the defined constraints. n: number of adjacent joints; v_0... v_n : adjacent joints; e_0 ... e_n: incident strokes connecting these joints with the processed joint v_p. The strokes e_i are ordered in descending fashion by their proximity to 90 degree.

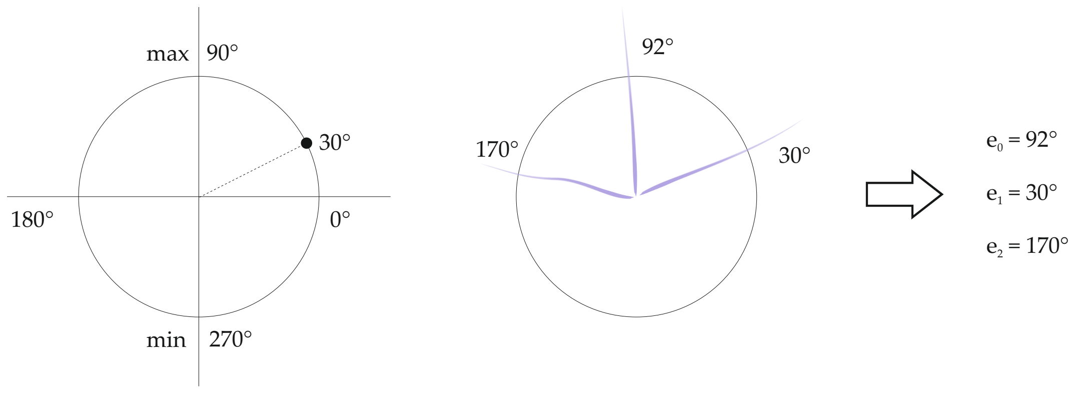


Figure 4.27.: Ordering of strokes by their proximity to 90 degree. Left: a typical polar coordinate system; Middle: a joint with three strokes and their respective angle; Right: the resulting order of the strokes.

```
        project_to_plane($v$, horizontal_plane($v_p$))
                                                        ▷ Switch $v_p$
        if type($v_p$) == FLOATING then
            error("This case branch can never be reached.")
        end if
        if type($v_p$) == FLOATING ∨
type($v_p$) == BUILDING_FOUNDATION_EDGE ∨
type($v_p$) == FOUNDATION_PROLONGED then
            type($v$) ← FOUNDATION_PROLONGED
        end if
        if type($v_p$) == PROLONGED_HORIZONTALLY then
            type($v$) ← PROLONGED_HORIZONTALLY
        end if
        if type($v_p$) == FLOATING ∨
type($v_p$) == BUILDING_ROOF_EDGE ∨
type($v_p$) == BUILDING_SIDE_EDGE then
            type($v$) ← PROLONGED_HORIZONTALLY
        end if
    end for                                             ▷ mark $v_p$ as processed
    $Q_p \leftarrow Q_p \cup v_p$
    for all $v_a \in V_{adj}$ do
        if $v_a \notin Q_p$ then
            insert($Q, v_a$)
        end if
    end for
end while
```

where

- Q is the priority queue or joint processing
- Q_p is the set of processed joints
- insert(Q, v) inserts an element v into a queue Q
- max(Q) returns the element with the highest priority in a queue
- neighborhood(v) returns all nodes in the graph adjacent to v
- project_to_plane(V, p) projects all joints from V to the plane p
- max_edge(v, V_{adj}) returns the joint v_{max} from V_{adj}, so that the edge (v, v_{max}) is closest to 90 degree in the polar coordinate system
- raise_vertically(v_p, v) projects the joint v "above" the joint v_p, i.e. so that the projection of v on the city plane falls on v_p
- horizontal_plane(v) returns the plane which contains the joint v and is parallel to the city plane

Discussion

In addition to the precise, formal specification of the reconstruction algorithm outlined in the sections above, here a more verbal description of the algorithm will be given by answering the following questions:

- How and why does the presented reconstruction algorithm work?
- What are the requirements and assumptions about the drawn sketch?
- What is the final result? What is the quality of the reconstructed shape?

The algorithm assumes the sketch represents a prismatic shape which has mostly horizontal or orthogonal to the city plane edges. It "travels" along the vertices (joints) of the sketch graph and applies the presented constraints until there are no more vertices to process.

The algorithm starts with the base contour. Joints which lie on the base contour but are also connected to a vertical edge "raise" this edge and set the position of the adjacent joint connected by this edge. After all base contour joints have been processed, the algorithm starts processing the "raised" joints. If such a joint has at least three strokes that start from the joint, it is considered a ceiling corner. Strokes which start from a ceiling corner and are not orthogonal to the city plane are projected to the plane parallel to the city plane and passing through the ceiling corner. This process continues until all joints have been processed. In the end, all joints are assigned a 3D position.

The algorithm does not always yield the correct 3D shape of the sketch. It depends to high extent on line precision, orthogonality of angles, and connectivity of the input graph. None of these conditions is universally maintained in hand-drawn sketches. On the contrary, drawings are usually imprecise and incomplete, which is also their advantage towards CAAD modeling [42].

The approach presented here is rather an exploration attempt towards 3D reconstruction than a robust solution. Nevertheless, it works for many sketches. In order to produce a plausible result, the following requirements should be fulfilled:

- The sketch contains only straight lines or lines close to being straight.
- There is a clearly recognizable base contour of the drawn building which lies on the city plane and is attached to the base contour of another building.
- Straight lines are drawn uninterrupted (without lifting the pen).
- Vertical edges are orthogonal to the city plane. In general, all facades should be vertical (90 degree with the city plane).
- Hidden lines must be drawn, too.
- Every line describes an edge and there are no accidental joints (joints which occur due to accidental edge or vertex collision).
- No lines are missing.

At first, these assumptions seem to be too hard and constraint the sketching to a great extent. In practice, designers follow them even when drawing on normal paper. For instance, if two corners coincide in the 2D view of the sketch, the user can fail recognizing them as different corners. Designers also rarely lift the pen in the middle of a straight line as it is hard to prolong it without reducing its quality.

Some requirements impose limitations, indeed. For example, common sketches often have a missing line or even few faces, which does not impede the ability of the reader to understand it. The presented algorithm, however, will probably fail to reconstruct the correct shape. Thus, the designer is forced to draw all lines.

Despite its simplicity and the large set of requirements, the presented approach produces good results. A more advanced and generalized algorithm is going to be a part of future research.

4.4.3.5.3. Projecting the sketch to 3D space

The steps of the algorithm described so far delivered the following information:

- The 3D positions of the vertices in the sketch
- The strokes which connect the vertices, in 2D screen space

This information is sufficient to construct the 3D shape. The next step is to create the lines connecting 3D vertices and compute the faces of the 3D shape. One way to do this is to connect the 3D vertices with straight lines and compute the faces out of the graph topology (this can be easily done with a face-search algorithm [70]). However, this representation would differ visually from the original sketch. Formally speaking, the resulting sketch would not coincide with the original one, if back-projected to the 2D screen space.

The goal of the 3D mode is to stay as close to the original sketch as possible, not to convert the hand-drawn strokes to a perfect 3D shape. The reconstructed sketch should therefore look like it is drawn by hand. To achieve that, all points of the stroke are projected on a plane defined by the ends of the stroke.

There are infinitely many planes which pass through two points. The one which minimizes the distortion caused by the projection is finally chosen and designated as plane P. The distortion refers in this case to the summed average distance from the projected point to the approximated 3D line (see figure 4.28). It can be minimized by choosing the plane being orthogonal to the one defined by the camera position and the two stroke ends (see figure 4.29). The latter is computed with the following formula:

```
$L \leftarrow (J_2 - J_1) \times (C - J_1)$
$L \leftarrow L/\,||L||$
$Op \leftarrow J_1$
$Np \leftarrow (J_2 - J_1) \times L$
$Np \leftarrow N_p/\,||N_p||$
```

where J_1 and J_2 are the two joints connected to the stroke, C is the camera position, L is an auxiliary line which lies on the plane P, and O_p and N_p are the origin and the normal of the plane respectively. O_p and N_p are the parameters which define P uniquely. P has the following properties:

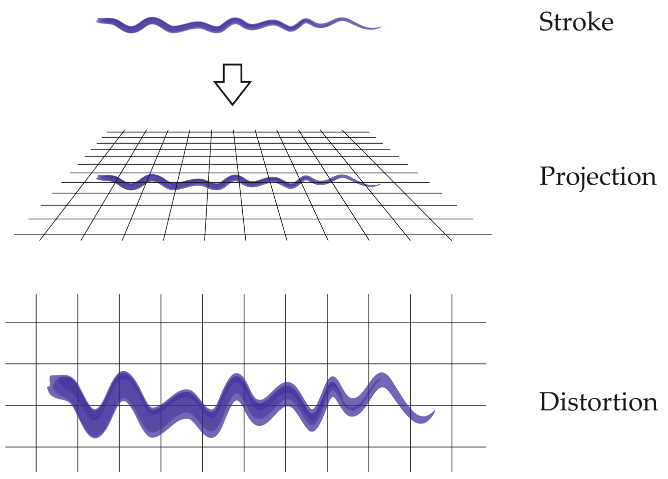


Figure 4.28.: Projecting imperfect strokes increases the error and sometimes causes visible distortion.

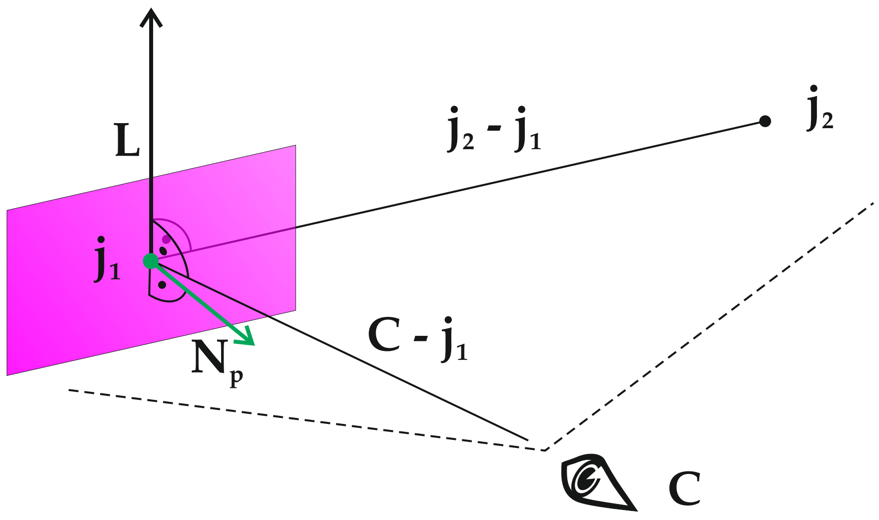


Figure 4.29.: Finding the plane P which minimizes distortion. C: camera; J_1 and J_2: two stroke ends; L: a line on P; N_p: the normal of P. Green quad: the plane P.

- Both joints J_1 and J_2 lie in the plane
- It is orthogonal to the drawing direction

All points of the stroke are then projected to this plane using simple ray-plane intersection. Notice that sketch points are represented as rays shot through pixels in the viewport.

4.4.4. GUI

The user interaction in the presentation mode is slightly different than traditional WIMP user interfaces can supply. The reason for this is the interaction medium - a large scale multi-touch screen. The interaction consists mostly of sketching, but also requires buttons, menus, and a means to change the sketching mode. To fulfill these needs, a specialized GUI framework was developed. The framework is largely based on QT, but implements some features unique for the presentation mode.

The following section 4.4.4.1 gives an overview of the framework and how this framework incorporates the design goals of CDP. This provides a more detailed insight into the implementation.

4.4.4.1. Visual appearance and interaction

One of the goals of CDP is to minimize the distraction caused by the system on the designer. The user interface should be therefore simple, smooth, and self-revealing [4]. It should be also clearly arranged and easily extendable.

At the beginning, the display is almost free of UI elements. The screen space is optimized for the standard sketching mode since this mode is intuitive, easily explored, and requires no additional interaction.

A click on a surface selects it and activates the extended surface mode. Clicking on another surface transfers the focus to the respective surface. Clicking the same surface again leads back to standard mode. The more advanced sketch modes are activated using the GUI.

Table 4.4 lists the three GUI elements which are always visible on the screen. All of them can be activated by "pulling them out". The two paper-edge elements have an additional icon to distinguish them from one another and to give a hint about their function.

When opening the sketch paper mode, a sheet of sketch paper slides over the screen and covers it, leaving small gaps at the top and bottom edges to remind the designer that this is a layer over the virtual 3D scene. In addition, there is a thumbnail bar at the lower end of the screen which contains snapshots of the sketches drawn on the paper. Each snapshot corresponds to one layer and clicking the thumbnail makes that layer the current layer. Each thumbnail has two buttons – one to apply (or "bake") the sketches to the scene and one to discard the layer. In addition, there is a slightly larger "bake" button on the top-left of the current layer (see figure 4.30).

The bar has also a button on the left which generates new layers. The newly created layers slide smoothly from the left. This way the visual appearance of the "new layer"

[4]"Self revealing" refers to the intuitiveness of a user interface. A user interface is self revealing if it requires no training before usage [71]

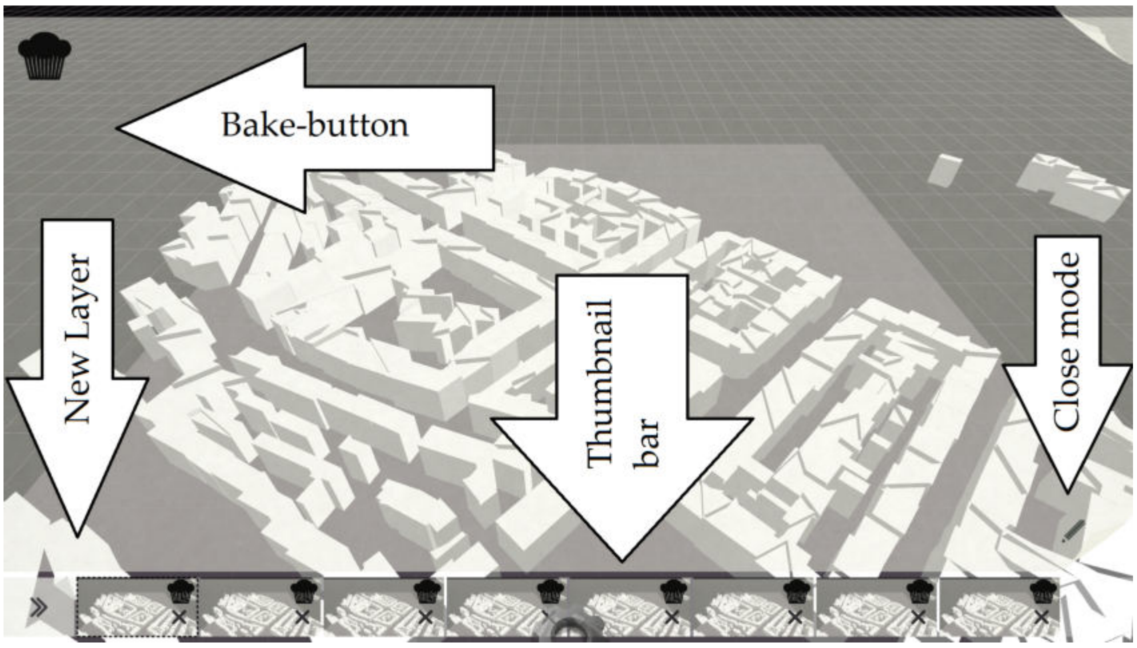


Figure 4.30.: The UI elements of the sketch paper mode.

GUI Element	**Location**	**Function**
Sheet edge with a pen-icon	Top-left corner	Activates the sketch paper mode
Sheet edge with a cube-icon	Top-right corner	Activates the 3D sketching mode
Cog-wheel	Centered in the lower edge, semi-hidden	Opens the settings menu

Table 4.4.: Main GUI elements and their properties.

button gives a hint about its function. The "new layer"-button, similarly to the paper edge which activates the sketch paper mode, is designed to mimic the appearance of real sketching paper, which is essentially a roll with one loose end. Pulling the end uncoils more paper from the roll.

The thumbnail bar can be swiped left and right to scroll over the thumbnails in case that not all of them could fit in the display. This way of interaction also mimics real sketch paper. Architects commonly overlay the sketch paper on top of other drawings to redraw or extend their content. In order to generate additional versions of the same drawing, architects slide the paper tape to a new position and sketch again. The result is a paper tape with different versions of the same sketch arranged in a row. The designer can slide the tape over the original drawing and compare the versions against each other. Figure 4.31 illustrates common usage of sketch paper in architecture.

Closing the *sketch paper mode* is done similarly to opening it – one has to slide the edge of the paper towards the screen boundary. By doing so, the program returns to the standard sketching mode.

The 3D sketching mode is started by pulling the paper edge at the *top right* of the screen. It has a similar GUI like the sketching mode, but it has no thumbnail bar in the current implementation.

The *settings menu* can be used to adjust program related parameters or to activate additional functions. In the current implementation, the following options are available:

- *Free camera*. This option will unlock all axes of the 3D mouse controlling the camera. It could be useful to people used to working with a 3D mouse and aware of its sensitivity.
- *Clear scene*. This option will clear all sketches drawn so far.
- *Eraser*. When this option is activated, a single click will erase the sketch strokes belonging to the clicked surface instead of triggering the "extended surface" mode.
- *Camera sensitivity* is a slide bar, which controls the sensitivity of the 3D mouse. Sliding it towards its maximum will increase the speed the camera moves with.

The GUI has no "turn off" button. The presentation mode is switched on whenever the 3D mouse is put on the table surface, and switched off as soon as the mouse is missing from the table for several seconds. Unfortunately, the current implementation of the fiduciary marker tracking module, which tracks the mouse, is slightly unstable and sometimes cannot find the camera marker. This causes an unexpected shutdown of the presentation mode. To avoid this problem, the presentation mode was programmed to run until the whole system is shut down. This setting will be changed in the future when a more stable fiduciary marker tracking algorithm is developed.

4.4.4.2. Implementation overview

This subsection gives some implementation details about the GUI framework used to implement the user interface presented in this thesis. The framework consists of three classes:

- Element class

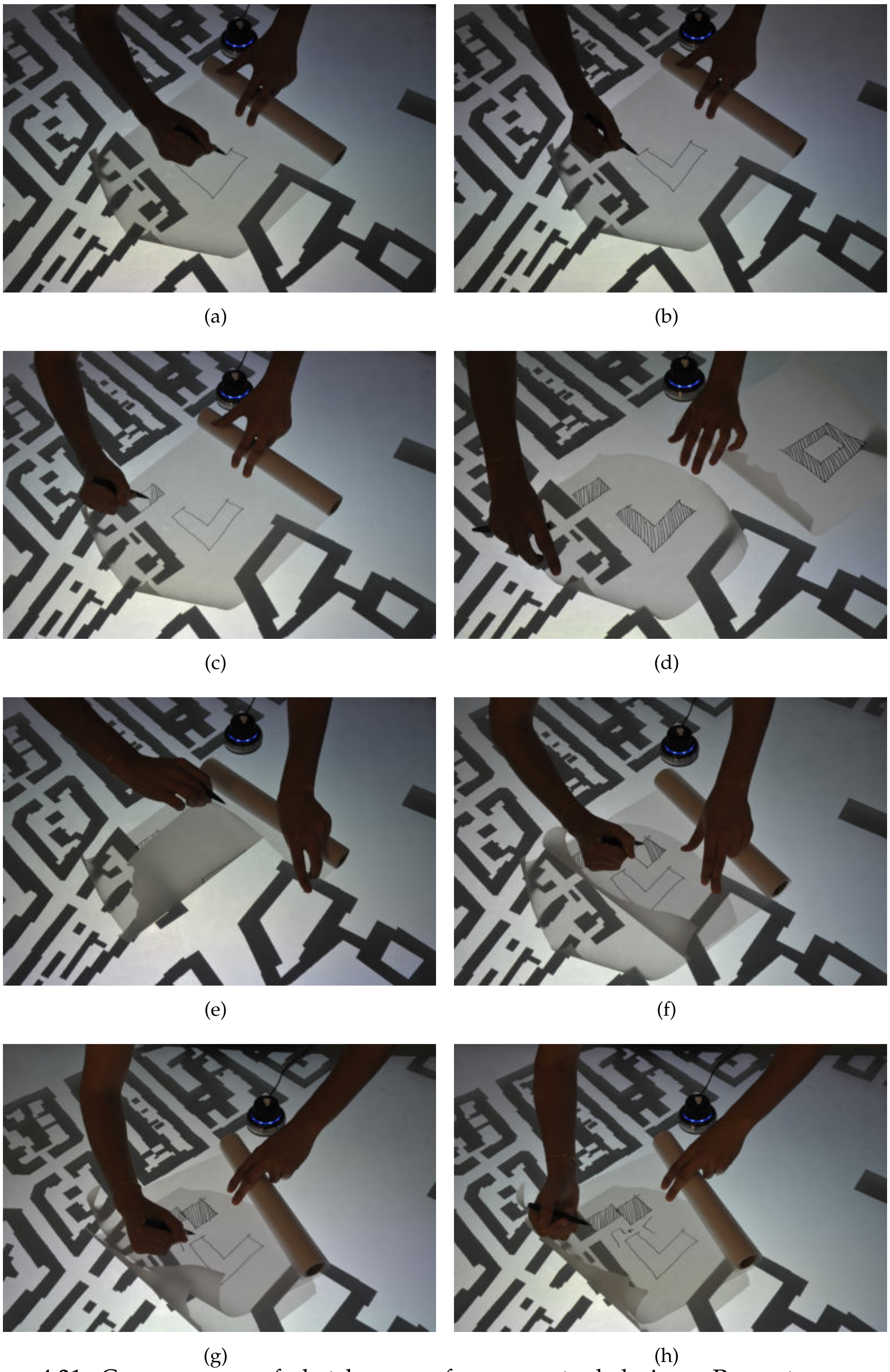


Figure 4.31.: Common use of sketch paper for conceptual design. Paper transparency makes it ideal for layering and versioning. Changing the sheet is easy, one has to unveil the roll further.

- Container class
- Menu class

An *element* is the base class for all GUI elements – it handles mouse and keyboard events, and can gain or lose focus. It is also the basis of the GUI hierarchy which arranges elements in a parent-child relation tree. The element's parent can serve as a coordinate system origin, causing an element to move together with its parent. For example, buttons in the settings menu move together with the menu when it slides across the screen.

A *container* is a specialized type of element which can hold an arbitrary number of elements (children). The children are organized in a list which defines their order of appearance. This order is also used to capture mouse events – elements which are displayed on top are transitioned first when propagating a mouse event. The framework checks each element in a container for collision with the mouse pointer using this ordering. If the collision detection returns "true" for a given element, the event is passed to it. Each element can choose if it will consume the event or pass it further to other elements. For example, the thumbnail bar of the sketch paper mode can be used to select a thumbnail or to slide the bar. If the detected gesture is a "drag", the bar slides in the drag direction and consumes the event. If the gesture is a "click", the bar passes it further to the clicked thumbnail image.

A *menu* is a specialized type of container. Menus can be shown and hidden with an animation. The animation mechanism which is used is simple interpolated morphing with two frames [72]. The animation is defined by two key-frame images (one for a hidden and one for an open menu respectively), start and end transparence values, and time interval between the two frames. The result is a smooth fading from one appearance to the other, simulating movement. It is possible to extend this concept to full key-point animation if necessary.

The menu class is used not only for the settings menu but also for the sketch paper mode and for the 3D mode. The appearance of these modes closely resembles that of a menu – they are pulled into the screen, have to be animated, and contain further UI elements.

There are also other GUI elements which extend the base *Element* class. These include:

- Button
- Slider (has values between 0.0 and 1.0)
- Switch button (has "on" and "off" states)
- Drag button; this button can be dragged and influence the position of another UI element; it is used to model the paper edges activating the additional sketching modes.

The sketch paper layers and the thumbnails used in the sketch paper mode also extend the base UI elements class.

4.4.5. Camera control

The camera is controlled by a 3D mouse [48], which has a fiduciary marker glued on its bottom. The fiduciary marker is used to track the position and orientation of the camera on the table top, while the 3D mouse controls the height and forward tilting of the camera.

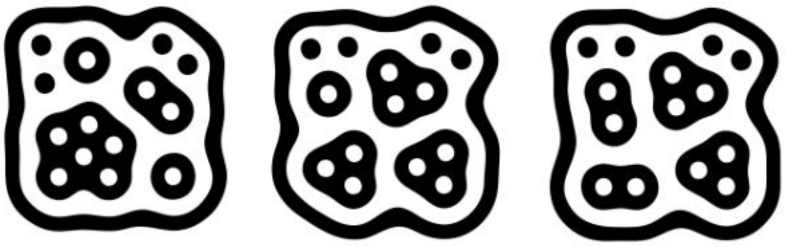


Figure 4.32.: The "amoeba" fiduciary markers used in reacTIVision [7].

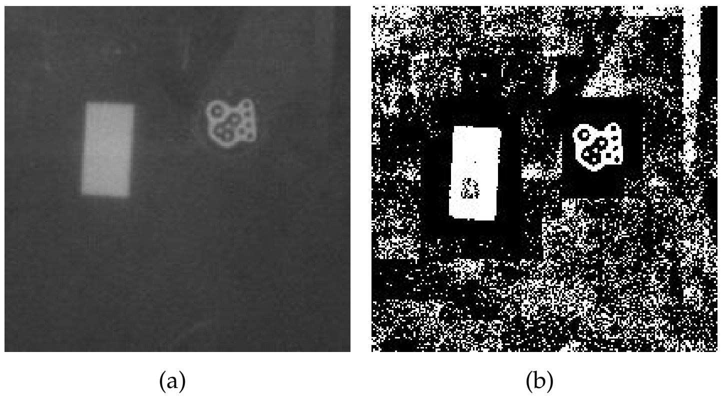


Figure 4.33.: Berensen threshold. (a) The source monochrome image; (b) The binarized image after performing a Berensen threshold.

4.4.5.1. Tracking the fiduciary marker

The algorithm used to track the fiduciary marker is taken from the open-source project ReacTIVision [7]. The fiduciary marker tracking library used there is called "libfidtrack" and has several advantages over older implementations, such as d-touch [20]. For instance, it is up to 16 times faster, supports fiduciary markers with 50% smaller size, or alternatively a larger set of markers as compared to d-touch. The fiduciary markers used by "libfidtrack" are called "amoeba" (see figure 4.32) and are generated according to a technique called "left heavy depth sequences" [19]. The relation between a marker and its identification number is computed based on the topological structure of the marker, which results in a better resistance against noise and distortion. Furthermore, it is less probable that two markers will be recognized as one and the same marker.

Libfidtrack uses an adapted Berensen thresholder [73] to binarize the image (convert it to black and white pixels) – see image 4.33. Then, a segmentation step is performed, which constructs an adjacency graph containing all black and white regions. This graph is traversed and searched for fiduciary markers which have a specific topology. Finally, the position and orientation of the fiduciary markers is computed. Libfidtrack can also handle fiduciary markers which are only partly detected or distorted due to fast movement.

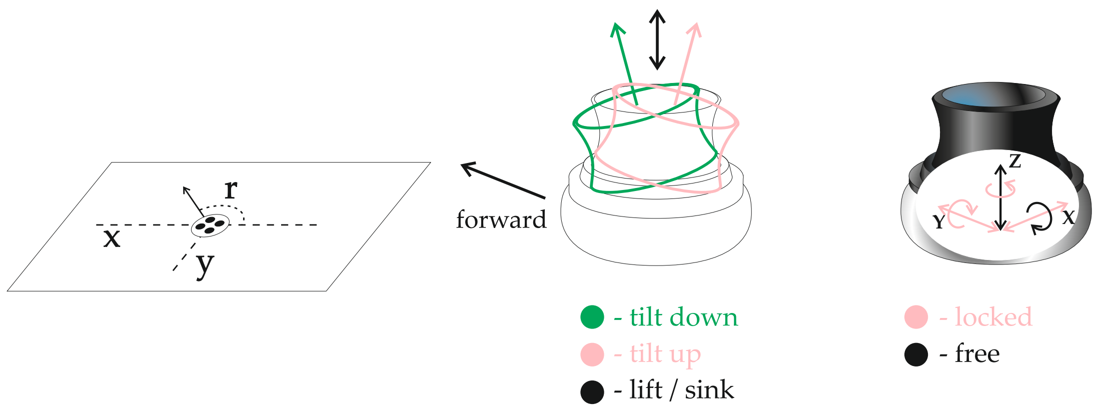


Figure 4.34.: Fiduciary marker and 3D mouse used for camera control. Left: 2D position and orientation (x, y and r) of the camera is determined by the fiduciary marker on the table surface; Middle: permitted directions of movement of the 3D mouse to control the camera; Right: free and locked axes of the 3D mouse controller.

4.4.5.2. Navigating with the 3D mouse

The implementation of the camera uses 3Dconnexion [48], a popular 3D mouse used frequently for navigation CAAD applications. Although it provides a complete freedom of movement (all 6 DOF in 3D space), it takes time to get used to precisely navigate with the mouse. Since one of the requirements of the CDP is to be usable by novice users and people without background in architecture, the mouse usage was simplified.

Only two axes of movement were used – one translational (the "z", or "up", direction) to lift or sink the camera, and one rotational (the rotation around "x", or "side") to tilt it. Figure 4.34 illustrates the permitted movement directions. The other four axes remain locked, unless the designer explicitly set them free. This way, everyone can use the mouse in an intuitive way but it is also possible to customize it for more advanced users.

This approach has a small disadvantage – the 3D mouse is wired via USB cable to the computer, and the cable lies on the top of the table, which could be misinterpreted as user interaction. In practice however, the cable was never an issue and participants in the evaluation later barely noticed its presence.

4.4.5.3. Alternative implementations

There are alternative ways to implement a camera in an augmented virtuality environment such as CDP. One alternative would be to put the camera on a stand with several flexible joints which can be twisted and bent until the desired camera orientation is reached. The tip of the construct would hold a marker. The position and orientation of the marker would be tracked by a camera above the table (see figure 4.35). This is a valid and likewise intuitive implementation of a virtual camera. However, it is far more difficult to implement and it involves the design of additional hardware.

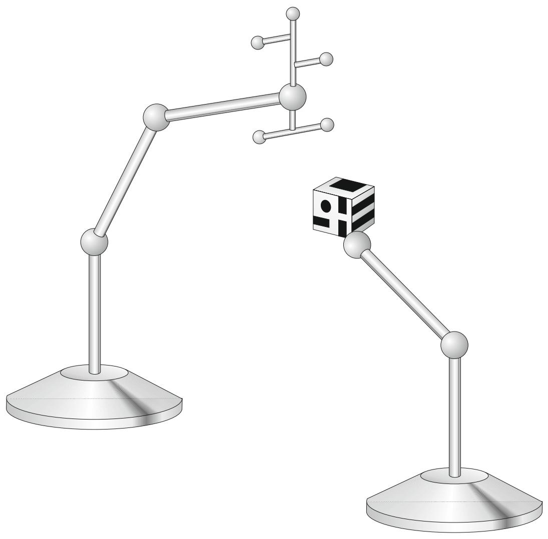


Figure 4.35.: An alternative camera implementation with 3D markers on a flexible arm.

It is also possible to implement camera control entirely on the table top. The user would move and rotate the camera with finger gestures and there would be an additional slider to adjust the vertical position. This approach is simpler to implement, but it is also less convenient. The designer has to redirect his attention to another user interface every time he needs to move the view, which is more demanding than just moving a physical object.

4.5. Bugs, problems, and performance issues

The current implementation has surprisingly only a few bugs, considering the many libraries, tools, and methods used within the project. This chapter points to some of the fundamental problems encountered during the development to which no direct solution was found.

4.5.1. Shared OpenGL contexts

The presentation mode uses two libraries which are based on OpenGL and render to the screen – VL and QT. VL renders the 3D content of the scene, while QT draws the GUI elements on top of it. Consequently, these libraries need to share a common OpenGL context in order to render to the same frame buffer. Although QT supports shared OpenGL contexts, which come from other applications, it was extremely difficult to get this feature to work. The final solution used a graphics context generated by QT and adopted by VL. Both libraries had to be adjusted accordingly:

- QT has to make sure to reset all OpenGL rendering states before the end of a rendering cycle

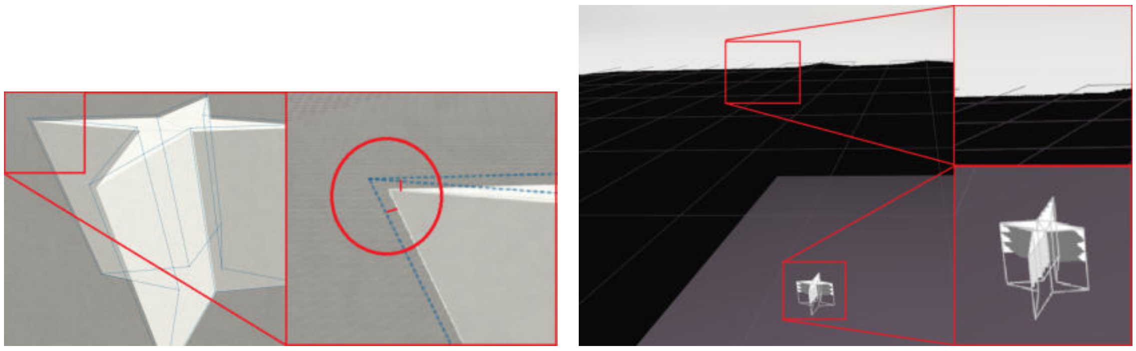

(a) (b)

Figure 4.36.: The two types of computational errors which depend on the chosen metric; (a) numerical errors; (b) depth artifacts.

- VL has to call special QT methods prior to and after rendering the GUI – beginNativePainting() and endNativePainting(), respectively

Even with the proposed work-around, it was impossible to use native QT GUI elements and all GUI classes had to be implemented from scratch.

4.5.2. Numerical instability and depth fighting

A seemingly unimportant property of data representation is the unit metric (i.e. what is the corresponding length of one virtual unit in the real world?). One could argue that choosing the scale is merely a cosmetic matter. However, this property introduces two major computational problems – *numerical instability* (if the scale is too small) and *depth fighting* (if the scale is too large).

Numerical instability refers to the limitations imposed by floating point arithmetic on modern computers. A small value transformed several times with matrices which also have small values results in a visible offset on the screen (see figure 4.36a). Using numbers with double precision might solve this problem, but would also slow down the computation. Moreover, graphics hardware is optimized for single precision calculations. Thus, coordinate values should not be too small.

The coordinates cannot be large numbers either. The reason for this can be traced back to the way conventional graphics hardware works. In order to resolve polygon ordering in the scene, the rendering pipeline utilizes a so-called "depth buffer" [74]. The depth buffer stores depth values of rendered polygons and uses them to discard polygons, which lie farther than what was already rendered. This ensures that polygons are rendered in the correct order and overlapping polygons are showed correctly. The depth buffer technique works very well and has the advantage of speed, but fails in case of almost-co-planar polygons.

The presentation mode superimposes sketches over surfaces, but also surfaces over other surfaces. This causes so-called **depth fighting**, if the displayed primitives are coplanar [74]. Depth fighting, or "z-bleeding", is a common phenomenon in 3D computer graphics which produces unpleasant visual results (see figure 4.36b). A work-around to this problem is the OpenGL extension "polygon-offset" [75], which modifies the depth values of selected polygons, so that these do not "bleed through". This technique works only if the unit distance between polygon vertices is relatively small compared to the size of the camera frustum (more precisely, the distance between the near and the far clipping planes).

Using larger coordinate values increases the absolute distance between vertices, which causes z-bleeding. The bigger the scale, the more visible are the depth artifacts. Therefore, a sufficiently small scale had to be chosen.

A balanced metric system was found empirically:

```
1 virtual unit = 1cm
```

This metric reduces the numerical error to less than a pixel (such that rendered objects appear on their correct position). The z-bleeding effect on the other hand is not entirely eliminated but it appears only on distant polygons far into the horizon, which are barely visible.

4.6. Unimplemented features

For various reasons, some of the features presented in the concept were not implemented in the presentation mode and remain subject to further research. The following list summarizes unimplemented topics.

- Visual synchronization between the CDP table top and the presentation mode (i.e. sketches drawn in the presentation mode to appear on the table surface).

  Currently, the CDP is in a protypical stage and its software framework is in a preliminary transient state. Therefore, the software framework has no unified semantic data model yet. Effort is made to create such model and to implement consistent interfaces, which will enable transparent data flow between the components of the CDP, including the presentation mode. As soon as the software model supports this kind of data flow, the presentation mode will be synchronized with the table top.

- Visualization of data produced by the simulation applications (e.g. simulation of sunlight).

  Currently, simulations produce only 2D data. Although it was possible to visualize the 2D data in the presentation mode (i.e. as a texture on the city plane), it was considered a better solution to transform the simulations to produce 3D data first, and then work on a solution that visualizes the 3D data. Creating simulations which produce 3D data depends on the software framework (see above), thus it is also a part of future research.

- Unlocking the camera (so that it can be navigated solely using the 3D mouse).

  This feature raises additional conceptual problems which could disrupt the consistency of the presentation mode. "Freeing" the camera implies that it is no longer bound to the fiduciary marker on the table surface. Consequently, the relation between the physical controller and the virtual camera is destroyed, as the virtual camera no longer corresponds to the physical position of the 3D mouse. This feature could confuse users and requires further conceptual clarification (end probably a user study) before implementing it.

- Sketching on the table top.

  Sketching on the table top is a separate interdisciplinary project developed by Saburo Okita. A special pen is used which emits infrared light with a specific frequency to let the table distinguish between finger gestures and sketching on the table surface.

5. Evaluation

The presentation mode was evaluated in few pilot case-studies which will be presented in this thesis. The goal of the evaluation was to measure the usability of the CDP system, in particular, the usability of the presentation mode in collaborative conceptual designing.

The usability of a tool for the aims of conceptual design in architecture depends to a great extent on psychological factors, such as personal preferences, experience, and feeling of comfort. Hence, usability appears to be subjective. Given that designing is an individual process, it is not sufficient to assess the usability of a tool by simply calculating the average feedback of all participants in the evaluation tasks. For this reason, the evaluation tasks in this thesis strive more towards qualitative than to quantitative feedback. One should consider, however, that a comprehensive answer to the question "How usable is the presentation mode?" could be provided when the system is tested by more users in later, more advanced stages of its development. Here, our attention is focused on evaluating the most frequently used features of the system by users and how well users interact with the system.

5.1. Design of the evaluation

5.1.1. Evaluation criteria

The hypothesis of the evaluation was that the new tool for visualization and sketching does not impede the creative thinking of the architect; furthermore, it helps discover and express new ideas with ease. It is difficult to prove this hypothesis with significance due to the limited number of cases analyzed here and the missing control – another established mixed reality designing tool – for plausible comparison to the presentation mode of CDP. A comparison with the traditional pen-paper approach is not plausible, since creativity and spatial cognition differ significantly for strictly physical and for augmented design tools [76].

Thus, the aim of the evaluation presented in this thesis was to explore and provide rationale for this hypothesis by means of the following test criteria:

- Ability of the user to recognize the correspondence between the mixed reality environment on the multi-touch table and the presentation screen
- Easy and intuitive navigation using the tangible camera
- Switch frequency between the sketching modes

The first two criteria are subjective and could not be quantitatively measured. Hence, architects had to give their verbal feedback for the analysis of these criteria. The last criterion was automatically measured by the software. Switch frequency refers to the rate with which users need to switch from one sketching mode to another.

We assume that these three criteria influence the concentration of the user on his actual task of designing. The more time and efforts is invested into understanding the system and interacting with its user interface, the higher the probability that architects could diverge from their current activity or even forget it. This is exactly the effect we wanted to test here.

In addition, the usage duration of each of the four sketching modes was recorded (only the total duration and not the individual sessions spent with each mode were recorded). This data was used to evaluate which kind of interaction is most natural and desired by architects.

5.1.2. Participants, setup and task

A total of 10 students in architecture participated in the evaluation. In groups of two students, they had to solve a given task together. The concrete setup used in the evaluation and the formulation of the task given to the participants were as follows.

A static figure-ground diagram [footnote: a map where buildings are colored in black and the background is white] of a city is loaded on the multi-touch table. The scene is also displayed on the vertical screen in 3D. The 3D scene contains both virtual and physical objects. The participants can sketch directly in the virtual scene. Both pens are assigned the same color in order to obtain a collaborative concept.

In addition to sketching, physical models can be cut from styrodur foam using a styrocut device and utility knife. The goal is to create collaboratively a conceptual design of a living area. The exact formulation of this task (the one provided to the students) is as follows:

> *"Design the concept of a living area on the provided free urban space within the Mariahilfplatz in Munich. The objectives are to explore the initial distribution of space and volume, as well as basic ideas about the design of the facades. As supplementary tools you can use the provided materials and the interactive sketch tools based on the CDP platform. Mind the principles of urban design, the internal relationships of the buildings to each other, as well as the design of the facades. The height of all buildings has to be 20 meters*[1]*.*
>
> *Let your ideas flow freely, and try to ignore the technical limitations of the system in its development stage.*
>
> *The quality of the design is not part of the evaluation."*

5.1.3. Dependent variables

During the evaluation, the following information was gathered:

- Which activities are performed on the table surface and on the presentation screen, respectively?

[1]The limitation that all buildings have to be 20 meters was dictated by the fact that the 3D object recognition was in a development stage during the evaluation of the presentation mode. Therefore, it was not possible to scan the precise height of the buildings. To compensate for this technical difficulty, all objects were assumed to be of height 20m.

- Is the camera view aligned with the view of the table surface? Do the users have difficulties to recognize the correspondence between the two displays?
- Where do users look when moving the building blocks – at the blocks on the table top or at the presentation screen?

In addition, the use duration and frequency of switching between the sketching modes were recorded as part of the evaluation.

5.1.4. Interview

At the end of each evaluation session, the participants of each group were interviewed. The questions and answers of the interview are presented in section 5.3.3. The participants were also asked to fill in a standard SUS survey (System Usability Scale [77]) and an AttrakDiff survey [78]. The SUS measures the overall usability of a system in a "quick-and-dirty" fashion, without considering the context it is used for. AttrakDiff measures the attractiveness of interactive systems by exploring their pragmatic and hedonic properties. AttrakDiff consists of numerous word-pairs, each one representing the extreme opposite. Each word-pair provides rating on a 7-point Likert-scale.

The results of both surveys as well as the data obtained about mode usage and switches between different sketching modes are presented in the following chapter. The results obtained by verbal communication with the participants are summarized in chapter 5.3.

5.2. Quantitative feedback

Quantitative feedback refers to the (quantitative) data collected from the SUS and AttrakDiff surveys, the usage duration of the sketch modes, and the switch rate from one sketch mode to another.

5.2.1. Sketch mode usage and number of switches

The data demonstrating how long each sketching mode was used and how often the users switched from one mode to another was recorded automatically by the software of the presentation mode. No distinction between the participants within a group could be done in this regard as the system cannot detect who is sketching at a certain time. Furthermore, the system records only active usage of the sketching modes and neglects idle states. A sketching activity is considered active in the time span between touching the screen surface and lifting the pen up. A switches is defined as the transitions from an arbitrary sketching mode to a different sketching mode. Only four groups were measured by the system because of technical difficulties.

Figure 5.1 shows the duration of use for each of the four sketching modes (by user group) in seconds. Data about four user groups are displayed. The results indicate a high deviation between the groups regarding time of use of different modes. In addition, mean percentage of usage duration per sketching mode was calculated (100% = total time spent for drawing). The lowest mean value was observed in the case of the 3D mode (8% total usage). This can be explained with the low level of stability of the 3D mode at the time of

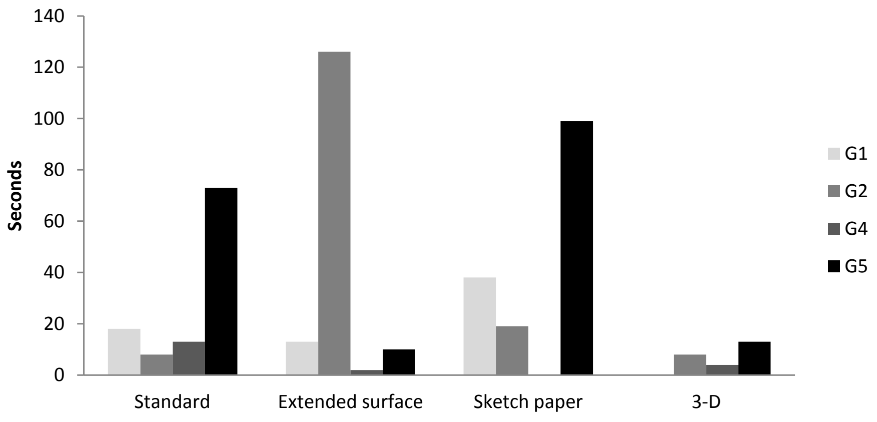


Figure 5.1.: Usage duration of a sketching mode by each group (G1, G2, G4, and G5)

evaluation. Users were disappointed when the 3D mode failed to recognize the original 3D shape of the drawn sketch. The mean values for the other modes were as follows: standard mode – 34%, extended surface – 28%, and sketch paper – 29%. For all sketching modes, a high deviation from the mean values was observed. (standard mode: 34% total use, SD: 26%; extended surface: 28% total use, SD: 34%; sketch paper: 29% total use, SD: 28%; 3D mode: 8% total use, SD: 9%), which points that all five groups taking part in the evaluation are likely to have very diverse preferences regarding the use of sketching modes.

The total execution duration of the assigned task was about 30 minutes. The sketching modes were changed in average 27.75 times (SD: 14.81) during the execution. This corresponds to about one switch per minute. The high standard deviation indicates that some participants in the system evaluation felt more often the necessity to change the sketching mode than others.

5.2.2. SUS

The SUS survey was evaluated using the scheme given in [77] where a score of 100 indicates the highest usability and score of 0 indicates the lowest general usability of the system. Figure 5.2 presents the results of the SUS survey by questions. Answers indicting stronger agreement in the case of odd-numbered questions (Q1, Q3, Q5, Q7 and Q9) increase the usability score, while answers indicating stronger agreement in the case of even-numbered questions (Q2, Q4, Q6, Q8 and Q10) decrease the usability score of SUS. This dependence belongs to the design of the SUS where odd-numbered questions are positively formulated and even-numbered questions are negatively formulated.

The mean usability score calculated from the SUS data of all participants in the evaluation was 85.83 (SD: 9.27), pointing to relatively high usability of the presentation mode. The results on questions 8 and 9 indicate a need for improvement towards better user experience and increasing the confidence of users when working on the CDP and its presentation mode. The higher variance in the answers to question 4 could be explained with the different levels of experience of the participants. The rest of the questions indicate a good usability of the presentation mode.

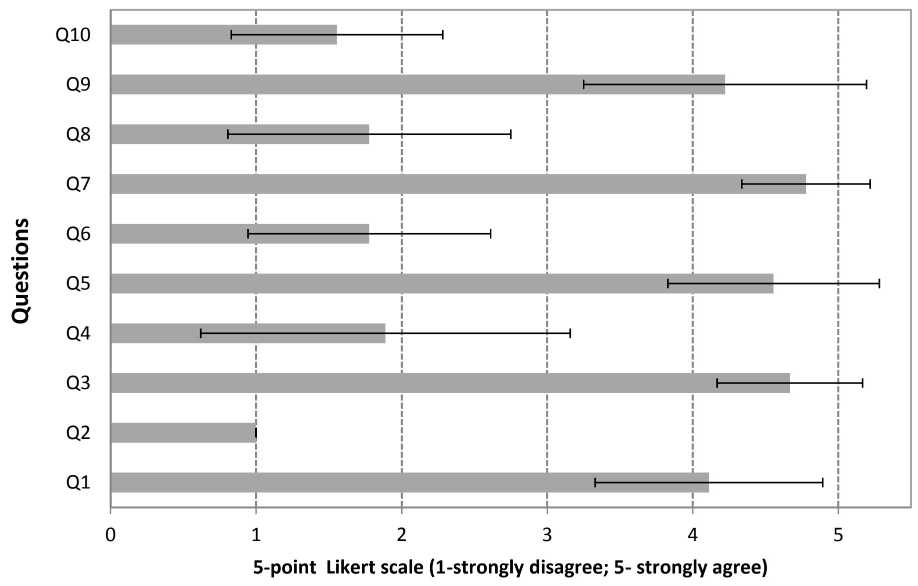


Figure 5.2.: Average response to the SUS questions. List of questions:
Q1: I think that I would like to use this system frequently
Q2: I found the system unnecessarily complex
Q3: I thought the system was easy to use
Q4: I think that I would need the support of a technical person to be able to use this system
Q5: I found the various functions in this system were well integrated
Q6: I thought there was too much inconsistency in this system
Q7: I would imagine that most people would learn to use this system very quickly
Q8: I found the system very cumbersome to use
Q9: I felt very confident using the system
Q10: I needed to learn a lot of things before I could get going with this system

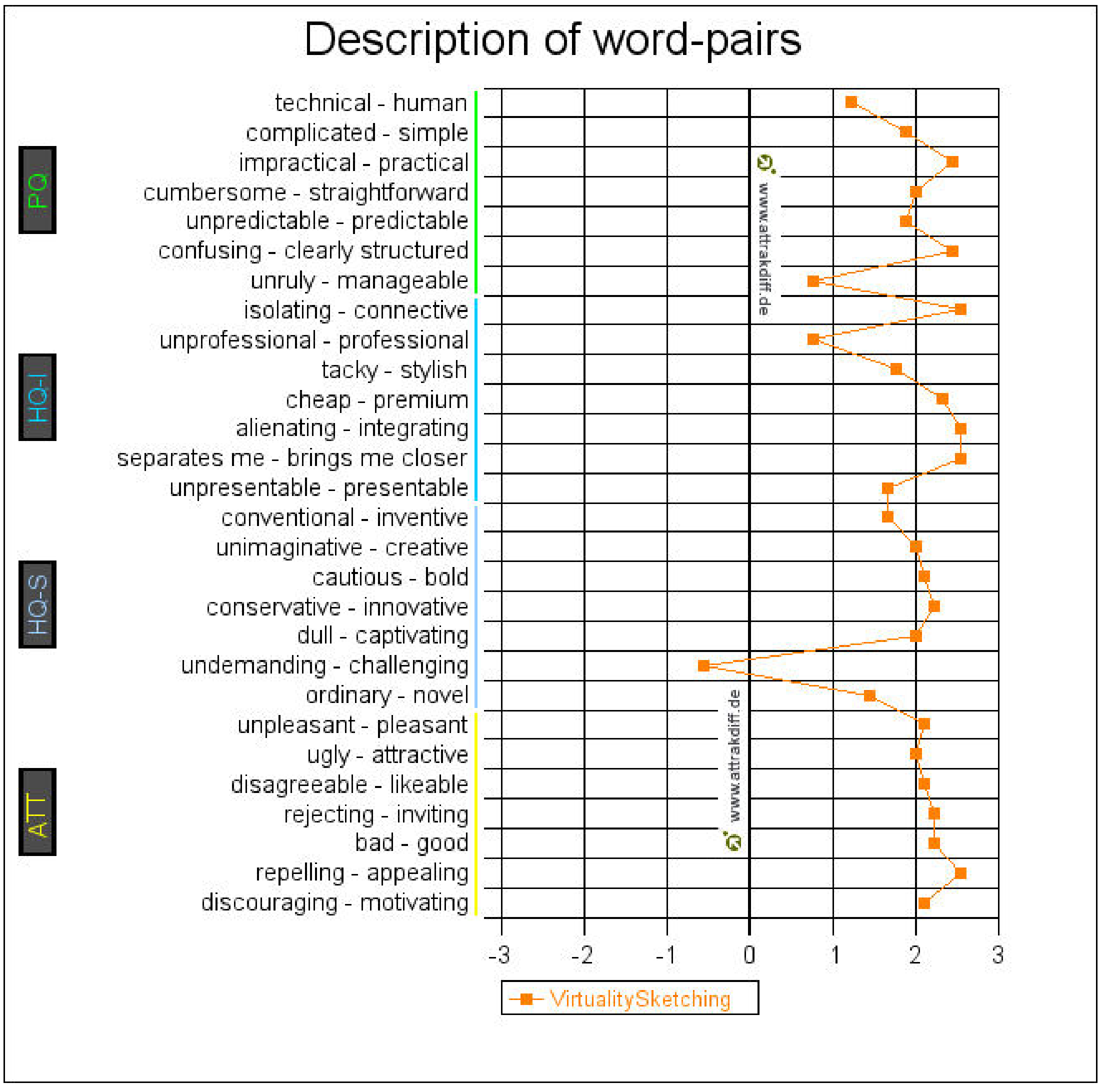


Figure 5.3.: AttrakDiff average values of the presentation mode assessment.

5.2.3. AttrakDiff

The results of the AttrakDiff survey were processed on the official website of the survey [78]. With AttrakDiff the user ranks 28 properties of the product on a 7-point Likert scale ranging from -3 to 3. The properties are described by 28 word-pairs presented in figure 5.3. The results of the AttrakDiff survey are shown in figure 5.4 as graphical presentation of mean values of the presentation mode properties assessed by all participants in the evaluation.

The 28 properties are grouped in four categories:

- Pragmatic quality (PQ)
- Hedonistic quality – identity (HQ-I)

- Hedonistic quality – stimulation (HQ-S)
- Attractiveness (ATT)

The first category relates to the usability of the product and how well it assists users in their task. The second and third categories describe how well users can identify themselves with the product and how well the product stimulates users' need to improve their skills and be more productive, respectively. The last category of questions measures the general product attractiveness to users.

According to the average values of the AttrakDiff data, the presentation mode is clearly structured, practical, and simple to use regarding its pragmatic quality (figure 5.3). On the negative side, it seems that users had slight difficulties to cope with the system (see word pair "unruly-manageable" (figure 5.3). This could be explained with the insufficient calibration of the multi-touch table which caused some objects and the camera to shake sometimes.

Regarding its hedonic quality, users perceived the presentation mode as innovative, stylish, and bringing users together (figure 5.3). However, there is a demand for more challenge and professionalism (see word-pairs "unprofessional-professional" and "undemanding-challenging" (figure 5.3).

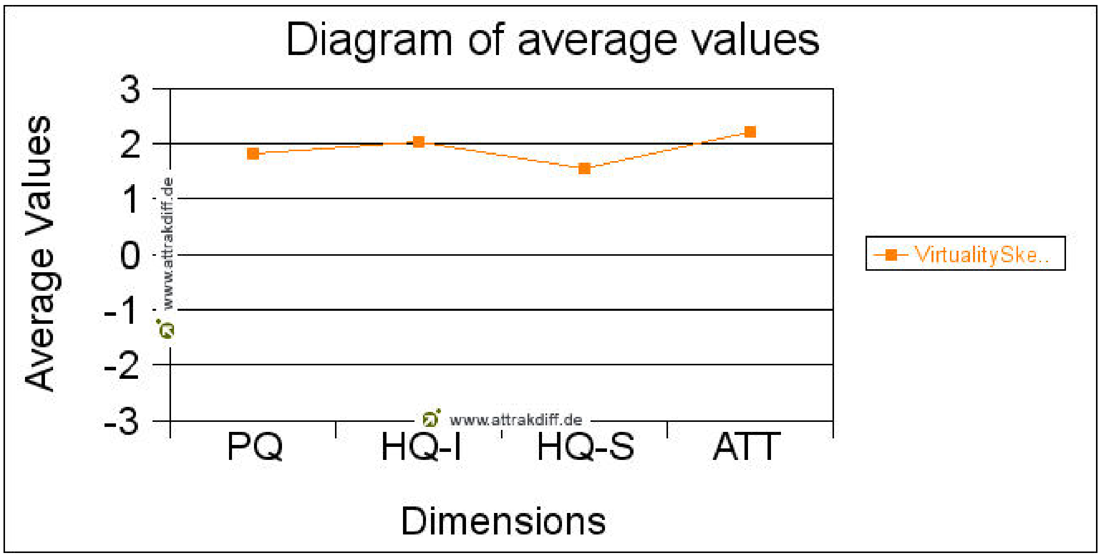
By analyzing AttrakDiff data, mean values of the pragmatic and the hedonic quality are plotted in a graph to shape the compound desirability of the product. A confidence interval based on the homogeneity of the values is also computed. The confidence rectangle corresponds to the certainty of the obtained result. Figure 5.4(a) shows the respective AttrakDiff data obtained for the presentation mode. The participants evaluated the product as highly attractive. The position and the size of the confidence rectangle in figure 5.4(b) indicate the product is highly desired with good confidence. But there is also a room for improvement towards better pragmatic quality and better hedonic stimulation of the presentation mode.

5.3. Qualitative feedback

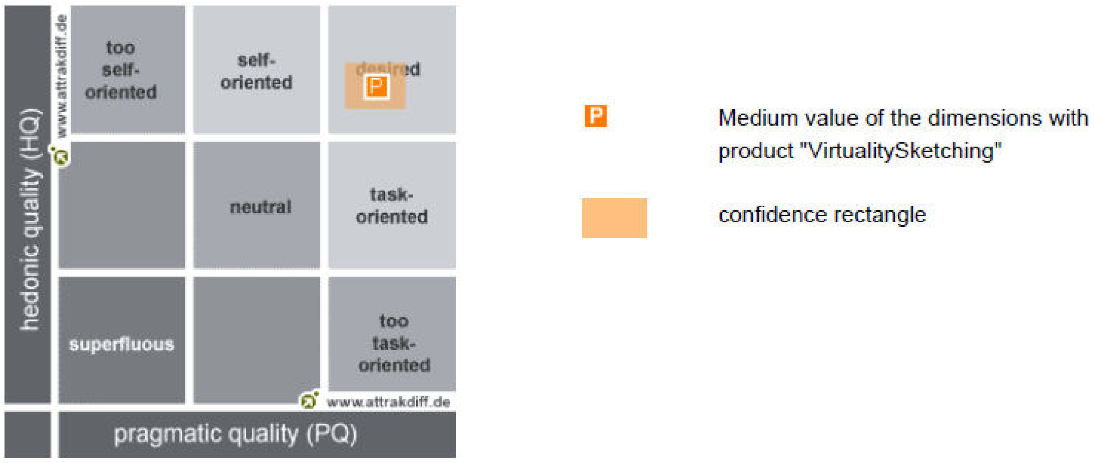
This section summarizes the qualitative feedback (direct and indirect) received during the evaluation of the presentation mode. This information was gathered by means of:

- observing user activities, difficulties, and communication during the task performance (indirect feedback)
- gathering remarks and suggestions of the participants during the task performance (direct feedback)
- interview with all participants after completing the task

During the interview, both direct and indirect feedback was collected. For example, during the interview some users came across to ideas being provoked by a particular question, but not directly related to the answer of the question. These were analyzed as direct feedback. An example for indirect feedback collected during the interview is when users started playing with the presentation mode while answering a question. To present the



(a)



(b)

Figure 5.4.: AttrakDiff results for the presentation mode. (a) Diagram of average values representing the average score of each of the four AttrakDiff categories. (b) Medium value and confidence rectangle based on PQ and HQ data of the presentation mode.

qualitative feedback consistently and avoid confusions, all results are grouped in three categories – indirect feedback, direct feedback, and interview – and presented in the following three sections – 5.3.2, 5.3.3, and 5.3.1 respectively.

5.3.1. Indirect feedback

Observing activities and behavior of the participants during the execution of the assigned task aimed to provide indirect information about how users perceive and interact with the system.

Most participants started prototyping by cutting models from the styrodur foam. Once they had a first version of their model, they put it on the table top started sketching on the presentation screen. One group took the inverse path, first sketching an outline of the model on the presentation screen and then cutting it out from styrodur foam.

The sketching modes were sometimes used differently than initially intended by the presentation mode. For example, users sketched two different versions of a single idea, overlapping these two sketches while using the standard mode, even though they could have used two different layers for each sketch within the sketch paper mode. Another example was activation of the extended surface mode in cases when the drawn sketch did not exceed the boundaries of the selected surface.

Participants often sought collaboration while working together. For example, they often tried to sketch simultaneously on the digital model [2]. Two groups divided the work – one user was sketching on the presentation screen while the other one was cutting styrodur shapes.

A few activities were not successful when using the presentation mode. For example, buildings with closed inner yards were recognized as if they had no yard. Also, selecting a surface failed sometimes; in these cases, the system recognized the click-gesture by mistake as a short stroke and did not mark the selected surface. Furthermore, two groups had difficulties with the thumbnail bar of the sketch paper mode. Sketching on the bar failed as the system interpreted the strokes as an attempt to scroll the thumbnails.

Some user expectations could not be optimally fulfilled by the system. All groups remarked the lack of precision in object recognition and placement of objects. In addition, digital buildings shook slightly sometimes, while their physical counterparts stood still. The system performed even worse when buildings were placed near the edge of the table. Such technical limitations often frustrated the users and impeded their work.

All participants had almost no troubles with learning to interact with the system. They tend to act intuitively and tried to apply features which were not implemented. In other cases, feature implemented in the system were not intuitively found and used. For example, participants often tried to zoom into the screen – a feature which is not directly supported by the screen but enabled by moving the camera in the desired direction.

All participants were fascinated from the system's ability to scan and digitalize objects interactively. The option to draw directly on the virtual world was accepted with satisfaction and curiosity. Users highly appreciated that the sketches did not disappear when moving the underlying building; moreover, the sketches move along and remain connected to the building.

[2]This produced scrambled sketches, because the presentation mode supports input from only one user at a time

5.3.2. Direct feedback

Each participant was given the possibility to share his own ideas for new features of the presentation mode or suggest changes for improvement of its functionality. The most interesting and novel ideas of the participants are summarized in this section.

A major part of all suggestions concerns additional utilities for editing and manipulating content in the presentation mode. All groups criticized the lack of different colors and stroke sizes for the pen. Three groups demanded implementation of the widely accepted "undo" and "redo" options. Furthermore, four groups suggested the option to export and import data, to take snapshots of the current workspace and to organize these snapshots in a consistent versioning mechanism. Two participants shared the opinion that a platform for sharing conceptual information in a collaborative scenario should be able to store the generated ideas consistently and methodologically, so that the collected ideas can be reviewed later. They suggested to achieve that by storing content changes cumulatively in a branch-like fashion.

Another set of suggestions referred to an improved semantic classification of buildings. Assignment to multiple colors, for example, to designate different types of buildings (living, industrial, commercial...) was suggested by three groups of participants. Their idea was that the system may provide different visualization styles based on semantic information as well. In addition, two groups suggested an option to hide all buildings temporarily and show only the urban terrain.

Four groups liked using the sketch paper mode, but lacked some beneficial features. Two groups proposed that the layers can be stacked on top of each other, arranged in a specific order, and even grouped semantically, similar to CAAD software. Users associated the sketch paper mode with pictograms frequently used in conceptual design in architecture. It was suggested the sketch paper mode can be used to create such pictograms and place them permanently into the virtual world.

A demand for less constrained and more customizable camera control was indicated. Some of the ideas were to create and use several modes for the camera; for example, a pedestrian mode and a "free-flight" mode in addition to the standard mode. Three groups asked for an option to instantly fly the camera into a "bird's eye" perspective to enable a view on a large part of the scene.

Three groups suggested an elaborated volume manipulation. One participant proposed an option for creating cavities into existing objects by drawing a contour and "cutting" the contour into the volume along a given direction. Another idea was to pull facades and surfaces along the direction of their normal vector in order to adjust an existing building, similar to SketchUp by Google [79].

All participants felt constrained that they couldn't sketch on the table surface. Participants of two groups lacked the option that more than one user can draw on the presentation screen simultaneously. Three groups also felt the absence of a compass and a precise indication of scale. One of the suggestions was to visualize these dependencies on the table surface, similarly to land maps.

One of the participants proposed two operation modes for the CDP – "basic" and "expert" – in order to optimally fit the requirements of both unprofessional and professional users.

5.3.3. Interview

The interview comprised nine questions, grouped into three themes. In the following, the questions together with the most frequently given answers are summarized.

Theme 1: GUI and sketching

> *Q1: What was positive or negative about the sketching modes – "standard", "extended surface", "sketching paper", and "3D mode"?*

In general, participants liked the user interface provided for sketching. The most positive impressions were expressed for the sketch paper mode and the 3D mode, despite that the 3D mode was not in a mature state during the evaluation. The users pointed out some missing functionality as well, such as "undo/redo" and an option for control of pen color and stroke width.

> *Q2: What was positive or negative about the way how each mode is activated?*

Users found the switching mechanisms convenient and intuitive. The only flaw concerned the transition between the standard mode and the extended surface mode, where a click on a surface was sometimes recognized as a short stroke.

Theme 2: Camera and orientation

> *Q3: What was positive or negative about the control of the camera?*

All participants liked how the camera is controlled. Three groups criticized that the camera moves a bit sluggish and slow. One group suggested that positioning the presentation screen along the length of the table top would improve the navigation in the virtual world.

> *Q4: Did you have problems recognizing the correspondence of the two views?*

There were no problems in this regard. All participants were of the opinion that architects are used to work with several views of the same data.

Theme 3: Ideas, collaboration, distraction

> *Q5: Did you feel constrained by the system while exploring ideas? Did you give up ideas, because the system was unable to express them?*

The users were constrained mostly by the bugs in the system, such as inaccurately recognized objects and improperly reconstructed sketches in the 3D mode. Three groups remarked that the system should be faster and more dynamic to better face the needs of architects for a rapid move and change of objects while using the system. Two participants expressed difficulty in sketching with the pen due to its large size and bad ergonomics.

> *Q6: What is better using the digital system and using traditional tools, such as pen and paper?*

Answers to this question were very diverse as every participant had a different working style. While some prefer to start sketching on paper and move to digital tools later in the development, others would do it in a reverse order or even could mix both styles iteratively. All participants agreed that the approaches are quite different and each has unique advantages. For example, digital tools can visualize the prototype within a real or imaginary environment, while paper feels more realistic by the user and provides better haptic experience. Nine of ten users conveyed that they could not abstain from using traditional tools, but all of them demonstrated interest in new and innovative digital tools with a natural user interface such as the CDP.

Q7: Would you prefer the digital system or traditional tools? Why?

None of the users provided a definite answer to this question. In general, users pointed out that the choice of tools depends on the task and that digital tools could not entirely replace traditionally employed methodology. One participant conveyed that he would lay his choice on the CDP, if he was forced to exclusively choose one tool.

Q8: To what extent did the system supplement the creative process?

All five groups agreed on that the system supplies and enhances creation of ideas. As an argument, the users pointed out the possibility to design the buildings directly in their future environment. "Also, the virtual camera gives you the feeling of navigation, which is invaluable for the early stages of conceptual design", says one of the participants, while putting her palms on the table surface and observing the result. "Cool", she says, when seeing two virtual buildings with the shape of her hands.

Q9: How would you improve the camera navigation, sketching modes, and the mode switching?

For summary on this question, refer to section 5.3.2.

5.4. Discussion

The assumption that architects use different approaches to design was confirmed during the evaluation of the CDP and its presentation mode. This observation was not only based on the significant difference in the usage duration of each of the four sketching modes but also on the manner they were used. There was no clearly distinguishable common workflow among the five groups. Each of them approached the given problem differently.

With regard to collaboration, users often divided their work or attempted to work simultaneously on the same concept. They also tried sometimes to sketch at the same time, neglecting the information that only one pen can be recognized by the system. Apparently, architects strive to collaborate in a dynamic and parallelized manner.

With respect to the hedonic quality, users perceived the presentation mode as highly attractive, innovative and desirable. They found the user interface easy to use and well structured. However, the values in AttrakDiff word-pairs "undemanding-challenging" and "unprofessional-professional" could mean that users wish a more complicated and mature user interface.

Another important observation was that users require more functionality. Both CDP and the presentation mode were designed with simplicity in mind, with a requirement to be easy to learn and easy to use. The evaluation results showed that users would rather sacrifice simplicity in favor of more features of the system. One of the participants suggested implementation of an "expert mode" which can be activated by more advanced users. This mode could resolve the issue of "simple interface versus rich functionality".

The sketching modes were designed to mimic the behavior of a physical pen in order to bring the presentation mode closer to a real-life experience. However, users demanded features for editing such as "undo/redo" and volume operations, although these features are typical for CAAD tools and not for the pen-and-paper approach. It appears that users expect a digital tool to provide features typical for digital tools, even if it is designed to mimic non-digital tools.

In summary, architects will most probably not abstain completely from using traditionally established methods such as pen-and-paper, no matter how good a digital equivalent could work. On the other hand, they declare interest and readiness to use the CDP and the presentation mode as a supporting tool. Furthermore, they would switch to the CDP due to the advantage of its 3D capabilities and its ability to visualize buildings in the context of a city during early stage of conceptual design. Interestingly, one of the participants claimed that if he has to choose between the CDP and traditional tools, he would choose the CDP. It can be speculated that architects are less used to working with digital tools for conceptual design, because they never used them in this stage of design. If they had a tool that is more convenient and less restrictive than established CAAD software, they will be more willing to use it.

6. Conclusion

This thesis presented the implementation and evaluation of the presentation mode – a novel approach for conceptual design in urban architecture which combines the convenience of physical tools for designing with the power of digital processing. The presentation mode is built on top of the CDP, a large-scale multi-touch table top which is able to scan physical models of buildings and embed them into an augmented virtuality setting, containing also the geospatial context of a virtual city. The presentation mode visualizes the augmented virtual world using an additional large-scale vertical presentation screen. The user can navigate in the world by moving a tangible camera on the multi-touch table. The user can also sketch directly on the display using four different sketching modes – standard, extended surface, sketch paper, and 3D mode.

The usability and the attractiveness of the presentation mode were tested in a pilot evaluation with ten students in Architecture, who designed a small living area using the CDP and the presentation mode. The students liked the new user interface and its features, especially the possibility to navigate freely in a virtual city and to sketch within the city in a natural manner. All participants were of the opinion that the presentation mode assisted them in being more creative and offered possibilities for being innovative, exploring and sharing ideas.

The evaluation pointed to a demand for a richer and more customizable user interface which would make better use of digital technology. Users would like to customize the sketching, to have more freedom controlling the virtual camera, and to be able to create custom profiles fitting their personal requirements. The users also require more functionality from the system, such as storing of version history, working with different content layers, and improved content management.

Very high interest provoked the 3D sketching mode, despite of its early stage of development. Users liked the idea to let the system recognize the 3D shape of their drawing and would like to see a more elaborate and better working version of this feature.

In conclusion, the participants in the evaluation admitted they would not abstain from traditional tools such as pen and paper in favor of a digital tool for conceptual design, regardless of the quality and the power of the tool. However, they showed high interest towards the CDP and the presentation mode and would readily use them as supplementing tools for conceptual design.

Appendix

A. Projective space $\mathbb{P}^2$

Sketches produced using the presentation mode are based on a 2D input mechanism, but are assigned a 3D meaning within a 3D virtual world. This paradigm requires 2D and 3D information to be related to each other in an intuitive and understandable way for the user.

In computer vision, the mathematical tool used to convert 3D into 2D information and vice-versa (a process called projection and reverse-projection) is called "projective space". There is a lot of literature on this topic. Details on the mathematical foundation of projective space can be found in specialized literature [80, 81].

The projective space, like any other mathematical space, can be of arbitrary dimension. In this thesis, 2D projective space ($\mathbb{P}^2$) is used since we deal with 2D projections of a 3D world. 3D computer graphics applications usually employ the 3D projective space since they deal with 3D projections.

In the $\mathbb{P}^2$ space, camera viewpoint coincides with the origin of a 3D cartesian coordinate system and camera view-vector of the camera coincides with the z-axis of the coordinate system, whereas the up-vector coincides with the y-axis of the coordinate system [1]. A 2D image of a 3D scene can be seen as the projection of the 3D scene objects on the plane $z = 1$. The reverse process, inverting a 2D image back to the original 3D shape, is possible only if the depth of each pixel is known.

In $\mathbb{P}^2$, a 2D pixel is represented by a 3D ray passing through the middle of the pixel (x, y) in image space (see figure A.1). There are infinitely many rays starting at $(0, 0, 0)$ and passing through $(x, y, 1)$ and all of them correspond to the same pixel (x, y). The set of these rays is called the homogeneous coordinate of image point (x, y). If the depth of the image point is known, it is possible to compute its 3D position by taking the single ray which has length equal to the depth of the pixel.

The presentation mode attempts to find the depth of all pixels along a 2D stroke in image space by examining the underlying 3D scene as seen from the camera. For example, in the standard mode for sketching, the intersection of the $\mathbb{P}^2$ ray (representing a pixel in image space) with the closest surface is computed. The distance from this intersection to the camera origin is assumed to be the depth of the corresponding pixel.

An important characteristic of the projective space is that it adds the so-called "line at infinity", which is a plane in the 3D case. Parallel lines in Euclidean space intersect at a point on this line. Consequently, the notion of parallelism, orthogonality, and all other kinds of angular dependencies, are different in the projective space and do not correspond to their equivalent in the Euclidean space.

Currently, the only angular comparison is made in the 3D mode for sketching. The mode checks whether a stroke (drawn in the projective image space of the presentation display) is orthogonal to the city plane in the 3D virtual world [2]. Strictly seen, such a comparison

[1] In a realistic 3D scene, the camera is not positioned at the origin of the coordinate system, but can be moved freely in the virtual world. In that case, the projection plane moves together with the camera.

[2] The stroke is simplified to a line segment, which is defined by the start and the end point of the stroke

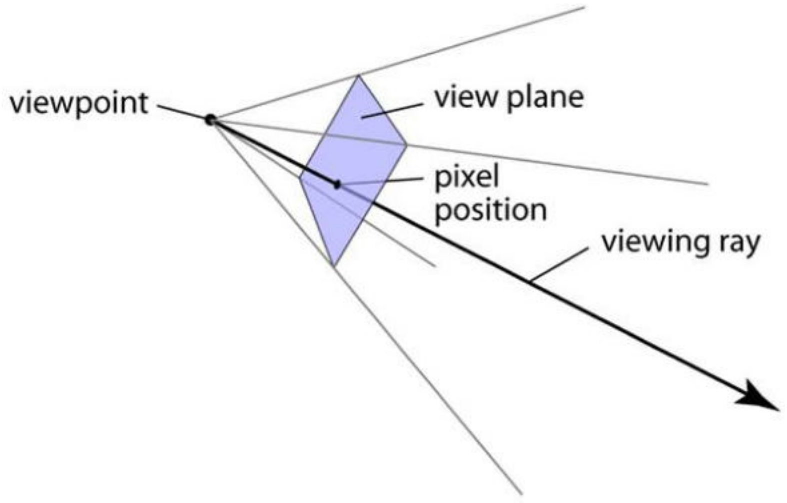


Figure A.1.: Conversion of a pixel coordinate to a homogeneous coordinate in the $\mathbb{P}^2$ space.

is mathematically not plausible, because the stroke in projective space corresponds to infinitely many strokes in 3D space (due to the lost depth component). However, the stroke with the *minimal angle* among all possible 3D strokes can be found. All 3D strokes lie in one plane – the plane which describes the stroke in projective space. The minimal angle of all lines in this plane to another plane in 3D space (in this case, the city plane) is the same as the dihedral angle between these planes.

respectively.

Glossary

Acronyms

API Application Programming Interface.

CAAD Computer Aided Architectural Design.

CDP Collaborative Design Platform.

CUDA Compute Unified Device Architecture.

DOF Degrees of Freedom.

GUI Graphical User Interface.

HDMI High-Definition Multimedia Interface.

SDK Software Development Kit.

TUI Tangible User Interaction.

WIMP Windows Icons Menus Pointer.

Glossary

Application Programming Interface is an interface implemented by a software program that enables it to interact with other software. It facilitates interaction between different software programs similar to the way the user interface facilitates interaction between humans and computers.

Computer Aided Architectural Design refers to software for creation of digital content in architecture design. CAAD programs are used to create a precise digital model instead of drawing it by hand.

Collaborative Design Platform is a large-scale interactive table top which supplements the early stages of conceptual design in urban architecture.

Compute Unified Device Architecture is a technology developed by nVidia, implementing the principle "single instruction, multiple data" (SIMD) on graphical processing units. Hardware and software emloying this paradigm enable massively parallelized algorithms to be executed in interactive time, whereas a sequential execution would take much more time to execute.

Degrees of Freedom is the minimum number of independent parameters which define the state of a system.

Graphical User Interface is a type of user interface which communicates with the user over images instead of textual input/output. Graphical interfaces are very common nowadays in almost all kinds of computer software, ranging from personal computers to complex CAD tools for 3D design.

High-Definition Multimedia Interface is an interface for transferring high definition video and audio data, typically to high-resulution output devices.

Software Development Kit is typically a set of development tools that allows for the creation of applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar platform.

Tangible User Interaction is a kind of user interaction where the user communicates with a system via physical manipulation of graspable tokens.

Windows Icons Menus Pointer is a user interface paradigm implementing the four interaction elements – windows, icons, menus, and pointers. This kind of user interface was developed by Xerox PARC in the early 70's and popularized by Apple's Macintosh and Microsoft Windows in the following decades.

Fiduciary Marker or fiducial, is an object used in computer vision and augmented reality applications to mark a location and/or orientation. Typically, the fiducial is recognized and tracked by a vision system and represents a virtual object in the augmented scene.

Bibliography

Bibliography

[1] G. Schubert, E. Artinger, F. Petzold, and Klinker G. Tangible Tools for Architectural Design. In *ACADIA*, 2011.

[2] H. W. J. Rittel and W. D. Reuter. *Planen, Entwerfen, Design: Ausgewählte Schriften zu Theorie und Methodik.* Stuttgart: W. Kohlhammer, 1992.

[3] C. Gänshirt. *Tools for ideas: an introduction to architectural design*. Basel: Birkhäuser, 2007.

[4] W. Buxton. *Sketching user experience: Getting the design right and the right design*. San Francisco, Calif: M. Kaufmann, 2007.

[5] Dimitra Figa and Gernot Nalbach. *The first sketch*. Förderkreis Dortmunder Modell Bauwesen, 2003.

[6] `http://www.microsoft.com/en-us/pixelsense/default.aspx`.

[7] Ross Bencina and Martin Kaltenbrunner. The design and evolution of fiducials for the reactivision system. In *Conference on Generative Systems*. Music Technology Group, Audiovisual Institute, 2005.

[8] O. Bergig, N. Hagbi, J. El-Sana, and M. Billinghurst. In-place 3d sketching for authoring and augmenting mechanical systems. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pages 87 –94, oct. 2009.

[9] Pierre Wellner. Interacting with paper on the digitaldesk. *Commun. ACM*, 36(7):87–96, July 1993.

[10] Brygg Anders Ullmer. *Tangible Interfaces for Manipulating Aggregates of Digital Information*. PhD thesis, Massachusetts Institute of Technology, 2002.

[11] K. Galloway and Rabinowitz. Hole in space. `http://www.ecafe.com/getty/HIS/`.

[12] Mark Stefik, Gregg Foster, Daniel G. Bobrow, Kenneth Kahn, Stan Lanning, and Lucy Suchman. Beyond the chalkboard: computer support for collaboration and problem solving in meetings. *Commun. ACM*, 30(1):32–47, January 1987.

[13] Mark Weiser. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3):3–11, July 1999.

[14] Hiroshi Ishii, Minoru Kobayashi, and Kazuho Arita. Iterative design of seamless collaboration media. *Commun. ACM*, 37(8):83–97, August 1994.

[15] Nobuyuki Matsushita and Jun Rekimoto. Holowall: designing a finger, hand, body, and object sensitive wall. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, UIST '97, pages 209–210, New York, NY, USA, 1997. ACM.

[16] Norbert A. Streitz, Jörg Geißler, Torsten Holmer, Shin'ichi Konomi, Christian Müller-Tomfelde, Wolfgang Reischl, Petra Rexroth, Peter Seitz, and Ralf Steinmetz. i-land: an interactive landscape for creativity and innovation. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, CHI '99, pages 120–127, New York, NY, USA, 1999. ACM.

[17] Björn Hartmann, Meredith Ringel Morris, Hrvoje Benko, and Andrew D. Wilson. Pictionaire: supporting collaborative design work by integrating physical and digital artifacts. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, CSCW '10, pages 421–424, New York, NY, USA, 2010. ACM.

[18] Eran Ben-Joseph, Hiroshi Ishii, John Underkoffler, Ben Piper, and Luke Yeung. Urban simulation and the luminous planning table. In *Journal of Planning Education and Research*. Journal of Planning Education and Research, December 2001.

[19] R. Bencina, M. Kaltenbrunner, and S. Jorda. Improved topological fiducial tracking in the reactivision system. In *Computer Vision and Pattern Recognition - Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, page 99, june 2005.

[20] Enrico Costanza and John Robinson. A region adjacency tree approach to the detection and design of fiducials. In *VVG*, pages 63–69, 2003.

[21] Michael Haller, Peter Brandl, Daniel Leithinger, Jakob Leitner, Thomas Seifried, and Mark Billinghurst. Shared design space: Sketching ideas using digital pens and a large augmented tabletop setup. In Zhigeng Pan, Adrian Cheok, Michael Haller, Rynson Lau, Hideo Saito, and Ronghua Liang, editors, *Advances in Artificial Reality and Tele-Existence*, volume 4282 of *Lecture Notes in Computer Science*, pages 185–196. Springer Berlin / Heidelberg, 2006. 10.1007/11941354_20.

[22] Morten Fjeld, Kristina Lauche, Martin Bichsel, Fred Voorhorst, Helmut Krueger, and Matthias Rauterberg. Physical and virtual tools: Activity theory applied to the design of groupware. *Computer Supported Cooperative Work (CSCW)*, 11:153–180, 2002. 10.1023/A:1015269228596.

[23] Ehud Sharlin, Benjamin Watson, Yoshifumi Kitamura, Fumio Kishino, and Yuichi Itoh. On tangible user interfaces, humans and spatiality. *Personal Ubiquitous Comput.*, 8(5):338–346, September 2004.

[24] Brygg Ullmer and Hiroshi Ishii. The metadesk: models and prototypes for tangible user interfaces. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, UIST '97, pages 223–232, New York, NY, USA, 1997. ACM.

[25] Patrick Baudisch, Torsten Becker, and Frederik Rudeck. Lumino: tangible blocks for tabletop computers based on glass fiber bundles. In *Proceedings of the 28th international*

conference on Human factors in computing systems, CHI '10, pages 1165–1174, New York, NY, USA, 2010. ACM.

[26] Hiroshi Ishii, Eran Ben-Joseph, John Underkoffler, Luke Yeung, Dan Chak, Zahra Kanji, and Ben Piper. Augmented urban planning workbench: Overlaying drawings, physical models and digital simulation. In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, ISMAR '02, pages 203–, Washington, DC, USA, 2002. IEEE Computer Society.

[27] John Underkoffler and Hiroshi Ishii. Urp: a luminous-tangible workbench for urban planning and design. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, CHI '99, pages 386–393, New York, NY, USA, 1999. ACM.

[28] John Underkoffler and Hiroshi Ishii. Illuminating light: an optical design tool with a luminous-tangible interface. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '98, pages 542–549, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[29] Katja Knecht. Augmented urban model. Bauhaus Universität Weimer.

[30] `http://www.reactable.com/products/`.

[31] John Frazer. An Evolutionary Architecture, 1995.

[32] J. Frazer. Three-dimensional data input devices, 1982.

[33] Kai Schäfer, Volker Brauer, and Willi Bruns. A new approach to human-computer interaction – synchronous modelling in real and virtual spaces. In *Proceedings of the 2nd conference on Designing interactive systems: processes, practices, methods, and techniques*, DIS '97, pages 335–344, New York, NY, USA, 1997. ACM.

[34] Ben Piper, Carlo Ratti, and Hiroshi Ishii. Illuminating clay: a 3-d tangible interface for landscape analysis. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, CHI '02, pages 355–362, New York, NY, USA, 2002. ACM.

[35] H Ishii, C Ratti, B Piper, Y Wang, A Biderman, and E Ben-Joseph. Bringing clay and sand into digital design - continuous tangible user interfaces. *BT Technology Journal*, 22:287–299, 2004. 10.1023/B:BTTJ.0000047607.16164.16.

[36] P. Lapides, E. Sharlin, M.C. Sousa, and L. Streit. The 3d tractus: a three-dimensional drawing board. In *Horizontal Interactive Human-Computer Systems, 2006. TableTop 2006. First IEEE International Workshop on*, page 8 pp., jan. 2006.

[37] E. Sachs, A. Roberts, and D. Stoops. 3-draw: a tool for designing 3d shapes. *Computer Graphics and Applications, IEEE*, 11(6):18 –26, nov. 1991.

[38] Mark Billinghurst, Sisinio Baldis, Lydia Matheson, and Mark Philips. 3d palette: a virtual reality content creation tool. In *Proceedings of the ACM symposium on Virtual*

reality software and technology, VRST '97, pages 155–156, New York, NY, USA, 1997. ACM.

[39] I. Poupyrev, N. Tomokazu, and S. Weghorst. Virtual notepad: handwriting in immersive vr. In *Virtual Reality Annual International Symposium, 1998. Proceedings., IEEE 1998*, pages 126 –132, 18-18 1998.

[40] Pedro Company, Ana Piquer, Manuel Contero, and Ferran Naya. A survey on geometrical reconstruction as a core technology to sketch-based modeling. *Computers & Graphics*, 29(6):892 – 904, 2005.

[41] Beom-Soo Oh and Chang-Hun Kim. Progressive reconstruction of 3d objects from a single free-hand line drawing. *Computers & Graphics*, 27(4):581 – 592, 2003.

[42] Hod Lipson. *Computer Aided 3D Sketching for conceptual design*. PhD thesis, Israel Institute of Technology, 1998.

[43] Lynn Eggli, Ching yao Hsu, Beat D Brüderlin, and Gershon Elber. Inferring 3d models from freehand sketches and constraints. *Computer-Aided Design*, 29(2):101 – 112, 1997. ¡ce:title¿Solid Modelling¡/ce:title¿.

[44] P. A. C. Varley and R. R. Martin. The junction catalogue for labelling line drawings of polyhedra with tetrahedral vertices. In *International Journal of Shape Modeling (IJSM)*, 2001.

[45] D. Kang, M. Masry, and H. Lipson. Reconstruction of a 3d object from a main axis system, 2004.

[46] Pedro Company, Manuel Contero, Julian Conesa, and Ana Piquer. An optimisation-based reconstruction engine for 3d modelling by sketching. *Computers & Graphics*, 28(6):955 – 979, 2004.

[47] A. Wolin, B. Paulson, and T. Hammond. Sort, merge, repeat: an algorithm for effectively finding corners in hand-sketched strokes. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, SBIM '09, pages 93–99, New York, NY, USA, 2009. ACM.

[48] `http://www.3dconnexion.com/products/spacenavigator.html`.

[49] `http://www.optoma.co.uk/projectordetailshe.aspx?PTypedb=High%20Definition%20Home%20Cinema&PC=HD87`.

[50] `http://www.xbox.com/en-US/kinect/`.

[51] `http://www.samsung.com/us/business/commercial-display-solutions/LH65TCPMBC/ZA`.

[52] `http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-580`.

[53] `http://wiki.nuigroup.com/Diffused_Illumination`.

[54] http://opencv.willowgarage.com/.

[55] http://www.nvidia.com/object/cuda_home_new.html.

[56] http://depts.washington.edu/aimgroup/proj/dollar/.

[57] http://www.opentk.com/.

[58] http://pointclouds.org/.

[59] http://www.openni.org/.

[60] http://www.boost.org/.

[61] http://usa.autodesk.com/autocad-map-3d/.

[62] http://www.oracle.com/us/products/database/overview/index.html.

[63] http://www.visualizationlibrary.org/.

[64] http://qt.nokia.com/.

[65] http://assimp.sourceforge.net/.

[66] http://www.cmake.org/.

[67] http://www.visualizationlibrary.org/documentation/pag_key_features.html.

[68] H Lipson and M Shpitalni. Optimization-based reconstruction of a 3d object from a single freehand line drawing. *Computer-Aided Design*, 28(8):651 – 663, 1996.

[69] http://en.wikipedia.org/w/index.php?title=Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm&oldid=499894315.

[70] M. Shpitalni and H. Lipson. Identification of faces in a 2d line drawing projection of a wireframe object. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(10):1000 –1012, oct 1996.

[71] G. Schubert, E. Artinger, F. Petzold, Klinker G., and Yanev V. Tangible Tools for Architectural Design. Accepted 06.2012, 2012.

[72] http://en.wikipedia.org/w/index.php?title=Interpolation&oldid=498656992.

[73] J. Bernsen. Dynamic thresholding of grey-level images, 1986.

[74] http://www.opengl.org/archives/resources/faq/technical/depthbuffer.htm.

[75] http://www.opengl.org/archives/resources/faq/technical/polygonoffset.htm.

[76] Mi Jeong Kim and Mary Lou Maher. The impact of tangible user interfaces on designers' spatial cognition. *Human-Computer Interaction*, 23(2):101–137, 2008.

[77] J. Brooke. Sus-a quick and dirty usability scale. In *Usability evaluation in industry*, 1996.

[78] M. Hassenzahl, M. Burmester, and F. Koller. Attrakdiff: Ein fragebogen zur messung wahrgenommener hedonischer und pragmatischer qualität. In *Mensch & Computer*, pages 187 – 196, 2003.

[79] `http://sketchup.google.com/`.

[80] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge Univ Press, 2003.

[81] O. Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. The MIT Press, 1993.