Aalto University
School of Chemical Engineering
Master's Programme in Chemical, Biological and Materials Engineering

Otso Pietikäinen

VRChem:

A molecular modeling software for Virtual Reality

Master's Thesis
Espoo, 5.2.2018

Supervisor: Professor Antti Karttunen
Advisor: M.Sc. Jarno Linnera

Aalto University
School of Chemical Engineering
Master's Programme in Chemical, Biological and Materials Engineering

ABSTRACT OF MASTER'S THESIS

Author:	Otso Pietikäinen		
Title:	VRChem: A molecular modeling software for Virtual Reality		
Date:	5.2.2018	**Pages:**	51
Major:	Chemistry	**Code:**	CHEM3023
Supervisor:	Professor Antti Karttunen		
Advisor:	M.Sc. Jarno Linnera		

Understanding the often complex atomic-level three-dimensional structure of molecules and materials is crucial in chemistry, and many tools have been designed and created to help chemists visualize and examine these structures. Currently, the most popular tool is molecular modeling software designed for desktop computers. The 2D-representations of molecules these applications render, however, are not ideal for studying the three-dimensional structure, and better tools with support for true stereoscopic molecular 3D-representation are needed.

For this purpose, we have created VRChem, a novel molecular modeling application for Virtual Reality featuring a modern 3D user interface. With VRChem and the necessary hardware, users can construct and modify molecules in a virtual reality environment using their bare hands.

This thesis covers the development and technical details of the VRChem application and what is being planned for future versions. VRChem is built on the Unity game engine for multi-platform support and it has thus far been an adequate platform for the application. The current version was built as a prototype for user interface testing and is not suitable for professional or research use, but the planned updates, such as improved graphical user interfaces, performance optimizations and more accurate energy minimization methods could make it a functional application in the future.

Keywords:	Virtual Reality, VR, modeling, molecular model, software, Unity
Language:	English

Aalto-yliopisto
Kemiantekniikan korkeakoulu
Master's Programme in Chemical, Biochemical and Materials Engineering

DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Otso Pietikäinen		
Työn nimi:	VRChem: Virtuaalitodellisuuteen perustuva molekyylimallinnusohjelmisto		
Päiväys:	5.2.2018	**Sivumäärä:**	51
Pääaine:	Kemia	**Koodi:**	CHEM3023
Valvoja:	Professori Antti Karttunen		
Ohjaaja:	DI Jarno Linnera		

Molekyylien ja materiaalien usein monimutkaisen atomirakenteen ymmärtäminen on äärimmäisen tärkeää kemiallisessa tutkimuksessa ja reaktioiden suunnittelussa. Tätä tehtävää varten onkin luotu useita työkaluja, joiden avulla voidaan mallintaa kemiallisia rakenteita, tutkia niiden reaktiivisuutta ja suunnitella halutun rakenteen luomista. Nykyaikana käytetyimmät mallinnustyökalut ovat tietokoneohjelmistoja, joita käytetään hiirellä ja näppäimistöllä ja joka piirtää kuvaa rakenteesta tavalliselle tietokoneen näytölle. Valitettavasti näyttöpäätteen kaksiuloitteisesta kuvasta johtuen kemiallisten yhdisteiden rakenteiden kolmiuloitteisuus, jolla on suuri merkitys reaktiivisuuteen, on tietyissä tilanteissa vaikea hahmottaa. Tästä syystä alalla olisi kysyntää työkaluille, jotka kykenisivät tarkempaan 3D-rakenteen visualisointiin.

Kehittämämme VRChem-ohjelma on virtuaalitodellisuuteen perustuva stereoskooppinen molekyylimallinnusohjelma, joka kykenee erittäin realistiseen kolmioloitteiseen molekyylien visualisointiin ja niiden muokkaamiseen käyttäjän omin käsin, myös ilman käsissä pidettäviä ohjaimia.

Tämä diplomityö käsittelee VRChem-ohjelman kehitystyötä ja teknistä toteutusta, ja kuinka ohjelmiston kehitystä voitaisiin jatkaa paremman kokemuksen tarjoamiseksi. VRChem on kehitetty Unity-pelimoottorin avulla, ja sen nykyinen prototyyppiversio on täysin toimiva kokonaisuus, jota on käytetty käyttäjäkokeissa virtuaalitodellisuuden hyödyllisyyden tutkimiseksi. Ohjelma ei ole valmis käytettäväksi kemiallisessa tutkimuksessa tai opetuksessa, mutta suunnitellut ominaisuus- ja suorituskykyparannukset voivat tulevaisuudessa tehdä tästä mahdollista.

Asiasanat:	Virtuaalitodellisuus, VR, molekyylimalli, molekyylimallinnus, ohjelmisto, Unity
Kieli:	Englanti

Acknowledgements

I wish to thank Krupakar Dhinakaran for co-developing the VRChem application, and Antti Karttunen for his excellent supervision, help, and tolerance for my disregard of deadlines.

Espoo, 5.2.2018

Otso Pietikäinen

Abbreviations and Acronyms

AR	Augmented Reality
CAMD	Computer Aided Molecular Design
CAVE	Cave Automatic Virtual Environment
DOF	Degree Of Freedom
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HID	Human Interface Device
HMD	Head-Mounted Display
IPT	Immersive Projection Technology
SDK	Software Development Kit
UI	User Interface
VR	Virtual Reality
VSEPR	Valence Shell Electron Pair Repulsion

Contents

Chapter 1

Introduction

VRChem is a new molecular visualization application intended for immersive and intuitive molecular model building in a virtual reality environment. VRChem has a comprehensive feature set for building medium-sized organic molecular models and could be the first modeling application with full molecular building and editing support in virtual reality. The application is still at a prototype stage but will remain under active development and will be improved in the near future. The application project file is available at `https://version.aalto.fi/gitlab/VRChem/VRChem_Unity`.

While the VRChem application can be used with a traditional mouse and keyboard, our focus has been on developing a novel, user-friendly interface for virtual reality with comparable accuracy and comfort to a traditional interface. The virtual reality interface of VRChem uses a head-mounted display and a hand tracking device and allows the user to walk around or inside molecules and interact with them using their bare hands. The purpose of developing VRChem was to test whether or not virtual reality could be beneficial for chemical research or for teaching chemistry.

VRChem is developed by Otso Pietikäinen and Krupakar Dhinakaran. Dhinakaran's thesis covers the design of the virtual reality interface and how test users reacted to the new interface[1], while this thesis covers the technical details of the application and its development. First, though, we examine how molecular visualization has developed and how virtual reality could be the next step for molecular graphics.

Chapter 2

Background

Understanding molecular geometry is absolutely crucial in chemistry. Molecular geometry can be measured and studied using a variety of methods, including various spectroscopic methods, diffraction methods etc. [2], but presenting this data in a clear and unambiguous form can be a challenge. The primary issue is often how to display the three-dimensional structure of a molecule on a two-dimensional medium, such as paper or a computer screen. For very simple molecules the stereochemical designation of a Lewis structure or the depth cues of a shaded image may be sufficient for understanding the full structure, but both become impractical when used for larger structures like proteins. Molecular modeling software can alleviate the problems by allowing the user to rotate, zoom and often modify the molecular model and thus examine the structure in more detail. This presentation, however, still relies on the illusion of depth conveyed by the computer screen, limiting the user to some degree. Combined with an often lacking or unintuitive user interface, many chemists still prefer to use physical kits for building and examining chemical structures in select cases.

The depth perception issue with two-dimensional media has long been understood and acknowledged, and a number of true stereoscopic visualization methods have been introduced for studying chemical structures. The next section contains a short history of molecular visualization and some the stereoscopic visualization techniques.

2.1 Development of molecular graphics

Before the rise of computer graphics, the most widely used method for molecular structure visualization was physical models [3] and kits for assembling these models have been commercially available at least since the 1930's [4].

Obviously, though, physical models could not be used to represent a molecule of any significant size while maintaining practicality from a research point of view.

The pivotal point for molecular visualization and the birth of molecular graphics is often attributed to the landmark article of Cyrus Levinthal released in 1966, although similar development was being conducted elsewhere simultaneously [3, 5]. Levinthal was using an early computer system with support for real-time interactivity at Massachusetts Institute of Technology (MIT) to both build and display a vector representation of a molecule, such as the one presented in figure 2.1[6]. The model could be rotated on a graphics display using a device not too dissimilar from a trackball and the rotation of the molecule would convey information of the three-dimensional structure via the *kinetic depth effect* [7].

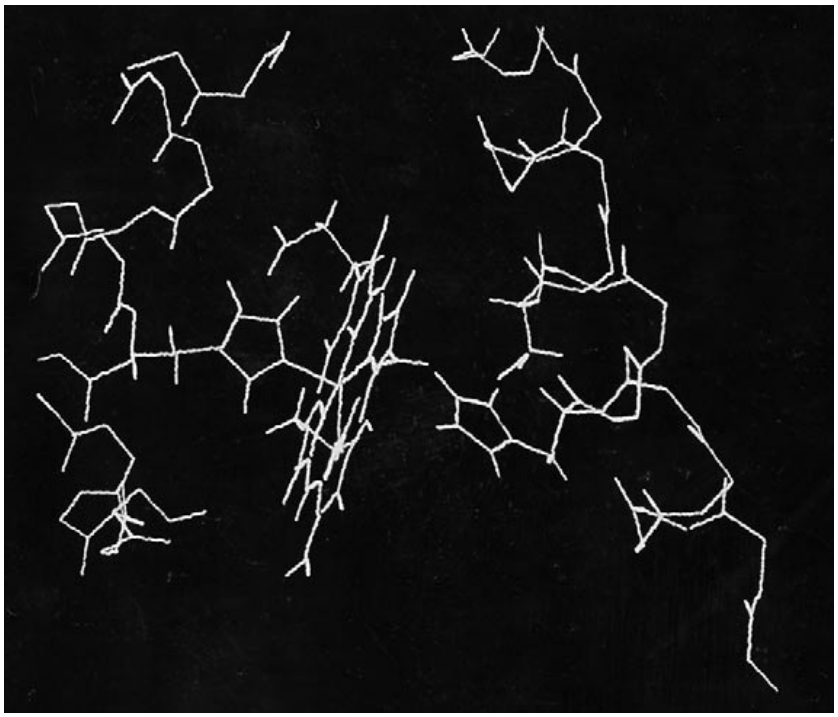

Figure 2.1: Photograph of a part of the structure of the myoglobin protein as displayed on the system Levinthal used.

While the visual quality of molecular models has steadily grown with the advancement of computers and computer displays, the main principle of depth perception in molecular visualization has been mostly the same since 1966. Modern graphics processing units, advanced shaders and real-time

raytracing can help the depth perception by providing some depth cues like shadows and blur [5], but the kinetic depth effect is still the primary depth cue in most applications. Figure 2.2 illustrates the effects of advanced shading techniques like ambient occlusion on the perceived depth of a structure.

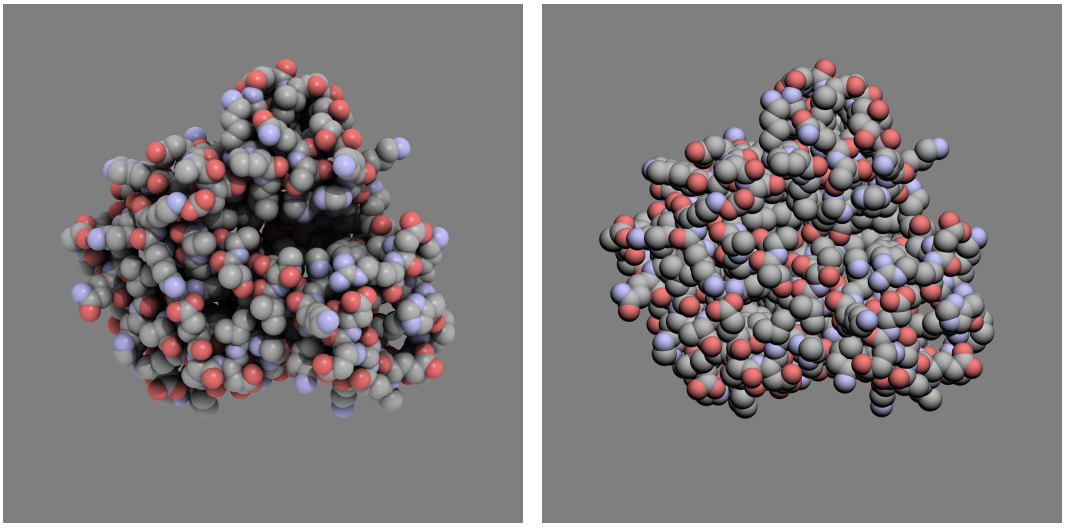

Figure 2.2: Two pictures of a myoglobin protein illustrating the depth cues of modern shading techniques. Both pictures are rendered from the same angle, but the picture on the left is rendered using directional illumination, self-shadowing, ambient occlusion and more [8]. The picture on the right is rendered using only direct illumination. The pictures are rendered using QuteMol [9].

To further enhance the perceived depth of an image, stereoscopic methods are required. By feeding two different images rendered from slightly different angles to a person's eyes, their brain can be tricked into perceiving the images as a three-dimensional scene in a phenomenon called stereopsis. The cost of stereoscopic display setups, however, has reduced interest for stereoscopic molecular graphics software until recently.

The most common form of stereoscopic displaying is the active shutter-based system used in modern 3D-movie theaters, televisions and computer screens. The system uses a high framerate screen or projector to alternate between displaying an image for the left and the right eye, and using a pair of synchronized shutter glasses to block the image from the other eye. The technique is suitable for stationary work and does not require very expensive hardware. However, the depth effect is limited by the static nature of the image i.e. the user cannot, for example, peek behind an object due to at

least a usual lack of head tracking.

The active shutter system has also been tested in room-scale. Immersive projection technology (IPT) systems like the *Cave automatic virtual environment* (CAVE), as shown in figure 2.3, project the alternating images on set of walls. Active shutter glasses then allow the user to see objects floating inside the room and a motion tracking system allows them to walk around the object while maintaining the illusion. The ability to walk around and inspect objects from different angles is a defining feature that makes the CAVE a proper virtual reality (VR) system instead of a 3D display and helps users become immersed in the projected world. CAVE setups have been used to demonstrate chemical reactions to students, and have helped the students understand the reactions and also be more enthusiastic about the subject matter in general[10]. However, the cost and space requirement of a stationary CAVE system are serious limitations for widespread use.

Figure 2.3: A CAVE virtual reality system displaying an astronomical simulation.

Another approach for stereoscopic viewing is mounting a pair of screens directly in front the user's eyes. Head-mounted displays (HMD), such as the one in figure 2.4, are less expensive than a CAVE, more portable and do not require as much space to use. Modern examples of HMDs like the Oculus Rift

and HTC Vive feature high resolution and high framerate displays for decent image quality, a wide field of view that comes close to the human eye's natural field of view, and very accurate position and orientation tracking. This allows for an immersive virtual environment where the user can walk around and interact with the virtual world in a surprisingly natural way. While head-mounted displays have been in development for half a century, recent leaps in high definition screen manufacturing and computing power have created a new industry of consumer grade HMDs and accompanying software. Among these are several virtual reality molecular visualization applications, which are discussed in the next section.

Figure 2.4: Image of the Oculus Rift CV1 head-mounted display.

2.2 Existing software for molecular visualization in VR

In recent years, the availability of consumer grade VR head-mounted displays has attracted the attention of both academic and commercial software developers, and several virtual reality molecular visualization applications have already been released. The following subsections will introduce some of the more noteworthy releases.

2.2.1 Molecular Rift and Molecular Vive

Molecular Rift is an open-source protein visualization application developed in 2015 at Lund University, Sweden, for the Oculus Rift HMD [11]. The application was designed for pharmaceutical researchers and drug designers and its main purpose is to display protein structures. The software features several rendering methods for the primary and secondary protein structures and uses a Microsoft Kinect motion sensing device for rotating, zooming and panning molecules and interacting with a menu system. Another feature of the application is the integration of the open source chemistry toolkit Open Babel [12]. The toolkit is used to parse and correctly display the protein database files and to implement force fields for energy minimization (mmff94 [13], uff [14] and ghemical [15]). Molecular Rift was built using the Unity game engine and software development kit (SDK). Figure 2.5 shows the Oculus Rift and Kinect devices required for Molecular Rift as well as a mirrored image of the stereoscopic view rendered for the two displays in the headset.

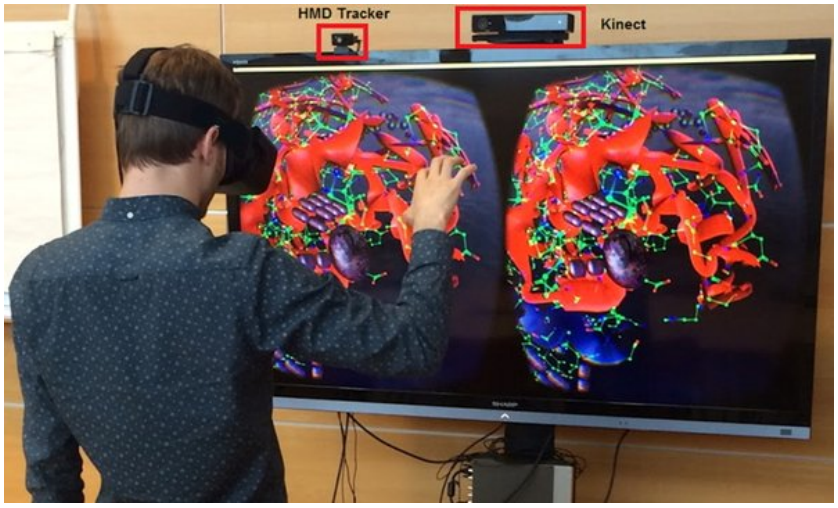

Figure 2.5: Image of the Molecular Rift setup and a mirrored image of the view inside the HMD.

We did not have a chance to test the Molecular Rift application but instead evaluated the software based on the article and the application's change logs. We were impressed by the different visualization schemes but found the small feature set to be quite limiting for proper design work. Moreover, the motion tracking device Kinect is no longer in production making a major part of the application obsolete [16]. Molecular Rift has not been updated since 2016 and the last update to a fork targeting the HTC Vive VR system

was in March 2017 [17, 18].

2.2.2 Caffeine molecule viewer

The Caffeine molecular viewer is another VR application for molecular visualization developed in 2016 [19]. It is similar in features to the Molecular Rift application but has a few distinct advantages over it. Like Molecular Rift, Caffeine is strictly for visualization i.e. it has no interactive capabilities with the molecular models. Also, it too is primary designed for working with proteins, has the usual rendering modes for proteins, and includes the Open Babel toolkit. However, the Caffeine viewer is not built on a video game engine but rather using OpenSceneGraph, an open source 3D graphics engine. This may make for a more efficient application by reducing the computing overhead, since, unlike in for example Unity, there is no audio engine, physics engine, unnecessary networking or other video game-related subsystems running. In a test conducted by the authors of the software, Caffeine was able to display a molecular system with 356,000 atoms in a CAVE virtual environment while maintaining a frame rate above 55 frames per second. The performance was compared with that of VMD, a competing molecular viewer with CAVE support, which only managed between 2 to 7 frames per second. The frame rate and latency of a VR environment has been proven to have an effect on the perceived realism of the environment, and a frame rate below 15 FPS has been shown to dramatically reduce test subject performance in a virtual reality task [20]. Figure 2.6 shows a caffeine molecule rendered using the Caffeine software. The figure also illustrates the applications ability to overlay volumetric orbital data alongside the molecular structure.

In our opinion, Caffeine looks very promising from a visual and performance standpoint but the lack of molecular modeling tools for editing the molecules is a drawback and limits the application's usefulness. The orbital data visualization, however, is an intriguing feature. Caffeine does not support current HMDs and has removed support for the older Oculus Rift development kits[21].

2.2.3 ChemPreview

ChemPreview is another open-source Unity-based molecular augmented reality application, but unlike Molecular Rift it supports molecular building, editing and measuring using a hand-tracking system embedded to the Meta 1 augmented reality headset [22]. Augmented reality (AR) differs slightly from virtual reality though the core concepts of accurate depth perception

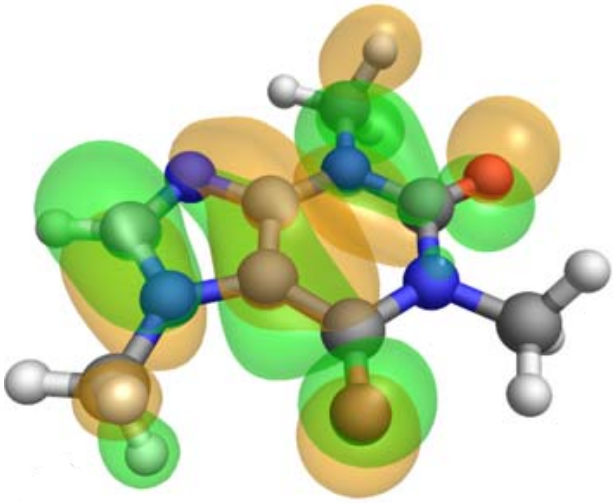

Figure 2.6: A caffeine molecule rendered using the Caffeine molecular viewer. The image features the ball-and-stick representation of the molecule as well as semitransparent isosurfaces of the highest occupied molecular orbitals.

and digital visualization remain. The main difference is that in VR the physical world is replaced by a virtual environment whereas in AR, the digital features are projected as a part of the physical world. In ChemPreview for example the molecules can appear as floating holograms on the user's desk or palm, like in figure 2.7. Furthermore, since an AR device allows users to see their physical surroundings better, it enables more natural communication and collaboration between coworkers and might therefore be more suitable for professional working environments.

ChemPreview supports some hand gestures, primarily an open palm and pointing a finger. Using these hand gestures, the user can build molecules, select, rotate, move and resize atoms and measure bond lengths and angles. However, the UI does not appear to be specifically designed for VR/AR use and the their gesture-based interactions seem difficult to use, making the application somewhat impractical. ChemPreview was last updated in 2016 [23].

2.2.4 Commercial VR applications

In addition to the experimental non-commercial applications already discussed, some commercial molecular visualization application developers have also begun to show interest in supporting virtual reality. For example, Au-

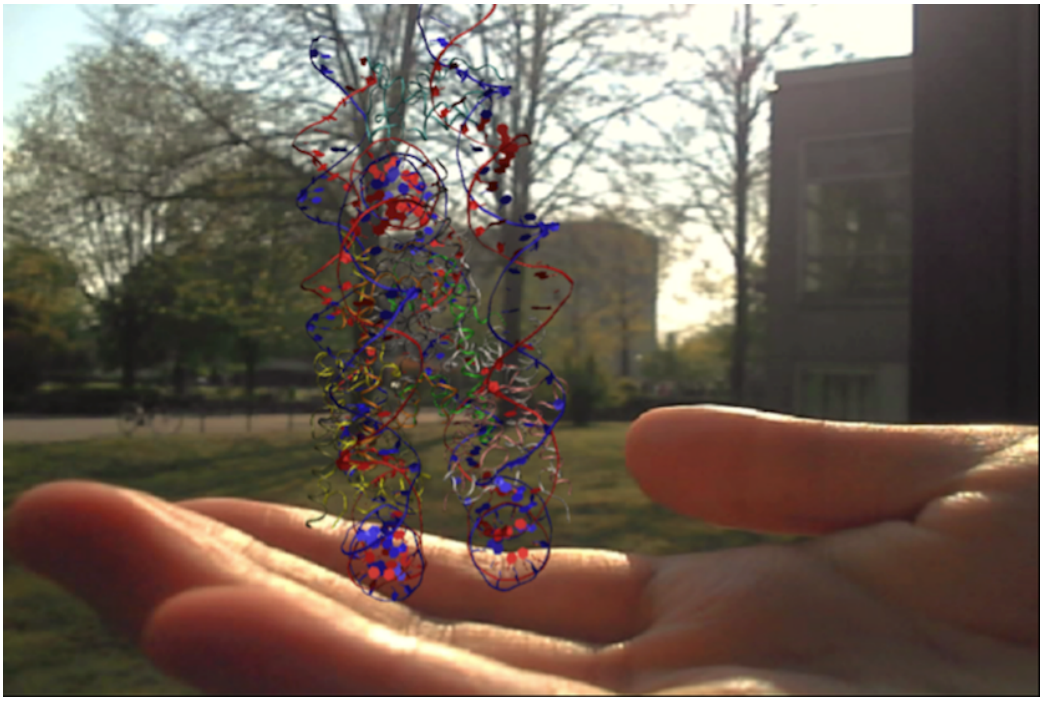

Figure 2.7: A view of a protein rendered by the ChemPreview augmented reality molecular modeling application.

todesk's Molecule Viewer has added support for the common commercial HMDs and smartphone app stores feature visualization applications such as MoleculE VR. However, both of these, like most academic VR applications, lack any building tools which seems to be a recurring problem in existing software.

Based on this research into existing molecular graphics software we concluded that a molecular modeling application for virtual reality with a properly designed graphical user interface is a novel experiment and worth developing. We also began to plan how VR could be used in the future to replace laboratory experiments by incorporating computer aided molecular and reaction design, which is discussed in the next section.

2.3 Interactive computer aided molecular design

While visualizing molecules is a fundamental part of chemical research and improving our visualization capabilities may help researchers and students in

understanding and developing complex chemistry, incorporating computers and computation into a chemist's workflow could have an even more profound effect in other areas. Computer aided design (CAD) and computer simulations have revolutionized many areas of engineering, such as mechanical and electrical engineering, and bringing similar tools to chemical reaction design could allow for rapid prototyping and less laboratory work for chemical researchers as well. Thus far, though, attempts at computer aided molecular design (CAMD) have suffered from either a lack of accuracy or too long response times for any real-time reaction engineering. In CAMD the fundamental problem to solve is the computation of interatomic forces of molecules, minimizing the energies of reagents, reaction intermediates and products and finding a minimum energy path between them. Empirical force field-based models, while quick, are too inaccurate for general use or are accurate for only a small subset of molecules. On the other hand, quantum chemical approaches have until recently been too computationally intensive to use in an interactive, real-time application. [24, 25, 26, 27]

Today, however, parallel computing with modern graphics processing units (GPU) can provide enough computing performance, and researchers have managed to combine quantum mechanical Hartree-Fock[28], density functional theory (DFT)[24] and density-functional tight-binding (DFTB)[25] methods with modern GPUs to simulate reaction systems of up to several hundred atoms. The simulations have reached an update rate above 1 Hz, which can be considered adequate for real-time, feedback-based reaction design[24, 26, 29]. These quantum chemical CAMD methods have also been successfully combined with augmented reality to provide a very promising way to examine reactivity.

Quantum chemical calculations output large amounts of data and are often difficult to interpret numerically. To address the issue, several researchers have begun to look into using haptic pens instead to decode the data. Haptic pens are computer input devices that can be used as a type computer mouse that can move a cursor in three directions, instead of the traditional two. The pens can also use actuators and motors to apply a force to the user's hand, resisting or assisting the hand's movements. An example of a haptic pen is shown in figure 2.8.

The haptic pen can be used to probe the potential energy surface (PES) of a molecule and explore feasible reactivity by moving atoms and molecules around and rendering the calculated force to the user's hand via the haptic pen. For example, trying to pull apart a strong bond results in the pen pushing against the user's motion, while bringing two carbon atoms close to each other would result in the opposite. In other words, the force feedback of the haptic pen resists the user's hand movement when pushing against

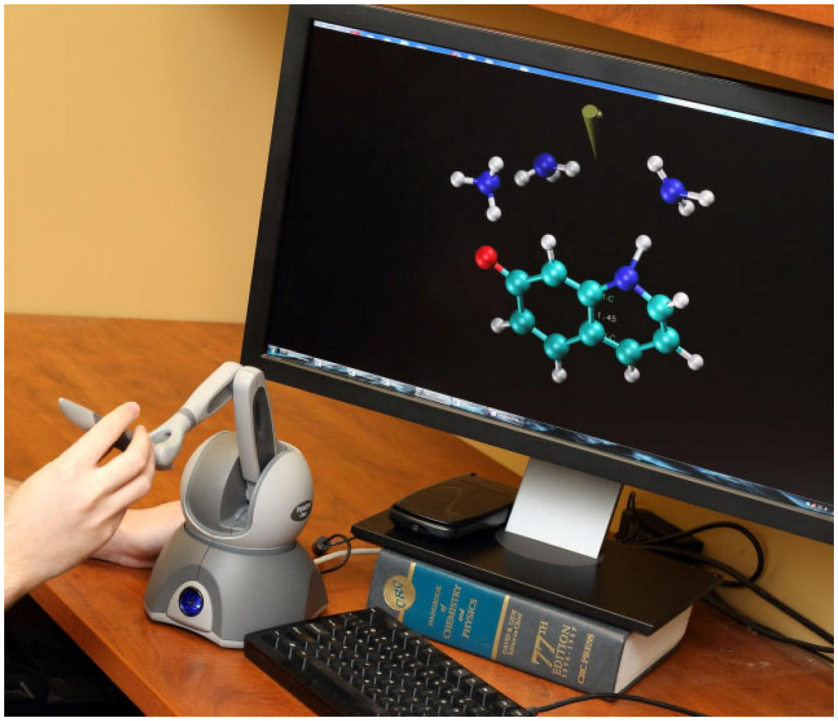

Figure 2.8: An example of force feedback-based chemical reactivity exploration using a haptic pen.

a potential energy barrier, and pulls towards any close by minima in the potential energy surface. This helps the user explore otherwise potentially unexpected minimum energy pathways and avoid unlikely reactions with high barriers. Combining the haptic pen and its 3D-cursor with the accurate depth perception of a VR or AR headset could be the preferred media for reaction engineering in the future.

Chapter 3

Environment

Development of the VRChem prototype was started by Dhinakaran who laid the foundation of the application. Dhinakaran was developing VRChem for his master's thesis for the school of Arts Design and Architecture, and was researching intuitive VR interfaces for professional applications[1]. Dhinakaran selected the Unity game engine and software development kit as the base for VRChem, and a Leap Motion hand tracking device as the primary user input device. In short, Unity was selected because of its ease of adaption, VR support, available programming languages and good documentation, while Leap Motion was chosen because of the peripheral-free user interactivity it could facilitate [30, 31]. The Leap Motion technology had been successfully deployed in an earlier project by Dhinakaran *et al.* to provide a gesture-based interface for 3D characted posing and animation [32]. This familiarity with gesture-based intarfaces was one of the main reasons behind our cooperation. This chapter covers the hardware and software components used in VRChem and explains their functionality and purpose in the project.

3.1 Hardware

Hardware required for developing and testing VRChem included a virtual reality HMD, a Leap Motion hand tracking device, and a moderately fast computer with a minimum of 4 processor cores and a dedicated GPU.

3.1.1 HTC Vive head-mounted display

HTC Vive is one of the two mainstream, consumer-grade HMDs currently available for purchase. The headset features two 3.6" AMOLED screens at 1080x1200 pixels per eye and Fresnel lenses that both expand the field of view

of the screens to 110° and move the focal plane of the screens further from the user's eyes to prevent eye strain. The resolution of the screens is relatively low in relation to the distance from the eyes and the pixelation is easy to notice, but not distracting. The Vive system also tracks the position of the HMD with 6 degrees of freedom (DOF) (translation and rotation in x, y and z) using a suite of sensors and lasers. This allows the user to both move their head freely and walk around a predefined area while maintaining full position and orientation tracking. Based on our personal experiences and the experiences of our colleagues we can safely say that the virtual reality experience is very natural and immersive as long as the sensors are unobstructed. [33]

The Vive system is shipped with a pair of wireless hand controllers. The controllers use a similar sensor suite to provide 6DOF tracking of the user's hands and can be used to point at virtual objects and interact with them by pressing buttons on the controllers. However, since our initial plan was to avoid using cumbersome controllers, another human interface device was needed for control purposes.

3.1.2 Leap Motion

The device we chose for interacting with our virtual environment, the Leap Motion, is a small accessory which mounts to the front of a VR headset. The device has two wide-angle IR-sensitive cameras that capture and combine stereoscopic image data and use image recognition algorithms to track the user's hands and fingers. This system is surprisingly accurate and works well in most situations. The obvious drawback of the technology is the limited view that the fixed mounting point can offer. In cases where the user's fingers are hidden behind their palm or another object, the device can not accurately determine the position of those fingers. Instead, the algorithms try to infer the probable position of the fingers based on the human skeleton and seem to favor some commonly used hand gestures. This limits the amount of reliably recognized hand gestures we can use for interacting with our application.[34] Figure 3.1 shows the VR setup used in developing VRChem.

3.2 Software

Like most applications, VRChem builds upon existing software. The core of the application is based on the Unity game engine which is the framework it runs on, but we have also included features from Open Babel, a common open source chemical programmer's toolkit.

Figure 3.1: An photograph of the HTC Vive head-mounted display and the attached Leap Motion hand-tracking device, and the Vive Wand hand controllers.

3.2.1 Open Babel

Open Babel started as a tool for translating between different molecular file formats but has since grown to include tools for measurements, molecular analysis, fingerprinting and energy minimization [12]. At the moment, VRChem only uses Open Babel for optimizing molecular structures with the mmff94 force field, but should in the future also include at least saving and loading files in the formats supported by Open Babel.

3.2.2 Unity

Unity is a proprietary cross-platform game engine and game development environment for Microsoft Windows and Mac OS X. Unity supports both 2D and 3D application development and includes e.g. full rendering and physics engines, a UI system for menus and other user interfaces, and supports C# and JavaScript scripting across all platforms. As of 2017, Unity supports more than 25 platforms on desktop systems, mobile devices, game consoles and VR and AR platforms. The Unity engine is, according to its developers, the most commonly used development platform for Virtual- and Augmented Reality applications, which was one of the main reasons we chose to use it. [30, 31]

For its cross-platform support, Unity uses Mono developed by Xamarin. Mono is a platform-agnostic open source toolkit for compiling and using Microsoft's .NET framework outside the Windows operating system. [35] Before

running an application, Unity compiles the C# or JavaScript code written by the developer to dynamically linked library (DLL) files. At runtime, these are compiled using the .NET virtual machine on Windows or the Mono virtual machine on other platforms using a Just-in-Time (JIT) compiler [36]. This system allows Unity developers to use the popular C# language for programming applications for all platforms. The downside of this appears to be the deep integration of Mono into the Unity engine and the resulting slow pace of updates to the feature set offered by Mono. At the start of this project, Unity was at version 5.6.3p2 and supported the .NET 2.0/3.5 equivalent, which was released in 2007. We quickly noticed that finding plugins written for a ten-year-old framework can be difficult and lead to unnecessary work if one is not available. Fortunately, we managed to get Open Babel to work in Unity. Furthermore, a late 2017 beta version of Unity contained an updated version of Mono and added support for .NET 4.6 equivalent which should remove compatibility issues.

An important detail with Unity is that it uses a left-handed coordinate system for object positions, and therefore careful consideration is needed when calculating object coordinates. For example, many chemical file formats use a right-handed coordinate system. This could lead to unexpected behaviour between programs unless the coordinates are converted correctly.

3.2.2.1 Unity application structure

A basic understanding of the Unity application structure is necessary to fully follow the technical examination in a later chapter. The following subsections covers the basic building blocks of a Unity application and their purpose.

3.2.2.2 Scenes

A scene in principle contains all program data concurrently loaded into the computer's memory. In video games, scenes are usually used to load and unload different sections of the game to memory if the whole game would be too large for the system's RAM. VRChem currently only has one scene but scenes could be used to switch between a VR and a desktop version of the application.

3.2.2.3 Objects

Scenes are populated by objects, which can be considered as data containers for individual entities within the application. Video games made with Unity usually have objects for characters and physical objects within the game, but also for abstract parts like cameras and user input devices. Objects can also

contain child objects and be enabled or disabled. Disabling an object hides the object and its children from the application and halts the execution of script in the object. VRChem has objects for each atom and bond, a parent object for all atoms and bonds, objects for the different cameras, empty objects that only exist to host a script, and more.

3.2.2.4 Components

Components are the data that objects contain. Each component in an object gives the object a property, such as a 3D mesh that gives the object a shape. Components also include collision meshes for physics simulations, light emissions and particle effects, shaders for object color and surface material and so on. Components are also responsible for all program logic in the form of scripts.

3.2.2.5 Scripts

Scripts are the most commonly used component at least in VRChem. All of Unity's programming is done using scripts, which have to be linked to an object to run. There is, for example, no accessible main-loop for doing general computing and all code has to be instead written to a script and tied to some object in the object hierarchy. Scripts in Unity can be thought of as C# classes and can hold variables and contain methods.

Methods in scripts can be invoked by two procedures: from special built-in methods and via events. At specific points of execution, like before rendering a new frame, the Unity engine calls an associated method in each enabled script. For example, the Update() method is called between each frame and can be used for animating atom movement.[37] Unity also supports C# events which are used in VRChem for user interactions. Events are used to notify objects and classes that a predetermined set of conditions has been met and appropriate action should be undertaken. We have created an event system where user actions, such as clicking or grabbing an atom, sends out an event that an event listener has subscribed to. The event listener then handles the different events by calling the appropriate methods in other scripts. [38]

3.3 Plugins and extensions

One of the core features of the Unity SDK is an extensive support for third-party extensions and plugins. [30, 39] Extensions, in this case, refer to downloadable packages of Unity-native objects and scripts while plugins refer to externally compiled .NET code assemblies. VRChem currently uses three

extensions from Unity's Asset Store and two managed plugins from other sources. This subsection covers the purpose of these components and their licensing terms.

3.3.1 SteamVR Unity plugin

SteamVR plugin is a native Unity extension developed by Valve and is designed to allow Unity application to work with the HTC Vive head-mounted display. The extension contains an application programming interface (API) for OpenVR which provides positioning and interaction data from the device. The extension also adds VR support for Unity's camera objects and corrects distortion caused by the lenses in the device, allows stereoscopic rendering in Unity and provides virtual models for the hand controllers. SteamVR Unity plugin is released under *BSD 3-clause "New" or "Revised"* license which allows commercial usage. [40]

3.3.2 Virtual Reality Toolkit

Virtual Reality Toolkit (VRTK) is an auxiliary plugin for virtual reality applications in Unity developed by Sysdia Solutions. The cross-platform extension supports several VR devices and can be used to easily add common VR features to Unity applications. Supported features includes locomotion within the virtual world and interacting with virtual objects and UI elements with VR controller accessories. VRChem uses VRTK to select and modify atoms and to interact with the UI menu system when using the Vive hand controllers. VRTK is released under the MIT license which allows commercial use. [41]

3.3.3 Leap Motion Unity plugin

The Leap Motion plugin provides an API for accessing the Leap Motion hand-tracking device and can be used to add hand-activated object and UI interactions to an Unity application. We used the Leap Motion plugin to track hand gestures used for building and editing molecules, to interact with our UI menu system and to render a pair of low-polygon virtual hands in the VR environment. The plugin's terms and conditions allow commercial use of the plugin. [42]

3.3.4 OBDotNet

OBDotNet is a precompiled .NET assembly for the Open Babel API and enables the use of Open Babel's many features in external applications. VRChem currently only uses Open Babel for molecular geometry optimization but will add other features in a later update, such as saving and loading files in supported file formats. Open Babel and OBDotNet are released under the *GNU General Public License* version 2.0 which allows copying and redistributing the licensed work, if the derivative application is released under the same licence terms and its source code is made available. [12, 43]

3.3.5 CsvReader

The final plugin used in VRChem is a small, fast CSV parser by Sébastien Lorion. The plugin was chosen simply because it is written using .NET 2.0 and is therefore supported in Unity. The parser is released under the MIT license and does not set any additional restricting stipulations for releasing VRChem.

Chapter 4

Development goals

The goal of the VRChem prototype was to develop a modeling application with support for visualization and molecule building in VR. The application should also have a traditional desktop interface for UI testing and comparison. No performance targets or restrictions were set. Before starting the software development, Dhinakaran studied some commonly used modeling applications and formulated a list of required functions for VRChem. The features chosen were

- Creating molecules by adding or subtracting atoms and bonds
- Arranging, moving and rotating sub-structures of molecules
- Stretching and shrinking bond lengths
- Showing ball-and-stick figures, CPK colouring [44], and secondary structures of proteins
- Measuring distances, angles, torsion of and between bonds
- Carbon atoms automatically adding hydrogens (saturation), ability to replace a hydrogen with something else.
- Offering a short list of elements to choose from, with a button to open a whole periodic table

Secondary structures were later removed from our priority list, and force field-based structure optimization was added.

Chapter 5

Implementation

VRChem was developed as a prototype for testing the applicability of virtual reality as a research tool and is not intended for release in its current form. The prototype, however, is fully functional and has been demonstrated at numerous venues, receiving genuine interest from chemists, educators and policymakers alike. The ongoing development was temporarily halted to allow for the writing of this thesis and the application will therefore contain some deprecated constructs and features that are being reworked. This chapter covers the application structure of VRChem and what improvements we have planned for the future.

5.1 Scene structure

VRChem consists of a single scene with four primary parent objects. We have attempted to maintain a logical object hierarchy in the application by separating objects and scripts by their function into four groups:

1. A virtual room
2. Cameras and controllers
3. Graphical User Interface (GUI) system
4. The virtual molecule

A more detailed explanation of these groups is available in the following subsections.

5.1.1 The virtual room

The virtual room is a large, empty cube with a single light-emitting object inside, pictured in figure 5.1. The purpose of the room is simply to give the user a virtual environment to work in, rather than floating in an empty void. Having this room might not be necessary, but some test subjects did report that removing the room and especially the floor made them feel uneasy.

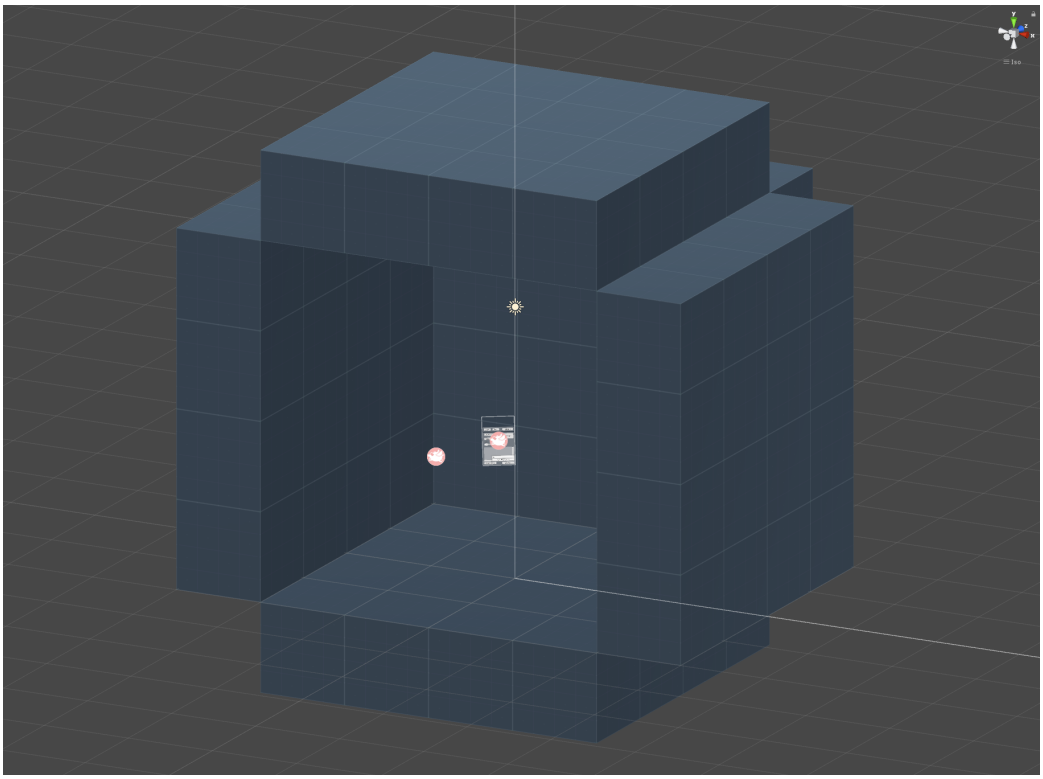

Figure 5.1: A view of the virtual room displayed in the Unity editor. The front wall of the room is hidden in this picture. The dimensions of the room are 2-3 times larger than the user's height.

5.1.2 Cameras and controllers

This parent object groups together all of our supported human interface devices (HID): a keyboard, mouse and screen, a head-mounted display with a Leap Motion hand tracking device, and a head-mounted display with hand controllers. The parent object has a script attached to it that enables switching between the HIDs mid-execution.

5.1.2.1 Head-mounted display with Leap Motion

The HMD with Leap Motion hand-tracking device was our first supported HID. The object is a modified prefab, i.e. a ready-made Unity object complete with components and properties, from the Leap Motion Unity plugin. It contains a camera object with head tracking for rendering, a pair of virtual hands simulating the user's real hands, and an event system and listener for calling methods when certain hand gestures are used. Figure 5.2 shows a user building and rotating a molecule in VR using the Leap Motion HID.

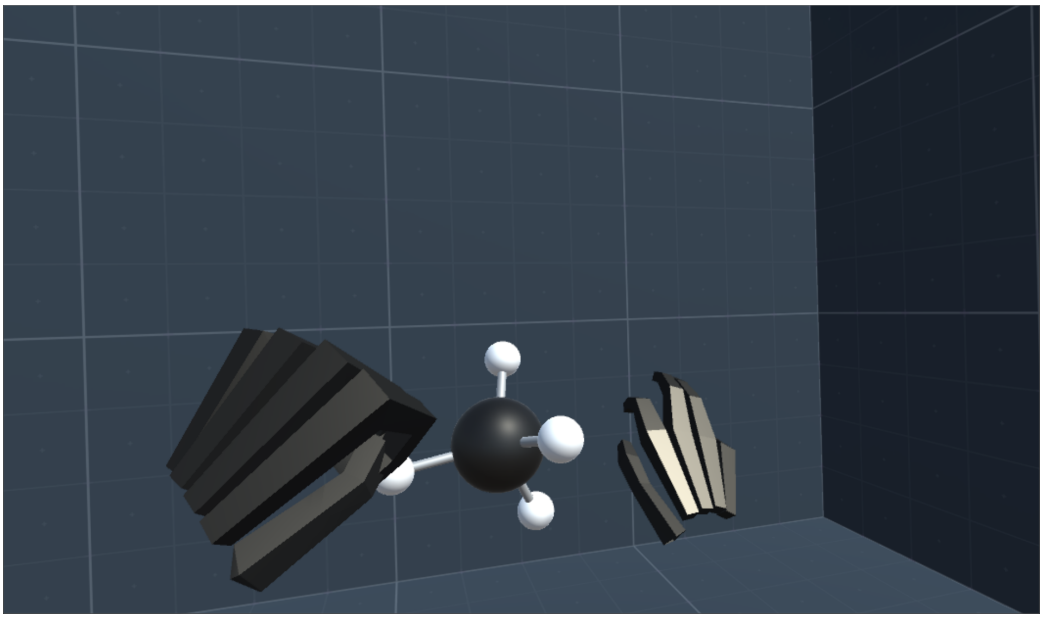

Figure 5.2: An image of the user's view in VRChem showing the Leap Motion virtual hands and a methane molecule. The field of view in the picture is significantly narrower than the view rendered in the HMD.

The virtual hands consist of two objects each. One object has a low-polygonal mesh of a hand and a mesh renderer and will be visible for the user. The other object, called "attachments", is not rendered and is responsible for tracking the user's hand gestures. We have noticed that only a few gestures are recognized with satisfactory accuracy and reliability and for this reason have chosen to use the following gestures in VRChem:

- Pinching or touching one's index finger and thumb together
- Using an extended index finger to touch objects
- Using a clenched fist to grab objects

- Using an extended index finger and thumb in a finger gun gesture to touch objects

At this stage, all gestures have a defined action they perform and cannot be edited by the user. They are also hand-specific and for example the pinch gesture only works with the user's right hand. A future update should at least allow the user to switch gestures between hands for more comfortable use for both left-handed and right-handed users.

The pinching gesture is used for creating new atoms and bonds in the model. To reduce misinterpreted pinching gestures, we chose to also track position of the user's three other fingers by adding a logic gate to the pinch gesture: a pinch event is only propagated if the user's index finger and thumb touch while all other fingers are fully extended. While this does reduce accidental pinch events, we noticed in user testing that the gesture is unintuitive and many subjects quickly forget to extend their three fingers. We would like to try out other gestures for creation or adding some color or other effects to the current gesture to guide users.

The extended index finger is used to select atoms and bonds in the molecule and has been proven to be a reliable and easy to use gesture even with minimal guidance. It is also used to press buttons and other UI elements in our VR graphical user interface (GUI).

We use the clenched fist gesture to grab and move atoms with the right hand, and to grab and rotate the whole molecular structure with the left hand. We have not spotted any major issues with this gesture and all test subjects have been able to reliably rotate and move atoms with their hand.

Finally, the finger gun gesture is an example of the limitations of this control scheme. We use the gesture for deleting objects by touching objects with the tip of the index finger, and while detection of this gesture is solid and it causes fewer misinterpretations that an extended thumb gesture we used previously, it is probably not an intuitive gesture that a user could figure out without help. For this reason, frequent testing of VR applications and interfaces might be more important than with desktop applications.

5.1.2.2 Head-mounted display with hand controllers

We decided to add support for VR controllers after conducting our user tests, where we noted that the freedom of movement provided by the Leap Motion did not help users in accuracy or speed in our test cases. The obvious downside of the Leap Motion is the lack of tactile feedback that buttons and other physical devices have. We speculated that combining the 6 DOF tracking of the handheld controllers with their physical buttons could make

for a more accurate control method compared to the pure hand tracking. However, we have not yet done this comparison as we are still refining both control schemes.

For the controller, we chose a point-and-click type control scheme utilizing raycasts. Between each frame, we check if a ray sent from the controller collides with an atom, bond or GUI element. When a button press event is intercepted, we check if the ray is currently colliding with an object and handle the situation depending on the object and what button was pressed.

This control method has been developed for the Vive Wand controllers, shown in figure 5.3, and has not yet been tested on other equipment. On the Wand, the trigger button functions like the pinching gesture and is used to edit atoms and activate GUI buttons. The grip button equates to the clenched fist gesture used with Leap Motion and is used to rotate the molecule. Lastly, pressing the top sector on the trackpad is used to remove objects while pressing the bottom sector selects bonds.

Figure 5.3: Picture of a HTC Vive Wand controller with labels for the buttons primarily used in VRChem.

Since we have yet to do user testing with the Leap Motion and the VR controller, we have to evaluate the controllers from a personal perspective. In my opinion, the tactile feedback of physical buttons is vastly more accurate than the gesture detection and pointing the controller at objects is a more precise way to select objects than attempting to touch them with my virtual hands. The controller can also provide a larger set of interactions by using button combinations and different modes.

5.1.2.3 Keyboard and mouse

The keyboard and mouse interface was created for Dhinakaran's UI testing [1] but has not been a focal point of this prototype. The HID object has a main camera for rendering a picture on a regular screen and an event listener for molecule and UI manipulation. The event system uses raycasts to determine if the user clicks on an object and modifier keys to determine what action is to be taken. For example, if the user holds down the C key while clicking, a new atom is created, while pressing D will delete the pointed atom. While functional, this system is in its current form not very intuitive and requires a rework.

5.1.3 Graphical User Interface (GUI) system

The GUI parent object contains all of our GUI elements. We designed a conventional overlay for desktop use, however this has been disabled in a recent update and will be redesigned in the future. Dhinakaran also designed and developed a menu for VR use and UI testing, which is pictured in figure 5.4.

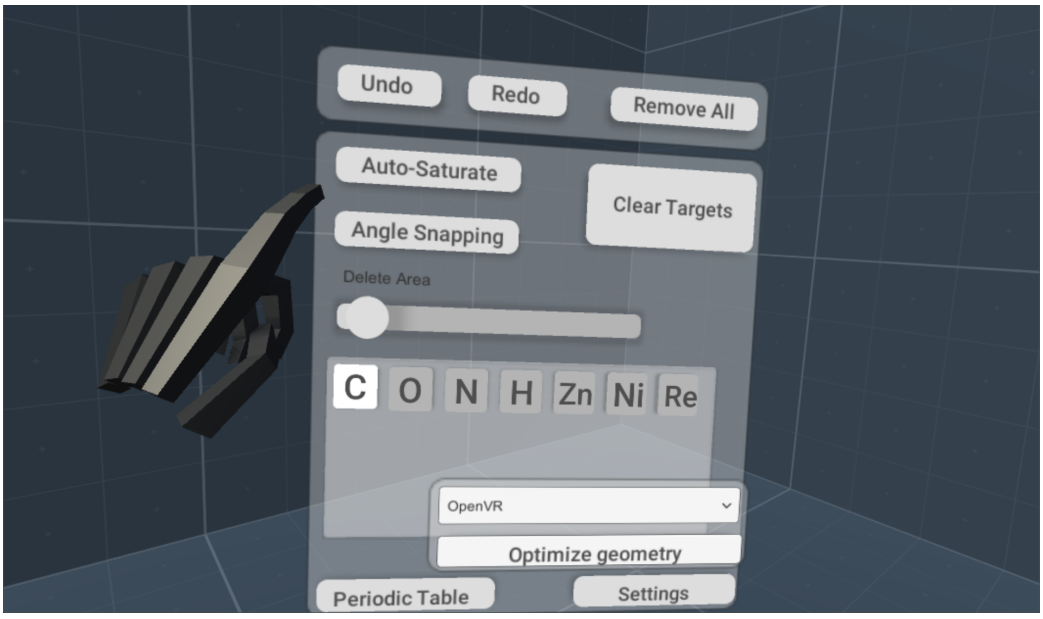

Figure 5.4: Picture of the floating menu used with Leap Motion hand tracking. The buttons are activated by touching them with the virtual hands, but the lack of physical feedback often makes this difficult.

The VR menu consists of a regular Unity canvas and a few canvas objects for settings and other functions, mainly for selecting an element for building

molecules and toggling settings. We can use a single menu for both Leap Motion and VR controllers since the Leap Motion Unity plugin and the Virtual Reality Toolkit add support for canvas object interactions. Developing separate menu layouts for the different controllers is under consideration. When using a Leap Motion, the menu is set to float at a set distance from the user camera and in a fixed direction. This way, the user has an unobstructed view when looking forward, and can access the menu by turning their head slightly. This system works, but users occasionally lose track of the direction of the menu. Discussions with other VR game and application developers have brought up alternative menu types that we would like to test. In controller mode the menu canvas is set to float above one of the controllers, and the user can choose which controller the menu is attached to or freeze the canvas to float in space. This UI is supposed to resemble a painter's palette, from where the user can pick atoms to place.

5.1.4 The virtual molecule

The last parent object, internally called "AtomsBonds", contains all chemistry-related scripts and all atom and bond objects are created as its children. The AtomsBonds object is designed to be HID-agnostic and relies on easily adaptable events and delegates to make adding new control schemes quick.

All atoms and bonds are created as children of the AtomsBonds object. This makes rotating the whole molecule easy, since we can simply rotate the parent object, but manipulating only a part of the atoms requires additional work. We approached this problem by adding an empty child object to AtomsBonds and when the user tries to e.g. rotate half of a molecule, we find the individual atom and bond objects the user's action will affect, move them temporarily to this child object and rotate the child instead. Our search algorithm for the affected atoms and bonds resembles a tree search, where we hop along the atom chain following the bonding data of the atoms.

We have created prefabs for the atoms and bonds, so that new objects could be created by instantiating new prefab instances. The prefabs have a primitive mesh (a sphere for atoms and a cylinder for bonds) and a material for rendering purposes and a script for managing the atom or bond object. The scripts hold properties and bonding data for the object, and contains methods for editing the object.

The AtomManagerScript contains elemental data including element name, symbol, covalent radii, VSEPR parameters and RGBA color values, a list of predicted bond orientations, and a list of bonds and bonded atoms. The manager script has methods for e.g. adding and removing bonds from the atom and safely deleting the atom.

The BondManagerScript holds references to the atoms in the bond as well as the multiplicity, direction and length of the bond. Methods in the script can be used to e.g. set and edit the bond length and to safely delete the bond.

Currently, each instantiated atom and bond contains a copy of associated manager script. This approach has worked so far, but could become a problem in larger molecules. For example, at the moment each atom individually stores some data about its chemical properties. Since this data is identical for atoms of same elements, duplicating the data is unnecessary. Also, we could move the bonding data, which is currently stored in individual lists for each atom, to a single large matrix to reduce memory use. Completely rewriting or even removing the object managers and using a centralized script instead is a serious consideration.

The AtomsBonds object also holds most of our actual program logic, which is explained in the next section.

5.2 Program logic

The primary features of the current version of VRChem are creating atoms and bonds, removing them, editing parts of the molecule and using Open Babel to optimize the structure in real time. All features are called by a device-specific event listener, which receives events from the current control device and then calls the appropriate logic from below.

5.2.1 Build logic

Creating atoms is a two-part process with a StartCreate() method and an EndCreate() method. The outcome depends on which objects are targeted when the methods are invoked. The process is illustrated in figure 5.5.

If the initial target is null, we create a new atom object in the pointed location and attach it to the hand or controller with a fixed joint. This allows the user to move the atom around the scene until a release event is detected. Else, if the initial target is a bond, the bond multiplicity is increased to a double or triple bond or cycled back to a single bond. Lastly, if the initial target is an atom, a temporary atom object is created at the target atom and bonded to the target.

Releasing the creation button or gesture calls EndCreate() to finalize the building logic. If the initial or second target was null, the temporary atom is detached from the input device model and saturated with hydrogen atoms if the setting is turned on and the necessary chemical data is available. Since

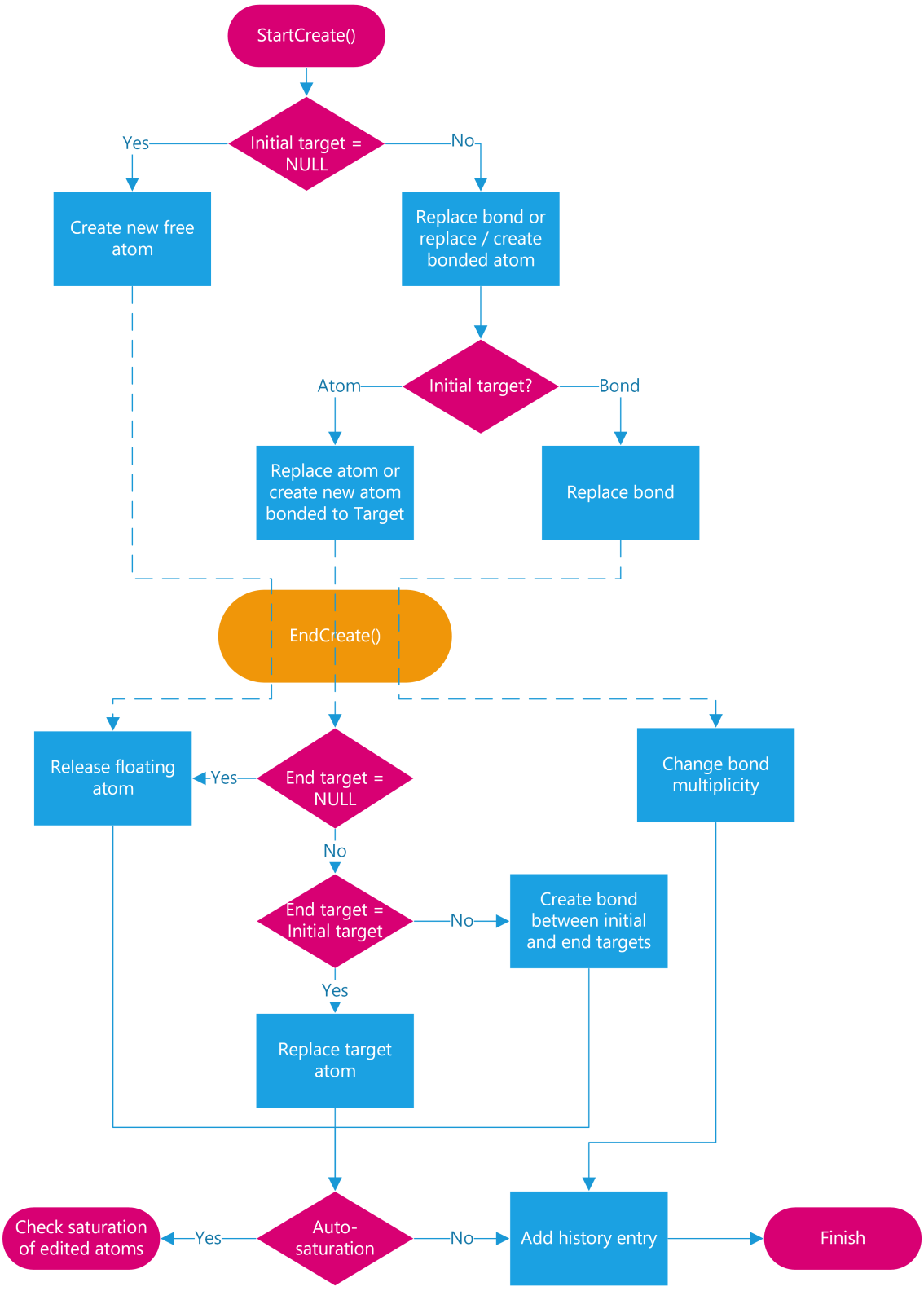

Figure 5.5: Flow chart of the object creation process.

this has not been a priority, only C, H, N and O elements can be saturated in the current version. Else, if the second target is the same atom as the initial target, the target atom is replaced with an atom of selected type. Lastly, if the second target is an atom but not the initial target, a new bond is added between the target atoms. In both of the last two cases the temporary atom object is destroyed and saturation of both targets is checked.

At the end of the build process as well as some other processes we create a history entry for an undo/redo feature. The current implementation saves the position and element type of each atom at every history step in a list. The feature could be improved by only saving atom positions for atoms that have moved since the last history entry. This should decrease memory use as well.

Atom saturation and a rough geometry optimization is done using valence shell electron pair repulsion (VSEPR) theory if the saturate-toggle is on. This process is illustrated in figure 5.6.

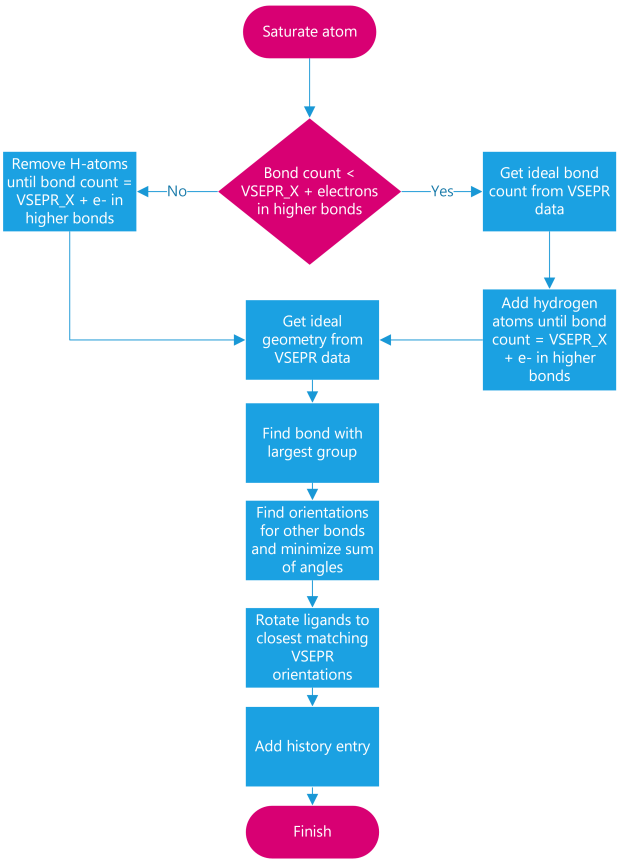

Figure 5.6: Flow chart of the atom saturation process.

VSEPR predicts molecular geometry by presuming that the electron pairs on the valence shell of an atom assume such orientations that the repulsion forces between electron pairs is minimal. The ideal orientation can be looked up from a table by using the atom's electron configuration. In our saturation process we first check if the current bond count for an atom is above or below the expected amount. Hydrogen atoms are then added or removed until the correct value is reached. Once the atom has a correct bond count, the predicted VSEPR orientation is looked up from a data table. Next, the bonds are rotated to the predicted orientations by first finding the bond with the largest ligand and assigning it to the first orientation in the VSEPR geometry. This bond will be stationary for the rest of the process and is frozen to reduce unnecessary movement on the screen. For the remaining bonds, we calculate the sum of angles between current and ideal orientations for different bond indexing configurations and choose the configuration with the smallest sum of angles. In the final step, the remaining bonds are rotated to these closest orientations and the bond lengths are calculated and adjusted based on the covalent radii of the bonded atoms.

Removing objects also depends slightly on the target object. When removing objects from a scene, all references to the object should be purged beforehand. If the target is a bond, the bonded atoms are updated and set to not share a bond before the object is destroyed. If instead the target is an atom, each bond the atom has must first be removed. This process is illustrated in figure 5.7.

5.2.2 Editing logic

VRChem allows users to edit bond lengths, rotate atom groups around bonds and modify dihedral bond angles using two gestures. The selection process is based on a context-sensitive delegate logic, which is illustrated in figure 5.8.

A delegate can be described as a variable for methods. A delegate can be set to "contain" a method, and calling the delegate will execute the contained method. In this selection logic, the event listener script calls SelectionDelegate when the appropriate gesture is detected, while the method SelectionDelegate points to depends on the state of the application. Initially, when nothing is selected, selecting a bond will call FirstSelection. The target bond is set as the base bond for modification and an outline is drawn around the bond to indicate a successful selection. In addition to the outline, a bond length value is displayed next to the bond and modifying the bond length is enabled. Grabbing one of the atoms in the base bond will search for and group together the bonds and atoms connected to the grabbed atom and allows the user to modify the selected bond length while preserving the

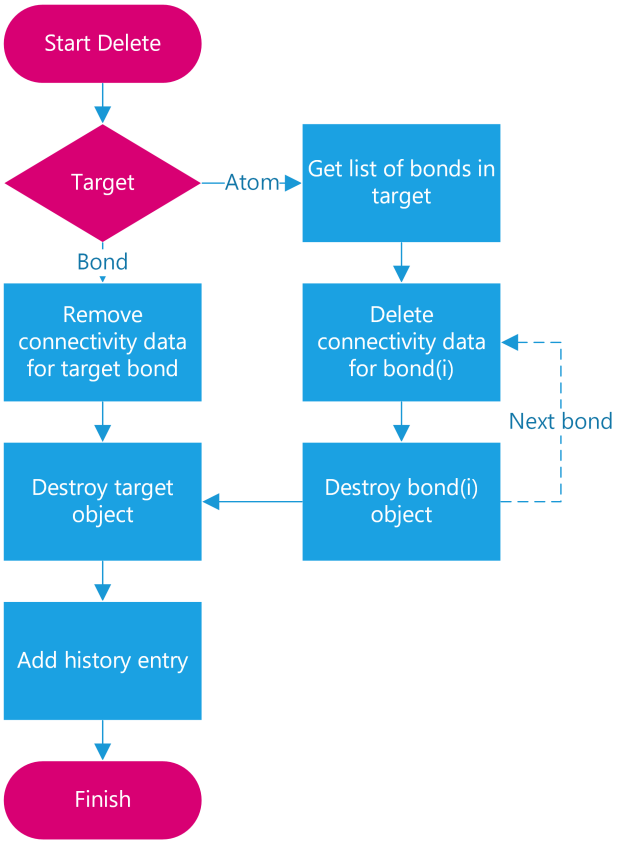

Figure 5.7: Flow chart of handling object removal.

rest of the molecular structure. When a base bond has been selected, the SelectionDelegate is set to point to SecondSelection.

Selecting a second object enables rotation editing. If the second selection target is one of the atoms in the base bond, the highlight is extended to the atom and a rotation indicator is drawn around the base bond. A grabbing gesture will group together atoms connected to the selected atom and the atom group can be rotated around the axis of the base bond. However, if the second selection target is a bond adjacent to the base bond, the user can grab and modify the dihedral angle between the two selected bonds. In both cases, SelectionDelegate is set to ResetSelection and selecting a third object will then remove all highlights and measurement indicators, deselect all bonds and atoms and reset the SelectionDelegate to FirstSelection.

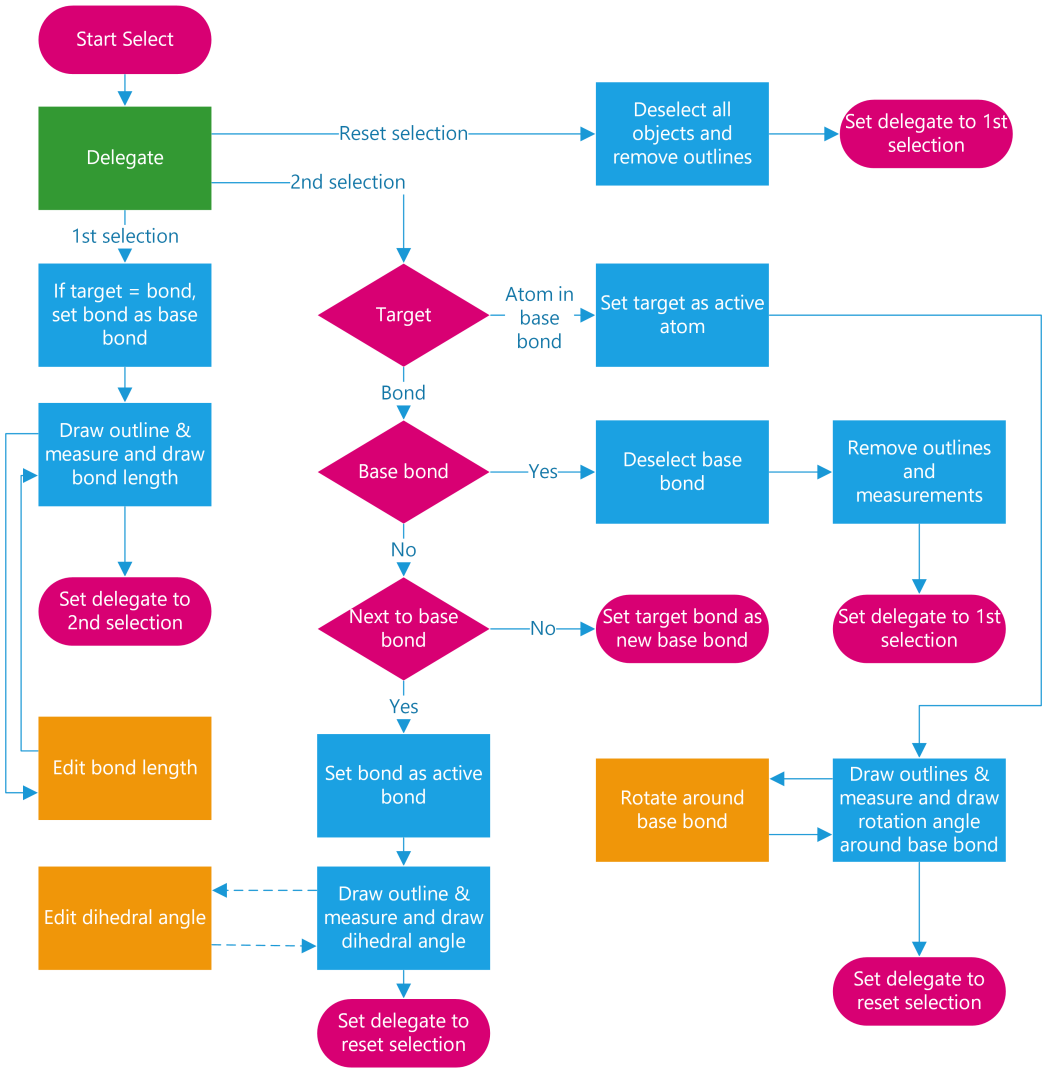

Figure 5.8: Flow chart of the selection logic using a context-sensitive delegate.

5.2.3 Structure optimization logic

In the saturation phase we used VSEPR and covalent radii to build a rough, idealized structure for the molecule. The resulting structure is sometimes suitable for simple visualization tasks but is rarely accurate. The next step towards more realistic models is using empirical force field based chemical models to optimize the geometry. These methods are not excessively heavy to compute and can be iterated at 60 Hz, which is our approximate target framerate. We used the Open Babel chemical toolkit to integrate a continuous structure optimization loop, which is illustrated in figure 5.9.

The loop runs as long as the optimize toggle is set in the GUI and updates the positions of all atoms between each frame. The loop begins by checking if the molecule has been altered by the user since the last update. This is a necessary step because the force field has to be re-initialized if bonding in the molecule has been altered, since most force fields do not take into account bond formation or breaking, unlike quantum chemical methods. A change in the molecule is indicated by a Boolean hasChanged flag, which is set to true at the end of all atom or bond creation or destruction methods. If the flag is set, we chose to completely remove the force field and reconstruct it from the current molecule state since it does not seem to cause any performance issues at our test scale of around 100 atoms. Reconstruction begins by converting the native atom and bond classes of VRChem to OBAtoms and OBBonds used in Open Babel. This is a mostly straightforward task because both classes are coincidentally nearly identical. In the next step the force field is loaded and initialized. The current choise, mmff94, cannot be changed by the user but will be in an upcoming version. If the force field initializes without errors, the OBAtoms and OBBonds are combined into an OBMol molecule class and validated. The initialized force field is used to move the atoms in OBMol one iteration step towards an energy minimum, after which the new OBAtom positions are read and updated to the VR scene. The optimization coroutine then yields and loops back to start after the next frame has been rendered. If no changes have been made to the molecule, execution skips directly to the iteration step. This force field optimization runs well even at high framerates and does not cause any noticeable lag or slowdowns in the application performance.

In the next chapter we assess if our prototype meets the set development goals, how functional it is and how we aim to improve the application.

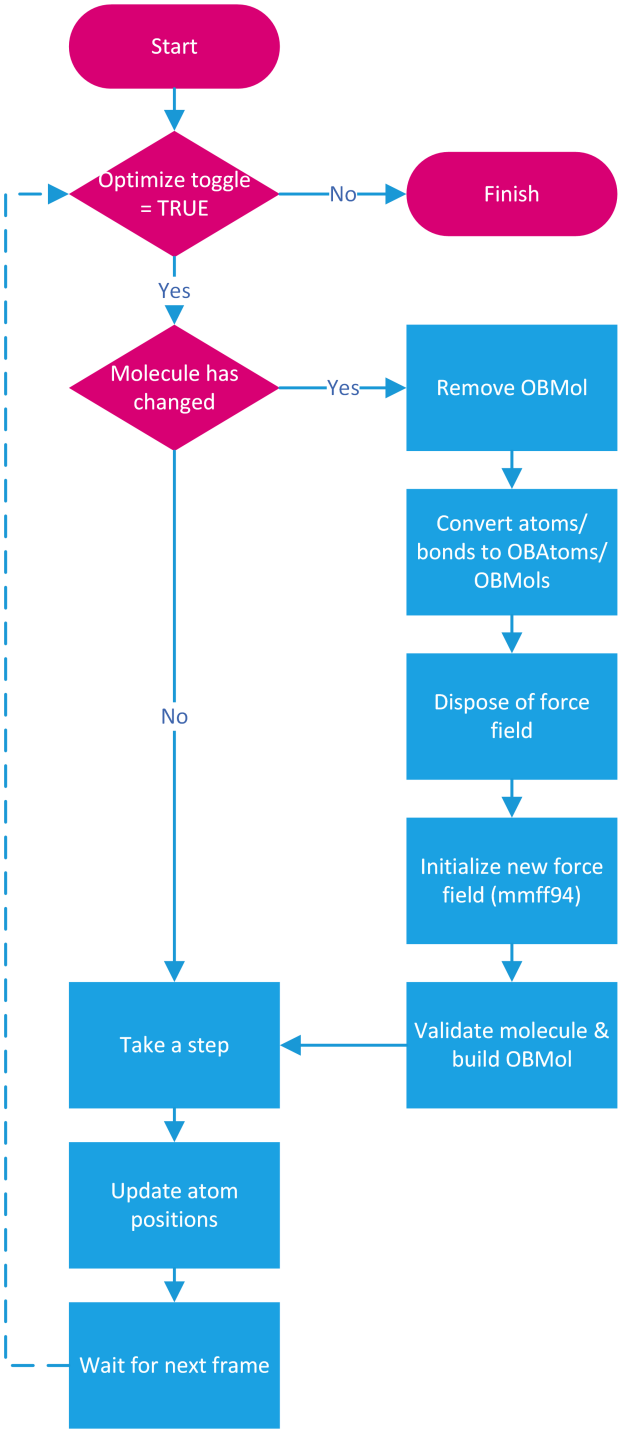

Figure 5.9: Flow chart of the real-time structure optimization logic.

Chapter 6

Evaluation of practicality

VRChem has been in active development for 9 months and is steadily improving in usability and receiving new features. Our current feature set, which contains all of our initial development goals, covers most basic use cases and is almost equal to feature sets of commonly used modeling applications like Avogadro[45] or JMol[46], although our lack of options for molecule representation needs to be addressed. Crucially, VRChem is among the first applications to support molecular modeling and editing in VR.

We would consider VRChem a successful prototype or tech demo and are content with its current status as a test platform for molecular modeling in VR. In UI testing the test subjects were able to complete building and editing tasks on average only 25% slower than with a traditional interface, and most reported that using the VR application was enjoyable[1]. We expect the speed to increase as familiarity with VR environments and UIs increases and as the quality of the UI itself improves. We have also demonstrated the application at various venues and have noticed genuine interest towards our work. However, VRChem is certainly not ready for professional use in any capacity.

6.1 Future development goals

One of the most important unknowns at this stage is measuring the limits of the application and extending them. We have not tested how VRChem handles large molecules but expect the maximum functional atom count to be lower than what other applications can handle. This could be caused by e.g. unoptimized memory use or an excessive polygon count. We are planning to optimize the code base for better performance soon.

We also have plans for new features that could make VRChem more usable

or useful for actual research work. We have sketched several alternative GUIs for VR that we wish to test and compare to the current floating menu. Moreover, deeper integration with the Open Babel API would allow us to use the format conversion and other tools in Open Babel. We would also like to develop VRChem more towards computer aided molecular design and experiment with combining CAMD with virtual reality. While some recent quantum mechanical methods are approaching suitable iteration rates for proper real time use, the reactive force field model ReaxFF[47] might be more suitable for the computing power and rendering framerate we are aiming at. This is unless the actual quantum chemical calculation could be efficiently offloaded to a network connected parallel computing server.

I am personally elated with VRChem and its development. During the project I learned C# and Unity, neither of which I had prior experience in, and am fully committed to continuing the development and improve on our work. I believe VRChem or a similar application could have real value for chemists in the future.

Chapter 7

Conclusions

Virtual reality is quickly becoming commonplace with the advent of inexpensive consumer grade VR devices and growing number of VR applications, and we believe that VR or AR could be helpful for chemical research like drug discovery and material design, as well as for teaching purposes. For this reason, we set out to design and develop a new molecular modeling and visualization application with VR support, and created VRChem.

While molecular visualization applications with VR support have been developed before, VRChem appears to be the first one to include support for building and editing molecules in VR. We have created a novel, fully functional application suitable for common modeling tasks with three distinct interfaces for UI testing purposes. Although the current version of VRChem is strictly a prototype, the development process has given us valuable insight into VR software development and UI requirements and will help us create an improved version in the future.

"VRChem2" will be more efficient and can handle larger molecules, and will be used to test more intuitive GUI layouts and human interface devices. We will also attempt to introduce computer aided molecular design into the application by adding reactive energy minimization routines and other features. The outcome of this project is a good starting point for an actual, production-ready VR molecular modeling application.

Bibliography

[1] Krupakar Dhinakaran. "VR-Chem - Developing a Virtual Reality Interface for Molecular Modelling". MA thesis. Aalto University, 2017.

[2] Jonathan Clayden et al. *Organic chemistry*. Oxford University Press, 2001.

[3] Eric Francoeur. "Cyrus Levinthal, the Kluge and the origins of interactive molecular graphics". In: *Endeavour* (26(4) 2002). DOI: 10.1016/S0160-9327(02)01468-0.

[4] Eric Francoeur. "The Forgotten Tool: The Design and Use of Molecular Models". In: *Social Studies of Science* (27:7 1997). DOI: 10.1177/030631297027001002.

[5] Jon Peddie. *The History of Visual Magic in Computers. How Beautiful Images are Made in CAD, 3D, VR and AR*. Springer, 2013.

[6] Cyrus Levinthal. "Molecular Model-Building by Computer". In: *Scientific American* (214(6) 1966).

[7] Hans Wallach and D. N. O'Connel. "The kinetic depth effect". In: *Journal of Experimental Psychology* (45(4) 1953). DOI: 10.1037/h0056880.

[8] Marco Tarini, Paolo Cignoni, and Claudio Montani. "Ambient Occlusion and Edge Cueing to Enhance Real Time Molecular Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* (12(5) 2006).

[9] Marco Tarini, Paolo Cignoni, and Claudio Montani. "Ambient Occlusion and Edge Cueing for Enhancing Real Time Molecular Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 12.5 (2006), pp. 1237–1244. ISSN: 1077-2626. DOI: http://dx.doi.org/10.1109/TVCG.2006.115.

[10] Maria Limniou, David Roberts, and Nikos Papadopoulos. "Full immersive virtual environment CAVE in chemistry education". In: *Computers & Education* (51:2 2008). DOI: 10.1016/j.compedu.2007.06.014.

[11] Magnus Norrby et al. "Molecular Rift: Virtual Reality for Drug Designers". In: *Journal of Chemical Information and Modeling* (55 2015). DOI: 10.1021/acs.jcim.5b00544.

[12] Noel O'Boyle et al. "Open Babel: An open chemical toolbox". In: *Journal of Cheminformatics* (3:33 2011). DOI: 10.1186/1758-2946-3-33.

[13] Thomas A. Halgren. "Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94". In: *Journal of Computational Chemistry* 17 (5-6 1996). DOI: 10.1002/(SICI)1096-987X(199604)17:5/6<490::AID-JCC1>3.0.CO;2-P.

[14] A. K. Rappé et al. "UFF, a Full Periodic Table Force Field for Molecular Mechanics and Molecular Dynamics Simulations". In: *Journal of the American Chemical Society* (114 (25) 1992). DOI: 10.1021/ja00051a040.

[15] Austin Acton et al. *The ghemical homepage*. URL: www.bioinformatics.org/ghemical/ghemical/index.html.

[16] Mark Wilson. *Exclusive: Microsoft Has Stopped Manufacturing The Kinect*. URL: https://www.fastcodesign.com/90147868/exclusive-microsoft-has-stopped-manufacturing-the-kinect.

[17] *Molecular Rift v2 GitHub*. URL: https://github.com/JBostrom/MolecularRiftv2.

[18] *Molecular Vive GitHub*. URL: https://github.com/UoA-eResearch/MolecularVive.

[19] Andrea Salvadori et al. "Immersive virtual reality in computational chemistry: Applications to the analysis of QM and MM data". In: *International Journal of Quantum Chemistry* (116 2016). DOI: 10.1002/qua.25207.

[20] Paul Richard et al. "Effect of Frame Rate and Force Feedback on Virtual Object Manipulation". In: *Presence: Teleoperators and Virtual Environments* (5:1 1996). DOI: 10.1162/pres.1996.5.1.95.

[21] *SMART - Visualization and Molecular Dynamics*. URL: http://smart.sns.it/?pag=vmd.

[22] Min Zheng and Mark Waller. "ChemPreview: an augmented reality-based molecular interface". In: *Journal of Molecular Graphics and Modelling* (73 2017). DOI: 10.1016/j.jmgm.2017.01.019.

[23] *ChemPreview Github*. URL: https://github.com/wallerlab/chem-preview/.

[24] Moritz Haag and Markus Reiher. "Real-time quantum chemistry". In: *International Journal of Quantum Chemistry* (1 2013). DOI: 10.1002/qua.24336.

[25] Moritz Haag et al. "Interactive Chemical Reactivity Exploration". In: *ChemPhysChem* (15 2014). DOI: 10.1002/cphc.201402342.

[26] Alain Vaucher, Moritz Haag, and Markus Reiher. "Real-Time Feedback from Iterative Electronic Structure Calculations". In: *Journal of Computational Chemistry* (37 2016). DOI: 10.1002/jcc.24268.

[27] Todd Martínez. "Ab Initio Reactive Computer Aided Molecular Design". In: *Accounts of chemical research* (50 2017). DOI: 10.1021/acs.accounts.7b00010.

[28] Nathan Luehr, Alex Jin, and Todd Martínez. "Ab Initio Interactive Molecular Dynamics on Graphical Processing Units (GPUs)". In: *Journal of Chemical Theory and Computation* (11 2015). DOI: 10.1021/acs.jctc.5b00419.

[29] Ross Walker and Andreas Götz. *Electronic Structure Calculations on Graphical Processing Units*. Wiley, 2016.

[30] *Unity 2017: The world-leading creation engine for gaming*. URL: https://unity3d.com/unity.

[31] *Unity for VR and AR*. URL: https://unity3d.com/unity/features/multiplatform/vr-ar.

[32] Mikko Kytö et al. "Improving 3D Character Posing with a Gestural Interface". In: *IEEE Computer Graphics and Applications* (37(1) 2017). DOI: 10.1109/MCG.2015.117.

[33] *Vive VR Features*. URL: https://www.vive.com/us/product/vive-virtual-reality-system/.

[34] *How Does the Leap Motion Controller Work?* URL: http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/.

[35] *Mono, cross platform, open source .NET framwork*. URL: http://www.mono-project.com/.

[36] Adam Tuliper. *Unity : Developing Your First Game with Unity and C# - MSDN Magazine Blog*. Aug. 2014. URL: https://msdn.microsoft.com/en-us/magazine/dn759441.aspx.

[37] *Unity Documentation - Execution order of event functions*. URL: https://docs.unity3d.com/Manual/ExecutionOrder.html.

[38] *Events (C# Programming guide)*. URL: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/events/.

[39] *Unity Manual: Plugins*. URL: https://docs.unity3d.com/Manual/Plugins.html.

[40] *Steam VR Unity Plugin Github*. URL: https://github.com/ValveSoftware/steamvr_unity_plugin.

[41] *Virtual Reality Toolkit Github*. URL: https://github.com/thestonefox/VRTK.

[42] *Leap Motion Unity plugin*. URL: https://developer.leapmotion.com/unity.

[43] *OpenBabel Wiki*. URL: https://openbabel.org/wiki/Main_Page.

[44] Walter L. Koltun. "Space filling atomic units and connectors for molecular models". US3170246 A. 1965.

[45] *Avogadro: an open-source molecular builder and visualization tool. Version 1.2.0*. URL: http://avogadro.cc/.

[46] *Jmol: an open-source Java viewer for chemical structures in 3D*. URL: http://www.jmol.org.

[47] Adri C. R. van Duin et al. "ReaxFF: A Reactive Force Field for Hydrocarbons". In: *The Journal of Physical Chemistry A* (105(41) 2001). DOI: 10.1021/jp004368u.

Figure sources

Figure 2.1.	Levinthal C., from *Molecular model-building by computer* [6].
Figure 2.2.	Pietikäinen O., own work.
Figure 2.3.	US Department of Energy, US Public Domain (`https://commons.wikimedia.org/wiki/File:U.S._Department_of_Energy_-_Science_-_393_007_005_(9558079932).jpg`).
Figure 2.4.	Public Domain.
Figure 2.5.	Norrby, M., Grebner, C., Eriksson, J., & Boström, J., from *Molecular Rift: virtual reality for drug designers* [11]
Figure 2.6.	Salvadori A., Del Frate G., Pagliai M., Mancini G. & Barone V., from *Immersive virtual reality in computational chemistry: Applications to the analysis of QM and MM data* [19].
Figure 2.7.	Zheng H. & Waller M. P., from *ChemPreview: an augmented reality-based molecular interface* [22].
Figure 2.8.	Martinez, T. J., from *Ab Initio reactive computer aided molecular design* [27].
Figure 3.1.	Pietikäinen, O., own work.
Figure 5.1.	Pietikäinen, O., own work.
Figure 5.2.	Pietikäinen, O., own work.
Figure 5.3.	Cropped and edited from an image by CES2016_HTCVive_Pre_Winters, CC BY 2.0. (`https://www.flickr.com/photos/92587836@N04/24177102722/`)
Figure 5.4.	Pietikäinen, O., own work.
Figure 5.5.	Pietikäinen, O., own work.
Figure 5.6.	Pietikäinen, O., own work.
Figure 5.7.	Pietikäinen, O., own work.

Figure 5.8. Pietikäinen, O., own work.
Figure 5.9. Pietikäinen, O., own work.