FAKULTÄT FÜR !NFORMATIK

Dynamic Differential Geometry in an Educational Augmented Reality Application

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Medieninformatik

eingereicht von

Marie-Theres Tschurlovits
Matrikelnummer 0125975

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuer: Univ. Ass. Mag. Dr. Hannes Kaufmann

Wien, 20.08.2008

(Unterschrift Verfasserin) (Unterschrift Betreuer)

Abstract

In this thesis a number of geometry software packages leading both to static and dynamic constructions and their particular features will be presented. Afterwards Construct3D [14] a 3D dynamic geometry construction tool will be introduced. It is based on the Augmented Reality System Studierstube. Construct3D's greatest advantage compared to other dynamic geometry software is the possibility for users to see the real environment augmented with virtual content with the aid of a head mounted display. That gives the users, mainly high school and university students, the opportunity to actually construct, explore and interact with three dimensional content in "real" 3D space.

The practical part of this thesis was the implementation of a number of new functions for Construct3D. Several tools have been developed to enhance the understanding of the term curvature of curves and surfaces. To complement the already available sweep function of Construct3D helical and general sweeps have been implemented.

Kurzfassung

In dieser Arbeit wird eine Reihe von Geometrie Software Paketen präsentiert, die sowohl zu statischen als auch zu dynamischen Konstruktionen führen. Anschließend wird Construct3D eingeführt, eine Software zu Erstellung dynamischer 3D Konstruktionen, die auf dem Augmented Reality System Studierstube basiert. Der größte Vorteil von Construct3D gegenüber anderen dynamischen Geometrie Software Paketen ist die Option mit Hilfe eines Head mounted displays die echte Umgebung angereichert mit virtuellen Inhalten betrachten zu können. Das eröffnet den Usern, vornehmlich Oberstufenschülern und Studenten, die Möglichkeit im "richtigen" dreidimensionalen Raum dreidimensionale Objekte zu konstruieren, zu erforschen und in weiterer Folge damit zu interagieren.

Der praktische Teil dieser Arbeit umfasste die Implementierung einer Reihe von neuen Funktionen für Construct3D. Es wurden einige Werkzeuge entwickelt um das Verständnis des Begriffs der Krümmung von Kurven und Flächen zu fördern. Um die bereits vorhandenen Sweep Funktionen zu ergänzen wurden außerdem Schraub- und Schiebeflächen implementiert.

Acknowledgements

First of all I would like to thank my supervisor Hannes Kaufmann for his comprehensive support. He supported me in every problem or question I had, also at night, weekends and during holidays. His colleague Mathis Csisinko helped me when problems with Construct3D occured.

I am also grateful to Marion Papp who supported me with geometrical background informations and was one of the first testers of my work.

My partner Thomas Weiß advised me in programming questions and was the man for mental support.

Last but not least I want to thank my parents who gave me the opportunity to study. I know my father is very proud I studied at the university where he has worked himself for so many years.

Contents

1 Motivation

Mathematics and descriptive geometry are subjects in high school which will either be loved or hated. On the one hand there are the students who have the luck to really understand most of what they have to do in these subjects but for most of the students mathematics and geometry are closed books.

Fortunately I was one of the former and so in the 11th grade, I decided to take descriptive geometry instead of an emphasis in natural sciences. Rapidly it became on of my favorite subjects. Because of my ability to visually imagine my tasks I easily understood the subject.

Construct3D, an augmented reality application for mathematics and geometry education, is a tool which was mainly developed to improve the spatial abilities of its future users, high school and university students. The user has the opportunity to actually see in 3D what he only has to imagine in conventional geometry classes. The traditional pen and paper based methods used for construction and even computer based visualizations only provide 2D projections of 3D content. The advantage of computer based approaches over pen and paper based methods is that the students can really concentrate on the construction and geometric understanding rather than to achieve a perfect drawing.

In Construct3D users achieve a real 3D impression of the virtual geometric content projected in a real environment. Users can explore the space of modeling by walking around the constructions and looking at them from different points of view. Construct3D is also a dynamic tool so the user has the possibility to change a part of the construction and in real time the whole constructions also changes. The user gains a deeper insight in the relationship between the parts of the construction. In our geometry room in high school there was a cone for demonstrating the conic sections. It was built out of plastic in different colors and could be disassembled. For the purpose of demonstrating the conic sections it was sufficient and useful but there are other geometric constructions which are more difficult to understand. Sometimes the possibility to dynamically change parameters of the constructions provide a better understanding of the subject.

For understandable reasons choosing a field concerned with geometry education was selfevident for me being a really useful combination of informatics and geometry. Implementing differential geometry functions was interesting because these functions were—up to now—rarely implemented in conventional geometry packages and if, they are not really dynamic. Differential geometry in a dynamic way in an augmented reality application was completely new.

I hope with my implementation of generalized sweep objects and functions of differential geometry I will help future users of Construct3D to gain a deeper understanding of these fields and to have more fun with geometry.

2 Related Work

This section should give an overview over some software packages which are related to the work in this thesis. There will be a focus on dynamic geometry software but there are also connections to other software for differential geometry or CAD applications. In this work first software packages for dynamic geometry for working in 2D and mostly for educational purposes will be introduced. After that follow software packages for three dimensional geometry. Finally some CAD packages which provide differential geometry functions will be presented.

2.1 History of dynamic geometry software

For a very long time geometric constructions were created only with the aid of pencil and ruler. Static visualizations were generated and later changes in the construction were not possible. The instructions at school were augmented only with a few models with flexible parts. These models were expensive and often only the teacher was allowed to work with them. Therefore, the students had mostly no possibility to examine and explore the models. In the seventies of the previous century film material was sometimes used to demonstrate dynamic behaviour. But their development was complex and expensive and the students were again only observers.

With the geometry software of the first generation in the eighties one could do exactly the same things as one could do with pen and paper. Therefore, an added value could not be identified and this type of software was rarely used.

At the beginning of the nineties the first dynamic geometry software packages were developed, e.g. Cabri Géomètre and Geometer's Sketchpad. The characteristic feature was the so-called "drag mode". The base points of a geometric construction could be dragged only with the help of the mouse without the need of keyboard input. This was and still is an intuitive way to change the parameters of a geometric construction dynamically. The interrelationship between the construction's components will not be affected through the change of the base points and at any time the user has a mathematical valid construction. At that point of the development the construction task itself lost its importance and the exploration and examination became the work's main part. The students got the chance to explore the concepts themselves, everybody in its own speed and way. The student's role changed from the passive observer to that of the active user and explorer of geometry. The construction process itself became a lot easier and the remaining tasks were to understand the relationships and to draw the correct conclusions. Another main characteristic of dynamic geometry software is the possibility to generate so-called "loci". Loci originate from the trace of a construction point B when another point A of the construction is moved and affects point B. In the course of time

also the generation of interactive worksheets for students became more important. Instead of starting with a blank worksheet the students get a specific task to solve and can check their results themselves. Often the installation of the software is not needed anymore because most software packages run as applets in a web browser.

Initially dynamic geometry software was developed only for geometry in two dimensions. But at the beginning of the 21st century dynamic geometry software was also developed for 3D. Features and handling are very similar but sometimes more difficult and complex because the mouse as 2D input device has to deal with three spatial directions.

Construct3D as an augmented reality application takes this progress a step further and leads the geometry students to a real 3D environment where they can explore geometry problems in their genuine space. A main goal of the use of dynamic 3D geometry software is to improve the student's spatial abilities. At the moment, there exist several research projects for analyzing the concrete effects of dynamic 3D geometry software on spatial abilities but first tests and projects have already shown that students really benefit. [6, 7]

2.2 Dynamic geometry software

In this section several software packages will be introduced. Neither offers all the important features of Construct3D—three dimensional construction, examination in augmented reality, interactive behavior, changes applying in real time and support of differential geometry—but inspired in some way the concept of Construct3D and its features. Each of these software packages has its own feature set and strengths and could be used for partially solving the problems concerning differential geometry which can both be solved and properly visualized to full extent only with Construct3D by now.

2.2.1 Dynamic 2D Geometry software

Euklid DynaGeo and Cinderella are popular and widely used examples of dynamic 2D geometry software. These two software packages have several features in common mainly that they are interactive and changes are applied in real time. A further relatively new dynamic 2D geometry software is GeoNext [31], developed by the University of Bayreuth, Germany, since 2003. The software package GeoGebra [13] is a so far unique combination of a dynamic geometry software and an algebra software.

Euklid DynaGeo Euklid DynaGeo [20] is a shareware software for dynamic 2D geometry for Windows (see figure 1) developed by Roland Mechling since 1994.

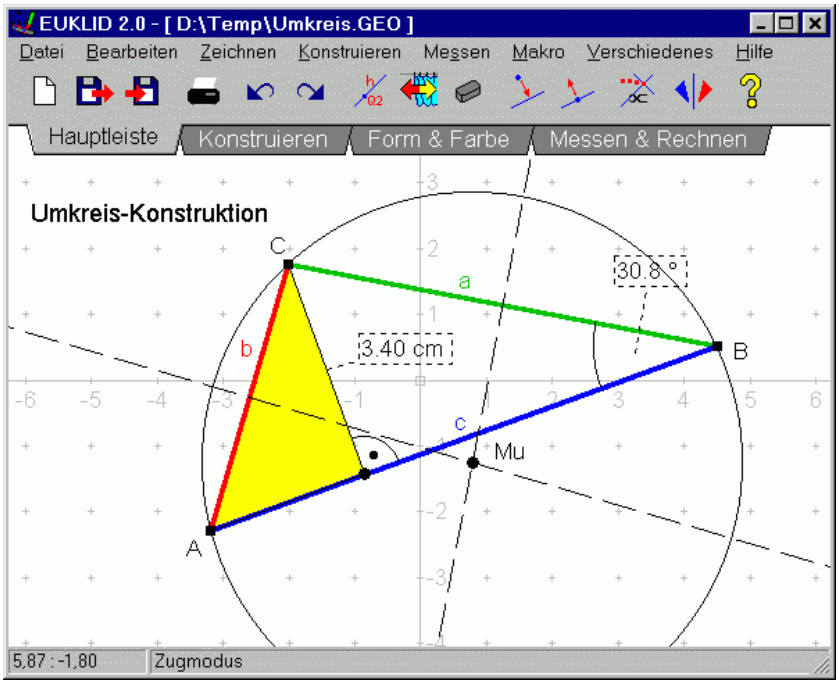

Figure 1: This figure shows a screenshot of Euklid DynaGeo [20]. Below the menu bar there is a tabbed bar with buttons for the main functions, construction, form and color, measurement and calculation. The screenshot shows the construction of a circle's circumcircle Mu. The input elements, the straight lines a, b and c between the points A, B and C are colored in green, red and blue.

This is a popular software for geometry, mathematics and descriptive geometry education, mainly used in the lower grades of high schools. It was one of the first available software packages aiming with its high usability especially at beginners with geometry programs.

As in every dynamic geometry software in Euklid DynaGeo some base points of the constructions can be dragged with the mouse without loosing the interrelationship between the components of the construction. Constructing a triangle's circumcircle it will always pass through all corners of the triangle, regardless where a corner of the triangle is moved.

The generation of dynamic loci gives answers to questions like "how does a point on a bicycle tire move in space when the bike is in motion?" Macros which combine multiple construction steps can be generated in an easy point-and-click manner. All lines and points can be colored and labeled freely. Distances and angles between the objects can be measured and construction dependent terms can be calculated. Animations can be generated and the projects can be exported to various file formats.

Cinderella The Cinderella project [16] is the product of a sequel of three projects done between 1993 and 1998. Cinderella is a dynamic 2D geometry software developed by Ulrich Kortenkamp and Jürgen Richter-Gebert since 1998. It is written in Java and therefore available for Windows, Linux and Mac.

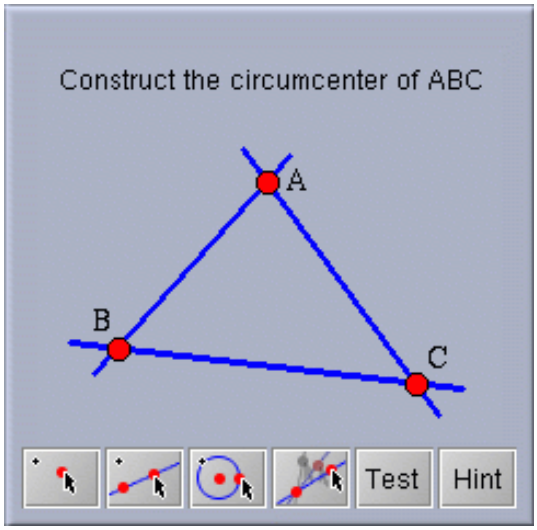

Figure 2: This figure shows an interactive student exercise in Cinderella [25]. The input elements are marked in red. At the bottom there are tools that can be used to solve the exercise "construct the circumcenter of a triangle using only straightedge and compass". There are also buttons for giving hints and for testing the correctness of the student's solution.

In its so-called "drag mode" the parts of a construction can also be moved with the mouse without destroying the interrelations between the components of the construction. No keyboard input is necessary. In its main features Cinderella is very similar to Euklid DynaGeo but provides several additional features. Unlike Euklid DynaGeo Cinderella not only supports Euclidean geometry but also non-Euclidean geometry, like hyperbolic and elliptic geometry.

The problem of jumping elements resulting from ambiguities (e.g. a circle and a line can have none, one or two intersections) was solved in implementing the Cinderella kernel in a complex vector space. To avoid singular situations results of analytic function theory were used. This system established the base of a reliable randomized theorem checking, which tests automatically the correctness of geometric theorems. It became also possible to generate complete geometric loci.

In its actual version 2.0 Cinderella was augmented with several new features which were also motivated by user's suggestions. Cinderella now also supports

some kind of macros. A newly added feature are transformations and transformation groups which shall support the user at more advanced constructions. A quite independent part is the so-called CindyLab, a physics simulation engine which provides a particle, mass and force simulation paradigm. New is also CindyScript, a full-featured, mathematically oriented, high level programming language. An interesting new feature is also the support for the recognition of hand sketches, so Cinderella could be used with pen and tablet, an interactive whiteboard or a PDA. It is also possible to create interactive exercises for students while providing only selected functions that students shall use for solving a problem (see figure 2) [16, 17, 25].

2.2.2 Dynamic 3D Geometry software

Archimedes Geo3D and Cabri 3D are examples for dynamic 3D geometry software. They extend the concept of the software of the previous section to three dimensional space. Meanwhile they share the same basic concepts of real time execution and interaction.

Archimedes Geo3D Archimedes Geo3D [11] has been developed by Andreas Goebel since 2006 and is available for Windows, Linux and Mac (see figure 3). It expands the idea of dynamic geometry to the third spatial dimension. Its idea of moving the base points of a construction, the "drag mode", is the same as for dynamic 2D geometry programs. At the beginning, the handling of a dynamic 3D geometry software is more complicated because the mouse as a 2D input device has to be combined with pushing the keyboard buttons <shift> and <strg> to navigate in three spatial dimensions. Archimedes Geo3D supports also an extended drag mode where all (not only the base objects) can be moved and also rotated freely. The handling of the program is simplified through the possiblity of using keyboard shortcuts.

Like the 2D programs Archimedes Geo3D supports the creation of loci, the traces of points, i.e. curves. Because of the third spatial dimension it is also possible to create traces of curves, i.e. surfaces. Input cannot only be given through the creation of points and basic shapes but also through entering of mathematical terms. Therefore Archimedes Geo3D supports also vector analysis as well as curves and surfaces can be defined through their equations. Further available features are texturing, animation creation and shadow generation. Macros can be used to record multiple construction steps and can also be called recursively. Finally, Archimedes Geo3D supports "real" 3D display with the aid of anaglyph images or shutter glasses. [12]

Cabri 3D Cabri 3D [3] is a dynamic 3D geometry software developed by Cabrilog. The first Cabri software programs were developed in the research

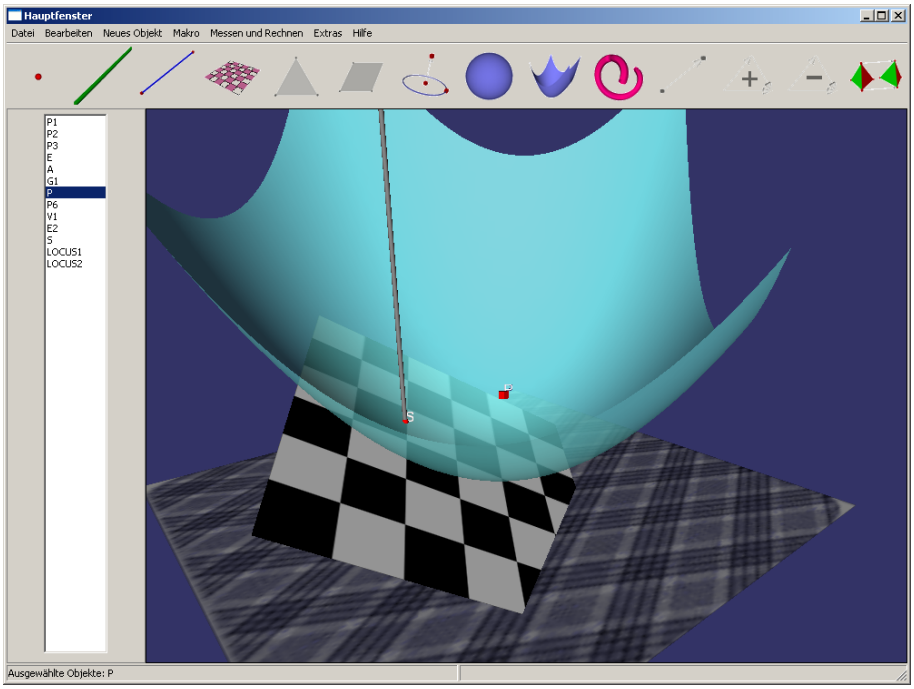

Figure 3: This figure shows a screenshot of Archimedes Geo3D [11]. At the top the menu bar is located with options for file actions, object construction, measurement, calculations and macro generation. Below the toolbar can be seen which shows buttons for actions available for the current selection. On the left there is an object box where objects can be selected and deselected. With a double click on a point its coordinates can be changed.

labs of the France Centre National de la Recherche Scientifique (CNRS) and at the Joseph Fourier University in Grenoble. In 1985 Jean-Marie Laborde, co-founder of Cabrilog, had the vision to make 2D geometry easier to learn for students and for teachers more enjoyable to teach.

A Cabri 3D document consists of a set of pages with one or more views on each page that can be freely manipulated. The user can choose from over fifteen standard projections and can change the viewpoint on a construction unrestrictedly. The pages can be enriched with comments in rich text and the constructions can be augmented with colors, textures and graphic styles (see figure 4). Lenghts, angles, areas and volumes can be measured and further calculations can be performed with these results. Expressions can be created using algebraic concepts like numbers, variables and operations. Animations can be used for modeling physical phenomena. A tool replays the user's previously performed construction steps. A nice feature is the unfolding of all polyhedra into a printable net.

The file format of Cabri 3D is based on the XML standard (CabriML) and

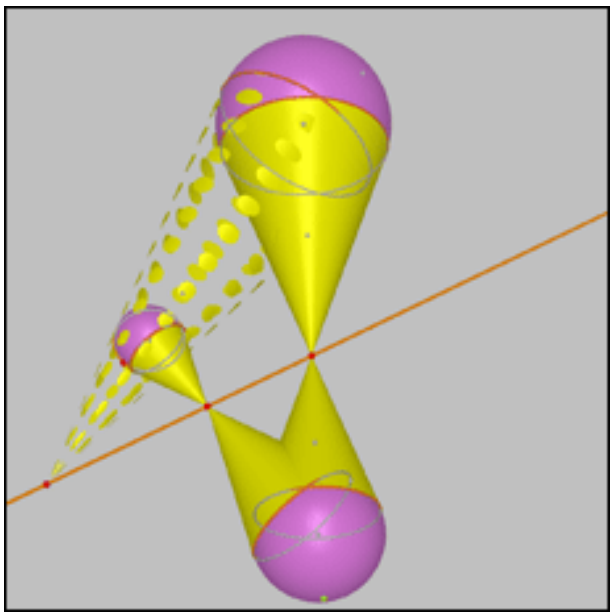

Figure 4: This figure shows a screenshot of a Cabri 3D construction. It demonstrates that the apexes of three double cones (yellow) which are distinct and tangent to two among three spheres (purple) are collinear.

is therefore easy to understand and to modify. High resolution images can be copied to the clipboard. Every project can be exported as interactively manipulable figure to a web page. The required plugin is available for free for Windows. These figures can also be incorporated in Microsoft Office documents. [3, 27]

2.2.3 CAD software providing differential geometry functionality

Rhinoceros and Pro/Engineer are examples of commercial CAD software that also supports differential geometry and are therefore related to this work. Both software packages have in common that they are not interactive in a sense that changes are not applied in real time compared to the software packages presented in the previous two sections.

Rhinoceros Rhinoceros [19], abbreviation Rhino (see figure 5), is a commercial software for NURBS (non-uniform rational B-Splines) modeling for Windows developed by McNeel. Rhinoceros is used by constructors, architects and designers. A particularity of Rhino is that all surfaces are constructed of NURBS. With Rhinoceros NURBS curves, surfaces or volumes can be created, edited, analyzed or converted, regardless of their complexity, grade or size. Polygon meshes and point clouds are also supported.

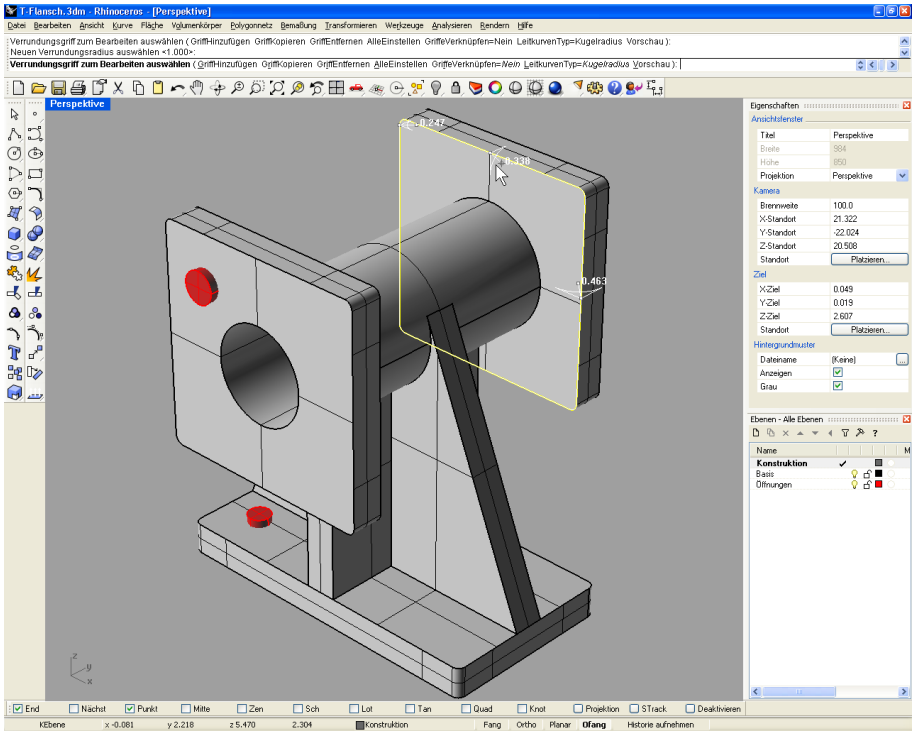

Figure 5: This figure shows a screenshot of the Rhino 3D [9] user interface. The shown object can be analyzed with a wide range of analysis tools for surface curvature and geometric continuity.

Rhino supports 3D freeform modeling without any restrictions and is nevertheless extremly precise. Rhino is also compatible with a wide range of other CAD or modeling software packages and can therefore be seen as a connector between other programs. It supports also a variety of input and output devices like 3D digitizers, 3D scanners and 3D printers.

The link to this thesis is Rhino's support of differential geometry functions. There are analysis tools for surface curvature, geometric continuity, deviation, curvature graphs on curves and surfaces, Gaussian curvature, mean curvature and minimum or maximum radius of curvature.

Helices and spiral curves can be constructed. Rotational surfaces and extrusions are also available. It also supports some forms of curvature measurement.

In its latest version 4.0 many features of Rhino were enhanced, including texturing, rendering and animation. Advanced custom display modes are now supported, as well as dual-screen support and stereo display.

Pro/Engineer Pro/Engineer [21] is a professional parametric 3D CAD software package also known as ProE or Pro/E developed by the Parametric Technology Corporation (PTC). It is available for Windows, Linux, Irix, HP-Unix

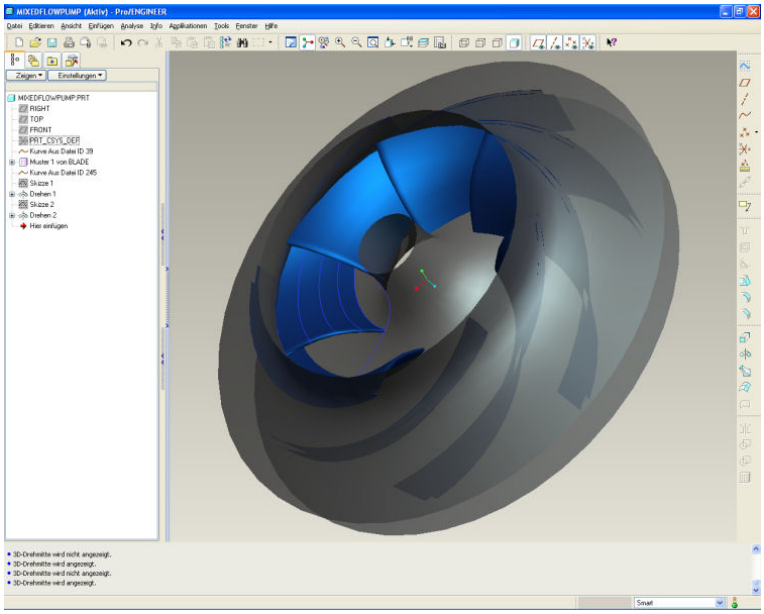

Figure 6: Screenshot Pro/Engineer

and Solaris.

In Pro/Engineer all objects are constructed in 3D and in succession the corresponding drawings can be extracted. The term parametric means that everything has a dimension and a change in a part of model changes the whole model geometry. It is also bidirectional associative, so every change in the model geometry follows a change in all abbreviations of the model like drawings and constructions groups and vice versa.

The main application areas are the automobile industry (construction of motors and gearboxes) and machine construction (see figure 6).

3 Technological Foundations

This sections shall explain the components which build the basement of Construct3D and therefore of the new implemented functions.

3.1 Open Inventor and Coin3D

The Open Inventor rendering library is a framework of C++ classes that implement a scene graph based rendering library using OpenGL. It supports an event driven programming style where the application is normally composed as a set of call back functions that react to framework events. [23]

It is a library of objects and methods for creating interactive 3D graphics applications. So it offers a range of objects which can be used, modified and extended. There are three types of objects: database primitives, interactive manipulators and components.

All information of the created 3D objects e.g.—shape, size, color, texture, location—is stored in a scene database. This approach offers the flexibility not only to render these objects on a screen but also to move the objects, view them from different viewpoints, change their properties, to highlight, animate and to interact with them. The Open Inventor programming model is intuitive to use because it is based on the "real world" we live in. [32, 33]

Open Inventor originally was developed by SGI but after it's release under Open Source license in 2000 SGI discontinued the further development. In 1995 the company Systems In Motion started the development of Coin as a 3D graphics rendering library. After a few years Coin was rewritten from scratch and "inventorized" and now implements the whole Open Inventor API including a number of extensions. For Free Software development Coin3D is available under the GNU GPL license. [30]

3.1.1 The Open Inventor Toolkit

At the programming level Open Inventor offers the following tools:

- *3D scene database*: used to create a hierarchical 3D scene. (shape, property, group, engine and sensor objects)
- *Node kits*: prebuilt groups of Inventor nodes.
- *Manipulators*: objects in the scene database users are able to interact with. (handle box, trackball)
- *Inventor component library*: library for high-level interactive tasks (render area, material editor, viewers, utility functions) [32]

3.1.2 The Scene Database

The scene database consists of basic elements called nodes. A node is a C++ class type with additional functions that aggregates objects called fields that store a value of a certain type (e.g. string or integer). A node contains information like surface materials, shape descriptions or geometric transformations. Every 3D shape or light source itself is also a node. To form a hierarchical structure a node of type SoGroup can associate a list of other nodes, called children. The children of a group node are ordered and can be accessed from left to right with index 0 up to $n-1$. The ordered collection of all these nodes is the scene graph which is stored in the database. A mechanism called action traverses the scene graph recursively to compute data.

Classes of database primitives include the following:

- *Shape nodes*: sphere, cube, cylinder, cone, quad mesh, etc.
- *Property nodes*: material, lighting model, textures, etc.
- *Group nodes*: separator, switch, etc.
- *Engines*: Engines can be connected to other objects to animate parts of the scene or otherwise constrain some parts of the scene.
- *Sensors*: A sensor can detect changes in the underlying database or reacts to a timer and in succession calls a function. [23, 32]

3.1.3 Node Kits

A node kit is a collection of nodes with a specified arrangement which helps to keep things in order. There are some sort of templates telling you which kinds of nodes can be added to a node kit and where to place them. There is also the possibility to extend Open Inventor in creating new customized node kits. [32]

3.2 Studierstube

Studierstube [26] is a research software system for augmented reality applications. It consists of a set of extension nodes to the Open Inventor [32] (see section 3.1) rendering library. It includes support for interaction based on 3D tracking events. There are rendering and output modes for virtual and augmented reality output devices. It also provides tools for developing distributed applications and user management functions for multiple user support.

3.2.1 3D event system

Studierstube extends the Open Inventor library to support not only standard desktop input devices such as keyboard and mouse but also trackers with six

degrees of freedom (position and orientation in the space). These user interface events are handled with a dedicated Handle3DEventAction. The base class Base3D implements the basic methods called during the traversal of the scene graph after a user interface event.

3.2.2 Widget system

Widgets are graphical objects that react to input events and change their state. A state change means a different graphical representation (e.g. a released button is a box with a given height and a pushed button is a box with a lower height and a different color) and a change to the fields of the widget (e.g. a Boolean is set from false to true). Studierstube implements a standard set of widgets such as toggle, push and radio buttons, lists and sliders which are represented by 3D geometry. The personal interaction panel (PIP) is a tracked tablet overlaid with a traditional 2D graphical user interface consisting of widgets. It combines the advantages of a physical representation, the natural way of interaction with the virtual widgets and the haptic feedback when an interaction device, mainly a pen, collides with the tablet with the flexibility of switching between different sets of widget groups.

3.2.3 Dynamic application loading

Applications are implemented as a sub scene graph in a Studierstube process and can be defined by implementing a new application node class. On the one hand the application can use any existing sub scene graph to create required elements, on the other hand own specialized node types can be defined. All the time all Open Inventor operations can be used on the application data.

Because of Open Inventor's support of serialization of a scene graph to and from a file, an application can be loaded and saved at runtime. At any time the application's scene graph represents the current state of the application because all data is stored in fields or nodes of a sub scene graph.

3.2.4 Collaborative work

Studierstube provides the necessary functionality for collaborative work. It supports the simultaneous connection of several display devices which can provide personalized views and also supports an unlimited number of input devices. A typical dual user setup consists of two head mounted displays (HMD), two interactions devices (pen) and two personal interaction panels (PIP), that is six tracked devices and two output windows. Because of the tracking of the input devices the user can see the scene through his HMD from his own viewpoint. It's also possible to render a private sub scene graph for each user which gives the possibility of a personalized view. [23]

3.3 Construct3D

Construct3D [14] is a 3D dynamic geometry construction tool. It is based on the Augmented Reality System Studierstube. Users see the real environment augmented with virtual content. That gives the users, mainly high school and university students, the opportunity to actually construct, explore and interact with three dimensional content in "real" 3D space (see figure 7). Until now students had to construct geometry with traditional pen and paper based methods or with desktop computer software that also only shows a 2D projection of a 3D content. Construct3D offers the opportunity to walk around a geometric construction and view it from different angles. It is also a colloborative Augmented Reality application and can be used by several users simultaneously which can interact with each other.

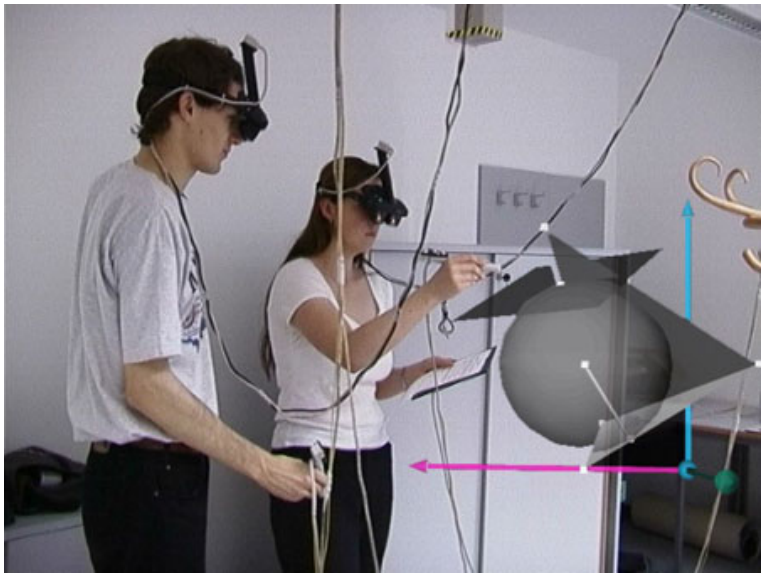

Figure 7: This figure shows two users of Construct3D working together on a three dimensional construction. The users wear head mounted displays (HMD) and interact with the construction with the aid of pen-based input devices. A tracked physical tablet with a projection of a menu system, the Personal interaction panel (PIP), is used to access Construct3D's functions.

A further key feature of Construct3D is the support for dynamic 3D geometry. By moving one part of the construction the other parts of the construction adjust themselves in real-time. The relationships between the components of the construction can be explored and a deeper understanding of the matter may be achieved. Construct3D is also a tool that should improve the spatial abilities of the users [6].

3.3.1 Design

At the moment Construct3D offers functions for the construction of points and two and three dimensional objects and also for planar and spatial operations on these objects. It also provides functions for measuring, structuring content into layers and basic system functions.

At the start up a 3D window with infinite size is initialized to cover all available space. The users work with a user interface consisting of the Personal Interaction Panel (PIP) and a pen-based input device. The PIP is a tracked physical tablet with a projection of a menu system on it. The pen is used for manipulating the PIP but also for direct manipulation of the objects.

In point mode the user points with the pen at the location where he wants a point to be and clicks and at an absolute position in space a point will be created. When point mode is turned off the nearest point or object in the scene will be highlighted. Then the user can select points or objects with the pen and can create new objects with the previously selected objects as input elements. After selecting an object the user can interact with them, e.g. selecting and moving a point on a circle and the circle will automatically adjust its size and position. A preview function available for most constructions shows the resulting object in wireframe mode. The 3D constructions can be projected on orthogonal planes to provide classical 2D views of the 3D content (e.g. ground view, upright projection, profile).

When the application is used with several users, each user's constructed points and objects get an unique color. Nevertheless each user can modify constructions of another user.

Every construction step is stored in a command list and an undo/redo history offers the possibility to move backward and forward in the construction history.

3.3.2 Basic object types

Construct3D offers a range of basic object types:

- Points: freely positioned or fixed on curves or surfaces
- Lines
- Planes
- Circles and ellipses
- Cuboids
- Spheres
- Cylinders
- Cones

- B-Spline curves with control points and interpolated B-Spline curves
- NURBS surfaces with control points and interpolated NURBS surfaces
- Sweep surfaces

3.3.3 Geometric operations

The following geometric operations can be performed on the basic object types and also on more complex objects generated throughout the construction process:

- Boolean operations (union, difference, intersection)
- Intersections
- Planar slicing of objects
- Rotational sweep around an axis
- Surface normals
- Tangential planes
- Tangents
- Common tangent to two circles
- Plane normal to a line
- Line normal to a plane
- Plane of symmetry
- Angle bisector
- Mid point [14]

3.3.4 New functions

As result of this diploma project the following new functions have been implemented which will be explained later in the sections 4.3, 4.4 and 4.5:

Differential geometry

- Center of curvature
- Circle of curvature
- Plane of curvature
- Meusnier point
- Frenet Frame

Sweeps

- Helical sweeps
- General sweeps

3.4 ACIS

Geometric modeling is a very difficult task concerning higher mathematics and complex programming issues and also practical problems like numerical accuracy. Therefore after considering to implement these functions themselves the developers of Construct3D decided to use the 3D geometric modeler ACIS [28] which is widely used from CAD software to virtual reality applications. The ACIS modeler is a software library developed by the Spatial corporation for representing and manipulating shapes. It features an open, object-oriented C++ architecture that enables robust 3D modeling capabilities. ACIS integrates wire frame, surface and solid modeling functionality with both manifold and non-manifold topology and geometric operations. [4]

3.4.1 Boundary representations

ACIS is an exact geometric modeler that represents shapes by modeling their boundaries. This representation is called boundary representation or B-rep for short because ACIS calculates the equations of the curves and surfaces that lie on the boundary between the inside and the outside of a solid 3D object.

Imagine a thin wooden plate with a circular hole in it. This board consists of the following surfaces: two planar surfaces on the upper and the lower side of the board, four planar surfaces on the exterior and a cylindrical surface for the hole. The intersections of these surfaces form curves and the meeting points of the curves form points. So a boundary representation consists of vertices (points), edges (curves), faces (surfaces). The only problem with this approach is that, with a few exceptions e.g. points, circles and spheres, classical analytical geometry has no boundaries. The solution for this problem is to explicitly define these boundaries.

Curves are bounded by pairs of points lying on the curve and surfaces are bounded by curves lying on the surface. The relationships of the geometric elements are called topology and can be visualized as a graph where the nodes represent points, curves and surfaces. Also the sharing of bounding entities (e.g. the corners of a cube bound three different edges and each edge bounds two faces) can be stored in such a graph. These entities are called topological entities because they define how things connect:

- The shape of a face is a surface bounded by edges associated with it.

- The shape of an edge is a curve bounded by a pair of vertices associated with it.
- The location of a vertex is the position of a point.

Only the three topological entities face, edge and vertex are really needed but in practice the introduction of a few more topological entities is useful. Below there is an explanation how a path through the graph shown in figure 8 can be read:

- *Coedge*: In most cases edges lie between two faces but there are operations which need only the boundary of an individual face. So an edge is associated with a number of coedges, one coedge for each face.
- *Loop*: The connection of coedges which form a high-level representation of a faces boundary is called loop.
- *Face*: A collection of loops can be summarized to a face.
- *Shell*: The connection of faces which form a high-level representation of a surface is called shell.
- *Lump*: A collection of shells which define a separate piece of an object is called lump.
- *Body*: A collection of lumps can be summarized to a body.

Finally there are a few restrictions to the ACIS boundary representations to guarantee only manifold objects:

- Every edge must lie between two faces.
- Faces and edges do not self intersect.
- Every entity in the model is bounded. [4]

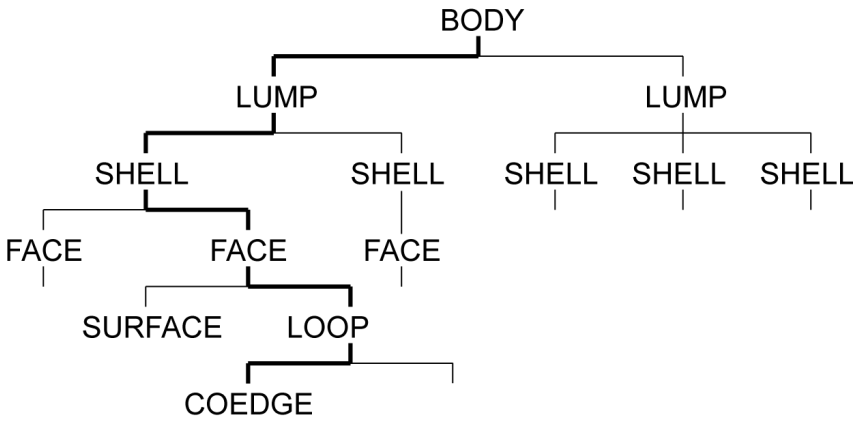

Figure 8: Boundary representation hierarchy in ACIS. A possible path through this graph (bold line) is described in section 3.4.1.

3.4.2 Geometric representations

There are several ways to represent geometry, e.g.:

1. Explicit: $y = f(x)$
2. Implicit: $f(x, y) = 0$
3. Parametric: $x = f(t)$, $y = f(t)$

The explicit form is the best known and least useful representation because of its numerous possible special cases. For geometric modeling the implicit form which defines two half-spaces is more useful. Half-spaces can be imagined to be generated when an infinite plane cuts the 3D space in two. Given a point's coordinates on the boundary, the implicit equation evaluates to zero. On either side the equation evaluates to more or less than zero. The drawback of the implicit representation is that it cannot be easily derived for all surfaces used by ACIS.

The third form, the parametric representation, can be created for all surfaces used by ACIS. The parametric form of a curve is defined by equations given in terms of a single independent variable t. A mathematical function with two parameters u and v can be used to define every point on a surface. That means every surface has its own (u, v) coordinate system.

Because of its specific advantages and drawbacks ACIS uses analytic, implicit and parametric geometry representations. The analytical curves and surfaces have a low level implicit representation as well as higher level parametric representation. General smooth curves and surfaces, called splines, are represented through parametric representation. There are several forms of splines but all are constructed by specifying a number of control points which determine their shape. [4]

4 Geometric Background

Elementary differential geometry deals with curves and surfaces in three dimensional space and their curvature properties. Differential geometry can be seen as a synthesis between analysis, in particular infinitesimal calculus, and elementary geometry. When handling differential geometry problems numerically, the need arises to calculate derivatives of complicated functions. Therefore knowledge on numerical derivation methods is useful. For a comprehension of the concepts of differential geometry a basic understanding of these topics is required.

In the following section the mathematical and geometrical foundations are explained which are needed for understanding the differential geometry tools implemented for this thesis. At the beginning a short overview over different numerical derivation methods is given which are fundamental because all differential geometry tools need at least the second derivation of a curve's function. After this the parametric form of a curve representation both in two and in three dimensions will be introduced. Next, the differential geometry concepts on a curve are explained: the Frenet frame and the center, the circle, the sphere and the plane of curvature. Afterwards the concepts for surfaces are introduced, starting with an explanation of surface curvature and their types, followed by the Frenet frame in surface points and the Meusnier point. Finally the geometric properties of the implemented surface classes—helical sweeps and general sweeps—are introduced.

Detailed descriptions of the mathematical and geometrical foundations of this thesis can be found in [15, 29, 34].

4.1 Numerical Derivation Methods

In this section some concepts of calculating derivations of a function in a numerical way will be presented. Unfortunately it is unknown how ACIS performs the derivation of functions and therefore it is also unknown how Construct3D performs these calculations. The numerical derivation methods presented in the following shall only give a short overview to get an idea how ACIS could perform these tasks.

4.1.1 Differentiation

The derivation of a function $f(x)$ at a point x is the linear image of the function which approximates the change of the function best. The geometrical correspondent of the first derivative of a function at a given point is the gradient of its tangent in that point. To estimate the gradient of the tangent we can

assume the gradient of a secant, a straight line through two points of the function curve

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \tag{1}$$

The gradient of the secant is called difference quotient. Then we let the distance between the points $f(x)$ and $f(x+h)$ converge to zero. A function is called differentiable at position x if the limit

$$\lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0} = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h} \tag{2}$$

exists. This limit is called differential quotient or derivative of f with respect to x and can also be written as $f'(x_0)$ (Lagrange's notation) or $\frac{df}{dx}(x_0)$ (Leibniz's notation).

The derivatives of all elementary functions can be calculated considering the concepts of the difference and the differential quotient but in daily practice the derivatives and the antiderivatives are well known or can be looked up in a reference book. [1, 22]

4.1.2 Higher derivatives

If the derivative of a function f is derivable, a second derivative can be obtained as derivative of the first derivative. A function f can be derivable up to n times, i.e. up to its n-th derivative. [1]

$$f'' = f^{(2)} = \frac{d^2 f}{dx^2}, f''' = f^{(3)} = \frac{d^3 f}{dx^3}, \ldots, f^{(n)} = \frac{d^n f}{dx^n} \tag{3}$$

4.1.3 Taylor expansion

The so-called Taylor expansion is used to approximate complicated functions by a series of polynoms. Often a function can be fairly good approximated by a Taylor expansion truncated after only a few terms. Given a $(n+1)$ times continuous derivable function f in its interval I then we can propose for all a and x within I the Taylor formula

$$f(x) = T_n(x) + R_{n+1}(x) \tag{4}$$

where

$$\begin{aligned} T_n(x) =& \sum_{k=0}^{n} \left(\frac{f^{(k)}(a)}{k!}(x-a)^k \right) \\ =& f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n \end{aligned} \tag{5}$$

with the remainder term

$$R_{n+1}(x) = \frac{1}{n!} \int_a^x (x-t)^n f^{(n+1)}(t) dt \tag{6}$$

A function f that is indefinitely derivable is called a smooth function and the Taylor formula can be expanded to the Taylor series. The Taylor series is a power series of a real or complex function f that is infinitely derivable in a neighborhood of a real or complex number a [1]

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{7}$$

4.1.4 Numerical approximation by difference quotients

A numerical solution is necessary if either a precise solution is not available or an exact solution can not be computed because of its huge costs. There are some methods of calculating a derivative numerically and for each problem one has to balance the pros and cons concerning cost and accuracy of the solution. When computing the derivative of a function there are several potential sources of error, e.g. the round off error and the truncation error.

Error estimate for a asymmetrical difference quotient Assuming one chooses an arbitrary small number for h, e.g. $h = 0,0001$ and use the equations of section 4.1.1 h has no exact binary representation, neither has $x+h$ in most cases. Therefore these values are represented with some fractional error, called round off error, depending on the floating point precision of the machine where the calculation is executed. As a consequence there is also a fractional error in the derivative and in the following calculations. To avoid this error one has to choose a value h where it is guaranteed that h can be exactly represented. If h is exactly represented the round off error of the difference quotient is

$$e_r \sim e_f \left| \frac{f(x)}{h} \right| \tag{8}$$

where e_f is the fractional accuracy with which f is computed. For a simple function the fractional accuracy is comparable to the machine accuracy $e_f \approx e_m$.

The truncation error comes from higher terms in the Taylor expansion. For the function $f(x+h)$ one can assume the following Taylor expansion

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(x) + \dots \tag{9}$$

Therefore one can assume a Taylor expansion for the difference quotient:

$$\frac{f(x+h) - f(x)}{h} = f'(x) + \frac{1}{2}hf''(x) + \dots \tag{10}$$

The truncation error of this Taylor expansion is in the order of

$$e_t \sim |hf''(x)| \tag{11}$$

If h is varied to minimize the sum of errors $e_r + e_t$ one can assume the optimal value of h to be

$$h \sim \sqrt{\frac{e_f f}{f''}} \approx \sqrt{e_f} x_c \tag{12}$$

where

$$x_c \equiv \sqrt{\frac{f}{f''}} \tag{13}$$

is the curvature scale or characteristic scale of the change of function f. If no further information is available a first assumption is $x_c = x$. If x is nearly zero a different assumption should be used. With the proposition $h \approx \sqrt{e_f} x_c$ the fractional accuracy of the computed derivative can be estimated

$$\frac{(e_r + e_t)}{|f'|} \sim \sqrt{e_f}\sqrt{\frac{ff''}{f'^2}} \sim \sqrt{e_f} \tag{14}$$

From this it follows that the difference quotient from section 4.1.1 gives at best the square root of the machine accuracy dependent on the machine's floating point precision. [22]

Error estimate for a symmetrical difference quotient A more accurate result can be achieved if it can be afforded to calculate two functions instead of one. Then the symmetrical form of the difference quotient can be used

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{15}$$

The truncation error for this form of difference quotient is $e_t \sim h^2 f'''$ and the round off error stays the same as before. In this case the optimal choice of h is

$$h \sim \sqrt[3]{\frac{e_f f}{f'''}} \sim \sqrt[3]{e_f} x_c \tag{16}$$

Therefore the fractional error e_f is

$$\frac{e_r + e_t}{|f'|} \sim \frac{\sqrt[3]{e_f{}^2}\sqrt[3]{f^2}\sqrt[3]{f'''}}{f'} \sim \sqrt[3]{e_f^2} \tag{17}$$

That means that the accuracy of the symmetrical difference quotient is between one or two orders of magnitude better than of the asymmetrical difference quotient. Therefore h should be also chosen the correct power of e_f or e_m times a characteristic scale x_c. [22]

4.1.5 Advanced methods of numerical approximation

Unfortunately, the simple approaches of different types of the difference quotient do not yield the desired results in respect of accuracy. To achieve better results, the exploration of the function's behavior and assumptions of smoothness or analyticity are required to get higher-order terms in a Taylor expansion that have some meaning. The drawback of these methods is the need of multiple evaluations of the function f.

Richardson's deferred approach to the limit According to the general approach of Richardson's deferred approach to the limit [24] one seeks to extrapolate $h \to 0$. The result are finite difference calculations with smaller and smaller finite values of h. Using Neville's algorithm, an algorithm for polynomial interpolation, each new finite difference calculation is on the one hand used for the extrapolation of higher order and on the other hand for the extrapolation of previous lower orders but with a smaller h-value. Implementations of this algorithm lead to an approximate derivative and an estimation of its error. [22]

Chebyshev polynomial approximation If a fairly smooth function is given and it will be evaluated at several arbitrarily chosen points then a Chebyshev polynom could be used to approximate the function within a given interval. Instead of derivating the function itself, derivates of the Chebyshev polynom could be used. A Chebyshev polynom $T_n(x)$ of degree n has the explicit formula

$$T_n(x) = \cos(n \arccos x) \tag{18}$$

Details can be found in [22].

4.2 Curve representation

In the following section the parametric form of a curve representation both in two and in three dimensions will be introduced. Also the concepts of the parameter transformation and the so-called arc length will be explained. The knowledge of these things provides the background for the differential geometry calculations on curves explained in section 4.3.

4.2.1 2D curve representation

Plane curves are mostly described analytically in a parametric form:

$$x = x(t), y = y(t) \tag{19}$$

where x and y are the Cartesian coordinates of the point $P(x, y)$. $x(t)$ and $y(t)$ are real unique functions of a real parameter t which are continuous in a shared interval and not constant. The point $P(x, y)$ describes an arc $\widehat{AB}$ of a real continuous plane curve c. To exclude some special cases it is further requested that the mapping between t in its interval and of the points P of the continuous arc $\widehat{AB}$ has to be continuous and reversible unique. It is also required that the functions $x = x(t)$ and $y = y(t)$ have continuous derivatives relative to t in its closed interval. These two derivatives also must not be zero at the same time. For further exploration of the curvature the second and third derivatives of the curve, in some cases up to the n-th derivative, are required. The coordinates of the curve points P can also be seen as the coordinates of the position vector x of point P:

$$\mathfrak{x} = \mathfrak{x}(t) = x(t)e_1 + y(t)e_2 \tag{20}$$

where e_1 and e_2 are the unit vectors of the coordinate axes' directions. The derived position vector $\mathfrak{x}'$ is also continuous within its interval and never equal to zero:

$$\mathfrak{x}' = \mathfrak{x}'(t) = x'(t)e_1 + y'(t)e_2 \neq 0 \tag{21}$$

Geometrically interpreted $\mathfrak{x}'$ is the tangent vector of curve $\mathfrak{x}$ in point t. The length of the tangent vector depends on the parametric scale of t and its direction is the same as the curves'. [29]

Parameter transformation Often the parameter t is replaced by a new feasible parameter s

$$t = t(s) \tag{22}$$

where $t(s)$ is a monotone and continuous derivable function relative to s and

$$\frac{dt(s)}{ds} \neq 0 \tag{23}$$

If $\frac{dt}{ds} > 0$ the orientation of curve c stays the same otherwise the orientation is swapped. After the parameter transformation the curve c can be written as [29]

$$\mathfrak{x} = \mathfrak{x}(s) \tag{24}$$

Arc length The vector differential

$$d\mathfrak{x}(t) = \frac{d\mathfrak{x}(t)}{dt}dt = \mathfrak{x}'(t)dt \tag{25}$$

of the smooth curve

$$\mathfrak{x} = \mathfrak{x}(t) = x(t)e_1 + y(t)e_2 \tag{26}$$

is parameter invariant relative to t. Its inner square

$$(d\mathfrak{x}(t))^2 = \mathfrak{x}'^2(t)dt^2 \tag{27}$$

is also invariant relative to movement.

After the parameter transformation $t = t(s)$ and $\frac{dt}{ds} \neq 0$ the position vector $\mathfrak{x}(t)$ becomes $\mathfrak{x}(t(s)) = \mathfrak{x}(s)$ and the inner square is

$$\mathfrak{x}'^2(t)dt^2 = \mathfrak{x}'^2(s)ds^2 \tag{28}$$

If for all s of the arc it is imperative that

$$\mathfrak{x}'^2(s) = x'^2(s) + y'^2(s) \equiv 1 \tag{29}$$

i.e. the tangent vector has the fixed length of 1 then parameter s is special. s is called the arc length of curve and ds is called the arc element or the arc differential.

The square of the arc element is

$$ds^2 = (d\mathfrak{x})^2 = dx^2 + dy^2 \tag{30}$$

The arc length s of the curve between the points t_0 and t can be obtained by the integration of the arc element ds

$$s = \int_{t_0}^{t} \sqrt{\mathfrak{x}'^2(t)}dt = \int_{t_0}^{t} \sqrt{x'^2(t) + y'^2(t)}dt = s(t) \tag{31}$$

The arc element is invariant relative to parameters and to movement. It is also the differential of lowest order with these features, i.e. the simplest differential invariant of the curve. Through integration the arc length s can be obtained which is also the simplest integral invariant of the curve. ds is called the natural differential and s the natural parameter of the curve. All further obtained values and vectors based on the natural parameter are also parameter invariant.

Geometric meaning To obtain the length of an arc $c = \widehat{AB}$ of the curve $\mathfrak{x} = \mathfrak{x}(t)$ we inscribe the arc c with arbitrary intermediate points $P_v = (x(t_v), y(t_v))$ with parameter values $t_a = t_0 < t_1 < t_2 < \cdots < t_{n-1} < t_n = t_b = t$ into a polyline $\mathfrak{P} = (A, P_1, P_2, \ldots, P_{n-1}, B)$ and determine its length:

$$s(\mathfrak{P}) = \sum_{v=1}^{n} |\mathfrak{x}(t_v) - \mathfrak{x}(t_{v-1})| \tag{32}$$

The more intermediate points P_v are used the better is the estimation of the real length of the arc according to the triangle inequality. If the set of polylines $\{s(\mathfrak{P})\}$ which can be inscribed into the arc is bounded above the arc c can be assigned with a finite arc length $s = \sup s(\mathfrak{P})$ and is called rectified.

Theorem: Each continuous differentiable curve $\mathfrak{x} = \mathfrak{x}(t)$ can be called rectified. Its arc length $s = s(t)$ is a continuous derivable function of t described with the integral [29]

$$s = s(t) = \int_{t_0}^{t} \sqrt{\mathfrak{x}'^2(t)}dt = \int_{t_0}^{t} |\mathfrak{x}'(t)|dt \tag{33}$$

4.2.2 3D curve representation

The basic principles for curves in three dimensions are mainly the same as for curves in 2D, see section 4.2.1. The parametric representation of a three dimensional curve consists of three in its interval $t_a < t < t_b$ unique continuous derivable functions $x(t)$, $y(t)$ and $z(t)$ relative to a feasible parameter t which derivations are not equal to zero at the same time. These functions determine as coordinates of the position vector

$$\mathfrak{x} = \mathfrak{x}(t) = x(t)e_1 + y(t)e_2 + z(t)e_3 \tag{34}$$

a continuous derivable 3D curve.

The tangent vector at a position t is given through the derivation of the position vector

$$\frac{dx}{dt} = \mathfrak{x}' = \mathfrak{x}'(t) = x'(t)e_1 + y'(t)e_2 + z'(t)e_3 \tag{35}$$

Because of the continuity of the tangent vector $\mathfrak{x}'(t)$ the curve can be called rectified and its arc length s is given through the integral

$$s = \int_{t_0}^{t} \sqrt{\mathfrak{x}'^2(t)}dt = \int_{t_0}^{t} \sqrt{x'^2(t) + y'^2(t) + z'^2(t)}dt = s(t) \tag{36}$$

The arc element of a 3D curve is also independent of the parameter representation $\mathfrak{x}(t)$ and the spatial position of the curve. The arc length s is called the natural parameter of the 3D curve and is bounded above by the length of an inscribed polyline. Choosing the arc length s as parameter of the curve

$$\mathfrak{x} = \mathfrak{x}(s) = x(s)e_1 + y(s)e_2 + z(s)e_3 \tag{37}$$

then the tangent vector

$$\mathfrak{t} = \mathfrak{x}' = \mathfrak{x}'(s) = x'(s)e_1 + y'(s)e_2 + z'(s)e_3 \tag{38}$$

has a fixed length of one [29]

$$|\mathfrak{t}| = |\mathfrak{t}'(s)| = 1 \tag{39}$$

4.3 Differential geometry on a curve

On the following pages the mathematical and the geometrical basic principles necessary for understanding the concepts of differential geometry on a curve presented in this thesis will be explained. At the beginning the concept of the Frenet frame will be presented and the terms curvature and torsion will be defined. In the following subsection the concepts of the center, the circle and the sphere of curvature will be introduced. Finally the plane of curvature will be explained.

4.3.1 Frenet frame in curve points

The Frenet frame (see figure 9) is a local coordinate system in each point of a curve. Using the natural parameter s one can exploit the advantage that the derivatives $\mathfrak{x}'(s)$, $\mathfrak{x}''(s)$ ...of a curve $\mathfrak{x}(s)$ are invariant of movement and parameter.

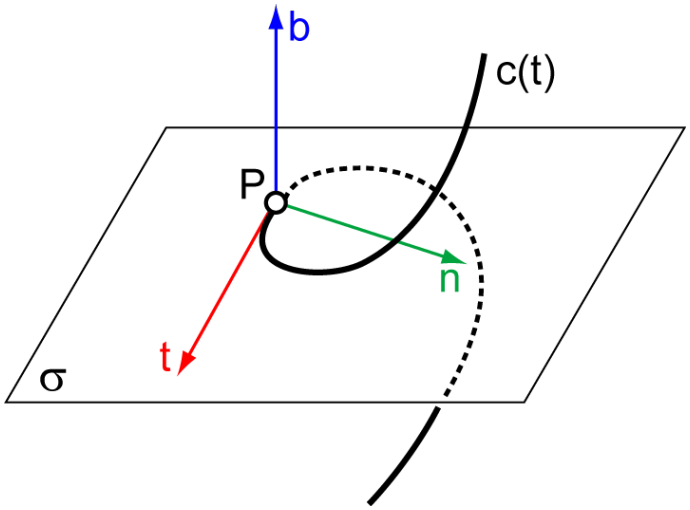

Figure 9: This figure shows the Frenet frame in a curve point P. The Frenet frame consists of three vectors: the tangent vector t, the normal vector n and the binormal vector b.

For $\mathfrak{x}''(s) \neq 0$ the Frenet frame can be defined as:

$$\begin{aligned} \mathfrak{t} &= \mathfrak{x}'(s) \ldots \text{tangent vector} \\ \mathfrak{n} &= \frac{\mathfrak{x}''(s)}{|\mathfrak{x}''(s)|} \ldots \text{normal vector} \\ \mathfrak{b} &= \mathfrak{t} \times \mathfrak{n} \ldots \text{binormal vector} \end{aligned} \tag{40}$$

For each natural parameter s $\mathfrak{x}'(s)$ is a unit vector. From this it follows that

$$\mathfrak{x}'^2 = \mathfrak{t}^2 \equiv 1 \tag{41}$$

and therefore

$$\mathfrak{x}'\mathfrak{x}'' = \mathfrak{t}\mathfrak{t}' \equiv 0 \tag{42}$$

i.e.

$$\mathfrak{t} \perp \mathfrak{n} \tag{43}$$

From this it follows that both unit vectors, the tangent vector $\mathfrak{t}$ and the normal vector $\mathfrak{n}$, are pairwise orthogonal and form a right-handed coordinate system.

The naming of the three vectors forming the Frenet frame can be explained as follows: $\mathfrak{t}$ is an orientation vector of the tangent at position s of curve $\mathfrak{x}(s)$. A straight line orthogonal to a tangent is called a normal and both the normal vector and the binormal vector fulfill this condition.

If the curvature

$$k(s) = |\mathfrak{x}''(s)| \tag{44}$$

in $\mathfrak{x}(s)$ exists, there is also a frenet frame. $\mathfrak{x}''(s)$ can be seen as a measure for the variation from a straight-lined run of the curve. If $|\mathfrak{x}''(s)| = 0$ in an interval the run of the curve is straight-lined in this range. The curvature of a curve in 2D can also be regarded as a measure how fast the tangent is rotating around its curve point.

If we want to know the variation of the vectors of the Frenet frame when moving along the curve we assume the following system of equations:

$$\begin{aligned} \mathfrak{t}' &= a_{11}\mathfrak{t} + a_{12}\mathfrak{n} + a_{13}\mathfrak{b} \\ \mathfrak{n}' &= a_{21}\mathfrak{t} + a_{22}\mathfrak{n} + a_{23}\mathfrak{b} \\ \mathfrak{b}' &= a_{31}\mathfrak{t} + a_{32}\mathfrak{n} + a_{33}\mathfrak{b} \end{aligned} \tag{45}$$

Given that $\mathfrak{t}'(s) = \mathfrak{x}''(s) = k\mathfrak{n}(s)$ we can state that $a_{11} = a_{13} = 0$ and $a_{12} = k$. Because $\mathfrak{t}$, $\mathfrak{n}$ and $\mathfrak{b}$ form a trihedron, i.e.

$$\mathfrak{t}^2 = \mathfrak{n}^2 = \mathfrak{b}^2 = 1 \tag{46}$$

and

$$\mathfrak{t}\mathfrak{n} = \mathfrak{n}\mathfrak{b} = \mathfrak{b}\mathfrak{t} = 0 \tag{47}$$

we can solve the former equations for a_{ik}:

$$\begin{array}{lll} a_{11} = \mathfrak{t}'\mathfrak{t} = 0 & a_{12} = \mathfrak{t}'\mathfrak{n} = k & a_{13} = \mathfrak{t}'\mathfrak{b} = 0 \\ a_{21} = \mathfrak{n}'\mathfrak{t} & a_{22} = \mathfrak{n}'\mathfrak{n} & a_{21} = \mathfrak{n}'\mathfrak{b} \\ a_{31} = \mathfrak{b}'\mathfrak{t} & a_{32} = \mathfrak{b}'\mathfrak{n} & a_{33} = \mathfrak{b}'\mathfrak{b} \end{array} \tag{48}$$

Concerning

$$\mathfrak{t}\mathfrak{t} = \mathfrak{n}\mathfrak{n} = \mathfrak{b}\mathfrak{b} = 1 \tag{49}$$

it is imperative that

$$\begin{aligned} \mathfrak{t}'\mathfrak{t} + \mathfrak{t}\mathfrak{t}' &= 0 \\ \mathfrak{n}'\mathfrak{n} + \mathfrak{n}\mathfrak{n}' &= 0 \\ \mathfrak{b}'\mathfrak{b} + \mathfrak{b}\mathfrak{b}' &= 0 \end{aligned} \tag{50}$$

i.e.

$$\mathfrak{t}\mathfrak{t}' = \mathfrak{n}\mathfrak{n}' = \mathfrak{b}\mathfrak{b}' = 0 \tag{51}$$

From

$$\mathfrak{t}\mathfrak{n} = \mathfrak{n}\mathfrak{b} = \mathfrak{b}\mathfrak{t} = 0 \tag{52}$$

it follows that

$$\begin{aligned} \mathfrak{t}'\mathfrak{n} + \mathfrak{t}\mathfrak{n}' &= 0 \\ \mathfrak{n}'\mathfrak{b} + \mathfrak{n}\mathfrak{b}' &= 0 \\ \mathfrak{b}'\mathfrak{t} + \mathfrak{b}\mathfrak{t}' &= 0 \end{aligned} \tag{53}$$

i.e.

$$\begin{aligned} a_{12} &= -a_{21} \\ a_{23} &= -a_{32} \\ a_{13} &= -a_{31} \end{aligned} \tag{54}$$

We also state that

$$w(s) = \mathfrak{n}'\mathfrak{b} = \mathfrak{n}\mathfrak{b}' \tag{55}$$

and call it the torsion at position $\mathfrak{x}(s)$. The torsion of a curve point can be regarded as a measure how fast the corresponding plane of curvature is rotating around the tangent. A curve without torsion lies within its plane of curvature.

From these assumptions follow these differential equations, the Frenet formulas:

$$\begin{aligned} \mathfrak{t}' &= k\mathfrak{n} \\ n' &= -k\mathfrak{t} + w\mathfrak{b} \\ \mathfrak{b}' &= -w\mathfrak{n} \end{aligned} \tag{56}$$

The torsion $w(s)$ is a measure for the variation of a planar run of the curve. If $w(s) = 0$ in an interval the run of the curve is planar in this range. Because of $\mathfrak{b}'(s) = -w(s)\mathfrak{n}(s)$ the variation of the binormal can be seen as a rotation around the tangent which at the same time is a rotation of the plane spanned out of $\mathfrak{t}(s)$ and $\mathfrak{n}(s)$. So the torsion $w(s)$ brands the "winding" of the curve out of this plane.

Example: straight line:

$$\mathfrak{x}(s) = x + s\mathfrak{a}$$

where $|\mathfrak{a}| = 1$ and $s \in \mathbb{R}$.

1^{st} and 2^{nd} derivation:

$$\begin{aligned} \mathfrak{x}'(s) &= \mathfrak{a} \\ \mathfrak{x}''(s) &= 0 \end{aligned}$$

So both the curvature and the torsion of a straight line which run is straight-lined and planar, amount to 0.

$$\begin{aligned} k(s) &= 0 \\ w(s) &= 0 \end{aligned}$$

Example: circle:

$$\mathfrak{x}(s) = O + r\cos\frac{s}{r}\mathfrak{i} + r\sin\frac{s}{r}\mathfrak{j}$$

where $r > 0$ and $s \in \mathbb{R}$.

1^{st} and 2^{nd} derivation:

$$\begin{aligned} \mathfrak{x}'(s) &= -\sin\frac{s}{r}\mathfrak{i} + \cos\frac{s}{r}\mathfrak{j} \\ \mathfrak{x}''(s) &= -\frac{1}{r}\cos\frac{s}{r}\mathfrak{i} - \frac{1}{r}\sin\frac{s}{r}\mathfrak{j} \end{aligned}$$

From this it follows the curvature of a circle

$$k(s) = |\mathfrak{x}''(s)| = \frac{1}{r}$$

is constant and inversely proportional to its radius.

The torsion of a circle which is a planar geometric primitive amounts to [15]

$$w(s) = 0$$

Example: helix: As for a straight line and a circle a helix has a constant (independent of parameter s) curvature and torsion:

$$k(s) = \frac{r}{r^2 + h^2}$$

$$w(s) = \frac{h}{r^2 + h^2}$$

If $w > 0$ (or $h > 0$) it is a left-handed helix and if $w < 0$(or $h < 0$) it is a right-handed helix. [15]

4.3.2 Center, circle and sphere of curvature

Given the parametric representation $\mathfrak{x} = \mathfrak{x}(s)$ of a curve.

In the case of $k(s_0) \neq 0$ we look for a circle that approximates the curve in point $\mathfrak{x}(s_0)$ as good as possible. If we find the midpoint M we can calculate the radius r as the distance between the midpoint M and the point $\mathfrak{x}(s_0)$ on the curve:

$$r = |\mathfrak{v}(M\mathfrak{x}(s_0))| \tag{57}$$

The circle is coplanar with the plane defined by the midpoint M and the tangent on the curve in point $\mathfrak{x}(s_0)$.

Given the midpoint M and the radius r we can state the function

$$f(s) = (\mathfrak{v}(M\mathfrak{x}(s)))^2 - r^2 \tag{58}$$

This function can be seen as a measure for the deviation of the distance between the midpoint M and the point $\mathfrak{x}(s)$ from the radius r. That is the deviation of the curve from the sphere $S(M, r)$ with the midpoint M and the radius r. According to the Taylor expansion the approximation is good if

$$f(s_0) = 0$$

$$f^{(v)}(s_0) = 0, \text{ for as much } v \text{ as possible}$$

With the Frenet formulas in mind and assuming that $\mathfrak{m}$ is the position vector of M and $\mathfrak{r}(s)$ is the position vector of $\mathfrak{r}(s)$ we can state:

$$f(s) = (\mathfrak{r}(s) - \mathfrak{m})^2 - r^2 \tag{59}$$

$$f'(s) = 2(\mathfrak{r}(s) - \mathfrak{m}) \cdot \mathfrak{r}'(s) = 2(\mathfrak{r}(s) - \mathfrak{m}) \cdot \mathfrak{t} \tag{60}$$

$$f''(s) = 2\mathfrak{r}'(s) \cdot \mathfrak{t} + 2(\mathfrak{r}(s) - \mathfrak{m})\mathfrak{t}' = 2 + 2k(\mathfrak{r}(s) - \mathfrak{m})\mathfrak{n} \tag{61}$$

$$f'''(s) = 2[k'(\mathfrak{r}(s)-\mathfrak{m})\mathfrak{n}+k\mathfrak{t}\mathfrak{n}+k(\mathfrak{r}(s)-\mathfrak{m})(-k\mathfrak{t}+w\mathfrak{b})] = 2(\mathfrak{r}(s)-\mathfrak{m})[-k^2\mathfrak{t}+k'\mathfrak{n}+kw\mathfrak{b}] \tag{62}$$

So

$$f(s_0) = (\mathfrak{r}(s_0) - \mathfrak{m})^2 = 0 \quad \text{, i.e. } \mathfrak{x}(s_0) \in K(M, r) \tag{63}$$

$$f'(s_0) = 2(\mathfrak{r}(s_0) - \mathfrak{m}) \cdot \mathfrak{t} = 0 \quad \text{, i.e. } \mathfrak{r}(s_0) - \mathfrak{m} \perp \mathfrak{t} \tag{64}$$

that is $\mathfrak{x}(s_0)$ is a point on the circle and the radius is perpendicular to the tangent in $\mathfrak{x}(s_0)$.

If we assume that $\mathfrak{r}(s_0) - \mathfrak{m} = x\mathfrak{n}(s_0) + y\mathfrak{b}(s_0)$ and insert into equation 61 we get

$$f''(s_0) = 2(1 + k(x\mathfrak{n} + y\mathfrak{b}\mathfrak{n})) = 2(1 + kx) = 0 \tag{65}$$

i.e.

$$x = -\frac{1}{k(x_0)} \tag{66}$$

Equation 62 leads us to

$$f'''(s_0) = 2\left(-\frac{1}{k}\mathfrak{n} + y\mathfrak{b}\right)[-k^2\mathfrak{t} + k'\mathfrak{n} + kw\mathfrak{b}] = 2\left(-\frac{k'}{k} + kwy\right) = 0 \tag{67}$$

i.e.

$$y = \begin{cases} \dfrac{k'}{k^2 w} & \text{for } w(s_0) \neq 0 \\ \text{arbitary} & \text{for } w(s_0) = k'(s_0) = 0 \end{cases} \tag{68}$$

The midpoints of all spheres with $f(s_0) = f'(s_0) = f''(s_0) = 0$ fulfill the equation

$$M = \mathfrak{x}(s_0) + \frac{1}{k}\mathfrak{n} - y\mathfrak{b} \tag{69}$$

and lie all on a straight line a, called axis of curvature which is parallel to the binormal of the curve in $\mathfrak{x}(s_0)$. a goes through $M(s_0) = \mathfrak{x}(s_0) + \frac{1}{k}\mathfrak{n}$ with the orientation vector $\mathfrak{b}$. On all this spheres lies a circle with midpoint $M(s_0)$, the so-called center of curvature (see figure 10), and radius

$$\rho(s_0) = \frac{1}{k(s_0)} \tag{70}$$

the so-called radius of curvature.

The midpoint of the circle of curvature (see figure 10):

$$M(s) = \mathfrak{x}(s) + \rho(s)\mathfrak{n}(s) = \mathfrak{x}(s) + \rho^2(s)\mathfrak{x}''(s) \tag{71}$$

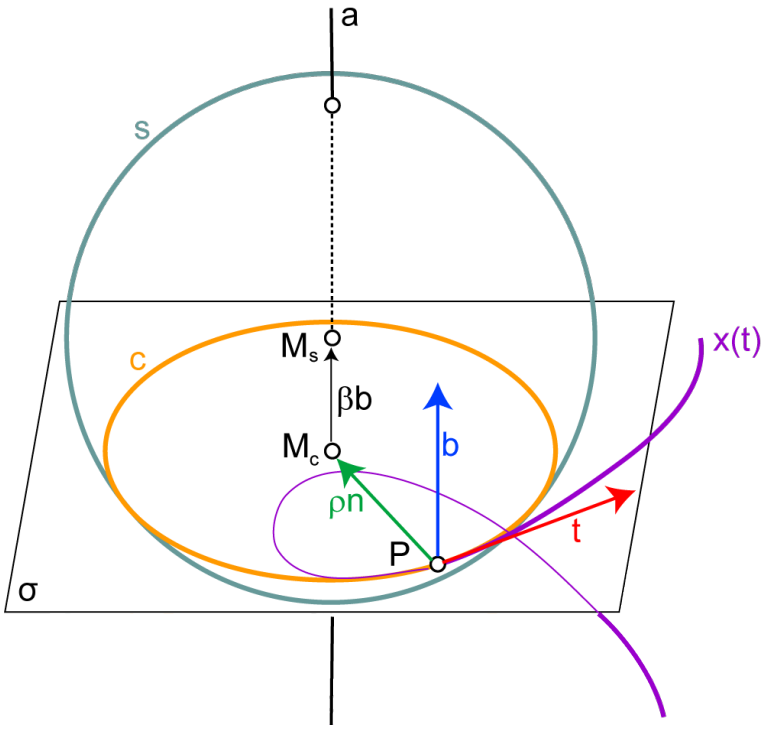

Figure 10: This figure shows an illustration of the center M_c, the circle c, the sphere s and the plane of curvature σ in a curve point P of curve $x(t)$. The Meusnier point M_s (see section 4.4.4), the center of the sphere of curvature s, is also shown. The Frenet frame consisting of the tangent t, the normal n and the binormal b is shown in curve point P. The amount to move in the direction of the normal for achieving the center of curvature M_c is labeled as ρn and the amount in the binormal's direction to get the Meusnier point M_s as βb.

Theorem: The curvature $k(s)$ of curve is equal to the inverse of the radius of the curvature $\rho(s)$, i.e.

$$k(s) = \frac{1}{\rho(s)} \tag{72}$$

A sphere that touches the curve at at least third order exists if $w(s_0) \neq 0$ or $w(s_0) = k'(s_0) = 0$. It is called sphere of curvature (see figure 10) with midpoint

$$M = \mathfrak{x}(s_0) + \frac{1}{k}\mathfrak{n} - \frac{k'}{k^2 w}\mathfrak{b} \tag{73}$$

if $w(s_0) \neq 0$ or

$$M = \mathfrak{x}(s_0) + \frac{1}{k}\mathfrak{n} - y\mathfrak{b} \tag{74}$$

if $w(s_0) = k'(s_0) = 0$ and y arbitrary. [15]

4.3.3 Plane of curvature

Given a twice continuous derivable and not straight-lined curve in space

$$\mathfrak{x} = \mathfrak{x}(t) = x(t)e_1 + y(t)e_2 + z(t)e_3 \tag{75}$$

We choose a fixed point t_0 on the curve and close-by two movable points t_1 and t_2. These points determine a plane

$$\sigma(t_0, t_1, t_2) = \mathfrak{a} \cdot \mathfrak{y} + k = 0 \tag{76}$$

where $\mathfrak{a}$ is the normal vector of the plane, $\mathfrak{y}$ is the position vector of an arbitrary point on the plane and k is a scalar. The plane of curvature (see figure 10) in point t_0 of the plane is obtained if the movable points t_1 and t_2 simultaneously and independent of each other tend to the fixed point t_0. It follows that the plane of curvature $\sigma(t_0)$ contacts the curve $\mathfrak{x}(t)$ on position t_0 in three points.

If

$$f(t) = \mathfrak{a} \cdot \mathfrak{x}(t) + k = 0, \quad a \neq 0 \tag{77}$$

is true then the point t lies on the plane σ.

If the curve $\mathfrak{x}(t)$ is twice continuous derivable then the function $f(t)$ has a first and a second derivation $f'(t)$ and $f''(t)$. Because the three points t_0, t_1 and t_2 lie on the plane σ it follows that

$$f(t_0) = 0, f(t_1) = 0, f(t_2) = 0 \tag{78}$$

Assuming a differentiable function $f(x)$ whose derivative is a continuous function and $f(a) = 0$ and $f(b) = 0$ $(a < b)$ then there must be at least one

point c between a and b at which $f'(c) = 0$. Therefore there has to be a position t_{01}^* between t_0 and t_1 and a position t_{12}^* between t_1 and t_2 where

$$f'(t_{01}^*) = 0, f'(t_{12}^*) = 0 \tag{79}$$

There's also a position t_{01}^{**} between t_{01}^* and t_{12}^* where

$$f''(t_{01}^{**}) = 0 \tag{80}$$

If the positions t_1 and t_2 tend to t_0, $t_1 \to t_0$ and $t_2 \to t_0$, then t_{01}^* and t_{01}^{**} also tend to t_0. From this it follows that

$$f(t_0) = 0, f'(t_0) = 0, f''(t_0) = 0 \tag{81}$$

or

$$\begin{aligned} f(t_0) &= \mathfrak{a} \cdot x(t_0) + k = 0 \\ f'(t_0) &= \mathfrak{a} \cdot x'(t_0) + k = 0 \\ f''(t_0) &= \mathfrak{a} \cdot x''(t_0) + k = 0 \end{aligned} \tag{82}$$

If the cross product at position t_0 is not equal to zero

$$x'(t_0) \times x''(t_0) \neq 0 \tag{83}$$

then the normal vector of the plane of curvature σ at position t_0 is

$$\mathfrak{a} = \lambda(x'(t_0) \times x''(t_0)), \qquad \lambda \neq 0 \tag{84}$$

Because

$$k = -\mathfrak{a}\mathfrak{x}(t_0) \tag{85}$$

and the formula of the normal vector can be cancelled by λ we can state the equation of the plane of curvature at position t_0 as

$$\begin{vmatrix} X - x(t_0) & x'(t_0) & x''(t_0) \\ Y - y(t_0) & y'(t_0) & y''(t_0) \\ Z - z(t_0) & z'(t_0) & z''(t_0) \end{vmatrix} = 0 \tag{86}$$

Theorem: The twice continuous derivable non straight-lined curve in space $\mathfrak{x}(t)$ has in all points t_0, provided that the derived vectors $\mathfrak{x}'$ and $\mathfrak{x}''$ are linearly independent, i.e. $\mathfrak{x}'(t_0) \times \mathfrak{x}''(t_0) \neq 0$, a uniquely determined plane of curvature $\sigma(t_0)$. [29]

In other words, the plane of curvature is unique for $k(s) \neq 0$. It is determined through the point $\mathfrak{x}(s)$ and its normal vector, the binormal $\mathfrak{b}$ of the curve in point $\mathfrak{x}(s)$. The plane is spanned of the tangent $\mathfrak{t}$ and the normal $\mathfrak{n}$ in point $\mathfrak{x}(s)$:

$$\sigma = \mathfrak{x}(s) + s\mathfrak{t} + t\mathfrak{n} \tag{87}$$

There are another two important planes through point $\mathfrak{x}(s)$ spanned of the vectors of the frenet frame, the normal plane and the rectifying plane. The normal plane is perpendicular to the tangent $\mathfrak{t}$ of the curve in point $\mathfrak{x}(s)$ and is spanned of the normal $\mathfrak{n}$ and the binormal $\mathfrak{b}$ in point $\mathfrak{x}(s)$. The rectifying plane is orthogonal to the normal $\mathfrak{n}$ of the curve in point $\mathfrak{x}(s)$ and is spanned of the tangent $\mathfrak{t}$ and the binormal $\mathfrak{b}$ in point $\mathfrak{x}(s)$. The normal plane, the rectifying plane and the plane of curvature are each normal to one another and spanned of two of the vectors forming the Frenet frame, the tangent, the normal and the binormal vector of the curve in point $\mathfrak{x}(s)$. [15]

4.4 Differential geometry on a surface

As for the curves in section 4.3 some basic mathematical and geometrical principles necessary for differential geometry on a surface will be introduced in this section. At the beginning, the idea of surface curvature and their different types will be explained. After that, the concept of the Frenet frame in surface points (similar to the Frenet frame in curve points presented in subsection 4.3.1) will be presented. Next, a special point, the Meusnier point, will be explained. Finally a short explanation of general sweeps is given.

4.4.1 Surface curvature

The curvature of a given point on a surface stands for the variation of the normal vector in this point when moving the point on the surface. The principle of curvature is easier explained with curves but can easily be extended to surfaces.

Figure 11 shows a surface and the surface normal n at a given point P. The corresponding tangent plane of the surface in that point (which is determined by the point and the surface normal n) is spanned by two vectors e_1 and e_2. If one imagines each of these two vectors, e.g. e_1, to span a plane with the surface normal n this plane is orthogonal to the tangent plane, i.e. it is a normal plane. The result of the intersection of this normal plane and the surface is a plane curve, e.g. c_1, which lies within this normal plane and goes through point P. This curve c_1 has a specific curvature in point P and this type of curvature is called *normal curvature* of the surface.

Of course, there is an infinite amount of directions and corresponding intersection curves at a given point on a surface. The two most interesting

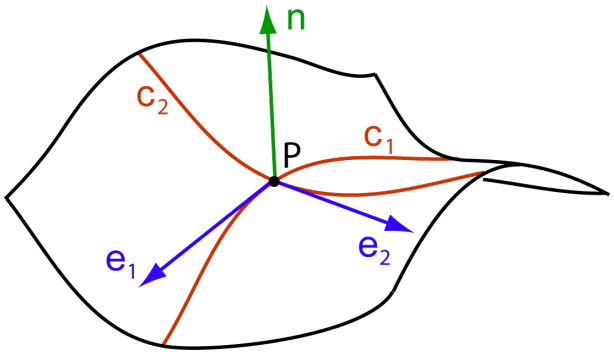

Figure 11: This figure shows a surface and the surface normal n in surface point P. The vectors e_1 and e_2 indicate the principal curvature directions. Each one of these directions can span a plane with the surface normal n. The resulting normal planes can further be intersected with the surface. This results in the plane intersection curves c_1 and c_2. The curvatures of these intersection curves in point P are the maximum and the minimum curvature magnitudes κ_1 and κ_2.

directions, the directions of the *maximum* and the *minimum curvature* are called principal curvature directions e_1 and e_2. The corresponding scalar curvatures are referred to as principal curvature magnitudes κ_1 and κ_2. As κ_1 denotes the maximum curvature $\kappa_1 \geq \kappa_2$ is always true. The curvature of a surface point in any direction can be calculated from the principal curvature directions. [8]

Surfaces can be regarded as an image of a plane region G, i.e.

$$\mathfrak{x} = \mathfrak{x}(u_1, u_2) = (x(u_1, u_2), y(u_1, u_2), z(u_1, u_2)) \tag{88}$$

This parameter representation will be feasible, if and only if it is triply continuous derivable and the partial derivatives $\mathfrak{x}_1$ and $\mathfrak{x}_2$ are not parallel.

If one parameter is constant, the equation is similar to the parameter representation of a curve. If we assume parameter u_1 constant, we obtain a curve with a parameter u_2 and vice versa. Tangent vectors of these parameter lines can be determined by partial derivatives of the curve

$$\mathfrak{x}_i = \frac{\partial \mathfrak{x}(u_1, u_2)}{\partial u_i} \tag{89}$$

For a feasible parameter representation it is also mandatory that [15]

$$\mathfrak{x}_1 \times \mathfrak{x}_2 \neq 0, \qquad \forall (u_1, u_2) \in G \tag{90}$$

4.4.2 Types of surface curvature and special points

Based on their principal curvature, magnitudes κ_1 and κ_2 surface points and the corresponding surface curvatures can be classified in the following categories (see figure 12):

- *Elliptical points*: $\kappa_1, \kappa_2 > 0$, $\kappa_1 \neq \kappa_2$, convex curvature, elliptical surface curvature
- *Hyperbolic points*: $\kappa_1 > 0$, $\kappa_2 < 0$, surface is saddle-shaped, hyperbolical surface curvature
- *Parabolic points*: κ_1 or κ_2 is zero, surface is locally cylinder-shaped, parabolic surface curvature
- *Umbilical points*: $\kappa_1 = \kappa_2$, locally sphere-shaped or planar, curvature directions are not well-defined. Spheres and planes consist only of umbilical points with constant curvature. [8]

4.4.3 Frenet frame in surface points

Given a feasible parametric representation of surface $\mathfrak{x} = \mathfrak{x}(u_1, u_2)$, an arbitrary surface point $\mathfrak{x}(s)$ and an arbitrary surface curve $\mathfrak{x}(t) = \mathfrak{x}(u_1(t), u_2(t))$ through point $\mathfrak{x}$. The tangent vector

$$\mathfrak{x}' = \mathfrak{x}_1 u_1' + \mathfrak{x}_2 u_2' \tag{91}$$

lies in the plane ϵ spanned of $\mathfrak{x}_1$ and $\mathfrak{x}_2$.

Given a second feasible parametric representation of the same surface with the parameters $\overline{u_1}$ and $\overline{u_2}$, the tangent vectors $\overline{\mathfrak{x}_1}$ and $\overline{\mathfrak{x}_2}$ of the parametric lines $\overline{u_1}$ and $\overline{u_2}$ through point $\mathfrak{x}(s)$ lie within the plane ϵ. That means the plane ϵ is independent of the chosen parameter values of the surface and is called tangent plane.

The cross product

$$\mathfrak{N} = \mathfrak{N}(u_1, u_2) = \frac{\mathfrak{x}_1 \times \mathfrak{x}_2}{|\mathfrak{x}_1 \times \mathfrak{x}_2|} \tag{92}$$

is orthogonal to the plane ϵ and is called the normal vector of the plane.

The triplet

$$\mathfrak{x}_1, \mathfrak{x}_2, \mathfrak{N}$$

is called Frenet frame of the surface.

The Frenet frame is a right-handed system with

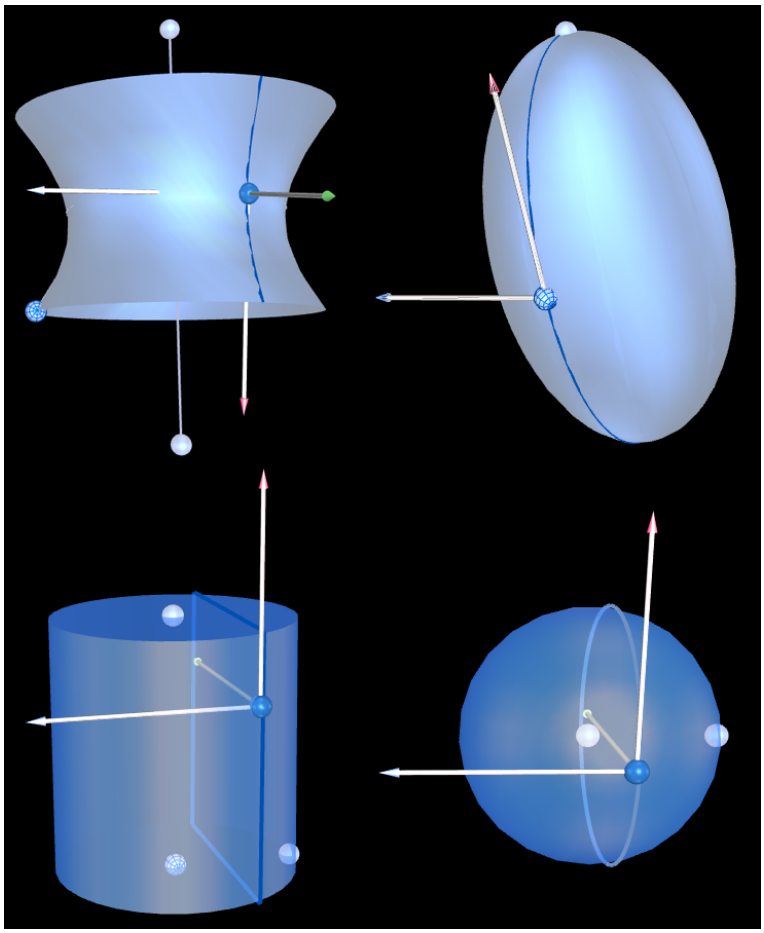

Figure 12: This figure shows different types of surface curvature and their corresponding special points (from left to right and from top to bottom) as explained in section 4.4.2 for arbitrary curves on the surface: a hyperboloid (hyperbolic points), an ellipsoid (elliptical points), a cylinder (parabolic points) and a sphere (umbilical points). The Frenet frame in a point of a surface curve is also shown for every object.

$$\mathfrak{x}_1, \mathfrak{x}_2 \perp \mathfrak{N} \tag{93}$$

and

$$\mathfrak{N}^2 = 1 \tag{94}$$

although generally $\mathfrak{x}_1$ and $\mathfrak{x}_2$ are not orthogonal unit vectors.

In general $\mathfrak{x}_1$ and $\mathfrak{x}_2$ are dependent of the chosen parameter values of the surface whereas $\mathfrak{N}$ is uniquely determined except of the prefix. [15]

4.4.4 Meusnier point

Given an arbitrary surface and a point $\mathfrak{x}(s)$ on this surface the tangent plane τ in point $\mathfrak{x}(s)$ approximates the surface in a linear way. Given an arbitrary tangent t in point $\mathfrak{x}(s)$ there are countless surface curves with this tangent t. Each of these curves c has its own circle of curvature. Meusnier has shown that all these circles of curvature lie on a shared sphere, called Meusnier sphere. The midpoint of the Meusnier sphere is called Meusnier point (see figure 17). [10]

4.5 Geometric properties of implemented surface classes

This section explains the geometric properties of the surface classes implemented in Constuct3D for this thesis. At the beginning the general properties and the concept of a helical transformation will be explained. After that, the special features of helices (helical transformation of a point) and helical sweep surfaces (helical transformation of a curve) and their special types will be introduced.

4.5.1 Helical transformations

A helical transformation is a spatial movement composed of a rotation around an axis and a proportional translation along that axis. It is imperative that

$$s = c \cdot \sigma \tag{95}$$

where s is the translation distance, σ is the angle of rotation and $c \neq 0$ is the helical parameter. If the helical parameter c is equal to zero, it will be only an ordinary rotation.

The corresponding distance of the translation $h = 2c\pi$ to a full revolution of $\sigma = 2\pi$ is called full height of the screw and can be directly measured on

a helical object. The helical parameter c will be obtained if the full height of the screw is known:

$$c = \frac{h}{2\pi} \approx \frac{h}{6} - 5\% \tag{96}$$

Assuming that the axis of the screw is the z-axis of a polar coordinate system (r, ϕ, z). The screw transformation of a point $P_0(r_0, \phi_0, z_0) \rightarrow P(r, \phi, z)$ can be described as

$$\begin{aligned} r &= r_0\sigma \\ \phi &= \phi_0 + \sigma \\ z &= z_0 + c\sigma \end{aligned} \tag{97}$$

Written in Cartesian coordinates where

$$x = r \cdot \cos\phi y = r \cdot \sin\phi \tag{98}$$

for the transformation $P_0(x_0, y_0, z_0) \rightarrow P(x, y, z)$ the following equations can be obtained:

$$\begin{aligned} x &= x_0 \cos\sigma - y_0 \sin\sigma \\ y &= x_0 \sin\sigma - y_0 \cos\sigma \\ z &= z_0 + c\sigma \end{aligned} \tag{99}$$

Looking along the axis contrary its orientation two types of helical rotation can be distinguished: if the rotation is counterclockwise it is a right-handed helical rotation, $c > 0$, otherwise a left-handed, $c < 0$. [34]

Table 1 summarizes surface types that can be generated by a helical transformation.

object to screw	helical object
point	helix, helical curve
curve	helicoid, helicoidal surface, helical sweep surface
straight line	ruled helicoid
circle	cyclic helicoid
surface	enveloping helicoid
plane	developable helicoid, helicoidal torse
sphere	tubular helicoid

Table 1: Summary of surface types generated by a helical transformation.

Helix A helix (see figure 13) is the path of a point subordinated to a helical transformation. The following parametric representation of a helix with fixed values x_0, y_0, z_0 and a variable angle of rotation σ can be obtained assuming that the x-axis of the coordinate system contains point P_0, i.e. $x_0 = r$ and $y_0 = z_0 = 0$:

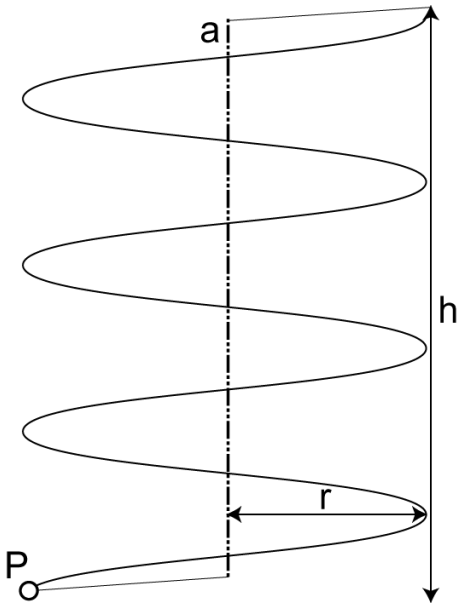

Figure 13: This figure shows an illustration of a helix in front view. A point P is rotated around an axis a. The radius is labeled as r and the full height of the helix is labeled as h.

$$\begin{aligned} x &= r \cdot \cos\sigma \\ y &= r \cdot \sin\sigma \\ z &= c\sigma \end{aligned} \tag{100}$$

The equations of y and z can be considered as a the upright projection of the xz-plane and the projection of the helix is a sine wave

$$y = r \cdot \sin\frac{z}{c} \tag{101}$$

Recapitulatory the upright projection of a helix on a plane parallel to the axis is a sine wave with a full height of $h = 2c\pi$ and the ground view on the xy-plane is a circle with radius r.

The helix has several special features: a helix lies on a coaxial cylinder Γ with radius r and intersects all generators at the same angle. That means the helix is a loxodrome of the cylinder. If the cylinder Γ is developed into a

plane the helix becomes a straight line, i.e. the helix is also a geodesic of the cylinder, i.e. the shortest path between two points.

Helical sweep surface A helical sweep surface Φ (see the screenshot in figure 19) is formed when a curve is transformed by a helical motion, i.e. rotated and simultaneously moved along an axis. The curve c is the generator of the helical sweep surface Φ. Below some types of helical sweep surfaces will be described. Details can be found in [2, 34].

There are several special cases of helical sweep surfaces: i.e. ruled helicoids and cyclic helicoids.

Ruled helicoids A ruled helicoid is generated if a straight line s is screwed around an axis. A ruled helicoid is called *closed* when the axis and the straight line intersect, otherwise it is called *open*. It is called *right* when the axis and the straight line are orthogonal, otherwise it is called *skew*. Hence the following cases can be discerned:

- *right closed ruled helicoid*:
  It is generated if a straight line s orthogonal to the axis, that also intersects the axis, is screwed. This type is also called right helicoid. The stages of a spiral staircase lie on such a type of helical surface.
- *right open ruled helicoid*:
  This surface is generated if a straight line s orthogonal to the axis, but not intersecting the axis, is screwed.
- *skew closed ruled helicoid*:
  This helical sweep surface is generated if a straight line s that intersects the axis in a skew angle γ is screwed.
- *skew open ruled helicoid*:
  This is the most general type of a helicoid. A straight line s that crosses the axis in a skew angle γ is screwed.
- *developable helicoid*:
  A developable helicoid consists of all tangents of a helix.

Circle helicoids If a circle c is screwed a circle helicoid is generated. Dependent of the position of the circle relative to the axis there exist several different cases:

- *circle helicoid*:
  The plane of the circle c is normal to the axis. It is only necessary to specify the helix m of circle's midpoint and the radius of the circle. All points of the circle create helices congruent to m. A circle helicoid can

also be created if the circle is swept along the helix m and therefore it is also a sweep surface.

- *meridian circle helicoid*:
  This type of helicoid is generated if the circle's plane lies in the same plane as the axis.
- *tubular helicoid*:
  If a sphere (assumed its midpoint lies not on the axis) is screwed, a surface will be generated that is both a helical surface and a tubular surface. The mid line of the tubular helicoid is the helix of the midpoint of the sphere. The surface of the tubular helicoid is touched by a great circle of the sphere whose plane is perpendicular to the mid line. Therefore this tubular helicoid can also be generated by this circle. [34]

4.5.2 General sweeps

For generating a general sweep surface (or translation surface) a profile curve p is moved along a second curve, the leading curve l. In most cases both curves share a point O. The profile and the leading curve may both be two and three dimensional curves.

For instance could a cylinder not only be seen as a straight line rotated around an straight-lined axis but also as a straight line (the profile curve) moved along a circle (the leading curve) perpendicular to the profile curve (see the screenshot in figure 20). The profile and the leading curve can also always change their roles. The cylinder can also be generated if the circle (the profile curve) is moved along a straight line (the leading curve). [34]

5 Design

This section will give an overview over the design ideas which have been implemented in the course of this thesis. First the start up process of Construct3D is described shortly. After that the functions which have been implemented are described generally together with their necessary Construct3D input elements and the resulting output elements. Finally, the extension of the user interface which became inevitable for integrating the new functions is mentioned.

5.1 Construct3D startup

Construct3D [14] is a 3D dynamic geometry construction tool and it is based on the Augmented Reality System Studierstube [26]. For further details of the general concept of Construct3D see section 3.3.

At its start up Construct3D initializes a 3D window with the largest possible size to create the illusion of covering the whole 3D space. After that the user interface is initialized, the personal interaction panel (PIP), a hand-held tracked tablet with the projection of a menu system on it. New points can be created by turning on the point mode on the PIP and pointing in the 3D space and pressing a button on a tracked pen. After turning off the point mode points and previously created objects can be selected with the pen.

5.2 Workflow for creating a new object

To create a new object in Construct3D first of all the user has to select the required input elements. If a user does not know which input elements are required this information will be found in the *Help notes* box positioned on the top of the PIP. By moving the pen over a construction menu item a help text is shown. For example, to create a circle of curvature, a curve and a point on this curve have to be selected. Points can be created when the point mode of Construct3D is turned on by clicking on the `SoToggleButton` *Point* on the right of the PIP with the aid of the pen (see figure 21). Then, by clicking the pen's button, a point in space will be created. After that the point mode has to be turned off again. Now the object or point closest to the tracked pen will be highlighted (the point gets overlaid with a white wireframe structure) and can be selected by clicking the pen's button. When creating a *Points on curve* curve the resulting curve will go exactly through the control points. Therefore the points have to be selected in the correct order. After that the sub menu *3D* in the PIP's construction menu has to be selected. By clicking on the `SoPushButton` *Points on curve* a curve through the previously selected point is generated (a `SoPushButton` is the implementation of an ordinary button which goes up and down and forces the immediate execution of a command).

Now this new curve has to be selected. After turning on the point mode again a point can be positioned on the curve by clicking close by the selected curve. The new point will now be positioned on the selected curve. The point mode has to be turned off again to be able to select the curve and the point on the curve. Now the *Diffgeo* sub menu must be chosen. At one time there are only four of six sub menus visible. If the *Diffgeo* sub menu is not visible in the construction menu bar the user will have to click on one of the little arrows left of the construction menu bar for switching between the sub menus. From the *Diffgeo* sub menu the *Circle of curvature* button ought to be chosen. If the user moves the pen over the button without clicking he will get a preview of the circle in a dotted line style. Finally, by clicking the button, the circle of curvature will be generated (see figure 16).

5.3 Differential geometry functions

This section describes shortly the general concepts of the newly implemented functions for differental geometry. To create an object in Construct3D the input elements have to be selected first. For every function the required input elements and the resulting output element will be given.

5.3.1 Frenet frame

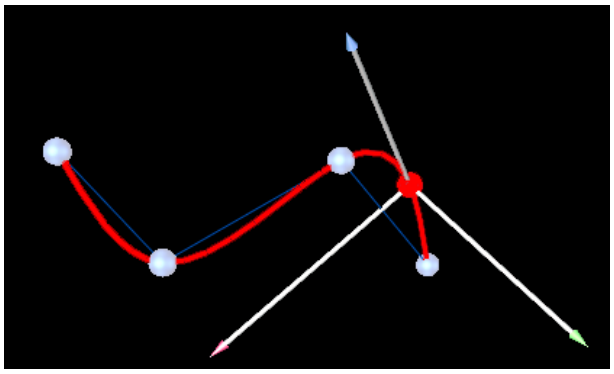

Figure 14: This figure shows a screenshot of a Frenet frame in Construct3D. The input elements, a curve and a point on this curve, are colored in red. The output element is the Frenet frame in the previously selected curve point consisting of the tangent, the normal and the binormal of the curve in that point.

The Frenet frame is a tool for the exploration of curves. It is a coordinate system that can be "glued" to a point on a curve. If the point on the curve is moved the Frenet frame will also move and adapt its orientation automatically. Using the natural parameter s one can exploit the advantage that also

the derivatives $\mathfrak{x}'(s)$, $\mathfrak{x}''(s)$ ...of a curve $\mathfrak{x}(s)$ are invariant of movement and parameter. For further details see sections 4.3.1 and 4.4.3.

Input: To create a Frenet frame (see the screenshot in figure 14) in a curve point in Construct3D, a curve and one or more points on this curve have to be selected.

Output: The Frenet frame is displayed in all previously selected curve points.

5.3.2 Plane of curvature

The plane of curvature in a curve point P is a plane that approximates this curve $\mathfrak{x}(s)$ best. It is spanned out of the tangent $\mathfrak{t}$ and the normal vector $\mathfrak{n}$ of curve $\mathfrak{x}(s)$ in point P. Its normal vector is the binormal vector $\mathfrak{b}$ of curve $\mathfrak{x}(s)$. For further details see section 4.3.3.

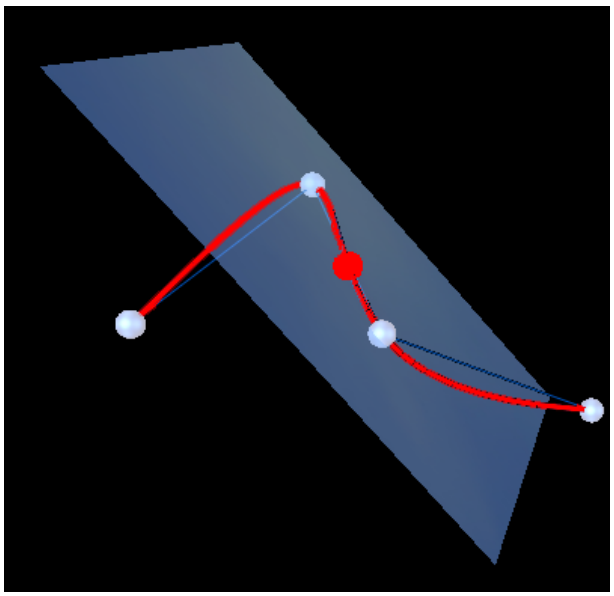

Figure 15: This figure shows a screenshot of the plane of curvature in Construct3D. The input elements, a curve and a point on this curve, are colored in red. The output element is the plane of curvature in the previously selected point.

Input: To create the plane of curvature (see the screenshot in figure 15) in a curve point in Construct3D a curve and a point on this curve have to be selected.

Output: The plane of curvature is displayed in the previously selected curve point.

5.3.3 Center and circle of curvature

The circle of curvature is the circle which approximates a curve $\mathfrak{x}(s)$ in a curve point P best. The circle of curvature's tangent in point P matches exactly the curve's tangent in point P. Its radius, the radius of curvature, is the reciprocal value of the curve's curvature in point P. The center of curvature is the midpoint of the circle of curvature. Because of the variation of the curve's curvature, the circle of curvature approximates the curve only in a very small neighbourhood of point P. For further details see section 4.3.2.

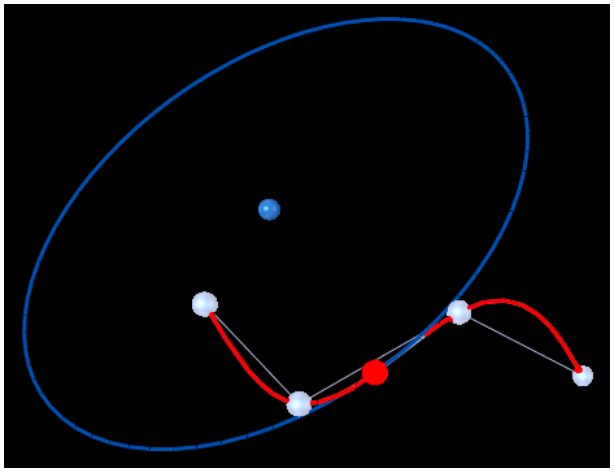

Figure 16: This figure shows a screenshot of the center and the circle of curvature in Construct3D. The input elements, a curve and a point on this curve, are colored in red. The output elements are the center and the circle of curvature in the previously selected point. The center and the circle of curvature are only displayed in one figure to illustrate the relationship between them. The center and the circle of curvature have to be constructed independently.

Input: To create the center or the circle of curvature (see the screenshot in figure 16) in a curve point in Construct3D a curve and a point on this curve have to be selected.

Output: The center or the circle of curvature are displayed in the previously selected curve point.

5.3.4 Meusnier point

Given an arbitrary surface and a point P on this surface. Given an arbitrary tangent $\mathfrak{t}$ in point P there are countless surface curves with this tangent. Each of these curves has its own circle of curvature. All these circles of curvature lie on a shared sphere, the Meusnier sphere. The midpoint of the Meusnier sphere is the so-called Meusnier point. For further details see section 4.4.4.

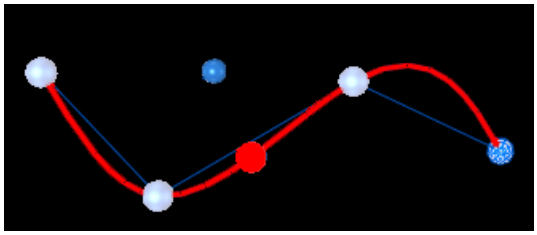

Figure 17: This figure shows a screenshot of the Meusnier point in Construct3D. The input elements, a curve and a point on this curve, are colored in red. The output element is the Meusnier point of the previously selected point.

Input: To create the Meusnier point (see the screenshot in figure 17) of a curve point in Construct3D a curve and a point on this curve have to be selected.

Output: The Meusnier point is displayed for the previously selected curve point.

5.4 Sweep functions

This section describes the general concepts of the implemented functions for sweep objects, i.e. helical and general sweeps. Again the input elements have to be selected first. For every function the required input elements and the resulting output element(s) will be declared.

5.4.1 Helical sweeps

A helical transformation is a spatial movement composed of a rotation around, and a proportional translation along, that axis.

Helix A helix is generated when a point P is rotated around, and proportionally moved along, an axis.

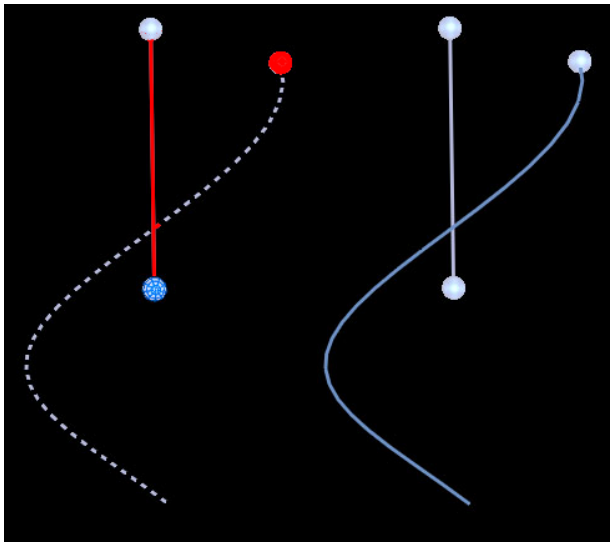

Figure 18: This figure shows a screenshot of a helix in Construct3D. The input elements, an axis (a straight line) and a point, are colored in red. The output element is the helix. On the left a preview of the helix in a dotted line style can be seen and on the right the generated helix.

Input: To create a helix in Construct3D (see the screenshot in figure 18) an axis (a straight line) and a point have to be selected.

Output: Output is the helix generated through a rotation around and a translation of a point along an axis.

Helical sweep surface A helical sweep surface is formed when a curve is transformed by a helical motion, i.e. rotated and simultaneously moved along an axis.

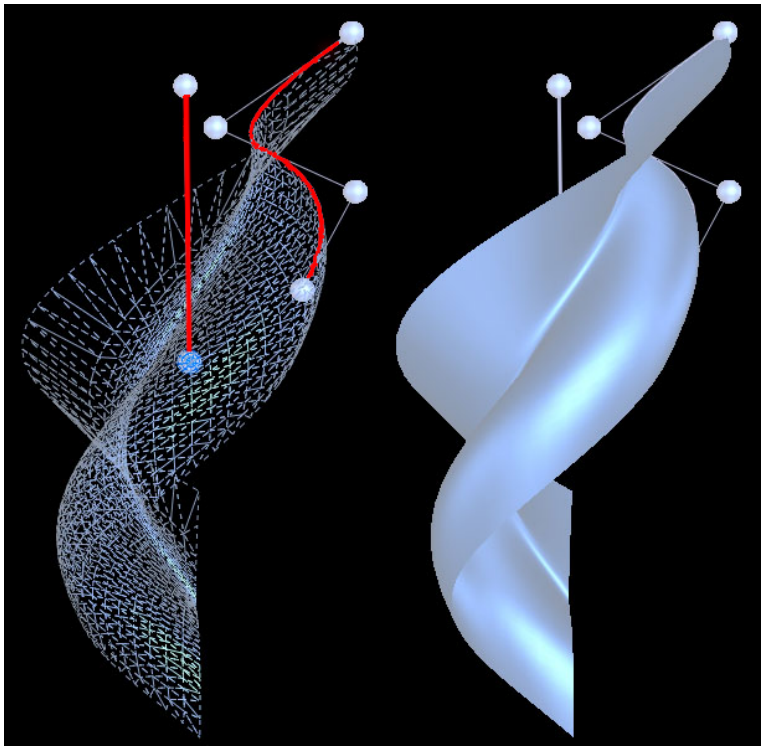

Figure 19: This figure shows a screenshot of a helical sweep surface in Construct3D. The input elements, an axis (a straight line) and a curve, are colored in red. The output element is the helical sweep surface. On the left a preview of the helix in wireframe can be seen and on the right the generated helical sweep surface.

Input: To create a helical sweep surface in Construct3D (see the screenshot in figure 19) an axis (a straight line) and a curve have to be selected.

Output: Output is the helical sweep surface generated through a rotation around and a translation of a curve along an axis.

5.4.2 General sweeps

For generating a general sweep surface a profile curve p is moved along a second curve, the leading curve l. The profile and the leading curve may both be two and three dimensional curves.

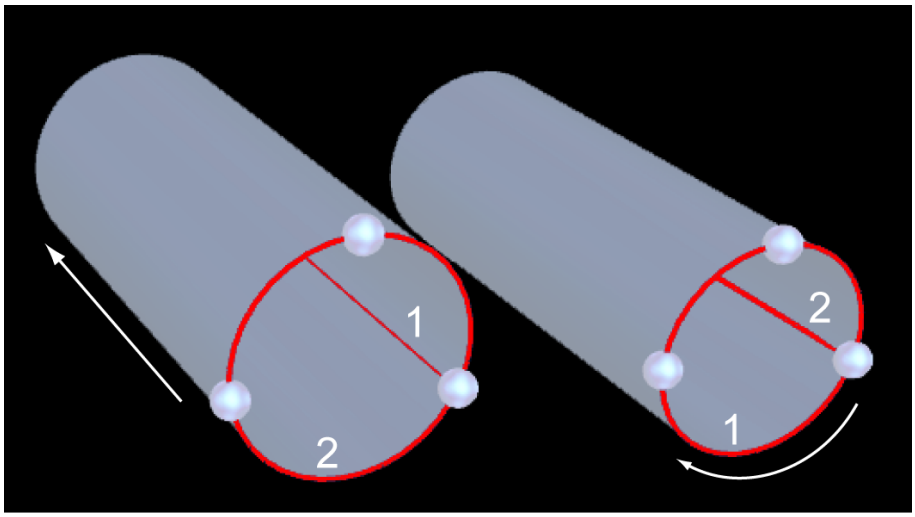

Figure 20: This figure shows a screenshot of general sweep surfaces in Construct3D. The input elements (colored in red) are in both cases a straight line and a circle. On the left the leading curve is the straigt line (1) and the circle is the profile curve (2). On the right leading and profile curve have changed their roles. The result is in both cases a cylinder (see section 4.5.2).

Input: To create a general sweep surface in Construct3D (see the screenshot in figure 20) an axis (a straight line) and one or more profile curves (straight lines or curves) have to be selected.

Output: Output are the general sweep surfaces created through a translation of the profile curves along the leading curve.

5.5 The Personal Interaction Panel (PIP)

The following section describes the extensions of the user interface which became necessary due to the introduction of the new construction menu item *Diffgeo*. Afterwards a short description of working with Construct3D in a desktop setup is given.

5.5.1 Extension of the user interface

To integrate the new functions in the existing layout of the personal interaction panel (PIP) new menu items had to be inserted (see figure 21).

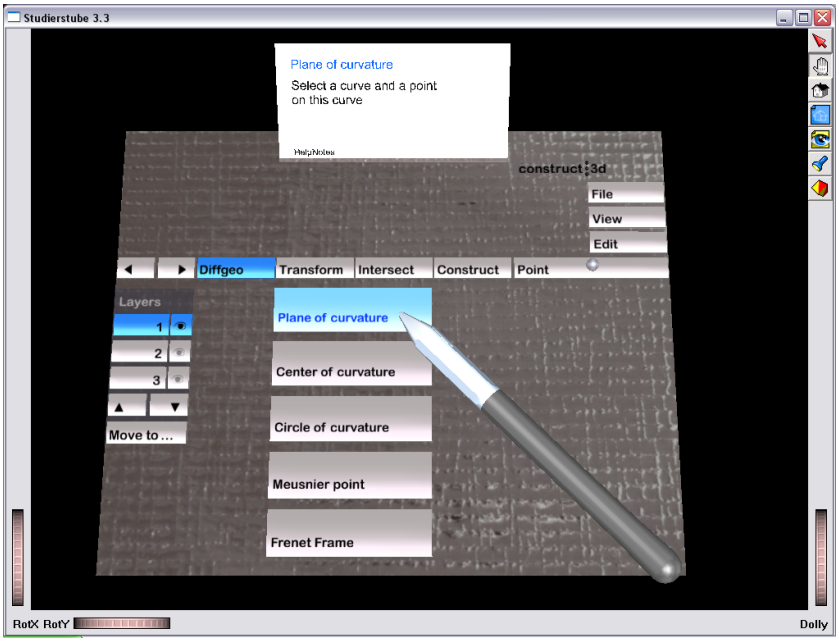

Figure 21: Screenshot of the Construct3D Diffgeo menu: it shows the submenu items of the new diffgeo functions. In the help notes field at the top the objects which have to be selected for the creation of the requested object can be seen. At the left from the construction menu there are two arrows for cycling through all the construction menu's options.

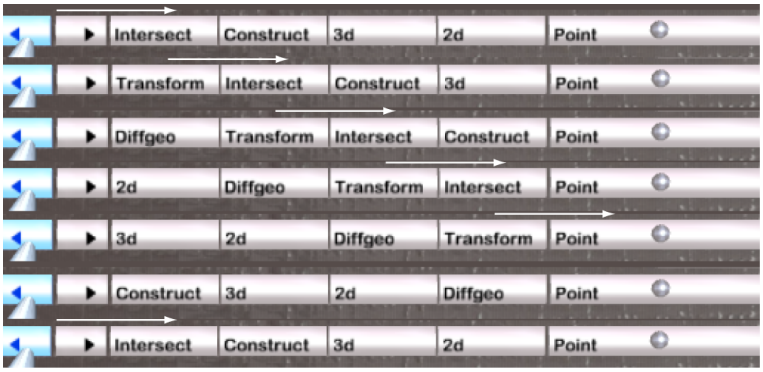

Figure 22: This figure illustrates the scrolling function for the PIP's construction menu items. Every new construction menu bar stands for a click on the left arrow button. After clicking the left arrow button the construction menu items shift one step to the right. Respectively one new menu item appears from the left and the rightmost menu item disappears.

To maintain the balanced layout of the PIP relative to size and position of the displayed items the new menu button *Diffgeo* (abbrevation for differential geometry) should not just be positioned next to the other menu items. Instead a scrolling function for the menu items was choosen (see figure 22). Two arrow buttons left to the construction menu items can be used to shift the menu's items left or right. After the last menu item the first menu item is displayed again, i.e. the menu items build a cycle.

See section 3.3 and [18] for further general details on the PIP. See section 6.6 for details on the implementation of the changes described above.

5.5.2 Keyboard shortcuts for the desktop setup

For quick testing during the implementation process working with Construct3D's desktop setup is the most efficient way. In this setup no extra hardware like a head mounted display, a tablet, a pen or a tracking system is necessary. The user works with a conventional personal computer displaying desktop virtual reality. The mouse can only be used to change the viewpoint. For moving the PIP and the pen the keyboard must be used (via the cursor keys and page up and page down). To simulate some of the possibilities of a setup with a freely movable PIP and a pen some keyboard shortcuts can be used (see table 2).

shortcut	function
home	view point returns to home position showing the PIP
end	view point moves backwards showing the PIP and the coordinate system
0	moving the pen
1	moving the PIP
w	pen returns to home position
e	counter-clockwise rotation around x-axis
d	clockwise rotation around x-axis
g	counter-clockwise rotation around y-axis
t	clockwise rotation around y-axis
r	counter-clockwise rotation around z-axis
f	clockwise rotation around z-axis

Table 2: Some useful keyboard shortcuts for the desktop setup. The specifications are for a right-handed three-dimensional coordinate system with the y-axis pointing away from the user.

6 Implementation

This section explains the general class structure of Construct3D. Besides it will show how the new functions for differential geometry (center, circle and plane of curvature, Meusnier point, Frenet frame) and sweep objects (helical and general sweep objects) have been integrated into existing classes. Then the program flow of Construct3D during the execution of a new function will be explained. Finally the integration of new menu elements in the Personal Interaction Panel will be described.

6.1 Class overview

The main class of Construct3D (see class overview in figure 23) is the user interface class `C3D`. Its super class is the Studierstube node kit class `SoContextKit` which contains the application functionality. It also provides the window and the PIP sheet geometry. The user interface class `C3D` is responsible for the connection of the PIP's widgets to methods for object creation. To handle incoming events efficiently `C3D` also inherits methods of Studierstube's `Base3D` class. Geometric objects are initialized by methods of `C3D` such as `addPoint()` or `addCurve()`. These methods create objects of subclasses of the `Object3DKit`. `Object3DKit` is the super class for all geometric objects in Construct3D. To enable the dragging of the objects `Object3DKit` is derived from Studierstube's `SoDragKit`. For event handling purposes regarding the drag and drop of the geometric objects `SoDragKit` is derived from Studierstube's `Base3D` class. It is also derived from the class `SoBaseKit`, the toplevel super class for node kits.

All geometric objects in Construct3D are derived from `Object3DKit`: `SoPointKit`, `SoLineKit`, `SoPlaneKit`, `SoCubeKit`, `SoCurveKit`, `SoSurfaceKit`, `SoSphereKit`, `SoCylinderKit`, `SoConeKit`, `SoBoolKit`, `SoIntersectionKit` and `SoTextKit`. `Object3DKit` contains general functionality needed by all geometric objects of Construct3D such as setting the selection and highlighting state, assigning of colors and materials (defined in `MaterialConstants`), determining an object's layer, recording the user identity and object deletion. `Object3DKit` has also a method for detecting a user's wish for dragging an object or for the generation of fixed objects that cannot be dragged. General constants for Construct3D are stored in `C3DConstants`.

6.2 Object creation

To create new objects buttons on the PIP have to be pressed and in succession the corresponding functions for adding a new object in the function `findButtonMethod()` are called. For the creation of the differential geometry

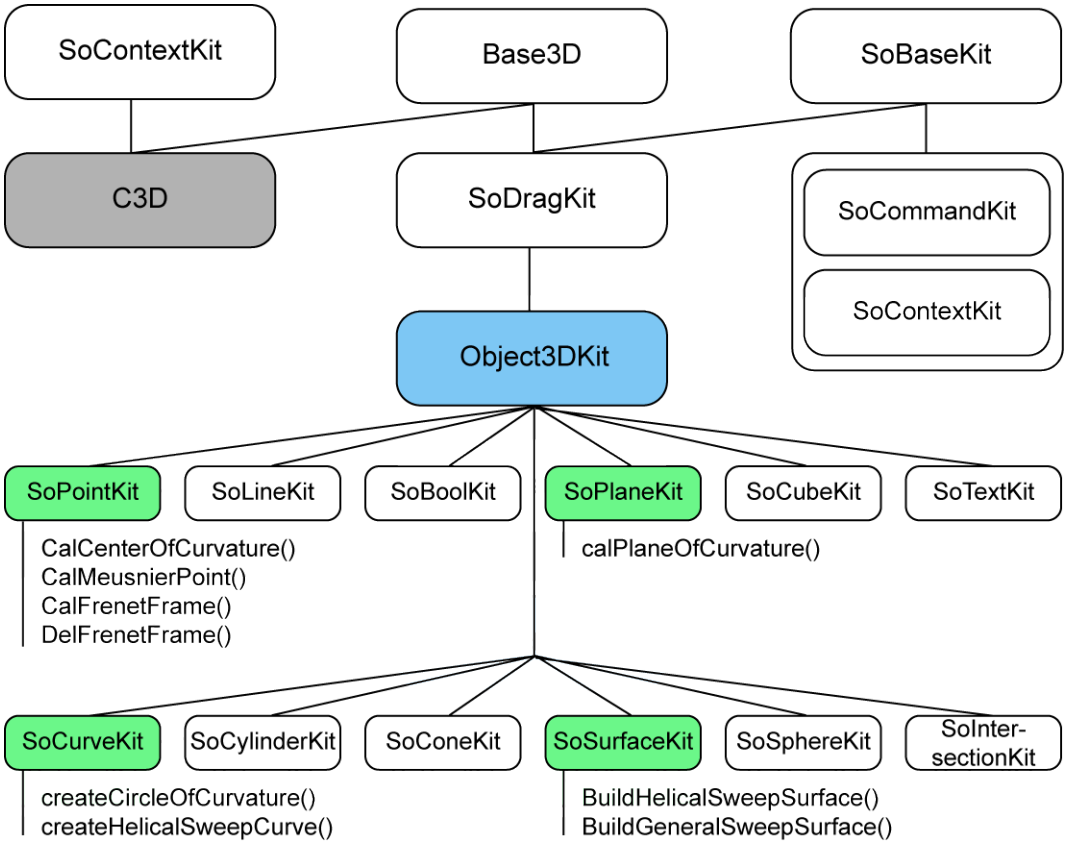

Figure 23: Class hierarchy of Construct3D: At the bottom of the class diagram the classes `SoPointKit`, `SoPlaneKit`, `SoCurveKit` and `SoSurfaceKit` are shown, augmented with the newly added functions for differential geometry and helical and general sweep functionality.

tools a curve and a point on that curve have to be selected. For the helical sweep objects a straight line (the axis) and another straight line or curve (the object to be swept) and for the general sweep objects at least one object to be swept must be selected.

In the body of such a function, e.g. `addCenterOfCurvature()` the number of selected objects is calculated. If enough objects, in most cases at least two objects, have been selected a new `Kit` object is generated, for example a `SoPointKit`. If the object creation was successful, properties like the draw style will be set. After that the new object is added to the scene graph.

The classes `SoPointKit`, `SoPlaneKit`, `SoCurveKit` and `SoPlaneKit` are subclasses of the general class `Object3dKit`. Further important class dependencies can be seen in figure 23.

For the implementation of the tools for differential geometry the classes

`SoPointKit`, `SoCurveKit` and `SoPlaneKit` have been extended by functions for the calculation of the center, the circle and the plane of curvature as well as the Meusnier point and the Frenet frame. The geometrical theory of these concepts is described in section 4. The Frenet frame was implemented as a tool which can be switched on and off for previously selected points by pressing a specific button.

For the creation of helical and general sweep objects the classes `SoCurveKit` and `SoSurfaceKit` have been augmented by functions for the computation of helices, helical sweep surfaces and general sweep surfaces.

6.3 Differential geometry functions

6.3.1 Constructor and destructor

In all extended classes further constructors were added. A new parameter, the object type as an enumeration, was added to distinguish between the different types of objects to be generated.

Within the constructor some properties of the object to be generated like size, name, number, material and behaviour relative to translation and rotation are set. In the next step it is checked if enough objects of the correct type have been selected.

For the calculation of the differential functions always a curve and at least one point on that curve have to be selected. If the selected point does not lie on the curve the calculation will be aborted and a warning message will be sent to the console output. Otherwise the object type is saved in an enumeration and the names of the objects are stored in objects of the type `SoSFName`.

For the helical objects a sweep object and an object to sweep around have to be selected. In succession several other functions refreshing the object lists are called.

The functions `CreateSensors()` and `setUpConnections()` are called to create and attach sensors which will react to every change applied to the underlying objects and result in a recalculation of the dependent objects. Afterwards the function `UpdateAcisObject()` is called where a function for generating the object corresponding to the object type is called.

In the destructor the previously generated sensors are detached and finally deleted.

6.3.2 Common precalculations

For the center, the circle and the plane of curvature, the Meusnier point and the Frenet frame the same precalculations have to be made. After the initialization

of variables necessary for the following calculations, the selected curve and the point on the curve are obtained with help of their object names.

There are two types of curves which have to be treated separately. Ordinary curves and curves resulting from an intersection have a different structure in their boundary representation (see 3.4.1) and are therefore stored as an EDGE (`a_Curve`) or a BODY (`a_intCurve`).

Then the actual position of the point and the corresponding parameter value on the curve are obtained. Using the ACIS function `eval()` that takes the parameter and the position of the point as parameters, the first and the second derivation of the curve at the given position can be calculated. From the cross product out of the first and the second derivation the binormal vector is computed. Because the second derivation is not necessarily orthogonal both to the tangent vector (the first derivation) and the binormal vector, the normal vector is calculated as the cross product out of the first derivation and the binormal vector.

The curvature of the curve at that position is calculated as the length of the binormal vector divided by the length of the tangent vector to the power of three

$$\text{curvature} = \frac{|\text{binormal vector}|}{|\text{tangent vector}|^3} \tag{102}$$

6.3.3 SoPointKit

The class `SoPointKit` was extended by several functions for the calculation of the Frenet frame (section 4.3.1), the center of curvature (section 4.3.2) and the Meusner point (section 4.4.4).

CalCenterOfCurvature() As stated in section 4.3.2 the radius of a circle is the reciprocal value of its curvature. The midpoint of the circle of curvature, the center of curvature, is obtained as the point on the curve is moved in direction of the normal vector by the amount of the curvature radius

$$\text{center of curvature} = \text{point} + (\text{norm}(\text{normal vector}) * \text{curvature radius}) \tag{103}$$

Finally the position of the new point has to be converted into coordinates for the point.

CalMeusnierPoint() For the calculation of the Meusnier point the first three derivations are needed. These can be obtained with the ACIS function

`evaluation()` that takes the parameter and position of the point as parameters. If the third derivation is equal to zero the Meusnier point cannot be calculated and an error message is returned.

After the calculation of the curvature also the torsion has to be computed. First the torsion determinant must be specified where the colums are the vectors of the first three derivations. To compute the torsion itself the torsion determinant is divided by the square of the length of the binormal vector.

$$\text{torsion} = \frac{\text{torsion determinant}}{|\text{binormal vector}|^2} \tag{104}$$

If the determinant of the torsion is equal or nearly equal to zero an error message is sent to the console output because in that case no Meusnier point exists.

In the next step the derivation of the curvature expression 102 has to be calculated. The Meusnier point is obtained as the center of curvature is moved again in direction of the binormal vector by the amount of derived curvature radius divided by the torsion

$$\begin{aligned} \text{Meusnier point} &= \text{point} + \\ (\text{norm(normal vector)} &* \text{curvature radius}) + \\ \left(\text{norm(binormal vector)} * \frac{\text{deriv(curvature radius)}}{\text{torsion}}\right) & \end{aligned} \tag{105}$$

Finally the position of the point has to be converted in coordinates.

CalFrenetFrame() The function `CalFrenetFrame()` takes a curve as parameter because a point can be part of more than one curve. After the initialization of the variables the rotation node of class type `SoRotation` for the Frenet frame is fetched. As stated above the derivations at the given curve position are calculated. Then the rotation field of the rotation node is set according to the directions of the calculated derivations. Therefore the attached node with a set of tiny coordinate axes is rotated accordingly. Finally the Frenet frame is set visible.

DelFrenetFrame() The purpose of the function `DelFrenetFrame()` is to set the previously calculated Frenet frame of this point invisible.

6.3.4 SoCurveKit

The class `SoCurveKit` was extended by a function for the calculation of the circle of curvature (see section 4.3.2).

createCircleOfCurvature() Because the center of curvature is the midpoint of the circle of curvature the same calculations as for the center of curvature have to be made (see section 6.3.3). After obtaining the position of the center of curvature an ellipse is created with the ACIS function `api_mk_ed_ellipse()`. The function takes amongst others the position of the midpoint, the normal vector of the circle's carrier surface and the distance between the midpoint and the point on the curve (the radius). If no circle can be created, an error message will be sent to the console output.

6.3.5 SoPlaneKit

The class `SoPlaneKit` has been augmented with the function `calPlaneOfCurvature()` for the calculation of the plane of curvature at a given point on a curve.

calPlaneOfCurvature() After obtaining the position and the parameter of the point on the curve the first and second derivation of the curve at that position are calculated via the ACIS function `eval()`. Here it is only necessary to calculate the binormal vector from the cross product of the first and second derivation. With the aid of a Construct3D intern macro which takes the position of a point on the plane and the normal vector of the plane (the binormal vector at that curve point) as parameters a plane is created. This generic plane has to be converted into a plane of a predefined size. Finally the plane has to be translated to center it around the point on the curve by manipulating the vertex properties of the plane's construction points.

6.4 Sweep functions

6.4.1 SoCurveKit

The class `SoCurveKit` was extended by the function `createHelicalSweepCurve()` in order to construct helices (4.5.1).

createHelicalSweepCurve() For the calculation of a helix some parameters have to be set as the helical parameter and the angle of rotation. At the moment these parameters are static but Construct3D can easily be adapted to dynamically change these parameters through the user interface.

First it is checked whether enough suitable objects were selected, in this case a straight line (the axis) and a point. If the rotational angle has been defined in degrees because this measure is easier to understand for the user, it has to be converted in polar coordinates.

In Construct3D straight lines are defined through two points and have therefore a defined length. Only straight lines resulting from a calculation, e.g. a straight line normal to a plane, have an infinite length. To ensure that the helix has the user defined height and does not end when the axis ends the straight line is explicitly extended to infinity.

With the ACIS function `api_edge_helix()` which takes as arguments the closest point to the point to screw on the axis, the endpoint of the helix' axis, the vector from the point to screw (the start point of the helix) and the start point of the helix' axis and also its distance and the handiness of the helix. If an error occures during the calculation, an error message will again be sent to the console output.

6.4.2 SoSurfaceKit

The class `SoSurfaceKit` was extended by two similar types of sweep surfaces, a helical sweep surface (function `BuildHelicalSweepSurface()`, see section 4.5.1) and "general" sweep surfaces (function `BuildGeneralSweepSurface()`).

For the creation of a sweep there exists a general function in ACIS `api_sweep_with_options()`. So-called sweep options have to be set to specify the properties of the resulting sweep surface like the behaviour at self intersection and whether the surface has to be solid.

BuildHelicalSweepSurface() Basically there is no great difference between the precalculations for a helix and a helical sweep surface. Also the helical parameter and the rotational angle have to be specified. Unlike to the helix where the order of the selected objects was of no relevance for a helical sweep object the order of the selected objects is important for providing the correct parameters to ACIS' sweeping function. At the moment the axis has to be selected first and the curve that shall be swept second. Then an arbitrary point from the curve is taken and the helix of this point around the axis is calculated. The ACIS function `api_sweep_with_options()` takes the object to sweep and the helix where the object is swept along as parameters. Finally if the calculation of the helical sweep surface was successful the resulting surface has to be triangulated. In case of an error an error message is displayed.

BuildGeneralSweepSurface() The ACIS function `api_sweep_with_options()` is a general function which takes an object to sweep and a path to sweep along as parameters. In case of the helical sweep curve the path to sweep along is a helix. Therefore it is easy to generalize this function to a general sweep function. At least two objects have to be selected. The first selected object, a straight line or a curve, becomes the path to sweep along and all following selected objects are swept along this path.

6.5 Undo/redo list

To provide an undo/redo functionality for every generated object an entry is stored in a special node kit, the so-called `UndoRedoListKit`. The `UndoRedoListKit` also provides a field containing the current position in the undo/redo list. For a preview of an object only a preview is shown and no entry in the undo/redo list is made. As a consequence a form of history is generated. If an inventor file is loaded at the startup of Construct3D only the commands in the undo/redo list will be processed. If an object is deleted the corresponding entry in the list will also be deleted. For an undo or a redo the counter in the `UndoRedoListKit` has to be decremented or incremented. Further details see in [5].

6.6 Extension of the user interface

The following code fragment of the Open Inventor file defining the PIP's layout (see section 3.2.2) shows the Open Inventor item hierarchy of the construction bar. After the definition of a `Separator` for the construction bar a `SoWidgetLayoutGroup` with attributes like width, depth, height and number of colums is defined to which the following items will be added. The buttons for shifting the menu items to the side are `SoPushButtons`. Some attributes like the used texture which is defined in a separate Open Inventor file only for textures are set. For the realization of the shifting functionality a `Switch` is used. All possible combinations of menu items are defined in `SoWidgetLayoutGroups`. The menu items itself are of type `SoToggleButton` because the visibility of the sub menu items is turned on and off every time the button is pushed. Already defined `SoToggleButtons` can be reused via the keyword `USE`.

```
DEF CONSTR_BAR Separator {
   SoWidgetLayoutGroup {
      width 0.036 depth 0.01 height 0.003
      numOfCols 2
      ...
      elements NodeKitListPart {
         containerNode Group {
            DEF CONSTR_LEFT_BUTTON SoPushButton {
               onGeometry Separator {
                  USE ON_GROUP USE
                  CONSTR_LEFT_BUTTON_TEX
                  USE LABEL_GEOMETRY }
               ...
               statusBoxText "65"
```

```
            }

            DEF CONSTR_RIGHT_BUTTON SoPushButton {...}
         }
      }
   }
   ...

   DEF CONSTR_SWITCH Switch {
      ...
      SoWidgetLayoutGroup {
         ...
         elements NodeKitListPart {
            containerNode Group {
               DEF DIFFGEO SoToggleButton {...}

               USE TRANSFORM
               USE INTERSECT
               USE CONSTRUCT
   ...
```

The items of the Diffgeo menu are defined in another `Separator`. Because there are submenu items for every menu item of the construction menu again a `Switch` is needed. The buttons for the differential geometry functions are `SoPushButtons` except for the Frenet frame which is from type `SoToggleButton` because the display of the Frenet frame can be switched on and off.

```
DEF WORK_SPACE Separator {
   DEF WORK_SPACE_SWITCH Switch {
      DEF DIFFGEO_PART Separator {
         SoWidgetLayoutGroup {
            elements NodeKitListPart {
               containerNode Group {
                  DEF CURVATURE_PLANE SoPushButton {...}
                  DEF CURVATURE_CENTER SoPushButton {...}
                  DEF CURVATURE_CIRCLE SoPushButton {...}
                  DEF MEUSNIER_POINT SoPushButton {...}
                  DEF FRENET_FRAME SoToggleButton {...}
   ...
```

The "Transform" menu was augmented with two buttons for the generation of a helical and a general sweep (see figure 24). The Transform menu has also

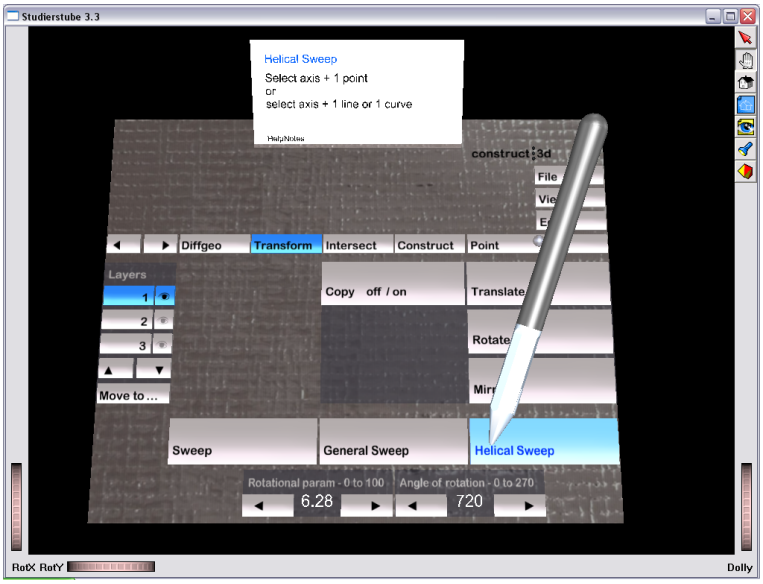

Figure 24: Screenshot of the Construct3D Transform menu: At the bottom the two options for the general and the helical sweep have been added. The prerequisites for a helical object can be seen at the top in the help notes field. At the bottom there are the fields prepared for the dynamic change of helical parameter and angle of rotation.

been prepared for the dynamic change of the helical parameter and the angle of rotation of a helical sweep.

The following code fragment shows the specification of a `Separator` for the help notes which defines the text shown if the pen moves over the Diffgeo menu button. `Translation` sets the position of the element depending on the elements higher in the hierarchy. Also the color of the font, the material and the text to be shown can be specified.

```
DEF SHOW_HELPNOTES Switch {
   ...
   Separator {
      Translation { translation 0 0.016 0 }
      USE FONT_COLOR AsciiText
         { justification LEFT string ["Diffgeo"] }
      Translation { translation 0 -0.00974 0 }
      Material { diffuseColor 0 0 0 }
      AsciiText { justification LEFT spacing 1.12985 string
         ["Opens the Diffgeo Menu -",
          "actions in this menu require that",
          "objects be selected first."] }
   ...
```

6.6.1 Routing of user interface events

The following section explains the definitions for the behaviour of the user interface scripted in Open Inventor.

Displaying the diffgeo menu First of all a `SoConditionalTrigger` for the diffgeo menu is defined. If the trigger's comparison evaluates to true the trigger field fires.

```
DEF DIFFGEO_TRIGGER SoConditionalTrigger {
   boolIn = USE DIFFGEO.on triggerBool TRUE token "0" }
```

Next the `SoFanIn`, an engine which is used to connect a number of fields to a single field, is augmented with the trigger for the diffgeo menu.

```
DEF CONSTR_FAN SoFanIn {
   type MFString
   in0 = USE DIFFGEO_TRIGGER.tokenOut
   in1 = USE TRANSFORM_TRIGGER.tokenOut
   in2 = USE INTERSECT_TRIGGER.tokenOut
   in3 = USE CONSTRUCT_TRIGGER.tokenOut
   in4 = USE THREE_D_TRIGGER.tokenOut
   in5 = USE TWO_D_TRIGGER.tokenOut
   in6 = USE CONSTR_NONE_TRIGGER.tokenOut
}
```

If another trigger for the diffgeo menu is fired the diffgeo state will be set on.

```
DEF DIFFGEO_STATE_ON SoConditionalTrigger {
   intIn -1 = USE CONSTR_FAN.out triggerInt 0
}
```

Finally routes are created via `SoRoute` between the previously defined fields. The output of the `CONSTR_FAN` field determines the child of the switch `WORK_SPACE_SWITCH`. If `DIFFGEO_STATE_ON` evaluates to true the diffgeo `SoToggleButton` is switched on.

```
SoRoute {
   from "CONSTR_FAN.out" to "WORK_SPACE_SWITCH.whichChild"
}

SoRoute {
   from "DIFFGEO_STATE_ON.boolOut" to "DIFFGEO.onIn"
}
```

Scrolling the menu items First two counters are defined via `SoCounter` which are engines that count from a specified minimum to a specified maximum value with a given step size each time a specified trigger fires. Because it is more intuitive the construction menu's items shall shift to the left if the right button is pressed and vice versa.

```
DEF NUM_CONSTR_LEFT SoCounter {
   min 0 max 5
   step 1
   trigger = USE CONSTR_RIGHT_BUTTON.triggerOut
   reset 0
}

DEF NUM_CONSTR_RIGHT SoCounter {
   min 0 max 5
   step 1
   trigger = USE CONSTR_LEFT_BUTTON.triggerOut
   reset 0
}
```

In the next step it is calculated which items of the construction menu are currently visible with the aid of a `SoCalculator`.

```
DEF CUR_CONSTR SoCalculator {
   a = USE NUM_CONSTR_RIGHT.output
   b = USE NUM_CONSTR_LEFT.output
   expression ["oa=a-b+6", "ob=oa\%6"]
}
```

Dependent of the `SoCalculator`'s output a route is defined from the result of the calculator to the construction menu's switch. As stated above, several different cases for the display of the construction menu's items have been defined and the result of the calculation routes to the correct menu items.

```
SoRoute {
   from "CUR_CONSTR.ob" to "CONSTR_SWITCH.whichChild"
}
```

7 Results

This concluding section shall give a short and, of course, only fragmentary survey of the possiblities of the newly implemented functions. The following screenshots are only simple presentations of the new functions. Hopefully the wide potential of the novel functions will be explored in future learning sessions.

7.1 Differential geometry functions

7.1.1 Frenet frame

The Frenet frame, a tool for understanding the meaning of curvature, can be displayed for every point on an arbitrary curve. Figure 25 shows the Frenet frame for an arbitrary 3D curve. The Frenet frame adapts dynamically when the curve or the point is changed.

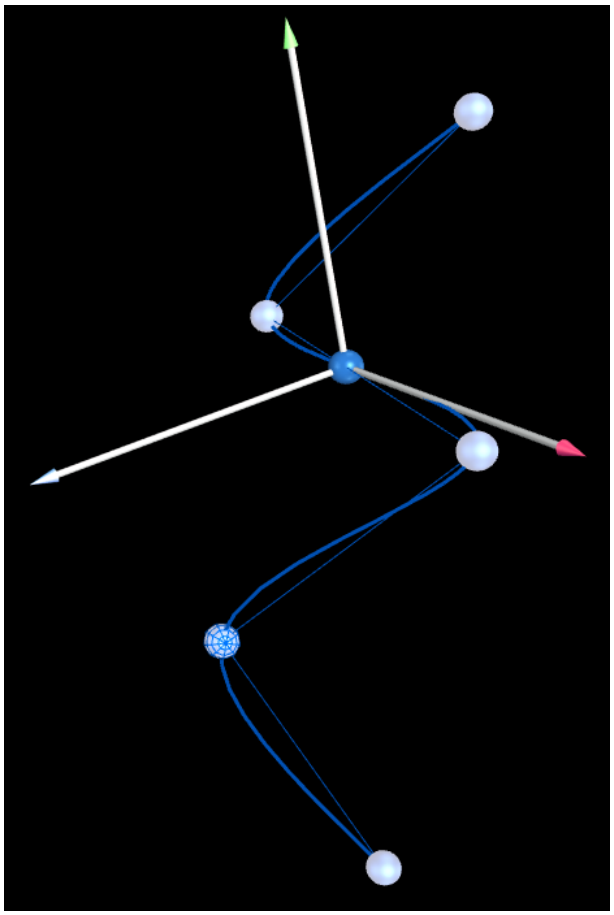

Figure 25: This figure shows the Frenet frame in a point on an arbitrary 3D curve.

7.1.2 Plane of curvature

The plane of curvature in a curve point is a plane that approximates this curve best. It is spanned out of the tangent and the normal vector of the curve in a point on a curve. Figure 26 shows the plane of curvature of a given curve point on an arbitrary 3D curve.

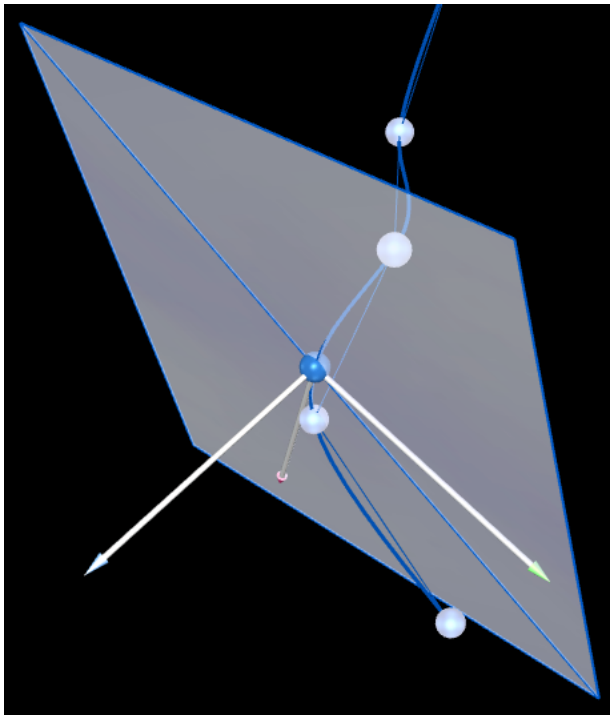

Figure 26: This figure shows the plane of curvature in a curve point on an arbitrary 3D curve.

7.1.3 Center and circle of curvature

Like the plane of curvature the circle of curvature is a basic form which approximates the curvature of a curve in a given curve point best. Figure 27 shows the circle of curvature in a given point on the intersection curve of two cylinders. The center of curvature is basically the midpoint of the circle of curvature.

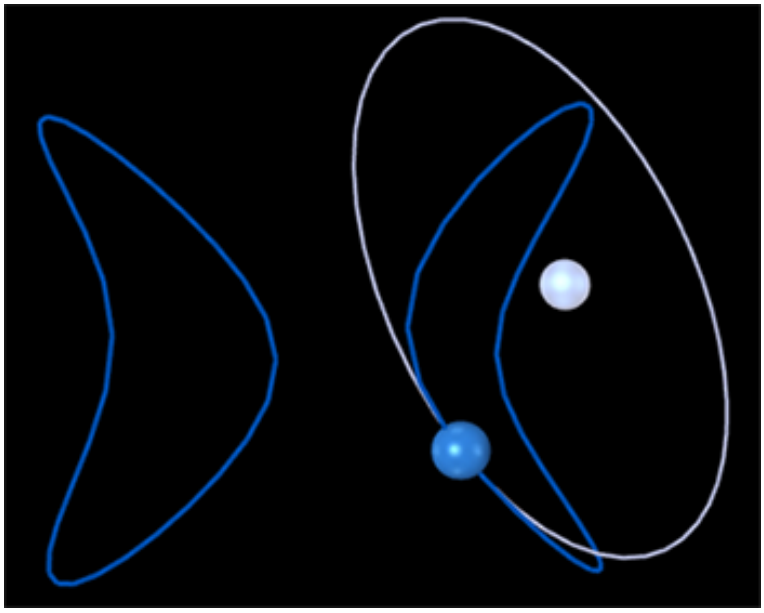

Figure 27: This figure shows the center and the circle of curvature in a curve point on the intersection curve of two cylinders.

7.1.4 Meusnier point

As explained in section 4.4.4 the Meusnier point is a special point. It is the midpoint of a sphere that contains all circles of curvature which pass through a specific point with a given fixed curve tangent. Figure 28 shows the Meusnier point and the Meusnier sphere for a given point on a 3D curve.

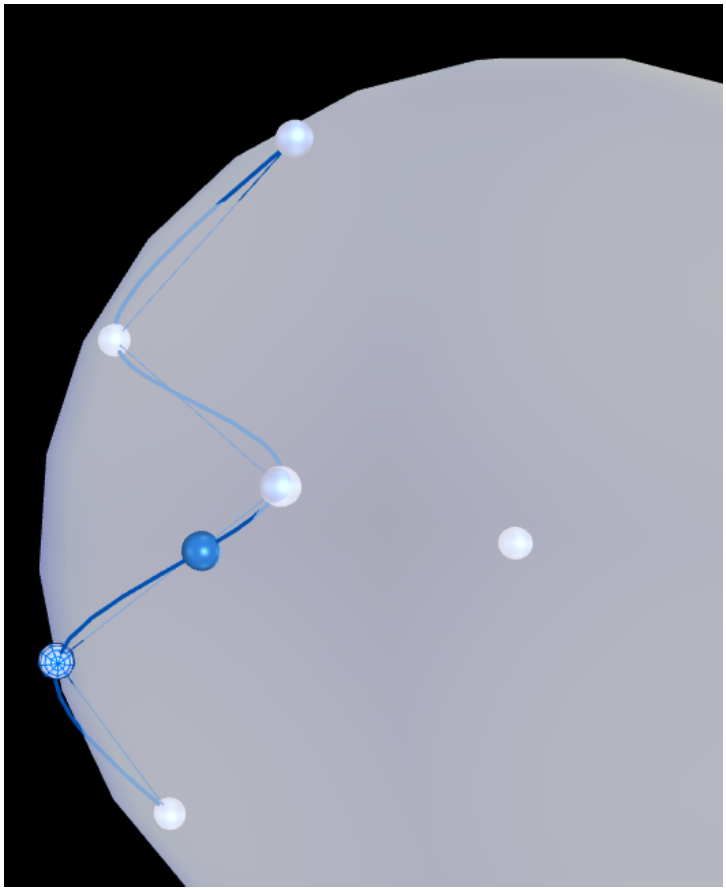

Figure 28: This figure shows the Meusnier point and the Meusnier sphere for a curve point on a 3D curve.

7.2 Sweep functions

7.2.1 Helical sweeps

Construct3D was augmented with functions to build helical sweep surfaces. These are sweeps where the sweep object is not only rotated around the sweep axis but also translated along that axis during the rotation. Sweep object can be a point or an arbitrary curve.

Ruled helicoids There are different forms of ruled helicoids (see section 4.5.1). A right closed ruled helicoid (see figure 29) is generated if a straight line normal to the sweep axis that also intersects the sweep axis is screwed. The stages of a spiral staircase lie on such a type of helicoid.

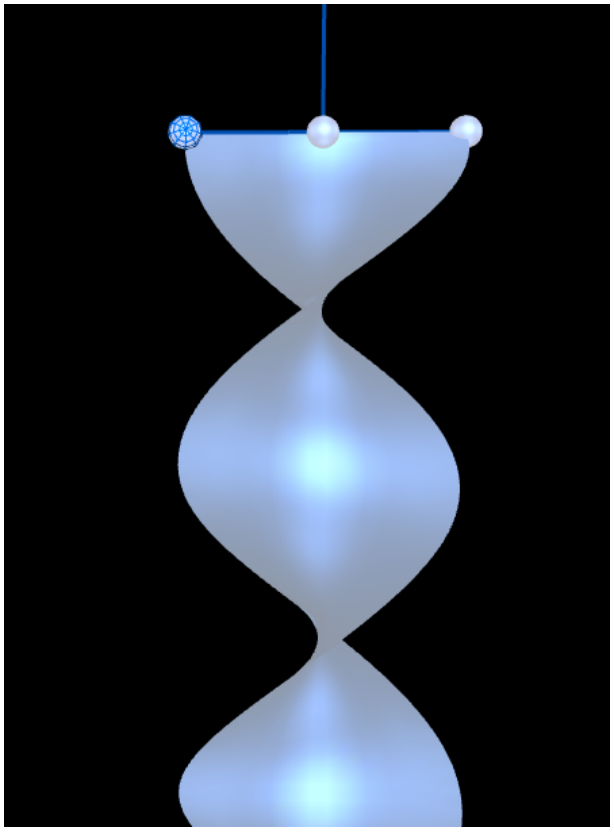

Figure 29: This figure shows a helical sweep surface with a straight line both as sweep axis and as sweep object.

Circle helicoids A circle that lies in the same plane as the axis which leads to a meridian circle helicoid (see figure 30).

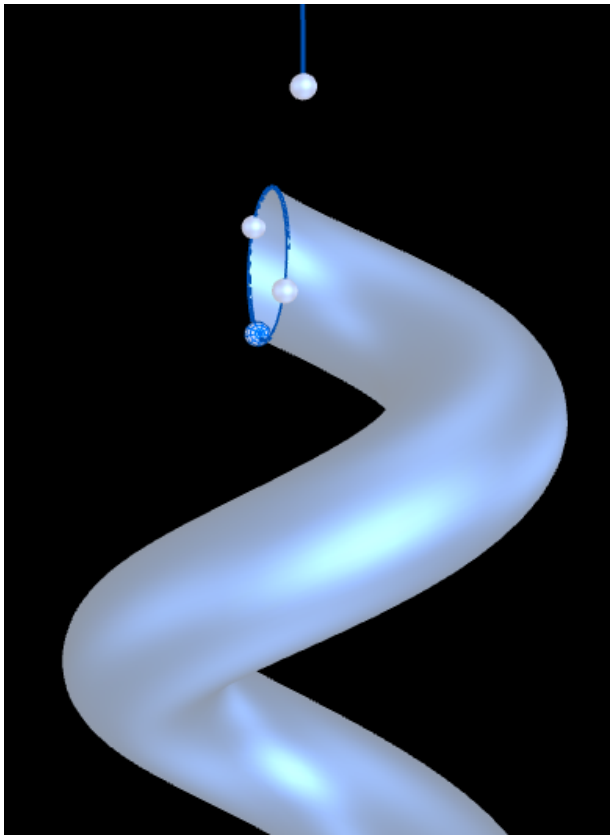

Figure 30: This figure shows a helical sweep surface with a straight line as sweep axis and a circle as sweep object.

7.2.2 General sweeps

A general sweep surface (or translation surface) is created when one or more profile curves are moved along a leading curve. Figure 31 shows a general sweep consisting of three straight lines forming a triangle that have been translated along a fourth straight line.

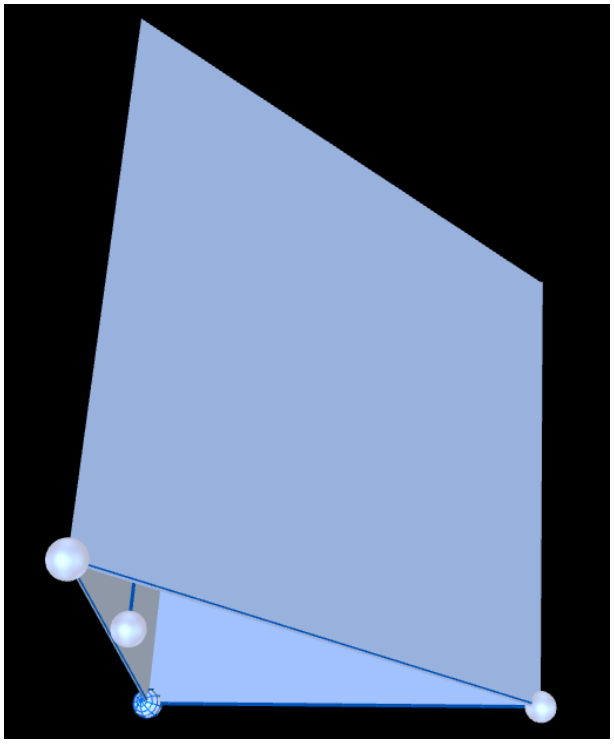

Figure 31: This figure shows a general sweep surface with a straight line as leading curve and three further straight lines as profile curves.

8 Conclusion

The aim of this thesis was to describe the newly implemented functions for Construct3D, a 3D dynamic geometry construction tool based on the Augmented Reality System Studierstube.

In the preceding chapters some software packages for static and dynamic geometry have been presented. Afterwards the technological foundations of this work have been described, the Open Inventor implementation Coin3D, the Studierstube framework, Construct3D and ACIS, a geometric modeler.
Following the basic geometry principles for understanding the new curvature tools—the Frenet frame, the plane, the center and the circle of curvature and the Meusnier point—as well as the helical and general sweep objects have been explained.
Adjacent the foundations of the practial work, the design and the implementation, have been described.
Finally some figures generated by Construct3D using the new functions have been shown.

On the one hand the author hopes that in the future students will get to know Construct3D and use its possibilities. On the other hand, hopefully, future students will help to expand the features of Construct3D. Anyway, may these people gain new insights, because that is what life is for.

List of References

[1] Gerd Baron and Peter Kirschenhofer. *Einführung in die Mathematik für Informatiker*, volume 2. Springer-Verlag, Wien, 1996.

[2] Heinrich Brauner. *Lehrbuch der konstruktiven Geometrie.* Springer-Verlag, Wien, 1986.

[3] Cabrilog. *Cabrilog > Products > Cabri 3D.* http://www.cabri.com. last checked: August 21, 2008.

[4] Jonathan Corney and Theodore Lim. *3D modeling with ACIS.* Saxe-Coburg Publications, Stirling, UK, 2001.

[5] Mathis Csisinko. *Long Distance Distribution of Virtual and Augmented Reality Applications*. Master's thesis, Vienna University of Technology, 2006.

[6] Andreas Dünser, Hannes Kaufmann, Karin Steigbügel, and Judith Glück. *Virtual and Augmented Reality as Spatial Ability Training Tools.* In *Proceedings of the CHINZ 2006, University of Canterbury, Christchurch, New Zealand*, pages 125–132, 2006.

[7] Hans-Jürgen Elschenbroich. *Unterrichtsentwicklung und Medieneinsatz im Fach Mathematik: Auf dem Weg zum Lernmittelkonzept – Eine Beratungshilfe.* In *Medienberatung NRW*, 2006.

[8] Klaus Engel, Markus Hadwiger, Joe M. Kniss, Christof Rezk-Salama, and Daniel Weiskopf. *Real-Time Volume Graphics.* A K Peters, Ltd., Wellesley, Massachusetts, US, 2006.

[9] Filou Software GmbH. *Rhinoceros 3D - der Nurbs-Modeller unter Windows.* http://www.rhino3d.de. last checked: August 21, 2008.

[10] Georg Glaeser. *Geometrie und ihre Anwendungen in Kunst, Natur und Technik.* Spektrum Akademischer Verlag, München, 2005.

[11] Andreas Goebel. *Archimedes Geo3D.* http://www.raumgeometrie.de. last checked: August 21, 2008.

[12] Andreas Goebel. *Archimedes Geo3D - WissensSchule.de - SHOP.* http://www.wissensschule.de/schulshop/5/54/ArchimedesGeo3D.php. last checked: August 21, 2008.

[13] Markus Hohenwarter. *GeoGebra.* http://www.geogebra.org. last checked: August 21, 2008.

[14] Hannes Kaufmann. *Geometry Education with Augmented Reality.* PhD thesis, Vienna University of Technology, 2004.

[15] Benno Klotzek. *Einführung in die Differentialgeometrie.* Verlag Harri Deutsch, Frankfurt am Main, 1995.

[16] Ulrich H. Kortenkamp. *Cinderella : Die interaktive Geometrie-Software Cinderella.* http://cinderella.de. last checked: August 21, 2008.

[17] Ulrich H. Kortenkamp and Jürgen Richter-Gebert. *The interactive geometry software Cinderella: interactive geometry on computers.* Springer-Verlag, Berlin Heidelberg, 1999.

[18] Valérie Maquil. *Automatic Generation of Graphical User Interfaces in Studierstube.* Bachelor's thesis, Vienna University of Technology, 2004.

[19] McNeel. *Modeling tools for designers.* http://www.rhino3d.com. last checked: August 21, 2008.

[20] Roland Mechling. *Euklid DynaGeo Homepage.* http://www.dynageo.de. last checked: August 21, 2008.

[21] Parametric Technology Corporation. *PTC: Pro/ENGINEER.* http://www.ptc.com. last checked: August 21, 2008.

[22] William H. Press, William T. Vetterling, Saul A. Teukolsky, and Brian P. Flannery. *Numerical Recipes in C++. The Art of Scientific Computing. Second Edition.* Cambridge University Press, 1992.

[23] Gerhard Reitmayr. *On Software Design for Augmented Reality.* PhD thesis, Vienna University of Technology, 2004.

[24] Lewis F. Richardson and J. Arthur Gaunt. *The Deferred Approach to the Limit.* In *Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, volume 226, pages 299–361, 1927.

[25] Jürgen Richter-Gebert. *Cinderella Documentation.* http://doc.cinderella.de. last checked: August 21, 2008.

[26] Dieter Schmalstieg, Anton Fuhrmann, Gerd Hesina, Zsolt Szalavari, L. Miguel Encarnacao, Michael Gervautz, and Werner Purgathofer. *The Studierstube Augmented Reality Project.* In *Presence – Teleoperators and Virtual Environments, MIT Press*, volume 11, pages 33–54, 2002.

[27] Sophie and Pierre René de Cotret. *Cabri 3D User Manual.* Cabrilog SAS, 2005.

[28] Spatial Corporation. *Spatial: 3D ACIS Modeler.* http://www.spatial.com. last checked: August 21, 2008.

[29] Karl Strubecker. *Differentialgeometrie. Kurventheorie der Ebene und des Raumes.* Walter de Gruyter & Co., Berlin, 1964.

[30] Systems in Motion AS. *3D Graphics Development Tools – www.coin3d.org.* http://www.coin3d.org. last checked: August 21, 2008.

[31] Universität Bayreuth. *GEONExT: Startseite.* http://geonext.uni-bayreuth.de. last checked: August 21, 2008.

[32] Josie Wernecke. *The Inventor Mentor – Programming Object-Oriented 3D Graphics with Open Inventor, Release 2.* Addison-Wesley Longman Publishing Co., Inc., Boston, Massachusetts, USA, 1994.

[33] Josie Wernecke. *The Inventor Toolmaker – Extending Open Inventor, Release 2.* Addison-Wesley Longman Publishing Co., Inc., Boston, Massachusetts, USA, 1994.

[34] Walter Wunderlich. *Darstellende Geometrie II.* Bibliographisches Institut AG, Mannheim, 1967.