COMPUTER AIDED MANUFACTURING (CAM) DATA GENERATION FOR SOLID FREEFORM FABRICATON

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ONUR YARKINOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

SEPTEMBER 2007

Approval of the thesis:

COMPUTER AIDED MANUFACTURING (CAM) DATA GENERATION FOR SOLID FREEFORM FABRICATON

submitted by **ONUR YARKINOĞLU** in partial fulfillment of the requirements for the degree of **Master of Science in Mechanical Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____________________
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Kemal İder _____________________
Head of Department, **Mechanical Engineering**

Assist. Prof. Dr. Buğra Koku _____________________
Supervisor, **Mechanical Engineering Dept., METU**

Prof. Dr. Eres Söylemez _____________________
Co-Supervisor, **Mechanical Engineering Dept., METU**

Examining Committee Members:

Assist. Prof. Dr. Merve Erdal _____________________
Mechanical Engineering Dept., METU

Assist. Prof. Dr. Buğra Koku _____________________
Mechanical Engineering Dept., METU

Prof. Dr. Eres Söylemez _____________________
Mechanical Engineering Dept., METU

Assoc. Prof. Dr. Veysel Gazi _____________________
Electrical Engineering Dept., TOBB

Assist. Prof. Dr. Melik Dölen _____________________
Mechanical Engineering Dept., METU

Date: 04 / 09 / 2007

ABSTRACT

COMPUTER AIDED MANUFACTURING (CAM) DATA GENERATION FOR SOLID FREEFORM FABRICATON

YARKINOĞLU, Onur

M.S., Department of Mechanical Engineering
Supervisor : Assist. Prof. Dr. Buğra KOKU
Co-Supervisor : Prof. Dr. Eres SÖYLEMEZ

September 2007, 110 pages

Rapid prototyping (RP) is a set of fabrication technologies that are used to produce accurate parts directly from computer aided drawing (CAD) data. These technologies are unique in a way that they use an additive fabrication approach in which a three dimensional (3D) object is directly produced.

In this thesis study, a RP application with a modular architecture is designed and implemented to satisfy the possible requirements of future rapid prototyping studies. After a functional classification, the developed RP software is divided into View, RP and Slice Modules. In the RP module, the process parameter selection and optimal build orientation determination steps are carried out. In the Slice Module, slicing and tool path generation steps are performed. View Module is used to visualize the inputs and outputs of the RP software. To provide 3D visualization support for View Module, a fully independent, open for development, high level 3D modeling environment and graphics library called Graphics Framework is developed.

The resulting RP application is benchmarked with the RP software packages in the market according to their memory usage and process time. As a result of this benchmark, it is observed that the developed RP software has presented an equivalent performance with the other commercial RP applications and has proved its success.

Keywords: Solid Freeform Fabrication, Rapid Prototyping Software, Model Slicing, Path Planning, STL

ÖZ

KATI SERBEST FORMLU İNŞA YÖNTEMLERİ İÇİN BİLGİSAYAR DESTEKLİ ÜRETİM VERİSİ OLUŞTURULMASI

YARKINOĞLU, Onur

Yüksek Lisans, Makina Mühendisliği Bölümü
Tez Yöneticisi : Yrd. Doç. Dr. Buğra KOKU
Ortak Tez Yöneticisi : Prof. Dr. Eres SÖYLEMEZ

Eylül 2007, 110 sayfa

Hızlı prototipleme (HP), bilgisayar destekli tasarım (BDT) verisinden kesin doğrulukta parça üretilmesini sağlayan bir dizi üretim teknolojisine verilen addır. Bu teknolojiler üç boyutlu (3B) bir parçanın üretimi sırasında kullandıkları malzeme eklemeli üretim yaklaşımı açısından eşsizdirler.

Bu tez çalışmasında, gelecekte gerçekleştirilmesi muhtemel hızlı prototipleme çalışmalarında doğabilecek gereksinimleri gidermek amacı ile kullanılacak, modüler yapıya sahip bir HP yazılımı tasarlanmış ve üretilmiştir. Fonksiyonel bir sınıflama sonrasında HP yazılımı, Görüntüleme Modülü, HP Modülü ve Kesitleme Modülü olmak üzere üç ana parçaya bölünmüştür. HP modülünde işlem değişkenlerinin seçilmesi ve uygun üretim pozisyonlamasının gerçekleştirilmesi, Kesitleme Modülünde kesitleme ve üretim yollarının çıkarılması işlemleri gerçekleştirilmektedir. Görüntüleme modülü HP yazılımındaki girdi ve çıktıların görselleştirildiği modüldür. Görüntüleme Modülüne 3B desteği sağlamak amacı ile Grafik Destek Sistemi adında, tamamen bağımsız, geliştirilmeye açık, üst seviye bir 3B modelleme ortamı ve grafik kütüphanesi geliştirilmiştir.

Elde edilen HP yazılımı piyasadaki diğer yazılımlarla kıyaslanmıştır. Bu kıyaslamanın sonuçları hafıza kullanımı ve işlem sürelerine göre değerlendirilmiştir. Bu değerlendirmeler sonucunda, geliştirilen HP yazılımının, piyasadaki ticari HP yazılımlarına denk bir performans sergilediği ve başarısını kanıtladığı gözlemlenmiştir.

Anahtar Kelimeler: Katı Serbest Formlu Üretim, Hızlı Prototipleme Yazılımı, Model Kesitleme, Üretim Yolu Planlaması, STL

To my family

ACKNOWLEDGMENTS

I would like to thank my thesis advisor A. Buğra KOKU and my co-advisor Eres Söylemez for their support, collaboration, and guidance. I also would like to thank to my project colleague and my dear friend Faruk Yazıcıoğlu for his technical support and sensibility.

I sincerely thank to the people who worked with me for days and nights during this hard period of time. Thanks to all my friends Lütfi Taner Tunç, Süleyman Bideci, Umut Koçak, Emre Sezginalp and Arda Özgen for their understanding, support and friendship. Without their help and support, it would be difficult to overcome the faced problems throughout my MSc. study and thesis research.

Special thanks are given to my family, especially to my dear sister Oya Yarkınoğlu Gücük for their moral support and encouragement. Their sensibility gives me extra strength to finalize this study.

Additionally, this study was supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) as a part of a research project with project code 105M135.

TABLE OF CONTENTS

PLAGIARISM.. iii
ABSTRACT.. iv
ÖZ.. vi
DEDICATION.. viii
ACKNOWLEDGMENTS ix
TABLE OF CONTENTS.. x
LIST OF TABLES.. xiii
LIST OF FIGURES..xiv
CHAPTER
1 INTRODUCTION.. 1
1.1 Aim of the Thesis.. 2
1.2 An Overview of the Thesis.. 2
2 RAPID PROTOTYPING.. 4
2.1 Fundamentals of Rapid Prototyping.. 4
2.2 Historical Development of Rapid Prototyping.. 6
2.3 Advantages of Rapid Prototyping.. 7
2.4 Classification of Rapid Prototyping Techniques 10
2.4.1 Photopolymer Curing Techniques.. 10
2.4.1.1 Stereolithography.. 10
2.4.1.2 Solid Laser Plotter..11
2.4.1.3 Solid Ground Curing.. 11
2.4.1.4 Perfactory.. 12
2.4.1.5 Other Techniques.. 13
2.4.2 Material Deposition Techniques.. 13
2.4.2.1 Fused Deposition Modeling.. 13
2.4.2.2 Multi Jet Modeling.. 14
2.4.2.3 InVision Three Dimensional Printing.. 15
2.4.2.4 Precision Metal Deposition.. 15

2.4.2.5 Direct Metal Deposition..15
2.4.2.6 Other Techniques..16
2.4.3 Powder binding Techniques.. 16
2.4.3.1 Selective Laser Sintering.. 17
2.4.3.2 Three Dimensional Printing..17
2.4.3.3 Other Techniques..18
2.4.4 Object Lamination Techniques 18
2.4.4.1 Laminated Object Manufacturing..18
2.4.4.2 Solidimension Three Dimensional Printing................19
2.4.4.3 Other Techniques..19
2.5 Rapid Prototyping Software.. 18
2.6 STL Data Format..20
2.7 Process Planning in Rapid Prototyping Software.. 23
2.7.1 Selecting Process Parameters.. 24
2.7.2 Determining Optimal Build Orientation 25
2.7.3 Generating Support Structures..25
2.7.4 Slicing the Model..26
2.7.5 Planning Tool Path..29
2.8 A Survey on Rapid Prototyping Software Market.. 30

3 SOFTWARE DESIGN AND IMPLEMENTATION..34
3.1 Requirement Analysis..34
3.2 Software Design and Implementation Tools.. 35
3.2.1 Programming Language..35
3.2.2 3D Application Programming Interface 37
3.3 Software Architecture..38
3.4 Graphics Framework.. 39
3.4.1 Application Independency.. 41
3.4.2 Multi Screen Support..42
3.4.3 3D Modeling Environment..42
3.4.4 Graphics Library..44
3.5 Modules.. 45
3.5.1 View Module.. 45
3.5.1.1 Generic File Format Support.. 45
3.5.1.2 Part Organization and Selection..46

3.5.1.3 Undo/Redo Support……………………..……….….. 48
3.5.2 Rapid Prototyping Module……………………….……….…..48
3.5.2.1 Collision Detection …….……………………..…….. 49
3.5.2.2 Part Transformation …….………………………..….. 50
3.5.2.3 Direct Printing Interface………………………….…..57
3.5.3 Slice Module…………………………………...…….……57
3.5.3.1 Part Slicing …….…………………………………….……58
3.5.3.2 Tool Path Generation ………………………………….. 64

4 DISCUSSION AND CONCLUSION…………………………….……….….. 71
4.1 Performance Benchmarking……………………..……………….….. 71
4.2 Conclusion………………………………………….………..…... 74
4.3 Future Work and Further Improvements……………….……………. 75

REFERENCES…………………………………………………………………77

APPENDIX
A COMPARISON OF RAPID PROTOTYPING TECHNIQUES AND SYSTEMS ………………………………………….…………………… 80
B RAPID PROTOTYPING SOFTWARE MARKET SURVEY DETAILS……..... 82
B.1 Viscam Software Technical Specifications…………………………….. 82
B.2 Magics Software Technical Specifications…………………………….... 84
C REQUIREMENT ANALYSIS OF RP SOFTWARE…………………………… 88
D DETAILS OF INTERFACES AND STRUCTURAL CLASSES………….…. 90
E CASE STUDY…………………………………………...……………….102

LIST OF TABLES

TABLES

Table 2.1: RP software packages and systems developed by different manufacturers........ 31
Table 2.2: Functionality and price information of Viscam Software modules.................... 32
Table 2.3: Functionality and price information of Magics Software modules..................... 33

Table 4.1: Memory usage comparison of developed RP software, Magics and Viscam..... 72
Table 4.2: Process time comparison of developed RP software, Magics and Viscam 73

Table A.1: Comparison of Rapid Prototyping Techniques and Systems (Castle, 2007)..... 81

LIST OF FIGURES

FIGURES

Figure 1.1: Growth of Rapid Prototyping Systems (Wohlers, 2006)................................... 2

Figure 2.1: Three dimensional objects that are built layer by layer (Weiss, 1997) 5
Figure 2.2: Overview of the entire process of rapid prototyping... 5
Figure 2.3: A complex model produced by using a RP technology (Hart, 2005)................. 8
Figure 2.4: Area of use of rapid prototyping systems (Wohler, 2006) 9
Figure 2.5: Diagram of stereolithography machine with essential parts
(M2 Systems, 2007)... 11
Figure 2.6: Diagram of solid ground curing machine with essential parts (EFunda, 2007) 12
Figure 2.7: Diagram of multi jet modeling machine with essential parts (Erkut, 2007)...... 14
Figure 2.8: Diagram of direct metal deposition machine with essential parts
(Erkut, 2007)... 16
Figure 2.9: A faceted object with three faces and four vertices... 20
Figure 2.10: STL representation in ASCII format of the object shown in Figure 2.8 21
Figure 2.11: STL representation in binary format of the object shown in Figure 2.8.......... 22
Figure 2.12: Common topological problems: (a) flipped triangles, (b) missing triangles, .. 24
(c) - (d) invalid edge sharing and (e) misplaced triangles
(Venuvinod and Ma, 2004).. 24
Figure 2.13: Common geometric problems: (a) (b) (c) degenerate triangles,
(d) face overlapping (Venuvinod and Ma, 2004). 24
Figure 2.14: Support structures: (a) gusset support, (b) base support, (c) web support,
(d) column support, (e) zigzag support (used in FDM), (f) perimeter and hatch support (Venuvinod and Ma, 2004)... 26
Figure 2.15: Staircase effect in uniform slicing (Venuvinod and Ma, 2004)...................... 27
Figure 2.16: Marching algorithm for slicing (Gültekin, 2003).. 28
Figure 2.17: Raster tool path segment creations for a slice contour (Gültekin, 2003)......... 30

Figure 3.1: RP software modules and their sub parts. 39
Figure 3.2: Relations between RP Software, Graphics Framework and other APIs............ 40
Figure 3.3: Multi screen support provided by the Graphics Framework 42
Figure 3.4: Top and top front views of different 3D objects. 43
Figure 3.5: Different objects that are selected by using hierarchical tree structure. 47
Figure 3.6: Platform and workspace are displayed with a set of parts inside. 49
Figure 3.7: Former and latter positions of a translated part. 51
Figure 3.8: Former and latter orientations of a rotated part. 52
Figure 3.9: Former and latter sizes of a uniformly scaled part. 52
Figure 3.10: Former and latter sizes of a non-uniformly scaled part. 53
Figure 3.11: Slice preview of a set of parts along X direction 60
Figure 3.12: Slice preview of a set of parts along Y direction 61
Figure 3.13: Slice preview of a set of parts along Z direction 61
Figure 3.14: Slice preview of a different set of parts along Z direction 62
Figure 3.15: View of a set of parts after a production slicing operation. 64
Figure 3.16: View of a set of parts after a production slicing operation. 65
Figure 3.17: Hatching parameters and hatching direction of a contour. 66
Figure 3.18: View of a contour after a tool path generation operation. 67
Figure 3.19: View of a set of parts after a tool path generation operation. 68

Figure B.1: The modular architecture of Viscam Software (Marcam, 2007) 82
Figure B.2: Functionalities of View module of Viscam Software (Marcam, 2007)............ 83
Figure B.3: Functionalities of Mesh module of Viscam Software (Marcam, 2007)............ 83
Figure B.4: Functionalities of RP module of Viscam Software (Marcam, 2007)................ 84

Figure E.1: Creating a new project by using the main menu of the software 102
Figure E.2: Importing parts in STL format by using the main menu of the software........ 103
Figure E.3: Choosing the parts to be imported by using open file dialog.......................... 103
Figure E.4: Selecting a part from the graphical user interface to translate 104
Figure E.5: Moving the selected part by using the translation menu 104
Figure E.6: Selecting a part from the graphical user interface to scale.............................. 105
Figure E.7: Scaling the selected part by using the scaling menu 105
Figure E.8: Front view of the oriented parts 106
Figure E.9: Top view of the oriented parts 106
Figure E.10: Isometric view of the oriented parts.. 107

Figure E.11: Choosing the print preview functionality from the main menu 107
Figure E.12: Previewing the slices of the parts along X direction 108
Figure E.13: Previewing the slices of the parts along Y direction 108
Figure E.14: Previewing the slices of the parts along Z direction 109
Figure E.15: Choosing the print functionality from the main menu 109
Figure E.16: Printing the slices of the parts by using parameters 110
Figure E.17: Displaying parts after printing operation 110

CHAPTER 1

INTRODUCTION

Thanks to today's rapidly developing technology, nearly every day the world become familiar with a new product that makes life easier. However, from the manufacturers' perspective, day by day, the product development speed is increasing to attain this technology growth and it becomes nearly impossible to design new competitive products by following traditional design steps. This fact forces the companies to design better products by spending less money and less time with respect to their competitors. Consequently, the prototyping becomes more important since the visual and the functional verification of the design is performed by means of prototypes before making time and money consuming manufacturing investments. Therefore, in the industry, rapid prototyping (RP) technologies are widely used to produce better, faster and cheaper prototypes. In the last years, extensive use of RP technologies in the industry is resulted in a sudden growth in the market of RP technologies (Figure 1.1). This growth is also accelerated by the new coming RP systems and technologies developed by specialized research laboratories of different universities and companies. Today, the market is still growing so fast that "If the worldwide economy remains relatively strong, an estimated 15,000 3D printers are expected to sell annually by 2010" (Wohlers, 2006). This expectation makes RP an attractive field of research for universities all over the world and creates a competition between different universities and companies to develop better technologies to fulfill the future requirements of the market.

However, in Turkey, the situation is nearly contradictory to the world's trend. Unfortunately, only a few research studies in this topic are completed so far and most of these studies are only a collection or a revision of the currently available RP technologies (Erkut, 2007) and only a few research projects are in progress. Moreover, none of the universities in Turkey has a laboratory specialized in RP or any infrastructure that can be used in a comprehensive study. Another drawback is the lack of academic staff specialized in this field. Since the main parameter of industrial growth is the design and production of new and better products in a faster, cheaper and more competitive manner, this state is far beyond being sufficient for a developing country like Turkey. So this situation makes every new research in this field more and more important.

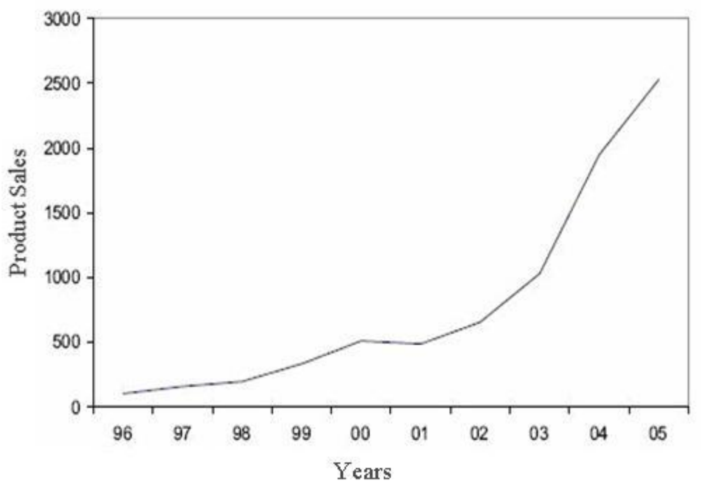


Figure 1.1: Growth of Rapid Prototyping Systems (Wohlers, 2006)

1.1 Aim of the Thesis

Considering the current state of RP technologies in Turkey, this thesis attempts to create (design and implement) a software framework which can satisfy the possible requirements of future RP studies and hence facilitate them. In this manner, it is desired to obtain an open for development and easy to use RP software package with all the necessary functionalities that are required to perform a RP process.

1.2 An Overview of the Thesis

The thesis study is presented in logically distinct four chapters, each of which contains some number of sections that are somehow related.

In this chapter, a brief introduction to the topic of RP is provided and the aim of the thesis is described.

The aim of the second chapter is to give the reader a general understanding of RP techniques, systems and related software packages. Therefore, the second chapter starts with a detailed explanation of RP concept. After this explanation the chapter continues with a general classification of RP systems. This classification is based on the production techniques used by different RP systems. At the end of the chapter basic concepts of a RP software is discussed and a survey on RP software market is provided.

The third chapter attempts to give the reader a general understanding of the architecture of the software and the methodology used during the design and implementation of this architecture. Therefore, in this chapter mainly design and implementation of the software is discussed. The chapter starts with a detailed requirement analysis of the software. After the requirement analysis, the tools used in the implementation and their alternatives are listed and the reasons of choice are justified. After this justification, core part of the software, which is called Graphics Framework, is discussed. Its application independency, multi-screen architecture, 3D visualization environment and graphics libraries are explained in detail. At the end of the chapter, the modules of the software and their properties are presented in detail by using screenshots taken from the software. The algorithms used in slicing and tool path generation operations are also discussed in this chapter.

In the last chapter, a general performance benchmark is presented. In the benchmarking process, the resultant software is compared with the commercial software packages listed in the second chapter according to their memory usage and process time. After this benchmark study, possible future work and further improvements are discussed and the chapter is completed.

CHAPTER 2

RAPID PROTOTYPING

This chapter provides a detailed survey on RP concepts to facilitate a better understanding of RP technologies, systems and software packages. The basic concepts and definitions given in this chapter facilitate the reader to easily understand some given information in forthcoming chapters.

2.1 Fundamentals of Rapid Prototyping

Rapid prototyping is a term which embraces a set of fabrication technologies that are used to produce accurate parts directly from computer aided drawing (CAD) data (Pham and Gault, 1998). These technologies are unique in a way that they use an additive fabrication approach in which a three dimensional (3D) object is produced by building new layers on top of other existing layers by incrementally adding materials (Figure 2.1) (Venuvinod and Ma, 2004). Because of the use of different terms and concepts in the definition of RP, these technologies are also known by different names like additive fabrication, three dimensional printing, layered manufacturing and solid freeform fabrication (Grenda, 2006).

The entire process of RP can be divided into three main parts that are performed in three different units. These units can be listed as computer aided design (CAD) software, RP software and RP machine. The prototyping process starts in the CAD software with the design of a model and ends in the machine with the production of the prototype (Figure 2.2).

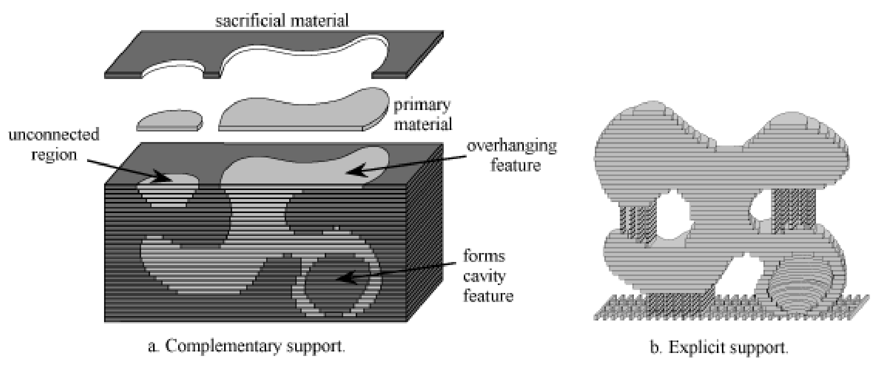


Figure 2.1: Three dimensional objects that are built layer by layer (Weiss, 1997)

The RP process starts with a design. When a design is completed, it is modeled in CAD software by using a solid or surface representation. From this model, triangulated data to be used in the production of the model in a RP system is generated. This data is exported from the CAD software in a worldwide standard RP file format called STL[1]. This exported file is imported by RP software to perform the next steps in the RP process.

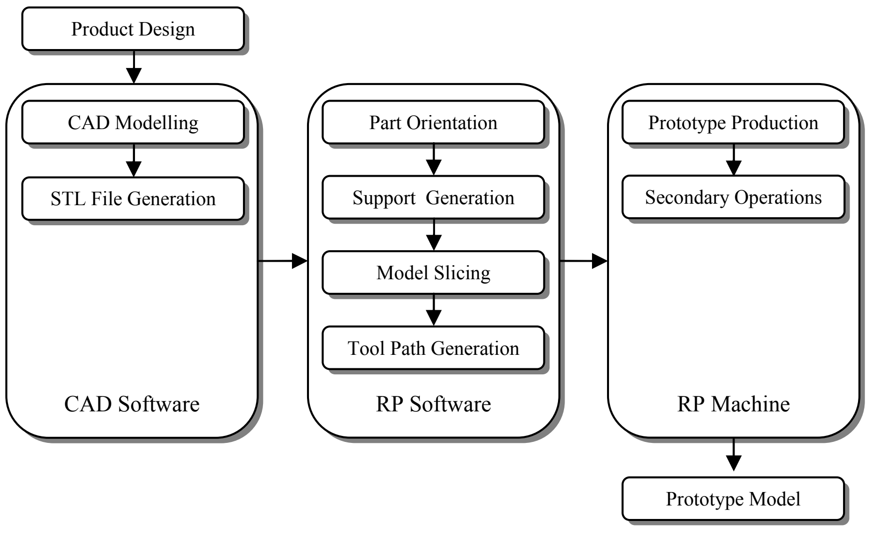


Figure 2.2: Overview of the entire process of rapid prototyping

1 STL is the native file format of the stereolithography CAD/CAM software created by 3D Systems Company.

Main purpose of RP software is to generate computer aided manufacturing (CAM) data that are used in the production phase. In the generation of CAM data, special process parameters like workspace dimensions, layer thickness, tool compensation, etc. are used. These parameters are set according to the properties of RP machine. In a standard RP process, the process inside the software starts with the import of a 3D CAD file. When the file is imported, the part is oriented inside the workspace by considering different parameters like surface quality, production time, production cost, etc. In the next step, support structures are generated according to the orientation of the part. The formed workspace, the model and the support structures, are sliced into layers. After the slicing operation, the generation of CAM code is completed with the generation of the tool paths.

When CAM code is generated, the physical side of the production is started. The code is sent to the machine by the software and the machine produces the part by using a particular RP technique. In most of the RP systems the production process is fully automated, so no technician is needed during the fabrication of the part. In some systems, secondary operations can be needed after the production. Some of these secondary operations are manually done by a technician and some of them are performed automatically in different units of the RP system.

2.2 Historical Development of Rapid Prototyping

In the presentation of historical development of RP it is necessary to be familiar with the concept of prototyping and the meaning of word “prototype”.

Prototype used as a word in a product development and manufacturing cycle means “an approximation of a product (or system) or its components in some form for a definite purpose in its implementation” (Hornby and Wehnmeier, 2000). This is a very general description which not only contains meaning of being physical but also covers all kinds of prototyping issues like mathematical models, pen sketches or any virtual models of a product. Actually, as it is understood in our daily lives prototyping is a general process of realizing a design. From this point of view historically development of prototyping concept can be classified in three phases like manual prototyping, soft or virtual prototyping and RP (Chua, 2003).

When the age of mechanization had started in 1770, a requirement to use prototypes has emerged. Until the mid of the 20th century, handmade models were the only way to assess the form, fitness and functionality of a design before making a significant investment in tooling (Pham and Gault, 1998). Pen sketches were also used to visualize designs without fabricating a physical model. During this early phase, prototypes manufactured manually were not an accurate copy of the real product. Although, the prototypes were not very sophisticated, their fabrication took weeks or months depending on the skill of the craftsmen, level of complexity and representativeness (Chua, 2003).

With the usage of computers in the design and manufacturing of products in the mid 20th century, the prototyping entered its second phase. During this phase, computers and mathematical models were used to visualize the models and to analyze their physical attributes and properties. With increasing computation power it has become possible to design more complex products by spending less effort. Use of computers has also changed the way of production with the emergence of high precision computer numerical control (CNC) machines. This rapid change in technology has opened a new era in prototyping. Although, visual aids and mathematical models can be used to realize the product, still there has been a need to physically fabricate the prototypes of new products. By the introduction of CNC machines and computer technology it has become possible to produce more complex (about twice the complexity as before (Metelnick, 1991)) and more accurate models with a disadvantage of extended production time (weeks or months for complex models) and high production cost (Chua, 2003).

In 1988, with the release of the first commercial RP system, the next phase of prototyping has started. The invention of RP technology makes it possible to produce relatively three times more complex models compared to the parts produced in 1970s (Figure 2.3). Also the time and money required to produce these models has tremendously decreased. Required time to build a complex model has started to be expressed by months or even days (Chua, 2003). Today, 57 different companies from 11 different countries are producing RP systems by using 65 different RP techniques and nearly 700 service providers are giving prototyping service (Erkut, 2007).

2.3 Advantages of Rapid Prototyping

RP systems have a large area of use compared to other manufacturing processes (Figure 2.4). Today’s RP systems can directly be used to produce functional parts in small quantities. However, usually the accuracy and surface finish of these parts are not as good as the machined one. On the other hand, in some advanced RP systems near tooling quality parts can be produced as final products. Also in some other RP systems material qualities and physical properties can be improved by using different secondary operations. More fundamentally, when the time and the cost to produce a complex part by using a RP system is considered, these deficiencies are considerably affordable compared to other machining techniques. Therefore, with the increasing surface quality and improved material properties, the RP systems are the most significant competitor of conventional production systems.

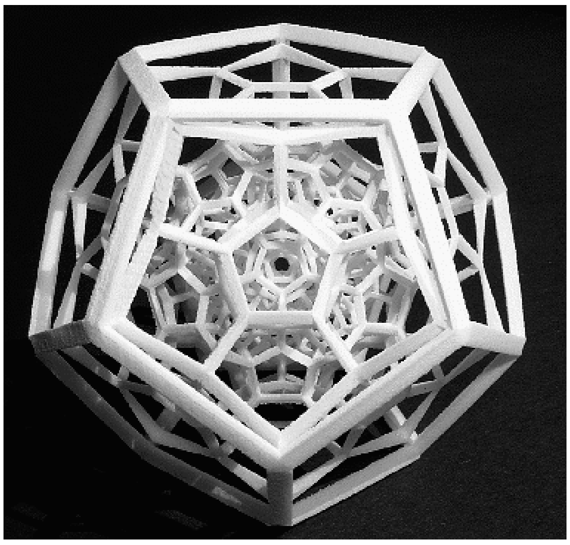


Figure 2.3: A complex model produced by using a RP technology (Hart, 2005)

When it is considered that the products in the market have increased in complexity, the companies, which need experimentation with physical objects of any complexity, are benefited more from the RP systems (Metelnick, 1991). During the development and design process, sometimes producing parts near to the final product or products that can be used in functional testing is important, however the time needed to manufacture a single final product is enormous. On the other hand, a RP system offers the companies a great opportunity to produce these products in a relatively short period of time with a low cost (Chua, 2003).

RP systems used in the production of a final product give designers the opportunity of designing more complex parts, so more functional or aesthetic forms can be used in the design. Also, without any manufacturing limitation, part designs can be optimized and the number of parts can be decreased by combining features in a single part. Decrease in number of parts results in a decrease in analysis time, assembly time, assembly difficulties and number of fastening parts (Chua, 2003). When RP systems are used in final product manufacturing, there are fewer constraints in the design of a part. The parts that cannot be set up for machining and the structures with production difficulties like thin walls, holes outside the limits of manufacturing tools, empty shapes, which causes high material removal and high production cost, can be designed without any manufacturing limitation. Also parts can be optimized for their strength / weight ratios without regarding the machining cost. Another advantage is the minimization of the time consumed during the discussions and evaluations that is made on the manufacturing possibilities (Chua, 2003).

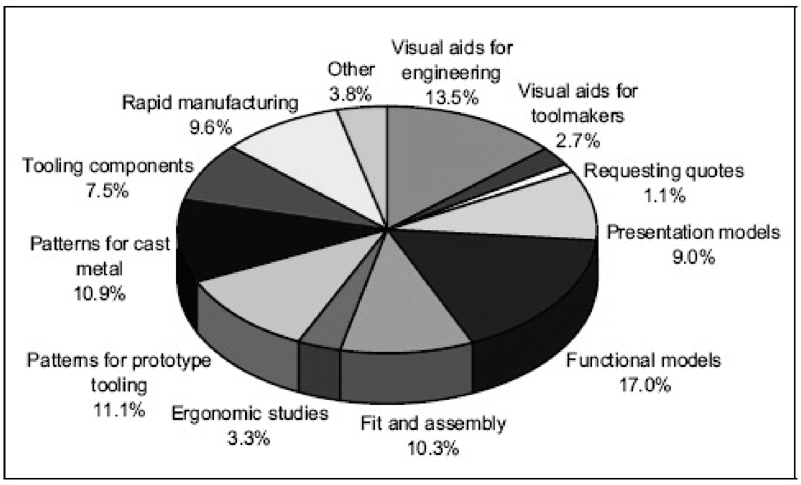


Figure 2.4: Area of use of rapid prototyping systems (Wohler, 2006)

RP systems producing final products causes a reduction in labor cost since stock of different tools and many special purpose machines is not needed. Trained personnel cost is reduced as well, since setting up of machinery and CNC programming is eliminated. As an indirect effect, less training cost is required because less qualified operator is needed. Also there is no need to stock materials in different sizes; as a result, the material costs are reduced. The change in a dimension of a part can be applied to the production without rewriting all the CNC code; hence an enormous amount of time is gained (Chua, 2003).

All these factors show their effect as more aesthetic and functional products with lower manufacturing costs; this way, the product prices decrease while the user satisfaction and competition power of companies increase.

2.4 Classification of Rapid Prototyping Techniques

RP technologies have shown a rapid growth since the release of the first commercial system in mid 80s. After this triggering event various systems using different techniques have been developed and commercialized all over the world. Today, worldwide 57 different companies are producing RP systems by using 65 different RP techniques and still new technologies are emerging (Appendix A) (Erkut, 2007). All these systems used worldwide can be grouped under four main classes based on the characteristics of production techniques. These classes can be listed as photopolymer techniques, deposition techniques, powder binding techniques and lamination techniques. In the following sections different RP techniques are presented in detail. The information used in these sections is taken from the technical report of Erkut (2007).

2.4.1 Photopolymer Curing Techniques

In photopolymer curing techniques, a photosensitive material layer is laid out to the production platform and the required parts of this layer are cured by using a light source. When this photosensitive material is cured, it solidifies and bonds with the previous layer(s). This procedure is repeated until all the layers of the three dimensional model of the prototype is completed. This curing operation can be performed by using two different approaches. In the first approach required parts of a layer is scanned point by point by using a laser beam. In the second approach, a mask that is covering the unnecessary parts of the layer is used and each layer is cured at a single time by using a single UV lamp. In the forthcoming subsections, systems using photopolymer curing techniques are described in detail.

2.4.1.1 Stereolithography

In the stereolithography (SLA) technique, liquid photopolymer accumulated in a tank is solidified layer by layer by using a laser beam that is positioned on top of the production platform (Figure 2.5). When curing of a layer is completed, the platform is plunged into the tank by a layer thickness and the curing operation of the next layer is started. After all the layers are formed, the platform is raised to the liquid level. Throughout the manufacturing process, some special structures called support structures are also produced with the prototype. These support structures are used for different purposes and they are manually removed from the prototype after the production process. The details of these structures are described in the forthcoming sections of this chapter. Mostly, after the first curing operation, the physical properties of the manufactured model are not ideal, so the model is kept in a special UV lamp oven for a period of time to improve its physical properties.

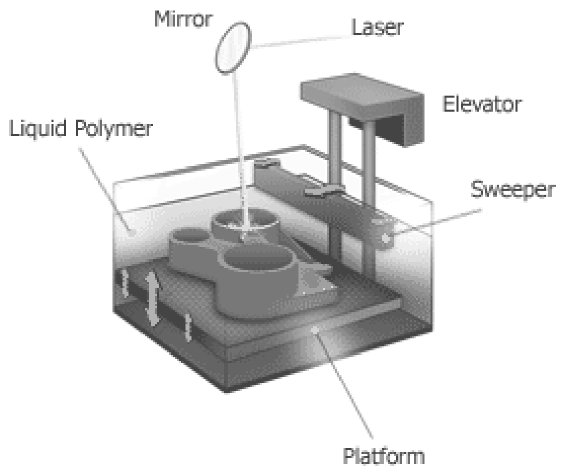


Figure 2.5: Diagram of stereolithography machine with essential parts (M2 Systems, 2007)

2.4.1.2 Solid Laser Plotter

In the solid laser plotter (SLP) technique, liquid photopolymer accumulated in a transparent tank is solidified under a production platform by using a laser beam that is positioned on top of the liquid tank. During the fabrication, the platform is positioned outside the liquid level and completed layers are adhered to the bottom of the platform. When curing of a layer is completed, the platform is moved upward by a layer thickness and the curing operation of the next layer is started. So, model is fabricated layer by layer in an inverse manner when compared to SLA.

2.4.1.3 Solid Ground Curing

In the solid ground curing (SGC) technique, liquid photopolymer is laid out to the production platform as a thin layer. Then a mask of the layer is formed on a glass with electrophotography technique by using photocopy toner. When the mask is formed, a single UV lamp cures the photopolymer to form a solid layer (Figure 2.6). Curing is performed by using a high density UV lamp, so a second curing process used in SLA is not needed. After the curing operation is completed, remaining liquid photopolymer is pumped out to be used in the next cycle and the empty parts are filled with liquid wax. Support wax is solidified by applying pressure with a water cooled metal sheet. For the next layer the surface of the existing layer is shaven by a blade. This process cycle is repeated until the fabrication of the model is completed.

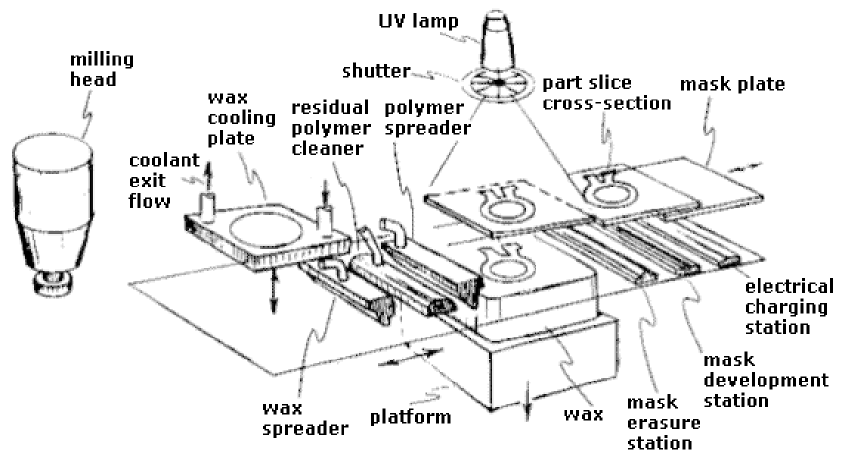


Figure 2.6: Diagram of solid ground curing machine with essential parts (EFunda, 2007)

2.4.1.4 Perfactory

In the perfactory technique, the light diffused from the UV lamp is directed to the photopolymer layer through a masking projection system that is driven by digital light processing (DLP) technology. In DLP, the direction of the light is controlled by thousands of micro mirrors positioned on a silicon chip. The state of these mirrors is controlled by electric signals in two stages (on / off). By changing the states of these mirrors, mask of a layer is formed and the fabrication of each layer is completed like it is performed in the SLP technique.

2.4.1.5 Other Techniques

Some other techniques that are using photopolymer curing approach are also available. However, these techniques have a more limited area of use compared to the stereolithography, the solid laser plotter, the perfactory and the solid ground curing techniques. In some applications the masking methods or the physical properties of the photopolymer material may show differences from the popular photopolymer curing techniques. Use of an LCD screen as a mask or use of a solid photopolymer material can be examples of these applications.

2.4.2 Material Deposition Techniques

In material deposition techniques, production material in a liquid or a cemented state is sprayed or plastered to the desired points of a layer in a controlled fashion. In this group of methods solidification of the material usually occurs by a state change of the material from a liquid to a solid form or by a chemical reaction. The material is sprayed or plastered to the production platform by using single or multi nozzle head systems. During the same production process, different materials can be deposited for production or supporting purposes by using different nozzles. As a result of this, it is possible to produce multi-material complex models by using different materials with different physical properties. In addition, assemblies composed of different parts with different materials can also be produced during a single build operation.

2.4.2.1 Fused Deposition Modeling

In the fused deposition modeling (FDM) technique, production material that is heated and held under the melting temperature is extruded through a thin nozzle and plastered to the production platform to form the layers. When a layer is produced, the platform moves downward and the deposition of the next layer starts. This cycle is repeated until the fabrication of the model is completed. The production material in the form of filament is fed to the nozzle by using a position controlled feeding system. During the production process, two different types of materials can be used as support structure. First type of support structure is produced by using an easy to break type plastic material, so the supports are manually removed from the model. The second type of support structure is produced by using a special material that dissolves in water, so the removal process is automatically handled in an external unit. In this production technique, different materials like acrylonitrile butadiene styrene (ABS), polyamide, casting wax and high temperature resistant engineering plastics can be used as production material.

2.4.2.2 Multi Jet Modeling

In the multi jet modeling (MJM) technique, melted material is sprayed to the platform by using an inkjet printer head (Figure 2.7). This inkjet printer head contains 352 micro nozzles that are driven by a piezoelectric actuation system. In production, printing the head moves in the direction of X and the platform moves in the direction of Y and Z. Owing to multi nozzle head structure, the printing head can cover Y direction in eight passes. In each layer, the order of passes in Y direction is randomized to prevent an error accumulation that can be originated from a material blockage in a nozzle. After production of each layer, a roller system passes along the layer to provide the uniform layer thickness. Support structures are produced by using the same material with the model and removed manually after the model is hold under the temperature of 10 degree in an external unit. In this production technique a paraffin based polymer is used as production material.

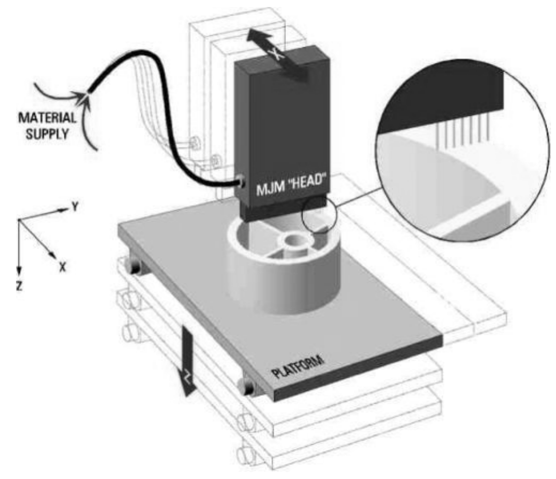


Figure 2.7: Diagram of multi jet modeling machine with essential parts (Erkut, 2007)

2.4.2.3 InVision Three Dimensional Printing

In the InVision 3D printing technique, an acrylic photopolymer material in gel form is heated until its melting point and sprayed to the platform by using an inkjet printer head similar to the one used in MJM. When a layer is formed in solid form it is cured by using an UV lamp. By repeating this cycle the fabrication of the model is completed. During production process, a special wax is used as a support material. Support structures are removed after the fabrication by heating the model. When compared to MJM, this technique can produce more durable materials at higher speed and better accuracy. In the market there are different implementations of this method.

2.4.2.4 Precision Metal Deposition

In the precision metal deposition (PMD) technique, a metal wire is melted and deposited to a platform by using a laser beam. During the fabrication, an inert gas is exerted from the wire feeding system to the deposition point to prevent corrosion. In PMD, compared to other methods, a lesser amount of melted material is produced and smaller amount of thermal stress is generated. So, parts with better metallurgical properties and a reduced amount of distortion can be produced. This makes PMD a superior example of use of RP systems in direct manufacturing of parts and molds. PMD is also used in aerospace industry to repair the expensive and complex titanium and stainless steel parts and molds.

2.4.2.5 Direct Metal Deposition

In the direct metal deposition (DMD) technique, metal powder is melted and deposited by using a carbon dioxide laser beam (Figure 2.8). The metal powder is fed by a funnel shaped feeding system that is positioned around the laser beam. The temperature of the melted powder and the crystal structure of the metal are controlled during the production. By using this technique, multi material parts can be produced by using different materials in the same fabrication process. DMD is an example of use of RP systems in direct manufacturing of parts and molds. DMD is also used to repair damaged or corroded parts and molds.

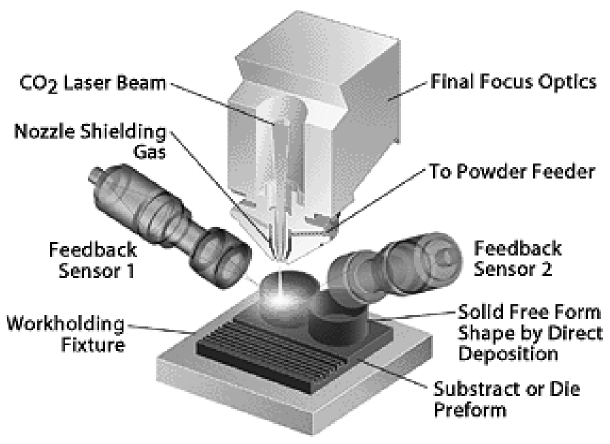


Figure 2.8: Diagram of direct metal deposition machine with essential parts (Erkut, 2007).

2.4.2.6 Other Techniques

Some other techniques that are using material deposition approach are also available. However, these techniques have more limited area of use compared to the ones introduced above. Some of these techniques are liquid metal jet printing, ballistic particle manufacturing, Pligraphy, electrochemical fabrication, micro-droplet fabrication and bio-plotter.

2.4.3 Powder Binding Techniques

In the powder binding techniques, material in powder form is laid out to the platform and is heated or glued in necessary sections of the layer by using a laser or electron beam or an inkjet type glue gun. Powders that are not bonded are used as support structure throughout the production. In this type of techniques secondary operations like removing unused powder and cleaning fabricated model is required. In model fabrication plastic, metal, ceramic or multi-material powders can be used as production material.

2.4.3.1 Selective Laser Sintering

In the selective laser sintering (SLS) technique, a heat fusible powder is laid out to the production platform as a thin layer and melted selectively by using a laser beam. The melted powder particles are bound together and form a solid layer. Then the platform moves downward and the production of the other layers are done by following the same steps. In SLS, the required powder is fed to the system by using another platform which is moving in the opposite direction of production platform. When the production platform moves downward, the feeding platform moves upward and feeds the required powder. The fed powder is laid out as a thin layer by using a roller. The production chamber is held under the melting temperature of material to be able to bind the powders faster and easier. In model fabrication, plastic, metal, ceramic or multi-material powders can be used. By using these materials, functional final products and ceramics molds can be produced. The most important disadvantage of this technique is the porous structure of the fabricated parts. To resolve this issue mostly a secondary heat treatment process is required to improve the physical properties of the model. Also, other secondary operations like removing unused powder and cleaning the fabricated model are necessary after production.

2.4.3.2 Three Dimensional Printing

In the three dimensional printing (3DP) technique, the material in powder form is laid to the production platform by using a similar feeding system that is used in SLS. But different from SLS, the powder is bonded in the required areas by using a multi nozzle glue gun. After a layer is finished the production platform is moved downward and the powder of the next layer is laid. This process cycle is repeated until all the layers of the model are completed. The unbound powder particles are used as a support structure throughout the production phase and removed for recycling by using a vacuum cleaner at the end of the production. In this technique, secondary operations like sintering and a different material saturation can be applied to improve the physical properties of the model. The advantages of this technique can be listed as high production speed and low production cost. 3DP systems are the fastest prototyping systems in the market. On the other hand, the products fabricated with 3DP have some deficiencies like low surface quality, low dimensional accuracy and being fragile. 3DP systems are mostly used in the visualization of conceptual designs or in the production of tooling rather than to be used in final products. Different parts like casting molds, filters and steel-bronze alloy materials can be produced by using this technique.

2.4.3.3 Other Techniques

Some other techniques that are using powder binding approach are also available. But, these techniques have a more limited area of use compared to selective laser sintering and three dimensional printing techniques. These techniques can be listed as sintering technique of EOS Company (EOSINT), selective laser melting, electron beam melting, selective mask sintering, laser-forming and lasercusing combined processing.

2.4.4 Object Lamination Techniques

In object lamination techniques, thin solid materials are used in production. The layers are formed by cutting the production material. After the cutting operation, the formed layer is pasted to the other layers. By repeating this cycle the model is formed layer by layer. In these systems paper, plastic, foam, metal sheets and some different materials saturated with ceramics and metals are used in production.

2.4.4.1 Laminated Object Manufacturing

In the laminated object manufacturing (LOM) technique, a special paper that is saturated with polymer adhesive is laid out to a platform by using a cylindrical feeding system. This material is pasted to the previous layer by using a hot roller. After positioning, the profile of the model section is cut on the material by using a laser cutter. Scrap parts are hatched to facilitate the removal process. These scrap parts are used as support structure throughout the production. This process cycle is repeated until the model is completed. A considerable amount of smoke is generated during the cutting operation. So a ventilation system and an air filter are used to remove the polluted air. The supports are removed manually. This process can be painful and time consuming in complex models. Also some secondary operations like retouching and sanding can be required after production. The fabricated models have similar physical properties with wooden models.

2.4.4.2 Solidimension Three Dimensional Printing

In the Solidimension 3DP technique, a PVC material in a sheet form is laid out to the platform and an adhesive material is applied onto the layer by using a roller system. Then the profile of the model section is cut from the material by using a blade. The uncut parts are used as support structures. A special anti-glue chemical is sprayed to these parts to facilitate the structure removing process. Compared to LOM more durable parts can be manufactured by using this technique.

2.4.4.3 Other Techniques

Some other techniques that are using object lamination approach are also available. But, these techniques have a more limited area of use compared to the laminated object manufacturing, the Solidimension 3D printing and the trusurf techniques. These techniques can be listed as trusurf, shape adhesive hot press, CAM-LEM, offset fabbing, stratoconception and customLAM.

2.5 Rapid Prototyping Software

RP systems can be studied in three sub groups which are software, hardware and the technique. The main concern of this thesis study is the software part of an RP process.

In the previous section, detailed information about the RP techniques is given. There are two reasons for this. First is the importance of RP techniques in the concept of RP. Therefore, a RP technology presentation without a detailed RP techniques survey is considered incomplete. The second reason is the effect of the RP techniques on the process planning part of the RP software packages. Consequently, the basic principles of production techniques have a great effect on the way how the software works.

On the other hand, the RP hardware is strictly dependant on the used RP technique. Each machine is designed to use a specific technique so a detailed presentation in RP techniques also reflects the details of RP hardware. Also, the effects of hardware on the RP software is limited; only some production parameters in the process planning part are set by the hardware, thus the technical details of hardware are considered as a different issue which is outside the boundaries of this study. To be able to use the software with different types of hardware, all of these parameters are programmed in a parametric fashion, so a detailed study on technical details of hardware is not needed and therefore not presented.

In the forthcoming sections of this chapter, the concern is only on the details of RP software packages. The required information about the STL data format, RP process planning steps and RP software market is presented.

2.6 STL Data Format

STL is a file format specification for importing CAD models from different design programs to RP systems. The specification was initially developed by 3D Systems Inc. for their stereolithography systems. Now, STL is the standard RP data format for all RP systems worldwide (Venuvinod and Ma, 2004).

STL file format mainly uses faceted objects to represent a model (Figure 2.9). To perform this representation, initially the surfaces of the model are reconstructed by using triangles. After the completion of re-construction, the data is stored in a file by using one of the two different STL representations. These representations can be listed as ASCII and binary. Both of these representations provide a list of triangles that is forming the model. Triangles are represented by a normal vector and coordinates of three edges. Each of these values is stored in floating point accuracy (Venuvinod and Ma, 2004).

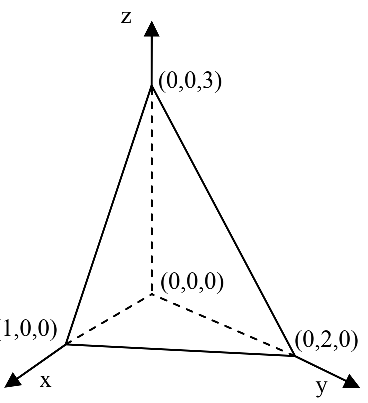


Figure 2.9: A faceted object with three faces and four vertices.

In ASCII format the data is stored in text format by using some keywords and values (Figure 2.10). This representation can be easily read and checked since all the values required to define a triangle is listed with its attribute name and corresponding value. However, an ASCII format representation results in a large file size since an ASCII value of each character is used to store the data (Venuvinod and Ma, 2004).

```
solid irregular tetrahedron
        facet normal    -1.0    0.0     0.0
                outer loop
                        vertex  0.0     0.0     0.0
                        vertex  0.0     0.0     3.0
                        vertex  0.0     2.0     0.0
                end loop
        endfacet
        facet normal    0.0     -1.0    0.0
                outer loop
                        vertex  0.0     0.0     0.0
                        vertex  1.0     0.0     0.0
                        vertex  0.0     0.0     3.0
                end loop
        endfacet
        facet normal    0.0     0.0     -1.0
                outer loop
                        vertex  0.0     0.0     0.0
                        vertex  0.0     2.0     0.0
                        vertex  1.0     0.0     0.0
                end loop
        endfacet
        facet normal    0.85714286      0.42857143      0.28571429
                outer loop
                        vertex  1.0     0.0     0.0
                        vertex  0.0     2.0     0.0
                        vertex  0.0     0.0     3.0
                end loop
        endfacet
endsolid irregular tetrahedron
```

Figure 2.10: STL representation in ASCII format of the object shown in Figure 2.8

On the other hand, in binary format only the required data is stored in a predefined order by using the byte representations of the values (Figure 2.11) which results in 85% file size reduction in a typical file. The reduction in file size also affects the read, write and transfer times. Hence, mostly the binary STL representation is used in the market

(Venuvinod and Ma, 2004). Despite its simplicity, the STL file format has some inherent problems that have to be dealt with when processing these files.

```
 (Start of File)
84 bytes  –  header record
          80 bytes  –  unformatted general information such as file
                         name, part name and comments
          4 bytes   –  number of facet records each facet record
                         defines one triangle
50 bytes  –  first facet record
          12 bytes  –  facet normal vector
                         4 bytes  –  i coordinate
                         4 bytes  –  j coordinate
                         4 bytes  –  k coordinate
          12 bytes  –  first vertex
                         4 bytes  –  i coordinate
                         4 bytes  –  j coordinate
                         4 bytes  –  k coordinate
          12 bytes  –  second vertex
                         4 bytes  –  i coordinate
                         4 bytes  –  j coordinate
                         4 bytes  –  k coordinate
          12 bytes  –  third vertex
                         4 bytes  –  i coordinate
                         4 bytes  –  j coordinate
                         4 bytes  –  k coordinate
          2 bytes   –  optional facet attributes like color
50 bytes  –  second facet record
          :
          :
          :
50 bytes  –  the last facet record
(End of File)
```

Figure 2.11: STL representation in binary format of the object shown in Figure 2.8

The most obvious problem of the STL file format is the large file size. When complex models are converted into STL format, huge files are created which makes file transfer more difficult. Also the time required to read or write a file increases with the increasing file size. When a single surface is represented by too many triangles, the memory usage also increases since all the required data is held in the memory during the data manipulation. This problem comes from the fact that the STL file format holds redundant data to represent a model. In a closed surface, the outer and inner surfaces can be found from the given data, so the normal vector information can also be calculated from the given vertex list. On the other hand, for a typical triangle mesh each mesh is shared by six neighboring triangles which implies that the same vertex data is hold for six times. So, when the vertices are ordered properly, the information of the shared vertices can also be found from the given triangle list. So it is entirely unnecessary to hold the normal vector information and the shared vertices information in STL files (Venuvinod and Ma, 2004).

The STL file format also has some deficiency in the accuracy of the representation of the model. Since an approximation is done during the conversion of a CAD model to a faceted model, the precision of some surfaces and some details can be lost. If the surface is planar this conversion is performed accurately, but if the surface is curved then the accuracy of the conversion is controlled by the number of triangles used in the approximation. Also the level of details can be affected from the conversion tolerances. These two error sources can result in inappropriate model or low surface quality in RP.

STL data can also contain some topological and geometric problems which are originated from surface approximation. The topological problems can be listed as flipped triangles, missing triangles, invalid sharing and misplaced triangles (Figure 2.12). These problems can be solved through face flipping, local re-triangulation and edge, triangle and vertex reconnection, insertion and deletion. The geometric problems can be listed as the degenerate triangles and face overlapping (Figure 2.13). Such problems can be solved by deleting degenerate triangles or by vertex repositioning (Venuvinod and Ma, 2004). In today’s RP systems these problems are handled before the production process by using RP software packages or individual STL file repair applications.

2.7 Process Planning in Rapid Prototyping Software

The operations performed in RP software packages can be classified and analyzed in five groups. These five groups can be listed as selecting process parameters, determining optimal build orientation, slicing model, planning tool path and generating support structures (Marsan, 1998). Although all of these steps can be considered as necessary in a standard RP process, in some cases one or more of these steps can show differences or even be neglected. As an example, in a process planning of a RP system using the LOM technique, the support generation step can be neglected depending on the system.

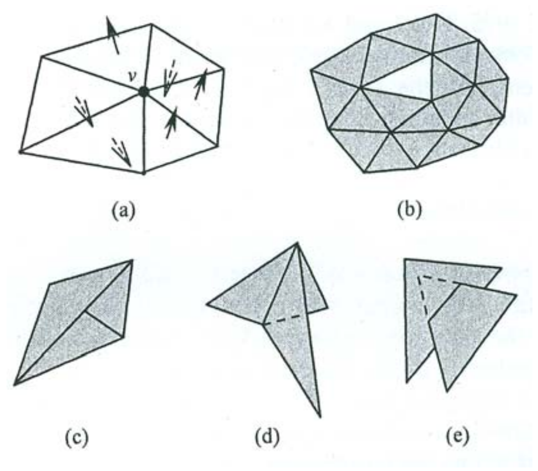


Figure 2.12: Common topological problems: (a) flipped triangles, (b) missing triangles, (c) - (d) invalid edge sharing and (e) misplaced triangles (Venuvinod and Ma, 2004).

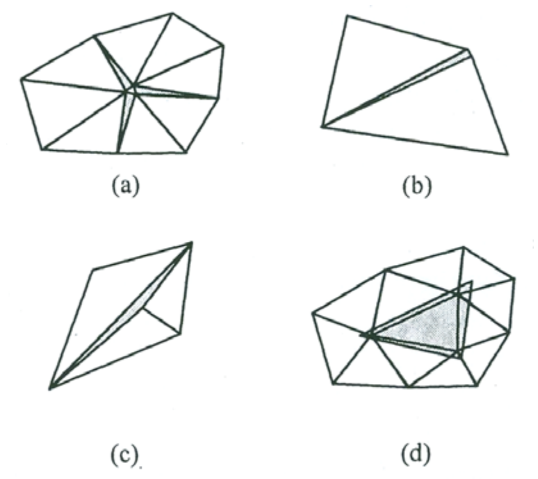


Figure 2.13: Common geometric problems: (a) (b) (c) degenerate triangles, (d) face overlapping (Venuvinod and Ma, 2004).

2.7.1 Selecting Process Parameters

The parameters that are used by the software to perform a RP process can be referred to as process parameters. These parameters can show differences from one technique to another, but in all techniques the proper selection of process parameters has a great effect on the production time, production quality and production cost. Workspace dimensions, slice thickness, production direction, extrusion head radius, material flow rate, head speed, acceleration and deceleration can be the examples of some process parameters that can be required in a RP process.

2.7.2 Determining Optimal Build Orientation

Orientation of a part on the production platform has a decisive affect on the surface quality, production time, production cost, material properties in different directions and the amount of support structures needed. Originally, build orientation of parts are decided by the operator by placing the parts on a production platform by rotating and translating parts in different directions. However, with too many parts in a single production process, it is very difficult to place maximum number of parts by finding the best possible part orientations. Therefore, various optimization methods are employed to find the optimal build orientation of parts. On the other hand, these software packages still give the operator the opportunity to pick and place parts manually. More detailed information on automatic part placement methods can be found in the study of Marsan, et al (Marsan, 1998).

2.7.3 Generating Support Structures

Different types of "support structures are used for variety of reasons (Figure 2.14), including supporting overhangs, maintaining stability of the part, supporting large flat walls, preventing excessive shrinkage, supporting components initially disconnected from the main part and supporting slanted walls" (Marsan, 1998). On the other hand, these problems can also be solved or at least minimized by finding the optimal build orientation of a problematic part. So generating support structures and finding the optimal build orientations of parts are related problems that must be considered as one and solved together. Also in some techniques like SLS, LOM and 3DP the support structures are not needed since the excessive production material are used as support structures. However quite a number of RP techniques still need support structures. Therefore, deciding when, where and which kind of a support structure is needed during manufacturing are the critical questions to be answered. To answer this question, different types of rules and automatic support structure generation algorithms have been developed for different techniques. More detailed information on these rules and algorithms can be found in the study of Marsan, et al (Marsan, 1998).

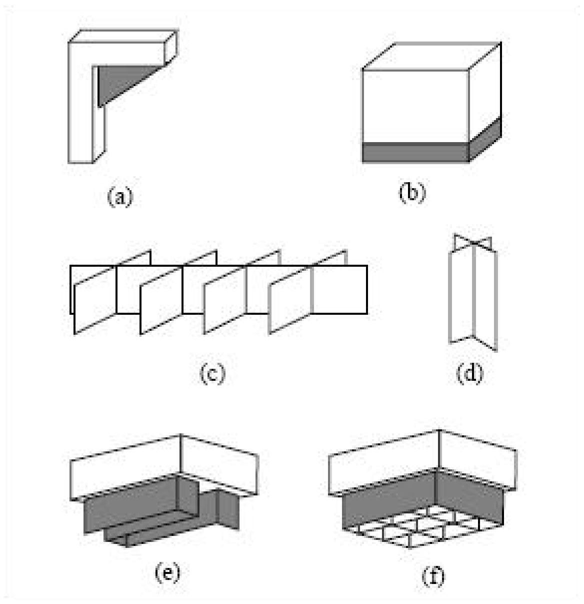


Figure 2.14: Support structures: (a) gusset support, (b) base support, (c) web support, (d) column support, (e) zigzag support (used in FDM), (f) perimeter and hatch support (Venuvinod and Ma, 2004).

2.7.4 Slicing the Model

For all RP production techniques, slicing operation is the most important process planning step. In this step, intersection of a model with a set of parallel planes is computed and the contour curve of each slice at a particular height is formed (Marsan, 1998). Originally, the parallel cutting planes are positioned perpendicular to building direction with uniform distance between the planes. Because of this uniform order this approach is known as uniform slicing.

The major problem of uniform slicing is the staircase effect (Figure 2.15). Staircase effect can be a great problem for near-vertical surfaces since it reduces the surface quality. In addition, some part details and dimensions smaller than the slice thickness can be lost during slicing operation.

In most RP systems the material is deposited in only one direction. Consequently, the stairway effect can only be minimized by changing the orientation of the part and using a smaller slice thickness value. However, minimizing slice thickness value has some counter effects like increase in production time and cost. Alternatively, a different slicing approach called adaptive slicing can be used as a more general solution for the problem. In adaptive slicing, different slice thickness values are used during the slicing operation. So, to minimize the stairway effect, the sections with near-vertical planes or small dimensioned details can be sliced by using smaller thickness values. On the other hand, the production time and cost can be minimized by using higher thickness values in the other sections of the model.

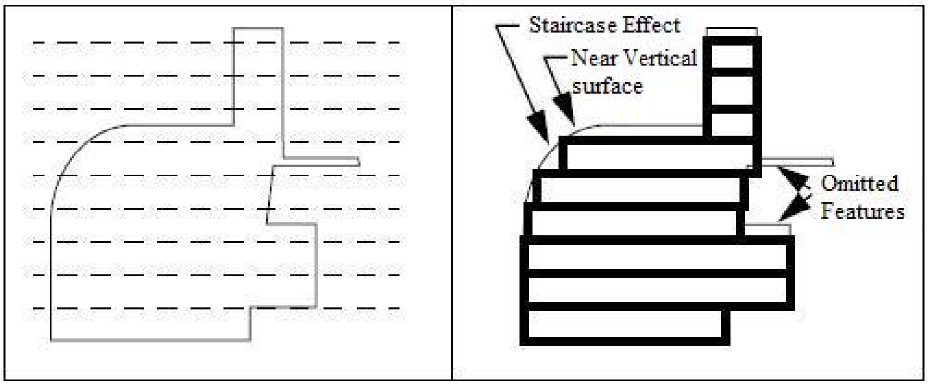


Figure 2.15: Staircase effect in uniform slicing (Venuvinod and Ma, 2004).

In addition to staircase effect, the slicing operation can also be negatively influenced from the STL data since the faceted data has some problems that originate from the nature of being a simple approximation of a true model. This fact can result in an inaccurate outer surface or coarse surface quality for the parts with complex surface designs. This deficiency can be solved by using a different slicing approach called direct slicing. In direct slicing, the original NURBS (Non Uniform Rational Bezier Spline) surfaces are used instead of their faceted approximations. So, the data is not imported in STL format but in some standard formats like IGES (Initial Graphics Exchange Specification), STEP (Standard for the Exchange of Product Model Data) or in native formats of different CAD programs.

Although there are some studies on adaptive and direct slicing, uniform slicing approach is still the mostly used slicing approach in the commercial RP software market. More detailed information about the adaptive and direct slicing algorithms can be found in the study of Marsan, et al (Marsan, 1998).

Faceted data provides a great advantage since only a plane to plane type intersection is computed to find the resultant intersection curves. However, commonly used STL data format does not contain connectivity information of the triangles (Marsan, 1998). As a result of this, a time consuming searching operation is required to find the triangles that are intersecting with a cutting plane positioned at a particular height. Hence, finding the connectivity information of the triangles increases the efficiency of the slicing operation.

Rock and Wozny (1991) used a topological based marching algorithm to find each intersection curve (Figure 2.16). In this study, a cutting plane intersects with an edge of a triangle and the other two edges of the same triangle are checked for another intersection. When two edges of a triangle are cut by a single cutting plane, the first line segment of the intersection curve is obtained. Then the next triangle is found by using the common edge shared by adjacent triangles. So the cutting operation is marched to the forthcoming triangles until the intersection curve at this particular height is closed. The same algorithm can also be used by spatially partitioning the triangles according to their heights (Gültekin, 2003). This partitioning improves the slicing speed.

Chalasani, et al (1991) used an approach different from marching algorithm. In this approach, for each cutting plane, all triangles are checked for an intersection. The slicing operation is done randomly between the triangles. So, when all intersections of a particular cutting plane are found, the resultant lines are ordered to satisfy the continuity of the contour curve. In a study using the same algorithm, parallel processing is used to speed up the slicing operation (Kirschman and Jara-Almonte, 1992).

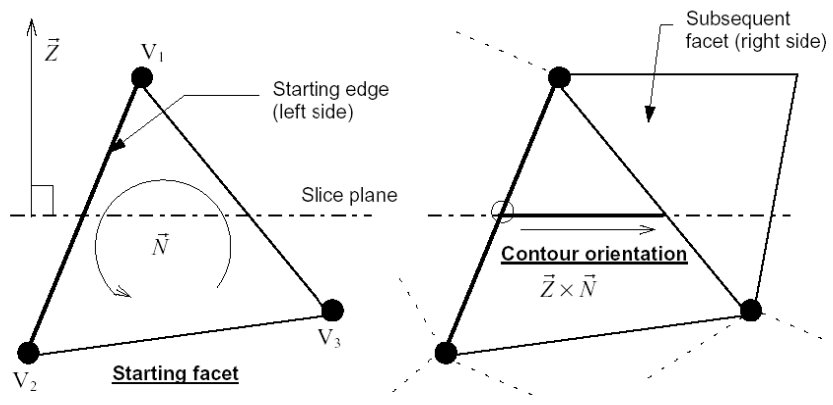


Figure 2.16: Marching algorithm for slicing (Gültekin, 2003).

2.7.5 Planning Tool Path

In the path planning step the tool paths which are used to build each layer are determined. In this operation the resultant contour curves of the slicing operation are used as input data. This operation can be classified in two categories. In the first category, the layers are formed at once so the contours are used directly as the tool paths. Laminated object manufacturing and solid ground curing techniques can be the examples of this category. In the second category, the tool paths are generated in a way that the inside of the layers are incrementally filled with the material, the tool paths may or may not include the contour curves. Fused deposition modeling and stereolithography are the two examples of this category (Marsan, 1998).

The tool paths followed by the production head affects different manufacturing parameters like the production time, surface accuracy, surface quality, stiffness, strength and post-manufacture distortion. During the fabrication, the tool head is re-positioned in the start of each layer and accelerated and decelerated to make the necessary direction changes. This movement influences the production time. So, the distances between the start points of each layer and the number of direction changes should be minimized while generating the tool path. In the tool path generation, the deposition width and depth of the material should be considered to satisfy the surface accuracy and the exact part dimensions. In production techniques where the material is deposited as molten, the temperature difference between the previously deposited layers and the added material may lead to residual stresses and subsequent warpage or other distortion of the part. A suitable choice of tool paths can minimize this effect, so, this effect should also be considered in tool path generation. Finally, the contact area between the newly deposited material and the previous layers plus the time passed between the depositions of these materials has an important effect on the stiffness and the strength of the part. In some techniques, the deposition direction of subsequent layers can affect the strength and the stiffness. Therefore, changing the deposition direction by 90 degrees in each layer can improve these properties. More detailed information about the studies made on these factors can be found in the survey made by Marsan and Dutta (1997).

Different methods and algorithms can be used for filling a layer. In the study made by Chang (1989), 3D cubic elements called voxels are used to generate the tool paths by superimposing the STL model. In this method interior of the model is filled with voxels whose heights corresponds to the slice thickness. The tool paths are generated by moving through the neighboring voxels in horizontal or vertical directions. Other methods for determining tool paths are reported by Rock and Wozny (1991) and Chari and Hall (1993). In these studies, parallel rays are intersected with the contour of each layer and the rays are connected to the next ray at the point of intersection so the tool paths are generated (Figure 2.17). But, in this method, the biggest problem is the possibility to miss some small interior features such as holes. To solve this issue a method that finds small features inside the model is presented by Tate and Fadel (1996). Another approach is to trace the counter curves by giving an offset rather than filling the interior of the object by using parallel lines (Yang, 1995). By doing so, the resultant tool paths are longer compared to the tool paths generated by the raster fill method, so the production time is minimized. Also, the need for support structures can be reduced since it enables the construction of an overhang by a sequence of offsets (Marsan, 1998). On the other hand, the models fabricated by using raster line methods have better strength and stiffness properties than the models produced by using offset contour fill methods. There are some variations of this approach that use different type of solutions to the problem of finding the offset contours. The details of these alternative methods can be found in the survey of Marsan and Dutta (1997).

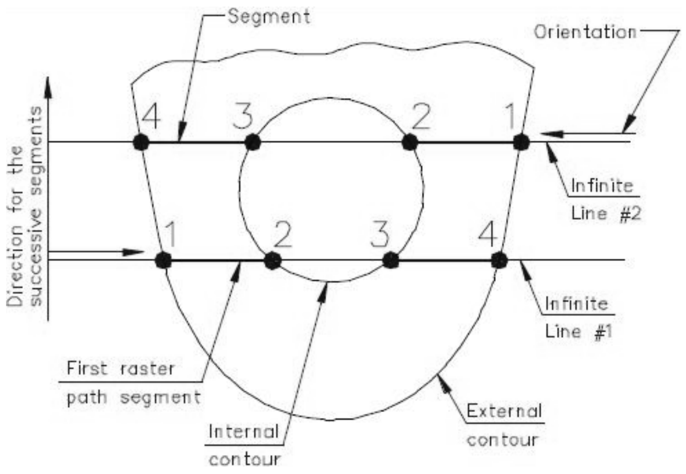


Figure 2.17: Raster tool path segment creations for a slice contour (Gültekin, 2003).

2.8 A Survey on Rapid Prototyping Software Market

RP system manufacturers in the market use different software solutions in their products. These solutions can be classified in two groups. The software packages in the first group can be referred to as product software that is specific to an RP machine. The second group is mostly known as the complete RP software solutions.

Product software packages are developed by the RP system manufacturers. These software packages are the system specific applications that are designed to satisfy the basic requirements of the manufacturer's product (Table 2.1). The purpose of this type of software packages is to generate the required control data for a specific RP machine by performing the necessary process planning steps.

Table 2.1: RP software packages and systems developed by different manufacturers

Company	**Software**	**RP System**
3D Systems	InVision™ System Software	InVision 3D Printers
	3D Lightyear™ Software	SLA Systems
	LS™ Software	SLS Systems
Stratasys	Insight™ Software	Stratasys Systems
	3D Printing Catalyst™ Software	Dimension Systems
Z Corporation	ZPrint™ Software ZEdit™ Software Mimics Z™ Software	3D Printing Systems
Solidscape	ModelWorks™ Software	Solidscape Systems
EOS	RP Tools™ Software	EOS Systems
Objet	Objet Studio™ Software	Objet Systems

Complete RP software solutions are developed to satisfy all the requirements of a RP system. These software packages support different types of RP systems from different manufacturers and are mostly distributed by RP manufacturers with their systems. These software packages have modular architectures. They consist of different modules performing different functions. Mostly the main modules perform tasks like part visualization, STL file repair, part editing, smart part placement, support structure generation, part slicing, tool path generation, CAD file formats to STL conversion, production time and cost estimation and etc. Magics software developed by Materialise company and Viscam software developed by Marcam Engineering company can be listed as the market leaders. The basic modules of the Viscam and Magics software packages and the pricing and functionality information of these modules can be found in the Tables 2.2 and 2.3. The technical specifications of these software packages are given in Appendix B.

Table 2.2: Functionality and price information of Viscam Software modules

Modules	**Functionality**[2]	**License**[3] **Price**	**Service Price**
Viscam View	Import of STL, 3DS, VRML, DXF, PLY, ZCP and VFX files, visualization of STL problems, detection and separation of included solids, model info like dimension, volume and surface area, visual measuring and annotation functions	Free	Free
Viscam Mesh	Detect defective edges, holes and triangles, automatic STL problem solving, accurate reduction, smoothing or filtering of triangles, cut and split the model along defined section planes, editing of solids, surfaces, triangles, Boolean operations, offset and extrusion functions, attach text, logos or bitmaps and create base solids, trim, cut and punch meshes with definable poly-lines, calculation and generation of error-free hollow models	1990€	300€
Viscam RP	Integrated database with more than 150 RP machines, copy, insert, orientate, scale and place individual parts, fully automatic import, placement and nesting of parts, adjustable build time estimation and cost calculation, fast and exact slice generation, compensation of material shrinkage and spot radius hatch generation for laser and jet based RP machines, support generation, file export support CLI, SLC, F&S, SLI, ISO, NC, SSL, STD, DXF, HPGL, BMP, PNG, TIFF formats	5980€	900€
Viscam Import	IGES, VDA, STEP file format support	995€	150€
Total		**8695€**	**1350€**

[2] Functionality information has been taken from the specifications of Viscam software.

[3] License and service price information is requested from the Marcam Engineering company by mail as of 6 February 2007.

Table 2.3: Functionality and price information of Magics Software modules

Modules	**Functionality**[4]	**License**[5] **Price**	**Service Price**
Magics Base	Visualization, measuring and manipulation of STL files, fixing STL files, uniting shells, trimming surfaces, double triangle detection, cutting STL files, punching holes, extruding surfaces, hollowing, applying offset, boolean operations, triangle reduction, smoothing, labeling, part nesting, collision detection	5200€	1040€
Magics RP	The platform concept, Build time estimation, Quotation Making, Slice verification		
Magics Import	IGES, VDA, STEP, Unigraphics, Pro/E and Catia (V4.5x and V5) and STL file format support	2200€	400€
Magics Slice	Slice parts, generates tool paths, writes out slice files for 3D systems, EOS, Stratasys and Sanders	1500€	300€
Magics Support	Fast and easy generation of support structures	2200€	400€
Total		**11100€**	**2140€**

The functionality and architectural organizations of these commercial software packages are used as guidance in the design and implementation of the software developed in this thesis study. The concepts explained in this chapter are widely used and referenced in the next chapter where RP software is presented in detail. The most of the knowledge used in the development of the RP software is mainly build on the information provided during this literature survey.

[4] Functionality information has been taken from the specifications of Magics software.

[5] License and service price information is requested from the Materialise company by mail as of 7 February 2007.

CHAPTER 3

SOFTWARE DESIGN AND IMPLEMENTATION

In this chapter, design and implementation of the software is discussed in detail. First, requirement analysis of the software based on the aim of the thesis and the commercial RP software market is given. The design and implementation tools are listed with their alternatives and the reasons of choice are justified. The software architecture is described and architectural and functional properties of the individual software parts are presented.

3.1 Requirement Analysis

When the aim of the thesis and the RP software market is considered, a RP application, which claims to satisfy the possible requirements of future RP studies and hence facilitate them, must be a complete and fully independent software package with an open for development and easy to use architecture. To be able to design an extendible and application independent software package, this RP software should be considered as a multipurpose application, which can be used in different engineering studies, rather than a specific RP application.

The software package must have a 3D modeling environment with 3D hardware visualization support to be capable of displaying different types of inputs and outputs. A multi screen support should be used to be able to present different information at the same time. A graphics library that facilitates the design of new graphic objects must be developed to use the same application core in different engineering applications where different types of visualization requirements and 3D objects can be required.

In the RP software, a modular architecture must be used to take advantage of software modularity in the design, development, release and update of the resultant software package. The software modules must be well organized according to their functionality.

A core module that supports a generic file transfer system can facilitate the use of different file types, which can be required in a future study. This interface is required to support multi file import and export. In the software, more than one file must be visualized at the same time. Since more than one part is imported, a part organization system can be developed to group parts. Part selection functionality has to be implemented to be able to perform different operations to different parts. In addition to these, undo/redo functionality may be provided for the sake of easy usage.

RP software must satisfy the process planning steps of RP. Therefore, additional modules with part modification, part transform, part slicing and tool path generation functionalities must be developed. In the current phase, the support generation step can be neglected since it is not required for all RP systems. On the other hand, support generation functionality should be considered as a future improvement in the design of the modules. Functionalities that facilitate easy modification and orientation of parts should be provided to the user. Parts should be scaled uniformly or non-uniformly, moved absolutely or relatively and rotated around three different directions. Additional functionality like part slicing in X, Y and Z directions to preview the production or to see the interior of the model can be provided to the user. Finally, a direct printing interface must be designed and implemented to be used in the RP machines that can be designed in future studies.

3.2 Software Design and Implementation Tools

The tools used in the design and implementation of a 3D application software can be studied in two groups. These groups can be listed as programming language and 3D application programming interface (API). In the forthcoming sections these groups are presented in detail.

3.2.1 Programming Language

When the programming languages that are used in different 3D applications in the market are considered, C, C++, C# and Java programming languages can be listed as the most commonly used options. These programming languages have different advantages and disadvantages, so, while making the choice, these languages should be considered in depth according to their performance, 3D support and ease of use.

The C programming language is the oldest programming language in the alternatives listed above. Today, it is mostly used in the applications where performance is critical since an application written in C executes faster than the applications written in the other three programming languages. In addition, all the popular 3D APIs can be used in a 3D application written in C. On the other hand, it is difficult and time consuming to develop a complex program by using C (Troelsen, 2005). Hence, only the performance critical parts of the complex 3D applications are written in C programming language.

The C++ programming language is a C based object oriented programming language, thus, programming complex software needs less effort and less time compared to C. However, compared to new generation programming languages like C# and Java, it is still time consuming to write complex applications. In C++, memory management, which is a problematic procedure that requires a serious amount of time and effort, is handled by the programmer (Troelsen, 2005). Besides, the amount of code written in C++ is considerably high compared to C# and Java to implement the same application. On the other hand, 3D applications written in C++ executes faster than the applications written in C# and Java. Also, C++ supports all the popular 3D APIs. So, it is still widely used in 3D applications where performance is a more important concern than time.

With the expansion of internet usage, the portability of applications has become an important issue. At this phase, a new generation of programming language called Java was released. An application written in Java is compiled into a processor independent intermediate language. This intermediate language can be transferred over the internet and recompiled at the target system in runtime. The compilation in the runtime is performed by using client software compatible with the target system. Consequently, the portability of the application is increased. With its portability, Java becomes the mostly used programming language in internet based applications. On the other hand, as a side effect of processor independency, the runtime performance is decreased. Java is rarely used in 3D applications because of its poor performance in 3D applications (Troelsen, 2005).

The newest programming language among the candidates is the C# programming language. C# is a new generation, C based object oriented programming language. Some of the software developers claim that C# contains the power of C and the portability of Java (Robinson, 2004). With its new generation of memory management system and .NET framework support, writing a piece of code for a specific task requires less time and less effort compared to C++ with an affordable performance decrease (Troelsen, 2005). In addition, one of the most popular 3D APIs that can be used in a 3D application is written in C#. Consequently, C# programming language is used both in 3D desktop and internet based applications.

In this study, a relatively complex 3D application must be design and implemented in a relatively short time. Therefore, C can be eliminated because it is not an object oriented programming language and Java can also be eliminated for its low 3D performance. Then, a choice between C# and C++ programming languages must be made according to ease of use and performance. In our study, as a consequence of time limitations, ease of use is a more important issue then performance. So, C# programming language is chosen as the programming language of our RP software. Another reason for choosing C# is that, as of the time this thesis work is carried out, C# is the mostly preferred language among fellow students who are expected to build applications upon the open architecture provided as a result of this work.

3.2.2 3D Application Programming Interface

The most popular 3D application programming interfaces can be listed as DirectX and OpenGL. Nearly all the 3D applications in the market use one of these 3D APIs. These APIs have similar properties and functionalities but different compatibility and performance issues. Therefore, while making the choice, these criteria should be considered in depth.

DirectX API is a product of Microsoft Company. It is mostly used in game programming. It is fully compatible with C# and Microsoft Windows operating systems. Consequently, compared to OpenGL, DirectX shows a better performance in a 3D application written in C# (Managed DirectX, 2003).

OpenGL API is used in both engineering and gaming applications. It is released by an independent consortium composed of different hardware and software developers. It is more portable compared to DirectX. It can be used in open source operating systems. But it is not fully compatible with C# programming language. There are different APIs that supports OpenGL in C#. But these APIs are not official solutions for the compatibility problems of OpenGL and C#.

Since both APIs have similar properties and functionalities that satisfy the requirements of our RP software, due to the compatibility and performance issues, DirectX API is preferred to be used in our RP software.

3.3 Software Architecture

The commercial RP software packages in the market contain independent software parts with different functional properties that are somehow related. These independent software parts are called modules. Modules communicate with other parts of the software through interfaces to abstract their functions and properties. Modules ensure their independency with their self sufficient structures and communication interfaces. The relations between the modules are provided by a single part which is called base module. The base module holds the necessary information about the functionalities of the other modules and uses them to perform the necessary tasks. Each module can be added to the base module independently. The architecture that is used by these software packages is called modular architecture. This architecture provides advantages to the developer in the design, development, release and update of a software package.

Owing to modular architecture, each part of the software can be designed independently. Since modules consists of related functionalities that are independent form the other parts of the software, these distinct parts can be designed by different programmer groups. So, the programmers can focus on a specific functionality rather than focusing on the entire software. Therefore, a better design can be achieved.

The independency of the modules also shortens the development period of the software. Since each part has a well defined independent functionality, they can be implemented and tested independently to satisfy the functional requirements. So, more stable parts with fewer errors can be developed.

In addition to these, with the help of modular architecture, the software companies release different package options with different functionalities that satisfy the needs of different customer profiles. Therefore, their customers can choose to buy only the required functionalities of a software package rather than giving extra money to unnecessary functionalities that they do not use.

In a modular application, if an update is required, only the parts that need revisions can be updated, hence none of the other parts is affected from any version change of an individual part. Thus, a problem can be fixed by using smaller update files without re-installation of the application software. This means that the customers can easily update their software packages over the internet in a relatively short time.

By considering these benefits, a modular architecture is implemented in the developed RP software. A functional classification depending on requirement analysis is made and the RP software is divided into three modules (Appendix C). These modules are named as View Module, RP Module and Slice Module. Each module consists of two sub parts called user interface and engine (Figure 3.1). Engine is the part where the necessary functional operations of the module are performed. User interfaces are the parts that are used by the engine to get the necessary input from the user.

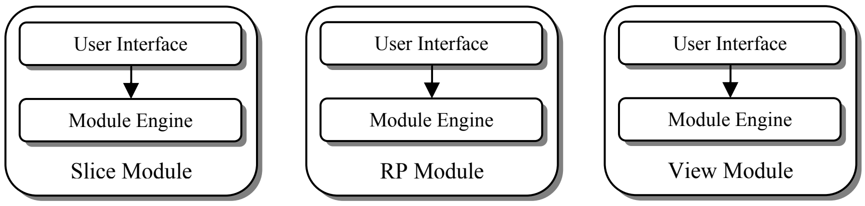


Figure 3.1: RP software modules and their sub parts.

The View Module is the base module of the application. This module uses the RP Module and Slice Module to perform the necessary process planning steps. The data processed in these modules are displayed by the View Module. The View Module uses a 3D display framework to abstract the details of 3D rendering process. This framework is called Graphics Framework. Graphics Framework uses .NET Framework, Microsoft Direct3D and Microsoft Windows (WIN32) APIs to render 3D objects (Figure 3.2). All the models, slices and 3D graphical user interfaces that are sent by the View Module are displayed in the Graphics Framework.

3.4 Graphics Framework

The Graphics Framework (in Figure 3.2) is designed and implemented as a part of the developed RP software to provide 3D visualization support. It can be described as a high level 3D engine that can be used in engineering applications where 3D visualization is required. The Graphics Framework is an application independent, multi-screen 3D modeling environment and graphics library that is based on the Microsoft Sample Framework.

The Microsoft Sample Framework is an open source layer “used by most of the Microsoft Direct3D samples and is built on top of the WIN32 and Direct3D APIs. Its goal is to make Direct3D samples, prototypes, and tools as well as professional games more robust and easier to build. It simplifies the WIN32 and Direct3D APIs for typical usage and is designed to help make simple to moderately complex Direct3D applications" (Microsoft, 2005).

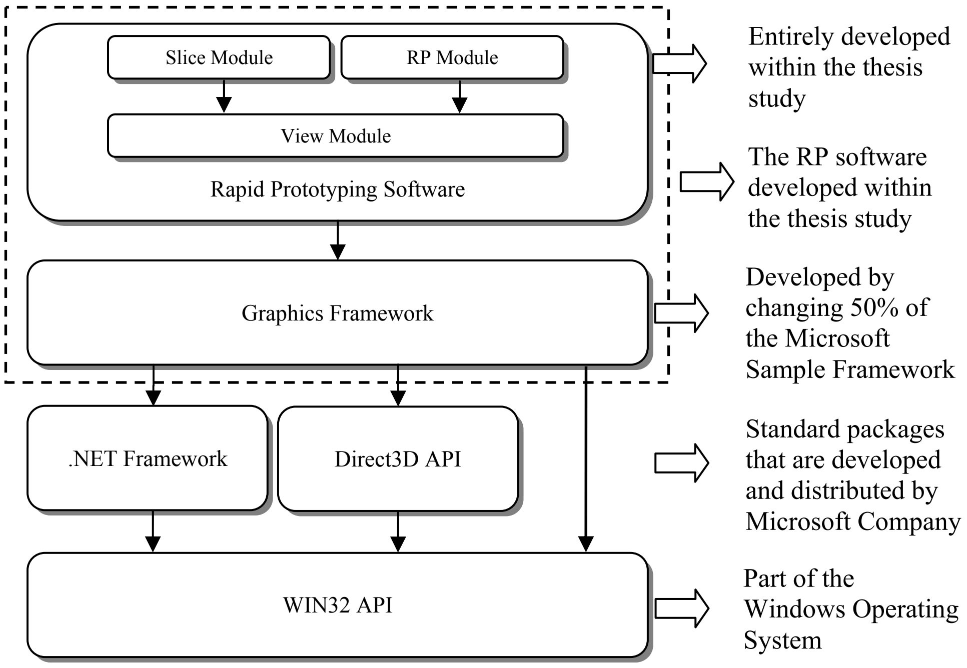

Slice Module
RP Module
View Module
Rapid Prototyping Software
Graphics Framework
.NET Framework
Direct3D API
WIN32 API
Entirely developed within the thesis study
The RP software developed within the thesis study
Developed by changing 50% of the Microsoft Sample Framework
Standard packages that are developed and distributed by Microsoft Company
Part of the Windows Operating System

Figure 3.2: Relations between RP Software, Graphics Framework and other APIs.

The Microsoft Sample Framework is designed to be used in single or full screen gaming or demonstration purpose 3D applications. In this framework 3D user interfaces are used instead of classical Microsoft Windows user interfaces. The internal architecture is designed to be used with DirectX mesh objects which are mostly used in texture based gaming applications where the complex models with textures are required. Therefore, to be able to develop a graphics framework to be used in the visualization of 3D engineering applications, more than 50% of the Microsoft Sample Framework is rewritten or completely changed and a considerable amount of additional architectural property is added. Application independency, multi-screen support, 3D modeling environment and graphics library can be listed as these architectural properties. In the forthcoming sections these architectural properties are expressed in detail. More general information about the Microsoft Sample Framework can be found in Microsoft DirectX Programmer's Reference (2005) and the book of Miller (2004).

3.4.1 Application Independency

Application independency can be described as being self sufficient to perform its functional responsibilities without using any other application. Graphics Framework is a fully independent, open for development, high level 3D modeling environment and graphics library that can be used in engineering applications where 3D visualization support is required.

In some of the engineering applications the visualization needs are met by utilizing different 3D CAD software packages as graphical user interfaces. The thesis study presented by Gültekin (2003) can be given as an example of this type of usage. In this study, an RP application is developed as an add-on of AutoCAD software package and the visualization needs of this application are utilized by using this CAD software. Therefore, the application is written fully dependent on AutoCAD.

Two deficiencies can be listed for this type of usage. First one is the functional limitations of the used application. The limits of the further studies are dependent on this application since it cannot be modified or further improved according to the needs of the study. Secondly, for each copy of the software, an additional license price must be paid. Use of commercial software package may also cause some copyright problems in the distribution of the study.

The Graphics Framework can also be used in this type of academic purpose engineering applications to overcome these deficiencies owing to its application independent architecture. With its open for development architecture it can be modified or improved with additional functionalities to satisfy special requirements of different type of studies without any copyright problems.

In addition to the RP software presented in this thesis study, the Graphics framework is also used in the study of Sezginalp (2007) to develop a 3D localization and mapping application for mobile robots. In these two application examples, the application independency of the Graphics Framework is verified. When these two applications are considered, it can be said that, moderately complex 3D applications can be designed and implemented in a considerably short time by using the Graphics Framework.

3.4.2 Multi Screen Support

The Graphics Framework provides a multi screen interface to display more than one 3D modeling environment at the same time (Figure 3.3). All the functionalities of the framework can be used concurrently by all the windows in the multi screen interface. Owing to multi screen support of Graphics Framework, the user can display many files in different windows to choose the suitable parts to create a printing job, while performing another printing operation of an existing project, in a different window. With the multi screen support, in a single application that uses the Graphics Framework, different operations can be performed independently and simultaneously in different windows.

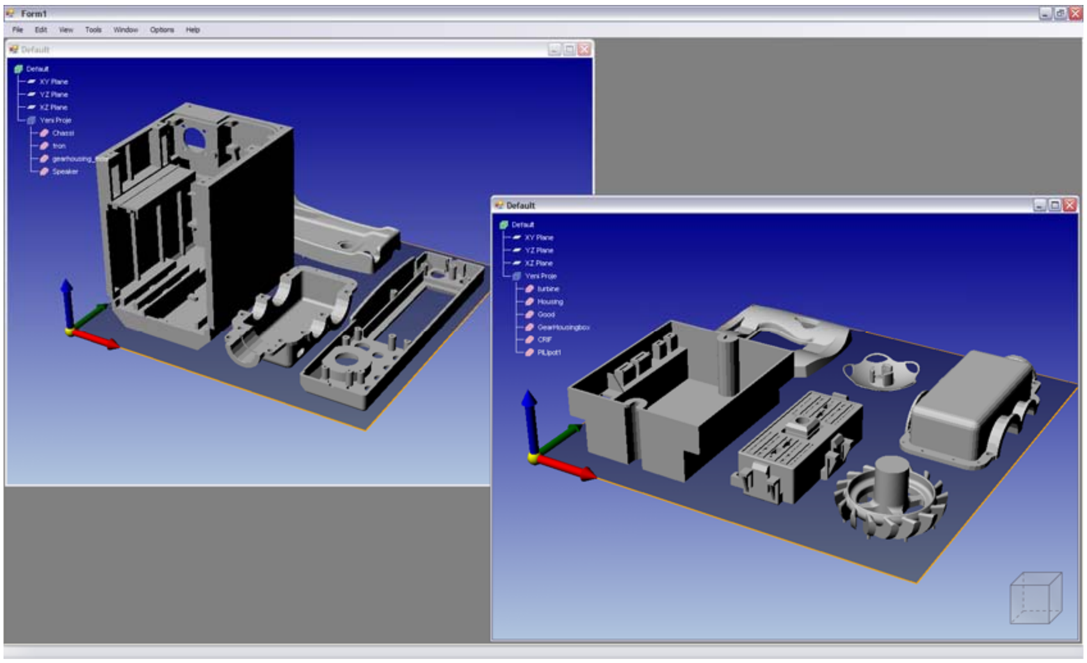


Figure 3.3: Multi screen support provided by the Graphics Framework

3.4.3 3D Modeling Environment

The Graphics Framework abstracts the complex low level functions of the Direct3D API and WIN32 API to a high level 3D modeling environment where 3D objects can be displayed. A camera with a perspective view cone is used to show the objects in the environment. The 3D environment is illuminated with the use of directional lighting technique. Perspective view cone and directional lighting technique creates the 3D view effect.

The 3D modeling environment uses the WIN32 API to get the user actions through the keyboard and the mouse. These actions are used to create different views of 3D objects by transforming the positions of the camera and the lights. Panning, free handed rotation and zooming actions can be performed by the user to see the environment from different directions. Also, there are some predefined views. These views are the ones that are mostly used in the CAD software; front, back, left, right, top, bottom and isometric views (Figure 3.4).

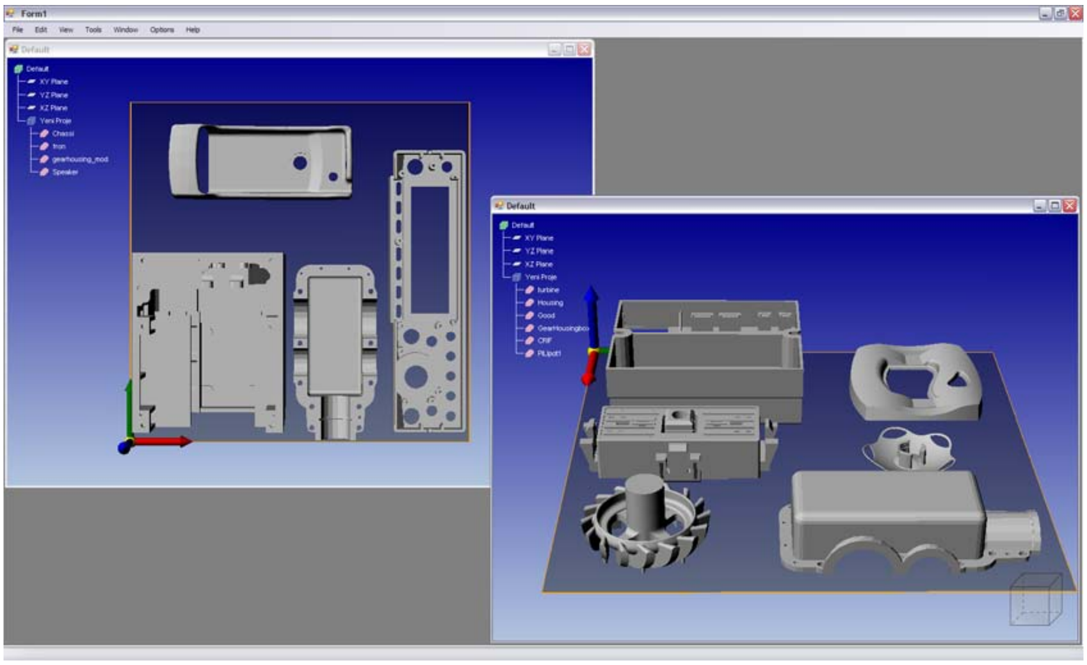


Figure 3.4: Top and top front views of different 3D objects.

User defined objects compatible with the Graphics Framework can be created by using two methods. As the first method, a special interface called IDrawable can be used to create a Graphics Framework compatible object in any complexity and functionality (Appendix D). However, using this method requires a high level of Direct3D API knowledge and experience since all the required low level rendering functions of the 3D object must be implemented by the user. As a second method, the existing 3D objects in graphics library can be used to create new 3D objects. In this method, new objects can be created by using inheritance rule of the object oriented programming approach. This is an easy to use method since only the basic object oriented knowledge is required. When 3D object is created by using the graphics library, all the low level rendering functions that are required for the Graphics Framework compatibility are handled by the parent graphics object.

3.4.4 Graphics Library

As a part of the Graphics Framework a completely new graphics library is developed to be used in engineering applications. In this library there are different types of objects that can be classified in two groups.

The graphics objects in the first group are the structural objects. This type of objects can only be used to create new objects. Their instances cannot be created in runtime therefore they cannot be directly used as 3D objects. There are two objects in this type. The first one is called Entity. Entity is the lowest level graphics class. It implements an interface called IDrawable in order to be compatible with the Graphics Framework. Entity is a generic class that can be used to create any type of 3D objects. The second one is a class called Feature. Feature is a higher level structural class which is inherited from the Entity class. Hence, it has all the properties and functionalities of the Entity class. It also has higher level functionalities like selection, transparency, object bounding box and object transformation. In the creation of 3D objects mostly the Feature class is preferred since it is easier to implement a 3D object by using the Feature class rather than the Entity class. The details of IDrawable interface, Entity and Feature classes can be found in the Appendix D.

The graphics objects in the second group are the functional objects. These objects are used by the Graphics Framework to visualize some necessary functionality such as Bounding Box, Background, Line and Plane objects. Bounding Box is used by the Graphics Framework to display the bounding boxes of the high level objects. Background is a plane used in all rendering windows to form a background picture. Line and Plane objects are used as the sub elements of the Bounding Box and Background objects. These objects are also accessible from the applications that are using the Graphics Framework and therefore they can also be used by these applications as well.

3.5 Modules

The RP software that is developed in this thesis contains three different modules to perform the process planning steps that are described in the "Process Planning in Rapid Prototyping Software" section of the first chapter. In this section, how these steps are performed is explained by using the algorithms and the architectural structures of modules and some other functionality of the modules are discussed.

3.5.1 View Module

The View Module is the base module of the RP software. It is the most important module of the application because it is structurally the core part of the whole process. It is used in the visualization of the inputs and outputs of the application. All the user interfaces, modules and their functionalities are handled by this module. In addition to these, it also contains different structural properties that are critical for future expansion of the RP software. These properties can be listed as generic file format support, part organization and selection support and undo/redo support.

3.5.1.1 Generic File Format Support

All applications, which use a specific type of data to perform some of its functionalities or generate a specific type of data as a result of an operation, must support one or more standard file types to import or export data. For an open for development RP application that is designed to be used in research studies, data import and export support for different file formats is an indispensable requirement. Therefore, in the RP software package, a generic data transfer structure is implemented as a part of the View Module to be able to support different file formats.

All the data transfers between the RP software and external sources are performed by using this generic data transfer structure. This structure consists of three elements. The first two elements are the data transfer interfaces that are called IWriteable and IReadable (Appendix D). These interfaces define the properties and functions that are required for the generic data transfer structure compatibility. The IWriteable and IReadable interfaces contain the definitions for write and read operations, respectively. The third element of the structure is an object list that stores a special type of objects called file format. File format objects perform read and/or write operation(s) for a specific file type.

Support for a new file format can be added to the View Module by performing a series of operations. Firstly, a file format object must be developed. According to the supported operation type, the developed object must implement IWriteable and/or IReadable interface(s). With the implementation of one or two of these data transfer interfaces, the file format object becomes compatible with the generic data transfer structure and can be added to the object list to be used in file transfer operations.

In a file transfer operation, a request for a specific file type is sent to the generic data transfer structure. According to this request, all file format objects in the interface list are searched for the requested file format and operation. If the requested file operation and file format is supported, then the operation is performed by the corresponding file format object. In case of a read operation, the data in an external source is read by the corresponding file format object and send back to the RP software by using the IReadable interface. In case of a write operation, this time IWriteable interface of the corresponding file format object is used to write the send data to an external source.

3.5.1.2 Part Organization and Selection

RP software can import more than one file or create part groups that contain multiple parts; therefore, a part organization structure with a selection support is required to clarify the relations between these objects and to make it possible to use different functionalities for different parts.

Part organization structure used in the RP software contains three elements that are named as project, product and part. The first element; project is used to define a build job. For each project a new screen with a 3D modeling environment is created by the View Module. The second one, product, can be defined as a collection of related parts. Products can be created as a part of a project or another product. The last element; part is used to define a part file that is imported from an external source. Part can be created as a member of a project or a product.

Each build job can be saved as a project file in which the relation information and part data is stored as an individual file or as a reference file. When project is saved as an individual file, all the relations and the parts included in this project are stored in a single file. This type of file is not affected from the changes made on the part files. If the project is saved as an external reference, only the relations and the locations of the parts are saved. In this case, when a referenced part file is modified all the projects referencing this particular part file also change.

In the internal structure of the RP software, project and product elements are simple collection objects that hold the related elements, while, the part elements are held in a special object called Model. The Model is a 3D object that is created by using the Feature object as it is explained in the "Graphics Library" section of this chapter. In the Model all the information and data about a part file is stored. All the part related operations are performed by using the Model object. Thanks to the object inheritance, the Model object can use all the properties of the Feature object including the transparency.

In the view module, the transparency functionality of the Model object is used to emphasize the selected parts (Figure 3.5). When a part is selected, all the unselected parts are displayed transparent and all the part related functionalities are performed only for the selected parts. The selection of a part or product can be performed through a graphical user interface (GUI) displayed in the project screen (Figure 3.5). In the GUI, the relations between the parts and products are displayed in a hierarchical tree structure by using the user defined names of these elements. User can select part(s) and/or part groups by using their corresponding product(s).

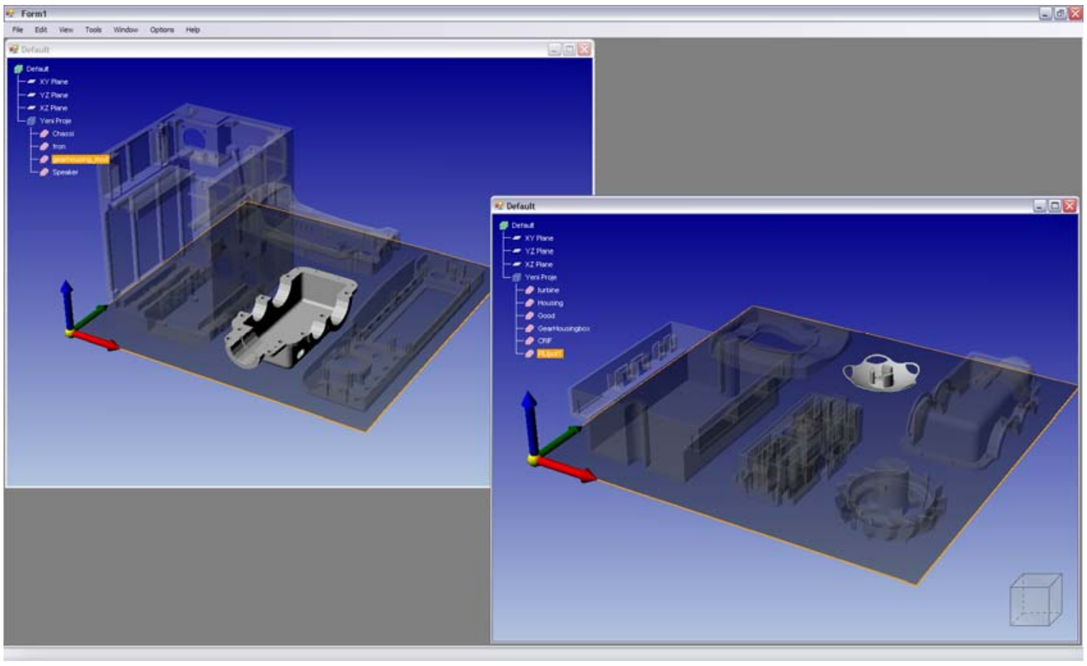


Figure 3.5: Different objects that are selected by using hierarchical tree structure.

3.5.1.3 Undo/Redo Support

One of the functionality provided by the View Module is the undo/redo support. In the RP software, the user can navigate through the executed user commands in the order of occurrence which means that a previously executed user command can be unexecuted or a previously unexecuted user command can be re-executed by the user. To handle this series of execution and un-execution operations, a command managing structure is used by the View Module.

In this structure, a command object is created for each user command where an undo/redo operation can be performed. This command object uses a special structure to execute or un-execute the related user command. The execution of a user command is a simple task since only the ordinary operation is performed by the command object. On the other hand un-execution of a user command is a hard to implement procedure since the state of the software must be reversed to a previous position by preserving the stability of the software. To perform a un-execution step, all the changes in the system must be stored by the command object. When a un-execution command is received, the related command object reverses all the changes in the system in the reverse order of occurrence to preserve the stability. This is a memory consuming process since all the state changes are held inside the command object; therefore a limited number of commands are held in the memory.

All the command objects are stored in a special storage system called command manager. The command manager is a part of the command managing structure. All the necessary information about the undo/redo operation like command objects, their orders, the point of navigation and the capacity of command object storage is hold by the command manager. The command manager handles all the execution and un-execution operations by keeping track of the user navigation. It reorders all the command objects after each undo/redo operation. Another responsibility of the command manager is to handle the memory usage by counting the command objects hold in the storage. When the number of command objects exceeds the limit, the oldest command object is removed from the storage to free up a place for the new coming command object.

3.5.2 Rapid Prototyping Module

In this module, the functionalities that facilitate easy modification and orientation of parts are developed. In addition, a direct printing interface for the supported RP hardware is implemented. The user can translate, rotate and scale parts, perform collision check between the parts and the fabrication limits and directly send the generated CAM code to a supported machine by using this module. In the forthcoming sections, these functionalities of RP module are presented in detail.

3.5.2.1 Collision Detection

In RP software, it is assumed that all the RP process is performed in a limited virtual 3D volume called workspace. Workspace is the virtual representation of production compartment of the machine, so it is a RP machine dependent process parameter that is used to define the limits of all fabrication specific functionalities. In RP software, architecturally, the workspace is defined by a 3D object that is called with the same name. This object is parametrically defined therefore different values can be set for different RP machines to visualize machine specific limits. Workspace object can be displayed in two different forms; a rectangular prismatic shaped 3D volume and a square shaped 2D platform (Figure 3.6). The platform shape is used to present the production platform of the machine and the 3D volume represents the whole production compartment.

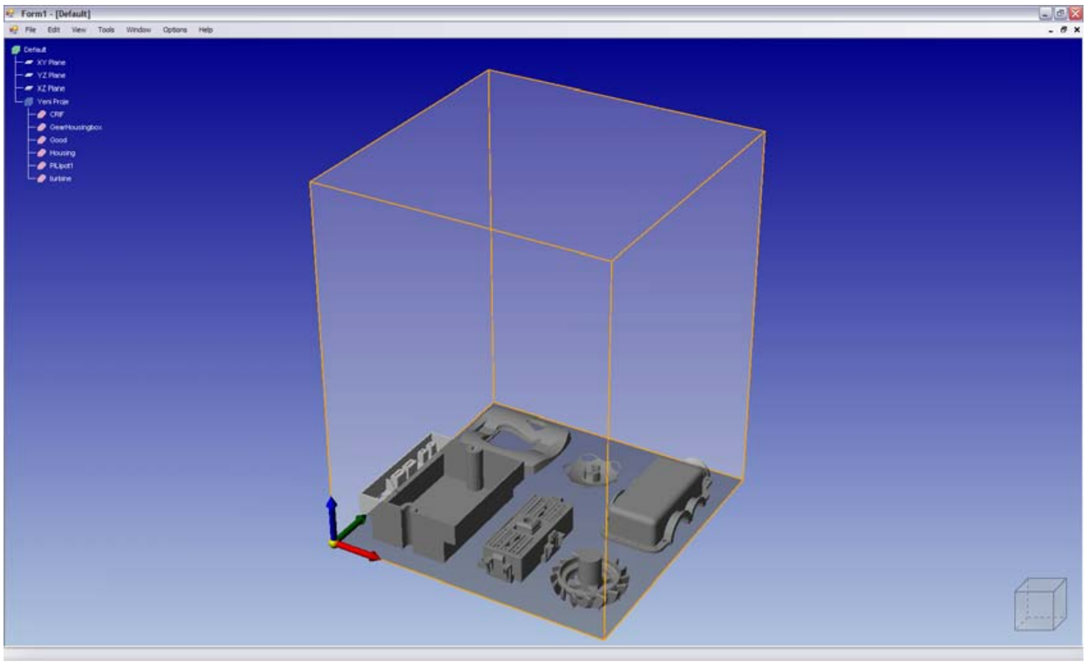


Figure 3.6: Platform and workspace are displayed with a set of parts inside.

In the RP software, all the parts must be oriented inside the workspace to be fabricated. Therefore, a collision between the parts and the boundaries of the workspace must be checked before each production operation. This collision detection is performed by the RP module by using a volumetric base comparison between the bounding boxes of the objects and the workspace. Each time a part is oriented, its bounding box is calculated and before a slicing operation, a collision between this bounding box and workspace is checked to continue the fabrication process.

3.5.2.2 Part Transformation

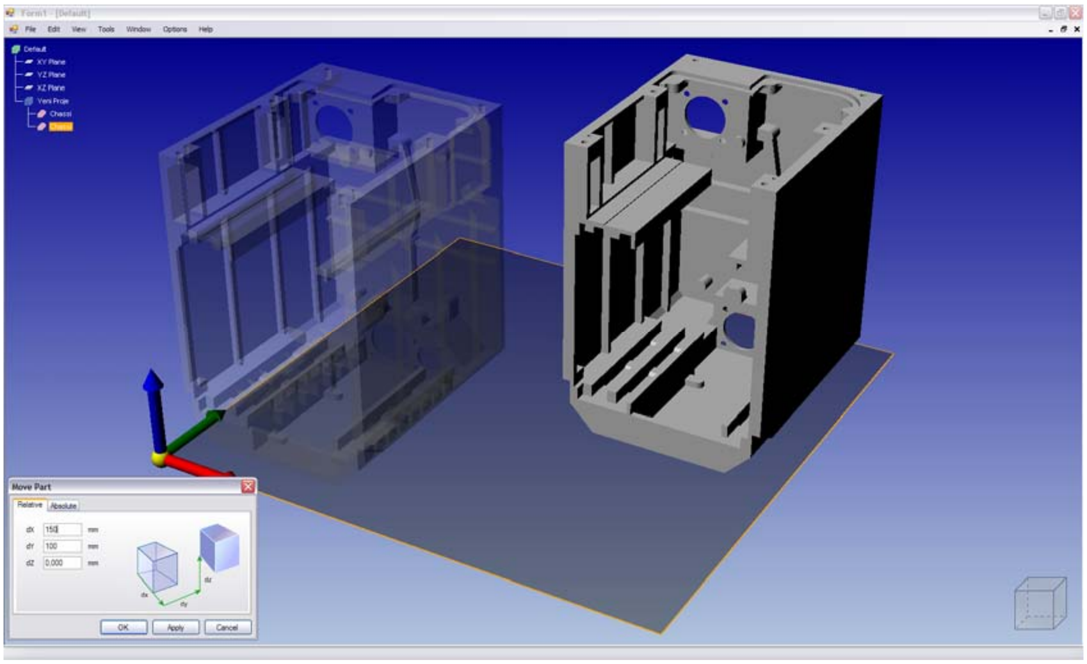
The RP module uses transformation matrices of Direct3D API to perform dimension and position modifications of the parts. For each operation a transformation matrix is generated according to the user inputs and sent to the corresponding Model object that supports a 3D transformation functionality inherited from Feature object. When the transformation matrix of the Model object is set, all the position and/or normal vector data of the part is modified. This modification can be performed by using two methods. The method of transformation can be chosen by the user.

The first way is to use object specific transformation support of the 3D modeling environment. 3D modeling environment provides a transformation matrix for each displayed 3D object. When this matrix is set, the 3D environment displays the 3D object as transformed without changing the real data. This method can be used to preview the effect of a transformation. A transformation operation performed by using this method can be reversed only by changing the transformation matrix of the 3D object to identity since the original data is preserved.

The second way is to directly transform the original position and normal vector data of the part. If the user requests a data transformation, the Feature object can perform all the position and/or normal vector data transformations by using 3D point and vector transformation functions provided by the Direct3D API. This operation is not reversible since the original data is changed. Therefore this method is only used when the original data is required in the transformed form for the slicing operation.

Both of the methods described above are used by the RP module to perform translation, rotation and scaling operations. When the user applies a transformation, the transformed forms of the parts are previewed by using the first method. By this way, an opportunity to cancel the operation is provided to the user since the first method is reversible. If the user confirms the previewed transformation, then the part data is modified by using the second method to be able to perform following operations by using the transformed form of the object.

The user can orient the parts in the build space by translating them absolutely to a given position or relatively with a given displacement (Figure 3.7) or by rotating them around X, Y and Z axes with respect to their centers (Figure 3.8). For each operation a different transformation matrix is generated by the View Module. In translation, generated transformation matrix is applied only to the position data of the part. The normal vector data is a direction representation; therefore it is not scaled since it is not affected from a translation operation. On the other hand, in a rotation operation, both position and normal vector data are modified to be able to rotate the part.



dX 150 mm
dY 100 mm
dZ 0,000 mm

Figure 3.7: Former and latter positions of a translated part.

The parts loaded into the 3D environment can be scaled in two ways: uniformly in all directions (Figure 3.9) or non-uniformly in different directions (Figure 3.10). In a scaling operation, a scaling matrix is generated by the View Module. This matrix is applied only to the position data as it is done in a translation operation.

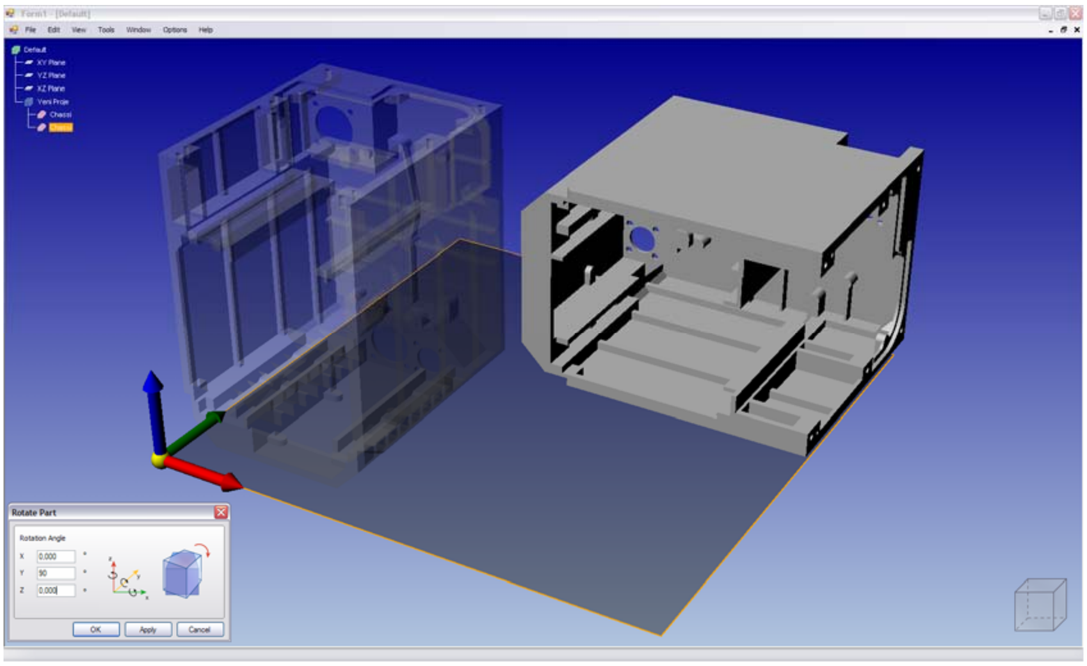


Figure 3.8: Former and latter orientations of a rotated part.

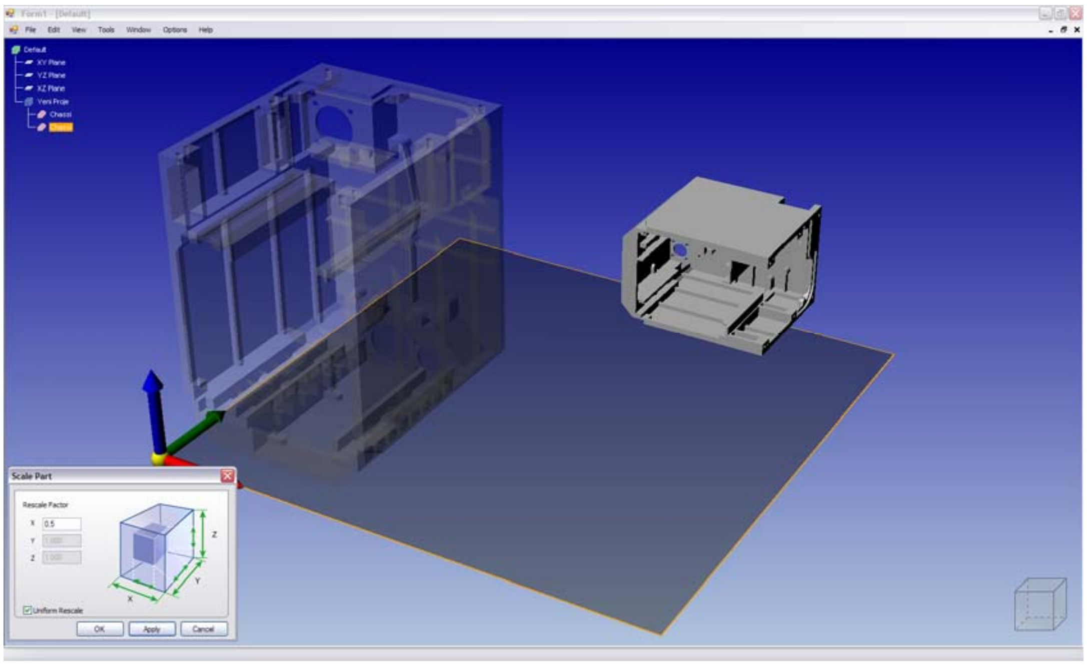
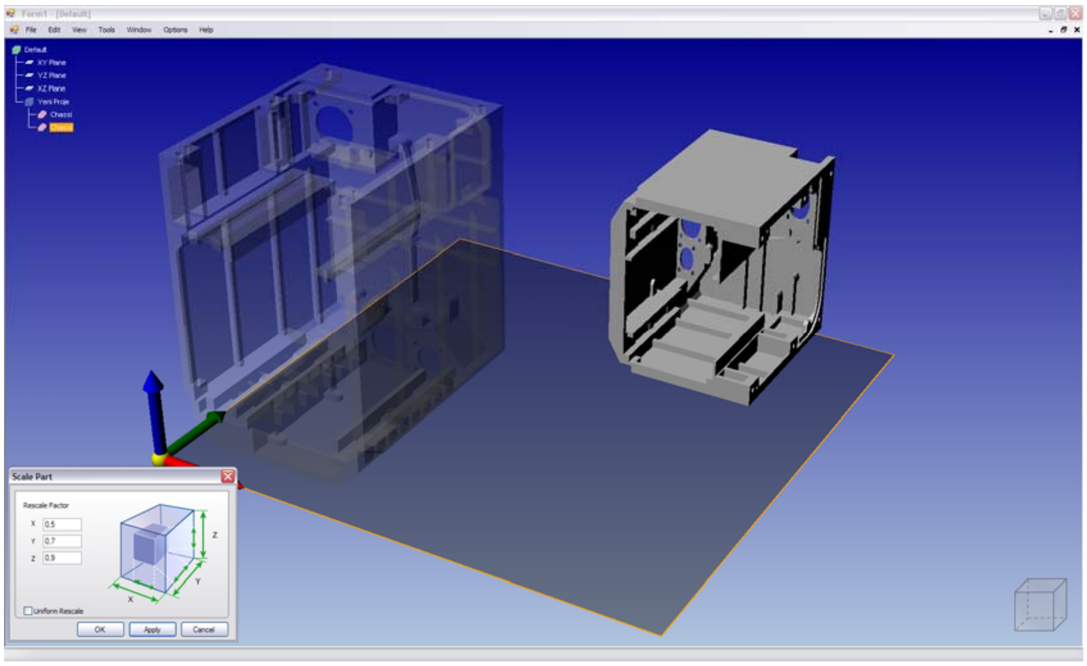


Figure 3.9: Former and latter sizes of a uniformly scaled part.



Rescale Factor
X 0.5
Y 0.7
Z 0.9
Uniform Rescale

Figure 3.10: Former and latter sizes of a non-uniformly scaled part.

The mathematical background of the part translation, rotation and scaling operations are provided in the following sections.

3.5.2.2.1 Part Translation

According to the Hearn and Baker (1996) the matrix expression for the translation of a position $P = (x, y, z)$ relative to its original position can be written as

$$\begin{bmatrix} x^1 \\ y^1 \\ z^1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.1}$$

where translation parameters t_x, t_y and t_z are assigned any real values. When this transformation is applied to the position data of the part, the resultant coordinates of the position can be expressed as

$$x^1 = x + t_x \qquad y^1 = y + t_y \qquad z^1 = z + t_z \tag{3.2}$$

To translate a part to an absolute point in the 3D environment the following transformation sequence must be applied.

1. Translate the center of the part to the origin
2. Translate the center of the part to the given absolute position

The resultant matrix of this transformation sequence can be written as

$$T(x_a, y_a, z_a) \cdot T(-x_c, -y_c, -z_c) = \begin{bmatrix} 1 & 0 & 0 & -x_c + x_a \\ 0 & 1 & 0 & -y_c + y_a \\ 0 & 0 & 1 & -z_c + z_a \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

where x_c, y_c and z_c are the coordinates of the center point of the part, x_a, y_a and z_a are the coordinates of the absolute point, $T(-x_c, -y_c, -z_c)$ and $T(x_a, y_a, z_a)$ are the translation matrices.

3.5.2.2.2 Part Rotation

According to the Hearn and Baker (1996) the matrix expression for the rotation of a position $P = (x, y, z)$ around the x, y and z axis can be written as

$$\begin{bmatrix} x^1 \\ y^1 \\ z^1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.4}$$

$$\begin{bmatrix} x^1 \\ y^1 \\ z^1 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.5}$$

$$\begin{bmatrix} x^1 \\ y^1 \\ z^1 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.6}$$

for a left-handed coordinate system where rotation parameters θ_x, θ_y and θ_z specifies the rotation angles around x, y and z axes, respectively. These matrix expressions can be defined in more compact forms as

$$P^1 = R_x(\theta_x) \cdot P \qquad P^1 = R_y(\theta_y) \cdot P \qquad P^1 = R_z(\theta_z) \cdot P \tag{3.7}$$

A general form of rotation matrix can be obtained by rotating an object around three axes in x, y and z order as expressed in Equation 3.8.

$$R(\theta_x, \theta_y, \theta_z) = R_z(\theta_z) \cdot R_y(\theta_y) \cdot R_x(\theta_x) \tag{3.8}$$

To rotate an object around an axis with respect to its center, the following transformation sequence must be applied.

1. Translate the center of the part to the origin
2. Perform the specified rotations around the axes
3. Retranslate the center of the part to its original position

The resultant matrix of the given transformation sequence can be found by performing the matrix operation given in Equation 3.9.

$$R_T(\theta_x, \theta_y, \theta_z) = T(x_c, y_c, z_c) \cdot R(\theta_x, \theta_y, \theta_z) \cdot T(-x_c, -y_c, -z_c) \tag{3.9}$$

where x_c, y_c and z_c are the coordinates of the center point of the part, $T(x_c, y_c, z_c)$ and $T(-x_c, -y_c, -z_c)$ are the translation matrices, $R(\theta_x, \theta_y, \theta_z)$ is the general form of rotation matrix and $R_T(\theta_x, \theta_y, \theta_z)$ is the resultant transformation matrix.

3.5.2.2.3 Part Scaling

According to the Hearn and Baker (1996) the matrix expression for the scaling transformation of a position $P = (x, y, z)$ relative to the coordinate origin can be written as

$$\begin{bmatrix} x^1 \\ y^1 \\ z^1 \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.10}$$

where scaling parameters s_x, s_y and s_z are assigned as a positive single value for uniform scaling or positive different values for non-uniform scaling. When this transformation is applied to the position data of the part, the resultant coordinates of the position can be expressed as

$$x^1 = x \cdot s_x \qquad y^1 = y \cdot s_y \qquad z^1 = z \cdot s_z \tag{3.11}$$

Scaling of a part with respect to the origin results in both dimension and position change if the part is positioned away from the origin. Repositioning of the part can be prevented by scaling the part with respect to its center point; to do this the following transformation sequence must be applied.

1. Translate the center of the part to the origin
2. Scale the part with respect to the origin by using Equation 3.1
3. Translate the center of the part back to its original position

The resultant matrix of this transformation sequence can be written as

$$T(x_c, y_c, z_c) \cdot S(s_x, s_y, s_z) \cdot T(-x_c, -y_c, -z_c) = \begin{bmatrix} s_x & 0 & 0 & (1 - s_x)x_c \\ 0 & s_y & 0 & (1 - s_y)y_c \\ 0 & 0 & s_z & (1 - s_z)z_c \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

where x_c, y_c and z_c are the coordinates of the center point of the part, $T(x_c, y_c, z_c)$ and $T(-x_c, -y_c, -z_c)$ are the translation matrices and $S(s_x, s_y, s_z)$ is the scaling matrix.

3.5.2.3 Direct Printing Interface

Main purpose of the RP software is to generate CAM data required to perform the fabrication of an imported part, however, properly transferring this data to a RP machine is as important as generating it since the generated CAM code becomes useless if it cannot be transferred to a production system. Therefore, in the RP software, a CAM data transfer interface is developed to be able to directly transfer the generated CAM code to a supported RP machine.

The RP software can connect to an external application by using its dynamic link library (DLL) if it implements an interface called IConnection (Appendix D). This interface defines the rules of communication of RP software with an external application. Therefore, an application DLL must be written for each RP machine that will be used with RP software. The user must implement IConnection interface to this application DLL and add it to the "Driver" directory of the RP software.

When a direct printing operation is requested by a user, the RP software searches its "Driver" directory to find the application DLLs implementing the IConnection interface and displays the supported RP machines that correspond to the found application DLLs to the user. The user can choose the printing device among the supported RP machines. Once a machine is chosen by the user, the RP software connects to the corresponding application DLL and sends the generated CAM code in Common Layer Interface (CLI) Format.

The CLI Format "is a simple, efficient and unambiguous slice format for data input to all RP systems. In CLI, each layer is represented by a set of contours and hatches. Contours define the boundaries of the solid material within a layer and are represented by polylines. Each contour should be closed and have no intersections with itself or with other contours. A hatch is also defined, as a set of independent straight lines each defined by a start and an end point. Hatches are used with open polylines to define support and filling structures. Polylines representing internal contours are ordered clockwise and those of external contours counterclockwise when viewed along the negative Z-axis. No non-geometric information is defined in CLI" (Marsan, 1998).

3.5.3 Slice Module

All the production based operations are performed in the slice module. In this module, the CAM code required for production is generated by slicing the parts and generating the tool paths. Once the tool paths are generated, these paths can be converted to CLI Format to be saved as an external file or can be directly sent to a supported RP machine.

3.5.3.1 Part Slicing

The most important functionality of the slice module is the slicing operation. In this module the parts oriented in the workspace can be sliced by using two different algorithms that perform two different slicing functionalities. These functionalities can be listed as preview slicing and production slicing. Both of these functionalities use slicing algorithms performing uniform slicing by using the same mathematical background.

3.5.3.1.1 Mathematical Background

In both algorithms, a plane to plane intersection between a triangle and a cutting plane is used. However, when the intersection between a triangle and a cutting plane is considered, it can be seen that the intersection occurs between the edges of the triangles, which are straight lines, and the contact line of the cutting plane. Therefore, a cutting operation can be simplified as an intersection of two three dimensonal lines.

According to Mahir (1999), a three dimensional line can be defined in a parametric form as

$$\left\{ (x, y, z) \in R^3 \,\middle|\, \begin{array}{l} x = \lambda a_1 + (1-\lambda) b_1, \\ y = \lambda a_2 + (1-\lambda) b_2, where \quad \lambda \in R \\ z = \lambda a_3 + (1-\lambda) b_3, \end{array} \right\} \tag{3.13}$$

where $R^3 = \{(x, y, z) | x, y, z \in R\}$, $a_1, a_2, a_3, b_1, b_2, b_3 \in R$ and $(a_1, a_2, a_3) \neq (b_1, b_2, b_3)$.

Cartesian equation of a three dimensional line can be found by eliminating λ from the parametric form given in Equation 3.13. To eliminate λ, the parametric form of a three dimensional line must be redefined as

$$\begin{aligned} x &= \lambda a_1 + (1-\lambda)b_1 = b_1 + \lambda(a_1 - b_1) \\ y &= \lambda a_2 + (1-\lambda)b_2 = b_2 + \lambda(a_2 - b_2) \\ z &= \lambda a_3 + (1-\lambda)b_3 = b_3 + \lambda(a_3 - b_3) \end{aligned} \tag{3.14}$$

$$\begin{aligned} a_1 - b_1 \neq 0 \quad &\Rightarrow \quad \lambda = \frac{x - b_1}{a_1 - b_1} \\ a_2 - b_2 \neq 0 \quad &\Rightarrow \quad \lambda = \frac{y - b_2}{a_2 - b_2} \\ a_3 - b_3 \neq 0 \quad &\Rightarrow \quad \lambda = \frac{z - b_3}{a_3 - b_3} \end{aligned} \tag{3.15}$$

From this expression the Cartesian equation of a three dimensional line can be extracted as

$$\frac{x - b_1}{a_1 - b_1} = \frac{y - b_2}{a_2 - b_2} = \frac{z - b_3}{a_3 - b_3} \tag{3.16}$$

In a case, where a cutting plane in the x direction with an equation of $x = c$ is intersected with a three dimensional line with start and end points $P_1(a_1, a_2, a_3)$ and $P_2(b_1, b_2, b_3)$, the x value of the intersection point will be equal to c since the intersection point must be positioned on the cutting plane. Therefore, the intersection point can be found by putting the $a_1, a_2, a_3, b_1, b_2, b_3$ and c values to the Equation 3.16. When the necessary operations are performed the resultant point $P(x, y, z)$ can be expressed as

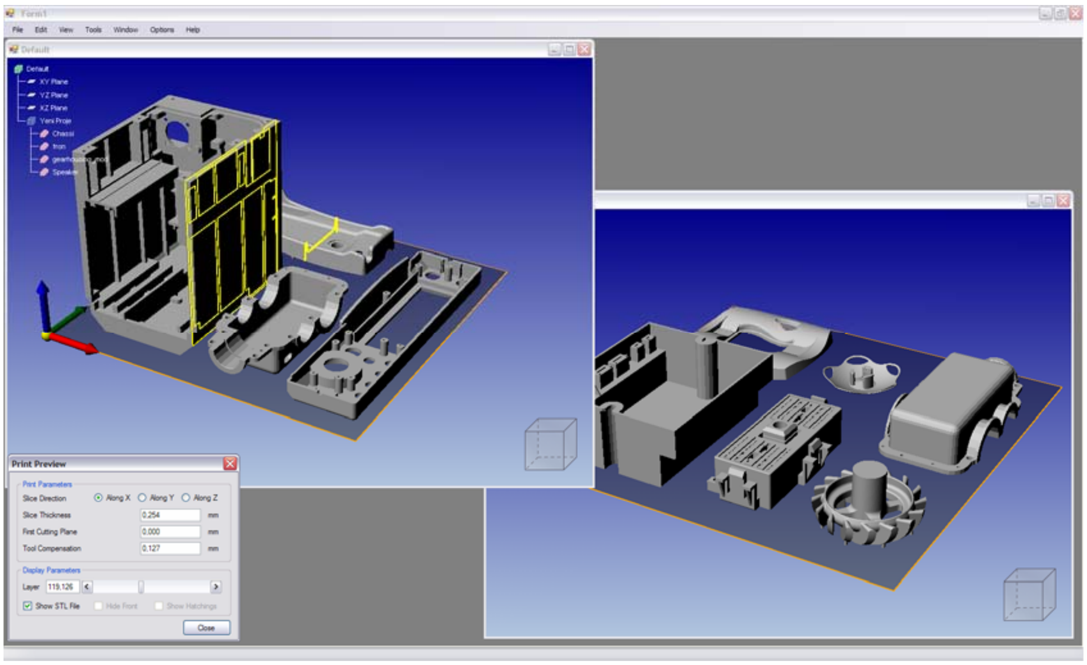
$$\begin{aligned} x &= c \\ y &= \frac{(a_2 - b_2)(c - b_1)}{a_1 - b_1} + b_2 \\ z &= \frac{(a_3 - b_3)(c - b_1)}{a_1 - b_1} + b_3 \end{aligned} \tag{3.17}$$

The same operations can be performed for other cutting directions to find the intersection lines between the cutting planes in different directions and the triangles.

3.5.3.1.2 Preview Slicing

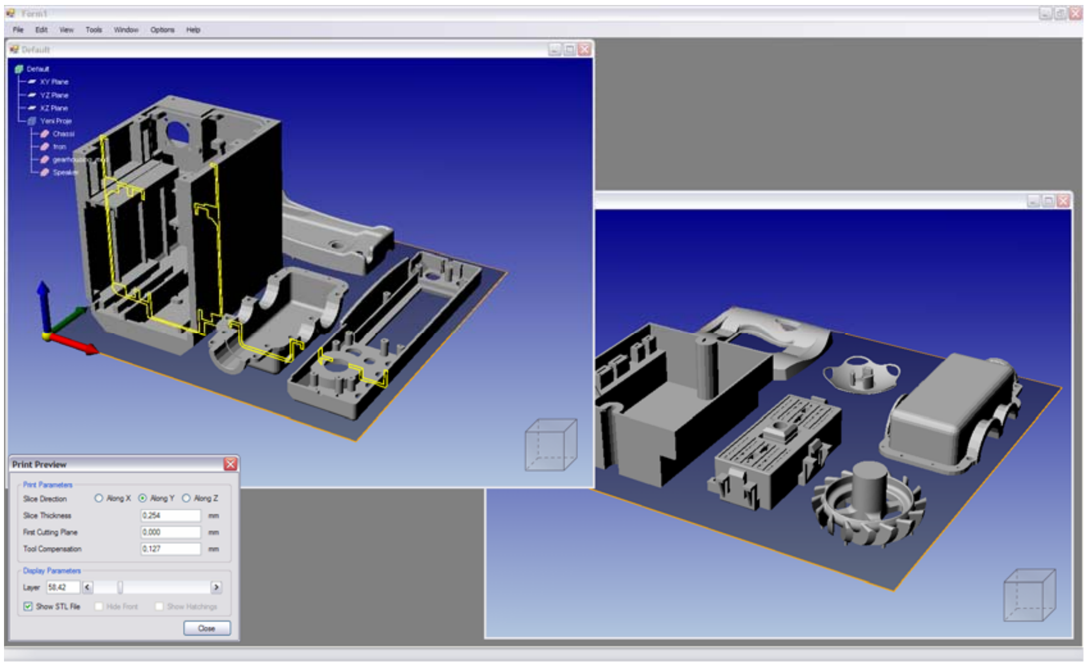
The first functionality using a slicing operation is the slice preview. In the slice preview, the parts in the workspace are sliced in X, Y or Z directions in order to display a single slice at a particular width, height or depth. This feature can be used to verify the slices used in the production or to visualize the inside of an object.

In the slice preview, each triangle in the workspace is checked according to their positions. If a triangle inside the bounds of a cut is found, then a cutting plane in the given position is intersected with an edge of this triangle. After the first intersection, the other edges of the same triangle are checked for another intersection. When two edges of a triangle are cut by a single cutting plane, the first line segment of the intersection curve is obtained. Then the other triangles are checked to find another triangle laid in the cutting bounds. This procedure continues until all the triangles in the workspace are checked for an intersection. When the procedure is completed, all the intersection lines in this particular height are found in mixed order. The lines are not ordered in order not to loose time since the slice preview data is used only for visualization purposes. The contour generated at a particular position is displayed in the 3D environment (Figure 3.11, 3.12, 3.13 and 3.14).



Slice Direction [✓ Along X] Along Y Along Z
Slice Thickness 0.254 mm
First Cutting Plane 0.000 mm
Tool Compensation 0.127 mm
Layer 119.126
[✓ Show STL File] Hide Front Show Hatchings

Figure 3.11: Slice preview of a set of parts along X direction



Print Preview

Print Parameters

Slice Direction: Along X [✓ Along Y] Along Z

Slice Thickness: 0.254 mm

First Cutting Plane: 0.000 mm

Tool Compensation: 0.127 mm

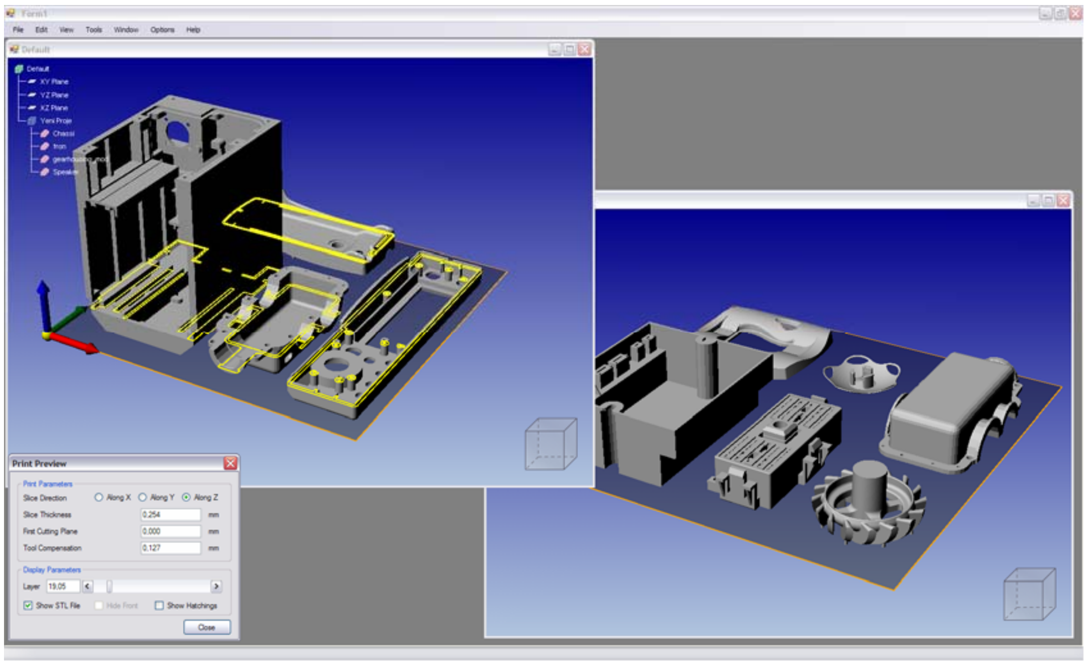
Display Parameters

Layer: 58.42

[✓ Show STL File] Hide Front Show Hatchings

Close

Figure 3.12: Slice preview of a set of parts along Y direction



Print Preview

Print Parameters

Slice Direction: Along X Along Y [✓ Along Z]

Slice Thickness: 0.254 mm

First Cutting Plane: 0.000 mm

Tool Compensation: 0.127 mm

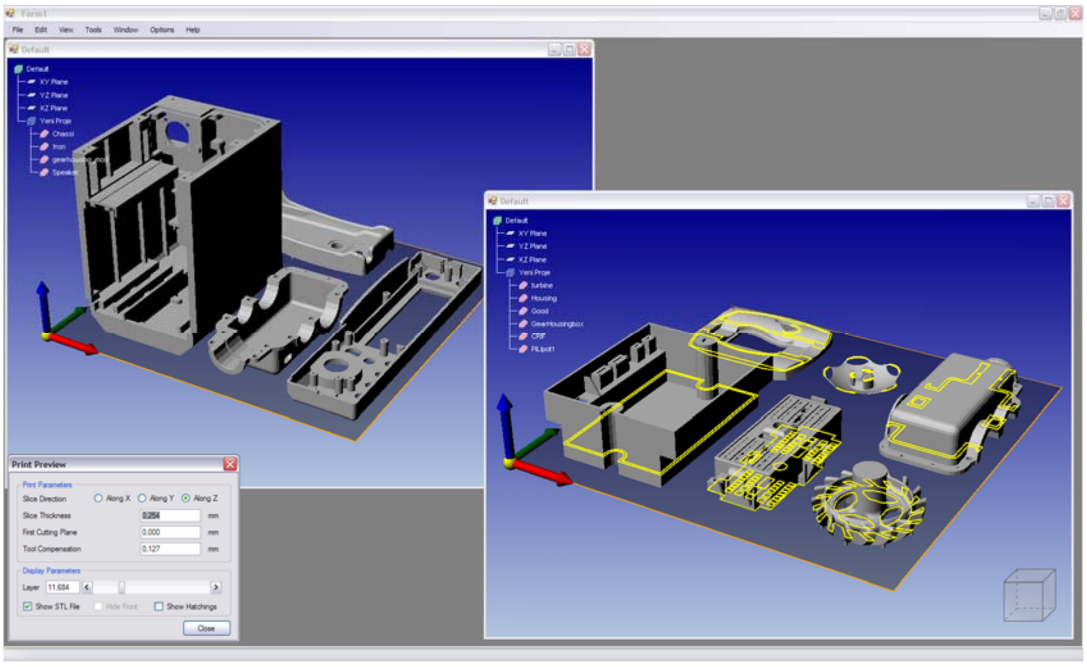
Display Parameters

Layer: 19.05

[✓ Show STL File] Hide Front Show Hatchings

Close

Figure 3.13: Slice preview of a set of parts along Z direction



Print Preview

Print Parameters

Slice Direction ( ) Along X ( ) Along Y [✓ Along Z]

Slice Thickness 0.254 mm

First Cutting Plane 0.000 mm

Tool Compensation 0.127 mm

Display Parameters

Layer 11.684

[✓] Show STL File [ ] Hide Front [ ] Show Hatchings

Close

Figure 3.14: Slice preview of a different set of parts along Z direction

3.5.3.1.3 Production Slicing

In the production slicing, instead of finding a single slice in a particular position, all the slices in the workspace must be found to generate the slice contours that are used in the tool path generation step. Therefore, the production slicing is a more time consuming procedure, compared to the slice preview.

In finding a single slice, all the triangles of the parts are searched to find the triangles intersecting with a particular cutting plane. This searching operation consumes a considerable amount of time. Even worse, in a complex part with millions of triangles, thousands of slice contours must be generated by searching the entire triangle list for each slice which means an incredible amount of process time. Therefore, methods like partitioning of triangles according to their positions or finding the adjacent triangles to apply a marching algorithm have been developed in different studies to solve this problem. Although, these methods decrease the process times in considerable amounts, they still have disadvantages like unnecessary searching and high memory usage.

When the method of finding the adjacency of triangles is examined, it can be seen that the searching operation is performed only to find the adjacency of triangles. Once the adjacency of triangles is found, the remaining slicing operation marches through all the triangles by using the shared edges. However, still a huge amount of time is consumed for searching operation to find the adjacency of triangles. In addition, more than one call for a single triangle must be performed since in most of the cases more than one cutting planes cut the same triangle. On the other hand additional memory is allocated to hold the adjacency information of triangles so the memory usage increases.

In the method of triangle partitioning, triangles are partitioned into the buckets according to their heights in the cutting direction, so the list of triangles corresponding to a cutting plane is known before starting the slicing operation. This partitioning operation can be performed in a single search, therefore, the search time and the number of triangle calls decreases considerably compared to the method of finding adjacency of triangles. However, for a triangle cut by different cutting planes, the reference of a single triangle must be held in more than one bucket. As a result, when millions of triangles are considered, a huge amount of memory is required to hold the partitioning information. Additionally, the partitioning only provides information about the relations between the intersection lines and the slices. The triangles in the buckets are cut by the cutting planes in a random order. Hence, after the slicing operation, another time consuming search operation must be performed to order the intersection lines.

In production slicing feature, a third method different than the ones described above is used to generate slice contours. In this method, all the cutting planes correspond to a randomly chosen triangle and the intersections between this triangle and cutting planes are calculated in a single call (Figure 3.15). Therefore, a reverse approach is applied where the triangle is decided randomly and all the cutting planes are calculated rather than searching an item in an existing list. In the other methods, the cutting plane is decided first and then a search operation is performed among a huge triangle list to find an intersection. When the process times of searching a particular triangle among millions of triangles and calculating the positions and IDs of at most tens of cutting planes are compared, it can be seen that summation operation takes significantly less time than searching. Therefore this method consumes considerably less time compared to the other two methods. In addition, in this method no additional memory is used to hold partition buckets or adjacency information.

In this method each cutting plane is identified with an integer slice ID. This slice ID is calculated with the cutting plane by using the first cutting plane, slice thickness value and the minimum and maximum bounds of the triangle.

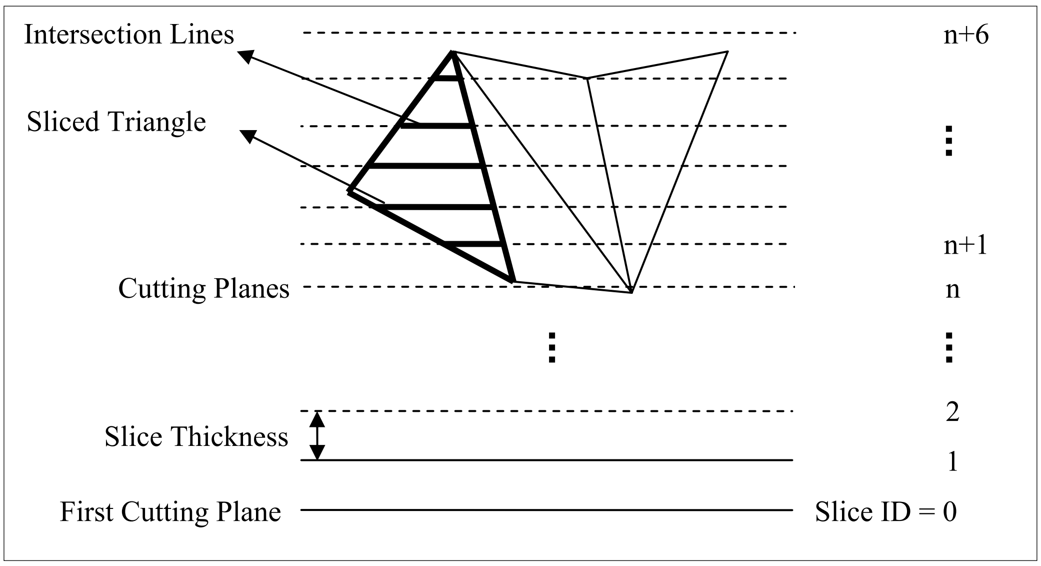
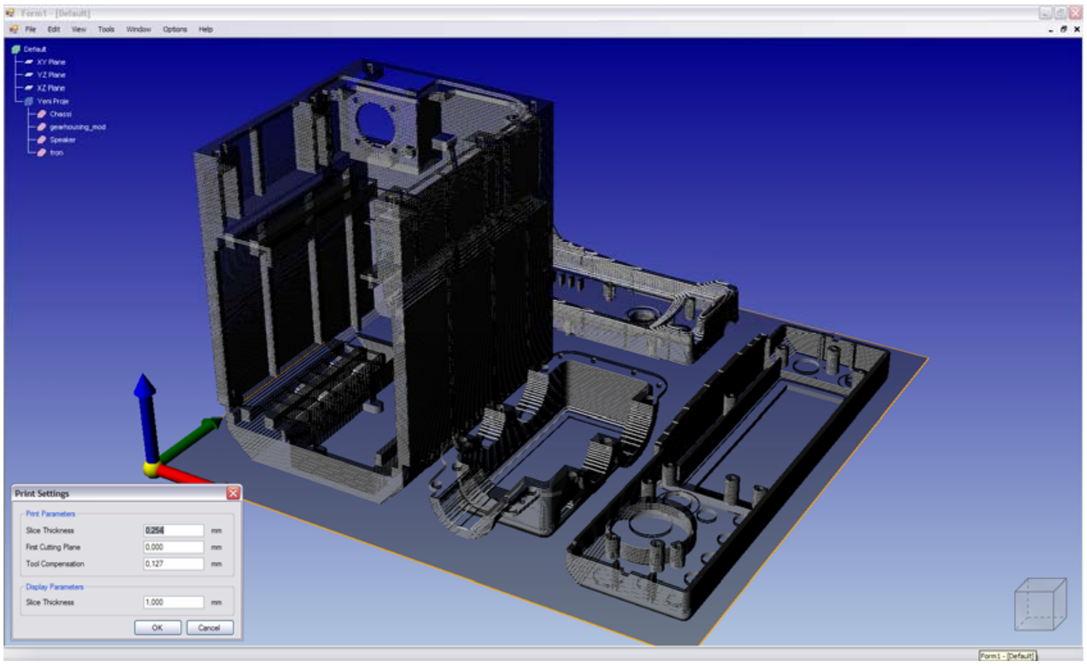


Figure 3.15: View of a set of parts after a production slicing operation.

After the slicing operation is performed, like in the triangle portioning method, the intersection lines must be ordered to form the contour curves. In this ordering the slice information of the lines and their start and end points are used. The first search process is performed to place intersection lines into slice lists and the second search is performed among the slice lists to order lines. However, these searches take less time than a portioning search, since only an equality check is performed to find order lines into slice. When, the slicing operation is completed the resultant slice contours are displayed in the 3D environment (Figure 3.16).

3.5.3.2 Tool Path Generation

The generation of the tool path is the final process planning step before fabricating a part. In this stage, the contour curves generated by the production slicing functionality are used as the input. The generated contour curves are also used as the tool paths to fabricate the outer bounds of each layer. To complete the CAM code generation, the inside of these contours must be filled by raster method to form the final tool paths. The tool paths generated after raster process must be as continuous as possible and avoid any unnecessary direction changes. The consecutive raster segments must be alternated in direction to improve the physical properties of the fabricated part. If one layer of raster segments is laid in the horizontal direction, the next layer of raster segments must be laid in the vertical direction. Between the layer changes the continuity of the deposition of the production material can be an important concern in some production techniques. Therefore it is better to design the tool paths as continuous as possible in the layer changes, as well. This can be achieved by linking the end of a raster segment to the beginning of the next one and oppositely orienting the direction of the raster segments of consecutive layers (Gültekin, 2003).



Print Parameters

Slice Thickness	0.254	mm
First Cutting Plane	0.000	mm
Tool Compensation	0.127	mm

Display Parameters

Slice Thickness	1.000	mm

Figure 3.16: View of a set of parts after a production slicing operation.

The raster segments are generated for one contour in a time. Tool path generation of a contour is started when the whole tool path generation process of a previous contour is completed. The raster segments are generated by intersecting the closed slice contours with equally spaced parallel infinite rays laid along a common direction. The process of generating tool paths by using these parallel infinite rays is also called hatching. The rays can be generated in two different directions, x and y. These two directions are called hatching directions (Figure 3.17). The hatching direction is alternated in subsequent contours.

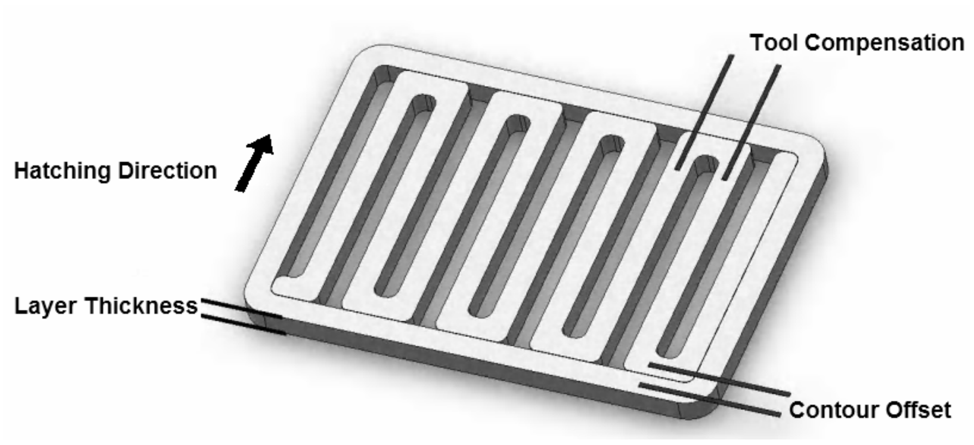


Figure 3.17: Hatching parameters and hatching direction of a contour.

Two parameters, called tool compensation and contour offset, are used in the hatching operation. Tool compensation value defines the distance between the parallel infinite rays. Contour offset value defines the distance between the internal raster segments and the inner and outer contours. Tool compensation and contour offset values are set at the beginning of the process and the same values are used throughout a single tool path generation step. On the other hand, different values can be used in different tool path generation operations. These values and their physical meanings can be seen in Figure 3.17.

Hatching operation starts with the creation of new offset contours. The existing contour curves are regenerated inside the outer and outside the inner contours with a distance equal to contour offset. When all the offset contours are generated, the minimum and maximum values of all new contours in the perpendicular direction of hatching are found. These values are used to form a hatching interval. The parallel infinite lines are generated in this interval with equal spacing defined by tool compensation value. Afterward, the intersections between the rays and offset contours are calculated and the resultant intersection points are hold in a point list. Once all the intersection points are found, the points are ordered according to ascending values of the coordinates in the hatching direction. The ordered points are used to create the line segments that fill the inside of the contours. The line segments are connected to their neighboring lines at the point of intersection so the tool paths are generated (Figure 3.18). These connections are performed by obeying three intersection rules listed by Gültekin (2003).

1. A line segment can be linked to another line segment if and only if the segments are consecutive. If not, the link might cross other line segments.

2. A link is possible only between the tail and the head of a segment. This guarantees proper orientation.

3. A raster segment can be linked to another raster segment if and only if the two vertices to be connected originate from the same contour. If not, the link could overlap some existing line segments.

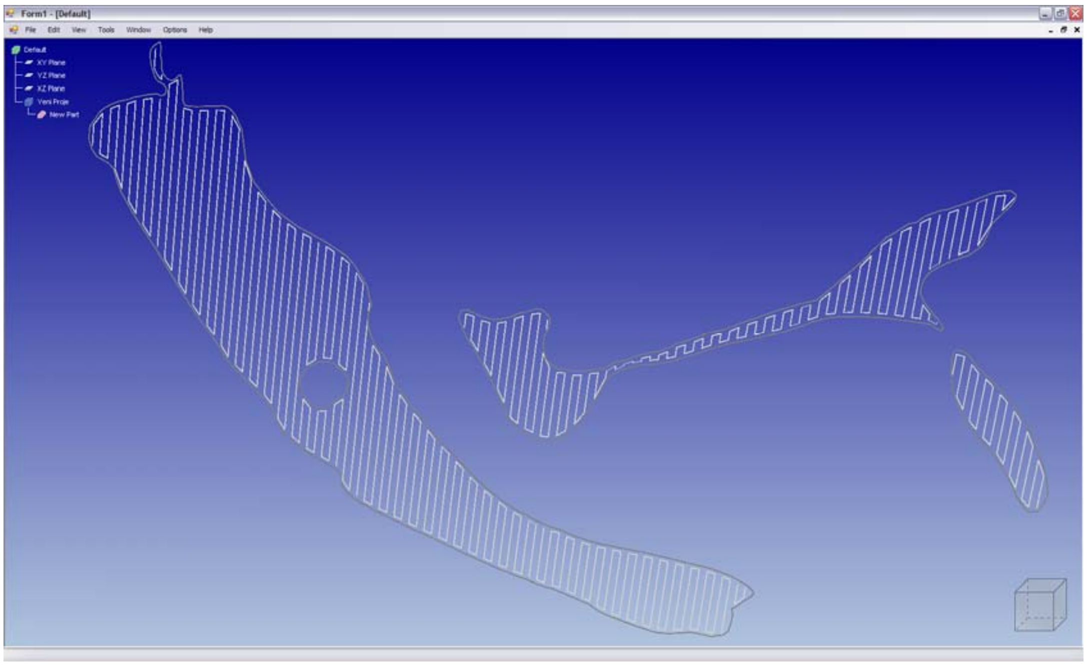


Figure 3.18: View of a contour after a tool path generation operation.

When all the line segments are connected to each other by obeying these rules, the tool path generation stage of the current contour is completed and the same steps are repeated for the subsequent contours. After all the contours in the workspace are hatched (Figure 3.19), the generated tool paths are converted into CAM code by saving this data in CLI Format or by sending the generated data to a supported RP machine through the direct printing interface provided by the RP module of the RP software.

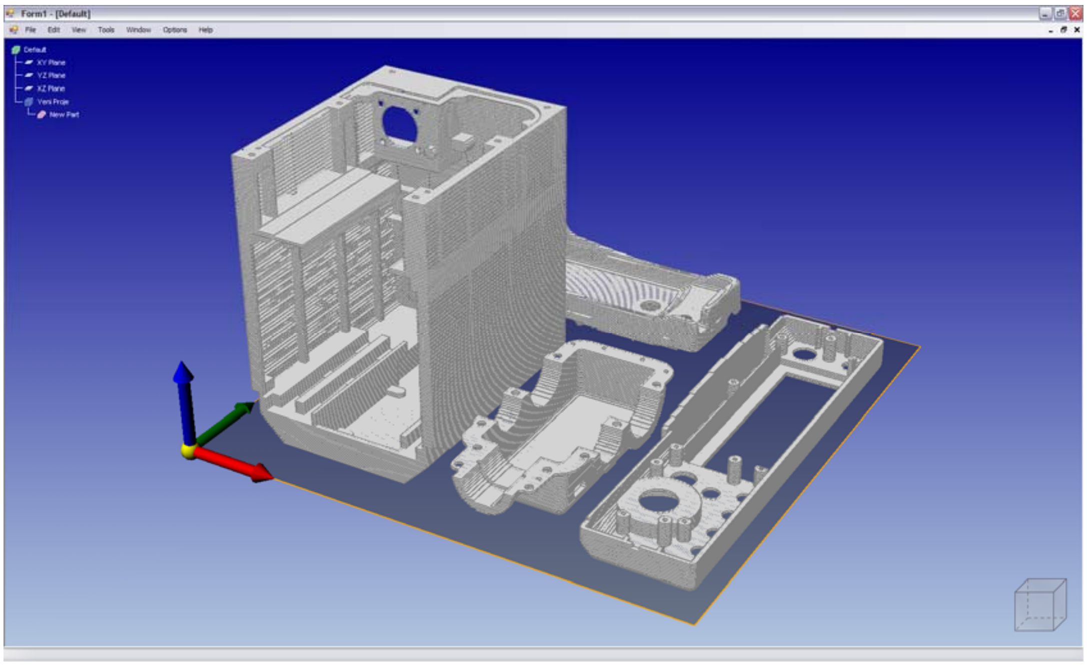


Figure 3.19: View of a set of parts after a tool path generation operation.

3.6 Software Technical Specifications

The RP software developed within the thesis is designed to be used in Microsoft Windows XP operating system. The software is implemented by using C# language and Microsoft .NET Framework 2.0. DirectX April 2007 Software Development Kit is used to support Direct3D hardware acceleration which is used in the display of 3D inputs and outputs of the software. To be able to use this software in a personal computer (PC), the target system must have Microsoft XP operating system, .NET Framework 2.0 and Managed DirectX April 2007 libraries.

Different functionalities are supported in different parts of the software. A modular architecture is used to be able to distribute different parts of the software independently. Furthermore, the modular architecture has its benefits while updating the program and adding new functionalities to the existing software. Therefore, the software is designed in four different parts as three different modules and a graphics framework. Each of these parts performs different functionalities of the software.

The software can display three dimensional objects by using a multi screen interface. Different windows can be opened simultaneously by using a single application. All the windows provide an independent 3D modeling environment with user controls like rotation, zoom and pan, display options like wireframe, flat shading and point view, predefined views like top, bottom, front, back, left, right and isometric views. In addition, each window can be displayed in full screen mode by covering the whole screen of the PC.

The software supports STL file format to import CAD data and CLI file format to export generated CAM data. Software can import multi STL files at the same time. The imported files are listed in the graphical user interfaces of their parent windows by means of their filenames. These files can be grouped in special structures called Product. The parts and products can be selected by using the graphical user interfaces of their windows. Therefore, different functions can be performed to different parts or part groups. The parts can be visualized individually in different colors. The bounding box of each part can be displayed by using the software. The software also supports undo and redo functionality to facilitate the ease of use.

A workspace and a platform are displayed by the software to define the virtual workspace of the target RP machine. An imported part can be oriented in this virtual workspace by translating the part absolutely or relatively to a given position and by rotating the part around X, Y and Z axes with a given angle. Moreover, sizes of the parts can be modified by scaling the part uniformly or non-uniformly along X, Y and Z directions. The oriented parts are automatically checked for a collision with the virtual workspace to prevent any failure in manufacturing.

The user can see the previews of the slices of the parts by using slice preview functionality. The user can verify the slicing operation and slicing parameters before generating the CAM code or see the interior sections of the parts by using this functionality. The parts in the workspace can be previewed in three directions, X, Y and Z. The user can display any slice in any position by using a scroll bar or by entering the position of the required slice. The user can also choose to display the hatches of the slice contours.

The CAM data generation is automatically performed by the software. The user only sets the required parameters; tool thickness, first cutting plane, tool compensation and contour offset. The user can change or use the default values of these parameters before tool path generation. The process parameters do not have soft limits. Therefore, the user can set any value to these parameters. However, in large data, the small parameter values may results in high computation times and high memory and processor usage.

The user can control the slicing operation through the slice thickness and first cutting plane parameters. The user can change the start point of slicing operation and the thickness of the resultant layers by changing these parameters. The slicing operation is performed uniformly. All the parts in the workspace are sliced by using a single slice thickness value provided by the user.

The user can control the hatching operation through the contour offset and tool compensation parameters. These parameters can be used for different purposes like manufacturing porous parts or decreasing the production times by rarefying the tool paths in the interior of the parts. By using the tool compensation value, the distances between the hatches and by using the contour offset value the distances between the contours and the hatches can be controlled. The same values are used throughout the tool path generation process. The hatches can only be spaced uniformly. The software only generates vertical and horizontal tool paths. The direction of hatching is changed by the software in subsequent layers. The software does not optimize the tool paths to improve the physical properties of the fabricated part. Therefore, these parameters must be used by considering this fact.

The generated tool paths are sent to a supported RP machine through a special architecture called direct printing interface. Support files for different machines can be written by the user by using some predefined rules of communication. The generated CAM code can also be converted into CLI format and saved to an external file to be used in any application where CLI format is required.

CHAPTER 4

DISCUSSION AND CONCLUSION

In this last chapter, the resultant RP software is compared with the other commercial RP software packages according to their memory usage and process time to verify the success of this study. After this benchmark study, the results of this comparison and achievements of the thesis study are concluded. At the end of the chapter, possible future work and further improvements are discussed.

4.1 Performance Benchmarking

A performance benchmark study based on memory usage and process time can be performed between the developed RP software and other commercial RP software packages in the market to validate the success of this thesis study.

The memory usage of the developed RP software can be tested by comparing the amount of memory used in different states with Magics and Viscam Software packages. These states can be listed as the state when the software is just initialized, the state when the software displays a set of parts, the state when the software created two different build jobs at the same time and the state when the software is generated the tool paths for a set of parts. When the memory usages of these three software packages are tabulated at these stages, the results listed in the Table 4.1 is obtained. The values listed in the Table 4.1 are taken from Microsoft Windows XP operating system taskbar when the memory usage of the program is stabilized after the processes or operations listed in the table 4.1 are completed. The set of parts used in this comparison can be found in the Appendix E. The creation of two build jobs simultaneously is given as a criterion to emphasize the importance of the multi screen interface. Since the other commercial software packages do not support multi screen interface, this state is performed by running two copy of the same application with different build jobs. In the developed RP software, this criterion is performed by using the multi screen interface. The set of parts used in this comparison can be seen in the case study provided in Appendix E.

Table 4.1: Memory usage comparison of developed RP software, Magics and Viscam

	Developed RP Software	**Magics Version 11.1**	**Viscam Version**
Initialized Software	23.580 KB	40.628 KB	13.812 KB
Opened a set of parts	31.488 KB	55.556 KB	31.612 KB
Opened more than one build job	40.348 KB	121.760 KB	71.472 KB
After tool path generation of set of parts	49.328 KB	55.656 KB	55.928 KB

The overall process performance of the developed RP software can be tested by comparing the process completion time of these three software packages. These software packages can be compared according to the time needed to open a complex file, the time needed to open multiple files and the time needed to slice a complex part. When the process completion time of these three software packages are tabulated according to these criteria, the results listed in the Table 4.2 are obtained. The process completion time of the developed RP software is measured in milliseconds by using the processor clock. The measurement is performed by implementing additional timer codes into the source code of the software. However, process completion time of the other software packages are measured by simultaneously executing an external timer application and manually stopping the timer at the end of the process. Consequently, some measurement error is generated in the values tabulated in Table 4.2.

Table 4.2: Process time comparison of developed RP software, Magics and Viscam

	Developed RP Software	**Magics Version 9.52**	**Viscam Version**
Opening a complex file[6]	0,47 sec	1,34 sec	3,20 sec
Opening a set of files	0,29 sec	0,67 sec	0,76 sec
Tool path generation of a complex part	6,42 sec	2,37 sec	4,57 sec

The memory usage and the process completion time of the developed RP software are comparable with the other commercial systems. However, this comparison can not be considered as an exact result since the development platforms of the software packages and additional functionalities my show varieties from one software package to another. The software packages discussed in the benchmark study is developed by using different libraries. The developed RP software is using .NET framework libraries and a managed memory management system that depends on the structure of C# language. On the other hand, the other two software packages are developed by using C++ language and different software libraries with an unmanaged memory management system. Therefore, the memory usage values displayed in the Microsoft Windows XP operating system taskbar can not be considered as an exact comparison criterion. In addition, the commercial software packages listed in the table have additional functionalities like STL file repair, STL file modification, support structure generation, etc. as listed in Appendix B. These additional functionalities and the methods used in measurement affect the accuracy of the memory usage and process completion time values of the software packages. However, the results displayed in Table 4.1 and 4.2 are only presented to show the similarity of the developed RP software with the other commercial RP software packages in a performance wise comparison. The comparison is not performed to compare the developed RP software in a commercial point of view. For that reason, the exact memory usage and process time values are not crucial to discuss the success of the developed RP software. Therefore, by using these values, it can be said that this study is a reasonable implementation of the functionalities discussed in this thesis study.

[6] The complex part used in this comparison is a STL file with a size of 11 926 KB and it contains 245 258 number of triangles.

4.2 Conclusion

In this study, a detailed survey on RP technology is presented. In this presentation the fundamentals and basic concepts of RP technology, its historical background and advantages, classification of RP techniques and basic concepts of RP software packages like STL data format, process planning steps of RP software are explained as detailed as possible. At the end of this RP technology presentation, a market survey of RP software packages is given to the reader to justify the requirement analysis of the developed software.

As a result of this study, a RP software package that satisfies the possible requirements of future RP studies and hence facilitate them, with a complete, fully independent, open for development and easy to use architecture is designed and implemented. This software is developed as an extendible and application independent, multipurpose software package, which can be used in different engineering studies, rather than a specific RP application.

The software package supports a 3D modeling environment with 3D hardware visualization support to be capable of displaying different types of inputs and outputs. A multi screen support is used to be able to present different information at the same time. A graphics library that facilitates the design of new graphic objects is developed to use the same application core in different engineering applications where different types of visualization requirements and 3D objects are required.

In the RP software, a modular architecture is used to take advantage of software modularity in the design, development, release and update of the resultant software package. The software modules are organized according to their functionality.

A core module that supports a generic file transfer system is designed to facilitate the use of different file types, which can be required in a future study. This interface also supports multi file import and export. In the software, more than one file can be visualized at the same time. Since more than one part is imported, a part organization system is developed to be able to group parts. Part selection functionality is implemented to be able to use different functionalities for different parts. In addition to these, undo/redo functionality is provided for the sake of easy usage.

A RP software package that satisfies the process planning steps of RP is designed and implemented. As a part of this software package, modules with part modification, part transform, part slicing and tool path generation functionalities are developed. In the current phase, the support generation step is neglected since it is considered as a future improvement. Functionalities that facilitate easy modification and orientation of parts are provided to the user as a part of the RP module. By using this module parts can be scaled uniformly or non-uniformly, moved absolutely or relatively and rotated around three different directions. An additional functionality like part slicing in X, Y and Z directions to preview the production or to see the interior of the model is provided in a different module called Slice module. Finally, a direct printing interface is designed and implemented as a part of the RP module to be able to support different RP machines that can be designed in future studies.

When the application is completed it is tested for different conditions by using different part files. Finally to test the performance of the developed RP software, it is compared with commercial RP applications sold in the market according to memory usage and process time. This comparison has verified the success of the RP software with acceptable memory usage and process times compared to other commercial RP applications. Therefore, it can be said that, the resultant RP software is comparable with other commercial RP applications in case of functionality and performance.

4.3 Future Work and Further Improvements

The basic functionalities that are necessary to perform a standard RP process are implemented in this version of RP software. However, some additional functionalities and modules must be added to the developed RP software to be able to use it with all RP systems in the market.

Support generation module, STL file repair and edit module, data import and export module can be implemented to the RP software in the future. Automatic and manual support generation, automatic STL file problems recognition and repair, boolean operation and 3D shape generation and direct CAD data import support for STEP and IGES formats can be listed as the functionalities that can be provided by these new modules.

In addition to these new modules, current modules can be improved to support new functionalities. Automatic part nesting, determining optimal orientation, build time and cost estimation, build animation and RP machine library supports can be listed as the functionalities that can be added to the existing RP and View Modules. Additionally, the current algorithms used in the Slice module can be improved with adaptive and direct slicing algorithms to perform more accurate slicing operations.

Some architectural properties like multi language support, a completely independent modular expansion support, exception handling and automatic system recovery support, online update and system logging support can also be developed in a newer version of the RP software.

REFERENCES

Castle Island's Worldwide Guide to Rapid Prototyping, Rapid Prototyping Equipment, Software and Materials, http://home.att.net/~castleisland/rp_int1.htm, Last accessed September 27, 2007, Last updated September 3, 2007.

Chalasani, K. L., Grogan, B. N., Bagchi, A., Jara- Almonte, C. C., Ogale, A. A. and Dooley, R. L., An algorithm to slice 3d shapes for reconstruction in prototyping systems.

Chang, W., CAD/CAM for the Selective Laser Sintering Process, M.S Thesis, University of Texas, Austion, TX, 1989.

Chari, J. K. and Hall, J. L., Robust Prototyping, Solid Freeform Fabrication Symposium 1993, H. L. Marcus et al., eds. University of Texas, Austin, August 1993, pp. 135-142.

Chua, C.K., Leong, K.F., Lim, C.S., Rapid prototyping : Principles and Applications. World Scientific, Suite 202, 1060 Main Street River Edge, NJ 07661, 2003.

EFunda Engineering Fundamentals, Rapid Prototyping: SGC, Solid Ground Curing Process Figure, http://www.efunda.com/processes/rapid_prototyping/sgc.cfm, Last accessed September 27, 2007.

Erkut, N., Towards perfection in manufacturing: Autofabrication technologies. http://www.turkcadcam.net/rapor/autofab/index.html, Last updated June 26, 2007, Last accessed August 19, 2007.

Gültekin, Murat, Design and Development of a Rapid Prototyping Machine, Master Thesis, The Graduate School of Natural And Applied Sciences of University of Gaziantep, 2003.

Grenda, E. P., Printing the Future; The 3D Printing and Rapid Prototyping Source Book, Castle Island Corporation, 119 Webster Street Arlington, MA 02474 U.S.A., 2006.

Hart, George W., "Creating a Mathematical Museum on your Desk", Mathematical Intelligencer, 27, No. 4, Winter, 2005

Hearn, D., Baker, M. P., Computer Graphics, C Version, Second Edition, Prentice Hall, May 24, 1996.

Hornby, A. and Wehnmeier, S. (Editor), Oxford Advanced Learner's Dictionary of Current English, 6th edition, Oxford University Press, Oxford, 2000.

Kirschman, C. F. and Jara-Almonte, C. C., A parallel slicing algorithm for solid freeform fabrication processes, Solid Freeform Fabrication Symposium, pages 26–33, August 1992.

M2 Systems, Diagram of stereolithography machine with essential parts figure, http://www.m2-systems.com/prototyping/stereolithography.php, Last accessed September 27, 2007.

Mahir, N., Analitik Geometri; Uzayın Analitik Geometrisi, T.C Anadolu Üniversitesi Matematik Öğretmenliği Bölümü, T.C Anadolu Üniversitesi Yayınları, 1999

Marcam Engineering GmbH, Viscam RP Software Technical Specification, http://www.marcam.de/Eng/default.htm, Last accessed September 27, 2007.

Marsan, A. L. and Dutta, D., Survey of Process Planning Techniques for Layered Manufacturing, Proceedings of DETC'97, 1997 ASME Design Engineering Technical Conferences, September 1997, 14-17,

Marsan, A. L., Kumar, V., Dutta, D., Pratt, M. J., An assessment of data requirements and data transfer formats for layered manufacturing. U.S. Department of Commerce, NISTIR 6216, September 1998.

Materialise Company, Magics RP Software Technical Specification, http://www.materialise.com/materialise/view/en/240063-Magics+brochure+pdf.html, Last accessed September 27, 2007.

Metelnick, J., How today's model/prototype shop helps designers use rapid prototyping to full advantage, Society of Manufacturing Engineers Technical Paper, pages MS91–457, 1991.

Microsoft DirectX Development Team, DirectX 9.0 Programmer's Reference, Microsoft Corporation, August 2005.

Miller, T., Managed DirectX 9 Kick Start: Graphics and Game Programming, Sams Publishing, Inc., 800 East 96th Street Indianapolis, IN 46240 USA, October 22, 2003.

Miller, T., Beginning 3D Game Programming, Sams Publishing, Inc., 800 East 96th Street Indianapolis, IN 46240 USA, December 3, 2004.

Pham, D. T., and Gault, R. S., A comparison of rapid prototyping Technologies, International Journal of Machine Tools and Manufacture, 38 (10):1257–1287, October 1998.

Proceedings of the 1991 ASME Computers in Engineering Conference, pages 209–216, August 1991.

Robinson, S., Nagel, C., Glynn, J., Skinner, M., Watson, K., Evjen, B., Professional C#, Third Edition, Wiley Publishing, Inc., 10475 Crosspoint Boulevard Indianapolis, IN 46256 USA, 2004.

Rock, S. J. and Wozny, M. J., Utilizing topological information to increase scan vector generation efficiency, Solid Freeform Fabrication Symposium, pages 28–36, August 1991.

Sezginalp, E., Simultaneous Localization and Mapping for an Outdoor Mobile Robot, Master Thesis, The Graduate School of Natural And Applied Sciences of Middle East Technical University, 2007.

Tata, K. and Fadel, G., Feature Extraction from Tessellated and Sliced Data in Layered Manufacturing, Solid Freeform Fabrication Symposium 1996, H. L. Marcus et al., eds. University of Texas, August 1996, pp. 587-595.

Troelsen, A., Pro C# 2005 and the .NET Platform, Third Edition, A Press, 2560 Ninth Street, Suite 219, Berkeley, CA 94710, 2005

Venuvinod, P. K., and Ma, P. K., Rapid Prototyping - Laser-Based and Other Technologies, Kluwer Academic Publishers, Post Office Box 322, 3300 AH Dordrecht, The Netherlands, 2004.

Yang, D. C. H., Jou, Y., Kong, T. and Chuang J., Laser Beam Diameter Compensation for Helisys LOM Machine, Proceedings of the Sixth International Conference on Rapid Prototyping, R. P. Chartoff and A. J. Lightman, eds. University of Dayton, pp. 171-178, June 1995.

Weiss, L. E., Rapid Prototyping Process Overview, Generic Fixturing Figure, http://www.wtec.org/loyola/rp/02_01.htm, WTEC Hyper-Librarian, Last accessed September 27, 2007, Published March 1997.

Wohlers, T.T., Wohlers Report 2006; Rapid Prototyping Tooling State of the Industry; Annual Worldwide Progress Report, Wohlers Associates, Inc., New York, May 2006.

APPENDIX A

COMPARISON OF RAPID PROTOTYPING TECHNIQUES AND SYSTEMS

A detailed comparison of commercial RP techniques and systems is presented in the Table A.1. The techniques are presented with their representative vendors and acronyms. They are compared according to their general qualitative features by means of maximum build chamber size, production speed, production accuracy, surface finish and system price. Additionally, the strengths and the weaknesses of the techniques and their typical applications are provided in the table. At the end of the table, the materials used by each technique are listed with production costs.

Table A.1: Comparison of Rapid Prototyping Techniques and Systems (Castle, 2007)

Technology	**Stereolithography**	**Jetted Photopolymer**	**Selective Laser Sintering**	**Single Jet Inkjet**	**Laminated Object Manufacturing**	**Fused Deposition Modeling**	**Three Dimensional Printing**
Acronym	**SLA**	**J-P**	**SLS**	**MM**	**LOM**	**FDM**	**3DP**
Representative Vendor	**3D Systems**		**EOS GmbH**	**Solidscape**	**Solidimension**	**Stratasys**	**Z Corp.**
Maximum Build Chamber(inches)	20 x 20 x 24	11.75 x 7.3 x 8	27.5 x 15 x 23	12 x 6 x 9	6.29 x 8.26 x 5.31	24 x 20 x 24	20 x 24 x 16
Speed	average	good	average to good	poor	good	poor	excellent
Accuracy	very good	good to very good	good	excellent	fair	fair	fair
Surface Finish	very good	good to very good	good to very good	excellent	fair	fair	fair
Strengths	large part size, accuracy	accuracy and finish, office OK	accuracy, materials,	accuracy, finish, office OK	office OK, price, size	office OK, price, materials	speed, office OK, price, color
Weaknesses	post processing, messy liquids	size and weight, post processing	size and weight, system price, surface finish	speed, limited materials, part size	limited materials, finish and accuracy	speed	limited materials, fragile parts, finish
Typical Applications	•Very detailed parts and models for fit & form testing •Trade show and marketing parts & models •Rapid manufacturing of small detailed parts •Fabrication of specialized manufacturing tools •Patterns for investment casting •Patterns for urethane & RTV molding	•Very detailed parts and models for fit & form testing •Trade show and marketing parts & models •Patterns for investment casting, especially jewelry and fine items •Patterns for urethane & RTV molding	•Slightly less detailed parts and models for fit & form testing compared to photopolymer-based methods using engineering plastics •Rapid manufacturing of parts, including larger items such as air ducts •Parts with snap-fits & living hinges •Parts which are durable and provide the •Patterns for investment casting	•Most detailed parts and models available using additive technologies for fit & form testing •Patterns for investment casting, especially jewelry and fine items, especially medical devices •Patterns for urethane & RTV molding	•Somewhat less detailed parts and models for fit & form testing compared to other methods •Patterns for urethane & RTV molding •Larger patterns for sand-casting	•Detailed parts and models for fit & form testing using engineering plastics •Detailed parts for patient- and food-contacting applications •Plastic parts for higher-temperature applications •Trade show and marketing parts & models •Rapid manufacturing of small detailed parts •Patterns for investment casting •Fabrication of specialized manufacturing tools •Patterns for urethane & RTV molding	•Concept models •Parts for limited functional testing •Color models for FEA and other engineering related applications •Architectural & landscape models •Color industrial design models, especially consumer goods & packaging •Castings
System Price Range	$75K-800K	$40K-85K	$200K-1M+	$70K-80K	$15K-240K	$19K-300K	$20K-70K
Available properties& characteristics	•Acrylics (fair selection) •Clear and rigid •ABS-like •Polypropylene-like (PP) •Flexible or elastomeric •Water-resistant	•Acrylics (limited selection) •Elastomeric	•Nylon, including flame-retardant, glass-, aluminum-, carbon-filled and others providing increased strength and other properties •Polystyrene (PS) •Elastomeric •Steel and stainless steel alloys •Bronze alloy •Cobalt-chrome alloy •Titanium	•Polyester-based plastic •Investment casting wax	•Bonded PVC-based plastic film •Bonded paper	•ABS •Polycarbonate (PC) •Polyphenylsulfone •Elastomer	•Bonded plaster / plaster composite •Elastomeric •Investment & direct casting
Material Costs $/pound							
plastics	$75-110	$60-200	$25-60	$100	$18	$115-185	
metal			$35-115				
other			$5 (foundry sand)		$5-8 (paper)		starch: $0.35 / cu in plaster: $0.60 / cu in + infiltrant

APPENDIX B

RAPID PROTOTYPING SOFTWARE MARKET SURVEY

In Appendix B.1 and B.2, the technical specifications of Viscam and Magics software are given, respectively.

B.1 Viscam Software Technical Specifications

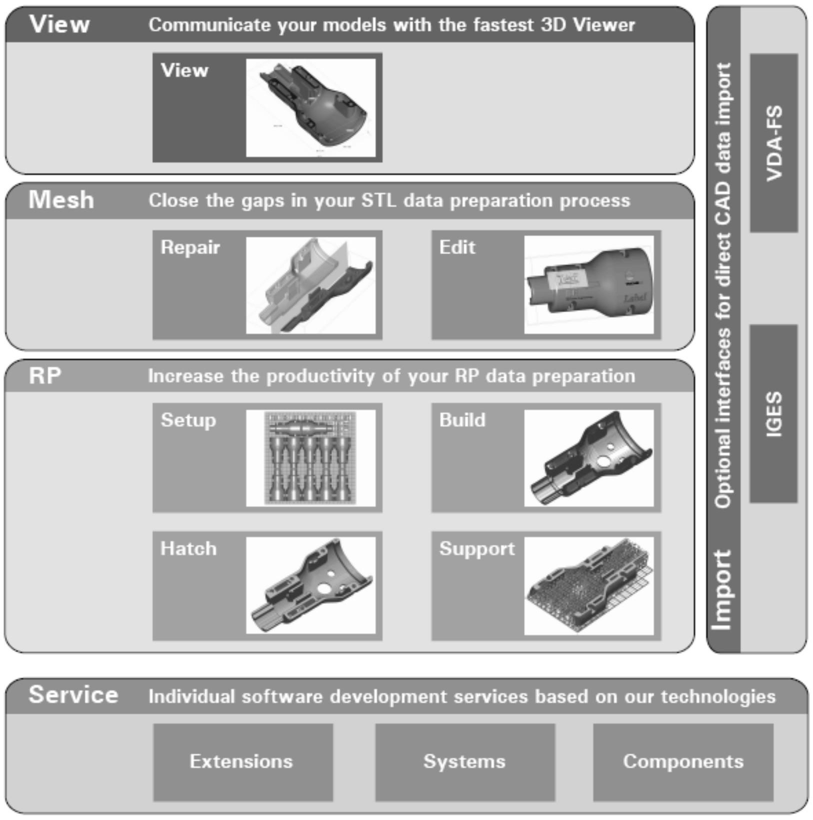
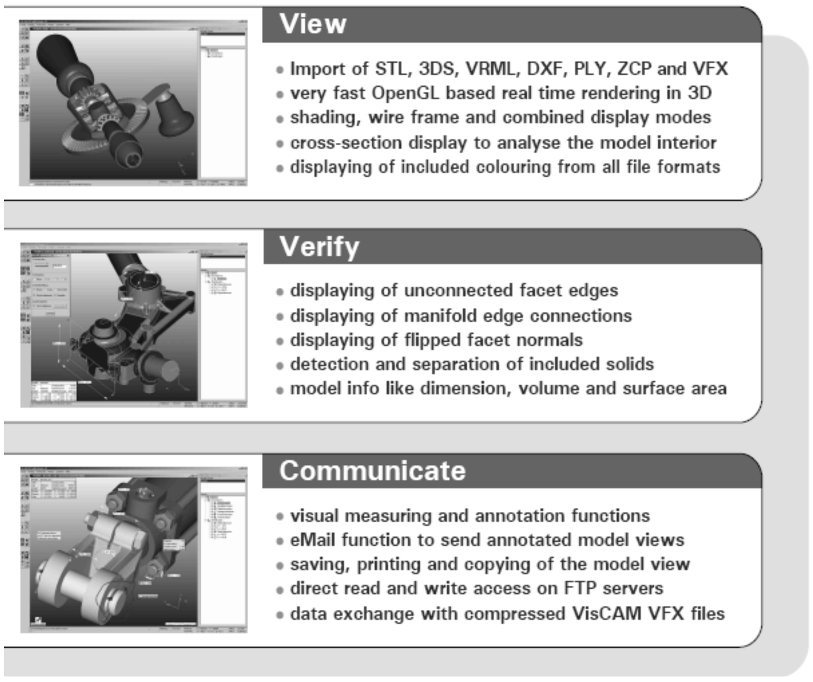

View
Communicate your models with the fastest 3D Viewer
View
Mesh
Close the gaps in your STL data preparation process
Repair
Edit
RP
Increase the productivity of your RP data preparation
Setup
Build
Hatch
Support
Import
Optional interfaces for direct CAD data import
VDA-FS
IGES
Service
Individual software development services based on our technologies
Extensions
Systems
Components

Figure B.1: The modular architecture of Viscam Software (Marcam, 2007)



View

- Import of STL, 3DS, VRML, DXF, PLY, ZCP and VFX
- very fast OpenGL based real time rendering in 3D
- shading, wire frame and combined display modes
- cross-section display to analyse the model interior
- displaying of included colouring from all file formats

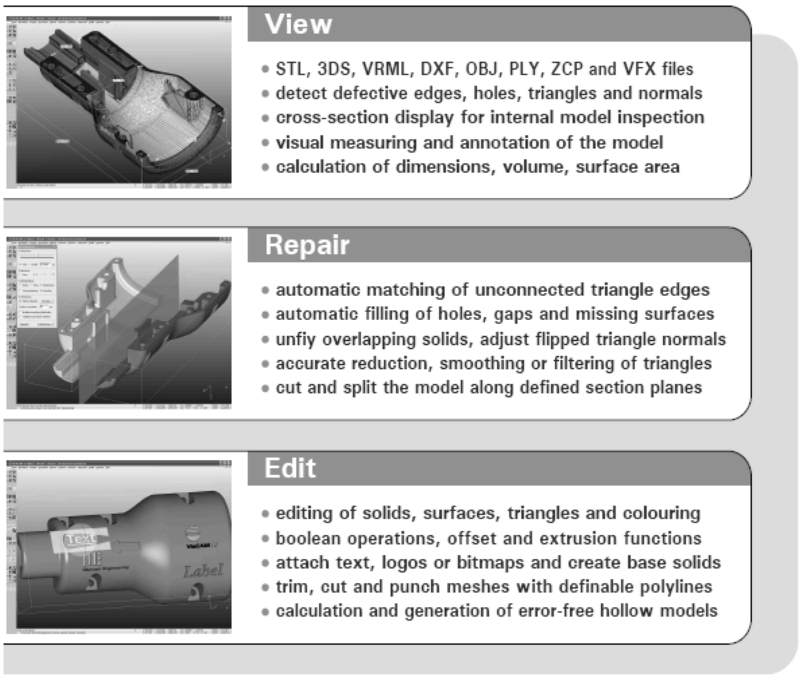
Verify

- displaying of unconnected facet edges
- displaying of manifold edge connections
- displaying of flipped facet normals
- detection and separation of included solids
- model info like dimension, volume and surface area

Communicate

- visual measuring and annotation functions
- eMail function to send annotated model views
- saving, printing and copying of the model view
- direct read and write access on FTP servers
- data exchange with compressed VisCAM VFX files

Figure B.2: Functionalities of View module of Viscam Software (Marcam, 2007)



View

- STL, 3DS, VRML, DXF, OBJ, PLY, ZCP and VFX files
- detect defective edges, holes, triangles and normals
- cross-section display for internal model inspection
- visual measuring and annotation of the model
- calculation of dimensions, volume, surface area

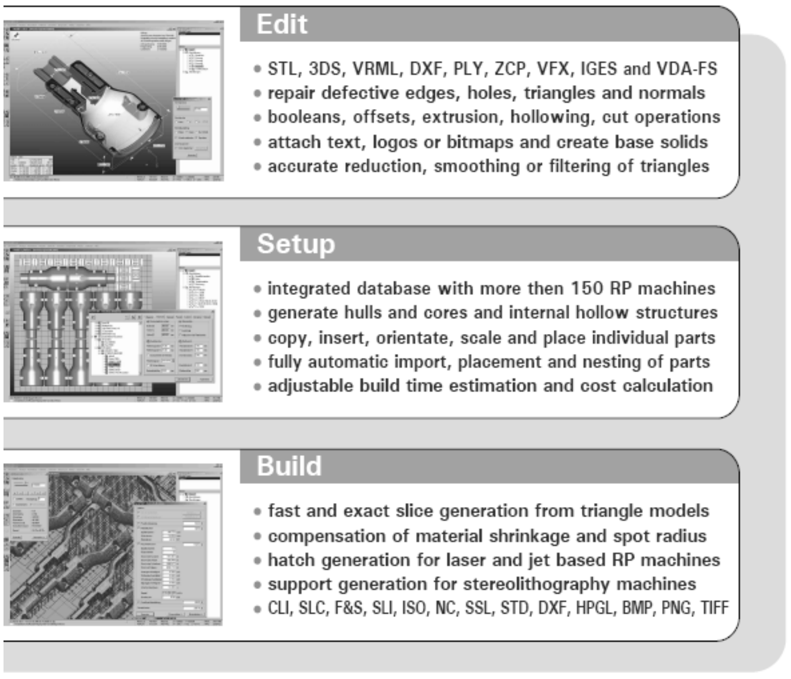
Repair

- automatic matching of unconnected triangle edges
- automatic filling of holes, gaps and missing surfaces
- unfiy overlapping solids, adjust flipped triangle normals
- accurate reduction, smoothing or filtering of triangles
- cut and split the model along defined section planes

Edit

- editing of solids, surfaces, triangles and colouring
- boolean operations, offset and extrusion functions
- attach text, logos or bitmaps and create base solids
- trim, cut and punch meshes with definable polylines
- calculation and generation of error-free hollow models

Figure B.3: Functionalities of Mesh module of Viscam Software (Marcam, 2007)



Edit

- STL, 3DS, VRML, DXF, PLY, ZCP, VFX, IGES and VDA-FS
- repair defective edges, holes, triangles and normals
- booleans, offsets, extrusion, hollowing, cut operations
- attach text, logos or bitmaps and create base solids
- accurate reduction, smoothing or filtering of triangles

Setup

- integrated database with more then 150 RP machines
- generate hulls and cores and internal hollow structures
- copy, insert, orientate, scale and place individual parts
- fully automatic import, placement and nesting of parts
- adjustable build time estimation and cost calculation

Build

- fast and exact slice generation from triangle models
- compensation of material shrinkage and spot radius
- hatch generation for laser and jet based RP machines
- support generation for stereolithography machines
- CLI, SLC, F&S, SLI, ISO, NC, SSL, STD, DXF, HPGL, BMP, PNG, TIFF

Figure B.4: Functionalities of RP module of Viscam Software (Marcam, 2007)

B.2 Magics Software Technical Specifications

The Information provided in this section is taken from the (Materialise, 2007).

Magics Base

Magics gives a control over STL-files among the offered functionality:

- Visualization, measuring and manipulation of STL Files
- Fixing STL files, uniting shells, trimming surfaces, double triangle detector
- Cutting STL files, punching holes, extruding surfaces, hollowing, applying offset
- Boolean operations, triangle Reduction, smoothing, labeling
- Nesting, collision detection
- Coloring STL-files

Magics RP

This version offers some extra Rapid Prototyping functions like:

- The platform concept, Build time estimation, Quotation Making
- Slice verification
- Z-compensation

Modules "IGES", "VDA", "STEP", "UG", "Pro/E", "CATIAV4.2x" "Catia V5" and "Slice to STL" import

Magics is compatible with all major CAD formats like IGES, VDA, STEP, Unigraphics, Pro/E and Catia (V4.5x and V5) and STL. In combination with the STL fixer, Magics enables you to send any file to the RP system or use it in a tooldesign.

Module "Pointcloud-import"

Point-cloud import allows Magics to import directly the pointclouds generated by 3D scanners. Magics will triangulate the pointclouds by connecting the points by triangles. A wide range of parameters is available, allowing you to optimise the triangulation. It is also possible to import VDA pointclouds.

Module "Support Generation"

Support generation is one of the key issues for stereolithography and metal sintering. Fast and easy generation of support structures is crucial but also the verification and adaptation of these generated supports is essential for the final quality of the part. Magics offers several support types and combinations of these different support structures on one surface.

Module "SG Volumes"

Sand parts are fragile when they are lifted out of the build envelope. The volume supports avoid that the part breaks or drifts away. The volume supports give extra stability to the part and large overhangs. The part and supports are automatically placed on a sintered platform to enable easy lifting of the built construction.

Modules "C-Tools" and "Slice"

The Slice module writes out files for 3D systems, EOS, Stratasys and Sanders. Slices are automatically repaired and before the slicing is done, the slice preview allows you to inspect the slices. The Ctools module writes out contours and hatching for 3D Systems SLA 250 machines in the SLI format.

Module "RapidFit"

Quality control is an important issue in the Rapid prototyping industry, speed too. With Magics RapidFit you can quickly and semi-automatically create fixtures (» supports) which can be produced with any rapid prototyping technique. These supports are constructed so they can be placed on so called base plates or beams. This setup can be used as measuring caliber, supports for easy measuring or for avoiding deformation when storing the part.

Module "Tooling"

RP Machine constructors are offering techniques to produce mold inserts. However the tool design is still an elaborate work. Rapid Tooling software allows you to create the tool directly from the STL design in a matter of minutes. The parting line is automatically generated. Tooling elements and draft are easily added.

Module "EDM"

The EDM module is included in the tooling module. It is specifically designed for a fast and easy design of electrodes for spark erosion. Reference points can be defined to determine coordinates for the setup of the EDM machine. A report file helps managing the spark erosion project.

Module "Tooling Expert"

With this module quotes for tools for the injection molding process can be made in a fast, reliable and consistent way. In a 1-2-3 approach, a complete analysis of the part is done (step 1), a reliable price estimation is obtained (step 2) and a well-documented quote is automatically generated (step 3).

Module "Remesh"

Optimize your STL models for FEA purposes. Magics' Remesh Module enables to quickly and easily transform badly shaped triangles into more or less equilateral triangles. The more geometrically 'regular' the triangles are, the better and more reliable the results of the FEA calculations will be. Apart from the automatic remeshing option, different techniques are available to further improve the quality of the triangles manually.

Module “SmartSpace”

SMART – Save Material And Reduce Time. SmartSpace assures an optimal load for your sintering machine and this in a very easy and fast way. Considering the parts’ geometry, the software automatically nests your parts, maximising the number of parts in the build envelope and/or minimising the build time. At the same time, the software ensures that none of the parts collides with another part nor the container.

Module “Hearing Aid”

The purpose of the Hearing Aid module is to automate the platform preparation for the production of Hearing Aid shells. With this in mind, the module combines existing Magics functionality in a one-button solution.

Module “2D Drawing”

Generate powerful feedback on your STL models by creating and printing 2D drawings. This module offers fast and easy 2D drawing generation from even the most complex STL models. You can generate all types of views in both 1st angle and 3rd angle projection. Adding and editing measurements and annotations takes only seconds and the title blocks are completely customizable.

Module “ActiveX”

Use the ActiveX interface to integrate Magics in your business process. The ActiveX module enables direct communication between your software applications (like databases, quoting programs, web applications) and Magics. Using the ActiveX interface, Magics can provide these applications with all the imaginable parameters and pictures of your STL-files. This will bring the automation of your business processes to a higher level: saving you time and eliminating human errors.

APPENDIX C

REQUIREMENT ANALYSIS OF RP SOFTWARE

View Module

Visualization

- Hardware acceleration: support of Direct3D
- Rotation, zoom and pan of parts.
- Flat shading
- Individual visualization of parts
- Cross-section display to analyze the model interior
- Visualization of included color information

File Input/output

- STL file format input.
- Multiple file reading

View-modes

- Flat shading (opaque), triangle (the STL-file), and point view.
- Part bounding box: shows the bounding box around the part
- Full screen display.
- Multi screen support
- Different parts can be shown in different colors

Tools

- UNDO and REDO functions

RP Module

Hardware Communication

- Direct printing interface for the Hermes.

Orientation

- Easy orientation of the part on the workspace.
- Parts can be moved absolutely and relatively
- Parts can be rotated around three axes.

Modification

- Parts can be scaled uniformly.
- Parts can be scaled in three different directions

Slice Module

Slicing

- Fast and exact uniform slice generation from triangle models
- No slice thickness limitation
- Section in X,Y,Z planes

Tool Path Generation

- Fast and exact tool path generation from slices

APPENDIX D

DETAILS OF INTERFACES AND STRUCTURAL CLASSES

```
/// <summary>
/// Define the methods required to draw an object in graphics engine
/// </summary>
public interface IDrawable
{
        /// </summary>
        /// Method that will be called when device is lost
        /// </summary>
        void Dispose();
        /// <summary>
        /// Method used to render the object in a 3D environment
        /// </summary>
        /// <param name="device">Rendering device</param>
        /// <returns>State of the rendering process</returns>
        bool Render(Device device);
        /// <summary>
        /// Method that will be called when device is reset
        /// </summary>
        /// <param name="device">Rendering device</param>
        /// <returns>State of the rendering process</returns>
        bool Reset(Device device);
}
```

```
/// <summary>
/// Define the methods required to create a selectable object
/// </summary>
public interface ISelectable
{
        /// </summary>
        /// Required to set or get selection information
        /// </summary>
        bool IsSelected { get; set; }
        /// </summary>
        /// Required to set or get the selection color
        /// </summary>
        Color SelectionColor { get; set; }
}

/// <summary>
/// Define the rules of reading a file and adding it to Graphics Framework
/// </summary>
public interface IReadable
{
        /// </summary>
        /// Reads a file and returns it as a feature type object
        /// </summary>
        /// <param name=" targetLocation ">Location of the target file</param>
        /// <returns>Read feature object</returns>
        Feature Read(string targetLocation);
        /// <summary>
        /// Gets supported file extension as a string
        /// </summary>
        string FileExtension { get; }
}
```

```
/// <summary>
/// Define the rules of writing a object to a file
/// </summary>
public interface IWriteable
{
        /// </summary>
        /// Writes a the given object to a file
        /// </summary>
        /// <param name=" object ">Object to be write to the file</param>
        /// <param name=" targetLocation ">Location of the target file</param>
        /// <returns>Status of the operation</returns>
        bool Write(Feature object, string targetLocation);
        /// <summary>
        /// Supported file extension
        /// </summary>
        string FileExtension { get; }
}

/// <summary>
/// Defines the rules of communication with an external application
/// </summary>
public interface IConnection
{
        /// </summary>
        /// RP software starts connection by using this method
        /// </summary>
        /// <returns>Status of the process</returns>
        bool OpenConnection();
        /// <summary>
        /// Sends toolpath as an Vector3 array to the external application
        /// </summary>
        /// <param name="device">Tool path</param>
        /// <returns>Status of the process</returns>
        bool SendCode(Vector3[] toolPath);
}
```

```
/// <summary>
/// Base class for all drawable colored objects. When a class is inhereted from this class, it
/// must set vertices and(optionally) indices of this base class. This can be easily made by
/// adding ": base(vertices, vertexType, vertexFormat)" or ": base(vertices, vertexType,
/// vertexFormat, indices, indexType)" /// statements to constructor of inhereted class or by
/// using "SetVertices(Array vertices, Type vertexType, VertexFormats  vertexFormat)" and
/// "SetIndices(Array indices, Type indexType)" methods. Once vertices and (optionally)
/// indices are set then any changes made on them can be reflected to drawing by setting
/// "IsResetRequired" property as true which will yield a resetting process in base class.
/// </summary>
public abstract class Entity : IDrawable, IDisposable
{
        /// <summary>
        /// Constructor of the entity class
        /// </summary>
        public Entity();
        /// <summary>
        /// Constructor of the entity class
        /// </summary>
        /// <param name="vertices">Vertices of the entity</param>
        /// <param name="vertexType">Format of the vertices</param>
        /// <param name="vertexFormat">Type of the vertices</param>
        public Entity(Array vertices, Type vertexType, VertexFormats vertexFormat);
        /// <summary>
        /// Constructor of the entity class
        /// </summary>
        /// <param name="vertices">Vertices of the entity</param>
        /// <param name="vertexType">Format of the vertices</param>
        /// <param name="vertexFormat">Type of the vertices</param>
        /// <param name="indices">Indices of the entity</param>
        /// <param name="indexType">Type of the indices</param>
        public Entity(Array vertices, Type vertexType, VertexFormats vertexFormat, Array
                indices, Type indexType);
```

```
/// <summary>
/// Indices of entity
/// </summary>
protected Array IndexArray { get; }
/// <summary>
/// Buffer which is holding indices
/// </summary>
protected IndexBuffer IndexBuffer { get; }
/// <summary>
/// Type of the indices in the indices array
/// </summary>
protected Type IndexType { get; set; }
/// <summary>
/// Indicates whether a reset for the device is required or not
/// </summary>
protected bool IsResetRequired { get; set; }
/// <summary>
/// Gets or sets the  material color of the entity
/// </summary>
protected Color MaterialColor { get; set; }
/// <summary>
/// Type of the primitive that will be used in rendering
/// </summary>
protected PrimitiveType PrimitiveType { get; set; }
/// <summary>
/// Gets or sets the material of the entity
/// </summary>
public Material SurfaceMaterial { get; set; }
/// <summary>
/// Indicates whether indices are used or not
/// </summary>
protected bool UsingIndices { get; set; }
```

```
/// <summary>
/// Indicates whether material is used or not
/// </summary>
protected bool UsingMaterial { get; set; }
/// <summary>
/// Vertices of entity
/// </summary>
protected Array VertexArray { get; }
/// <summary>
/// Buffer which is holding vertices
/// </summary>
protected VertexBuffer VertexBuffer { get; }
/// <summary>
/// Format of the vertices in the vertices array
/// </summary>
protected VertexFormats VertexFormat { get; set; }
/// <summary>
/// Type of the vertices in the vertices array
/// </summary>
protected Type VertexType { get; set; }

/// <summary>
/// Method that will be called when device is lost. When you override this method, to
/// improve performance call disposable objects of the class if IsResetRequired is
/// false.
/// </summary>
public virtual void Dispose();
/// <summary>
/// Method used to render the object in a 3D environment
/// </summary>
/// <param name="device">Rendering device</param>
/// <returns>State of the rendering process</returns>
public virtual bool Render(Device device);
```

```
        /// <summary>
        /// Method that will be called when device is reset. When you  override this method,
        /// to improve performance call objects of the class that will be reseted if
        /// IsResetRequired is false.
        /// </summary>
        /// <param name="device">Rendering device</param>
        /// <returns>State of the rendering process</returns>
        public virtual bool Reset(Device device);
        /// <summary>
        /// Method used to set buffers of device
        /// </summary>
        /// <param name="device">Rendering device</param>
        /// <returns>State of the process</returns>
        protected bool SetDeviceBuffers(Device device);
        /// <summary>
        /// Sets the indices of the entity class
        /// </summary>
        /// <param name="vertices">Indices of the entity class</param>
        /// <param name="vertexType">Type of the indices</param>
        protected void SetIndices(Array indices, Type indexType);
        /// <summary>
        /// Sets the vertices of the entity class
        /// </summary>
        /// <param name="vertices">Vertices of the entity class</param>
        /// <param name="vertexType">Type of the vertices</param>
        /// <param name="vertexFormat">Format of the vertices</param>
        protected void SetVertices(Array vertices, Type vertexType, VertexFormats
                vertexFormat);
}

/// <summary>
/// Base class for drawable objects that needs selection, custom colorization, transparency
/// and transformation functionality. When a class is inhereted from this class, it must set
/// vertices and (optionally) indices of this base class. This can be easily made by adding
/// ": base(renderColor, vertices, indices)" statement to constructor of inhereted class or by
/// using "Vertices" and (optionally) "Indices" properties. Once vertices and (optionally)
/// indices are set then any changes made on them can be reflected to drawing by setting
/// "IsResetRequired" property as true which will yield a resetting process in base class.
/// </summary>
public abstract class Feature : Entity, ISelectable
{
        /// <summary>
        /// Constructor of the feature class
        /// </summary>
        public Feature();
        /// <summary>
        /// Constructor of the feature class
        /// </summary>
        /// <param name="renderColor">Rendering color</param>
        public Feature(Color renderColor);
        /// <summary>
        /// Constructor of the feature class
        /// </summary>
        /// <param name="renderColor">Rendering color</param>
        /// <param name="vertices">Vertices of the feature</param>
        public Feature(Color renderColor, CustomVertex.PositionNormal[] vertices);
        /// <summary>
        /// Constructor of the feature class
        /// </summary>
        /// <param name="renderColor">Rendering color</param>
        /// <param name="vertices">Vertices of the feature</param>
        /// <param name="indices">Indices of the feature</param>
        public Feature(Color renderColor, CustomVertex.PositionNormal[] vertices, int[]
                indices);

        /// <summary>
        /// Center of the object
        /// </summary>
        public Vector3 Center { get; }
```

```
/// <summary>
/// Identifier of the feature.
/// </summary>
public int Identifier { get; set; }
/// <summary>
/// Indices of feature
/// </summary>
public virtual int[] Indices { get; set; }
/// </summary>
/// Required to set or get selection information
/// </summary>
bool IsSelected { get; set; }
/// <summary>
/// Maximum bound of the object
/// </summary>
public Vector3 MaximumBound { get; }
/// <summary>
/// Maximum distance of object boundary from center of the object
/// </summary>
public float MaximumDistanceToCenter { get; }
/// <summary>
/// Minimum bound of the object
/// </summary>
public Vector3 MinimumBound { get; }
/// <summary>
/// Name of the feature.
/// </summary>
public string Name { get; set; }
/// <summary>
/// Number of triangles hold in the list of the model
/// </summary>
public int NumberOfTriangles { get; }
```

```
/// <summary>
/// sets or gets the render color of the feature
/// </summary>
public Color RenderColor { get; set; }
/// </summary>
/// Required to set or get the selection color
/// </summary>
Color SelectionColor { get; set; }
/// <summary>
/// Is entity render its bounding box
/// </summary>
public bool ShowBoundingBox { get; set; }
/// <summary>
/// This property indicates whether the feature will drawn
/// transparent or opaque.
/// </summary>
public bool ShowTransparent { get; set; }
/// <summary>
/// Tranformation matrix of the object
/// </summary>
public Matrix Transformation { get; }
/// <summary>
/// Transparency value of the feature. This value must be 0 for full transparent and
/// 255 for opaque. Values greater /// than 255 and smaller than 0 will be set to 255
/// and 0, respectively.
/// </summary>
public byte Transparency { get; set; }
/// <summary>
/// Is feature using its transformation matrix
/// </summary>
public bool UsingTransformation { get; set; }
/// <summary>
/// Vertices of feature
/// </summary>
public virtual CustomVertex.PositionNormal[] Vertices { get; set; }
```

```
/// <summary>
/// The indexer returns a triangle based on a numerical index.
/// </summary>
/// <param name="i">Indexer</param>
/// <returns> Triangle that corresponds to given indexer </returns>
public Triangle this[int i] { get; }

/// <summary>
/// Permanently apply transformation matrix to the vertices of the feature.
/// </summary>
public void ApplyTransformation();
/// <summary>
/// Method that will be called when device is lost. When you override this method, to
/// improve performance call disposable objects of the class if IsResetRequired is
/// false.
/// </summary>
public virtual void Dispose();
/// <summary>
/// Iterator method of the model class
/// </summary>
/// <returns>Triangles of the model in a foreach syntax</returns>
public IEnumerator<Triangle> GetEnumerator();
/// <summary>
/// Method used to render the object in a 3D environment
/// </summary>
/// <param name="device">Rendering device</param>
/// <returns>State of the rendering process</returns>
public virtual bool Render(Device device);
/// <summary>
/// Method that will be called when device is reset. When you override this method,
/// to improve performance call objects of the class that will be reseted if
/// IsResetRequired is false.
```

```
    /// </summary>
    /// <param name="device">Rendering device</param>
    /// <returns>State of the rendering process</returns>
    public virtual bool Reset(Device device);
    /// <summary>
    /// Sets transformation matrix
    /// </summary>
    /// <param name="isAbsolute">
    /// Is given transformation according to absolute values
    /// </param>
    public void SetTransformation(Matrix tMatrix, bool isAbsolute);
}
```

APPENDIX E

CASE STUDY

In this appendix, a standard rapid prototyping process is performed for a set of parts. The screenshots of the software after each operation is provided in operation order. The process is started by creating a new project. Different STL parts are imported to this project by using the main menu of the software. The imported parts are oriented in the production platform. All the oriented parts are displayed in different predefined views. Slices of the parts are previewed in X, Y and Z directions. At the end of the process, the parts are printed, sliced and hatched, by using the printing functionality of the software.

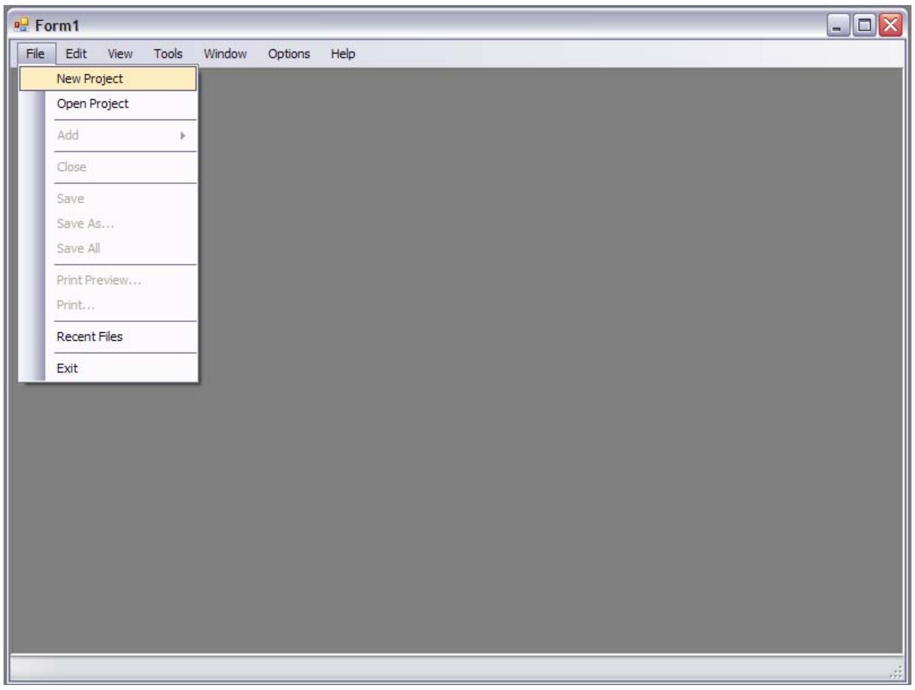


Figure E.1: Creating a new project by using the main menu of the software

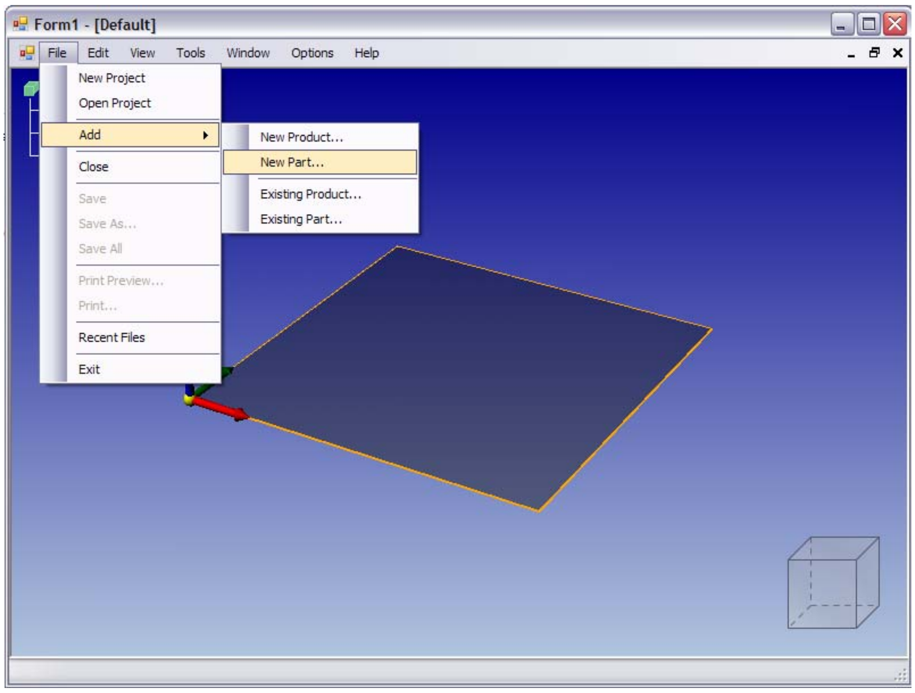


Figure E.2: Importing parts in STL format by using the main menu of the software

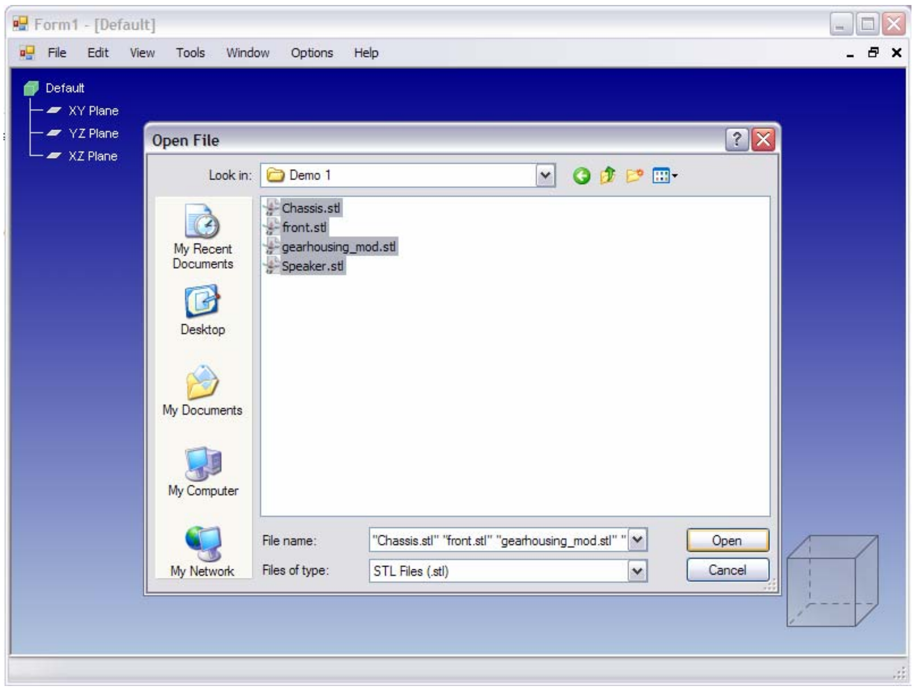


Figure E.3: Choosing the parts to be imported by using open file dialog

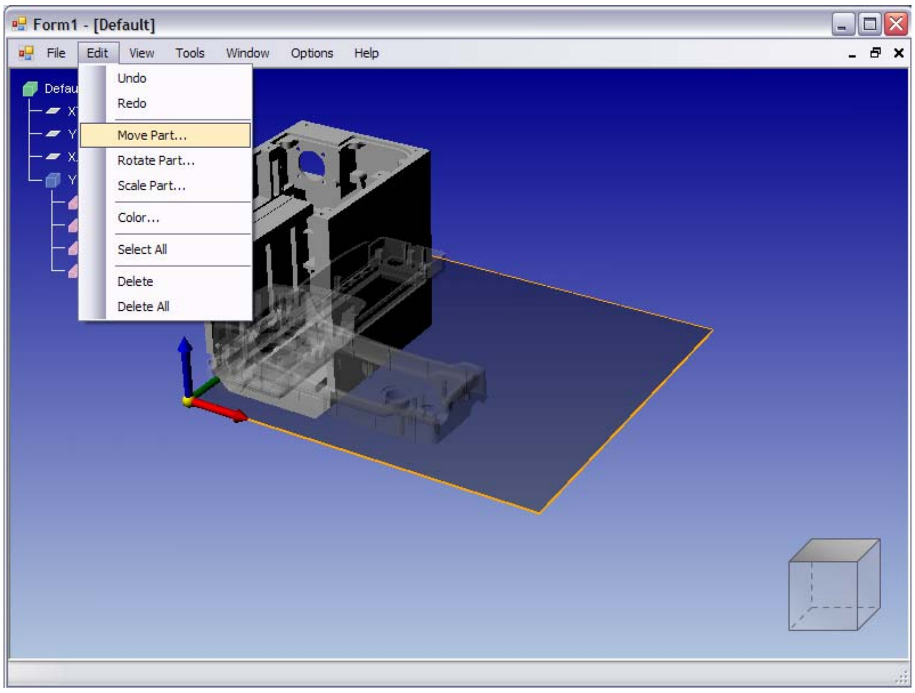
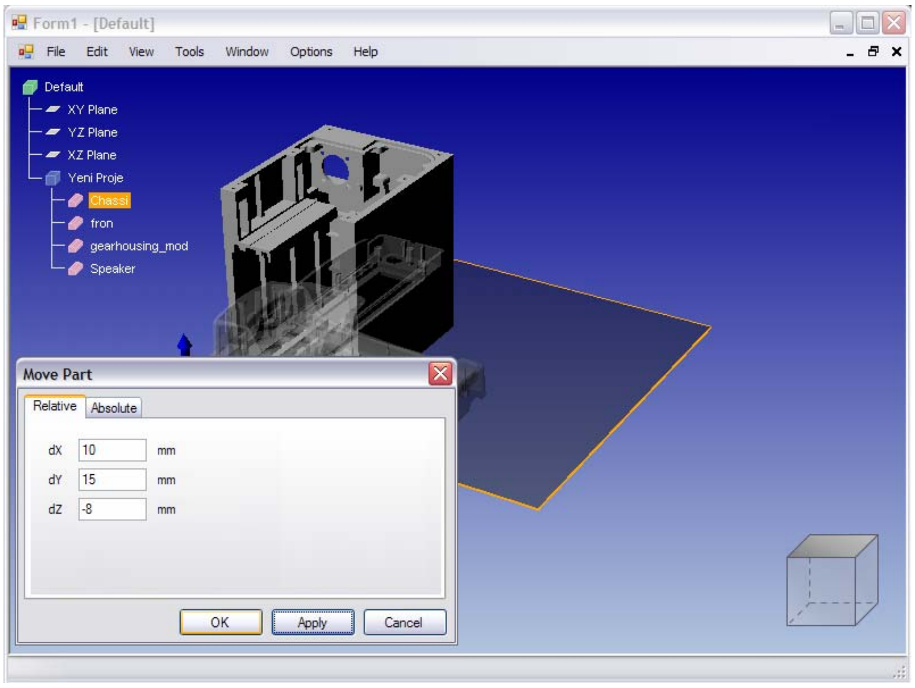


Figure E.4: Selecting a part from the graphical user interface to translate



dX 10 mm
dY 15 mm
dZ -8 mm

Figure E.5: Moving the selected part by using the translation menu

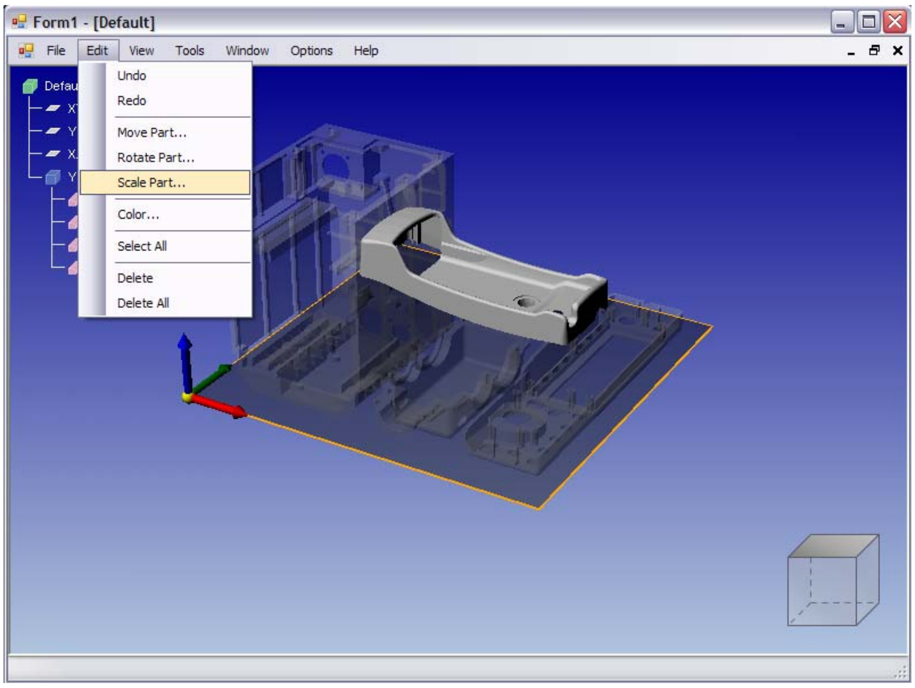
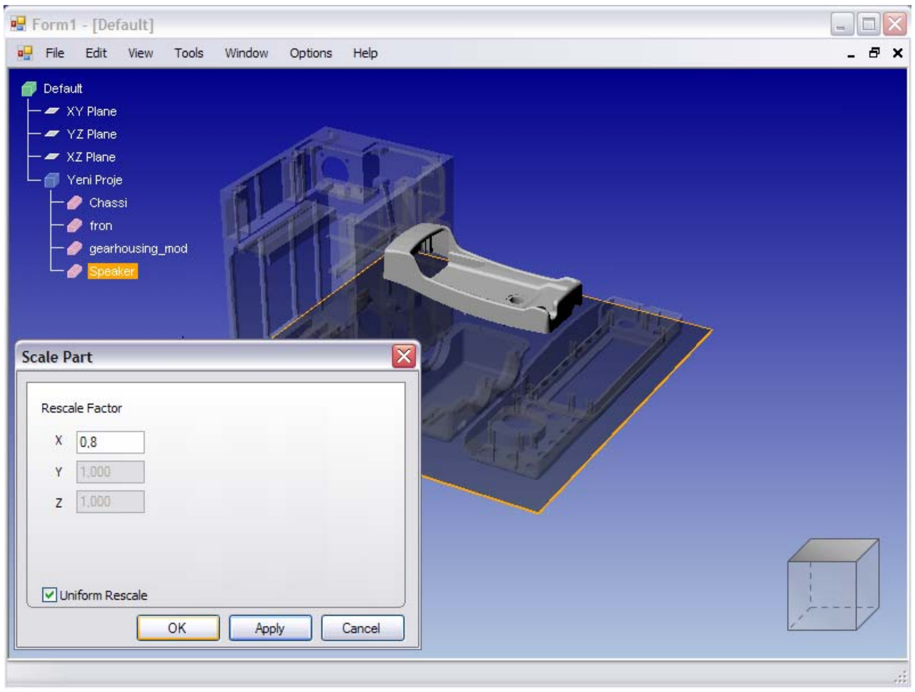


Figure E.6: Selecting a part from the graphical user interface to scale



Rescale Factor
X 0,8
Y 1,000
Z 1,000
Uniform Rescale ✓

Figure E.7: Scaling the selected part by using the scaling menu

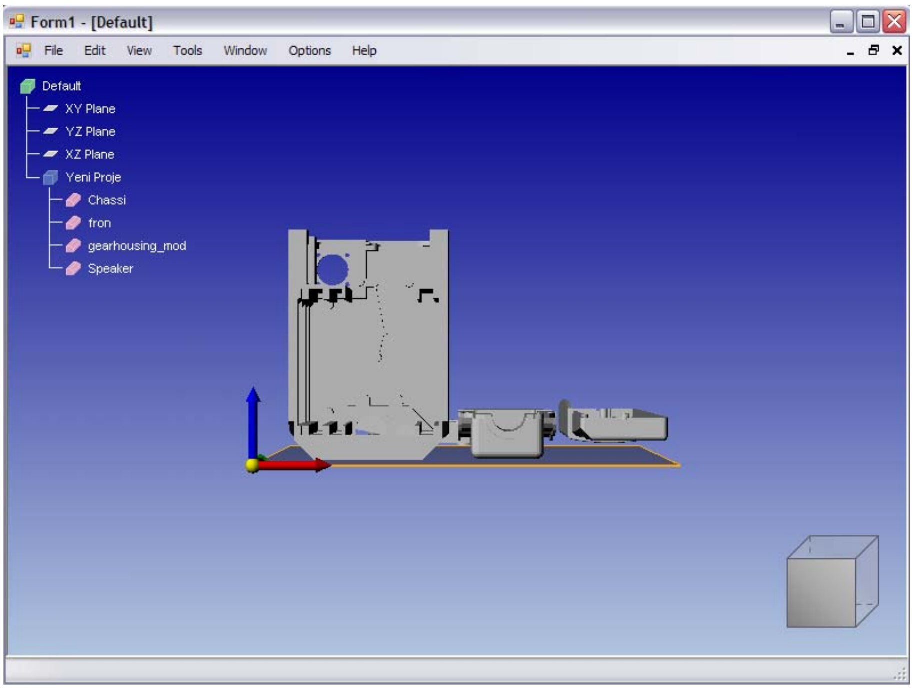


Figure E.8: Front view of the oriented parts

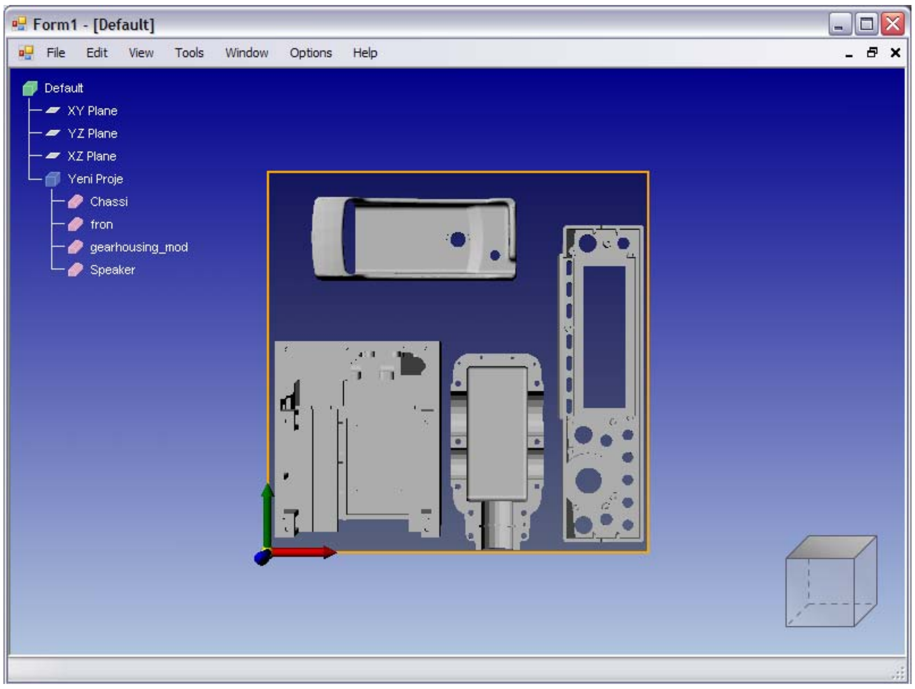


Figure E.9: Top view of the oriented parts

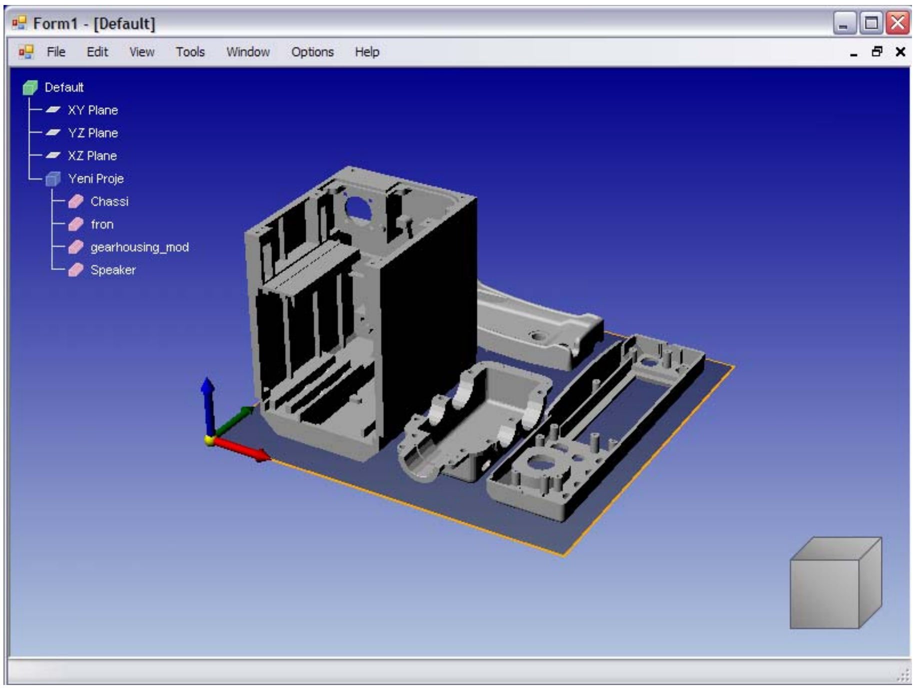


Figure E.10: Isometric view of the oriented parts

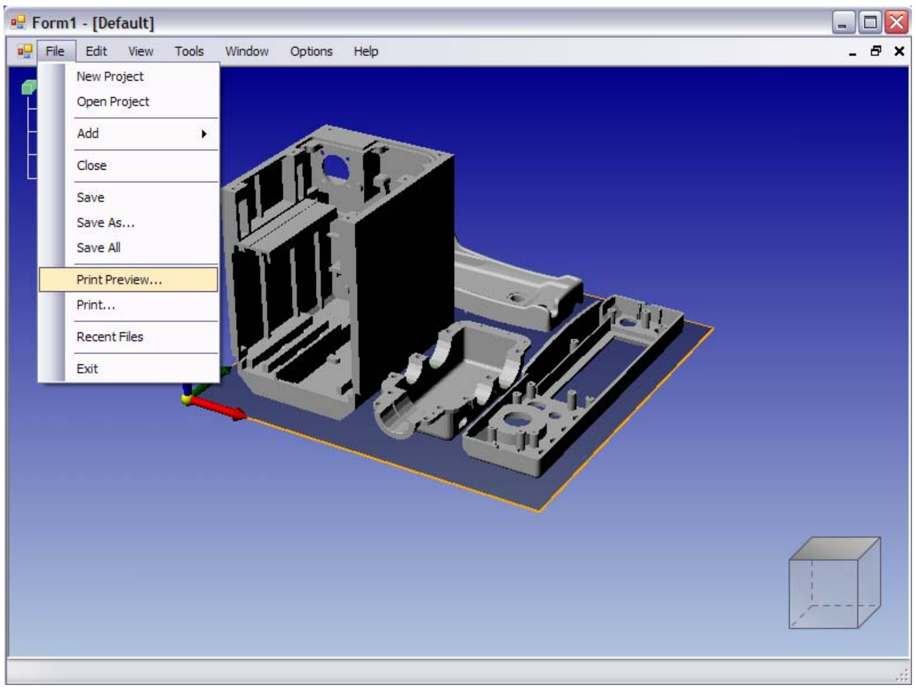
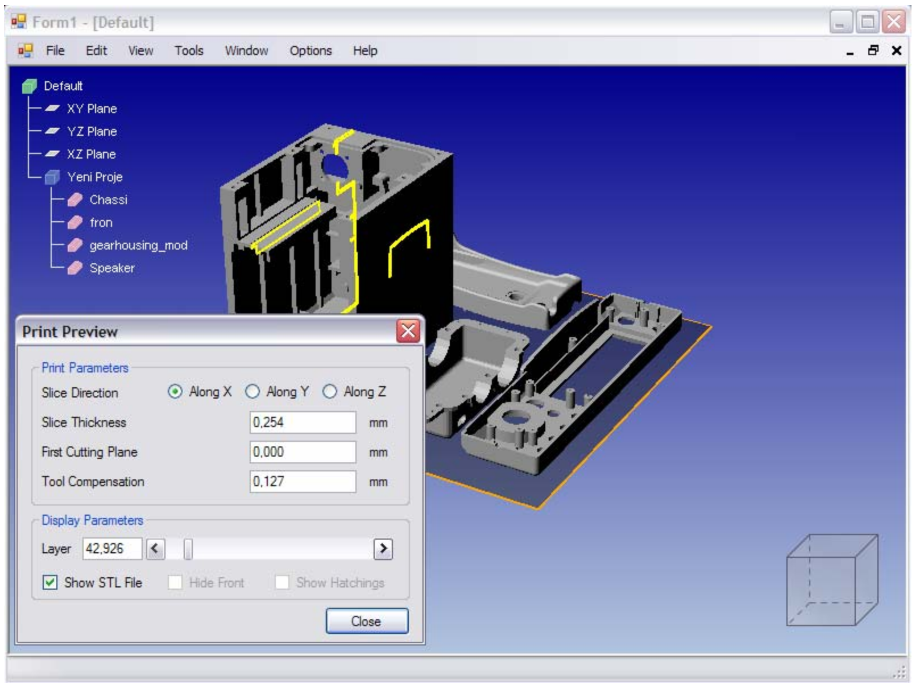


Figure E.11: Choosing the print preview functionality from the main menu



Print Preview

Print Parameters

Slice Direction [✓ Along X] Along Y Along Z

Slice Thickness 0,254 mm

First Cutting Plane 0,000 mm

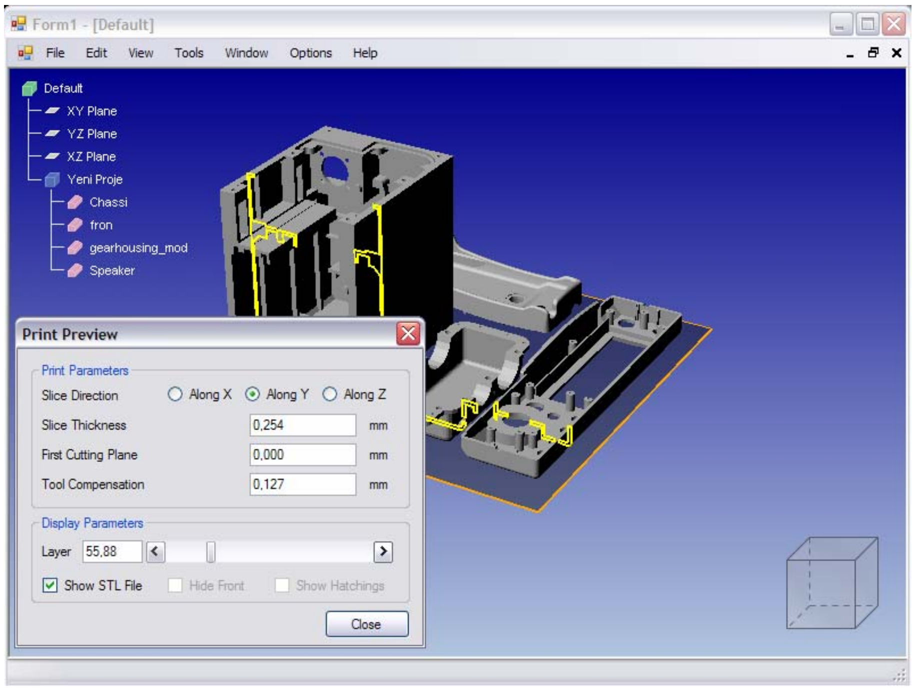
Tool Compensation 0,127 mm

Display Parameters

Layer 42,926

[✓ Show STL File] Hide Front Show Hatchings

Figure E.12: Previewing the slices of the parts along X direction



Print Preview

Print Parameters

Slice Direction Along X [✓ Along Y] Along Z

Slice Thickness 0,254 mm

First Cutting Plane 0,000 mm

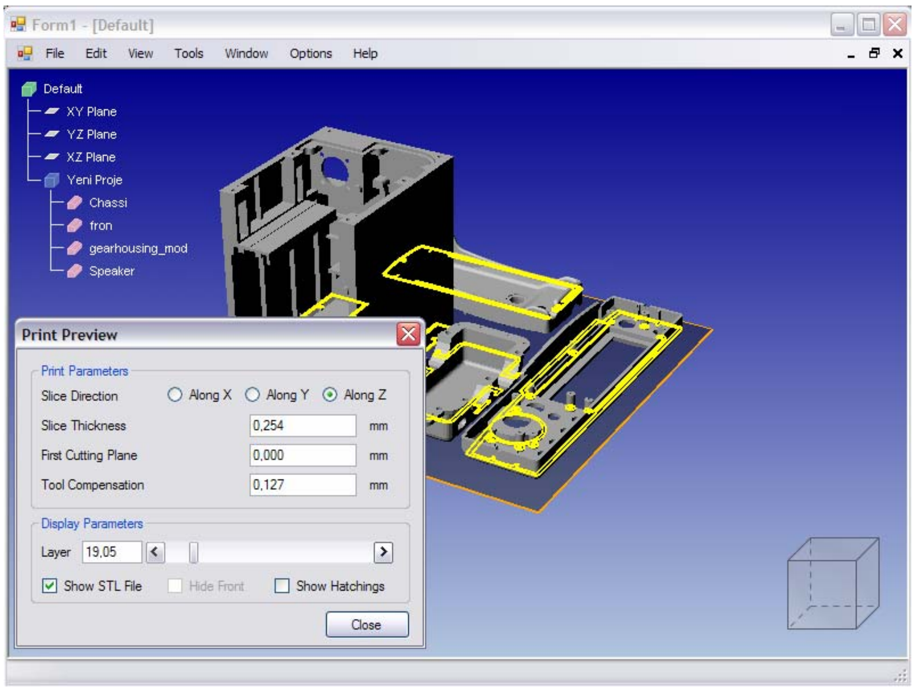
Tool Compensation 0,127 mm

Display Parameters

Layer 55,88

[✓ Show STL File] Hide Front Show Hatchings

Figure E.13: Previewing the slices of the parts along Y direction



Slice Direction: Along X / Along Y / [✓ Along Z]
Slice Thickness: 0,254 mm
First Cutting Plane: 0,000 mm
Tool Compensation: 0,127 mm
Layer: 19,05
[✓ Show STL File] Hide Front / Show Hatchings

Figure E.14: Previewing the slices of the parts along Z direction

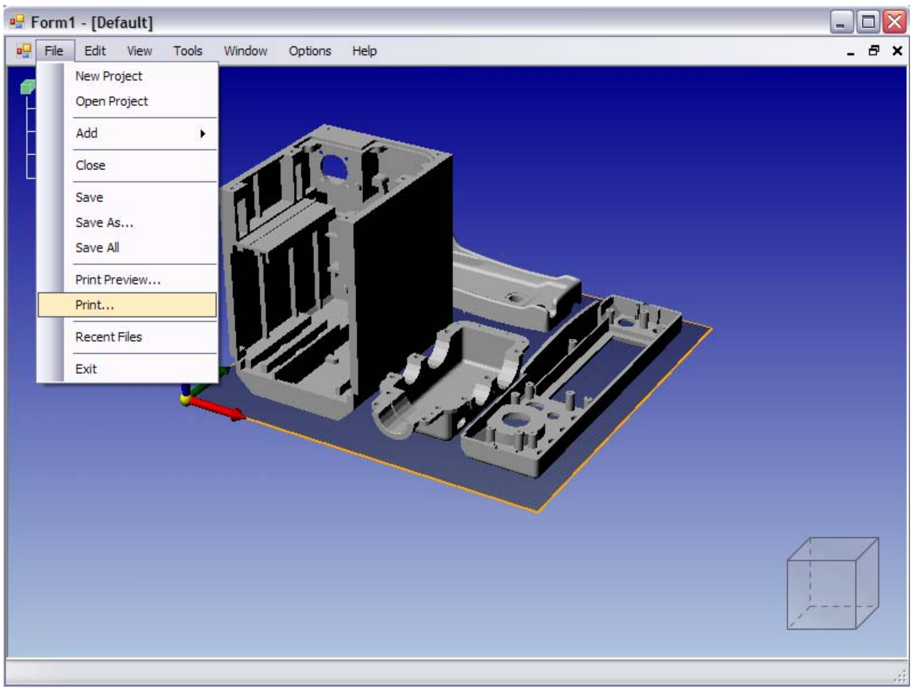
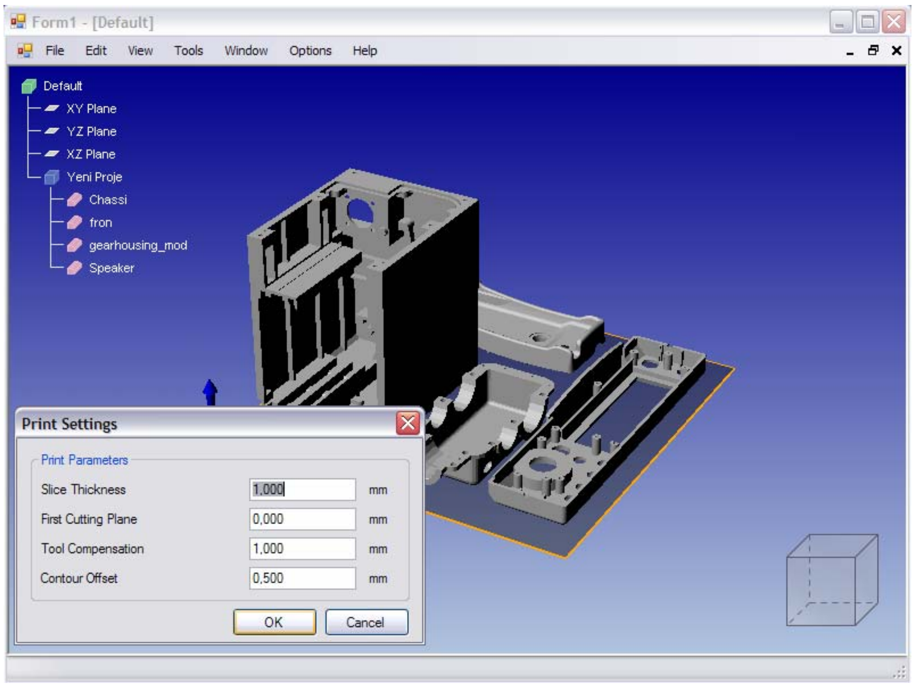


Figure E.15: Choosing the print functionality from the main menu



Print Settings

Print Parameters

Slice Thickness	1,000	mm
First Cutting Plane	0,000	mm
Tool Compensation	1,000	mm
Contour Offset	0,500	mm

Figure E.16: Printing the slices of the parts by using parameters

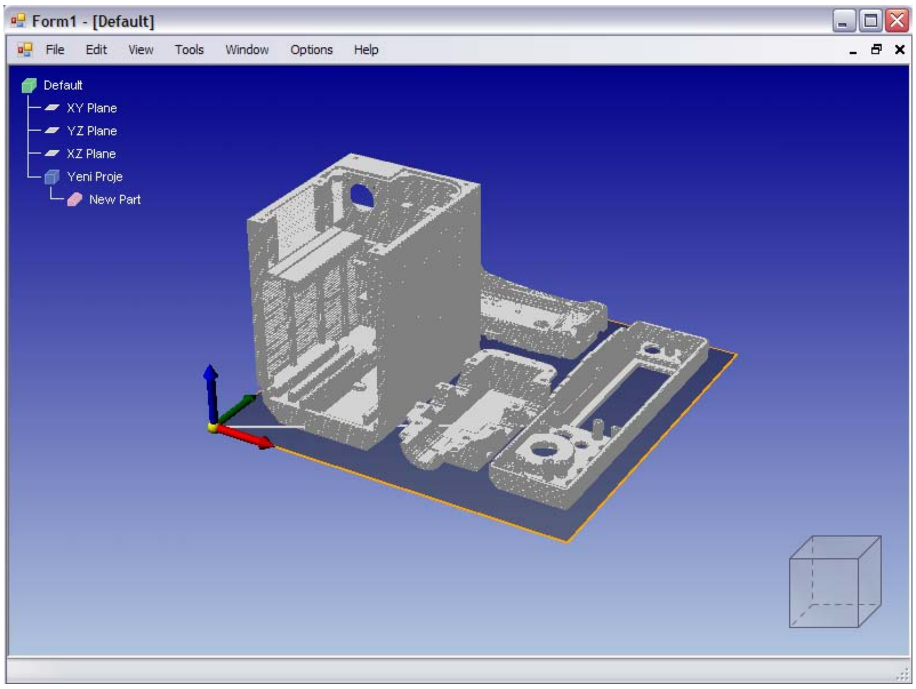


Figure E.17: Displaying parts after printing operation