

Implementation of user-defined features in web-based CAD applications

by

Tarun Kanakachari Kandala

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical Engineering

Waterloo, Ontario, Canada, 2009

© Tarun Kanakachari Kandala 2009

Authors Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Computer-aided Design (CAD) and Computer-aided Manufacturing (CAM) play an important role during the design and production phase of a product. CAD allows for generation of two and three-dimensional models of the product with the help of solid modelers and CAM allows for production planning of the product using tools such as a CAM package. Both CAD and CAM tools are highly specialized in nature and at the same time expensive to own. Large industries can afford to own such systems and experts to operate them but small and custom product industries cannot afford these benefits due to high design and manufacturing costs involved and complexity involved in using these systems. This has led to less popularity of CAD and CAM systems in custom product industries.

A web based design tool can offer the functionality of a CAD system to custom product industries by allowing them to create and design three-dimensional models over the web. This method helps in simplifying the complexity involved in solid modeling by automating the commonly performed design operations using design algorithms. At the same time a web based manufacturing tool can allow for automatic generation of tool-paths for machining using a CNC machine.

Although a web based design tool offers the required benefits to custom product industries, offering just the functionality of a CAD system may not be useful to the users of the web-based system who are usually carvers and artisans with minimal or no knowledge of CAD. An alternative method and its implementation are presented in this thesis. The method allows for creating user-defined parametric features using simple tools that can be offered in a web based application. The system takes advantage of the built-in API tools in a solid modeler and

advanced web-based technologies to integrate them into a simple and easy to use web based design system.

Identification of key elements in designing user-defined features and a framework for implementing them are discussed. Also different types of user-defined features that can be offered in a web application with examples of their implementation in a real world application for designing custom wooden signs are presented.

Acknowledgements

I would like to thank my supervisor Dr. Sanjeev Bedi for giving me the inspiration and encouragement throughout my research and without whom this thesis would not have been possible. It was a great experience to work under Dr. Sanjeev Bedi and I greatly appreciate the guidance I received throughout my graduate studies.

I would also like to thank all my fellow researchers in E3-3134 for the help and support they have provided me to make the progress of my research fun and exciting.

Table of Contents

List of Figures	viii
Chapter 1: Introduction	1
1.1 Design and Manufacturing.....	1
1.1.1 Programming in solid modeller	3
1.2 Internet as a data exchange media.....	3
1.3 Project Scope	4
1.4 Research Scope	5
Chapter 2: Literature Review	7
2.1 Computer Aided Design	7
2.1.1 Solid Modeling	9
2.1.2 Different models in CAD designing.....	9
2.1.3 Parametric Feature based solid modeling.....	13
2.2 Automating CAD modeling.....	14
2.2.1 Application Programming Interface (API).....	15
2.2.2 Structure of the existing system - WatCAD	17
2.2.3 Issues in implementing web-based CAD systems.....	18
2.2.4 Other applications.....	19

Chapter 3: Design paradigm	20
3.1 WatCAD system	20
3.2 User defined parametric feature based designing	22
3.2.1 Regenerative algorithms	23
3.2.2 Configuration for regenerative algorithm inside the web application	25
Client-side only:	26
Server-side only:	26
Client-server mixed environment:	27
3.2.3 Identification of key elements for user-defined features	28
Basic geometric elements	28
Transformation tools	29
Algorithm builder	29
3.3 Modified WatCAD system with user defined features	30
AJAX – Asynchronous JavaScript and XML [20]:	31
Chapter 4: Types of user-defined features	33
4.1 Introduction.....	33
4.1.1 Types of user-defined features	33
Type I user-defined feature: Symbols.....	33
Type II user defined features:	35
Type III user defined features	39
4.2 Design of user defined feature system	41
4.2.1 Adding relationships and constraints.....	42
Chapter 5: Implementation	44
5.1 Development of web-based sign design application – iSketche™	47
5.1.1 Functionality	48
5.1.2 Components of the iSketche system	51
Creation and display of geometry	51
Client - Server communication	52

Generation of 3D models	53
Load SVG file	53
CAD Initialization.....	54
Parse SVG and Create 3D model.....	54
Apply material and save snapshot	54
5.2 Implementing user defined parametric features	55
5.3 Example	59
Chapter 6: Conclusions and Recommendations	66
6.1 Conclusions.....	66
6.2 Recommendations	68
6.2.1 Development of sketchpad	68
6.2.2 Types of user-defined features	69
6.2.3 3D viewing	69
Bibliography	70

List of Figures

Figure 1 : CAD model of a computer mouse [32]	8
Figure 2 : Product life cycle - Design and Manufacturing [32]	8
Figure 3 : Cell Decomposition [3]	10
Figure 4 : CSG tree [3]	11
Figure 5 : Boundary Representation [3]	12
Figure 6 : MisumiUSA [19] Linear guides with lengths – 280mm and 960mm.....	15
Figure 7 : Table leg sections [7]	17
Figure 8 : Existing WatCAD system architecture	21
Figure 9: (a) Shaft with 2” larger diameter; (b) Shaft with 3” larger diameter	24
Figure 10 : Constraints applied to a window in a wall - Sketch.....	24
Figure 11 : 3D view of the rooms with the two different windows from Figure 10	25
Figure 12 : Regenerative algorithm in a Client-side only setup	26
Figure 13 : Regenerative algorithm in a Server-side only setup	27
Figure 14 : Regenerative algorithm in a Client-Server mixed setup	27

Figure 15 : Extrusion and sweep feature in a CAD software	29
Figure 16 : Modified WatCAD system with user defined features	31
Figure 17 : (a) Simple user-defined feature with six splines; (b) Three instances of the user defined feature I in (a) to form the University of Waterloo logo	34
Figure 18 : Type II user defined feature	35
Figure 19 : Circles C1 and C2 with no existing relation between them.....	36
Figure 20 : Modification of C1 does not affect C2 as no relation exists between them.....	36
Figure 21 : Type II user defined feature - relation added between C1 and C2	37
Figure 22 : Propagating type II user defined feature in a tree structure	38
Figure 23 : Three circles C1, C2 and C3 with no existing relation between them.....	38
Figure 24 : Type II feature relation added to the three circles C1, C2 and C3.....	39
Figure 25 : (a) Petal in a pattern with “angle” and “number of petals” as parameters; (b) Pattern applied with “number of petals” = 5; (c) Pattern applied with “number of petals” = 9	40
Figure 26 : Type III user defined feature procedure.....	41
Figure 27 : Programming structure for adding user defined feature	42
Figure 28 : Architecture of the iSketche application.....	45
Figure 29 : Machined wooden sign with chipped lettering	46
Figure 30 : Signs with varying shapes, backgrounds and borders.....	47
Figure 31 : Drawing tools in iSketche application	49
Figure 32 : Editing tools in iSketche application	49

Figure 33 : iSketche sketchpad design	50
Figure 34 : (a) Sample code of a SVG document with <i><defs></i> element; (b) generated shape from code in (a)	56
Figure 35 : Sketchpad with a random circle and rectangle drawn and their properties in the “Parameters” tab on the right	57
Figure 36 : Relation added between the circle and rectangle from Figure 35.....	58
Figure 37 : Pattern tools in the iSketche sketchpad.....	59
Figure 38 : 2010 Canadian Winter Olympic logo [30].....	59
Figure 39 : Drawing the first element of the 2010 Canadian Winter Olympic sign and saving it as a Definition.....	60
Figure 40 : Drawing 5 random circles for creating the Olympic rings.....	61
Figure 41 : (a) Adding relations between the circles in Figure 40; (b) Sketchpad updates with the new relations between the circles and adds it to the definition	62
Figure 42 : Creating an instance of the “Vancouver 2010 Olympic logo” and the generated 3D model	63
Figure 43 : Creating another wooden sign with Olympic rings as a background and the generated 3D model.....	64

Chapter 1

Introduction

1.1 Design and Manufacturing

Due to the globalization of the economy and increased international competition, engineering companies are being forced to cut down the costs in design and manufacturing. Also due to high labor cost in North America, manufacturing is being outsourced to countries with lower labor costs. Companies are trying hard to reduce cost at every stage of the product life cycle.

The product life starts with a *need* in the market. This is followed with the development of one or more *concepts* to meet the need. The concept is followed by *design* of the product or products. Once the design of the product is finalized it is *manufactured*, checked for *quality*, *packaged*, *marketed*, *sold*, etc. During the life cycle, Computer Aided Designing (CAD) software plays an important role. The role of CAD starts at the concept stage and continues until the design is finalized. One can also argue that CAD has a role in quality assessment, marketing and sales. CAD tools are expensive and operating these tools requires an experienced operator. In addition, the operator needs regular training to keep up with software changes. These expenses are necessary and any savings in these are attractive to companies today. Computer Aided Manufacturing (CAM) software also plays a similar role during

Introduction

manufacturing, quality checking and packaging. Similar to CAD, CAM has recurring expenses of software, operator and training associated with it.

In large manufacturing companies where the same product is made again and again, the cost of designing can be distributed over a large number of products. In this case the cost of CAD or CAM is easily justified. However, for small batch production or for production of customized products such as address plaques, bio-medical inserts and machine control panels, the cost of designing each part individually can be prohibitive.

Use of CAD and CAM systems in custom product markets has been limited due to the high-costs involved in design and manufacture. Although CAD and CAM coupled with computer code controlled (CNC) machines offer the potential of manufacturing customized products, the cost associated with use of design software and the cost associated with the manufacturing software makes this option less popular in industry today. It is rare to find a CNC machine being used by an artist or carvers to realize their art or concepts. The result is that the designs manufactured by wood carvers or by artisans are not accurate and carving takes longer and is not repeatable. If the combination of CAD, CAM and CNC can be offered as a tool to artists and carvers, then the quality of the product they produce will change because this tool offers reliability, accuracy and repeatability. Lack of technology innovation in offering easy to use and affordable software products to custom product markets is a deficiency in the existing CAD and CAM systems.

A combined CAD-CAM system solves this problem by offering functionality of designing and manufacturing in a single package and is often easy to setup. Another advantage of using such systems is that these software systems can be designed to target specific markets. An example of such software at the time of writing is DELCAM's ArtCAM [11] which provides a

Introduction

combined CAD and CAM package into one for designing customized wooden and metal products.

1.1.1 Programming in a solid modeller

The evolution of CAD progressed from wireframe modeling to surface modeling to solid modeling [3]. Solid modellers were developed to create a database of geometry and topology and were given functionality that allows users to obtain answers to geometric queries. This ability to answer geometric queries allowed solid modeling systems to offer an application programming interface (API). An API allows automated design and design based on rules. This API can be found in all solid modellers because the inherent idea of creating a solid model was to use the database of geometry and topology to answer geometric questions.

An API also offers designers and manufacturers to algorithmize the commonly performed steps using a CAD system and with these algorithms the steps can be repeated automatically and unattended. The API concept is also available on CAM systems and analysis packages like Finite Element Analysis (FEA) and other similar systems. The idea of capturing the design and manufacturing process in algorithms has been of attraction to engineers for a long time but it has not been captured in a proper manner that would make the use of CAD and CAM systems accessible to designers, artists and to industries that make customized products.

1.2 Internet as a data exchange media

The algorithms development using an API in solid modellers often requires hiring an experienced CAD operator. The expert needs to know solid modeling and also be an expert in software development. This expert however is only needed for a short span of time and once

Introduction

his expertise has been captured in a form of an algorithm, the algorithm can be used repeatedly. To offset the additional cost to hire the expert or experienced CAD operator or software developer, it is important that the company has access to a wider market. The tool that offers global marketing or reach to customers anywhere in the globe is the internet.

Internet has become an important source of marketing. For the custom product markets to strive in the global economy and to reach out to the end customer the internet can offer great options. With the development of new web technologies such as Web 2.0, the internet is becoming a primary place of choice to showcase the products through web pages with NikeID [12] being one example. The advantage of implementing such internet based marketing strategies is that the end user gets a feel of uniqueness when designing online.

One of the issues that custom product markets face is the time lag involved in getting the design made and manufactured and getting it delivered it to the customer. By the time the designer designs a product and gets the user to see the design for approval, the user may already lost interest in buying the actual product. This is more apparent in products that need to be designed first in CAD software and later has a unique manufacturing strategy of its own.

1.3 Project Scope

The focus of the current project is to simplify the design and manufacturing process to custom product markets. The solution is offered in the form of a web-based design and manufacturing system that helps in automation of CAD designing and tool-path generation. The system operates by running pre-defined macros on the server side and a user interface on the client side of the web application. Based on the user input, the macros create a solid model automatically inside the solid modeller and present the user with a preview of the CAD model

Introduction

generated. After the user is satisfied with the design, the CAM macros on the server side analyze the designed part and create a tool-path automatically. This tool-path can be used in a CNC machine for machining the part.

During the product life cycle, CAD designing and production planning consume the maximum amount of time. Automating such a process significantly reduces the time to create a part and provides an edge in today's competitive market. The web based designing and manufacturing system offers the required automation process and has many advantages over conventional methods of using the CAD and CAM software. Designers can use the online system to design parts faster and easier as commonly performed tasks are automated through pre-recorded macros. The simplicity involved in operation of the online web system significantly reduces the time taken in training of CAD operators, thus reducing overhead costs.

For developing the web based design and manufacturing system, the concept of Web 2.0 technology was implemented. Web 2.0 is the second generation based website designing that offer users with more interaction giving a sense of control over the content rather than a simple information based website. An example of a Web 2.0 based application is YouTube[®] that allows users to upload and share videos online.

1.4 Research Scope

The goal of this research is to create a user-interactive online CAD designing system to allow users to create their own reusable parametric 3D models. The system will be integrated with the automatic tool-path generation system for developing a fully fledged web-based automated CAD-CAM system. The online system should be able to allow users to create customized

Introduction

parametric features that will not only be used to design artistic 3D models but also allow the model to be shared with other users using the system. User defined parametric features offer a new and innovative way for non-CAD designers to create actual 3D models over the internet without prior training.

To build the application, previous works were critiqued and new ideas were developed. As part of the research, new web based technologies had to be utilised for getting the desired results. Creating online designs in a client application needs to be managed using client scripts and the back-end server end 3D modeling needs to be managed by the use of algorithms written in server-side scripting languages such as ASP (Asynchronous Server Pages) or PHP (Hypertext Preprocessor). Also implementing user defined parametric features online has its own challenges. To overcome these challenges, the concept was carefully organized and developed with the help of the available web technologies and previous work on online CAD systems. Algorithms known as regenerative algorithms were formulated to help create user-defined parametric features.

Chapter 2

Literature Review

2.1 Computer Aided Design

Computer aided design (CAD) is used for 2D drafting and generating 3D models with the help of modeling software. Figure 1 shows an example of a model created with a CAD package. A CAD package such as SolidWorks or CATIA enables a user to create these models in a graphical user interface (GUI) with the help of mathematical and parametric tools.

CAD modeling plays an important role during the design and manufacturing phases of a product life cycle. Figure 2 shows a simple product life cycle in the form of a flow chart with emphasis on the engineering design and prototype building. Through CAD modeling, a virtual representation of the final product that needs to be manufactured can be created. This helps in simplifying the creation and modification of designs throughout the prototype and final design processes. This is also beneficial when designing parts that will be part of an assembly such as the piston or the crankshaft inside an automotive engine block. CAD modeling is also used to get solutions to geometric information such as volumes, surface area and sectional properties of the designed part.

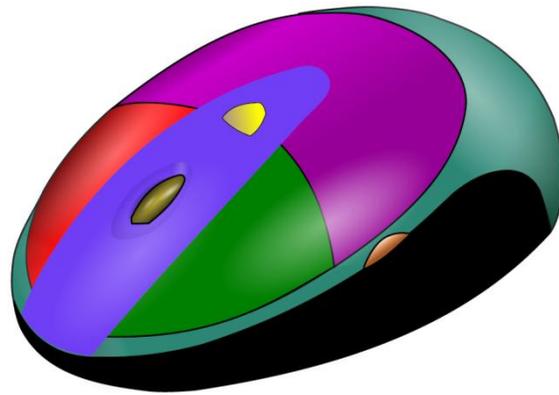


Figure 1 : CAD model of a computer mouse [32]

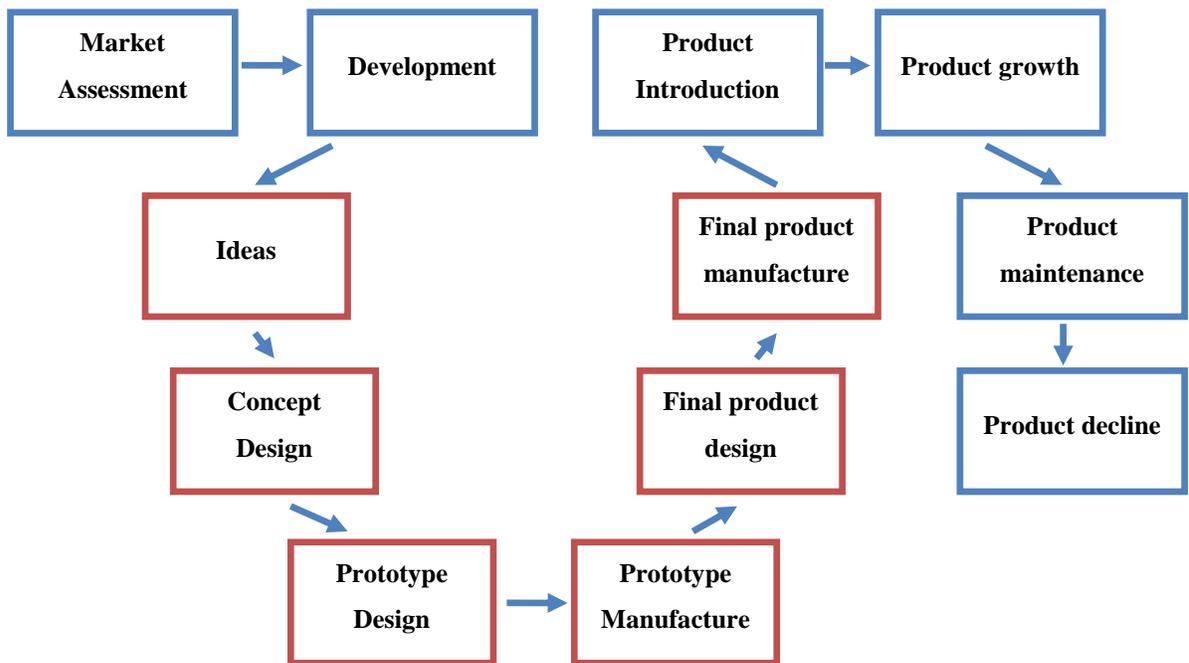


Figure 2 : Product life cycle - Design and Manufacturing [32]

Literature Review

CAD encompasses the entire array of computer tools that are used as design aids. Historically, CAD was used as a drafting aid for making engineering drawings. In time it evolved into wireframe modeling and subsequently to hidden line removed models. The real development in the field began with a search for methods to represent solids and the evolution of solid modeling.

2.1.1 Solid Modeling

Solid modeling deals with the complete physical representation of solid objects algorithmically. Solid modeling in computers is done using various techniques such as Constructive Solid Geometry (CSG) or Body Representation (B-Rep). In CSG, solid models are constructed using Boolean operations like union and intersection whereas in the B-Rep technique, solids are constructed by developing a data structure to model the topology and represent the geometry of the part in terms of boundary surfaces [3]. A solid modeler based purely on CSG modeling has an easy to use interface but it is difficult to visualize the models created in CSG whereas a B-Rep model is hard to construct but lends itself for efficient viewing. Most of today's solid modelers are hybrids. They use CSG as a user interaction tool and maintain the internal representation in terms of B-Rep model.

2.1.2 Different models in CAD designing

The early works on actual representations of solid models has been extensively covered by Requicha et al [10] and by Martti Mäntylä [3] during the 1980s. Different representations such as the Cell Decomposition models, half-plane models, CSG, B-Rep, etc., were discussed extensively in [10] and [3] for computer-based solid modeling systems to represent rigid solid objects.

Literature Review

Cell decomposition models described 3D models with the help of cells also known as the building blocks that are glued together to form a solid – Figure 3. The cell itself is typically a curved polyhedron described by points and the solid is constructed by means of a collection of semi-disjoint cells that necessarily do not have common interior points. The problem with this type of modeler is that the user does not have direct access to the solid primitives of the 3D models.

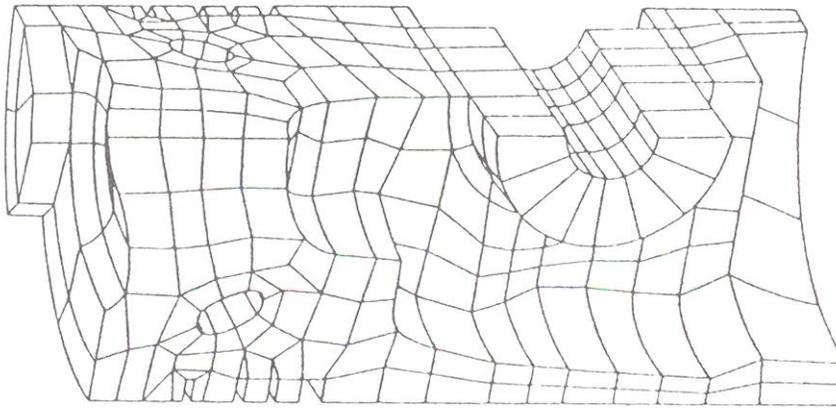


Figure 3 : Cell Decomposition [3]

CSG based solid modeling on the other hand uses CSG tree to represent solids, and is defined as follows:

$$\langle \text{CSG tree} \rangle ::= \langle \text{primitive} \rangle \mid \langle \text{CSG tree} \rangle \langle \text{set operation} \rangle \langle \text{CSG tree} \rangle \mid \langle \text{CSG tree} \rangle \langle \text{rigid motion} \rangle$$

The $\langle \text{primitive} \rangle$ leaf is an instance of a solid primitive such as a sphere or a cube. $\langle \text{rigid motion} \rangle$ is a translation or a rotation applied to the primitive, and $\langle \text{set operation} \rangle$ is a Boolean operation. An example of CSG tree is shown in Figure 4.

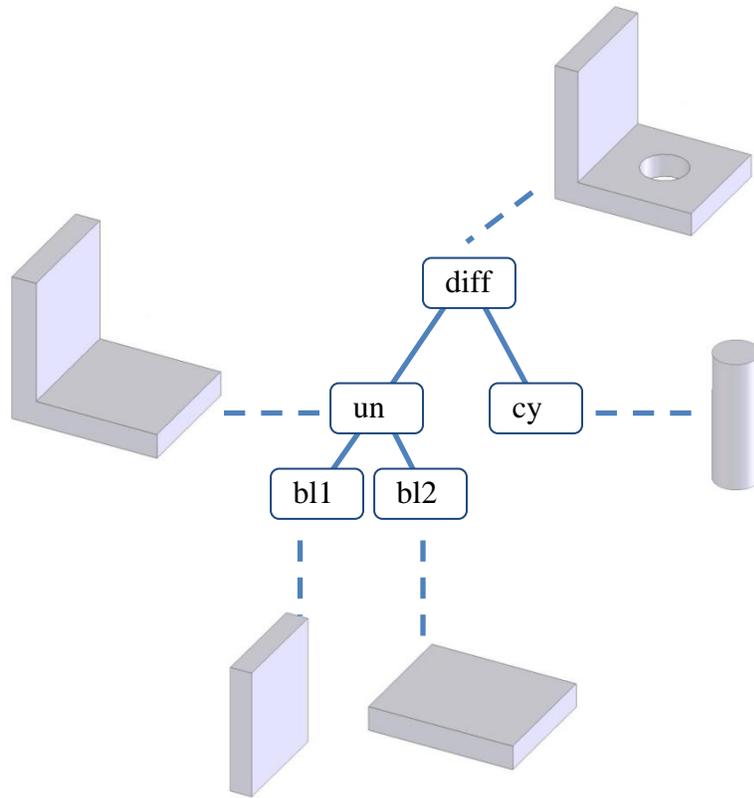


Figure 4 : CSG tree [3]

In Figure 4 the two rectangular slabs *bl1* and *bl2* are combined using the union operation *un* to create the *L-bracket*. Using the cylinder *cy*, a difference operation *diff* is used to create the final piece with a hole on the horizontal slab.

Boundary Representation technique is another type of solid representation that was discussed in [3] and [10]. As mentioned earlier, a B-Rep model gives a complete description of the bounding surfaces of the object. It has advantages over CSG in that it better represents the hidden lines and also the faces and edges of the solid model. But construction of the B-Rep model itself is harder as compared to the CSG modelers. Figure 5 gives an illustration of

Literature Review

the B-rep model. Figure 5 (a) shows the surface of an object divided into faces, which are represented as bounding polygons in Figure 5 (b) and in terms of edges and vertices in Figure 5 (c)

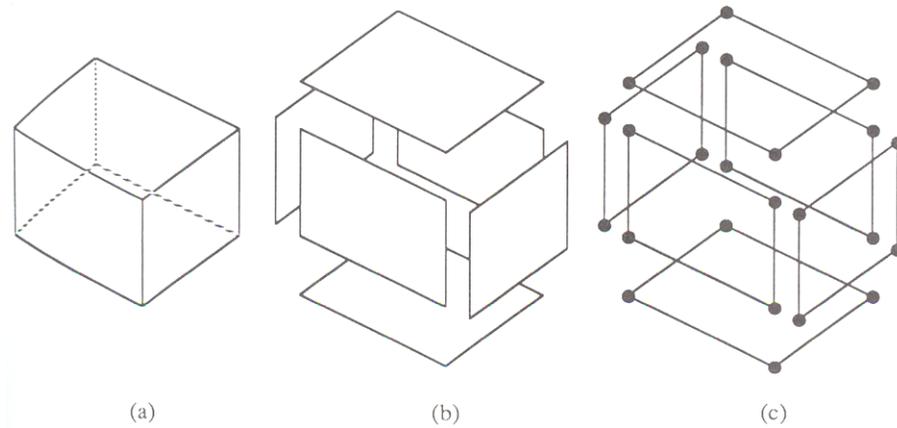


Figure 5 : Boundary Representation [3]

During the 80's, solid modeler packages were introduced in the commercial markets. These implemented the CSG and B-rep techniques in them. UniSolids released by Unigraphics, currently known as NX [13], used the CSG based kernel for solid modeling and Romulus [14], by Ian Braid and Charles Lang, used the B-Rep model to construct solid geometry. With the development of different computer workstations and better 3D rendering support, the solid modelers kept emerging as one of the basic means for engineering modeling. One of the most important features of a B-rep is that the database of geometry and topology can be used to obtain answers to geometric queries using algorithm. This feature of algorithm based solutions to geometric queries can also be extended to building a database using geometric algorithms. These abilities are the bases of an Application Programming Interface (API) and are available in most solid modelers.

Literature Review

Later CAD systems used hybrid methods that implemented both CSG and B-Rep techniques. In all modern systems, feature based designing is used to create 3D models. In feature based designing, a solid model is created by adding features one after the other sequentially by using a parent – child pattern. The models are created using sketcher based features which build 2D sketches that are extruded or swept to create 3D objects.

Chen et al. [1] presented a feature based designing system as opposed to a CSG or B-Rep techniques to design CAD models. According to [1], feature based designing have several advantages over CSG and B-Rep like the ability to better present the product geometry and functionality, a better method for designers to work on higher level of geometric features and overall standardization of the method for improved manufacturability and product quality.

With the advancement of the CAD systems, the functionality of the system itself played a major role when designing and editing complex geometry. The most common method followed in engineering design is the parametric feature based design. Javier Monedero [2] has presented a review of how models in architecture can be easily represented so that it is easy to both create and modify the models using the parametric modeling method.

2.1.3 Parametric Feature based solid modeling

Most modern CAD packages employ the concept of parametric designing to create 3D models. In parametric based designing, geometric models are built based on parameters. For example, a circle would have the parameters radius and position of center whereas a line is defined by its start and end points. When it comes to 3D models, a cylinder can be created using a circle extruded to a given height. Extrusion is a parametric feature with base sketch and extrusion height as its primary parameters. Examples of other kinds of parametric features include drafts, chamfers and holes.

Literature Review

Another feature of parametric modeling is constraints or relationship based design. Relationships such as perpendicularity, parallelism, and concentricity can be added between geometric elements to create a constrained sketch. This helps in maintaining the properties of the model when modifying it by changing the parameter values. This is of great advantage as it makes modeling simpler and faster. Parametric design has been around for many years and has been implemented in most of the commercially available CAD packages. For example, SolidWorks [15] has a user interface that incorporates parametric based design that allows users to create sketches that involve adding constraints between sketch elements. Gossard and Light [4] [5] have defined a variational based geometric modeling algorithm that allows adding constraints between elements of a geometric model using a method called the matrix method which can determine the shape of the geometric model by solving constraint equations. The algorithm presented by Gossard et al [4] [5] allows for solving under-constrained geometry and at the same time flag the user in case of over-constrained geometry.

2.2 Automating CAD modeling

From the above discussion, it can be said that CAD packages have a lot of capability that allows designers and engineers to create 3D models using techniques such as parametric feature based modeling. But CAD modeling itself is a professional job and requires both knowledge and experience to create well constrained geometry when designing complex models. Modeling gets complex when parts need to be edited many number of times during the engineering design process. Also industry typically specializes in making one or more products of the same type and they all require similar methods of modeling. By automating commonly performed CAD operations, an industry can simplify design and modification issues. The method to generate 3D models can also be captured in a generalized way that can

Literature Review

be automated. For example, when designing linear guides, as shown in Figure 6, many different lengths of guides can be designed just by changing the value of the parameter length and keeping other dimensions fixed and also the number of threaded holes on the guide.

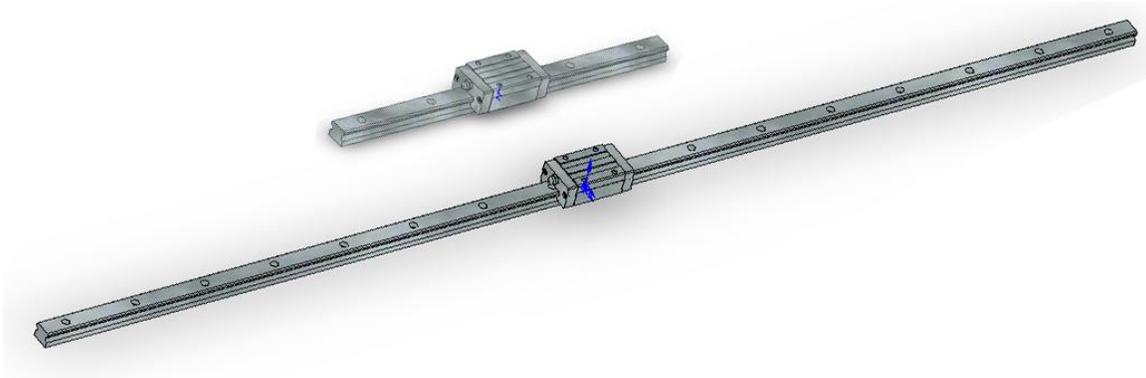


Figure 6 : MisumiUSA [19] Linear guides with lengths – 280mm and 960mm

2.2.1 Application Programming Interface (API)

To automate the design process, CAD packages offer built-in APIs that allow building 2D and 3D models procedurally. With the API feature a user can create and run macros that can speed up the design process by sequencing operations like emboss and sweep in an automatic algorithm to design an object with user specified dimensions and parameters. The functionality of an API has not been completely explored or utilized in a production environment but if implemented properly it can play an important role in any product's life cycle.

Creating macros using an API involves an intricate knowledge of solid modeling, geometry and topology. In addition it requires programming know how. This means that only designers with good programming skills can create these macros and implement them into the

Literature Review

existing design system. In the case of large manufacturing industries such as automotive or aerospace industries where the cost of designing is distributed over the large number of parts produced, having such a designer on board can be justified. But in small and mid-sized industries or even custom product markets, having the same designer may not be affordable. In addition to these costs, there is a cost of CAD package involved as well. In such a scenario, automating design and customization process would be an attractive and potential solution to these industries.

One of the early works on automating designing and manufacturing was CyberCut [9] that allowed users to design simple machinable CAD models over the web using Java applets. The technique used to design CAD models is based on destructive solid geometry. Unlike CSG, destructive solid geometry technique is used to create objects by removing material from a *stock* part similar to CNC machining. Using this technique, users can visualize designing based on how a CNC machine would machine the same product.

Zeng et al [16] have also proposed an architecture for building web based CAD models and made use of JAVA and CORBA to implement the system. But the system can only support basic curves and surfaces to be drawn using the GUI.

Another similar web-based system called WatCAD [7] was designed and built at University of Waterloo. WatCAD allows users to create 3D wooden table leg models using ASP.NET. In this system, the user is presented with an online form where values to different parameters such as length of the table leg, add-on features such as braid feature or a star pattern are provided by the user. The server running a CAD system interprets the user inputs to generate a 3D model of the table leg on the fly. The user has the option to go back and edit the features and also view the 3D model in the browser using an STL viewer. This system has the potential of growth and thus will be studied further.

2.2.2 Structure of the existing system - WatCAD

The WatCAD system provides a simple method of how an online web-based CAD application can be implemented in real world situations such as the designing of decorative table legs in this application. The user is presented with a GUI that gives step by step instructions to start building a table leg from scratch. The procedure followed to create the design is similar to the usual procedure followed in a CAD system. For example, the table leg has five sections:

- (a) Connector – top part of the leg that is attached to the table
- (b) Transition – the connection between the connector and the middle part of the leg
- (c) Middle – this is the longest section of the leg which also has the design features on it
- (d) Second transition – another connection between the middle section and the bottom part of the leg
- (e) Bottom piece or the “foot” – the section of the leg that touches the floor

Figure 7 shows different table legs with the sections labeled [7].

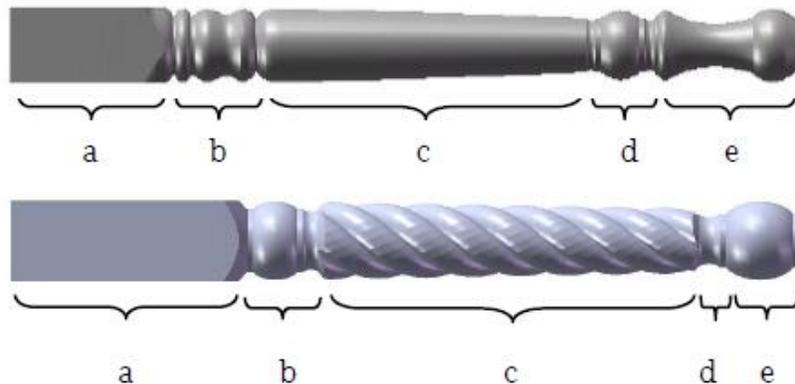


Figure 7 : Table leg sections [7]

Literature Review

The WatCAD system uses the totem pole approach to design the CAD model of the table leg based on the above mentioned sections. To automatically generate these CAD models the WatCAD system uses the built-in APIs of the solid modeler. To accomplish the task of designing over the web, WatCAD uses ASP.NET [17] technology to design the web application. Inside the web application, the information from the client web browser is transferred to the server side solid modeler through an ASP manager. The ASP manager runs pre-defined geometric algorithms that create CAD models inside the solid modeler. After the appropriate CAD model of the leg has been created, the user or the client is presented with a snapshot of the model. In this manner the user gets to see the results of the model designed inside the web browser itself.

The WatCAD application uses constructive geometry model to create the wooden table leg model instead of the destructive geometry model approach that was used in the CyberCut [9] system. The advantage of such a system is that it gives the designer the sense of working in a virtual CAD environment.

2.2.3 Issues in implementing web-based CAD systems

From the above mentioned web based design methods it can be said that if a paradigm can be implemented that allows design to be done automatically without any human intervention then it is of a great advantage. If the modeling is exposed over the web, an average web-user can create a 3D model in few seconds without any prior CAD knowledge.

But the configuration of the web based system needs to be designed appropriately to allow design over the web to be simple and robust. Factors such as the different types of web browsers, server-side configurations and the internet connection speed between the client and server can affect the performance of the web based CAD system. Also, the web-based systems

Literature Review

need to be light-weight, in the sense that they cannot offer many options or functions to the client.

The WatCAD system mentioned earlier tackles most of these issues by using the ASP.NET technology to design the architecture of the web application and by running complex designing algorithms on the server instead of running them on the client side java applets. Although it is a good solution to online CAD applications, it has the weakness of the user not being able to modify the offered designing features or tools to create new features. The offered features may not be useful or could be complicated to work with if the user is not a CAD expert. Another weakness of the WatCAD system is that the paradigm separates the users and the experts. The experts put on features without knowing what the users want and the users cannot add features that they desire. This is a major weakness in the WatCAD system.

2.2.4 Other applications

If the above mentioned issues can be resolved, the web based CAD design can be used successfully in other types of applications. The paradigm allows users to take advantage of the expert's experience. Furthermore, if the experts create tools that allow users to build their own features then the system has uses beyond table legs. The web-based design system can be used in other custom product markets where users create their own custom products online. As discussed earlier, a custom table leg was one of the applications that uses a back-end server system connected to a CAD package to design 3D models. Another application which is in the current scope of the project is personalized wooden signs which will be discussed in the later chapters.

Chapter 3

Design paradigm

As mentioned in the earlier chapter, the architecture of the WatCAD system only allows the experts to add features in the system. Experts have direct access to the server and can develop features and install them for use by clients. But experts are not artisans and are not connected to the market, thus their designs may not be useful in the market place. By offering the ability to add user defined parametric features within the WatCAD system the web application can offer a solution. This chapter evolves the design paradigm behind WatCAD to allow user defined features.

3.1 WatCAD system

As discussed in the earlier chapter, the WatCAD system implemented the parametric feature based design. But these parametric features were pre-defined and the user was only allowed to re-create these features with different parameter values. At the same time, to implement the user-defined parametric features into the WatCAD system, the architecture of the system needs to be completely analyzed. Figure 8 shows the present architecture of the WatCAD system.

Design Paradigm

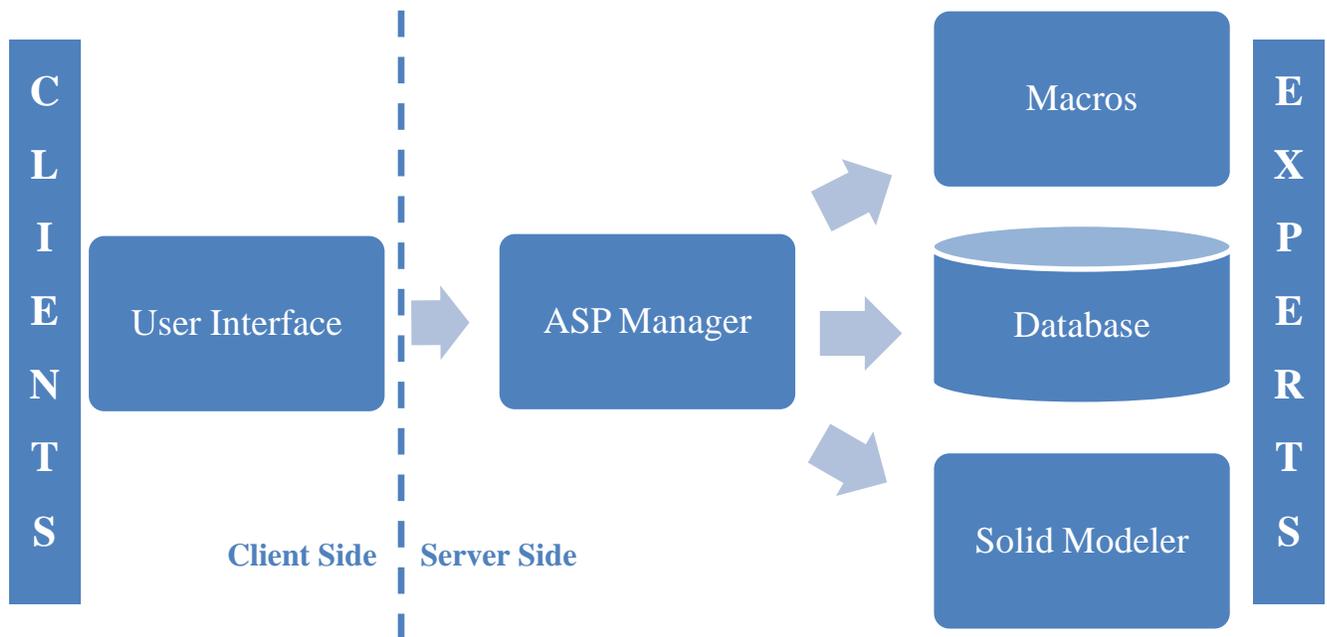


Figure 8 : Existing WatCAD system architecture

The WatCAD system used ASP.NET [17] technology and VB.NET language to design the web application. This choice was made on the basis that the solid modeler being used supported Visual Basic as the language for programming in its API.

Looking at the architecture of the WatCAD system, the system has a user interface that allows users to enter different parameter values of the features inside a web form. These values are then submitted to the server using an ASP manager where all the macros are stored. Depending upon the type of feature requested by the user, the ASP manager initiates the appropriate macros and creates a 3D model inside the solid manager. During the process of analyzing user inputs and creation of 3D model inside the ASP manager, the user interface just displays a loading message inside the user's browser and no other activity happens inside

Design Paradigm

the browser. Once the model has been generated by the solid modeler, the ASP manager responds back to the user browser with a snapshot of the 3D model created. The user at this point has the option of using another feature or modifying already created features.

From the above configuration, it can be said that the architecture of the WatCAD system is an example of the server-side only CAD system and a server-side only system has disadvantages that the user has no knowledge of how long a particular action is going to take.

3.2 User defined parametric feature based designing

Parametric feature based design is used in all modern design software systems and is one of the widely used methods of designing CAD models. Parametric designing allows for constraint based and dimension-driven solid modeling which simplifies model regeneration after the variables have been changed [18]. This regeneration of models is done with the help of algorithms often termed as *regenerative algorithms* which will be discussed here in detail. Regenerative algorithms are a set of functions inside a solid modeler that process the modification of geometrical entities when their original parameter values are changed to new values. To design the architecture for designing user-defined features, the concept of regenerative algorithms [3] can be used.

In web based CAD applications, offering such a dimensional constraint feature can be of a great advantage in creating user-defined parametric features. This will simplify complex designing tasks that were not possible in the WatCAD system with a limited number of features. With the help of the regenerative algorithms, a concept for implementing user-defined parametric features can be formalized. Although this concept can be implemented into the existing WatCAD system to design custom table legs, it was decided that the concept

Design Paradigm

would be applied to simpler applications such as flat objects. The application thus chosen was custom flat wooden signs and this implementation will be discussed in later chapters.

3.2.1 Regenerative algorithms

The challenge in building user based parametric features is that the web application does not know what kind of a parametric feature the end-user is looking for. To overcome this issue, drawing elements such as circle, line and spline that are fundamental to making a feature can be offered to the user. The regenerative algorithms need to be robust enough to properly interpret the user inputs and at the same time offer enough flexibility to the user.

To implement regenerative algorithms, the basics of parametric designing in solid modelers have to be re-visited. In CAD software, parametric designing has been used to modify the design of the model by maintaining relationships and constraints. For example a simple shaft can be constructed in a solid modeler with the larger cylindrical part with Ø2” and the smaller cylindrical portion with Ø1” as shown in Figure 9 (a). If at some point the design of this shaft needs to be changed, the 3D model gets updated automatically and still maintains any constraints. For example, if the larger shaft diameter changes to Ø3” then the mode just updates as shown in Figure 9 (b).

Design Paradigm

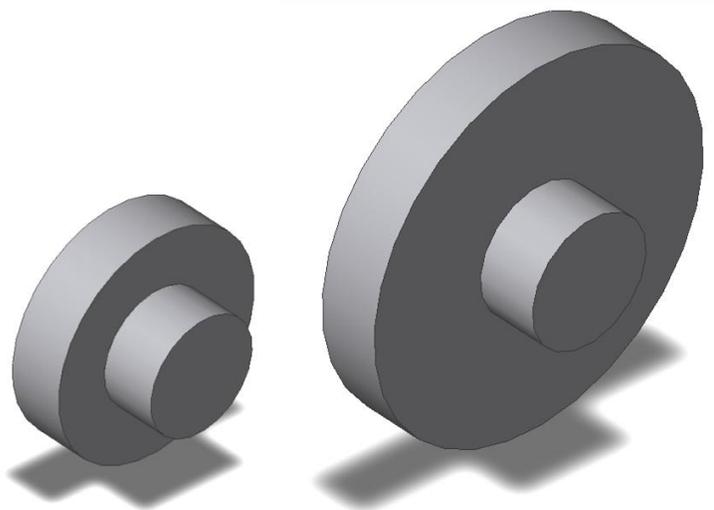


Figure 9: (a) Shaft with 2" larger diameter; (b) Shaft with 3" larger diameter

In the above example, constraints were the dimensional constraints and the concentricity between the circles. CAD packages also allow other types of constraints to exist between geometric elements such as perpendicularity and parallelism. Figure 10 shows parallelism constraints applied to the sketch lines of a window with different dimensions in a wall.

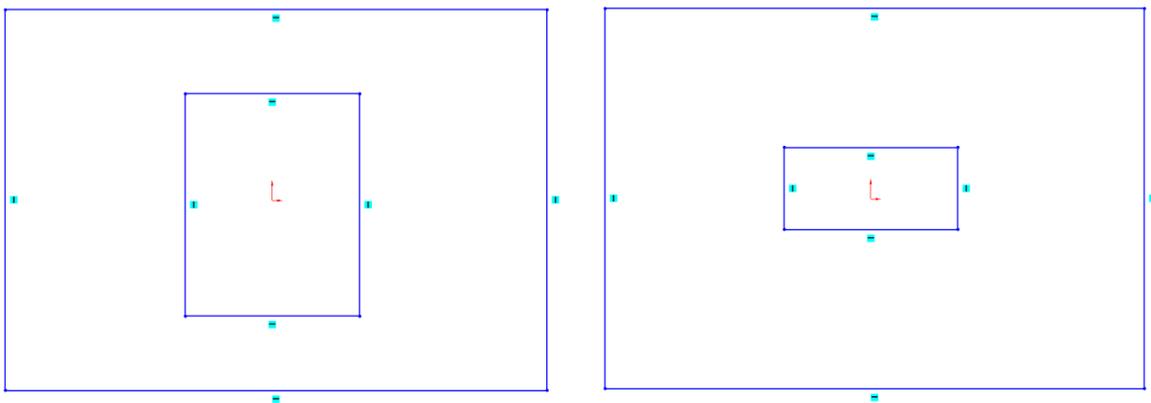


Figure 10 : Constraints applied to a window in a wall - Sketch

Design Paradigm

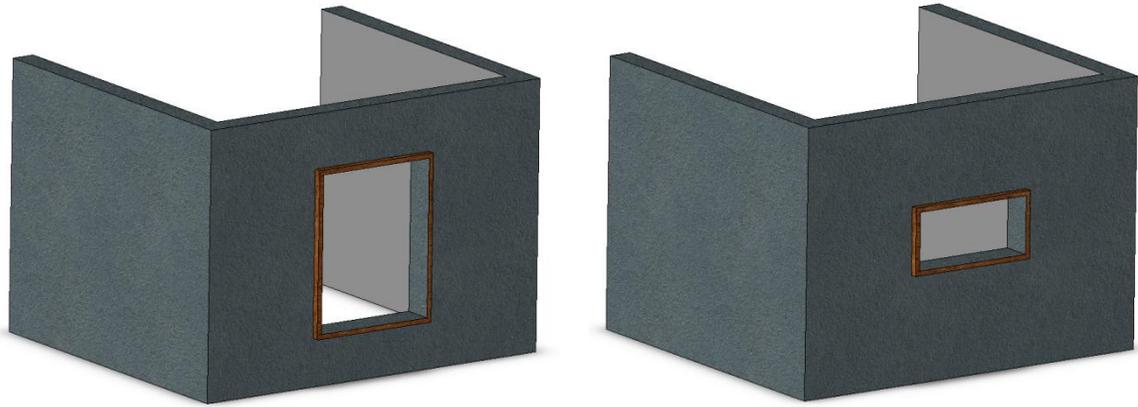


Figure 11 : 3D view of the rooms with the two different windows from Figure 10

The job of the CAD software simply is to provide the user with a list of available constraints and a GUI to help add and edit these constraints. Following the same approach, the web-based CAD system can be designed with the help of regenerative algorithms that help the user design models inside a web browser.

3.2.2 Configuration for regenerative algorithm inside the web application

For implementing the user-defined parametric features inside the web application, several different configurations for regenerative algorithms were considered. Due to the nature of a web application, the regenerative algorithms can be configured to execute on the client side only, or server side only or a mixture of both server and client procedures. Different configurations have their pros and cons and are discussed below.

Design Paradigm

Client-side only: (Figure 12)

In this kind of a setup, the user generates basic geometric entities and applies relationships between them using only client scripts such as JavaScript. The macros on the client side interpret the commands issued by the user and present the user with the results. During the designing step of the model, no information is transferred from the client computer to the server. The response of the application is fast and the user gets a feel of working in a desktop like environment. But the disadvantage is that the application can become sluggish when many algorithms run at the same time inside the user's browser. Also, due to the lack of a database on the client side, the application needs to build a database to save any user's actions.

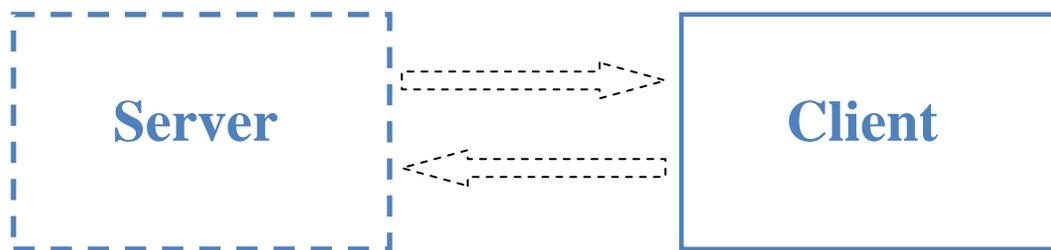


Figure 12 : Regenerative algorithm in a Client-side only setup

Server-side only: (Figure 13)

In this type of a setup, the regenerative algorithms run on the server side and depending on the user inputs, the server responds with the results. The advantage of server based regenerative algorithms is that it decreases the complexity of the client interface and the algorithms reside on the server protecting intellectual property. With a database on the server side, the user can save the work. The problem with this kind of a setup is that any delay in communication between the server and the client due to slow internet speed could affect the response of the application.

Design Paradigm

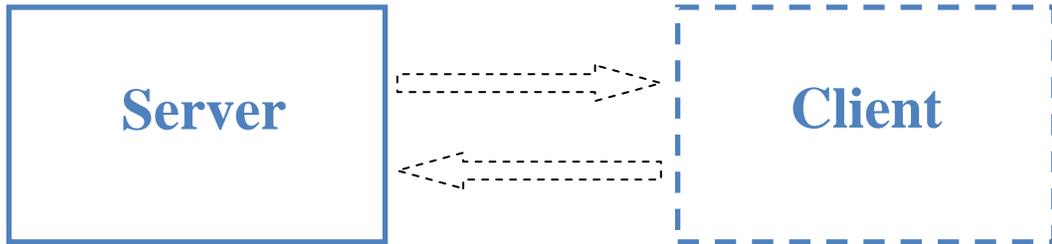


Figure 13 : Regenerative algorithm in a Server-side only setup

Client-server mixed environment: (Figure 14)

In this setup, the regenerative algorithms can be designed to incorporate the advantages of both the server based and client based setups. Also, by incorporating fewer functions on the client side and using the database support and larger processing power on the server side, the application can become more interactive and useful.

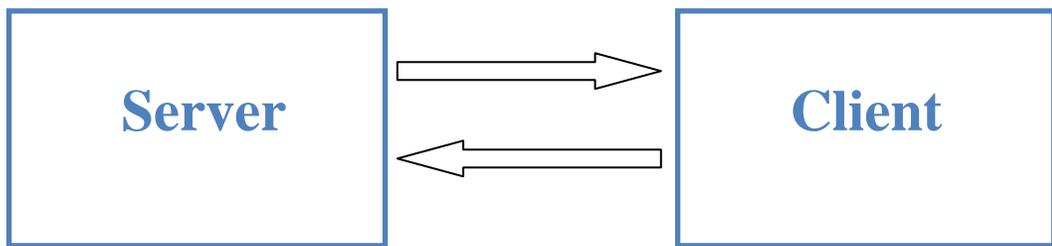


Figure 14 : Regenerative algorithm in a Client-Server mixed setup

3.2.3 Identification of key elements for user-defined features

The WatCAD system only allowed the user to use features that were pre-defined for the application. But the system did not have any tools that would help in creation of new features and storing them into a database for later use. The advantage of user-defined parametric features is that the user can create features that can be reused later thus saving designing time to a great extent. This also helps in reducing the number of built-in features that need to be offered to the user and keep the application simple and robust. Another advantage of user defined parametric features is that the user can share the features with other users of the same application. This enables contribution to the user-defined features database making the web based CAD application much better.

For developing such a user-defined feature system, few key elements need to be identified. These key elements or tools become the building blocks of the user-defined feature. These key elements are discussed below:

Basic geometric elements

Most features used in WatCAD are a combination of entities built with basic elements and operations such as extrusion, sweep, revolve and holes – Figure 15. All these features use geometric elements such as circles, rectangles or lines as basic elements in creating the feature. These basic geometric elements can be created using simple sketch tools outside a CAD system. The operations on the entities are more complex and can only be done within a CAD system. So it is proposed to offer a 2D sketching method on the client side and data collected is added to the geometric model of the part with operations such as extrusion and sweep on the server side.

Design Paradigm

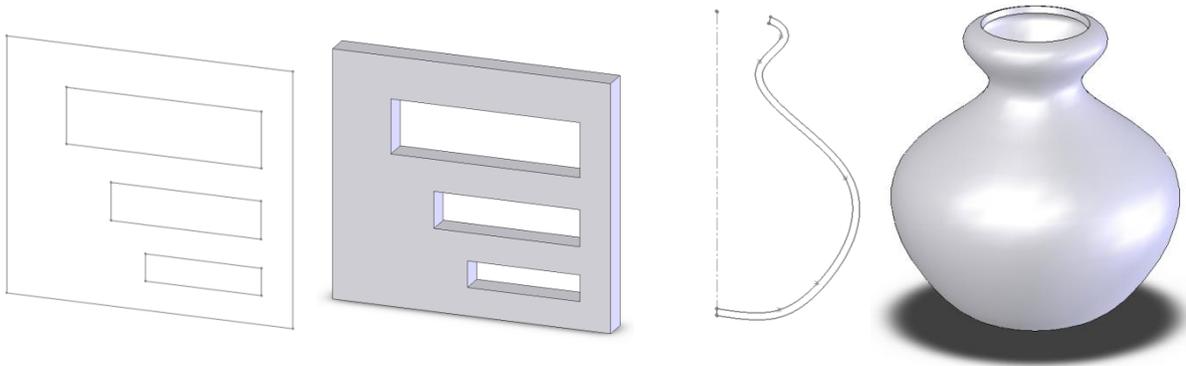


Figure 15 : Extrusion and sweep feature in a CAD software

Transformation tools

Along with the drawing tools, transformation tools also offer building of custom designs and shapes. Tools such as translation, scale and rotate allow modification of the basic geometric elements and these tools are provided as a GUI in the sketchpad.

Algorithm builder

In addition to the sketch elements, the 2D sketchpad also allows for adding of constraints or relations between sketch elements. This constraint based designing allows for easy modification and better designing of 3D models. To allow the user to be able to add such constraints and relations, the GUI provides the user with functions that have conditional statements similar to if-else statements or loops to constraint the elements in the feature. When the user adds such constraints, the information is sent to the server, analyzed and recorded in a database. This information analysis and management on the server side is managed by the algorithm builder or the macro generator.

Design Paradigm

By offering the user with these basic elements of sketching, the creation of user-defined features can be accomplished. The way to offer these features is to have this sketchpad available to the user.

3.3 Modified WatCAD system with user defined features

For implementation of the key ideas into the existing WatCAD system, a full featured set of sketch tools and transformation tools need to be offered in the form of a sketchpad inside the client user interface. The algorithm builder functions can be added to client side macros. In addition to these tools, a macro generator on the server side needs to be created to record the algorithms created by the user and give the user access to those algorithms later when needed. Figure 16 shows the modified architecture of the WatCAD system with the above mentioned elements added to it.

To start designing in the modified system, the user generates a 2D model using the sketchpad and adds constraints to the model. After the user has finished generating the design, he submits the design to the ASP manager through the client side macros. This method of transferring information to the server using client side macros instead of a HTML page post-back ensures that the user can still continue to work on the design in the sketchpad and at the same time run background functions on the server. This process can be done through AJAX [20] which is explained later. The macro generator on the server side receives the request from the client browser and makes a record of any user-defined features. The macro generator then initiates already created macros inside the ASP manager and the process of creating a 3D model is repeated as in the WatCAD system.

Design Paradigm

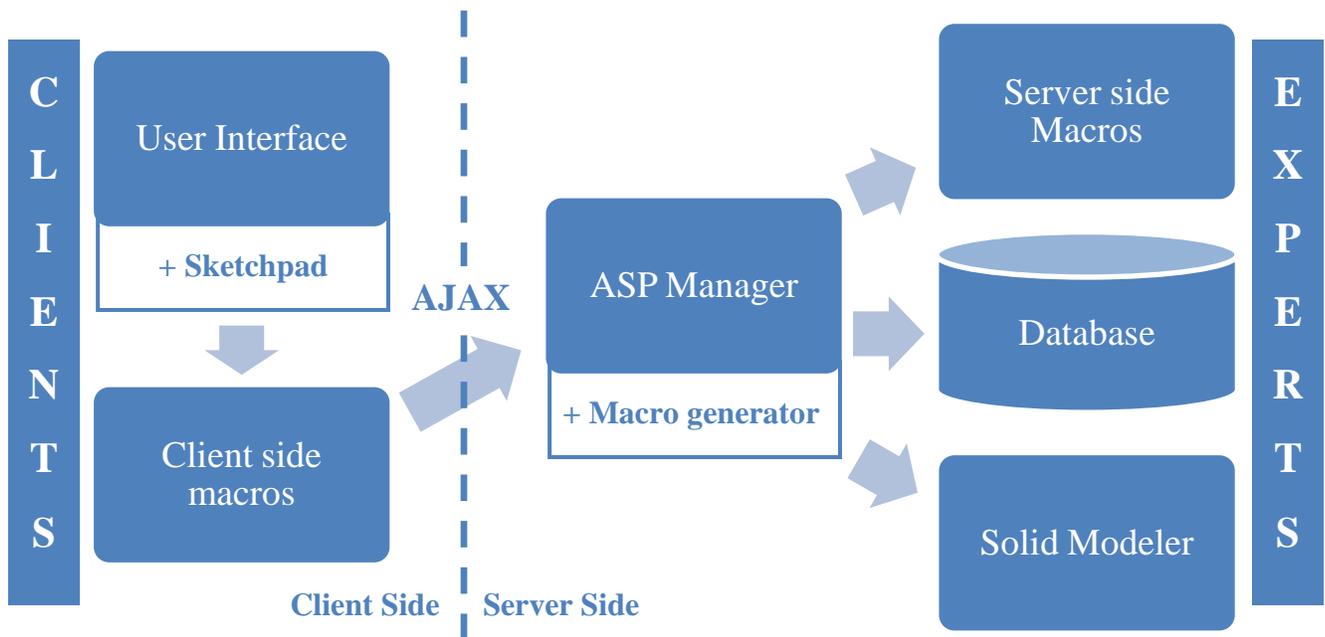


Figure 16 : Modified WatCAD system with user defined features

AJAX – Asynchronous JavaScript and XML [20]:

As mentioned earlier, the new architecture makes use of AJAX to transfer information between the client and the server. This type of a communication between the client and the server is different from the regular method of information transfer using HTML page post-back. In the case of AJAX, only parts of information are transferred to the server. The advantage of this method is that it reduces the time of information exchange between the server and client, and at the same time, the user browser allows the user to continue working on the GUI without any interruption.

In this chapter, the configuration of the original WatCAD system and its weaknesses were discussed. For user-defined features to be implemented into the existing WatCAD system, the

Design Paradigm

concept of regenerative algorithms and its architecture were discussed. Also, the basic tools for creating user-defined parametric features were identified and these tools were implemented into a modified version of the WatCAD system.

The next chapter will introduce different types of user-defined features that have been created. Although these features may seem like simple features, they have the flexibility to allow user to create complex shapes.

Chapter 4

Types of user-defined features

4.1 Introduction

Due to the lack of a solid modeler within a web browser, the generation and managing of a user defined parametric feature is complex. But since the purpose of a web application is not to imitate a solid modeler, a different approach needs to be followed to generate user-defined features inside the web application.

4.1.1 Types of user-defined features

For user-defined features, the features need to be structured based on relationships between geometric entities. This will enable the features to be implemented easier into the designing model. Different types of user-defined features can be implemented in to a system where basic tools for creating geometrical models are used as primitives.

Type I user-defined feature: Symbols

A simple user defined feature can be defined as shown in Figure 17 (a). The object is made up of basic elements which in this case are six splines with positioning as relationship between

Types of user-defined features

each of them. When the user designs such a model, the relative position between the elements is captured by macros and stored. This allows for easy instancing of the model in the design table to make the University of Waterloo logo as shown in Figure 17 (b).

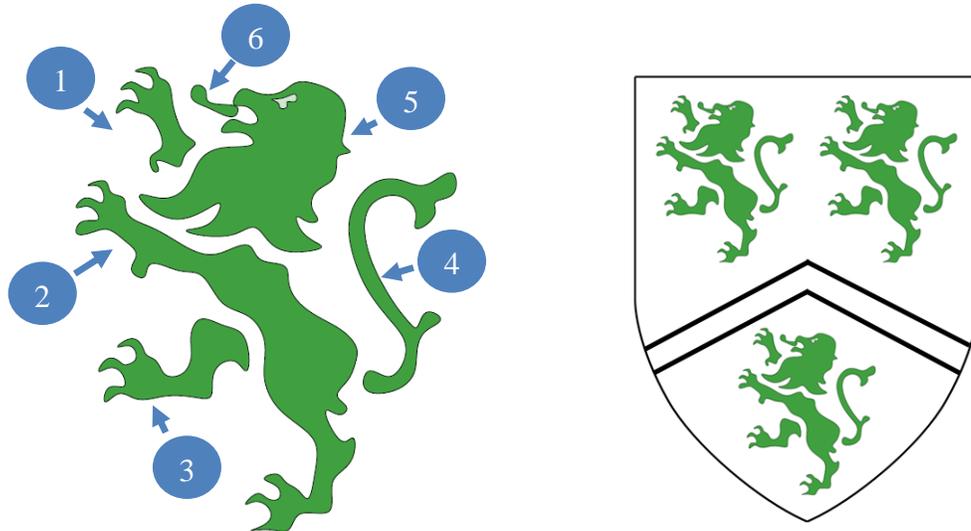


Figure 17 : (a) Simple user-defined feature with six splines; (b) Three instances of the user defined feature I in (a) to form the University of Waterloo logo

The way to implement and process this type of feature is to define the elements in the model as a group. When an instance of the feature needs to be generated, transformation matrix can be applied to all the entities to give the symbol an orientation and scaling.

A Type I feature does not depend on any other feature or element in the drawing area of the sketchpad or there are no parametric relationships with other external elements. This means that modifying the feature does not affect other elements.

Types of user-defined features

Type II user defined features:

Type II features allow for adding external relationships (constraints) between the geometric elements. The processing structure for this type of feature gets complicated as the algorithms should be able to build and manage the associations between the elements.

Unlike the constraint adding model in major CAD modelers, a progressive processing method is used in this model to add constraints or relationships. This means that when a constraint is added between two parameters, one of the parameter behaves as the parent, and the other as a child as shown in Figure 18. This system is advantageous because changing the parent parameter updates the child parameter but not vice versa. This allows for quick relationship building between the elements in the design. As seen earlier in Figure 11, when the dimension of the window in the wall is changed, the dimension of the wall does not change.

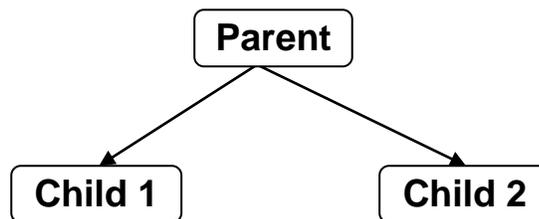


Figure 18 : Type II user defined feature

Following examples also illustrate this concept.

Circles are defined with the parameters – center (cx, cy) and radius (r) . If there are two circles then the parameters can be represented by $[(cx1, cy1), r1]$ and $[(cx2, cy2), r2]$ for circle 1(C1) and circle 2 (C2) respectively as shown in Figure 19 below.

Types of user-defined features

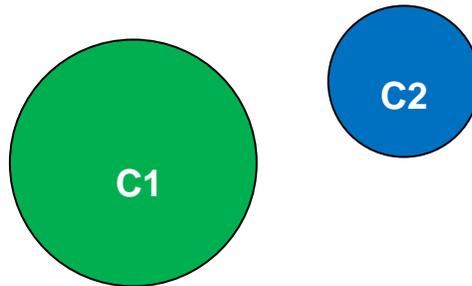


Figure 19 : Circles C1 and C2 with no existing relation between them

Since both the circles do not have any existing relation between them, modifying one of the parameters for circle 1 will not affect circle 2 as shown in Figure 20 below.

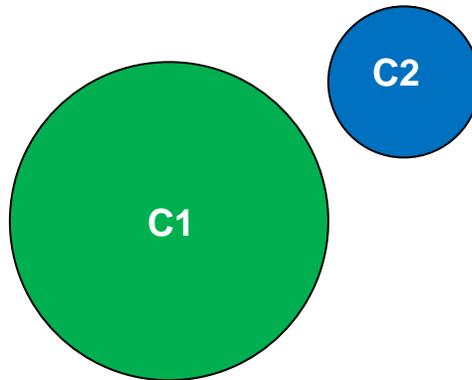


Figure 20 : Modification of C1 does not affect C2 as no relation exists between them

A relation can be added to circle 2 such as $r_2 = r_1$ equating the C2 radius with C1. In this case, radius of C2 acts like the child parameter and radius of C1 as the parent parameter so whenever the radius of C1 is changed, radius of C2 changes automatically as shown in Figure 21.

Types of user-defined features

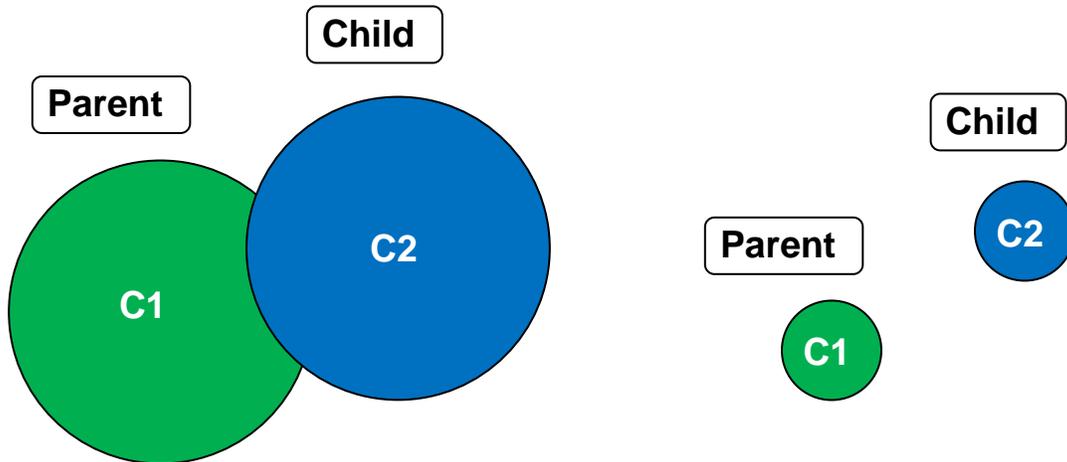


Figure 21 : Type II user defined feature - relation added between C1 and C2

Similar relationships can be added and modified to get the desired result. One of the rules in adding this relation is that circle 2 remains a child to circle1 as long as the relationship is not changed.

Another type of relationship building feature is the concept of propagating constraints through a tree based structure as shown in Figure 22.

Types of user-defined features

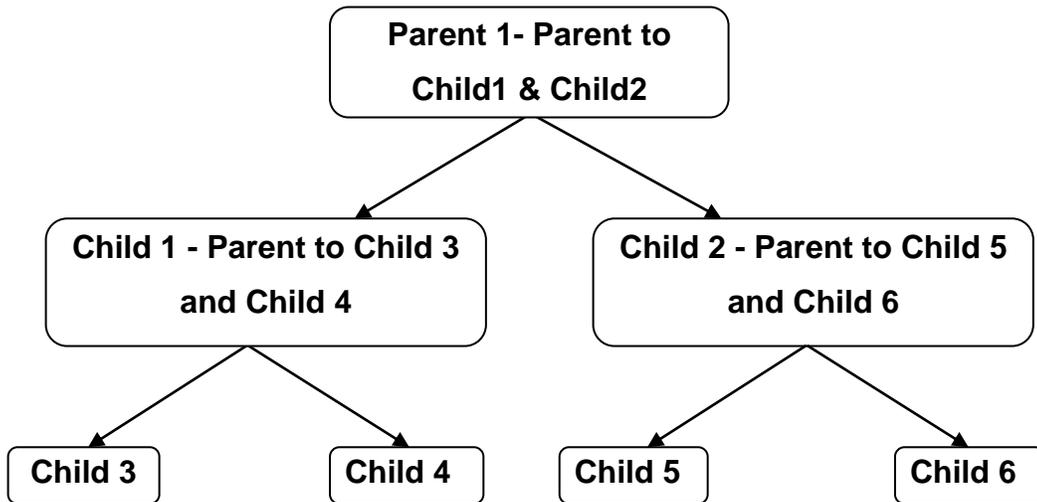


Figure 22 : Propagating type II user defined feature in a tree structure

For example, if there are three circles and circle 2 is a child of circle 1 and circle 3 is child of circle 2, then changing circle 1 changes both circle 2 and circle 3 following the relationships they maintain. The following example illustrates this idea.

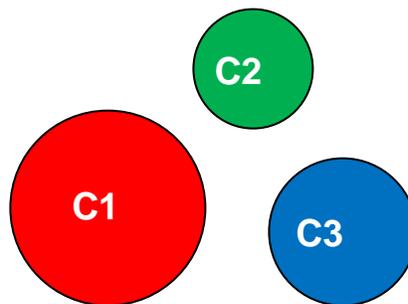


Figure 23 : Three circles C1, C2 and C3 with no existing relation between them

Types of user-defined features

The three circles in Figure 24 have different radii as a relationship between them do not exist. When r_2 is set to be equal to r_1 and r_3 is set to be equal to r_2 then the result shown in Figure 25 is obtained.

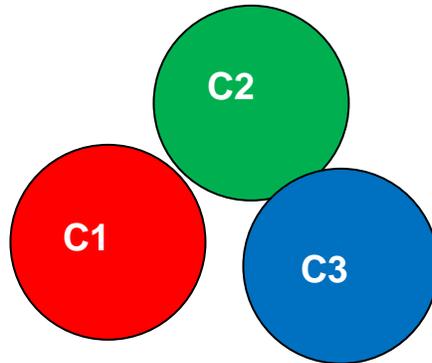


Figure 24 : Type II feature relation added to the three circles C1, C2 and C3

Expressions as in the above example need not be just simple blocks but can include even other mathematical expressions such as $x*y$ and $x + y$ depending upon the programming flexibility.

Type III user defined features

Type III features allow for automatic conversion of relationships to parameters and storing of these parameters for the user to instance the procedure rather than just the object. The advantage of this type of feature is that the procedure can be applied to any geometric element and the user does not have to repeat the procedure to different elements.

For example, a feature called “flower” can be created that has as parameters the number of petals and the angle between each petal as shown in Figure 25(a). Instances of the procedure are shown in Figure 25(b) and Figure 25(c).

Types of user-defined features

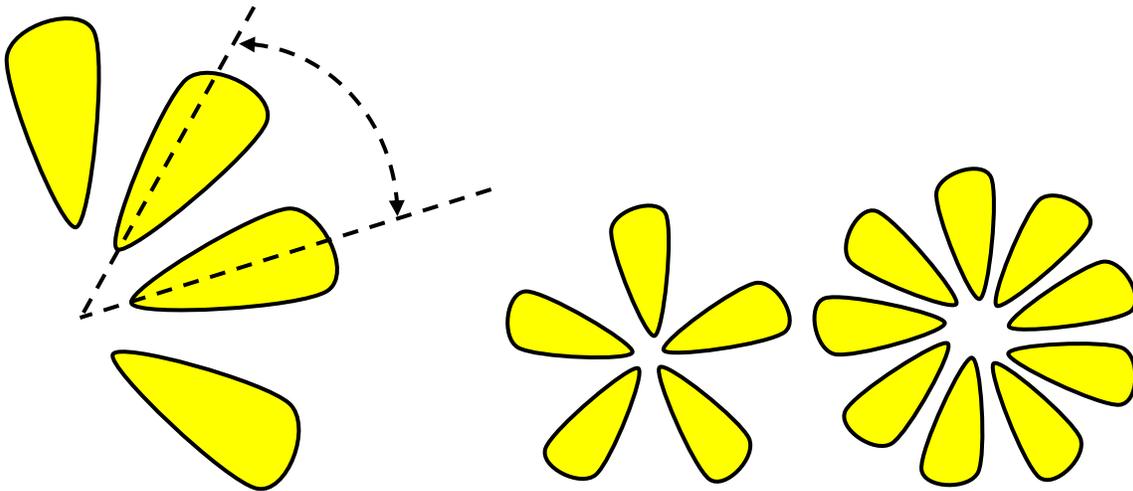


Figure 25 : (a) Petal in a pattern with “angle” and “number of petals” as parameters; (b) Pattern applied with “number of petals” = 5; (c) Pattern applied with “number of petals” = 9

The method for building a Type III feature is that the user starts generating a geometric element such as a circle or a spline and identifies it as the starting element of the design. Using pattern designing, the user creates a pattern with the geometric element. For example, when creating the flower feature the user adds “copy” and “rotate” as parameters to the “petal” generating the “flower” pattern. Once the user is done adding the parameters, the regenerative algorithm converts the procedure into a feature with the parameters added inside the pattern. This method then is stored into the database and is instantiated when required. A process flow diagram is shown in Figure 26.

Types of user-defined features

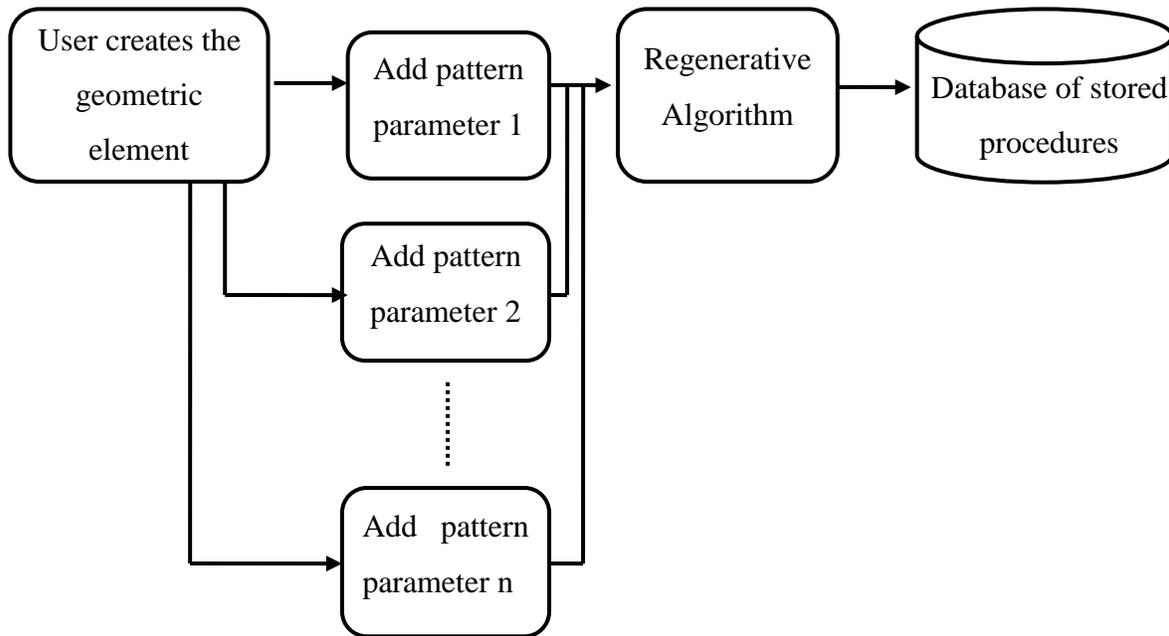


Figure 26 : Type III user defined feature procedure

4.2 Design of user defined feature system

The data flow management of the user-defined feature system was explained in earlier chapter in context of the application design itself. To create or manage user defined features, a database structure needs to be implemented to allow the user to save the newly created feature and generate instances of the feature in the application. Using the knowledge of the different systems involved in the web application, a concept was introduced that allows the creation of user defined features.

4.2.1 Adding relationships and constraints

As seen earlier in the three types of user defined features, the key concept was to allow users to add relationships and constraints also known as expressions, between the basic sketch elements. The programming structure shown in Figure 27 for adding constraints was designed to be implemented into any application.

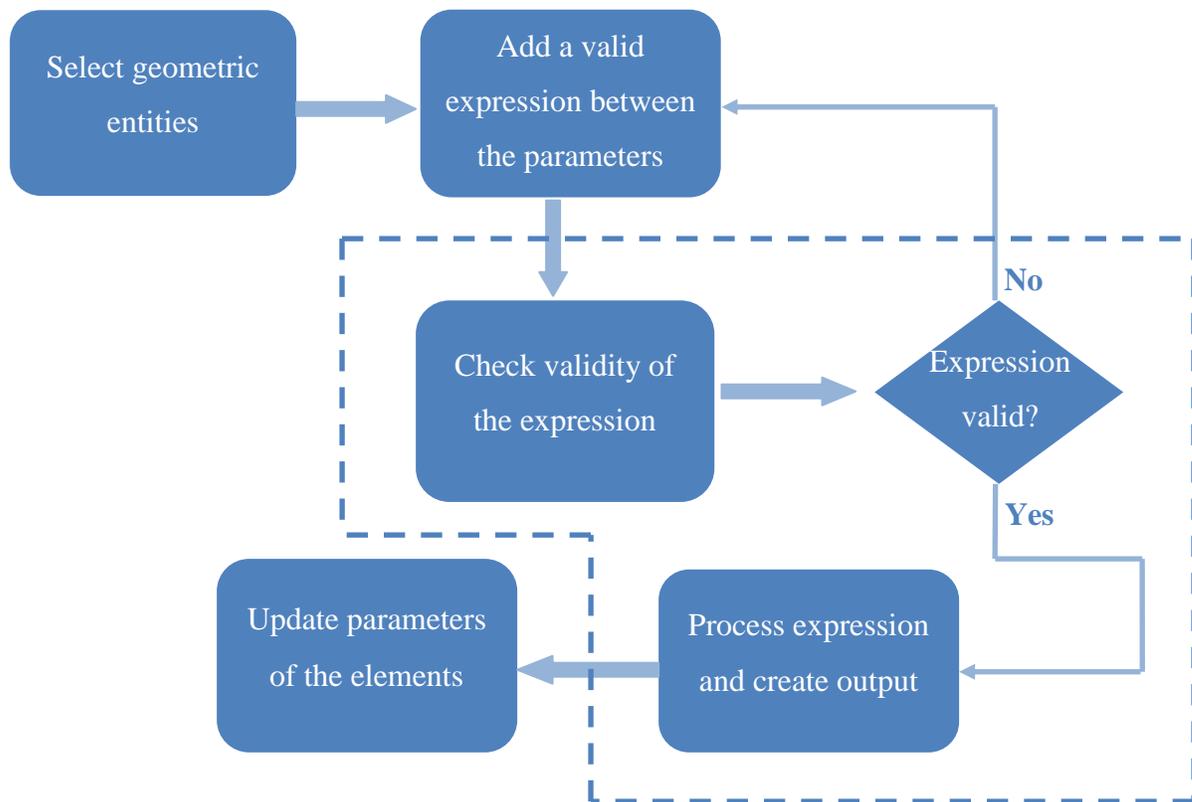


Figure 27 : Programming structure for adding user defined feature

In Figure 27, the main interpretation of the client's commands and execution is done by the regenerative algorithm that was discussed in the chapter 3. To help give the regenerative

Types of user-defined features

algorithm a syntax and a structure, a language or a platform needs to be chosen for the application to be implemented in.

The next chapter deals with the implementation of user defined parametric features in a real world application with the help of the architecture, process flow and types of user-defined features discussed so far. The chapter also discusses the different platforms and languages that were considered for developing the application.

Chapter 5

Implementation

The main goal of the automated designing of user-defined features was to allow a user to customize an application. The program iSketche™ was developed to test user-defined features where clients use a web application to design their own wooden signs. The iSketche application brings automatic CAD designing of wooden signs to the general public who do not have the knowledge of using complex design packages for creating signs.

Wood signs can be designed on any CAD or modeling system. However, most of these systems are seen on proprietary machines and require training. To offer the functionality of these CAD systems in a web application requires a large bandwidth that is not offered by the internet technologies available today. In light of this the web based sign design system will require a subset of capabilities available in a CAD package. As discussed in the previous chapter to implement the user defined features, a client based web technology was selected. The block diagram showing its components is given in Figure 28.

Implementation

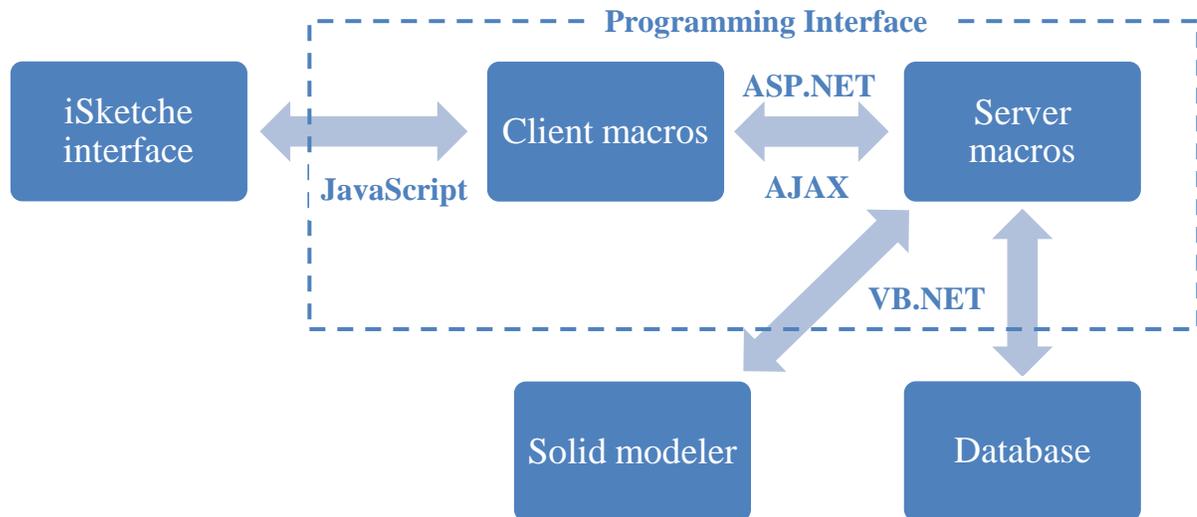


Figure 28 : Architecture of the iSketche application

The iSketche application was programmed to connect to the server algorithms that ran the solid modeler in the background and provided the outputs to the web browser. The server application creates geometric models and a preview of the model generated is sent to the web browser. At this point the user has the option to further modify the design by changing the parameters or adding new shapes to the existing design. To take user feedback and communicate it to the server applications a combination of ASPX forms and a client side sketchpad application were used. The sketchpad application is also discussed in section 5.2.

5.1.1 Prototypes

To gain experience with wooden sign making and to understand the relevant issues, few signs were modeled in SolidWorks 2006 and machined on a 3-axis CNC mill machine with a spindle speed of 5000rpm and feed rate of 1500rpm.

Implementation

One of the observations made from these prototypes was that the sign design cannot have any features such as holes or spacing between each feature smaller than the machining tool diameter. If they are present the tool will not be able to penetrate into the smaller areas and leave the wood un-machined at those spots. So when designing a logo, the features have to be big enough to be machined properly

Another observation made was that the features shouldn't be thin. The reason for this is that due to the material properties of the wood, any feature that is thin can chip off during machining. To avoid this issue, the logos were designed to not have any sharp edges and the fonts with no serifs were used. An example of a chipped text in one of the prototypes is shown in Figure 29.



Figure 29 : Machined wooden sign with chipped lettering

5.2 Development of web-based sign design application – iSketche™

The goal of the iSketche system is to allow users to design customized wooden signs using a web based design interface. Custom wooden signs can be of different shapes and sizes and incorporate different elements such as logos, text, border and background according to the user's choice. The domain of possible wooden signs is infinite. A full-fledged CAD system would be required to design a random member of this domain. Examples in this domain are shown in Figure 30. These signs have variable shapes, backgrounds and borders.



Figure 30 : Signs with varying shapes, backgrounds and borders

For the purpose of this project a subset of the domain was selected and since the focus of this project was demonstrating user based feature design, the size, shape and border were fixed. To demonstrate the concept of user based features, the user was allowed to design custom logos. iSketche accomplishes this goal by providing a graphical user interface that is interactive and tools that are easy to use.

In the iSketche application, the selection of shape and the size of the wooden sign were fixed to a rectangle and $85\text{cm} \times 65\text{cm}$ in dimension respectively. This was done to limit the

Implementation

user to a single option for the shape of the sign but it could be easily expanded to the pre-defined selection of shapes and sizes. Similarly the border was also fixed to a circular swept border following the edge of the sign. Again, this option can also be extended to a selection of borders such as twine or a profile tool border. A flat background is considered in this work. iSketche offers the user an option of selecting a predesigned logo or to design their own logo and convert it into a user defined feature if they desire.

As the focus of the iSketche application is to allow users to design their own logos along with the option of adding personalized text and placing them anywhere on the sign, basic drawing tools and text boxes were provided. The drawing tools were limited to 2D geometric elements such as circles or lines which are discussed later in the chapter. The text box capability was limited to selection of a few common fonts that do not have serifs such as Arial or Verdana.

For generating the 3D model of the designed sketches, the user is given the option to either emboss the sketches with three different emboss heights or sweep a circular profile along the sketches with three different sweep widths. This simplifies the selection process for the user to generate the 3D shape and at the same time keeps the demonstration simple. With the help of the server side scripts and a solid modeler, the iSketche system can take the user created sketch and create 3D models of the sketch in the wooden sign.

5.2.1 Functionality

To implement the online wooden sign system, different functionalities are needed. The user (client) needs to create the logo, add text, etc. All the features required for sign design are geometric in nature. So the best way to interact with the user in designing these features is to use a sketchpad. It has a main drawing board to sketch logos, a menu driven logo selector with

Implementation

capability of uploading pre-made designs, a text box feature, etc. The user selects a geometric entity such as a line and uses the design area to draw the line. The end points of the line can be defined by mouse clicks or by keyboard input. After one entity has been created other entities can be added as required to build the desired feature. The drawing entities that were chosen for the iSketche application are shown in Figure 31.

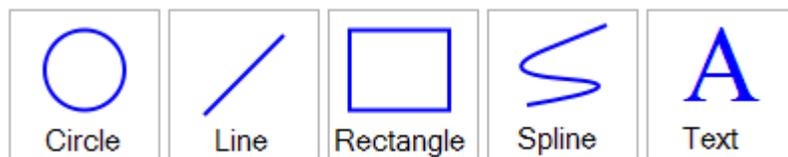


Figure 31 : Drawing tools in iSketche application

These simple drawing tools are adequate to demonstrate user defined features within the iSketche application as discussed in the earlier chapter.

All graphical interfaces must provide a tool that allows a user to erase a misstep. Such an undo feature is provided. The undo feature basically deletes the data entry from the database and updates the graphical screen. Furthermore to facilitate faster designing, transformation operations like copy, scale, translation, or rotate as shown in Figure 32 were implemented in the iSketche system.

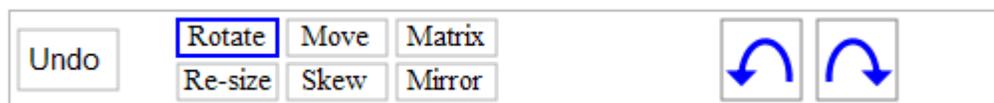


Figure 32 : Editing tools in iSketche application

Implementation

After the user is satisfied with the design and is ready to build the 3D model, the data is collated by the client script and is sent to the server side CAD for creation of a sign model. The sign model can then be visualized on the client side inside the drawing board as a 2D image or using a 3D viewer. The final iSketche sketchpad designed is shown below in Figure 33.

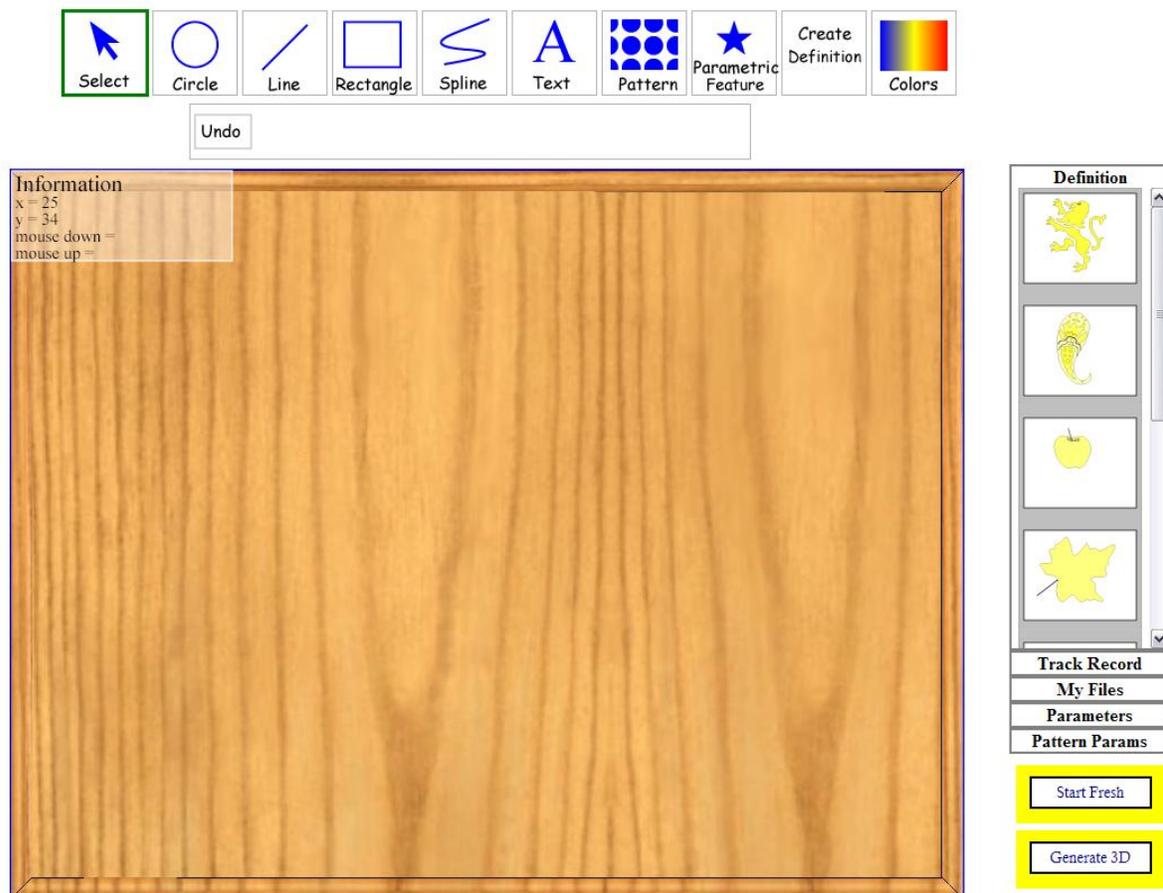


Figure 33 : iSketche sketchpad design

Implementation

5.2.2 Components of the iSketche system

The development of the sketchpad was divided into modules. The main modules were the creation and display of geometry in iSketche, data transfer between the client and server, server macros, and generation of 3D models.

Creation and display of geometry

As discussed earlier, the general drawing and editing tools were required to allow the user to create 2D designs. A sketchpad application that provides this functionality was designed after detailed analysis on available web technologies.

In many research projects that involve building web based GUIs, Java [21] has been the first choice. Java has a simple programming structure and is an open source language making it an ideal option for building the web-interfaces. Adobe® Flash [22] was also considered in this scenario as it is used for developing interactive web based applications.

But for the iSketche application SVG (Scalable vector graphics) [23] with JavaScript [24] as the scripting language was chosen to be used as the sketchpad interface. SVG is a 2D image format in XML (Extensible Mark-up Language) [25] that can be used to display both static and dynamic images over the web. The reasons for this choice are discussed below:

- **Vector based graphics** – Unlike other image formats, SVG is completely based on vector based formats which means that the image does not lose its resolution upon scaling. Also, the basic elements such as the circles, lines and splines are already built-in inside SVG and can be defined using parameters which is a requirement for building parametric features

Implementation

- **Support for scripting** – SVG supports JavaScript, which is required for dynamic modification of elements, a basic and important requirement for the application
- **XML format** – XML is a well structured format that is both simple and human readable which allows for easy modification of the file using any text-editors
- **Supported by all major browsers** – SVG is supported natively by most of the browsers and can be viewed on other browsers with the help of external plug-ins like the Adobe SVG viewer [26] plug-in for Internet Explorer
- **Open source** - SVG is supported by the huge open source community and continues to gain support in developing next generation web graphics

To implement SVG into the sketchpad application, the client macros were written using JavaScript. Everything including the sketch tools, editing tools and the drawing area of the iSketche application was designed using SVG. The JavaScript macros capture user mouse inputs and output the results to the user browser. For example, when the user clicks the circle command, a JavaScript function *start_circle()* initiates the circle object. When the user starts drawing inside the drawing area, a circle is created dynamically with information captured from the user's mouse movements.

Client - Server communication

When it comes to transferring information from the client web application to the server side geometric modeler, various languages as discussed in earlier chapters were analyzed. For the iSketche application, ASP.NET AJAX was implemented to transfer information between the server and the client.

Apart from transferring user created designs between the client and the server, AJAX was used to activate the 3D generation of the part on the server side. When this activation

Implementation

command is initiated by the client browser, the server validates the command and starts talking to the solid modeler and the database. After the model is generated, the server macro responds to the client macro with a snapshot of the model. At this point the user is presented with the image in his drawing area. All during this process, the user is presented with a “Loading” image to indicate the progress of the work.

Generation of 3D models

The server side scripts as mentioned earlier were written in VB.NET [27]. These scripts were written inside the code-behind file for the webpage. For example, the VB.NET code for *Default.aspx* is stored in *Default.aspx.vb*.

The code for generating 3D models for the iSketche application was written in one of the code-behind files. The scripts are initialized by the client macro when the user clicks the *Generate 3D* button on the web-page as shown earlier in Figure 33. The functions that were created inside the server scripts are discussed below. Also these functions are mentioned in the order in which the data flows when generating 3D objects inside the solid modeler.

Load SVG file

The SVG file that was created by the user using client scripts is loaded by the server script for parsing. The library used for this operation was LINQ – (Language Integrated Query) [28] which allows for querying XML files.

Implementation

CAD Initialization

After the SVG file has been loaded and the contents in the file are verified, the CAD Initialization function is called. This function creates an instance of the solid modeler being used and creates a new file inside the modeler for generating the 3D model.

Parse SVG and Create 3D model

This function starts parsing the SVG file and looks for the different geometric elements created by the user. These elements include circle, line, spline, rectangle, polyline and text. When the function finds any of these elements, appropriate functions to create these elements are called to sketch the elements inside the solid modeler. For example, to create a circle, a function called *create_circle* is called. This function adds a circle element into the solid modeler's sketch with the parameters specified by the user. A 3D model is created out of this sketch by either extruding the sketch, or sweeping the profile of the sketch. This option is also specified by the user inside the client sketchpad.

Apply material and save snapshot

After the 3D model is created, the solid modeler applies a material property and saves a snapshot of the model. The snapshot is sent to the user browser and is displayed inside the drawing area of the sketchpad. At this point the user can add more elements directly to the image and generate the 3D model.

The above section gives a complete description of the components inside the iSketch application and inter-connectivity between them. For implementing the user-defined features, a framework for creating and storing the user-defined features using these components was built and is discussed below.

5.3 Implementing user defined parametric features

To design user defined parametric features in SVG, functions in JavaScript were programmed to capture the user inputs from the SVG drawing area and store them into the SVG document. The types of user defined features implemented into the iSketche system are discussed below.

The first type of user-defined feature was the symbol feature. To be able to create the symbol feature, the user draws a sketch inside the drawing area. After the sketch is done, the user clicks on the “*Create Definition*” button inside the elements toolbar. At this point, the JavaScript functions collate the entities inside the sketch and create a container for these sketch entities. By using the built in property known as definitions (<defs>) [29] inside SVG, the sketch entities are stored inside the SVG document. <defs> property acts as a container for sketch elements that are only rendered when referenced inside the SVG document. An example is shown Figure 34 (a):

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="850" height="650"
  xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
  <desc>Local URI references within ancestor's 'defs' element.</desc>
  <defs>
    <g id="c_spln1">
      <path d="M253,364 C247.3,353.2 234.4,345.5 220,342 C198.5,336.7 173.5,340.7 163,328
        C158.5,322.64 156.6,314.4 152,306 C141.2,286.5 115.7,266.2 123,257
        ...
        C254.1,401.1 261.09,379.45 253,364"
        stroke="black" fill-opacity="0.5" fill="yellow" transform="">
      </path>
      <line id="line2" x1="255" y1="375" x2="98" y2="495" stroke="black" stroke-width="6" transform="" />
    </g>
  </defs>
  <use x="0" y="0" xlink:href="#c_spln1" />
</svg>
```

(a)

Implementation

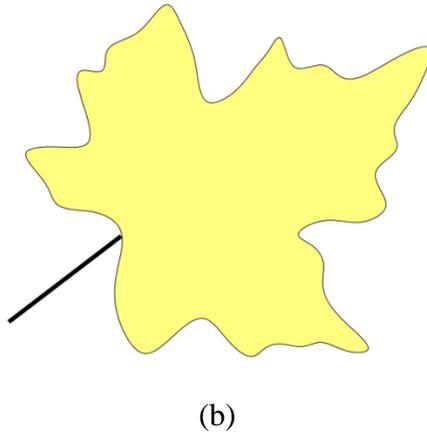


Figure 34 : (a) Sample code of a SVG document with *<defs>* element; (b) generated shape from code in (a)

At this point, a reference to the *<defs>* tag is created inside the *Definitions* tab on the right hand side of the application to allow the user to instance the type I user-defined feature.

Type II user defined features allow adding of relationships (or constraints) between the sketch elements. Since SVG defines geometric shapes in terms of parameters, these parameters can be changed to update the already created sketches inside the drawing area of the iSketche application.

To generate type II features, the user creates a geometric element inside the SVG drawing area. After the user is done drawing the sketch, a JavaScript function creates a HTML tag inside the Parameters tab that has the properties of the sketch elements created by the user. For example, if the user creates a circle and a rectangle, the properties of the elements are displayed to the user with their values as shown in Figure 35.

Implementation

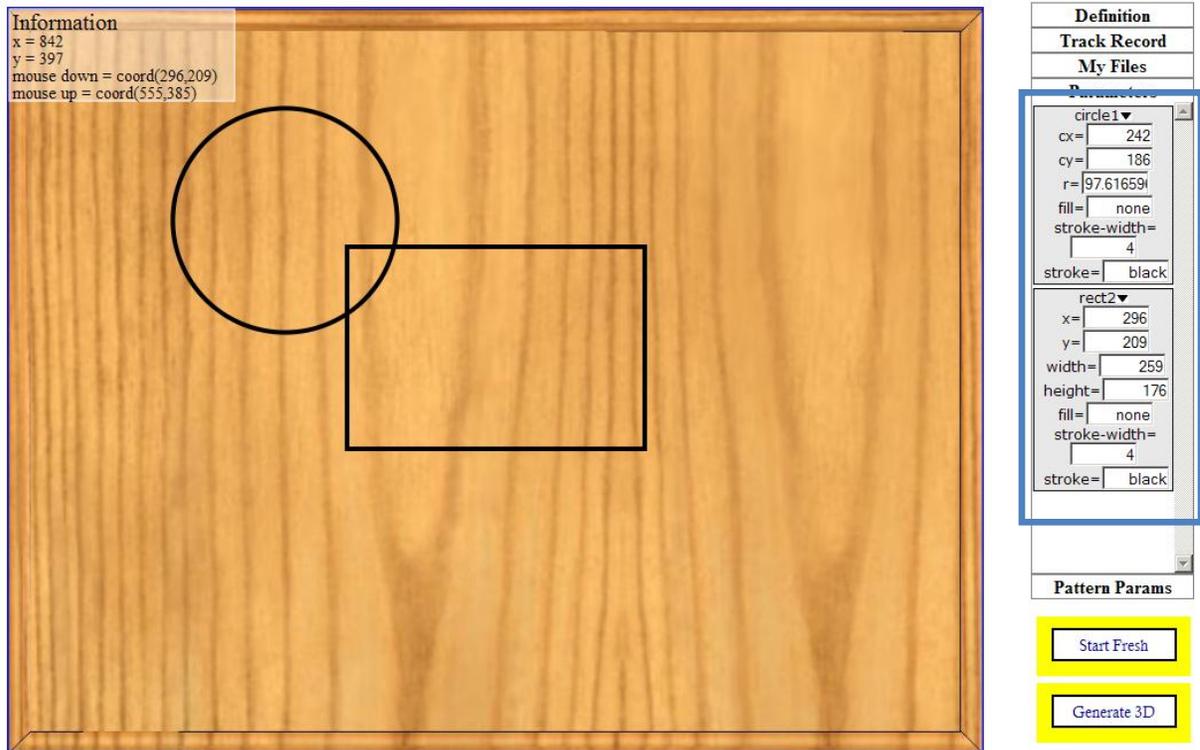


Figure 35 : Sketchpad with a random circle and rectangle drawn and their properties in the “Parameters” tab on the right

The model inside the drawing area can be updated by changing the parameter values of the individual geometric elements. Apart from just changing the values, the user can also add relations in the input fields. An example would be, changing the center of the circle to coincide with the top left corner of the rectangle by equating the value field of cx to $rect2:x$ and cy to $rect2:y$ as shown in Figure 36 below. A JavaScript function updates the model immediately with the result.

Implementation

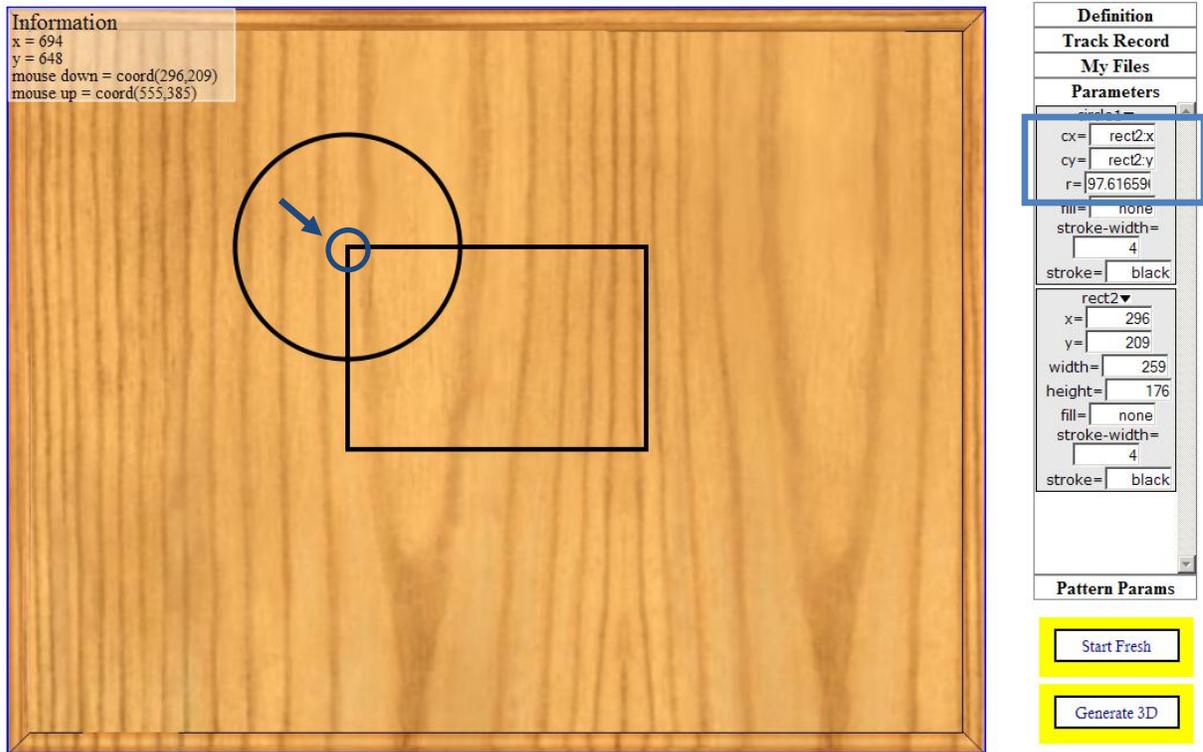


Figure 36 : Relation added between the circle and rectangle from Figure 35

As seen here, the relationship added was just a simple expression that merged the center of the circle to one of the corners of the rectangle. Any change in the position of the rectangle also changes the position of the circle with it.

Type III user defined features were the pattern features and to implement it, a toolbar as shown in Figure 37 was developed. Using this toolbar, the user can add patterns such as linear, rotate or re-size (scale) to the sketch elements inside the drawing area. The steps followed to create the pattern are saved into a database on the server and can be used to apply to other sketch elements. In the iSketche application, the pattern feature was not completely implemented but this feature would definitely simplify designing to a great extent if offered.

Implementation

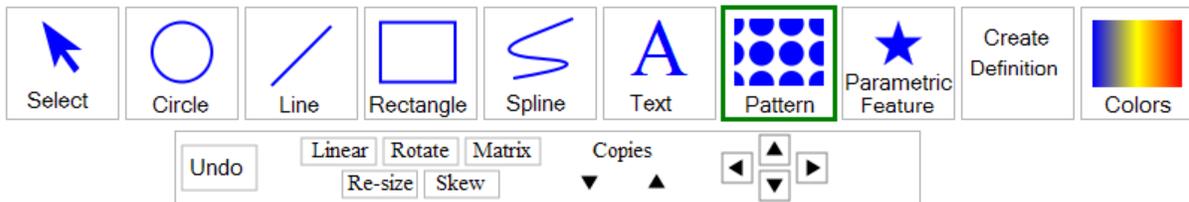


Figure 37 : Pattern tools in the iSketche sketchpad

5.4 Example

Using the iSketche application and with the help of user-defined features, a test case of customized wooden sign was selected. The task is to design the 2010 Canadian Winter Olympics logo [30] also known as Ilanaaq – Figure 38 using the iSketche application.



Figure 38 : 2010 Canadian Winter Olympic logo [30]

Implementation

The elements that can be identified in the logo are the two images at the top and bottom and a text at the center. One of the methods to create this logo is to create all the elements inside the drawing area of the iSketche application and generate a 3D model. But to personalize the logo or use it multiple times, the logo has to be saved to be used more than one time.

With the help of the user-defined features, the elements of the logo can be created and saved inside the Definition tab of the iSketche application. This will allow for easy instancing of the logo for creating a personalized Canadian Winter Olympic wooden sign. To do this, each of the elements are created as shown in Figure 39 and saved as a Definition which is the Type I user defined feature.

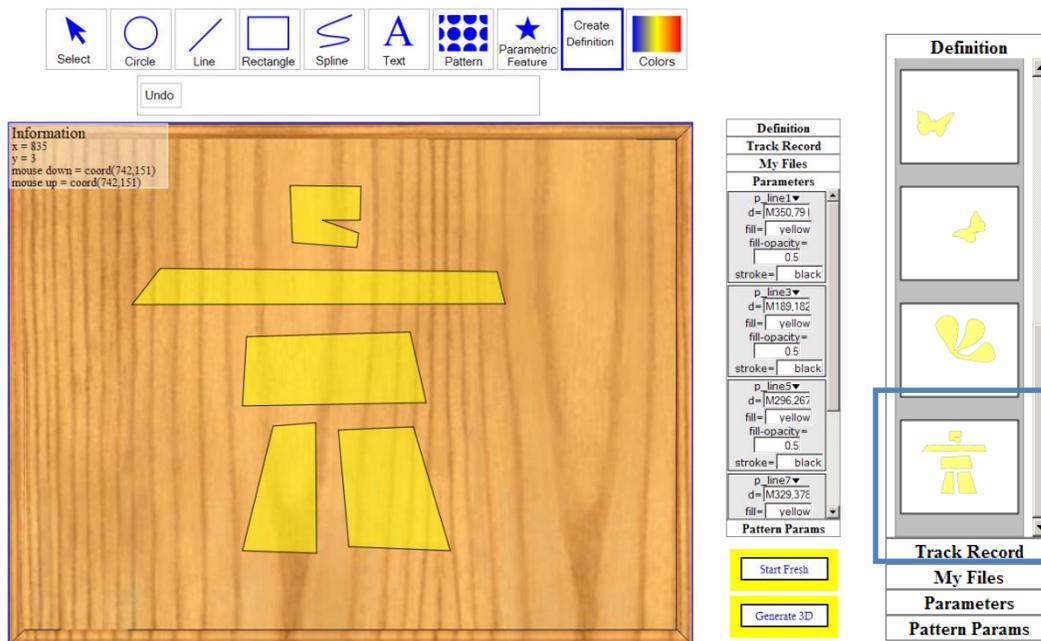


Figure 39 : Drawing the first element of the 2010 Canadian Winter Olympic sign and saving it as a Definition

Implementation

For the second logo, it was tough to manually draw the Olympic rings by maintaining the same radii and relative positions between the circles. To resolve this issue, five random circles were created as shown in Figure 40 and by using Type II user defined feature, the parameters of the circles were modified.

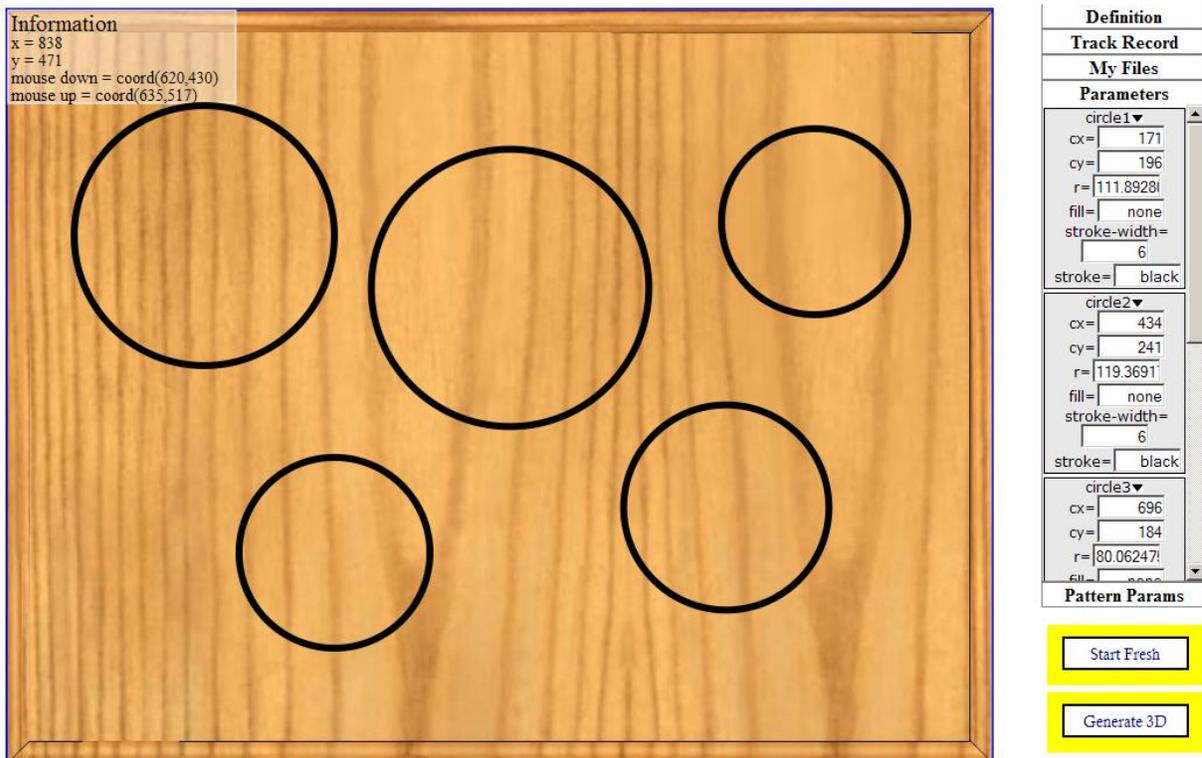
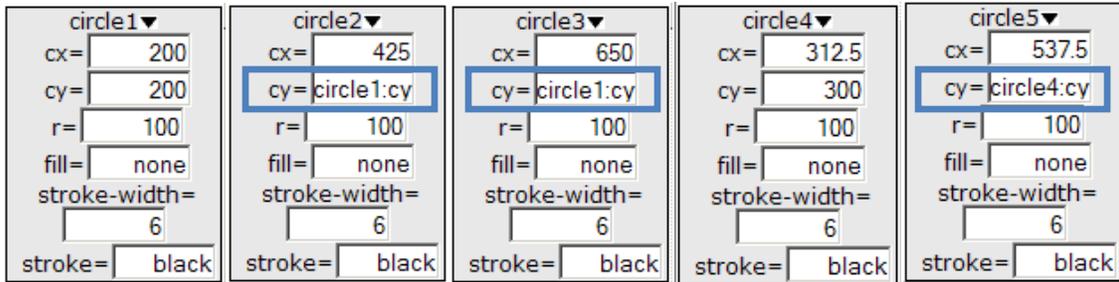


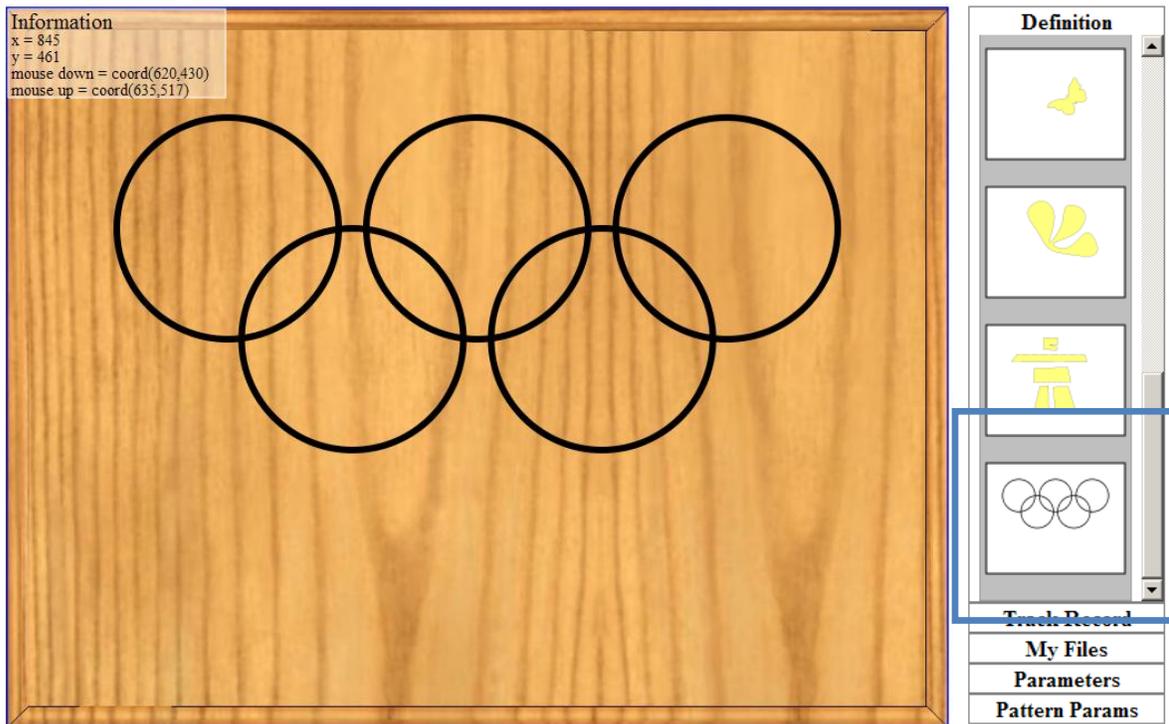
Figure 40 : Drawing 5 random circles for creating the Olympic rings

By modifying the values of the radii and the cx, cy values of the circles' center coordinates inside the Parameters tab, the final logo of the Olympic sign was created as shown in Figure 41.

Implementation



(a)



(b)

Figure 41 : (a) Adding relations between the circles in Figure 40; (b) Sketchpad updates with the new relations between the circles and adds it to the definition

Implementation

After creating the definitions, an instance of these elements were created and added to the design area along with the text – “Vancouver 2010”. A 3D model was automatically generated by the geometric modeler on the server side and was presented to the user inside the drawing area itself as shown in Figure 42. A snapshot of the actual generated 3D model inside the solid modeler is also shown in Figure 42.



Figure 42 : Creating an instance of the “Vancouver 2010 Olympic logo” and the generated 3D model

Implementation

The Olympic ring Definition was also used as a background for the wooden sign and a new sign was created as shown in Figure 43(a) with the 3D model snapshot in Figure 43(b).



Figure 43 : Creating another wooden sign with Olympic rings as a background and the generated 3D model

Implementation

5.4.1 Performance of the iSketche system

The complete iSketche system was hosted on a pc with Windows XP as the operating system and IIS 5.0 as the hosting server. The pc had an Intel Core 2 Duo processor with 2.4 GHz speed, 3GB of RAM and 256MB of video memory.

The iSketche application was accessed from an external computer within the local network area of the hosting pc. The total time taken to generate the first Olympic sign in Solidworks 2006 and returning the final snapshot to the user took about 5-6 sec. The time can increase by 2 or more seconds depending upon the internet connection speed between the external computer and the hosting pc if not on the same local network. The most time consuming parts of the design generation was concluded to be the generation of 3D model inside SolidWorks. This was due to the fact that information being transferred between the client and the server was only an image of size ~69kb.

Looking at the results, it can be said that the total time of 6 sec taken for generation of the 3D model and presenting the user with a snapshot image is acceptable. By changing the configuration of the pc to have a better video card and increased RAM, a better result can be achieved and the total time can be significantly reduced.

In case of concurrent users designing using the iSketche application, the IIS server allows for ten users to be connected to the system at a time. To allow the access of SolidWorks to more than one user, iSketche queues the design process and the models are generated for the each use on a first-come-first-serve basis. To differentiate between the users, ASP Session ID is used by the iSketche application. This way in concurrent sessions, the ASP manager can identify each user and send the snapshot images to their respective users.

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

Computer aided design and manufacturing play an important part in developing a product during the product life cycle. Automation of this design and manufacturing processes can bring significant savings in the final cost of the product and also save time in the overall development of the product. However, a valid and simple solution to automate the design and manufacturing process was never given importance and there has not been any research on this topic. To make CAD and CAM affordable to small industries and automate the development process, it is necessary to offer solutions that bring significant savings in time and money to small and custom industries. A web-based design system was demonstrated with implementation to a custom wooden sign application. A user-defined parametric feature application was also introduced and its feasibility was discussed.

Through the iSketche application, it was shown that the user-defined feature system has an advantage over the WatCAD system that offered only pre-defined features to the client. The simplicity involved in designing using a sketchpad and the communication response between the client and the server scripts helped in bringing the online design application to small and custom industries without any significant investments in special CAD programs. It was shown

Conclusions and Recommendations

that the concept of user-defined features can be applied not only to design flat wooden signs but also to other applications that involve designing of custom products over the web.

In defining the user-defined features concept, the idea of developing different kinds of simple user-defined features and the framework to implement them turned out to be a success. The identification of basic elements and implementing them using a sketchpad were simple enough to allow any non-CAD designer to use the online system and generate 3D models instantly. The combination of both the sketchpad on the client side and using AJAX to transfer bits of information between the client and the server proved to be a great concept to provide an interactive application to the user.

The choice of SVG as the design medium inside the iSketche application turned out to be invaluable as it offered the parametric based design functionality that the author was looking for. The simplicity involved in programming it and implementing it using the JavaScript functions made it a valuable solution to offer user-defined features in a much more interactive way than the WatCAD application.

In conclusion, the online web based CAD system and the concept of user-defined parametric features save significant amount of time and cost in designing a product over the web. These methods can be used as a method to solving product design issues that arise in custom product industries and can be easily implemented along with any new concepts that may come out later. It was shown that by using built in APIs inside a solid modeler, the automation of design process can be accomplished. The API offers a unique way to generate and modify 3D models on the go with least user involvement during the whole design process. By using the functionality of the API it was possible to design a web based design system that offered the same functionality as a solid modeler. To properly make use of the API

Conclusions and Recommendations

functionality and offer a simple online design tool to non CAD experts, the user-defined features turned out to be a valid solution.

6.2 Recommendations

6.2.1 Development of sketchpad

The sketchpad used in the iSketche application had minimal tools offered to the user and was sometimes non-responsive due to the large number of functions inside the client side. The fact that the user browser may not be able to handle large client scripts made the application crash at times. Debugging of the application was also slower due to the amount of interlinked functions written inside the client scripts. To overcome this issue, fewer functions can be offered to the client to make debugging easy and also the functions can be written in external files that are only downloaded to the user browser when required.

The complete functionality of the sketchpad was managed by JavaScript, which is visible to the user and can be copied or distributed without any knowledge of the author. This can cause an issue as it could expose any intellectual property that was implemented inside the JavaScript functions. A JavaScript obfuscator or adding a license to the beginning of the JavaScript functions can minimize any illegal copying of the code inside the client scripts.

The design tools within the sketchpad itself were basic and offered little functionality to edit or delete any elements created inside the drawing area of the iSketche application. This was due to the amount of functions that needed to be written from scratch to offer any functionality to the iSketche application. By choosing a better way of offering such tools to the user should be undertaken if the functionality of the application needs to be improved.

Conclusions and Recommendations

It is also recommended that another way of offering the sketchpad application be researched. The author came up with the choice of designing using the Microsoft Silverlight technology that offered similar functionality as SVG. The advantage is that the application designed in Silverlight can be compiled on the server itself thus avoiding any exposure to intellectual property to the client and at the same time offer much more functionality to the user.

6.2.2 Types of user-defined features

The type III user defined feature was never completely implemented into the iSketche system. The functionality of type III user defined feature was discussed earlier and if implemented, the tool can offer great possibilities in creating designs. Also, more functionality could be added to make it a valuable tool.

The user-defined features discussed were simple and can be used without any issues for applications such as the flat wooden sign design. But when it comes to other applications such as table legs, other types of user-defined features need to be designed as well to make the application useful.

6.2.3 3D viewing

The purpose of the web-based CAD system is to create a 3D model over the web but in the iSketche application the user does not have the option of viewing the model in actual 3D inside the web browser. Although the functionality is an added feature for flat wooden signs, other applications might require 3D viewing as part of the system such as in the case of table

Conclusions and Recommendations

legs where only one view of the table leg is not enough to give the complete picture of how the table leg might look after manufacturing.

By offering a built-in 3D viewer inside the browser, the user can get a better picture of the actual model generated by the solid modeler. To accomplish this task, many different options are available. One of them is the STL viewer using Java applets. This method uses OpenGL [31] technique to render the STL file to a 3D model so giving the ability to view the actual 3D model. Another option is to generate series of images that are basically snapshots of the 3D model and combine them using a script to simulate rotation of a 3D model.

Bibliography

1. Yuh-Min Chen, Ching-Ling Wei. "Computer-aided feature-based design for net shape manufacturing". *Computer Integrated Manufacturing Systems*, Vol. 10, n. 2, pp 147-164, 1997
2. Javier Monedero. "Parametric design. A review and some experiences". <http://info.tuwien.ac.at/ecaade/proc/moneder/moneder.htm>. accessed July 27th, 2009
3. Martti Mäntylä. "An introduction to solid modeling". *Computer science press*, 1988
4. V.C. Lin, David C. Gossard, Robert A. Light. "Variational Geometry in computer-aided design". *Computer Graphics*, Vol. 15, n. 3, pg-171-177, August 1981
5. Robert A. Light, David Gossard. "Modification of geometric models through variational geometry". *Computer Aided Design*, Vol 14, n 4, July 1982
6. Fredy Jara. "About Solid Modeling by using feature-based models". <http://www.cs.uml.edu/~fjara/oldcourses/review3.html>. accessed August 4th, 2009
7. Gerardo Salas Bolanõs, Stephen Mann, Sanjeev Bedi. "Targeted WebCAD". *Computer Aided Design and Applications*, Vol 6, No. 5, pp 639-644, 2009
8. Jae H. Kim, Fu-Chung Wang, Carlo H. Séquin, Paul K. Wright. "Design for Machining over the internet". *ASME*, 1999
9. Sung H. Ahn, V. Sundararajan, Charles Smith, Balaji Kannan, Roshan D'Souza, Ganping Sun, Ashish Mahole, Paul K. Wright, JaeHo Kim, Sara McMains, Jordan Smith, Carlo H. Séquin. "CyberCut: An Internet based CAD-CAM system". *ASME*, Vol 1, pg 52-59, 2001
10. Aristides A. G. Requicha. "Representation for Rigid Solids: Theory, Methods, and Systems". *Computing surveys*, Vol 12, No. 4, pp 437-464, 1980

Bibliography

11. Delcam plc – ArtCAM. <http://www.artcam.com/>. accessed June 18th 2009
12. Nike – NikeID. <http://nikeid.nike.com/nikeid/index.jsp#home>. accessed August 24th 2009
13. Siemens AG – NX. http://www.plm.automation.siemens.com/en_us/products/nx/. accessed June 22nd 2009
14. Romulus. [http://en.wikipedia.org/wiki/Romulus_\(b-rep_solid_modeler\)](http://en.wikipedia.org/wiki/Romulus_(b-rep_solid_modeler)). accessed June 22nd 2009
15. Dassault Systèmes – SolidWorks Corporation, SolidWorks 2006. <http://www.solidworks.com/sw/products/mechanical-engineering-cad-software.htm>. accessed June 22nd 2009
16. Jianjiang Zeng, Wenliang Chen, Quiling Ding. “A Web-based CAD system”. *Journal of Materials Processing Technology*, Vol 139, pp 229-232, 2003
17. Microsoft Corporation, ASP.NET - .NET framework. <http://www.asp.net/>. accessed August 12th 2009
18. Donald E. LaCourse. “Handbook of Solid Modeling”. *McGraw-Hill Inc*, 1995
19. Misumi USA, Linear Guides – Medium load, regular clearance. <http://www.misumi-ec.com/usa/eusaen/ItemDetail/10300047720.html>. accessed August 12th 2009
20. Microsoft Corporation, ASP.NET AJAX – Asynchronous JavaScript and XML. <http://www.asp.net/ajax/>. accessed August 15th 2009
21. Sun Microsystems, Java. <http://www.java.com/en/>. accessed August 15th 2009
22. Adobe Corporation – Flash Player. <http://www.adobe.com/products/flashplayer/>. accessed August 16th 2009
23. World Wide Web Consortium – W3C, Scalable Vector Graphics v1.1 – SVG. <http://www.w3.org/Graphics/SVG/>. accessed August 17th 2009

Bibliography

24. W3C schools, JavaScript. <http://www.w3schools.com/JS/default.asp>. accessed August 18th 2009
25. W3C schools, XML – Extensible Mark-up Language. <http://www.w3schools.com/xml/default.asp>. accessed August 18th 2009
26. Adobe Corporation, Adobe SVG viewer 3.03. <http://www.adobe.com/svg/viewer/install/mainframed.html>. accessed August 19th 2009
27. Microsoft Corporation, Visual Basic Developer Center. <http://msdn.microsoft.com/en-us/vbasic/default.aspx>. accessed August 19th 2009
28. Microsoft Corporation, LINQ: .NET Language-Integrated Query. <http://msdn.microsoft.com/en-us/library/bb308959.aspx>. accessed August 19th 2009
29. SVG Document Structure, ‘defs’ element. <http://www.w3.org/TR/SVG/struct.html#DefsElement>. accessed August 20th 2009
30. Olympics – 2010 Vancouver Olympic games. <http://www.vancouver2010.com/>. accessed August 21st, 2009
31. Silicon Graphics International, OpenGL. <http://www.opengl.org/>. accessed August 21st 2009
32. Wikipedia, Computer Aided Design. http://en.wikipedia.org/wiki/Computer-aided_design. accessed June 21st, 2009