ELSEVIER

Computers in Industry 1634 (2002) 1–12

COMPUTERS IN INDUSTRY

www.elsevier.com/locate/compind

A framework of web-based conceptual design

S.F. Qin[a,*], R. Harrison[b], A.A. West[b], I.N. Jordanov[c], D.K. Wright[a]

[a]Department of Design, Brunel University, Runnymede Campus, Surrey TW20 0JZ, UK
[b]Department of Manufacturing Engineering, MSI Institute, Loughborough University, Loughborough LE11 3TU, UK
[c]Department of Computer and Information Sciences, De Montfort University, Milton Keynes MK7 6HP, UK

Received 4 June 2001; accepted 20 March 2002

Abstract

A web-based conceptual design prototype system is presented. The system consists of four parts which interpret on-line sketches as 2D and 3D geometry, extract 3D hierarchical configurations, allow editing of component behaviours, and produce VRML-based behavioural simulations for design verification and web-based application. In the first part, on-line freehand sketched input is interpreted as 2D and 3D geometry, which geometrically represents conceptual design. The system then infers 3D configuration by analysing 3D modelling history. The configuration is described by a parent–child hierarchical relationship and relative positions between two geometric components. The positioning information is computed with respect to the VRML97 specification. In order to verify the conceptual design of a product, the behaviours can be specified interactively on different components. Finally, the system creates VRML97 formatted files for behavioural simulation and collaborative design application over the Internet. The paper gives examples of web-based applications. This work forms a part of a research project into the design and establishing of modular machines for automation manufacture. A consortium of leading automotive companies is collaborating on the research project.

Keywords: Sketch; Conceptual design; Behavioural simulation; Web application

1. Introduction

Economic globalisation is creating competitive pressures on industry to minimise the time to bring products to market. Project timing through the whole production process: conceptual design, detailed design, analysis and test, installation, to maintenance must be compressed wherever possible. Today, information technologies and the web are challenging, and changing the way industry works. It is believed that web-based conceptual design techniques can be applied to improve efficiency by first building conceptual design models to represent products' geometry, structures, and behaviours, and then distributing the models over the web for remote evaluation and verification of the design correctness. The web is seen as the ideal method to achieve this, because a web-based system has a universal interface, uses open standards, and is globally supported [1,2].

Conceptual design is an early stage of the product development process having characteristics of fuzzy

* Corresponding author. Tel.: +44-1784-431341-244; fax: +44-1784-472879.
E-mail address: sheng.feng.qin@brunel.ac.uk (S.F. Qin).

0166-3615/02/$ – see front matter © 2002 Published by Elsevier Science B.V.
PII: S0166-3615(02)00117-3

problems, tolerating a high degree of uncertainty. During the conceptual stage of design, designers generate ideas, turn them into quick sketches with basically two-dimensional (2D) tools like pencil and paper, while at the same time these activities are guided by function design. Designers not only need to determine the physical structure of the design, but also need to verify design functions. Conventional CAD systems do not readily support this conceptual design process, since they usually require complete, concrete and precise definitions of the geometry, which are only available at the end of the design process. To provide computational support for computer aided conceptual design (CACD), studies [3–5] indicated that a CACD system must allow sketched input. On the other hand, during conceptual design, collaborating designers or partners (e.g. customer, manufactures), may work in different sites all over the world. To some extent, there is a lack of consistent visualising tools to view, share, and evaluate conceptual design models or results.

Our research investigates sketch and simulation based design tools to allow users to quickly model their design ideas and test their design by performing products' behavioural simulation on the Internet. A possible application scenario is shown in Fig. 1. Designers first sketch out their conceptual design and transform the design into a simulation model, then send or broadcast the animated simulation model of the conceptual design over the Internet to enable collaborative working with designers, manufactures, and potential customers who wish to evaluate and verify the initial design ideas. The designers can thus quickly get feedback from their partners. Simulating and testing various design ideas in a rough model at the early stages of design facilitates the integration of the geometric design with the product's behavioural description. Our research focuses on geometric modelling and simulation, rather than discrete event simulation [6]. Many geometric simulators [7] have been explicitly developed for simulation of robots, e.g. CimStation and RobCAD. The last can be used for professional robot simulation and production cell animation, but for a number of reasons the required functions of a conceptual simulation model cannot be created with these software systems [8]. For example, the processing of sketched input is generally unavailable. For the viewing of sketch-based modelling, some efforts [9–11] in recognising 3D objects from a set of sketched 2D input have been made. These efforts

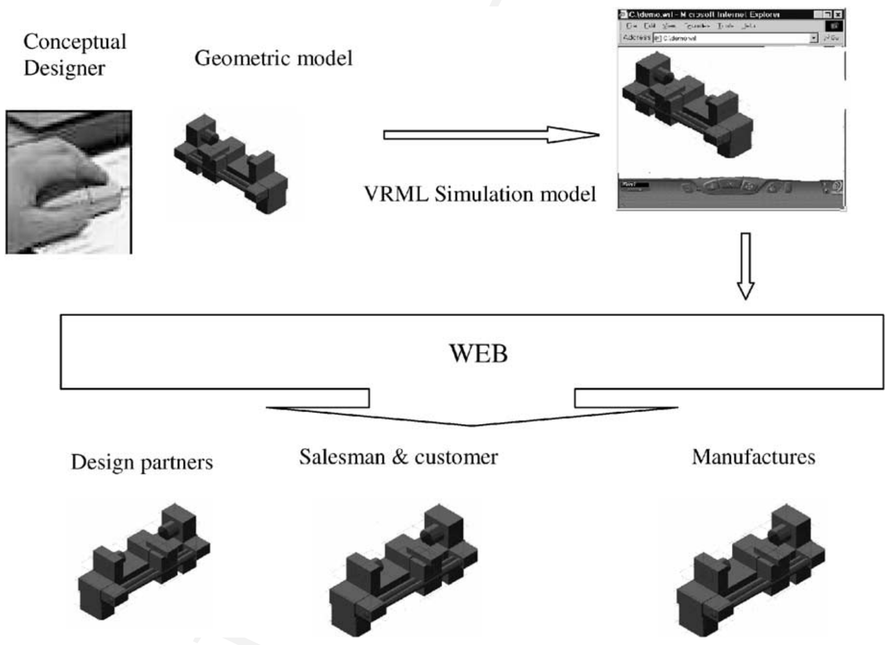


Fig. 1. A possible application scenario.

focused only on geometric descriptions in a global co-ordinate system. However, a simulation model should be described in a hierarchical way and be associated with behavioural descriptions embedded within the design. Our research integrates sketch-based 3D recognition techniques with simulation modelling techniques to support web-based conceptual design activities. In Section 2, our initial sketch-based modelling system is briefly described. The process of obtaining hierarchy information is presented in Section 3. In Section 4, our approach to specify behaviours is discussed. Finally, examples are given and conclusions made.

2. Sketch-based modelling

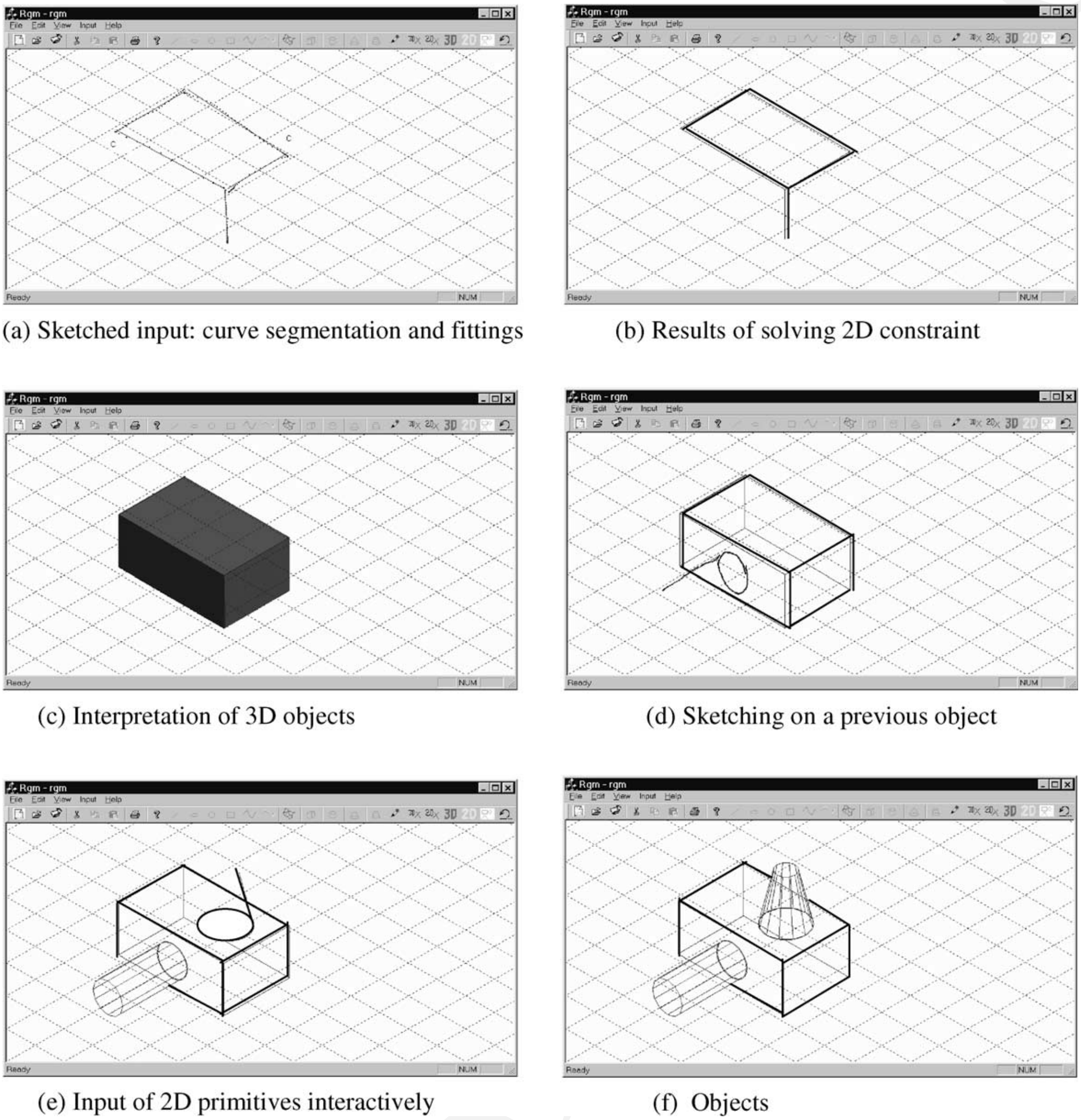
2.1. Initial sketch interpretation system

Our initial sketch interpretation system [12] has been developed in three phases: segmentation and curve fitting, constraint solver and 2D geometry, and 3D interpretation. Here, a brief introduction to the system is given to describe its functions and discuss our newly developed work. In the first phase, a conventional mouse is used as the input device. While sketching, the system gets a sequence of input data from mouse button presses, mouse motion and mouse button release events. This sequence of data represents a freehand curve that may consist of several sub-curves. The investigated segmentation approach accepts the input of on-line free-hand sketch, and segments it into meaningful parts, by using fuzzy knowledge in terms of sketching position, changes of drawing direction, drawing speed and acceleration. After segmentation, each sub-curve represents one 2D primitive. The sub-curve is then classified into one of the following 2D primitives: straight lines, circular arcs and elliptical arcs, or free-form curves, it is then fitted with corresponding parameters. As a result of the segmentation and curve fitting, a set of 2D primitives or free-form curves are obtained. These 2D entities are roughly placed at their proper positions and directions. In general, however, they are not connected together correctly to reflect users' intent. A geometric constraint inference engine and a constraint solver are utilised to capture the designers' intention, and to generate a possible solution. At the end of this phase, 2D entities have their correct positions and 2D constrained connections. In the last phase, rule-based feature interpretation and manipulation techniques are investigated. While drawing, the 2D geometry is accumulated until it can be interpreted as a 3D feature. The feature is then placed in a 3D space and a new feature can be built incrementally upon previous versions. Therefore, this 3D recognition process automatically assembles features in 3D space. Once a feature is created, a user can examine it in a wire-frame model or in a shaded solid model from different views. In addition to the sketched input, users can input 2D primitives interactively. This gives more freedom in inputting 2D information. The system currently supports only extruded objects. Fig. 2 shows the sketch-based modelling process. The background diagonal lines parallel to the axes of the isometric projection are auxiliary lines for assisting sketch.

In Fig. 2a, three strokes were drawn. Two of them contain two straight lines with corner points marked with letter "C". Another stroke is a vertical line. The system first found the corner points by segmentation processing, then sketches were classified as straight lines, and fitted with lines. These sketched lines initially were not connected properly and were not parallel to the axes of the isometric projection to reflect a user's intention. However, the system examined those sketched input to the extract 2D constraints: connection relations between those entities, and unitary relations such as vertical direction. Consequently, the system produced a 2D solution (geometry) for extracted constraints shown in Fig. 2b. The lines became parallel to the axes of the isometric projection with proper connections. This reflects the user's intention. Based upon the 2D geometry, the system interpreted the input as a 3D box feature illustrated in Fig. 2c. Afterwards, the user continued sketching a cylinder on the left face of the box (Fig. 2d). After receiving the cylinder, a truncated cylinder was entered by interactive input of an ellipse and a line on the top face of the box (Fig. 2e). The user can choose to input 2D entities by either sketched input or interactive input. As a result of the 3D interpretation, combined 3D objects were received (Fig. 2f).

2.2. Improved prototype system

In order to construct simulation models, hierarchy information and relative positioning information are



(a) Sketched input: curve segmentation and fittings

(b) Results of solving 2D constraint

(c) Interpretation of 3D objects

(d) Sketching on a previous object

(e) Input of 2D primitives interactively

(f) Objects

Fig. 2. Modelling process.

needed. We have improved our initial sketch interpretation system to support the simulation modelling processes. From the geometric modelling processes, the hierarchy information is extracted and is described as a tree structure shown in Fig. 3.

The root node in Fig. 3 is a null object. It just defines a global co-ordinate system, in which the positive *X*-direction points to the right, the positive *Y*-direction points up, and the *Z*-direction points out from the screen. It also provides three co-ordinate planes as reference planes. A parent-object is linked with its child-objects. This means that the parent-object is used as a reference to further build its child-objects. Thus, one parent-object can be classified as a child-object when referencing to its parent-object, and it can also be identified as a parent when looking at its children.

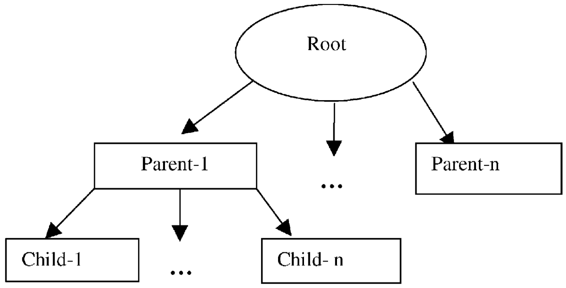


Fig. 3. Tree structure.

2.2.1. Hierarchy information

During sketching, after finding closed profiles, extrusion edges and directions, e.g. an ellipse and a line drawn from the ellipse in Fig. 2d, the system can recognise the sketched input as an extrusion feature. Then it has to find a reference plane from previous 3D objects in order to obtain information about features' sizes and their positions (transformation information). If the reference plane exists, the 3D transformation information can be extracted. The referenced 3D object will become a parent-object, and the new object (feature) will become a child-object linked to the parent. If the reference plane comes from one of the three global co-ordinate planes, the new object will be linked to the root. Brother or sister relationships can be formed when two or more objects come from the same parent.

To determine a reference plane, the system first computes the centroid of a closed profile. If the inferring feature is a cylindrical object, its centroid is the centre of the ellipse (closed profile). If the feature is a non-cylindrical object, the centroid can be received by

$$x_{cd} = \frac{\sum_{i=1}^{n} x_i}{n, y_{cd}} = \frac{\sum_{i=1}^{n} y_i}{n}$$

where n is the number of elements involved in the closed profile, x_i and y_i the pair of co-ordinates of the end points for each element.

After obtaining the centroid position, the system continues to conduct a containment test [13] between the centroid point and the closed profile (a polygon or ellipse), which is a projection of a face of a previous 3D object. If the centroid is within two or more closed profiles (or projection areas of faces), the system will further determine which face is a reference plane by finding an minimum angle between the extrusion direction vector and projected normal vectors of candidate faces. For example in Fig. 2d, the centre of the ellipse is within the projection areas of the left face and the bottom face of the box object. The extrusion direction is parallel to the normal vector of the left face. It is obvious that the angle between the extrusion direction vector and the projected vector of the normal of the bottom face is bigger than the angle between the extrusion direction vector and the projected vector of the normal of the left face. Thus, the left face of the box object is determined as a reference plane. Consequently, the box object becomes a parent-component of the new component (cylinder). In turn, the cylinder is a child of the box object.

2.2.2. Relative positioning

Each object is described in three co-ordinate systems in terms of the object (or primitive), relative and global co-ordinate systems. The object co-ordinate system is related to a graphics rendering program, e.g. OpenGL and VRML97 [14]. In order to easily transfer models into VRML97 format for the web-based application, the object co-ordinate system is consistent with shape and geometry definition in VRML97. For example, a cylinder can be defined in its object co-ordinate system by a radius and a height as shown in Fig. 4. In the relative co-ordinate system, an object coupled with its object coordinate system is defined in its parent's object co-ordinate system. The definition includes transformation of a child's co-ordinate system to its parent's co-ordi-

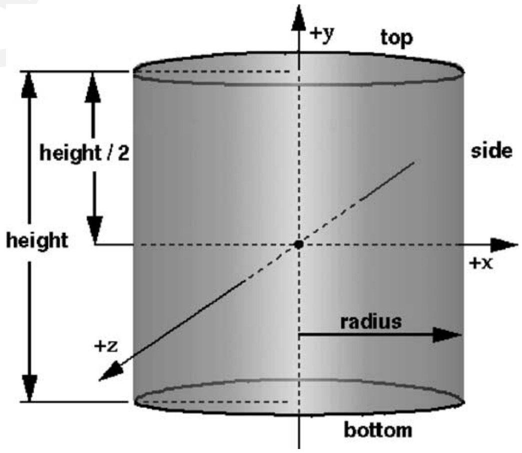


Fig. 4. An object co-ordinate system.

nate system and geometric descriptions in its own object co-ordinate system. In order to display objects and produce projection of faces of objects, descriptions of objects are finally transformed to the global co-ordinate system.

Each object is internally represented by an object-oriented class, which encapsulates modelling data: dimensional and positional parameters, derived data from the model such as B-rep (boundary representation) information about faces, edges, and vertices, and its member functions (methods) for building the model, producing B-rep information, accessing the data and so on. Each object model is an instance of its corresponding class. This representation can take full advantages of the features and properties of object-oriented design, e.g. data encapsulation and code reuse through the inheritance mechanism. Taking a box part as an example, its corresponding class can be defined as follows:

```
class Box::Object
{
protected:
double length, width, height; //dimensional parameters
double relative_tx, relative_ty, relative_tz; //relative translation parameters
double relative_ax, relative_ay, relative_az; //relative rotation axis
double relative_angle; //rotation angle about the relative rotation axis
double globle_x, globle_y, globle_z; //globe translation parameters
...
public:
Box(Object referentObject, double Length, double Width, double Height);
void Draw2D();
void Draw_frame3D();
void DrawShade3D();
void generating_faces();
void get_data_of_faces();
...
};
```

This class named *Box* is derived from an existing public class named *Object*. In the data field, we declare modelling data and derived data as protected type. The construction function takes a reference object and dimensions of the box as input and generates relative positions to its parent (the reference object) and globe positioning information.

To compute relative positions of a child-object to its parent-object, the object co-ordinate system $O_pX_pY_pZ_p$ of the parent is assumed as in Fig. 5. The object co-ordinate system $O_cX_cY_cZ_c$ of the child is transferred to a new position from its initial position that is coincident with $O_pX_pY_pZ_p$. The transformation processes can be identified as a rotation about an axis vector to make the Y_c consistent with (pointing to) the normal direction of the reference plane, and a translation to let the origin O_c has a distance of d from the reference

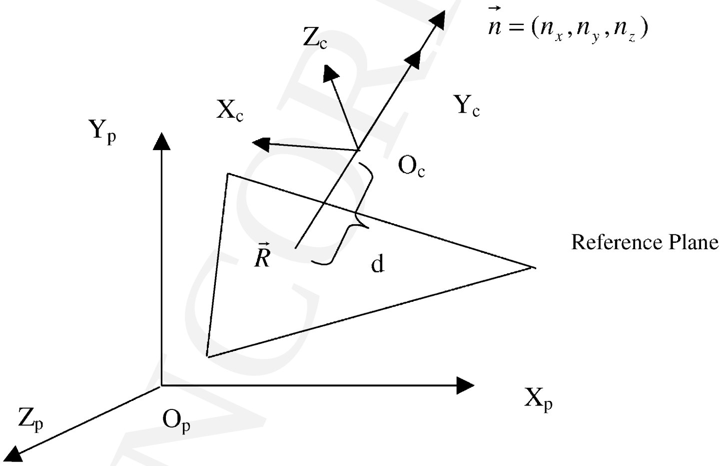


Fig. 5. Relative position.

plane to enable a right size of the child-object. Let $\vec{R}$ be a vector from the origin O_p pointing to the intersection point of the axis Y_c and the reference plane. The relative positions are computed in terms of a rotation axis vector, a rotation angle and a translation.

(1) *Computing the rotation axis vector*: The rotation axis vector can be represented as

$$\vec{A} = \vec{y}\vec{n}$$

where $\vec{y}$ is a unit vector along Y-axis, $\vec{n}$ is a unit vector of the normal of the reference plane. In case of $\vec{y}$ parallel to $\vec{n}$, $\vec{A}$ is assigned as a unit vector along Z-axis.

(2) *Computing the rotation angle*: The rotation angle can be computed by

$$\theta = \arcsin(||\vec{A}||)$$

When $\vec{y}$ is equal to $\vec{n}$, θ is assigned a value of 0; while $\vec{y}$ is opposite to $\vec{n}$, θ is assigned a value of π.

(3) *Obtaining the translation*: The translation vector $\vec{T}$ can be given by

$$\vec{T} = \vec{R} + \mathrm{d}\vec{n}$$

In order to obtain global positions of a child-object, the system will accumulate transformations from the root to the child.

3. Behavioural description

From the sketch recognition, a hierarchical structure of the design is received. Design structures can be classified as static and dynamic structures. In a static structure, design components, their attributes, and their relationships are fixed, and there is no active component or process in the structure. They are assumed not to change their structures with time, e.g. civil engineering design. Whilst in a dynamic structure, design components, their attributes and their relationship to one another can be changed with time by external effects or driving events. For instance, when a car is started, its engine will run. These driving events (input to design) and their corresponding structural changes (output of the events) can be defined as design components' behaviours in relation with function design of a product. While designers sketch out their design structures (geometry definition), the functional relationships are being considered. After structural design, the designers can verify functional design by specifying the behaviours of design components and simulating them later. The behavioural simulation is commonly used for functional design verification [15]. The simulated and desired behaviours are compared in order to determine to the degree of functionality achieved.

In our system, the designer can specify behaviours to a selected object. The designer first selects a geometric object representing a design component, and then inputs the behaviour in a dialogue window shown in Fig. 6. Behaviour can be triggered by a driving input (an event). In order to produce a driving event at real-time simulation, a touch sensor is attached to the selected geometric object. In a simulation environment, if users simply click the geometric object, the touch sensor will be activated to drive the corresponding behaviours. After receiving a driving event, the design component will continuously change its initial state to a set of different states. In the input window, designers specify the name of the behaviour, and input time intervals corresponding to serial states. If the states are related to the position changes of the design component, the designers can continue to enter key positions in relation to state changes. If there is no position change of the design component during the state transitions, the designers can specify different colours of the geometric object to represent different states.

Our approach is limited as the behaviours must be known or specified by the designers. Design verification is achieved by ensuring that values of design variables meet the functional requirement.

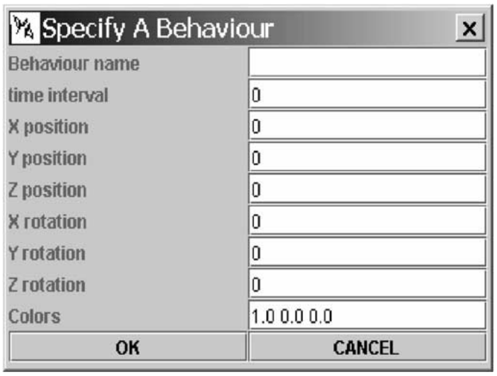


Fig. 6. Input window for behavioural descriptions.

4. Virtual reality mark-up language (VRML)-based simulation

A composite geometric and behavioural model is constructed in our sketch-based modelling system. This model enables designers, during a conceptual design stage, to effectively communicate their intent by simulating and verifying dynamic behaviours. In order to effectively and easily conduct the simulation, we output the model data into VRML formatted files. VRML has an open structure and is easy to access over the Internet. The VRML files can be visualised graphically and animated interactively using a web browser with a VRML plug-in, e.g. CosmoPlayer. This simulation model can be shared with different partners (co-designers, manufacturers, customers, etc.). The simulation may be visualised and controlled remotely over the Internet. Designers can use the models to simulate the products' performance, to determine part clearance, interference and collision detection, and thus improve their designs. Moreover, utilising VRML, designers can potentially further develop the simulation models into multimedia-based product presentations for a advertising purposes by integrating additional multimedia data.

It is easy to transfer the VRML-based model into commercial CAD packages as an initial input for detailed design. This allows design ideas to be consistently transferred from the conceptual stage to the detailed design stage.

5. Examples and discussion

We have, as an example, used our prototype system to conceptually model a milling machine. Firstly, a conceptual model of the milling machine was built on sketched input. Its shaded model is given in Fig. 7.

After obtaining the conceptual geometric model, the system extracted the hierarchical information for the design. For example, the vertical carriage is linked to its parent component (the vertical base pillar). It has a child (the table moving along X-axis) and a grandchild (the workpiece holder moving back and forth). The vertical carriage can move up and down. These provide motion in three directions. Based on the geometric model and hierarchical structure, we interactively selected components to specify their behaviours. For example, we included a touch sensor to the workpiece (a cylinder on the workpiece holder), specified a movement of the vertical carriage from current position to 80 cm within a time interval of 10 s, specified the table's motion of moving 10 cm backwards and defined the workpiece holder's motion of moving 40 cm to the right. We also changed the colours of the components for appearance modification. The geometric model and the behaviour definitions formed a simulation model of the drilling machine.

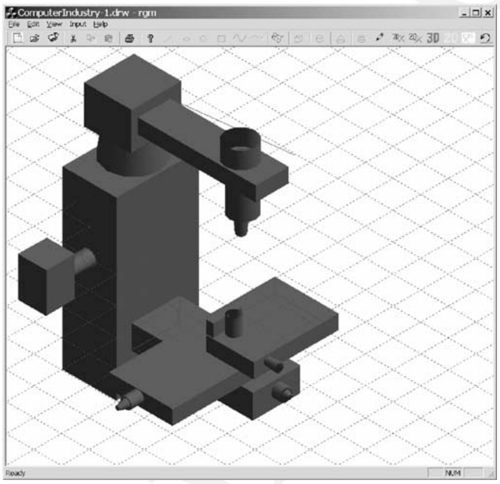


Fig. 7. A shaded model of the machine.

Finally, we outputted the simulation model into a VRML97 formatted file and loaded this file in an Internet browser. Fig. 8 shows the initial state of the drilling machine. When the touch sensor was activated, the defined behaviours were performed; the final state was shown in Figs. 9 and 10. If the file is linked to a web server, the simulation can be executed remotely over the Internet for design verifications.

After receiving conceptual design models in VRML formatted files, we may use web technologies for supporting the active feedback from the users (clients) of the system and their collaborative work. The web computing architecture [18,19] of the system could be a three-tier client/server architecture: presentation tier, application tier and share data tier as shown below.

Shared data might be stored in a database server in a collection of VRML files or a corresponding relational database. A separate application server runs the collaborative design application logic (Java-application),

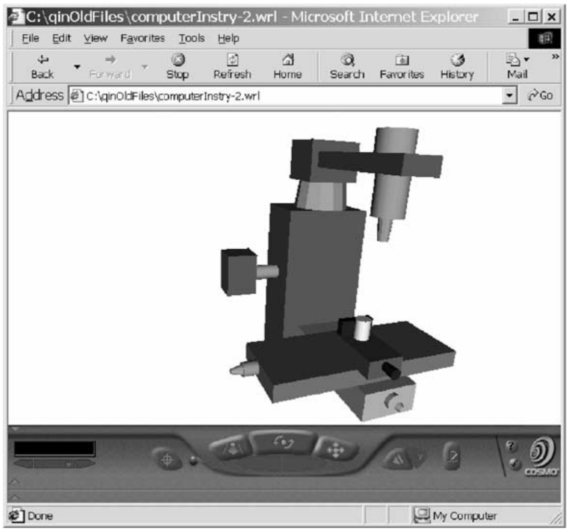


Fig. 8. Initial state of the machine.

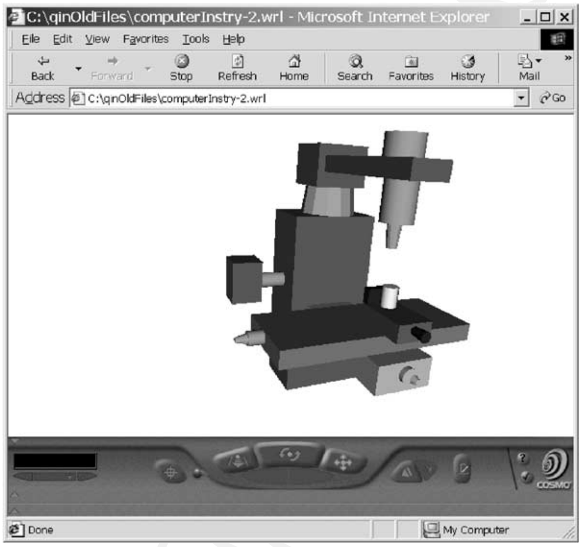


Fig. 9. Final state of the machine.

which mediates between shared data and presentation (web servers and web browses), and manipulates the shared data. It takes inputs and requests through Java remote method invocation (RMI) and common gateway interface (CGI) or the extendible mark-up language (XML) mechanisms from the presentation (HTML or XML), decides what needs to be done, decides what shared data should be accessed or must be updated, manipulates that data appropriately, e.g. creating a new design version, and responds to the presentation. The shared data answers queries from the application logic through JDBC or structured query language (SQL) mechanisms, and the application logic determines what data is stored and what queries are needed. In the presentation tier, web servers interact with web browsers supporting the

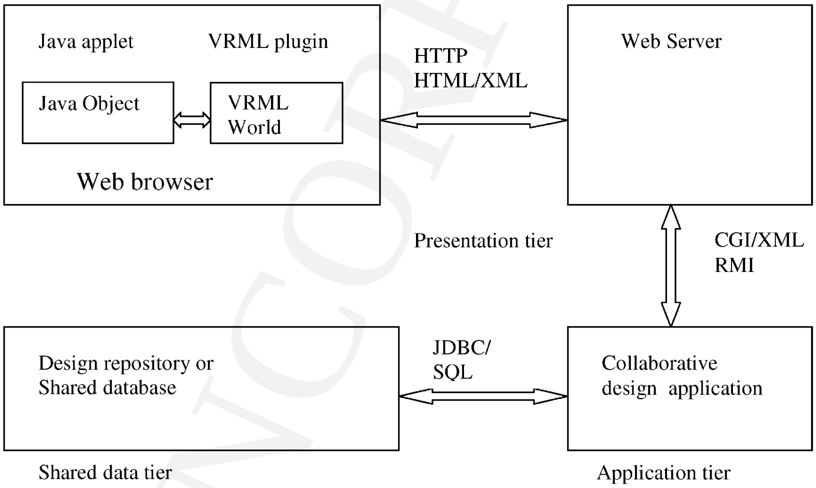


Fig. 10. The web computing architecture of the system.

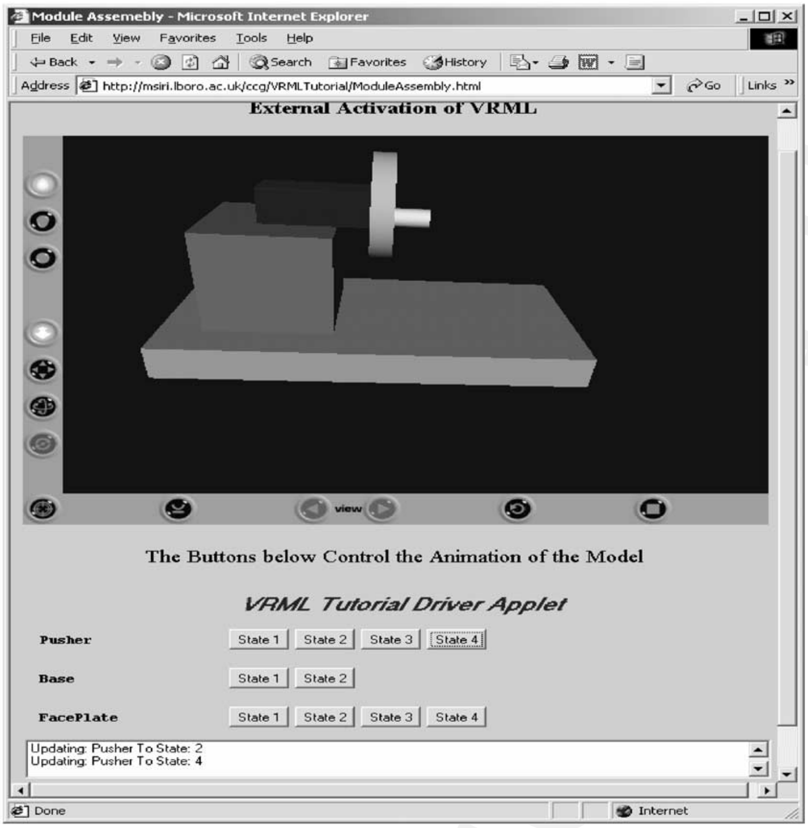


Fig. 11. External activation of a VRML model.

users. The users are able to navigate through the VRML worlds by using a VRML-browser. The external authoring interface (EAI) makes it possible to control the VRML worlds dynamically via the Java applets or Javascripts. For example, users can interactively verify or evaluate a design model by interacting with a Java applet that activates (http://msiri.lboro.ac.uk/ccg/vrmltutorial/moduleassembly.html) the VRML model shown in Fig. 11. Users can click on different state labels defined in the Java applet to activate design simulations. In a similar way, users might dynamically modify the design by changing design attributes, e.g. parameters of a cylinder, update their design to collaborative participants' browsers (depending on authorised rights) or request to store their design as new versions in the database or send their evaluation feedback by e-mail.

6. Conclusion

From the examples, some features of the prototype system can be identified as follows:

(1) Using an on-line sketch, users can rapidly and easily create, and edit a 3D design geometric model for any purposes.
(2) With behavioural definition attached to the geometric model, designers can explore their ideas not only in a static model form, but also in a dynamic simulation form. This will provide an effective evaluation mechanism for verifying their conceptual designs. Instead of working with confusing paper drawings, designers can have a real-time shaded and animated view of their design models, without the need for

expensive CAD hardware or software, and without extensive training.

(3) This tool lets the designers publish their designs on the Internet. Designers can share models and data with their partners involved in the product development process without the need for of expensive workstations and CAD software. This VRML-based simulation is more accessible to non-expert users. Non-CAD users such as customers, suppliers, and managers may evaluate the design and quickly give feedback. This communication mechanism may compress the timing from the conceptual design to manufacturing and marketing, and support distributed engineering of manufacturing machines [20].

(4) With this tool, the designers could quickly transfer their conceptual design ideas bounded with approved modelling data into commercial CAD packages to rapidly realise detailed design and manufacturing processes. Comparing with the current design process, time is saved by directly importing VRML-based models into CAD packages.

In summary, the authors believe that this tool has the potential to save time and money by: (i) rapidly developing a product model; (ii) improving understanding design ideas for all parties involved; (iii) facilitating communication so less time is spent in face to face meetings; (iv) reducing the need to invest more CAD workstations and software; (v) using simulation to reduce the number of costly physical prototypes.

The next stage of this work will include an evaluation of the tool in design applications at our collaborating manufacturing companies.

Uncited references

[16,17].

Acknowledgements

The support of the EPSRC (GR/M83070 and GR/M53042), the Ford Motor Company Limited, Cross-Hueller Limited, Johann A. Krause GmbH and the other collaborating companies in carrying out this research is gratefully acknowledged.

References

[1] M. Rezayat, The enterprise-web portal for life-cycle support, Computer Aided-Design 32 (2000) 85–96.

[2] M. Bender, R. Klein, A. Disch, A. Ebert, A functional framework for web-based information visualisation systems, IEEE Transactions on Visualisation and Computer Graphics 6 (1) (2000) 8–23.

[3] D.L. Jenkins, R.R. Martin, Applying constraints to enforce users' intentions in free-hand 2-D sketches, Intelligent Systems Engineering 1 (1) (1992) 31–49.

[4] C.G.C. Van Dijk, New insights in computer-aided conceptual design, International Journal of Design Studies 16 (1) (1995) 62–80.

[5] T. Hwang, D. UIlman, Recognise features from freehand sketches, ASME Computers in Engineering 1 (1994) 67–78.

[6] C.D. Pegden, R.E. Shannon, R.P. Sadowski, Introduction to Simulation Using SIMAN, Second ed., McGraw-Hill, New York, 1995.

[7] P. Kilingstam, P. Gullander, Overview of simulation tools for computer-aided production engineering, Computers in Industry 38 (1999) 173–186.

[8] M. Weyrich, P. Drews, An interactive environment for virtual manufacturing: the virtual workbench, Computers in Industry 38 (1999) 5–15.

[9] P. Chen, S. Xie, Freehand drawing system using a fuzzy logic concept, Computer-Aided Design 28 (2) (1996) 77–89.

[10] L. Eggli, C.Y. Hsu, B.D. Bruderlin, G. Elber, Inferring 3D models from freehand sketches and constraints, Computer-Aided Design 29 (2) (1997) 101–112.

[11] R.C. Zeleznik, SKETCH: an interface for sketching 3D scenes, in: Proceedings of the ACM SIGGRAPH, 1996, pp. 163–170.

[12] S.F. Qin, D.K. Wright, I.N. Jordanov, From on-line sketch to 2D and 3D geometry: a system based on fuzzy knowledge, Computer-Aided Design 32 (14) (2000) 851–866.

[13] I. Zeid, CAD/CAM Theory and Practice, McGraw-Hill, New York, 1991.

[14] J. Hartman, J. Wernecke, The VRML 2.0 Handbook: Building Moving Worlds on the Web, Addison-Wesley, New York, 1996.

[15] Y.M. Deng, G.A. Britton, S.B. Tor, Constraint-based functional design verification for conceptual design, Computer-Aided Design 32 (14) (2000) 889–899.

[16] Y. Umeda, M. Ishii, M. Yoshioka, Y. Shimomura, T. Tomiyama, Supporting conceptual design based on the function-behaviour-state modeller, Journal of Artificial Intelligence for Engineering, Design, Analysis and Manufacturing (AI-EDAM) 10 (1996) 275–288.

[17] J. Dorsey, L. McMillan, Computer graphics and architecture: state of the art and outlook for the future, ACM SIGGRAPH Computer Graphics 32 (1) (1998) 45–48.

[18] D.G. Messerschmitt, Networked Applications: A Guide to the New Computing Infrastructure, Morgan Kaufmann, Los Altos, CA, USA, 1999.

[19] A. Eliëns, Principles of Object-Oriented Software Development, Second ed., Addison-Wesley, UK, 2000.

[20] R. Harrison, A.A. West, R.H. Weston, R.P. Monfared, Distributed engineering of manufacturing machines, in: Proceedings of the Institution of Mechanical Engineers, Journal of Engineering Manufacture B 217–231 (2001).

S.F. Qin, a lecturer, the Department of Design, Brunel University. Research interests: sketch based Computer Aided Conceptual Design, web-based application, and simulation modelling.

R. Harrison, a senior lecture at MSI in the Loughborough University. Research interests: systems modelling, integration and control by creating globally distributed virtual environments to underpin the life cycle engineering of component-based machines and process control systems.

A.A. West, a lecturer, at MSI in the Loughborough University. Research interests: the lifecycle engineering of intelligent, distributed, component-based manufacturing control and monitoring systems.

I.N. Jordanov, a senior lecturer in the Department of Computer and Information Sciences, the De Montfort University. Research interests: neural network training (local minima problem), stochastic methods for global optimisation, object-oriented programming and applications.

D.K. Wright, a reader in the Department of Design, the Brunel University. Research interests: interaction between people and products, CAD tools in conceptual design, biomechanical modelling for product design, virtual prototypes and rapid prototyping techniques.