Journal of Intelligent Manufacturing (2000) **11,** 515–534

Object-oriented cam design through the internet

JONATHAN LARSON and HARRY H. CHENG

Integration Engineering Laboratory, Department of Mechanical and Aeronautical Engineering, University of California, One Shields Avenue, Davis, CA 95616
E-mail: hhcheng@ucdavis.edu

We have developed a Web-based interactive cam design package under the programming paradigm of the C^H language environment. This package was initially developed as a teaching and learning tool for educational use in an undergraduate Computer-Aided Mechanism Design course. Because the system is Web-based and implemented through a client/server model with the user interface through the Web browser, it is easy to use and maintain. The system can also be used to solve practical engineering cam design problems with flat or roller followers and with translating or oscillating motion types. The system can be used to generate the cam profile, transmission angle, position, velocity, and acceleration of the follower. Once a cam/follower system is designed, animation of the cam/follower system can be performed. At the end of the design, the CNC code for manufacturing the designed cam can also be produced through our Web-based cam design system. The package consists of a number of modules including various Web pages, common gateway interface (CGI) programs, a C^H program called `cam.ch`, and the CCam C^H class which performs the necessary computation for cam design. Two different versions of the cam design package have been designed and implemented. One runs the cam design program on the client machine as a C^H applet, and the other runs the cam design program on the Web server through CGI. In this paper, details of design and implementation of Web-based cam design package will be described. Two application examples with different motion types for the follower will be used to illustrate features of the applet-based and CGI-based implementation schemes. The ideas of the Web-based software design presented in this paper can be applied to other application areas.

Keywords: Cam, web-based design, CGI, applet, Ch

1. Introduction

Cam is one of the most commonly used mechanisms in automation and assembly systems. Because of the wide application and the underlying principles behind its design, cam design can provide students with valuable experience in the design process, and is part of the undergraduate mechanical design curriculum in most engineering schools. For engineering practice, designers may use commercial software packages such as I-DEAS (Structural Dynamics Research Corporation, 1996) and Working Model (Knowledge Revolution, 1989) for design and analysis of cams. The application of these software packages in cam design requires significant expertise and time. For educational purpose, some cam design packages for student learning have been developed (Erdman and Sandor, 1997; Norton, 1992). These packages typically can display cam profile and follower motion characteristics such as position, velocity, and acceleration. Wang (1997) developed software utilities based on Working Model for animation of cam and follower systems. Aziz (1996) also developed a Window-based cam design package with animation and generation of CNC code for manufacturing of the designed cam. However, all these software packages and utilities are not suitable for Web-based cam design and animation in a network computing environment. In addition, they are not suitable for collaborative design and distance learning.

Under the paradigm of network computing, users are not required to install all the software in local machines. Information can be easily exchanged and software can be downloaded as it is needed. The World Wide Web (WWW) takes network computing to a new level by providing a friendly and convenient user interface. Because of the ubiquitous nature of The World Wide Web, it is ideally suited for use in distance learning. Students can take advantage of remotely located design tools to complete projects, or experiment and learn on their own. We have developed a Web-based interactive cam design package that runs under the C^H language environment. The C^H language environment is a superset of C, with extensions for mechanism design and network computing (Cheng, 1993a, 1993b, 1997). This cam design software package was designed specifically for use in a network computing environment. In the current implementation, it can be used to design cams with either translating or oscillating, flat-faced or roller followers and different motion characteristics such as harmonic and cycloidal motion types. Based on the principle of modular software design in the client/server model, this package uses a number of small programs and utilities, including dynamically created Common Gateway Interface (CGI) programs (Cheng, 1996; Felton, 1997) and C^H applets, to perform the necessary computations for cam design. The user can access our software through a Web browser, therefore, the system is easy to use. After the user types in cam design parameters and CNC manufacturing parameters through a Web browser, the system can produce cam profile and follower motion characteristics such as position, velocity, acceleration, and transmission angle. Animation of the designed cam and follower system can also be performed through the Web browser. Even the CNC code for manufacturing the designed cam can be generated automatically. In this paper, details of design, implementation, and application of our Web-based interactive cam design package will be described.

2. User interface for design of cams with translating or oscillating followers

The user interface for the cam design package is implemented as a set of Web pages, some of which are created dynamically in response to user input. The Web provides an interface that is user friendly and easy to use, hiding the details of the data generation for cam design. Users can design cam profiles for either translating or oscillating, flat-face or roller followers with either harmonic or cycloidal motion characteristics. In this section, the Web interface and user-selectable parameters for cam design are described.

2.1. *Follower type selection*

As shown in Fig. 1, the first of the Web pages allows the user to select the cam follower to be either translating or oscillating. The two choices are presented in graphical form. The followers are selected by clicking on the respective pictures. Next, as illustrated in Fig. 2, the follower type is selected more specifically to be either a flat-faced or a roller follower and the number of cam sections (a change in follower position or a dwell) is selected. These data are submitted to a CGI program that creates the next Web page dynamically for entry of the cam parameters. Alternately, users may select the sample cam design with a flat-faced follower in which all of the cam parameters are pre-selected.

2.2. *Cam design and CNC manufacturing parameters*

Figures 3 and 4 show the subsequent page where the user can input the cam parameters, some of which are dependent on the follower type, and the CNC cutter parameters. Note that Figs. 3–5 are the same Web page. All cam parameters must be set for the program to work properly. However, if the user does not want the CNC code for manufacturing the designed cam as an output, no modifications to the CNC parameters are needed, as they do not affect the calculations of the cam profile.

The user-selectable parameters used in the cam design program are described in the sections below (Erdman and Sandor, 1991; Mabie and Ocvirk, 1975; Martin, 1982). Figures 6–9 illustrate the basic parameters for the four supported follower cases.

2.2.1. *Parameters common to both translating and oscillating follower types*

There are a few parameters that are common with both follower types. The first of these is the base radius, measured in inches. It is the initial radius of the cam.

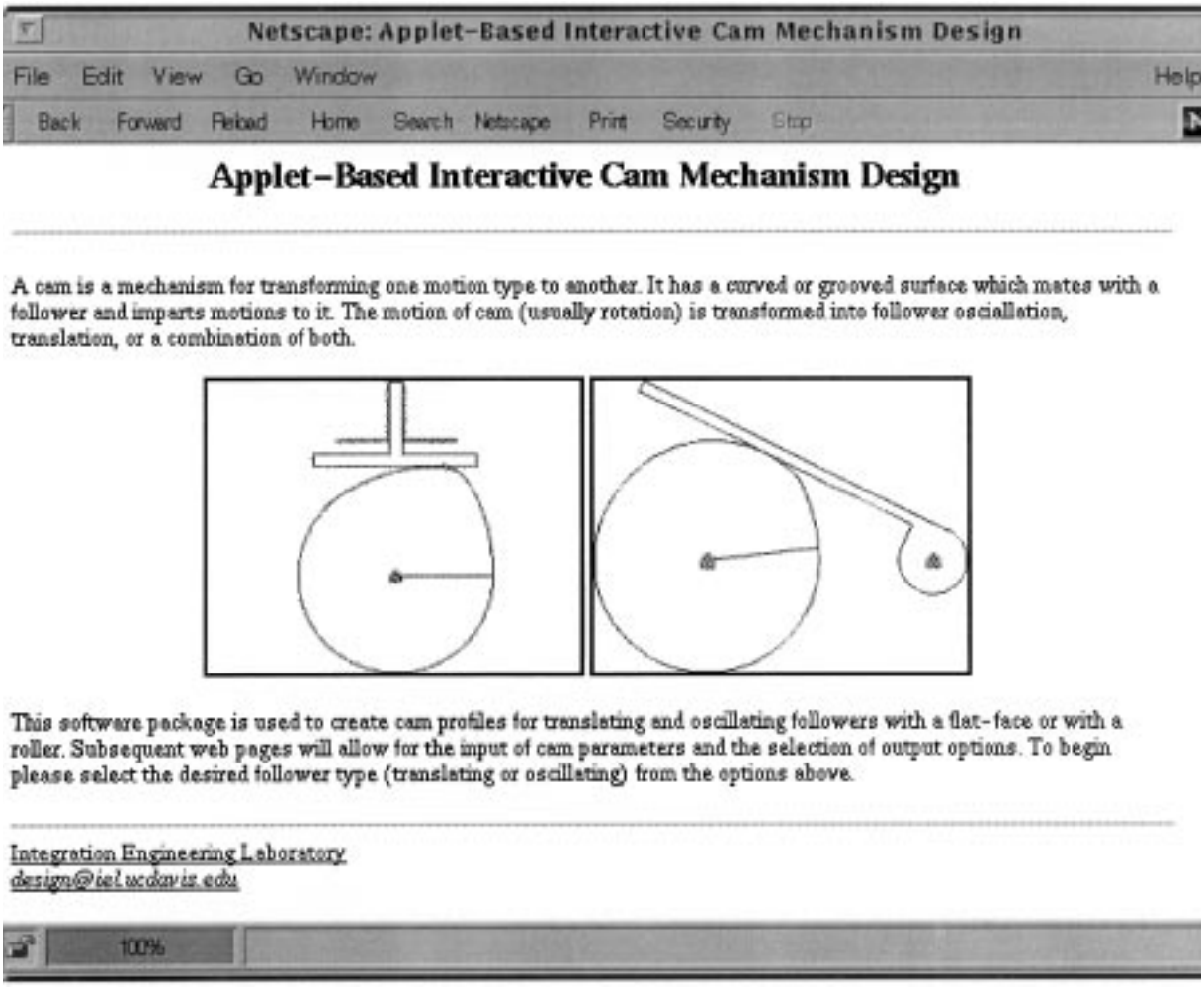

Fig. 1. Homepage for web-based interactive cam design.

The cam profile specified by the user is built up from a disk of this size. Duration is the angular size of the section measured in degrees. Profile points specified how many points are used to calculate the cam profile. The cam angular velocity is measured in rad/s and is positive in the clockwise direction. The final parameter is the motion type. This is the type of the follower displacement profile. It can be chosen to be either harmonic or cycloidal. For cam sections where the follower does not change position, the section is circular and this parameter has no effect.

2.2.2. *Translating follower specific parameters*

The follower offset is the distance from the center of the cam to the line of motion of the follower. To be consistent with (Erdman and Sandor, 1991), a positive follower offset is to the right for flat followers and to the left for roller followers. In both cases the follower offset is specified in inches. The roller radius is only used for the roller follower and is specified in inches. The final parameter unique to translating followers is lift. Lift, specified in inches, is the change in follower displacement for the section. This is a positive number if the follower is to move away from the cam and a negative number if the cam is to move towards the cam. A lift of zero is entered if the follower displacement is to remain constant for the duration of the cam section. For the last section of the cam, the duration and change in lift will be chosen automatically to form a continuous cam profile.

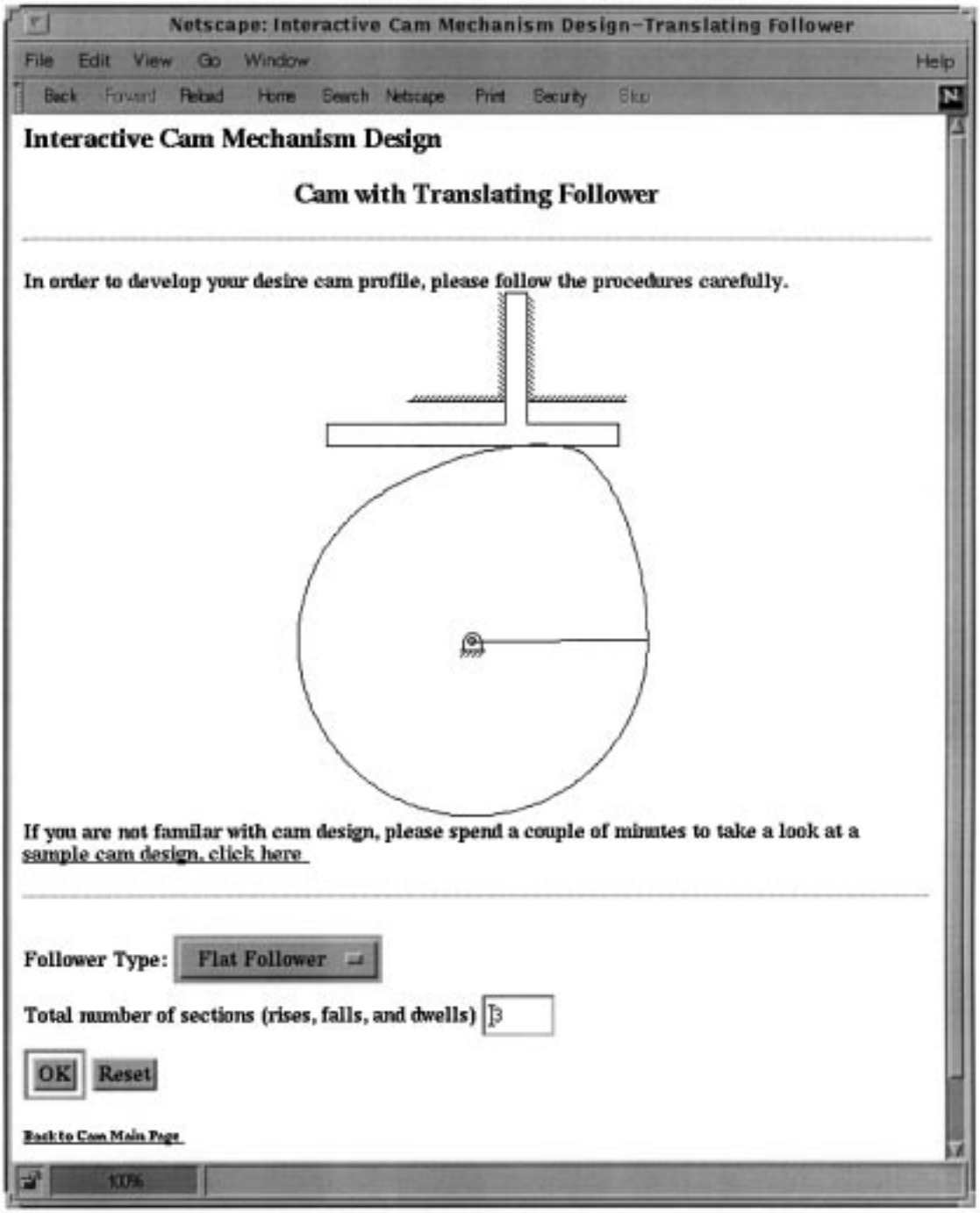

Fig. 2. Selection of the number of cam sections.

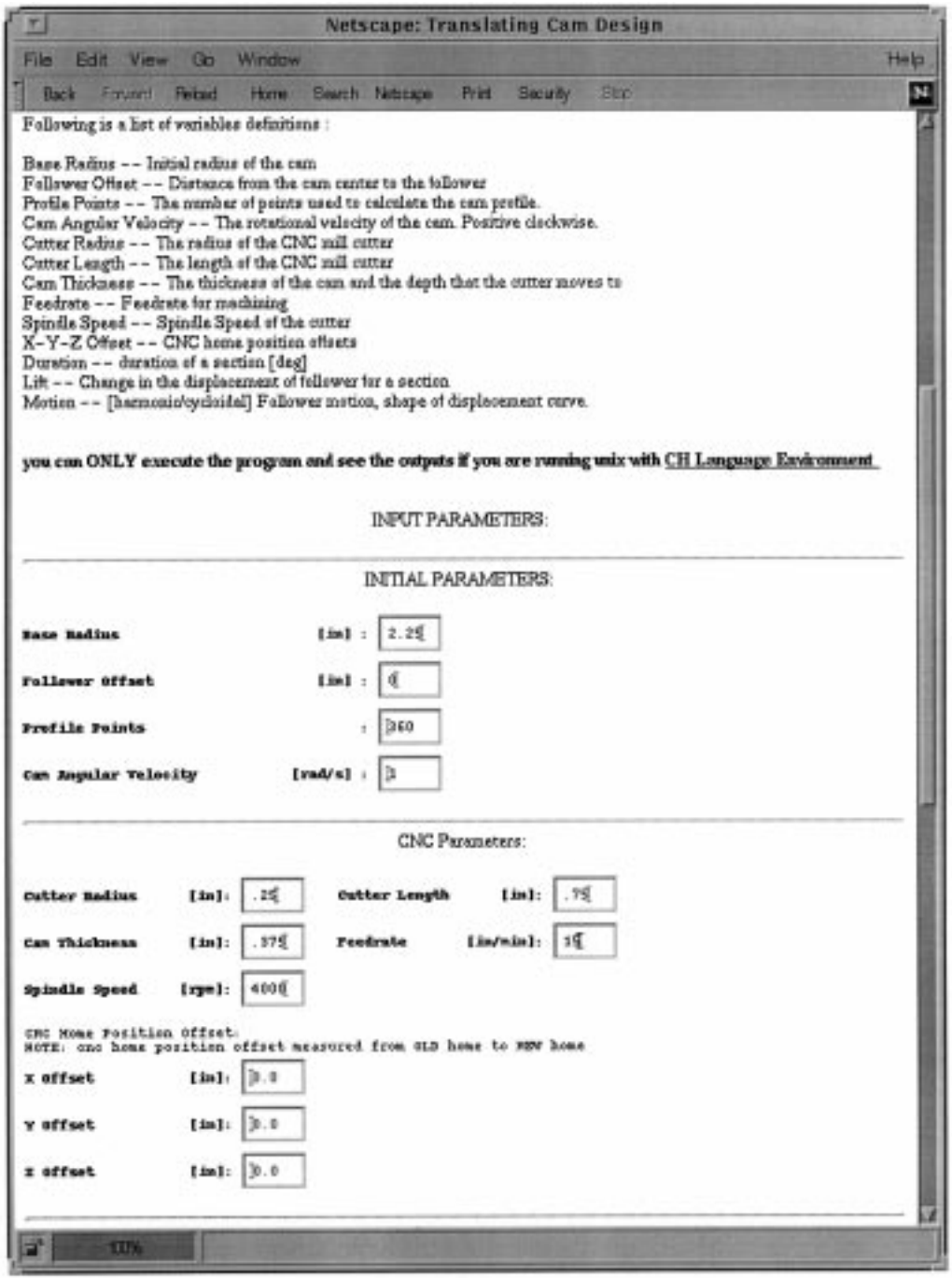

Fig. 3. Base radius and CNC manufacturing parameters.

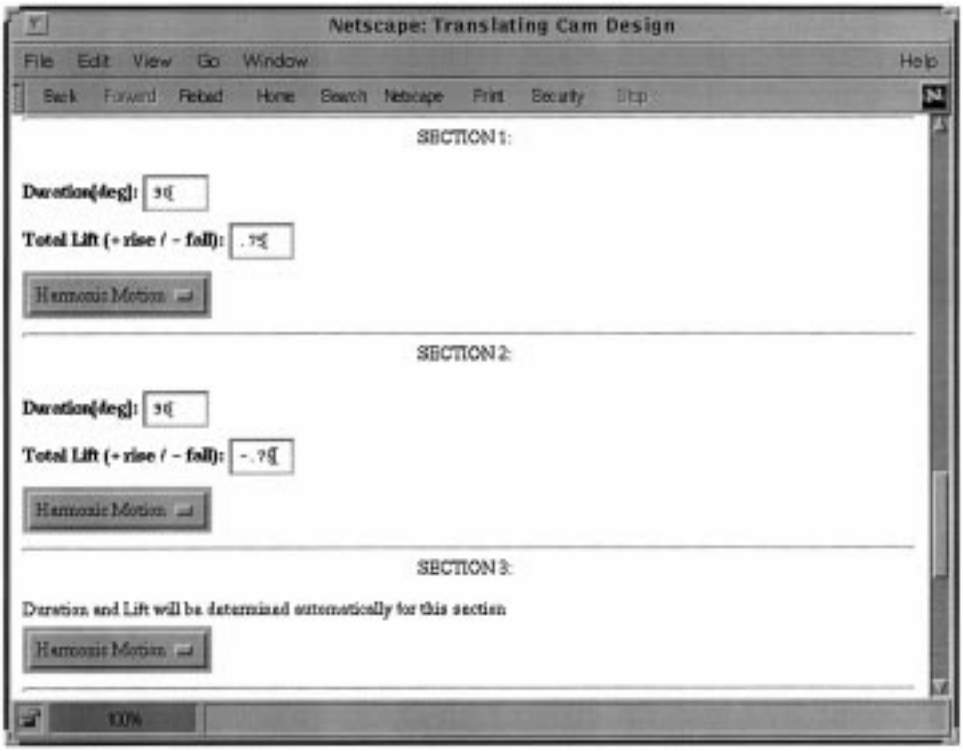

Fig. 4. Section parameters.

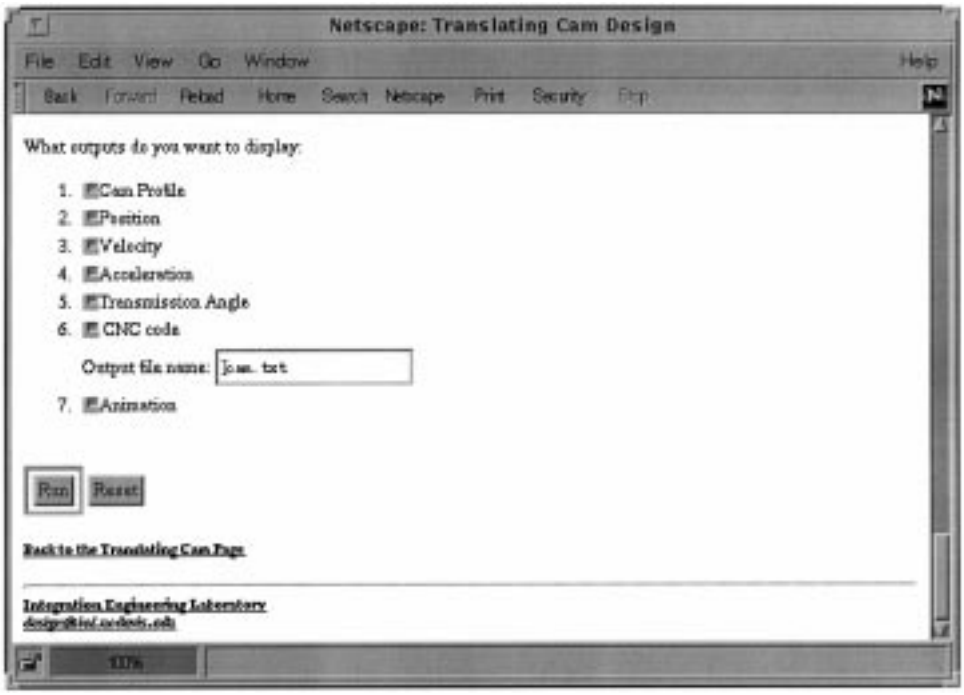

Fig. 5. Selection of output.

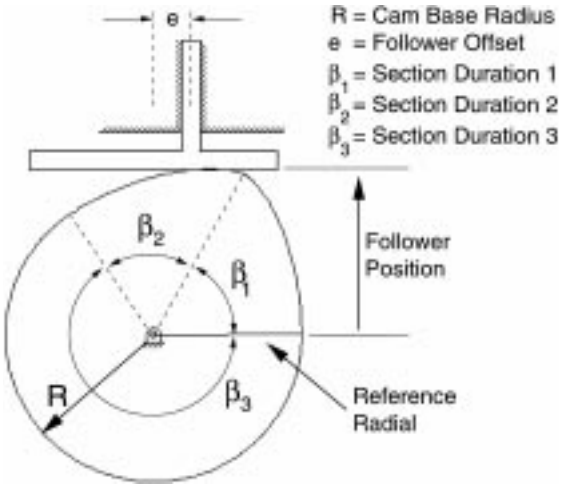

Fig. 6. Translating flat-faced follower parameters.

2.2.3. *Oscillating follower specific parameters*

The oscillating follower has a few parameters that are unique to it. The first of these is the distance between the cam and follower. This is measured from the center of the cam to the follower pivot point and is in inches. This is used for both flat and roller followers. Flat followers also use the follower offset, measured in inches. This is the distance from the follower face

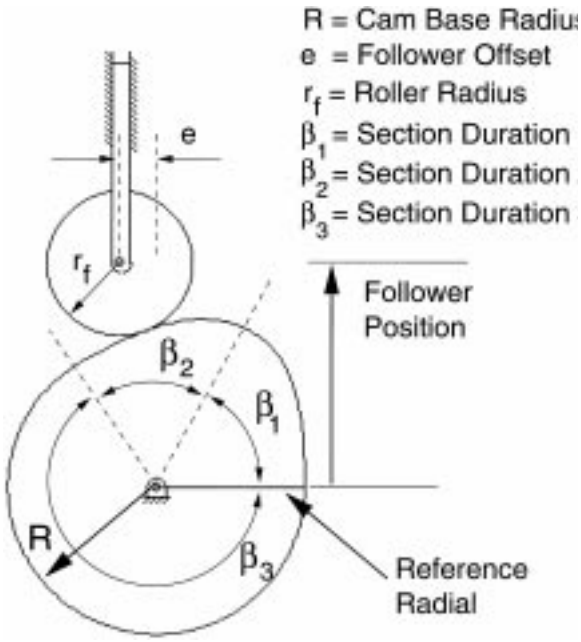

Fig. 7. Translating roller follower parameters.

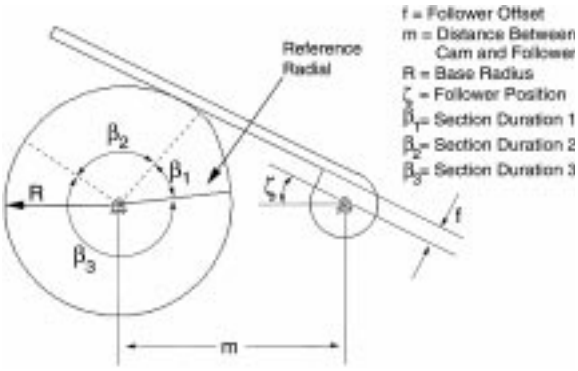

Fig. 8. Oscillating flat-faced follower parameters.

to the follower pivot point. Roller followers use arm length and the roller radius. The arm length is the distance from the follower pivot point to the roller center. Both the roller radius and arm length are in inches. The final oscillating follower parameter is oscillation angle. It specifies the change in the follower displacement for the section and is measured in degree. Like lift, this is a positive number if the follower is to move away from the cam and a negative number if the cam is to move towards the cam. A oscillation angle of zero is entered if the follower displacement is to remain constant for the duration of the cam section. For the last section of the cam, the duration and change in oscillation angle will be chosen automatically to form a continuous cam profile.

2.2.4. *Parameters for generation of CNC code*

For users who desire to create CNC code to manufacture the cam they design, a number of CNC

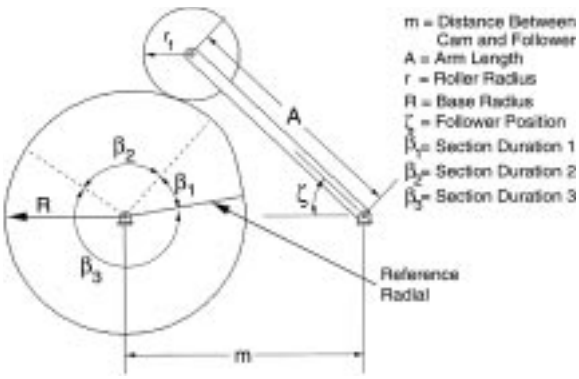

Fig. 9. Oscillating roller follower parameters.

hardware parameters are needed. The cutter length and cutter radius, as shown in Fig. 10, are the dimensions, in inch, of the cutter used by the CNC machine. The cam thickness specifies the thickness of the material used to manufacture the cam and, for code generation, is used as the depth to which the cam is cut. This depth should be less than the cutter length. Feedrate, in inches per minute, is the rate at which the workpiece is moved during machining. The spindle speed is the rotational speed of the cutter, measured in RPM. The CNC home position offset is used if the default home position does not coincide with the desired location of the cam center. As shown in Fig. 11, the home position offset, specified in inches, is measured from the old home position to the new home position.

2.3. *Output options*

The output options are chosen following selection of the cam and CNC parameters, as shown in Fig. 5. The output options include: plots of the cam profile, follower position, follower velocity, follower acceleration, and transmission angle, CNC code for manufacturing of the designed cam, and animation of the cam/follower system. There are two different implementations in our Web-based cam design, one is applet-based and the other is CGI-based. In the applet-based implementation the user may select as many of these options as desired and results are displayed externally to the Web browser. In the CGI-based implementation, the user can only select one output option at a time and the results are displayed inside the Web browser.

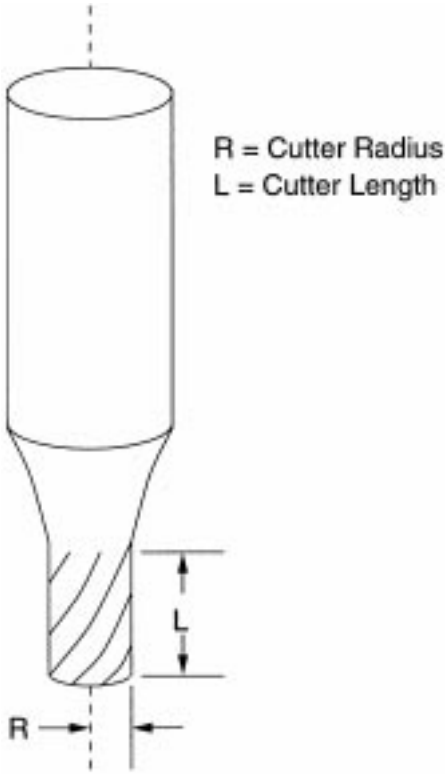

Fig. 10. Cutter dimensions.

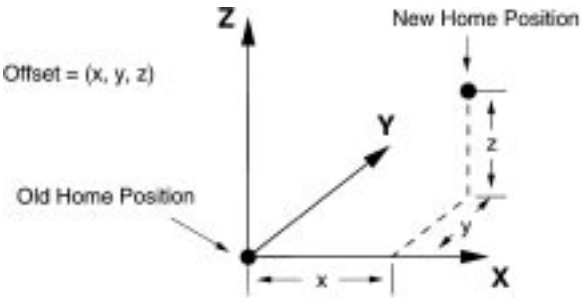

Fig. 11. Home position offset.

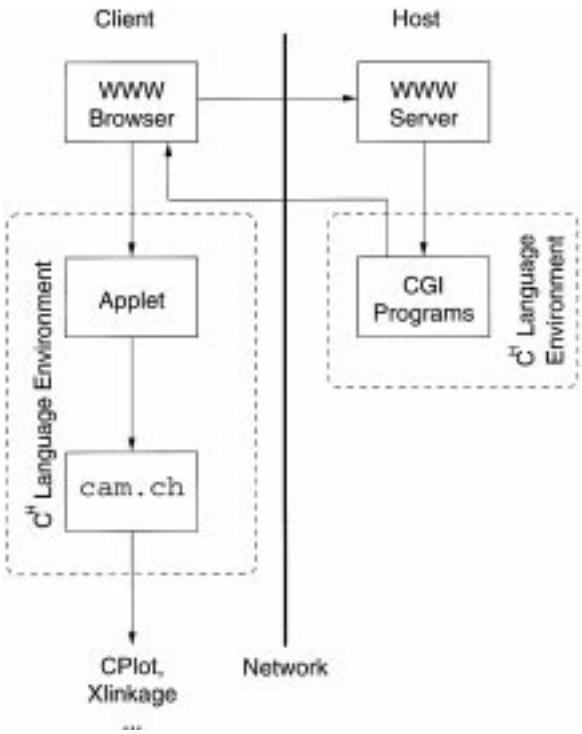

Fig. 12. System architecture for applet-based cam design.

3. Applet-based cam design and implementation

In both versions of the Web interface, once the cam parameters are selected, they are submitted to the WWW server. On the server the parameters are sent to `applet_trans_cam.cgi` or `applet_osc_cam.cgi`, depending on if a translating follower or an oscillating follower was selected, where they are processed and the C^H code necessary for cam design is created. In the applet-based version, as its name implies, this code is a C^H applet. This applet is sent over the network, and is executed on the client machine (Cheng, 1996). The Web browser on the client machine is configured to recognize the applet as a C^H program, and executes it in the C^H language environment automatically. Once executed, the applet in turn executes the `cam.ch` program, which is part of a software library built into the C^H language environment. Program `cam.ch` relies on the lower-level C^H functions found in the CCam and CPlot C^H classes. Program `cam.ch` performs all of the computation for designing the cam through calls to the CCam class. Generation of the data is done by the CCam class, but display of the data is not. Plotting of results is performed using the CPlot class. CPlot in turn uses plotting packages that are already installed on the client machine to display the plots in a separate window. Data generated by the CCam class are sent to the plotting routine for display. For display of the cam profile, in addition to the profile data, the maximum value of the coordinates is also sent to the plotting routine. This value is used for both *x* and *y* coordinates to ensure that the profile is displayed with the proper proportions. CNC code (Vickers *et al.*, 1990) is written by the CCam class directly to a file specified by the user. Animation data are also written to a file first, then sent to the Xlinkage program for display. Xlinkage, a software component for Web-based simulation and analysis of planer mechanical systems (Van Katwyk and Cheng, 1997), reads the data file and animates the designed cam/follower system in a separate window.

This scheme has a number of salient features designed for a network computing environment. C^H applets are typically very small because they utilize the functions and features that are built-in to the C^H language environment. This makes both the generation of the applet and its transmission to the client very efficient because large volumes of source code do not need to be generated or sent with the applet. Because the applet is executed on the client machine, the majority of the computational load for a particular task does not rest on the host machine (the Web server). This is particularly advantageous in cases where a large number of people are accessing the Web pages, as is the case in the undergraduate mechanism design course where this package has been used. If the Web server were required to do intensive computation, such as animation, and a large number of users invoked the cam package at the same time, it is possible that it would be bogged down to the point where it would be rendered virtually unusable. The use of applets does, however, have one slight drawback. Because the applet makes use of software resident to the client, the user must install C^H in the local client machine. Although this is at worst only an inconvenience, as C^H is available for downloading on the WWW, some potential users may not wish to take the time to download and install it.

4. CGI-based cam design and implementation

The CGI-based interface performs all of the same computational functions that the applet-based interface does. The primary difference between the two is

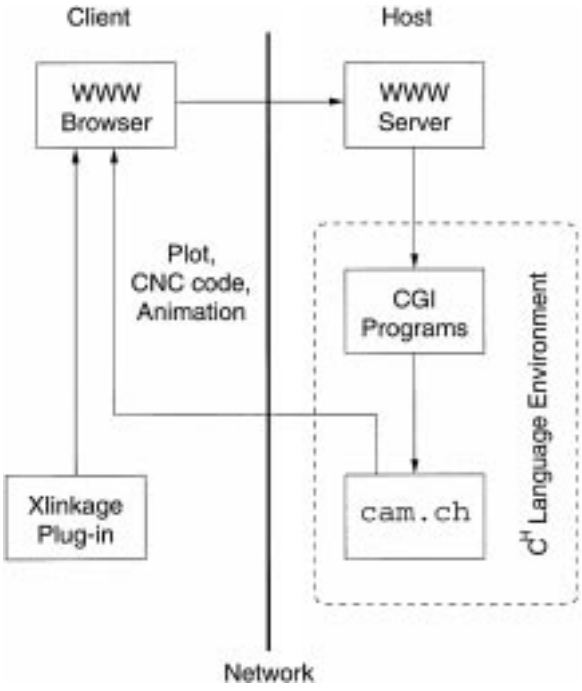

Fig. 13. System architecture for CGI-based cam design.

that for the CGI-based interface the computation required for cam design is done in the host computer. When the cam parameters are submitted to the WWW server, they are sent to `cgi_osc_results.cgi` or `cgi_trans_results.cgi`, depending on the follower type, and processed. These CGI programs then execute a second CGI program named `cgi_make_osc_cam.cgi` or `cgi_make_trans_cam.cgi` which in turn executes `cam.ch` on the server and sends it the processed parameters. The `cam` program then performs the calculations for cam design and generates the data for the desired output. Plots of the results, CNC code, and animations can be viewed, but only one of them can be viewed for each submission to the `cam` program. Unlike the applet-interface, the CGI programs do not depend on software installed on the client machine. Because of this, all results are displayed in the browser window itself.

Plots of the cam profile, follower position, follower velocity, follower acceleration, and transmission angle are sent to the browser as GIF files. These GIFs are also generated using the C^H CPlot class. As shown in Fig. 14, `cam.ch` passes the data to the CPlot class with an option set for generating a GIF file as the standard output stream. The Web browser then takes the GIF data from this stream and displays it directly in the browser window. The CNC code, if selected, is also displayed in the browser window, where it may be saved for later use. Animations of the cam and follower are displayed using a modified version of the Xlinkage program. This version of Xlinkage was created to run as a Netscape plug-in. A plug-in is a third-party extension to the capabilities of the Netscape Navigator. Implementing Xlinkage as a plug-in allows animations to be displayed within the browser window. The animation data created by

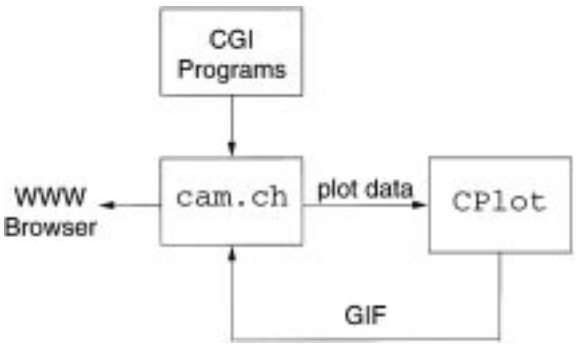

Fig. 14. Plot generation for CGI-based cam design.

`cam.ch` are sent directly to the Web browser where they are interpreted by the plug-in and animated on the screen. A difficulty with displaying the results in the browser window is unwanted caching of previous results by the Web browser. Previous results can sometimes be re-displayed, even if different output is selected or different parameters are submitted. An attempt to solve this problem was made by sending the statement `Pragma: no-cache` to the web browser before the output results. This statement was intended to stop the results from being stored in the browser cache, but it was discovered that it is not supported by all browsers. An alternate solution that was tried was to explicitly inform the browser of when a cached document was to expire using the `Expires:` directive (Network Working Group, 1997). However, this too was found to be unsupported by some browsers. The method that was ultimately used was to pass the cam parameters along with the name of the CGI. This creates a unique link name each time a different output option is selected or a cam parameter is changed. As a result, different results are displayed for each submission to the Web server.

Although the CGI-based interface has the potential to slow down the host machine, it has the advantage of, with the exception of the plug-in, not requiring any special software to be installed on the client machine. The CGI-based implementation also has the advantage of being very easy to maintain. Because all of the software used by the CGI is on the host machine, the user does not need to be concerned with what version of `cam.ch` or what version of C^H is being used, or with updating software on the local client.

5. Program `cam.ch` and CCam class

The CCam class is the heart of the cam design package. It contains the functions that perform the actual calculations to create the cam profile based on the user input. The CCam class generates the data for plotting, the CNC code, and the data for cam animation. All of the cam parameters, and the functions need to use them, are contained in the CCam class. A summary of these functions is located in Fig. 15. The primary purpose of `cam.ch` is to provide an interface to the CCam class. Program `cam.ch` allows the design capabilities of the CCam class to be used easily, without the

need to create a separate C^H program. The cam parameters and output options are set through a simple command-line interface. Program `cam.ch` processes these arguments and calls the appropriate CCam functions. This modular structure allows for a great deal of simplification in the implementation of the previously described CGI programs. Only the small amount of C^H code necessary to call `cam.ch` needs to be generated each time a cam design is submitted.

6. Web-based cam design examples

In this section, two sample cam design problems will be presented to illustrate the features and applications of Web-based cam design.

Problem 1: Using the applet-based cam design Web pages, generate a cam profile for a translating flat-face follower. The cam should have a base radius of 2.25 inches, no follower offset, and an angular velocity of 1 rad/s. During the first 90° of cam rotation the follower should move outward 0.75 inches with harmonic motion. During the next 90° the follower should move inward 0.75 inches with harmonic motion. For the remainder of the cam rotation the follower should not change position. Three hundred and sixty points should be used to calculate the results. Generate plots for the follower position, follower velocity, follower acceleration, the transmission angle, and the cam profile. Also generate an animation of the cam using 12 positions and produce CNC code using the following parameters. The cutter has a radius of 0.25 inches and a length of 0.75 inches. The spindle speed should be set to 4000 rpm and the feedrate should be 15 inches/minute. The thickness of the cam is 0.375 inches. No home position offset is used.

In Problem 1, we will generate plots for the follower position, follower velocity, and follower acceleration, transmission angle, as well as a plot for

Function	Description
AddSection()	Add a cam section to a previously declared instance of the cam class.
AngularVel()	Set the cam angular velocity.
Animation()	Display an animation of the cam.
BaseRadius()	Set the cam base radius.
CNCCode()	Set the filename for CNC code output.
CutDepth()	Set the cut depth for CNC code generation.
Cutter()	Set the cutter parameters for CNC code generation.
CutterOffset()	Set the cutter home position offset for CNC code generation.
DeleteCam()	Remove and data from a previously used instance of the CCam class.
Feedrate()	Set the feedrate for CNC code generation.
FollowerType()	Set the cam follower type.
GetCamAngle()	Get the cam angular position data.
GetCamProfile()	Get the cam profile data.
GetFollowerAccel()	Get the cam follower acceleration data.
GetFollowerPos()	Get the cam follower position data.
GetFollowerVel()	Get the cam follower velocity data.
GetTransAngle()	Get the cam transmission angle data.
MakeCam()	Generate the cam data and write CNC code to a file.
PlotCamProfile()	Plot the cam profile data.
PlotFollowerAccel()	Plot the cam follower acceleration vs. the cam position.
PlotFollowerPos()	Plot the cam follower position vs. the cam position.
PlotFollowerVel()	Plot the cam follower velocity vs. the cam position.
PlotOutputType()	Set the cam plotting output type.
PlotTransAngle()	Plot the cam transmission angle vs. the cam position.
SpindleSpeed()	Set the spindle speed for CNC code generation.

Fig. 15. Member functions of class CCam.

the cam profile. We will also output the CNC code and animation of the follower and cam. The Web pages used for entry of the parameters for this problem are shown in Figs. 1–5 in the previous sections. The C^H source code shown in Fig. 16 could also be used to solve this problem. Figures 17–19 show the plots of the follower position, velocity and acceleration. Figure 20 shows the transmission angle and Fig. 21 shows the plot of the cam profile. Figure 22 is an excerpt of the generated CNC code. Figure 24 shows the cam manufactured using the CNC code and Fig. 23 shows the animation.

Problem 2: Using the CGI-based cam design pages, generate a cam profile for an oscillating roller follower. The base radius of the cam should be 4 inches. The follower should have a radius of two inches, should have an arm length of 12 inches and should be located 10 inches from the cam center. four hundred points and an angular velocity of 1 rad/s should be used. During the first 120°
of cam rotation the follower should move out five degrees with cycloidal motion. During the next 120° the follower should move back to its original position with cycloidal motion. The follower will remain at rest for the remainder of the cam rotation. Plot the follower position, follower velocity, follower acceleration, transmission angle and the cam profile. Animate the cam/follower system.

In Problem 2, we will also generate plots of the follower position, follower velocity, follower acceleration, transmission angle and the cam profile, as well as an animation of the cam. The Web pages for entry of the oscillating follower parameters are similar to those for the translating follower, the primary difference being that the oscillating follower has a few additional parameters. To produce the desired output we must submit the parameters a total of six times, once for each plot and once for the animation. The resulting plots of follower position, follower velocity, and follower acceleration are shown in Figs. 25–27. The plot of the transmission angle is shown in Fig. 28. The plot showing the cam profile is shown in Fig. 29 and the animation is shown in Fig. 30. Unlike the applet-based implementation, all of these results in the CGI-based implementation are displayed within the browser window.

```
#include <cam.h>

int main() {

    class CCam cam;

    cam.FollowerType(CAM_FOLLOWER_TRANS_FLAT, 0);
    cam.BaseRadius(2.25);
    cam.AngularVel(1);
    cam.AddSection(90, .75, CAM_MOTION_HARMONIC);
    cam.AddSection(90, -.75, CAM_MOTION_HARMONIC);
    cam.AddSection(CAM_DURATION_FILL, 0, CAM_MOTION_HARMONIC);
    cam.CNCCode("cam_code.nc");
    cam.Cutter(.25, .75, 1);
    cam.SpindleSpeed(4000);
    cam.Feedrate(15);
    cam.CutDepth(.375);
    cam.CutterOffset(0, 0, 0);
    cam.MakeCam(360);
    cam.PlotFollowerPos(NULL);
    cam.PlotFollowerVel(NULL);
    cam.PlotFollowerAccel(NULL);
    cam.PlotTransAngle(NULL);
    cam.PlotCamProfile(NULL);
    cam.Animation(12);

}
```

Fig. 16. C^H source code for problem 1.

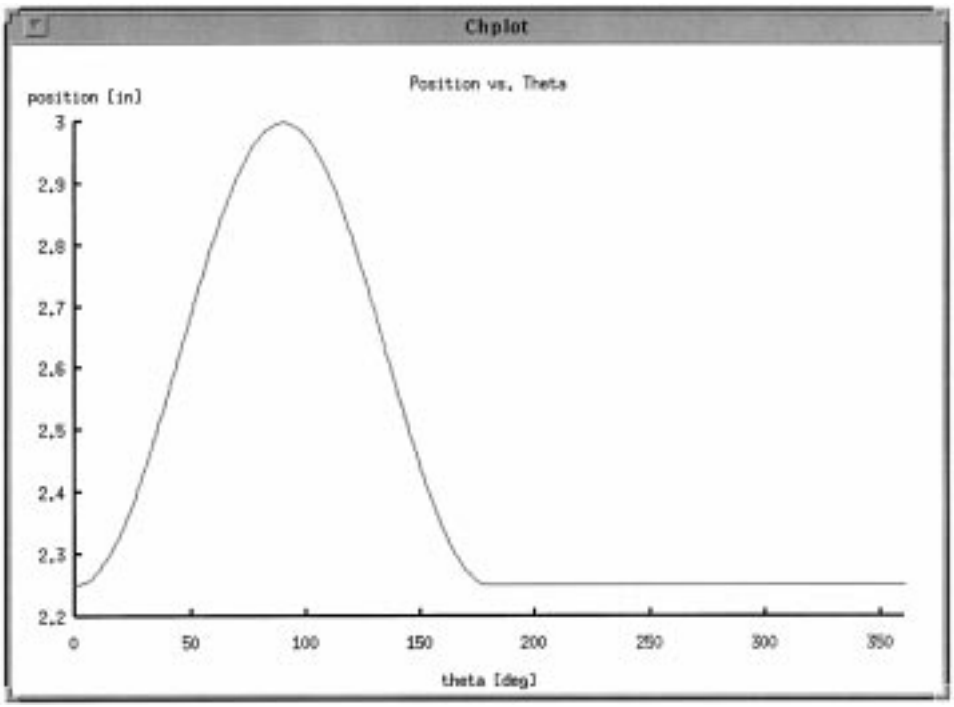

Fig. 17. Translating follower position.

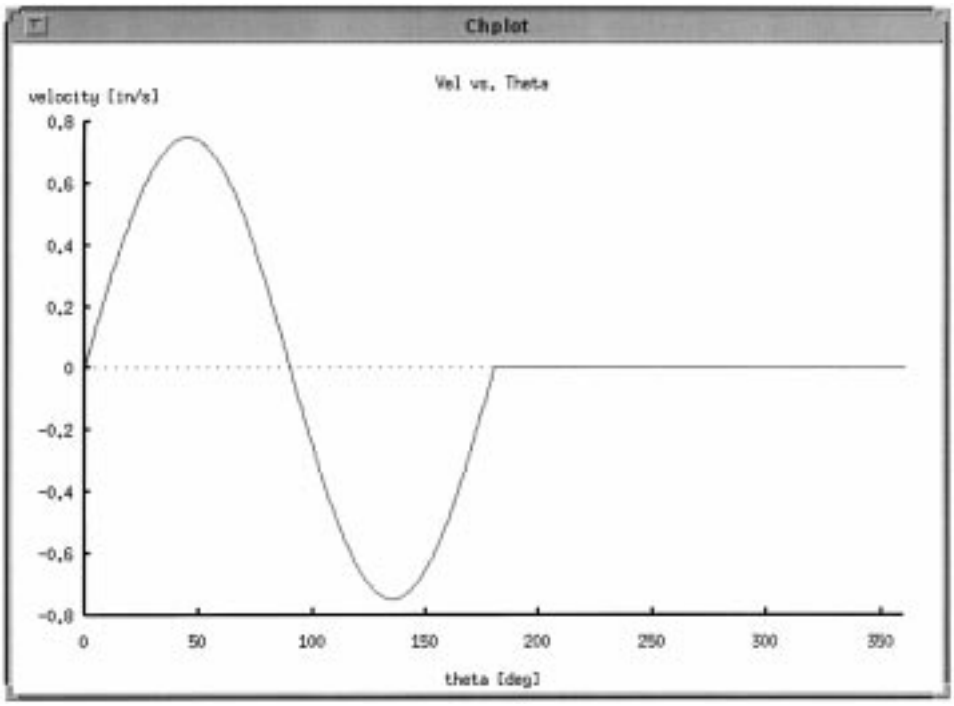

Fig. 18. Translating follower velocity.

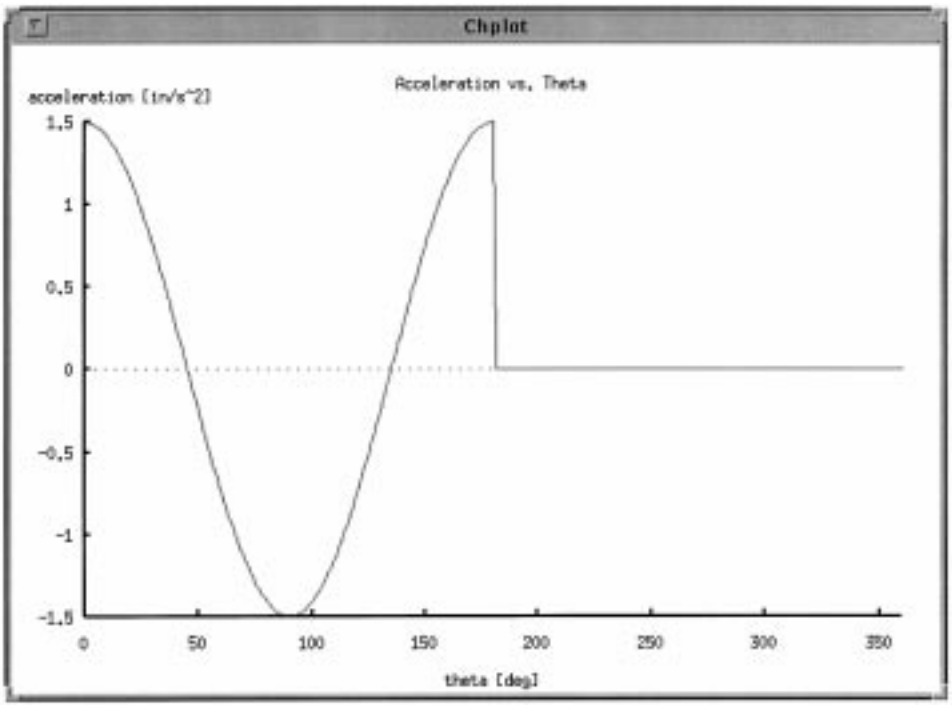

Fig. 19. Translating follower acceleration.

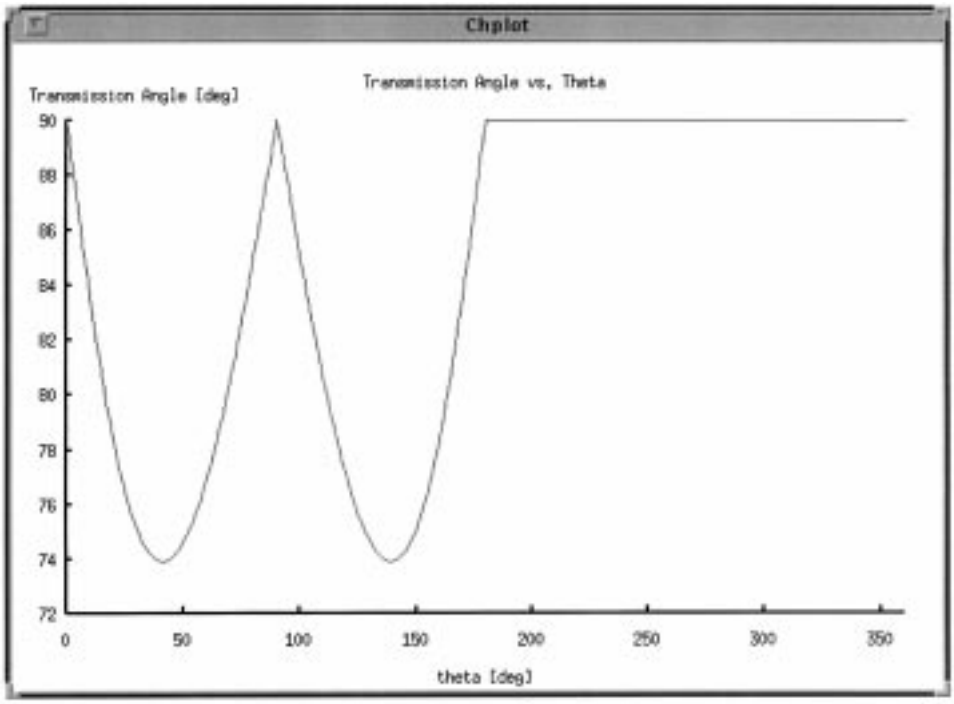

Fig. 20. Translating follower transmission angle.

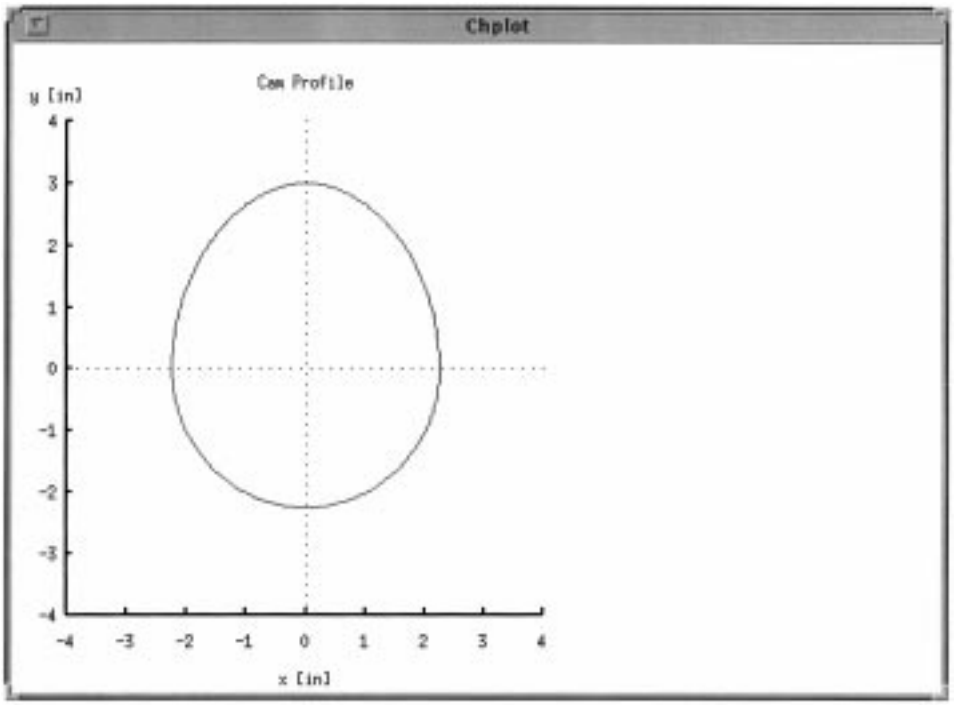

Fig. 21. Translating follower cam profile.

7. Conclusions

We have developed a web-based cam design software package under the network computing paradigm. Unlike conventional cam design packages, our software package is modular, easy to use and maintain. It can be accessed simultaneously by multiple users through Web browsers worldwide. At the current implementation, the system can be used to design both translating or

```
N10 G90 G00 X0.000 Y0.000 Z0.000; move to home
N20 M08; coolant on
N25 S4000 M3; spindle on
N30 X0.000 Y0.080; move to starting point
N35 G01 Z-0.375 F15.000;
N40 X2.500 Y0.080
N45 X2.499 Y0.070
N50 X2.498 Y0.140
.
.
.
N930 X-2.498 Y0.140
N935 X-2.499 Y0.070
N940 X-2.500 Y0.000
N945 G03 X2.500 Y-0.000 I0.000 J0.000; circular CCW
N950 G90 G01 Z0.500;
N955 M05; stop spindle
N960 M09; coolant off
N965 G90 X0.000 Y0.000;
N970 M22
```

Fig. 22. CNC code for the manufacturing of the CAM shown in Fig. 24.

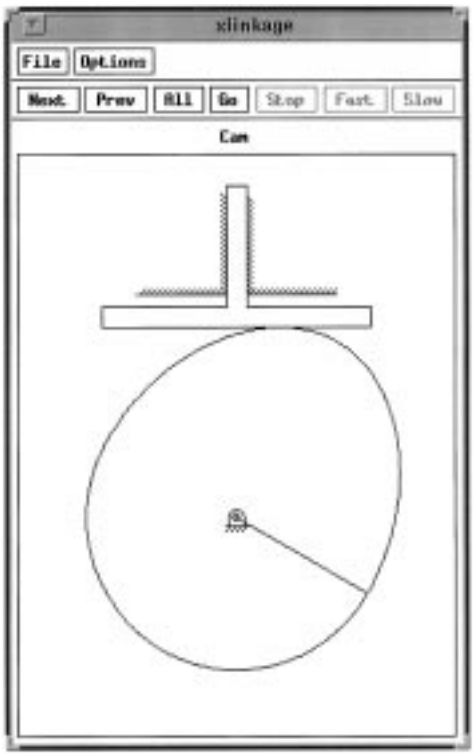

Fig. 23. Animation of CAM with translating follower.

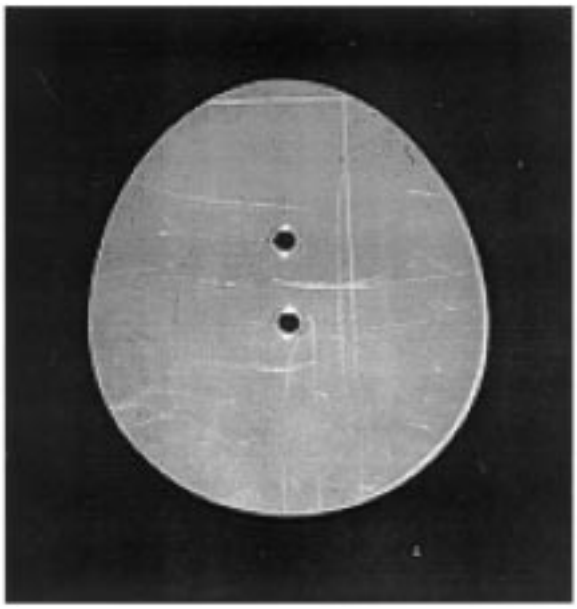

Fig. 24. Manufactured cam.

oscillating cams with flat-face or roller followers and harmonic or cycloidal motion types. The user can obtain cam profile, transmission angle, and position, velocity, and acceleration profiles of the follower. The animation of cam/follower system can also be performed through the web. The system can also generate a CNC code for manufacturing of the cam.

The software package is implemented in C^H, an object-oriented language environment, through a cam class called CCam written in C^H. The applet-based implementation of the user interface makes a good use of the decentralized nature of the Web by utilizing locally available software and hardware resources. Because the size of an applet code is small, it is easy to generate and inexpensive to transmit. Applets are executed locally on the client machine. Therefore, they do not burden the server with intensive calculations such as animation. The CGI-based implementation, on the other hand, does not rely on locally available resources, placing the majority of the

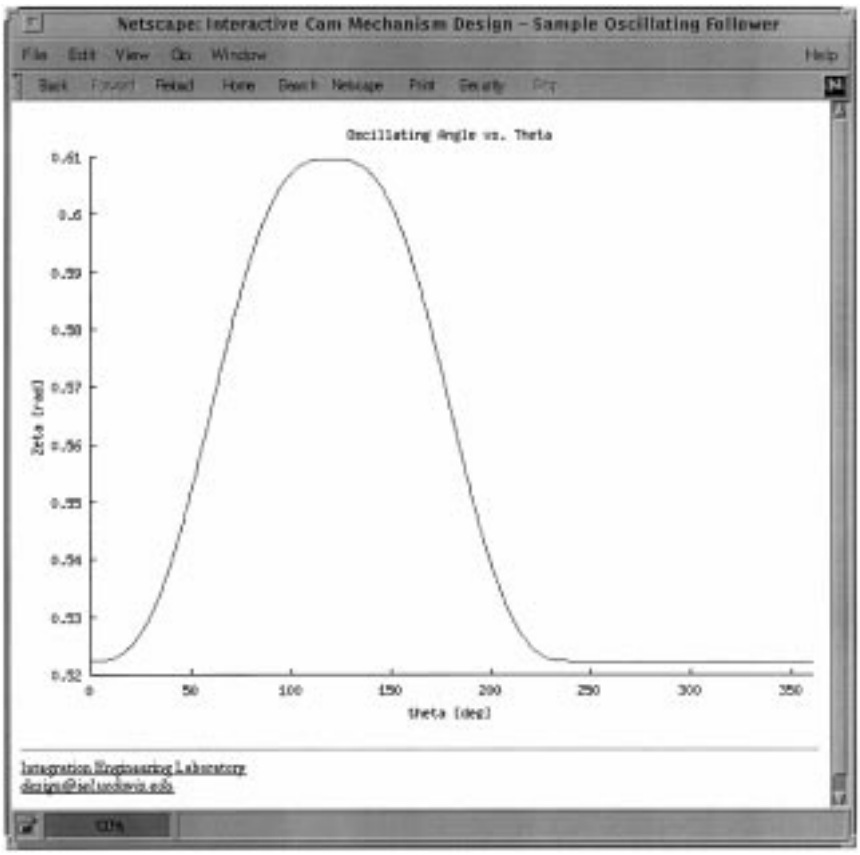

Fig. 25. Oscillating follower position.

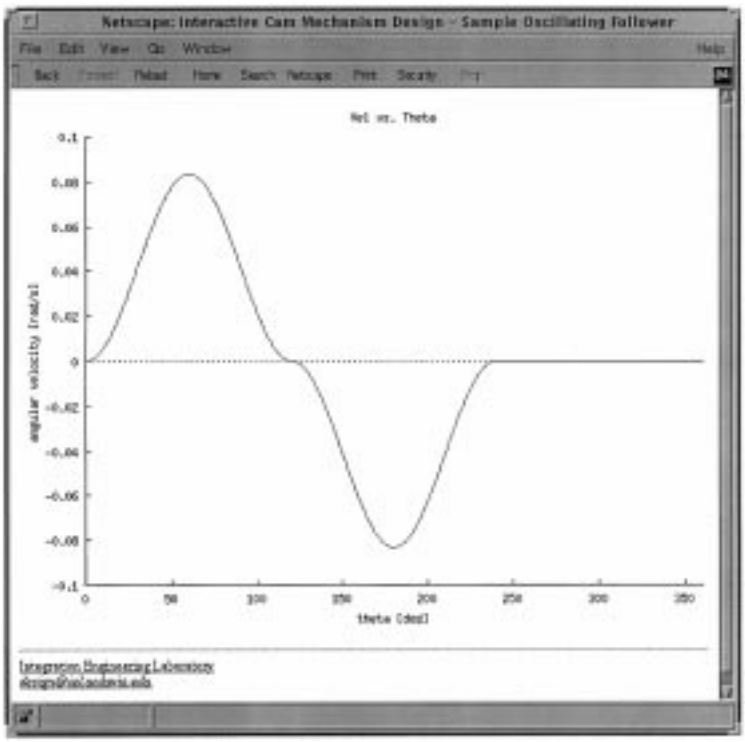

Fig. 26. Oscillating follower velocity.

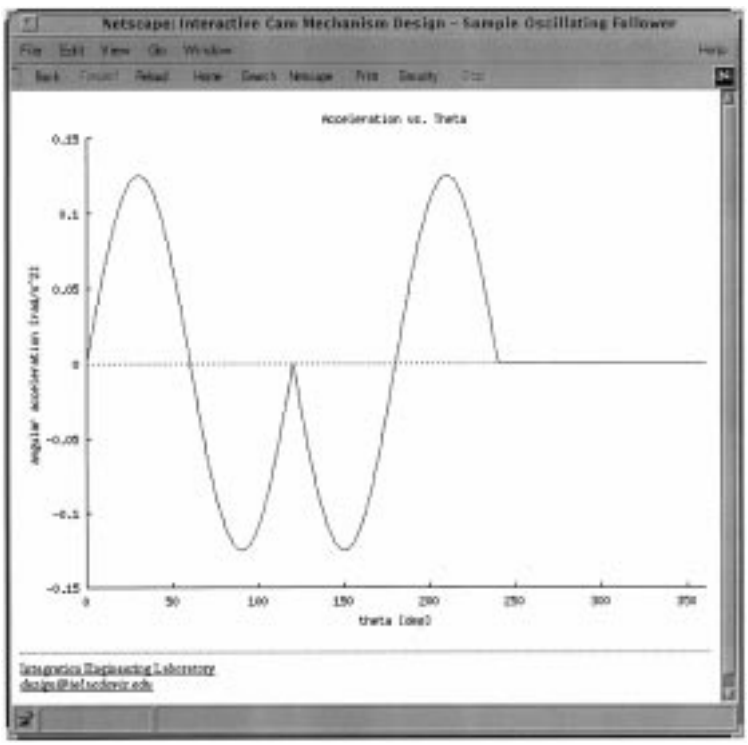

Fig. 27. Oscillating follower acceleration.

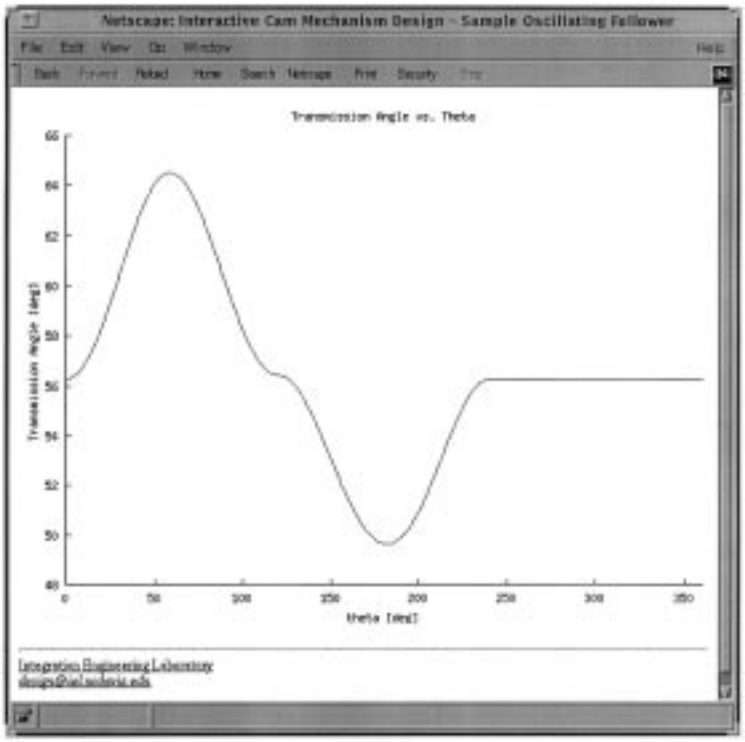

Fig. 28. Oscillating follower transmission angle.

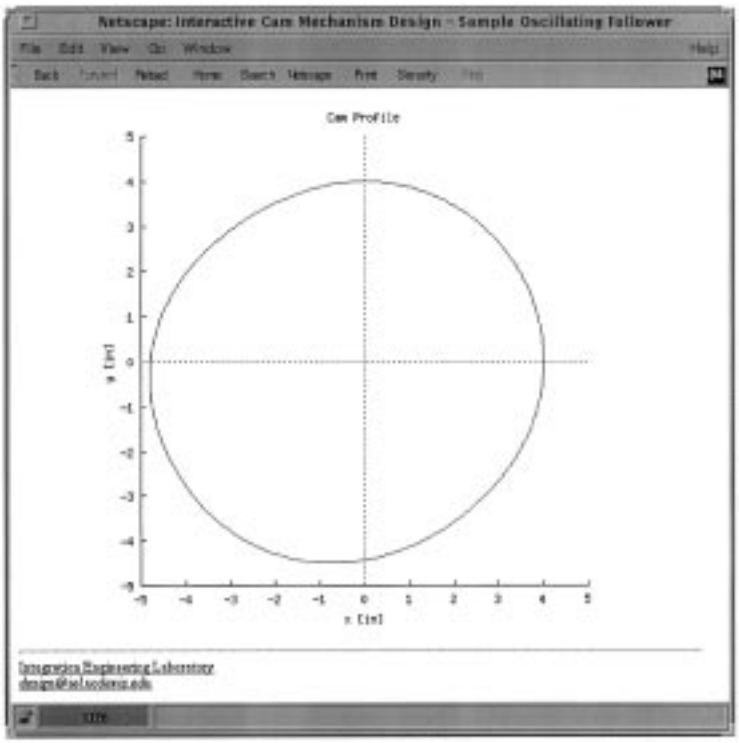

Fig. 29. Oscillating follower cam profile.

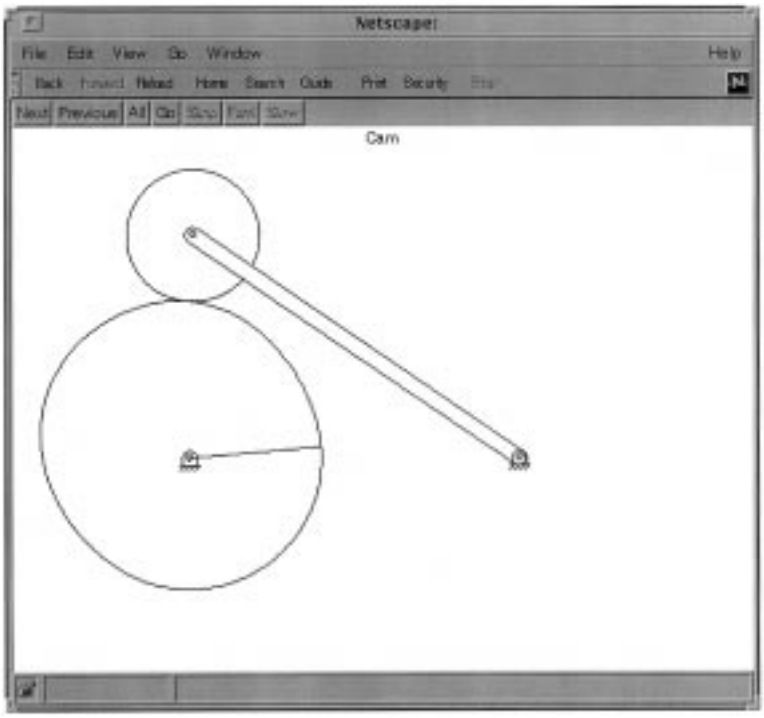

Fig. 30. Animation of cam with oscillating follower.

computational burden on the host machine. But, the CGI-based interface can be used with little special software, which makes it immediately accessible to nearly any computer with a Web browser. In addition, because of the CGI programs' centralized nature, it is easier to maintain than the applet-based implementation.

Our web-based cam design system has been used in an undergraduate design class and used by many students in other campuses and design engineers through the Web. The principle of Web-based software design and implementation in C^H under the network computing paradigm presented in this paper can be applied to other areas of design and manufacturing.

Acknowledgment

This work was supported in part by the National Science Foundation under Grant DMI-9309207 and DMI-9622588, by the University of California, Davis through an Undergraduate Instructional Improvement Grant, by industrial affiliates of the Integration Engineering Laboratory, BSDI, Datacube, Delta Tau Data Systems, HP, Intel, JR3, Lynx Real-Time Systems, Microsoft, Motorola, Panasonic Broadcast & Television Systems, and Unmanned Solutions.

References

Aziz, R. (1996) Development of an integrated system for cam design and manufacture with graphical user interface. *CD-ROM Proc. 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, Irvine, CA.

Cheng, H. H. (1993a) Scientific computing in the C^H programming language. *Scientific Programming*, **2**(3), 49–75.

Cheng, H. H. (1993b) *The C^H Programming Language*,

Revision 1.1, Department of Mechanical and Aeronautical Engineering, University of California, Davis.

Cheng, H. H. (1996) CGI Programming in C, *C/C++ Users Journal*, November, 17–21.

Cheng, H. H. (1997) A network computing language environment for design and nanufacturing. *CD-ROM Proc. 1997 ASME Design for Manufacturing Conference*, Sacramento, CA.

Erdman, A. G. and Sandor, G. N. (1998) *Mechanism Design Analysis and Synthesis*, Vol. 1, 3rd Edn, Prentice Hall, Englewood Cliffs, NJ.

Felton, M. (1997) *CGI Internet Programming with C++ and C*, Prentice Hall, Upper Saddle River, NJ.

Knowledge Revolution (1989), *Working Model User's Guide*, Knowledge Revolution.

Network Working Group (1997), Hypertext Transfer Protocol – HTTP/1.1, http://ds.internic.net/rfc/rfc2068.txt.

Mabie, H. H. and Ocvirk, F. W. (1975) *Mechanisms and Dynamics of Machinery*, 3rd Edn, John Wiley and Sons, New York, NY.

Martin, G. H. (1982) *Kinematics and Dynamics of Machines*, McGraw-Hill, New York, NY.

Norton, R. L. (1992) *Design of Machinery*, McGraw-Hill, New York, NY.

Structural Dynamics Research Corporation (1996), *I-DEAS Master Series Student Guide*, Structural Dynamics Research Corporation.

Van Katwyk, K. and Cheng, H. H. (1997) Xlinkage: A web-based analysis and simulation tool for planar mechanical systems. *CD-ROM Proc. 1997 ASME Design Engineering Technical Conferences*, Sacramento, CA.

Vickers, G. W., Ly, M. H. and Oetter, R. G. (1990) *Numerically Controlled Machine Tools*, Ellis Horwood, New York, NY.

Wang, S. (1997) Motion simulation of gears and cams using working model. *CD-ROM Proc. 1997 ASME Design Engineering Technical Conferences*, Sacramento, CA.