Chapter 5

The MachineShop System

The design and construction of automata can be intimidating for the adult novice and even more daunting for a child. Appropriate tools and knowledge support materials can make the process of both building and learning easier and improve the quality of the user's experience. By extension, this increases the likelihood that a user will continue to create automata, with the increased probability that her skill and understanding will both increase. The MachineShop system was created to fill this role by providing users with a collection of core and supplemental tools to support this task.

This chapter begins with an overview of the MachineShop software, a discussion of its constituent parts and their functions, and an examination of the files the software creates and uses. This is followed by an introduction to recent advances in affordable computer controlled machine tools and output devices with an emphasis on carbon-dioxide laser cutters of the type used in this research. Examples of how MachineShop can work with other devices are also provided. The chapter concludes with a description of the supplemental materials created and appropriated to support users as they make their own automata.

5.1 The MachineShop Software

MachineShop was created to provide children aged 11 to 13 years with a computer-aided design and computer-aided manufacturing (CAD/CAM) tool that they could use to make the mechanical components needed to construct their own contemporary automata. Commercial design

tools (see Section 3.2) are too complex, feature laden, and general purpose for children to use in specialized domains such as automata design. This is frustrating for children and causes them to spend too much of their time with the software trying to discover and remember the steps necessary to do what they want. The intent with MachineShop was to create a tool specific to the task that supported users as they acquired domain knowledge and mechanical reasoning skills through the process of building.

This section details the current form of the MachineShop software and how it focuses on supporting users as they create automata. It discusses the major challenges that have been faced in the creation of the software and how they have been addressed through both design decisions and domain constraints.

5.1.1 High Level Considerations

To insure that MachineShop would meet the needs of its intended users A set of six core metrics were derived empirically and they were then used to guide the initial design choices for the MachineShop software.

(1) The purpose of the MachineShop system is to support the activities that children do as they design and build contemporary automata. This support should be as extensive as possible while keeping the system simple to learn and easy to use.

(2) Children have access to computers that come from a variety of manufacturers. These computers are of varying ages and capabilities, and use an assortment of operating systems. This means that software has to be created for as many of these platforms as possible in ways that do not make unwarranted computational demands on hardware that is one, two, or even three generations old.

(3) Because the software will be used for the creation of physical objects, users should be able to create and manipulate these objects in as natural a way as possible.

(4) The interface should present information to the user in a consistent and meaningful fashion. Where controls are duplicated from one tool to another they should be, as much as possible, in the same locations. Information should be presented in appropriate forms and multiple forms should be used when needed. These multiple forms should be placed near each other so that the user may take advantage of them without spending time trying to locate them.

(5) The version of the MachineShop software that was used for the research described in this document should be considered only the initial version. It should be extensible by anyone who cares to obtain the source code and modify it.

(6) The MachineShop software should be available to anyone without cost.

Most of these constraints can be addressed by the appropriate choice of a programming language and supporting libraries in which to write the software.

(1) The language must have a compiler, interpreter, or runtime environment for each hardware and operating system combination on which the software will be expected to run and all executable files created and libraries used must be free from licensing arrangements that would restrict the use of the software or impose fees for its use.

(2) The underlying structure of the language and its libraries should map logically to the problem domain.

(3) The language must have a freely available set of graphical libraries for both the creation of the user interface and the display of the graphical information needed by the user.

(4) The language should be one that is easily learned and has a large user community to enable persons other than the original author to make changes to the software in order to fill evolving needs.

(5) Implementations of the language should be available at no cost for a wide variety of hardware and operating system combinations.

These requirements led to the choice of Java (in particular the Java 2 standard edition) for the creation of the MachineShop software. A Java Virtual Machine (JVM) is freely available for almost every platform that a child might have access to. It has a library of graphical interface components (Swing) as well as libraries for 2-D and 3-D graphics that can be easily and freely obtained. The Java language has a large (and growing) user community and can be obtained for developing on many platforms without cost. Everything that the MachineShop software should need to do could be done in Java without the need for commercial extensions to the language or paying licensing fees.

The interface design choices were developed using the methods of Task Centered Design [70] and recent work on the cognitive aspects of using figures and animations to support text based learning [13]. Because the intended users of MachineShop are children, particular care was taken to make the interface accessible and meaningful to that population.

A survey of almost one hundred sample automata was conducted with the goal of finding the best initial capabilities for the software in the areas of supported tasks and mechanism creation. Although the mechanisms in these automata form a continuum from very simple to very complex, they do have much in common both in process and in mechanical composition. The construction of an automaton can be divided into three tasks. First, an idea is refined into a workable form. Second, a mechanism is devised and tuned to provide the desired behavior. Finally, the automaton is fabricated, assembled, and decorated. Not all of these tasks need software support but identifying the ones that do, especially with the intended user community in mind, suggests three tasks in which the software should assist the user.

- The software should assist the user in acquiring knowledge about mechanisms and movement.

- The software should assist the user in designing components that provide the behaviors the user wants.
- The software should assist the user in combining components into mechanisms that preserve the individual behaviors of components and combine them into the desired behavior for any given automaton.

To this list is added the task of providing persistent storage of component and mechanism designs; something that is not commonly done by automatists. These are the four tasks the MachineShop software supports (Figure 5.1).

Figure 5.1: When the MachineShop software is launched the user is asked to select which of the four core tasks she wants to undertake.

The types of components seen during the survey presented some surprises. Even with the unique nature of the automata examined and the differences in the complexity of their mech-

anisms, cams, levers and cranks, and gears comprise the overwhelming majority of mechanical components and are often found combined into more complex assemblies. The MachineShop software needs to support the creation of a wide range of these basic components and allow for the future addition of new component types.

5.1.2 Exploring Movement

To help users understand how mechanisms work the MachineShop software includes a *movement explorer*. Figure 5.2 shows the interface to this tool.

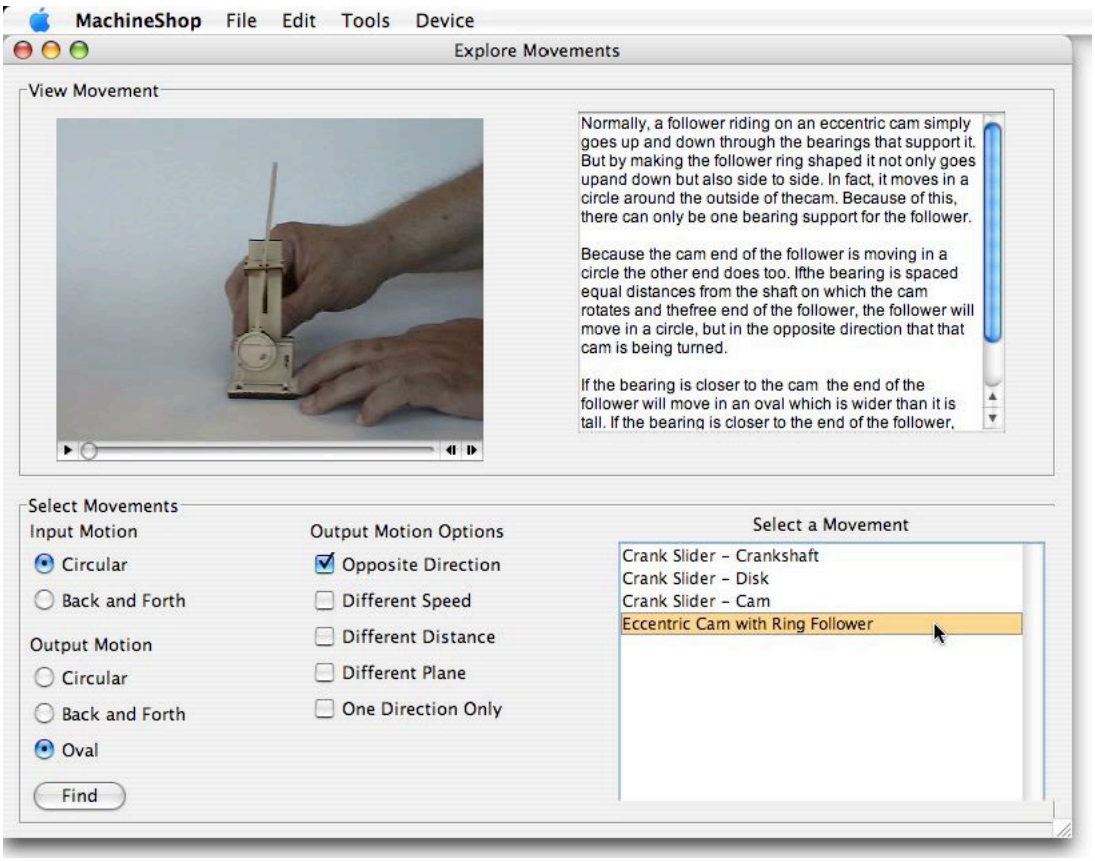

Figure 5.2: To help users understand the behaviors of the components which can be designed and built in MachineShop, the movement explorer provides examples and explanations.

Most children today have no knowledge about how machines work in contrast to their counterparts of half a century ago. As recently as the mid-twentieth century many common devices

used mechanisms that could be easily examined, but the move over the last three or four decades has been to sealed units with “no user serviceable parts” inside. Simple mechanisms (with the exception of bicycles) are no longer common to children’s experience, leaving them with no base knowledge about machines on which to build. To successfully build automata this situation needs remediation.

One approach is to show children examples of automata. This is a worthwhile approach, and has value for deepening their understanding of what automata are and can be (see Section 6.1.1). But it can be hard for children to understand how the individual components of a mechanism function together, or which piece or pieces of an automaton’s motion a particular component contributes to when presented with a finished machine. A better approach is to provide children with examples of simple mechanisms that are composed of only one or two components and allow them to experience these movements in isolation. Once the behavior of individual components is understood, combining them into new mechanisms and decomposing existing mechanisms becomes an easier task. Not every child who works with MachineShop will have access to examples of this kind and even those who do will be unlikely to have examples of many of the mechanisms that can be made with the tool. To mitigate this situation, the software incorporates a tool for this purpose called the *movement explorer*. Figure 5.2 shows the interface to this tool.

The window is divided into four sections. Initially, only the lower left section is available for the user to interact with. These controls allow the user to think about the kinds of motions in which she is interested without requiring any knowledge of the behaviors of the components that might provide it. She needs to think about the kind of input motion she wants (rotational or translational), the general type of output motion she desires, and any special properties she wishes the output motion to have. For instance she may want circular input and output motions, but to have the output rotate in the opposite direction of the input or at a different speed. Once she has made her choices, she clicks the **Find** button and is presented with a list of movements in the lower right of the window that meet her criteria. If none exist, a dialog box appears to inform

her that there are no example movements currently in the database that match her choices.

Any entry she chooses from this list has two corresponding elements that appear in the upper section of the window. A short video of the movement appears on the left while a textual description of the movement appears on the right. The user can watch the video clip while comparing what she sees to the description given for the movement and can move back and forth between them as she finds necessary.

5.1.3 Component Editors

Once a user has determined which components are required for a particular mechanism, she needs to design them. This task is more than just making a component of a particular type and requires that the user accurately choose the parameters that will define the operation, behavior, and suitability of each component so that the desired mechanism behavior is achieved. To assist in this process, MachineShop provides an editor for each family of components that the software can create: cams, gears, and levers. Within each of these editors the user has the ability to either precisely define a component's parameters in a "bottom up" design process or to define a component's behavior in a "top down" design. These design styles can be used interchangeably as it seems most natural in any given circumstance.

Each editor is presented using a standard layout (see Figure 5.5 for one example). The window is divided into three areas and the functionality of each is repeated from editor to editor. In the upper section of the window are the representations of the behavior and motions of the component or component collection being designed. Below that are the controls needed to define basic parameters for the components and to specify the materials from which they will be fabricated. At the right side of the window is a visual representation of the components as they are currently defined that can be animated so that their movement can be observed.

To keep the user interface consistent across editors, a number of controls in the lower panel are repeated in each editor. Choosing the diameter of the shaft or shafts for components as well as picking a material and thickness from which to make the components are shared. The row

of buttons below these controls are also shared and allow the user to load a previously saved file into the editor, save a file to the library for reuse or sharing, create a fabrication file for use with a computer controlled output device (see Section 5.2), or to send the component's profile to a laser or inkjet printer.

5.1.3.1 Cams

Cams are one of the most intriguing of the components found in contemporary automata. At their most basic, they turn rotary motion into translational or oscillating motions of the devices that move along their outer edges. These devices, known as cam followers or simply followers, can either be the terminus for the cam's motion or can be used to transfer the cam's motion to other components[1] .

A lobe has three primary features: lift, dwell, and shape. Lift is simply the change in distance between the cam center and the minimum and maximum distances on the perimeter for the lobe. Dwell, sometimes called duration, is the rotational angle through which the lobe acts on the follower. Shape is the profile of the perimeter of the cam and varies from one type to another. Both lift and dwell are user defined parameters in Machineshop unlike shape which the software produces as a function of the cam type. A cam can have more than one lobe and the follower will move through a complete cycle once for each lobe on the cam. Figure 5.3 shows the lift and dwell of one lobe of a five lobed snail cam. Note that in this example the dwell and lift of each lobe are not the same.

Cams come in a variety of types and supply a corresponding variety of motions and behaviors to their mechanisms. Surveying existing contemporary automata showed that the cams most commonly used fall broadly into two types; eccentric cams and snail cams (see Figure 5.4. Eccentric cams have a circular profile but have their shaft offset from the disk center. This provides a smooth and regular movement to the follower effectively creating a single lobe with a lift of twice

[1] Strictly speaking, a cam is protuberance on the edge of a circular disk. MachineShop uses the more common term lobe for these deviations from the circular, and refers to the composition of lobe and disk as a cam.

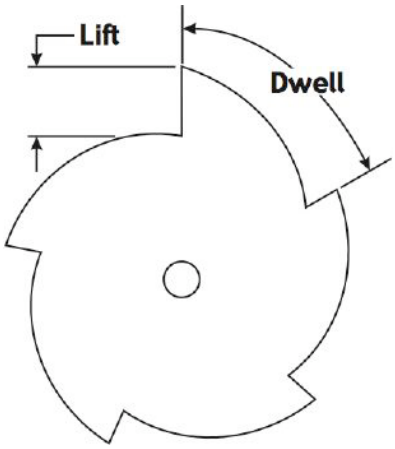

Figure 5.3: The lift of a lobe is the distance the cam follower moves relative to the rotational axis of the cam. The angular period over which this change takes place is the dwell.

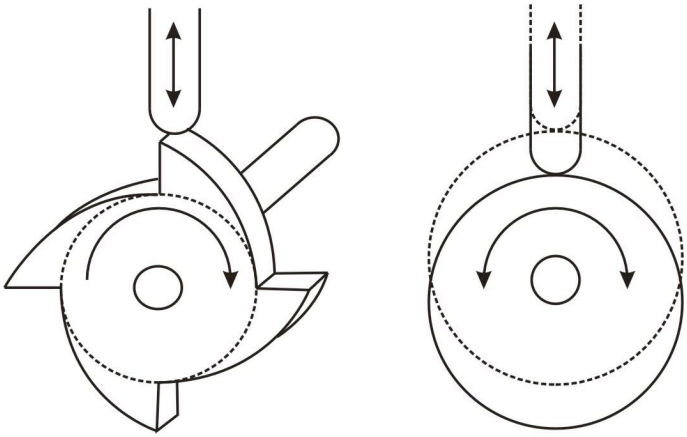

Figure 5.4: The snail cam (left) lifts its follower gently before letting it return rapidly to its starting position. The eccentric cam (right) moves its follower up and down with a smooth motion. Note that the snail cam can turn in one direction only while the eccentric cam can turn in either direction.

the shaft offset and a duration of 360 degrees. Snail cams (named for the snail shell like profile of the single-lobed instance) give their followers a gradual rise in lift followed by a rapid return to their original location. In contemporary automata mechanisms cams are commonly used to create a back-and-forth or up-and-down motion in one or more of the tableau elements such as the camel's head, hump, and tail in Spooner's *Camel Simulator* (Figure 4.7).

Figure 5.5 shows the interface for the cam editor during the process of designing a three-

lobed snail cam. In the upper panel, the user is presented with a motion profile that the cam follower will follow for the current cam. She may alter this profile by using the mouse to move any of the squares at the inflection points of the graph. This allows her to design top down, creating the path the follower must take while the software creates a cam profile that will deliver that motion.

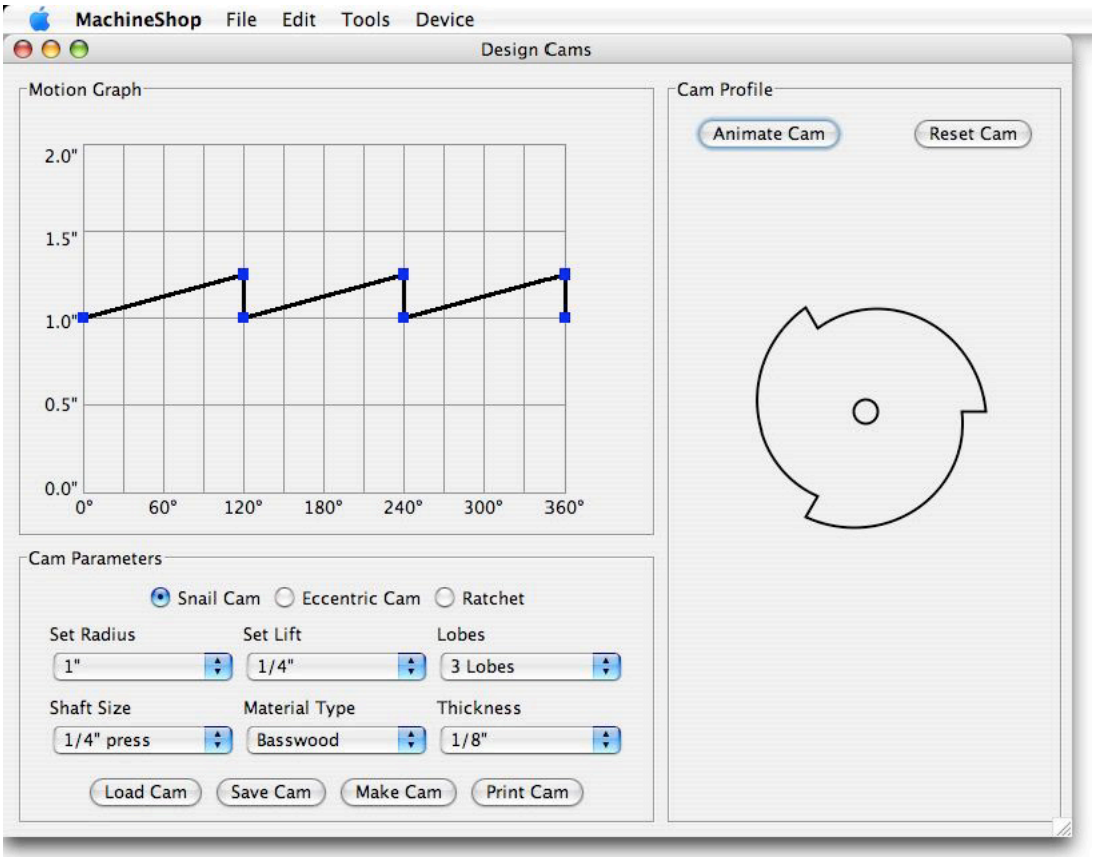

Figure 5.5: The cam editor presents the user with controls to set the parameters of the cam (lower left), a graphical display of the cam follower's motion that can be changed by dragging with the mouse (upper left), and the profile of the current cam (right). The cam profile can be animated to simulate the motion of the finished cam.

The lower panel contains controls that allow the user to set parameters directly to define the cam. Here the user can choose the base radius of the cam, the number of lobes (except in the case of eccentric cams) and the default lift for all lobes. Once these are chosen, the user may still independently modify the lift and dwell of each lobe in the upper panel.

The right hand panel shows the cam profile that will provide the follower motion shown in the graph in the upper panel. This profile cannot be directly manipulated by the user, but responds in real time to changes to the graph. At any time while the user is changing the graph, the current cam profile and a gray representation of the changes being applied to the graph are shown in this panel so that the user may see how her changes are affecting the cam's profile. When she finishes modifying the graph (by releasing the mouse button) the new profile is displayed. At any time the user may also animate the cam profile causing it to turn so that she can see the motion that the current profile will produce.

In addition to designing snail cams, the user may also design eccentric cams and ratchets. Eccentric cams (Figure C.7) are circular disks with their shaft holes offset from the disk center. The effective lift of an eccentric cam is therefore twice this offset (once above and once below the center) and the dwell is 360 degrees. Eccentric cams can have two different types of followers, each of which produces a different final motion. By using a rod follower as shown in Figure 5.4, an eccentric cam produces a smooth back and forth motion at the other end of the follower. By using a ring follower as shown in Figure C.7, an eccentric cam can be used as a crank-slider that produces an elliptical motion, opposite in direction to the rotation of the cam, at the far end of the follower. The software has the option of producing an appropriately sized ring follower for any eccentric cam the user may create.

While not technically a cam, MachineShop also allows users to create ratchets which share many physical similarities with snail cams and are included with cams for that reason. Ratchets are most commonly used to constrain rotation to one direction only or to supply intermittent linear motion from rotational motion. The interface for ratchet design is shown in Figure C.8.

5.1.3.2 Gears

Gears transfer rotary motion from one location to another. In the process, it is possible to change the speed, force and direction of rotation by the appropriate choice of the number of and ratios between the gears. A gear set with an even number of gear axes (since multiple gears can

exist on the same axis) will have the output (driven) gear rotate in the opposite direction to the input (drive) gear. A gear set with an odd number of gear axes will have the output gear rotate in the same direction as the input gear. The ratio between the sizes of the input and output gears in any gear pair determines both the change in rotational speed and the change in force that will occur in that pair. When the input gear is smaller than the output gear, the input gear will rotate more than one complete revolution for every complete revolution of the output gear. At the same time, the effort applied to the input gear will be amplified at the output gear. Both of these occur as functions of the gear ratio of the pair.

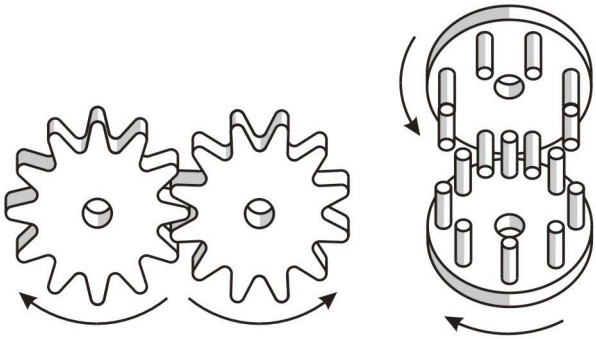

Figure 5.6: Spur gears (left) keep the axes of rotation parallel. Pinwheel gears (right) allow the axes of rotation to be 90 degrees apart. Note that the rotation direction of meshed gears in both types are in opposite directions.

Because the teeth of one gear mesh with the teeth of another, this rotational transfer is accomplished without slippage. Frictional losses of effort are an important concern in gear design and over time have been addressed primarily in the areas of tooth profile and angle and mesh properties between gears. To maximize the efficiency of gears, designers in the real world must deal with a large number of parameters [2] when creating gears [11, 29].

Like cams, gears come in a wide variety of styles to suit the numerous functions for which they are used. In the contemporary automata surveyed, the gears most commonly found were of the spur and pinwheel varieties (Figure 5.6). Spur gears keep the rotation axes parallel but allow

[2] No less than 58 parameters are defined as necessary for designing gears in [6].

them to be separated by some distance. Pinwheel gears move the rotational axis of the output gear 90 degrees from that of the input gear.

MachineShop allows the user to create gear sets composed of either two (pinwheel and spur gears) or three gears (spur gears only), in various ratios. It is of interest to note that gears are the one component currently implemented in MachineShop that exist only in sets. Since rotary motion is transferred from one gear to another, it makes no sense to think of a single gear in the context of a mechanism. The gears in MachineShop have been simplified such that there are only three user specified parameters: the number of gears in the gear set, the spacing between the input and output shafts (for spur gears) or the size of the larger gear (for spur gears), and the ratio between the input gear's size and the output gear's size. There are, of course, other parameters that must be known in order to create gears in MachineShop; diametral pitch and tooth profile for example. These values are currently established by the software and are not directly available to the users.

Figure 5.7 shows the interface for the gear editor during the process of designing a gear set of three spur gears. In the upper panel, the user is shown a representation of the gear set presenting its important properties. The relative sizes of the gears are shown, each with an arrow. The direction of rotation of a gear is indicated by the direction of its arrow, while the relative lengths of the arrows indicate the amount each gear moves relative to the amount its neighbor moves. The thickness of the arrow indicates relative force or effort in the same fashion. Unlike this panel in the cam (Section 5.1.3.1) and lever (Section 5.1.3.3) editors, the user cannot manipulate the gears directly. This decision was made because gears require an integer number of teeth. This limits the gear sizes available to a discrete number rather than the virtually unlimited sizes available to the other components. While the user could have been permitted to change the size of individual gears in one tooth increments with the resultant non-integer ratios, it was decided that most of what a user would want to do could be done within these constraints.

The lower panel contains controls that allow the user to set the parameters that define the properties of the gear set. Here the user can choose the number of gears that she wants (currently

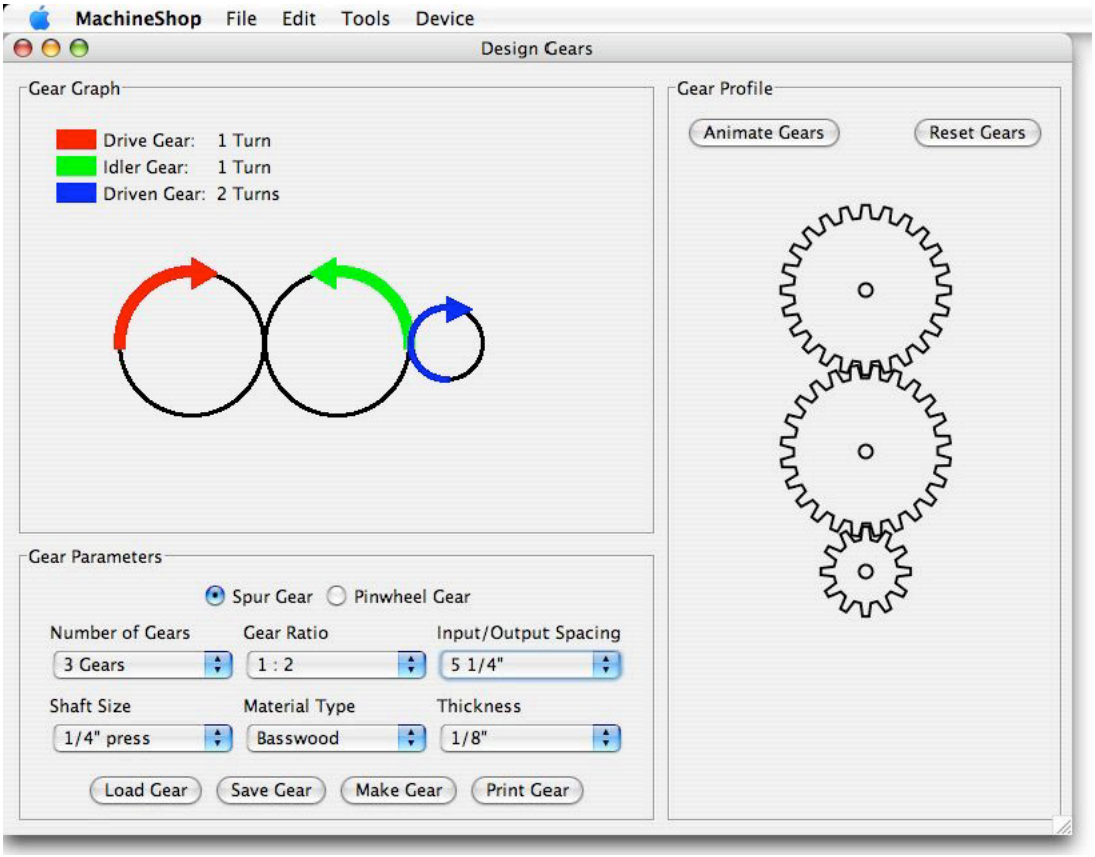

Figure 5.7: The gear editor presents the user with controls to set the parameters of the gear set (lower left), a symbolic representation of the direction, speed, and force delivered by each gear in the set (upper left), and the profiles of each gear in the current gear set (right). The gear profiles can be animated to simulate the motion of the finished gears.

only spur gear sets can contain 3 gears), the ratio between the input and output gears with respect to their sizes, speeds, and power, as well as the distance between the input and output shafts (spur gears) and the sizes of the gears (pinwheel gears). The decision to limit the number of gears in a set was deliberate. By choosing to provide only two or three gears in a set, MachineShop avoids presenting a confusing number of options to the novice user while still supporting more advanced users who can create more complex gear arrangements by combining smaller sets.

The right hand panel shows the profiles of the gears that compose the current gear set. The input gear is shown at the top with the output gear at the bottom. As the user changes parameters for the gear set with the controls, the profile is updated to reflect these changes. The user can also

animate the gear profiles to see what the motion of the gear set will be. The display of pinwheel gear profiles presented a special challenge since the faces of the gears are not located in the same plane. The profile display presents the gears in their actual orientation showing one gear face on and the other edge on (see Figure C.10).

5.1.3.3 Levers

Levers are arguably the oldest machines known to man and are one member of the set of devices known as simple machines [98]. The earliest purposes to which levers were put was in the amplification of forces in order to move heavy loads. In order to do this, the lever must have a fulcrum, or pivot point, upon which it can move. Levers are divided into to three classes or orders based on the relationships between the location of the force applied, the location of the work done, and the location of the fulcrum (see Figure 5.8).

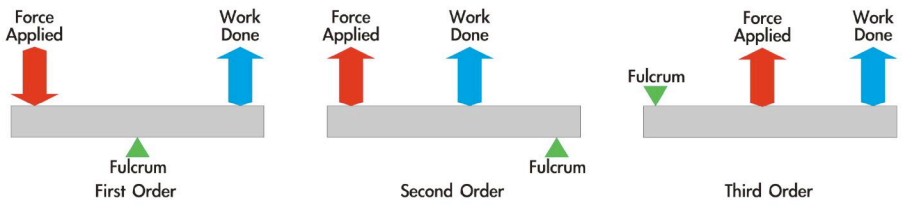

Figure 5.8: Levers are categorized according to the locations of their applied force, work done, and fulcrum. Examples of each order are a see-saw (first order), a wheelbarrow (second order), and a fishing rod (third order).

A lever of the first order has its fulcrum located between the location of the applied force and the location of the work done. A see-saw is an example of a lever of the first order. A lever of the second order has its location of the work done located between the location of the applied force and the fulcrum. A wheelbarrow is an example of a lever of the second order. A lever of the third order has the location of its applied force located between the location of the work done and the fulcrum. A fishing rod is an example of a lever of the third order.

Levers serve a wide variety of functions in the mechanisms of contemporary automata and are seen more frequently than either cams or gears. Levers are used to transfer linear motion from

one location to another, reversing its direction in the process. Many times this transfer of motion results in a change of distance travelled between the input and the output. Levers are used as cam followers, as pawls to constrain the rotation of ratchets, and as cranks that act as the interface between the viewer's hand and the mechanism.

Figure 5.9 shows the interface for the lever editor during the process of designing a lever of the first order. In the upper panel, the user is presented with a schematic representation of the lever along with its force and effort. She may relocate the locations where the force is applied, the work is done, or the lever pivots on the fulcrum by dragging any of the colored squares with the mouse. She can also choose to design a lever of another order by selecting the appropriate button in this panel.

The lower panel contains controls that allow the user to set parameters directly to define the lever. The user can choose a length and width for the lever and set the location of the fulcrum. The user can move back and forth between these controls and direct manipulation of the lever in the upper panel to create any lever she may desire. This panel also contains the controls that allow the user to select which type of component she wants to design.

The right hand panel shows the profile of the lever that matches the settings in the upper and lower panels. The profile in this panel cannot be directly manipulated but responds in real time to changes made in the upper panel. At any time that the user is changing the force, effort, or fulcrum locations or modifying the length of the lever in the upper panel, a gray profile representing the changes being applied appears over the current profile so that the user can determine how the changes she is making will affect the shape of the lever. When she releases the mouse button this new profile replaces the existing profile in the panel. At any time she may also animate the lever to see how it will move when used in a mechanism.

In addition to levers, this editor also allows users to design and create cranks and pawls (see Figure C.13). Cranks are levers with the location of either their applied force or returned effort coincidental with the location of their fulcrum. As commonly used in contemporary automata, the user applies force to the crank at some distance from the shaft that acts as both fulcrum and

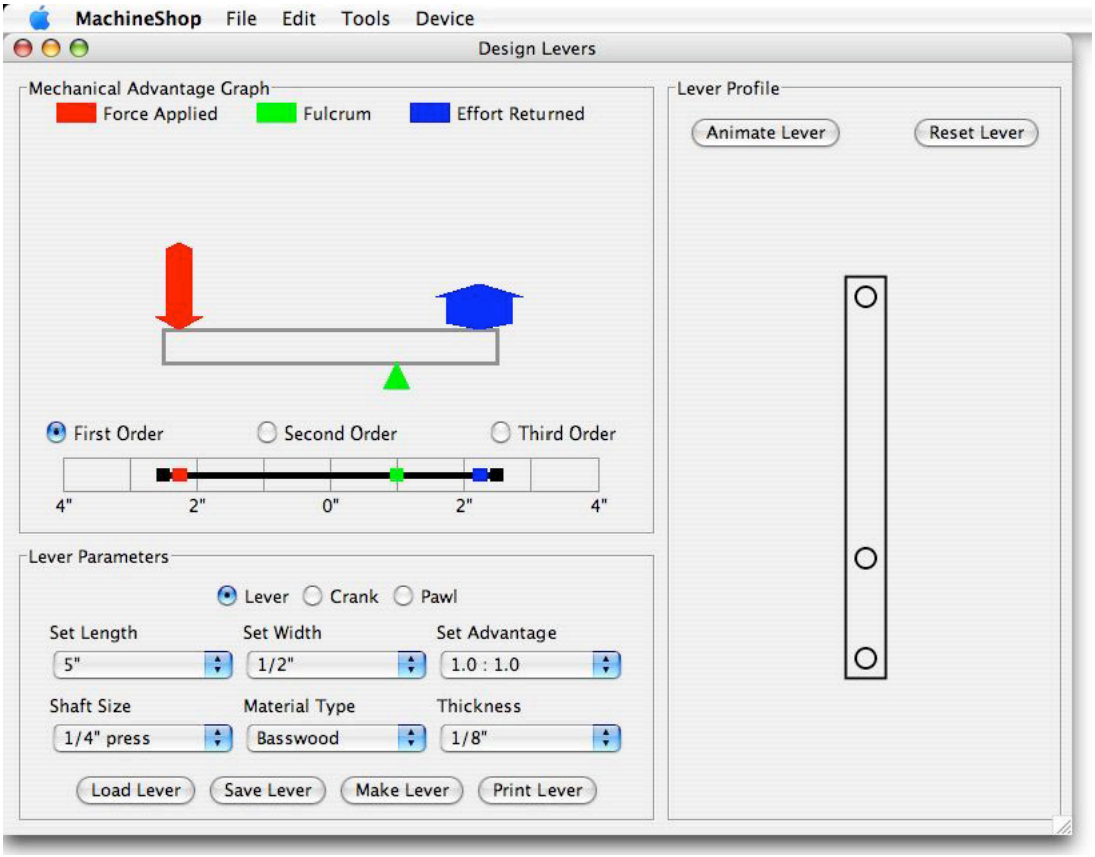

Figure 5.9: The lever editor presents the user with controls to set the parameters of the lever (lower left), a symbolic representation of the lever that can be changed by dragging with the mouse (upper left), and the profile of the current lever (right). The lever profile can be animated to simulate the motion of the finished lever.

source of the effort applied to the rest of the mechanism. Pawls are physically similar to cranks, but are used with ratchets (Section 5.1.3.1) to constrain rotary motion to one direction only.

5.1.4 Browsing Libraries of Components and Mechanisms

While the component editing tools in MachineShop allow users to quickly and accurately design the mechanical components that they require, doing so for each and every automaton that the user builds is a waste of effort. Additionally, it is desirable for users to be able to share their designs with each other so that either duplicate or derivative copies can be easily made. MachineShop has the ability to store and retrieve both the mechanisms that a user creates as well

as the individual components that comprise those mechanisms. Browsers allow users to visually examine each entry in the libraries, delete library entries, or load them into the appropriate editor. Users add components to the component library by using the **Save** button in that component's editor. Users can also load components from the library by using the **Load** button in the editor, bypassing the browser when the component's name is known.

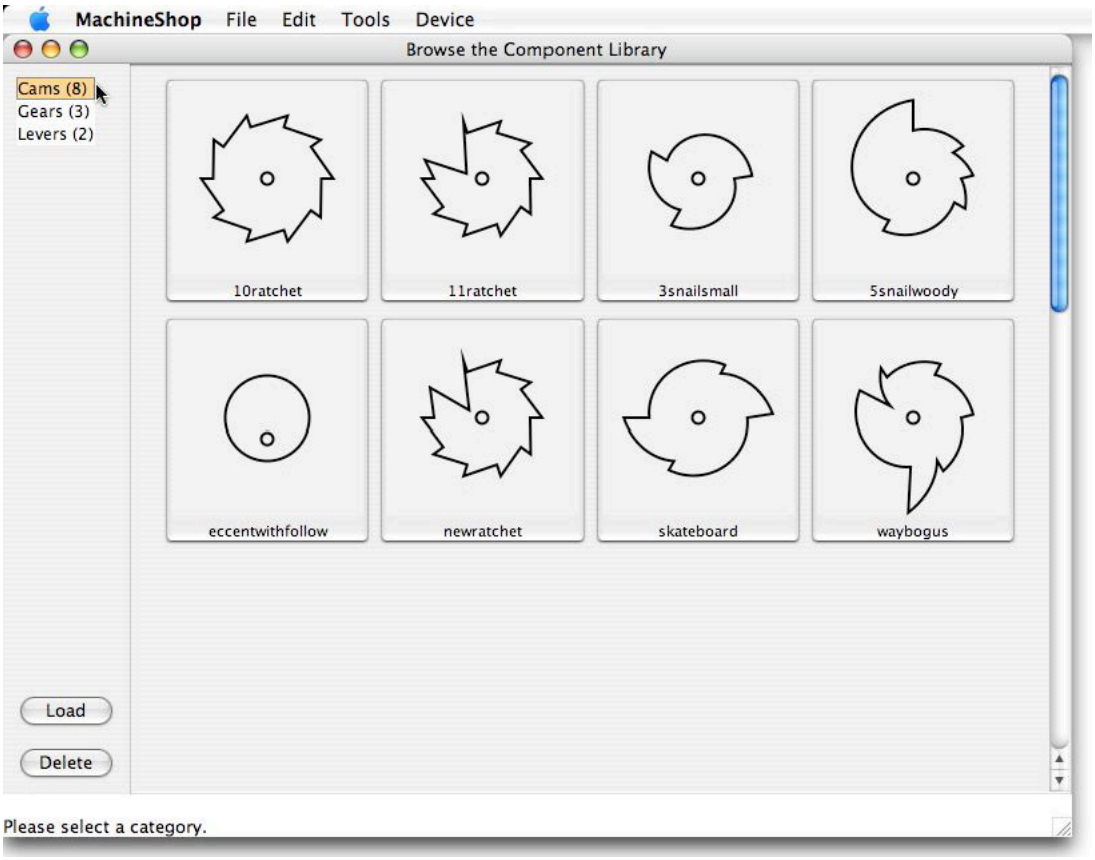

Figure 5.10: The component browser allows the user to quickly view all components saved in the library. By selecting from the component types (upper left) she is presented with images of the profile of all components of that type. If she wishes, she can select a component from this window and delete it, or she can load the component which will start the editor with the component ready to edit or fabricate.

Figure 5.10 shows the interface for the component browser while looking at previously created and saved cams. The left panel contains a list of the types of components stored in the library; currently cams, gears, and levers. When additional component editors are added to the

software they will have corresponding entries in this panel. The right hand panel contains small images of each component of the currently selected type that exist in the library. The user selects a component from this collection by clicking on it with the mouse. Once selected, she can choose to load the component into the appropriate editor or to delete it permanently from the library using the buttons in the left panel. She can, at any time, choose another component of the current type or choose to view components of another type.

Like components, mechanisms can be saved to and loaded from the library in the assembly tool (Section 5.1.5) or selected using the browser. Mechanisms are saved as assemblies of components with each component's description included in the mechanism's file. This method simplifies the reading and writing of mechanism files and allows users to not only learn about the mechanism, but also about its constituent components by reading the file.

5.1.5 Designing Mechanisms

The mechanisms that can be designed and created in MachineShop are more than just collections of components. Every mechanism also includes the shafts and bearing surfaces on and in which the components move, the support structures that hold the components in their relative positions and maintain alignment between them, and often other structures or pieces as well. Having a virtual space in which to fit components to create mechanisms can help users understand and visualize both the relationships between components and the relationship of a component to the mechanism and how those relationships affect the proper functioning of the mechanism. Designing a tool of this type for children to use is especially challenging. Conceptually, designing a case to hold mechanical components exposes many more variables than does designing a component. Where at most a handful of components can provide some desired mechanical behavior, an almost limitless variety of structural frameworks can be created to support those components. The software designer must proceed carefully when constraining those choices to avoid severely limiting users' creativity.

These choices affect what can be made and how it can be made in a variety of ways. For ex-

ample, looking at a variety of contemporary automata shows that most have structural frameworks which are some variation on a cube[3] . By restricting frameworks in MachineShop to be similarly constructed, the software can be greatly simplified and the user can be gently guided in making her mechanisms in empirically proven styles. Little seems to be lost by not allowing spherical or octahedral structures.

The structural framework of a mechanism exists primarily to allow the moving components of the mechanism to perform as the designer intended. But the outer framework is seldom sufficient to accomplish this task and very often must be augmented with additional structural members whose purpose is to allow free movement of components and to keep the components properly aligned, oriented, and located. These pieces are in turn aligned, oriented, and located by the structural framework of the mechanism. Intrinsic in the proper operation of the moving components are a number of mechanical requirements beyond the functional characteristics that the designer has selected.

Shafts with or about which components move need to be supported in at least two places and the greater the distance between these supports the greater will be the stability of the component. Holes in which shafts fit or turn must be appropriately sized. Cam followers need to be supported in two places and their performance is also affected by the distance between those supports. Components which operate in sets, like gears, need to maintain an optimum spacing regardless of their orientation within the mechanism. While important, much of this should not be forced on the user and can be dealt with by the software without the user needing to know about it. This allows the user the opportunity to acquire this knowledge while building mechanisms rather than needing the knowledge before mechanisms can be built.

The mechanism assembly tool has been the most complicated of all of the tools in the MachineShop software. It has been written, discarded, rewritten, and discarded numerous times. It seemed that every time a strategy had been devised for its creation, results from the user test-

[3] Few, if any, contemporary automata have frameworks that are truly cubes. Most are stretched or shrunk in some dimension and often one or more sides are missing or abbreviated to provide a view into the interior of the mechanism.

ing would reveal serious flaws with whatever approach was being used at the time and a new incarnation would need to be started. The incomplete version that exists at the time of this writing incorporates the lessons learned from the user tests (Section 6.2.5) as well as guidance from current cognitive theory (Section 2.3) and practices gleaned from the work of designers of contemporary automata. The issues that have arisen with respect to the design of the assembly tool that need to be addressed fall into two categories: conceptual and implementational.

The conceptual issues reflect concerns of overwhelming users with details of the domain and determining how best the software can take on less central, but still necessary, tasks to make the design of mechanisms less complicated. Foremost among these issues are:

- How much of the knowledge required to design and construct properly functioning mechanisms should a user need to know to successfully work in the domain? Will the user have opportunities to expand her understanding of the full range of knowledge?

- At what level or levels should the software support the assembly task? What advantages or disadvantages would the user have if static displays of the assembly were replaced with animated displays? Would anything be gained from simulating and modeling the physics of the operation of a mechanism?

- How much freedom in manipulating pieces in the virtual assembly should a user have? Should the distances a piece can be moved or the locations a piece can be placed be limited in some way or should the user have total freedom? Should pieces that are part of a logical group be constrained to move together when appropriate or should each piece be allowed to move independently?

The implementational issues concern making such a potentially complex tool available to the intended user audience. Among these issues are

- Since the mechanism assembly tool is one of a suite of tools, there should be as much consistency in the choice and placement of controls and information displays between

this and other tools as possible.

- Views of the mechanism should be displayed in appropriate forms. This may mean 2-dimensional views, 3-dimensional views, or some combination of the two.

- Manipulation of the components and structural elements in these views should be as simple and easy to learn as possible. It should be difficult to inadvertently rotate an object when attempting to move it. It should be easy to select and manipulate the desired object without interference from neighboring objects.

With respect to the conceptual issues it makes sense to insulate the user from some of the finer details of mechanism design. Other tools in the software already do this; the gear editor determines many of the important parameters transparently to the user for example. As previously mentioned, the sizes, features, and numbers of the support elements for components are very important, but asking the user to worry about these additional pieces would likely confound the design process by doubling or tripling the number of pieces the user would be responsible for creating correctly. MachineShop will add file specific data for default support elements when the user saves a component to the library or makes a fabrication file. Some components (cranks or pawls for example) will not have their own support pieces but will rely on other components' pieces for support.

Other issues can and will be addressed in the mechanism assembly tool interface shown in Figure 5.11. The window is divided into four panels, one more than in the component editors but with a similar logical grouping. While the implementation has only just begun, the following describes the intended functionality. The upper left panel contains a two-dimensional orthographic view of the mechanism in its current state. The user selects the view she wants from the six possible views (front, back, top, bottom, right, or left) using the pull down menu at the top of the panel. This panel is where the work of arranging and aligning the pieces of the mechanism is done using the mouse to select and move them. By providing a single view of the mechanism the user is not given the task of integrating multiple views [30]. Further, by restricting all of the ma-

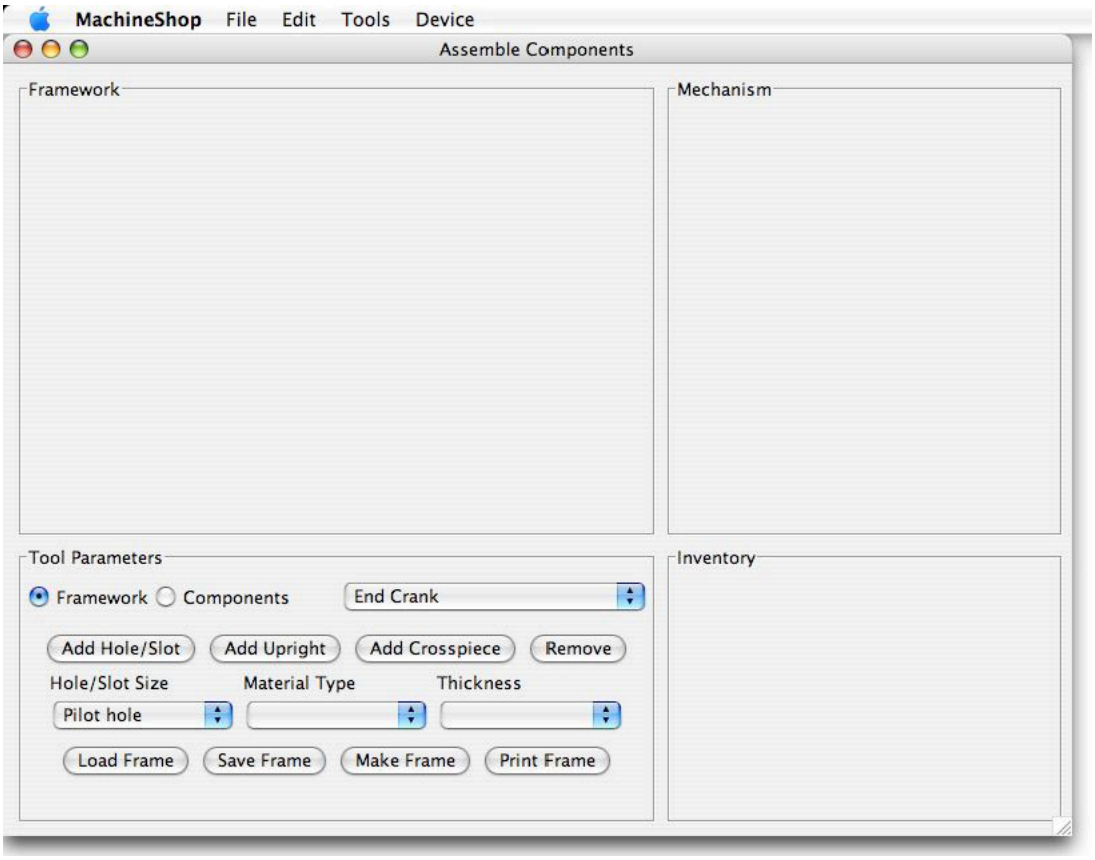

Figure 5.11: The mechanism assembly tool presents the user with controls to work with both components and structural elements of the mechanism (lower left), a user selectable view from any of the orthographic projections for the mechanism where components and structural elements can be placed with the mouse (upper left), an oblique (3-D) view of the mechanism (upper right), and an inventory of components for the mechanism (lower right). This figure shows the current, incomplete state of the tool with provisions made for adding the necessary functionality.

nipulation of the mechanism to a two-dimensional view, any ambiguity as to direction or distance is significantly reduced [36]. The upper right panel contains a three-dimensional oblique view of the mechanism as it is currently configured. Changes made in the two-dimensional view will be reflected here as they are made. The initial implementation of this panel will not provide the ability to animate the mechanism as can be done with components in the editors (Section 5.1.3).

The lower left panel contains the controls that allow the user to choose which category of pieces to work with and to set or change the properties of those pieces. When the **Framework** button is selected, the user can choose between framework types for this mechanism. She can

also choose to add a new support element to the framework, remove an element selected in the upper view, or change the orientation of the selected element in the framework. She can select the material type and thickness for any element and she can set or change the size of any existing holes or slots or add new holes or slots to the selected element. When the **Components** button is selected, she has a similar set of controls for adding, deleting, and positioning components. By separating the tasks of working with the structural members from working with the components, the number of controls immediately available to the user is reduced and each task becomes more structured. The lower right hand panel contains an itemized listing of the components included in the mechanism and corresponds to the components that will be included in the mechanism file when the current assembly is saved to the library. Adding a component to this inventory puts its profile in a default location of the two-dimensional view, ready for placement. Selecting a component in the list allows it to be removed from the assembly or loaded into its editor for manipulation.

Choosing between the component and framework categories not only changes the set of controls that the user has access to, it also affects the two-dimensional mechanism display. Pieces which belong to the selected category are displayed as solid objects while pieces of the other category are shown as wire frame outlines to make it more obvious which task the user is engaged in. But even with these differentiations there are still dependencies between the components and the structure. For example, since components and their supports share shafts, moving a component in a direction perpendicular to the axis of its shaft will cause its supports to also move. Moving a component along its shaft will force any supports that it contacts to move as well.

Little is known about how a tool such as this will be used or what effect it will have on users' understanding of the domain or mechanical reasoning. Children designing automata with MachineShop seem to have fundamentally different needs than mechanical engineers designing water treatment plants with AutoCAD. It seems prudent, therefore, to simplify as much as is reasonable until data are available to guide further development. For this reason the three-dimensional view will not allow for the mechanism to be animated, movement and placement of mechanism pieces

will be constrained to a grid, and any necessary spatial relationships between components and supports will be enforced. Further testing will indicate the efficacy of these choices.

5.1.6 File Types

Among the early design constraints for the MachineShop software were two that dealt with what appropriate file types might best support the goals established for the system. First, the software should make the design of components as transparent as possible to the user. This meant that whatever file encoding method was used by the component editors and libraries, it had to be human readable so that users could not only make sense of a component's file by looking at it but that these files could be easily exchanged. Second, the software would use one or more machine tools (section 5.2.1) to fabricate the mechanical components that the users would design. This meant that the software needed to be able to convert the profiles and parameters for components into forms that these machines could use.

To address the first constraint it was decided that files for the editors and libraries needed to be in some text format, using words and numbers that would make sense to the users. Because the intended users were children, the initial desire was files that could be read almost like text. This proved impossible as the parsers and generators necessary were not easily implemented and would not scale well. Pursuing this path would have made adding new component types and extensions to the software that relied on reading these files a potentially arduous task. A compromise position was to store these files in Extensible Markup Language (XML) format [119]. XML has become a standard encoding format for many large software companies over the last few years and a wide variety of tools are available for reading and working with files in the XML format. While XML files are not prose, they can be encoded in text and the ability to create meaningful tags goes a long way toward simplifying the reading and understanding of these files. For each component type tags were created that represent each property that the editor uses to define the component. These tags are then assigned the values given those properties by the editor.

Dealing with computer controlled output devices was a trickier proposition. These ma-

chines vary widely in the file formats that they will accept and it was necessary to consider carefully the types of machine tools that might be available to the intended target audience (see Section 5.2 for a discussion of this point) and the corresponding file encodings that they would accept. Many of these devices expect input in some vector format so the software was designed in such a way that component profiles could be easily saved as vectors. The first device that MachineShop was used with was a carbon-dioxide (CO_2) laser cutter that took its input from files encoded in the Hewlett-Packard Graphics Language (HPGL). Developed by Hewlett-Packard as the printer control language for their family of pen plotters, vector based HPGL became the standard control language for most plotters made by all manufacturers. Many commercial CAD and graphics programs can both write and read HPGL making it an appropriate format for many different tools.

5.2 Output Devices

While the software is a major constituent of the MachineShop system, much of the system's ability to bring the complicated undertaking of automata design and construction within a child's capabilities lies in the ways in which it supports the fabrication of mechanical components after they have been designed. Throughout the history of automata, the tools used to make the pieces needed have been used by skilled practitioners. These skills were often the result of years spent learning the craft as an apprentice. Even when this knowledge was codified in books and schools were established to train workers in these areas, true mastery was still the result of years of practice. While MachineShop can support users as they learn to use hand and power tools during the fabrication of automata, one of its major strengths lies in its capacity to minimize the necessity of mastering tools and materials while still producing high quality automata. This is due in large part to the abundance of machine tools which can be controlled programatically by computers.

5.2.1 The Landscape of Computer Controlled Machine Tools

Controlling machinery programatically is nothing new. The Jacquard loom of the early nineteenth century [34] and the census tabulating machines developed by Herman Hollerith [3]

both used punch cards to control the operation of the machines. Punched paper tape was used in Alexander Bain's automatic telegraph (later perfected by Thomas Edison) [2] which in turn spawned the teletype. Computer Numeric Control (CNC) was developed at the MIT Servomechanisms Laboratory in the late 1940s [130] and also used paper tape to transfer programs to machine tools such as lathes and mills. The holes in the tape controlled the actions of servo-motors which in turn controlled all aspects of the machining from tool selection to material or tool speed to tool depth. This advance allowed repeatable fabrication of objects with more precision than was possible any other way. Adapting a machine to use CNC was costly however, and even after manufacturers began selling tools already equipped for CNC, these machines were found only in large production machine shops.

But the microcomputer revolution of the 1980s changed things. CNC machines no longer read paper tapes, but were controlled by a small computers reading files written in G-Code [73] from floppy diskettes. By the mid-1990s smaller lathes, mills, and other machines equipped to use CNC were becoming available to small shops and hobbyists. It was also around this time that a variety of new fabrication tools were developed. The following list, while not exhaustive, shows how computers have changed the face of manufacturing.

- Shaping tools. Lathes (which create cylindrical objects by cutting material from blocks of spinning metal) and mills (which can create objects of arbitrary shapes by applying spinning cutting tools to stationary material) were the earliest of the modern breed and, in most cases, continue to use programs written in G-Code just as they did half a century ago. Routers have joined these tools and are used for decorative embellishment of wooden products like furniture and for tasks such as prototyping printed circuit boards.

- Cutting tools. For decades, repetitive production cutting of plate steel had been accomplished by first creating a reduced scale pattern of the shape to be cut and then using a pantograph with an acetylene cutting torch attached to create the full size pieces. CNC

allowed this crude form of automation to be significantly refined, eliminating both the time consuming tasks of creating the pattern and the tedious hand control of the pantograph. While these are still in use in many places, plasma torches are replacing acetylene where metals other than steel need to be cut (plasma torches can cut stainless steel, aluminum, titanium, and other metals) and where the smoother cuts produced by the plasma torch significantly reduce cleanup of the cuts. New technologies have also been applied to automated cutting in tools like high pressure water jet cutters and a variety of laser cutters. These new tools further expand the range of materials that can be precisely and repeatedly cut to wood, plastics, stone, and many others. Even cutting with knives has been automated in tools such as sign cutters and inkjet printer/cutters.

- Rapid prototyping tools. This relatively recent class of machines differs radically from those previously described. Rapid prototyping machines (sometimes called 3D printers) create objects by building them up from base materials in powder, string, or liquid forms rather than removing or cutting material from existing stock. These machines either deposit the material in the desired areas one thin layer at a time (deposition printing) or use lasers to solidify liquid in a tank (stereolithography).

The availability of these types of tools has profound implications for designers of software like MachineShop. The place where we stand today is reminiscent of where software designers were two decades ago when color printers were first becoming available. Those printers were expensive and there were few ways in which people could obtain some advantage from their use. But astute software developers started writing applications that could use color printers in ways that users found desirable. As an increasing number of people started using these applications, the demand for access to color printers rose. As supply increased, prices dropped. More people now had access to color printers and more software was written to take advantage of them. This spiral continues to this day and it is often the case that the purchase of a new computer will include a free or very low-priced color printer.

This same process is occurring with respect to computer controlled machine tools. A variety of expensive tools are currently available but the number of people who have applications for these tools is still small. MachineShop and other programs that take advantage of these tools can provide increased demand for these tools. Economies of scale will reduce prices and manufacturers will be able to add features to these lower cost machines, increasing their desirability. This is not speculation; it has been happening over the last decade. A laser cutter that cost $10,000 US six years ago can now be replaced by one with twice the power and three times the features for that same $10,000. While stereolithography machines still command prices of $250,000 US (and more), deposition printer prices have dropped to the point that several manufacturers offer machines for less than $20,000 US. These machines are still expensive, but they are quickly coming into the price range where a high school could buy one for use in shop classes or a community center could buy one for use in its after-school programs. Even hobbyists can acquire these devices with some carbon-dioxide laser cutters selling for as little as $5,000 US.

From the beginning, a key design consideration for MachineShop was that it would leverage as many members of this growing family of output devices as made sense. While the only tools that it has used so far have been laser cutters, the components that it creates can be fabricated by most of the tools listed here. The necessary classes to convert the software's internal representations to appropriate fabrication file encodings would not be difficult to create.

5.2.2 The Kern Laser

The first machine to be used with the MachineShop software was purchased from Kern Electronics. This device (Figure 5.12) had been used as a research test bed by the company and we were able to obtain it used for $10,000 US.

Despite its low power (10 watts), this was a laser cutter of industrial design. The laser sat on heavy steel framework with the horizontal beam being directed downwards by a 45° mirror to be focused on the workpiece by a lens. A jet of air supplied by an external air compressor was directed down on the workpiece, collinear with the laser beam. The laser itself was cooled by

Figure 5.12: The initial installation of the Kern CO_2 laser cutter in the lab. The laser is the black box at the top with the control cables and chilled water hoses exiting on the right. The air compressor that supplies air to the beam is located on the shelf below the laser and the water chiller sits on the floor beneath the compressor. The optics are attached to the left end of the laser and focus the beam down on the workpiece sitting on top of the moving table. The exhaust hose to the blower can be seen attached to the left of the moving table. The lead screw which moves the table in the X-axis can be seen on the table base plate while the Y-axis screw is visible to the right of the material support box. Not seen here is the computer which controls the table and the laser.

recirculating water from an external chiller. The workpiece was placed on a 12" x 24" aluminum box with a honeycombed top and vent through which smoke and cutting debris were drawn by an external fan. The box itself was supported on a 24" x 48" steel plate and was moved under the laser beam by two servo motors turning lead screws, one in the X-axis and one in the Y-axis. The servo motors were controlled by an EISA card inside the dedicated computer that also controlled the power level of the laser.

The laser came with proprietary software to control all aspects of the cutting process. This software could read fabrication files written in HPGL (Section 5.1.6) and in many ways the process of cutting with this laser was similar to using a pen plotter driven by HPGL. For example, the cutting table moved in two orthogonal directions as does the pen holder on a plotter. The

software took this one step further and defined eight *pens*, each of which corresponded to a power setting for the laser and a speed for the movement of the table. Pen plotters typically have carousels that hold eight pens and from which the plotter can retrieve and replace them during plotting.

This implementation of HPGL made creating plot files in the software an easy task. The Java 2D libraries use vectors to display component profiles and provide methods to convert those profiles to lists of points. These lists are converted to HPGL commands to move and locate the laser beam. It was also possible to establish a table of pen settings (power and speed values) and map material types and thicknesses to pen numbers. With the Kern laser the software could create fabrication files that set all of the necessary parameters for cutting without operator intervention.

But the Kern laser had its shortcomings as well. It was time consuming to set up and difficult to keep aligned. The exhaust fan, air compressor, and water chiller combined to make it very noisy to operate. The laser itself was a low-power unit which sometimes required multiple cutting passes to completely cut through thicker or harder materials. And, being used, it could no longer operate at its original ten watts which further exacerbated the problem. Despite all this, the laser helped guide the development of the software and produced many parts before finally succumbing to age and use. The cost of replacing the failed laser tube was more than the original cost of the system, making it prudent to explore options which had become available since the initial purchase.

5.2.3 The VersaLaser

The second output device used with the MachineShop software was a VersaLaser, model VL-300, manufactured by Universal Laser Systems (Figure 5.13) and was purchased after the Kern laser had failed. This design is one of the first intended for desktop use, and is targeted at small companies who make plaques and trophies. The laser is a 30 watt CO_2 unit and, in addition to etching and cutting wood, plastic, paper, cardboard, cloth and leather, it can etch on cylindrical objects with a rotary workpiece holder. The cost of this machine was $16,000 US but it is far more

capable than the machine it replaced.

Figure 5.13: The VersaLaser installed in the lab. The laser tube is mounted to the rear of the cabinet seen here and is air cooled. The moving optics are located under the clear safety lid. The tank at the left contains compressed nitrogen that is used to clear the kerf and reduce instances of flaming when cutting wood. The exhaust fan remains from the Kern laser and exhausts smoke and fumes into the lab's ventilation system. The computer that controls the laser is seen at right. Compare this installation to that in Figure 5.12 to see the kinds of changes that have occurred in these devices in only a 5 year period.

The VersaLaser uses an air cooled laser tube that eliminates the need for chilled water while providing three times the output power of the Kern unit. The workpiece support, the optics, and the laser are all contained in a single unit which occupies only about half the footprint of the Kern laser. Compressed nitrogen is used to clear the kerf and reduce the chances of flaming which can happen when cutting very thick or hard materials that require high power settings and slow speeds. The smoke and fumes are exhausted into the lab's ventilator using the fan that we had previously acquired. The computer which controls the laser sits to its right on the same desk.

Programatically, there are some differences between the Kern laser and the VersaLaser. Because the intended audience of the VersaLaser is not necessarily technical, or one that has a

background in manufacturing, the VersaLaser uses a printer driver similar to what would be used by any other printer attached to the computer. Rather than using a dedicated software application to control the laser as the Kern laser did, this printer driver can present the VersaLaser to many programs just as if it were a laser or inkjet printer. This adds a step to the fabrication process by requiring that the fabrication files be imported into an application on the controlling computer that can read the HPGL files and interface with the driver for the VersaLaser. Fortunately, a number of applications can do this and the one we are currently using is CorelDraw. The VersaLaser was used exclusively during the user tests (Chapter 6).

5.3 Ancillary Materials

The MachineShop software and the VersaLaser were the primary system components used during the user tests. While the test users were working one-on-one with a researcher comfortable with making contemporary automata, not every user of MachineShop will have that help when first starting out. To help support users, the MachineShop system includes a number of supplementary materials that introduce new users to the domain, guide them as they create their own automata, and help them explore the domain in order to expand their understanding. None of the ideas presented here are necessarily new, but by constructing these materials around and with MachineShop they become part of a consistent whole and users are encouraged to move freely among all of the components of the system.

5.3.1 Example Automata and Mechanisms

Providing examples of automata and simple mechanisms is one way in which MachineShop allows users to learn about the mechanical principles found in contemporary automata. During the testing presented in Chapter 6, users had to a small number of automata that had been created by adult users of the software in its early forms. In addition, nearly a dozen simple mechanisms that had been created and video-taped for use as Quicktime movies in the movement explorer (Section 5.1.2) were on hand at every testing session. These examples were made from wood and

plastic to help users gain an understanding of how these materials affected the operation of the mechanisms and to provide some insight into how the choice of materials might affect a finished automaton. A few of these examples are shown in Figure 5.14.

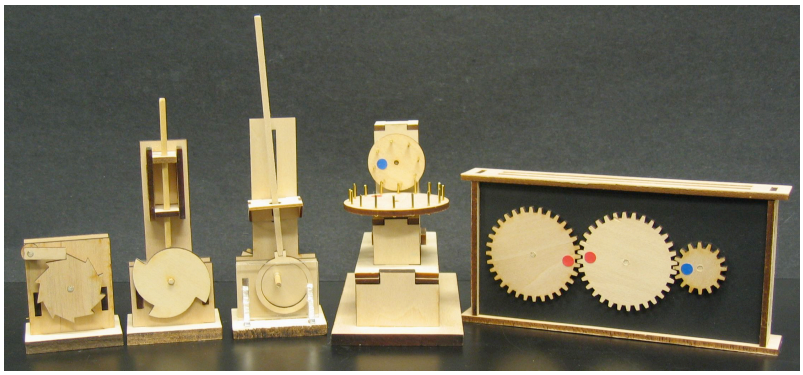

Figure 5.14: A selection from the simple movements used to explore motion during the user testing. From left to right: a ratchet and pawl, three-lobed snail cam, eccentric cam with ring follower used as a crank-slider, pinwheel gears with a 2:1 ratio, and a set of three spur gears with a 1:2 ratio.

These mechanisms proved invaluable when working through ideas with the test users. Many times a user would have an idea for an automaton and a sketchy notion of the desired behavior and motions but without a clear idea of how to make it happen. Very often, spending five or ten minutes examining and discussing the sample mechanisms either led to a possible approach for the desired automaton mechanism or led to a desirable change to the original idea.

5.3.2 Kits and Instructions

During the Spring 2004 semester, an undergraduate student became involved in this research with the goal of determining what would be needed to package an automaton into a kit that could be built by a novice system user before she set about designing her own automaton. To do this he took the fabrication files for the earliest of the automata built with an early MachineShop version and fabricated, assembled, and learned about the process with no input from any outside source. He created a set of instructions for assembling this automaton but because of time constraints was unable to present the kit and instructions to children for evaluation.

5.3.3 Videos

Seeing automata in motion reveals much more about them than seeing them at rest. Videos have been made not only of the simple mechanisms that are used in the movement explorer, but also of all of the automata that have been built with the system. While certainly inferior to having the chance to examine and manipulate automata personally, videos can supply many additional examples without the need to fabricate duplicates every time one is needed.

5.3.4 Posters

Posters of sample automata and mechanisms are simple to create but can be powerful learning tools. Because they can be annotated and important concepts or key points highlighted, they make an effective learning tool when displayed in the work area. They provide a resource to which a user can return as often as necessary to more fully understand concepts and ideas and they are valuable objects around which topics can be discussed. Because they are easy to create, users can refine their understanding by creating a poster for others to use.

5.4 Summary

The MachineShop system is composed of a computer-aided design and computer-aided manufacturing (CAD/CAM) application, a computer controlled device for fabricating components produced by the software, and a collection of educational materials to support children as they learn about making mechanisms for contemporary automata. The software supports four key tasks in the design process. First, it allows users to define the motions they wish an automaton to have and enables them to search for mechanical components that will supply those motions. Second, it allows users to create instances of those components specifically tailored for their needs and to create files from which those components can be fabricated. Third, it will allow users to assemble components into virtual mechanisms to insure that once fabricated a mechanism will perform as intended. Finally, it allows users to save components and mechanisms to libraries, to browse

the libraries and reuse components or mechanisms, and to share those saved components and mechanisms with other users. Many of these capabilities can be found in systems designed for adults, but MachineShop is unique in making them available to children.

The MachineShop system can make use of a number of computer controlled output devices to fabricate components created with the software. By automating the fabrication of mechanism pieces, MachineShop helps young users over the twin hurdles of tool use and properties of materials; neither of which most children are adept at or knowledgeable about. Having access to machines of this type allows children to design far in excess of what they are capable of fabricating. The landscape of these kinds of tools is continuing to grow and the software is capable of creating files for many different devices, not just the laser cutters that were used during this research.

The MachineShop system uses a variety of materials and examples to assist users as they learn about creating automata. These materials come in a number of forms; example automata and mechanisms, posters describing mechanisms and movement, kits that can be assembled into working automata, even videos that show and explain concepts and techniques. The combination of all of these elements into the MachineShop system is intended to help users succeed and have fun while becoming a more accomplished builders. Chapter 6 examines the how successful this process has been for the users who tested the system.