MARK-IT: A MARKING USER INTERFACE FOR CUTTING DECOMPOSITION TIME

Nicholas C. Yang[1] Andrew S. Forsberg[1] Jason F. Shepherd[2] Ricardo M. Garcia[2]
Karl G. Merkley[3]

[1]*Brown University, Providence, RI, U.S.A. {nyang, asf}@cs.brown.edu*
[2]*Sandia National Laboratories, Albuquerque, NM, U.S.A. {jfsheph, ricgarc}@sandia.gov*
[3]*Elemental Technologies, Inc., American Fork, UT, U.S.A. karl@elemtech.com*

ABSTRACT

We present Mark-It, a marking user interface that reduced the time to decompose a set of CAD models exhibiting a range of decomposition problems by as much as fifty percent. Instead of performing about 50 mesh decomposition operations using a conventional UI, Mark-It allows users to perform the same operations by drawing 2D marks in the context of the 3D model. The motivation for this study was to test the potential of a marking user interface for the decomposition aspect of the meshing process. To evaluate Mark-It, we designed a user study that consisted of a brief tutorial of both the non-marking and marking UIs, performing the steps to decompose four models contributed to us by experienced meshers at Sandia National Laboratories, and a post-study debriefing to rate the speed, preference, and overall learnability of the two interfaces. Our primary contributions are a practical user interface design for speeding-up mesh decomposition and an evaluation that helps characterize the pros and cons of the new user interface.

Keywords: meshing, decomposition, marking user interface, gestural commands, time-to-mesh, post-WIMP

1. INTRODUCTION

The goal of meshing is to produce a discretization of a space, a mesh, that is suitable for the simulation of a given engineering problem. Meshing, particularly hexahedral meshing, is a complicated procedure that often requires the use of rather sophisticated, interactive tools (i.e. CUBIT, TurboMesh, SolidMesh) to produce a usable mesh. For the user, decomposition can be the most time-consuming part of the meshing process; it includes determining and specifying operations such as dividing and simplifying a model's geometry into more primitive pieces. Decomposition is usually required before an automatic meshing algorithm can be applied to components of a model. In many cases, the decomposition as a whole cannot be automatically performed and must be done manually; oftentimes, the user is an irremovable part of the meshing process [1]. Algorithmic and user interface (UI) research are the current techniques for alleviating this bottleneck . This study focuses on the latter by testing whether the use of marking gestures, in the context of the 3D tasks at hand, can significantly reduce decomposition time. As noted in [2], any type of automation or optimization in decomposition could "vastly reduce the time to mesh for hexahedral elements."

Decomposition operations must be accessed through a user interface in any meshing system. Two popular UI styles are Windows-Icon-Menu-Pointer (WIMP) and command-line. In general, user interfaces have remained mostly unchanged for the past two decades and predominantly consist of WIMP UIs [3]. Traditional WIMP UIs are widely popular mainly because they have proven to be relatively easy-to-learn. Command-line UIs originated much longer ago but are sometimes preferred for systems with large numbers of operators and operands. Oftentimes, command-line UIs are significantly more efficient than WIMP UIs once mastered.

In this study, we applied the concept of a post-WIMP UI to a sophisticated meshing system, Sandia National Laboratories' CUBIT, which has been in development for about 8 years and combines the best aspects of both conventional WIMP and command-line UIs. CUBIT is a 2D and 3D finite element mesh generation tool that produces meshes for finite element analysis. Development for the marking interface was done exclusively in CUBIT, but we expect the concepts of the implemented marking UI to extend over to other meshing systems as well.

Specifically, we developed a post-WIMP *marking UI*, which we have termed "Mark-It." With Mark-It, users draw 2D lines in the context of 3D Computer-Aided Design (CAD) models to specify meshing operations and operands. We hypothesized a marking UI would be effective due to the graphical nature of meshing tasks. The intent was to discover the advantages and disadvantages of a specific marking UI targeted at mesh decomposition. Results from this study are preliminary data that show how effective a marking post-WIMP interface could be in a meshing context. Our results and evaluation suggest that for purposes of meshing, our post-WIMP marking interface is faster than most traditional WIMP interfaces. Furthermore, we also received feedback about several possible improvements to our marking UI that would make it faster and easier to learn. The advantages and disadvantages of both WIMP and post-WIMP interfaces are discussed in [3].

The non-marking CUBIT interface we compared Mark-It to consists of toolbars, menus, point-and-click selection, and a command-line. Users in our study were asked to click on separate toolbar icons or to use hot-keys in order pan, rotate, zoom, and enter certain entity (e.g. surface, volume, curve, vertex) selection modes when using the non-marking interface. To execute a decomposition command, users had to select appropriate tools using the WIMP GUI and then type in the commands based on entity IDs. Mark-It aims to extend meshing UIs of this type to go beyond using the mouse in the 3D graphics window for only picking and navigating. It allows the user to use the mouse to perform several different decomposition operations by drawing directly on a model.

2. RELATED WORK

Time-to-mesh is the total time it takes to create a finite element mesh given a geometric model. A common goal of many researchers is to minimize the time-to-mesh. As stated above, decomposition is necessary because many meshing algorithms require pre-processing of the geometric model. In many cases, especially cases with high problem complexity, a user must manually make decisions, using a tool such as CUBIT, to prepare a model for a given meshing algorithm. These human-made decisions can improve overall efficiency in a way automatic algorithms cannot. For example, the user may improve efficiency by reducing the total number of elements created, improving the quality of elements created, or by minimizing the number of steps to achieve the required decomposition. Nevertheless, this section goes into more detail on non-user-based algorithmic approaches to reduce the time-to-mesh and also discusses related work in marking post-WIMP UIs.

2.1 Algorithmic Approaches

Historically, a great deal of effort has been spent in reducing the time-to-mesh by introducing new meshing algorithms or improving the scope of existing algorithms. Because of limitations in current hexahedral meshing algorithms, algorithms for decomposing models into mesh-able primitive shapes have received a great deal of attention. Several examples of algorithmic work to reduce the time-to-mesh will be highlighted in this section.

Two of the most highly used algorithms in hexahedral meshing today include the "mapping" and "sweeping" methods [4][5]. Enhancements to each of these algorithms have been made specifically to reduce the amount of decomposition necessary to produce a mesh, thereby speeding up the time to produce the mesh [6] [1] [7]. Specifically, the "submapping" algorithm [1] and the many-to-many sweeping tools [8][9][10] [11] are the algorithms used heavily in CUBIT today. Both of these techniques are geometry pre-processing algorithms that extend the existing "primitive" algorithm. Many-to-one sweeping and multi-sweeping involve extruding surface meshes from multiple source surfaces, as opposed to the conventional sweeping algorithm which allows only one source surface. Volume submapping involves breaking up volumes into sub-regions that are quicker and easier to mesh as separate entities, instead of as a whole, using standard mapping transformations.

Another study tried to completely automate the decomposition process entirely by using a technique called Feature Recognition (FR), which was able to automatically generate mesh-able volumes directly from a class of imported ACIS models [12]. Although the results of FR were impressive, the implemented system was not general enough to supplant the need for human interaction. The implemented FR system also lacked some options, such as the ability to specify automatic meshing patterns to guide the decomposition.

Several other algorithmic approaches for reducing time-to-mesh exist as well, and not surprisingly, none are fully automatic. [13] reduces datasets to wavelet representations and coefficients, but it also requires obtaining triangulation calculations in a lookup table and performing inverse wavelet transformations. [14] uses features on a "reference mesh" to automatically generate mesh-able volumes on a desired model, but consequently, it requires having to find the appropriate reference meshes for specific geometries. [15], [16], [17], and [18] try to improve time-to-mesh by using an octree approach to produce an automatic, all-hexahedral meshing algorithm. However, these approaches impose some additional work for dealing with elements on the boundary, and they do not inherently work well for producing contiguous meshes across material interfaces or boundaries of the simulation model.

The goal of these techniques is the same as the goal of this study: to reduce the time-to-mesh. Rather than improving on the underlying algorithms to further automate the meshing process, Mark-It focuses primarily on optimizing the user interface.

2.2 Post-WIMP UIs

Several research studies have shown the potential of integrating marking UIs into various application areas, as is discussed in the following paragraphs.

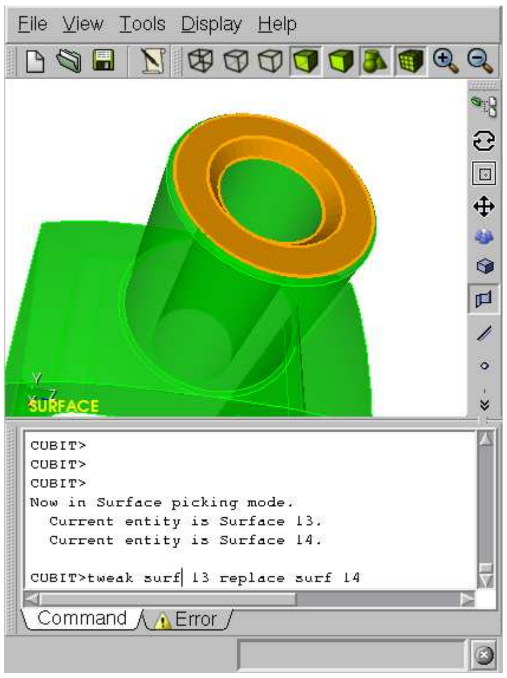
In [19] gestural movements and marks were used to create 3D geometry in a program called SKETCH. Mark-It also uses gestural marks to perform certain operations, but it is different because it contains a considerably greater number of gestures. There is no simple set of SKETCH-like gestures that can be directly applied to mesh decomposition.

In [20], a stylus pen was used in combination with a marking menu for composing music. This post-WIMP interface received much positive feedback and was even noted by many users to be preferred over existing alternatives. [20] relates to Mark-It in a number of ways. First, both use a marking interface involving gestures and marking-menu-like mechanisms. Second, both interfaces call for a sensible approach to learning and teaching a gestural interface. Third, they both apply the idea of being able to draw what is desired directly in the graphics window. The main differences between [20] and Mark-It is the number of commands supported and the dimensionality of the problem domains.

In [21], which specifically studied the use of marking menus, a radial menu or straight-line marks were used to select operations to perform. The study showed that once learned, marks were used more often and were more efficient, on average being 3.5 times faster than conventional pull-down menus. There is a noted learning curve, however. Both the speedup and learning curve are expected in Mark-It as well. [21] only implemented menus containing even numbers of operations, where each menu contained 12 operations at most, because it was determined in [22] that such a setup tended to increase user performance. Due to the large number of possible decomposition commands, Mark-It was implemented as a variation of such a marking menu system, using more than just radial and straight-line marks. The number of operations is only bounded by the number of distinct marks that can be recognized and learned by the user. This implementation was intended to address the limitation of traditional marking menus by combining the advantages of marking menus with a gesture recognition system.

3. THE TWO INTERFACES

3.1 Non-Marking CUBIT UI



```
CUBIT>
CUBIT>
CUBIT>
Now in Surface picking mode.
  Current entity is Surface 13.
  Current entity is Surface 14.

CUBIT>tweak surf 13 replace surf 14
```

Figure 1: The CUBIT window. In the non-marking UI, users must select modes either by using the toolbars or by using corresponding shortcut keys on the keyboard. Modes include entity (surface/volume/curve/vertex) selection and camera pan/zoom/translate. Selection is required to discover IDs of model entities. At the bottom is a command-line prompt for users to type in commands to execute on entities, based on IDs.

The original CUBIT UI's decomposition operations were primarily executed using combinations of toolbars, menus, key presses, mouse interactions, and a command-line window. Many operations can be invoked through either 2D widgets or the command-line. The mouse is used to change modes using the toolbar, to perform camera movements, and to make entity selections to find IDs. Figure 1 shows a screenshot of the CUBIT window. All of the toolbar commands used in this study have corresponding shortcut keys on the keyboard. It is also possible to use abbreviations in typed commands. Many commands require parameters. Parameters include ID numbers, coordinates, and fractional values. Specifying the right coordinates and fractions may require some previous knowledge of where the model is located in world-space, relative to the origin.

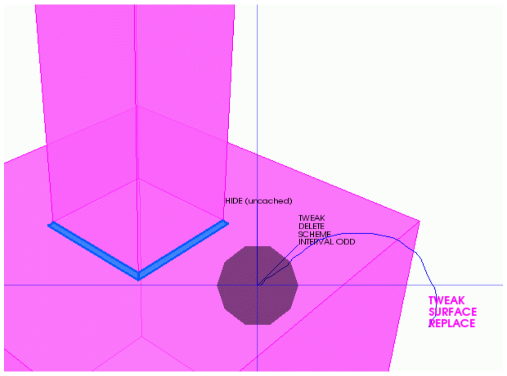


Figure 2: In this example, the four small surfaces drawn in blue that connect the rectangular solid to the cube are to be removed using the "tweak surface replace" command. In the non-marking UI, the user would need to know the IDs of all 5 surfaces and then type in or echo those IDs to the prompt. Using the marking UI, the user selects the four surfaces and then draws a "tweak surface replace" mark starting from the plane of the cube, as illustrated.

The syntax of typed commands must be known by the user. There is extensive documentation on CUBIT's command-line syntax available online at http://cubit.sandia.gov. At least 1000 different types of commands exist in CUBIT. While a complete language of marking gestures has not been devised for all of the commands, we believe that a marking interface for just the most common operations would still yield a significant improvement on the time-to-mesh.

At times, certain entities may not be selectable due to other occluding entities. For such cases, the user can use the "visibility off" command to hide occluding entities. Alternatively, the tab key can be used to cycle through the set of entities appearing underneath the cursor.

3.2 Marking CUBIT UI

The marking UI is intended to simplify the overall user interface for executing decomposition commands. Most interactions are done right in the graphics window and involve just the mouse. The hope is that this is fast, intuitive, and direct. In the WIMP GUI, users must frequently switch between graphics windows, toolbars, menus, and buttons. The time taken to switch between different panels or parts of the interface is nontrivial.

The marking interface does not rely on toolbar icon selections. All camera operations are done using a single mouse button [23]. Decomposition operations are performed by drawing marks with the left mouse button. The user executes an operation by first clicking to select one or more entities and then drawing an appropriate mark, possibly relative to a specific model feature, to execute a desired command. Different types of entities are selected by toggling modes; from the "surface selection" mode, the user double-clicks to enter or exit "body selection" mode and triple-clicks for "curve selection" mode. To perform an operation on all the entities in a model, the user should not manually select any entities.

There are many possible marks, making it hard to remember or distinguish some of the commands. Although the implemented system contains only a subset of the full range of CUBIT commands, the marking UI still implements over 50 different CUBIT operations, which can be overwhelming for beginners. Consequently, a primitive help system was implemented to draw help text or hints directly inside the graphics to give the user a better idea about how to execute different commands.

When a user draws a mark that corresponds to a valid command, a label will appear by the cursor, denoting the command that will be executed if the user releases the button at that moment. If the user makes a mistake while drawing a mark, the user may recover by moving the cursor to the cancel area, a circle at the start of the mark, release the button and draw the mark again. Examples of mark drawing are illustrated in Figures 2 and 3.

Mark-It's novel gesture recognition algorithm was inspired by marking menus [21]. The recognition system tracks which quadrants are intersected, relative to the mark's initial click point. Because the implemented gesture recognition is based on quadrants and which quadrant the user initially moves the cursor into, axis lines are drawn whenever the user makes a mark. Depending on which direction the user initially moves the mouse, certain commands will become valid or invalid. If the user holds down the left mouse button without moving, hints on which quadrant to start a particu-

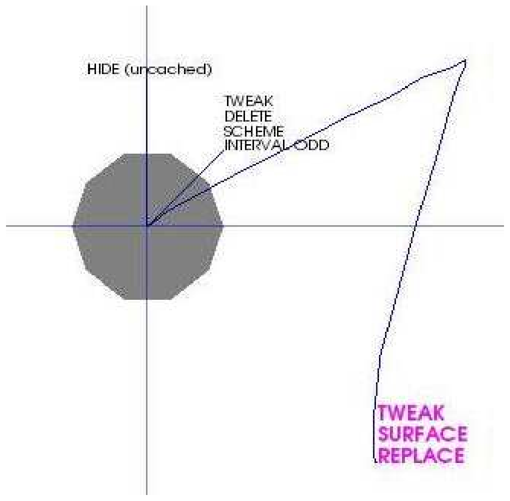


Figure 3: An example of drawing a mark to execute the command "tweak surface replace." Notice that axis lines are drawn at the start of the mark, to show the different quadrants, along with a circle denoting the cancel area. Because the user moved into the upper-right quadrant first, only the hints for that quadrant remain on the screen. When the user drags the mouse such that it makes the correct mark for "tweak surface replace," the command label appears by the cursor, as illustrated.

lar mark in are displayed. An illustration of all the possible hints in "surface selection" mode is shown in Figure 4. If the user enters the wrong initial quadrant, the user should cancel and draw the mark again.

For operations that require the user to specify a certain number, coordinates or fractions, the marking UI utilizes a "virtual slider" mechanism, which allows the user to drag left or right from any location to adjust a specific value. To confirm and select a value, the user need only hold down the left mouse button without moving for one second before releasing. Cancelling from the virtual slider consists of clicking once without moving the mouse.

Some operations require selecting the x, y or z axis (e.g. cutting perpendicular to the x axis) or selecting a group of vertices. When an axis selection is required, a tri-colored axis will be drawn at the relevant point on the model. The user clicks and drags from a point off the model until the "selection line" turns red, green, or blue before releasing the mouse button to make an x, y, or z axis selection, respectively (see Figure 5). The user can cancel at any time by bringing the cursor back to the axis origin before releasing. For selecting vertex groups, the user must click to select or deselect vertices. The user will only be allowed to select as many vertices as the operation will allow. To accept a vertex group selection and execute the operation,

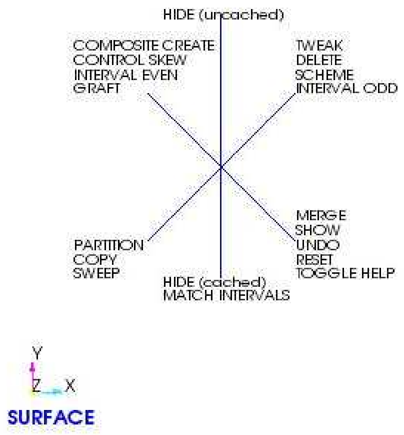


Figure 4: In the marking UI, when the user holds down the left mouse button without moving, hints for which quadrant to begin a mark in are displayed. The illustrated hints are for the "surface selection" mode.

the user makes a check-mark gesture (Figure 6, and to cancel, the user draws a line diagonally downright.

To deal with overlapping entities, the marking UI uses "cached" and "uncached" hiding, which allows users to hide surfaces and at the same time specify which hidden entities to use in the subsequently chosen operation. Because there is no mechanism for arbitrarily selecting which entities to hide and unhide, especially as the number of hidden entities increases, users should cache-hide all entities that they want to be parameters in the operation and use uncached-hiding for entities that are in the way (and need to be hidden) but should *not* be part of the operation. This implementation is just one possible design for handling overlapping entities in the marking interface. It should be noted that other methods may be implemented to replace this.

4. RESULTS AND DISCUSSION

To evaluate the effectiveness of the marking interface in comparison to the original command-line based interface, we had ten people at Brown University learn both UIs in a brief 15 minute warm-up tutorial and then decompose four models twice, one time for each UI. Each user was given a packet with instructions on how to use each UI to mesh the 4 models. The instructions for each model consisted of three columns, one for a picture of the model at the current decomposition step, one for a description about how to use the non-marking UI to execute commands, and the last column for instructions and picture references on how to use the marking UI to draw marks. We also had a refer-

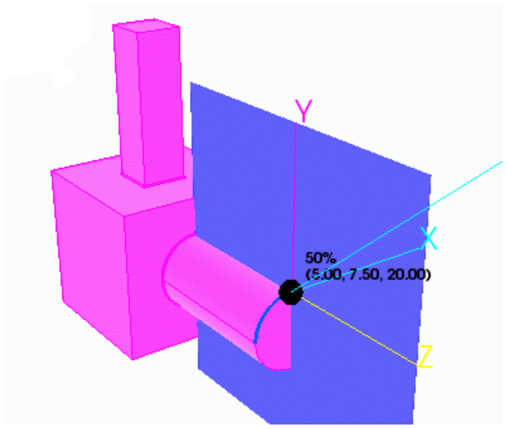


Figure 5: Selecting an axis. A "selection line" is drawn from the cursor to the axis origin. When the line changes to the corresponding axis color, the user can release the mouse button to select that axis. In this example, the body is being cut by a plane perpendicular to the x-axis, so a corresponding cutting plane is also drawn for this x-axis selection.

ence sheet of all the possible marks available for users to look at if they so desired (see Appendix). Because we did not require users to have any previous knowledge of CUBIT or decomposition, we allowed them to ask questions when they needed help remembering commands, had trouble completing steps in the procedure, or were confused about the general procedure. After users completed the study, they were asked to fill out a questionnaire that asked them to rate the learnability, speed, and preference of one interface over the other.

Each session lasted about 1-2 hours, depending on how well the user adapted to and learned the CUBIT system. We suspect that the time it took was highly correlated to the users' background experience, particularly experience having to do with 3D modeling or CAD programs. None of the users noted any discomfort, distortion in vision, fatigue, or disorientation during the study.

In addition to the marking UI, we also allowed users to try using two other devices in conjunction with the marking UI as post-WIMP options. We had a free 3D tracker with a 2-stage push-button that was capable of rotating and translating the camera, depending on whether the user pushed the button softly or with more force. Also installed was a Wacom tablet for helping users to draw marks, in case it was too difficult to draw marks with the mouse.

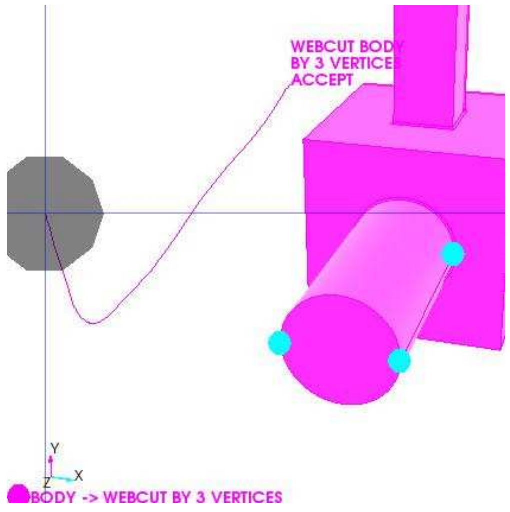


Figure 6: An example of selecting 3 vertices and accepting the selection by making a check-mark.

4.1 Demographics

All of the test subjects had experience in using both 2D and 3D graphics programs. The list of programs includes Photoshop, GIMP, Illustrator, CorelDraw, Maya, 3D Studio Max, AutoCAD, and Infini-D. Most of the programs were noted as having been used before for more than 10 hours. Only three users wrote that they used some programs for less than two hours. One user had prior experience in meshing, but it was not using CUBIT.

The age group ranges from 19 to 32, where 6 of the users are male and 4 are female. A majority of the users are currently students, one being a postdoctorate, and some of the listed concentrations include Applied Math, Geophysics, Education, Engineering, and Computer Science. The users who were not students had job titles such as UI Developer and media coordinator.

4.2 Ease of Use

Overall. Every user commented that the marking interface outperformed the non-marking interface. All but one of the users noted, though, that there was a significant learning curve that had to be overcome before the marking interface could be effectively used. About half of the users found that only in the later examples did the marking interface became easier and faster. The users agreed that using the marking UI, commands were more direct. They also appreciated how the marking interface eliminated the need to know entity IDs. There was one user who commented that learning either interface required roughly the same amount of work but that it would probably be more worthwhile to learn the marking UI.

Comparison. Both interfaces had their strengths and weaknesses. We noticed that several users frequently made a lot of typos in the non-marking UI, but similarly, users drew incorrect marks using the marking UI, especially in cases where the user had to to start the mark on a particular curve or surface. One of the noted advantages of the typing interface was the ability to enter commands without having to actively look at the graphics window the whole time, although it sometimes made it harder for users to notice when they made a typo. Users found the marking UI to be a lot more intuitive and simpler to perform certain operations. However, a distinct disadvantage of the marking UI was bad gesture recognition, which was particularly noticeable when users wanted to draw gestures quickly without having to confirm that a command was correctly selected. Users often executed undesired operations on the model by accident.

Modes. Half of the users thought that the toolbar mode selection of the non-marking UI was more straightforward for selecting different types of entities, as opposed to the required double and triple clicking of the marking UI. The modes of the marking UI was also confusing to users particularly in cases where a user had to be in "body selection" mode to cut a body, but the mark to be drawn had to start on a certain curve or surface.

4.3 Camera Controls

Non-Marking. About half of the users remarked that the camera functions of the non-marking UI were more intuitive than the single button camera (the Unicam [23]) of the marking UI. They had a better immediate understanding of how to navigate by clicking toolbar icons for rotating, panning, and zooming the camera. One user remarked that the controls were almost exactly identical to a CAD program that this user had previous experience with. The Unicam seemed to be difficult for users that had not used Unicam-like camera navigation before. Particularly, a Unicam is supposed to zoom or pan, depending on whether the user initially drags the mouse horizontally or vertically, but several users had difficulty in getting the mouse to initially drag vertically when they wanted to. One of the users mentioned that it was annoying to have to first move left or right to pan up or down; the user preferred moving the mouse directly up or down, instead of having to first move left or right. Another user, who had extensive experience using Maya, remarked that some "unlearning" of Maya was required in order to get used to the Unicam controls.

Marking. On the other hand, the use of Unicam also received much positive feedback from several users in the study. For instance, one of the users said that for many operations, the Unicam saved at least a half second of time that was required in the non-marking UI when switching between different camera or selection modes. Another user wanted to use the Unicam for both the marking and non-marking examples instead of having to resort back to the toolbar icons for non-marking examples.

4.4 Speed

Typing. About a quarter of the users noted that typing commands was more direct and faster than having to learn a gesture mark in order to execute a command. There were two users who simply preferred typing over mouse interactions in general, saying that mouse-drawn commands took too long to learn and that drawing the shapes of the various gestures was difficult to remember and do quickly, even with the reference sheet. One user said that had all of the IDs and arguments been known ahead of time, the non-marking UI might have been easier to use for the particular examples used in the study. A number of users did also complain, however, about the lengthy and confusing syntax of some of the commands.

Drawing Marks. The marking interface received mostly positive feedback. We had one user who had previous meshing experience, and this user expressed interest in the concept of the implemented marking UI. All the users said that the marking UI was harder to learn. They also commented, though, that with more practice, the marking UI would be faster than the non-marking UI. Certain annoyances that users noted in the non-marking UI include having to type in ID numbers and having to navigate from window to window in order to select a toolbar icon, type a command, or navigate the camera. The marking UI bundled all of these tasks into the graphics window alone. In particular, for longer and more repetitive examples, the marking UI was always preferred over the non-marking UI. Users commented that such examples were "tedious" and "painful" to complete using the non-marking UI. In the post-questionnaire, users rated the non-marking UI to be between 30-80% slower than the marking UI.

4.5 Learnability

Previous Experience. Many of the users who had previous experience in using 3D modeling or CAD programs commented that the non-marking UI was more intuitive to initially learn. Comparisons were made to programs with similar toolbar icon camera functions. Users also said that it was more obvious what command was going to be executed using the non-marking interface. To them, typing in actual commands using words was more intuitive than having to remember how to draw shapes or patterns. More than half of the users also thought that using the keyboard to input exact numbers was easier than using the slider-based mechanism of the marking UI. A common complaint was that there was not enough visual feedback or accuracy in using the slider. Users who said that their previous experience helped them in completing the tasks mostly said that their background facilitated camera navigations. Those who had never used a Unicam before found the Unicam hard to use. Although the hope for the marking UI was to have it readily learnable by any user, the eventual target audience will consist of people who perform decomposition on a regular everyday basis. Perhaps, we should have limited the user study to users with more background experience.

Making Marks. The marking UI was difficult to learn, even with the reference sheet, and we suspect that there was not enough time for users to become accustomed to the restrictions and rules for making marks. At the beginning, many users had to draw some marks several times before they were able to get the right command to show up. Most of the users believed that this problem was a simple matter of practicing. However, one user commented that the marking UI was not intuitive at all and that even with everyday practice, it would be difficult to learn. It should be noted, though, that the implemented gesture recognizer can easily be replaced with any other type of marking-menu or gesture recognition system. A possibility is using the stroke recognition techniques described in [24], which would probably significantly increase the accuracy of the recognition. The quadrant-based marking UI in this study is only a preliminary system that was implemented to demonstrate the possibility of using a marking post-WIMP UI in decomposition. There is a wide range of possibilities for further expanding and developing the work started here in this study.

Using the Mouse. About a quarter of the users found themselves clicking the wrong button while using the marking UI. Some users had difficulty in remembering to use the middle button for camera navigations and the left button for marking commands. While this is a minor detail, perhaps the learnability could have been improved had the controls been limited to one mouse button. Other complaints include having to hold down the button one second before a parameter was accepted by the virtual slider. One user commented that using conventional "OK" and "Cancel" buttons would have sufficed.

Help System. Users thought that the help system for the marking UI was indeed beneficial, but many thought that it was insufficient for becoming familiarized with the interface. Two of the users wanted a more interactive system that updated as marks were being drawn. Perhaps the number of commands was simply too overwhelming in the marking UI for novice users.

4.6 Preference

Nearly all of the users agreed that the command-line based UI was easier and more intuitive to learn. However, they pointed out that it was still rather annoying or frustrating to have to click the toolbar, graphics window, and then the command-line to execute a single command. One of the users pointed out that using just the mouse for all commands was a lot more convenient. For longer examples, all users preferred using the marking UI. A lot of users still missed the comfort and familiarity of using the traditional non-marking WIMP interface, however.

Some of the positive comments about the marking UI include the ability to make marks directly on the model, which was a noted improvement over having to echo ID numbers on the command-line after selecting entities. For commands such as cutting by a plane perpendicular to the x, y, or z axis, cutting by a plane defined by three vertices, deleting entities, and partitioning entities, this was a significant improvement. Two of the users said that the tasks themselves seemed inherently visual, and that is why they preferred the marking UI over the non-marking UI.

More than one user liked the idea of having a hybrid interface, where there would be a method for switching between the non-marking and marking UIs, because according to four of the users, certain tasks seemed better-suited for one UI than the other. Particularly, for tasks requiring number values such as fractions or coordinates, many users wanted to just type in the desired values as opposed to having to use the slider in the marking UI. Users also voiced a concern over not being able to select certain numbers when using the slider, as the values only change in 0.05 increments.

The marking UI was almost always noted as being more fun to use, because it was something different and unconventional, but the non-marking UI received more votes for being easier to learn and easier to use. The non-marking UI was straightforward to many users, although one user acknowledged the method was "old-fashioned." Many users were more confident using the non-marking UI, although two of the 10 users in the study did not note a significant difference in the difficulty of either interface; it was noted by all users that with practice, any implemented UI would become easier to use. The non-marking UI did receive more negative feedback in regards to being tedious, especially on models requiring repetitive steps. Three

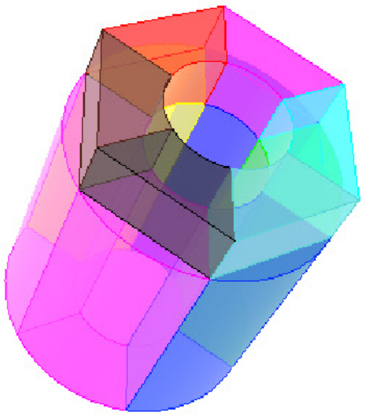


Figure 7: An example that was considered tedious by several users in the user study. Similar operations had to be done on all six sides of the model. The non-marking UI received especially negative feedback on this example.

users voiced a need for some more automation on these models. Figure 7 is an example of such a model.

5. ERROR ANALYSIS

There are several sources of error in the user study. Discussion of each follows:

First, it is possible that some of the controls were too sensitive. More than one user believed that some tuning was required, especially mouse movement in slider number selections. Many of the mouse and gesture movements could have been further tweaked or adjusted to accustom the different ways that users initially react to and interact with such a system.

Second, it is possible that given more practice with mesh decomposition or more knowledge about meshing concepts, the users might have reported differently on the two interfaces. Most of the users did not know the purpose or steps behind meshing. A more detailed explanation of the reason for each step may have contributed to more confident user feedback.

Third, there were several bugs in the prototype code. Given the focus of the study, these bugs were considered to be trivial enough to ignore. Some of the bugs included drawing errors in the graphics window and erroneously chosen rotation points for the camera. However, had such bugs been fixed perhaps there would have been more positive feedback regarding visuals in the graphics window, as some users noted not having enough visuals to confirm or know about what certain commands or settings were doing in the program.

Finally, the Wacom tablet and 3D tracker were also buggy. In the marking UI, clicks were detected only when the user did not move the mouse because any cursor movement during a click implied dragging. The Wacom tablet was too sensitive in that every time the user lifted the pen, it would oftentimes report a drag when the user actually wanted to click. The effectiveness of the Wacom might also be related to previous experience; the two users who had experience in using a Wacom-type of input device were the only ones who commented on the ease of use for making some of the marks. As for the 3D tracker, we had problems in the camera translation. Perhaps the Wacom tablet and the tracker would have shown better potential had they been more thoroughly tested and implemented, but they still demonstrated good possibilities for post-WIMP interfaces in a meshing environment.

6. FUTURE WORK

More Visuals. Mark-It's UI is only one possibility for a marking UI in a meshing system. Some of the goals that we wanted to achieve included avoiding unnecessary movement of the mouse and unnecessary interaction with the keyboard. Hence, we implemented double and triple clicking at arbitrary locations to avoid having to move the mouse or press keys on the keyboard in order switch modes. The same logic applies to the "1-second accept method"; instead of requiring the user to have to move the mouse to an "OK" or "Cancel" button, the user could click wherever the cursor was currently at, and release the mouse accordingly. One of the major problems of Mark-It's interface was a lack of visual feedback. Perhaps had there been more visual feedback, especially in the "1-second accept method," users would have been more receptive to the idea of not having "OK" and "Cancel" buttons. Creating more visuals may be a possible first step in improving the marking interface.

Hybrid Interface. More than one user expressed an interest in having some combination of typing and making marks as one unified user interface. We believe that such a system may especially be effective for users who are accustomed to previous programs that utilize WIMP techniques and wish to slowly transition to or complement their methods of interaction with a gestural marking interface.

More User Studies. Perhaps a more interesting user study would have involved implementing and comparing a user interface more comparable to such commercial software programs as Maya, and comparing that with the currently implemented marking UI. One of the users remarked that it was rather unfair to compare a command-line based UI to a more state-of-theart marking UI. A possibility would be to compare Mark-It to Tracking Menus [25] or to future versions of CUBIT.

More Post-WIMP. The Wacom tablet and 3D tracker showed very good possibilities for future implementations. In respect to the Wacom tablet, two of the users, who had experience in using pen and tablet input devices, liked the ease of use and natural motions of drawing marks by using a pen instead of the mouse. For the 3D tracker, most of the users commented that had the tracker been better implemented, they would have liked to use it more than they did during the study. This was especially true for the users that disliked the Unicam. One of the inconveniences about the tracker that was mentioned was having to deal with the cable getting in the way and making awkward rotations with their wrists. Some users said that they would have preferred using their non-dominant hand for doing camera navigations with the tracker while they did commands with their other hand. Others did not think a two-handed interface would be beneficial. The use of these two devices as well as the possibility of two-handed interfaces in a meshing system can serve as a basis for other user studies in future work. Further extensions to the tracking system may involve stereo viewing and head-tracking.

Better Learning. Finally, future work may also involve studies about improving the learnability of marking interfaces. The implemented help system only shows hints about how users should start a mark, but it does not actively update to show what hints are possible after drawing a partial mark. A more thorough and complete system that helps users avoid using the reference sheet would have been ideal.

7. CONCLUSION

The implemented marking UI has shown the potential of a post-WIMP marking interface when integrated with a complex meshing system such as CUBIT. The marking UI is only one of the many possibilities for a post-WIMP interface and could be substituted with variations or entirely different interfaces. An example would be using gesture recognition algorithms described in [24].

Being able to specify commands and operands by drawing marks inside the graphics window in the context of the 3D model helped users significantly in achieving the decomposition goals in less time. In the non-marking UI, much time was consumed in switching between different parts of the WIMP GUI. We believe that marking UIs are scalable and extensible to other meshing systems, and implementing a marking UI in other systems would be worthwhile in future studies.

Although this version of Mark-It is in its early stages of development, we believe that there is much potential for post-WIMP UIs in a meshing environment. We believe future studies should focus on the scalability of the UI, ability to learn the UI, and evaluating effectiveness.

8. ACKNOWLEDGEMENTS

This work was supported by LLNL Research Subcontract No. B527302. We would like to thank the CUBIT team at Sandia National Laboratories for their continued support and collaboration in this study. We would like to thank Jon Goldman for his help, including feedback and comments on designing the Mark-It prototype.

References

[1] White D.R., Mingwu L., Benzley S.E., Sjaardema G.D. “Automated Hexahedral Mesh Generation by Virtual Decomposition.” *Proceedings of the 4th International Meshing Roundtable*, pp. 165–176. Sandia National Laboratories, October 1995

[2] White D.R., Saigal S., Owen S.J. “Meshing Complexity of Single Part CAD Models.” *Proceedings of the 12th International Meshing Roundtable*, pp. 121–134. September 2003

[3] van Dam A. “Post-WIMP User Interfaces.” *Communications of the ACM*, vol. 40, no. 2, 63–67, 1997

[4] Cook W.A., Oakes W.R. “Mapped Methods for Generating Three-Dimensional Meshes.” *Computers in Mechanical Engineering*, pp. 67–72. August 1982

[5] Staten M.L., Canaan S.A., Owen S.J. “BMSWEEP: Locating Interior Nodes During Sweeping.” *Proceedings of the 7th International Meshing Roundtable*, pp. 7–18. Sandia National Laboratories, October 1998

[6] Knupp P. “Next-Generation Sweep Tool: A Method for Generating All-Hex Meshes On Two-And-One-Half Dimensional Geometries.” *Proceedings of the 7th International Meshing Roundtable*, pp. 505–513. Sandia National Laboratories, October 1998

[7] White D.R., Tautges T.J. “Automatic Scheme Selection for Toolkit Hex Meshing.” *International Journal for Numerical Methods in Engineering*, vol. 49, no. 1, 127–144, September 2000

[8] Blacker T. “The Cooper Tool.” *Proceedings of the 5th International Meshing Roundtable*, pp. 13–29. Sandia National Laboratories, October 1996

[9] Mingwu L., Benzley S.E., Sjaardema G., Tautges T. "A Multiple Source and Target Sweeping Method for Generating All Hexahedral Finite Element Meshes." *Proceedings of the 5th International Meshing Roundtable*, pp. 217–225. Sandia National Laboratories, October 1996

[10] Shepherd J., Mitchell S.A., Knupp P., White D. "Methods for MultiSweep Automation." *Proceedings of the 9th International Meshing Roundtable*, pp. 77–87. Sandia National Laboratories, October 2000

[11] White D.R., Saigal S., Owen S.J. "CCSweep: Automatic Decomposition of Multi-Sweep Volumes." *4th Symposium on Trends in Unstructured Mesh Generation*. July 2003

[12] Lu Y., Gadh R., Tautges T. "Volume Decomposition and Feature Recognition For Hexahedral Mesh Generation." *Proceedings of the 8th International Meshing Roundtable*. 1999

[13] Gross M.H., Gatti R., Staadt O. "Fast Multiresolution Surface Meshing." *Proceedings of the IEEE Visualization '95*, pp. 135–142. IEEE Computer Society Press, 1995

[14] Yamada A., Inoue K., Itoh T., Shimada K. "An Approach for Generating Meshes Similar to a Reference Mesh." *Proceedings of the 9th International Meshing Roundtable*, pp. 101–107. Sandia National Laboratories, October 2000

[15] Walton K.S., Benzley S.E., Shepherd J.F. "Sculpting: An Improved Inside-Out Scheme For All-Hexahedral Meshing." *Proceedings of the 11th International Meshing Roundtable*, pp. 153–159. Sandia National Laboratories, September 2002

[16] Tchon K.F., Hirsch C., Schneiders R. "Octree-Based Hexahedral Mesh Generation For Viscous Flow Simulations." *13th AIAA Computational Fluid Dynamics Conference*, AIAA-97-1980. AIAA, June 1997

[17] Marechal L. "A New Approach to Octree-Based Hexahedral Meshing." *Proceedings of the 10th International Meshing Roundtable*, pp. 209–221. Sandia National Laboratories, October 2001

[18] Schneiders R. "Octree-Based Hexahedral Mesh Generation." *International Journal of Computational Geometry and Applications*, vol. 10, no. 4, 383–398, 2000

[19] Zeleznik R.C., Herndon K.P., Hughes J.F. "SKETCH: An Interface for Sketching 3D Scenes." *Proceedings of SIGGRAPH'96*, pp. 163–170. 1996

[20] Forsberg A.S., Dieterich M., Zeleznik R.C. "The Music Notepad." *ACM Symposium on User Interface Software and Technology*, pp. 203–210. 1998

[21] Kurtenbach G., Buxton W. "User Learning and Performance with Marking Menus." *Proceedings of the CHI94*, pp. 258–264. 1994

[22] Kurtenbach G., Sellen A., Buxton W. "An Empirical Evaluation of Some Articulatory and Cognitive Aspects of "Marking Menus"." *Journal of Human Computer Interaction*, vol. 8, no. 1

[23] Zeleznik R., Forsberg A. "Unicam 2D Gestural Camera Controls for 3D Environments." *Proceedings of the 1999 symposium on Interactive 3D graphics*, pp. 169–173. ACM Press, 1999

[24] Lopresti D., Tomkins A., Zhou J. "Algorithms for Matching Hand-Drawn Sketches." *Proceedings of the 5th International Workshop on Frontiers in Handwriting Recognition*, pp. 233–238. 1996

[25] Fitzmaurice G., Khan A., Pieke R., Buxton B., Kurtenbach G. "Tracking menus." *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, pp. 71–79. ACM Press, 2003

9. APPENDIX

The figures in this section show operations that were implemented in Mark-It.

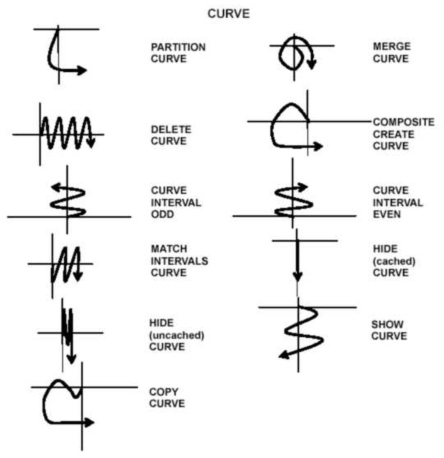


Figure 8: Curve operations.

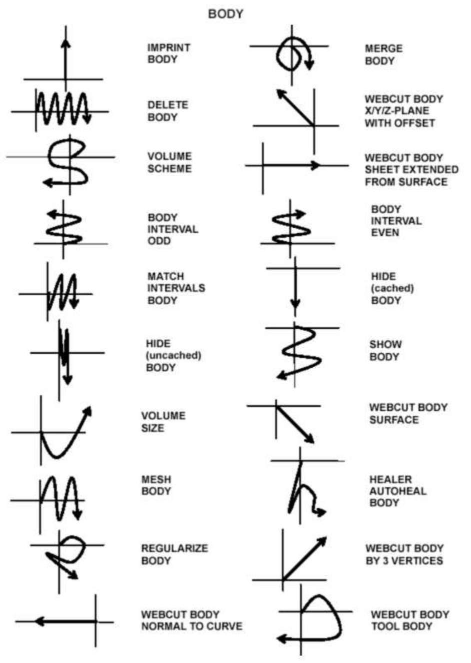


Figure 9: Body operations.

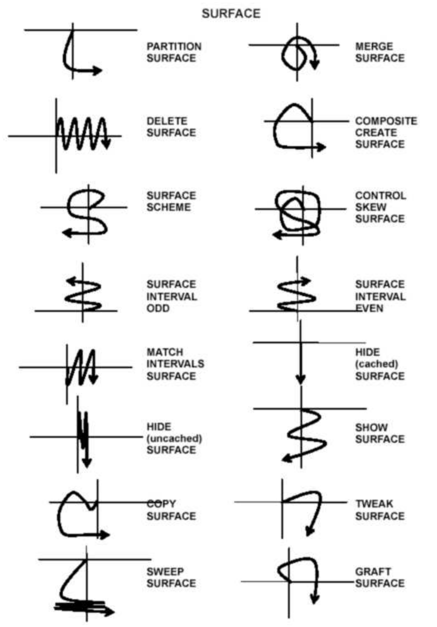


Figure 11: Surface operations.

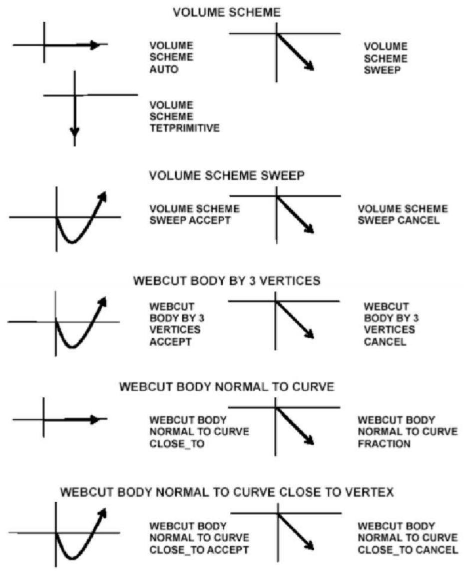

VOLUME SCHEME
VOLUME SCHEME AUTO
VOLUME SCHEME SWEEP
VOLUME SCHEME TETPRIMITIVE
VOLUME SCHEME SWEEP
VOLUME SCHEME SWEEP ACCEPT
VOLUME SCHEME SWEEP CANCEL
WEBCUT BODY BY 3 VERTICES
WEBCUT BODY BY 3 VERTICES ACCEPT
WEBCUT BODY BY 3 VERTICES CANCEL
WEBCUT BODY NORMAL TO CURVE
WEBCUT BODY NORMAL TO CURVE CLOSE_TO
WEBCUT BODY NORMAL TO CURVE FRACTION
WEBCUT BODY NORMAL TO CURVE CLOSE TO VERTEX
WEBCUT BODY NORMAL TO CURVE CLOSE_TO ACCEPT
WEBCUT BODY NORMAL TO CURVE CLOSE_TO CANCEL

Figure 10: Marks that are specific to various modes related to body selections.

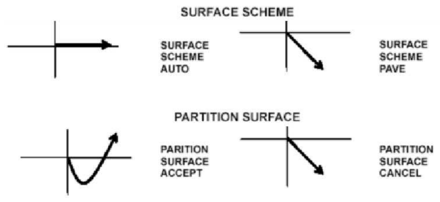


Figure 12: Marks that are specific to modes related to surface selections.