MeshFlow: Interactive Visualization of Mesh Construction Sequences

Jonathan D. Denning* William B. Kerr* Fabio Pellacini*†
*Dartmouth College †Sapienza University of Rome

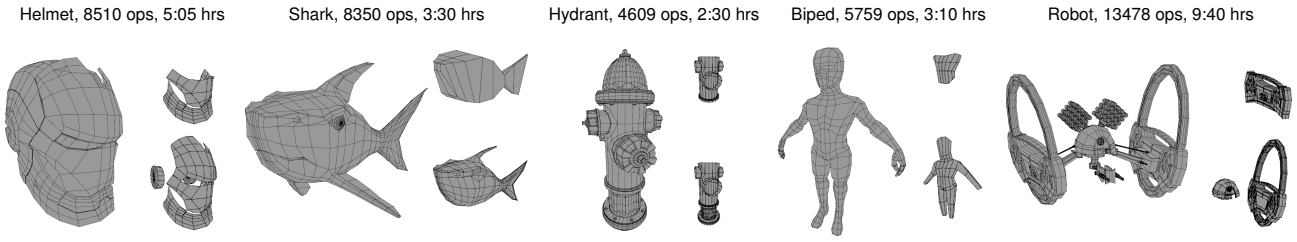

Figure 1: *Five input models, number of operations in construction history, and approximate time to complete.*

Abstract

The construction of polygonal meshes remains a complex task in Computer Graphics, taking tens of thousands of individual operations over several hours of modeling time. The complexity of modeling in terms of number of operations and time makes it difficult for artists to understand all details of how meshes are constructed. We present *MeshFlow*, an interactive system for visualizing mesh construction sequences. *MeshFlow* hierarchically clusters mesh editing operations to provide viewers with an overview of the model construction while still allowing them to view more details on demand. We base our clustering on an analysis of the frequency of repeated operations and implement it using substituting regular expressions. By filtering operations based on either their type or which vertices they affect, *MeshFlow* also ensures that viewers can interactively focus on the relevant parts of the modeling process. Automatically generated graphical annotations visualize the clustered operations. We have tested *MeshFlow* by visualizing five mesh sequences each taking a few hours to model, and we found it to work well for all. We have also evaluated *MeshFlow* with a case study using modeling students. We conclude that our system provides useful visualizations that are found to be more helpful than video or document-form instructions in understanding mesh construction.

1 Introduction

Mesh Construction. For many applications in Computer Graphics the shape of objects is represented as polygonal meshes, either rendered directly or as subdivision surfaces. In most cases, these meshes are modeled by designers using polygonal modeling packages, such as Maya, 3ds Max [Autodesk 2011], or Blender [Blender Foundation 2011]. Even for relatively simple shapes, such as the ones shown in Fig. 1, the construction of polygonal meshes remains a complex task, taking tens of thousands of individual operations over several hours of modeling time. The complexity of the modeling tasks in terms of number of operations and time makes it difficult for artists to understand all details of how meshes they did not build are constructed.

Without access to an instructor, it is common to use tutorials in either video or document format, e.g., from a book or website. For mesh construction, both of these formats have severe drawbacks. On the one hand, a video tutorial contains all the necessary details to construct the mesh, but long recording time (several hours) makes it hard to get an overview of the whole process. On the other hand, a carefully prepared document provides a good overview of the whole process, but skips many details that are necessary for correct construction.

MeshFlow. In this paper we present *MeshFlow*, a system for the interactive visualization of mesh construction sequences. These sequences are obtained by instrumenting a modeling program, in our case Blender, to record all operations performed by a modeler during mesh construction. In its simplest form, *MeshFlow* can be used to play back every operation made by the modeler, similarly to a video, while allowing the viewer to control the camera. The real strength of our system, though, is a hierarchical clustering of the construction sequence that groups similar operations together at different levels of detail. We motivate our clustering by an analysis of the frequency of repeated operations found in mesh construction sequences. To visualize the clustered operations, we introduce graphical annotations that we overlay on the model. Figure 2 shows examples of annotated clustered operations for the mesh sequences used to create the models in Figure 1.

In *MeshFlow*, the top level clusters provide an *overview* of the construction process, while the ability to change the level of *detail on demand*, all the way down to individual operations, ensures that viewer has all the information needed to reproduce the model exactly. Furthermore, we allow the viewer to *focus* on specific aspects of the construction process by filtering operations based on either their type or which parts of the model they affect.

Contributions. We believe that by combining automatically generated annotations with the functionality for overview, detail-on-demand, and focus, *MeshFlow* has the benefits of both video and document tutorials. We have validated this intuition by asking eight subjects to compare *MeshFlow* with traditional tutorials, finding that our tool is highly preferable. To the best of our knowledge, *MeshFlow* is the first system to support this type of interactive visualization of mesh construction sequences.

2 Related Work

Design-workflow Visualization. Our system for interactively visualizing mesh construction sequences is inspired by several re-

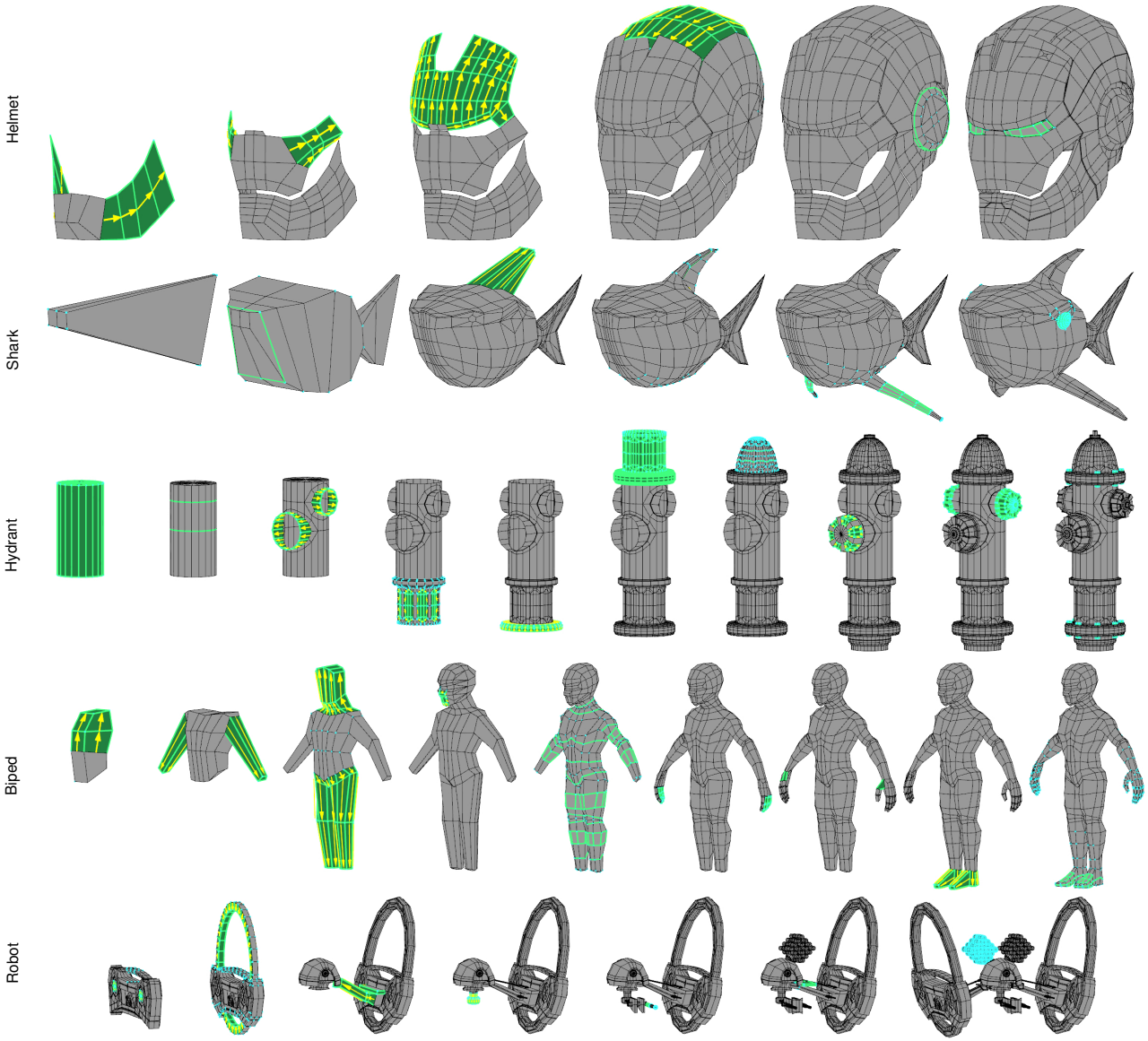

Figure 2: *Subset of clusters with annotations from level 10 for helmet, hydrant, biped, and robot; level 9 for shark. Green highlights indicate new, constructed geometry. Blue highlights indicate translated vertices. Yellow arrows indicate direction of extrusion.*

cent works on visualizing designers' workflow. VisTrails [VisTrails 2010], the closest system to our work, is a workflow provenance system. The system records actions performed in the application, displaying states as nodes in a graph, and allows the viewer to jump to any state in the workflow history (similar to an undo). Changes made to a previous state creates a version branch, and navigating the history involves traversing a version tree. However, when a single version grows deeper than a few hundred edits, exploring the branch becomes similar to searching a long video sequence. In *MeshFlow*, we assume that undos are performed to correct mistakes, and we concentrate on a specific aspect of the provenance visualization problem: the practical and effective visualization of long sequences of editing actions through hierarchical clustering. In future work it would be interesting to combine their model version branching with our hierarchical operation clustering. Additionally, *MeshFlow* annotates the mesh by the edits performed. [Grabler et al. 2009] have developed a system to automatically generate a photo manipulation tutorial directly from the recorded steps of the artist. The system was designed to handle sequences of operations that are orders of magnitude shorter than ours. While parameter tuning and repeated operations are grouped into single steps, long sequences of different operations are not grouped. [Grossman et al. 2010] describe an interactive system for visualizing and exploring long image editing histories. While their system is scalable to record and navigate several hours of work, the exploration of the edit sequence involves using a detailed timeline and before-and-after thumbnails, delimited first by save-times and then by edit-times. While this is effective for image manipulations, we found instead that for mesh modeling sequences clustering is necessary to provide a clear overview. Visualizing workflows is a well-explored topic in HCI research [Bergman et al. 2005; Berlage 1994; Kurlander and Feiner 1989; Nakamura and Igarashi 2008; Su et al. 2009]. Because they focus on a smaller number of individual steps rather than summarizing long sequences, these methods are not well-suited for very long sequences as navigation becomes difficult.

Summarizing Video Sequences. There is a large body of work on finding and visualizing a small set of representative keyframes for a video sequence [Assa et al. 2005; Barnes et al. 2010; Christel

Model	Vertices	Time	Operations					
			total	*view*	*select*	*trans*	*topo*$_a$	*topo*$_b$
Helmet	1342	5h05m	8510	4941	2020	1264	126	64
Shark	940	3h30m	8350	4668	1986	1563	61	51
Hydrant	10435	2h30m	4609	2430	1364	519	157	84
Biped	564	3h10m	5759	2741	1669	1236	60	31
Robot	16081	9h40m	13478	8296	2877	1648	347	151

Legend	
cam	Camera changes
vis	Visibility changes
$view$	cam or vis
$select$	Selection operations
$trans$	Transformation operations
$topo_a$	Loopcut, Subdivide, Extrude, Delete
$topo_b$	Add Edge/Face, Merge Vertices/Triangles

Table 1: *Input data statistics. This table breaks down the construction statistics of the five models visualized by our system.*

et al. 1998; Kang et al. 2006]. These approaches use image analysis and optimization to determine keyframes that are semantically important and should be present in the summary. In *MeshFlow* we take a different approach and summarize mesh sequences by only analyzing operation tags. We plan to extend our system to include geometry analysis to reap some of the benefits of the summaries presented in these works.

Tutorials. [Palmiter and Elkerton 1991; Harrison 1995] have shown that image-based tutorials are far more effective than video-based instructions, due to the fact that users are able to work at their own pace. [Narayanan and Hegarty 2002] report that the structure and content of instructional materials are important for learning and understanding. [Kelleher and Pausch 2005] has shown that graphical overlays help with focus and reduce confusion. Many of these previous studies focus on relatively short design tasks. In *MeshFlow*, we focus on design tasks that take several hours to compute. In our domain, we found that video and document tutorials fundamentally work at different levels of detail and each have strong benefits but significant drawbacks. In *MeshFlow*, we let the viewer choose the level of detail interactively to capture the benefits while avoiding the drawbacks. For a more in-depth comparison, refer to Section 6.

Complex Model Visualization. Many recent papers show how to effectively explore a complex model by showing how parts relate spatially and interactively to one another in the finished model. In order to focus on a particular part of a model, [Li et al. 2007] cuts into and hides parts that occlude, while [Li et al. 2004] splits and separates sections. [Mitra et al. 2010] visualize models of mechanical assemblies by indicating motions with annotations and causal chains. While all of these approaches isolate parts in a finished model, *MeshFlow* focuses on visualizing the temporal construction. For future work, we would be interested in combining these techniques with our work.

3 Mesh Construction Sequences

Data Capture. The input to our visualization system is a mesh construction sequence, where each step is defined by a polygonal mesh, a tag that indicates the operation performed by the modeler, the current camera view and the current selection. In our sequences, we capture a step for each operation that changes the mesh, its per-component visibility, the viewing camera, or the mesh's per-component selection. We store the mesh as a list of vertices, uniquely labeled, defining its geometry and a list of faces represented as vertex lists.

We record this sequence by instrumenting Blender [Blender Foundation 2011], an open source animation package, comparable, with regard to polygonal modeling, to commercial systems such as Maya or 3ds Max [Autodesk 2011] Our mesh construction sequences are generated automatically while the modeler is building a mesh; this is in contrast to tutorials that need to be authored after the modeler has built the mesh. We supply our instrumentation as supplemental material.

Mesh Sequences. While our data capture works for any mesh, we focus on visualizing mesh construction histories of single objects, rather than full scenes. To demonstrate the usefulness of *MeshFlow*, we recorded the construction of five meshes, shown in Fig. 1. Figure 2 shows a few steps of the construction process annotated by our system. We built the models using tutorials found on the web. The helmet and shark models were based on document tutorials [Jack 2011; Drincic 2004]; the hydrant, biped, and robot models were based on video tutorials [Tate 2009; Williamson 2010; Culum 2009]. Three different modeling "techniques" were used: box modeling (where a single mesh is subdivided to add detail), surface extrusion (where the surface is grown using successive extrusions), and modeling by parts (where individual components are modeled separately). All five sequences are supplied as supplemental material.

Table 1 shows various statistics for each of our models. Note how even for these simple models, hours of modeling time was employed. This is due to the need for several thousands of operations to construct the meshes, even in cases where only half of the model is built due to symmetry. The construction process is traditionally documented by video recordings or documents with textual explanations and images. When the process takes many hours, a video recording becomes tedious and difficult to search, for a viewer. It can be useful to condense this information into a document with illustrations, but even with considerable work in authorship, details will be selected and aggregated in a static way.

Operations in the modeling sequence range from user interface commands, to geometric transformations, to topological changes in the mesh. We define five groups of operation types, listed in Table 1: $view$ for operations that either change the camera (cam) or hide/show geometry (vis), $select$ for operations where geometry components are selected to be modified, $trans$ for translation, rotation, or scaling transformations, $topo_a$ for the topological operations of loopcut, subdivision, extrusion, and deletion, and $topo_b$ for the topological operations of add edge, add face, merge vertices, and merge triangles. We split topological operations into $topo_a$ and $topo_b$ because $topo_b$ operations are typically used as patchwork edits in conjunction with members of $topo_a$. We include a third type, $topo_c$, for the creation of disjoint geometric primitives such as creating spheres and boxes, but these operations are very uncommon compared to the others.

To gain the benefits of video and document tutorials without their drawbacks, we need to provide a way to view modeling sequences at different levels of detail. Our analysis of construction sequences on the five models revealed a great deal of repetition within and between operation types (see Table 1 and Sec. 5 for an analysis). We use this repetition to hierarchically group operations into clusters, from a high-level overview of the modeling process, all the way down to the individual low-level operations needed for reproducing the mesh exactly. We allow users to interactively choose the desired levels of detail gaining the benefits of both overview and detail-on-demand.

	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6	Level 7	Level 8
Helmet	*cam cam* (46)	*select select* (18)	*select select* (19)	*select trans* (22)	*trans trans* (40)	$topo_a$ *trans* (18)	$topo_a$ *trans* (21)	*cam* $topo_a$ (24)
	select select (15)	*select trans* (16)	*select trans* (17)	*trans select* (18)	*trans cam* (23)	*trans cam* (14)	*trans cam* (16)	$topo_a$ *cam* (24)
	select trans (13)	*trans select* (13)	*trans select* (13)	*trans cam* (14)	*cam trans* (21)	*cam* $topo_a$ (14)	*cam* $topo_a$ (15)	$topo_a$ $topo_a$ (18)
	trans select (11)	*cam select* (12)	*cam select* (12)	*cam select* (12)	$topo_a$ *trans* (4)	*trans* $topo_a$ (11)	*trans* $topo_a$ (13)	$topo_a$ $topo_b$ (7)
Shark	*cam cam* (45)	*select trans* (22)	*select trans* (22)	*select trans* (27)	*trans trans* (45)	*trans cam* (18)	*trans cam* (21)	*cam* $topo_a$ (29)
	select trans (12)	*trans select* (16)	*trans select* (16)	*trans select* (20)	*trans cam* (24)	*cam* $topo_a$ (16)	*cam* $topo_a$ (18)	$topo_a$ *cam* (26)
	trans select (9)	*cam select* (13)	*cam select* (14)	*trans cam* (14)	*cam trans* (23)	*cam trans* (12)	*cam trans* (14)	$topo_b$ *cam* (12)
	cam select (7)	*select select* (12)	*select select* (12)	*cam select* (13)	*cam* $topo_a$ (2)	$topo_a$ *trans* (10)	$topo_a$ *trans* (12)	*cam* $topo_b$ (9)
Hydrant	*cam cam* (43)	*select select* (32)	*select select* (33)	*select trans* (16)	*trans trans* (30)	*cam* $topo_a$ (15)	*trans cam* (17)	$topo_a$ *cam* (26)
	select select (18)	*select trans* (9)	*select trans* (9)	*trans select* (11)	*trans cam* (15)	*trans cam* (12)	$topo_a$ *trans* (16)	*cam* $topo_a$ (24)
	select trans (5)	*cam select* (9)	*cam select* (9)	*trans cam* (11)	*cam trans* (12)	$topo_a$ *trans* (11)	*cam* $topo_a$ (15)	$topo_a$ $topo_a$ (14)
	cam select (5)	*trans select* (6)	*trans select* (7)	*trans trans* (11)	*cam* $topo_a$ (7)	$topo_b$ $topo_b$ (10)	*trans* $topo_a$ (10)	*cam* $topo_c$ (10)
Biped	*cam cam* (33)	*select trans* (22)	*select trans* (22)	*select trans* (27)	*trans trans* (40)	*trans cam* (17)	*trans cam* (20)	*cam* $topo_a$ (30)
	select trans (15)	*trans select* (16)	*trans select* (16)	*trans select* (20)	*trans cam* (26)	*cam* $topo_a$ (15)	*cam* $topo_a$ (17)	$topo_a$ *cam* (29)
	trans select (11)	*cam select* (13)	*cam select* (14)	*trans cam* (15)	*cam trans* (25)	*cam trans* (14)	*cam trans* (16)	$topo_a$ $topo_a$ (9)
	cam select (9)	*select select* (13)	*select select* (13)	*cam select* (14)	*cam* $topo_a$ (2)	$topo_a$ *trans* (11)	$topo_a$ *trans* (12)	$topo_b$ *cam* (7)
Robot	*cam cam* (48)	*select select* (21)	*select select* (22)	*trans cam* (15)	*trans trans* (28)	*cam* $topo_a$ (16)	*trans cam* (18)	*cam* $topo_a$ (28)
	select select (11)	*cam select* (14)	*cam select* (14)	*trans trans* (15)	*trans cam* (22)	*trans cam* (15)	*cam* $topo_a$ (18)	$topo_a$ *cam* (28)
	cam select (7)	*select cam* (10)	*select cam* (10)	*cam select* (13)	*cam trans* (19)	$topo_a$ *trans* (14)	$topo_a$ *trans* (18)	$topo_a$ $topo_a$ (15)
	select cam (5)	*trans cam* (10)	*trans cam* (10)	*cam trans* (10)	*cam* $topo_a$ (6)	$topo_a$ *cam* (9)	$topo_a$ *cam* (10)	*cam* $topo_c$ (6)

Table 2: *Top four bigrams for each of the models at levels 1–8 (all levels available in supplemental material). Numbers in parenthesis indicate percentage of all bigrams from the sequence at that level.*

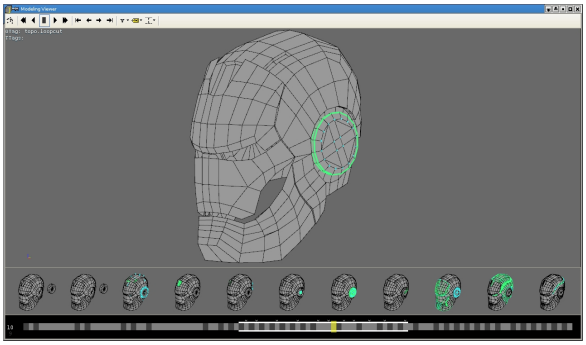

Figure 3: *User interface. A large view shows the mesh of the current cluster. Across the bottom is the timeline with indicators of the current cluster and any filtered clusters. The thumbnails show changes at different places along the timeline.*

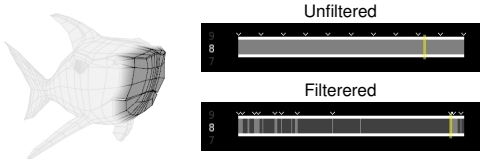

Figure 4: *Shark model with snout highlighted and the corresponding timeline with and without filtering. To focus on edits affecting only specific regions of the mesh, the viewer highlights the areas of interest, and the timeline is filtered to show the clusters that modify these areas.*

4 *MeshFlow*

Visualization System. In this section we describe briefly our visualization system from a user perspective. We suggest that the reader consult our video for a demonstration of the various concepts listed here. *MeshFlow* provides an interface for interactively exploring the mesh construction history. The interface includes a large view of the mesh, a timeline, and thumbnail views of the mesh at different places along the timeline. Fig. 3 shows a screenshot of our user interface. In its simplest form, the visualization system can be used to play back every operation made by the modeler, similarly to a video, except the viewer can control the camera in addition to using the original modeler's camera views.

Operation Clustering. The real strength of our system compared to traditional recordings comes from the use of interactive level of detail through operation clustering. Our approach differs from [Grabler et al. 2009; Nakamura and Igarashi 2008] in that we group operations together in a hierarchical fashion, where lower cluster levels have more details than higher ones. This allows us to get a visual summary of high-level changes in the mesh, while providing several levels of detail that can be accessed on demand. By changing the level of detail, a viewer can choose to see a summary of the edits or get details on demand. The timeline at the bottom of the interface is discretized into clusters, such that only one cluster is viewable at a time. The main view of our interface displays the resulting mesh from the clustered operations viewed from the average camera location (or a user controlled camera if so desired). To determine our clustering, we analyzed the recurrance of patterns of operations in the input sequence and found that clustering based solely on operation tags works well, without requiring geometric analysis. Section 5 covers our clustering methodology in detail.

Visual Annotations. We added graphical annotations to illustrate the types of operations that were performed in a cluster, which can be seen in Fig. 2, similar to [Grabler et al. 2009; Su et al. 2009]. These annotations color vertices, edges, and faces of the mesh to indicate mesh changes like adding topology (green), moving vertices (blue), and selection (orange). We further add annotations to indicate common operations such as arrows for extrusion and lines for loop cuts. Selection is usually active in many places on the mesh, so we allow it to be turned on and off when necessary to reduce clutter. The main view includes annotations indicating all operations performed in the current cluster. The thumbnails contain annotations indicating changes since the previous thumbnail, emphasizing modifications as in the timeline at that location. Section 5 covers these annotations in detail.

Filtering. We have found it useful to be able to focus quickly on subsets of operations. To achieve this we give viewers the ability to filter operations and clusters. This can be important for speeding up the viewing process, but also for visualizing how operations group over time and at what frequency. When a filter is activated, all clusters that match the filter are darkened in the timeline (see Fig. 4), made unselectable, and skipped during playback. We support two main filtering modes. First, filtering by operation type allows for operations and clusters tagged with that type (selection, transform, etc.) to be easily identified and skipped. This allows for focusing

	Clustering Regular Expressions
2	$(cam)+\ (cam)^{\diamond} \mapsto (cam)^{\diamond}$
3	$(view)\ (view)+ \mapsto (cam)$
4	$(select)\ (view\|select)*\ (select)^{\diamond} \mapsto (select)^{\diamond}$
5	$(select)\ (view)*\ (topo\|trans)^{\diamond} \mapsto (\cdot)^{\diamond}$
6	$(trans)+\ (view)*\ (trans)^{\diamond} \mapsto (\cdot)^{\diamond}$
7	$(\cdot)^{\diamond}\ (view\|(\cdot)^{\diamond})*\ (\cdot)^{\diamond} \mapsto (\cdot)^{\diamond}$
8	$(topo)^{\diamond}\ (view\|trans)*\ (trans) \mapsto (\cdot)^{\diamond}$
9	$(topo_a)^{\diamond}\ (view\|topo_b)*\ (topo_b) \mapsto (\cdot)^{\diamond}$
10	$(\cdot)^{\diamond}\ (view\|(\cdot)^{\diamond})*\ (\cdot)^{\diamond} \mapsto (\cdot)^{\diamond}$

Table 3: *Regular expressions used to generate levels 2 to 10. For each level the group of elements that matches the regular expression is replaced with a single cluster.* Legend: $*$ *and* $+$ *match 0-or-more and 1-or-more repetitions respectively;* $(\cdot)$ *matches anything;* $(a|b)$ *matches either* a *or* b*;* $\diamond$ *indicates a back-reference group.*

on different "techniques" used when modeling. Second, inspired by the Data Probe in [Grossman et al. 2010], filtering by vertex selection allows the viewer to highlight vertices and skip clusters that do not affect those vertices. This allows the viewer to focus on how specific parts of the model are built in their entirety. For geometry filtering, we further highlight the region of interest by deemphasizing the remainder of the model (see Fig. 4). Our system will automatically tag data during capture, but both modelers and viewers can provide their own custom tags, e.g., tagging operations spatially with labels like "torso" or "wheel", or temporally with labels like "blocking phase" or "refinement phase".

5 Operation Clustering

Clustering by Regular Expressions. The mesh sequences described in Sec. 3 contain a great deal of repeated operations. In order to provide a clear overview of how the model is built we need to group low-level operations into clusters representing high-level structural changes. To identify such groups, one might attempt to analyze geometric properties to learn when large semantic changes to the mesh have occurred. We have discovered, though, that clustering based solely on operation tags can establish meaningful levels of detail without attempting to learn semantics within the sequence (see Sec. 6). To group operations together, we apply substituting regular expressions defined on the operation tags. We derive these regular expressions by identifying repeated patterns of operations and combine them into clusters that can be visualized at once. As two examples, selections and vertex transformations are often achieved by many repeated atomic selection or transform operations. We can cluster these into a single cluster representing the net change in selection state or vertex locations.

To create a hierarchy of detail levels, we apply successive regular expression substitutions and let the user interactively choose the displayed level. In our implementation we provide 11 successive levels of detail. Table 3 shows a list of regular expressions used for each level of detail. Figure 5 shows an example of executing the regular expressions at different levels of detail. In the latter example, we show start and end states of a group of repeated extrusions and vertex movements, and then see each separate extrusion without viewing every individual selection and transform of vertices.

Removing Undos. The original sequence, called *Level 0*, will contain all operations a modeler has performed, including work that is undone. We assume that undos are used to correct mistakes, rather than used for exploration purposes. Our first cluster level, referred to as *Level 1*, cleans up the data stream by removing undone work. We look in the stream for identical mesh states and remove all operations in between, effectively making undos invisible.

Initial Clustering. To choose regular expressions that represent effective levels of detail, we analyze the mesh sequences for our data set. We measure the frequency of bigrams, or instances of pairs of operation types. Table 2 lists the four most frequent bigrams for cluster levels 1–8. Note that after we cluster undos (*Level 1*), repeated camera changes are the most frequent, roughly half of all bigrams in some cases, followed by repeated selections. Repeated camera movements likely come from the artists either viewing the model from different angles or simply adjusting the view carefully. Visibility operations (show/hide geometry), albeit not as frequent, are similarly motivated. Note also that repeated adjustments to display the mesh do not alter the mesh. For repeated selections, it is likely that the modeler was building up a large selection set for a successive operation, thus we can safely group them together. Similarly to view changes, these also do not alter the mesh. These observations motivate the next three levels of clustering.

In *Level 2* we replace repeated camera view changes with a single view cluster, picking the last camera view as the cluster view. *Level 3* clusters all repeated visibility and camera operations together. Visibility is clustered at this level for semantic reasons rather than a bigram frequency because it forms clusters affecting only view operations. We then cluster repeated selections together, in *Level 4*, as this is a highly common bigram and since this likely prepares larger selections for successive operations. We set the selection of the resulting cluster as the net result of the successive selections. At this point we have clustered together all operations that do not affect the mesh.

Clustering Editing Operations. After *Level 4* we begin to cluster operations that alter the mesh. At this point, transforms that follow selections are the most frequent bigrams on our sequences. This makes sense, since something must be selected to be edited. Thus, in *Level 5*, we cluster selection with the subsequent editing operation. The next most common bigram is repeated transformation. The combined effect of repeated translation, rotation, and scaling operations can be thought of as simply modifying the positions of vertices. We can cluster these together in *Level 6* such that the resulting vertex positions are the net change in position. Now we have another situation where semantics outweigh our bigram analysis. We take this opportunity to create a level of detail that clusters all repeated operations no matter what they are (essentially cleaning up repeated homogeneous topology changes), forming *Level 7*. In practice, we found this to be an effective level of detail with easily recognizable meaning. Note that topology operations are only clustered if they have the same tag, e.g., extrude with extrude, not extrude with loopcut.

Clustering Groups of Editing Operations. So far we have clustered together editing operations of the same type. We will now combine these clusters with each other to form higher level groups of operations with more heterogeneity. The most common bigram in *Level 7* is topology operations followed by transformations. This makes sense, since new topology is often shaped after being created. In *Level 8* we cluster topology changes with any subsequent transform cluster, combining, in most cases, the creation of new geometry with the shaping of that geometry. A good example of this is seen in *Level 8* of Figure 5.

Until now we have been thinking of topological operations together, but we now introduce the classification of types $topo_a$ and $topo_b$ (see Sec. 3 and Table 1). Operations in $topo_a$ represent major structural change to the mesh, often changing the number of edge loops or overall complexity, whereas $topo_b$ operations are used as patchwork in conjunction with $topo_a$ operations, filling holes and cracks by merging or connecting things. For example, on the crown of the helmet each edge loop is extruded and then attached to the head before starting the next extrusion. *Level 9* clusters instances of $topo_a$ with subsequent $topo_b$ operations. Finally, in *Level 10*, we cluster

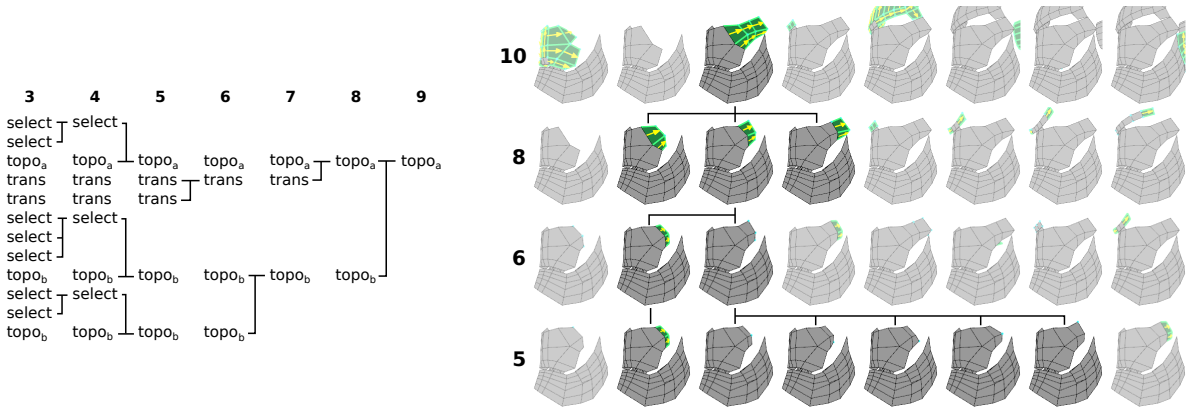

Figure 5: *Two examples of successively applying levels of clustering. The left figure shows the operation names for levels 3–9, while right figure shows screenshots of the model for levels 5, 6, 8, and 10. See Table 3 for clustering rules.*

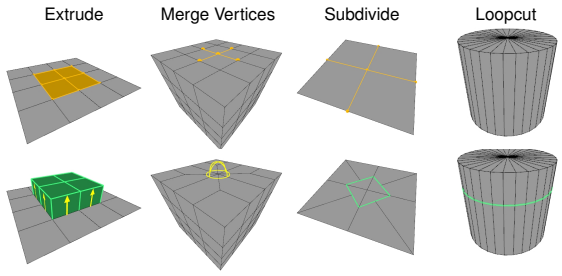

Figure 6: *Various automatically-generated annotatians. For illustrative purposes, the top row has selections drawn; the bottom row does not.*

repeated instances of the case from *Level 9*, visualizing large components of the mesh being constructed all at once. Depending on the model, *Level 9* or *Level 10* yields a concise overview that is easily visualized in a matter of seconds (see supplemental video). Though heterogeneous $topo_a$ pairs are our most common non-camera bigram past *Level 8*, we do not combine them here, because we find that this causes semantically ambiguous situations and unclear level of detail.

Visual Annotations. When drawing the mesh corresponding to each cluster, we highlight changes performed in the cluster to draw user attention. We use color coding to indicate simple changes: green for added geometry, cyan the transformed vertices, and orange for selection. For the most common topology operations, we overlay visual annotations on the resulting mesh to indicate what operations types are performed in each cluster. Figure 6 shows a summary of such annotations. We annotated *extrusions* by drawing yellow arrows on both sides of newly created faces. For *subdivisions* and *loopcuts*, the edges involved are highlighted in green. For vertex or face *merge* operation, we draw yellow circles at the location of the final vertex or face respectively.

6 Evaluation

Overview. We run our system on an Intel Core2 3.0GHz quad-core processor with 4GB of RAM, and an NVIDIA GeForce 9600GT GPU. On this system, exploring mesh sequences on all meshes in our dataset is interactive. We provide all source code and mesh sequence data files to allow readers to experience our visualization. We also include our Blender instrumentation. All visualization features and annotations shown in the paper and supplemental videos are automatically generated by our system with no authoring overhead for the modeler. We found that our regular expression grouping consistently works very well in reducing sequence complexity. Table 4 shows the number of operations at each level of detail, going from several thousands operations to just hundreds. This supports our claim that a simple frequency analysis of the operation is sufficient for reduction. We include videos showing three levels of detail for each mesh as supplemental materials, as well as an overview of the interface in our video submission. In the next subsection we will introduce a case study that supports our claim that these operation reductions are effective in aiding understanding for viewers.

Limitations. The most obvious limitation of *MeshFlow* is that it focuses solely on polygonal meshes, whereas other surface representations are also useful. It is our belief, though, that the majority of *MeshFlow* can be extended to support other surface representations such as NURBS. The primary limitation in performing such extension is that our clustering algorithm would need to address the presence of new operators, specific to modeling other geometric representations. We are confident, though, that this can be accomplished by analyzing operation frequencies and following our methodology. Additionally, the sequences used for analysis contained only a subset of the operations available in Blender. While this subset was able to construct a variety of models, in future work we would like to explore sequences containing operations from other modeling styles, such as sculpting. For polygonal meshes, our clustering based on only regular expressions could be improved. First, we only support clustering expressions sequentially, but it could be useful to investigate methods to cluster operations out of order to better highlight patterns on different parts of the model. Second, we made no attempt to determine what clusters have more semantic importance when editing a mesh. This would require some form of geometry analysis that could quantify the importance of mesh changes. Third, it would useful to be able to recognize parts of the model to create even higher level clusters. For example if we could recognize that a set of vertices is modeling the nose (rather than the eyes), we could automatically cluster all those together; this was done for images in [Grabler et al. 2009] using face recognition. Such semantics would allow us to automatically generate audio and text annotations. Last, because it is a completely automated system, *MeshFlow* is not a replacement for hand-authored tutorials. However, *MeshFlow* can be easily extended to allow author-specified hints and tips using the tagging metaphor. (see Filtering in Sect. 4) While we are interested in addressing these limitations in future work, the following section will show that artists found our current system very useful and a significant improvement over available methods.

Model/Level	0	1	2	3	4	5	6	7	8	9	10
Helmet	8510	8203	4274	4235	3190	1912	381	335	212	183	108
Shark	8350	8303	4587	4567	3762	2245	252	217	133	100	61
Hydrant	4609	4496	2579	2542	1483	1034	528	361	227	214	124
Biped	5759	5704	3826	3781	3118	1843	252	225	129	115	58
Robot	13478	13137	6809	6639	4321	3073	1247	998	639	596	326

Table 4: *Number of clusters for five models at each level of detail.*

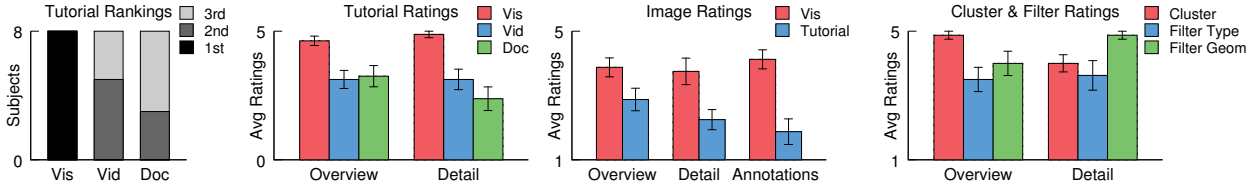

Figure 7: *Data from our case study. From left to right: preference rankings for* MeshFlow *(vis) compared to traditional video (vid) and document (doc) tutorials; ratings for* MeshFlow *compared to traditional tutorials for overview and detail usefulness; ratings for* MeshFlow *images compared to authored tutorial images for overview, detail, and graphical annotation usefulness; ratings for clustering and filtering features for overview and detail usefulness. Error bars represent standard error.*

6.1 Case Study

We conducted a case study in which subjects were asked to evaluate *MeshFlow* compared to video and document tutorials. The study included 8 college modeling students, all of whom had previously completed at least one course in mesh modeling and had experience in creating models by following tutorials. When asked to rate their confidence level in completing a mesh they had never tried before using a tutorial, all but one rated themselves 4 or higher on a scale of 1 to 5, with the other rating a 3. We are confident that all subjects have enough experience to put *MeshFlow* in context with real modeling tasks.

Methodology. We ask our subjects to make five comparisons of using *MeshFlow* to other options. For each comparison, subjects have 10 minutes to investigate a modeling sequence using *MeshFlow* and 10 minutes to investigate using the alternative. We guide the exploration by asking subjects to answer three specific questions about the modeling sequence (e.g., for the robot model, subjects were asked how the wheel was made to fit into the chassis). The investigator introduces the questions before the subject begins and remains on hand to guide the subject in using the interface. At the end of all five comparisons, we ask subjects to rate various aspect of *MeshFlow* and leave open comments regarding different aspects of the experience. Scanned questionnaires are supplied with supplemental materials.

First, we compare *MeshFlow* to traditional modeling tutorials for three of our models to determine whether *MeshFlow* is in fact effective as a visualization tool. We compare the helmet model in *MeshFlow* to its original document tutorial [Jack 2011]. We then compare the biped model in *MeshFlow* to the original video tutorial [Williamson 2010]. Finally, for the shark model, we compare just still screenshots automatically generated by *MeshFlow* at level 9 to authored images taken from the original tutorial [Drincic 2004] with the text removed. For each of these comparisons, half the subjects where shown *MeshFlow* first, and the other half were shown the traditional tutorial first.

Second, we compare the use of *MeshFlow* with and without the ability to filter and cluster operations, to evaluate the relative importance of these features in model sequence exploration. We use the robot and hydrant models for these comparisons. First, the subject is given the model at the lowest level of detail and asked to answer our questions without using clustering or filtering. We then allow the user all clustering levels of detail and filtering methods to compare.

Results. Figure 7 summarizes the ratings of subjects for the comparisons performed. In general, subjects are very enthusiastic about *MeshFlow*, and rate its features highly. We ask subjects to compare *MeshFlow* to video and document tutorials by rating the usefulness of each with respect to getting a general overview and in understanding details. Subjects rate *MeshFlow* superior to video and document tutorials in each of the two categories. This shows that *MeshFlow* not only has the benefits of traditional tutorials, but it outclasses them even in the area of their individual strengths. We also ask subjects to strictly rank their general preference between the three (*MeshFlow*, video, document). All subjects ranked *MeshFlow* as their preferred method. We also ask subjects to rate the set of images automatically generated by *MeshFlow* compared to the ones manually created for a document tutorial. Subjects rate each with respect to how useful they are in understanding an overview, the modeling details, as well as the clarity of the annotations. *MeshFlow* was rated much higher in all categories, with only one subject rating it lower than the tutorial images. We found this to be surprising, since *MeshFlow* was not designed to generate static image sequences, but interactive visualizations. Still, when comparing the automatically annotated clusters to hand-authored images, *MeshFlow* was found to be superior. Finally, we ask the subjects to rate the usefulness of clustering and filtering when trying to understand overview and details. For the most part, subjects rate all features high, indicating that clustering is the most useful feature for getting overviews, and filtering on specific vertices is the most useful for investigating details.

Observation. To support our previous analysis, we collected open feedback from subjects' questionnaires, and now report the following quotes. All subjects preferred *MeshFlow* over the traditional alternatives. When asked why, one responded with "the ability to customarily look at parts of the geometry and changes to it that I was interested in, rather than being dependent on what the tutorial author thought I would want to to know." And another subject, "the interactive vis gives me the option of the level of detail. [...] It has more detail than a document and can leave out irrelevant detail that a video often comes with." When comparing the *MeshFlow* images to hand-authored ones, "I thought that the interactive vis better explained how the model was built. I liked the color scheme / familiar interface, as well as the ability to easily distinguish/identify what was being altered." And another, "the graphical annotation [in *MeshFlow*] says much more than a normal tutorial." Regarding the ability to filter, many subjects found this useful, commenting "the painting tool which then shows you where changes pertaining to that which was selected on the timeline is a fantastic time saver

if you're focused on a detail". And another subject, "filtering by selected parts seemed very useful. Definitely fixed problem of having to guess or remember where in a tutorial or video a certain area is worked on." In terms of clustering, we found that subjects all had different interests, highlighting the importance of choosing the level of detail interactively. For example, one subject commented "clustering is key to finding the parts that you want to focus on" and another subject "clustering gives a good, rapid overview of the build," and yet another, "there are times when a general view is more helpful (clustering) and also times when a more detailed view is preferred (filtering). What sets the interactive vis apart is the ability to cater to both needs at any time." At least three of the subjects asked us after the study if we were going to release *MeshFlow* to the public, so they could start using it. One even wrote in the questionnaire "I would love to use this interactive vis tutorial in a digital arts modeling class. Though I suppose with it, the professor would not need to do much."

7 Conclusion

We have presented *MeshFlow*, a system for visualizing the construction process of polygonal meshes. *MeshFlow* combines overview, detail-on-demand, and focus by hierarchically clustering and filtering edits from a recorded modeling session. We based our clustering on an analysis of the frequency of repeated operations and implement it using substituting regular expressions on operations tags. We have tested *MeshFlow* on five mesh sequences and evaluated it with a case study. We conclude that our system provides useful visualizations that are found to be more helpful than video or document-form instructions in understanding mesh construction. For future work, we would like to focus on improving our clustering to support out-of-order grouping, highlight semantical changes and add voice and text annotations.

Acknowledgments

We would like to thank the tutorial authors of [Culum 2009; Drincic 2004; Jack 2011; Tate 2009; Williamson 2010]. This work was supported by NSF (CNS-070820, CCF-0746117), Intel, and the Sloan Foundation.

References

ASSA, J., CASPI, Y., AND COHEN-OR, D. 2005. Action synopsis: pose selection and illustration. *ACM Trans. Graphics*, 667–676.

AUTODESK, 2011. Autodesk 3ds Max and Autodesk Maya. http://www.autodesk.com/.

BARNES, C., GOLDMAN, D. B., SHECHTMAN, E., AND FINKELSTEIN, A. 2010. Video tapestries with continuous temporal zoom. *ACM Trans. Graphics*, 89:1–89:9.

BERGMAN, L., CASTELLI, V., LAU, T., AND OBLINGER, D. 2005. DocWizards: a system for authoring follow-me documentation wizards. In *Proc. ACM UIST*, 191–200.

BERLAGE, T. 1994. A selective undo mechanism for graphical user interfaces based on command objects. *ACM Trans. CHI*, 269–294.

BLENDER FOUNDATION, 2011. Blender. http://www.blender.org/.

CHRISTEL, M. G., SMITH, M. A., TAYLOR, C. R., AND WINKLER, D. B. 1998. Evolving video skims into useful multimedia abstractions. In *Proc. SIGCHI*, 171–178.

CULUM, A., 2009. Hailfire droid. http://www.cgwhat.com/hailfire-droid/.

DRINCIC, N., 2004. [Shark] Modeling Process. http://www.3dm3.com/tutorials/shark/.

GRABLER, F., AGRAWALA, M., LI, W., DONTCHEVA, M., AND IGARASHI, T. 2009. Generating photo manipulation tutorials by demonstration. *ACM Trans. Graphics*, 66:1–66:9.

GROSSMAN, T., MATEJKA, J., AND FITZMAURICE, G. 2010. Chronicle: capture, exploration, and playback of document workflow histories. In *Proc. ACM UIST*, 143–152.

HARRISON, S. M. 1995. A comparison of still, animated, or non-illustrated on-line help with written or spoken instructions in a graphical user interface. In *Proc. SIGCHI*, 82–89.

JACK, B., 2011. Helmet modeling. http://www.bracercom.com/tutorial/content/Ironman_Helmet_Modeling/ironman_helmet_modeling.html.

KANG, H.-W., CHEN, X.-Q., MATSUSHITA, Y., AND TANG, X. 2006. Space-time video montage. In *Proc. IEEE Computer Society Conference on CVPR*, 1331–1338.

KELLEHER, C., AND PAUSCH, R. 2005. Stencils-based tutorials: design and evaluation. In *Proc. SIGCHI*, 541–550.

KURLANDER, D., AND FEINER, S. 1989. A visual language for browsing, undoing, and redoing graphical interface commands. In *Visual Languages and Visual Programming*, S. K. Chang, Ed. Plenum Press, 257–275.

LI, W., AGRAWALA, M., AND SALESIN, D. 2004. Interactive image-based exploded view diagrams. In *Proc. Graphics Interface*, 203–212.

LI, W., RITTER, L., AGRAWALA, M., CURLESS, B., AND SALESIN, D. 2007. Interactive cutaway illustrations of complex 3d models. *ACM Trans. Graphics*, 31:1–31:11.

MITRA, N. J., YANG, Y.-L., YAN, D.-M., LI, W., AND AGRAWALA, M. 2010. Illustrating how mechanical assemblies work. *ACM Trans. Graphics*, 58:1–58:11.

NAKAMURA, T., AND IGARASHI, T. 2008. An application-independent system for visualizing user operation history. In *Proc. ACM UIST*, 23–32.

NARAYANAN, N. H., AND HEGARTY, M. 2002. Multimedia design for communication of dynamic information. *Int. J. Hum.-Comput. Stud.*, 279–315.

PALMITER, S., AND ELKERTON, J. 1991. An evaluation of animated demonstrations of learning computer-based tasks. In *Proc. SIGCHI*, 257–263.

SU, S. L., PARIS, S., ALIAGA, F., SCULL, C., JOHNSON, S., AND DURAND, F. 2009. Interactive visual histories for vector graphics. Tech. rep., Massachusetts Institute of Technology.

TATE, B., 2009. Model a detailed high poly fire hydrant in 3ds max. http://cg.tutsplus.com/tutorials/autodesk-3ds-max/model-a-detailed-high-poly-fire-hydrant-in-3ds-max/.

VISTRAILS, 2010. VisTrails Provenance Explorer for Maya. http://www.vistrails.com/maya.html.

WILLIAMSON, J., 2010. Character modeling in blender. http://cg.tutsplus.com/tutorials/blender/character-modeling-in-blender-basix/.