Visualizing Maya

Aylish Wood
University of Kent
School of Arts
Canterbury, CT2 7UG
United Kingdom
a.wood@kent.ac.uk

Pervasive animation in games, animation, films, numerous websites and stand-alone information interfaces, means that software increasingly influences how we imagine fictions, display information and interact with a digitally mediated world. Given its centrality in creating image, how can a 3D software package be made visible beyond the images seen in cinema, in games, commercials and visualizations? Is there such as thing as an interface language? This paper explores Autodesk® Maya® through a methodology based in visual studies, production culture studies, and science technology studies. The primary materials are moving images (games, films, animations, web sites), production culture materials, training manual and on-line tutorials, and interviews with software users. This diverse and extensive range of materials is approached through a focus on space. As the project remains a work-in-progress, a tentative conclusion is reached. The language of an interface is not something that can be straightforwardly read off the images of games, or fx sequence in movies, or animations. It emerges through software processes that make data legible and open to manipulability. It can be indirectly encountered as both a colonization of data by software users, and as a colonization of human users by data.


1. INTRODUCTION

Software increasingly influences the ways in which we can imagine fictions, display information and interact with a digitally mediated world. What we see as an image and hear as a sound emerges as an animator works in a sphere of influences. Software is part of a sphere of influences in the sense that it enables particular kinds of actions, and in doing so influences cultural and social patterns of behaviour, and systems of communication (MacKenzie 2006; Chun 2011, Manovich 2011, Kitchin and Dodge 2011).

If software enables particular kinds of actions and so influences systems of communication, how might this be visible? Might there be such as thing as an interface language beginning to emerge. Autodesk® Maya® is taken as a case study to explore if there is a language that reveals traces of a software package, which in turn influence how viewers and users and engage with audio-visual images.

A starting position is, very broadly speaking, that the language of an interface is data. This in itself is not an insight, after all a computer computes, and code becomes active when it operates as programmed software to generate data from other data. A computer operates generating digital data, values and measurements. Within Maya every frame that will be combined to create an image exists as a scene file. A scene file contains objects in a strictly defined language or data structure; it can contain geometry, viewpoint, texture, lighting, and shading information as a description of the virtual scene. To create an animation, the chosen renderer takes this data and converts the screen files from 3D images to 2D versions. When a gamer plays a game, data is rendered in real-time into game images through the game engine.

But since the viewers of animations, players of games, or users of the software, do not read data the question remains: in what ways does Maya present data in ways that are recognizable to humans? If the machine end language (data) does not offer a starting point, is it feasible to ‘read’ the language of an interface from the output imagery, where the output imagery has been rendered by another software as an animation or a game. Maya is a powerful program, designed for the fx and games industry. Initially released in 1998 by Alias|Wavefront, and used in Oscar winning fx (The
Matrix, 1999), the software has become established as a major tool in the fx, animation and to an extent, the games industry. Both skilled and novice artists working with the software say they resist the defaults, avoiding ‘out-of-the-boxness’ such as automated movement and textures as far as they can. This immediately makes it harder to look for connections between the operations of the software and the created images.

On top of this, artists work with the software to generate data according to the conventions required by the remit of project. An fx artist making trees in the Harry Potter franchise or monsters in John Carter (Andrew Stanton, 2012) will have a different set of parameters to work with than someone on the Audi Hummingbird commercial, or someone creating an animated feature such as The Lorax (Chris Renaud, 2012), or someone creating strands of folding proteins or simulations based on satellite data (NASA). Seeking to read an interface language from such a multiplicity of imagery seems wrongheaded, as the imagery is a consequence of merging sets of conventions, rather than solely determined by the influence of software.

A more productive way of thinking is that the language of the interface lies between the languages of data and visual convention. It can be found in the activities through which data is made accessible to human users, through the design of the software as executable code, the processes available to the users of the software (including coders who create scripts and plug-ins), and finally to the viewers and players who encounter the final imagery.

2. APPROACHING MAYA AND ITS MULTIPLE NARRATIVES

The developing pervasiveness of Autodesk Maya coincides with recent developments in academic studies of technology, in particular software studies. Software studies first emerged with Lev Manovich’s book The Language of New Media in 2002. Since then software studies has gained momentum, with a range of approaches emerging within the area of study. Though not mutually exclusive, some approaches study code, some study processes, while others look at processes in relation to the sociality of the software.

Manovich’s work on Photoshop is often described as formalist as it focuses on operations within the software (variability, modularity, transcoding), and how they become incorporated into ways of thinking about media (Manovich 2011). Looking at Photoshop Filters and Layers, Manovich links software techniques with a general principle of computer programming (the ability to re-assemble a collection of elements that can be independently altered). He argues that: ‘all elements of modern media software ecosystem...have not just one but two parents, each with their own set of DNAs: media and cultural practices, on the one hand, and software development, on the other (Manovich 2011:11).’

The purpose of formalism has been to explore a new language for thinking about digital media, one that is informed by software as much as existing traditions in works created using analog and digital media. Critics, while acknowledging that it has been successful in opening out a discussion of software, also point out that in seeing software distilled down to its operations and isolated processes the wider context for understanding the software is set aside. The processes of any software matter, obviously, but understanding them also involves saying more about their ‘neighbourhood (MacKenzie 2006).’

The approach developed in relation to Maya draws from strands of software studies that combine a process-oriented approach with an awareness of the sociality of the software. Adrian MacKenzie’s work Linux explores software sociality. He argues that we come to know Linux based on both what it does and also how it circulates: ‘code, the primary material of software becomes an involuted nexus connecting people, platforms, reading and writing conventions, power, law and creativity, distributed in time and space. If this nexus holds, it ties people together, but not seamlessly, effortlessly or without tensions (MacKenzie 2006: 91).’

What I aim to describe is the nexus of connections of which Autodesk Maya is a part, and from this perspective explore the language of the interface. This language exists not only in the processes of the software, but also in the discourses that circulate in relation to those processes. Through circulating discourse the processes of software are performative in both function and meaning. MacKenzie draws on Judith Butler’s notion of a performative, where: ‘a performative “works” to the extent that it draws on and covers over the constitutive conventions by which it is mobilized (Butler 1997: 51).’ The constitutive conventions of Maya can be found in moving images (games, films, animations, commercials, visualizations), production culture materials (especially ‘making of’ interviews and featurettes), training manuals and online tutorials, and interviews with software users. These together provide narrative through which to describe the sociality of the software.

To give more detail, interviews were carried out with animators working with Autodesk Maya in different contexts and at different stages of their careers. 18 people variously working at major
London-based fx houses, animation studios, games studios, and also people in academic environments at undergraduate and postgraduate programs were interviewed. A range of training manuals has been analyzed, as were videos created by authorized Autodesk training studios (Escape Studio and CGSociety), along with digital-tutors and others uploaded onto Youtube. Also explored are various help forums (creative.cow, as well as those linked to Escape and CGSociety). These materials are added to by direct use of Autodesk Maya, used to create (very) rudimentary pieces of work.

3. MAKING LEGIBLE SPACE

Each of these sets of materials reveals a connecting set of narratives about the software, some with conventions that have the potential to seamlessly knot together, while others pull against one another. Because Autodesk Maya is a very complex multi-functional software, the focus taken here is only on one particular area, that of space. Avoiding the detail of code, this involves looking at human meaningful elements and structures. Space is meant in both obvious and less obvious ways. As an animation tool Maya can be used to create locations, the setting for actions, the environment of a place. In a very literal way, the space is constructed from various elements. In the animated feature Rango (Gore Verbinski, 2011) there is a scene in which the chameleon Rango poses in front of a cactus in the desert (Figure 1). The constructed space includes everything from the sky, the clouds, the stones, grasses, the cactus, and also Rango.

For many people watching Rango the space is highly legible. The chameleon is in a desert, as signalled by the cactus, dry brush, and sandscape that extends out to the horizon. This is a cartoon chameleon wearing a flowery red and white shirt, so no ordinary chameleon, and for some reason ‘he’ is taking a pose like a cactus and seems to have things sticking out of his body, again visually echoing the cactus. There are two narratives legible here, the story of the nameless pet chameleon lost in the Mojave Desert who masquerades as Rango, a desert tough guy. The second probably less obvious but still legible narrative is around the conventions of spatial story-telling, developed over the history of film-making, cartooning, and in the more recent era of 3D computer animation.

Much less legible are the other spaces that come into existence in the making of those images. In making Rango, Autodesk Maya is described in the following way by Kevin Martel a lead animator at ILM: ‘This is such a detail-rich world, we needed software that would enable us to see and show as much of that detail as possible as we were animating. We needed to see all of those wrinkles, bumps, ridges, hairs, and scales as we went along, so we’d know what it was going to look like on the big screen. Maya was simply spectacular at letting us do all of that (http://area.autodesk.com/rango).’ It is more usual to approach a figure such as Rango as a character, and it is through this device that Martel draws attention to the possibilities of working with Maya: the details on the surface of Rango’s skin. But underneath this surface, we can draw out another way of thinking about the processes of Maya: creating the character is about building space. But this is a space that becomes legible in quite different ways to the story-world of Rango and the conventions of cinema.

What Maya does is allow a user to create space, build space and animate elements that combine into an image such as Rango, though equally a car engine in a commercial, a folding protein or animate a temperature map. Through the execution of code in the software, data is made meaningful because it becomes space-like, data is turned into shapes that are computable, demonstrable, and increasingly legible. As code is executed and data is made legible, it becomes entangled in narratives. Writing about expressive processing in computer games, Nick Wardrip-Fruin states that ‘the processes of digital media operate both on and in terms of humanly meaningful elements and structures (Wardrip-Fruin, 2009: 156).’ The narratives with which data become entangled make it legible for humans, and makes the space built within Maya sculptable for users of the software.

3.1. Shelves, Names and Space

Two sets of narratives are explored for the ways in which they make space legible. They are training narratives, found in video tutorials and manuals, and also modelling narratives, addressed mainly
through interview responses. Training narratives serve several purposes. They make sense of the complicated interface of Maya, and teach users how to use various tool sets. At the same time, training narratives give insight into how space is conceptualized within the Maya interface. Looking at modelling provides a different kind of narrative, one that is concerned with the creation of spatial elements in the digital void of Maya’s interface. Taken together, these narratives make data legible as a process-based and colonizable space.

This touches on a point often made by software users in interviews. When asked for three words to describe Autodesk Maya, the word vast, or a close equivalent such as enormous, was used by 16 out of 18 people interviewed. The extensiveness of Maya is both a frustration and an exciting challenge. The frustration lies in having to find the solution, while the excitement lies in being able to find the best solution for a problem. In this convergence of narratives, Maya begins to emerge as a spatial entity in itself, a multifaceted software whose entirety remains unknown to users. The vast program is populated with solutions to problems, and these are the means through which the animator, rigger or modeller begin to colonize the digital architecture. The shelf metaphor both enables and constrains this process: go ahead and customize and here is your shelf (a small edge in the vast architecture of the program).

The tutorials on project naming conventions also make the space of Maya visible, this time through the need to establish sites for recognition. The legibility of the space is to be shared amongst other users. Though admitting that project naming is rather mundane, author of How to Cheat in Maya 2010 Eric Luhta remarks: ‘Getting into the habit now will pay dividends in your future work. This is basic and introductory information, but it’s important enough to still be the first thing in this book!’ (Luhta 4).’ Dariush Derakshani too instills naming as good practice from the first simple project in Introducing Maya 2011: ‘Many professional studios have strict naming procedures and conventions to minimize the confusion their artists may have when working in a pipeline (Derakhshani 2010: 38).’

Similar advice is also found in introductory video tutorials for naming (see, for instance, project naming video on YouTube). The introduction of this naming process serves several purposes at once. It begins a route of enculturation in relation to studio working practice, but also is a method for tagging data, of bringing it into a world of human meaning. The explicit rationale is transparency for both the creator of the files, and for anyone else who has to work on the same set of digital assets somewhere down the pipeline. At the same time, data is taken hold, made legible through the process of naming.

The narratives of the shelf and project naming anchors people within the interface space of Maya, but in ways that are not without tensions. The constituent training narratives invoke the legible space of Maya as both customizable and protocol based. A user can inhabit the space well if they follow protocols, and they can make that space their own, or at least a shelf of that vast space their own. The narratives about Maya reveal the software to be what Donna Haraway ‘condensed
maps of contestable worlds (Haraway 1997: 8). John Law and Valerie Singleton suggest that technological storytelling are descriptions interfering with other performances of technoscience, propping them up, extending or undermining them, or frequently celebrating them (Law & Singleton 2000). In the training narratives of Maya, code is made legible and made humanly meaningful. What begins to emerge is a space that is vast and ordered, and this data as space is not only legible, but because it is legible it can be colonized, either through protocols or customization.

3.2. Modelling Space

Thinking about modelling reveals another set of narratives of legible space in Maya. In opening the software, a user encounters the equivalent of a blank page in Maya: an empty platform delineated with a grid. The default orientation of a new project is a single perspective view of a grid apparently floating in space (See Figure 2 to see the single perspective view of the grid). In fact the space is a void, empty of any thing. The grid is akin to a large red ‘your are here’ spot on a map, only you are in fact nowhere.

Writing about software inhabiting the everyday space of the world, Rob Kitchen and Martin Dodge say that: ‘Software matters because it alters the conditions through which society, space, and time, and thus spatiality, are produced (Kitchen & Dodge 2011: 13).’ This statement is also true of Maya, and not simply because it is possible create a cartoon desert for a shirt wearing chameleon. The software of Maya alters the conditions of constructing space. In Maya space is built in a void by filling it with elements that then become space. Filling the void involves uses the programmed processes available within the modelling toolsets of the software.

Though working on Maya involves sitting at a computer, and using either a mouse or a tablet, building models is very immediate. The software works by providing a set of primitive shapes that can be used to build up a more complex shape, which in turn can be sculpted to get the more nuanced details of curves and angles. In interviews about using the software, modellers describe the possibilities of Maya as ‘liberating,’ giving them the potential to ‘create anything that they can imagine.’ In addition, modellers talk about the sense of control they have over the object, the space that they are bringing into being within Maya.

Modelling in Maya involves using two different tool sets: polygons or NURBS (Non-Uniform Rational B-Splines). Amongst the community forums such as CGSociety and digital-tutor are debates about which is the best way to model, with a consensus that your choice is best made on the basis of what you are modelling. NURBS are good for modelling smooth curves such as those used in car engine parts, but also help with some rigging problems. Polygon modelling allows for detailed modelling and is less data dense, which has attractions for speed of work especially when it comes to rendering.

Whichever toolset is used, working up from primitive shapes is a visually immediate example of data re-presented in a meaningful way for human eyes. Instead of working with complex arrays of coordinates, the data is literally spatialized which enables an interaction with the data mediated by a toolset. Take, for instance, a polygonal primitive cube shape. The shape simply is the cube on the screen, and the modeller can use the tool sets to alter its orientation, its size. They can also alter its shape, add to it using more polygonal shapes. It can also be extruded, pulled outwards to take up more space. The immediacy of space can be an illusion. In discussion about polygons versus NURBS, Wirebird on CGTalk comments: ‘A hole in a polygon shape is an actual hole, whereas a hole in a nurbs object is more of an instruction not to show that area (forum discussion 2009).’ Put another way, space is at times only surface, with the void covered rather than filled.

Beyond depicting the shape of the model, there are other ways of making such a spatial construction legible, and the conditions through which spatialization is controlled and conceptualized becomes more overt within sets of narrative around processes. Modellers comment that learning Maya involves getting to grips with the step-by-step logic of the process. Video tutorials tend to emphasize this step-by-step quality by virtue of working stepwise through illustrative examples. These include understanding how things are grouped in a model, and how when animating models hierarchies of data groups need to be transparent so that changes can be made to individual parts of a model, and/or associated elements in a model. The moving model is not only a void filling entity, but is constructed from compartments of information, and understanding how they are linked helps ensure that the processes of the software are successful. This nodal understanding of data is discussed further below.

There is again more than one meaning given to space in Maya through this combination of narratives about modelling. There is a literal emergence of a void filling model through its changing configurations of a shape. Modellers say that this sometimes occurs through trial and error. Alternatively, multiple passes are used in which the model and animation is whitted down from something that is a clunky though adequate
approximation to a fully developed and model with fluid movement which fulfils what the animator had envisaged.

The narrative of modelling is about the creative work of the animator as they strive to design a moving entity that meshes well with their ideas for the model. The other aspect of space that emerges in these narratives runs counter to the apparent freedom of expression and creative liberation of which modellers and animators speak. Maya is a rule-based system, and getting things to happen, let alone, work requires both an understanding of and a willingness to adhere to that rule-based system.

Nodes are an example of a rule-based system within Maya. Nodes are essentially packets of information that define an object (Figure 3). Rather than thinking about a model terms of its shape on the screen, it exists as information visible as numeric values. The information includes what are called attributes: spatial co-ordinates, geometric descriptors, colour values, and so forth. The attributes define the object and how it will animate. Changing the attribute of a node changes the shape of an object. Complicating matters, nodes exist in a hierarchy, with some nodes parent nodes and others child nodes. A change in attribute to a parent is passed down to child nodes, but changes to child nodes are not passed back up to parents. As a consequence, the child node can have an action added to it, which does not affect the parent.

For instance, the first project of Derakhshani's Introducing Maya 2011 involves modelling and animating a nine-planet solar system. This project teaches the novice about nodes and their rules of hierarchies: 'By creating parent-child relationships, you can easily animate the orbit of a moon around a planet while the planet orbits the Sun. With the proper hierarchy, the animation of the planet orbiting the Sun automatically translates to the moon. In effect, the planet takes the moon with it as it goes around the Sun (Derakhshani, 2010: 56).'

Thinking about Maya through its hierarchical processes presents a different kind of space. Rather than being legible through the shapes of space filling polygons or space covering NURBS, it is legible in terms of the relationships between the elements that make up an object. This narrative configures space in a different way, as a consequence of the execution of an algorithm that connects elements in the model as compartments of information. Such language comes close to taking on a patina of neutrality: space is legible simply through the execution of a process. However, as a number of commentators have pointed out, algorithms are not neutral formal processes (Hayles 2005, MacKenzie 2006, Chun 2011). Algorithms instead operate in a neighbourhood of social, cultural and political relations.

The neighbourhood of processing in Maya brings together creative and technical know-how. No matter good an animator may be, if they lack sophisticated technical skills in working with the software, they will not be able to create good models or movements. When asked who does the doing of animation, the software or the animators, almost all interviewees say that the software is only a tool. And to an extent this is true. A person has to input data for the software to compute data, and that person exerts control over the details of the imagery that they are in the process of creating.

But at the same time modellers and animators are working within the processing structures of Maya, which requires the user to conceptualize spatial connections in particular ways. At this intersection of the expertise of a technically skilled artist and the possibilities of a software, space emerges as multiply legible through the physical dimensionality of the 3D objects modelled, the visual conventions that define the project that an animator is working within, and as processed space constructed from packets of data information.

Matthew Fuller refers to events in software or processes as ‘blips’ around which interrelations, collaborations and conflicts become visible: 'They [blips] have an implicit politics. Their aesthetics can be described as the result of the range of their potential combinatorial or isolatory capacity and its allowance of capture, invention, interrogation, or flight, the rhythms of peace or of compulsion that they put into place (Fuller 2003, 31). Working with Maya does not involve the same kinds of events to which Fuller refers, of which an example might be going to an ATM and getting a print out that tells you your bank balance is in the red. Consequently,
your approach to the hours ahead alters, is renegotiated in some way in the light of having too little cash.

In Maya, events are on-going, taking place in the coming together of creative effort involved in negotiating a way through the fault-lines of a multiply legible space. An entity enfolds ideas based on 3D physical models, the conventions of the project, and the processes that go into constructing an object. This seems more complicated than simply inputting commands, as it involves learning what commands do and do not work, of internalizing the logic of a rule-based system as part of means of creating an image.

In the earlier discussion of the shelf and project naming, I made the point that space in Maya begins to emerge as vast and ordered, and that the legibility of data as space makes it colonisable either through protocols or customization. What also begins to emerge through a discussion of the intersection of technical and creative skills when working with Maya is that the processes of Maya also colonize the decisions of the modeller and animator. Creative thinking enfolds technical skills and functionalities, and these in turn enfold creative thinking.

4. CONCLUSION

Early in this paper the idea of the language of an interface was raised. One way of conceptualizing such a language would be that there visible traces of a software package that influence how viewers and users engage with audio-visual images. The approach taken to understanding what such a language might be has been to explore a range of different narratives about Maya. Though not extensively discussed here, this has included the Maya produced images of fx cinema, games, commercials and science visualizations. It has also involved interviewing users of the software, analysing training manuals, and on-line training videos, as well as ‘making of’ interviews and featurettes.

Focusing on these extensive and diverse materials through the idea of space, it is possible to reach a tentative conclusion that the language of the interface of Maya is concerned with making data legible as space. Space is not limited to the final rendered images of a film such as Rango, but as the multiply legible spaces of the software.

One of problems of taking space to be multiply legible is that it begs the question of which is the ‘real’ space, and for whom or what does that reality emerge. The identities of things are fragile, and what is interesting is how particular realities become consolidated, or how things hang together:

Donna Haraway’s metaphor of a cat’s cradle visualises objects as many things at once. Cat’s cradles, by which Haraway means string entities formed of knots and patterns, are a kind of network. In a cat’s cradle, there is a sense of things becoming snagged, networks forming configurations of partial connections that hang together: ‘For the complex or boundary objects in which I am interested, the mythic, textual, technical, political, organic, and economic dimensions implode. That is, they collapse into each other in a knot of extraordinary density that constitutes the objects themselves (Haraway 1994, 63).’

In working between the spaces of Rango, the shelf and project naming, modeling and also nodal connections defining spatial relations, there is a density of narratives knotting together around the idea of making data legible as space. Data is legible as shapes that combine into complex models of figures and objects, as organizational protocols in the software’s architecture, and as compartments of information executed by software processes. Data, once legible through the processes of Maya can be manipulated, controlled, modified, transformed and then rendered into the enormous range of images that appear in worlds of visual media. In making space legible the language of the interface also becomes entangled with ideas around colonization. That is, the ability of human users of software to colonize data and of data in turn to colonize those human users.

Wendy Chun has recently argued that: ‘Key to understanding the power of software—software as power—is its very ambiguous thinginess, for it grounds software’s attractiveness as a way to map—to understand and conceptualize—how power operates in a world marked by complexity and ambiguity, in a world filled with things we cannot fully understand, even though these things are marked by, and driven by, rules that should be understandable, that are based on understandability (Chun, 2011: 57-58). The language of the interface is not straightforwardly readable in the images of games, or fx sequence in movies, or animations. The language of the interface emerges through software processes that make data legible and open to manipulability as multiply legible and potentially ambiguous constellations of space. This opens up routes for further work which focuses on thinking about how data is colonized by software and the circuits of power that circulate through software and its human users.

5. REFERENCES


