Examiner's commentary

The introduction to this well-researched essay provided context to the reader, reasons both for the choice of focus and the choice of algorithms and details of the methodology to be used. The essay was well-written in a consistent style in which the student's voice was clearly heard throughout. Excellent knowledge and understanding were shown both through the choice of source material and by the way in which this material was incorporated into the argument. Appropriate terminology was used throughout the essay with complex terms frequently being explained in more simple language. The inclusion at appropriate points of well-chosen explanatory diagrams aided the understanding and consequently the flow of the argument. The analysis was both thoughtful and critical as the student successfully attempted to link the algorithmic and mathematical properties of the two methods both to the quality of the final graphics and the degree of difficulty needed to produce them.

Polygons, NURBS patches, and Video Game Graphics

To what extent are polygon models more suited to video game graphics than NURBS?

Subject: Computer Science

Word Count: 3904

Table of Contents

1. Introduction

In our increasingly digitized era, the advancement of computer technology is a topic that has gained prominence. Digital devices, such as smartphones and personal computers, are continuously refined to handle complex processes quickly and efficiently, and advancements in the technology have compressed increasingly powerful processors into smaller and smaller sizes and greater efficiency. However, just as important is the graphical aspect of such digital technology, in which the results of computer's processing is organized and presented to the viewer. The use of 3D models is core to areas including video game graphics and CGI in movies, but also applications in architecture, medicine, business, and more. As such, its optimization is key to improving the functionality of computers and other digital machines.

There are a variety of ways these 3D models can be constructed, and the way this information is stored affects the way it interacts with the calculative and projection processes, thus affecting the aforementioned graphical processing efficiency. As these models function differently, there are certain advantages and disadvantages associated with each type, as each is more suited for different tasks. This paper will look at two modeling techniques and examine their characteristics: NURBS (non-uniform rational B-splines) and polygonal modeling, focusing on their implications in video game graphics and design. These were chosen because they are among the most commonly used forms of modeling, while also exhibiting significantly contrasting characteristics as opposed to some of the more recently-created modeling types that combine the features of these. The question driving this investigation is: to what extent are polygon models more suited to video game graphics than NURBS? Research will be conducted

using online sources that explain and discuss the features of these modeling techniques, primarily those directed at the graphic designers that work with these features on a regular basis, and conclusions will be drawn based on examining these structural characteristics and their implications on both the designer's workflow and the way they are processed by the computer.

2. Introducing 3D Models

Three-dimensional graphics rely on the construction of models, which are are instantiated in relation to each other, as a series of objects within a three-dimensional coordinate system, and not as a series of unrelated pixels. However, what shows up on the screen is a raster graphic, which is a 2-dimensional array of RGB information that will map directly to the pixels on the screen and do not have any sort of interrelation such as the spatial relationships among components of the 3D model. Due to this dissonance between the three-dimensional model and the two-dimensional monitor, the models must undergo a set of highly complicated transformations in order to display the image they describe on the screen. This is known as the rendering process, in which the model is projected onto the two-dimensional monitor, and can become extremely complex and time-consuming depending on the intricacy of the model.[1]

The coordinates delineating the locations of the components that make up the model are rotated and realigned to matched the intended camera perspective, taking into account factors such as depth and an object's distance from the viewport, and aggregating all of the individual objects into the same system of world coordinates. In order to produce realistic images from these models, however, transformative algorithms must also take into account other more complicated factors. For example, objects or parts of objects that fall behind the frontmost one

[1] Chris Woodford, "Computer Graphics," Explainthatstuff.com, last modified September 10, 2018.

should be obscured if it is opaque, and ray tracing algorithms can determine where light might fall realistically on a surface and what shadows might be produced if it were exist as an actual item. Other considerations for graphics intended to be used for animation may include the movement of smoke and flames, or the object's interactions with reflective surfaces, waves, clothing folds. During the rendering process, these factors are all blended together before producing the final raster image.[2]

While these processes concern the structure of the model itself, another very important aspect to creating these models are texture maps. These are essentially images wrapped onto the surface of the model and aligned with the object's coordinates in order to create added realism to the model, and are especially effective in simulating particularly rough or detailed surfaces. This reduces the complexity of the surface that needs to created— for example, instead of creating the individual hairs that make up a cat's pelt and adding lighting and shadows for each, a texture map can greatly simplify the process by adhering an image of shaded fur onto the model's surface. Other types of maps can generate surface topography (bump mapping, Fig. 1), or highlight transparent areas on the model. Their usefulness arises in the way they can reduce the calculations needed to render the object, by "baking" as much of the information as possible onto the two-dimensional image wrapped around it.[3]

---

[2] Eric Lengyel, "The Rendering Pipeline," in *Mathematics for 3D Game Programming and Computer Graphics, Third Edition* (Cengage Learning PTR, 2011), 1-9.
[3] "Start Mastering Important 3D Texturing Terminology," Pluralsight, last modified January 16, 2014.

Fig. 1. Bump maps can be used to hold additional topographic information. Source: Pluralsight, "Start Mastering Important 3D Texturing Terminology." 2014, Digital Image. Available from: Pluralsight, https://www.pluralsight.com/blog/film-games/cover-bases-common-3d-texturing-terminology (last modified January 16, 2014).

Ultimately, all of these considerations— lighting, textures, transparency, etc.— are blended together to determine the color of the fully shaded pixel that will appear, and only after all these are applied is the model finally mapped onto the pixels of the computer monitor in the form of a two-dimensional raster image. Due to this complexity, some of the more complicated graphics can take days or even weeks to render fully by powerful computers.[4] From this arises the issue of real-time rendering or pre-rendering. In films, the CGI can be pre-rendered to produce the video file that will be shown on screen, so the graphics can be made to be highly complicated and later passed to high-end machines to take their time rendering as the rest of the film is being produced. Conversely, much of video game rendering occurs in real-time as new

[4] Woodford, "Computer Graphics."

scenarios arise and the computer communicates this information visually to the player, and this is also why the quality and detail in video graphics appear to lag significantly behind the potential realism that current graphics technology appears to be capable of.

3. Polygonal Modeling

Polygonal models are defined by edges and vertices joined together to create many polygonal faces to model the surface of an object. These are created by describing the number and shape of each face, specifying the location of vertices that will comprise the faces, and then mapping each of these vertices to each face— for example, that the location of the first triangular face will be defined by "vertex 1," "vertex 3," and "vertex 4", all three of which have been defined separately. In this way, rather than directly creating each face with their spatial positioning, data space can be saved as fewer vertices will have to be defined, based on the principle that adjacent faces will share edges and vertices, and changing the location of a vertex will automatically adjust with it the faces that draw upon this information.[5] Each face of the mesh is comprised of at least three vertices, as three points define a plane in space, and in practice, triangular and quadrilateral faces are most common for their simplicity and versatility.

[5] "Introduction to Polygon Meshes," *Scratchapixel*, accessed November 10, 2018.

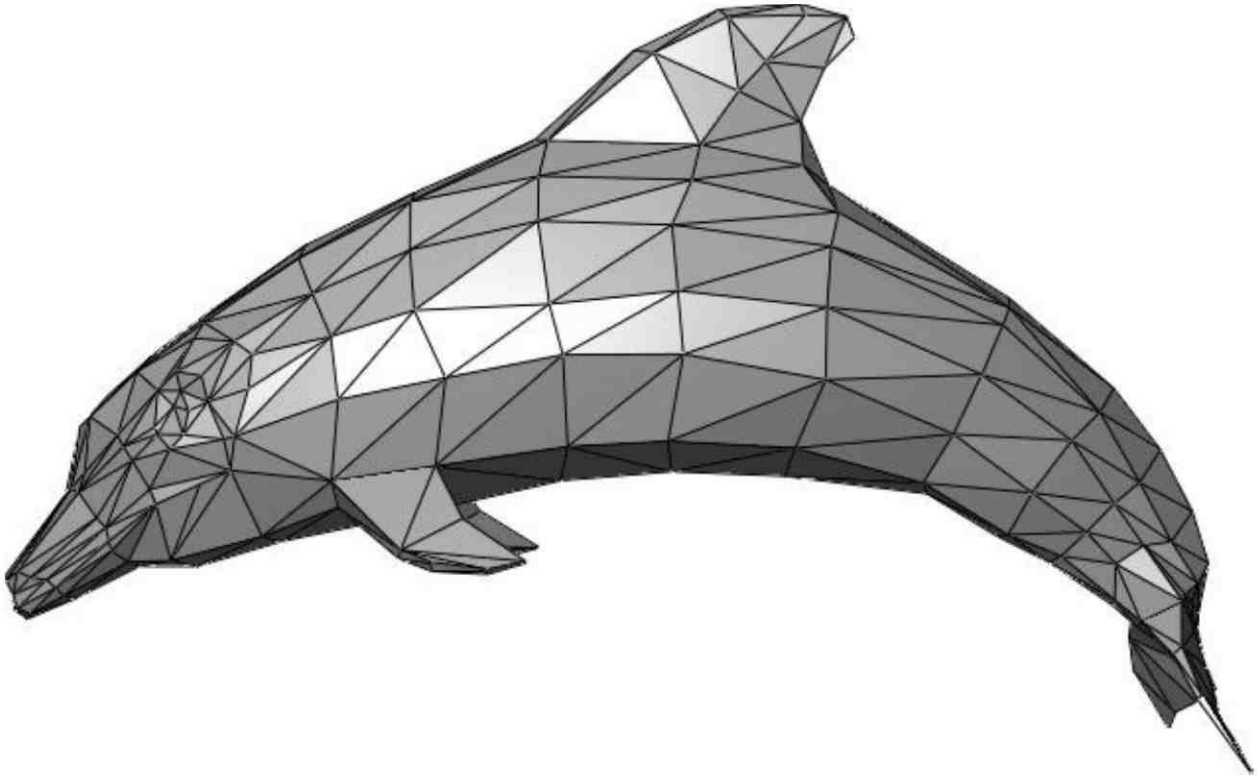

Fig. 2. Example polygon mesh of a dolphin. Source: Ken Power, "3D Object Representations." 2007, Digital Image. Available from: Glasnost - Carlow Institute of Technology, http://glasnost.itcarlow.ie/~powerk/GeneralGraphicsNotes/meshes/polygon_meshes.html (last modified October 22, 2012).

This is also how polygonal modeling is implemented in CAD (computer-aided design) software like Maya, where to create and edit the models graphic designers directly manipulate the edges and vertices of these polygons to generate the desired shapes necessary for creating the intended three-dimensional object. Because of this intuitivity, they are considered to be easier to use compared to NURBS.

4. NURBS Surfaces

NURBS stands for Non-Uniform Rational B-Splines, and is a type of mathematical function used to define curves to ultimately construct models. These are parametric functions, in which the x and y are evaluated based on the progression of a third variable, u.[6] For example, u = 0 will yield a certain x and y value, u = 1 will yield another x-y combination, and so on, until a continuous line is drawn.

The term "spline" was originally coined for the flexible strips of wood used by shipbuilders to draw curves, in which the strip was forced into a bent shape held down by a series of metal clamps called ducks.[7] According to the laws of physics, this strip would always bend in a way that would require the least energy to maintain, and this would create the smoothest possible curve that passed through these knots. B-spline curves were created to reflect a similar principle— a series of control vertices are defined, joining together to form a control polygon; the curve will curve up towards these vertices, and the rest of the curvature between these points is then evaluated using the Cox-de Boor recursion algorithm. Another important feature to the curvature is the weights, which determine the steepness of the resulting curvature. A heavier-weighted vertex will forces the function to curve more heavily towards the control point; a weight of zero will result in no change of curvature (Fig. 3).[8] Finally, these curves are defined in pieces, called "spans", before being joined together.[9] Each span is affected by a

---

[6] Autodesk.help, "The mathematics behind NURBS," Autodesk Knowledge Network, last modified January 22, 2015.

[7] Alastair Townsend, "A Brief History of the Computational Curve," AlaTown, last modified January 9, 2015.
[8] Markus Altmann, "About Nonuniform Rational B-Splines - NURBS," Worcester Polytechnic Institute, accessed November 10, 2018.
[9] "NURBS Terminology," Autodesk Alias Automotive, accessed November 10, 2018.

certain number of control vertices, and is manipulated individually, hence the name "non-uniform", and the higher the degree of the function, the more control vertices the span contains. While objects can sometimes be represented by single functions of very high degrees, it is much more efficient to divide these into a piecewise curve of many lower-degree functions (typically cubic, in practice) to reduce the calculative processes involved in coming up with an equation for an intended line.[10] Ultimately, these equations produce curved segments, the shapes of which can then be taken and used to build up an object. Users manipulate the construction of these objects by modifying the locations and weights of the control on the piecewise functions.

Fig. 3. Example NURBS curve with polygon net $\{P_0, P_1, P_2, P_3, P_4, P_5\}$. Curve passes through B if weight of P_3 is 0; passes through N if weight of P_3 is 1. Source: Markus Altmann, "About Nonuniform Rational B-Splines - NURBS." Digital Image. Available from: Worcester Polytechnic Institute, web.cs.wpi.edu/~matt/courses/cs563/talks/nurbs.html (accessed November 10, 2018).

[10] Autodesk.help, "The mathematics behind NURBS."

For modelling, these concepts are then taken and reapplied with a third dimension to generate the three-dimensional NURBS surfaces that are used on graphics models. Instead of using the control vertices to create control polygons with which to generate the curvature of the line, the curvature of the 3D surface is instead generated with a control net. NURBS surfaces take two parameters, u and v, instead of just u, extending in two directions instead of one.[11] Again, due to the limited degree of these three-dimensional surfaces NURBS surfaces are defined in patches, and later joined together to construct a larger, more complex surface.

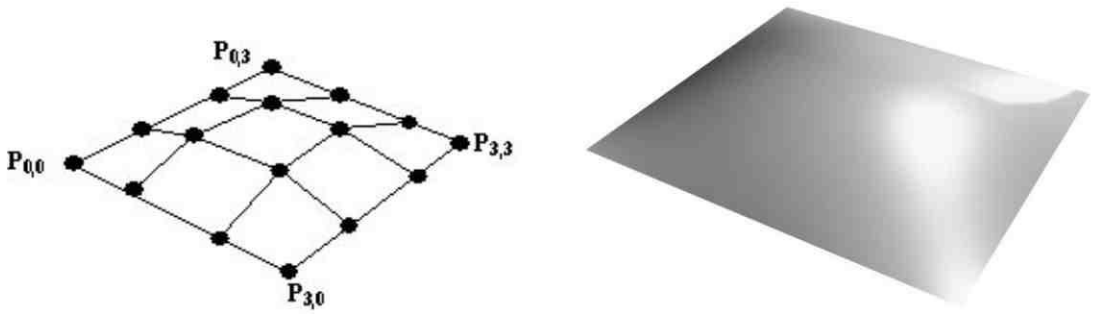

Fig. 4.1 and 4.2. Parametric control net and resulting 3D surface. Source: Dean Macri, "Using NURBS Surfaces in Real-time Applications." 2011, Digital Image. Available from: Intel Developer Zone, https://software.intel.com/en-us/articles/using-nurbs-surfaces-in-real-time-applications (last modified September 9, 2011).

5. Comparing Structures

Smoothness and Precision

One of the most important differences between the two types of modeling is the smoothness allowance. The mathematical nature of NURBs surfaces allows them to be perfectly

[11] "NURBS Terminology," Autodesk Alias Automotive.

smooth, modeling surfaces that satisfy equations exactly, and to reproduce mathematical figures such as conic sections perfectly. Overall, this smoothness combined with the curves' ability to produce natural-looking, organic curvatures becomes a valuable tool during modeling because it allows for the creation of incredibly realistic models.

Thus, a major limitation to polygonal modeling is the lack of accuracy in comparison to NURBS curves. Meshes can only create approximations of what NURBS does with perfect precision, and the models can never be perfectly smooth because they are stitched together with countless flat surfaces, inevitably creating sharp edges. While increasing the number of polygons involved and shrinking their sizes can reduce this problem, the quantity needed to create the semblance of smoothless approaching that of NURBS is extremely high. This is where subdivision algorithms come into play because they can smooth out the sharp edges of the object, able to optimize the model while significantly reducing the workload of the designer. One of the most well-known algorithms, and most commonly-used, is the Catmull-Clark algorithm. The polygonal mesh is refined by dividing the existing polygonal faces to become smaller and more numerous using the locations of the current vertices and averaging them to produce new ones, with roughly four new faces for each previously existing one. This ultimately smoothes out the previous mesh, while maintaining the integrity of its basic form in the averaging of preexisting points' coordinates (Fig. 5).[12] The simplicity of the algorithm means the process can be implemented repeatedly and conveniently to break a relatively low polygon-count mesh into one with many faces and thus quite smooth. Though the resulting mesh will never be able to fully match the perfect smoothness of a NURBS surface, it will reach a point where the visual

[12] Rory Driscoll, "Catmull-Clark Subdivision: The Basics," CodeItNow, last modified August 1, 2008.

difference between the two will be largely negligible due to the limited resolutions on device monitors.

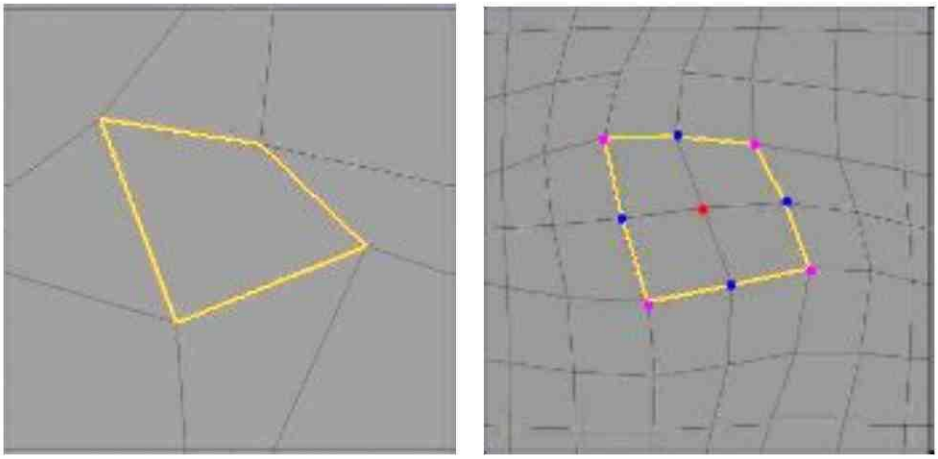

Fig 5.1 and 5.2. Original 3D surface (left) and resulting subdivided surface (right). Source: Rory Driscoll, "Catmull-Clark Subdivision: The Basics." 2008, Digital Image. Available from: CodeItNow, http://www.rorydriscoll.com/2008/08/01/catmull-clark-subdivision-the-basics/ (last modified August 1, 2008).

A smoothness issue that does arise with NURBS occurs along the edges of the patches. Because the surfaces are defined in pieces, the joining of the different patches is not always smooth. In order for the pieces to join seamlessly, not only must the surface be continuous, but the derivatives as well, and to fix these "cracking" issues involves significantly more manual effort, as well as storing additional connectivity information along these edges.[13]

Given the highly visual context of video game graphics, the discord of an unsmooth surface is never desirable. This occurs all over the surface of a polygon mesh, while it only occurs where patches are joined on NURBS models, which are generally much smoother.

[13] Dean Macri, "Using NURBS Surfaces in Real-time Applications," Intel Developer Zone, last modified September 9, 2011.

Conversely, this uneven distribution between smoothness and disjointedness across the surface of a NURBS model can also appear disorienting, compared to the evenly distributed angularity on polygon models. These can be overcome through manual adjustment for NURBS, through subdivision for polygons, or by hiding this defect using texture maps— but is unfortunately an issue that arises regardless of modeling type used.

Structural Constraints

A different issue with NURBS are the constraints arising from the structural definition of the surfaces. Due to its bi-directional parameterization, the control vertices must form a grid, in which the number of points in each row and each column is uniform throughout the patch. While this may not be a deal-breaking limitation, it can add unnecessary complexity to the patch's evaluation and occupying more storage than necessary. Wanting to add control vertices to only a certain area is impossible, for example, because extra control vertices must be added to the entire patch if one wants to make this one region more detailed.[14]

On the other hand, as polygons are comprised of free-floating points in space, this is a constraint that polygonal models are not restricted by. Polygonal models are capable of creating more organic, freeform 3D figures than their NURBS counterparts, a feature valued by many graphic designers for the greater freedom available and more suitable for games, which more often implement such abstract forms. As simplicity should be achieved whenever possible since all of the rendering occurs in real-time, complicating other regions of a model unncessarily from extra control vertices when using NURBS models also makes this a highly inefficient means of creating the models.

[14] Macri, "Using NURBS Surfaces in Real-time Applications."

Use with Texture Maps

Texture maps are overlaid onto the model with another set of coordinates that go in the U and V directions, which are two-dimensional and separate from the three-dimensional XYZ axes of the object's coordinates. These can be best visualized by taking the 3D surface and unwrapping them into flat cutouts, like the nets used in geometry classes to teach surface area of prisms and pyramids.[15] Using these UV coordinates, points on the unwrapped model are then matched up with locations on the texture, and areas of the texture are stretched or transformed as necessary to stretch smoothly.

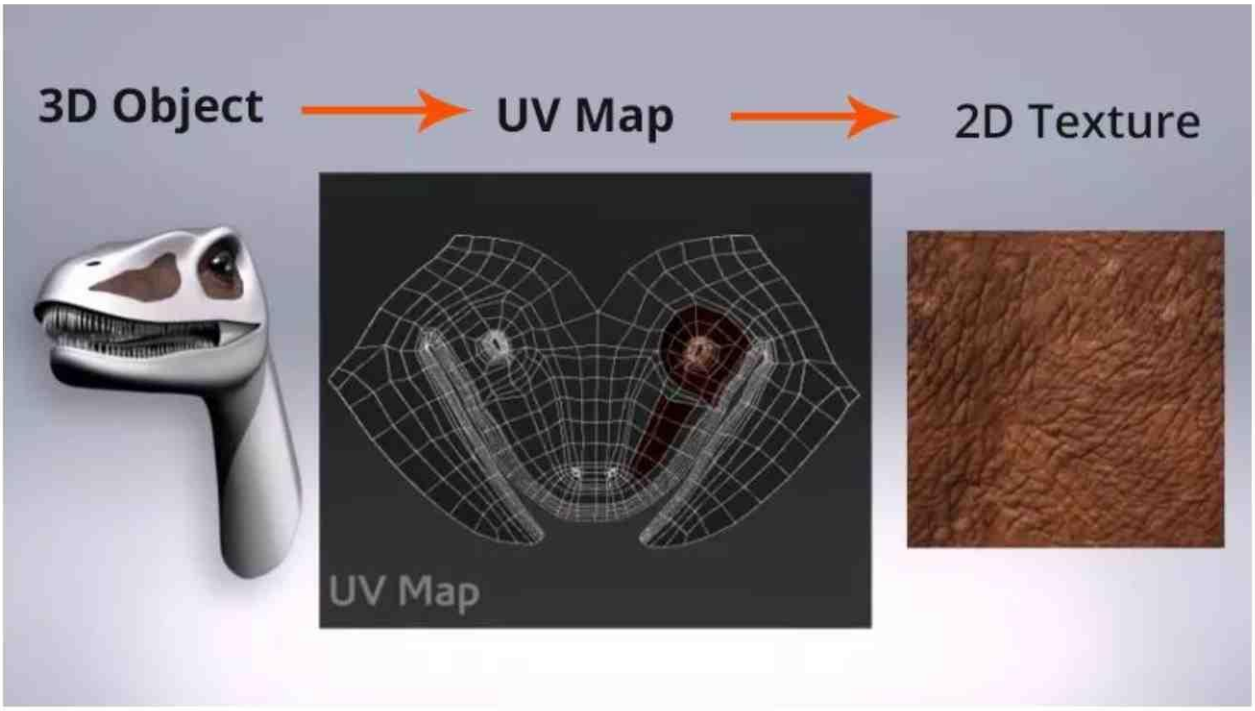

Fig. 6. 3D model (left), 2D map (right), and UV map (center). Source: Pluralsight, "Start Mastering Important 3D Texturing Terminology." 2014, Digital Image. Available from: Pluralsight, https://www.pluralsight.com/blog/film-games/cover-bases-common-3d-texturing-terminology (last modified January 16, 2014).

[15] "Start Mastering Important 3D Texturing Terminology," Pluralsight.

These UV coordinates are created separately from the three-dimensional polygonal mesh— the model exists in three dimensions, while UV maps are two-dimensional. However, generating a map is simple: the mesh can be unfolded along the edges like the aforementioned geometry nets by maintaining the distances and angles among each of the face edges and vertices, creating a two-dimensional object that can then be mapped to UV coordinates. This can be an automatic process (ex. generated algorithmically using CAD software and therefore computer programs), but the UV coordinates can be adjusted manually as well to fix the overlay of the map on the 3D object.

With NURBS surfaces, however, this is a different case, because UV coordinates are already implicit in the curve's definition. The UV coordinates here are the same U and V used for the parameters taken for the surface's evaluation, which means they cannot be edited manually without changing the surface curvature along with it. Though there are often solutions implemented by modeling software themselves in attempt to overcome this hurdle, the process of applying textures to NURBS remains much rockier than for polygon meshes. For example, Maya creates another explicit set of UV coordinates, separate from these implicit ones that are tied directly with the surface control vertices, that the user can then manipulate, to a limited extent, in order to apply the map. However, this also means that the geometry of this model should be complete before beginning this texturing process— the UV set for each surface is unique,[16] as in there is a one-to-one relationship between a surface and the related set of UV coordinates, which also means that editing the topography of a surface would result in a different set of UVs, and as such this topography should be left untouched after these UVs have already been created.

---

[16]Autodesk.help, "Edit NURBS UVs," Autodesk Knowledge Network, last modified September 8, 2014.

Video game graphics depend especially on texture maps because offloading as much visual information as possible onto the 2D image surface significantly eases the rendering occuring in real-time. However, this is an issue that presents itself for almost any modeling task, not just game graphics, and contributes to the perception that polygons are much easier to use than NURBS. While not affecting graphical performance in video games, it will instead affect the designer's workload in making the model's construction process much clunkier.

6. Rendering Efficiency

Rendering involves shading. To accomplish this, hypothetical rays of light are first checked against the model by first examining where it intersects, and then how much light would be reflected off the surface should it occur as an actual object. This is based on the angle between the ray and the surface,[17] which can be easily determined using the dot product of the light vector and the surface's normal at that point— a normal vector is perpendicular to the surface, and describes its orientation.

Finding the normal at a certain point is straightforward for both NURBS surfaces and polygon meshes. A plane is defined in terms of a normal vector; this can be computed given three points contained within a plane, and/or by forming two vectors between them and finding their cross product, which will produce a third vector perpendicular to both of these. This translates easily into the context of meshes, which consist of discrete flat faces that are polygon-shaped and therefore contain at least three vertices, from which these vectors can be calculated. This complements critically how objects are shaded, because the normals of each of

[17] "Ray-Tracing a Polygon Mesh," Scratchapixel, accessed November 10, 2018.

these faces, which describe their orientation, can be easily computed based on the coordinates of the object's vertices.

While NURBS surfaces are continuously curvaceous and therefore contain no flat regions, the same effect can be achieved by finding a tangent plane to the curve at that point. For the bi-directionally parameterized NURBS surface, this can achieved by finding two tangent vectors at a point, one in the u direction and one in the v direction, to define the plane. The normal to this tangent plane can then be found using the cross product, similar to the process for polygon meshes.[18]

However, models created using NURBS surfaces make this process much more complicated because of the infinitesimal changes at each point due to the curves' perfectly smooth nature. Light falling on a certain patch of a model must undergo these calculations at the differential level, while for meshes they only need to be performed for each of the faces— while these may still be small, they remain discrete and not nearly as miniscule as for NURBS. As a result, NURBS rendering falls significantly short of polygons in terms of efficiency, and in fact, it takes less processing power to just convert NURBS into triangular meshes before rendering.[19] Either way, simply because polygonal models already exist as meshes, this makes polygons much easier and much quicker to render than NURBS, and in processes that involve real-time rendering, such as video games, this can be a critical factor in the strain the application places on the graphics processor.

---

[18] Macri, "Using NURBS Surfaces in Real-time Applications."
[19] "Ray Tracing: Rendering a Triangle," *Scratchapixel*.

7. Conclusions

In practice, these models can be transformed between each other, such as the way triangle meshes are generated from NURBS during rendering. In fact, because NURBS curves are more compatible across platforms than meshes, any polygonal models often have to be converted into NURBS before they can be exported and used with these other softwares that may be important to an artist's workflow.[20] For this reason, the importance of the distinction between the forms that models are created in begins to smudge away. However, while generating meshes from NURBS surfaces is relatively simplistic (as they are constructed from a series of planar approximations of the parent curve), transforming from the other direction is not so simple. Rhino's MeshtoNurb command does not form a smooth NURBS curve from the vertices on the mesh, for example, but an angular one by creating a NURBS patch for every face on the polygon before joining them together, creating the aforementioned issue of disjointedness as well as occupying a very significant amount of data space.[21]

The primary value of NURBS arises from its mathematical nature and the realism that their smoothness allows for. The way they can exactly capture conic sections or other mathematical figures is very important in engineering-related and highly technical design tasks, but this significance falls off with respect to video game graphics, who do not require such technical exactness in their function.

The aesthetic aspect of video game graphics *is* quite important, and the potential realism that NURBS models offer can contribute significantly to a game's visual appeal to a potential

---

[20] Atkins, "How to select the best CAD modeler: recent engineering software uses much more than Nurbs. Here is a guide to the best approach."
[21] MeshToNurb, Rhinoscript Programmer's Reference, accessed November 10, 2018.

audience. The price of this, however, is efficiency. Achieving this realism means overcoming the inherent aesthetic issue of the disjointed seams by fixing the patches' continuity, and in comparison, repeated application of Catmull-Clark to make a polygonal model appear more smooth would require much less manual effort. The constraint of distributional uniformity of control vertices complicates the model unnecessarily, and the application of texture maps is not nearly as convenient as due to the inability to directly edit the UV mapping of the texture. None of these issues are associated with polygonal models, in which the freeform capabilities are arguably more suitable to the creative aspect of designing video game models. Most importantly, the process of rendering NURBS is much more complex and time-consuming than that of polygons. Particularly due to this last reason, any tradeoffs between these two factors that leans more heavily on the realism that NURBS offers places significantly heavier loads on the graphics processor due to the way these graphics are rendered in real-time. If a game's graphics causes the hardware sophistication requirements for players' to be extremely high, the appropriateness of this is questionable as video games are designed to be run on personal, home devices with limited processing power.

Above all, video game graphics rely on communicating as much visual information as possible, as simplistically as possible. NURBS can push visual detailing slightly closer to approaching realism, but at significantly higher performance costs. Thus, while perhaps NURBS may be used by designers for reasons of personal preference, polygon meshes are overall more suited for use in video game graphics for their greater versatility and efficiency.

Bibliography

Altmann, Markus. "About Nonuniform Rational B-Splines - NURBS." Worcester Polytechnic Institute. Accessed November 10, 2018. web.cs.wpi.edu/~matt/courses/cs563/talks/nurbs.html.

Atkins, Kevin. "How to select the best CAD modeler: recent engineering software uses much more than Nurbs. Here is a guide to the best approach." *Machine Design* 83.17 (October 6, 2011): 56+.

Autodesk.help. "The mathematics behind NURBS." Autodesk Knowledge Network. Last modified January 22, 2015. https://knowledge.autodesk.com/support/maya-lt/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/MayaLT/files/NURBS-overview-The-mathematics-behind-NURBS-htm.html

Autodesk.help. "Edit NURBS UVs." Autodesk Knowledge Network. Last modified September 8, 2014. https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2015/ENU/Maya/files/UV-sets-Edit-NURBS-UVs-htm.html.

Driscoll, Rory. "Catmull-Clark Subdivision: The Basics." CodeItNow. Last modified August 1, 2008. http://www.rorydriscoll.com/2008/08/01/catmull-clark-subdivision-the-basics/.

"Introduction to Polygon Meshes." Scratchapixel. Accessed November 10, 2018. https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-polygon-mesh.

Lengyel, Eric. "The Rendering Pipeline." In *Mathematics for 3D Game Programming and Computer Graphics, Third Edition.* Cengage Learning PTR, 2011.

Macri, Dean. "Using NURBS Surfaces in Real-time Applications." Intel Developer Zone. Last modified September 9, 2011. https://software.intel.com/en-us/articles/using-nurbs-surfaces-in-real-time-applications.

"MeshToNurb." Rhinoscript Programmer's Reference. Accessed November 10, 2018. https://developer.rhino3d.com/api/rhinoscript/mesh_methods/meshtonurb.htm.

Powers, Ken. "3D Object Representations." Glasnost - Carlow Institute of Technology, Last modified October 22, 2012. http://glasnost.itcarlow.ie/~powerk/GeneralGraphicsNotes/meshes/polygon_meshes.html.

"NURBS Terminology." Autodesk Alias Automotive. Accessed November 10, 2018. http://www.bluesmith.co.uk/nurbs2.htm.

"Ray-Tracing a Polygon Mesh." Scratchapixel. Accessed November 10, 2018. https://www.scratchapixel.com/lessons/3d-basic-rendering/ray-tracing-polygon-mesh/Ray-Tracing%20a%20Polygon%20Mesh-part-1.

"Ray Tracing: Rendering a Triangle." Scratchapixel. Accessed November 10, 2018. https://www.scratchapixel.com/lessons/3d-basic-rendering/ray-tracing-rendering-a-triangle.

"Start Mastering Important 3D Texturing Terminology." Pluralsight. Last modified January 16, 2014. https://www.pluralsight.com/blog/film-games/cover-bases-common-3d-texturing-terminology.

Townsend, Alastair. "A Brief History of the Computational Curve." AlaTown. Last modified January 9, 2015. http://www.alatown.com/spline/.

Woodford, Chris. "Computer Graphics." Explainthatstuff.com. Last modified September 10, 2018. https://www.explainthatstuff.com/computer-graphics.html.

International Baccalaureate®
Baccalauréat International
Bachillerato Internacional

Extended essay - Reflections on planning and progress form

Candidate: *This form is to be completed by the candidate during the course and completion of their EE. This document records reflections on your planning and progress, and the nature of your discussions with your supervisor. You must undertake three formal reflection sessions with your supervisor: The first formal reflection session should focus on your initial ideas and how you plan to undertake your research; the interim reflection session is once a significant amount of your research has been completed, and the final session will be in the form of a viva voce once you have completed and handed in your EE. This document acts as a record in supporting the authenticity of your work. The three reflections combined must amount to no more than 500 words.*

The completion of this form is a mandatory requirement of the EE for first assessment May 2018. It must be submitted together with the completed EE for assessment under Criterion E.

Supervisor: *You must have three reflection sessions with each candidate, one early on in the process, an interim meeting and then the final viva voce. Other check-in sessions are permitted but do not need to be recorded on this sheet. After each reflection session candidates must record their reflections and as the supervisor you must sign and date this form.*

First reflection session

Candidate comments:

For my EE, I wanted to learn more about computer graphics because it seems to naturally intersect with my interests in art a
nd math/computer science, and also correlates with my interest in video games (which makes extensive use of these). Beca
use it isn't covered by the IB CS curriculum, this also means I'll have to build up my understanding of it myself.
Based on my preliminary research over the summer, a problem I foresee is struggling with the math— computer graphics inv
olves a lot of geometry and functions to build 3-
dimensional models. For my project to be successful, I'm anticipating that I'll need to self-
study some of these concepts because they are indirectly related to my topic.
After some brief reading about my topic I've learned that there are different types of modeling structures used by graphic des
igners, and for my paper I wanted to compare a couple of these and evaluate what types of tasks each might be best suited f
or.

Date: 9/12/18

International Baccalaureate®
Baccalauréat International
Bachillerato Internacional

Interim reflection

Candidate comments:

I've narrowed the scope of my research by selecting two types of modeling to focus on: NURBS and polygon. In particular, NURBS was very interesting to learn about because complex curves can be defined with little information. Though I initially struggled to conceptualize how the different aspects tie together, I became more comfortable with this terminology after playing around with a NURBS curve generator I found online.
With my eye on the deadlines I've begun drafting disconnected paragraphs that explain basic concepts and brief comparisons. I'm continuing to think about how I can refine my research question and how I'll structure an argument around it; even though it's more focused I know that it's still rather vague.
Research is growing difficult because most sources are divided into two camps: practical information for graphic designers that doesn't align with my analytical aims, or extremely technical research papers far beyond the depth of what I believe would be appropriate.

Date: 12/4/18

Final reflection - Viva voce

Candidate comments:

I'm pretty confident in my analytical writing skills, and I think my project has logical structure and precise language to reflect that. However, attempting to describe the visual aspects of my topic through text was a new challenge for me. I added images to clarify some that were most difficult to visualize, but I'm not sure if all of my other image-
less explanations are clear and make full sense to others.
My most important takeaway was the importance of a good guiding question on the quality of an investigation. I think largely due to approaching it with a "research first, organize later" mindset that I never managed to free myself from, a lot of my analysis felt superficial. These issues could have been avoided if I managed to come up with a focused question earlier on— it would have concentrated my research better and allowed me think ahead in terms of organization.
Overall, I think I learned a lot about both the research process and 3D modeling, and I'm proud that I managed to complete such a big project.

Date: 3/1/19