PROFESSIONAL DEVELOPMENT

High poly to low poly workflows for real-time rendering

Nicky L Webster
Medical Illustration, Guy's and St Thomas' NHS Foundation Trust, London, England

ABSTRACT
This paper outlines a digital workflow that allows artists to add visual detail to 3D models whilst adhering to the strict polygonal budget that is vital to successfully render in real-time. Two versions of a model – a high-poly model, and a low-poly model – are produced to create a 2D normal map through a process known as ‘baking’, to achieve the illusion of the high resolution detail on the low-resolution model.

 ARTICLE HISTORY
Received 30 January 2017
Accepted 24 March 2017

KEYWORDS
Low poly; real-time; 3D; normal maps; modelling; rendering; medical illustration; visualisation

Introduction
As e-learning has become part of the mainstream in medical education (Ellaway & Masters, 2009), medical illustrators are required to use the latest technologies to provide appropriate content, to include real-time 3D applications such as virtual reality, augmented reality and educational gaming and apps. A tailored set of 3D modelling skills are therefore required to create assets that are suitable for use in real-time rendering engines.

Real-time applications take in the actions of a user and output them within a time frame that the user senses as immediate or current (Khan & Khan Pathan, 2016). The rendering engine is therefore required to re-draw the image on screen 30 to 60 times per second to ensure smooth continuous visual feedback. There is a computational cost to render each polygonal face belonging to a 3D model, and so as to ensure this fast render time the models must be low poly.

Three-dimensional shapes have faces, edges and vertices, as illustrated in Figure 1. Game engines count the number of vertices a 3D model contains (Blackman, 2011), and this vertex count is ultimately more important for performance and memory than the polygon count, but for historical reasons artists more commonly use the triangular polygon count as a performance measurement while modelling. This in turn gives an estimate of the vertex count, which is roughly half the number of triangular faces.

Modern PCs can easily handle hundreds of thousands of polygons, but the rendering speed will rely on the users’ graphics processing unit (GPU) and central processing unit (CPU). As outlined by Blackman, Unity (a global game development software company) have recommended a 30–40,000 vertex count for a fourth generation iPad (Blackman, 2011), therefore a medical illustrator creating 3D assets for a tablet application could aim to cap the poly count at 60,000–80,000 triangular faces.

Instruction and software
The below steps will demonstrate a high-poly to low-poly workflow to produce part of a detailed anatomical model. The instruction uses Autodesk Maya and Pixologic ZBrush for the model creation process,

Figure 1. Faces, edges and vertices.
and xNormal for the baking process. xNormal is freeware and can be downloaded from http://www.xnormal.net. An accompanying video entitled ‘High Poly to Low Poly Workflow’ uses further examples to clarify the more complex areas of the workflow and is located on the author’s YouTube channel, https://www.youtube.com/c/nickylwebster. A basic level of knowledge of 3D modelling is required.

**Example of the high standard of work in today’s AAA game industry**

*Figure 2,* Geralt of Rivia, is an example of a current real-time 3D model from The Witcher video games series. It has an abundance of detail, from the scar on the face to the intricate pieces in the armoury, and this would require millions of polygons to achieve using geometry alone, making it unsuitable for real-time rendering. A high-poly to low-poly workflow has been used to achieve the appearance of this visual information that catches the light as if it were represented by real geometry. This level of detail not only sets the bench mark for the games industry, but also influences the work that medical illustrators are expected to produce, as this is what our audience knows and expects.

**The 3D model production process**

*High-poly model creation*

The first step is to create a high-poly model. For this project example, a model of an iris has been created for use in an interactive app of the human eye. As portions of anatomy can commonly be viewed in isolation in interactive apps, the iris is designed as a separate piece of geometry that can be disconnected from the choroid. It is constructed from a primitive torus in Autodesk Maya, then imported into ZBrush and subdivided six times until the poly count reaches the millions, as a high resolution is required to sculpt fine detail. The final sculpted iris in *Figure 3* has 6,553,600 triangular faces.

*Duplicate and decimate the high-poly model*

Duplicate the model before decimating to keep the original high-poly sculpts unaltered. Use Decimation Master under the ZPlugin palette to reduce the polygon count while retaining sculpted detail. The lowering of the poly count is necessary, as many systems cannot export the larger size of model. In this example, the poly count is decreased to 2,621,400.
triangular faces. Export the decimated high-poly model as an obj file.

**Low-poly model creation**

There are several levels of Subdivision under the Geometry palette that can be used to export a low-poly version. Level 1 has 1600 triangular faces, but level 2 has more resolution with 6400 faces that better match the high-poly model in terms of shape, size and form, as can be seen in Figure 4. Export the low-poly version as an obj file.

**Alternative methods of creating the low-poly model**

An alternative method is to duplicate the model and then use the ZRemesher function, found under the Geometry subpalette, to retopologise the model.

**The preparation process**

**Import the high-poly model and the low-poly model into Maya**

To prepare the models for baking, import both the high-poly and low-poly obj files into Maya. Centre the pivots using Modify > Centre Pivot, then snap each model to the Origin using the Move Tool, holding down X on the keyboard to snap to the Grid. Ensure they are perfectly aligned with one another as shown in Figure 6. Vertices may need to be moved on the low-poly version in order to match the forms as closely as possible.

**UV unwrap the low-poly model**

UV unwrapping is the process of ‘unfolding’ a mesh, so that a 2D texture can be applied to the 3D object.
(What Does Unwrapping Do, 2016). The low-poly model is unwrapped into a 1:1 square texture space, using Maya’s UV Texture Editor. This is a large and difficult subject, and covered in more detail in the tutorial video using two different examples at the 06:25 timestamp.

**Harden and soften the vertex normals**

Select the model and soften the vertex normals – Mesh Display > Soften Edge. This helps to avoid the faceted look of a low-poly model. However, a good rule of thumb to follow is that any edges on the model that have an angle of more than 45 degrees should be re-selected and hardened – Mesh Display > Harden Edge, otherwise unwanted artefacts will occur (Normal Map Modeling, 2016). Where these hardened edges are placed, the corresponding edges in the UV map must be separated and moved away from each other.

In Figure 7, the low-poly model is unwrapped into five separate UV islands (the long strip around the edge has been cut into two to fit within the UV square), with the normals softened, and then hardened on the edges highlighted in orange.

**Triangulate the low-poly model**

Triangulate the low-poly model before baking the normal map. Many online tutorials bake the normal map with the model set to quadrilateral faces, and then allow the rendering engine to triangulate the model upon import; this is bad practise and offers no user control over the new topology. Triangulating first also ensures that the normal map is created specifically for this topology, and helps to avoid baking errors or artefacts once the normal map is applied. Duplicate the model first to retain a quad version if needed, then apply Mesh > Triangulate.

**Create a third ‘cage’ model**

To get the best results from xNormal, a third model called a cage is required. This cage model should encompass both the high-poly model and the low-poly model, and also have the exact same topology as the low-poly model. In Maya, duplicate the triangulated low-poly model and rename it as ‘cage’ in the Outliner. Within the Move Tool settings, change the Axis Orientation to Normal and then select all the vertices on the cage model. Drag out on the normals manipulator (marked ‘N’) so that the cage starts to inflate. Stop once it has scaled up just enough to encompass both the high- and low-poly models, as Figure 8 demonstrates.

**Export the high-, low- and cage models**

Delete the history of each model - Edit > Delete by Type > History. Then, freeze the transformations –
Modify > Freeze Transformations. Export each model as an obj. Name each model as xn_high, xn_low and xn_cage, so that it is clear which obj files should be imported into xNormal.

The baking process

Import the models into xNormal

Open xNormal, and under the High Definition Mesh tab, right click in the blank space, choose Add Mesh and import the xn_high.obj. Under the Low Definition Mesh tab, right click and import the xn_low.obj. Now, right click again on the imported xn_low.obj file and choose Browse External Cage File. Import the xn_cage.obj file.

Bake the normal map

Under the Baking Options tab tick ‘Normal Map’ and set the file to save in a preferred directory. Figure 9 shows the options set to bake out a 2k map, with the antialiasing set to 4x, and the edge padding increased to 16.

Edge padding duplicates the pixels along the inside of the UV edge and spreads those colours outward, forming a skirt of similar colours. This helps to prevent any colour bleeding on the model when the rendering engine uses a process called downsampling (Edge Padding, 2016).

Select Generate Maps to bake the normal map. Figure 10 shows the final normal map; xNormal has interrogated the difference in surface normals on the model.

Figure 8. Creating the cage.

Figure 9. xNormal baking options.
models, and baked this directional information down into RGB values in a 2D bitmap image.

Apply the normal map to the low-poly model and inspect for errors

My software of choice to review real-time assets is Marmoset Toolbag 2. Marmoset is a standalone real-time renderer that allows you to quickly test your assets in various lighting conditions. Import the xn_low obj file and under the obj Mesh Settings set the Tangent Space to xNormal. Apply a new material, and in the material’s Surface Normals tab, assign the normal map. Figure 11 shows the final low-poly model with and without the normal map in Marmoset.

To assess the normal map in Maya, apply a material to the model and then assign the normal map to the material’s Bump Map slot, and change Use As to Tangent Space Normals.
Further examples

Figure 12 illustrates the workflow on the cube model used in the video example. The edges of the low-poly cube are too sharp to react with the light, but the normal map has successfully mimicked the bevelled edges of the high-poly cube, allowing the light to catch the edge as it would do in real-life.

Figure 13 demonstrates the limits of a normal map on an antibody model. Many antibodies may be presented on the screen at once, requiring a very low resolution singular model. The normal map has added detail back into the model, but the faceted contour of the low-resolution model has not changed. However, this is not noticeable once the model is scaled down. The distance at which the model will be viewed, and how many assets will be on screen at one time, will help to determine the poly count for the low-poly model.

Outcome

Normal maps change how the light is calculated on the surface of a 3D model, but cannot change the geometry itself. If used strategically, they can be a great asset for medical illustrators to add detail to real-time models that will aid in the visual communication.

Acknowledgements

Mr Nathan Gallardo, Game Designer and Independent Developer
Ms Arezoo Alford, Medical Photographer, Guy’s and St Thomas’ NHS Foundation Trust
Mr Markos Kay, for the use of his image on the PC desktop in the accompanying video. Copyright Markos Kay. All rights reserved. www.mrkism.com
The members and contributors to Polycount.com forums
The members and contributors to ZbrushCentral.com forums

Figure 2, The Witcher is a trademark of CD PROJEKT S.A. The Witcher game is © CD PROJEKT S.A. All rights reserved. The Witcher game is based on the prose of Andrzej Sapkowski. All other copyrights and trademarks are the property of their respective owners.

Video soundtrack: Kevin MacLeod (incompetech.com), Licensed under Creative Commons: By Attribution 3.0 License, http://creativecommons.org/licenses/by/3.0
Disclosure statement

The author reports no conflicts of interest. The author alone is responsible for the content and writing of this article.

References

Blackman, S. (2011). Beginning 3D game development with unity (p. 924). New York: Apress.

Edge Padding. WWW Document. Retrieved from http://wiki.polycount.com/wiki/Edge_padding. 18 December 2016

Ellaway, R., & Masters, K. (2009). AMEE Guide 32: e-Learning in medical education Part 1: Learning, teaching and assessment. Medical Teacher, 30, 455. http://dx.doi.org/10.1080/01421590802108331

Khan, S., & Khan Pathan, A. (2016). Wireless sensor networks: Current status and future trends. CRC Press, p.172

Normal Map Modeling. WWW Document. Retrieved from Boca Raton, FL http://wiki.polycount.com/wiki/Normal_Map_Modeling. 18 December 2016

What does Unwrapping do and Why is it Important. WWW Document. Retrieved from http://blender.stackexchange.com/questions/38651/what-does-unwrapping-a-model-do-and-why-is-it-important. 18 December 2016