OPTIMIZING THE 3D CHARACTER MODELING PROCESS FOR GAME DEVELOPMENT

Henri Kuismin

Markus Turppa

Bachelor's thesis
March 2013
Degree Programme in Media

TAMPEREEN AMMATTIKORKEAKOULU

Tampere University of Applied Sciences

ABSTRACT

Tampere University of Applied Sciences
Degree programme in Media

HENRI KUISMIN & MARKUS TURPPA:
Optimizing the 3d Character Modeling Process for Game Development

Bachelor's thesis
Pages: 51
March 2013

The purpose of this thesis is to discover optimal ways to create 3D character models for games and to define how to find the best methods to deal with various situations that might be related to 3D character modeling.

The methods used in the thesis were qualitative analysis of the sources and practical study of a related game character modeling project.

The main conclusions of the thesis are that the optimal working methods for game character modeling are tied to every aspect of game development all the way from the initial ideas to the final game. Our analysis suggested that in order to optimize the various stages of 3D character modeling for games, understanding the possibilities in techniques, methods and software as well as all of the other closely related aspects of game development are the key to finding the best ways to work. The project study supported the analysis by showing us practical examples on how the different phases of modeling influence each other as well as various other relevant steps in the development of a game.

Key words: 3D modeling , Game development, Character design

CONTENTS

1 INTRODUCTION

The goal of this thesis is to test and find out what is needed in modeling 3d characters for games and how to optimize the creation process for optimal efficiency and results. Our means of doing this is researching and analyzing information from various sources as well as using experience from our past and present projects. The modeling process can vary along with the different models, games and their purposes but there might be some ways to make sure that it is done more efficiently in general. While we will try to find common, more widely applicable ways of optimizing the process we also want to analyze how to define the best approach for a more specific case and for this we create 3D character assets for our future game project which functions as an example throughout the thesis. The game project, called ‘Orcs hogging hogs’, mixes elements from role playing games, sports games and action games into a fast paced casual gameplay. The game is set in a fantasy world inhabited by orcs playing against each other on an ancient arena trying to catch a hog to sacrifice it on their own tribe's altar to appease their hungry gods. The characters of Orcs hogging hogs consist of three different types of orcs and the hog that they are trying to catch. They play an important and essential role in the game’s visuals and as such, their modeling process is used as a hands-on example throughout the thesis.

We research and test if we find methods that are optimal for game character modeling in general regardless of the models purpose and if we find out that they are not suitable for each individual situation, we will see what kind of limitations they may have and what kind of other methods there are that could work better. When there are a variety of different possible methods related to a topic, we will be trying to discover which qualities in the game, the model and even in the 3D modeler define the best approach for the task in question. Lastly our goal is to seek out the way in which these differences in the individual case can lead to the most optimal process of working with a game character model.

Our thesis will cover the process of creating a 3D game character mostly from the modeling point of view; however we will also deal with some other closely related subjects in game design from sketching all the way to texturing, rigging and animation. Our ap-

proach on the subject is rather practical and thus we include some hands on methods and tools that are used to achieve the best possible results.

First the relationships between game design, visual design and character design will be analyzed and used to define ways that are important for designing character models for game development. We go through different aspects of game design, such as game mechanics and narrative and see how they are connected with the overall visuals of the game, in order to find the link between them and the character models. The game world as well as the characters' interactions within the game will be reviewed on a general level to define the use of the characters personality, past and purpose in the game world and the effect these things can have on the way the characters should be designed and modeled.

Then we go on to give some basic understanding to the reader on what 3D modeling is and how the character and game design process might affect it and only then go into more detail with the different methods, tools and approaches in modeling.

2 GAME DESIGN AS A FOUNDATION FOR CHARACTER'S VISUAL DESIGN

In an article 'Game Design Cognition: The Bottom-up and Top-Down Approaches' by Gilliard Lopez and Rafael Kuhnen, 03.02.2013 (http://www.gamasutra.com/view/feature/130542/game_design_cognition_the_.php) developing a game is presented as a non-linear process; "*The cognitive process of designing a game begins with an idea. Sometimes it is a concept that we want to translate into play; sometimes it is gameplay that we want to turn into concept. The process of turning such ideas into palpable material, which then becomes a game, is composed of several journeys of thought and specification back and forth between these two extremes.*"

As pointed out in the article, game design is usually not straightforward enough to be constructed as a simple top-down or a bottom-up structure, but a much more complex process of designing parts and layers that are at the same time based and influencing each other. Even though the design often starts from a single idea; whether it is a story, a gameplay concept or something else; that idea usually also develops further along with the rest of the design.

Approaching upwards from the foundation when working on a game can lead to better and more consistent results and prevent any major changes and fixes afterwards but as the process is not a completely straightforward one, the design has to be revised and kept up to date. The foundation for the whole game comes often from a single idea that is then constructed into a design for a complete game. Using that design as a basis for all the different types of work done for the project makes the process smoother and assures that the project team is actually working with a mutual outcome in mind. As the concept changes along the way, the consistency of the project is better if everyone does these changes using the same design as a basis.

When designing a character for a 3D game, there must be an understanding on how the character is constructed and where to start in order to keep the character consistent with the rest of the game. As the character design is a part of the game design as a whole, it cannot be completely separated from it and it needs to evolve and change along with other aspects of the game. The 3D design of the character needs to have the game de-

sign as a foundation in order to make the needs of the model apparent. Even with games that start from an idea that may be essentially the character itself, the final design before starting the modeling process has to have the gameplay, the overall visual style and other game design aspects behind it so that all the modeling decisions can be made accordingly. When there is a storyline, character's personality and how the characters are viewed by the player can affect the character design greatly and should always be a part of the character creation process.

Game 3D character artists do not only work together with the rest of the visual and graphics oriented people involved in the project but with the whole team including the programmers, sound designers, game designers and others. The game design is what keeps the whole game project intact and it should also lead the general direction of the visual design and as a part of that the characters design process.

Game design, genre and theme usually describe a certain type of atmosphere for the game and can tell a great deal about the game’s environment and the world in which the game characters live. Together with the game's functionality and sounds, the visual design is supposed to bring out and support that atmosphere and the overall theme of the game. A good example of the use of a theme of the game in character design can be found on ‘*Brütal Legend*’ by Double Fine Studios (Electronic Arts, 2009) where a heavy metal music theme was integrated in the character design and also worked as the starting idea for the whole game (see image: 1).

(Image 1) Brutal Legend by Double Fine Studios (Electronic Arts, 2009) displays a good use of game's theme; heavy metal in the character design.
(Chris Kohler, 26.02.2013, http://www.wired.com/gamelife/2009/04/eyes-on-jack-bl/)

As implied in The Art of Game Design - A Book of Lenses by Jesse Schell (2008, Morgan Kaufmann publishers), the visual design of the game is not only to enhance the game's atmospheric experience, but also to support the game mechanics by working as visual clues for information or actions. They can be a very useful tool in letting the player know what to do and what to expect from certain things in the game and used as a method of indirect control over the players actions. These visual indicators can sometimes be as simple as color themes indicating for example characters hostility or friendliness or they can be used to actually take the players focus away from the background while highlighting the things that are meant to be interacted with.

It is important to know what the graphics or the visual objects in the game are used for, so that they can be designed in accordance with the functionality and their significance. The visual design itself should decide overall visual themes for the whole game but these themes can also be divided into sections when needed according to the game design. This could mean for instance visually defining characters with similar functionality or with a common interaction in the game. The visual definition for the character can also be done using elements outside the character design. The Sims by Maxis (Electronic Arts, 2000) uses speech bubbles and other indicators floating above the character's head to display the character's status and interaction (see image: 2).

(Image 2) The Sims by Maxis (Electronic Arts, 2000) uses indicators outside the character design to display their status and interaction.
(26.02.2013, http://www.thesimshub.com/2012/01/whats-needed-in-the-sims-4/)

2.1 Designing characters as a part of the overall design

As game character design is not completely separated from the other design areas of the game and its visuals, how should a character designer take the whole process into account while working? In his article 'Drawing Basics and Video Game Art: Character Design' (Drawing Basics and Video Game Art: Character Design, by Chris Solarski, 3.2.2013 http://www.gamasutra.com/view/feature/178656/sponsored_feature_drawing_basics_.php?page=2,), Chris Solarski shares his thoughts on how the character design process should be done: "*Develop cross-sections of the game rather than focusing on one aspect. This means developing the whole range of characters and environments in unison, rather than finishing one character or environment at a time.*" He then goes on to explain how this working method could lead to more consistent workflow among the whole development team: "Developing up from a high concept and a series of character

and environment concepts provides the entire team with common reference points against which to judge the suitability of design decisions. Only once you've defined your goal are you ready to begin the character development process.”

Solarski’s proposition of creating a wider concept and trying out sets of characters together with the other design elements within the game can help in understanding what is needed from the individual characters before getting into the finer details. This approach can make larger changes in the early stages of design much easier to implement but as pointed out earlier, the game development process is not that straightforward so this will not necessarily remove the need for all changes later on. Making initial plans for the whole game early on, however, will almost certainly keep the progress in character design more consistent and easier to maneuver.

What do I need to know about the other elements of the game before starting to design the character? A good starting point is to find out how the character is displayed to the player, whether it is a first-person or a third-person game, this information comes from the game design and is a good thing to know before starting to visualize the character design. 3D characters can be viewed in many different positions and angles as the game’s camera moves throughout the game world. This presents a challenge on how to choose which of the area’s in the character design needs the most focus to keep the character recognizable and visually at a high level in all the possible point of views.
To distribute the resources on designing the character on the most vital areas, mapping the possible viewpoints by picturing how the camera is positioned during gameplay and focusing the resources on those areas in the character design could be used to save time and resources as well as to keep the visual level high and consistent.

Many game worlds include more than one type of environment and in some cases the graphics might change quite dramatically for instance between different levels or stages. These variations should be considered also when designing the characters and they should be made so that they are suitable for all the separate parts of the game. In some games this might even mean changing the character's appearance along with the progress of the game but this has its own risks as the characters should usually be recognizable to the player at all times and it will increase the workload.

The relationship between different character designs is also crucial and even though the designs should all go well together, there should usually be some distinctive features for every individual character. There might be something about the character's personality or motive visible for the player through the design or there might be certain details on the character that are fundamental to the game itself and so need to be thought of in more detail and might have to be emphasized in some way. There could be a need for the player to be able to make a clear distinction between certain groups of characters. In some games this might mean the difference between a friend and a foe or in others it can be used to tell the player what kind of behaviour to expect from the certain character group. Designing the characters so that they are easily recognized as a part of a team or a group can be an important way of making the game experience more enjoyable for the player. Designing these visual distinctions may sometimes be tricky when, for example there’s a need to create a character that is a part of a certain team but at the same time needs to have an appearance that lets the player know that the character has a unique role or purpose within that team.

Designing multiple different characters requires a lot of resources, but sometimes the character design may allow it to be used among multiple characters without taking away anything from the game world's diversity. Such designs are for example characters with uniforms, apparel hiding recognizable individual areas such as faces, unknown life forms and animals, designs that are hard even in real life to tell apart. Re-using the same character design also provides information to the player of the characters possessing equal attributes. The player could learn what to expect from a character only by the way it looks and immediately know how to react. Just by differentiating the colors of the character, the player could see a distinction and would expect something new, thus giving the same character design a new value as a different entity with different attributes. This is an often used technique, for example in Halo 4 by 343 Industries (Microsoft Studios, 2012) the team multiplayer like in most team multiplayer games differentiate the teams with colors and the individuals in that team are separated with player customized armors and player names.

The character's purpose and role in the story of the game can be shown in the design in order to give that information to the player. If the player sees a large and muscular character in a game, he usually expects it to be strong and powerful and if the character is carrying a crossbow, it is expected to shoot with it instead of rushing into a melee battle.

In the same way, the character's personality, past and feelings can be made visible already in the visual design. An untrained, fearful civilian in a war game should look and be animated in ways that make those traits visible as opposed to a highly trained special forces veteran, who never runs from battle. However in games where the story is in a more pivotal point, for example such as point & click games, the personalities and past are something that should not be too self-evident to maintain the player's interest to the story by not knowing the motives and agendas of the characters from the first glimpse. If there's no background information about the character visible through its design, it can make the character more mysterious.

Jesse Schell brings up an interesting point in 'The Art of Game Design - A Book of Lenses' (2008, Morgan Kaufmann publishers, 314-328) about how in some games, the story is enhanced by transforming the main characters in a similar way that movies and books use. For example in the movie American History X, Derek Vinyard, a character played by Edward Norton goes through a transformation from a tough, violent and racist criminal into a remorseful and peaceful man who is concerned that his brother might become what he used to be. This transformation is not only portrayed by his behaviour and words in the movie but the character's appearance as well. His bald head gets covered by hair during the transformation and he starts to wear clothes that cover his tattoos including a swastika that plays a big role in the beginning of the movie. In the same way a game character can transform visually to bring forward other changes in the story or the environment within the game and changing their visual appearance can be even more effective way of narrating the change to the player. Changing the character along with the events in the game can make the story more meaningful and interesting to the player. In movies these changes affect the decisions and behaviour of the character but in a game world, those are sometimes directly controlled by the player so the visual clues play a much greater role. From a character's 3D designer's perspective this might mean that there needs to be variation or change in the character that is noticeable in the game but still lets the player recognize the character as the same person.

Keeping the character design up to date with latest game design changes and placing the resources where they are needed, the design choices can be reviewed inside the development team or with someone not familiar to the designs to test if they achieve what they were designed for. Sometimes it might be beneficial to question if something is

needed, and should it be designed at all if it does not bring anything new to the game or if it confuses the player as a something totally different from the rest of the design.

2.2 Drafting the character's visual appearance

Often albeit not always different people are in charge of different working areas on the same character; story, character design, 3D modeling, texturing, rigging, animation, so it is vital that all these people who work on the same character at some level at some point have a common visual goal for it in mind. The character needs to be visualised for the needs of the story, game design and style before starting to work on it in 3D. At this stage the visualisation should be done fast, easy and simple to keep it expendable as this is the best stage to make changes to the design and to create multiple different designs to choose from (See image: 3). After the 3D modeling has started it takes more time to make changes to the design and it may slow down the overall process.

(Image 3) Early orc sketches made for our project: "Orcs hogging hogs".

The character sketches are done in 2D, first drawing the overall visuals of the character design and then trying out different color themes. The designs are then reviewed and modified according to the results from the review. Once the design is completed, blue-prints or reference images of the character are drawn for the 3D modeler to easily trace the shapes in the 3D software to ensure the likeness is transformed from 2D to the 3D

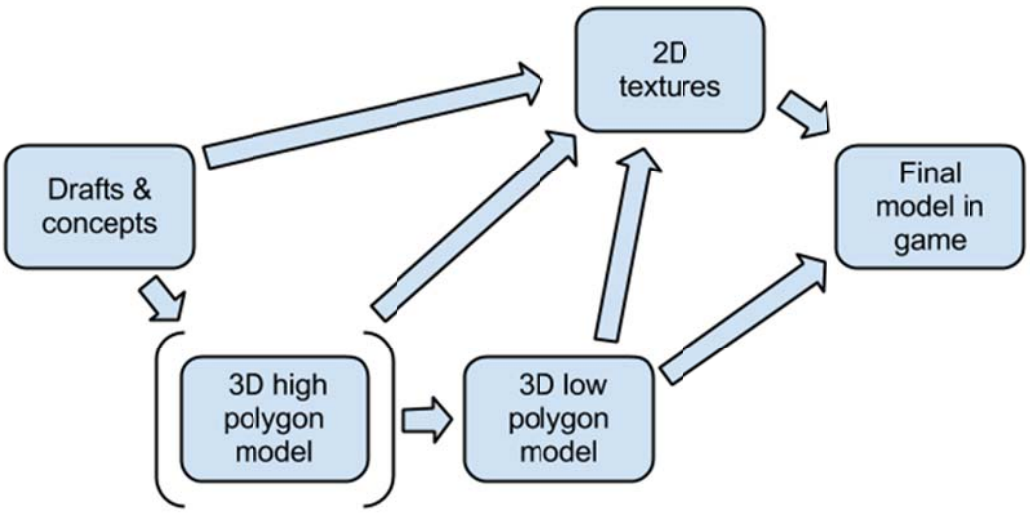
properly. The blueprints consist of at least the front and side profile of the character with separate reference images of details such as face or accessories. Often for additional accuracy for the conversion also top and 3 / 4 view reference images are included along with different poses to convey the way the character is meant to be presented (see image:4).

(Image 4) Face reference image of an orc.

3D can also be used for sketching, for some people it is easier to work in three-dimensional space than with traditional two-dimensional drawing methods. 3D sketching also produces more data if it is to be used as reference. It provides the whole coordinate info for the mesh, making it easier to get the likeness correct as the 3D-modeler can model the actual model on top of the sketch matching the mesh accurately. Traditional 3D modeling may be a bit too slow for this purpose, but using 3D sculpting in addition to the more traditional methods it is quite fast to visualize the basic shapes and dimensions of the character.

The main purpose of this stage is to get the character's appearance as finalized as possible, so the later phases won't be delayed because there may be changes needed to the design. The drafts and concept phase is the basis for the whole 3D character modeling process. At first, the drafts are utilized to create a high polygon 3D model of the character. From the high polygon 3D model, a low polygon version suitable for the game is created. The low polygon model is then used to create a UV map, a 2D image representation of the 3D model to draw the textures on. When the UV map is ready, the high polygon model can be used to create normal maps, a texture that can be used to create additional detail for the model without polygons. Drafts and concepts are used to help

creating the color scheme for the model's skin and clothing textures, which then are combined with the low polygon 3D model to a final model that will be used in game. Going back to correct mistakes in the drafts and concepts phase would affect all of the later phases creating a need to redo all of the later steps as well (See image: 5). However there are things that are hard to consider at the drafting stage, and they only appear during the process, such as technical limitations with the platform. This may cause the need of changes to the design and because of this, designing alternatives for the character in the initial phase is useful, as in the later phases when the needs arise, the changes may already have been made in another design. The alternative design can keep the process going without additional delay. These alternatives may also include clues for other possible future design changes and they will help to bring them out to the open for evaluation already in the beginning. Quality surpasses quantity also in this stage, but the quantity may have quality of its own as there have been more alternate routes explored.



(Image 5) The initial drafts and concepts are the basis for all the following phases in 3D game character creation.

As a way of saving the modeler from additional changes to the character and from redoing the same work later on in the modeling phase, drafting the character's appearance thoroughly is certainly an important way of optimizing the premise for game character modeling process.

3 TECHNICAL ASPECTS TO CONSIDER

3.1 Finding the most suitable game engine and 3D-software

A game engine is a set of code or a software system for creating and developing games for video game consoles and computers. A game engine functions as a framework for games to be built on and often includes tools and software used for the development.
In his article ‘Finding the game engine that can’ (Finding the game engine that can, by Paul Hyman, 2.2.2013
http://www.gamasutra.com/view/feature/132380/choosing_the_game_engine_that_can.php?page=3), Paul Hyman asks two game industry veterans, Alex Seropian and Ulf Andersson and an independent games consultant Mark DeLoura about how to decide on which game engine to use for a game project.
Seropian and Andersson seem to have quite different experiences on licensing game engines or creating one for your game, but both agree that developing a completely new engine is something that takes a lot of time and money and creates great risks. Seropian has been involved in various game projects in which the company has been developing game engines or additions to them but also licensed readymade engines. Based on his experience he says "*While some developers prefer to stick to one engine for all their projects, we're not afraid to use the right tool for the right job.*" Andersson on the other hand started developing a game engine with a company of two people when licensing game engines was much more expensive in 1997. At the time the article was written in 2009, he was still using and developing it with a team of 270 people. He explains the decision by saying:”*We have a very flexible platform that we can create anything with and not be one step behind because someone else decided what's going to be in the engine or not.*” Andersson however recommends new companies to rather license than build a game engine.

Mark DeLoura has surveyed a group of 100 developers (http://www.gamasutra.com/blogs/MarkDeLoura/20090302/581/The_Engine_Survey_General_results.php) and says that”55% are licensing someone else's engine” and that "*About 46% of those surveyed said that if they had their druthers and an infinite amount of time and money, they would prefer to create all their own tech*”. Deloura's poll also states that the primary motivations on choosing a game engine are”money, time, and whether the strategy is going to work for the game”.

Based on both DeLoura’s and Hyman’s articles and our own experience, developing a game engine from scratch only becomes a good option when there are a lot of resources behind the project or if the game is simple enough to allow the creation of a very simple engine. With a smaller development team, finding the best option from existing engines is in most cases much cheaper and less risky and as such usually a better way to develop games.

There are hundreds of game engines available from prices ranging from free to several hundreds of thousands depending on the licence type. Using the same engine from one project to another can save time as the tools are familiar and even using assets from previous projects becomes an option, but this approach can also make a single project more difficult as it might not be optimized for the same type of games.

The decision between game engines should be considered on the basis that it should make the development process as fluent as possible so in addition to the limitation from the game's perspective, the developers’ preferred working methods should also be a factor. The game design and the engine should be compared to see if the engine can offer the features needed and to figure out if there are some parts that would still have to be modified or developed separately. If the game is simple enough, it might also be a good idea to not use game engines that would be too complex, as that can make the advantages seem smaller in comparison when more work needs to be done just because the engine was designed for larger scale game projects. The game engine to be used also has an effect on the modeling software through compatibility issues and as such both should be considered together.

The most important aspect of choosing the software is that it works well together with the game engine since the models need to be able to be saved or exported to a format that the game engine supports.
Modern 3D software have fairly extensive set of export options, but when using a tool that has younger or smaller community and/or resources for development there may be problems with the exporting tools, which may cause errors when importing to the game engine. Sometimes these incompatibility issues might be unavoidable or too late to fix by switching programs, in which case it can be useful to try to find workarounds by adding an additional converter into the mix. This can be achieved for example by ex-

porting and importing the asset to another 3D-software from which it can be exported again to a compatible format for the game engine.

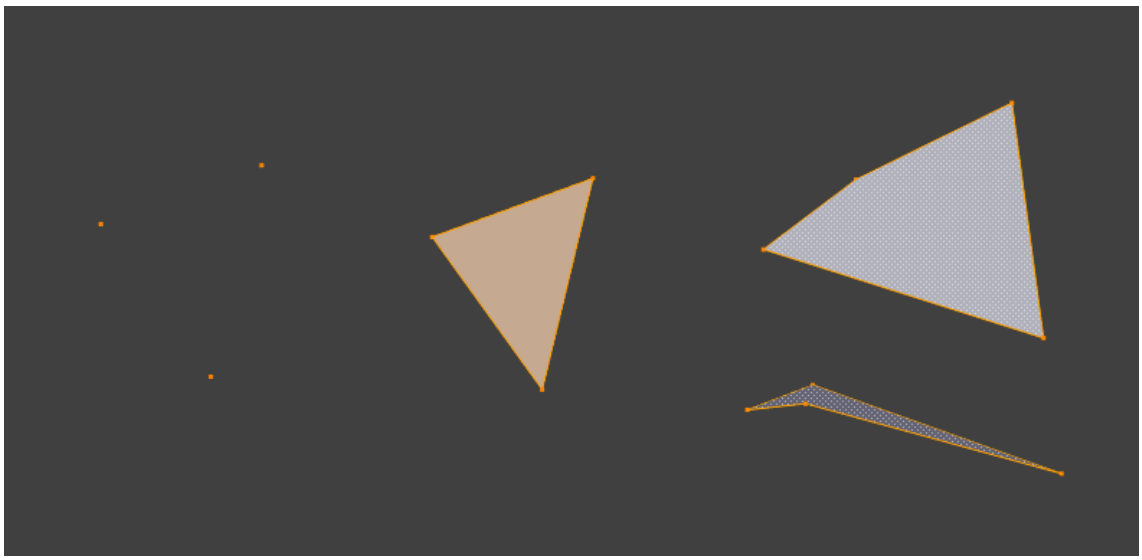
There are number of different platforms on the market with different technical limitations. This may affect which game engines can be used for the development. The current trend in the game engine industry has been the ability to output the finalized game for multiple platforms automatically. In the future this might mean that choosing the platform won't restrict which engines can be used. The output platform won't limit your choice on the 3D software as long as it is able to save in a format that the game engine of choice supports.

In Orcs hogging hogs we had an idea to later try the game in mobile platforms, so we chose a game engine that would support multiple platforms. Thus we ended up with Unity, which is currently receiving plenty of attention from indie developers and bigger commercial studios as it offers highly competitive features and prices. And as for our primary 3D modeling software, since we had previous experience with the tool we chose open-source software: Blender 3D. Both tools work well together and with our limited monetary resources they seemed like valid choises.

3.2 Polygonal limits

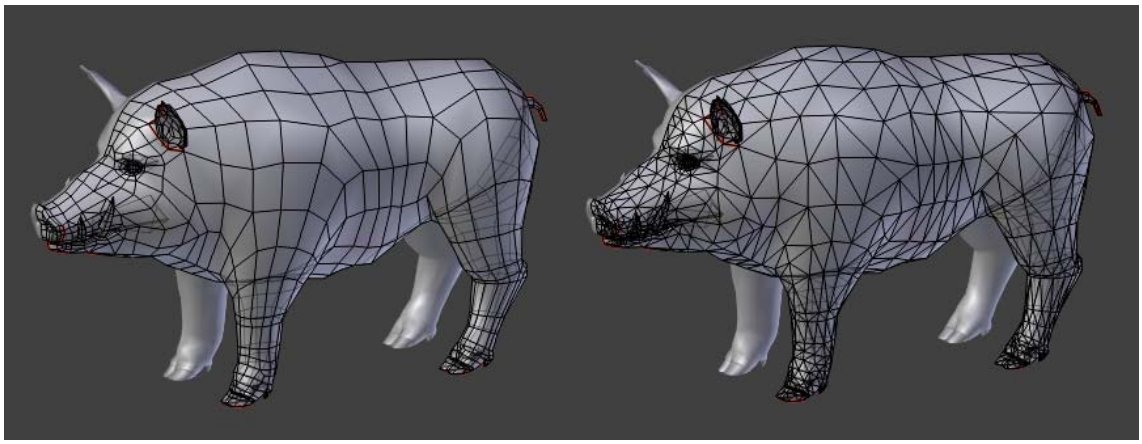
Vertices are to 3D games as atoms are for the physical world. Vertices themselves are not visible per se; they need to be connected together to form a polygon, a flat shape with the vertices acting as the corner points. As the vertices are placed on the X-, Y- and Z-coordinates and connected, they form 3-dimensional shapes that can then be expanded with more polygons resulting in 3D models that will ultimately populate the game world.

All polygons can be broken down to triangles, which only have 3 vertices, and the three vertices always fall on the same plane when they are connected. In a polygon with more than 3 vertices connected together the vertices may be on different planes (see image: 6), which makes them harder to calculate. In game development the models usually need to be triangles as they are easier for the game engine to calculate and therefore makes the game run faster.



(Image 6) In a triangle all vertices are on the same plane. In a polygon with more than 3 vertices one or more vertices can be on a different plane.

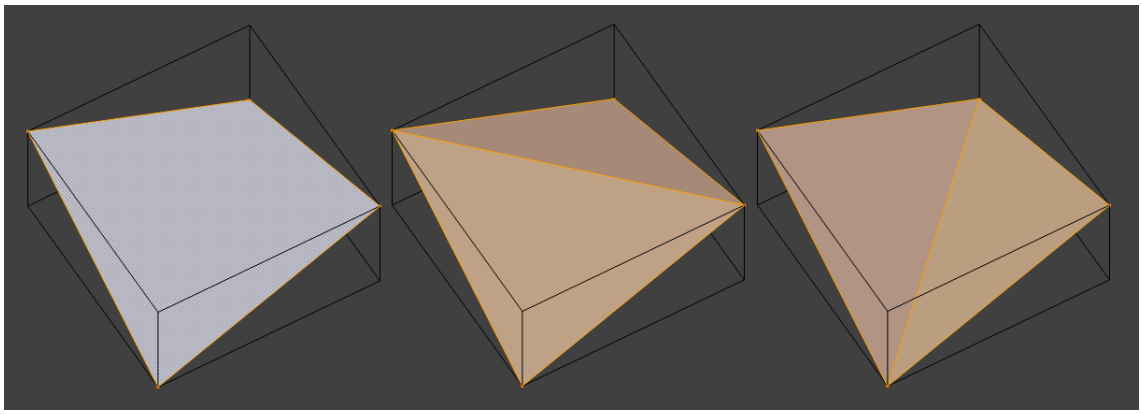
Polygons with more than 3 vertices do however serve a purpose in 3D-modeling process for games. Polygons with four vertices, also known as quads, function well with several different selection and transform techniques and tools used by the modern 3D -software, which will speed up the modeling process. Quads also make the model easier for the eye as it simplifies what the artist sees in the software (Image: 7), which in turn makes it easier to spot problems with the mesh and to fix them. This is why most artists usually preserve the quad polygons in the model as long as it is in the making, when it is ready to be imported to the game engine the quads are converted to triangles.



(Image 7) On the left, a hog model for “Orcs hogging hogs” with quad polygons and on the right the hog after converting the quads to triangles.

The game engines make the conversion from quads to triangles automatically, however this may sometimes result to undesired triangulation layout (See image: 8). A simple

quad can be converted to triangles in two ways that will result in two different outcomes that can have a dramatic impact on the appearance of the model. If there are problems with the conversion the changes to the converted model or a new conversion can be done manually in the 3D modeling software. Using more than 4 vertices in a polygon may lead to more complex undesired triangle layouts that can be time consuming to fix, so the artist should limit the use in triangles and quads.



(Image 8) A quad polygon converted to triangles can have two possible triangulation layouts creating either convex shape or a concave shape.

Rendering the vertices and polygons requires a reasonable amount of computing power and memory from the hardware. This is why there needs to be polygonal limits for the game models so that the hardware and game engine are able to run the game smoothly without compromising the gameplay performance. Before starting the actual 3D-modeling process, it is useful to know the approximate polygon limit for the models so it will not slow down the game while still having a high enough level of detail.

Polygon limits are progressively increasing as the hardware gets better and the game engines utilize new technologies more efficiently. For example mobile platforms such as Android and iOS are now able to produce 3D graphics nearly comparable to home consoles when just a decade ago the graphics on mobile platforms were limited to low resolution two-dimensional graphics. Knowing the platform and its hardware capabilities is important when considering the polygon limit. In addition to the hardware data, benchmarking is an excellent source of information, as other games, already produced for the platform, may give valuable clues for the possible limits.

Polygon count is not the only thing that affects the 3D model's performance hit on a game. A draw call is a function used by a program to tell the computer's graphics processing unit to draw something. Increasing their amount is usually the most costly operation that the character model may end up causing. This happens for instance when the character has more than one shader in use or consists of more than one mesh. Also in many cases, counting the vertices might be more essential measurement than polygons, as in a game engine vertices do not necessarily follow the expectations of the modeler. Many of the vertices that are shown as a single point in the modeling software might end up being considered two or more vertices in the final game as they might have to be processed separately for each UV-map border, material change or a switch between smoothing groups. Nevertheless, reducing the number of polygons and avoiding unnecessary ones is an advisable way to optimize the character modeling workflow.

4 MODELING CHARACTERS FOR GAMES

Heidi Landgraf quotes animator Dave Vasquez in her article, the increasing role of character animation in games: "*Your animation can usually be seen from all angles in the game so it must look correct from all points of view*" (The increasing role of character animation in games, Heidi Landgraf, 3.3.2013, http://www.animationarena.com/character-animation.html). Even though Vasquez is talking about animation specifically, the same reasoning can be used to all dynamic character models and their use in games.

A dynamic model is a movable and interactive model in the game world, as opposed to a static model, which is stationary and unchangeable.

When creating character models for games there is much less control over how the model is going to be seen or used compared to non-interactive use cases such as animation. In a game world, the character can usually move almost unrestrictedly, so often the graphics cannot be pre-calculated and rendered into readymade images or videos but instead the lighting, textures and most of the other effects have to be rendered in real time. This real time rendering is in most cases the bottleneck that causes performance problems with games. Animations, movie special effects and such on the other hand can use all of the available time and resources to render everything beforehand so they can use much more detailed and less cost-effective methods with modeling (see image: 9).

(Image 9) A comparison with pre-rendered and real time graphics. Images from Mass Effect 3 by Bioware (Electronic Arts, 2012).

Modeling a game character can still include making highly detailed models as well, but they are usually only used to make the actual final models in the game engine look better by using them to create the textures to get a more detailed result, especially with the lighting effects. The actual lower detail characters however are still the ones that the game renderer uses together with the textures to create the final results seen on the screen.

Texturing plays a really big role with game characters and the texturing should be considered a part of the whole modeling process as well, so that textures and the 3D meshes make use of each other in the best possible way. Textures are always an important part of the process but with games it becomes especially crucial to think of ways to use the topology of the model and the textures together to bring out the major shapes and to leave the finest details to texturing as it is the more cost effective way of doing it.

Because of the limits in polygon use in the game character models, their placement should also be more thoroughly thought and made sure that the polygons are placed to areas that need them the most. The need can come from simply just the shape of the character, but it can also be important for the animation and rigging process, so placing the polygons can be quite tricky if not planned properly.

4.1 Topology

Antony Ward writes in his article ‘Create the best character models for games’ about creating optimal topology for game character models: ‘Being handed a model with random edges and vertices placed across the surface not only looks awful, but makes any future tasks more difficult, and more than often means the model has to be rebuilt or edited. Remove anything that isn’t directly needed by the model and won’t enhance the way it looks or deforms.’ (Antony Ward, 24.02.2013, http://www.3dworldmag.com/2011/09/21/create-the-best-character-models-for-games)

In 3D modeling, topology refers to the models’ mesh layout and the positioning of the vertices, edges and polygons on its surface. Well-structured character topology is used to make sure that the real time rendering costs remain as low as possible while at the same time the deformations caused by the animations work in a desired way without causing problems. A good topology or “mesh flow” is also important for the artist to easily modify, add and or remove parts from the model. The greatest challenge usually is to keep the good deformations and the amount of polygons in balance. The topology of a game character can also be very dependent on the game environment and how the game is played. The character's size and distance related to the game camera position can often affect the model's polygon count quite a lot since the further the details are on a model, the harder it is to distinguish them and so they can become irrelevant.

With games, defining good character topology can be quite tricky, since the optimal amount of polygons can vary greatly depending on the game platform, the other objects within the game as well as the character in question. In general, there are still ways of defining what parts of the character's mesh require more polygons and which parts don't play such a great role. When making decisions on the mesh topology of a game character, it is important to know how the model is to be used in the game. Some decisions can be made by asking questions such as: What parts are to be animated, are some more

visible than the others or are some of them more important for the player? There even might be some parts of models - at least on first person characters - that are never seen by the player and so can be left with fewer polygons or even with none at all.

The individual polygons within the character model can sometimes cause problems because of poor mesh topology. Some of these problems are not easy to spot before the character is already placed into the game, since in most 3D modeling programs things such as z-fighting or polygon's normal directions are not easily visible. Z-fighting occurs when two polygons are overlapping and the graphic card is trying to render them at the same place and this causes the two polygons to flicker. In the modeling program the polygons are usually also shown as two-sided, even though in reality they have a direction in which the normal of the polygon is facing. In most cases the game engine renders the polygons as one sided and so if some of the polygon normals are not properly placed, they appear invisible in the game and may cause holes in the meshes as well as lighting problems as the lights and shadows are trying to affect the opposite side of the polygon. (See image: 10).

(Image 10) On the left some of the polygon normals are facing the wrong direction. On the right all the normals are facing the same direction

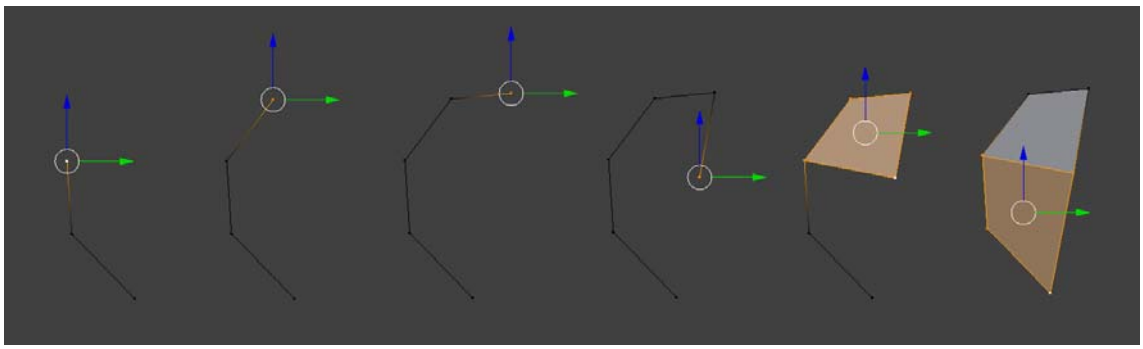
Optimizing character topology also affects character movement, animations, textures and the game performance and so if it is not done properly, there can be a negative ef-

fect on any of the following tasks and the whole process of game character creation. Working on topology carefully while keeping the rest of the character creation phases in mind can make a big difference in the overall progress.

4.1.1 Finding the most suitable techniques

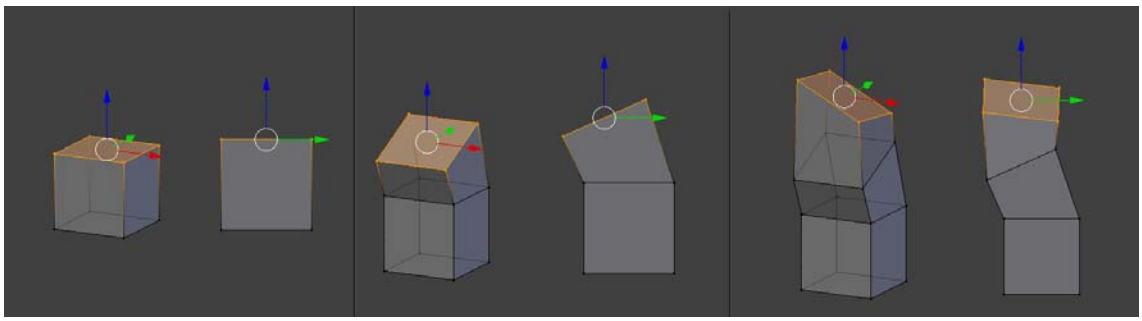
3D modeling can be done using various different techniques and working methods. Some of them can be more useful for game character modeling than others and having an understanding of the different ways in which the work can be done is important, as it can optimize efficiency even when the end result would be similar. We will go through many of the different approaches in this chapter.

Vertex by vertex modeling means extruding one vertex from another vertex one at a time and moving it to the correct position. This technique is extremely slow and it offers a poor overall view of the upcoming topology for the artist. Even though this technique is not really a good overall for modeling, it is a useful supporting technique when having to deal with complex details and correcting topology later in the modeling process. (See image: 11)



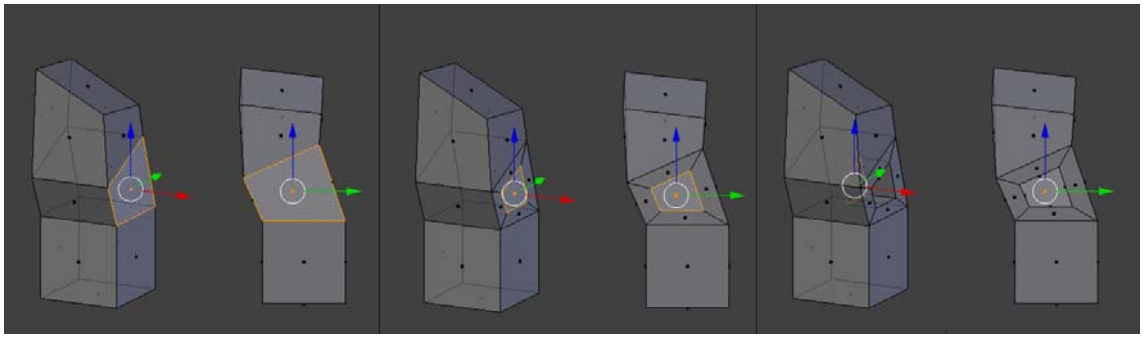
(Image 11) Vertex by vertex modeling

Box modeling is one of the easiest and fastest ways to start modeling as it allows a fast way to create a rough final version of the model. The technique consists of modifying a primitive shape such as a cube, sphere, cylinder, etc. By extruding the shape, the modeler can quickly fill in the basic figure of the model keeping the polygon count low and the topology of the 3D mesh clean. Box modeling is probably one of the most commonly used polygonal modeling techniques. (See image: 12)



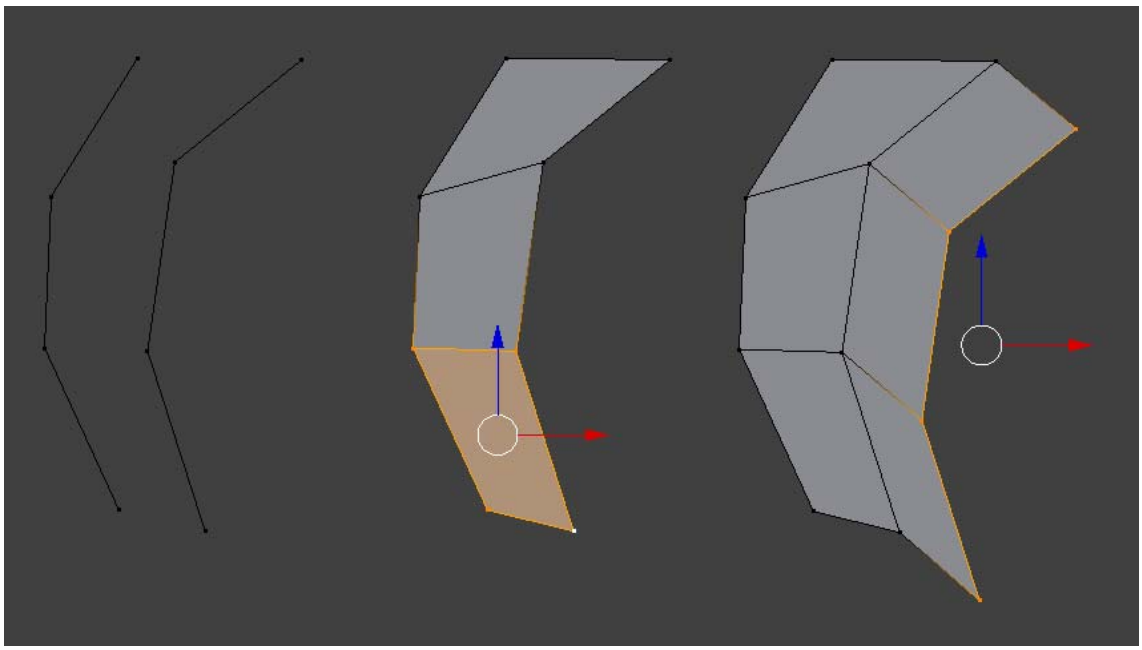
(Image 12) Box modeling

Subdivision modeling is a supporting technique often used in combination with box-modeling and other techniques to refine an already existing model topology by subdividing areas that need more polygons for mesh details or animation purposes.(See image: 13)



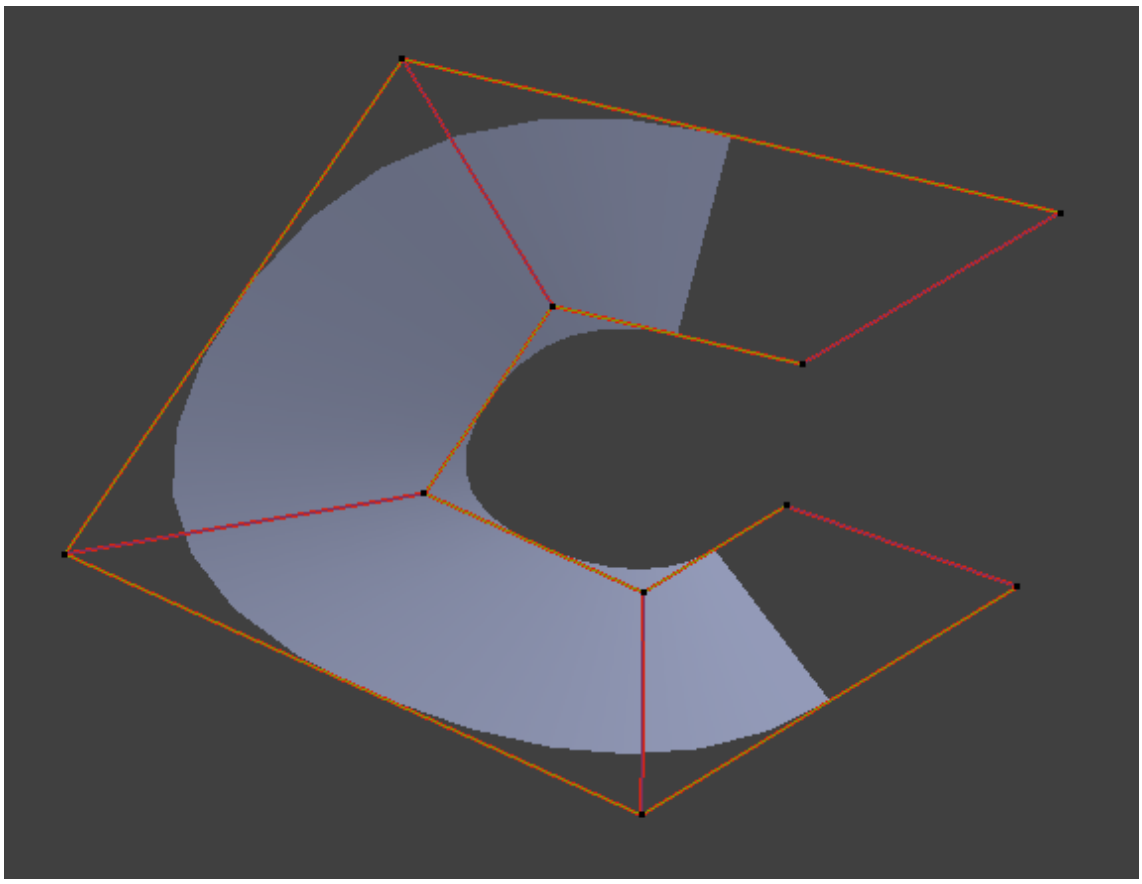
(Image 13) Subdivision modeling

Edge/Contour Modeling is a technique in which the model is built piece by piece by creating edge loops, placing them in prominent contours of the model and then connecting the loops to each other to form polygon faces. This is slightly more advanced and lower speed technique than box modeling, but it offers easier and more control over details when modeling difficult surfaces such as human faces. The loops can also be extruded and moved in the proper position in a similar way as in box modeling, saving time from connecting the vertices one by one. (See image: 14)



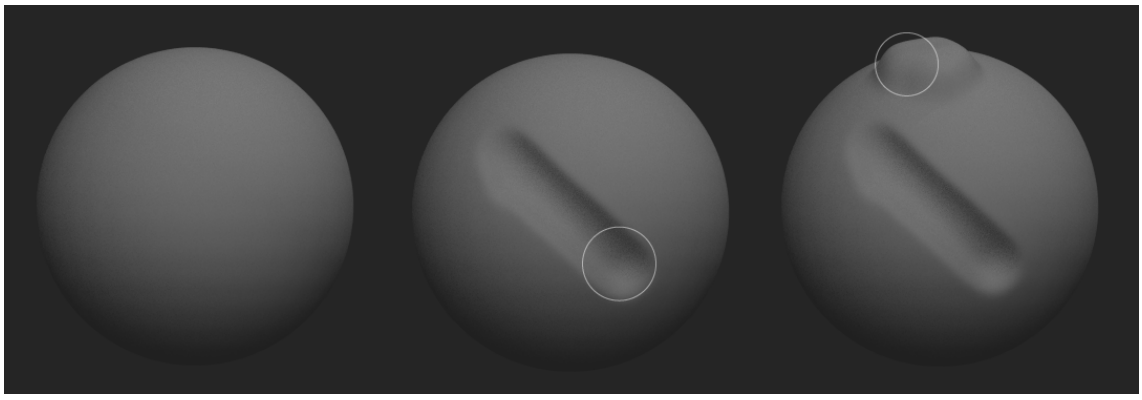
(Image 14) Edge/contour modeling

NURBS or **Spline Modeling** is a technique that uses Bezier curves to create smooth surfaces without vertices and polygons. This technique is mostly used in CAD- modeling, and sometimes also with animation. CAD-modeling is used for industrial design purposes like designing a 3D model blueprint for a product in making and it can also be used to actually print the product in industrial 3D printers. This technique however is not suitable for game modeling, as currently there are no game engines that support models with NURBS. There are ways to convert from NURBS to polygons, but as the NURBS rely on Bezier curves making the surfaces smooth, the conversion might come out as unusable because of the high polygon count. A model made with NURBS can however be used as a reference for polygon modeling, but this however is quite redundant as the same work would be done twice. (See image: 15)



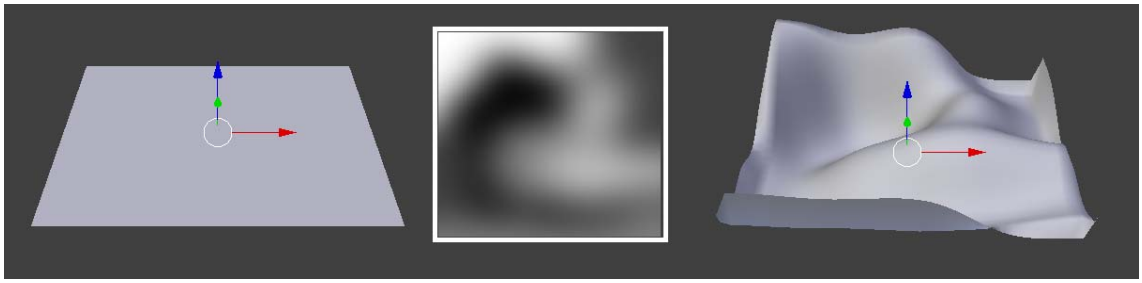
(Image 15) NURBS / Spline modeling

Digital sculpting is a technique where the artist uses different kind of brush tools to push and pull vertices and polygons as if sculpting digital clay. For game development sculpting is usually used for creating highly detailed models from already modeled lower polygon counterparts and using these sculpts as bump or normal maps for the lower polygon models to make them appear more detailed than they actually are. (See image: 16)



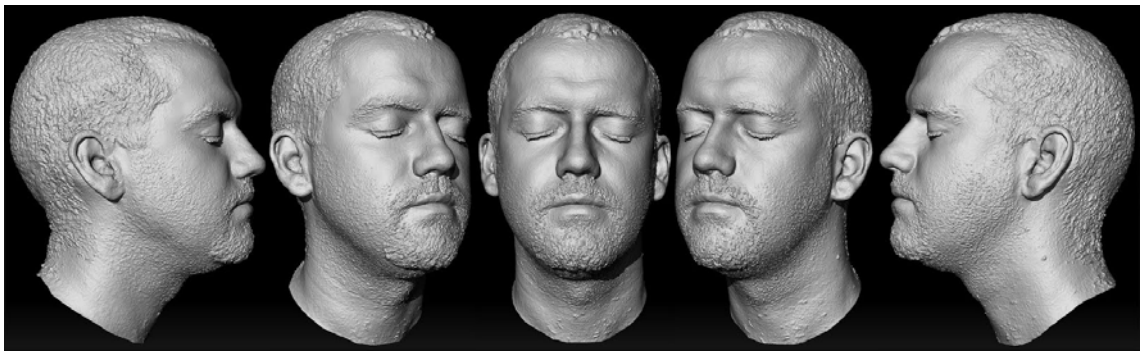
(Image 16) Digital sculpting

Procedural modeling is an algorithmic creation of scenes or objects based on different user definable variables and parameters. Procedural modeling is something totally different from the other techniques listed here, it is not actual visual 3D modeling, and it consists of creating a code or using a premade generator which will create the visuals according to the user input. One of the most common areas this technique is used in game developing is generating terrains by inputting an image file containing height values, black being the lowest height value and white being the highest. (See image: 17)



(Image 17) Terrain created with procedural modeling

3D Scanning is a technique that can scan real-life animate or inanimate objects into digital world as 3-dimensional data. The downside is that these scans are not usable per se for the game world as they use an overly high amount of vertices and polygons for the real-life to digital conversion by recreating every little bump and hair in 3D. The 3D data gained from the scan can however be used for reference, as the artist can start modeling the model on top of these highly accurate scans, which is a good way to get the appearance of a character spot on. (See image: 18)



(Image 18) 3D face scans by Infinite-Realities (http://www.ir-ltd.net/do-you-wanna-live-for-ever, 26.02.2013)

Before starting to model it is crucial to also consider what can and should be achieved with the mesh and what can and should be achieved with textures.

Sometimes using two-dimensional images to create features on a three-dimensional model or using the images as actual models may be better than actually modeling them, time and polygon wise, but the downside is that the images do not have any actual depth and they should only be used in places where the player won't be able to move around them or with matters that do not require much depth such as grass, paper or with character's hair.

Image based modeling technique however can create a depth map from the image, creating a three-dimensional relief model. This can be used to add depth to the image, but the downside to this technique is that as 2D images have only two-dimensional data it doesn't have information to create any depth in the backside of the model. This would allow the model to be viewed only within a certain degrees without revealing the lack of mesh. This technique might use, depending on the image, too much polygons which makes it not the best approach for using in game development. But using texturing techniques such as bump or normal mapping which create a fake 3D effect on the mesh's surface and simple mesh, the same result can be achieved with lower polygons.

The technique to choose for creating a 3D model depends on the comfort of use and what will be the use of the model once finished. It is very useful to flesh out the model before going into the details and this might be why box modeling has grown to be so popular. But once the process goes into the details this technique starts to be too plain to be used constructing advanced mesh topology. Relying only in one technique makes the quality and speed of the process uneven. This suggests that using a hybrid workflow of multiple different techniques in the different phases of the process, starting from techniques to create the big picture and then going to techniques more suitable for details and refining, which will help to smoothen the process and create a more well-rounded 3D model.

4.1.2 Modeling an animated character

Since the process of modeling an animated 3D game character is so closely related to the process of rigging and animating, the modeler should be aware of the limitations and aspects of the mesh topology that are set by the way in which the animations are to be done. In cases, where the modeling, rigging and animation are done by different people, there should be close collaboration as the mesh topology can have a major effect on

anyone’s work flow and changes in the mesh topology might affect all the related work as well.

Animation is one if the issues that Petri Lankoski brings up when he writes about the players relationship to the player character throughout his book ‘Character-driven game design’ (2010, Aalto University). He writes that the player can connect to game characters through empathy and make the gaming experience much deeper this way. One of the main ways that this reaction can be implemented through the character is to make it seem relatable, natural and human such as using its appearance together with gestures and animations in order to mimic our human interactions. As the word animation implies, it makes the character seem alive, which helps the players react to it in a desired way.

As brought up in Riccard Linde's book 'Game art: Creation, direction and careers', the quality of game character models and animations keeps going up as the computers become more powerful, but still they remain far more restricted than in for example animated movies. Rather, new possibilities in game character animation such as game physics, motion capture and animation blending keep making the game animation process more and more detailed and complex and at the same time they differentiate the process compared to pre-rendered animation process. As the bone placement for the rigs or skeletons of the character and their hierarchy become more detailed and character specific, their relationship with the model's topology also develops further and the modeler's understanding of the animation process becomes more important.

One of the things that greatly affect the way a game characters topology should be constructed is what kind of character it is, how it is expected to move and how it should be animated. Mechanical or robotic characters usually do not need as much polygons in the joint areas since there is probably no need for mesh deformation. In organic models, for instance humanoids, joint areas are meant to bend more in ways that mimic the real life movements of the muscles, bones and skin. Sometimes the character might also be a mix of these two types or even something very different, like a cartoon style model with evenly bending arms or legs with no visible joints in them at all. Keeping in mind the goal of the character’s animations can make the decisions about the character topology much clearer and easier. Animation workflow can also be affected by mesh topology and it is relevant to know in which way the animations will be made. Many times in

more realistic games, the animations are made with motion capture technology using real actors so the movements should follow the same limits and possibilities as the human body. But if the animations are made in a more traditional way by hand, the animator will have a greater role in deciding how the character moves.

When the animation needs for a character are defined, it helps to prioritize the time and resources for different areas. A character with facial animation needs requires more polygons and time to construct the correct mesh loops in the facial area, while a character with no facial animation can be managed with a considerably fewer amount of polygons. The facial rig and mesh are also typically the most detailed ones on the characters - at least if the goal is to have a somewhat realistic character - since as humans, the players are very good at reading facial expressions and can find even slight errors there disturbing. This also makes leaving out the facial animations of a character sometimes a great way of reducing the polygon count so that other parts of the model may have more mesh topology.

“As you build, keep thinking ahead. As your character moves around in the virtual world, it will look better and deform in a more convincing way if your topology mimics the natural muscle layout of a real person.”
(Antony Ward, 24.02.2013, http://www.3dworldmag.com/2011/09/21/create-the-best-character-models-for-games)

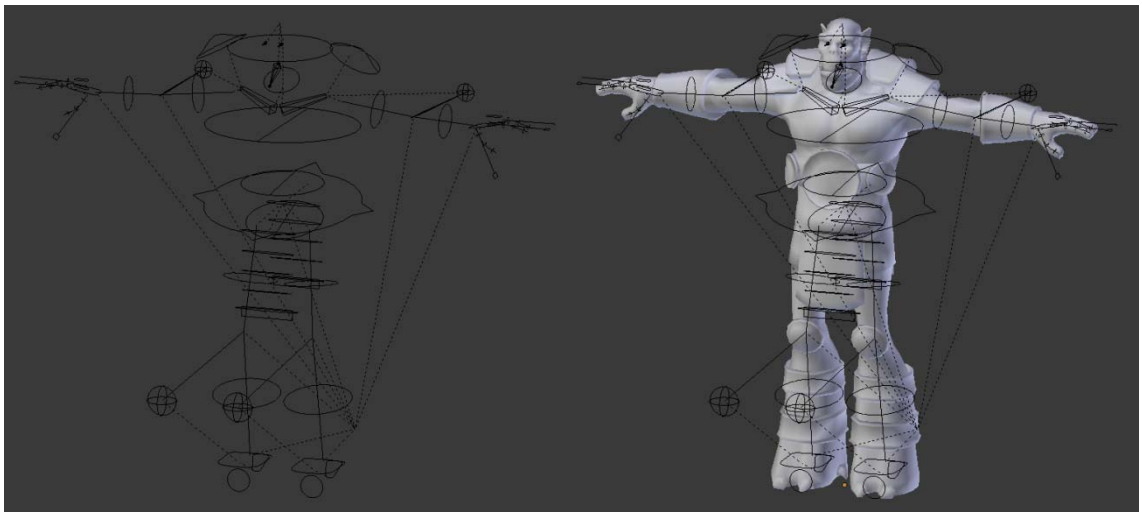
When modeling an animated character, it becomes important to know what kind of movement, poses and anatomy the character will have. The constraints in certain joints can mean that the mesh should not be completely symmetrical and for instance when modeling a realistic human knee, the front and back sides deform in a very different way. In general, the topology works better when there are more polygons used in the areas that have joints in the rig but how those polygons are added can be also defined better by how much and in which directions the mesh is meant to move and deform. With human models, there are already general guidelines and very specific information available on the optimal ways of modeling different body parts so that they work well with animations. Still whenever the character differs from this most basic humanoid type, the topology might not work in the same way and even slightly different body types with human characters can redefine the best approach on the topology.

Human or humanoid characters resembling humans need to be able to bend correctly from the limbs, as the players have knowledge on how their own and other people's bodies move, they are adept at noticing unnatural movement in a character and this may easily break the illusion of immersive game.

Bad quality of animation is often the reason for unnatural movement, but to be able to provide an environment for good animation there needs to be a properly working rig for the model. A rig is a skeleton that will determine the movement of different parts of the character. For a proper rig, the 3D mesh of the character needs to be modeled in a way that it will deform correctly when individual bones of the rig are moved, so the movement will happen in a natural looking way.

There are various different ways of making rigs. Simple rigs make the animation process easier and faster, but offer little control over small details that would help in creating a believable animation (see image 19). Sometimes some mesh problems are possible to solve with a rig, but it makes it more complex and possibly harder to animate. More complex rigs might be more work to animate but as they offer more control over the animation and movement of the mesh they can produce more believable animation (see image 20).

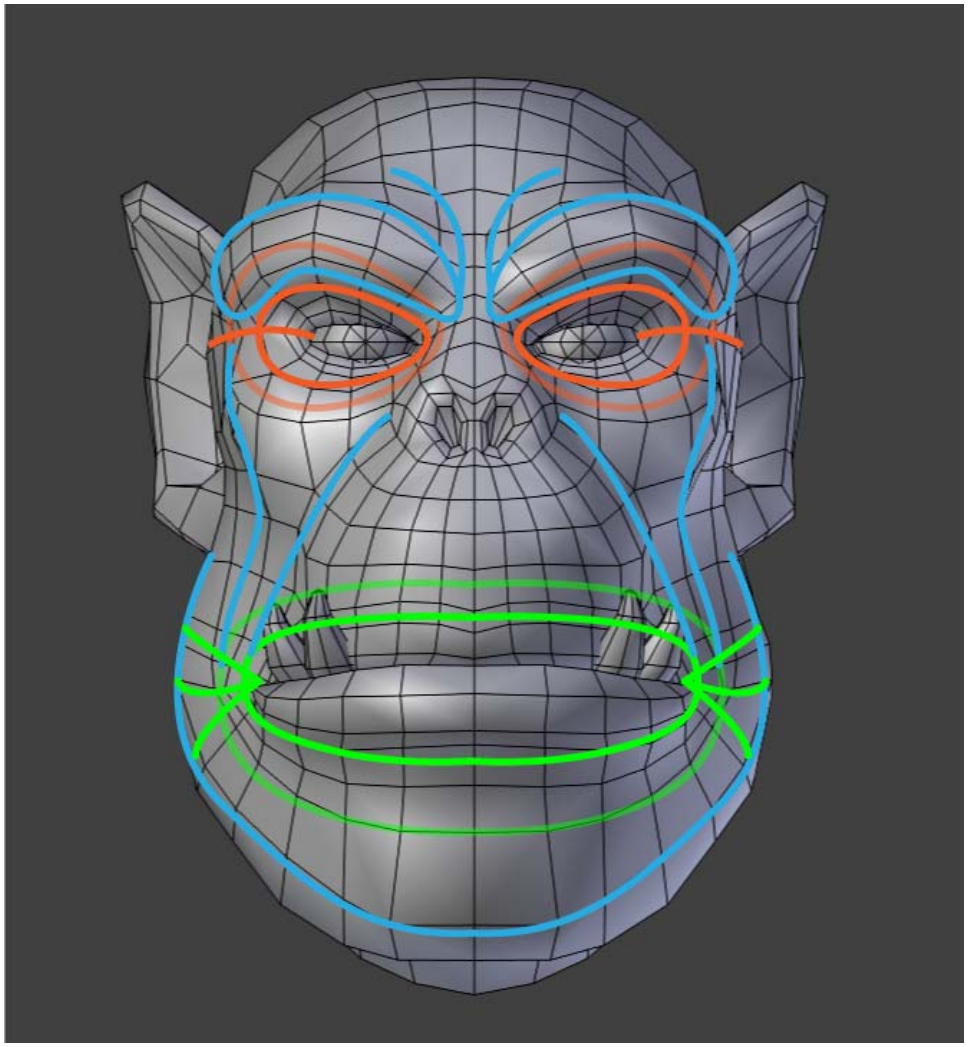
(Image 19) A simple rig showing only the most basic bones.



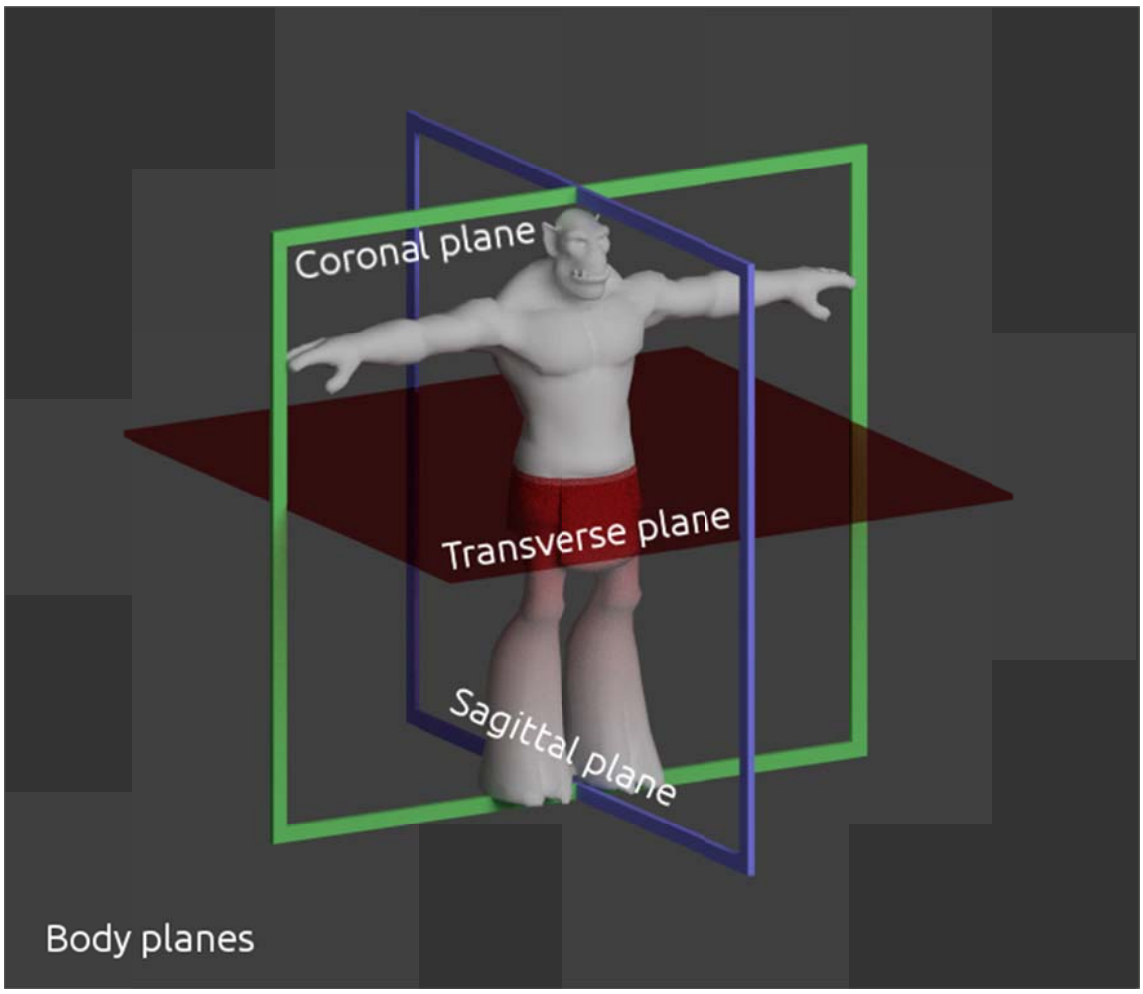
(Image 20) A more complex rig setup, which we used for the orc model and the gear he is wearing.

The head, and more precisely the face, is probably the hardest part to model for rigging. To be able to animate the face naturally, it needs more bones in the rig than any other part of the body and for the bones to work properly, it needs a complex mesh with enough polygons and correctly aligned mesh loops. The complexity of the face setup depends on the needs, if there is only need for some parts of the face to be able to move, the rig could be quite simple with less topology. The mesh loops in the face area overlap

on many areas and that’s why it’s important to make sure the mesh from brows, eyes, nose, chin, cheeks and lips merges smoothly to provide a good base to start rigging. (See image: 21)

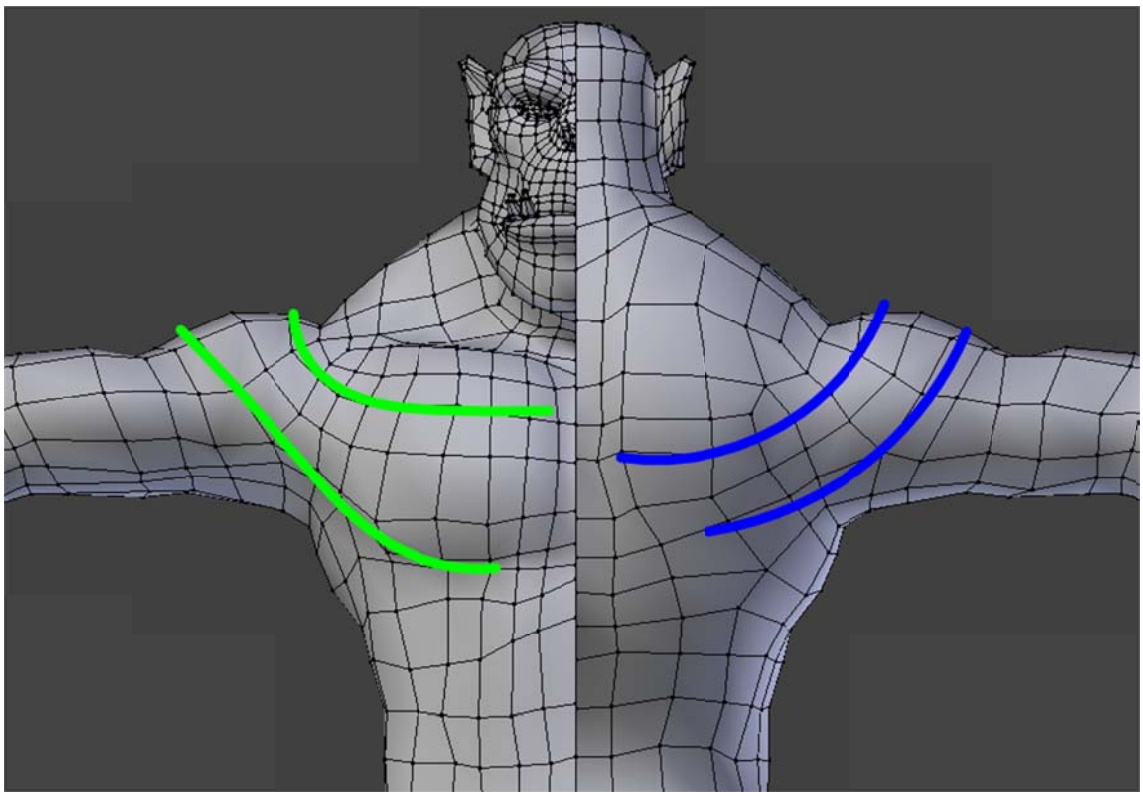


(Image 21) Orc model’s mesh loops in face area



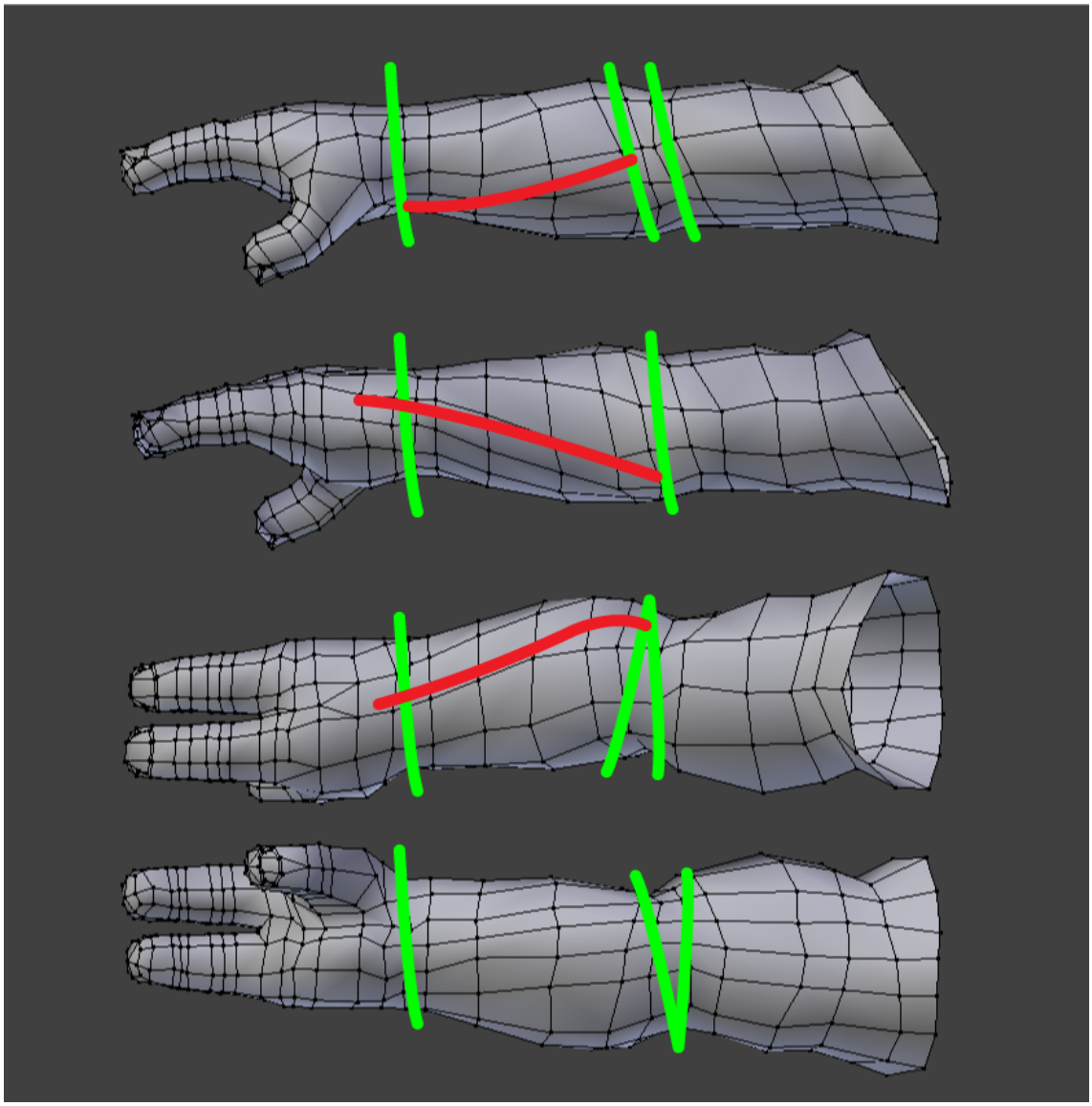
(Image 22) Body planes to explain the movement direction of joints. Green: Coronal plane, blue: sagittal plane and red: transverse plane.

The arms have three joints; the shoulder joint, the elbow joint and the wrist joint. The shoulder joint is a ball joint and it should allow forward-backward motion in the sagittal plane, sideward motion in the coronal plane and twisting along the arm axis (See image: 22). When looking at a real shoulder and its movement, the shoulder movement also affects the back and chest muscles, which should be kept in mind when creating the shoulder mesh to allow realistic rigging and animating of the shoulder. The mesh in the deltoid area should loop with the chest and back mesh to make them move more naturally with the shoulder (see image 23).



(Image 23) Shoulder area mesh loops from front and back.

The elbow joint allows flexion-extension of the arm in the sagittal plane and twisting along the arm axis. The wrist joint allows rotation of the hand in the coronal and sagittal planes. (See image 24).



(Image 24) Elbow and wrist mesh loops.

The hand area has only finger joints, all the fingers have three joints on the three knuckles that allow flexing to close the knuckle. The first knuckle joint on all fingers also allows small pivoting rotation for the finger. Mesh deformation here is quite simple, as all of the knuckles can rotate only about at a 90° degree angle in one axis and as the pivoting rotation of the first knuckle joint happens only in a minor angle, so it doesn't require a special loop in the mesh.

The torso usually consists of neck, shoulders, waist and pelvis. The neck rotates only in transverse plane with a small range of motion, so its mesh can be quite basic and low on polygons. The shoulder joint allows the up and down motion for the scapula and clavicle in the coronal plane and rotation in the transverse plane. The waist works similar to neck area with a little larger range of motion; it allows rotation in the transverse plane

bending the upper body. The waist doesn't need any additional mesh detail as long as there are few mesh loops to smooth the bending. The pelvis area allows forward-backward motion in the sagittal plane, rotation along the body's vertical axis and side to side swinging in the coronal plane.

The legs consist of three joints; the hip joint, the knee joint and the ankle joint. The legs are somewhat similar to the arms, but they have more restrictions in movement making them easier to model for rigging. The hip joint allows forward-backward rotation in the sagittal plane, sideward opening in the coronal plane and rotating along the thigh axis.
The knee joint allows flexion-extension of the leg in the sagittal plane and rotation along the shin axis. From the mesh deform point-of-view the crucial area is in the front and back part of the knee. The mesh in the back of the knee should collapse firmly but still keep the kneecap in the front in a correct form.
The ankle joint allows twisting and flexion-extension of the foot in the sagittal plane. The twisting of the ankle is limited so there is no real need to focus on that movement too much when creating the mesh. The flexion-extension movement should be kept as the main priority.

The foot part of the character usually only consists of the heel and the toes, and their movement is restricted to the sagittal plane. The toes are usually never rigged separately for games, as animating each individual toe rarely is necessary. Rigging and modeling the toes as one joint saves time and trouble, as they don't give much to the expression of the character and might not even be visible in the game because of the character's possible shoes and the overall visibility of individual toes, as they are usually the most distant part of the character's anatomy from the player's point of view. The mesh for the feet doesn't need that much focus on the moving aspect because of its simplicity, as long as there are enough polygons in the joint areas to allow the moment.

Modeling the joints or areas that need to be animated is a very important part of the modeling process, as it is something that will directly affect the results later on when rigging and animating. The mesh in those areas should be thought out carefully, so that it works properly with the rig before proceeding to animation or texture processes.

4.1.3 Setting up the model for texturing

In addition to animation and topology, texturing can also make a difference in the modeling process. The 3D mesh of a character model will have to be UV-mapped into 2D space in order to use texture images on its surface. UV-mapping is a process where the polygons of the model are divided into groups of polygons or polygon islands that are then placed onto a canvas for texturing. In game development texture images should follow the rule that their size must consist of a power of two in both dimensions for computational reasons. This means that they are either square with for example 512 pixels as both height and width, or they consist of two different powers of two such as 128x1028 or 256x2048. The UV-map of a character will most likely have to be divided into several groups and should have seams in them so that the 2D shapes will be more evenly spread on the canvas. The seams and edges of islands can sometimes cause problems on how the textures look on the 3D mesh and their placement on the model can be crucial.

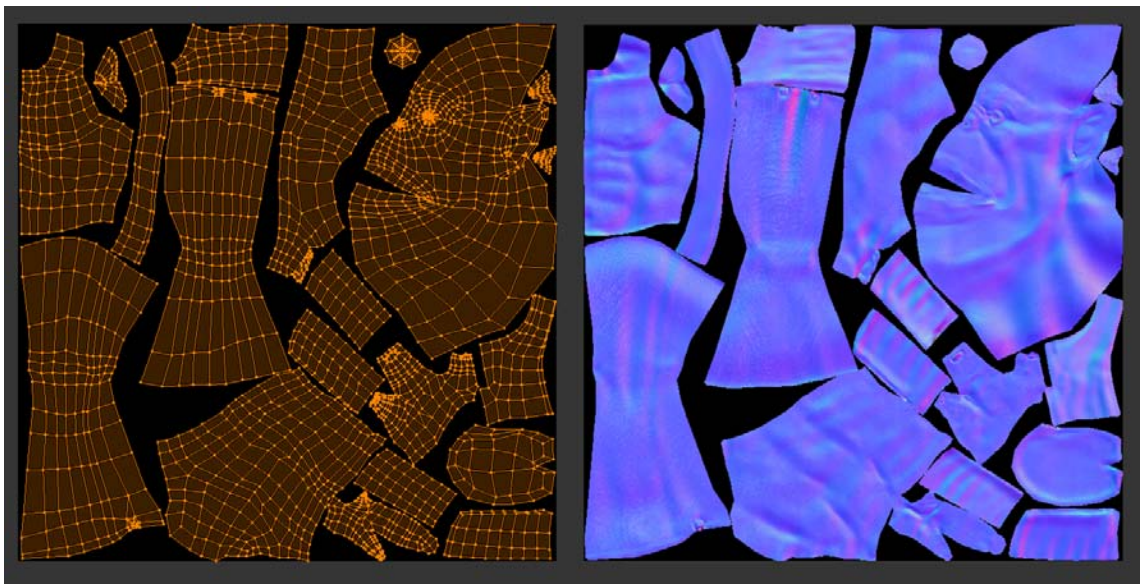
Keeping in mind the possible texture sizes and shapes of game characters and how the UV-seams can affect the outcome, the modeler can already think how the UV-mapping can be done with a certain character and make it easier by placing the polygons in a way that makes the process easier. Placing seams on polygon edges that are not that visible in the game or that might have a natural seam on them is usually a good way of hiding the possible problems with them. Areas such as the border between skin and clothing that already have contrast will probably also hide a seam effectively.

In 3D game development, rendering graphics is usually the bottleneck with the game's performance, so with textures and UV-mapping, optimizing everything is essential. Each separate textured material within the rendered view in the game world will add another draw call that has its own performance cost. The best way of UV-mapping the character is to put all of the surfaces on a single image map, unless there is a specific reason to separate them. However a single texture should not have any wasted space either but it should be only as big as is needed since the game engine will load all of the texture even if only half of it is mapped onto the character mesh. In order to optimize the use of textures, the character can be changed into a more simple to map shape already in the design and modeling phases. Adding detailed areas that have a natural seam

around them can make texturing and UV-mapping simpler and at the same time make the character more interesting and detailed.

As with almost any part of the character model's creation process, texturing is not always that straightforward. In certain situations, the relationship between textured models and performance costs has to be done in a more character specific ways. In some cases, the materials within a character can be so different from each other that the whole material has to be done separately. For example get the areas to react to light in a correct way. In those cases it is important that the model has a distinct border between the areas where the materials are applied since a single polygon cannot have more than one material applied to it. In other cases, especially with mobile games, the materials should not be made different from each other but made similar enough to allow the use of texture atlases. Texture atlases are combinations of several textures into a single material to get fewer draw calls and to really enhance the performance, but at the same time they limit the differences between the textures included in the atlas.

Texture baking is the main reason to create high polygon character models for game development. The high polygon models are used to bake more detailed parts of textures over the lower polygon models surface. This allows easier and better use of texture maps to fake finer details on the character's shape and lighting. For lighting, there are such as normal maps and ambient occlusion maps which can also be used to create more detail to other maps even if they are not used in the character's material as such. Knowing how the low polygon character's textures can be made more detailed through the use of the high polygon character, the low polygon character can be modeled so that those details are not included in the actual mesh topology but only in the textures (See image:25). Notice how the face area has a larger percentage of information, as there is more need for finer details. This can make a big difference in rendering costs and sometimes allow the use of more polygons elsewhere.



(Image 25) On the left: a UV-map for our orc model. On the right: the character's normal map for additional details without needing to model them.

Along with UV-maps and polygonal topology, game models usually use smoothing groups to differentiate between flat and smooth surfaces. Smooth faces create a smooth transition between them so that the edges between do not create as visible a corner as their flat counterparts. Game engines treat the edges that are located between smooth and flat surfaces as a physical break on the mesh surface and create separate chunks from them so the vertices between the two have to be duplicated to work as an edge for both types of smoothing groups.

The same thing happens whenever there is a transition from one material to another within a single continuous mesh or when the UV-map has a seam along the edge. In both cases, the amount of polygonal faces remains the same but the vertex count becomes larger and thus adds the need for hardware performance. One way to lessen the effect of the smoothing groups or material changes on the performance cost is to use it together with the UV-seams so that they follow the same edges and so only duplicate the vertices once instead of doing it separately for both the smoothing groups and the seams.

Texturing and UV-mapping can have an effect already in the modeling phase and are sometimes vital for lowering performance costs and avoiding problems with rendering. An optimal modeling method should take these steps into account from the start, so that there is less likely a need to go back and change things.

4.2 Level of detail -models

In his article, Antony Ward explains the purpose of the level of detail modeling: “By switching to lower-resolution models as the character or object moves away from the camera, you can lighten the overall polygon count on the screen without sacrificing visual quality.” (‘Generate more effective level of detail models’, Antony Ward, 3D World 2012-07)

Level of detail -models are used in games to make sure that the polygon count and the graphics details seen by the in-game camera do not take down the game performance unnecessarily but still retain the highest possible detail. The level of detail -modeling is not only character modeling specific, but still in many cases their use becomes most important in game characters as they usually slow down the game performance more than the static environment and their positioning is less predictable than that of their surroundings.

Different levels of detail are usually used in games which have the possibility to move the player’s view further and closer from the details in the game. The idea is to lower the details and the amount of polygons on the game character when the player doesn’t have a clear view on them and switch to higher detail as the player sees them closer up. This might allow the game for example to show a whole army of lower detail characters as the player’s view is really far away and still create the illusion that the player is still looking at the same high detail characters as he was when he was zooming at only a few of them.

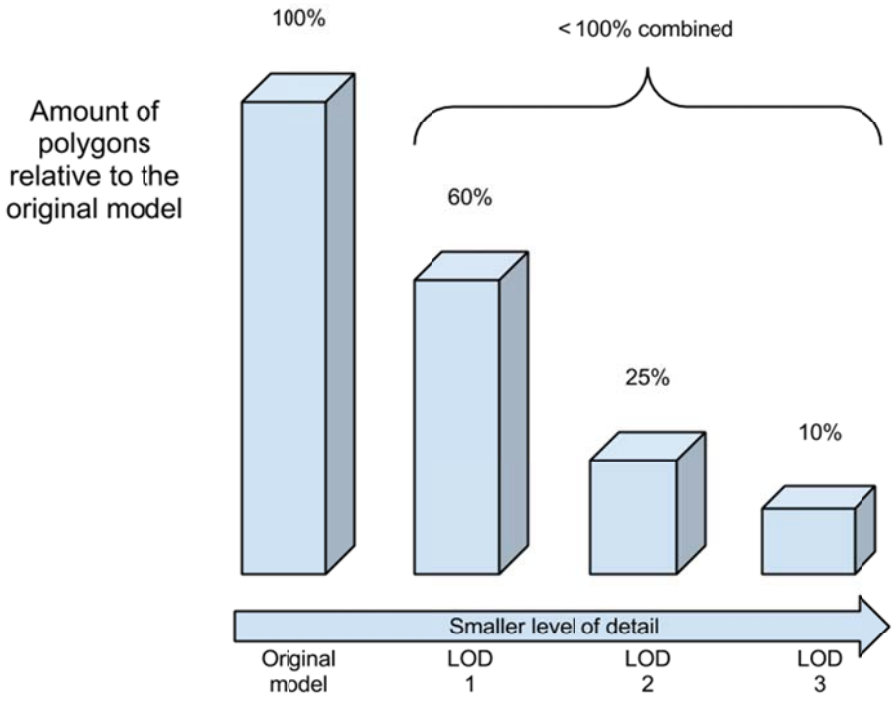
As with any game related modeling, the level of detail -model’s optimal use is very case sensitive and it should be considered as such. However there are certain aspects that should be thought of regardless of the game and the character’s use. Depending on the use of the character, there might be some aspects of the mesh that are more important than others, so viewing the models as they would be seen in the game can be a really useful place to start from.

The biggest visual problems with level of detail -modeling are usually encountered at the points where the model is switched to another level of detail. This can sometimes be

visible to the player and cause individual objects - or in worse cases most of the player's view - to stick out in an undesired way. This can be caused by a range of issues with the character models depending on the situation. Sometimes the textures of the character might seem to change dramatically, other times the shades and lighting make visible corners where they shouldn't be any and in many cases the character's animations seem to suddenly move in a wrong way. These issues differ from game to game and also between characters, so when tackling them, one should be thinking of the character's use in the game and which problems might be more acceptable than the others.

Defining the amount of detail in relation to the distance from the camera is a good way of working with level of detail models to make sure that the most problematic visual changes are hidden as well as possible from the player. Testing the outcome under various settings and environments while modifying the mesh may help finding the most visible problems and help fixing them.

There is an often used rule on how high or low the polygon count of a model's different levels of detail should have: All the polygons from lower level models combined should be no more than the original highest level model has and as the levels lower, the difference to the previous one should grow larger. This means that the difference between levels gets higher as the details go lower so that the changes become less visible when the character itself is more visible. So for example the switch from the highest detailed model to the next shouldn't be very radical because it is the easiest one to spot (See image: 26).



(Image 26) Approximate percentage of polygons in different level of detail models.

When optimizing a model for a certain level of detail, the modeler should be thinking about how far or close the character model should be seen within the game and what kind of details are the most and least important from that distance. For example if the character has a very detailed face with complex animations and rig on it but all the player can see is the movement of the jaw and eyes, then all the other details could and should be removed. The vertices with more than one bone moving them usually impact the game performance more as well, so removing vertices - as well as bones - from the parts that have more complex rigging like the face and hands is often more useful.

In many cases the lowest level of detail models are only seen as a silhouettes somewhere in the distance so with those models, the overall form of the character becomes really important even when the details are gone. It might be a good idea to for example put the lower level model on top the original one and try to make it take up most of the same space with all the other levels so that the silhouette seems similar from all directions.

Not all of the level of detail work is done in modeling, so thinking of ways that are done better with changes in texturing or rigging is also a part of the process. Sometimes re-

baking or switching to a different resolution of textures can help or replace some of the work that can be problematic with changing the actual mesh.

There are tools in 3d modeling programs that can be used to automatically create the level of detail models and some of them can in many cases be very cost effective. However like with most modeling, doing all of it by hand gives you a much greater control over what you are doing. The decision is not only limited to those two but sometimes it might be a very good idea to combine the two and for example make the model first automatically and then if there are obvious mistakes or problems with it, continue working on it manually and optimize it for the character's use in the game. Being aware of what the goal is for the character in question and knowing how it should look like is important even if the work is done automatically.

With level of detail models, the best way to optimize the modeling process is to first define which models are needed, based on how and from where the characters are to be seen in the game. While modeling the characters, the most important parts of a character should be preserved in each case and considered separately for each of the detail levels.

4.3 Avoiding overlapping work

As many 3D characters can be very similar especially within the same game, it is not always necessary to start each of them from a scratch. Even if the character seems to be really different from a previous one, there can be some parts of it that might be easier to create by using an existing mesh or a part of a mesh as a basis. When there is a character model that has a good topology designed and constructed, it might be better to edit that to suit the newer models instead of doing all of the work again from the beginning.

The same kind of methods can also be used with animations, rigging and texturing when these parts of the character are similar enough. When creating, for example, several characters with identical needs for rigging, the same rig can be modified to suit the slightly different character and as the bone structure stays the same in this way, the animations can usually be transferred between the two rigs and then also suited for the individual character. In cases like this, where several characters will be rigged and animated in a very similar way, their mesh should usually have a quite similar topology as

well, and duplicating and modifying these parts of the model can often save a lot of effort as they have already been tested to work.

Creating a single character first and using it in these ways to model other similar characters can optimize the process tremendously, especially when the game has a lot of characters whose rig can be developed from the same base rig. We created the orcs for our own project by first creating the topology and the rig for the medium sized orc, which we then used as a basis for the other two orcs.

5 CONCLUSIONS

Modeling a 3D character for game development can be a complicated process, as there are a vast number of ways to work. Balancing between the hardware performance requirements of the final character and the optimal visual outlook of the model along with animations and textures can result in a lot of tweaking and redoing if the process and the characters usage are not fully understood. We found out that one of the most important first steps when modeling a game character is to consider how the character is to be used and which traits are the most important to implement. Planning and deciding how those traits can all work together is needed for a consistent design when designing the model. Finding the connection between the character's design and its implementation becomes much easier if there is a general idea on how the model is to be created, part by part, keeping in mind all the limitations and the desired end result.

In order to have an optimal modeling process, one should have a clear character design to follow. Designing a character for a game includes taking into account all the other aspects of visual and game design. To keep the models design relevant to the game, the modeler should work together with the rest of the production team to make sure that they all have a common goal and in the end everything will work in the game. The modeling process should be optimized to support the gaming experience as a whole, so that the end result suits the following:

- Game genre
- Game environment
- Game mechanics
- Game narrative
- Characters personality and history

A character design concept is an important thing to have before starting the modeling as it often is crucial to all the parts throughout the whole development process.

Choosing the software for 3D character modeling is connected mostly to the artist’s preferences, but if the platform and game engine are not taken in account, there’s a good chance that additional software for exporting and importing the characters in to the game engine are needed. If the software and game engine are not compatible, transfer-

ring the character to the game world can be time consuming and problematic. Researching the tools beforehand can help in avoiding possible compatibility issues.

There are several methods and techniques available for modeling that all have their strengths and weaknesses. Not all of them are suitable for a game character modeling process, and it is important to know which ones simply will not create a character that works in a game environment. Changing the working methods according to the needs of the specific phase in the modeling process is an efficient way of optimizing the work flow. Box modeling for example is a good way of creating the basic form of the character without going to the details, but as the details are needed, moving to other methods that deal with individual faces, edges or vertices can produce better results.

We found that understanding the character model's usage plays an important role on its mesh topology and the modeling process. The positions of the character's joints and other parts that bend should be also considered from rigging and animation point of views as the movement of the mesh during animations is defined largely by the created mesh topology. In addition to rigging and animation, texturing and UV-mapping can also be made much easier and simpler by modeling the character while keeping in mind the later stages of the process. It is a good practice to use the vertices and polygons of the model to define the borders of different materials or smoothing groups and to plan good topology for the UV-mapping seams, so that they can be placed to areas where they cause fewer problems.

The optimization of the modeling process is not always solely about understanding the techniques and methods that are the most suitable for the task at hand, but sometimes it can be smart to use previously created assets and models for an easier and faster workflow. It is good practice to make use of similar models and other versions or level of detail models to create parts of new models. Sometimes it might even be a good idea to start a new character model by editing a completed, already existing mesh since creating everything from scratch can be time consuming and redundant.

As a whole, optimizing game character modeling is a rather complicated process as it seems that a modeler should be aware of all the preceding and following parts of the character creation while working on a model. Experience and knowledge in the whole game character creation process, as well as other parts of game development, make it

easier to consider as many aspects as possible while modeling. In a single project, everything is closely connected and an understanding of the bigger picture and other aspects of the game development can simplify the 3D modeling workflow by being able to notice possible problem areas beforehand and thus creating better functioning models with less need for going back and fixing mistakes.

In the future, character modeling for game development can become much more straightforward as computers and software keep on developing. With every new version of a 3D modeling software, there seems to be more and more automated functions that used to be made manually by the modeler or at least the existing tools become easier and faster to use. As the technology advances, the technological limits for 3D modeling start to become less and less dominant affecting the use of high polygon modeling techniques over low polygon techniques. The conclusions that we have made here, contemporary and we believe they will be so for some time. Things that we think will remain important even as modeling becomes easier to handle with newer tools are proper design process and the understanding of how 3D models are structured. Knowing how each part of the process works now will make it easier to spot problems and mistakes also in the future.

Our game character modeling project, Orcs Hogging Hogs resulted in four modeled characters with animations and basic functionality in Unity 3D game engine (See image: 27). We decided not to finish a complete game but instead focused on the parts most relevant for studying game character modeling. The three humanoid orc characters are controllable in the game and are able to interact with each other and the environment for example by tackling each other or turning their head and upper body to face in the direction of the hog character no matter where they move. We ended up having about 7500 polygons for each of the orc characters and 4400 for the hog, which should be a reasonable amount for a game targeted for console and PC platforms, as there are only six orc characters and one hog on the screen simultaneously. Our main focus on the topology was around the joints on the arms and legs so that the deformations when rigging the character would look smooth and natural. The face area of the mesh didn’t need to include facial expression animations, as it was not crucial for the gameplay, so we only used enough polygons needed for the geometrical details and jaw movement. The character modeling project was successful as a practical test for our thesis subject and

we managed to avoid any major problems in creating the characters all the way from drafts to controllable, animated characters within the game engine.

(Image 27) 3D Character models for our game project ‘Orcs hogging hogs”.

(Image 28) Screen capture from the game engine showing the textured medium sized orc character.

6 REFERENCES

Antony Ward, Master texturing for game characters, 3D World January 2012

Antony Ward, Generate more effective level of detail models, 3D World July 2012

Antony Ward, Animation tips for game artists, 3D World June 2012

Antony Ward, Character UV mapping tips for game artists, 3D World March 2012

Glen Southern, Tips and tricks for organic modeling, 3D World Magazine, August 2012

Thomas Bozovic, Alexandre Cazals, Julien Legay and Chao Ma, An insight into character design, 3D World March 2012

3-D Human Modeling and Animation, 2009, Peter Ratner, Wiley

Character-Driven Game Design, 2010, Petri Lankoski, Aalto University

Game Art: Creation, Direction, and Careers, 2005, Riccard Linde, Charles River Media

Rules of Play - Game Design Fundamentals, 2004, Katie Salen and Eric Zimmerman, The MIT press

The Art of Game Design - A Book of Lenses, 2008, Jesse Schell, Morgan Kaufmann publishers

http://www.3dworldmag.com/2011/09/21/create-the-best-character-models-for-games/, Antony Ward, 24.2.2012

http://www.gamasutra.com/view/feature/132380/choosing_the_game_engine_that_can.php?page=3, Paul Hyman, 2.2.2013

http://www.gamasutra.com/view/feature/2129/game_design_cognition_the_.php, Gilliard Lopez and Rafael Kuhnen, 03.02.2013

http://www.gamasutra.com/view/feature/178656/sponsored_feature_drawing_basics_.php?page=2, Chris Solarski, 16.01.2013

http://ontologicalgeek.com/the-philosopher-geek-mechanics-as-art-part-1-mechanics-as-art-support/, Bill Coberly, 17.01.2013

http://wiki.polycount.com/CategoryTopology, Eric Chadwick, 16.1.2013

http://www.ir-ltd.net/do-you-wanna-live-for-ever, Infinite-Realities, 26.02.2013

http://www.wired.com/gamelife/2009/04/eyes-on-jack-bl/, Chris Kohler, 26.02.2013

http://www.thesimshub.com/2012/01/whats-needed-in-the-sims-4/, thesimshub.com, 26.02.2013