Diploma Thesis

Game Art Asset Creation

handed in at the

FH JOANNEUM

Fachhochschulstudiengang Informations-Design

submitted in October 2006 by

Markus Hofer

Sonnenhang 5

A-8763 St. Oswald

Supervisor

Dietmar Mosbacher

Eidesstattliche Erklärung

Ich erkläre hiermit eidesstattlich, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Oslo, am 8.10.2006

Acknowledgements

Thank you for your help and support:
My parents Franz and Hildegard Hofer.
My grandparents Peter and Hildegard Steiner and Josef and Josefa Hofer.
Michael, Manuela, Katharina, Klemens, Heather and Lukas.
Helga, Herwig, David, Tanja, Martina, Daniel, Chris and many other of my fellow students.
Sonia, Dan, Mike, Vladas, Torstein, Grant, Didrik, Gaute, Pål, Kristian, Ruslan, Heidi, Razvan, Tibi, Robert and many more of my colleagues at Funcom.
My supervisors Dietmar Mosbacher and Terje Lundberg.

Kurzfassung

Diese Diplomarbeit beschäftigt sich mit der Erstellung von Grafik-Assets für Spiele, von den *Konzept-Zeichnungen*, über die *Modellierung* der 3D-Objekte bis zur *Texturierung*.

Der erste Abschnitt „Concepting“ erforscht die Aufgabe und Funktion von Konzept-Zeichnungen, beschreibt verschiedene Arten von „Concept Art“ und setzt sich mit den wichtigsten Stilen und Techniken auseinander.

Der zweite Abschnitt „Modeling“ behandelt die Erstellung von 3D Modellen für moderne Spiele-Engines. Er beschreibt die wichtigsten Werkzeuge und alle nötigen Hintergründe und Techniken zur Erstellung von low-poly Modellen, die für die Verwendung in Spiele-Engines dieser und der nächsten Generation zugeschnitten sind. Weiters behandelt er die Erstellung von high-poly Modellen und beschreibt mehrere Methoden diese zur Erzeugung von „*Normal Maps*“ zu verwenden.

Der dritte Abschnitt „Texturing“ beschreibt wie man nackten 3D Modellen ihr fertiges Aussehen verpasst. All die wichtigsten Bestandteile einer Textur werden erklärt und ein weites Spektrum von Techniken wird anhand von Beispielen erläutert.

Wo passend, werden Beispiele vom Werkstück dieser Diplomarbeit verwendet, dem „Palace of Kamula“, den ich für „Age of Conan“ (ein massively multiplayer online Spiel, das von Funcom entwickelt wird) erstellt habe.

Abstract

This diploma thesis focuses on the process of creating art assets for games, from *Concept Art* to *Modeling* to *Texturing*.

The first chapter "Concepting" evaluates what concept art is all about, describes different types of concept art and discusses some of the most important styles and techniques.

The second chapter "Modeling" is about the creation of 3D models for game engines on the edge of modern technology. It explains the basic tools and all the necessary backgrounds and techniques for creating low-poly meshes that are perfectly suited for use in current and next generation realtime 3d engines, how to create high-poly meshes and how to use them to create *normal maps*.

The 3rd chapter "Texturing" describes the process of giving a naked 3D model its finished look. It explains all the most important texture-components that are being used by current and next generation game engines and describes a wide array of techniques on several examples.

Where fit the examples are taken from the work-part of this diploma thesis, the "Palace of Kamula", which I created for "Age of Conan", a massively multiplayer online game developed by Funcom.

Contents

Introduction

This diploma thesis consists of a work part and a theoretical part.

The *work part* is the “Palace of Kamula” and will be found in the massively multiplayer online game “Age of Conan”, developed by Funcom.

This document is the *theoretical part* and focuses on the first part of the art production pipeline for games: Concepting, Modeling and Texturing.

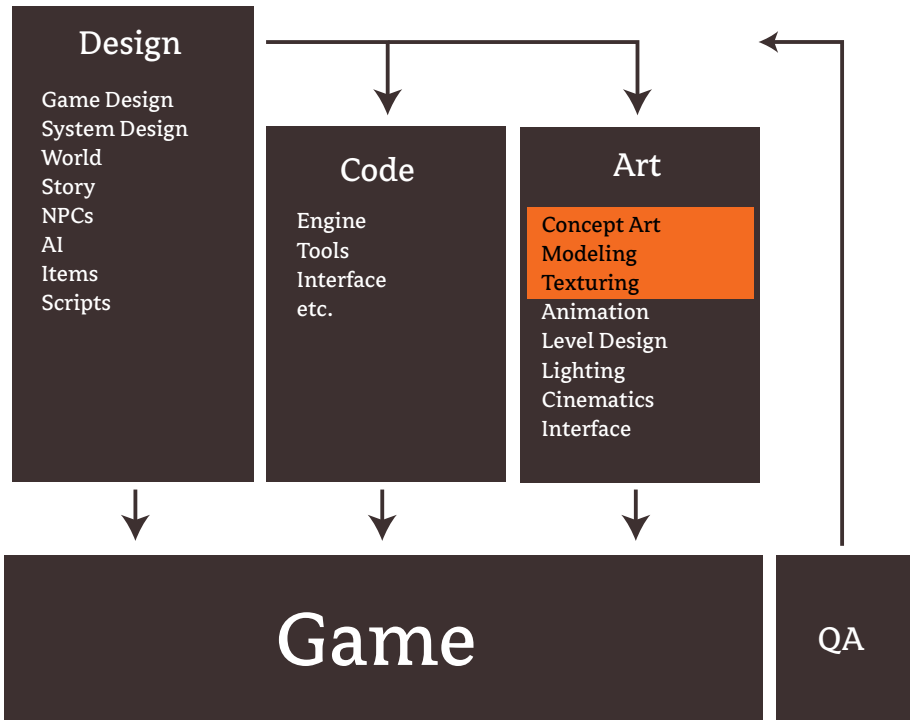


Fig. 1: The focus of this diploma thesis

Chapter 1:
Concepting

Introduction

In the process of creating assets for games, creating the concept drawings is the most creative part in the traditional sense of the word. While modeling and texturing don't require any less creativity, they require more of a *technical creativity*, creativity in dealing with the tools available.
But here it's coming up with unique-looking monsters, fascinating architecture, even entire art-styles for imaginary cultures. Often times you will almost start from zero. The world-design-document asks for some sort of ancient building, describes it in a few words, and from there on you're on your own. And it's not only about forms and proportions, it's also about feelings and emotions. Great concepts are not only there to visualize, they are also there to inspire!

This chapter is split into two parts. In the first part *"Inception"* I'll discuss techniques for creative thinking, talk about inspiration and describe how the art and architecture of past millennia can help you to create new wonders. In the second part *"Execution"* I'll deal with the technical aspects of creating good concept art and follow through several examples to illustrate different styles and techniques.

Inception

Where do ideas come from?

There are countless creativity techniques out there, from simple methods like "Brainstorming" and "Mind Mapping" to very obscure roleplaying-experiments, and as much as I believe that they can help, none can *guarantee* a good result. Ultimately it's always a matter of developing your own individual set of tools and techniques. Not everything that works for someone else will work for you. And you should always use what works best for you.

One central point of inspiration for all artists is nature. Because every human being is so used to seeing it, there is almost no way of creating a believable concept without mimicking nature in one aspect or another. And the sources of inspiration are almost endless. Even the most boring place is full of interesting ideas, it's just a matter of looking at it the right way. Inspiration, I believe, is largely influenced by personal attitude. You should not be afraid of looking at the world in your own way.
Looking at nature, photo-collections, movies, fine art, will only inspire if you look at it with your own eyes. A famous Picasso-quote reads "Good artists copy, great artists steal". The way I take it, being a great artist is not so much about reproducing something accurately up to the smallest detail, it's about understanding it. If you not only borrow, but steal, you *make it your own*.

Many artists use drawing as a means of developing and refining ideas. The Gnomon Analog DVD series is a great source for learning about different approaches:

In his DVD about character design Neville Page starts by drawing thumbnails on paper. He uses a black marker to quickly produce many silhouettes of animals, then takes the one that works best, blows it up and starts drawing on top of that. (cf. Page 2004a)

Ryan Church starts his architectural concept paintings with a basic idea and a dark and neutral canvas in PAINTER. He first draws a horizon line and some perspective lines and then begins to quickly block in the basic shapes of a composition. (cf. Church 2004a and Church 2004b)

In one DVD Carlos Huante demonstrates how he starts with no prior thought on the subject, lays out simple round shapes and just puts them together and lets the drawing take any direction it wants. “I don’t even have a concept in my head really at this point. But I’m seeing stuff, so I’m starting to emphasize things like arms, maybe a head, possibly... I really don’t know yet.” (Huante 2004a, 3 min 30 sec)

Scott Robertson has an interesting technique for coming up with interesting environments. He starts by drawing with markers on paper, just trying to come up with interesting, abstract shapes. Then he scans those, combines them in PHOTOSHOP and starts looking for interesting compositions while experimenting with all the different layer blending modes. He saves out the most interesting results and starts painting on top of those, just emphasizing and detailing what he sees in the abstract shapes. (cf. Robertson 2005)

Fig. 2: N. Page

Fig. 3: R. Church

Fig. 4: C. Huante

Fig. 5: S. Robertson

Not all of these techniques will always produce usable results. Especially when working on a big project, it might require many iterations and many discussions with the art director and the designers to come up with something that fits in. The final drawing will most likely never resemble the first idea. But that's okay, and it's all part of the process. Only with hard work come good results, and the job of a concept artist is it to keep going until it is good.

Often times the big challenge is to come up with something that looks like it took months and months to plan, but that only takes you a day to come up with. There's no way of achieving that without a good knowledge of art and architecture and a good collection of resources. There are many great art-history books, architectural and anatomical atlases and collections of patterns and textures. But it's best to have a basic visual dictionary in one's head and to use that to get all the basics right from the start.

This early in the process it's good not to worry all too much about the technical aspects, as not to limit the imagination. The modeling- and texture-artists know all the tricks and will find a way to execute even the most complicated ideas. Yet it's no mistake to keep at least some fundamental restrictions in mind. If your game-engine doesn't support reflections, don't plan a hallway of mirrors...

Execution

The wonders of modern technology and digital painting make it possible for this step to go hand in hand with the first one. After all of the necessary basics are thought out I usually get straight into making a very rough drawing, going into more detailed planning as individual challenges arise.

There are two extremes of concept art:

Mood-pieces that are full of feeling, show a moody lighting situation, that can be full of action and almost void of clear detail.

And *technical drawings* that contain little excitement, but are designed in such way that modelers and texture artists know exactly what to do.

Fig. 6: Mood piece - B. Gentile ("Prince of Persia - TTT")

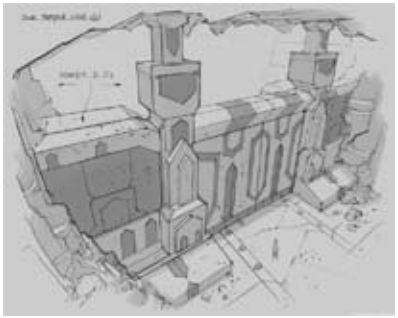


Fig. 7: Technical drawing - F. Zhu ("Battle for Middleearth II")

Of course there is an unlimited number of possibilities in between these two extremes. And depending on the task, it might be required to deliver quite a few different drawings with different purposes – if you are to design the main character of a game, it will not be enough to present one finished drawing. You might be required to do several mood-pieces to convey attitude and personality of the character, as

well as detailed anatomical drawings, often in T-pose, that can be used as blueprints for modeling. Maybe even detailed drawings of all the important poses, sketches of the character in different outfits and in different environments.
Yet when designing a simple table, a quick sketch will usually be enough.

Some concept paintings are specifically intended for use as marketing material. Those have to have a quite finished look, a high level of “coolness”, and should fit the style of the game perfectly.

All “real” concept art however should aim for these goals instead:
• To visualize the idea properly, but in an economical way.
• To make it easy for modeling- and texture-artists to create the finished asset.
• To inspire the team.

Everything is allowed if it helps these goals. If it does the job while saving time: add snippets of photos, build little models out of cardboard, attach newspaper-articles or send links to pictures on the web.

If we take the Picasso-quote from above one step further, we could say that great concept art is not about pretty pictures, it’s much more about developing the most interesting and plausible looks. Sometimes this requires a day of research and only results in a dirty paper-sketch. If the visual idea is great and the paper-sketch communicates it well enough, that’s perfectly alright.

Of course every concept artist should have outstanding drawing and painting skills, but not every concept of a house has to be of oil-painting-like quality. First and foremost concept art has to convey what it is really about, everything else is adornment.

Tools

The main tools for concept artists all over the world are a graphic tablet and Photoshop or Painter. Even though it's often good to start with a sketch on paper, there is no way around the speed and convenience of digital painting anymore.

In my examples I use Photoshop CS 2 and ArtRage 2.

Basics

It's best to start small and rough. Once everything is roughed out and the overall composition of the scene looks good, size up and start adding detail to the important regions.
The use of special brushes and textures will make the look of the image a bit more sophisticated and less sterile.

The use of reference pictures and color references is a great help and should not be forgone out of pride. It's the best result that counts.
Flipping and rotating the canvas regularly makes it easier to see if something works or not.

Now let's take a closer look at some basic techniques for two of the most important aspects of good concept art: perspective and light.

Drawing in perspective

If the concept is for an object that will be modeled in 3d, it is important to make sure

that everything in the drawing is possible. Furthermore it is important to choose the right perspective. If you are to make a *technical drawing*, the perspective should be chosen so the viewer gets a good look at the object. For a *mood piece* it's most important that the perspective is interesting. But in both cases it's important that the perspective looks fairly realistic.

I will quickly outline a few basic techniques that help to get the perspective fairly right, without taking too much time to construct.

Lines that are parallel in 3d space share a point at infinity. In a perspective drawing those lines go through one common vanishing point. A a cuboid with 6 faces has thus 3 different vanishing points. Drawing 3 vanishing points is however only necessary when the camera is tilted up or down or the object depicted is not aligned to the camera. This is mostly done for dramatic effect and is generally referred to as *three point perspective*.

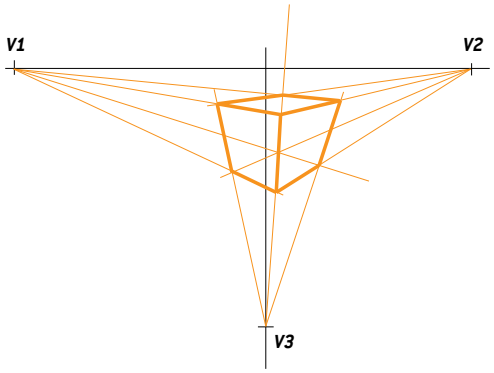


Fig. 8: 3 point perspective

If the camera is not tilted up or down the third vanishing point slides off into eternity and all height-lines can be drawn as parallels normal to the horizon. This is called *two point perspective*.

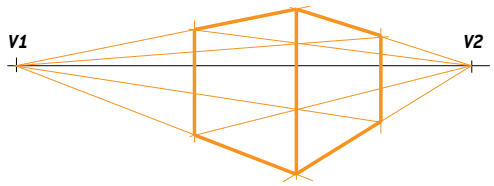


Fig. 9: 2 point perspective

When object and camera are aligned in such way that the front side is parallel to the camera plane, the 2nd vanishing point

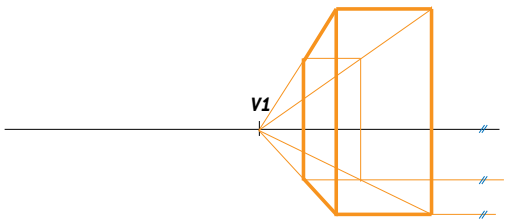


Fig. 10: 1 point perspective

slides off into eternity as well and we're only left with one vanishing point. Only depth lines are perspectively receding into a vanishing point now, so height and width are drawn parallel to the camera plane and normal to each other. - *One point perspective*.

For the purpose of doing concept drawings it's not so much important to construct every little detail; what is important is to get it to look right without spending a lot of time on it. This can be achieved by being clever about the way the construction is done. Don't construct more than necessary - a rough cage is usually enough guidance for drawing even complicated objects. The lines don't have to be perfect, a little bit of imperfection usually gives a natural touch to the image. Just make sure it looks plausible.

If aiming for a realistic look, the vanishing points should be set quite far apart. Studying photos, movie stills and paintings can help to give a good sense of how the vanishing points should be set to achieve specific results.

One way of working with perspective is to start with a large document, construct all the guides needed, and then crop the interesting area before starting to paint.
Scott Robertson has a clever alternative technique: Instead of using the **Line Tool** in **Fill Pixels** mode, he uses it in **Paths** mode, therefore drawing *paths* instead of *pixels on a layer*. This has several advantages: unlike pixels you can see paths outside the canvas, it's therefore possible to have the vanishing points outside the canvas which allows for setting the canvas to a much smaller area without having to very roughly guess where the vanishing points might be. And most importantly it's possible to move all the individual points easily. (cf. Robertson 2005: 1 h, 32 min)
Yet when constructing more complicated objects, a lot of lines might be needed and the paths might get in the way. In that case I like to turn different groups of paths into differently colored layers that I can put on top of my painting. Switching the paths off and setting the opacity of these new layers to a low value allows me to paint with a bit of guidance, but without the visually intrusive appearance of paths.

Translating real units and ratios into perspective

I'll now create a cuboid with a width and depth of 2 units and a height of 1 unit. Since lines appear foreshortened in perspective I can't just take measurements directly on the lines. I have to introduce an imaginary plane that is parallel to the camera plane and translate measurements from this plane into perspective. First I choose the **Eye-Point** E, a point where the object dissects this imaginary plane.

From there I get the first **Measuring Point** M1 by drawing a circle around V1 that has a radius of the distance between V1 and E.

I do the same with V2 and the radius of the distance between V2 and E and get M2. Then I draw a line through E that is parallel to the **horizon**. This line lies on our imaginary plane and is therefore in real size. I mark 2 units in each direction and connect each two resulting points with the **Measuring Points**. Now I only have to look for the right intersections with the perspective lines going from E, and I get my measurements in perspective.

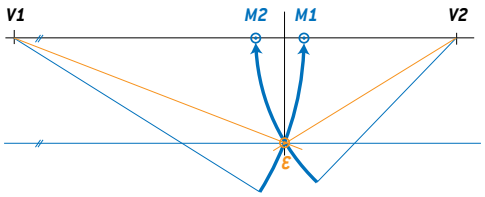


Fig. 11: Constructing the measuring points.

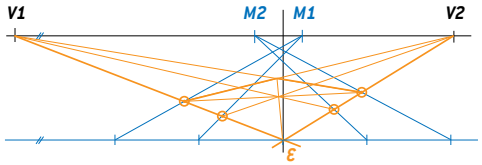


Fig. 12: Looking for the right intersections.

Since I'm working in a 2 point perspective, the height is easy. As mentioned above all height-lines are parallel to the camera-plane. A line that goes through E must therefore be on our imaginary plane and thus be in real size. The easiest way to get the height is to just draw a circle through the 1 unit mark and intersect it with the height. From there on it's a simple matter of connecting the right pieces and done is our wonderful cuboid. (cf. Treibergs)

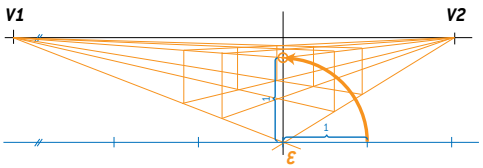


Fig. 13: Constructing the height.

Halving and doubling spaces.

The center of a quadrangle can be found out by intersecting the 2 diagonals. Drawing a line through this point and the appropriate vanishing point halves the face.

To double a space find the middle of the outer edge and draw a line from the corner through that point all the way until it intersects the base line.

If you only have a line, simply add any height to turn it into a quadrangle and proceed as described.

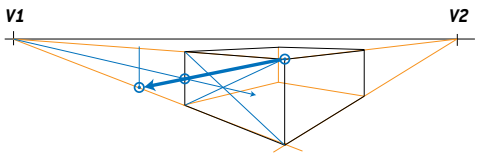


Fig. 14: Halving and doubling spaces.

Sloping surfaces

For drawing sloping surfaces the vanishing point can be elevated above the horizon.

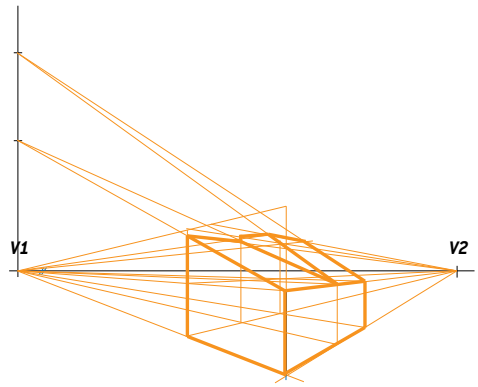


Fig. 15: Sloping surfaces

Shadows in perspective

First a light-source is needed. Straight underneath it on the ground is the shadow vanishing point. To find the outlines of the shadow of an object, draw lines from the light vanishing point through all the corners and intersect those with lines that go from the shadow vanishing point through projections of the corners on the ground plane. Connect the resulting points so the biggest possible space is covered. (cf. Larmann)

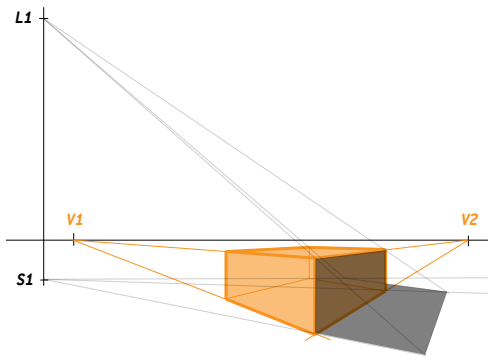


Fig. 16: Shadows in perspective

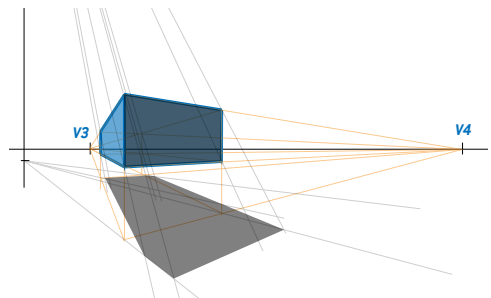


Fig. 17: Shadows in perspective

If you are planning a complicated object it might be faster to model it roughly and paint on top of a rendering. It's all about speed. Having a very rough 3D scene also makes it possible to spend a little time finding the most interesting perspective.

Light and Shadows

While a little bit of shading is enough to give plasticity to a rough sketch, a good and realistic lighting situation is much more complicated. After all it's not only important to understand how light affects a scene, it's also important to choose a good lighting setup to present the motive in the right light.

Let's start simple:

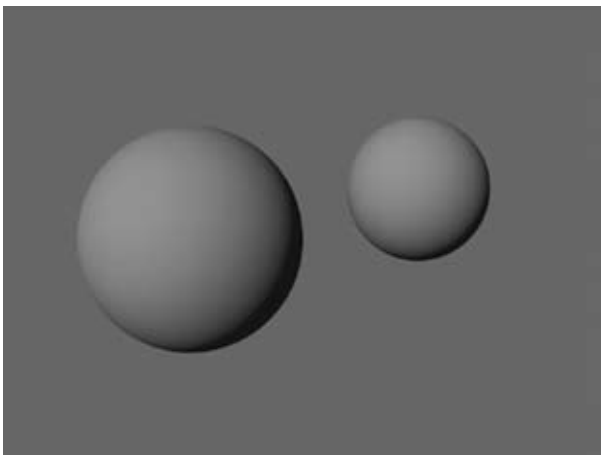


Fig. 18

A bit of diffuse lighting brings out the forms quite well, but without shadows it's hard to grasp the spatial layout. There's no way of telling how big the two spheres are and how far they are apart. Similar shape and color might lead us to believe that the objects could be of similar size and that the one on the right is simply positioned further back.

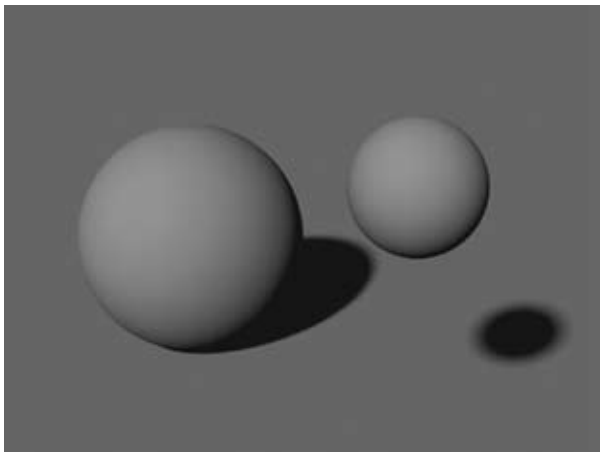


Fig. 19

Rendering the shadows shows us another possibility: The sphere on the right is smaller, hovering in plain air close to the camera. Notice how it's now possible to estimate their relative size, but we have no indication of what their absolute size might be as we have no familiar point of reference.

The entire scene looks very computer-generated because its lighting model is overly simplified.
Let's look at another rendering that is more realistic and analyze it's separate parts:

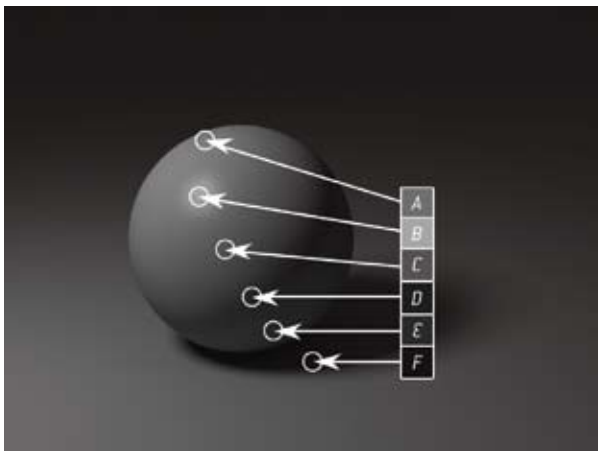


Fig. 20

A: Diffusely reflected light
B: Specular highlight – specularly reflected light. This depends on camera position as it is a mirror-like reflection of the lightsource.
C: Diffusely reflected light
D: Core shadow – begins where lightrays are tangent to the surface – This is the darkest part of the object visible to us, but it is not black as it picks up a little bit of reflected light from the surroundings.
E: This area is in the shadow of the Area light, but picks up quite a bit of reflected light from the ground and is therefore lighter than the previous sample.
F: The shadow on the ground.
(cf. Page 2004a)

The transition from light to shadow depends very much on the *type of light-source*. Let's have a look how shadows change with different light sources.

Fig. 21: Area light

Area lights create soft shadows. Lightrays are emitted from a large surface, which creates a soft transition from light to shadow, as the parts that face the lightsource more directly are hit by more light at once. The shadow can be split into 2 parts: The umbra, the dark part of the shadow that is completely invisible to the lightsource and the penumbra, the part that is visible to a part of the lightsource. (cf. Wikipedia: Umbra)
On an overclouded day the clouds scatter the sunlight and create the effect of a big area light.

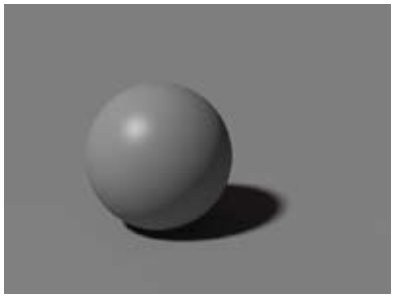


Fig. 22: Directional light

Sunlight (Directional light): Sunlight comes from so far away that its light-rays can basically be thought of as parallel. Direct sunlight produces quite hard shadows.

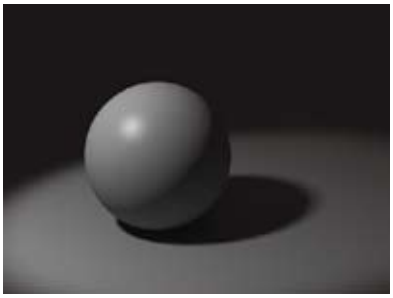


Fig. 23: Spot light

Spot lights are emitting light from a small surface and create hard shadows. Depending on the size and position of light and object, the shadows might look distorted.

Fig. 24: Ambient light

Ambient light is very diffusely reflected light from the surroundings.

Studio lighting setup

A typical studio setup consists of a key light (the main light-source), a fill light (to shed a little light on the rest of the motive), a rim light (placed behind the motive to bring out the contours) and eventually a background-light. (cf. Petrasch and Zinke 2003: 84f)

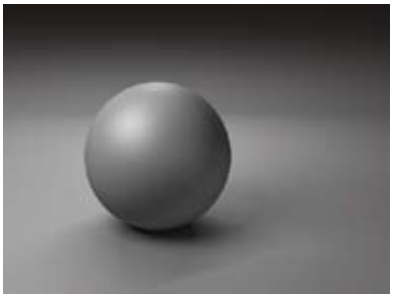


Fig. 25: Studio lighting setup

Surfaces and Light

Objects in the real world don't have perfectly flat surfaces. A bumpy surface creates a much less smooth look. Drawing on paper creates this effect more or less automatically and therefore often results in a more realistic look than perfectly clean images that were created digitally. A digital drawing can easily be made

Fig. 26: Studio setup with rough object

to look rougher by adding bits of textures or by painting with special brushes.

Real world lighting situations
Real world lighting situations can be a lot more complex than studio setups. Someone walking on a busy street at night is exposed to many different sources of light: the moon, streetlights, illuminated advertising signs, car-lights, reflected light from the surroundings and more.

Atmospheric Interference
When drawing huge outdoor scenes it is important to know how to deal with objects in the distance:
“Aerial or atmospheric interference with visual perception causes loss of contrast, detail and sharp focus.“ (“ATMOSPHERIC or AERIAL PERSPECTIVE”)
“The human brain interprets [..] depth based on values. So the further away an object is the lighter it should be in contrast.” (Robertson 2005: 3 min 20 sec)

Effects
Nature knows quite a few “special effects”. For that little extra touch of realism, it can be helpful to study nature and natural phenomenons: air is acting as a mirror over hot streets in the desert, caustics can give a very special look to scenes with water, frost and fog can make beautiful winter landscapes and the flow of water on a fountain can be unexpectedly beautiful, to name but a few examples. (see “Physics around us”) Just don’t base your concept on an effect like that if your game’s engine isn’t able to reproduce it.

Examples

Sword

I started with setting the rough proportions of the sword by drawing guides. From there I drew outlines and erased and corrected until I liked the result. On a different layer I painted it black to get a better impression what the silhouette was looking like, as the silhouette is what makes weapons look unique even when viewed from a distance. This is especially important in 3rd person games where you'll mostly see weapons quite small. If a game doesn't have swords with many different silhouettes, everything will end up looking similar, no matter how much work has been invested in the details of the models and textures.

Fig. 27: Sword - Work in progress

As next step I hid the silhouette, created a new layer underneath the outlines and started filling in color. After a basic color-scheme was established, I started adding highlights and shadows to add plasticity to the drawing.

In the end I added a bit of cheap fanciness: A very soft shadow and a glow around the lightest areas. Does this make the *concept* of

the sword better? No, but it makes the *picture* of the sword look nicer. It is important to remember, that a bit of polish might be a good idea for several reasons, but that nothing you add to help your picture will carry over into the game. Trying to hide flaws in the design by adding lighting effects or additional objects or backgrounds might make the picture nicer, but there is no way of hiding fundamental flaws from the people further down the production pipeline.

Fig. 28: Sword - Final

Simple Character Concept – Caricature of 3DS Max

The idea behind this character was to create a caricature of what 3d applications would look like if they were human. This example is my lovingly hateful take on 3ds max. I decided it would be quite old and packed full of tools, some modern, some ancient and some in weird places. I started with a very rough 2 min sketch on paper. I did this mostly as a means of coming up with ideas and didn't bother much about anatomical correctness or style at that point. When I though I had captured the most important aspects, I took a picture and brought it into ArtRage 2. I also imported some reference material from a great book that I always refer to when drawing human heads: "Drawing the Head and Hands" (Loomis, 1956)

Fig. 29: Work in progress

I chose a mid-level grey background in my **Paper Settings** and started roughing out the basic shapes with the **Pencil** tool. At this stage it's very important to take a step back every now and then and look away for a minute, just so you can come back and try to look at the picture with fresh eyes. Try to see if the shapes make sense. I helps to erase all lines and guides that are no longer needed, as they can create a misleading visual balance. (The 15 lines on one side of the body might cause the entire picture to look right, but once it's colored and you erase those lines it might suddenly look wrong...) It also helps to flip or rotate the image. (My art teacher in school would make me hold up my drawings against the window so I could look at them flipped.)
After I was sure the shapes were good, I moved on to

detailing the head, drawing the beard and adding the first couple of tools and gadgets.
Then I decided it was time to bring in some color and started painting a few basic shades on a new layer that I put in the background. I played with the colors until it looked okay and moved on to drawing good outlines on a new layer. I didn't want the picture to look too clean though, so I left the basic sketch in there too, but made it about 50% transparent. I adjusted the color-areas to the new outlines and started painting highlights and shadows. This part is tricky as it requires a lot of spatial sense and a basic understanding of light and materials.
Not only for beginners it's often very helpful to draw a 3d arrow onto the side of the canvas that shows the direction of the main lightsource and a sphere that shows the lighting situation and how the chosen material reacts to light.

In the end I added a bit of framing, a simple background and the tag-line "I've got this great new tool, if only I could remember where I put it".
This should be more than enough guidance for the creation of a great character model. The talented modeling- and texture-artists will fill in the rest in the same style.

Fig. 30: Workspace and final result

Lemurian Pillar

Fig. 31: Grant Regan's Concept

This is why everyone in the art pipeline should be able to draw basic concepts: Sometimes it's just so much faster to try out ideas by quickly painting them instead of working on the real thing.
This example illustrates how I explored different looks for the columns of the palace, when starting to work on their texture.

Grant Regan created the basic concept for the lemurian pillars, but as that contained no information about the coloring and was quite vague about the ornamentation, I decided to quickly draw a couple of variants on top of a rendering to see which options might work best.

Fig. 32

The Pillars of Kamula

Grant Regan created this concept for the Pillars of Kamula. Two versions of these statues were required:

- A ruined version, to be placed in the middle of a desert.
- A new version for the Palace of Kamula.

The concept shows a stage in between the two versions. The statues are quite ruined, but enough is left of them to imagine what the new version must have looked like. The inspiration for this concept came from the ruins of the ancient city of Persepolis combined with the look of one of the ancient monsters in the game.

Fig. 33

The Palace of Kamula

This concept by Grant Regan was the basis for the *work part* of this diploma thesis.

The palace was built in pre-Stygian times by the Lemurians and is therefore a quite unique site in the world of Hyboria.

Real world influences were Aztec architecture and ornaments, babylonian ruins and very early egyptian buildings.

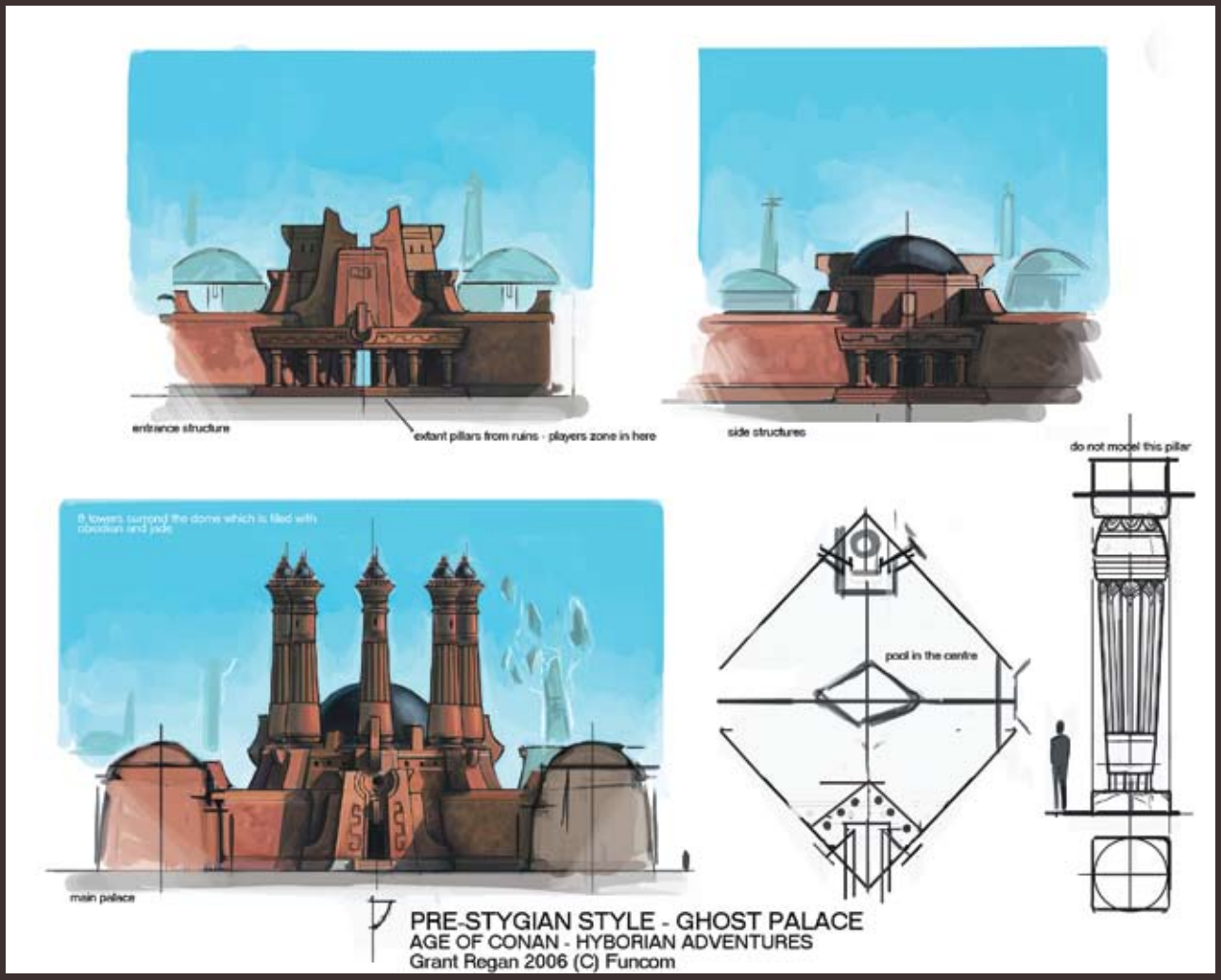

entrance structure
extant pillars from ruins - players zone in here
side structures
do not model this pillar
8 towers surrond the dome which is filled with obsidian and jade
pool in the centre
main palace
PRE-STYGIAN STYLE - GHOST PALACE
AGE OF CONAN - HYBORIAN ADVENTURES
Grant Regan 2006 (C) Funcom

Fig. 34

Location Concepts

With location concepts it's often good to not only create clean drawings for the modeling- and texture-artists, but to also add in some information about the surroundings. That way the world-designers can refer to them as well and the modeling- and texture-artists get a good idea of the conditions the buildings are standing in. A wooden hut standing in a swamp ages differently from a wooden house standing in the desert.

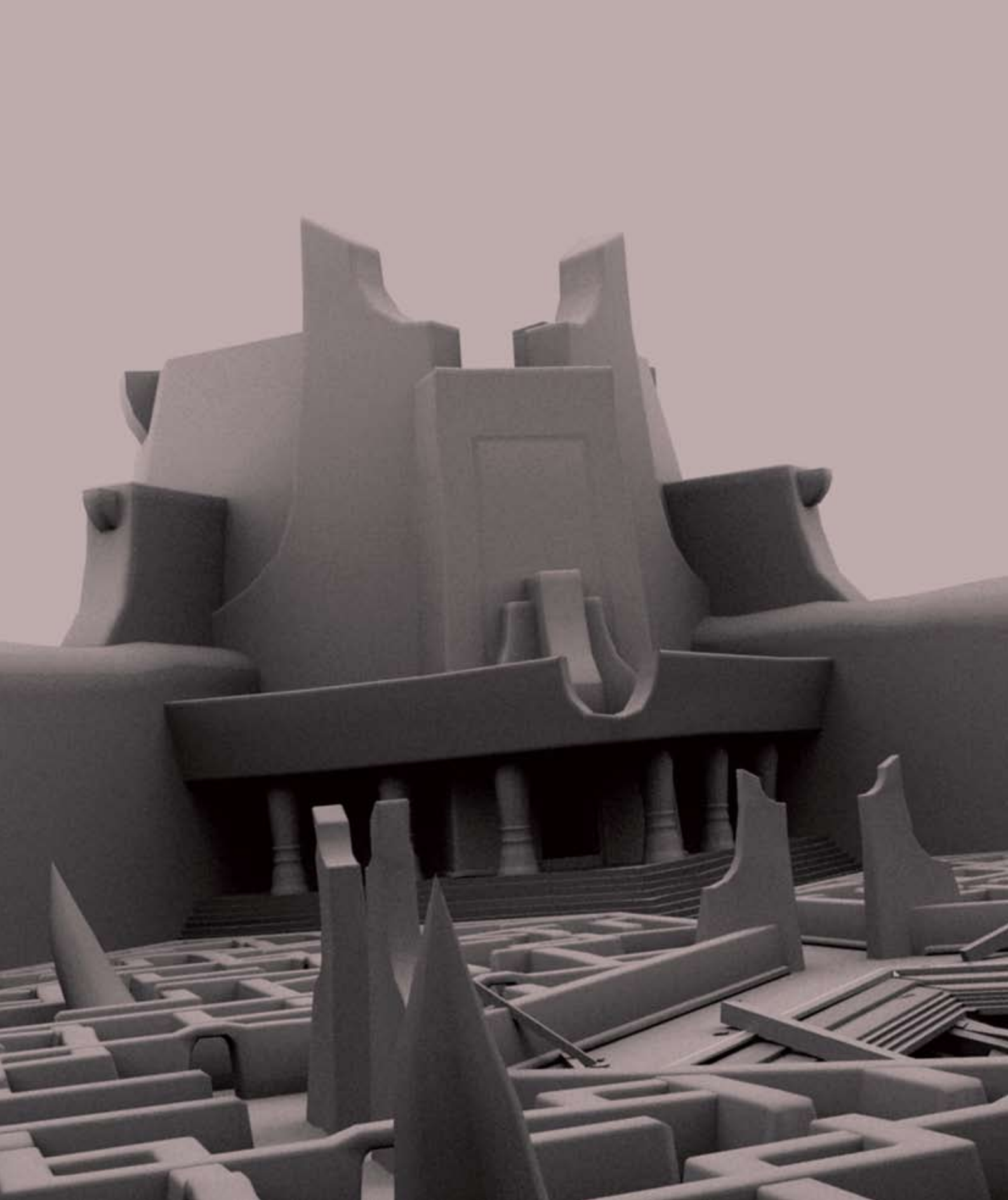
These are some of the concept drawings that Funcom's concept artists created for "Age of Conan":



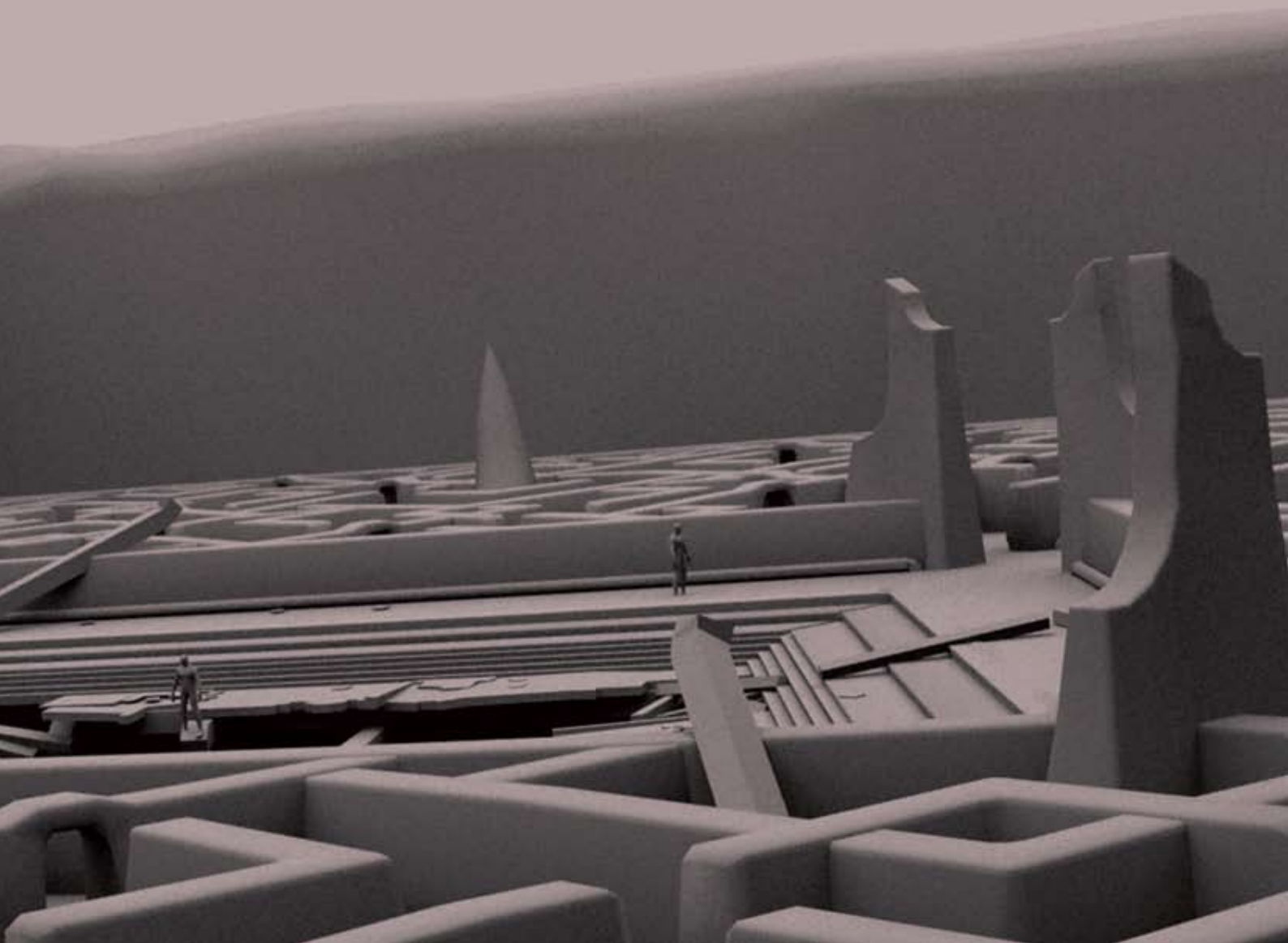
Fig. 35




Fig. 36



Chapter 2:
Modeling



Introduction

This is what we want to achieve:
Beautiful, efficient low-poly-models that look like great high-poly models, closely resemble the concept drawings and take as little time to build as possible.
Yes, modeling for real-time-engines is a demanding field. The expected quality-level is closing in on movie-special-effects, but the technological constraints are a lot tighter. It is not enough to have the artistic skills to sculpt a high-poly model, you also have to be able to transform your work back into a low poly model without losing the high level of quality.

To achieve that a 3D artist must have a good *spatial sense*, good knowledge about *geometric and artistic basics* and what I like to call *"technical creativity"* - the creative handling of the tools available to not only achieve the best result, but to do so in as little time as possible.

In this Chapter I will introduce some of the tools available, outline all the basics needed to model high- and low-detail assets and then go through quite a few examples to illustrate different workflows and describe different techniques and tricks.

Basics

Tools

3D modeling is still in its infancy. We are cavemen banging on rocks.
Even though modeling applications have improved a lot since their inception, they still haven't reached a level of intuitiveness and ease that would allow the user to work at a satisfying speed.
Mouse and keyboard just aren't very good tools for crafting 3D objects.
But while the industry-standard-dinosaur 3ds max is growing into an ever more horribly unorganized interface-monster, some fresh contenders have shown up on the scene and taken big steps ahead:

ZBrush introduced a whole new way of working. The ability to use a pressure sensitive pen to deform geometry made high poly modeling a big step more intuitive and a great deal more fun. More and more applications are introducing some form of displacement painting, but only Mudbox takes this approach as far as ZBrush. It will be very interesting to see how these two contenders develop over the next few years.

Modo (with its bold slogan "Model at the speed of thought") captured my heart with its very well thought out way of working, its solid tools and its streamlined and easily customizable interface.

And the road ahead looks bright. With a manifold of more or less revolutionary programs being released every year, all the big players are pushed to optimize their tools and interfaces to provide the best possible workflow.

While I don't think the time has come just yet to put the mouse aside and use our two hands with all 10 fingers to sculpt and form virtual objects using haptic modeling devices, I have high hopes that I'll live to see the day.

ZBRUSH 2 QUICKINTRO

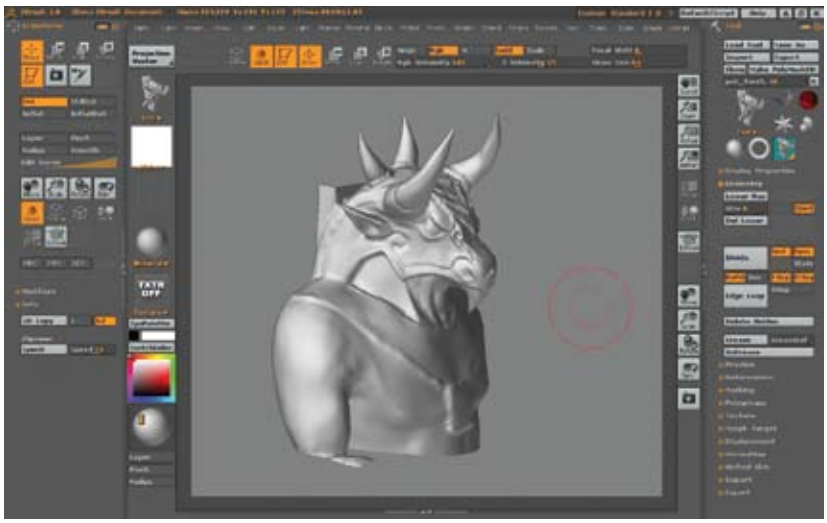


Fig. 37: ZBrush 2 interface

Navi- gation

Rotate: Click `Left Mouse Button` outside actual geometry and drag (hold `[Shift]` to snap to certain views).

Pan: Hold `[Alt]`, press `LMB` outside geometry and drag.

Zoom: Hold `[Alt]`, press `LMB` outside geometry, release `[Alt]`, drag up and down to zoom out and in.

Painting

Paint: Normal paint (`LMB`): **Zadd**, `LMB`+`[Alt]`: **Zsub**, `LMB`+`[Shift]`: **Smooth**

"`[`": draw size smaller, "`]`": **draw size** bigger.

Selection

Press `LMB`+`[Ctrl]` to paint **deselection** directly onto the object or press `[Ctrl]` and click `LMB` outside geometry and drag a **deselection rectangle** over geometry (Zbrush 2.5 will have **lasso selection**) – If `[Ctrl]`+`[Alt]` are pressed the methods described above will draw **normal selections**.

[Ctrl]+LMB on canvas will **invert the selection**.
[Ctrl]+LMB (draw little rectangle on canvas without enclosing any geometry): **select everything**

Hiding Geometry
Hide: press [Ctrl]+[Shift]+LMB and drag a rectangle around geometry: hides everything outside the rectangle.
Press [Ctrl]+[Shift]+LMB and drag a rectangle around geometry, but release [Shift] before you release LMB: hides everything inside the rectangle.
[Ctrl]+[Shift]+LMB and drag a little rectangle on the canvas without enclosing any geometry: **unhide** everything.

Use **set picot point** button to center the pivot point to the currently visible mesh.
Clear pivot point sets it back to its original location.

Other
RMB: quick menu.
[Tab]: switch the float menu on/off.

Customization
To save your configuration go to *Preferences › Config › Store Config*
To always start with the same canvas dimensions save a file with the right setup to Zbrush2\ZStartup\StartupDocument.ZBR

MODO 202 QUICKINTRO

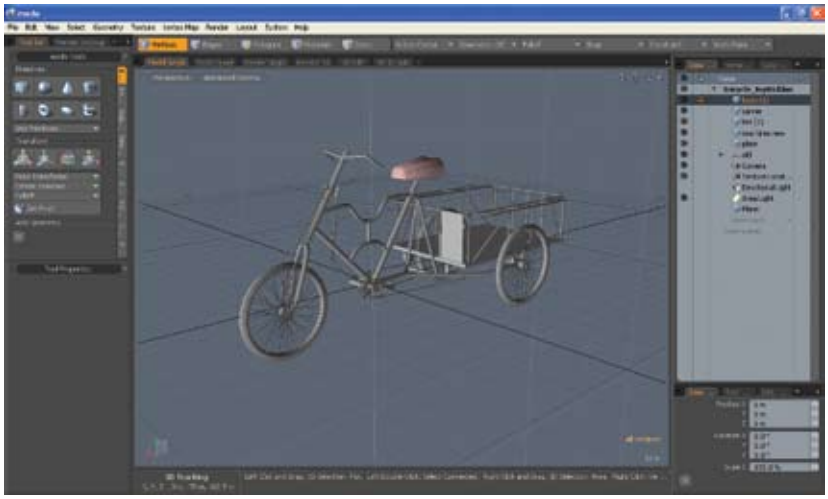


Fig. 38: Modo 202 interface

Navigation

Rotate: [Alt]+LMB-drag

Pan: [Alt]+[Shift]+LMB-drag

Zoom: [Alt]+[Ctrl]+LMB-drag

I like to switch off **Trackball Rotation** and activate **Orbit Selected**, so the camera is never tilted and always moves around the currently selected piece.

[a]: frame all, [Shift]+[a]: frame selection/active object

Selection

LMB: Select

RMB: Drag selection lasso, selects visible only.

MMB: Drag selection lasso, selects all.

[Space]: Circle between Modes: Vertex, Edge and Polygon Mode

[1]: Vertex Mode: LMB to select, doubleclick to select all

[2]: Edge Mode: LMB to select, doubleclick to select edge-loop (alt+l to select edge-ring)
[3]: Polygon Mode: LMB to select, doubleclick to select all connected polygons (l to select polygon-loop)
[4]: Material Mode
[5] or [Shift]+[Space]: Items Mode
[Up-Arrow]: **Continue selection** (something must be selected first), [Shift] + [Up-Arrow]: **Grow selection**
[Down-Arrow]: Undo last step of selection, [Shift] + [Down-Arrow]: **Shrink selection**
[Left-Arrow]/[Right-Arrow]: cycle through loop selections
"[": Invert Selection

Editing
Move-tool: [w]
Rotate: [e]
Scale: [r]
Element Move Tool: [t] (Just click on an element to move it, no matter if it is an polygon, edge or vertex.)
Transform: [y] (Universal gizmo with **Move**, **Rotate** and **Scale** all in one.)

[Ctrl]+[Space]: 3D-View Pie Menu
[Alt]+[Space]: Quick Access Popover
[Ctrl]+[Tab]: Modeling Pie Menu

UV-View: *Layout › Layouts › UV Edit*

Customization
Make your own **pie menus** for features that you often use, but that are hidden deep down in a menu-structure.

How to make a pie menu for switching between viewport modes:
Open the **Form Editor** [F3]
Create › New Pie Template
I name mine “Quick-Access Best of Viewport Controls”.
Start adding commands. Finding out what all the commands are called is easy: Just do what you want the pie menu to do and open the **command history** to look at its name. You can copy and paste it into the **Form Editor**, give it a name and description and move on to the next one. I create pie-menu-items to switch the wireframe on and off and to switch between the 3 shading modes that I use the most: Advanced OpenGL, Shaded and Reflection.

But I also want a quick function to turn grid, workplane and wireframe off all together with one short click. The easiest way to do this is to record a Macro. Just activate *System › Record Macro*, switch of wireframe, workplane and grid and click on *System › Record Macro* again so the recording finishes. Choose *System › Save to File*, save your macro to whatever location and try running it via *System › Run Script*. Then copy the command for running this script from the command history and paste it into the Form Editor. In this special case we have to do a little adjustment to make the macro work. For the **showWorkPlain**- and **showGrid**-functions modo is always using the argument “**True**” when toggling on and off. So if we always want our pie-menu-item to switch it off, we have to open the file in a text-editor and change “**True**” to “**False**”. Our finished macro-script should now look like this:

```
#LXMacro#
viewport.3dView wireframe:none
viewport.3dView showGrid:[False]
viewport.3dView showWorkPlane:[False]
```

Alternatively you could write a tiny Lua-, Pearl- or Python-script that combines these commands, and call that script from the pie-menu. I’ll do this for the script that

will switch everything back on again. This is what it looks like written in LUA:

```
-- lua
lx ( “viewport.3dView wireframe:colored” );
lx ( “viewport.3dView showGrid:[True]” );
lx ( “viewport.3dView showWorkPlane:[True]” );
```

Now all we have to do is assign it to a shortcut-key-combination. I use [CTRL]+[SHIFT]+key combinations for all my custom menus as I have my left hand on [Shift] and [Ctrl] all the time anyway and those combinations aren't taken yet by the default shortcuts. In this very case I choose [CTRL]+[SHIFT]+[x].

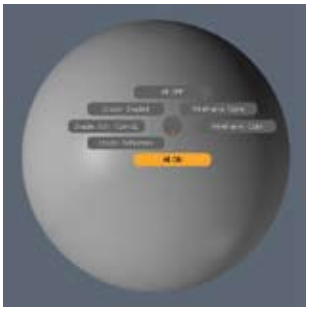


Fig. 39: Custom pie menu

Don't forget to quit modo and restart it so everything is saved. If it crashed all your customizations to the interface would be lost.

How to make a pie menu for selection sets:
The way selection sets work is a bit too clumsy for my taste. Having to click 3-4 times is too much for something I use quite frequently. So again I made a custom pie menu to speed things up:

Fig. 40: Custom pie menu

While modeling and especially during the uv-unwrapp process it's often necessary to make quite complicated selections. It's therefore very handy if you're able to save a

selection, so you can use all the selection tools (like **grow selection**, **continue selection**, **shrink selection**, etc.) without having to worry about messing up what you selected earlier.

I figured that I seldom need more than 2 **selection sets** at once, so this pie menu allows me to add elements to one of the two sets, recall the sets and empty them. If I'm done with a selection and I want to keep for later, I just convert it to a normal selection set.

Again: Building this pie menu was extremely quick and easy. I copied/pasted the commands for **add to selection set** and **recall selection set** to the **Form Editor**. The empty command is a **macro** that recalls the selection set and then removes the selection from the set, therefore clearing it. I adjusted the macro to work with an **Argument**, so I can use the same macro for both sets:

```
#LXMacro#
select.useSet %1 select
select.editSet %1 remove
```

In the **Form Editor** I just add the name of the **selection set** to the command that calls the script like this: `@{C:\...path...\mySelectionSets_empty.LXM} nameOfSet`
The "`%1`" in the macro is then replaced by the **Argument** given and I can now use the same macro-script to empty both my quick-selection-sets.

3ds Max 8 Quickintro

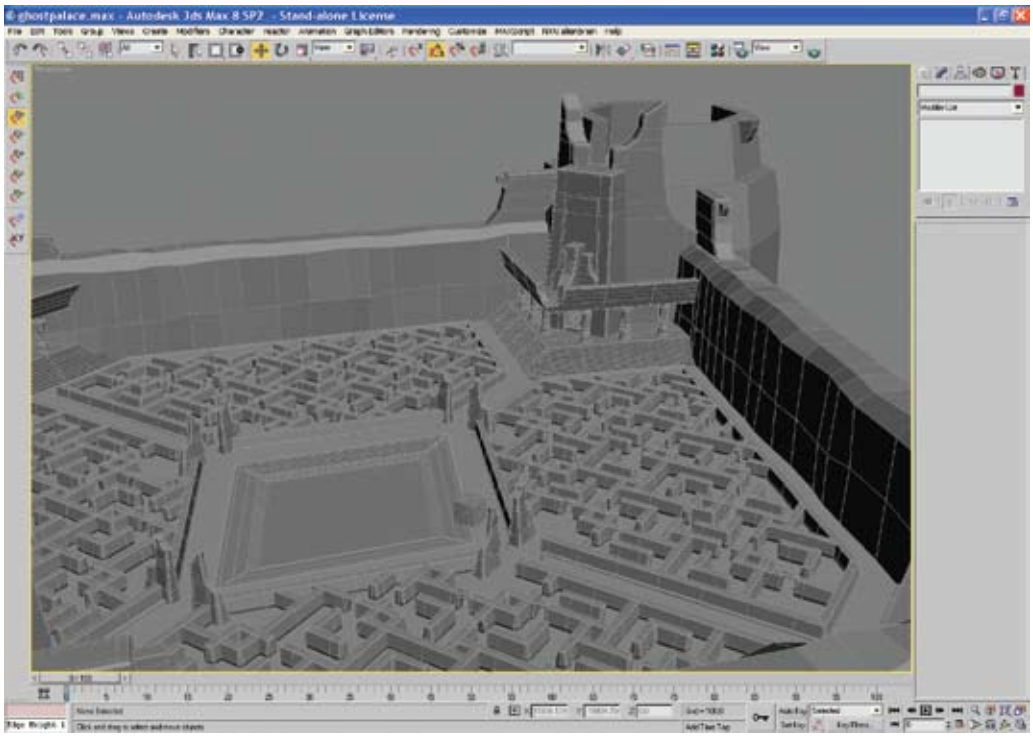


Fig. 41: 3ds max 8 interface

Navigation

Rotate: [Alt]+MMB drag

Pan: ([Ctrl]+)MMB drag ([Shift]+MMB drag: pan constrained to one axis)

Zoom: [Ctrl]+[Alt]+MMB drag up/down

Selection

In **Editable Poly** mode: [1] Vertex, [2] Edge, [3] Border, [4] Polygon, [5] Element

In **Editable Mesh** mode: [1] Vertex, [2] Edge, [3] Face, [4] Polygon, [5] Element

Ignore Backfacing is a vitally important checkbox for making good selections.

Editing

Move-tool: [w]

Rotate: [e]

Scale: [r]

Snap Toggle: [s]

[Shift]+RMB: **Snap Options** quad menu

Other

UV-Layout: Add an **Unwrap UVW** modifier to the **modifier list** of the object and click *Edit...* under **Parameters**.

RMB: Quad Menu

[Ctrl]+[x]: Toggle **Expert-mode**

Customization

3DS MAX is a really, really powerful tool. It can do everything. Everything. Unfortunately it has its own way of working and is not very well organized and therefore a bit clumsy to use. The only way to really work fast in 3ds max is to take the time, analyze how you work with its tools and then modify the interface accordingly. Be cruel. Change all the shortcuts, change all the pie-menus, change all the panels until all the features you use frequently are only one step away. If you don't use it for anything but modeling and texturing: scrap everything else.

Try to eliminate the most dangerous quirks. (For example: When in the **Unwrap UVW** window, [Ctrl]+[s] doesn't **save** the file anymore, the shortcut is overwritten by the **Snap Toggle**.)

Modeling Basics

The game's engine knows nothing of palaces and cities, nothing of monsters and heroes and nothing of swords and armor. For the engine every 3D object is basically just a list of vertices and some information on how these vertices are interconnected to form polygons that can be drawn on screen.

While artistic talent is what makes a great 3D model, a solid understanding of the underlying technologies is equally important. Without it there is no way of making sure that the carefully crafted creation will not lose its beauty when displayed in the game.

Another factor is experience. Only with experience comes a good feeling for how many polygons you need to build a certain object, and where to save and where to invest some additional geometry in order to make your model that extra bit better or more efficient.

So let's start at the very bottom and work our way up.

Modeling Basics I: Vertices, Edges and Polygons

Vertices, **edges** and **polygons** are the basic vocabulary of 3D modeling.

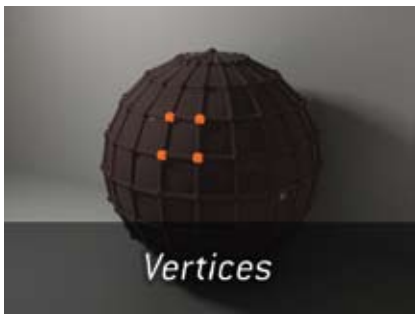


Fig. 42

Vertices: a vertex is an indefinitely small point in 3D space. *(For reasons of clearness I'm showing them slightly larger in my illustrations.)* Vertices are the basis of everything and edges and polygons can essentially also be thought of as groups of connected vertices.

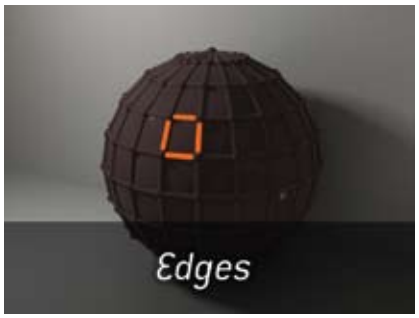


Fig. 43

Edges: an edge is the connection between 2 vertices. Two important kinds of edge-selections are the edge-loop and the edge-ring.

Fig. 44

Polygons: a polygon is a (planar) surface, defined by at least 3 vertices. Even though modeling is more convenient with 4-sided polygons (a.k.a. quads, quadrilaterals, quadrangles or tetragons) be aware that your model will ultimately be split into triangles (a.k.a. trigons), as that's what graphic cards work with.

Fig. 45: Single Edge, Edge-Loop, Edge-Ring.

Fig. 46: Single Polygon, Polygon-Loop, Triangle

Modeling Basics II: Modeling Tools

Now that we know what a 3D model is made of, how can we build one?

It works like this: First we create some basic geometry and then modify, extend and refine it until the model is done. Each 3D modeling application offers a big array of tools for this, that allow for many different approaches. We can either start with a primitive object like a sphere or a cube and work with that, or start from scratch by drawing polygons.

Working with **primitives** has the advantage, that they come with a default **uv-layout**. An example: you want to model a long and twisted tube. If you start by drawing the edges and then extend theses edges step by step, you'll have to manually unwrap

the entire thing in the end. But if you create a long cylinder primitive and deform that, you've already got a basic uv-layout that you can adjust to your needs.

The basic tools to create and modify geometry are similar in all modeling applications.
The **Pen** tool lets you create new polygons from scratch.
The **Extrude Polygon** tool moves a polygon and creates new polygons in between the original position and the new position.
The **Bevel** tool works the same way, but also scales the moved polygon.
Building an object is just a matter of knowing how all the different tools work and using the right tools in the right places.

With some talent and a bit of time it's possible to create pretty much everything you can imagine. But now we're getting to the point where the professionals split from the amateurs. I'm going to explain a few simple *modeling principles* and *techniques* that will help to not only get the job done, but to get it done in the most elegant and efficient way possible.
The key to a smooth look and an efficient model is good *polygon flow*. To understand what that means, we have to understand the way polygonal models are shaded by the engine.

Modeling Basics III: Shading Models

Polygons are flat surfaces, but for creating realistic imagery a smooth look would be far preferable over a faceted one. Subdividing the mesh of the 3D objects to the point where the faceted look would turn smooth was never an option as it would have increased rendering times far beyond acceptable limits, so other methods had to be found.
Let's take a look at the most important steps that have been taken towards a fast and

good-looking solution for shading rough polygonal meshes in real time.

Fig. 47

Wireframe

No lighting calculation is necessary, the renderer simply calculates the position of every vertex on screen and draws lines to connect them.

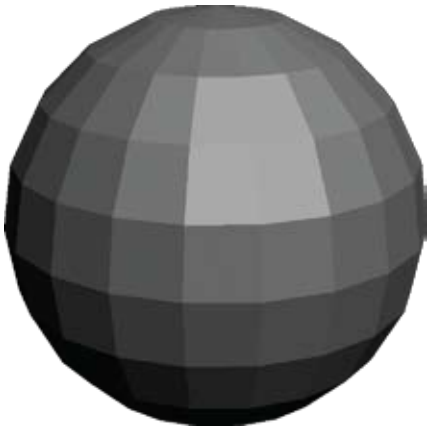


Fig. 48

Flat Shading

Each polygon's lightness is determined in accordance to a light-source and the renderer fills the screen-space of the entire polygon with that same shade.

Like in all basic lighting models the final shades are usually computed under consideration of these three components:

- The diffuse reflection (the basic shade of the surface, the color information), which is only dependent on the position of object and light. This is calculated using the incident angle of the light hitting the surface.
- The specular reflection, which is dependent on the position of the camera as well, as this is the part of the light that is reflected without being scattered. It is calculated from the direction of the camera and the vector of the reflected light on the surface.
- And Ambient lighting. This one "does not consider the location of light or the viewer at all." (Engel 2002) It just adds a general level of brightness to the entire model, suggesting a basic level of indirect illumination. (cf. Engel 2002)

Fig. 49

Gouraud Shading (per vertex lighting)

Then, in 1971 Henri Gouraud developed a method that allowed for a smoother look without requiring too much more computing power. It works like this: the renderer calculates the surface normal of each vertex by averaging the normals of all bordering polygons. Then it performs lighting calculations to determine the shade of every vertex and uses those to interpolate along the edges of the polygons. The screen-space of the polygon itself is "filled by lines drawn across the image that interpolate between the previously calculated edge intensities." (cf. Wikipedia, Gouraud Shading)

This method is fast to compute, but it looks quite rough and has one big disadvantage: Due to the nature of specular lighting and the fact that shades are only calculated per vertex it is possible that on rough meshes highlights get lost in between

Fig. 50: Lost specular highlight

vertices.
When rotating this simple sphere it is possible to reach a point where non of the 4 surrounding vertices produces a clear highlight. Therefore when the shades are interpolated the highlight is effectively lost.

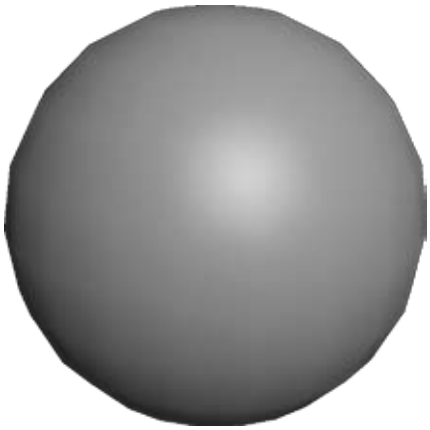


Fig. 51

Phong Shading (per pixel lighting, pixel shading)

Phong Shading doesn't just interpolate *shades* in between vertices, it interpolates the *surface normals* and then uses those to calculate the shades of the pixels.

> "The determination of the normal at a point on the surface of a polygon is achieved in the same way as the computation of the shading at that point with the Gouraud technique. The normal to the visible surface at a point located between two edges is the linear interpolation of the normals at the intersections of these two edges with a scan plane passing through the point under consideration. Note that the general surface normal is quadratically related to the vertex normal.
> From the approximated normal at a point, a shading function determines the shading value at that point." (Phong, p. 315)

In simpler words: Phong shading calculates a surface normal for every pixel, that is then used to calculate the shade of that pixel, which looks a lot more accurate than Gouraud shading as highlights aren't lost that easily.
The shading models used in current-generation games are mostly advanced versions of this *Phong Shading Model*.

Modeling Basics IV: Low Polygon Modeling Principles

Based on this knowledge we can now distill a small list of guidelines:

Keep the geometry clean. Avoid polygons with small angles, as the shading can yield weird results in that case. Don't arrange the geometry so more than 6-10 polygons share one vertex as this results in many small angles. For the same reason try to avoid very long and thin polygons.

Catch big changes in direction. Have a look at this spike:

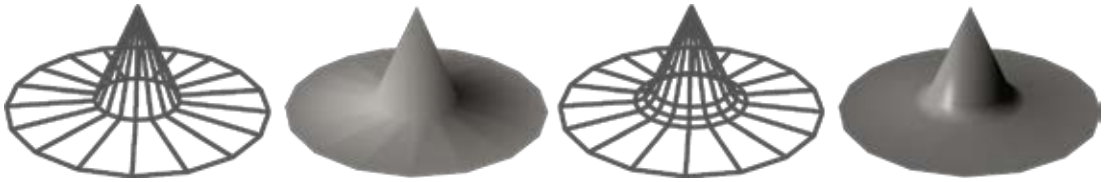


Fig. 52

Where it connects to the ground the interpolation reaches from the spike all across the base, and even though the base is perfectly flat, it now looks uneven. But if a few steps are added in between the spike and the base, the transition looks a lot more natural and the base looks flat again. The same is true for all sorts of rims and edges.

Avoid huge polygons. Every polygon is a space where the lighting is not accurately calculated but only calculated for the interpolated normal-vectors, and the bigger the space the more likely it is that the result looks unnatural. Most game engines use a **Level of Detail** system, where objects that are further away are displayed in less detail. In "Age of Conan", objects that are far away are not displayed with Phong shading, but with Gouraud shading. When building huge objects it's therefore necessary to make sure they look good in both shading models.

Following these simple principles will ensure a smooth-looking model. When working on a complicated object it's always helpful to look for something similar on the net and to try and understand why the creator laid out the polygons the way they are. Reading the forums at cgtalk.com and boards.polycount.net is always a good source for professional models and new tips and tricks.

Now we know how to keep everything smooth-looking. But what if a 3D object requires a sharp edge? Basically we have 4 ways to create sharp edges:

- We can break the model into separate pieces. (No smoothing occurs across polygons that are not connected.)
- We can use what is called "Smoothing Groups" in 3ds max. Smoothing Groups allow us to assign each polygon to one or more groups that will share one smooth surface.
- We can set a smoothing angle and let the modeling application break the smoothing automatically everywhere the angle between surface normals gets too big.
- We can increase the polygon-density of the mesh in certain places so that the object looks the way we want it to look even though it is smoothed across all polygons. This solution might require quite a few more polygons than the other solutions and is therefore less desirable from an economical point of view.

Modeling Basics V: Low Polygon Modeling Principles - Part 2

When preparing low-polygon base-models for work in ZBrush there are other factors to consider.

Resolution

As the **subdivision** works *per polygon* smaller polygons will have a “higher resolution“. If surface detail is especially important in a specific region the mesh must be split into smaller polygons there. If this is done correctly and you have higher resolution in the important places and lower resolution everywhere else, the performance of ZBrush will be much faster, as it doesn’t have to carry hundreds of thousands of unnecessary polygons.

Quads

While ZBrush works with triangles, the result will sub-optimal as the subdivided triangles can lead to weird-looking surfaces. Quads (4-sided polygons) look much better and it is therefore desirable to build the model out of quads.

Examples

Every project brings its own challenges and difficulties that require their own solutions. There's always a multitude of possible solutions, there is never just one way that leads to a good result. But the right ways are not always apparent from the start. The ability to recognize challenges from the start only comes with experience and study. In this section I want to share some of my experiences and results from the work part of this diploma thesis, the Palace of Kamula.

First I'm going to describe a *typical walkthrough* to give a general impression of the workflow and all the required steps and then I'm going to discuss several *examples*, detailing their individual challenges and how I solved them.

Typical Workflow

This is the workflow I follow most frequently:
Think › model low poly model › detail, create high poly model › unwrap › create normalmap › export

Think

Usually I start by analyzing the concepts. If necessary I collect reference material, research certain parts that I'm not familiar with, search architectural or anatomical atlases and go through the library of finished assets to see if I can save some time by reusing parts of a finished model.

Model low-poly version

Then I model the rough basic shapes, already forming a basic idea of how the high-poly model will look. I always keep an eye on the polycount and try to get the smoothing to look as natural as possible.
In some cases it is necessary to add some additional polygons to the model, so that it has enough "resolution" in the right places, once it's subdivided in ZBrush. In those cases it's necessary to rework the low-poly model after the high-poly version is done.

Model high-poly version

Now I take the model into ZBrush and start subdividing and detailing. I always try to establish a certain level of quality on the entire model and then give the most important parts an extra bit of polish.

Unwrap

Now before I can create the normalmap I need to unwrap the low-poly model. I like to save the unwrapping until this late in the process as the low-poly model often needs some adjustments after the high-poly version is done. Now is also a good time to make sure all the smoothing groups are well set up.

Create Normal Map

Once the texture-layout is done, I create the normalmap. The most convenient way of doing this is usually to use the ZMapper plugin in ZBrush. This step often also involves adjusting or reworking the low poly version to fit the high poly version and to assure good *polygon flow*.

Check and Export

Finally I check that the normalmap is looking good on the low-poly model and after the art director gives his okay, I export the model and move on to the next task.

The Palace Buildings

I started by looking at the concept drawings and since this location is closely tied to a quest, I read the quest document to get a good idea of about which spots are most important. Furthermore I read the world design document to get an idea of how this palace fits into this region of the world. I talked to the art director and started collecting some resources and reference images for the style we decided on.
Once I had a faint idea of how I wanted to approach this palace, I started modeling a very rough version. I didn't care about good polygon flow at this point, I just wanted to get the proportions of the concept drawing right and get a feel for the entire palace and its size and proportions.

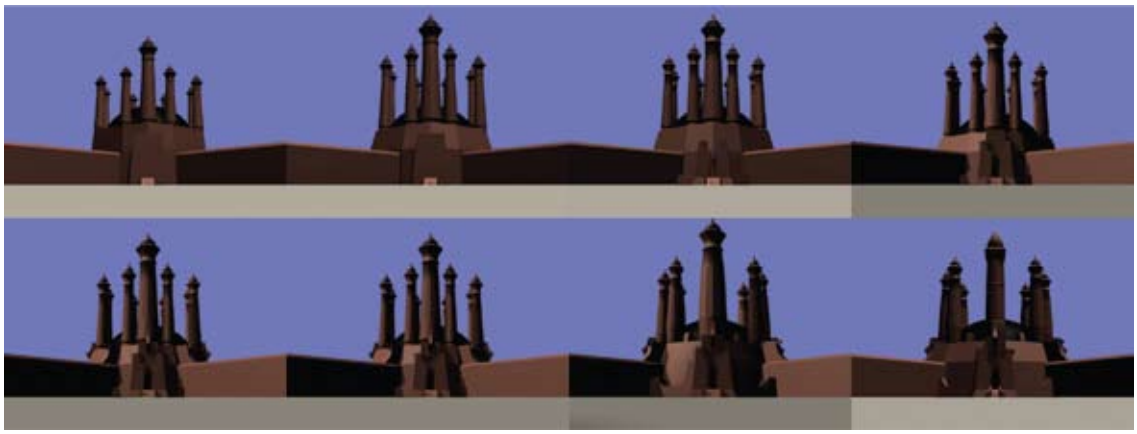


Fig. 53: The main building

I tried out several slightly different looks for every building and discussed a few perspectively ambiguous spots in the concept drawing with the concept artist. Once everything was to our satisfaction, I exported the raw model for the world-builders

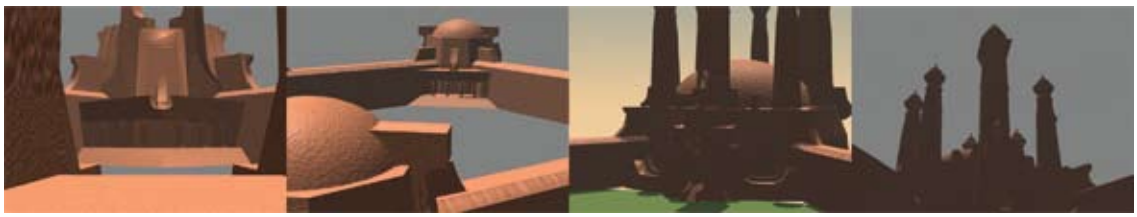


Fig. 54: The rough stand-in model with no uv-layout and a temporary noise-texture applied

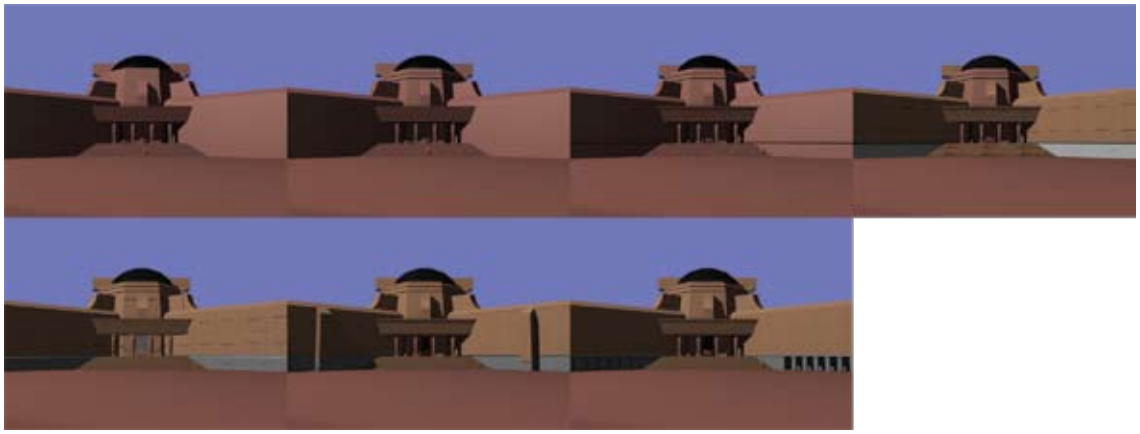


Fig. 55: The side structure

to use as a stand-in. I also put it in my own test-level and walked through it in the game, which gave me a good impression of which parts of the structure were catching the player's eye first and should therefore be modeled with the most care.
Before starting with the fine version, I did a little bit of research about big walls and round edges. I wanted to keep the polygon count as low as possible, but still achieve round edges and a fairly rough and uneven look. I came up with a good solution that I used consistently for all the round edges on the palace.
I placed several rough stand-in characters all over the place, so I could never lose the feel for the scale of things. Selecting one of them and rotating the camera around him was a good way to get into the player's perspective without having to export the mesh, place it in a level and start the game. Some of the factors that were decided "from the player's perspective" were the height of the stairs in relation to the height of the hedges, the size of the pool and the form and position of the gates.
From the start I wanted to give the players access to a spot where they could get an overview of the entire courtyard. I was considering several locations, but in the end I decided the front-panel of the entrance building was the best spot, as it gave a great view and I needed to model the inside of that building anyway. And while I really wish I could have created an inside for the main building and made the "observation deck" accessible from the other side of the entrance (so the players would have needed to walk through the courtyard before getting a look from above), unfor-

tunately the location just wasn't important enough to justify spending a lot of time creating additional assets. (And only creating a rough hall would have been a great disappointment for the players as the main building looks so gigantic and stunning from the outside.)

Once all the challenges were thought out, the actual modeling process went fairly fast. As I was also doing the texturing of the palace, I left some of the details for later, so I could add additional elements where the texture alone just wasn't interesting enough. One element that I added then was the tiled, snake-like band running around the 4 parts of the maze.

The unwrapping of the palace took quite a bit of thought. All the big spaces needed high-resolution texture-maps, but since the look would be fairly unique I wanted to reuse as much of the textures as possible. The solution I found will be discussed in more detail as an examples in the next chapter.

The last step in the modeling process was the creation of the collision mesh. To save processing time, the collision is not being calculated from the normal mesh, but from a much rougher model. There is no need for good polygon flow here, the main concerns are to cover all the important parts, so the player can't walk through something that looks solid, and to make sure there are no places where the player could get stuck. It is also allowed to "guide" the player a little bit by making some edges a bit smoother or extending certain parts so the player gets turned in the right direction if she accidentally walks into an obstacle. For our game, we're also adding special materials to the collision mesh that determine which footstep-sounds to play when the player steps on it.

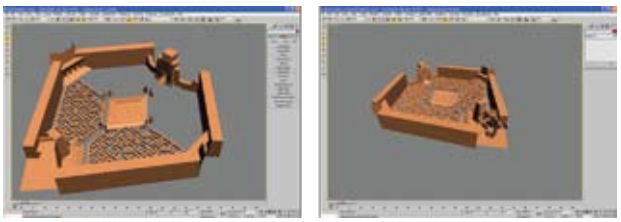


Fig. 56: Constructing the collision mesh

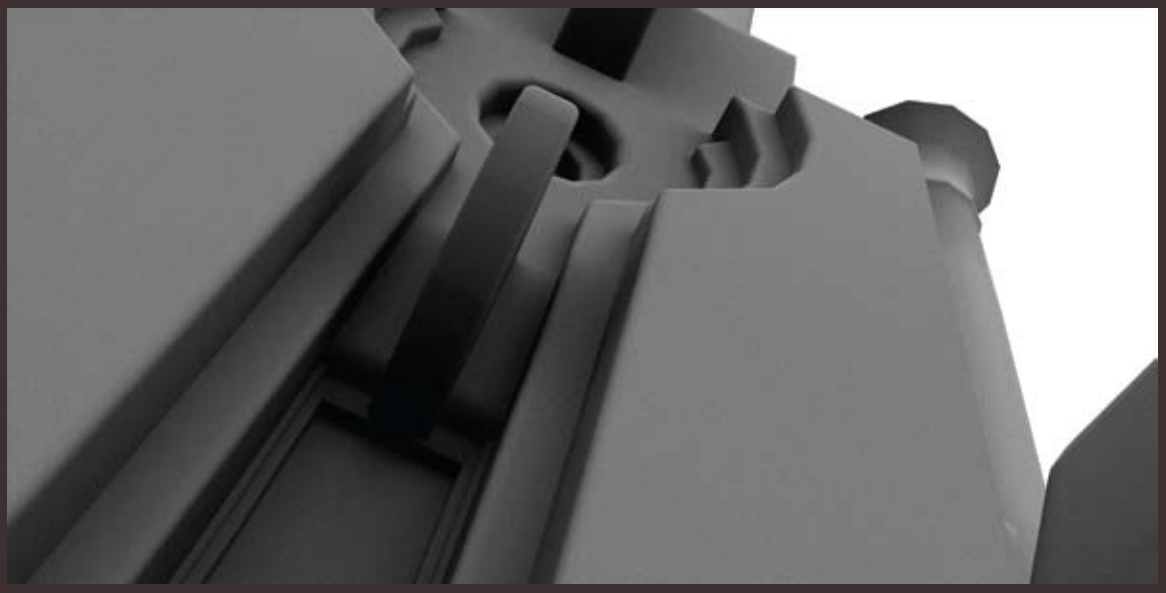


Fig. 57

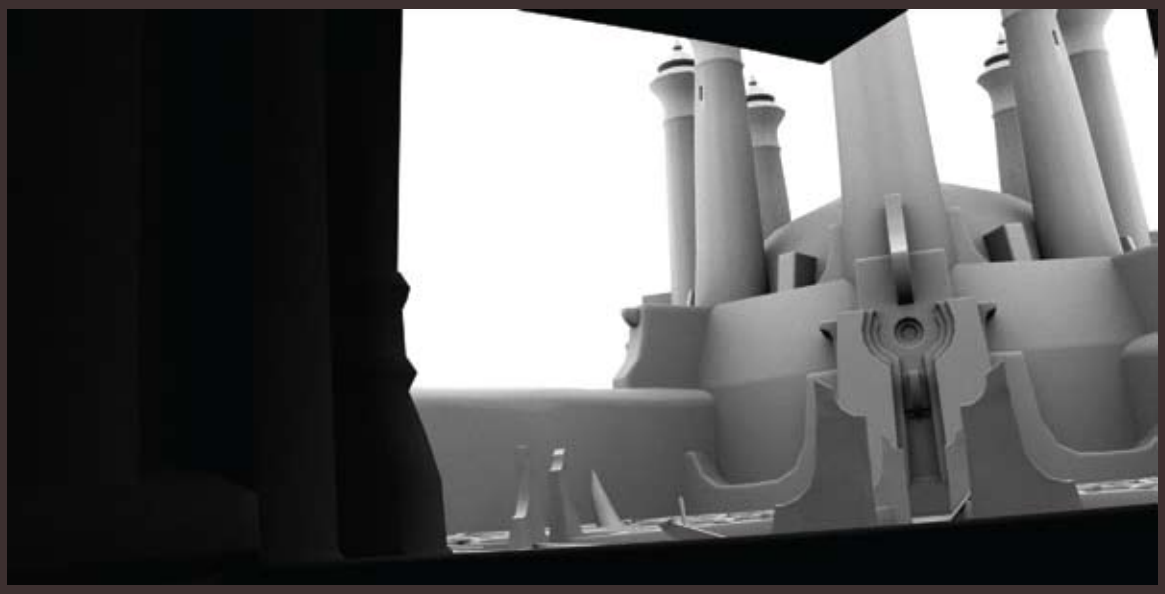


Fig. 58

Entrance Building Interior

This is more or less just the starting location of the player. The main focus here is to make it clear that this is a different time where the “Pillars of Kamula” were still part of a much bigger structure.
In the beginning I had big plans for the entrance hall, but then I figured the players wouldn’t spend much time here and only quickly walk through it anyway, so I tried to make everything as straight forward as possible. I arranged the room so the pillars where on one side and the exit to the courtyard on the other. Using the pillars as the doorway was a clever move, as it saved me from having to create a big main entrance. I created several generic pieces for the room and after those were done it was mostly a matter of stitching everything together.
I modeled one half, mirrored it and connected the two pieces. As I really liked the look of the Pillars of Kamula from above, and as I was already doing the staircase to allow the player to get up to the platform on the outside, I decided to create a simple overpath that connects the two staircases and gives the player a good look on the pillars from high above.

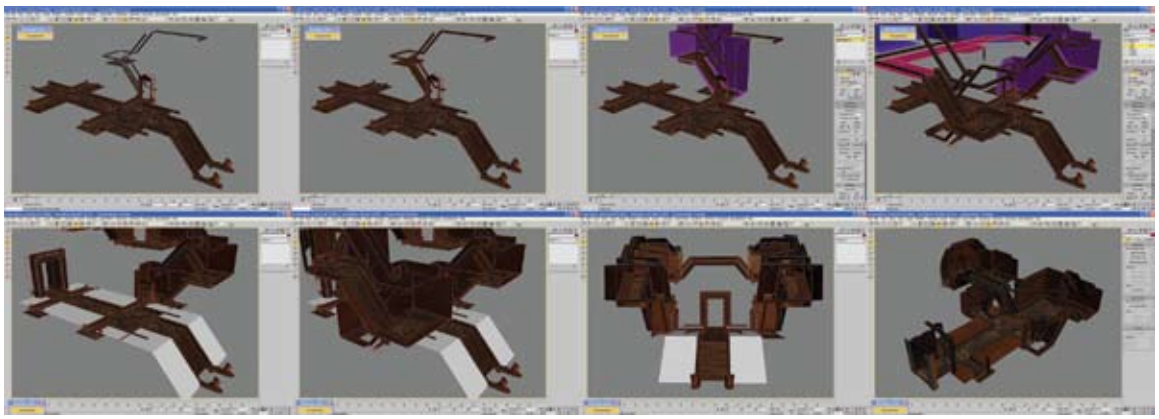


Fig. 59

As usual, modeling the collision mesh was the last step in the process.

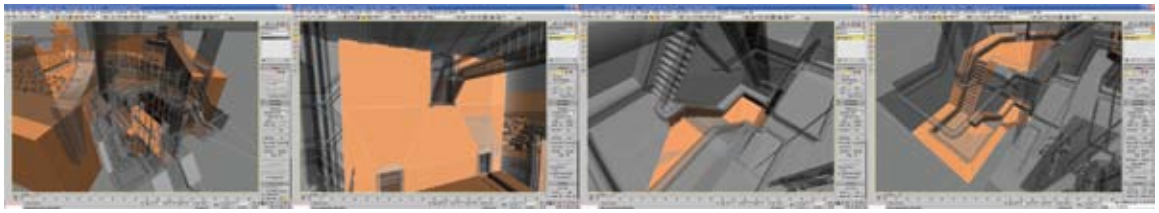


Fig. 60: Constructing the collision mesh

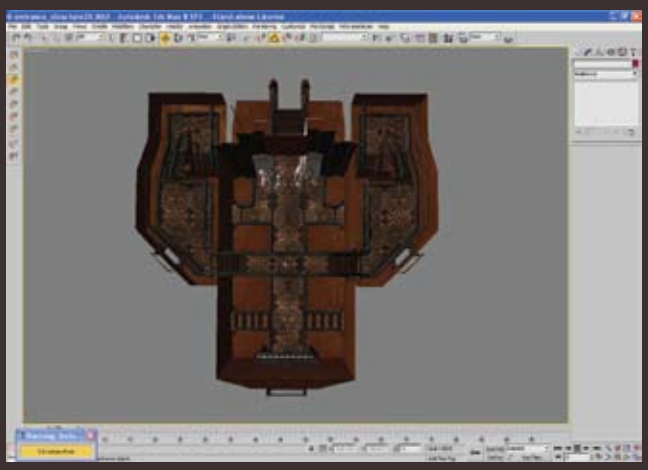


Fig. 61

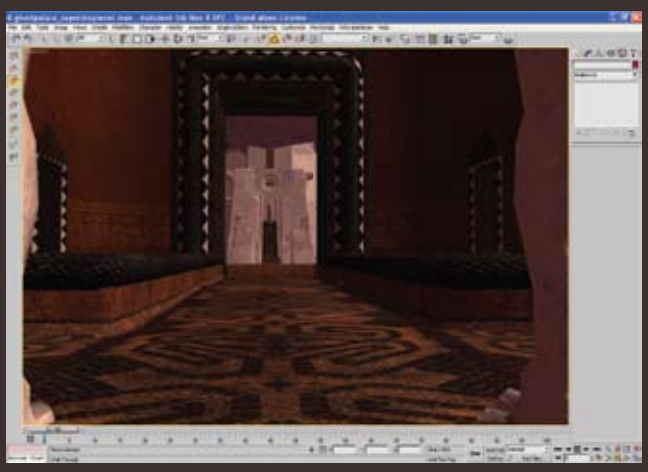


Fig. 62

The Pillars of Kamula

Using the tools and methods discussed in the last chapter I first constructed the low-poly version, then imported it into ZBrush, detailed it and used the ZMapper plugin to create the normalmap.

Low-poly Modeling

First I built the low-poly mesh in max. I reused some parts of other models and re-worked them to save a bit of time. I made sure everything was set up for subdividing and exported the model as in Wavefront OBJ format.

Fig. 63

High-poly Modeling - Typical ZBrush Workflow

In ZBrush I imported this file via *Tool › Import* and placed the object in the viewport. I activated edit object [T] and **deactivated RGB** (I did this out of habit, as I don't use ZBrush for painting and don't want to mess up my diffuse-map when I work on a textured object)

I started dividing with the **Subdivide Smooth Modifier** disabled, as I didn't want ZBrush to destroy all the hard edges that I wanted to keep (**Creasing** could have prevented ZBrush from doing that, but if the model is quite complex it just would have taken too long to select all the regions and set the right amount of **Creasing**). After I reached **subdiv-level** 3, I turned **smoothing** on again and divided 2 more times until I was at **subdiv-level** 5.

I set down the **Z-intensity** and started making everything nice and smooth and then

added all necessary details using a mix of **Zadd**, **Zsub**, **Smooth**, **Pinch** and **Nudge**. For this part of the process it was important to cleverly hide the right parts and reset the pivot point to rotate around the pieces that are not hidden, so I had a clear view on the surfaces I needed to work on.
I put in as much detail as I thought the **normal map** would be able to hold, but tried to keep the model rough as required by the background-story and the concept.

Fig. 64

UV-Unwrapping

Now that the high-poly version was done, I did the unwrapping of the low-poly-version. For this I used Unfold3D. This is a wonderful tool, but you need to feed it the right pieces. For that I needed to cut the lowpoly-model apart in the right places, which I did back in 3ds max.
I didn't use the original low-poly model from max, but the one I had at **subdiv-level** 1 in ZBrush, because the base topology had changed a bit while working in the high **subdiv-level**. I exported by switching down to **subdiv-level** 1 and choosing *Tool › Export*.
I opened the resulting file in 3ds max and cut it open where I wanted the **uv-seams** to go. Doing the cuts in all the right places requires quite a bit of experience and I will write more about this in the chapter about texturing. The big challenge is always to place the seams so that the model **unwraps** nicely (we want the texture as free from distortions as possible) and to keep the seams in places where they are as hidden as possible (You wouldn't want a texture-seam on the forehead of your character).
I exported the cut-apart model as .obj, opened Unfold3D, let it do the **unwrapping** (doing this manually for such a complicated model would have taken a lot longer. The

new **pelt-mapping** in MAX is fun, but no match in speed... The new **uv unwrap** tool in MODO would have been almost equally fast though.)
Then I took the model back into MAX to merge the pieces back into one surface and while I was there I also did some adjustments to the **uv-packing**. Some elements got a little bit more texture-space, less important pieces got a little less.
After I had exported the result once more to .obj, I was ready to generate the **normal-map**.

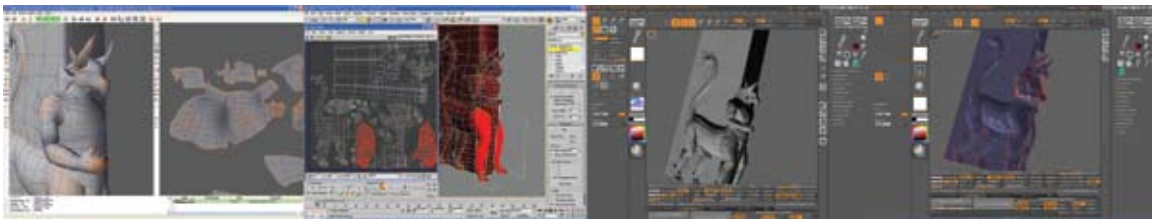


Fig. 65: Unwrapping, Adjusting UV Groups, Creating the Normalmap

NORMAL MAP GENERATION

For this I used the ZMAPPER plugin in ZBRUSH. But first I needed to update the ZBRUSH-model to the new version with the freshly created **uv-layout**. This I did by importing the latest .obj-file on top of the ZBRUSH-model at **subdiv-level** 1. (This only works if the model has the same count of vertices. If you optimized or changed something while uv-unwrapping you have to do things a little bit differently – see below) I started the ZMAPPER plugin, set all the correct values and clicked *Create Normalmap* under *Normal and Cavity Map*. This created a **tangent-space normalmap** that was placed in the **Texture slot** and that I could save out via *Texture › Export*. Now all that was left to do was to load the lowpoly-model and the normalmap in the previewer and inspect it. Everything looked fine and after the art director gave his final approval the model was done.

I will come back and texture it in the next chapter.

All in all the model turned out quite nice, but I wish I would have spent a few more

polygons on all the edges. The transition from the figure to the big block shows some imperfections, but I think I was able to hide it well enough with the texture.

Alternative Method 1

Zbrush Zmapper normal map creation from an object that is not a higher subdiv-level of the base-mesh.

Subdivide your highpoly model once more and go back down one level right afterwards. This is necessary because we want to capture all the details, but ZMapper doesn't start when you're on the highest **subdiv-level**. Now start ZMapper, go to the *Projection*-tab on the bottom and click *Capture Current Mesh*. This might take a while if your mesh is really highpoly. Once the mesh is captured, we leave the ZMapper plugin and import the low-poly mesh and place it wherever we want. It's important that the 2 meshes have the **pivot point** at the same spot, because then the ZMapper plugin will automatically align them. Trying to manually align them in ZMapper is (to say the least) hard... With the low-poly version in **edit-mode** we open the plugin again and choose *Create Projected NormalMap*. ZMapper calculates and places the result in the **Texture slot**. Now all we have to do is export the image via *Texture › Export* and check the result in the previewer.

Alternative Method 2

Normalmap creation in modo 202

In Modo 202 creating **normal maps** works via **object to object baking**.

Arrange the high-poly and low-poly models, so they are in an identical position.

Set up a material for the low poly version, make a new map and set it to be a normal map. In the *shader tree* right-click on the normal map and choose *Bake From Object*.

All visible objects that are in range will affect the normal map, no matter if they are marked as background layer or not. The range is determined by the **Displacement Distance** of the material set in the *Material Ref* part of the **shader**. Adjust this value if some height or depth gets clipped.

If the high-poly objects have a uv-layout in the same uv slot, this part of the normal-map might be overwritten, so be sure to either clear their uv layout or move it to a different slot before baking.

Alternative Modeling Approach

Model the high-poly mesh first, build a low-poly mesh afterwards.

This method offers more freedom at first, but can take a bit longer for simpler objects.

The advantage of this approach is that you can experiment in ZBrush without having to worry about the underlying low-poly mesh. You can model from scratch in ZBrush using **ZSpheres** and **deform**, **paint** and **bend** all you like.

After the high-poly mesh is done, there are several ways to create the low-poly version:

- Switch to the lowest subdiv-level, export it and rework it in 3ds max.
- Use Silo3D's "**Topology Brush**" to paint a low-poly version directly onto the highpoly-model. Silo also offers a handy "**surface snapping**" option, that allows you to pen polygons on top of another model.
- In modo a fairly similar workflow can be achieved by putting the high-poly mesh on a **background layer**, adding a "**Background Constraint**" to the **toolpipe** and using the **Pen tool** to sketch out the low-poly model on top of the high-poly version.

Once both versions are done, choose one of the 3 approaches described above to create the normalmap.

Fig. 66

Fig. 67

Hedge Maze

I wanted to create a reasonably big hedge maze, so the polygon-budget for all the individual hedge-pieces was very low. The timeframe was very short as well, so I had to keep it fairly generic for that reason too.
I started by modeling a very simple piece of hedge.
After that I created one tileable texture that I wanted to fit to all of the base parts, that I wanted to use to construct the entire maze. The tricky part here was that I wanted to mirror the texture to the other side, but I didn't want the mirrored seam on the top of the hedge. To work around that, I created a unique part for the top that tiles with the *front-part* on one side and the *same but rotated part* on the other side.
Once I had that texture, I modeled the rest of the basic parts and applied the texture to all of them. Now I just needed to copy the pieces and put them together to form the maze. At this stage I tried several different hedge-sizes in my test-level, so I could get a better feel for the maze.

Fig. 68: The hedge in my test-level.

In the end I constructed the collision mesh by creating a box in the right size and extruding polygons until everything was covered. I tried to model the mesh at the gateways in such way that the polygons point towards the opening, so if a player misses the opening just by a little bit she automatically gets turned in the right direction.

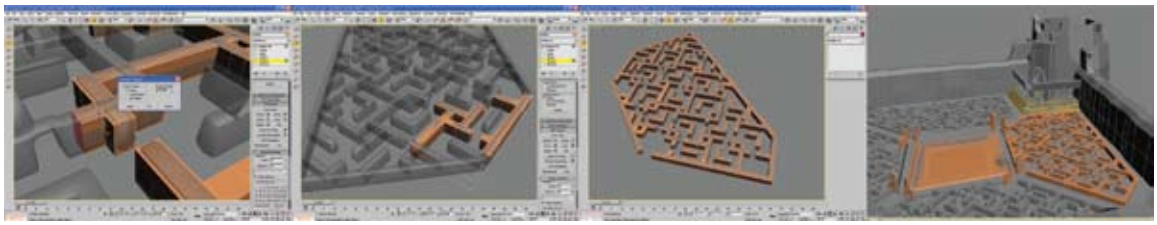


Fig. 69

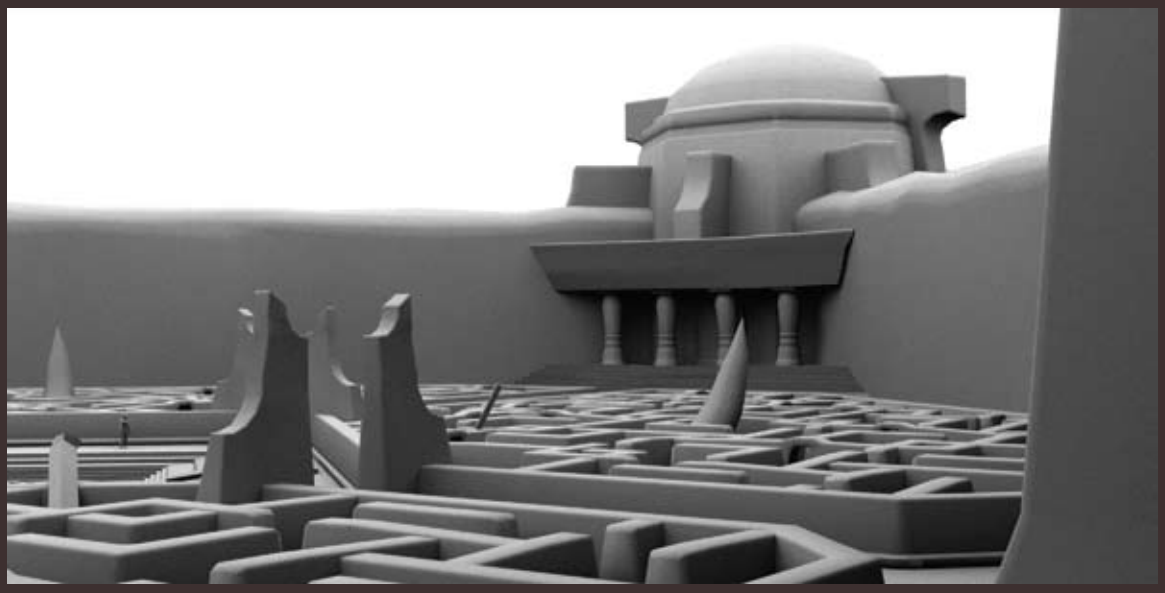


Fig. 70

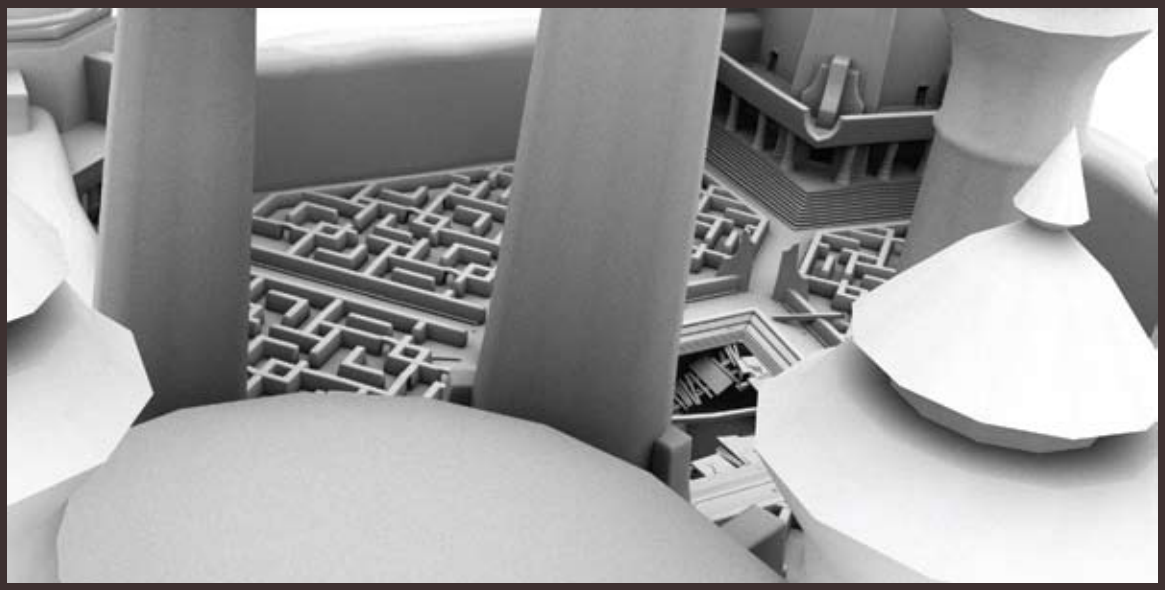


Fig. 71

The Palace Pool and the Eternal Well

This diamond shaped pool sits in the center of the courtyard and conceals the "Eternal Well", a vortex into nothingness. Once the ancient and evil well is stimulated, it swallows the floor of the pool and reveals how bluntly the Lemurians have built over this ancient site.

The normal pool is a very simple model, but the broken version has quite a few pieces of debris lying around and contains the vortex that is spiraling deep down below the palace.

I started by building a straight version of the vortex and used the **Vortex** tool in MODO (which is just an **Axis Rotate** combined with a **Cylindrical Falloff**) to twist it. The rest of the construction is mostly made up of a small collection of simple elements. I tried to create most of the textures beforehand or created at least a temporary texture, so I could unwrap and adjust the basic pieces before I duplicated them. I could have used instancing instead and unwrapped the basic pieces afterwards, but I did it this way so I could get a better impression of the overall look while positioning and adjusting. (Even without instancing, adjusting the uvs afterwards was no problem. I just select-

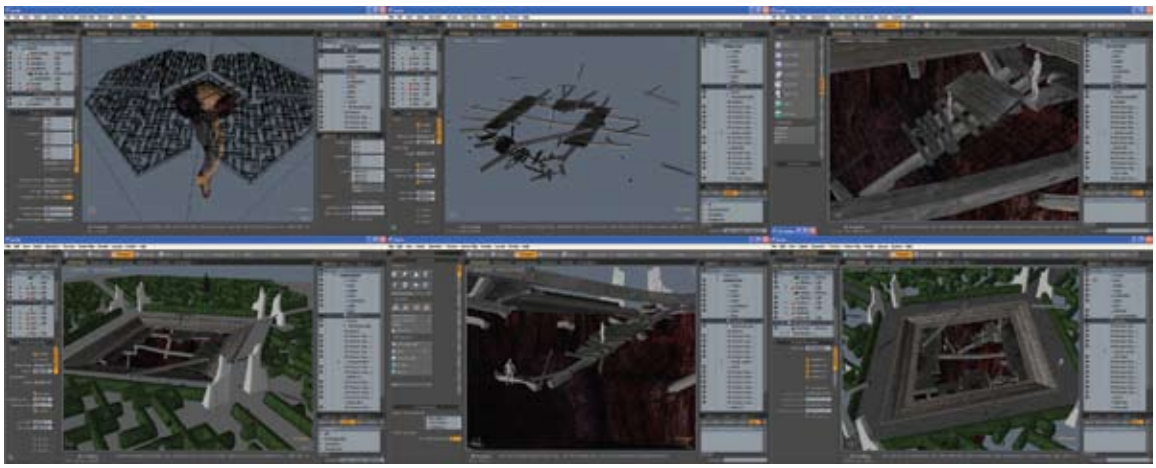


Fig. 72

ed all the objects and adjusted all their uvs at once.)

As always the last step was to create the collision mesh. Constructing rough forms around a chaotic scene was quite challenging. I didn't just want to model around all the individual pieces of debris, this would have just led to players getting stuck easily. Instead I wanted to build a single mesh that used slopes to make the player run smoothly over smaller pieces. There are still enough dangerous holes and pitfalls, but everything that looks like it should be no obstacle, isn't one.

This is exactly the sort of work where MODO excels. I made extensive use of the **action center** functionality and constantly realigned the **work plane**.
For moving individual pieces I set the **Action Center** to **Element**, I selected what I wanted to move, activated the **Move** tool, clicked on a polygon to align the **gizmo**, and dragged the element into its final position.

While working on pieces that were off axis, I aligned the **work plane** to them by hovering over one representative polygon in polygon selection mode and pressing

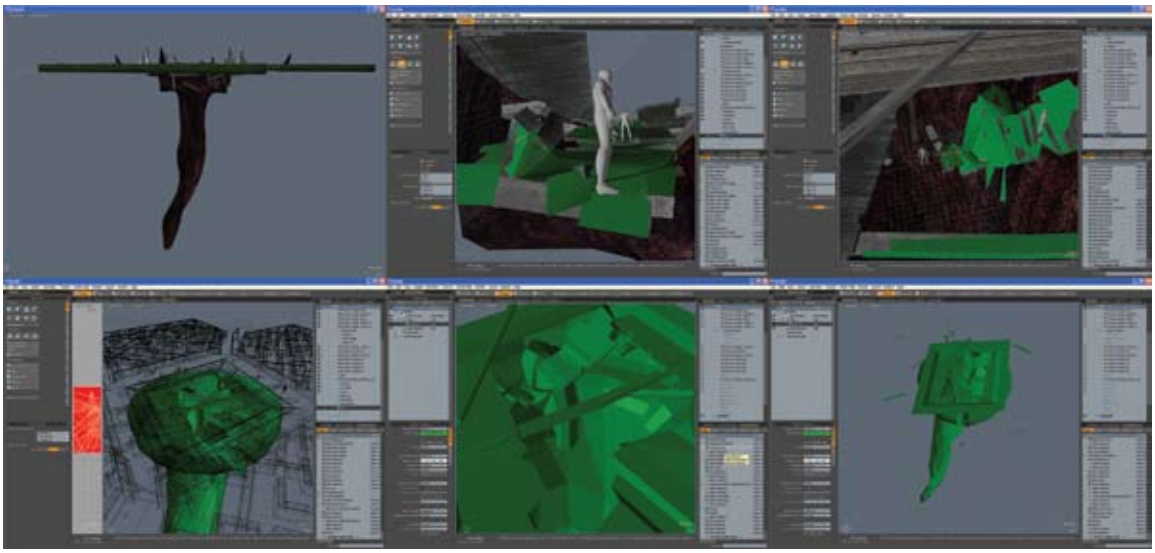


Fig. 73

[Pos1]. With the **Action Center** set to **Automatic** or **Selection Center Auto Axis**, all the **gizmos** of the tools would now automatically align to this plane, which made working fast and easy. After I was done, I could realign the **work plane** again or set it back to it's standard position by pressing [End].

Fig. 74

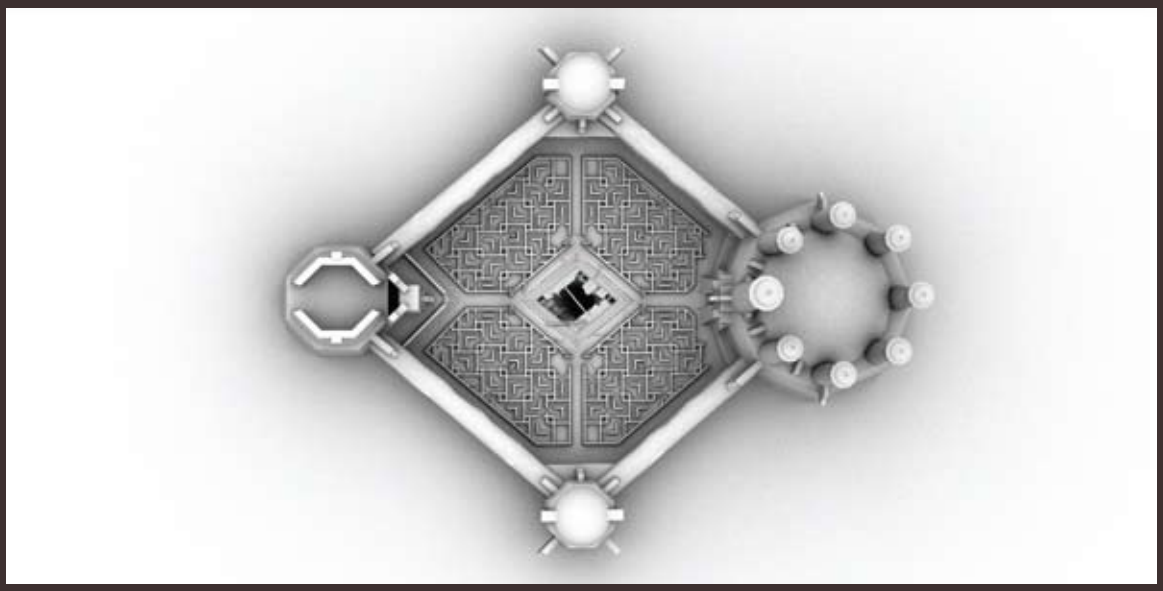


Fig. 75

Chapter 3: Texturing

Introduction

In the beginning of 3D computer games, texture artists were only creating single images with color information, but we have come a long way since then. We are not only applying color anymore, we're adding surface details, we define the feel of the material, set up transparency, translucency, incandescence and more.

Originally used for diffuse maps, the term "Texture" has come to describe the entire material. "Texturing" has come to mean the entire process of taking a naked 3D model and giving it its finished look.

Fig. 76: Naked model and textured model

The job of a texture artist is by no means less demanding than those of concept artists and modelers.
A texture artist has to be able to look at the physical world, analyze what she sees and split it into the right parts so she can recreate it as closely as possible with the tools available. She has to have a good understanding of *light and color*, *surfaces and forms* and a *good eye for detail* is just indispensable.
Furthermore it is always of high importance to find a good balance between quality (how good it looks), efficiency (how quickly an object can be finished) and effectiveness (how well it works with the game-engine, how little resources it requires).

In this chapter I will explain all the necessary basics needed for texturing 3D game assets. I will describe what maps and parameters are currently being used in game engines, how they influence each other and how to create and adjust them, and then walk you through several examples, each detailing a different approach and solution.

Basics

Components of a Texture

In the beginning of a project the parts that can make up a texture are defined under consideration of the game's art-style and the targeted system requirements. The programmers will then construct the game's graphic-engine to work with a list of images and parameters that can be defined by the artists. These maps and parameters all influence and modify the basic shading of the object that is calculated using the shading model. With every new generation of faster hardware this list grows longer, as game developers try to recreate reality more and more accurately.
The parameters available to the texture artists of "Age of Conan" are these:

Color
Detail map
Specularity
Normal map
Parallax map
Self illumination
Transparency
Translucency
Reflectivity
Light map

Let's look at them in more detail:

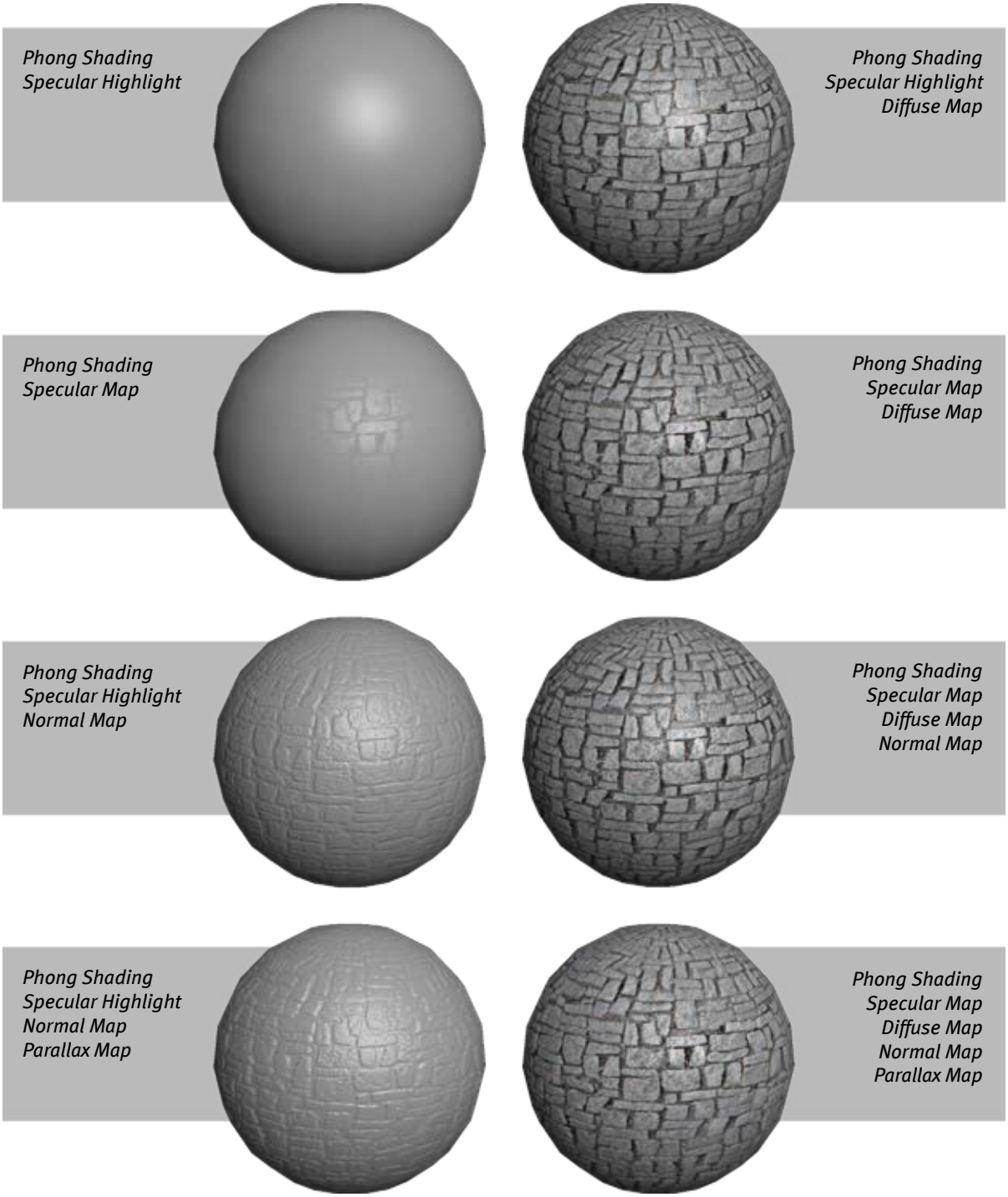


Fig. 77: The most common parts of a texture.

Color

Diffuse map – color information
Detail map – additional generic detail
Vertex painting – additional per vertex color information

What our eyes and brain perceive as the color of an object is essentially the wavelength of the light that is reflected from the object towards the eye.

"Light arriving at an opaque surface is either reflected "specularly" (that is, in the manner of a mirror), or scattered (that is, reflected with diffuse scattering), or absorbed – or some combination of these." (Wikipedia, Color)

In the diffuse map we want to capture only the scattered light, the plain color information; and preferably in a neutral lighting situation, as we want to leave the general lighting situation to the level-designer placing out the object. If we work towards a neutral lighting situation she will be able to use it in many different settings.

But even though diffuse maps should theoretically only contain pure color information it is often necessary to add quite a bit of additional information to the image as well. As several of the other options are only very rough approximations of the desired effect it is often good to paint part of the desired look into the diffuse map. Also it helps the overall quality quite a bit, if we try and make up for the shortcomings of the highly-simplified lighting model used in the engine. This can be done by adding details that can not be calculated in realtime, like ambient occlusion (the darkening of corners and parts of the surface that are less exposed to the open) or color bleeding ("The transfer of color between nearby objects, caused by the colored reflection of indirect light." (Birn 2002)). It won't be physically correct, but as long as it helps the look, no one will mind.

To prevent objects from looking too blurry when viewed up close, a *detail map* can be added. This is a small tileable image that blends over the diffuse map when the object is viewed from a close distance.

Vertex Painting allows the texture artist to add color information to each vertex in a model that is interpolated between vertices and is then used to modulate the color of the diffuse map. This is especially useful for adding a little bit of color variation to large areas that only consist of one tiled texture, as well as for adding shadows or a little bit of color bleeding to objects textured with atlases, where you can't add this to the diffuse map. Notice how the interpolation in between values is very rough, so good care must be taken about where to apply what amount of vertex color.

Specularity

Specularity map – amount of reflectivity
Specular color – color of the highlight
Roughness – *size and softness of the highlight*

The first thing we are going to put on top of the diffuse texture is the *specular highlight*. Specularity is an effect that simulates the reflection of a lightsource and provides much-needed highlights to the texture. While in realtime 3D engines specular highlights are circular or elliptical, in the real world they are actual reflections of the light-sources.

In this picture you can clearly see the reflection of the key lightsource, a simple table lamp.

Fig. 78

Furthermore there are reflections of the surrounding objects (notice the bookcover on the left side) and on the very edge there is a reflection of the window that acts as a rim light.

The example on the right shows that even surfaces that we wouldn't usually call "reflective" are of course also reflecting light, although in a much more diffuse way, and therefore the highlight is not clearly recognizable as a reflection of the lightsource, although that is exactly what it is.

Fig.79

You see, specular highlights are much needed to give a better impression of the type of material we want to represent. Rendering a uniformly colored object with a faint and soft highlight makes it look like cloth or paper. Rendering the same object with a sharp and small highlight makes it feel like plastic or metal.

Fig. 80

But textures usually don't contain only one type of material, so there is need for a way to control the amount of reflectivity across the texture. This is achieved by using a *specularity map*, an 8-bit grey-scale image that controls the amount of specularity on the surface. A black pixel means no specularity, a white pixel means the maximum

amount set in the material editor. Additionally it is usually possible to set the Roughness (also called Specular Power or Glossiness). This lets us adjust how smooth or sharp the specular highlight should appear and therefore how diffuse or glossy a surface will look.

Furthermore it's usually possible to set the *specularity color* for every material. This tints the specular highlight and allows us to adjust the fake reflection so it goes with the underlying diffuse map. (It would of course be possible to incorporate a specularity color map into the game-engine, but the improvement of the overall quality of the image would be barely noticeable.)

If the material is very diffuse or flat and doesn't require much variety, texture memory can be saved by not using a specularity map, but just setting a single value that will be used across the entire surface.
If the material is very diffuse and the specular highlight doesn't add much to the look, it might be best to switch it off entirely to save a little bit of rendering time.

ADDITIONAL SURFACE DETAIL

Normal map
Parallax map

Normal mapping (Dot3 Bumpmapping) and *Parallax mapping* are two ways of adding additional surface detail to a 3D model without actually increasing the mesh density.

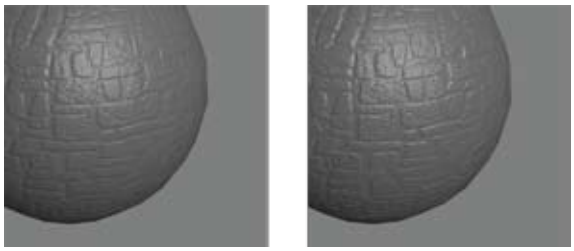


Fig. 81: Normal Map and Normal Map + Parallax Map

A *normal map* is a 24-bit-image, where the values stored in the R,G, and B channels represent X, Y and Z values of a *vector*. Basically what this means is that each pixel contains a small normal vector that tells the renderer how to change the lighting at that pixel on the surface: The renderer takes the rough polygon model and uses a variant of the phong shading model (per pixel lighting) to calculate the surface normal for each pixel. Then it modulates that surface normal according to the information stored in the related pixel of the normalmap and calculates the final shade of the pixel on screen. (cf. Wikipedia, Normal Mapping)

> "Normal maps [...] have the advantage that the expensive operations (computing the local surface normal by transforming the bump into the local coordinate frame) have already been performed in a preprocessing stage. All that remains to be done is to use the precomputed normals for shading each pixel." (Heidrich and Seidl 1999, p. 6)

The preprocessing mentioned here can either be done in a 3d application (by projecting surface detail from a high-density mesh onto a low-polygon model), in Photoshop (by converting a grey-scale heightmap using Nvidia's NormalMapFilter), or by a combination of both methods.
When modifying normal maps in Photoshop, some care must be taken: Taking up a color with the color-picker and painting it in RGB mode modifies all 3 channels at once. In most cases this will work just fine, but it's a good idea to check the individual channels afterwards, just in case something went wrong...

A normal map can be stored using different coordinate systems. The most commonly used one is tangent space, where X, Y and Z values "are relative to a [...] smoothly varying coordinate system (based on the derivatives of position with respect to texture coordinates)" (Wikipedia, Normal mapping)

This means the fake deformation is always relative to the surface of the object. This makes it possible to reuse normal maps that are not too specific. (cf. Cloward, p.2)

To check if a normal map is set up correctly look at the individual channels:

> "[T]he red channel should be the relief of the material when lit from the right, the green channel should be the relief of the material when lit from below, and the blue channel should be the relief of the material when lit from the front (practically, full except on the 'slopes')" (Wikipedia, Normal mapping)

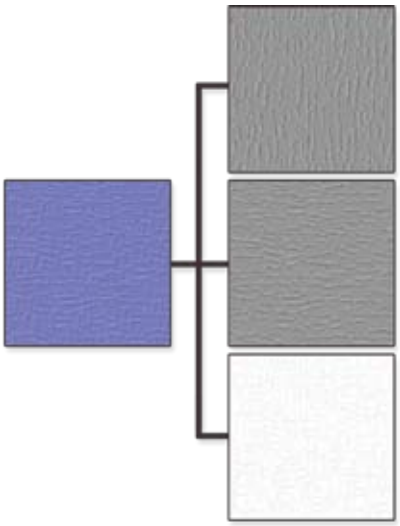


Fig. 82: The 3 Channels of the Normal Map

For characters and detailed objects the base normal map is usually generated by the 3d artist and the texture artist only has to add some more detail on top of that.
It is possible to put one normal map on top of another by setting its layer blend mode to overlay, but again a little care must be taken.
After any manual manipulation it is recommended to run the NVIDIA NORMALMAPFILTER in **Normalize Only** mode.

That the deformation is just a trick and that there is no actual geometry being added or modified becomes apparent when looking at the texture from an angle. The second method for adding additional surface detail is trying to address that problem:

Parallax mapping adds an impression of depth by moving color information on the surface, according to calculations based on a grey-scale height-map. If you look at a parallax-mapped object from an angle the silhouette will still be the flat, but the actual image will look as if it had depth.

Fig. 83: Parallax Mapping

> "With this method, motion parallax is realized by mapping a texture that is distorted dynamically to correspond to the destination represented shape. [...] We realize this texture generation process not by distorting the actual texture, but by shifting texture coordinates of each drawn pixels as the texture is mapped to the polygon online." (KANEKO 2003)

Parallax maps have to be used with care. First and foremost because they are quite expensive to compute and should therefore only be used where they can really help to improve the overall result. And secondly because they distort the color-information to achieve the impression of depth, which doesn't look very good when used too excessively.

Self Illumination

Self illumination amount
Self illumination map

Self Illumination gives the effect of a surface emitting light. It does not actually emit anything and doesn't influence any of the other objects, but what it does is to keep the texture visible up to a certain degree even if it's not exposed to any light and therefore make it look illuminated.

If only certain parts of texture should look illuminated it's a good idea to add a bit of bleeding to the outside of that region on the self illumination map. This makes it look less artificial and more like the surface is actually emitting a little bit of light. It might also be good to paint some bleeding into the diffuse map.

Transparency

Transparency map (3 possible modes: alpha testing, alpha blending, alpha safe)

Some objects are not only reflecting light, but also transmitting it in a linear way. This effect is called *transparency*. Light traveling through an object towards the eye means that we can see what's behind it. In the physical world light is usually refracted when entering and leaving transparent objects, which is why transparency alone is not a perfectly adequate way to portray transparent materials like glass and water.

There are several different ways to do transparency, the least expensive of which is *alpha testing*: It requires only 1 bit of information (but is usually stored as 8 bit alpha channel anyway); each pixel is either drawn on screen (white) or omitted (black).
A more expensive method is *alpha blending*: It is driven by an 8 bit grey-scale image, where the shade of grey determines how transparent the pixel is drawn onto the background. White means a pixel is opaque, black means it is fully transparent. To save some computing time our implementation of this doesn't make sure the sorting of the polygons inside one object is correct, so if the object uses several alpha-blended polygons on top of each other it is required to use *alpha safe*, which does the additional calculations and is therefore again a little bit more expensive to render...

Translucency

Translucency amount
Translucency map
Falloff
Fresnel
Tint

Translucency is a similar effect to transparency, but the light that is traveling through an object is scattered on the way and so the resulting effect is only a *shining through the object*.
By setting a higher Fresnel value the engine decreases the amount of translucency as the angle between camera axis and surface-normal gets smaller and smaller. An object with a high Fresnel value will therefore look more translucent at the edges and opaque in the center.

Reflectivity

Environment Map

Real reflectivity is quite expensive to compute, therefore games usually use all sorts of tricks to give the illusion of reflectivity. One of these tricks is using an environment map. No reflection is calculated, but an image (usually depicting the general light- and color-situation of the region) is applied to the object in such way that it looks like a reflection.

Texture Map Layout

All image-maps contained in a material share the same layout. The diffuse map must be applied to the object the same way as the normal map and the specular map and every other map used. (Theoretically it would of course be possible to use different layouts for each map, but it's usually done this way to keep things uniform and simple.)

The process of creating this layout is called *UV Unwrapping*. The name refers to the

translation of polygons from XYZ-space (a 3d space with practically no borders) to uv-space (a flat, quadratic space). This happens non-destructively, as it simply means spreading the same polygons in a separate 2d space, creating a map of how to apply parts of a 2d image to parts of the 3d object.

The perfect uv layout is free of distortions, gives important areas more space than unimportant areas, has the seams in places where they are not noticeable and is well-packed (uses the space efficiently).

The tools available are numerous. Every serious 3D package contains a toolkit for uv unwrapping, but especially in the last time there were many new developments that made people happy who had always regarded uv unwrapping as the least fun part of creating art assets.
UNFOLD3D manages to almost fully automate the process of uv unwrapping. (But only if you know what you are doing. Because it is still required to cut the model into proper pieces and only these pieces are then unwrapped and packed automatically. However for experienced artists this is a huge time-saver.)

The **unwrap tool** in MODO 201 is similarly efficient. Just hide everything you don't want to have unwrapped, select all the edges where you want the seams, activate the tool and drag until the result is satisfying. Using the **uv-relax** tool in **unwrap** mode with **interactive** checked, it is then possible to pin down any desired vertex and move those pins around with the rest of the uv layout deforming accordingly.

Fig. 84: The UV-Relax Tool in Modo

Version 8 of 3ds max has introduced **pelt-mapping**. This is an unwrapping technique that uses special relax methods to help the user create distortion-free layouts. This tool makes unwrapping complicated objects in max quite a bit easier, but not quite as easy and fast as it is in Unfold3D or Modo.

Types of Textures

Skin

The most specific type of texture is the skin. This is a texture, created for one object only, full of very specific details

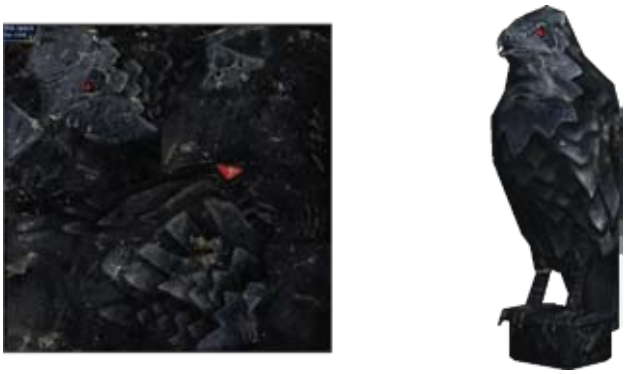


Fig. 85: Skin and Textured Model

Texture Atlas

This is a set of textures of a certain style combined into one big image that is used by many different objects. Usually this is set up like a construction kit that allows for texturing of a multitude of similar objects with the same atlas.
The reasons for creating texture atlases are not so much artistically as using a texture atlases is actually complicating the process in comparison to texturing with single textures. The mesh has to be cut in such way that the polygons can fit the tiles of the texture atlas in the uv layout.

Texture atlases are mostly used for performance reasons. Due to their generic nature, the same texture-map can be used by a great number of assets, which saves one of the currently most precious resources: texture memory.

Fig. 86: Texture Atlas and Textured Model

Tileable Texture

The common tileable texture is not only created for a single purpose, but is so general that it can be used all over. Stone, sand, earth and different types of walls are typical examples for this type.

As the level of detail in games is increasing, this type is becoming a little less important as most of the objects are either textured with texture atlases or get their own skins. They are still heavily used for terrain however. And there is always demand for cleverly designed textures that can easily be reused, as everything that can be reused saves time and resources.

Fig. 87: Tileable Texture

Tools

Traditionally Photoshop is the one big player in the realm of texturing, but 3D painting tools are becoming more and more important as they have several advantages over 2D tools: Being able to paint directly on the 3D model allows you to see the result of your work immediately; you don't have to save the file and switch to another application. It might even be possible to work on all the different maps at once. Paint a little bit of color here, a bump to the normalmap there, erase a bit of specularity in a third place, etc.
Another advantage is that it's really easy to hide seams as you can just paint over them directly on the model instead of having to try and paint in 2 places at once.
But there are some disadvantages as well: Sometimes it can be quite hard to get a good view of the polygons you want to paint. But this is not a big problem as it is usually possible to hide polygons wherever needed, or to just paint in the uv layout view.
And even though Bodypaint, ZBrush and Modo have quite advanced toolsets, especially working with photographic source material is just a bit more comfortable in Photoshop. But 3D painting tools are definitely starting to make their way into the production pipeline.

Photoshop Quickintro

Navigation
Zoom In: [Ctrl]+[Space]+LMB
Zoom Out: [Shift]+[Ctrl]+LMB
Pan: [Space]+LMB and drag
Hide Interface: [Tab]
Switch between **Screen Modes**: [f] (The big advantage of working in a Full Screen Mode as opposed to just maximizing the document window is that Photoshop then

lets you move around the canvas on screen.)

Selection
Select All: [Ctrl]+[a]
Deselect: [Ctrl]+[d]
Create Selection from Layer: [Ctrl]+LMB on layer thumbnail in layers palette

Layers
Select next/previous layer: [Ctrl] + "[" or "]"
Move selected layer up or down: [Ctrl] + [Shift] + "[" or "]"
Create new Layer: [Shift]+[Ctrl]+[Alt]+[n]
Select Layer from the canvas: [Ctrl]+LMB on a pixel of that layer that is more than 50% opaque. ([Ctrl]+[Shift]+LMB to add/remove a layer from the current selection)

Color
Set foreground/background color to default: [d]
Switch foreground/background color: [x]

Most important tools
Move Tool: [v]
Brush: [b]
Selection Mask: [m]
Rectangle/Line/etc. Tool: [u]

File Formats, Image Dimensions

The *color mode* is usually RGB, 8 bits per channel. The times of working with indexed

color images are pretty much over. The times of using higher dynamic range have not yet begun. CMYK was never an option as we're working for the screen.

The image *dimensions* should be set to a power of 2 (32, 64, 128, 256, 512, 1024, 2048, 4096). Most engines don't even accept images with other dimensions. Realistic values for current and next generation games range from 64x64 for small objects up to 4096x4096 for big texture atlases. A rough guideline for the art team of "Age of Conan" is to use 1 pixel for 1 cm. Our preferred file format is tga, but all the textures are later automatically converted to dds format.

The image *resolution* (ppi - pixels per inch) doesn't matter for our purposes at all. We're not going to print our textures, and since we're working in pixels from start to finish, a value that is used to translate from pixels into inches has no relevance whatsoever for us. (see Lebedev) Just leave it at 72ppi; the only time it might influences our work is when we use type – but if you set the image resolutions to 72ppi the type size in pixels equals the type size in points. However I recommend setting all values to pixels in the Photoshop Options anyway: *Edit › Preferences › Units & Rulers*

Work Big! Scale Down!

It's always a good idea to work as big as possible (and reasonable).
Especially when painting detail into the image manually, it's really hard to achieve a photographic level of quality, thus scaling the image down in the end is a good way of condensing the painted parts.

When scaling down tileable textures it's important to be careful about the *resampling mode*. While **Bicubic** usually gives a good result, with certain images it can show a faint border on the outer edges. But this problem is easy to solve, either by choosing **bilinear** as the resampling mode or by doing the following:

Tile the image 4 times (2x2), **flatten** and **scale** down the entire image.
Activate the **Rectangular Marquee** Tool, set its style to **Fixed Size** and set the **Width** and **Height** to the desired values.
Choose the right region (If it's a tileable texture with no special setup it doesn't really matter which part you choose. If there are other maps that have to go with this map, copy them all into the same file and crop everything together. If the framing matters, tile the image 9 times and crop to the one tile in the center.)
Crop to the selected region by choosing *Image › Crop*.
Done.

Source Material

Introduction

As we have learned, being a good texture artist is as much about delivering high quality as it is about working fast and efficient. Thus good source material makes every texture artist's life a lot easier. But where do we get it? The world around us is full of fantastic imagery, there are countless good pictures on the internet and there are companies selling DVD-collections full of material tailored to our needs.

In this Chapter I'll outline what good source material is all about, how to tell quality from mediocrity and how to overcome the most important challenges when you make your own.

What We Want

What we want are pictures of interesting surface textures, taken under soft light with no or minimal shadows, evenly lit across the entire picture.
We want images with interesting and easily reusable elements like cracks and dirt, either already extracted or easily extractable.
We want bumpmaps that are not just grey-scale-versions of the texture, but contain real height information.
We want heigh resolution and high quality.

A) Online-Archives/DVD-Collections

The main source of your material is usually either your studio's collection, your own photo-collection or an online archive.

There are countless sites and collections out there, but not all of them are good. And not all of them grant you the rights to use the images for whatever you want. This is especially important if you are working on a commercial project.

Here are the sites and collections I turn to frequently and why:

Online

www.mayang.com/textures
Free to use, 20 textures a day per ip-address, almost unprocessed, fairly high resolution, fairly good collection, no registration required, also available on DVD.

sxc.hu - stock.xchng
Free to use, huge collection but not only textures, so searching is a bit hard sometimes, some have usage restrictions/require authorization by the author, registration required.

photocase.de
Only free for 3 photos a day (or more if your photos are accepted into the pool), huge collection but only some textures.

flickr.com
Huge archive for photos of all sorts, but mostly copyrighted or not even available in big versions.

www.loc.gov
Library of Congress, big collection of photos and scans, mostly copyrighted though.

www.texturewarehouse.com

Free textures, use of these might be problematic as they are licensed under the Creative Commons License “Attribution-NonCommercial-ShareAlike 2.0” (if you’re working on a commercial project it might not be possible to share your results.)

stocktextures.com
My own project. An online archive for tileable textures and more, using creative commons licenses. Not officially launched yet, registration required.

images.google.com
Not the best source, it’s very hard to find good material here. Most images are copyrighted, too small and low-quality.

environment-textures.com
Paysite with a very fair price. Huge archive, all images are royalty free. The quality is alright, but I wish they would have used a better camera for all the great shots.

3d.sk
Paysite, human photo reference. Best resource for skin, hair, clothes, etc.

Collections

3D Total – Total Textures (http://www.3dtotal.com/)
High quality, a quite good collection.

Dosch Textures (http://www.doschdesign.com/products/textures/)
A good collection, fairly high quality, but some of the images are just too obviously computer-generated. I especially like “Ground“ and “Rust & Metal“.

dvGarage Water Damage (http://www.dvgarage.com/)

Small but great collection. Top-quality images and brush-presets.

Dover Pictorial Archive Series (http://www.doverpublications.com/)
Huge collection of books with patterns and designs of different cultures.

B) Photography

Contrary to popular believe it is not easy to take good pictures for texture-creation. Yes, it is easy to point a camera at a wall and press the shutter button, but taking a picture that is of highest quality and well-suited for texture-creation requires some skills:
We have to *know the camera* and which settings to use under which conditions to get the best result possible, we have to deal with *light, shadows and color*, think about the proper *size and framing* and be careful about perspective and distortions.

All of this I want to discuss in more detail on the following pages.

The Camera

For any serious user the choice of camera is an important one. The two main categories on the market of digital cameras are the **Compact Cameras** and the **Single Lens Reflex Cameras**.

Fig. 88: Compact Camera and SLR Camera

The one thing that speaks for compact cameras is their portability. I try to always carry at least a small camera with me – and if only to take a picture of the location so I remember to return with my SLR on another day.

The things that speak for an SLR are the higher level of control (Point-and-shoot cameras are often almost fully automated and don't give you the same level of control you get from a pro-sumer compact camera or an SLR), the wide variety of lenses you can use and the through-the-lens viewfinder (as opposed to a separate window in compact cameras).
Furthermore SLRs tend to have better and bigger image sensors. The image sensor is the light-sensitive device that captures the actual image.
While the Powershot S80 and the EOS 350D both take 8 megapixel pictures, the S80's sensor is less than 1/8th the size of the 350D's.

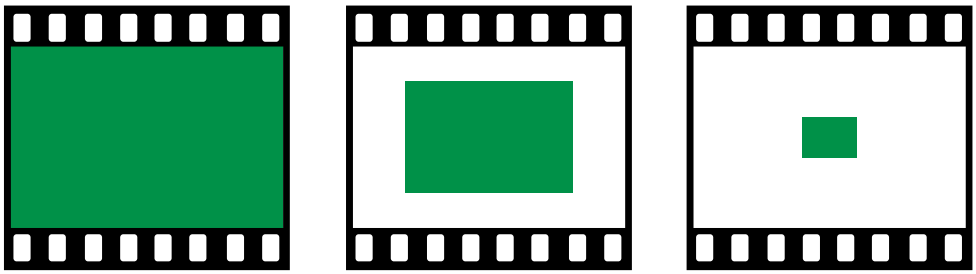


Fig. 89: Sensor sizes in relation to 35mm film

EOS 350D: 8 megapixels on 22.2 x 14.8 mm CMOS (Complementary Metal Oxide Semiconductor)
Powershot S80: 8 megapixels on 7.18 x 5.32 mm CCD (Charge-Coupled Device)

22.2 x 14.8 = 328, 6
7.18 x 5.32 = 38,2
328,6 / 38,2 = 8,6

In even more expensive cameras like the EOS 1Ds Mark II, the image sensor size can

go up to full 36 x 24 mm, capturing 16 megapixels.

The last thing that has to be considered is how the camera stores images. While JPEG is a great file format for keeping the file-size small, the lossy compression makes it impossible to store the full quality of what the image sensor is able to capture. Therefore it's preferable (though not always necessary) to use RAW formats, as most of those don't have lossy compression and stores the entire range delivered by the sensor without the data first being modified by whitebalance, contrast, saturation and other settings. This allows for changing many of the shooting parameters after the actual exposure. The downside is of course that it takes up quite a bit more memory, is slower to view and has to be processed with special software before it can be used. It might therefore be a good idea to set the camera to shoot in RAW and JPEG at the same time, or to batch-convert the RAWs to JPEGs for viewing-purposes and only return to the original raw when needed. (cf. Atkins)

It's better to have a usable image of mediocre quality than to have no image at all. But it's never wrong to strive for the perfection. The better the quality of the pictures, the more possibilities we have later on!

Taking the Picture

The 3 things that play together when taking a picture are the *film sensitivity*, the *f/stop* and the *exposure time*. These three elements have to be in balance to produce a good picture. There is however a great array of options as to how this balance is achieved.

A popular metaphor amongst photographers likens the taking of a picture to the filling of a bucket with water. You can fill the same bucket by pouring in a small stream for a long time or a fast stream for a short time.

> "It doesn't matter which combination of stream size and length of time you choose as long as the right amount of water ends up coming in. Film is the same; within limits, it is indifferent to the combination of time and amount of light as long as the right amount of light eventually arrives." (Cole)

Film sensitivity (ISO-number)

The sensitivity of the "film" means the *speed* of photographic negative materials. Since digital cameras use **image sensors** instead of **film** an *iso-equivalent* is given. A higher number indicates more sensitivity to light. This means that less light is needed to take a picture and thus we can either go down with the exposure time and/or up with the f/stop. *But* more sensitivity also brings more noise.
In our little metaphor the film-sensitivity is represented by the size of the bucket. (cf. "What Is... ISO")

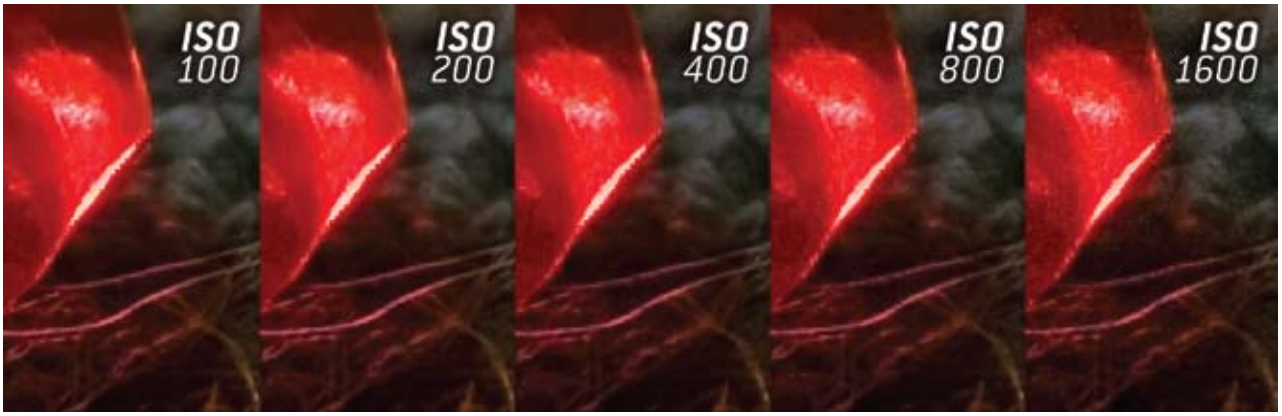


Fig. 90

f/stop

"The f/stop is [the ratio] [...] between the diameter of the aperture in the lens and the focal length of the lens" (http://www.uscoles.com/fstop.htm) Basically this tells you how far the **iris diaphragm** opens and how much light will come in at once.
A small f/stop means a lot of light is coming in, a big f/stop indicates little light coming in.

The available f/stops depend on the lens of the camera. Fixed lenses can usually do smaller f/stops than zoom-lenses. (Usually lenses are advertised by their maximum aperture = biggest opening of the iris diaphragm = most light coming it at once = smalles f/stop)
By setting the f/stop you control the depth of field around the focus point of your picture.
A small f/stop gives little depth of field, a large f/stop gives large depth of field.
"Depth of field (DOF) is the distance in front of and behind the subject which appears to be in focus." (Wikipedia, Depth of Field)

Fig. 91

Exposure time a.k.a. *shutter speed*
The *exposure time* describes how long the *digital sensor* is exposed to light and is measured in seconds and fractions of a second. (eg. 1/60) When taking pictures without a tripod it is especially important to keep an eye on this. With bare hands it is almost impossible to keep the camera steady enough to take sharp pictures if the exposure time is longer than 1/60th of a second.
A fast shutter speed makes it possible to capture fast moving objects, a slow shutter speed gives soft and blurry effects.

Fig. 92

Challenges

There are 3 main challenges we have to master to get great source material for textures:

- *Lighting/shadows/color*
- *Framing/size/resolution*
- *Perspective/distortions*

Lighting/shadows/color
Unless you want a very particular look, always aim for a neutral lighting situation. Soft light with no harsh shadows is ideal as this gives you the best colors and an equally lit image. In real world terms this means a cloudy sky during late morning or afternoon..

Taking pictures in bright sunlight is not optimal for our purposes. Direct sunlight creates hard shadows, which are undesirable if you're creating diffuse maps that have no special lighting situation already "baked in". Also the high contrast between highlights and shadows may supersede the dynamic range of the camera and there-

fore make it impossible to capture both areas at once. The contrast might even be so harsh that the light areas start bleeding into the dark ones.
All this might not be so bad if you only want to use bits and pieces of your image, but if you want to use large parts of your picture for a texture, you should consider returning another day when the lighting situation has improved. You might try the HDR-approach (see page 117ff), as it makes it possible to better an unbalanced lighting situation up to a certain degree, but if half of your motive is in plain sunlight and half of it is in shadow, the two parts will be just too different, even after you compressed the high dynamic range and adjusted brightness and contrast. (Soft light tends to hide the surface structure while hard light tends to bring it out more, so even if the two parts of the surface are adjusted they will still have different character.)

If the lighting situation is too dark, don't use the flash unless you really have to. It might sound like a good idea at first, but the resulting image will be darker at the edges, which makes it hard to use for tileable textures and the flash creates weird sharp shadows that look very unnatural. If I encounter a situation like that I set the exposure time to 1/60 sec and take a darker picture that I can maybe better in Photoshop, instead of a blurry one that I can't use at all. If you have to take a picture in a very dark spot, bring your tripod and do several longer exposures.

When composing a texture from elements of different photos the direction of light becomes very important. If you don't put much thought into it, the result may just look wrong, while you can't quite put your finger on what's the problem with it. Having shadows in the diffuse map of a texture can produce weird results, especially in combination with a normalmap and dynamic lighting. On the other hand it's sometimes good to bake a decent bit of soft shadowing into the diffuse map to make it look a little bit more realistic...

If you have the perfect motive, but it's never in the right light and you are really hardcore about this, you could get portable soft lights and return at night. But good lights

require a lot of power, so you'll need either an outlet or a generator. And you'll need several people to help carry and set up the set.

> "Flash on camera is sometimes useful. In very dark areas, it helps to have something like a Minicool w/ frosted glass diffuser. Beautiful, perfect lighting is not necessary. These images are going to be manipulated like crazy anyways. Avoiding hotspots (flash on camera vs. tile, etc.) and achieving good overall illumination is important." (Monolith 2006)

Familiarize yourself with the whitebalance-functionality of your camera. Color might not always be important to you, but it's never false to try and capture the color situation as good as possible anyway. A well-balanced picture holds a lot of potential for heavy use of the color adjustment tools in photoshop. In certain situations you might of course also try to tint an image towards a certain color to make it easier to extract parts of the image using aforementioned tools.

Be aware that you will also have to create normal, specular and other maps that go with your diffuse map. If you have your camera on a tripod and can take some more pictures with different settings, that might make it easier to generate those.

Framing/size/resolution

Choosing the right framing of the motive sounds like an easy thing to do at first, but actually requires some thought:
On one hand we are driven to take a smaller part as we want the image to have the highest possible amount of detail, but that might make it harder to texture huge objects. On the other hand we want to frame as big as possible so we can easily do large-scale textures, but then we might end up not having enough detail when we need it.

But always keep in mind that due to the way lenses work, the picture will lose sharpness towards the corners and might have a little bit of a vignette effect as well.

If you know exactly what your texture will be for, do some calculations and adjust the framing accordingly. The only general solution is to form a habit of always taking a series of pictures that covers all the important sizes and angles. Always take as many photos as possible (and reasonable). The more material you've got to select from, the better.

When in need of a really high resolution texture, consider taking many pictures and stitching them together. But be aware that the change of perspective from one side of the picture to the other plays a big role here, so choose quite a big overlap...

But often times the biggest problem will be to get a good angle on the motive. The best advise I can give here is: Use your environment!

Perspective/distortions

If possible try to take your pictures with the camera-vector normal to the surface. (This is especially important if you want your texture to look good from all directions!) If you have to take your picture from an angle you can still try to adjust the perspective in photoshop. But there are a few things to consider: If the angle is too far off normal you'll lose a lot of resolution when correcting the perspective. Also if the motive is very plastic or some parts are set into the surface the perspective adjustment will of course not be able to correct that.

Fig. 93: Original photo and corrected image

But even with the camera normal to the surface, perspective can become a problem: If we have a very plastic surface we see the left part from one angle and the right part from another. As mentioned in the previous paragraph, this will make it harder to stitch images together or to make one image tileable from one side to the other, as that's when these different angles collide.

Fig. 94

Theoretically it would be best to put on a big lens and zoom in from a great distance so that the difference in perspective is minimized, but that of course brings its own set of problems: More light is needed to do a proper exposure, it's much harder to keep the camera steady, and most importantly it's not always possible to get that far

away from a motive without hitting a wall or other things getting in the way.

Even though it's often only hard to notice, there's always a little bit of lens distortion in the photos. It's not always necessary and good to correct that as the recalculation of the image lowers the quality a little bit, but if your resolution is high enough (so you're scaling down in the end anyway), if it's too noticeable or if you need to adjust the perspective anyway, try the **Lens Correction Filter** in Photoshop: *Filter › Distort › Lens Correction*... It gives you quite a lot of control and makes it quite easy to correct perspective, lens distortion and vignette effect at once.

But with all these possibilities and challenges we should keep in mind that efficiency is about as important as quality. In a professional environment you won't be able to take several hours to get the *perfect shot*. But if you know your equipment and the basics by heart, you will be able to quickly make the right choices and get a *good shot* in a very short time.
And even if you can't get a good shot, a mediocre shot is often better than no shot at all.

A little Digression: Normal Map Photography

Ryan Clark has experimented with an interesting technique for creating normal maps out of photos. This is how it works: Take several pictures of the motive with the lighting arranged in such way that it resembles the way the different channels of a normal map look, and then desaturate and adjust them and combine them into one picture. (cf. Clark) This technique seems to work fairly well, but requires a lot of effort and is only applicable to objects that are fairly flat, small and moveable, yet complex enough to justify the long procedure.

A second little Digression: High Dynamic Range Images

HDR is a big buzz-"word" at the moment, and it's not always easy to understand what people are actually talking about when they say "HDR".
Some are referring to their use of HDR Images as light-probes for rendering 3d scenes.
Some are talking about the bloom- and auto-adjusting exposure effects found in the Half Life: Lost Cost tech-demolevel and more and more current games.
Some are talking about combining different exposures of the same scene into tone-mapped images.
And some are talking about file formats that allow storage of images with more than 8 bits per channel.

What I am going to outline here is not how to create HDR textures, but how to generating HDR images to use as source material for normal low dynamic range images.

So what does **High Dynamic Range** really mean?
High Dynamic Range images make it possible to store a very high range of values and can therefore contain shadow and highlight areas in high detail.

Traditional images use 8 bits per channel.
That means that each pixel in each of the 3 channels Red, Green and Blue can have a value between 0 and 2^8 (256). All 3 channels together give a range of 256^3 (16 777 216) different shades per pixel.

Now let's look at an HDR image with 32 bit floating point per channel:

> "Instead of using 32 bits to describe 4,294,967,296 integer numbers, 23 bits are allocated to a fraction, 8 bits to an exponent, and 1 bit to a sign [...] Practically speaking, this allows for an almost infinite number of tones between level '0' and '1', more than 8 million tones between level '1' and '2' and 128 tones between level '65,534'

and '65,535', much more in line with our human vision than a 32 bit integer image " (Bockaert)

Let's look at an example:
We are standing in a dark hallway. Up ahead we look into a room with two windows, but the blinds are closed, so only a moderate amount of light is coming in. In the far distance there is another hallway with an open door that is hit by bright sunlight.
The dynamic range (the distance between the light and dark parts) of this scene is big, bigger than what we can capture with a normal camera. If we take a picture with short exposure time, we can see the sunlit door in good detail, but everything else is close to black. If we take a long time exposure the shadow areas are clearly visible but everything else is burnt and white.

An HDR Image could contain all the details in the shadow region as well as all the detail in the highlights.

Fig. 95

So how do we get an HDR image if our camera can only capture a limited range?
We're using a method that was developed by Paul E. Debevec and Jitendra Malik at

the University of California at Berkeley and has recently been implemented in Photoshop.

> "[...] Multiple photographs of the scene are taken with different amounts of exposure. [... An] algorithm can fuse the multiple photographs into a single, high dynamic range radiance map whose pixel values are proportional to the true radiance values in the scene." (Debevec and Malik 1997)

This requires us to go through several steps.
First we have to take several photos with different exposure times. A good tripod is invaluable as automatically or manually aligning the images afterwards degrades the quality of the aligned images and therefore the overall accuracy of the conversion algorythm and thus the quality of the final result. A remote control also helps a lot in keeping the camera static. If you don't have a remote use the self-timer as to not move the camera by pressing the shutter button. But the ultimate solution would be to connect your camera to your laptop and use a remote shooting application to adjust all the settings and fire the shots. That way you wouldn't have to touch the camera at all and thus there'd be no chance of accidentally changing the view. But even then you still have to choose a fairly immobile motive as it takes at least a couple of seconds to do all the necessary exposures (preferably 3-5).

> "Each exposure should be separated by one to two stops, and these are ideally set by varying the shutter speed (as opposed to aperture or ISO speed). [... Each] "stop" refers to a doubling (+1 stop) or halving (-1 stop) of the light captured from an exposure." (McHugh)

The process of combining these images into one HDR image works like this:

In Photoshop CS2:
File › Automate › Merge to HDR... – Select Files/Folder – (check **"Attempt to Automatically Align Source Images"** if necessary) – *Ok* – Photoshop computes the camera response curves - A preview window opens and allows you to remove images from

your selection and set the **white point preview**. select a value that looks good - *Ok* – Photoshop creates the HDR file.

Now we've got an HDR image, but our monitor can only display low range, so we don't see a difference to 8 bits per channel images. Simply put, we're only seeing a certain exposure of our high dynamic range image.

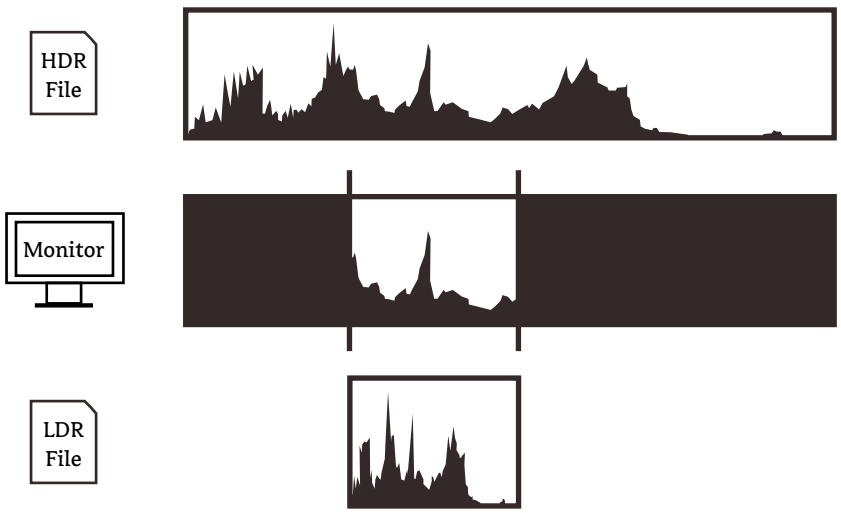


Fig. 96

But the file-size is 4 times larger and most of the PHOTOSHOP-filters and image adjustment tools are greyed out. This may sound like a disappointment at first, but fear not – the additional range is there. We just have to know how to view and how to use it:

Under *View › 32bit Preview Options* we find **Exposure** and **Gamma** sliders that allow us to set which part of the range we want to view. But careful! If you copy and paste from your HDR image into an 8bits/channel image, these settings are not taken into account.
The only safe way to convert to low range is by using the **HDR Conversion dialog**. Choose *Image › Mode › 16Bits/Channel* or *8Bits/Channel*. There you have the same

Exposure and **Gamma** sliders plus a handful of additional options. Now we can select the best setting for our current texturing-project, or even export several different 8bit/channel images at different exposure settings and work with those. “Wait! Isn’t that exactly what we started with?”, one might ask. Yes, we started with differently exposed photos, but now we have greater control without sacrificing image quality! Yes you can make 8bit/channel images lighter and darker, but if you do, you’re losing information as the **tonal range** is clipped or compressed when adjusting **Levels** or changing **Brightness and Contrast**. If you make a black spot lighter it becomes grey. But if you change the Exposure in a good HDR image the black spot just starts to reveal its detailed information.

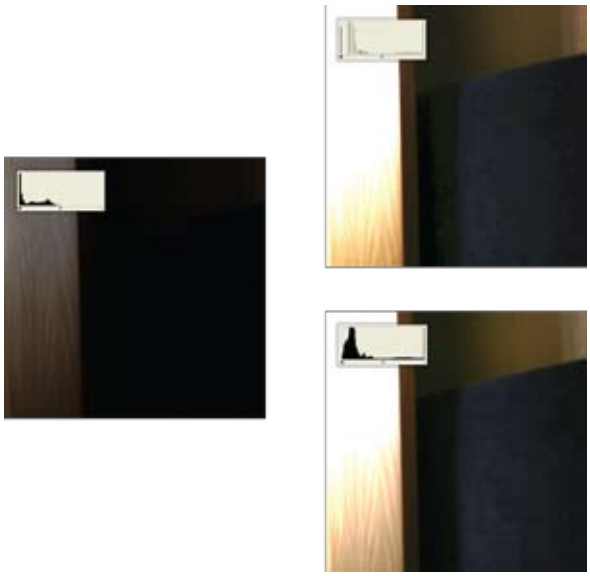


Fig. 97: Left: Original sample. Top-right: LDR. Lower-right: HDR

And that’s not all. Now we can use certain methods to turn our HDR image into an enhanced 8bit/channel image:

Photoshop CS2 – Local Adaption:
Image › Mode › 16 Bits/Channel… or *8 Bits/Channel › Local Adaptation › Adjust Radius/Threshold/Toning Curve and Histogram › Ok*

Photoshop - Photomatrix Tone Mapping Plugin:

Filter › Photomatrix › Tone Mapping...

Fig. 98

Photomatrix Pro:

HDRI › Reveal HDR

This basically holds the same functionality as the plugin described above.

"By hand":

Export several different exposures and compose them manually. (Paint on **layer-**

masks instead of using the eraser tool, so you can make details appear again by switching the brush-color from black to white.)

Another way of generating a tonemapped image is this: Take your picture in RAW, convert the RAW into 2 or more 16bit RGB images with different settings and tone-map those. Aim for the shadows when taking the picture. (cf. HDR Soft) Unlike the multi-exposure-technique, this method can be used to capture moving targets, but it will only enhance the picture a little bit as you can only work with the dynamic range that is in the RAW file.

As the technology is now, working with LDR pictures is good enough. And even in special cases where HDR would give a boost in quality, the additional work needed to create proper HDR images will probably be too much to consider it. But if you do have HDR source material it can give you a bit of extra versatility and quality.
Another use that I see for HDR images are as reference images. Tonemapped HDR images show the entire scene in equal clarity and are therefore ideal for preserving as much information as possible in one shot.

C) SYNTHESIS

There are basically two ways to "synthetically" generate image-material: Using specialized programs that calculate images using certain algorithms (procedural image generation) or by rendering 3D scenes.

Procedural Image Generation

A very basic example is the **Add Noise** filter that comes with Photoshop. It fills a layer with dark and light dots based on several input values. But there is much more to procedural image generation than simple noise. Development is going fast at the moment and there are several interesting programs on the market that are tailored for creating textures for games.

Darkling Simulations' DarkTree is a very versatile procedural shader authoring tool. Shaders created with this software can be rendered in many applications using the free Simbiont plugins. I will use the Simbiont.Max later in this chapter to create a base texture for the "Pillars of Kamula".

Allegorithmic Map|Zone is a photoshop-plugin that lets you combine all sorts of procedural patterns and effects. It's possible to move and erase parts of the procedural patterns and combine different procedural images in many ways.

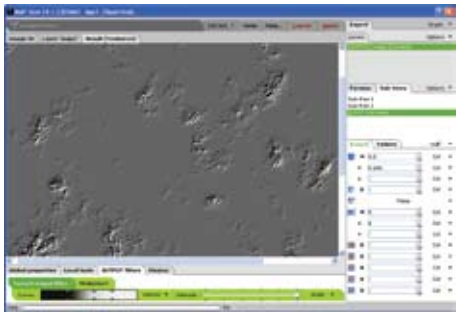


Fig. 99

The same company is currently working on ProFX, an attempt to make procedural textures more appealing to game developers by offering middleware that can be integrated into the game-engine and that generates the final textures on the fly instead

of storing them as pixel-images, thus saving disk-space. With Naked Sky Entertainment's RoboBlitz, a game that uses the Unreal Engine 3, they have a very interesting showcase of their technology.

3D Rendering

Rendering 3D scenes for use in textures is a very efficient way to create repetitive patterns that have a fairly complicated tactile surface. By modeling the surface it is possible to create a **normal map** straight from the model, or to render a **height map** that can then be turned into a **normal map**.

A basic example:
Creating a *chain mail texture* could be done like this:
Model a couple of elements and duplicate them several times to fill a certain area. Add a bit of noise to make the surface less uniform and modify little bits and pieces here and there. Create a **gradient** texture from black to white and apply it to the mesh from bottom to top (This can be done by projecting the texture across the mesh from the side, using **planar mapping**.)
Render the scene from above with an orthographic camera to get the **height map**.
If you now leave the mesh and camera untouched and add some good soft lighting and textures to the object, adjust all the necessary material parameters and render again you have the **diffuse map** aswell.

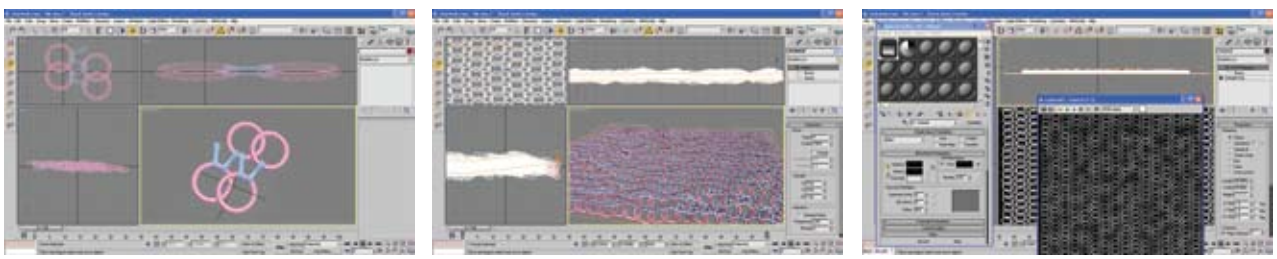


Fig. 100

Outro

It is not only important to have a collection of good source material, it's vitally important, that this collection is well-organized. Efficiency is everything! It must be easy and fast to find the best image for the job. A system of categories and tags might be perfect for this.
Build a solid collection. Take every chance to take a great picture. Every time you modify a texture, add it to your collection. Don't only collect photos, collect extracted elements and brushes as well.
Mark copyrighted images, so you don't forget and accidentally use them without asking permission.

Examples and Techniques

Now that we have a basic pool of material to work from, let's get right to work.
I'll start be describing the typical workflow of texturing an object.

Typical workflow

Think
Gather sourcematerial
Assemble, compose, paint, adjust the individual maps
Test, adjust and rework
Export

Think

Everything starts with a little bit of thinking.
I look at the task description and concept art, if available. If necessary I talk to the responsible people to find out as much as possible about the task.

Is the model already unwrapped?
If not, what kind of texture am I making?
Can I save some texture-space (and therefore texture memory) by using parts of the texture in multiple places, by reusing parts of other textures that are already in use in the same region (and therefore already loaded into memory anyway)? Or by mirroring the texture for the second half of the object if it is symmetrical?

What sort of maps do I need to achieve the desired result?

If I need any map only for a small spot, it would be a big waste of texture space to put it in the same texture and it might be good to make a separate small texture just for that part.

Gather Sourcematerial

After I know what I'm looking for, I start gathering image references and source material. I go through the library of finished assets to see if something similar has already been done for this project, so I can look at how it has been done and maybe even save some time by reusing parts of it.
Then I look through my collection of pictures, texture collections and websites until I have found enough good material to start working.
In more complicated cases I might also take a screenshot of the the model and do a quick concept drawing on top of that to help me find the right look.
If I am to create something that's made of a material that I'm not too familiar with, I try to gather pictures that show it under different lighting conditions, so I can analyze the material properties. Getting a good feel for the material makes it easier to create a good normalmap and to set up the specularity properly. Google-images and Flickr.com are invaluable sources for those pictures.

Assemble, compose, paint and adjust the individual maps

Now the real fun begins.
I usually start with the diffuse map. I build a basis-layer that I can work from and add rough details and colors to get a good feel for the overall look. Once I'm happy with the look, I start working out the details.

With some other materials I find it easier to achieve a good result if I first set up

everything but the diffuse map. I tweak all the maps and values until it has the right feel to it and only then I start thinking about the colors.

Test, Adjust and Rework until it's great

Texturing is an iterative process for me. Every so often I take a step back and try to look at the object with fresh eyes, try to see if it fits in with the rest of the game, try to see which parts need more work and if the overall look is fine.

Neville Page says something that is very true for any kind of art project:

> "I'm sure you've all had this experience when you're doing drawings. There's a point from the moment when you start adding value or paint to maybe about 75% of the way through it, that it looks like you need to quit. and it's a very difficult place to be. And you have to have the foresight which usually comes from experience to know that if you just stick with it it's going to be okay." (Page 2004a, 1 h 7 min)

I also ask colleagues about their first impression, as they might catch little inconsistencies that I have gone blind to during the hour- or even day-long work-session.
For the same reason I like working on at least two tasks at once. This allows me to go and work on something else for a while and then come back with a fresh mind.
It is important to talk to the art director every once in a while, to make sure it's going in the right direction and to get some good feedback.
From there on I work, test, adjust and rework until it looks great and the art director gives his final approval.

Aftermath

Once everything is done I check all the files again, export the final model and place

it at its final destination. I also try to make sure this texture's work-folder is in order, just in case someone needs to change something at a later point.
Finally I take a quick look through all the material to see if any individual pieces could be added to my collection of source material.

Customize your Tools! Automate repetitive Tasks!

Customization

Photoshop doesn't allow full customization of the interface, but being able to arrange and stack all the palettes is usually all you need to do, as the rest of the interface is clean and logical enough to not get in the way.
To get to all the often-needed tools fast, it's a good idea to familiarize oneself with the shortcuts, or customize them to fit the personal needs. Under *Edit › Menus* it is also possible to assign colors to menu items, so highlighting those items that you use most often, might aid as a visual guide through the menu structure.

Some of the tools in Photoshop are incredibly powerful. The brush tool is one of them and allows a lot of customization. Being able to build custom brushes can be a huge time-saver. There are countless options available to define how the brush should behave. Entirely new brush tip shapes can be added by simply drawing a selection around the desired area and clicking *Edit › Define Brush Preset*.

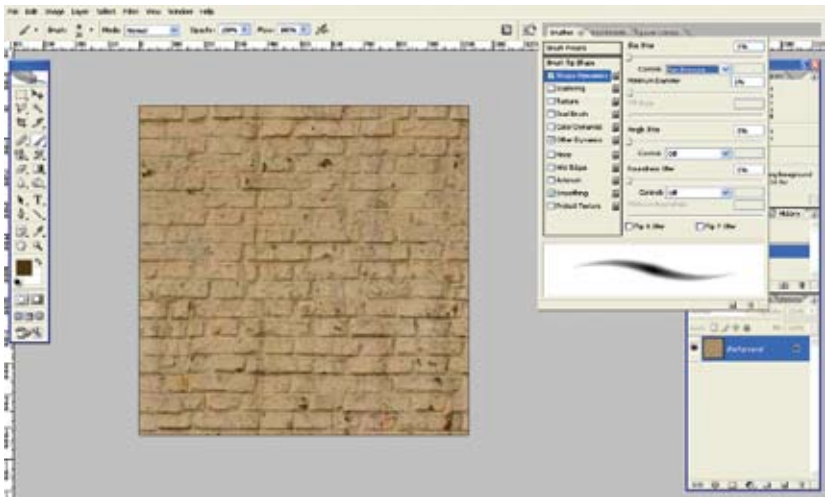


Fig. 101: The brushs palette in Photoshop CS2

Automation

Photoshop Actions

Actions are the **macros** of Photoshop. They allow the recording of several consecutive actions, that can then be replayed whenever needed. They can even be batch-applied to a collection of files using *File › Automate › Batch*.

Photoshop Scripts

Photoshop supports 3 scripting languages: **AppleScript**, **VBScript** and **JavaScript**. I'm writing in **Javascript** because it's the only platform-independent one of those three. Unfortunately the documentation that comes with Photoshop is not very good and partially incomplete. Also not all the features found in the program can be accessed via scripts. But the array of possibilities is still endless, and with a bit of experience it's usually possible to achieve something quickly.

Examples

Automated Export script

I keep all the maps belonging to one texture in the same Photoshop file and make a group for all the layers that make up one map. A custom script then allows me to export all of them at once. The script looks at all the groups in the file to find out which maps to export and then takes the contents of each group, assembles them in the correct way and exports them to tga, even choosing the right filename for me. For example: If it found a diffuse map group, it takes the content and puts it in a new file, if it also found an alpha map group, it puts the content of that into the alpha channel of the file and saves it with the right "_ab.tga"or "_at.tga" extension, depending on the type of alpha.

Since I have the scriptfile in the Photoshop\Presets\Scripts folder, I can easily assign a shortcut to it by going to *Edit › Keyboard Shortcuts*. There it is under Application

MENUS › FILE › SCRIPTS › NAME_OF_SCRIPT. I chose [F12] and overwrote the **Revert File** shortcut that had always scared me a little.

This script can be found on the CD that accompanies this diploma thesis, or it can be downloaded from my website at http://markus.stocktextures.com/thesis

Especially when getting close to the end of a task I need to export frequently to check the result in the previewer. If this script saves me only 1 minute every time I export and on average I use it 4 times a day, that saves me 20 minutes every week. This might not sound much, especially since it took me 2 hours to write it, but give it to a team of 20 people and let them work for 2 years and it saved 640 hours. But I didn't write it to increase my productivity, the main reason why I wrote it is that I wanted to accelerate a part of the process that is boring and takes away my concentration from the actual artwork.

Tile Image – Part 1: Actions
This was the first step towards an easy solution for previewing tileable textures. Here is how you create the action:

Click *Create new action* at the bottom of the **actions palette**. Let's give it a descriptive name, for example "preview 1024x1024 4x4 100%". Now the action is recording.
Press [Ctrl]+[a] to select the entire canvas
Press [Ctrl]+[Shift]+[c] to copy merged layers
Press [Ctrl]+[n] to create a new document and set the **image dimensions** to 4096x4096. click *Ok*.
Press [Ctrl]+[v] to paste
Move the layer to the top-left (snapping helps)
Press [Ctrl]+[j] to duplicate and drag and align it. (Again snapping helps)
Repeat this until the entire canvas is filled. If you want you can merge all the layers.
Now click the *stop recording* button at the bottom of the **actions palette**.

Done.
This action will only work for 1024x1024px files, but you can record more actions for other sizes. Just don't forget to save your actions. You can do this by clicking the little arrow at the top-right of the **actions palette** and choosing *Save Actions...*

These actions are great for speeding up the process, but each action only works for one size. With a script we could have all the necessary parameters calculated and even add some other neat features...

Tile Image – Part 2: Script
I wanted to make this script as versatile as possible. It should let me select how often I want to tile the image, if I want to have it scaled down, and I'd like some lines

Fig. 102: Script Popup

added to show me where the actual tiling occurred.

This script can be found on the CD that accompanies this diploma thesis, or it can be downloaded from my website at http://markus.stocktextures.com/thesis

Creating Tileable Textures - Photoshop

For this example I created a tileable cobblestone texture in Photoshop.
As source material I used 2 photos, taken during late afternoon with no direct sunlight involved – perfect conditions.

First I used the **Lens Correction Filter** (*Filter › Distort › Lens Correction...*) to correct the bit of distortion and adjust the perspective (the pictures were taken from an angle of about 10-15 degrees to the surface normal.)

Since I had good high-resolution source material I wanted to work as big as possible and created a canvas of 2048x2048 px. I inserted the two corrected pictures, looked for a good way to align them, and adjusted the colors a little bit. I wanted one image to fill the top half and the other one the bottom, so I started by working on the seam between the two.
I erased parts of the image on top in such way that the stones flowed together as naturally as possible and then used the **Healing Brush** with **Sample All Layers** activated to heal the seam by taking fitting bits and pieces from other parts of the image and painting them over the gaps.

I noticed that the color was changing too much from one side to the other, so now that I wanted to tile the image, the lighter and darker parts collided. I used an **Adjustment Layer** with a **Posterize** filter to show it clearer. (Having **adjustment layers** on top of the **layer-list** allows for turning them on and off quickly every now and then, which can help to recognize little problems.)
I could have started adjusting the lightness with the **Dodge** and **Burn** tools, but I decided to try the **High Pass** filter first (*Filter › Other › High Pass*) and it worked quite well. A medium setting was enough to make everything look more uniform. But before I used the filter I copied everything into a separate, bigger canvas, so that the **High Pass** filter is uniformly applied to the entire image. (When applied to several

pieces individually, the results will not match 100%, even if the same settings were used for all pieces.) Thereafter I dragged the filtered image back into my main composition. The image lost a bit of contrast and color, but I would add that back in in the end.

But first to the actual tiling:
I have 2 favorite ways of doing the tiling and I choose between them depending on the source material that I have available.

1) I position source-material so there is some good material left outside the canvas on two sides. Then I duplicate the entire layer, delete that part of it that is visible on the canvas (this helps me position it) and move one side inside the canvas and position it at the opposite edge. Then I do this for a second side and use the eraser tool to blend the two new pieces with the base-layer.

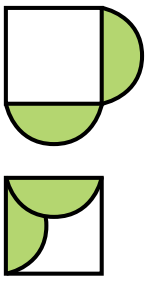


Fig. 103

2) I use the **Offset** filter (*Filter › Others › Offset*) with **Wrap Around** activated, so the seams of the texture move to the center of the canvas and use other parts of the source material to cover them up.

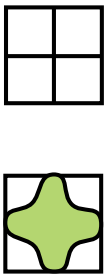


Fig. 104

For this texture I used the first approach. As soon as the basic tiling was done I used my tiling script to see how it looked tiled. I tried to identify the spots where the tiling becomes most obvious and worked over them. This is an iterative process, as the bigger problems are gone, lesser problems spring forward. It can take a while to make a texture look even enough so the tiling doesn't become apparent easily.

While it's preferable to make the result as good as possible, ultimately the porpuse of the texture determines how much time should be spent polishing.

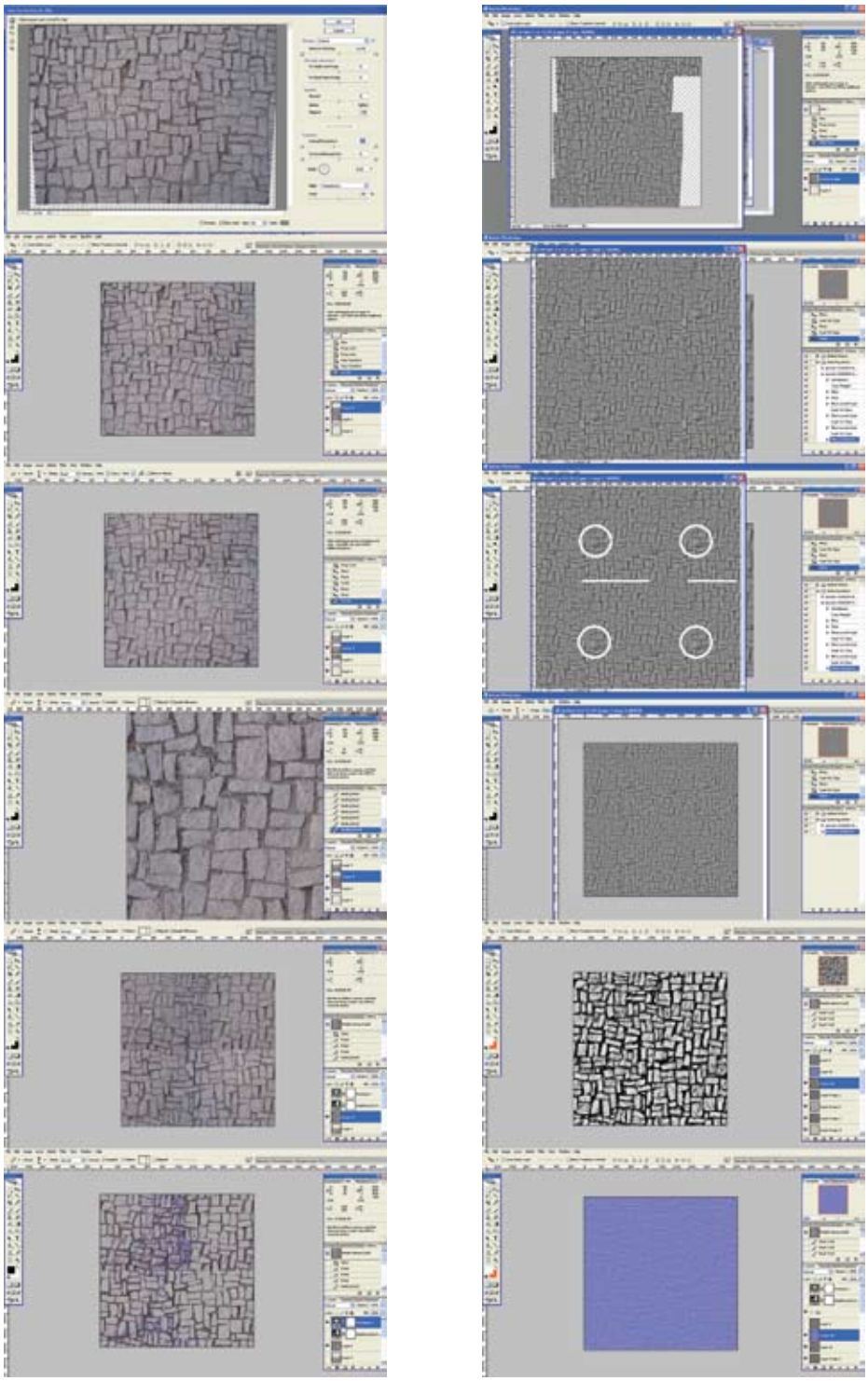
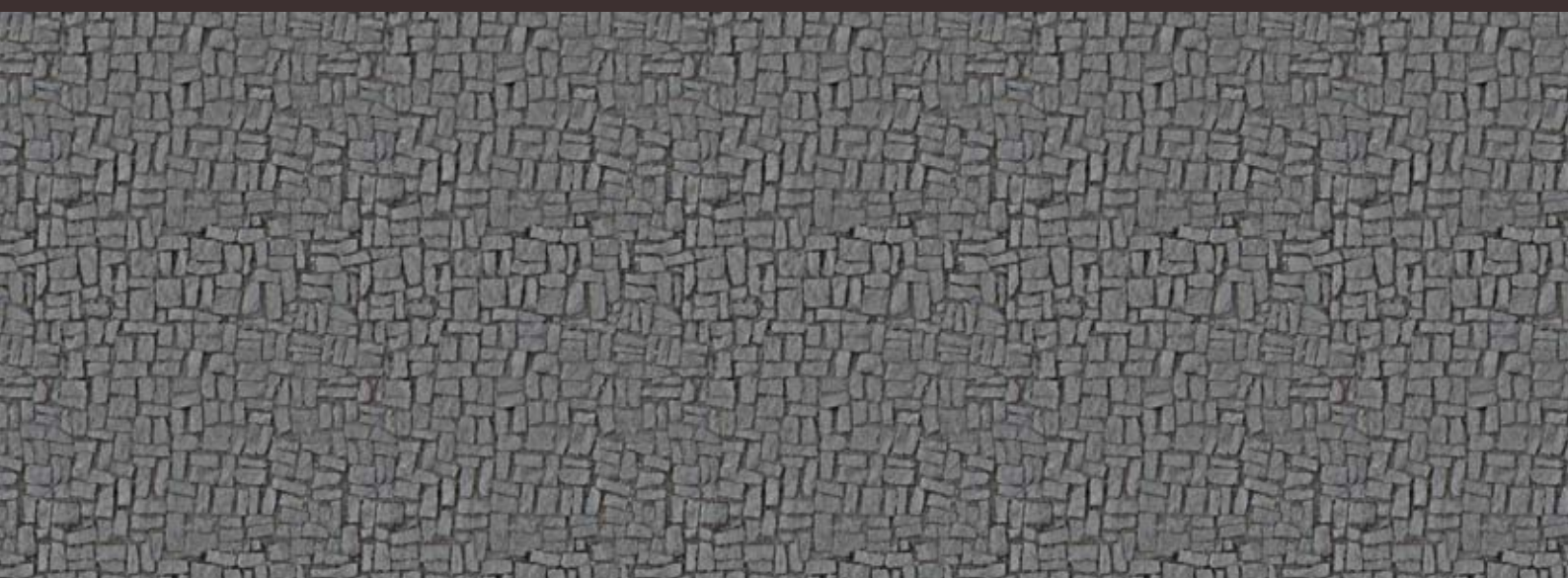


Fig. 105

Fig. 106: Finished diffuse map, normal map and specular map.

Fig. 107: Finished maps on a sphere.



Creating Tileable Textures - Photoshop and ZBrush

When creating tileable textures we can only go one of two ways: we can try to make it as interesting as possible and add specific details that make it look more natural, or we can try to make it tile as easily and good as possible and make it look even and generic.

While the last example was a very generic-looking one, I wanted to make this one interesting. I chose source material that was quite tricky, but in a real production environment we often have to make something out of source material that is not perfect and most of the time it's possible to use a couple of tricks to make the result look good anyway. The important thing is to know what you can and can't do with certain materials and to know when something is just not going to work.

The source material I chose was this photo:

Fig. 108

A beautiful wall, but not a good picture to use as source material for a texture. The lighting conditions are very distinct, there are hard shadows and there's not too much material there to be used (the upper left part is made unusable by the shadows and the right side has a big cable running through it.)
However, if the final texture is only used in a spot where the lighting conditions

match the situation captured in the photo it will do a good job. But it won't be possible to use it anywhere else. (I'll demonstrate why a little further down.)

I started by make a new file: *File › New*. I dragged in the source image and named the layer "Source". I adjusted size and position so that there was some usable material left outside the canvas. (Depending on the nature of the image 10-50% of the canvas-size will usually be enough. It's not necessary to leave space on all 4 sides, as we're only going to use 2.)
Now I duplicated the source layer by selecting it and pressing `[Ctrl]+[j]` (Usually I press `[Ctrl]+[d]` beforehand to make sure I have no selection, as in that case only the selected region would have been duplicated.)
Then I pressed `[Ctrl]+[a]` to select the whole canvas and then `[Alt]+[Backspace]` to fill the selected part with the foreground-color (in this case the default black). I did this to make it easy to see where I needed to tile the image. I took the layer and dragged it to one side holding `[Shift]`, so its vertical position didn't change.
I zoomed in and adjust it using the cursor-keys until the black was completely hidden outside the canvas.

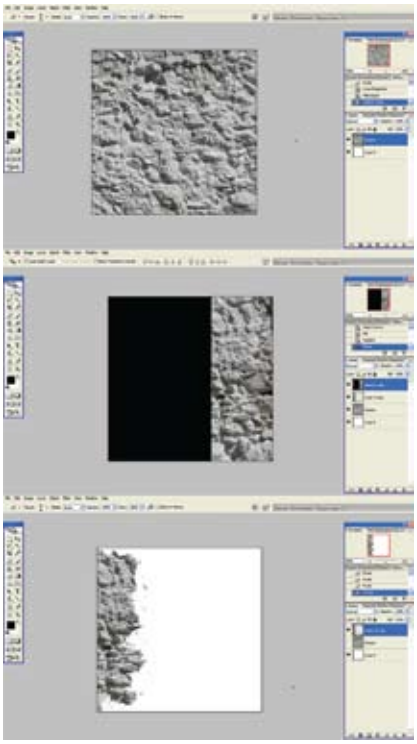


Fig. 109

Now all I had to do was erase the right side of the new layer and make sure the 3 other edges were clear, so the one side tiled perfectly and the other ones weren't changed.
Then I did the same thing for the vertical tiling with another duplicate of the base layer and done was the first iteration of this diffuse map.
I used my tiling script to get a first preview and started correcting those spots where the tiling became most obvious. After a few changes the

diffuse map was done.

In this special case making a good normal map was really hard. A very plastic normal map would have hurt the texture much more than it could have aided it. The additional plasticity that comes with the normal map would have worked against the quite distinct lighting situation in the diffuse map. Lit from the right direction the shadows in the diffuse map would have been amplified, lit from the opposite direction the entire illusion of depth would have been destroyed.

Fig. 110

I decided to create a very subtle normal map in ZBrush.
Even tough it's easier to paint deformation in ZBrush than to hand-paint a height-map in Photoshop, it's still not easy to get a good result.
But first I needed something to work on in ZBrush, so I used modo to create an object like this:

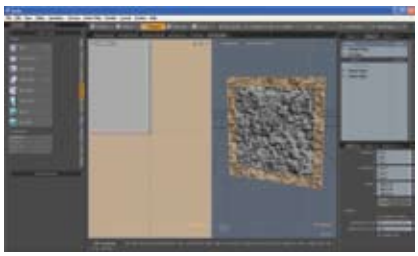


Fig. 111

In the center there is space for the entire texture and then there is a bit of space where it starts to tile.
To make everything go smoothly in ZBrush I had to do a tiny bit of uv magic: I unwrapped the object so that the central part filled the entire **uv-space** from 0 to 1 perfectly while its outer border was just a tiny little bit on the inside of the **uv-space** (I zoomed in very far to adjust this perfectly) then I selected the outer section where I wanted the tiling, tore it off and scaled it up just a little tiny bit so it started just on the outside of the **uv-space**.
By making the uv layout like this I made sure the two parts automatically became seperate **uv groups** in ZBrush and I could hide the outer part (*Tool › Polygroups › UV Groups* – then `[Shift]`+`[Ctrl]`+LMB on one of the groups to hide everything else) before I opened the ZMapper Plugin to create the normal map.

But first came the difficult part.
It was easy to get a result that looked fairly alright in ZBrush, but this impression was misleading. I had to change the light and rotate the object frequently to check how the shapes were coming along. In addition to that I turned off the diffuse map regularly and zoomed out all the way to get a good look at the entire object. After a bit of work I had a moderately satisfying result and used ZMapper to generate the normal map. After the normal map was saved I just had to adjust the edges a little bit to make it perfectly tileable and match the diffuse map.

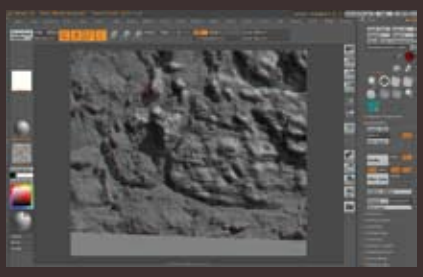


Fig. 112

Creating Tileable Textures - Imagesynth

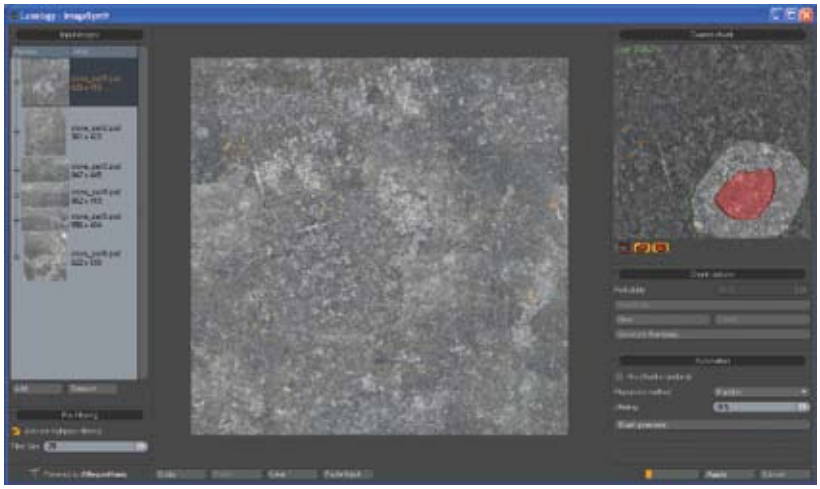


Fig. 113

The Luxology Imagesynth plugin for Photoshop is a tool for creating large tileable textures out of smaller source images.

First I have to prepare my source material, as imageSynth only accepts pictures that are smaller than the size of the area it is applied to. I take a look at several photos I took and save the best pieces as separate files.

Then I create the main document (1024x1024 px) and open the Plugin: *Filter › Luxology › imageSynth*.

I load all the pieces I created and start by placing the first one somewhere on screen. The second one I'll place so it overlaps the first one slightly. If I move it out of the left of side of the screen, the pixels come back in on the right side automatically. Once i click, imageSynth calculates how to best blend the overlapping area. With experience comes a better feeling for what the plugin might do, which makes it easier to plan the next move.

Then I just go on and on. As the canvas fills it becomes more and more important to choose the right source image part for the spot. I start using the **lasso tool** more and more to select only those regions that I really want, setting **outer limits** (green) and **inner limits** (red) before stamping the **chunk** on the canvas. The outer limit specifies

which region *can* be used, the inner limit which region *must* be used. The chunks can also be rotated and scaled, but care must be taken. If scaled up or rotated in non-90 degree steps, the result will look blurry and will stand out.

Fig. 114

Once I'm fairly happy with the result, I apply the filter and look at the result using my *tiling script*, trying to identify problematic areas.
I can rework as much as I want, as long as I don't touch the edges the tilability of the image will be preserved. If I need to work on the edges I just use the offset filter to move the problematic zones to the center of the canvas. I take some more parts of the original source material and rework some areas using the healing brush and the burn and dodge tools until I'm satisfied with the result.

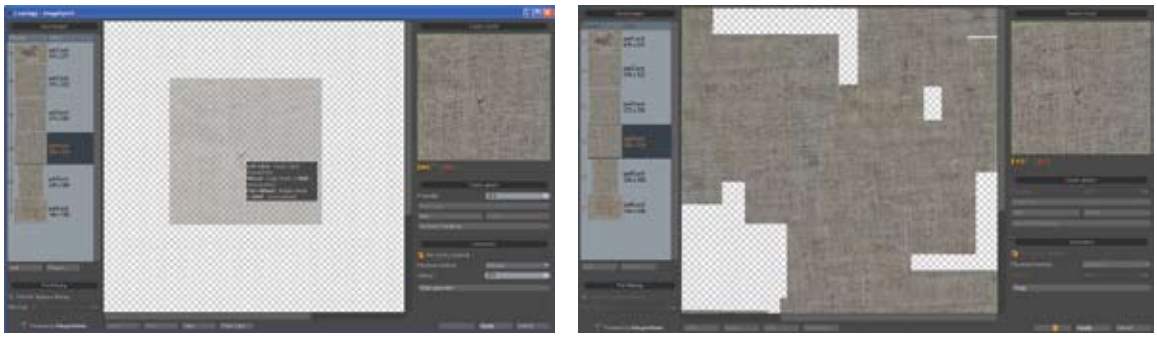


Fig. 115

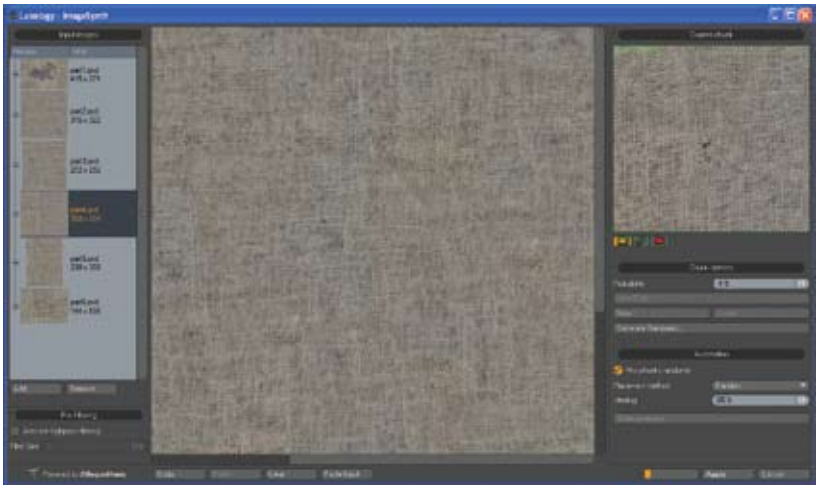


Fig. 116

But that's not all IMAGESYNTH can do. It also has an **automatic mode** where it places out all the chunks either randomly or semi-regularly.
Again I load in the source-material I prepared and split down individual images into smaller chunks using the rectangular or lasso selection tool. I select *Mix chunks randomly* and adjust the **Probability** value for each chunk according to its irregularity. Once I click *Start Process*, imageSynth starts placing the chunks. This takes a short while. When it's done I start reworking areas where elements are obviously repeating and try to better the result as much as possible. Sometimes I try the **automatic mode** several times and then go back to the best result clicking *Undo*, and take it from there.

My general attitude towards imageSynth is this:
It's a really good tool for creating a rough base layer to work on top of, or if I'm in a really big hurry and highest quality is not of utmost important, it gets the job done *fast*. But for creating really good tileable textures, working "manually" is still the better option for me. This is why:

It takes less sourcematerial
Getting good results in ImageSynth requires quite a bit of overlap, so effectively some space is lost with every chunk used. Wasting space means I either have to work smaller or get more good source material.

No repetitions in a single tile
ImageSynth is just not tailored to this need. Its way of working calls for small parts that repeat. With either a lot of source material or a lot of rotating and mixing chunks, it's possible to create a unique surface, but it takes quite a bit of care and consideration to get it done. When working "manually" I already choose the entire setup in such way that there are no repetitions in one tile. Furthermore the mixing of many different chunks may create a unnaturally "patched" look, and it is quite hard to get rid of that.
A possible solution to this would be to choose the source material wisely beforehand: Make one base chunk that is a bit smaller than the canvas and 2+ chunks that will cover up the seams. But if I already have those chunks, I'd rather just do it in Photoshop. It might be a little slower since I have to take care of the tiling manually (or via script), but:

I have more control
I can use all the tools, cut, paste, use the healing brush, adjust levels of parts before merging them down, paint with brushes, erase and more. In ImageSynth I can only trust that the algorithm does what I want. Of course I can rework it after the filter is applied, but then I don't save much time and could have worked with greater control from the start.

With the right source material ImageSynth does a good job, but manual labor by someone with a good eye for texturing will always yield higher quality in the end. Yet again it's a question of how much time is available and what level of quality is required...

Creating tileable textures - Paint from Scratch - Cobblestone

In some cases it is the best approach to start painting the texture from scratch and to add photographic elements, patterns and filters as the work progresses. When trying to create photorealistic textures I always find it helpful to at least overlay some parts of photos and use custom brushes to help a more realistic look without the tedious work of hand-painting every little detail.

The requirements for this example were to create a cobblestone texture to be used all over the city of Tortage.

I started by painting the basic layout of the stones on a new layer. Then I created a basic stone-texture that I put in the background, created a group with the stone-layout as layer-mask and started adding different stone textures to create the look of the spaces in between stones. I added other textures on top of everything to tie the image together. I used a very faint inner shadow layer style to bring the stones out a bit. After some experimentation with layer modes and some adjustments of all the values I had a basic look that I was happy with. From there on I added pits and pieces of dirt and grass and burnt and dodged to roughen up the fairly smooth look a little, but tried to keep it as easily tileable as possible. The result was a texture with very uniformly distributed density and little flashy detail, that was tileable across big spaces. This technique has the advantage that it's very easy to create variations once the groundwork is done.

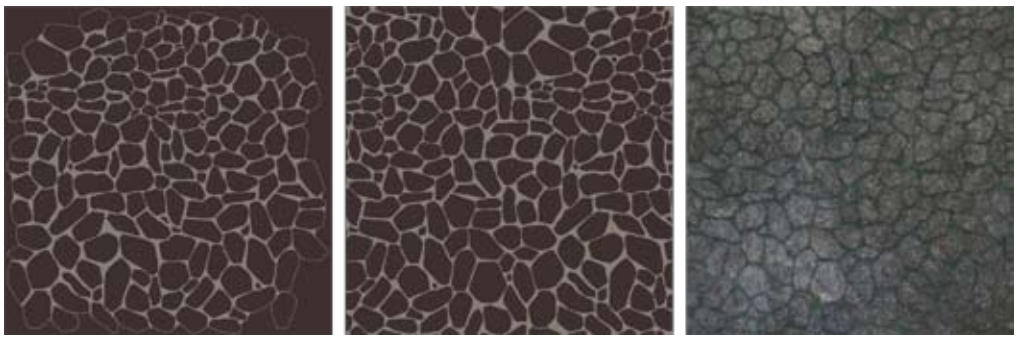


Fig. 117

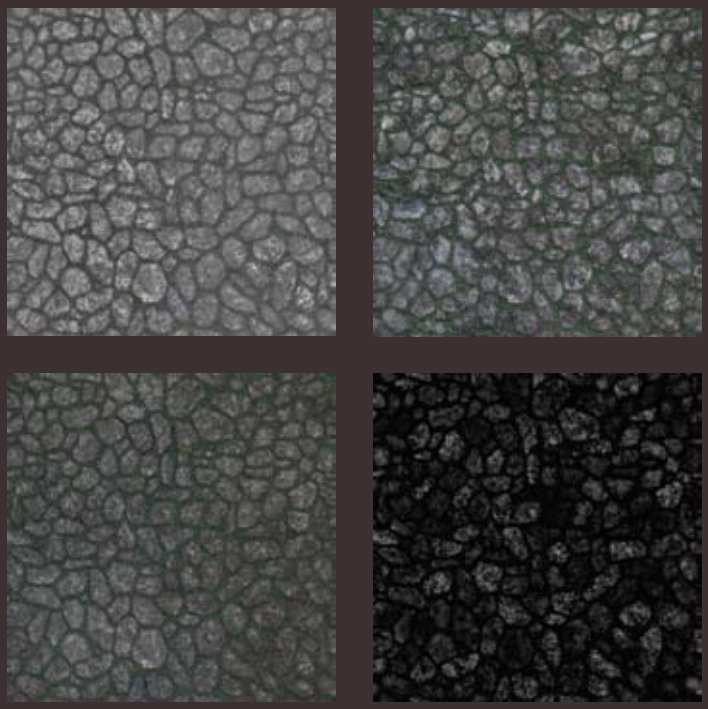
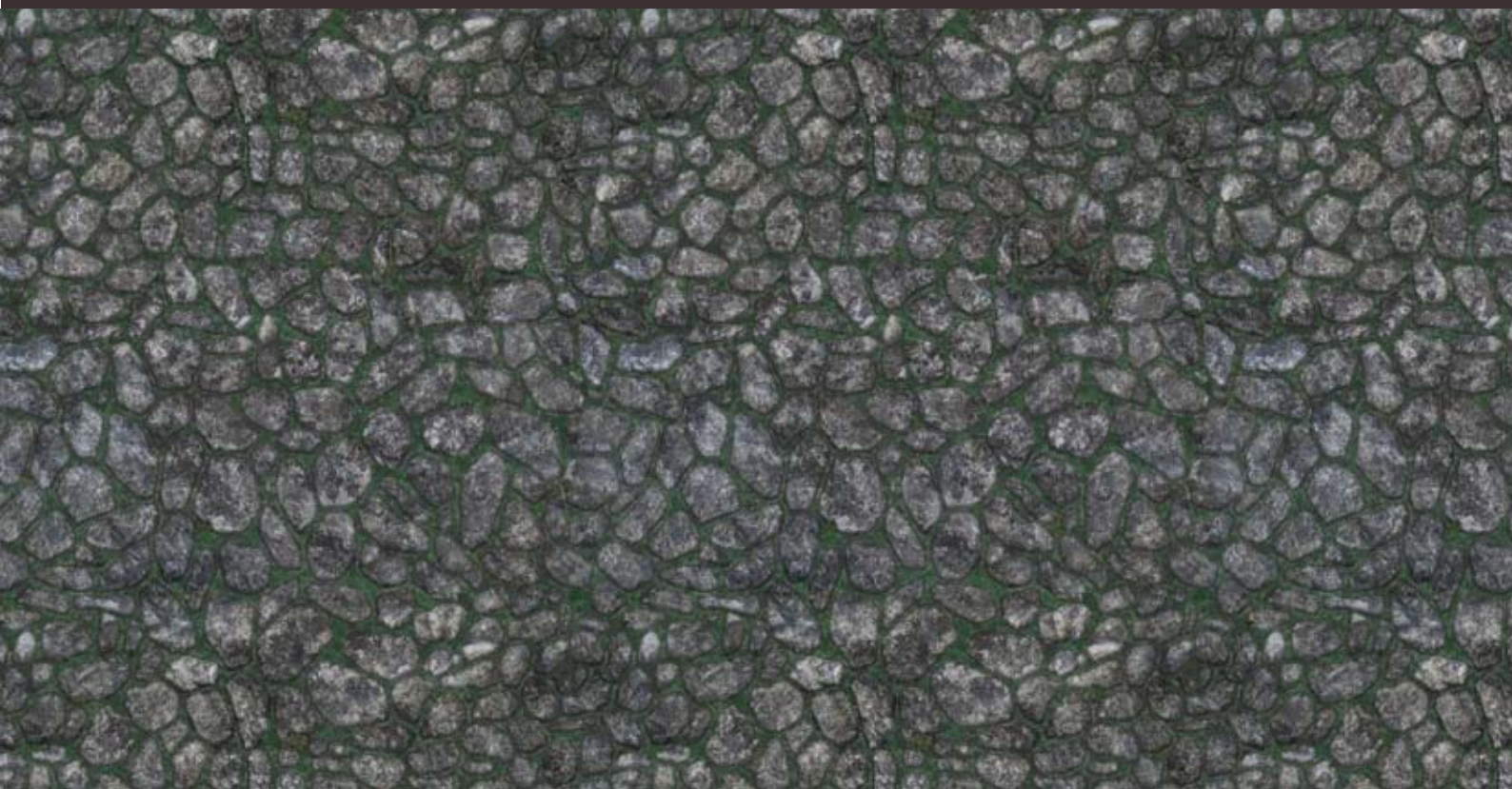


Fig. 118



Torn off hand - Working with the Healing Brush

I found this technique especially useful when working with skin.
I had a lot of great sourcematerial from 3d.sk and took the back of a naked lady as the base layer. I looked at several photos of hands, selected individual pieces, scaled and rotated them to the right size and orientation, and then used the healing brush with sample all layers activated to paint all the details onto the base-skin-layer.

After the basic hand was done, I created several custom brushes with splatters of blood and stamped them onto a separate layer. I used a simple brush to connect the splatters to the edge of the uv-patches and brought the texture into Bodypaint 3d to fix the seams.

I had already create the normal-map when modeling the hand, so all that was left to do was the specularity map. Since I had the blood and the skin on separate layers it was easy to copy and adjust them to work as a specularity map.

After a little bit of tweaking values in the previewer, the hand was done and ready for export.

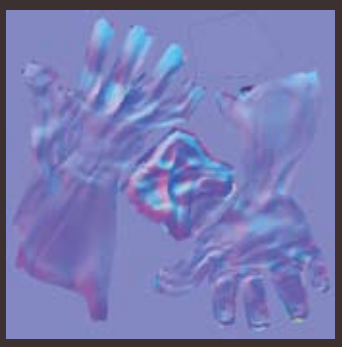
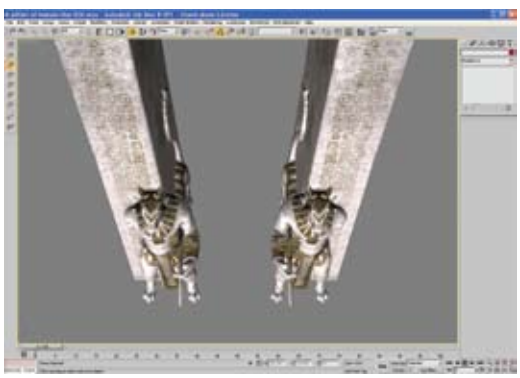


Fig. 119

The Pillars of Kamula

Two versions of the pillars were required: One ruined, to be placed in the middle of a desert and one fairly new, to be placed inside the ghost palace.

I decided to texture the *new* version first. I had the rough normal-map and the finished low-poly-model and decided to do the base-layer in 3ds max with the Simbiont.max plugin to easily get a good bit of dust in just the right places. I used the **Render to Texture** dialog to create this base image. (Note: Even if everything is set up correctly 3ds max shows the result in the **render-window** with lighting applied to it. Clicking the save-button in this dialog will save this wrong version. But is the dialog set to automatically save a file, this file is correct - even if the preview is wrong.)
From there on I started adding little details, gold-paint, dirt and glyphs, which I also added to the normal-map and specular-map at the same time. I frequently exported the three maps and checked the results in the previewer.
In the very end I decided to bake an **ambient occlusion pass** in modo. I imported the model and quickly set up a lighting situation that was similar to the one found at the spot where the model would stand in the game. I set the **render output settings** from **shader tree** to **ambient occlusion**, adjusted the **global illumination settings** and clicked *Render › Bake* to start rendering an ambient occlusion map with the currently selected uv layout. This map I put on top of the diffuse map in Photoshop to add a little bit more realism to the final result.



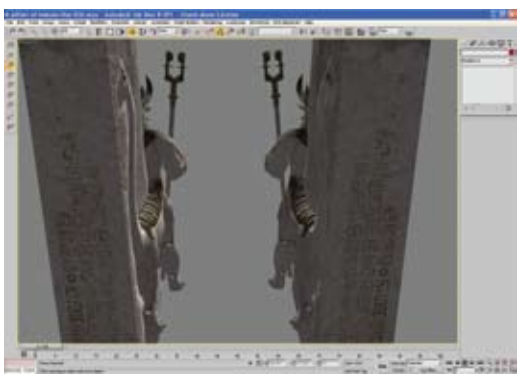


Fig. 120

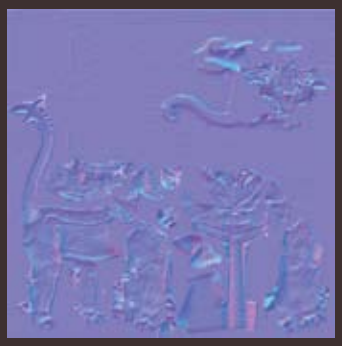
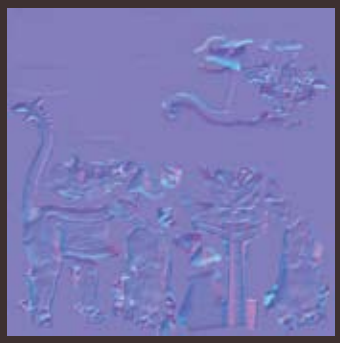


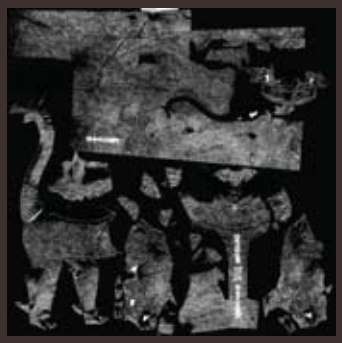
Fig. 121

For the *ruined* version I had planned to go back to my original high-poly model, make it look ruined and unwrap it anew, but due to time constraints I decided to go a quicker way: I would just add the backside and top to the unruined version, break off a couple of pieces here and there and texture that. In the uv layout I put the additional back-wall on top of some other pieces, which made texturing those parts a bit of a challenge, but I found a promising solution and went on to start the texture.

As discussed with the art director, it was my plan to suggest that these pillars were part of that ancient evil place that the Lemurians built on top of. The pillars have been here long before the palace and they remain after the palace has long vanished. After the player has seen the other old, dark pieces in the palace and in other places of the region he might just catch the drift, that the remains of a powerful ancient world are still down there buried under this very desert, maybe stretching out across the entire region. Long lost, but you never know what ancient evil still lurks down there and when and how it could break loose...

Back in 3ds max I again started the Simbiant.max plugin and built a base texture for the ruined version. I used similar settings, but this time with a dark rock-texture and sand from that area to get an old, dusty look. I reworked the result a little bit and added more details here and there. The specular map was especially important for a dark and dusty stone like that, so I took my time tweaking the map and adjusting all the **specularity** settings. In the end I added a very faint **translucency**-effect to make it look just a little bit weird.





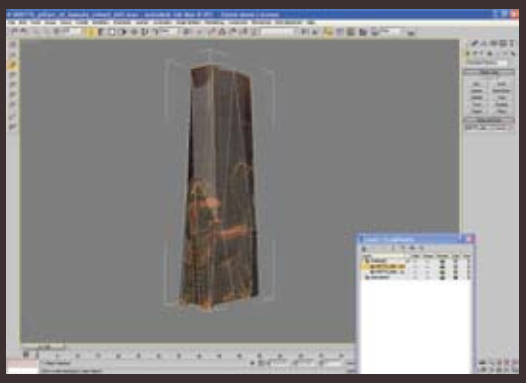


Fig. 122

Lemurian Pillar

My goal with this pillar was to save texture space by making some areas tileable. The pillar has 10 sides and I created individual textures for 4 of those, whereas I split them into 2 groups of 2 sides that were tileable with each other. I could thus alternate these pieces randomly to give each pillar a little individual touch. (For example: the sides of one pillar could be arranged in the pattern "ABABA", the sides of another one in "ABBAB", etc.)

Initially I tried to create the entire normal map in Photoshop, but however hard I tried to create a soft height map for the upper part of the pillar, it just didn't look as good as I wanted it to look. So I took just that part of the model into ZBrush and created the normal map there.

I spent a long time tweaking the look of the stone-texture and the mosaic, and I'm quite happy with the result. I used about 20 different parts of wall-photos to construct a natural-looking base for the texture and spent a good while adjusting color, highlights and shadows.

One little thing that I'm really happy about is how plastic the gems on the lower part look, even though their plasticity is solemnly based on the normal map.

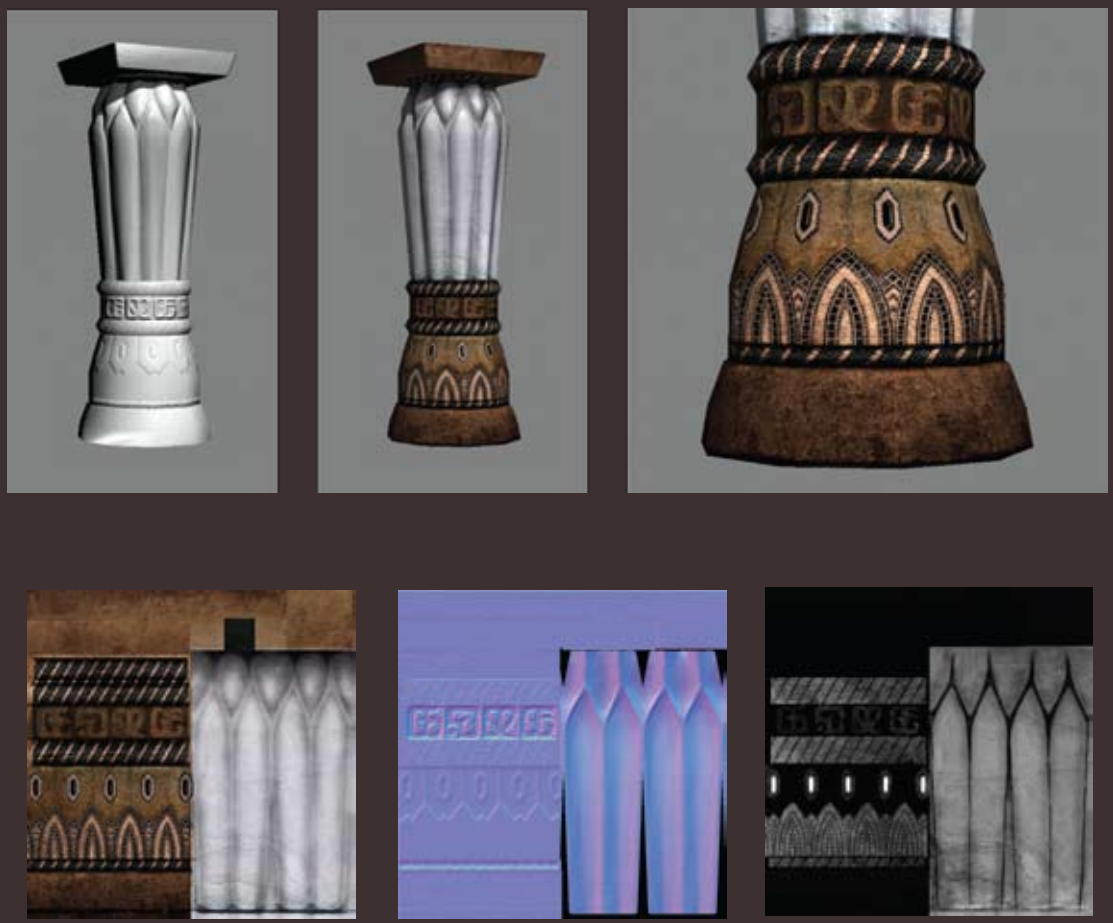


Fig. 123

The Palace of Kamula

The main challenge resulted from the huge size of the palace. I wanted the textures to look good from a great distance away, but at the same time I wanted them to be high-res enough so the player could walk up close to a wall without seeing only big pixels.

I decided to go as big as possible and compensate by reusing many parts. The solution I came up with was a big and fairly loose texture atlas plus smaller individual textures where needed.

Initially I had planned to make it look like it was built out of red rocks and I already had a rough version of one building ready when discussions with the art director and the quest designer lead to a change of plans: We decided to paint the entire palace in white with golden ornaments to underline that the player has traveled to a time where the palace was still very new and also to make the contrast to all the "normal" buildings in the world even bigger.

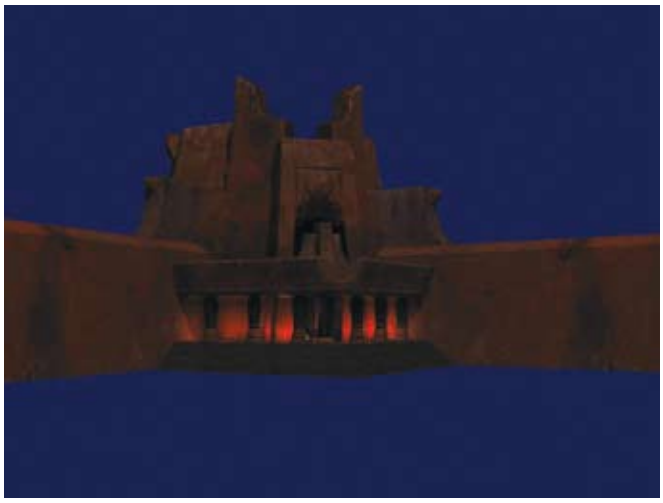


Fig. 124

I decided it would be best to start with the look of the walls and then base everything else on that. I did some research and looked through quite a few pictures online and found a few good ones of how I wanted the walls to look. I didn't have any good high resolution photos of white walls that looked primitive and rough enough to be used, so I had to construct my own. I built a base layer from several different images and added more and more elements to that. Overlaying some quite rough paint on top of everything helped the overall impression a lot. I ended up using more than 50 different pieces of photos from my collection and from online archives. Balancing all those images to get an even look was not an easy thing to do. With lots of adjusting, color-correction, blending and painting I finally managed to build this high-resolution texture of a roughly painted white wall:

Fig. 125

From there on I created a base set of ornaments and worked on the look of the gold so everything would look consistent. I tried all of this on one building and once I was happy with the look, I went on to the other buildings.








Fig. 126: Textures

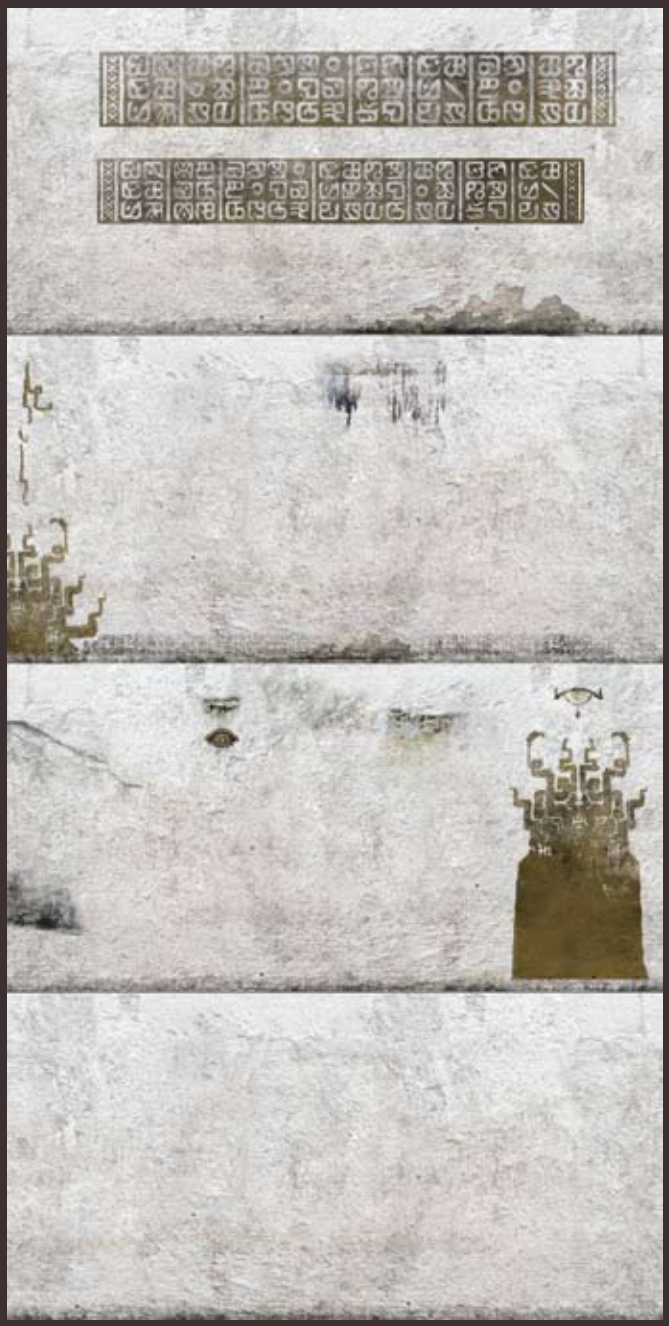


Fig. 127: Big, loose texture atlas

Fig. 128: The main building

Fig. 129: The entrance building

Fig. 130: The side structure

All these are screenshots from the previewer without shadows, fog, final lighting and effects. Look for the entrance to the final palace in Khopshef Province in “Age of Conan” (http://www.ageofconan.com/) or on the website to this diploma thesis (http://markus.stocktextures.com/thesis/)

References

Fleming, Bill (1999). *3D Modeling and Surfacing*. San Diego: Academic Press

Franson, David (2003). *2D Artwork and 3D Modeling for Game Artists.* Cincinnati: Premier Press

Franson, David (2004). *The Dark Side of Game Texturing.* Boston: Premier Press

Gantzler, Todd (2005). *Game Development Essentials: Video Game Art*. New York: Thomson Delmar Learning

Loomis, Andrew (1956). *Drawing the Head and Hands*. New York: The Viking Press

Mikunda, Christian (2002). *Kino spüren. Strategien der emotionalen Filmgestaltung.* Wien: Facultas Universitätsverlag.

O'Neill, Rory and Muir, Eden (1998). *Web developer.com guide to creating 3D worlds.* New York: Wiley Computer Publishing

Petrasch, Thomas and Zinke, Joachimg (2003). *Einführung in die Videofilmproduktion.* München Wien: Carl Hanser Verlag

Russo, Mario (2006). *Polygonal Modeling: Basic and Advanced Techniques (Worldwide Game and Graphics Library)*, Plano: Wordware Publishing

Vitruvius (1960). *The Ten Books on Architecture* [probably written between 27 and 23 BC]. Translated by Morgan, Morris H. Mineola: Dover

DVD

Church, Ryan (2004). *The Techniques of Ryan Church Volume 3: Rendering Hi-Tech Architecture* [DVD]. Gnomon Workshop and Design Studio Press

Church, Ryan (2004). *The Techniques of Ryan Church Volume 4: Rendering Low-Tech Architecture* [DVD]. Gnomon Workshop and Design Studio Press

Huante, Carlos (2004). *The Techniques of Carlos Huante Volume 1: Creature Sketching and Design* [DVD]. Gnomon Workshop and Design Studio Press

Huante, Carlos (2004). *The Techniques of Carlos Huante Volume 2: Digital Creature Painting* [DVD]. Gnomon Workshop and Design Studio Press

Page, Neville (2004). *The Techniques of Neville Page Volume 1: Character Design – Fantasy Wildebeest* [DVD]. Gnomon Workshop and Design Studio Press

Page, Neville (2004). *The Techniques of Neville Page Volume 2: Digital Painting – Fantasy Wildebeest* [DVD]. Gnomon Workshop and Design Studio Press

Robertson, Scott (2005). *The Techniques of Scott Robertson Creating Unique Environments* [DVD]. Gnomon Workshop and Design Studio Press

WWW

"2 pt Perspective Tutorial Part 2 - Mechanical Drawing Perspective Grid Techniques" http://www.khulsey.com/perspective_2pt.html [2006, Sept. 23]

Ahearn, Luke (2006). "3D Game Textures: Create Professional Game Art Using Photoshop".

http://www.gamasutra.com/features/20060622/ahearn_01.shtml [2006, Sept. 23]

Allegorithmic (2006) “Naked Sky Entertainment’s RoboBlitz”. http://allegorithmic.com/v2/ProFX_roboblitz.htm [2006, Sept. 23]

“ATMOSPHERIC or AERIAL PERSPECTIVE”, http://studiochalkboard.evansville.edu/ap-aerial.html [2006, Sept. 23]

Atkins, Robert M. (2004). “RAW, JPEG and TIFF”. http://www.photo.net/learn/raw/ [2006, Sept. 23]

Birn, Jeremy (2002). “Color Bleeding”. http://www.3drender.com/glossary/color-bleeding.htm [2006, Sept. 23]

Bockaert, Vincent. “Bits”. http://www.dpreview.com/learn/?/key=bits [2006, Sept. 23]

Bui Tuong Phong (1975). “Illumination for Computer Generated Pictures”. *Communications of the ACM*, 18.6: 311-317. available at: http://portal.acm.org/citation.cfm?id=360839&dl=ACM&coll=portal&CFID=11111111&CFTOKEN=2222222) [2006, Sept. 23]

Clark, Ryan. “Normal Map Photography”. http://66.70.170.53/Ryan/nrmphoto/nrmphoto.html [2006, Sept. 23]

Cloward, Ben. “RESOURCES - TUTORIALS - CREATING NORMAL MAPS”. http://www.bencloward.com/tutorials_normal_maps1.shtml [2006, Sept. 23]

Cole, Matthew (2005). “A Tedious Explanation of the f/stop”. http://www.uscoles.com/fstop.htm [2006, Sept. 23]

Debevec, Paul E. and Malik, Jitendra (1997). “Recovering High Dynamic Range Radiance Maps from Photographs”, Siggraph 1997 - http://www.debevec.org/Research/HDR/debevec-siggraph97.pdf [2006, Sept. 23]

Engel, Wolfgang (2003). “Implementing Lighting Models With HLSL”. http://www.gamasutra.com/features/20030418/engel_02.shtml – free account required [2006, Sept. 23]

Harris, Tom. “How Cameras Work”. http://science.howstuffworks.com/camera1.htm [2006, Sept. 23]

HDR Soft (2005). “Dynamic Range Increase Examples“. http://www.hdrsoft.com/examples.html [2006, Sept. 23]

Heidrich, Wolfgang and Seidl, Hans-Peter (1999). “Realistic, Hardware-accelerated Shading and Lighting”, Siggraph 1999 - http://www.cs.ubc.ca/~heidrich/Papers/Siggraph.99.pdf [2006, Sept. 23]

Larmann, Ralph M. “THE PERSPECTIVE OF SHADOWS”. http://studiochalkboard.evansville.edu/lp-shadow.html [2006, Sept. 23]

Lebedev, Artemy (2001). “§ 70. Screen resolution. And a little about the origin of 72 pixels per inch”. http://www.artlebedev.com/mandership/70/ [2006, Sept. 23]

Luxology (2006). “Real-world Rendering Techniques” [Quicktime Video] http://content6.luxology.com/event/2006/SIGGRAPH/Luxology_Greg_SIGGRAPH2006.zip [2006, Sept. 23]

McHugh, Sean. “HDR: HIGH DYNAMIC RANGE PHOTOGRAPHY”. http://www.cam-

bridgeincolour.com/tutorials/high-dynamic-range.htm [2006, Sept. 23]

Meyer, Jon (2004). "The Future of Digital Imaging - High Dynamic Range Photography". http://www.cybergrain.com/tech/hdr/ [2006, Sept. 23]

"Physics around us". http://fizyka.phys.put.poznan.pl/~pieransk/Physics%20Around%20Us/Physics%20around%20us.html [2006, Sept. 23]

TAKAHEI, Toshiyuki, INAMI, Masahiko, KAWAKAMI, Naoki, YANAGIDA, Yasuyuki, MAEDA, Taro and TACHI, Susumu (2003) "Detailed Shape Representation with Parallax Mapping". www.vrsj.org/ic-at/ICAT2003/papers/01205.pdf [2006, Sept. 23]

"THE PERSPECTIVE OF SHADOWS", http://studiochalkboard.evansville.edu/lp-shadow.html [2006, Sept. 23]

Treibergs, Andrejs. "The Geometry of Perspective Drawing on the Computer" http://www.math.utah.edu/~treiberg/Perspect/Perspect.htm [2006, Sept. 23]

"TUTORIAL - Box Modeling and Precision Texturing" http://206.145.80.239/zbc/showthread.php?t=20370 [2006, Sept. 23]

Walker, Chad (2001). "The Basics of Designing and Creating Low Polygon Models" http://www.gamasutra.com/resource_guide/20011119/walker_01.htm [2006, Sept. 23]

"What Is... ISO". http://www.photoxels.com/tutorial_iso.html [2006, Sept. 23]

Wikipedia. "Color". http://en.wikipedia.org/wiki/Color [2006, Sept. 23]

Wikipedia. "Depth of Field". http://en.wikipedia.org/wiki/Depth_of_field [2006,

Sept. 23]

Wikipedia. “Gouraud Shading”. http://de.wikipedia.org/wiki/Gouraud_Shading [2006, Sept. 23]

Wikipedia (german). “Gouraud Shading” http://en.wikipedia.org/wiki/Gouraud_shading [2006, Sept. 23]

Wikipedia. “Normal Mapping”. http://en.wikipedia.org/wiki/Normal_mapping [2006, Sept. 23]

Wikipedia. “Texture”. http://en.wikipedia.org/wiki/Texture [2006, Sept. 23]

Wikipedia. “Umbra”. http://en.wikipedia.org/wiki/Umbra [2006, Sept. 23]

Other

Monolith Productions (2006). “The Environment Art Pipeline in Condemned – Criminal Origins”. at Xfest 2006 Art Contend Track [Powerpoint Presentation]

List of Figures

Fig. 32: Hofer, Markus. *Simple Pillar Concepts* for *Age of Conan (Funcom)*.

Fig. 33-34: Regan, Grant for *Age of Conan (Funcom)*.

Fig. 35: Regan, Grant. *Aquilonian Location Concepts* for *Age of Conan (Funcom)*.

Fig. 36: *Tortage Location Concepts* for *Age of Conan (Funcom)*.

Fig. 37: Screenshot of Zbrush 2. Model by Markus Hofer for *Age of Conan (Funcom)*.

Fig. 38: Screenshot of Modo 202. Model by Markus Hofer.

Fig. 39-40: Hofer, Markus. *Custom Pie Menus*.

Fig. 41: Screenshot of 3ds Max 8. Model by Markus Hofer for *Age of Conan (Funcom)*.

Fig. 42-52: Hofer, Markus. *Illustrations for Modeling Basics*.

Fig. 53-58: Hofer, Markus. *Palace Buildings for Age of Conan (Funcom)*.

Fig. 59-62: Hofer, Markus. *Entrance Building Interior for Age of Conan (Funcom)*.

Fig. 63-67: Hofer, Markus. *The Pillars of Kamula* for *Age of Conan (Funcom)*.

Fig. 68-71: Hofer, Markus. Hedge Maze for *Age of Conan (Funcom)*.

Fig. 72-75: Hofer, Markus. *The Palace Pool and the Eternal Well* for *Age of Conan (Funcom)*.

Fig. 76: Hofer, Markus. *Naked model and textured model - Lemurian Pillar* for *Age of*

Conan (Funcom).

Fig. 77: Hofer, Markus. *The most common parts of a texture*.

Fig. 78-83: Hofer, Markus. *Illustrations for Texturing Basics*.

Fig. 84: Screenshot of modo 202. Model by Markus Hofer.

Fig. 85: Hofer, Markus. *Skin and Textured Model - Artifact* for *Age of Conan (Funcom)*.

Fig. 86: Hofer, Markus. *Texture Atlas and Textured Model - Entrance Building Interior* for *Age of Conan (Funcom)*.

Fig. 87: Hofer Markus. *Tileable Texture*.

Fig. 88: Hofer, Markus. *Compact Camera (Canon Powershot S50) and SLR Camera (Canon EOS 350D)*.

Fig. 89: Hofer, Markus. *Sensor sizes in relation to 35mm film*.

Fig. 90-92: Hofer, Markus. *Illustrations for Taking the Picture*.

Fig. 93-94: Hofer, Markus. *Illustrations for Challenges*.

Fig. 95-98: Hofer, Markus. *Illustrations for High Dynamic Range Images*.

Fig. 99: Screenshot of Map|Zone EX 1.7 Demo

Fig. 100: Hofer, Markus.

Fig. 101: Screenshot of Photoshop CS2. Texture by Markus Hofer.

Fig. 102: Hofer, Markus. *Script Popup*.

Fig. 103-104: Hofer, Markus. *Creating Tileable Textures – Techniques*.

Fig. 105-107: Hofer, Markus. *Tileable Texture*.

Fig. 108-112: Hofer, Markus. *Creating Tileable Textures – Photoshop and ZBrush*.

Fig. 113-116: Hofer, Markus. *Illustrations for Creating Tileable Textures – Imagesynth*.

Fig. 117-118: Hofer, Markus. *Cobblestone Texture for Age of Conan (Funcom)*.

Fig. 119: Hofer, Markus. *Torn off hand* for *Age of Conan (Funcom)*.

Fig. 120-122: Hofer, Markus. *Pillars of Kamula* for *Age of Conan (Funcom)*.

Fig. 123: Hofer, Markus. *Lemurian Pillar* for *Age of Conan (Funcom)*.

Fig. 124-130: Hofer, Markus. *The Palace of Kamula* for *Age of Conan (Funcom)*.