The Making Of "Shadow Of The Colossus"

Translated from: http://www.watch.impress.co.jp/game/docs/20051207/3dwa.htm

With the PS3 being announced for spring 2006, you might think the PS2 is at the point where it's reached it's limit. At this time, the world's attention tends to go to the next generation machine, but this is actually the time where all the technology achieved over the lifetime of the current machine bears fruit, and is the age where the masterpiece games appear.

You could probably say that this winter, where SOTC (Shadow Of The Colossus) appeared, represents such a maturing period for PS2.

Shadow Of The Colossus is a good game in itself, but it is the technology which is operational within the machine, PS2-wise, that gives it the true "next-gen impression".

So now, we have brought together the information from the development of Shadow Of The Colossus, because we had heard discussion on the technology we used and wished to present it.

[Fumito Ueda]

"I'm not sure where I could even start talking about it! Really! (laughing). The work on SOTC accumulated little-by-little, so when it finally came together, it seemed completely natural."

[Hajime Sugiyama]

"I tried hard to make the appearance look natural to the eye. I'm not sure if I was able to achieve it, but it is hard to tell because it is not cartoony. Don't you think? (laughing)"

[Takuya Seki]

"Rather than the programmer saying how about like this, he keeps making the kind of environment which the designer can adjust to whatever their heart's desire."

[Masanobu Tanaka]

"You can spend far too much time making detailed adjustments on a case-by-case basis, just to get the exact right motion on each character."

* The HDR rendering and dynamic exposure / tone mapping algorithm

After starting SOTC, the first impressive visuals you get are probably the view of the magnificent scenery seen outside from inside the sanctuary. It floods the external scenery which is seen from the interior to white, and light overflows through the opening of the columns which support the sanctuary. This is the trendy high dynamic range rendering (HDR - High Dynamic Range).

When you emerge from the dark sanctuary interior to the outside, the outside scene is drawn almost completely white, which settles shortly to it's proper brightness balance. This is called the dynamic tone mapping, which is part of the HDR effect.

SOTC's HDR rendering method is not physically correct, but the effect works well and creates very impressive visual results.

[Screen shot]

The effect in pseudo-HDR rendering is seen everywhere. It means that brightness balance has been centred around the protagonist under the cliff, and so the relatively bright sky tends to become almost white.

HDR rendering and tone mapping is explained in the article about "Half-Life 2: The Lost Coast", but we will give a brief overview of the concept here.

In order to render a HDR image without being restricted by the possible color range of the television display, the HDR rendering indicates a process where tone mapping is used to get the proper brightness balance for the TV display.

Originally, 3D graphics performed on a computer, is something designed to make an image which is close to the actual world. There is a huge variation in brightness in the actual world, from intense brightness down to gloomy darkness, but the human eye and the camera cope by adjusting the exposure, shutter speed of the camera, and the opening of the pupil, etc. Regarding 3D graphics, we process the light intensity of the actual world using a similar large range with high accuracy (HDR rendering), and the simulation of the camera and the eye will happen at a later stage (tone mapping).

It is difficult to perform true HDR rendering for PC GPUs before the DirectX 8 generation which most current consoles use, as these render only to 16 million-color framebuffers containing 8-bit RGB channels. (Technically it is not impossible but it would be too slow to be of any practical use). Because of this, the pseudo-HDR rendering has become popular in game graphics.

With Xbox 360 and PS3, it is possible for each RGB channel to be 16-bit floating point (FP16) , as the hardware was designed from scratch to be able to cope with HDR rendering. But there is no such potential for that on PS2. When you think of the graphics of Shadow Of The Colossus, pseudo-HDR rendering springs to mind.

Speaking of pseudo-HDR rendering, the glow effect where light overflows onto the surrounding areas during the post-processing (the bloom effect) is traditionally a general-purpose effect. Our implementation introduces exposure and simulation of the pupil with the proper brightness, which is not usually done. Furthermore, with SOTC, this is done according to the scene where the player is.

[Pseudo-HDR rendering]

At first, it tends to bright white (left picture), but the real-time effect settles gradually to it's proper brightness (right picture). The eye keeps adjusting.....

* Behind the scenes of pseudo-HDR rendering in Shadow Of The Colossus

> "To tell the truth, it uses the same basic principles as our previous title 'ICO'. When looking at it in a certain way, it would be fair to say it is a development of that."
>
> *- Takuya Seki, SCE lead production department*

With Shadow Of The Colossus, the scene defines boxes (trigger boxes) everywhere on the map, (for this article I've decided to call them scene boxes for convenience), which when the player is inside, it dictates the settings for the effect from that area.

In other words, if you think back to the sanctuary example, "because the inside of the sanctuary is always dark, when you looked at the bright outside from there, it appears saturated", this is because the scene definition instructs it to "do just that".

> "You do the rendering itself normally. Afterwards, depending on the scene effect that has been defined, it just applies the effect where light starts overflowing in the outside scene."
> *- Seki*

The rough algorithm is outlined below:

1. **Rendering the distant view, and save it.**
2. **Foreground rendering.**
3. **Composite the distant view and foreground together.**
   It uses the contents of the Z-buffer like a mask pattern (see image 2).
4. **Again using the Z-buffer, generate a separate image of the mask pattern.**
5. **Reduce this to 64x64 pixels, using a bilinear filter. (This makes it ultra-low resolution.)**
   This gets reused in step (8).
6. **Blend in the previous frame (from step 7) using a percentage specified in the scene box.**
7. **Store the image for use in step (6).**
   It is used for drawing the next frame.
8. **Expand step (5) to a suitable resolution, using the bilinear filter.**
   Thus, it becomes the material for the bloom effect where the distant view starts bleeding out from the shape of the foreground.
9. **Blend (7) and (8) using a ratio specified in the scene box.**

[1. Distant view rendering] **[2. Foreground rendering]** **[3. Blend of distant view and foreground]**

1. Distant view rendering ("SuperLow", described later)

2. Foreground (only the high and medium models)

3. Combination of (1) and (2) using the Z-buffer mask.

[4. Generating the image mask pattern]

Using the same Z-buffer mask, generate the original image for the bloom effect.

[8. Blurring the mask, and blending the previous frame]

Blurring of (4), and composition of the previous frame.

[9. The final image]

9. Final image

Occasionally, when you would move from a bright place to a dark place (and vice versa), it seems that the effect gradually moves to it's proper brightness. How did we do this kind of dynamic tone mapping?

When you move from inside the sanctuary to the the outside, because you have left the scene box in the sanctuary, the effect there stops being applied. This is done as a binary (on/off) decision, but if the scene were to change suddenly, it would look visibly unnatural. So, when the player leaves the box, there is a short interval where it gradually changes into the effect of the next scene box.

I perform the fake tone-mapping effect by changing smoothly from "Inside the sanctuary, outside scene is drawn bright" ==> "Outside the sanctuary, outside scene is drawn normally".

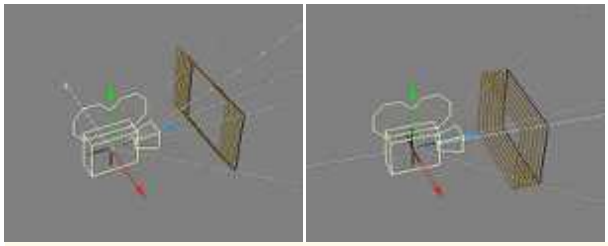
> "The downside to this method of scene boxes, which causes the PS2 to suffer, is that the time taken on the production side to set the boxes up is very high. SOTC has a very large world to manually set all this in."
>
> *- Hajime Sugiyama, SCE lead production department*

* Motion blur - with afterglow effect, and frame rate

With SOTC, when the camera moves, the "motion blur effect" occurs.

The motion blur - there are various techniques for achieving this, but with SOTC, we adjust the current frame after rendering is done, by the velocity of the camera, and then simply accumulate these images. In the PS2, where there is little memory, and it it slow to access, this is a fairly reasonable technique.

[Camera blur conceptual drawing]



By shifting the current frame according to the velocity, it accumulates due to translucency.

Original frame

Blurred result

Furthermore, this camera blur is not applied with respect to the character. If this camera blur was put on the character, the character would become too blurred and vague, which is not a good thing as it hinders gameplay.

But, when you look carefully, in SOTC when the player character runs etc, you can see (in the pictures below) that blur occurs regarding action. This camera blur occurs due to different processing reasons.

This is done by finding the difference between the bone positions in this frame and the previous frame, and forming a polygon from it. It applies the character image of the current frame as a texture for it.

On the occasion where the player has hung onto a colossus, and the colossus performs a large move, the view moves quickly from the sky to the land bewilderingly, but the sky brightness remains strongly visible. When the point of view is moving extremely, a similar effect happens, but this is when the human eye is looking at a bright light, and is known as the afterglow phenomenon. Because of this, the motion blur and the pseudo-HDR effect delays the brightness change, and the image remains. It seems to be a necessary hack to say "If the movement is excessive, keep the camera still". We just have to live with it. This is a unexpected by-product of step (6) in the HDR flow.

This afterglow effect is controlled by a value in every scene box. While playing you may see if you look carefully, that the afterglow does not happen in certain areas.

Well, SOTC is a game whose frame rate can increase and decrease wildly. Actually, we have incorporate the variable frame rate in the design - it increases and decreases with load balancing. Although there are cases when it reaches 60fps, there are also times when it falls to the 15fps range, but the motion blur helps to smooth over this, and the player's sensation of frame rate changes is held down to a minimum.

[Blur]

The blur effect appears even when the character moves quickly.

Adjusting to the movement of the colossus, when the view shakes violently, we get a strong blur effect where the whole view appears off-center. This greatly improves the visual appearance.

By turning the gaze, blur occurs.

* Generating shadows via the self-shadowing stencil shadow volume technique

Explanation of SOTC's shadow generation:

With shadow generation of this kind it is difficult enough to compute the hard-edged shadow (the penumbra), but we manage to generate your own shadow to fall not only on the ground, but also on your own body itself. For example, the shadow of the arm of main character is cast onto his own body. Because most PS2 games make do with casting the silhouette of the character onto the landscape, SOTC's shadows are considerably more immersive.

The shadow generation in SOTC adopts the same stencil shadow volume technique as DOOM3 and other games.

We generate our shadows with the stencil shadow volume technique, which also allows self-shadowing effects. Not many PS2 titles do this. Furthermore, at the time of writing this article, this technique will continue to last for a long time.

With this technique, first, we extrude the outline of the 3D object to form the contour along the direction of the light, forming a contained area (shadow volume) inside. Depth information of the scene from the camera's view is used for each pixel, to judge whether the pixel is inside or outside the shadow volume. The results are kept in a stencil buffer. Finally, using the contents of the stencil buffer, you perform the process that applies "shadow color" to the rendered image.

[Self shadow]


On the right is the visualization of the shadow volume from the same scene. Shadow volume extrusion is done via VU1 of the PS2's CPU.

The quality of the shadow formed by this technique is high, but because more polygons are drawn than are actually visible, and the processing of the shadow volume formation, it becomes difficult when there is a lot to draw. Fortunately, with "Shadow Of The Colossus" the game mostly centers around the hero, the horse and the colossus when it comes to shadows. Seki tells how "It was possible to limit most of what needed processing".

By the way, even in our previous title "ICO", the stencil shadow volume technique of this kind was used, but with that the dynamic real time shadow was applied to just 2 people - ICO and the heroine.

Well, in the case of shadow volume generation, we find the outline of the 3D character from the light source and extrude this to form a shadow domain, but this processing is done on VU1 of the PS2's CPU. However, as the number of outline increases, this can take a very long amount of processing time.

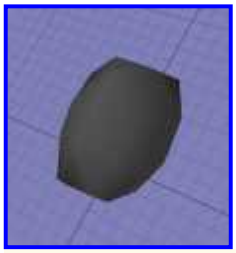
In games such as DOOM3 on the PC, the model used for generating the shadow volume is almost equivalent to the character itself. But with SOTC, in order to speed this up, we made use of a simpler model with much fewer polygons in.

The main character generally consists of 3,000 polygons, but the colossus can be around 18,000 polygons, depending on the type. But the model used for shadow generation will contain a substantially lower amount than this. For example, the simple model seen by the player will probably only use 1/40th of what the original model contained.

When in the game, if you look carefully, the shadow of the colossus appears to be less detailed than the actual model, but it is helped by the fact that the model and it's penumbra are rarely projected in a way such they can be compared, so it looks fine. Perhaps this fine-tuning is the art of a true craftsman?

In addition, the penumbra generated from the stencil buffer is applied using a soft blur to get soft shadows. This is not physically correct as it does not take into account the projection distance of the shadow, but with the resolution of the PS2 it seems to work quite well without looking wrong.

[Shadow model]


The model for shadow generation which is applied to the face of the player character. It deliberately does not cast any shadows onto itself because it would look


The model for shadow generation of the player character's body. The number of polygons has decreased, but the shape itself is accurate enough.

visually
unpleasing.


A comparison image which superimposes the shadow model and the real model on top of each other. The wireframe model is for shadow generation. The shadow model for the face is one big shape for the reason given earlier.

[Self shadow]

The shadow generation model of the colossus when the number of polygons has been reduced to 1/40th, although because of it's immense size, it doesn't look strange.

The image with the stencil shadow technique turned off.

The same image with shadows turned on. The self shadow appears to also cover itself. Furthermore, regarding this shot, the shadows are much deeper.

* Hair growing on the colossus using a fur shader

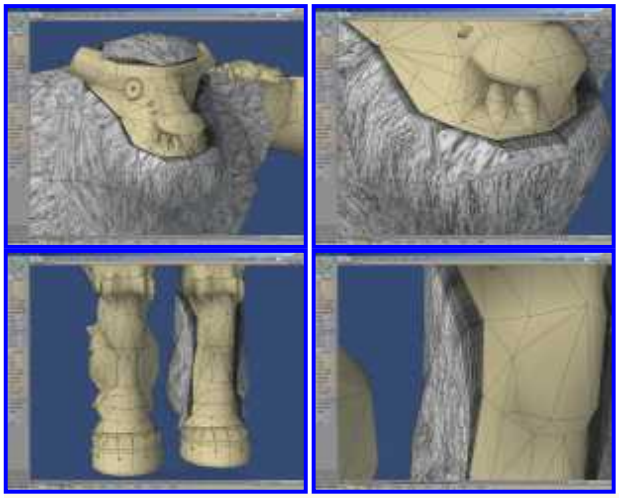
When you play Shadow Of The Colossus, it is the fur on the colossus that gives a striking impression. The 'fur shader' buzzword, when the programmable shader 1.x generation of GPUs appeared, caught on and even Xbox and PC games adopted it.

With SOTC, we implemented this without programmable shaders. During the 1.x generation, a popular technique was to construct fur using a multilayered shell system as the basis.

This rendering is done by constructing parallel layers, from the skin at which the hair grows and in fixed steps outwards. These translucent polygons accumulate using a cross-section texture of the fur, which build up to form the complete section of the hair. It is almost a kind of volume rendering effect.

[Fur]


The soft and fluffy feeling here comes from the fur shader. The waving, flowing and thickness of hair comes from the "hair spike" rising from the skin. It looks real because it is adjusted to be natural.



Development picture. When you add multiple layers to the colossus' skin at a fixed distance, if you zoom in you can see the polygons which make up the cross section fur texture.

Well, with this technique, the more polygons you use the more real the hair looks. But with SOTC, it tends to be limited to 3 - 6 layers due to the processing limits of the PS2.

The multilayer shell construction system does a good job of expressing the fluffy nature of fur, but when to imitate the high density of hair, you need to use a large number of layers. But, if you try to draw too many fur layers, mostly due to the amount of overdraw, this is too much to handle and starts to overload the video memory fill rate, which affects other drawing processes. Therefore in order to get a good balance between visual quality and workload, we add additional polygons of the hair drawn from the side, which we place vertically perpendicular to the skin.

The fur shader's multilayer shell construction is used even for the grass in certain outdoor areas. Try and spot it when you play.

The layered fur shells are easily visible at a sharp viewing angle, and the extra hair fins are equally visible at the opposite angle. But by combining these two techniques, we create a very natural feeling of hair for the colossus.

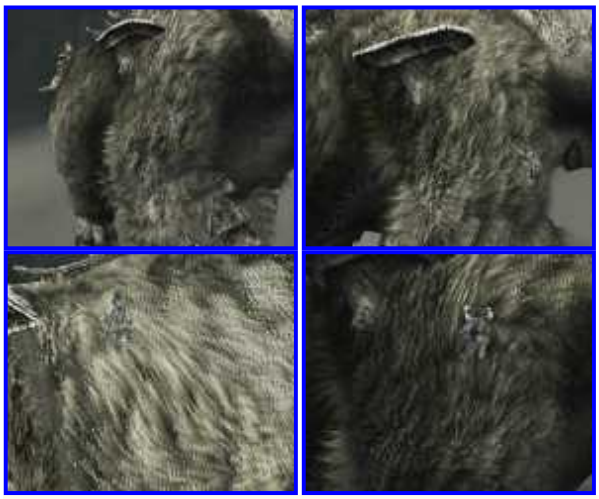
When we implemented this layered fur on a GPU from the 1.x generation, we used a normal map to convert the direction into a texture lookup, giving a very realistic per-pixel lighting method. In addition, something that changes the shade drastically is anisotropic lighting (the relationship between the viewing angle and position of the light for each pixel), which gives the fur it's characteristic highlight. Of course, this was not possible on the PS2.

However, with SOTC, we do the fur's lighting on the vector unit, to achieve the same results.

In addition, when the colossus moves, there are times when the skin goes into a strange shape and each layer of the fur deforms to match it. Therefore, correct simulation of the fur is not always done when the colossus moves, in order for it to not look wrong.

In any case, with PS2, the fact that the fur-effect is performed in real-time deserves special praise. We are pleased to draw attention to it as it is one of the key highlights of Shadow Of The Colossus' graphics.

[Fur]



Lighting is done on the vector unit, which can cause the shading of the hair to change during play.

You can verify the 5 layers of fur on the head of this large mammoth-style colossus.

* LOD system and landscape rendering

In SOTC, there are several colossuses, but in order to visit each one you have to travel around the immense landscape with your trusty steed.

It is during this time that we use the landscape rendering system to draw far away places, without cheating using distance fogging.

Landscape rendering is divided into 3 stages, the furthest background being either a rendered image or a texture, which is stuck on a distant polygon. The internal development team nicknamed this "Super Low".

Consequently, most of the medium range which goes from nearby to just past the furthest background, is rendered using a low-resolution landscape. This is managed in units of 600x600m, and as the player approaches it, once he gets nearer than a certain fixed distance, it changes to the high-res landscape model for the foreground. This switch is designed to blend well so it not obvious.

In addition before being combined down into one picture for the Super Low, most of the distant scenery being managed in 600x600m chunks has been reduced to around 1/30th - 1/100th of the polygons. Because there is little memory, we usually try and spread it around the landmark which represents that area. The combined image for the most distance scenery is done using this low-res model. With this, the low-resolution model appears seamlessly out of the Super Low image. By connecting the foreground and distant background well, we are able to produce a landscape rendering system which seems very natural.

[Landscape rendering]

The SuperLow far view is handled as 2D inside the scenes, but because it is updated using real time rendering, the change from SuperLow into the low-res model is done seamlessly.

The grassy area up to the tree is the foreground high-res model, the cliff in the mid-distance is the low-res model, and the distant background is the SuperLow part.

The trees behind the rocky mountain belong to the SuperLow distant view.

Furthermore, the foreground landscape centered around the player is done with a high-res model which is managed in units of 100x100m.

The colossus is staggeringly large, but we did not use a LOD system based on view distance like the landscape because there was no memory to spare. We use the high accuracy model even when it is far away.

By the way, you wander about this immense land but do you notice that there are no loading time? We think that many players haven't noticed this, because you move around so smoothly. While SOTC uses a conventional resource management system, as you wander about the landscape, in fact, we perform dynamic background reading.

When you enter into a new area, the landscape data and textures etc. of the new area are read, the data which is no longer needed is thrown out. As this repeats, the memory eventually becomes fragmented into used and unused sections. Memory efficiency becomes very bad then.

With SOTC, during the vertical retrace period, and there is time to spare, we perform rearrangement processing of fragmented memory areas in the background. The program obtains information about the relocation addresses, so it can maintain the original setup. This isn't a visible technology like graphics, but because high-level technology is utilized even in such an area, the total quality of the game is heightened.

* Collision with the colossus, and the "deforming collision system"

The "deforming collision system" buzzword was mentioned on an official website, and attracted huge attention. Here we explain it.

When it comes to collision, it often seems that the huge boss character found in a conventional game seems to be kind of "plain" in relation to the player and his actions.

For example, if you are crushed by that gigantic figure, then you just die..... or when you shoot, it just takes away damage.... and that's it.

> "With Shadow Of The Colossus, because we realized that the interaction would be very 'close', allowing the player character to hang onto the gigantic figure, a system of management and control was needed which was very unconventional. This is called the 'deforming collision system'. "
>
> *- Hajime Sugiyama*

[Clinging to the colossus]



The dynamics of Shadow Of The Colossus come from the close interaction with the giant character.

With 3D games, the collision shape is often separate from the obvious 3D graphics.

For example, a 3D game in which the character can climb up rock faces is now common, but this can need precise collision detection where the collision shape matches the rock model exactly.

On the one hand, if the game is a 3D shooter where the player shoots enemy characters, the collision decision is taken using a shape that roughly fits the shape of the enemy. It usually has a cube or sphere shape for each bone in the 3D model for the collision detection, to match the shape of the character, and that's it.

In SOTC, the colossus model is to some extent what could be called the "landscape", but when this model swings an arm around, or twirls his whole body around, using an inflexible collision model is not a great idea. If we were to use a system of cubes and spheres, it would be very apparent when you got up close to the colossus, which is not good.

So with SOTC, kind of like the example of the 3D shooter game, we use a 3D collision model which is close to the shape of the colossus. When the colossus changes it's pose, the bone inside it moves, but the collision model is programmed to follow the movement of the bone - it becomes "deformed". This is where the "deforming collision" name comes from. Furthermore, the collision model is at no point actually drawn on-screen, but you can imagine it using the same triangles as for the displayed model.

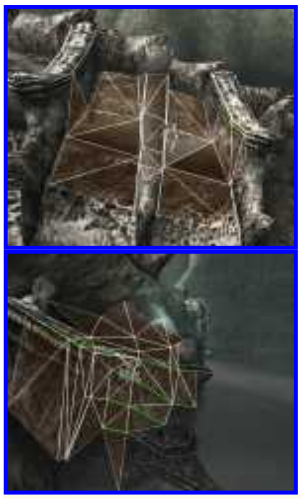
Because it is difficult to perform complex collision detection on the vector unit, we simplify this in order to operate in real-time on the PS2.

For collision between the player and the colossus, we consider the whole player as a sphere-shaped collision model, and perform a point-to-polygon test against the colossus' collision model. And when colliding the colossus against the scenery, this time we use a sphere for the whole colossus and perform a point-to-polygon test against the scenery's collision model. We use the appropriate standpoint depending on "who vs. who" is involved.

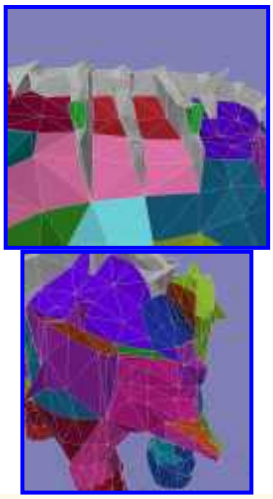
[Collision model of a colossus]

The image which visualizes the collision model of the colossus. The collision model tries to match the actual model well.

[Comparison of a collision model and rendering model for a colossus]

Rendering of the colossus' visible polygons.



The collision model of almost the same spot.

[Detail of collision deformation]

When the colossus moves, it also deforms the collision model. This is collision deformation

* Movement control of the colossus using inverse kinematics

In the simplest 3D games, movement of the character is done by calculating the next position from the player's input and applying it to a running animation. Sometimes with these games you can see a moonwalk effect as they run.

Because SOTC uses such a large colossus which the players looks at in close-up, it would look unnatural if we were to use such a simple method. When controlling such a large model, we literally need the kind of movement where the foot is literally "positioned onto the ground".

Therefore, for Shadow Of The Colossus, a real-time IK (Inverse Kinematics) mechanism is introduced for the walking animation system on the colossus.

By the way, the standard method is FK (Forward Kinematics), where you have a "child" joint and position, which are arranged relative to a "parent" joint by a given angle. Conversely, IK involves calculating the angles when given a desired end position and then setting the parent joint from that. A character with IK needs complex processing applied. Kinematics are an area which is very active in the field of robotics as a research theme, but, has recently reached the point where it has been part of research for 3D games. Shadow Of The Colossus is something that has continued this trend.

With SOTC, we used the colossus AI (described later) to calculate the movement direction and distance for the colossus, and then calculate the position needed for the foot to form each step successively.

When the colossus walks, the basis of the walk animation is handmade animation data, but in order to work with the position of the foot which has been calculated, we need to adjust the motion data as we go.

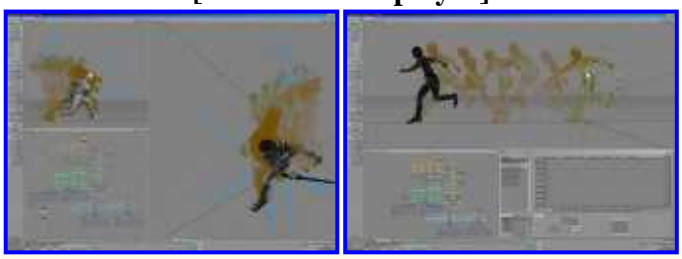
Because there are ups and downs in the landscape, we also need to adjust the angle the foot is placed at. When this happens, we also need to adjust the height - if we just changed the angle it would appear to stand strangely. So then, given the new height of the foot, we calculate the angle and direction for each bone as appropriate. Furthermore, the IK

for the dynamic height adjustment of each bone, is not only done for the colossus, but also for the player character and his horse.

[Motion of the colossus]

The dynamic movement of the colossus is created by a designer

[Motion of the player]



It is not just the playback of the motion data - feeding back the interaction from the scene is a characteristic part of SOTC.

Furthermore, there are sensors surrounding the colossus that cover various areas of the territory. When the player enters into that territory, it triggers the action associated to that sensor.

For example, when the player is on the ground within fixed distance of the front, it shakes it's club and attacks.... when it is facing the rear it tries to turn around to face the front..... if the player is hanging onto the back, it will struggle to shake you off. It depends on the condition that has been set.

A visualization of the AI sensor of the colossus.

Setting which movement is tied to each sensor.... this is a job we give to the scripter, not the programmer. The AI of the colossus is not merely a hard-coded program, it is a superior system that can be easily tuned for each scene.

* A true sense of reality comes from physics simulation

In most recent 3D games, the topic of "physics simulation" usually crops up somewhere. In SOTC, various simulations are done, but it is the "simulation of the pendulum" that is most important.

At times when the player character is hanging onto the colossus, the colossus becomes violent and tries to shake the player off. At that time, he is swung around from the hand where he is attached to it, using a physical simulation of a pendulum to perform it.

We do the simulation of the pendulum from the hand where the player holds on by to the body (the arm), and again from the body down to the foot (the leg); this is what we refer to as a "multiplex pendulum". In addition, expansion and contraction is allowed to perform the simulation of the "spring pendulum" in the arm.

> "Simply put, if it were just this, it would become wooden, but when this is extended with the human motion from the designers, the simulation will try to match this. This is known within our team as 'motion addition'. "
>
> *- Masanobu Tanaka, SCE lead production department*

When movement occurs for a character as a result of physical simulation, or from the player's inputs, we contribute this movement to the character's bones as "additional movement information". Generally this comes from the motion that was set up by the designer for the character.

However, this mechanism was not as easy as one would expect, as it calls for the adjustment of specific bones, which has to be done by hand. In an early stage of development, we added the ability for, say, a hand to fall down and automatically have the body or foot slip. We use the IK mentioned earlier to put this all together.

It is the only practical way, that the player's motion is a combination of "the motion from the IK processing" and "the designer's animation", which makes the final "physical result".

> "Movement becomes wooden if purely controlled by a program, but also a pre-made animation won't fit in right. It is the combination which makes it work. When I saw this working in real time, I was more than a little impressed! I think it has become a truly wonderful system to combine the animator and programmer."
>
> *- Fumito Ueda, SCE lead production department*

Furthermore, this motion addition system is not only for the player, but also the horse he rides on.

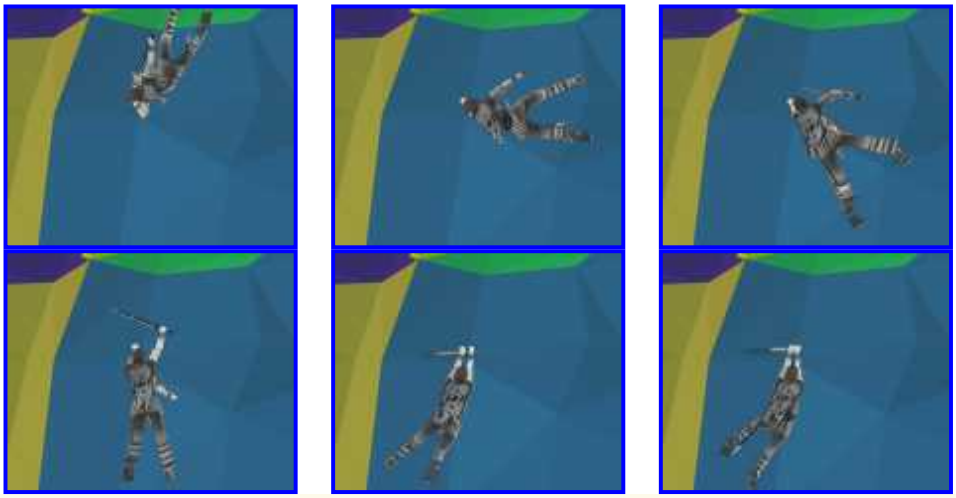
[The motion when holding onto a colossus]

Adjusting skill of the motion data and physical simulation the action also the developer has not seen is produced

< Pendulum and deformation collision >

The collision model and the debug rendering which demonstrates the pendulum simulation.

[Pendulum and motion addition]

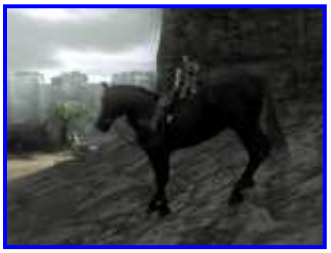


IK is used to ensure consistency between the result of the simulation and the pendulum.

[An example of IK applied to the horse]

Basic pose on level ground.

The solution with a simple 2-bone IK. The hind leg bends too much and looks unnatural.



The pose with the final adjustments. This was fine-tuned by hand, by a designer.

* The fake volume particles and fake light scattering with "high-quality illumination"

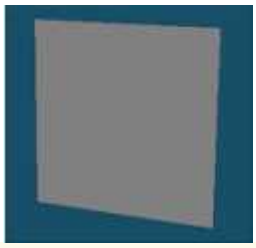
When playing Shadow Of The Colossus, you may notice the realism of the clouds of dust and smoke. Usually, in such an effect you prepare a texture which contains an image of the smoke, and you draw a large number of particles (a 3D sprite with the texture on) using polygons mapped with this texture. This gives the general impression, but sometimes it does not feel 3D enough.

With SOTC, we used a technique to draw 3D smoke, known inside our team as "fake volume particles".

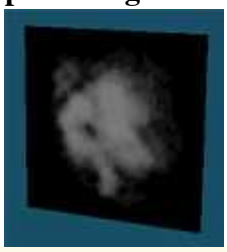
The shadows on the fake volume particle adapt to the lighting of the scene.

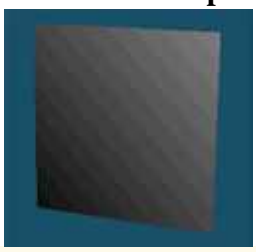
This is a method which performs lighting calculations using the relationship between the viewing direction and the lights affecting the particle, which creates an effect where the particle textures appear properly highlighted. The particles lighting calculation itself becomes expensive for the vector unit, but "the particle effect becomes 3D, because it is real-time and interactive, which is a big effect in itself." - *(Ueda)*

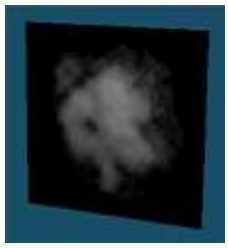
[Conceptual diagram of a fake volume particle]



With a normal particle, it is just displayed as a texture.

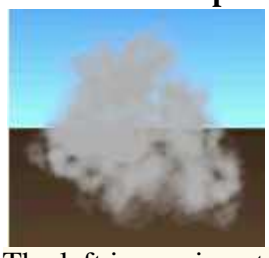






With the fake volume particle lighting done on the vector unit, the light and darkness of the particle texture is changed as a result.

[Experimental image of a fake volume particle]

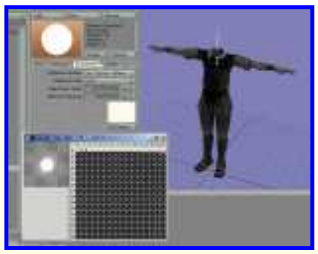


The left image is not lit, but the one on the right shows the lighting calculations from the vector unit.


Similar comparison. The lighting gives a prominent impression of a true 3D volume.

Another thing that was used throughout the game is the fake light scattering, known in the team as the "fairy shader".

For this, when the direction of the view faces a light source, and the character is in-between, we generate a simple white outline of a fixed brightness ratio, using a simple anisotropic lighting calculation. It produces a kind of specular anisotropy image.



The condition that is used is selected by a designer.

During gameplay, when the face, skin, or hair of the main character is backlit, we use an effect to make it become white. Light from the light source coming underneath the surface, appears to shine out again, giving the fake light scattering effect which makes it look kind of transparent.

As a similar technology, the horse's skin uses a "velvet shader" effect (team's internal name) which featured in the game.

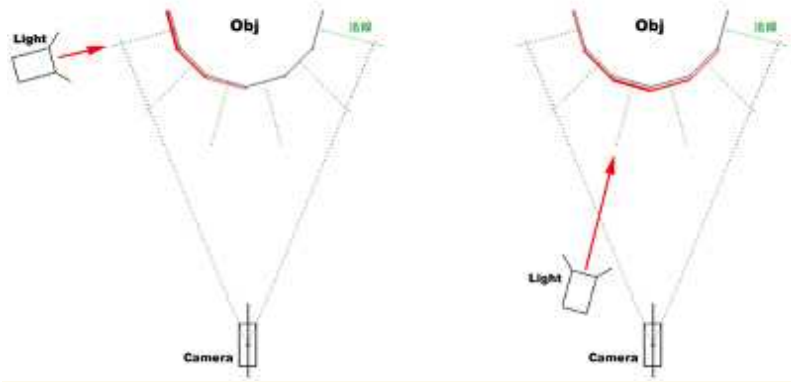
[Fairy Shader]

Picture without the fake light scattering effect (fairy shader).

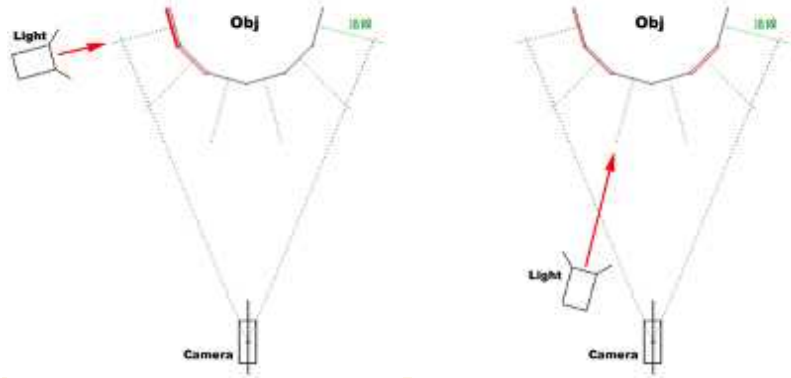
The same image with 30% fake light scattering applied. You can see the faint transparent impression appearing on the right-hand side outline.

This time with 100% fake light scattering, purposely to demonstrate the effect. The silhouette has almost become totally white.

[Velvet Shader]



If direction of the surface (normal vector) is facing the relative lighting direction (light vector), a strong highlight will appear.



With the velvet shader, when the normal vector and view vector (camera direction) are facing as well, we use anisotropic processing to weaken the highlight. When short hair is seen with the eye looking directly at it's end, the reflection is less visible. When it is seen with the eye looking from the side, the hair is fully visible and the reflection is maximized..... this is the basis for the fake reflection model.

* The fantasy visuals of ICO and Shadow Of The Colossus were the end result of much trial and error.

> "Shadow Of The Colossus took approximately 3 years to produce. With ICO, it is somewhere between a cartoon shader and photo realistic visuals. At the start of development it was hard to add things, such as the stencil shadow volumes, strengthening the environmental lighting, etc..... but it was a matter of trial and error, and eventually we arrived at the fantastic visuals."
>
> *- Fumito Ueda*

Also with SOTC, the effects it used such as the fur effect, the motion blur, motion addition, etc - were designed before the actual programming began, by first doing a simple test in LightWare by Mr. Ueda, and then passed down to the person in charge of implementing and adjusting the technique.

With both ICO and SOTC, the unique visual stylization wasn't just something to show off the technology. As well as the graphics programmer, this was something that was planned from the design side to create a consistent visual look, and it was brought to life by the skill of our artists and much fine tuning work.

By the way, we are in frequent contact with fans of SOTC, and they always wonder "why this was not made for the next-generation PS3". Because we had mentioned in the past that the previous game ICO had it's PS1 production cancelled and moved to PS2, which was the next-gen machine at that time, shouldn't we have done this again with the PS3?

> "Once interest in the next-gen happens, and we start performing the conversion to PS3, another console appears, and it never stops! But even if we did, it would have taken another 3 years! (laughing)"
>
> *- Fumito Ueda*

"Basically I feel it was good to complete this on PS2, as it pushes the boundaries of the current machine slightly."

- Hajime Sugiyama

People are concerned about whether a sequel will appear or not on PS3. The epic scale which Shadow Of The Colossus seems ideally suited to the high-def nature of PS3, but....

Most current games which handle collision between two simple objects, such as a sword-vs-sword, seem normal. But with the next-gen consoles, people have seen the interaction between the player and the huge colossus and will expect this to be the standard. I can't saying anything concrete about our next project, but it may use the same theme of "hold, and perform an action" from SOTC. It is possible that this game concept may become standard on the next-generation consoles..... so this seems a nice reason to have it completed now on PS2."

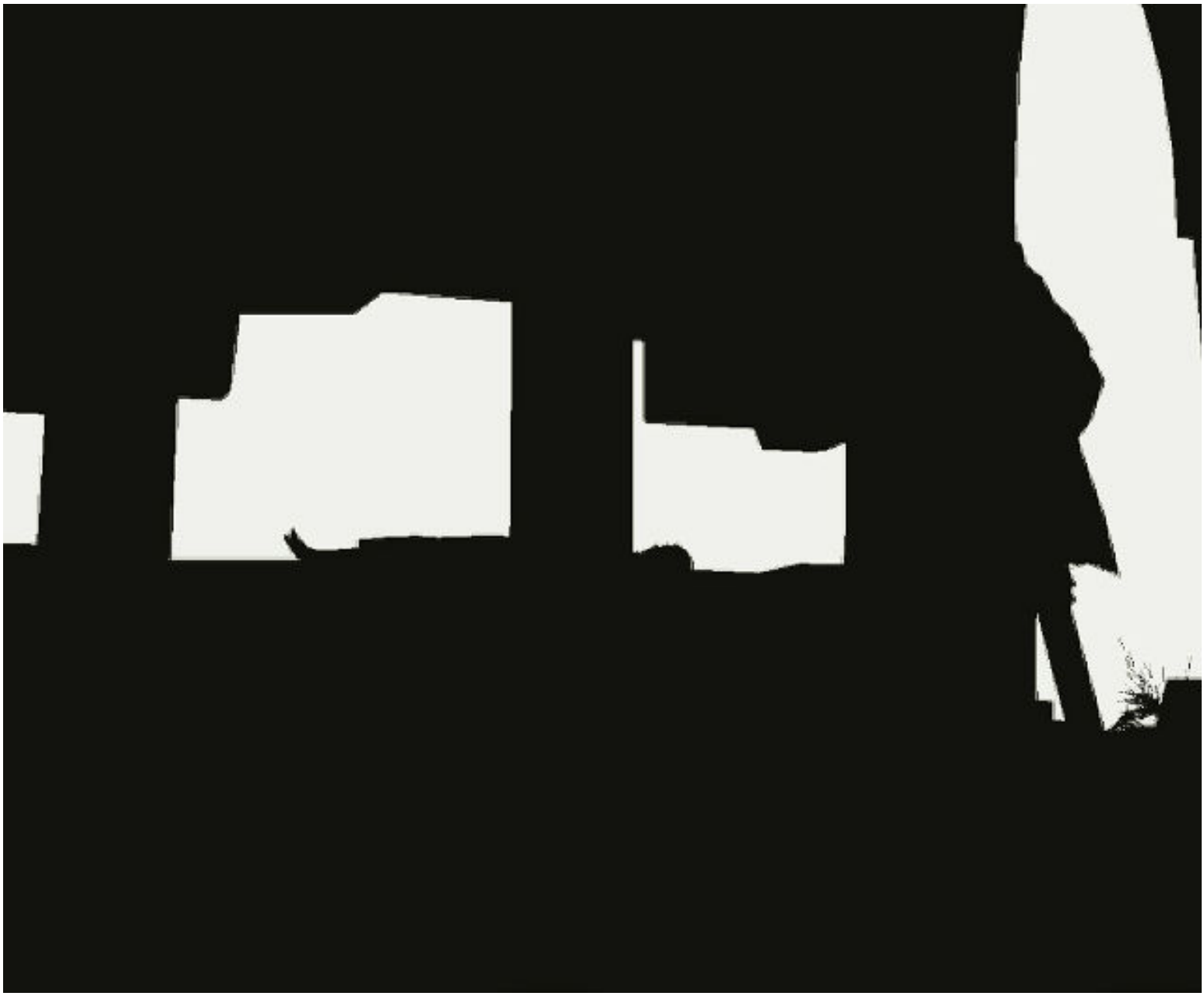
- Fumito Ueda

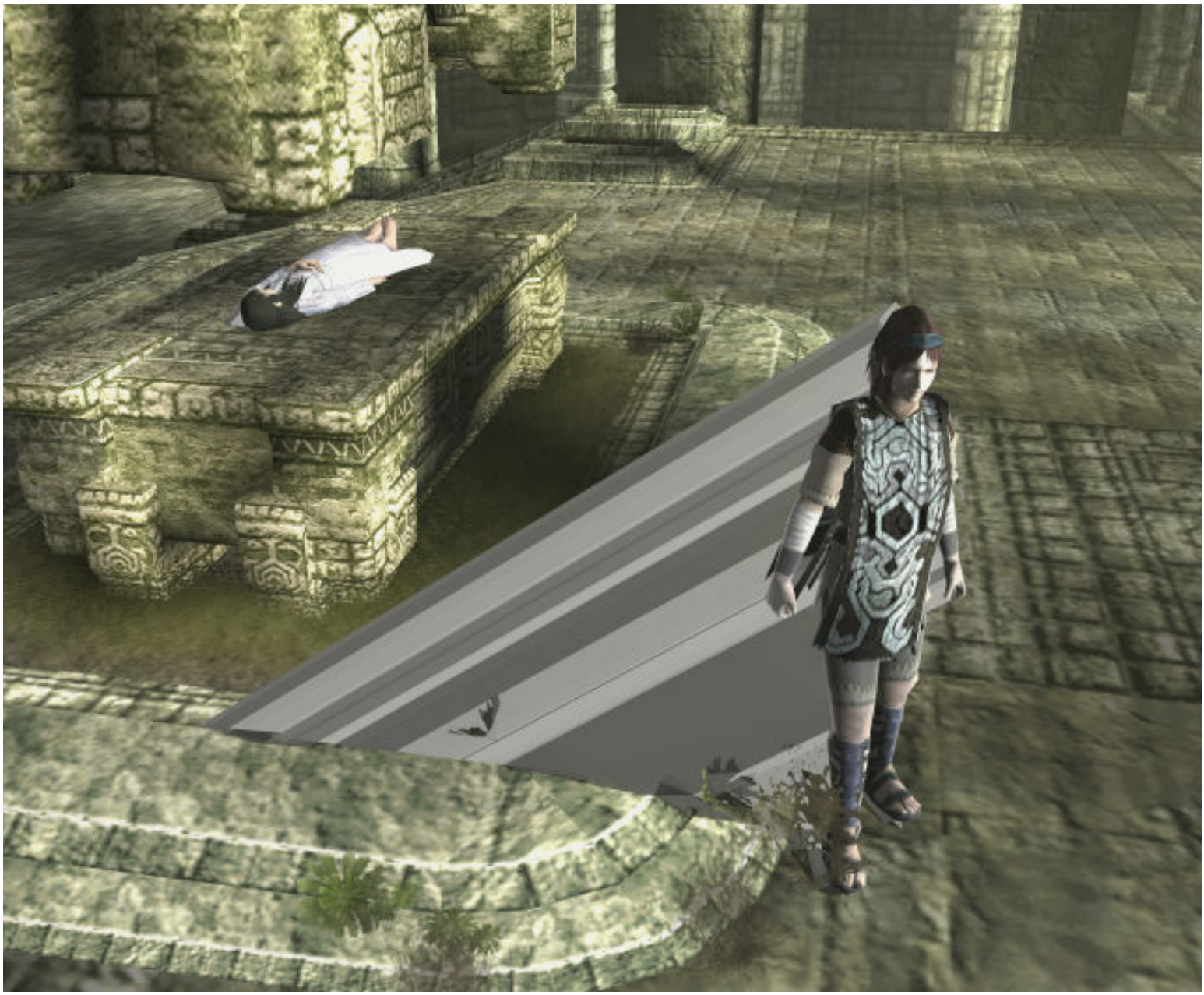
PlayStation home page:
http://www.playstation.jp/
Product information:
http://www.playstation.jp/scej/title/wander/

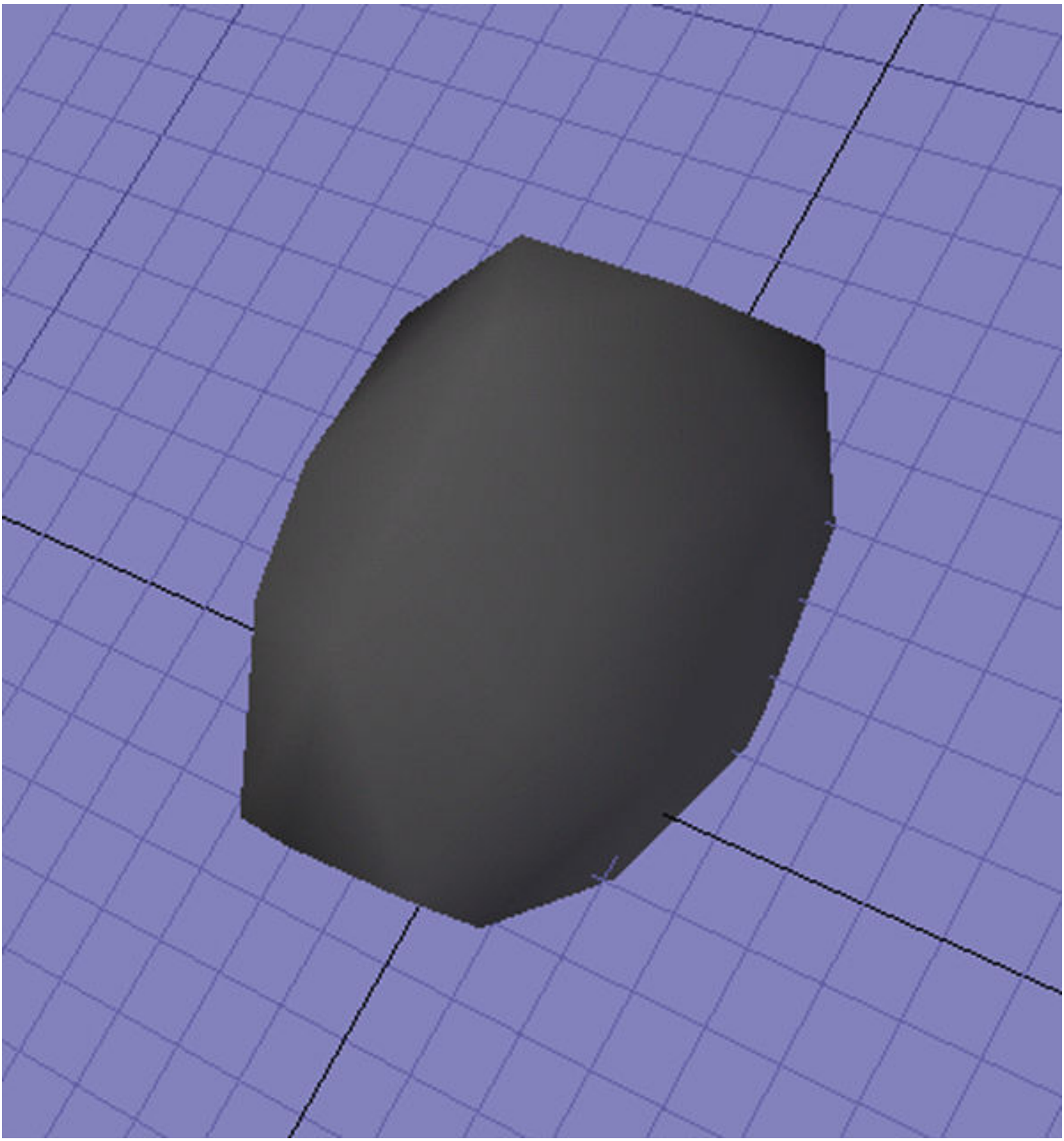
(2005 December 7th)

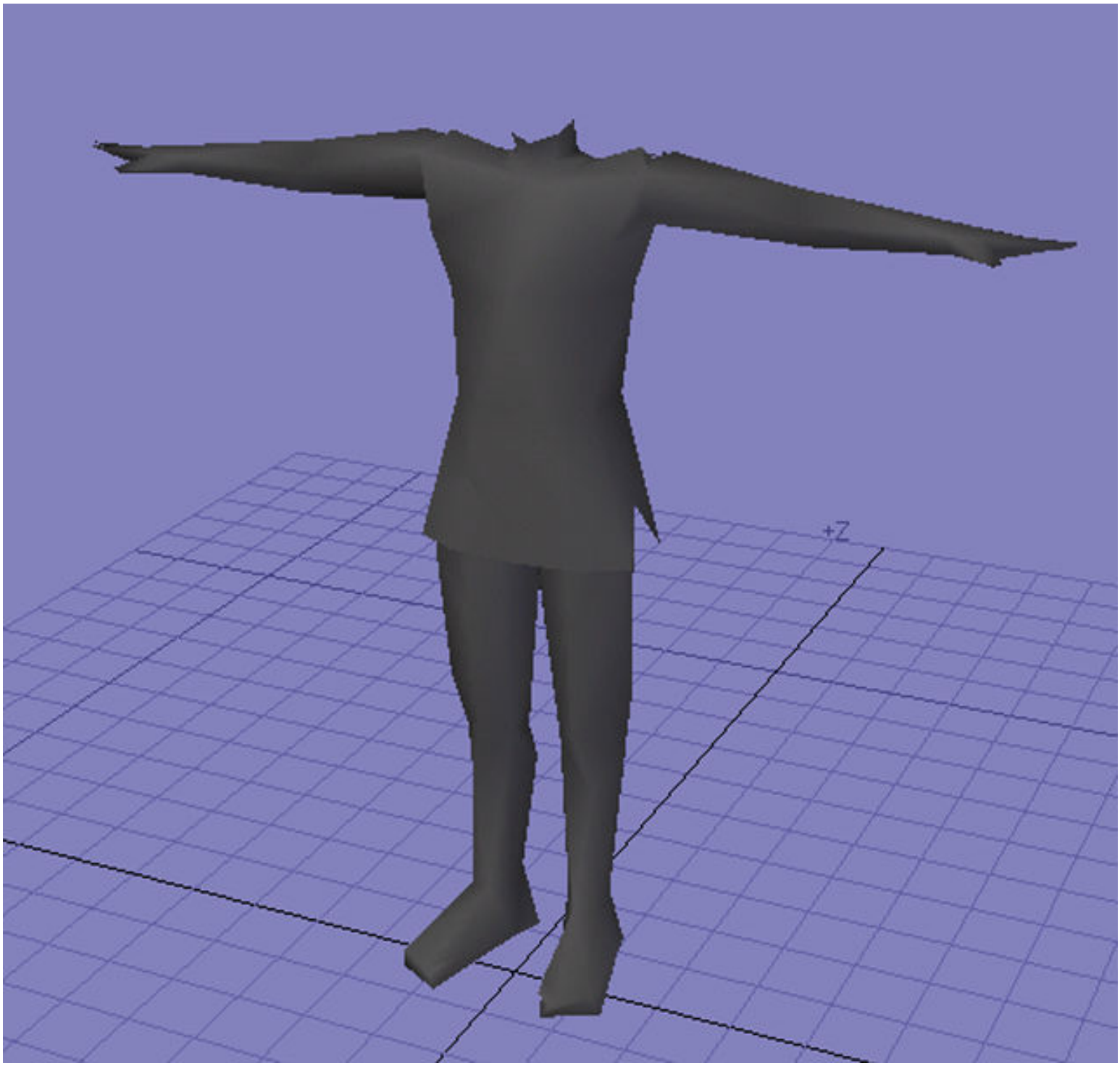
[Interview by Nishikawa]

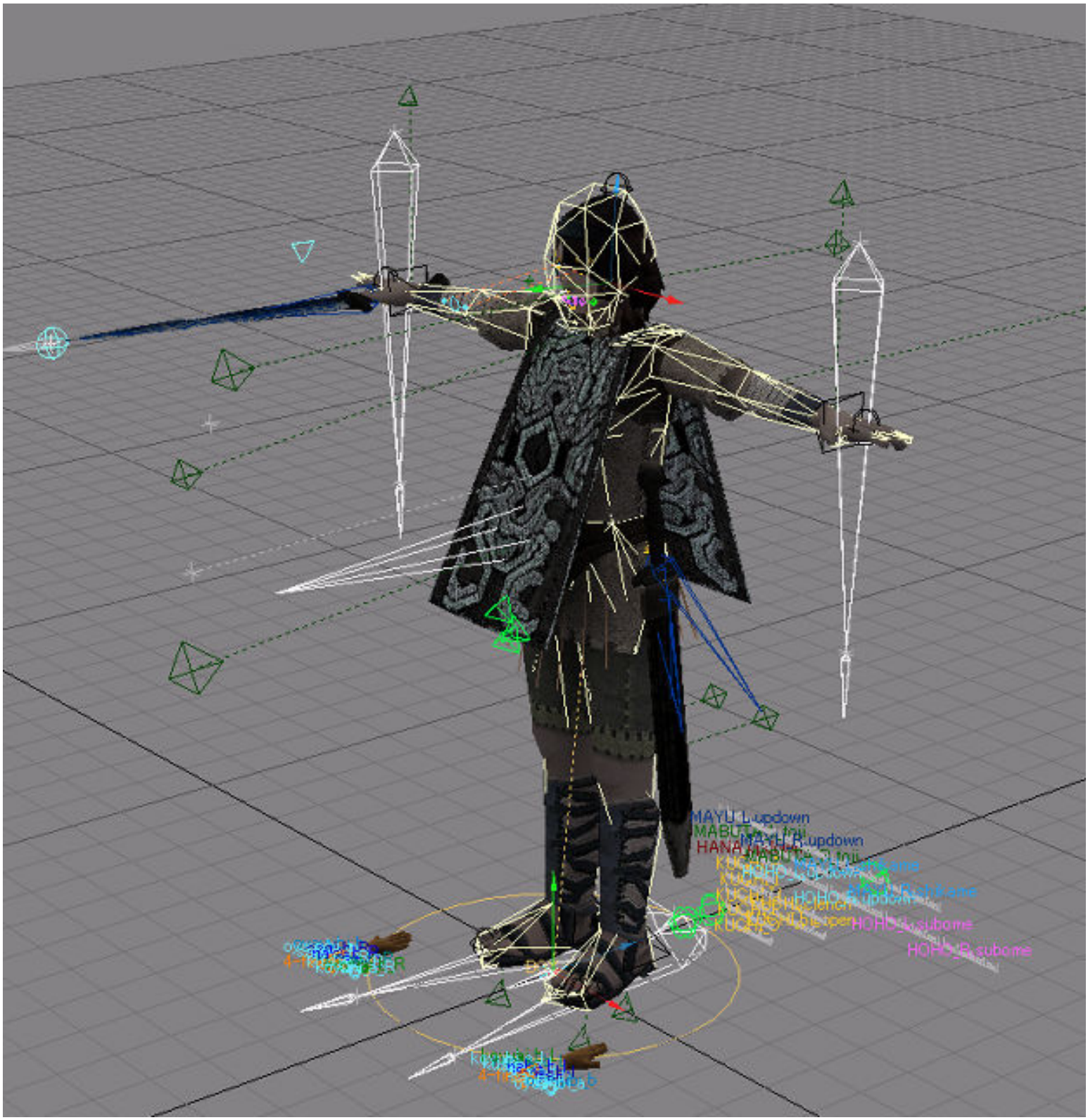

ワンダと巨像
© 2005 Sony Computer Entertainment Inc. All Rights Reserved.
new file
load file
options

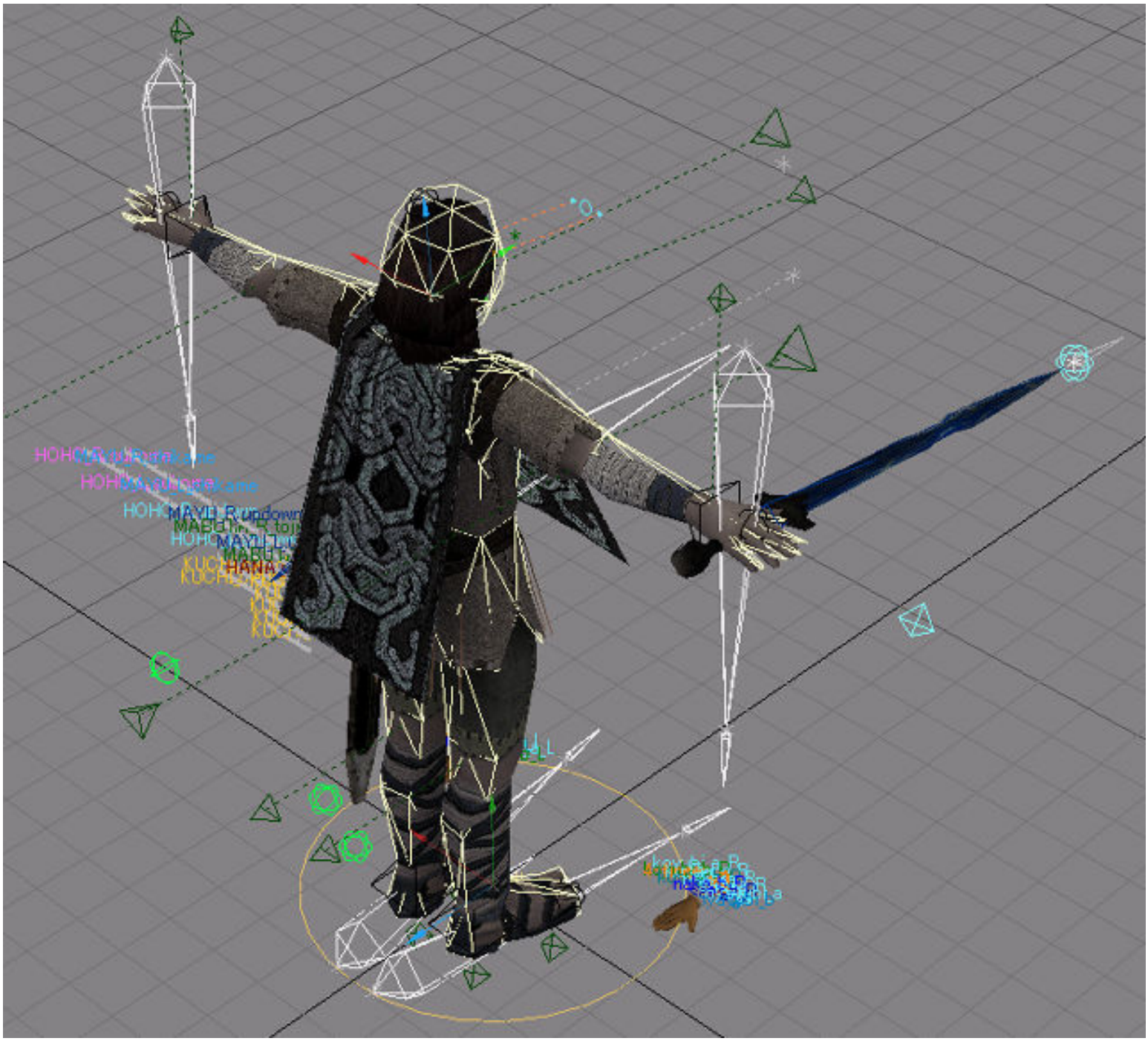


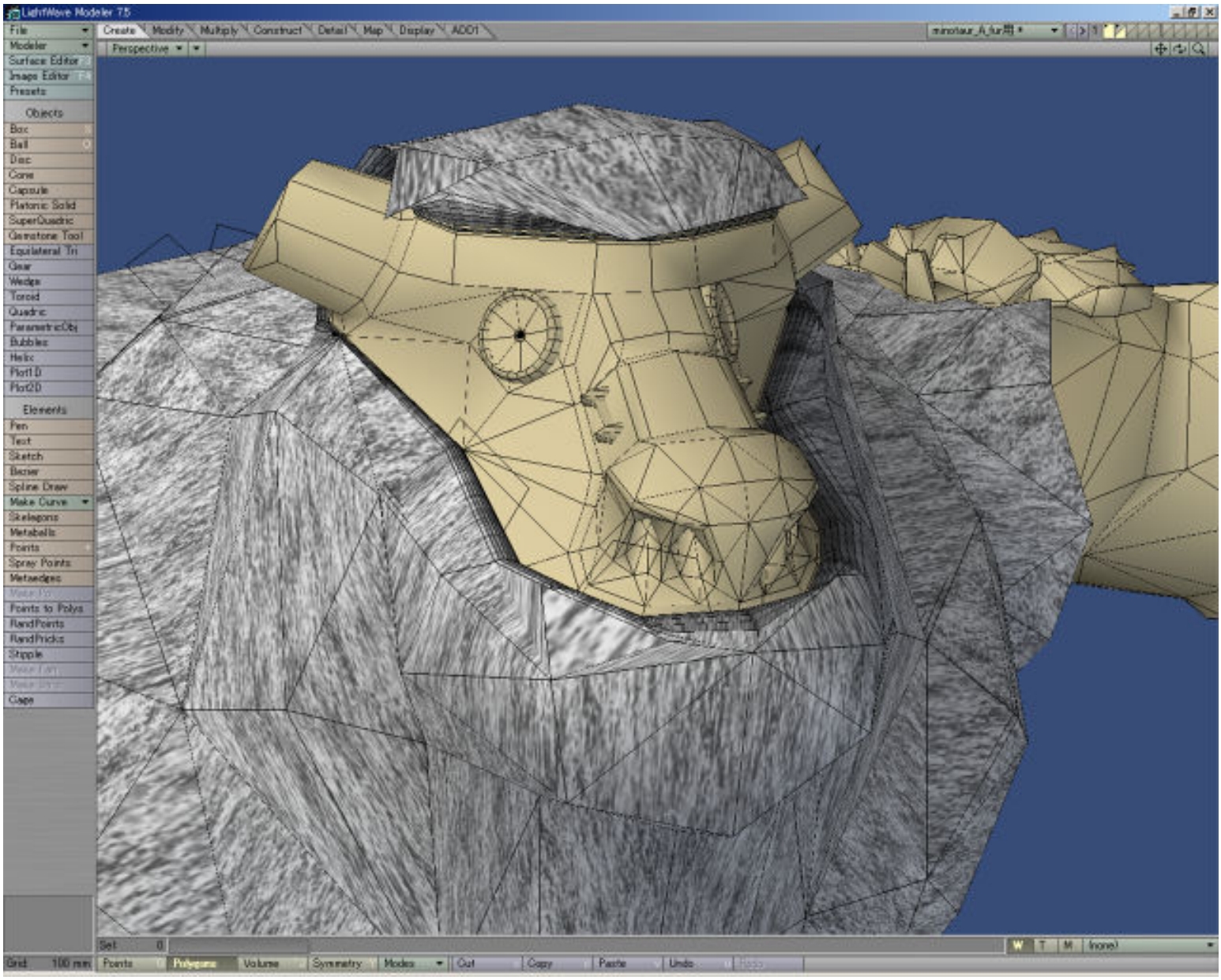


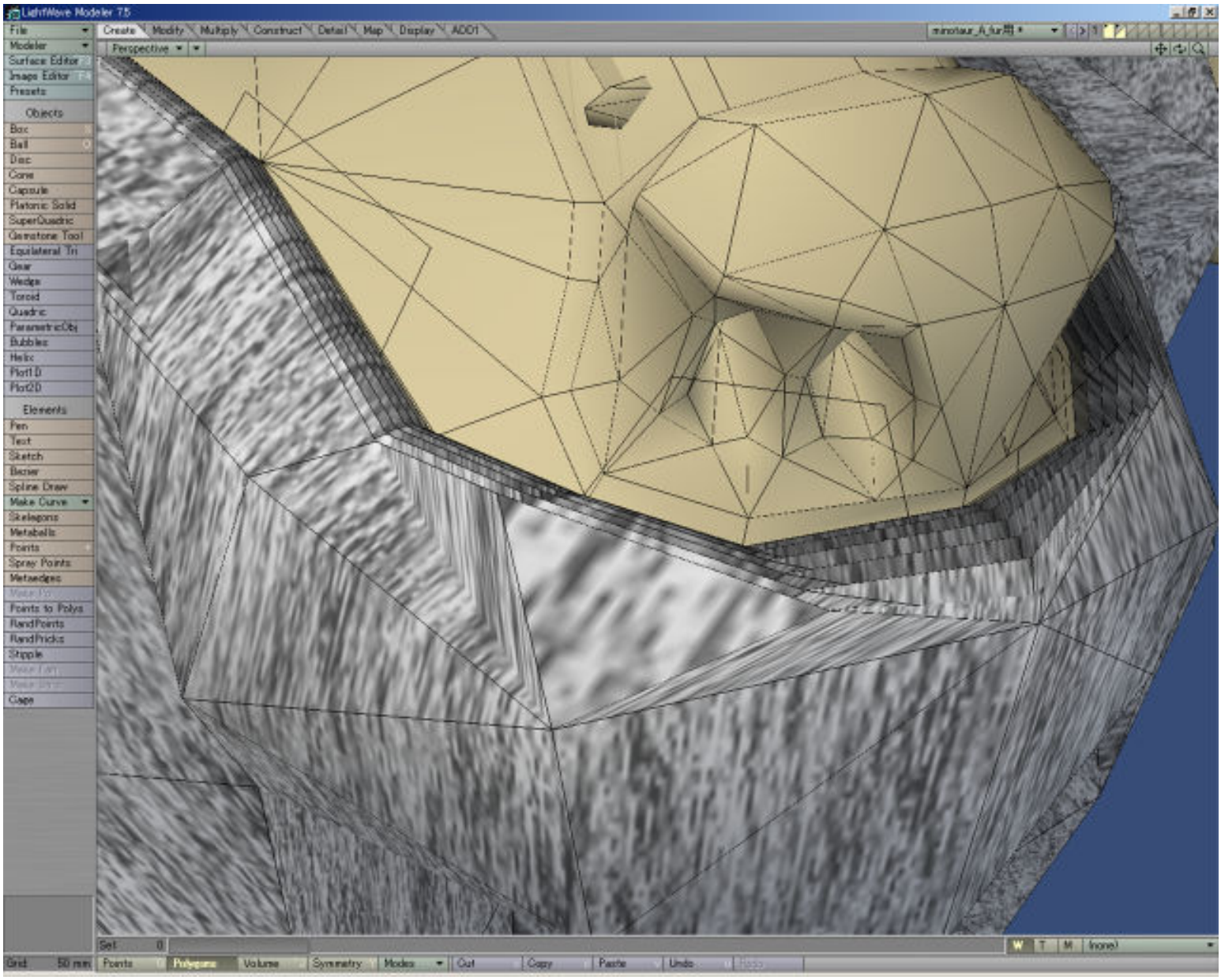


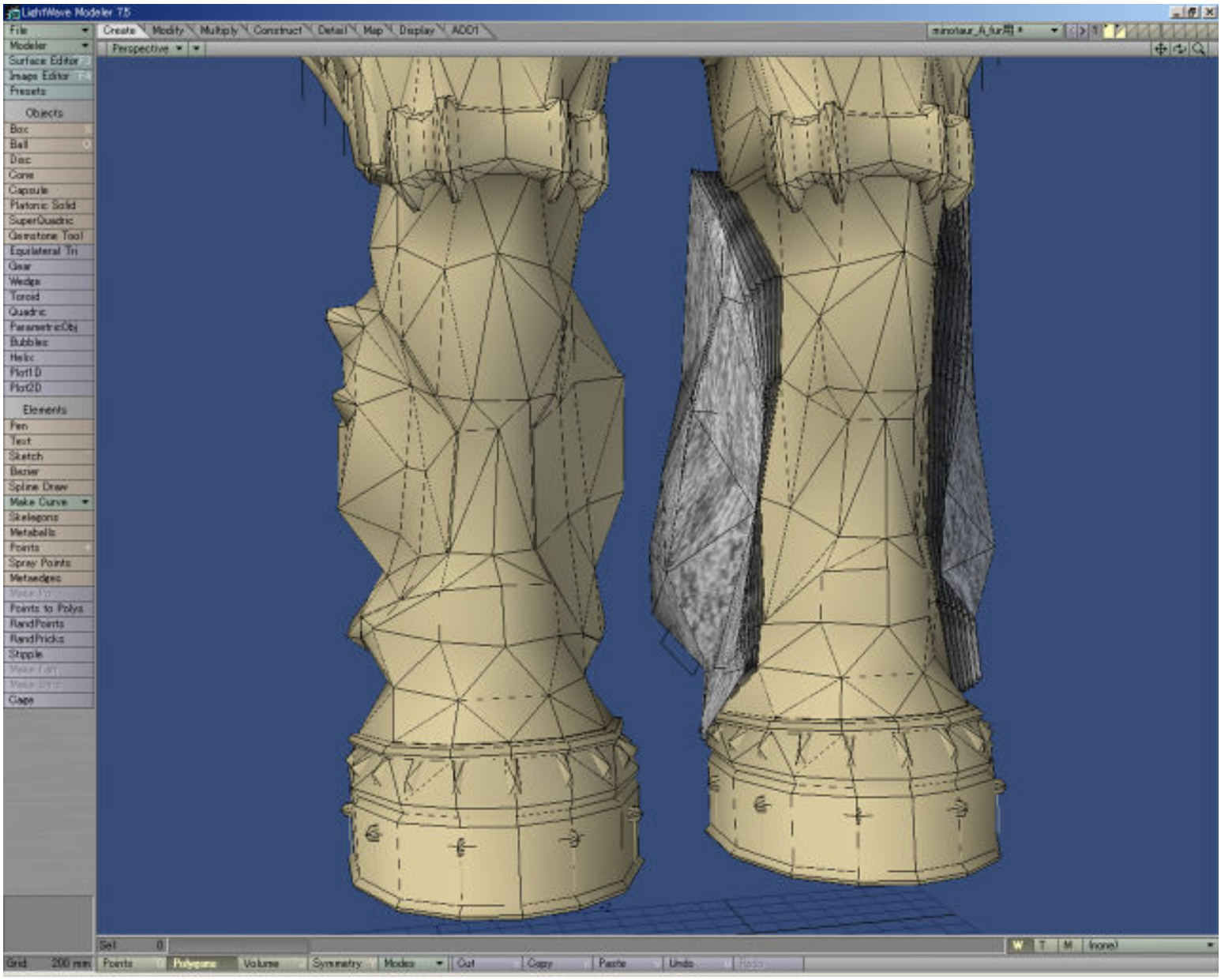

+Z

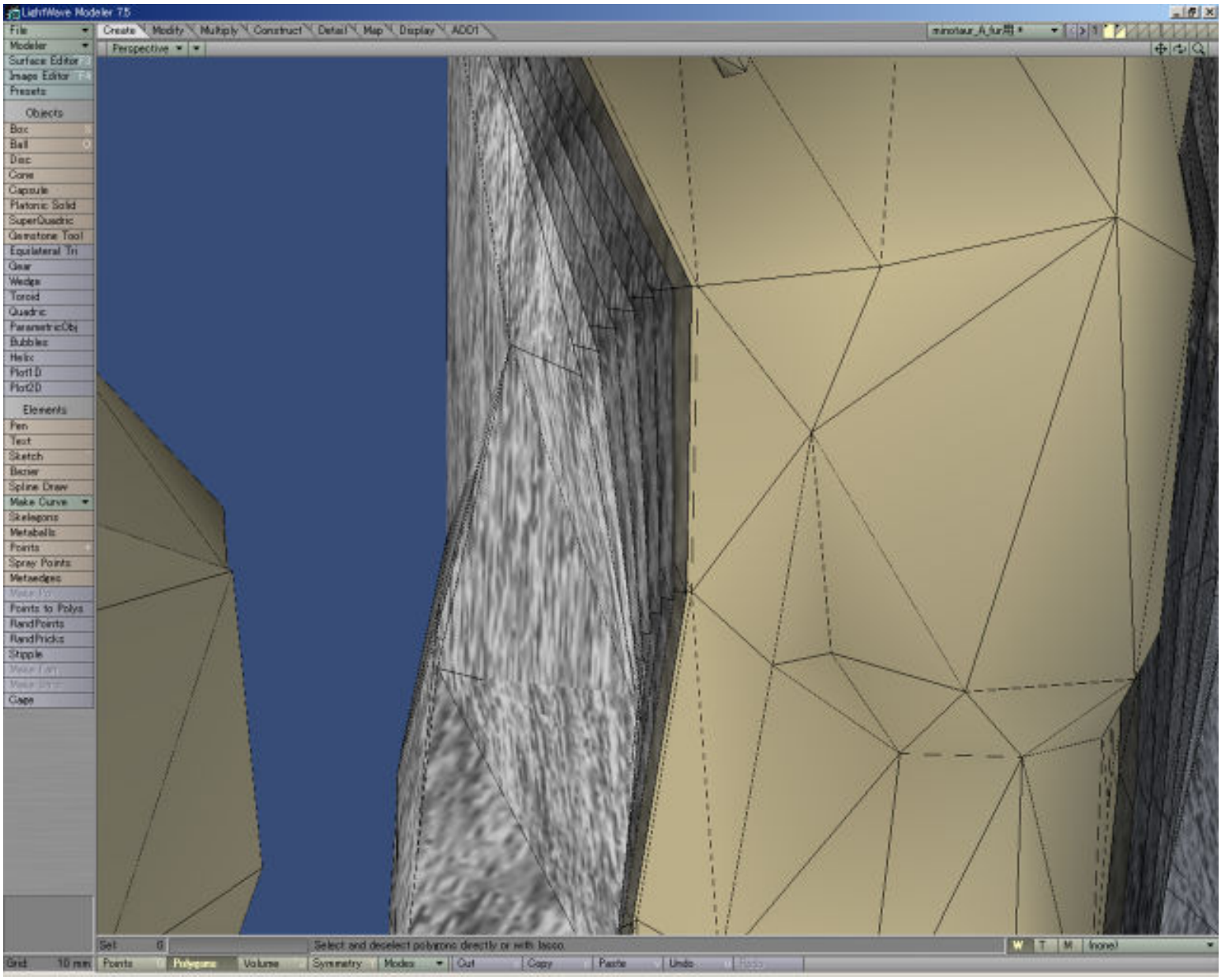

MAYU_L_updown
HANA_MABUTA_L_tsui
MAYU_R_updown
HOHO_L_subome
HOHO_R_subome

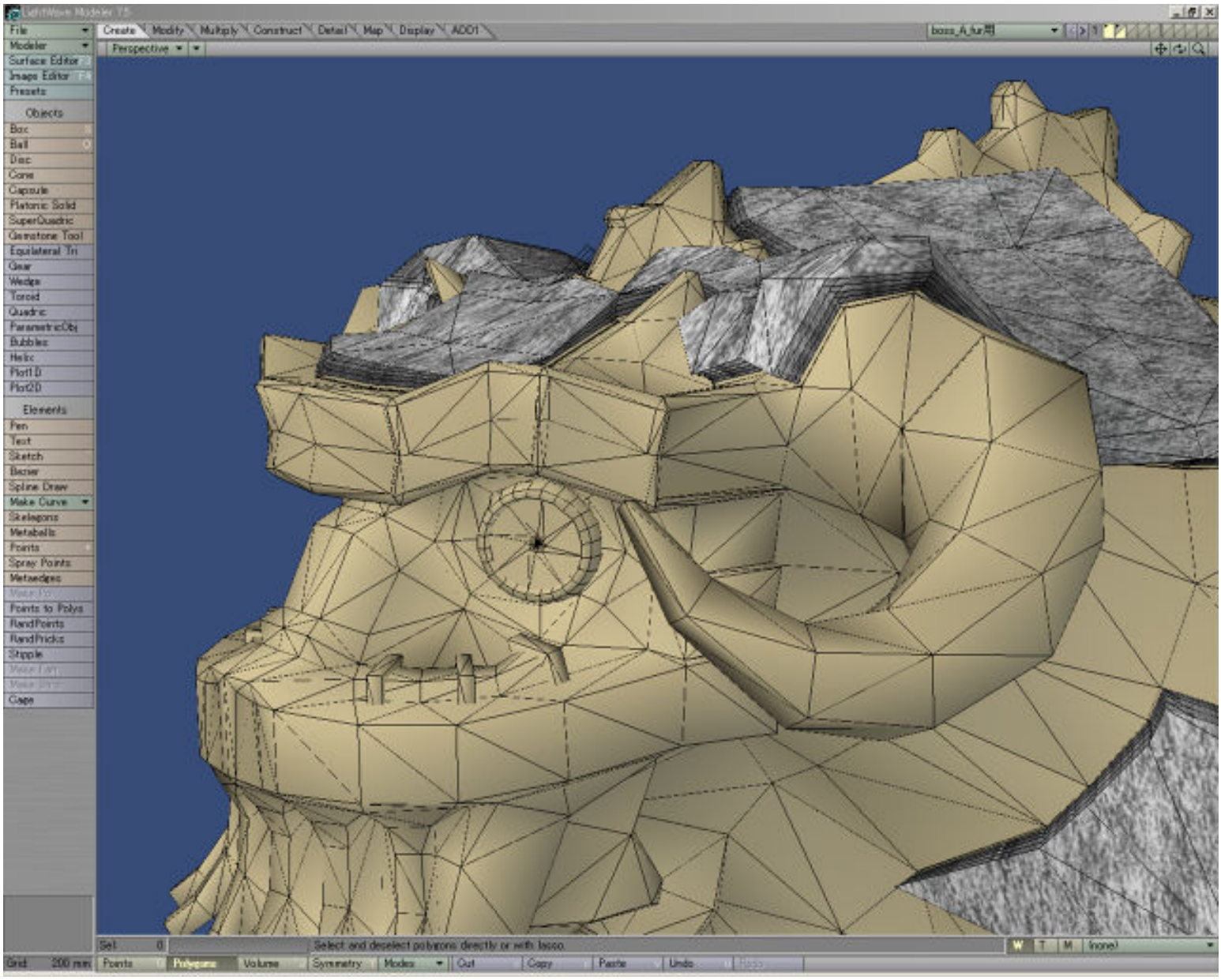


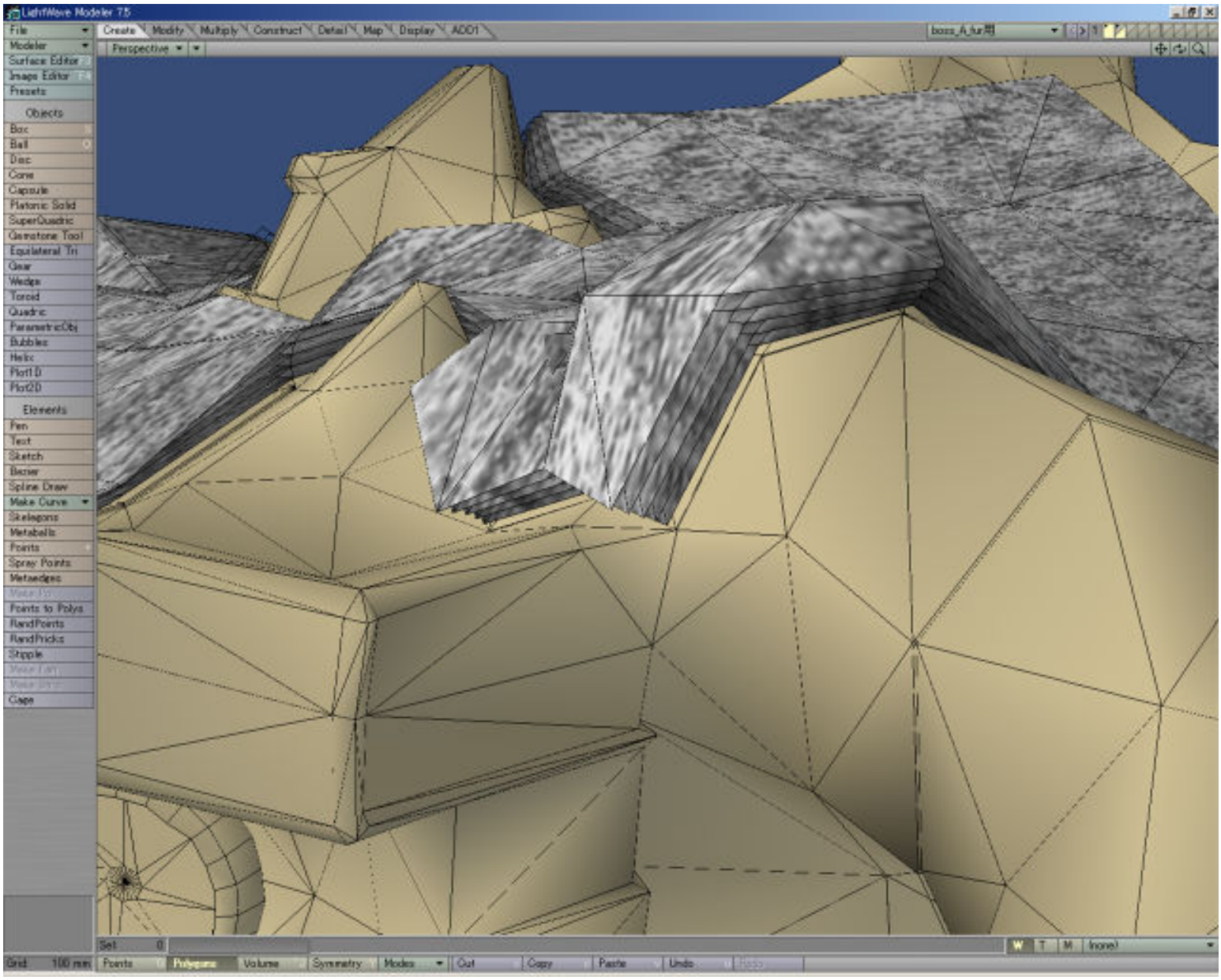

LightWave Modeler 7.5
File
Modeler
Surface Editor
Image Editor
Presets
Objects
Box
Ball
Disc
Cone
Capsule
Platonic Solid
SuperQuadric
Gemstone Tool
Equilateral Tri
Gear
Wedge
Toroid
Quadric
ParametricObj
Bubbles
Helix
Plot1D
Plot2D
Elements
Pen
Text
Sketch
Bezier
Spline Draw
Make Curve
Skelegons
Metaballs
Points
Spray Points
Metaedges
Points to Polys
RandPoints
RandPricks
Stipple
Cage
Create
Modify
Multiply
Construct
Detail
Map
Display
ADD1
Perspective
Set
Grid
100 mm
Points
Polygons
Volume
Symmetry
Modes
Cut
Copy
Paste
Undo
W
T
M
(none)

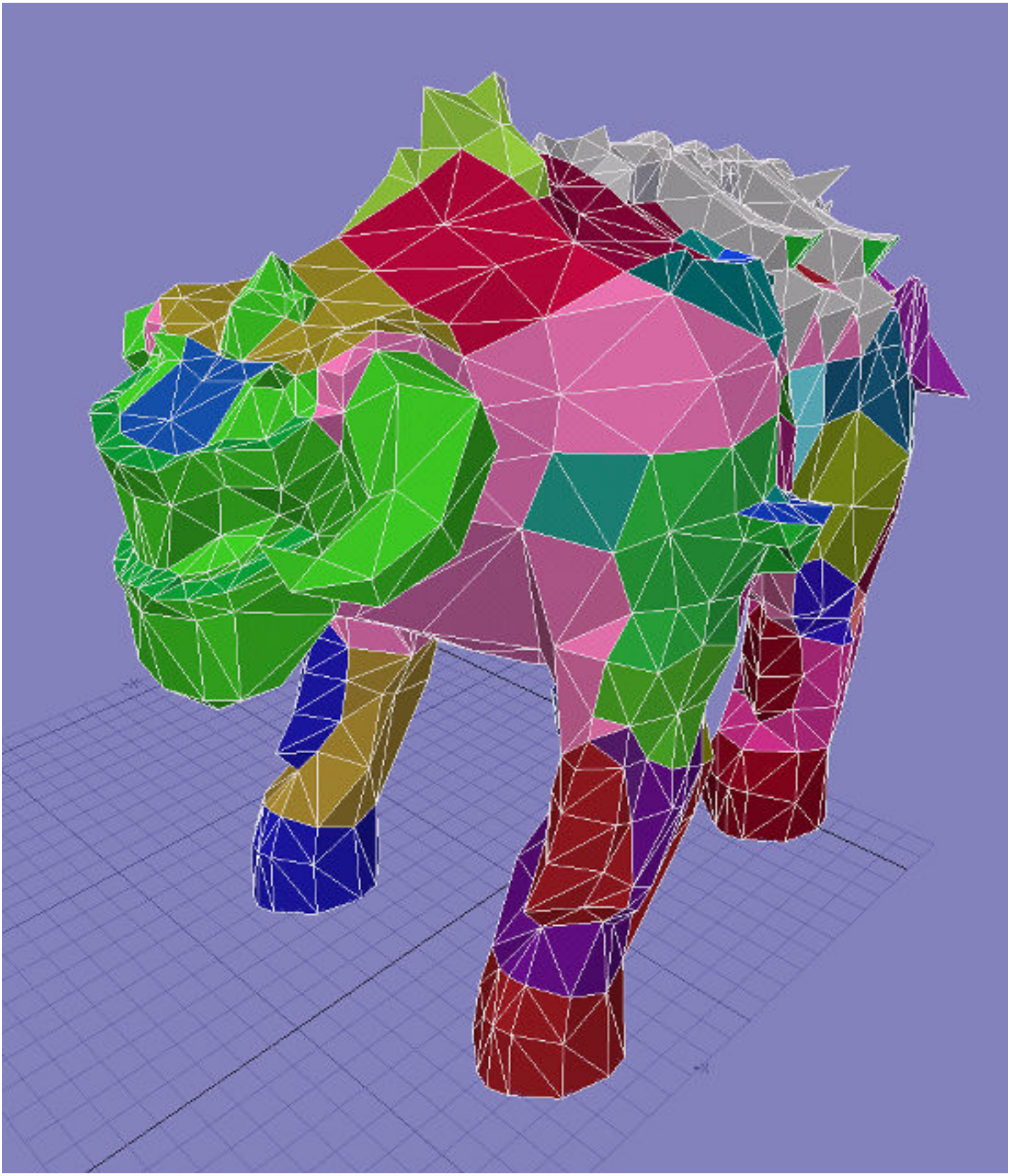

LightWave Modeler 7.5
File
Modeler
Surface Editor
Image Editor
Presets
Objects
Box
Ball
Disc
Cone
Capsule
Platonic Solid
SuperQuadric
Gemstone Tool
Equilateral Tri
Gear
Wedge
Toroid
Quadric
ParametricObj
Bubbles
Helix
Plot1D
Plot2D
Elements
Pen
Text
Sketch
Bezier
Spline Draw
Make Curve
Skelegons
Metaballs
Points
Spray Points
Metaedges
Points to Polys
RandPoints
RandPricks
Stipple
Cage
Create
Modify
Multiply
Construct
Detail
Map
Display
ADD1
Perspective
Set
Grid: 50 mm
Points
Polygons
Volume
Symmetry
Modes
Cut
Copy
Paste
Undo
W
T
M
(none)

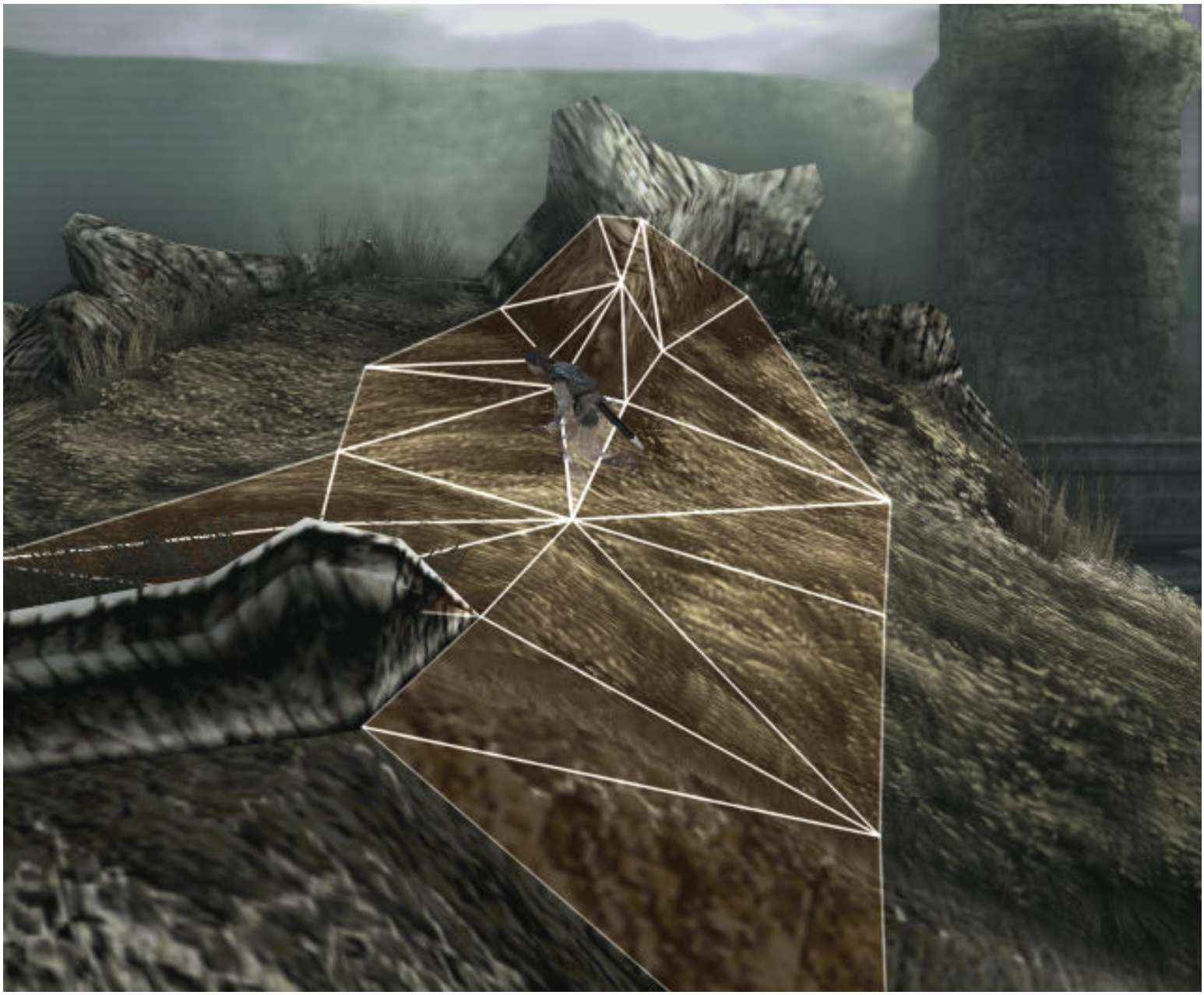

LightWave Modeler 7.5
File
Modeler
Surface Editor
Image Editor
Presets
Objects
Box
Ball
Disc
Cone
Capsule
Platonic Solid
SuperQuadric
Gemstone Tool
Equilateral Tri
Gear
Wedge
Toroid
Quadric
ParametricObj
Bubbles
Helix
Plot1D
Plot2D
Elements
Pen
Text
Sketch
Bezier
Spline Draw
Make Curve
Skelegons
Metaballs
Points
Spray Points
Metaedges
Points to Polys
RandPoints
RandPricks
Stipple
Cage
Create
Modify
Multiply
Construct
Detail
Map
Display
ADD1
Perspective
Sel
Grid
Points
Polygons
Volume
Symmetry
Modes
Cut
Copy
Paste
Undo
W
T
M
(none)

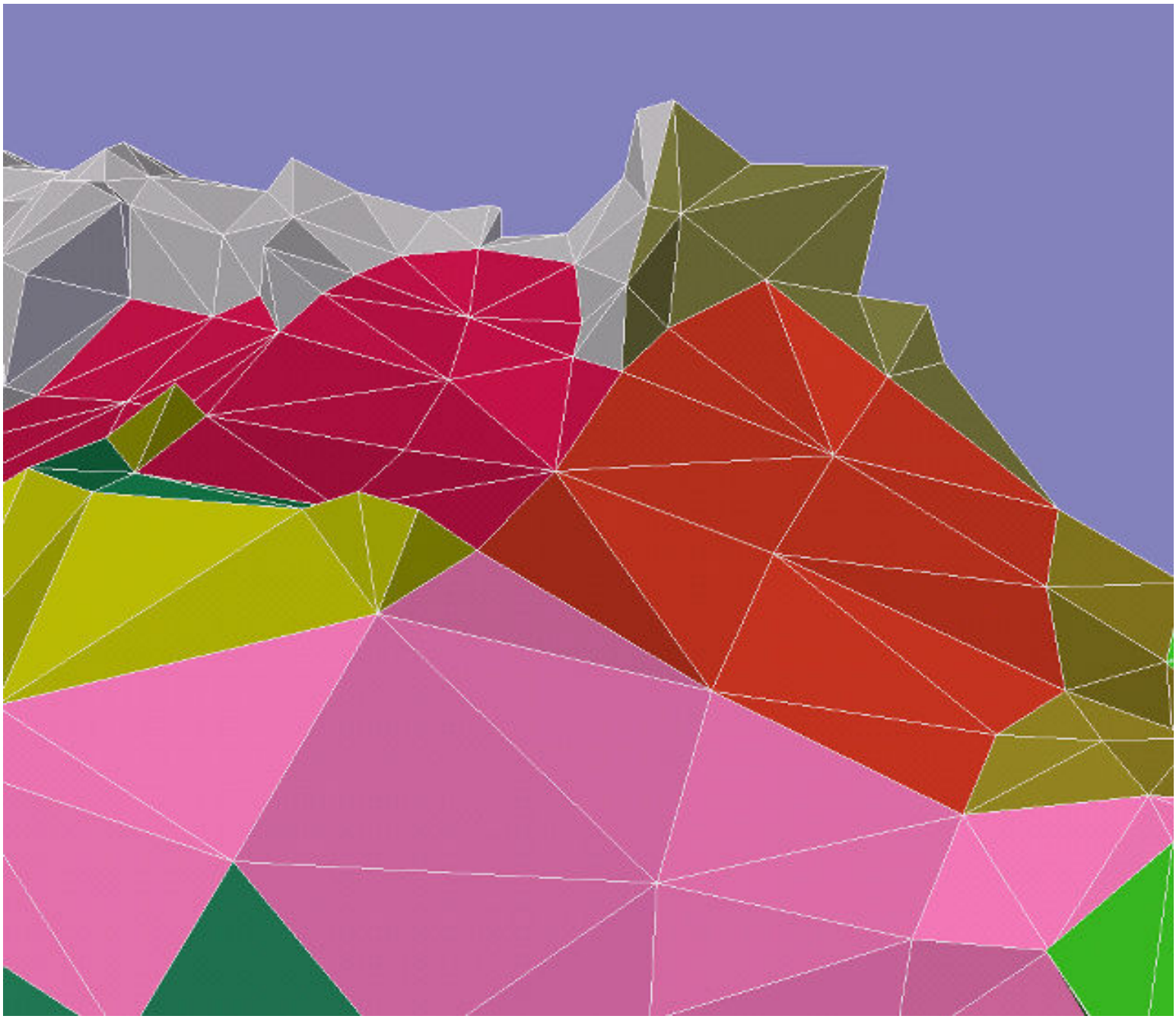

LightWave Modeler 7.5
File
Modeler
Surface Editor
Image Editor
Presets
Objects
Box
Ball
Disc
Cone
Capsule
Platonic Solid
SuperQuadric
Gemstone Tool
Equilateral Tri
Gear
Wedge
Toroid
Quadric
ParametricObj
Bubbles
Helix
Plot1D
Plot2D
Elements
Pen
Text
Sketch
Bezier
Spline Draw
Make Curve
Skelegons
Metaballs
Points
Spray Points
Metaedges
Points to Polys
RandPoints
RandPricks
Stipple
Cage
Create
Modify
Multiply
Construct
Detail
Map
Display
ADD1
Perspective
Sel
0
Select and deselect polygons directly or with lasso
W
T
M
(none)
Grid
10 mm
Points
Polygons
Volume
Symmetry
Modes
Cut
Copy
Paste
Undo

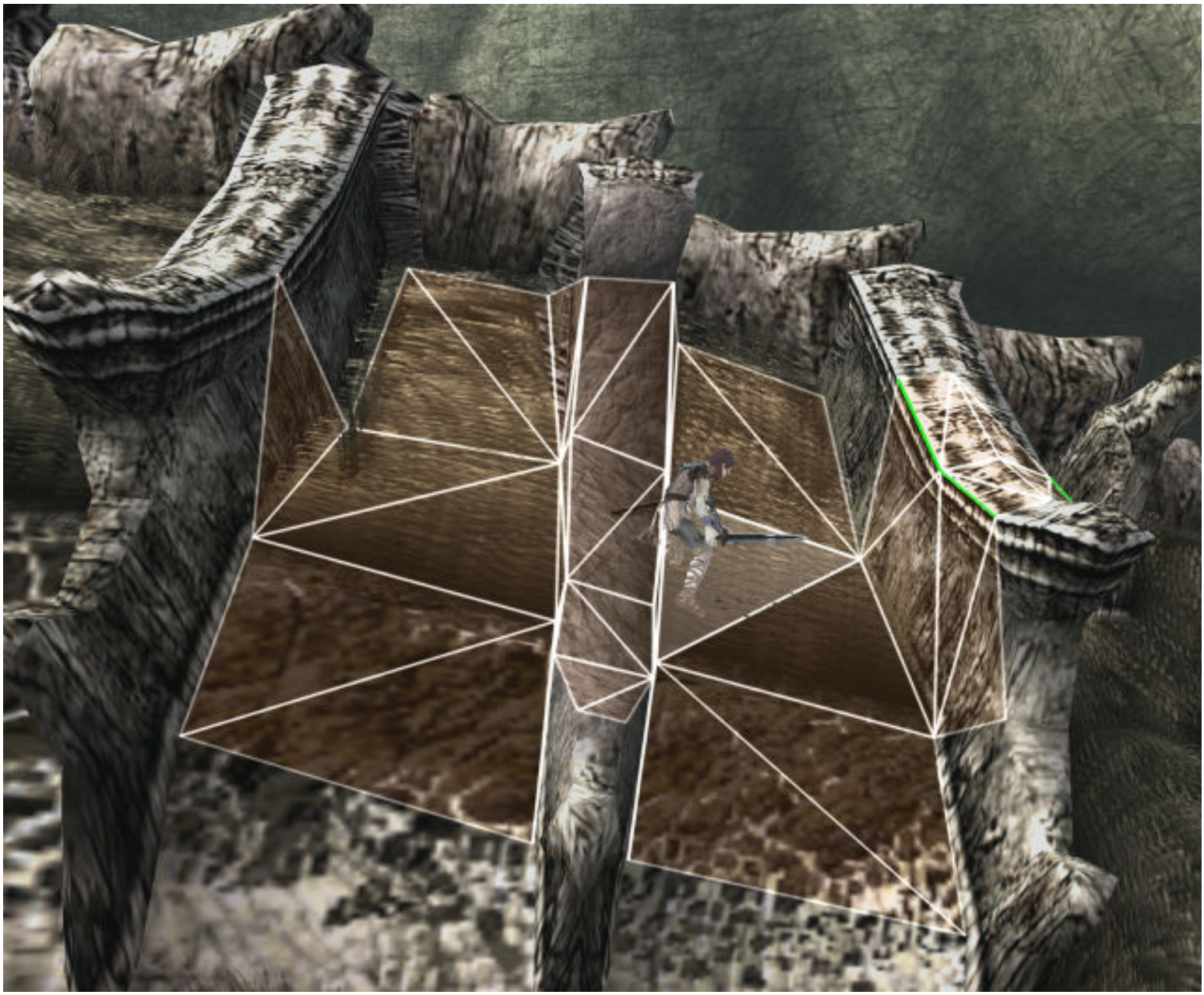

Create
Modify
Multiply
Construct
Detail
Map
Display
ADD1
Perspective
File
Modeler
Surface Editor
Image Editor
Presets
Objects
Box
Ball
Disc
Cone
Capsule
Platonic Solid
SuperQuadric
Gemstone Tool
Equilateral Tri
Gear
Wedge
Toroid
Quadric
ParametricObj
Bubbles
Helix
Plot1D
Plot2D
Elements
Pen
Text
Sketch
Bezier
Spline Draw
Make Curve
Skelegons
Metaballs
Points
Spray Points
Metaedges
Points to Polys
RandPoints
RandPricks
Stipple
Cage
Select and deselect polygons directly or with lasso
Points
Polygons
Volume
Symmetry
Modes
Cut
Copy
Paste
Undo

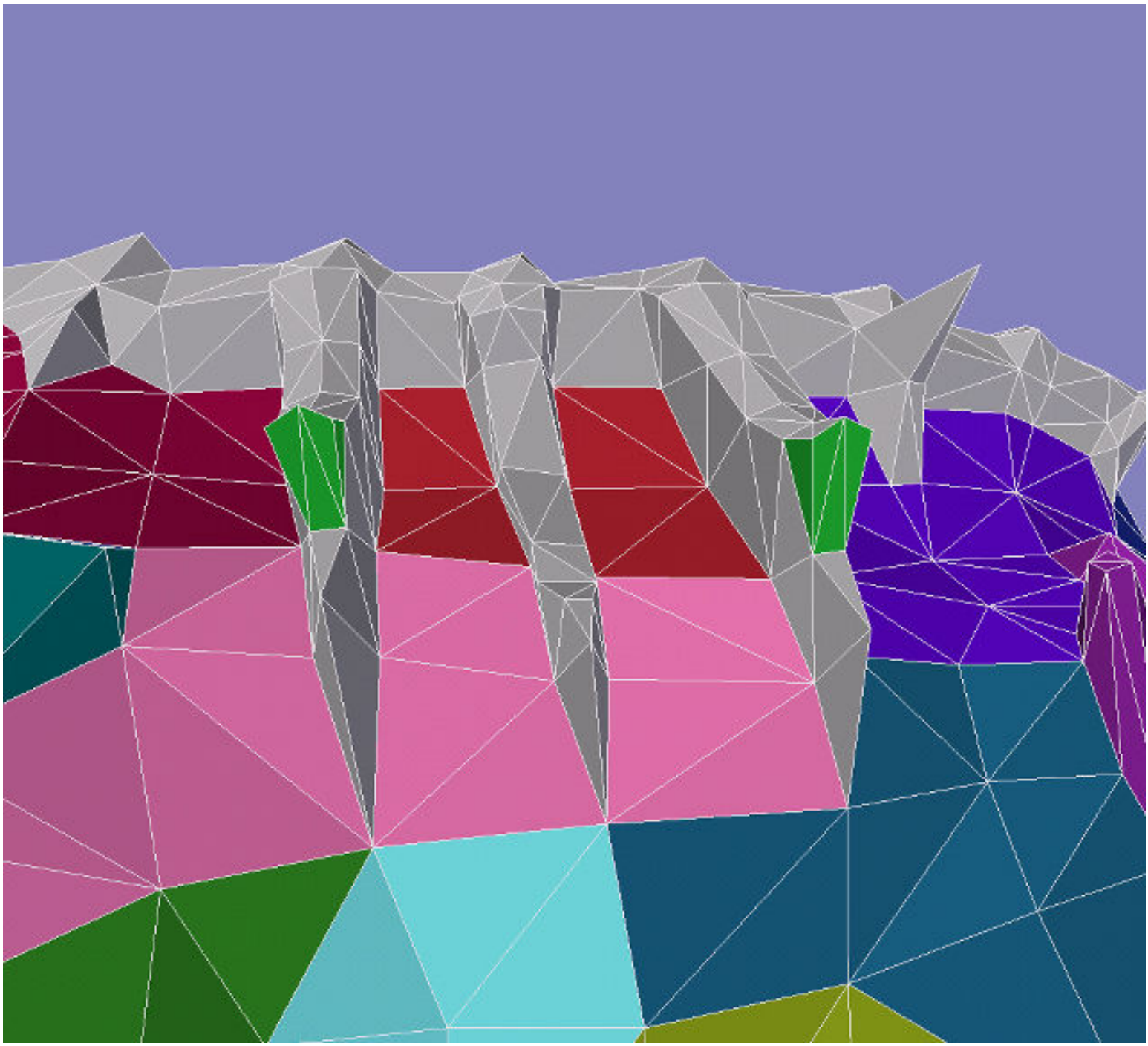

LightWave Modeler 7.5
File
Modeler
Surface Editor
Image Editor
Presets
Objects
Box
Ball
Disc
Cone
Capsule
Platonic Solid
SuperQuadric
Gemstone Tool
Equilateral Tri
Gear
Wedge
Toroid
Quadric
ParametricObj
Bubbles
Helix
Plot1D
Plot2D
Elements
Pen
Text
Sketch
Bezier
Spline Draw
Make Curve
Skelegons
Metaballs
Points
Spray Points
Metaedges
Points to Polys
RandPoints
RandPricks
Stipple
Cage
Create
Modify
Multiply
Construct
Detail
Map
Display
ADD1
Perspective
Set
Grid: 100 mm
Points
Polygons
Volume
Symmetry
Modes
Cut
Copy
Paste
Undo
W
T
M
(none)

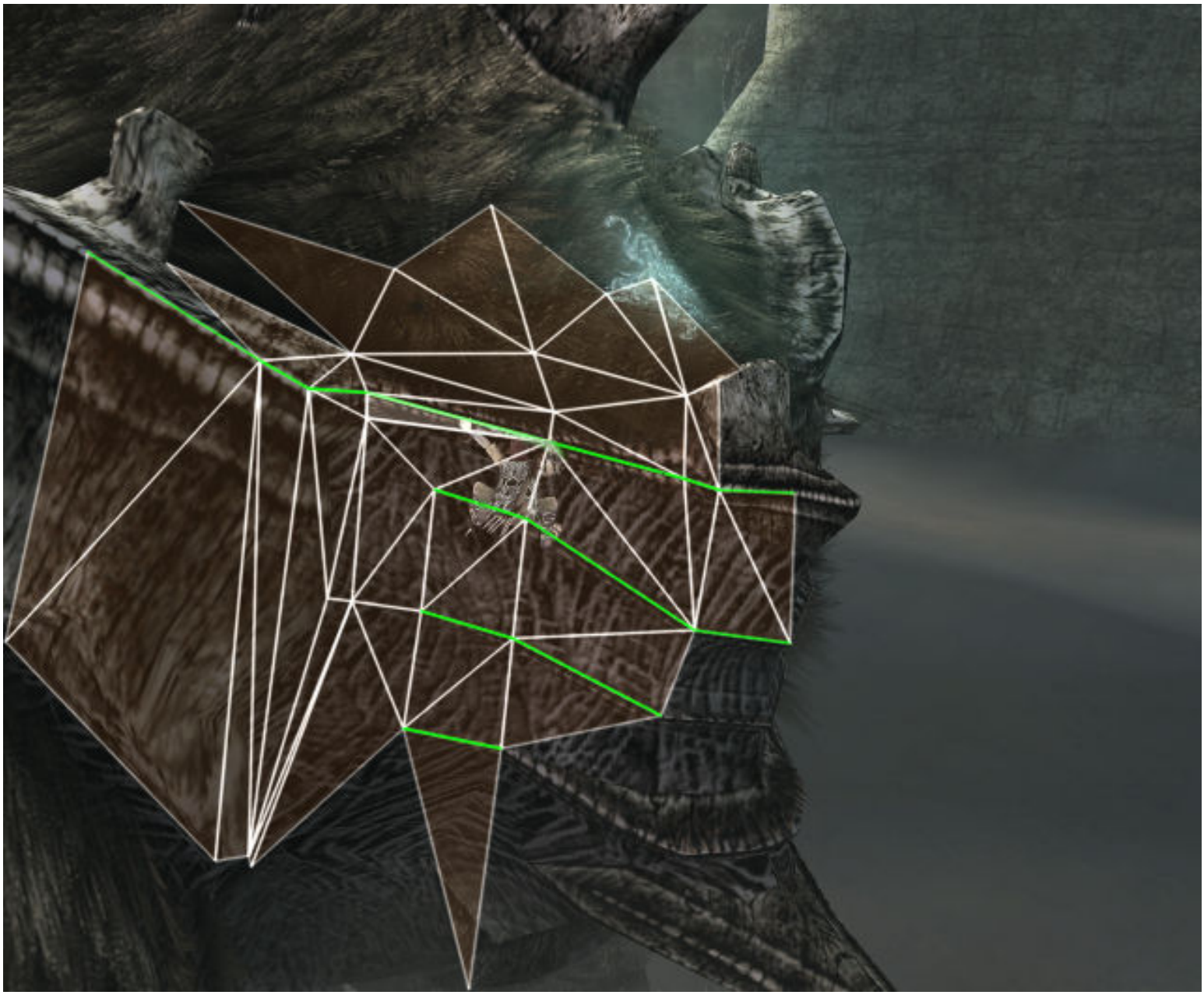


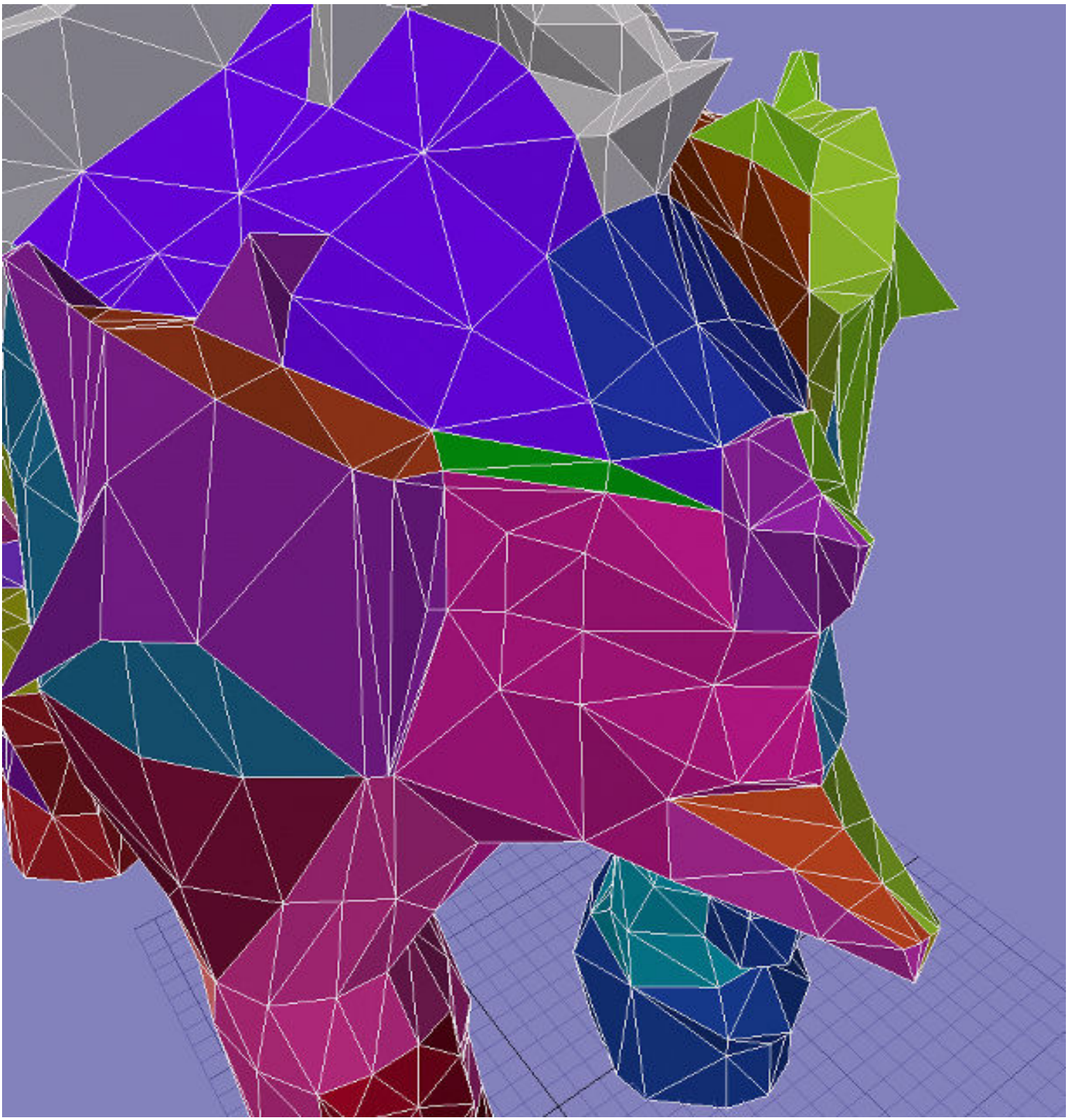


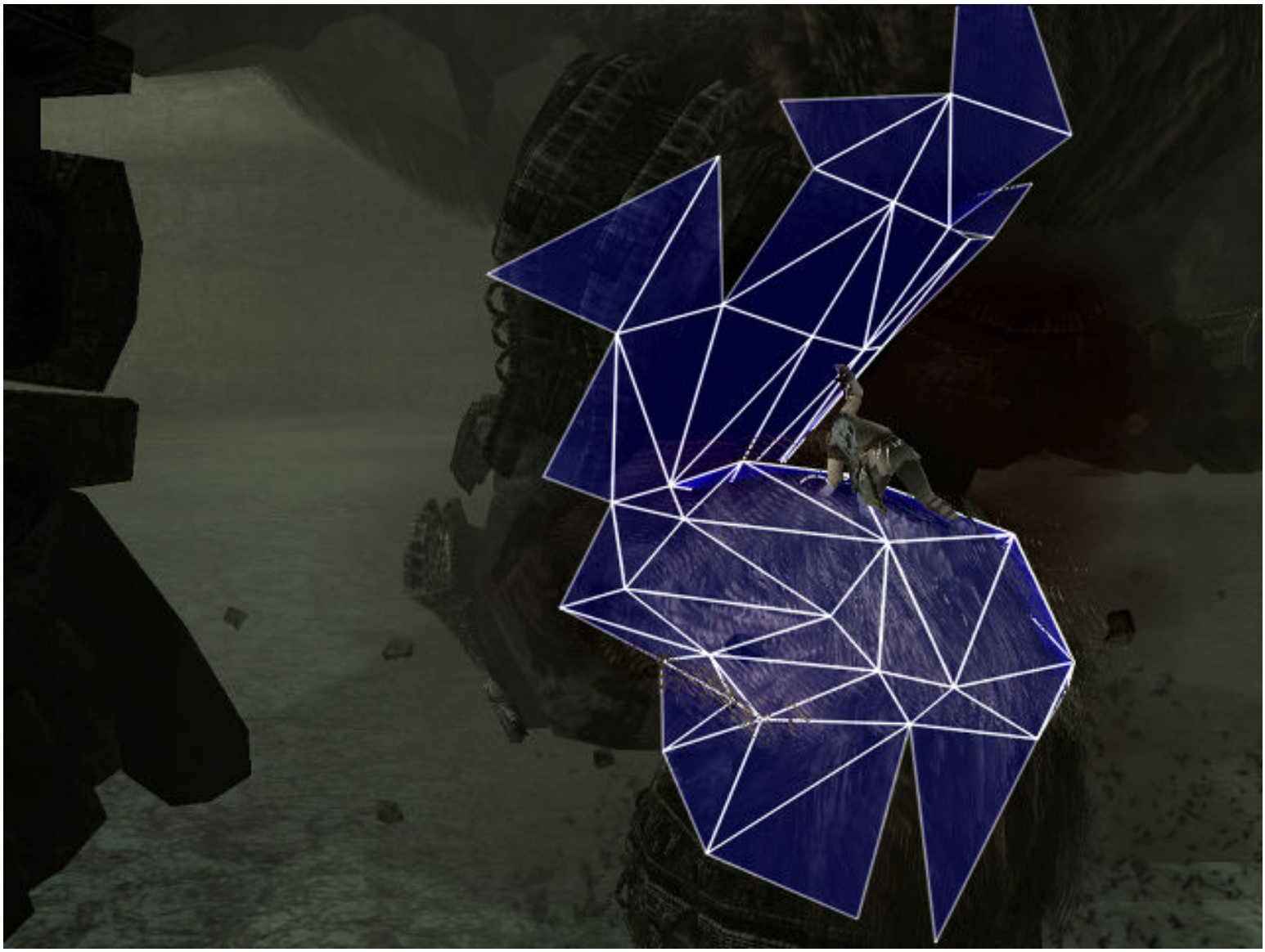


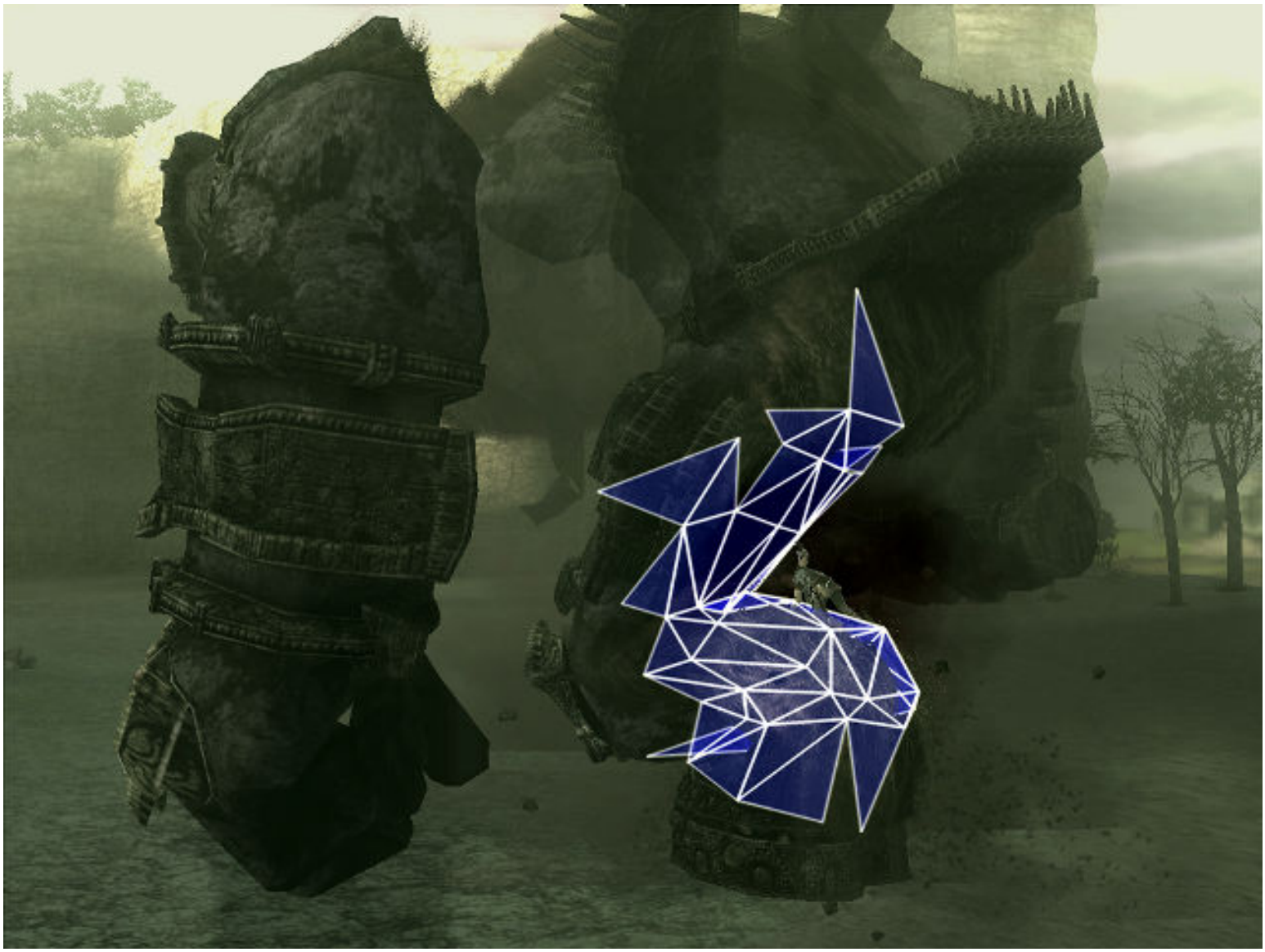


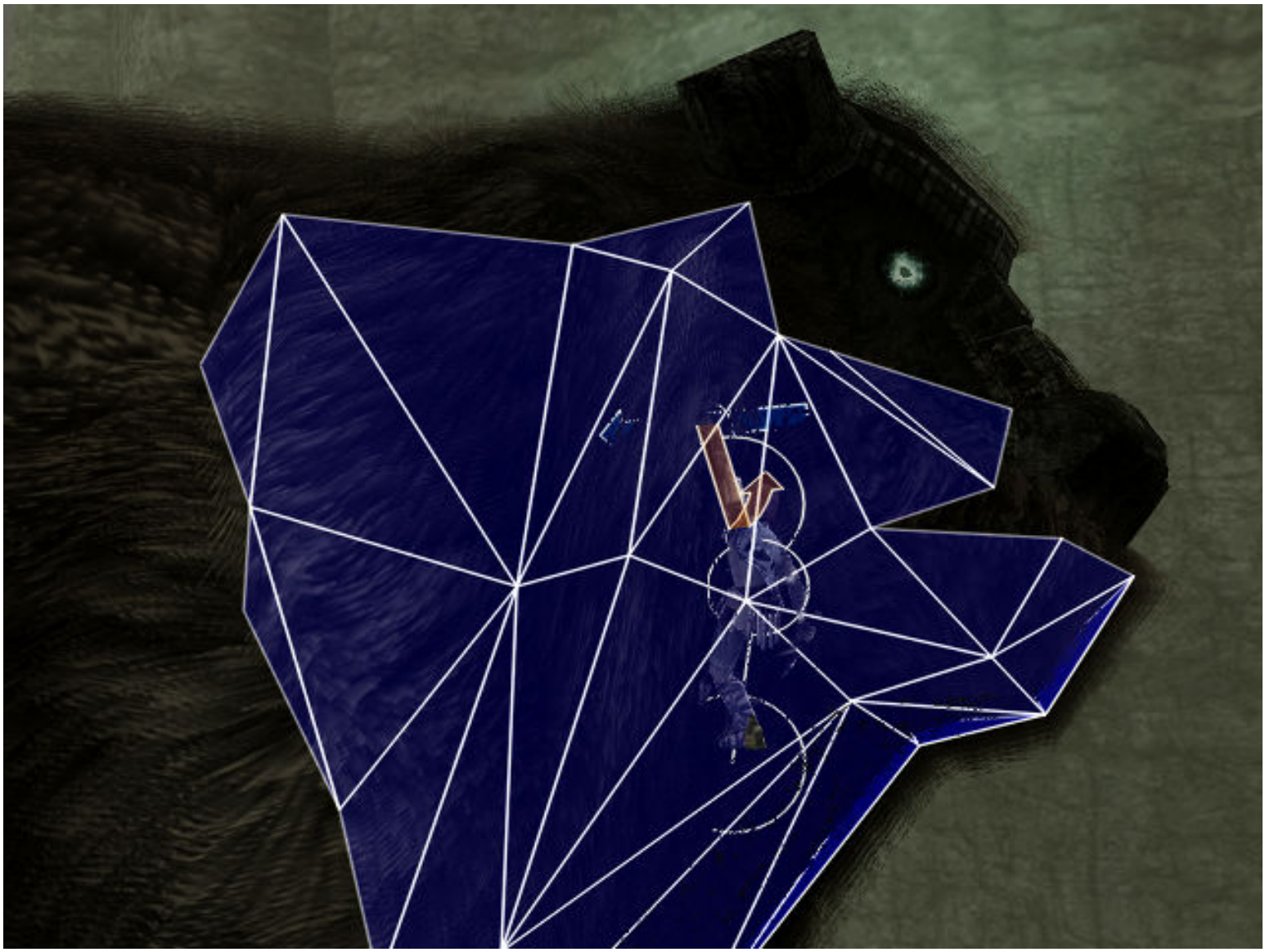


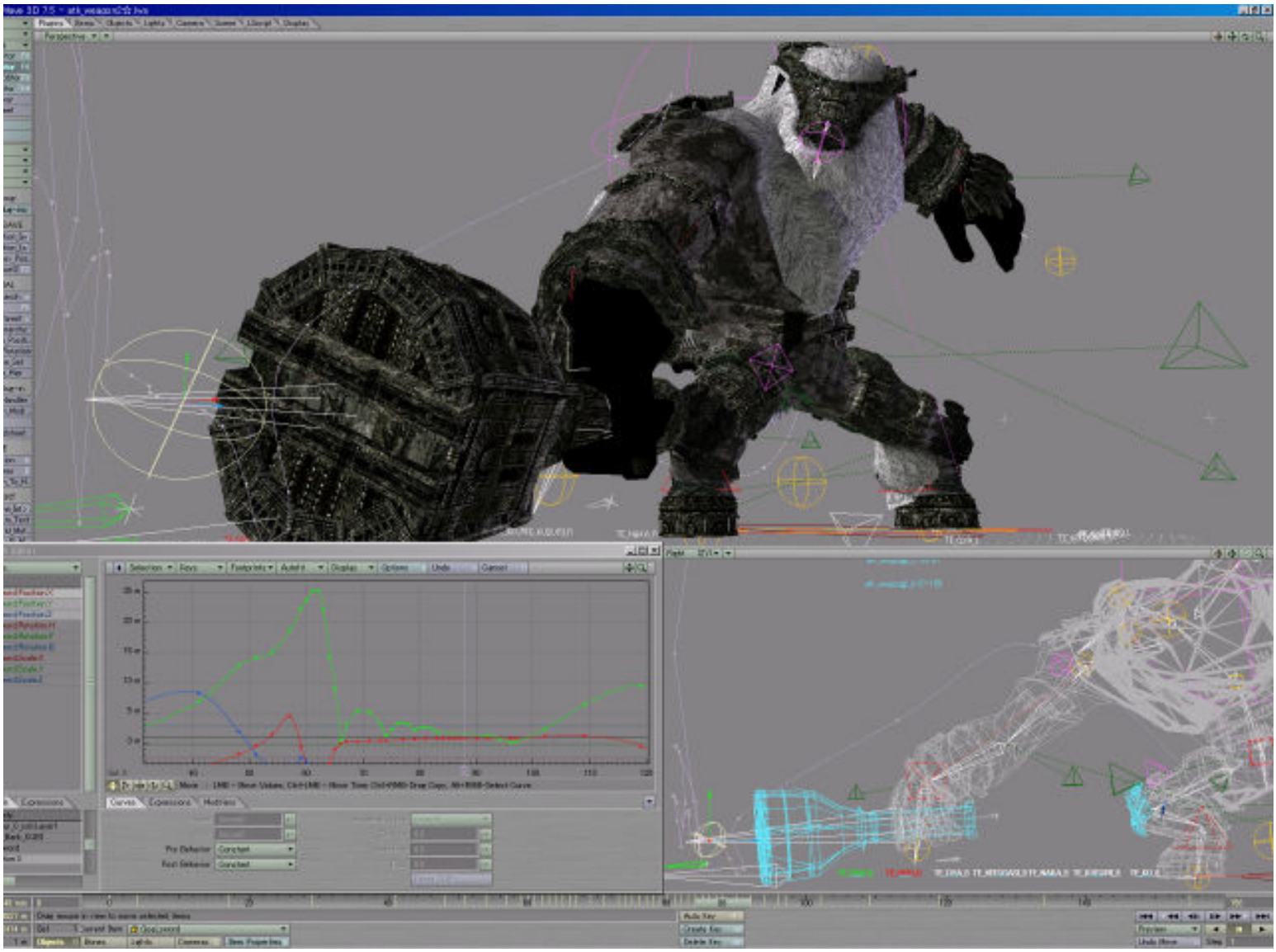


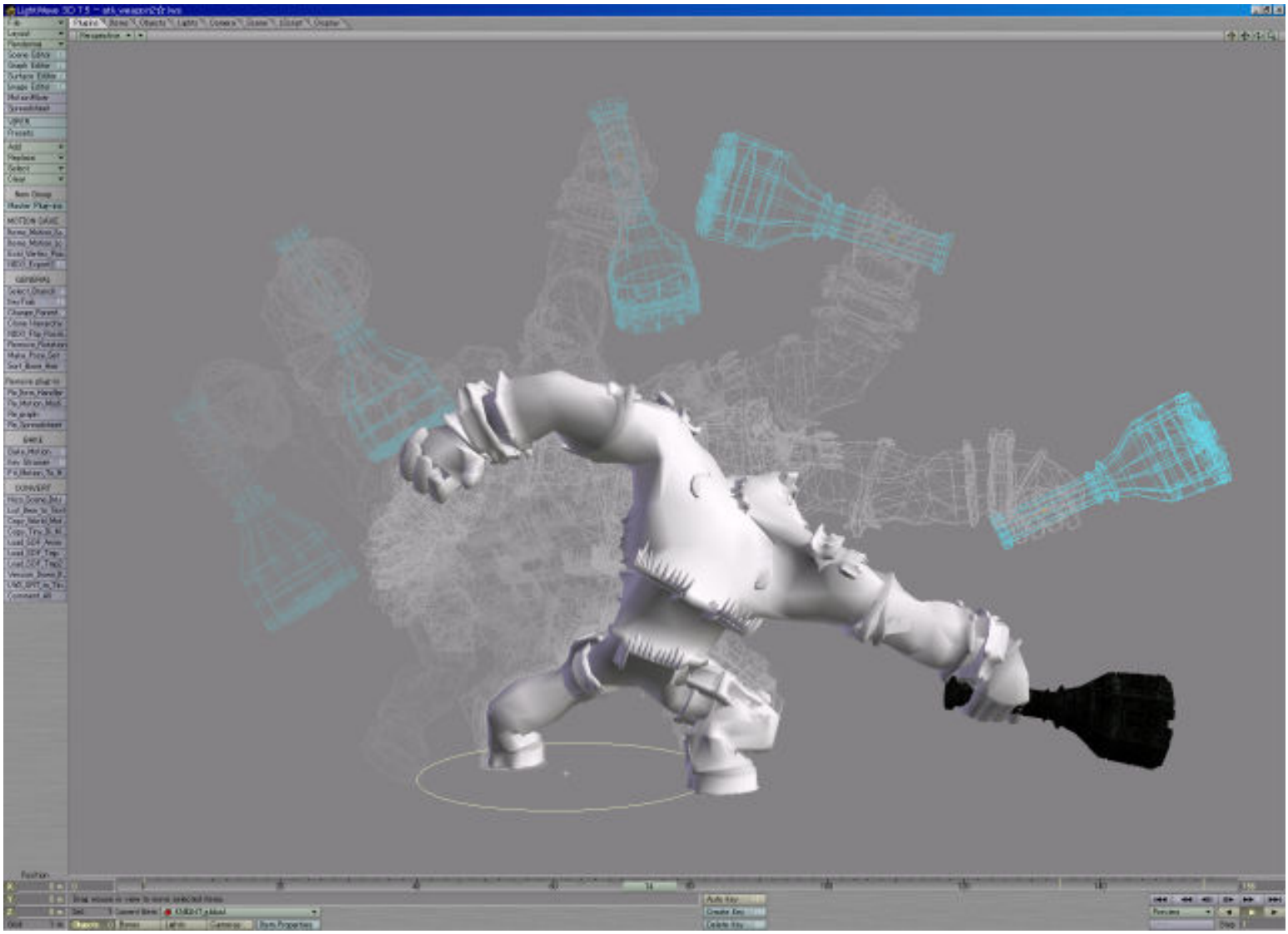


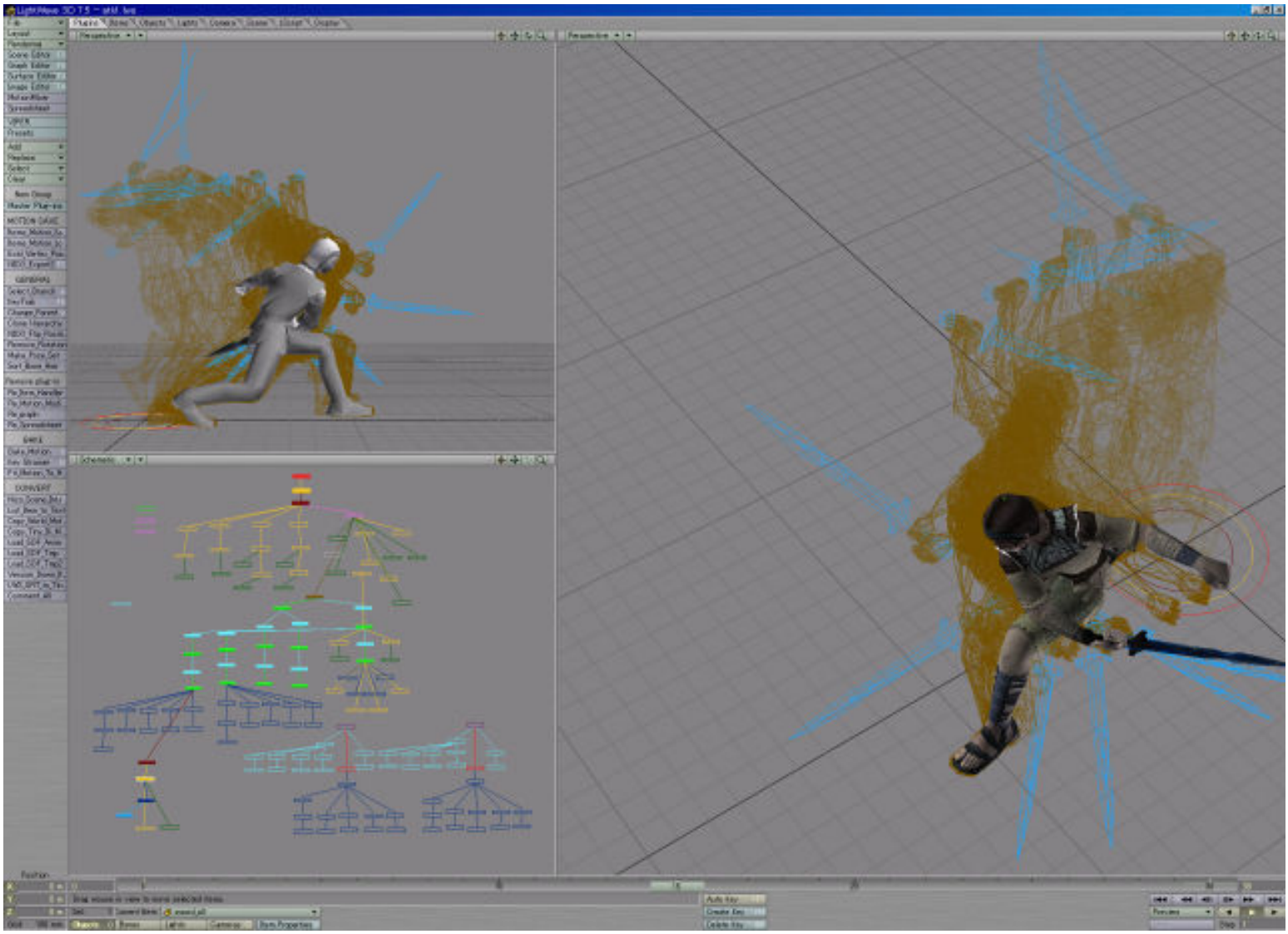


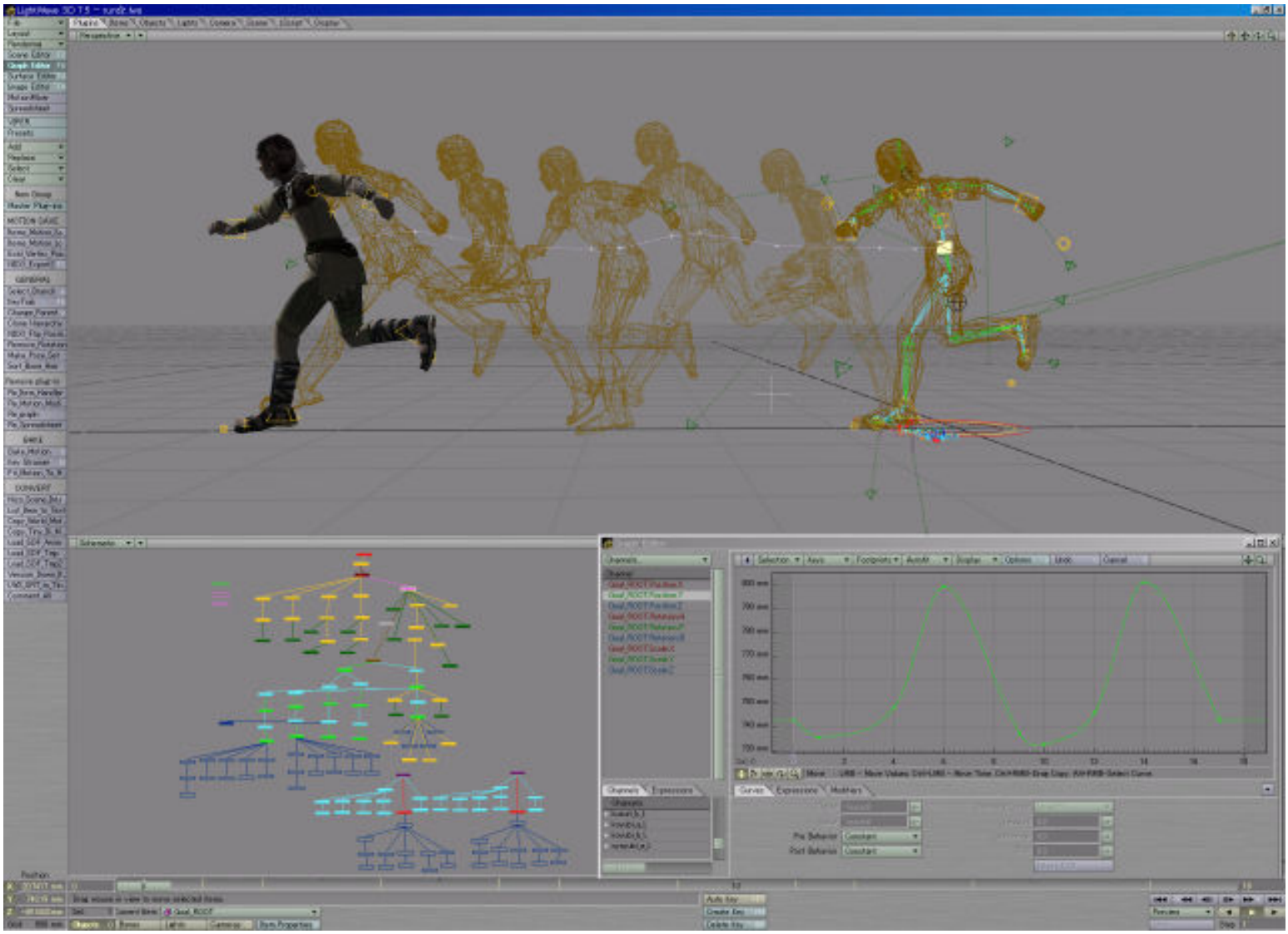


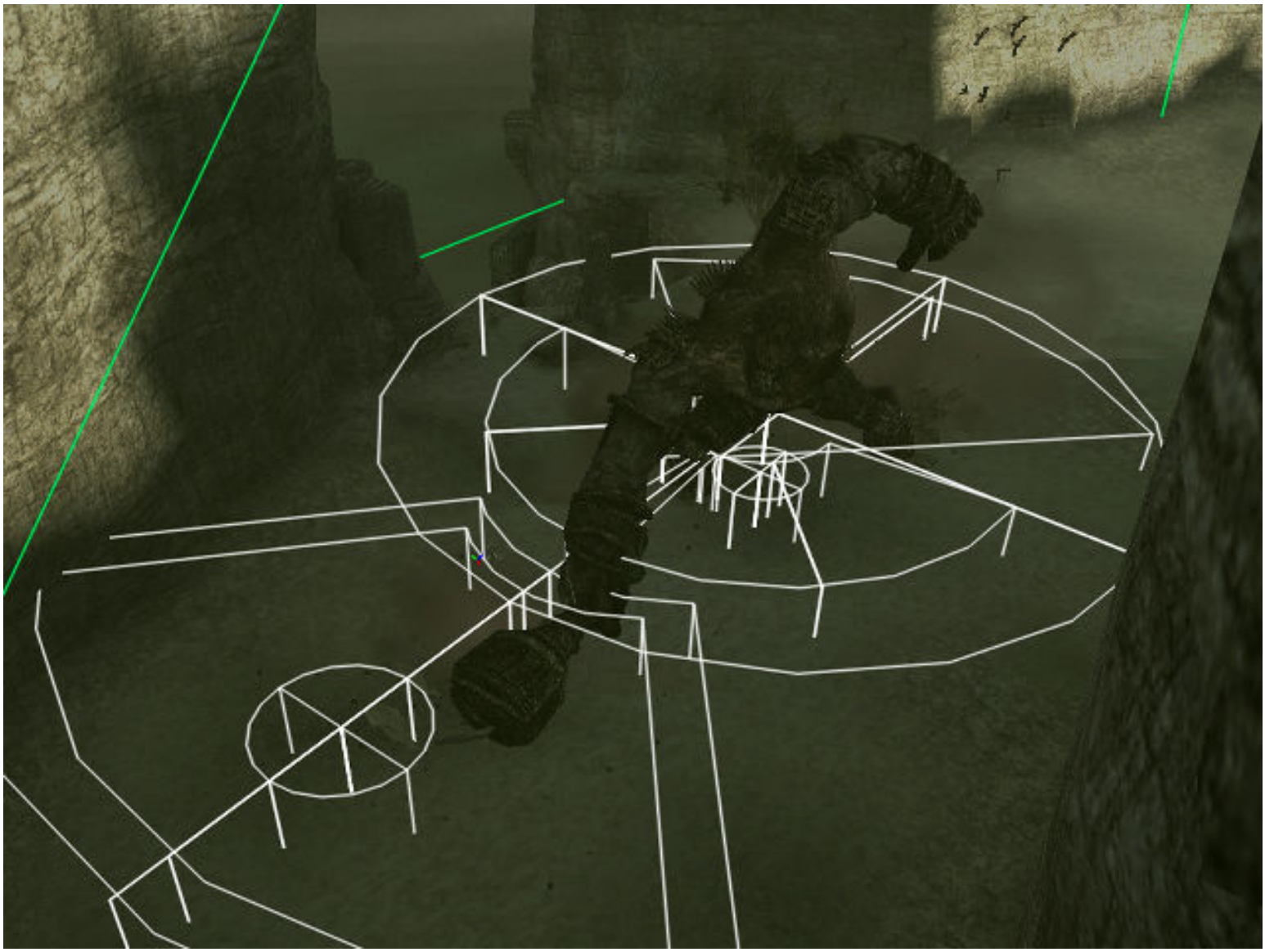


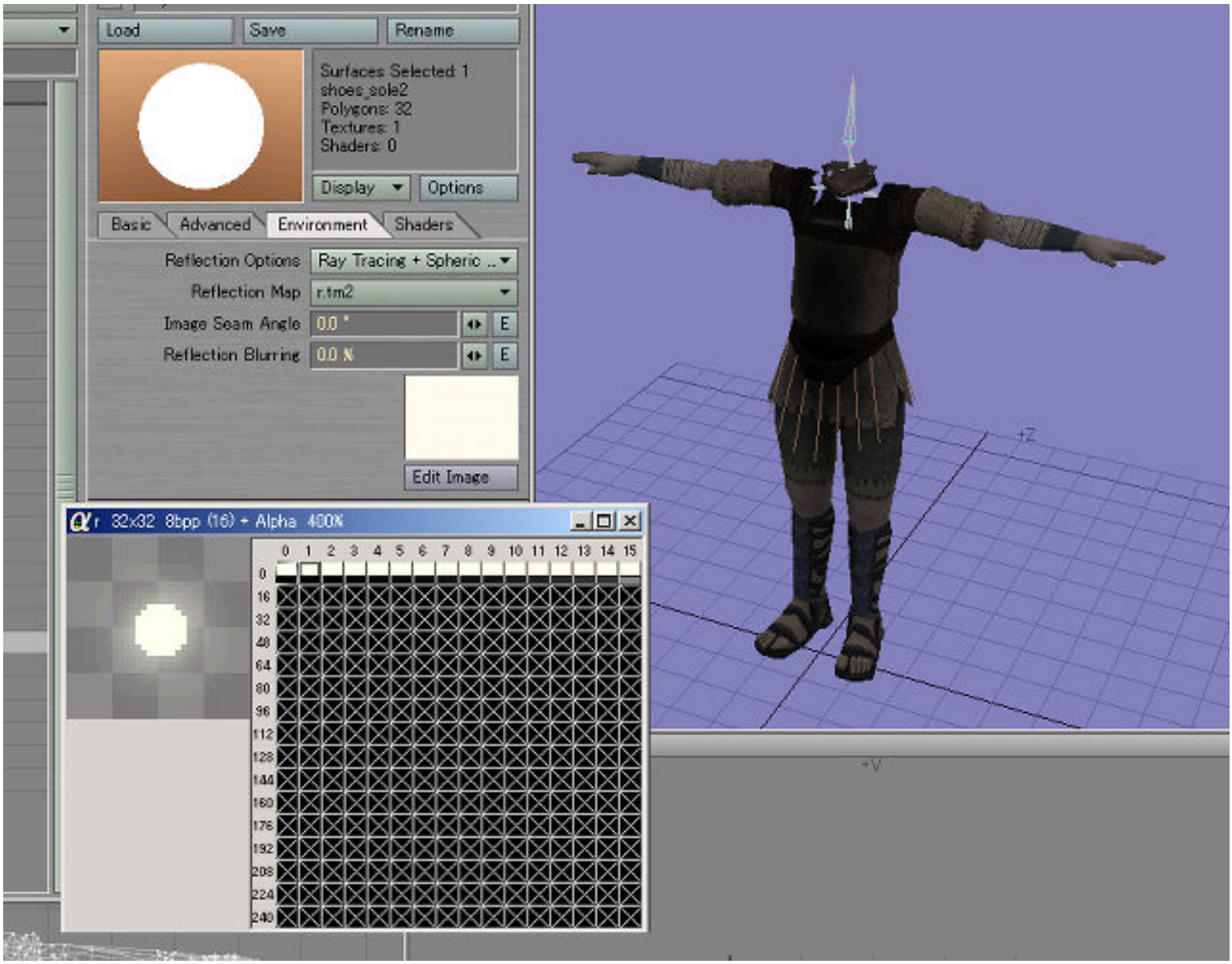











Load
Save
Rename
Surfaces Selected: 1
shoes_sole2
Polygons: 32
Textures: 1
Shaders: 0
Display
Options
Basic
Advanced
Environment
Shaders
Reflection Options
Ray Tracing + Spheric ..
Reflection Map
r.tm2
Image Seam Angle
0.0 °
Reflection Blurring
0.0 %
E
Edit Image
r 32x32 8bpp (16) + Alpha 400%
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0
16
32
48
64
80
96
112
128
144
160
176
192
208
224
240
+Z
+Y