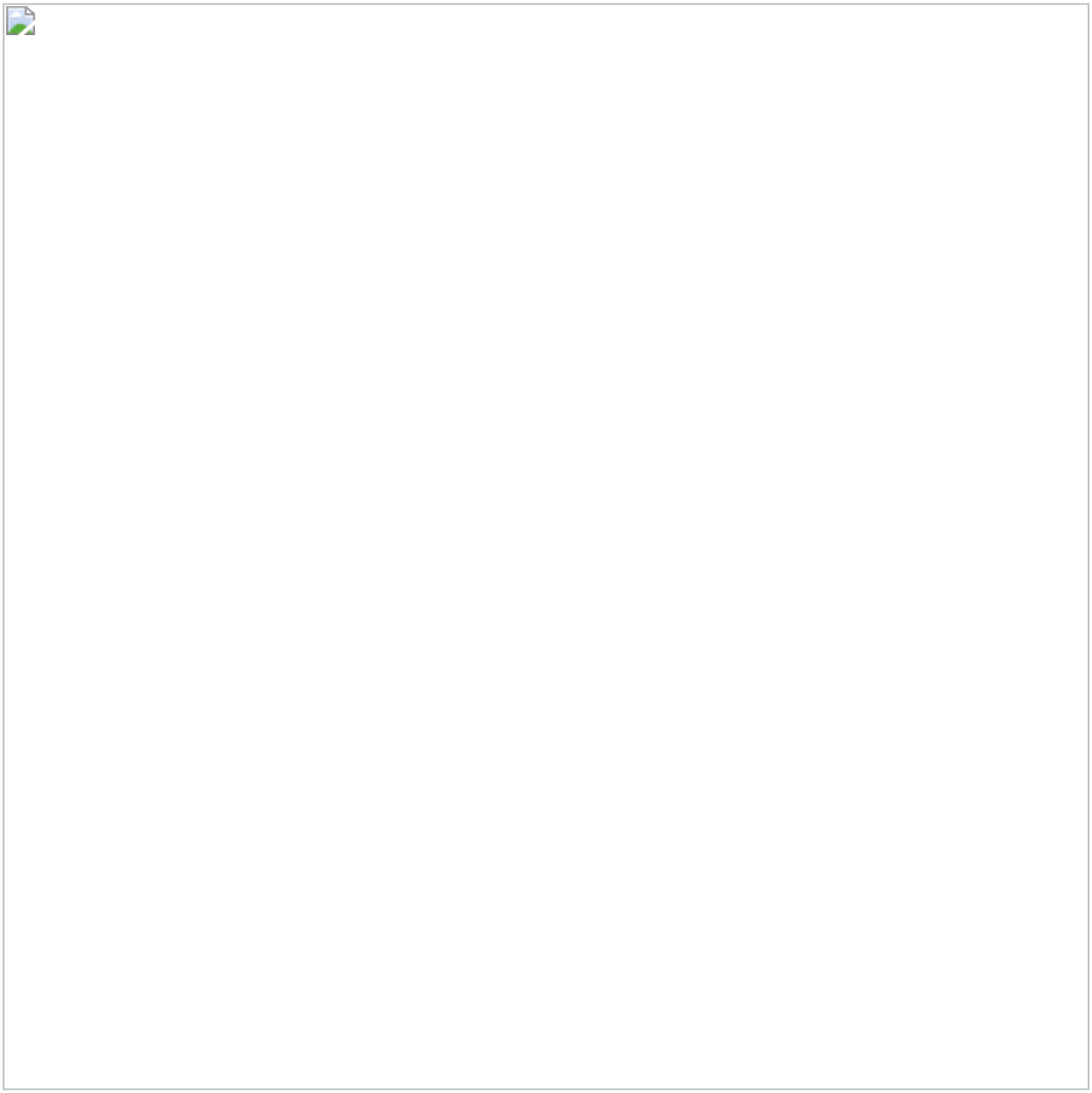
Creating Characters for DOTA 2

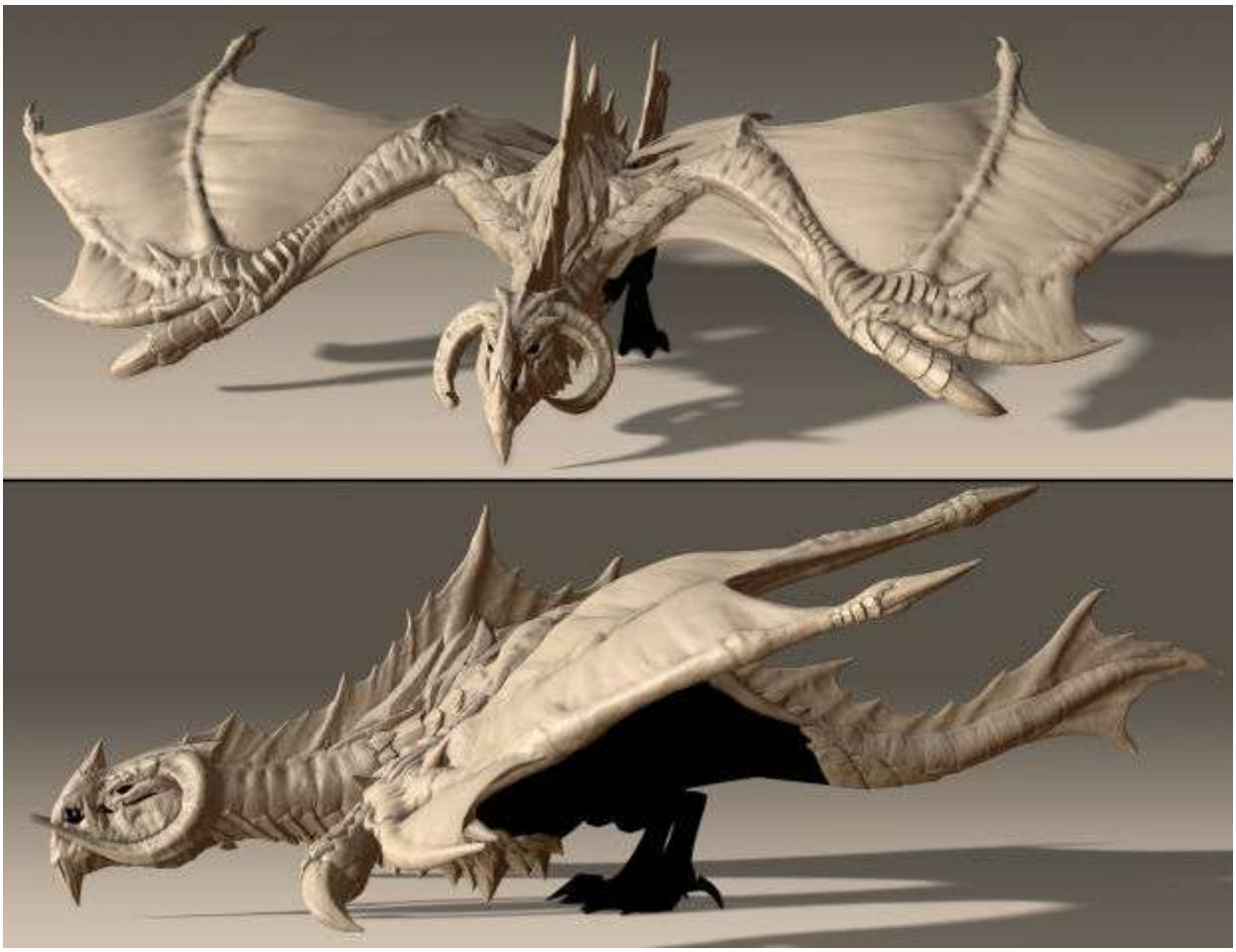
80.lv/articles/creating-characters-for-dota-2

Andrea Orioli February 20, 2017

Italian artist Andrea Orioli discusses the creation of his Dota 2 skins.

3D Character Artist Andrea Orioli talked about the production process behind his Dota 2 characters. Making money with your favorite game might be really easy.





1 of 2

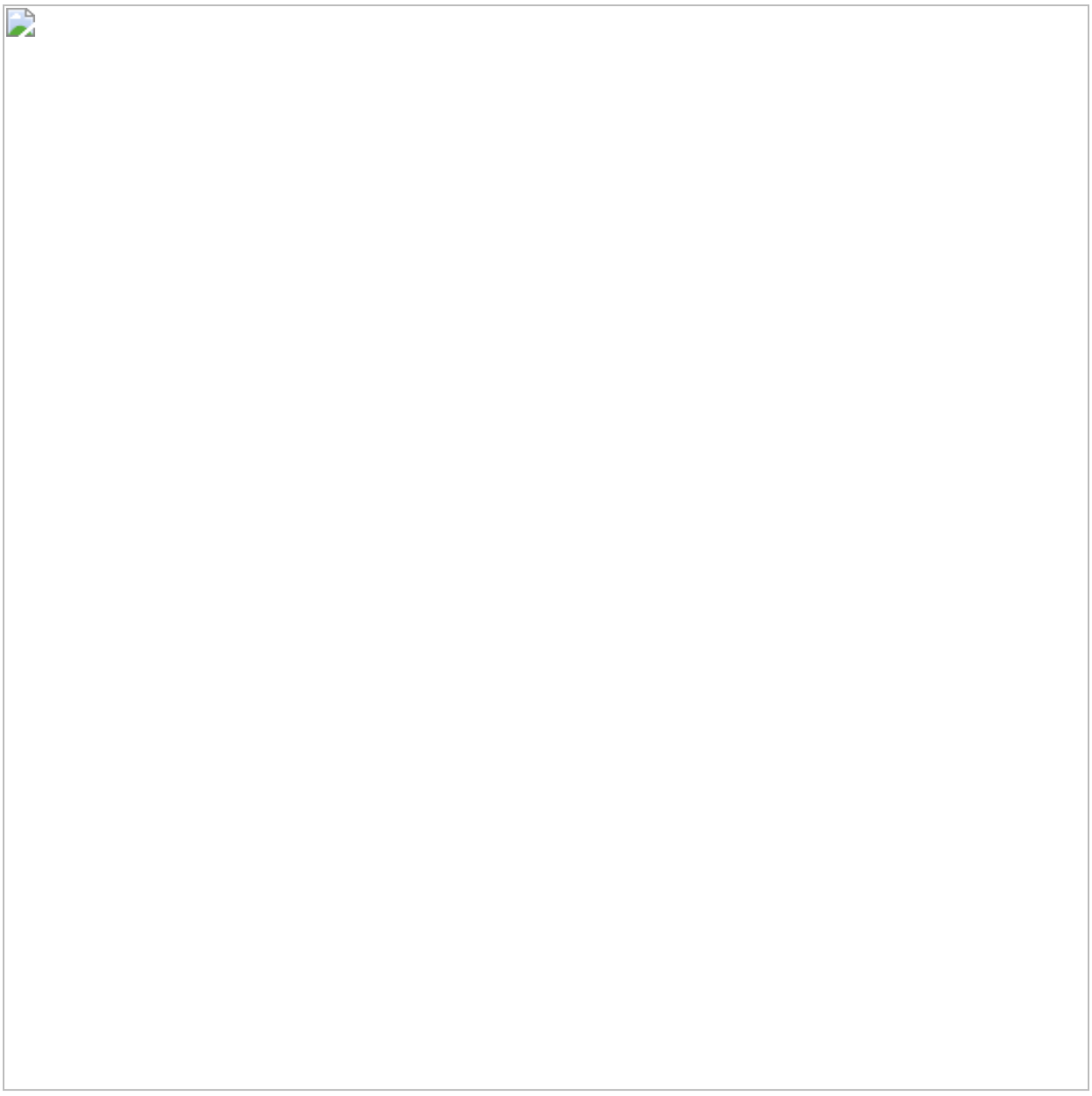
Introduction

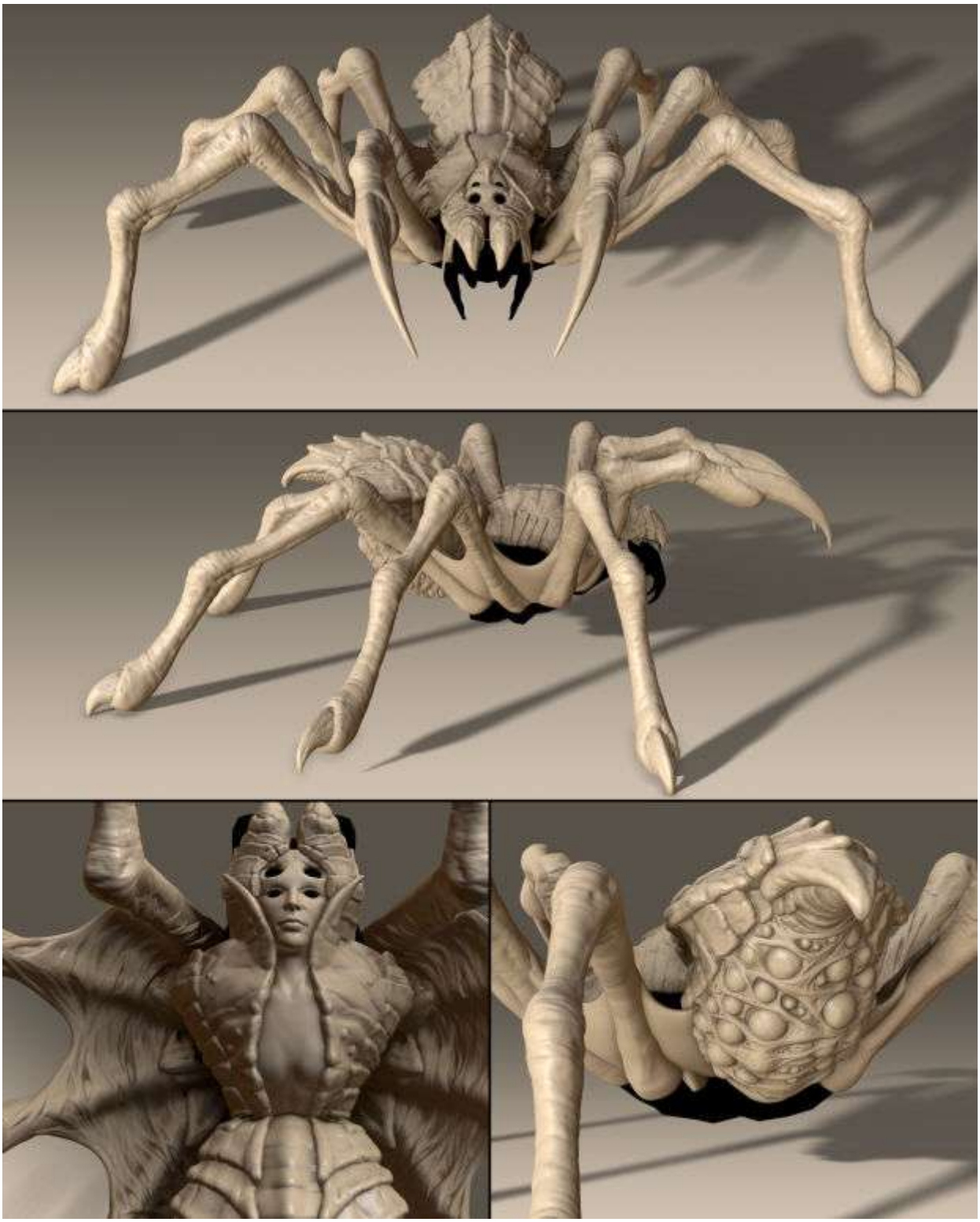
Hi, my name is Andrea, I'm an Italian 3D artist, huge nerd, avid gamer, and still in love with '90s punk-rock music.

1 of 3

1 of 4

It’s hard to define a precise moment in time when I decided “I wanna be a 3D artist for video games!”. I’ve been in love with video games since I had my first Sega Master System at the age of 6, and not much later I developed a love for drawing, so eventually these two passions converged into who I am today and what I do for a living. I still remember how impressed I was by Blizzard cinematics back when I had my first PC in mid-90s and that was sure something that pushed me to try 3D art for the first time with 3D Studio Max. Unfortunately back then in Italy there weren’t really schools that could give me an instruction on this subject, and having an internet connection was something that few could afford, so my only option was to grab a few books and start experimenting.



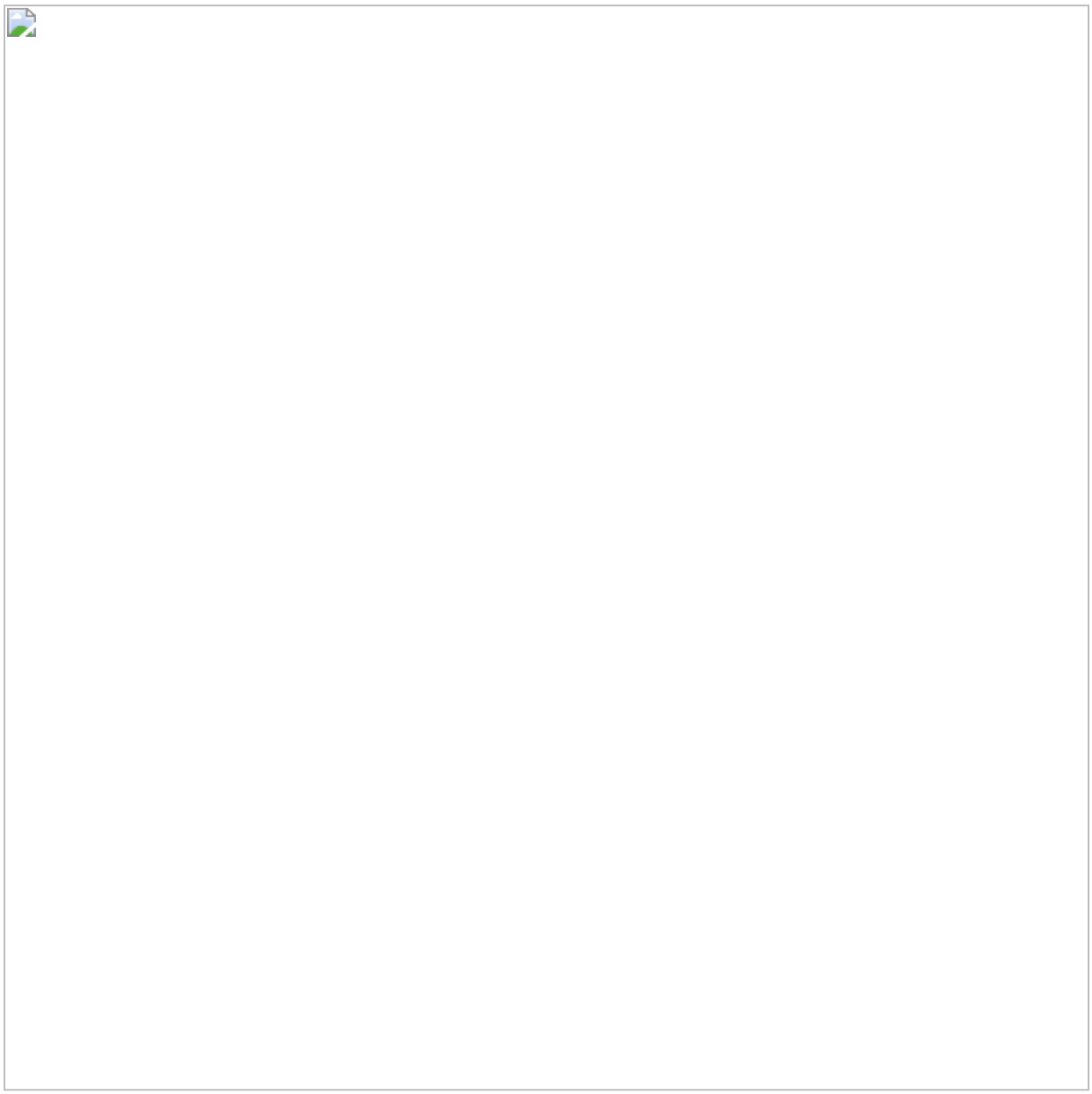


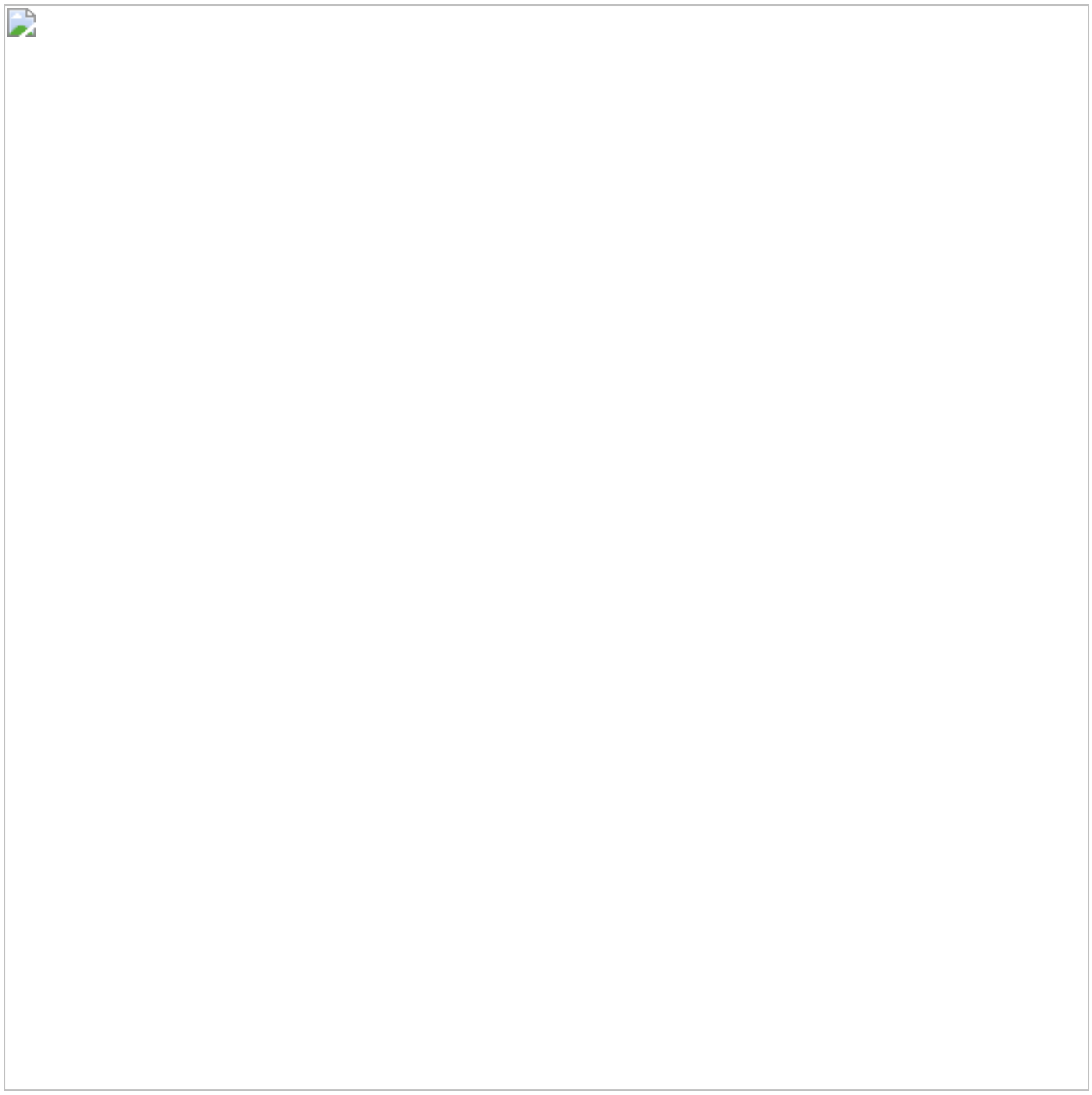
1 of 2

Eventually after few years of trial and error I improved enough to score a job at Milestone which is probably the biggest Italian video game studio, and from there my professional career started. My main focus initially has been hard-surface modeling though, it took several more years for me to start learning character art, and when I finally found out about the Dota 2 Workshop I had the perfect chance to focus on characters full-time.

Dota 2 Content

I first heard of the Dota 2 Workshop at the end of 2012 from a colleague of mine that was very much into the game. I already knew that Valve had created a workshop for Team Fortress 2 where game artists could earn some money from their creations, and I've always been curious to try it out, so the newborn Dota 2 Workshop seemed a great opportunity.





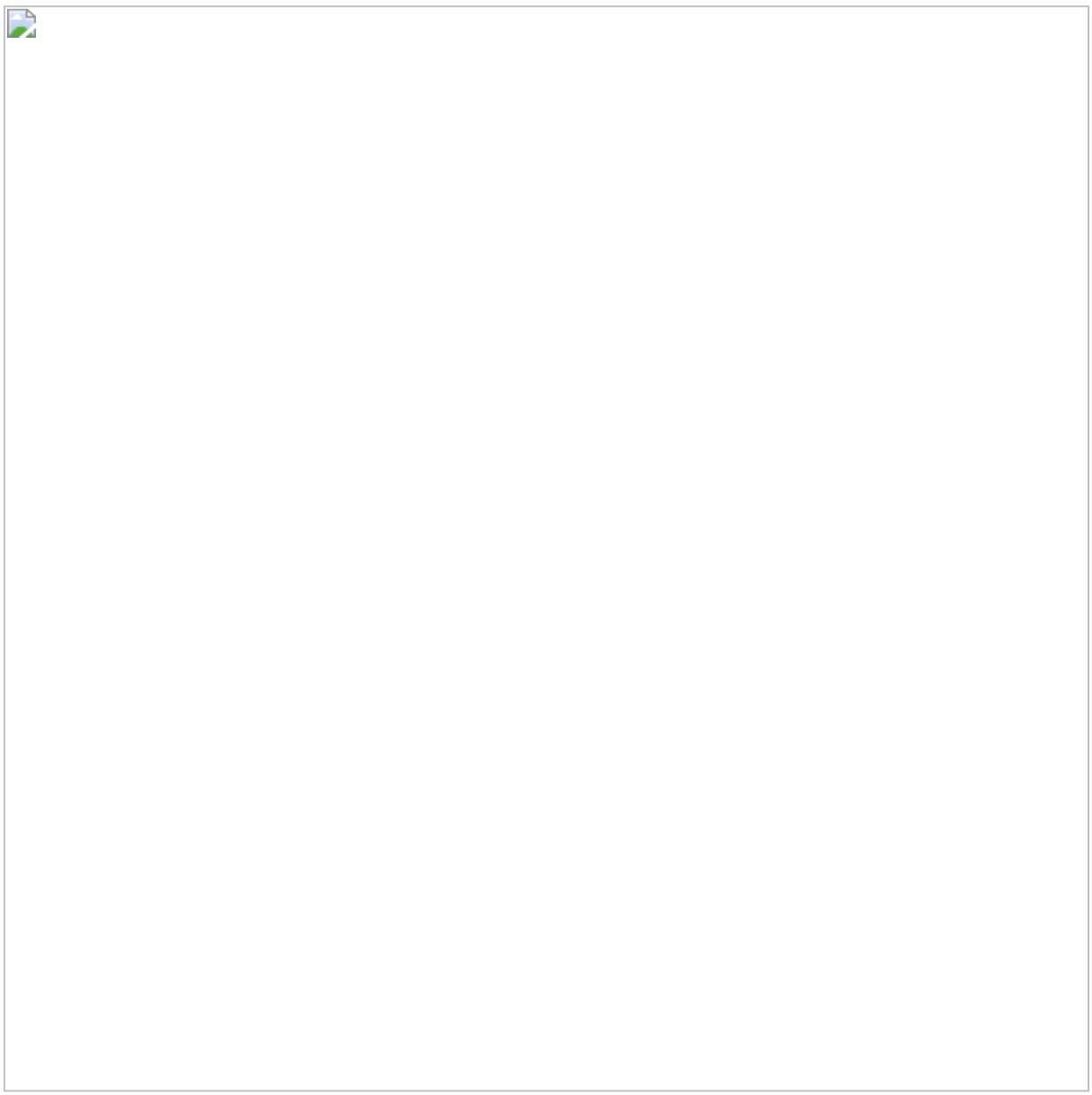
1 of 2

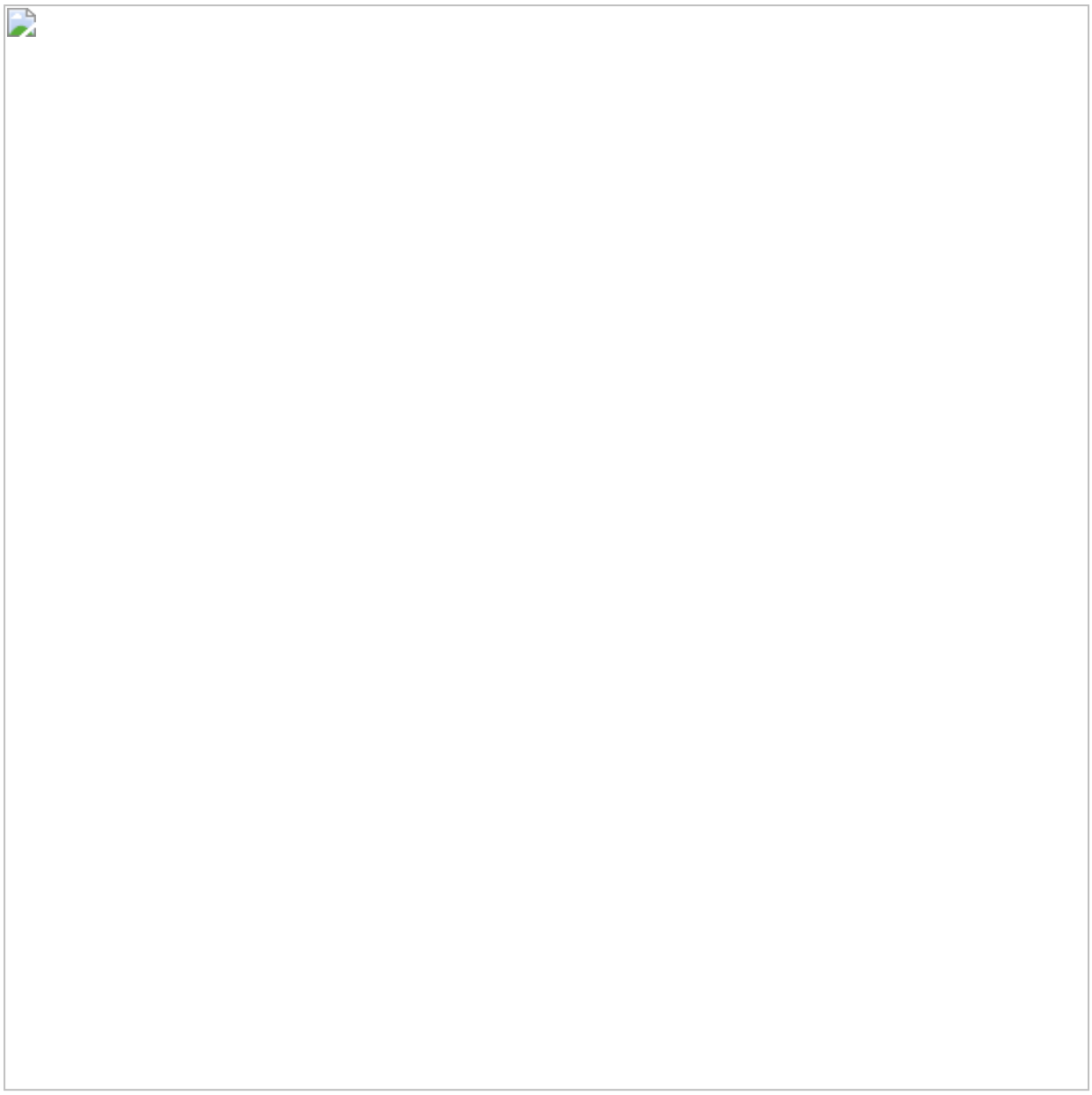
At the beginning I was quite rusty though. I never did video game modding before, and never even worked on a title with a fantasy art style, so I decided to start slow, creating a few simple weapons. I never though I could make a living by selling digital items though, such an idea was too far fetched in my opinion. I started realizing the potential of the Dota 2 Workshop only few months later when one of my items has been accepted. I took me another 6 months from that moment to have the guts and the economic stability to leave my full-time job and dive completely into creating items and sets for Dota.

Challenges

Because of the nature of its gameplay, with a top down camera, Dota 2 items have quite heavy restrictions in both polycount and texture resolution.

The biggest challenge of creating assets for a game like this is to push detail and design as far as I can, while keeping it within the polygon and texture restrictions at the same time. It may sound kinda obvious, but it’s actually quite hard to keep the necessary balance between “awesome” and few thousand polygons with a handful of 256 textures.





1 of 2

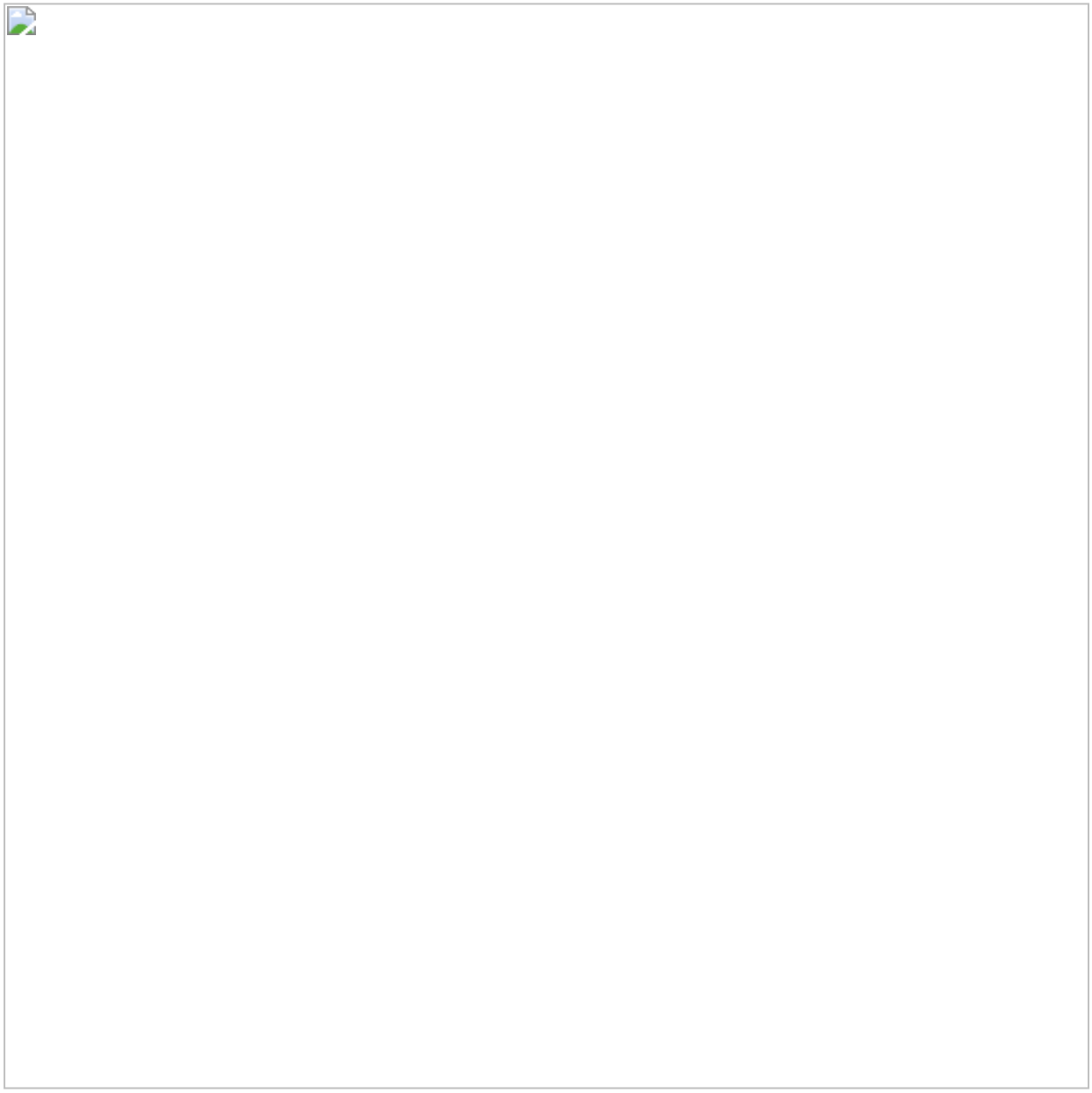
For example, in my early creations, I would start sculpting a model and adding all sort of interesting visual elements to it without worrying too much of what was to come after. I then found myself in the situation where I had to go back and simplify the silhouette or make certain items symmetrical instead of unique, because I didn't have enough poly budget or texture resolution. Having a rough plan of the whole pipeline and what you can or can't afford to try is hard but it can be learned with experience and eventually it will save a lot of potential rework time.

Low Poly Modeling

Usually I start my pipeline of production in Zbrush, with or without a defined concept. It's always nice to work with a drawn concept made by some great concept artist, but in the case I start a solo project, Zbrush is perfect for prototyping and try different ideas quickly.

I usually will spend several days deciding what looks good and what needs changes, until I reach a somewhat final stage of design, with all the elements in place. From there I start detailing things and adding minors touch-ups until I have a nice looking final sculpt.

If I've done my planning job right, the subsequent retopo pass in 3ds max is usually pretty smooth and fast. During retopo I always try to make an efficient use of my triangles budget, dedicating more geometry to silhouettes, and leaving surface details and shapes to the normal map.

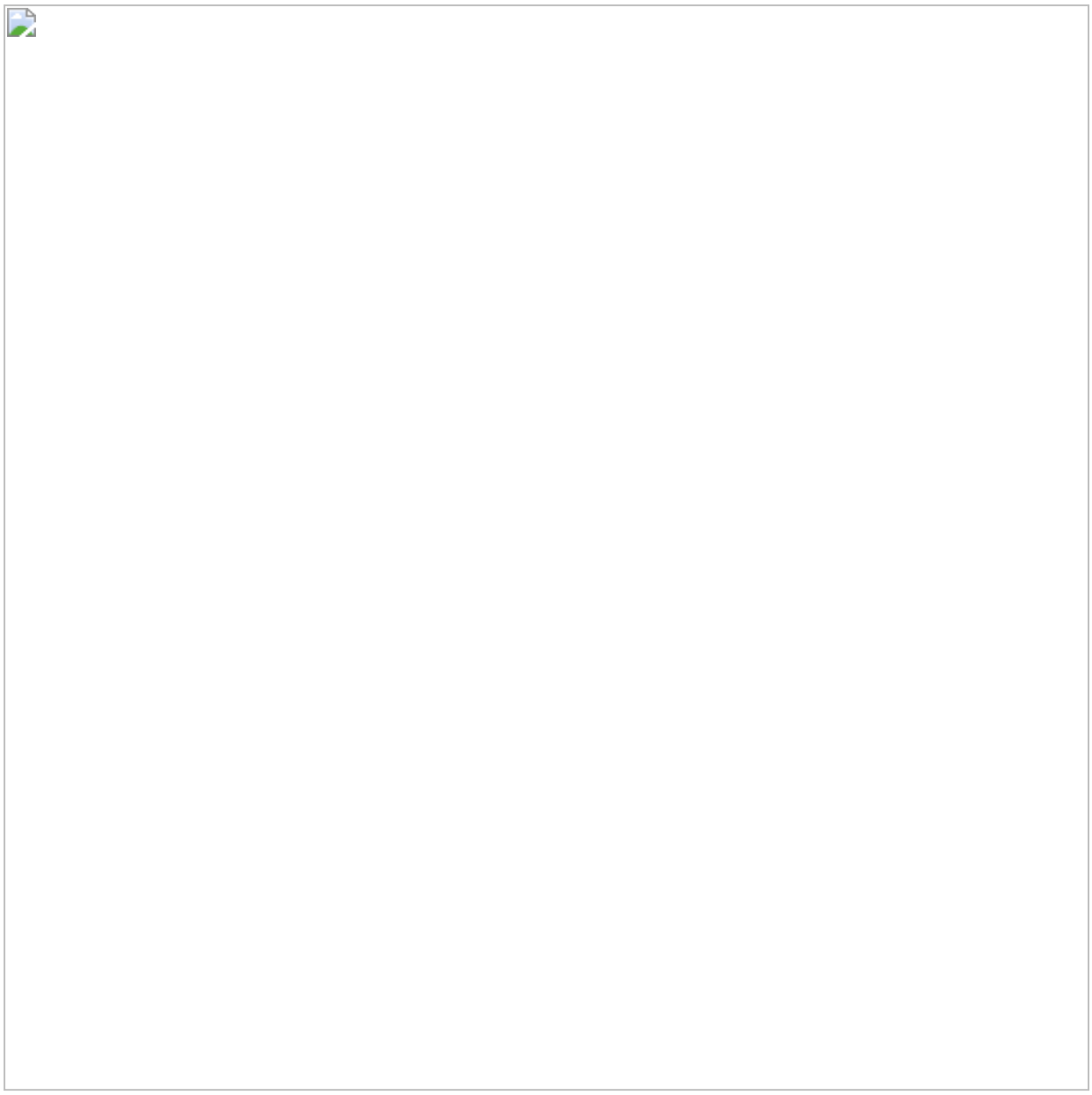


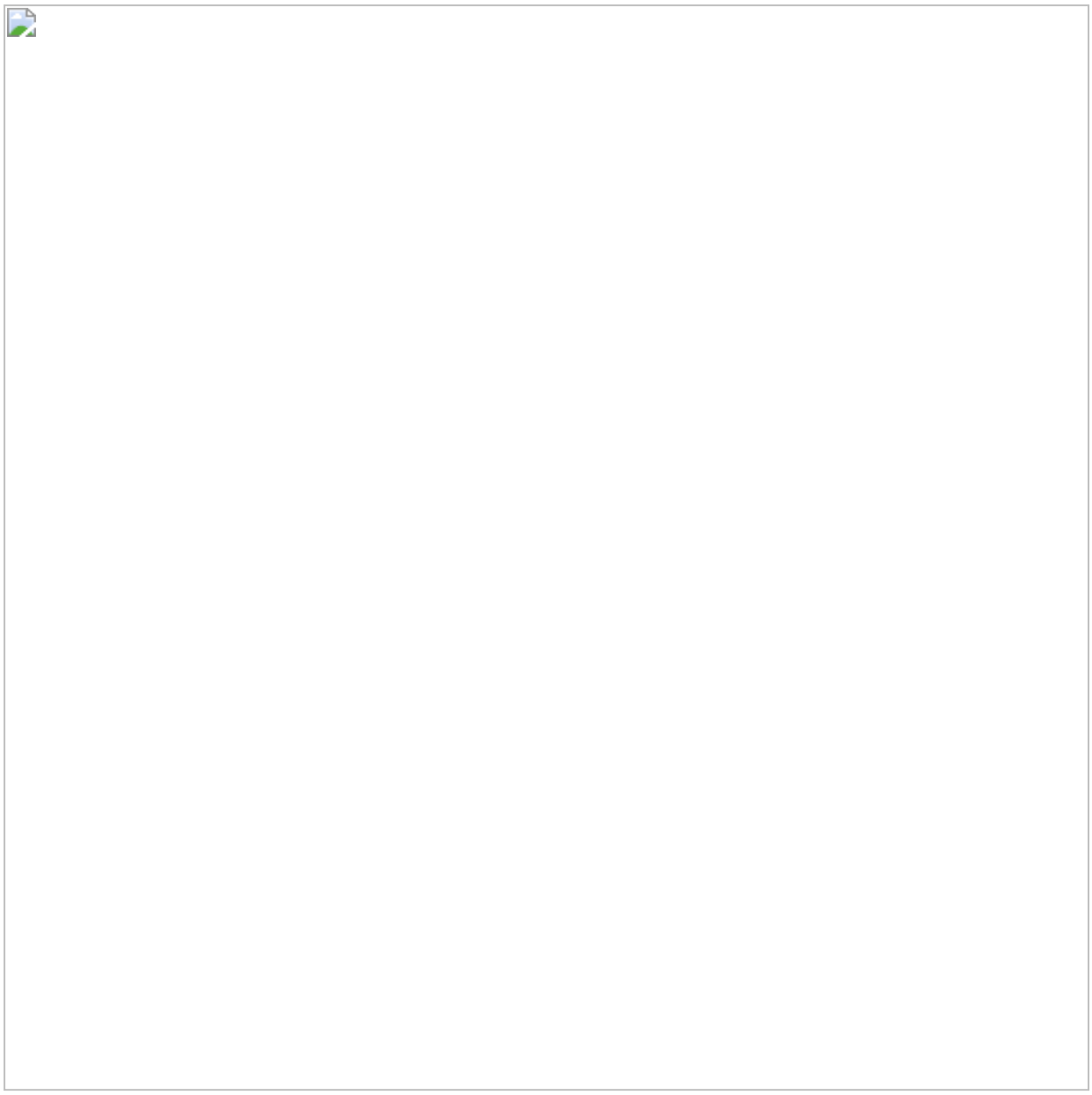
1 of 2

Materials

For my texturing process I rely a lot on the detail of my Zbrush sculpt and my baked textures.

Before starting to paint colors I go through the process of creating a base texture that allow me to balance information of curvature, directional light and ambient occlusion. Then on the top of this base map I start painting color, usually in Mudbox or 3D-Coat. This is probably my favorite phase of the whole creation process, because I finally begin to see the final result come to life and I can play with lots of color combinations and gradients.





1 of 2

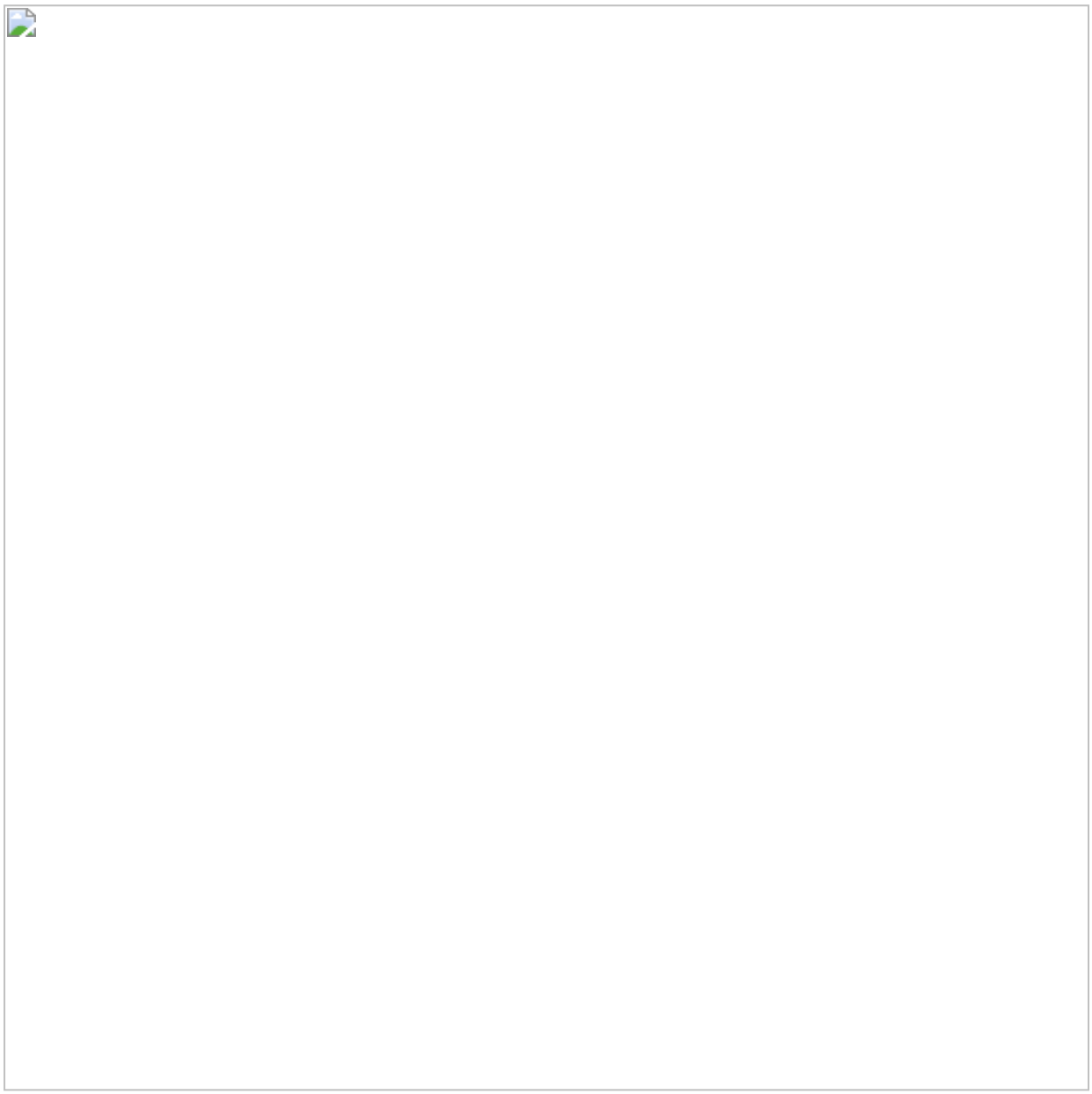
Making Money

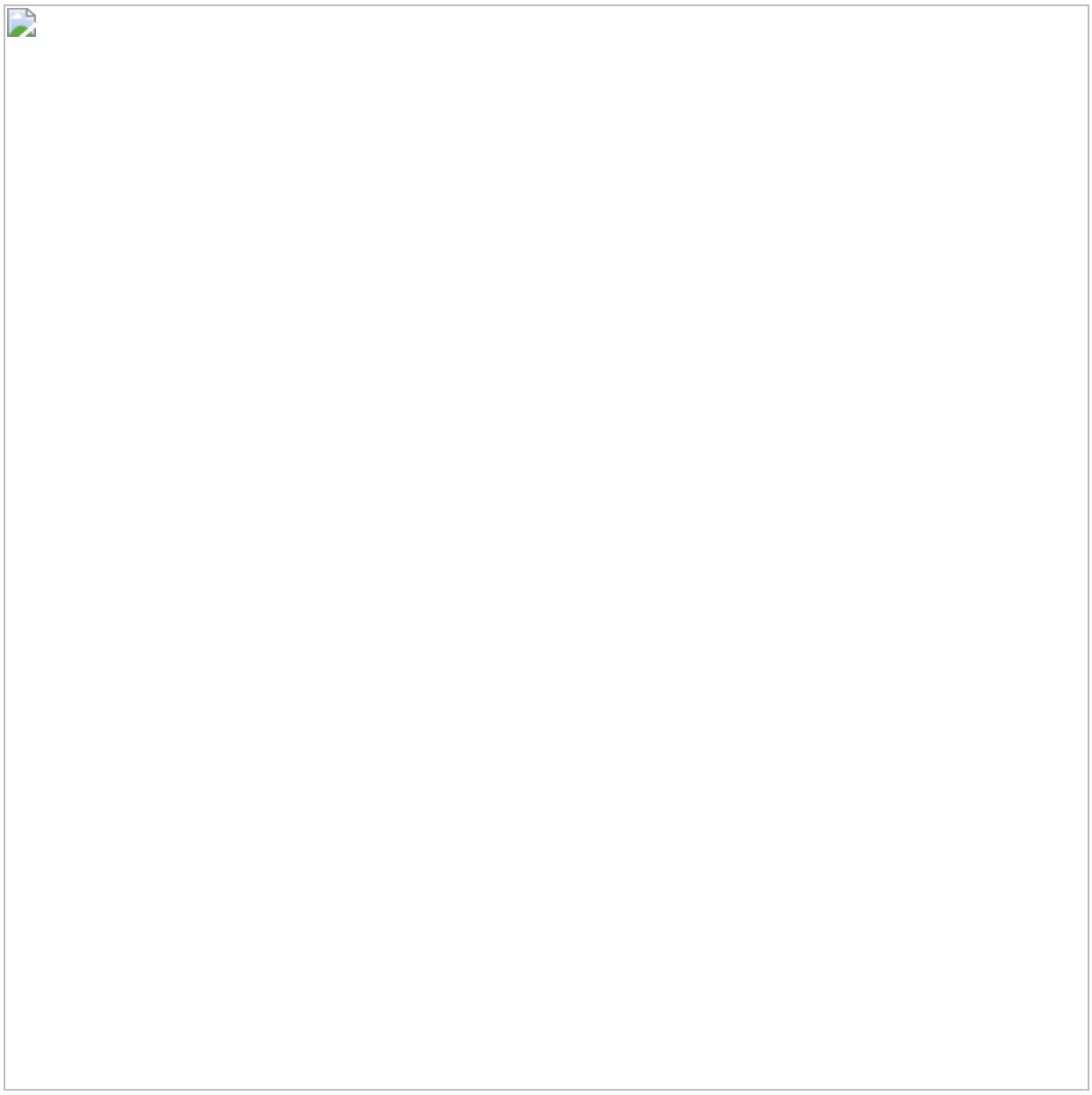
The Dota 2 Workshop works as a sort of marketplace, where every artist can upload his creations.

All these models can be voted by whoever has a Steam account, and eventually Valve will pick the best creations to be accepted into the game. Valve grants a percentage on the returns of every unit sold to the authors of an accepted asset. This percentage was 25% at the beginning, but it has been adjusted several times through the years. Now I kinda lost track of what it is precisely to be honest.

For me working for Dota 2 has certainly been an amazing experience so far, both artistically and financially, it allowed me to become financially quite indipendent even without having to rely on a full-time job.

I don’t think it’s ever unreasonable to try to work for the Dota 2 Workshop. The doors are always open for whoever wants to try, although the times have changed from the early days when the Workshop started. The environment has become increasingly competitive year after year, filled with crazy talented artists, so trying to be noticed and eventually get a model accepted is getting harder and harder even for seasoned creators.

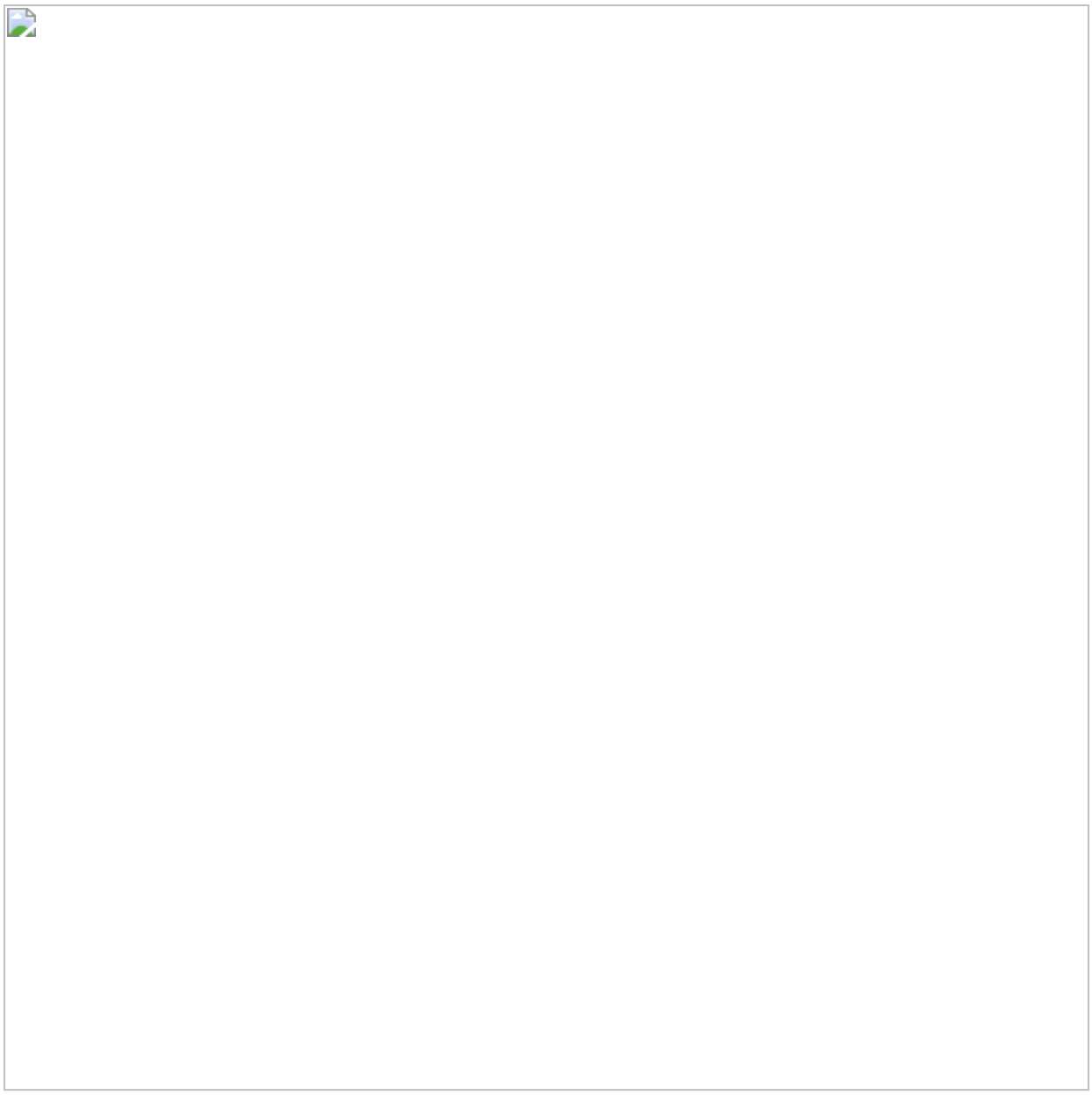


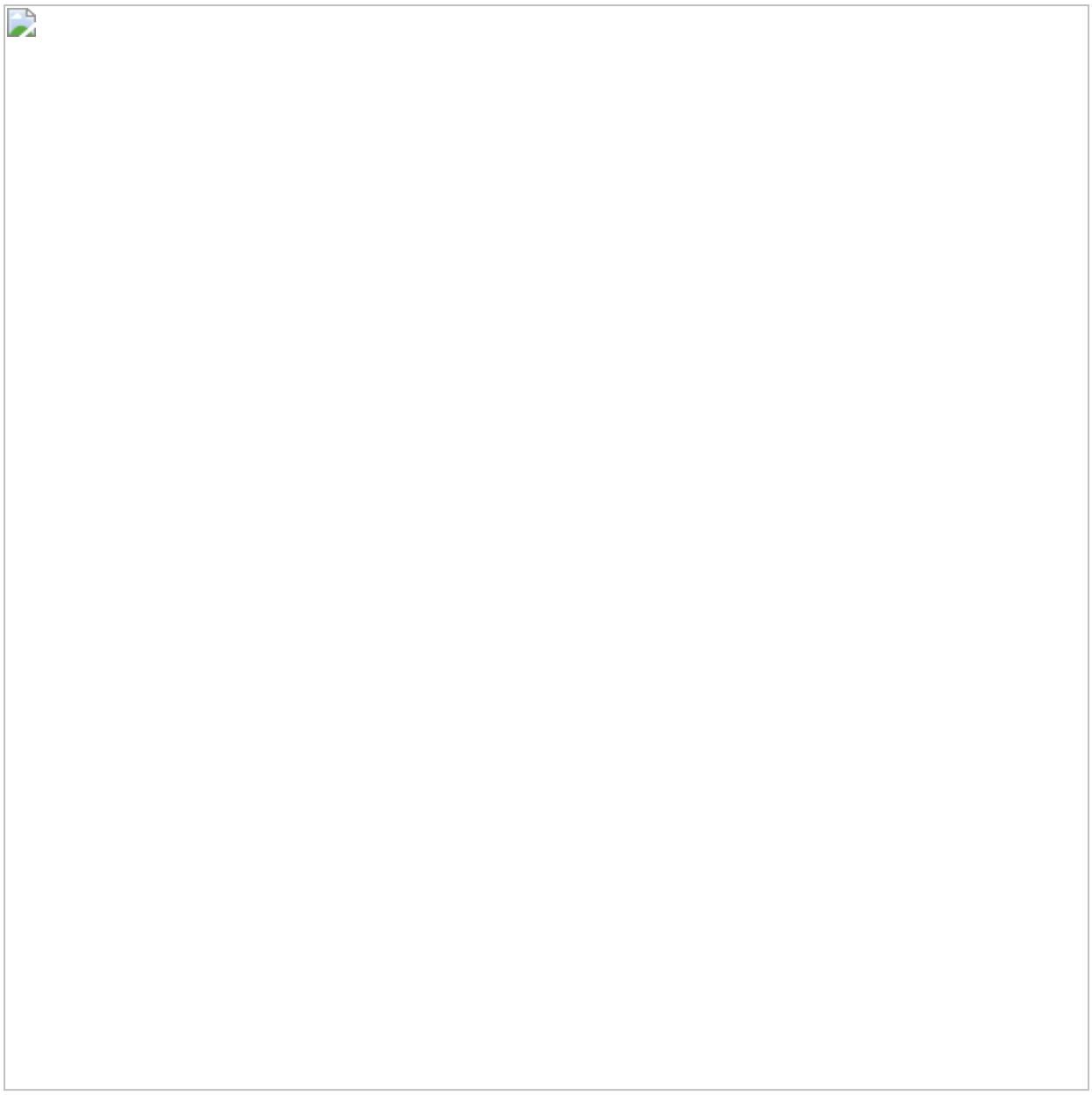


1 of 2

By nature trying a career in the Workshop is quite risky, cause you can’t be certain if and when one of your creation will be accepted in the game, so it’s wise to transition into it slowly while keeping your daily job as a safety net.

I think that it’s not unreasonable to consider the chance of making a living with UGC economies. Besides Dota 2, other games have embraced this philosophy, and I hope many more will in the future. Unity and Unreal Engine also offer quite interesting platforms for artists who want to create content for video games and sell them to other developers, so the options of choice are quite many.





1 of 2

I have some projects for Dota that I want to finish in the future months, and in the meanwhile I’m starting to look around in search of what could be my next adventure.

Andrea Orioli, 3D Character Artist

Interview conducted by Kirill Tokarev.

Designing a Stylized Creature in ZBrush, Blender & Marmoset Toolbag

80.lv/articles/designing-a-stylized-creature-in-zbrush-blender-marmoset-toolbag

Francisque Facon August 22, 2023

Francisque Facon has told us about the workflow behind the Peter Fishmouther project, explaining how ZBrush, Marmoset Toolbag, Blender, and 3ds Max were used to create the stylized creature and its hand-painted textures.

Watch Video At: https://youtu.be/DIgtkWGI9FQ

Introduction

Hi, everyone! I'm Francisque Facon, a French Character Artist who just graduated from Artside School's 3D character art program, and now I'm ready to dive into the industry!

I grew up playing video games, and I have always been very passionate about them. So, when it came to the point where I had to choose a career path, I decided to try playing around with 3D and fell in love with making characters in ZBrush. Initially, I enrolled in a school, but after two years, I made the decision to switch to online classes.

During this 2-year period, I discovered Marc Brunet's hand-paint tutorial. I made my first sword and character in this style. That's when my obsession with hand-painted textures came in. Since then, I've made most of my projects using this workflow. I also played with PBR mixed with handpainted workflow on my "Sharp Shooter" and "Rusty Sword" projects.

The Peter Fishmouther Project

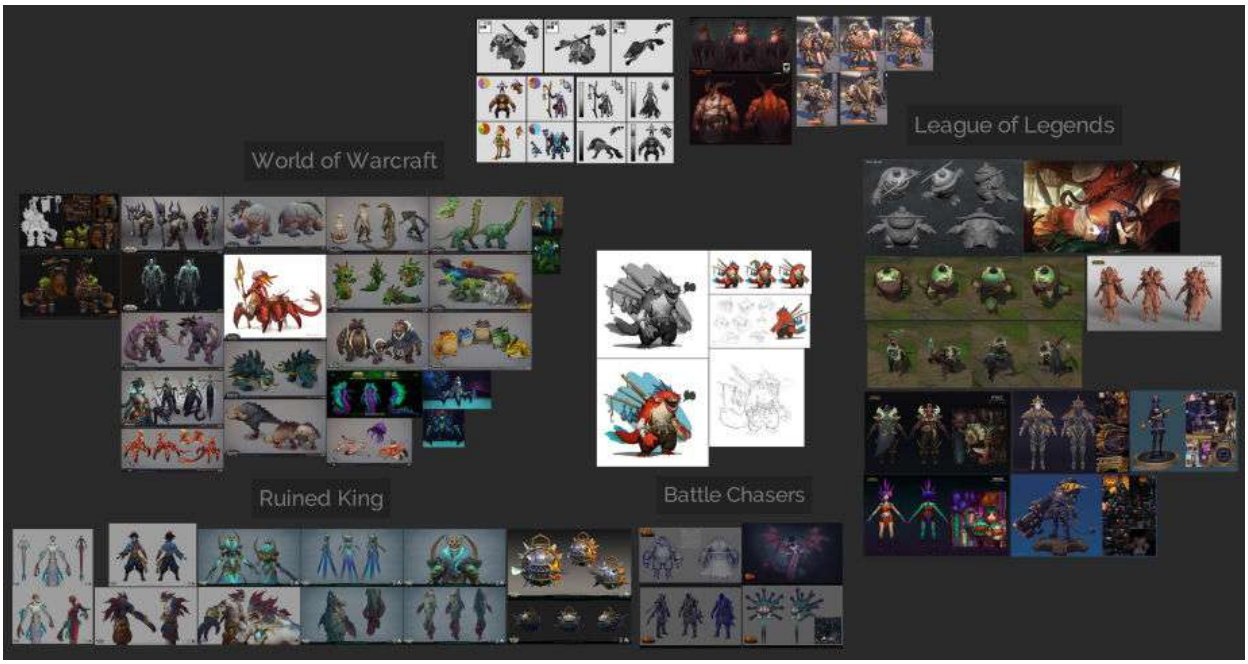
This project was done in a class with Marion Volpe as a teacher. I had a task to make a stylized creature. I chose the amazing "Peter Fishmouther" concept by Labros Panousis as a reference.

After some time using a PBR workflow earlier during the year, I wanted to go back to a more "diffuse only" approach on this project because it was always super fun to work with unlit textures. When I was painting that way, I didn't feel restrained by my Normal Map, so I could go crazy and add details wherever I like.

References

References are super important, especially with a big project like this one. My main references were League of Legends, World of Warcraft, and Ruined King. Those games had an amazing unlit art style, and they came first when I was looking for hand-painted references.

Here's a simplified version of my PureRef board:



Sculpting

The first thing to do was a rough blocking of everything. I tried to stay very low poly and keep the different parts of my character separated, so I could focus on what was important. When everything was in place, I merged my SubTools, used ZBrush's DynaMesh, and did a first pass of cleaning. Once I was happy with the result, I could add some details to have a decent supporting topology.

As you can see, the final sculpt is quite simple. This workflow didn't require it to be very detailed, since I would not be baking a Normal Map, and because I was going for something quite low poly on this project, I preferred to keep the small details (such as the scales) for texturing. That way I wouldn't lose too much time on the sculpting part.

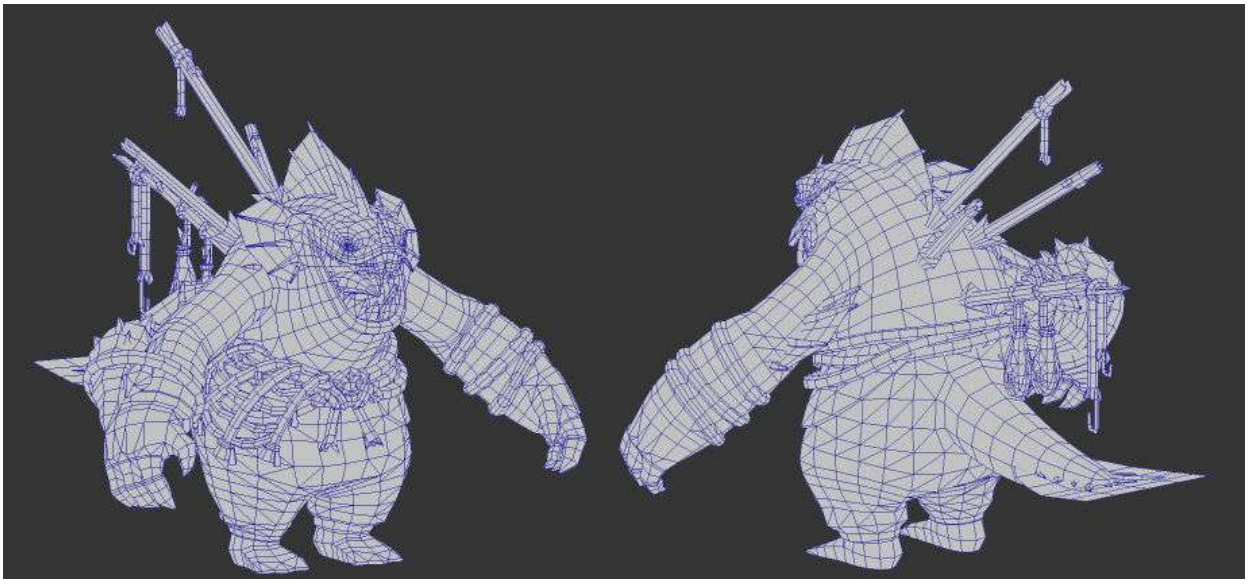
So why did I not model the low from the beginning in 3ds Max?

- It was easier for me to block out everything in ZBrush before working on the low. It helped me get a clean and strong silhouette with good primary and secondary shapes. It was also super important to stay consistent on this part.
- It allowed me to bake the first Ambient Occlusion Map as a base to start my painting process. That way, the lights were already blocked on most of the characters. Also, some details were faster to do in sculpting now, rather than later in texturing (ex: ropes, simple wood base for the harpoon).

Retopology

I did the retopology in Blender with the help of some plugins like "Retopoflow" and "Speed Retopo" because those two were great tools that I preferred to use instead of 3ds Max. When the biggest part was done, I finished fixing and cleaning the low poly in 3ds Max.

A small tip: if you don’t know how many triangles to aim for with your final model, choose a game art style that you like and want to match and take it as a reference. Here I tried to have a model that could fit in League of Legends and ended up with a 12k triangles character.



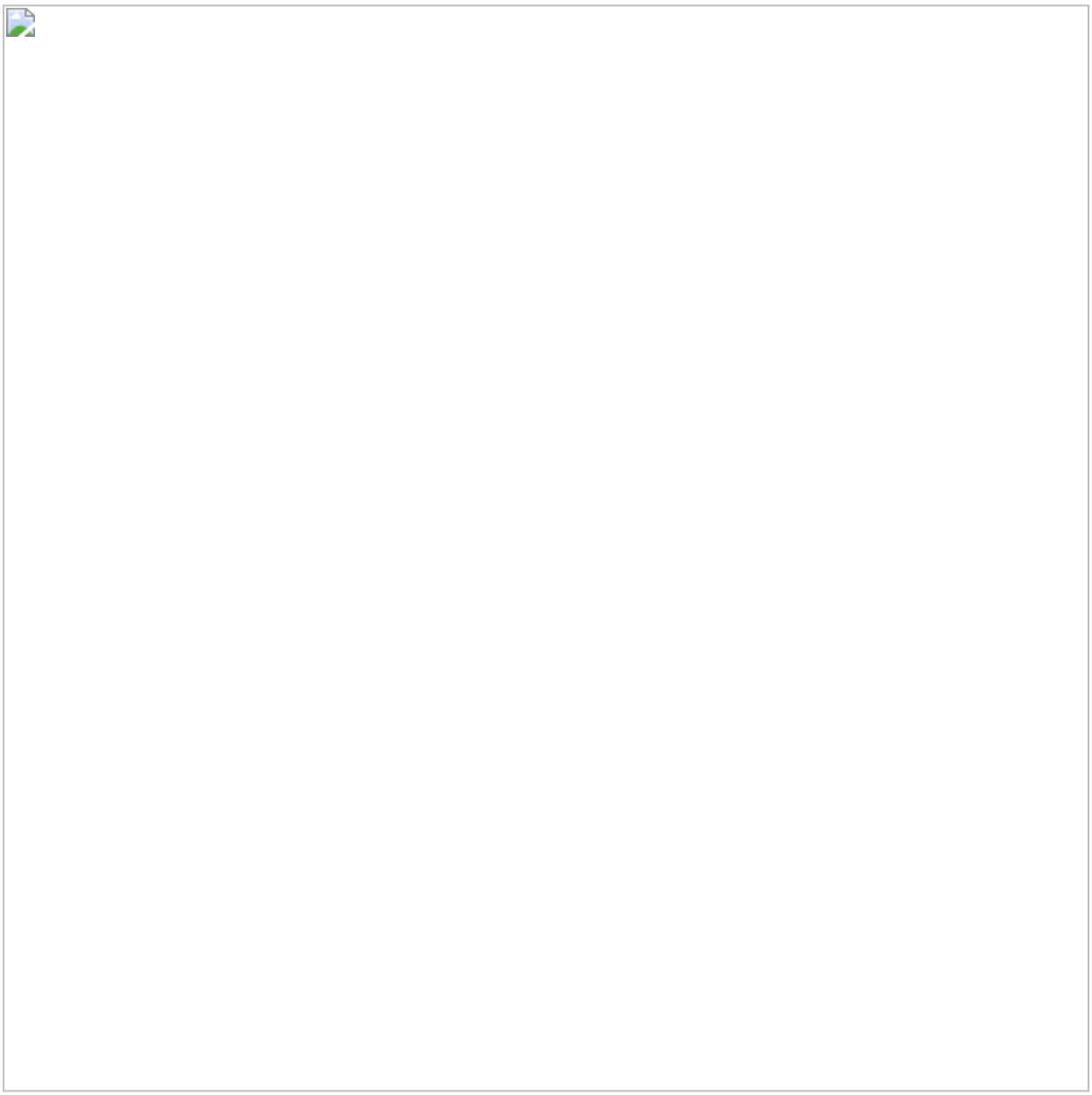
UVs

For the UVs, I divided them into two separate maps: one for the objects and another for the body. Although I could have fit everything into a single map by using a more tileable texture on the ropes, I wanted the light to affect different areas uniquely. Hence, I opted to be a little more expansive with my UVs for this particular section.

Still, I tried to duplicate and mirror other parts of the character, so I could gain some UV space and working time because everything duplicated or mirrored would be something less to paint. So if it doesn't impact the look too much, you should definitely do it!

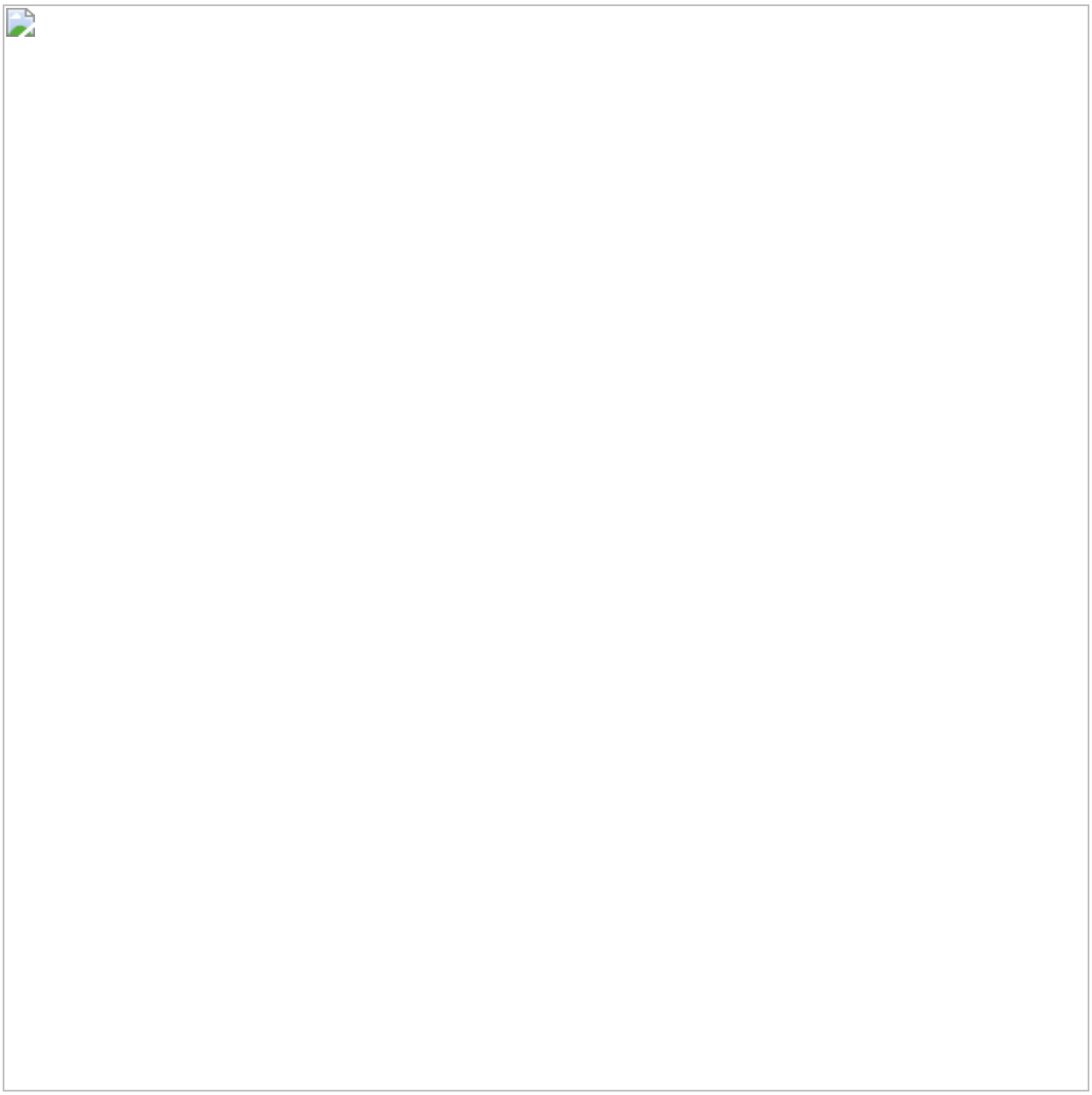
I tried to straighten my islands as much as I could while avoiding too much distortion.

Finally, I tried keeping my UV islands as big as I could, because it was much easier to work on something when everything wasn't split in a ton of pieces.



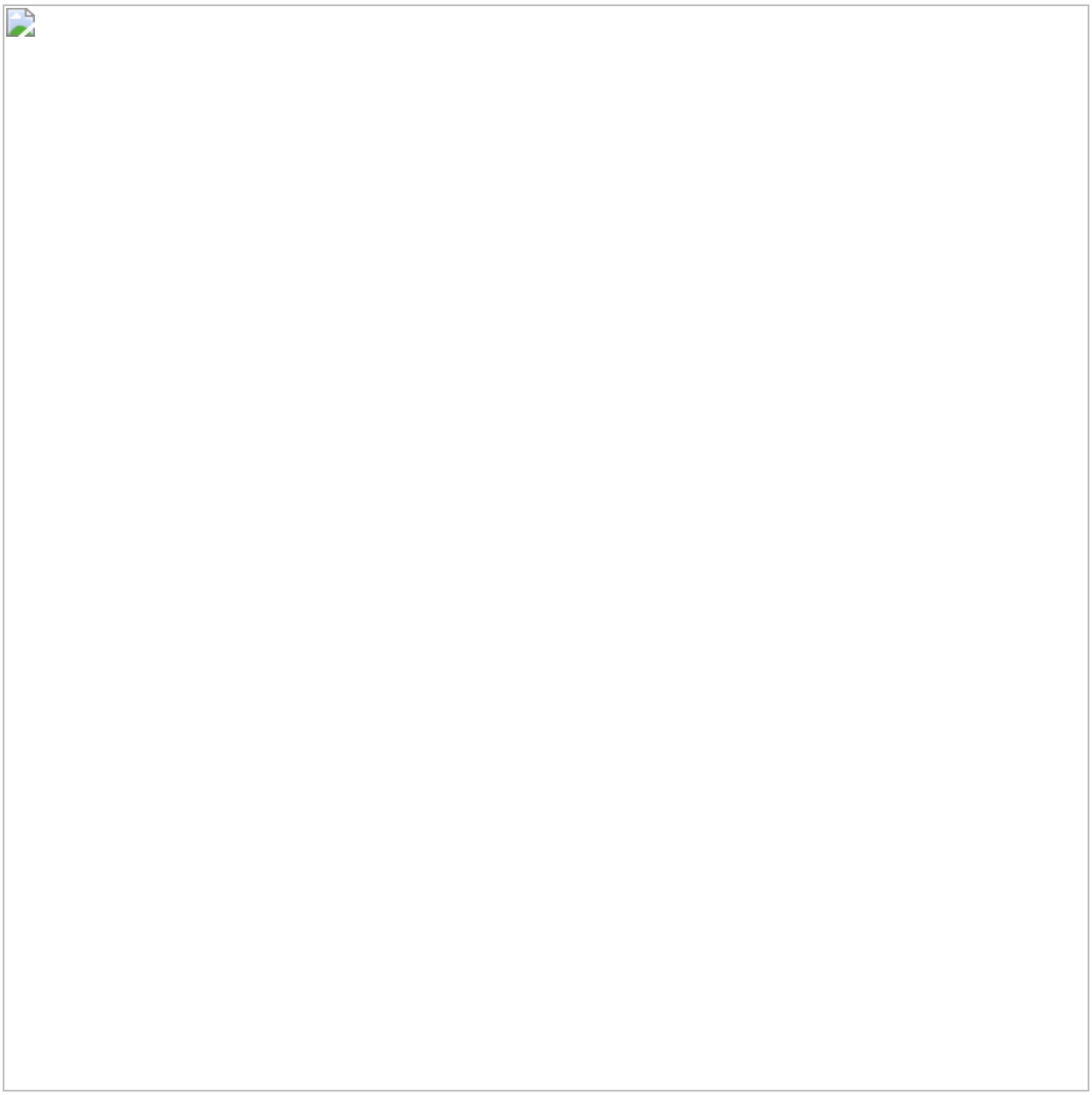
Baking

Speaking of baking, I did it in Marmoset Toolbag 4. Those are the maps that I baked and will use later:



There’s also a “contact AO” that I baked in 3DCoat using my low poly directly so that I could have proper contact shadows between my low poly pieces. In my opinion, the 3DCoat did a better job than Marmoset Toolbag on this part. It was also faster and easier to do it that way.

You can do the same by going to the "Textures" upper menu and clicking on "Calculate Occlusion". Here's the window that will appear, those are the settings that I used. You can play with the light count for better texture quality. I would not recommend going higher than 4096 because your software will probably freeze or crash.



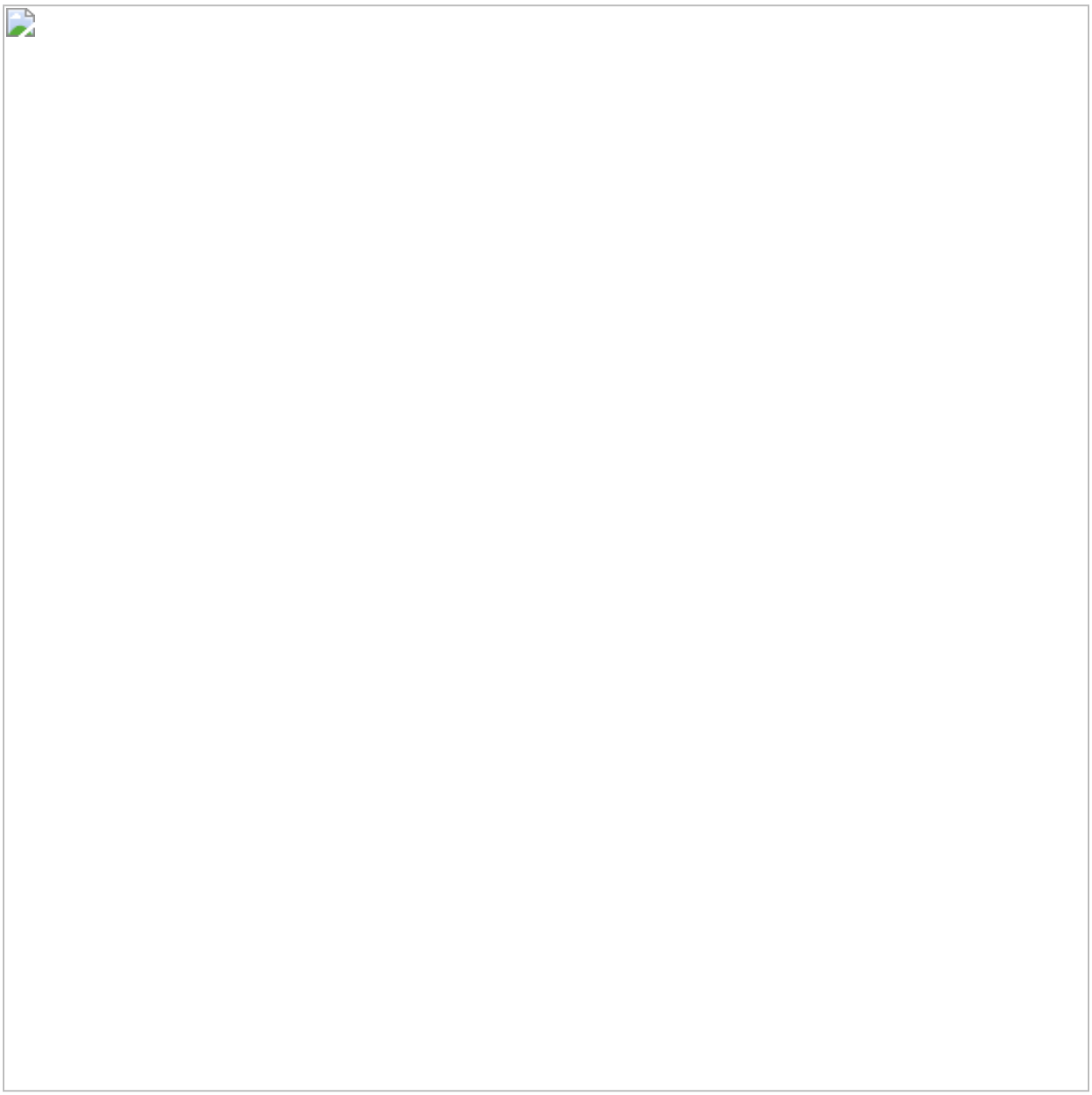
Composited Ambient Occlusion

Before starting to paint, I wanted to have a first base created by using all these Maps. I liked to mix those together and have some kind of grayish final result. I played a lot with the opacity and blending modes of my layers to get what I want.

Here's how I composed my Base Map with only 5 layers:

- First, I used all the Ambient Occlusion Maps coming from my baking process (base AO, AO with floor). The contact AO was added later in the texturing process.
- To get a Light Map, you can use the green channel from the Normal Object Map. To do so, import your Normal Object Map in Photoshop > select the layer > go to channels > select the green channel > press "ctrl+a" > press "ctrl+c" > select back RGB > press "ctrl+v" in the layers window.
  Next, it is necessary to duplicate the green channel from the "Normal Object Map" and place it on top of everything. This will act as a form of lighting, simulating light coming from the top and following the volume of the character.

- The gray layer is a simple layer with a "Multiply" blending mode to darken everything.
- The curvature is used here to sharpen the result. It is very important to use a low opacity on this one. You want to avoid having a strong curvature everywhere.
- The fix layer is a paintover layer to fix artifacts or displeasing mishaps.

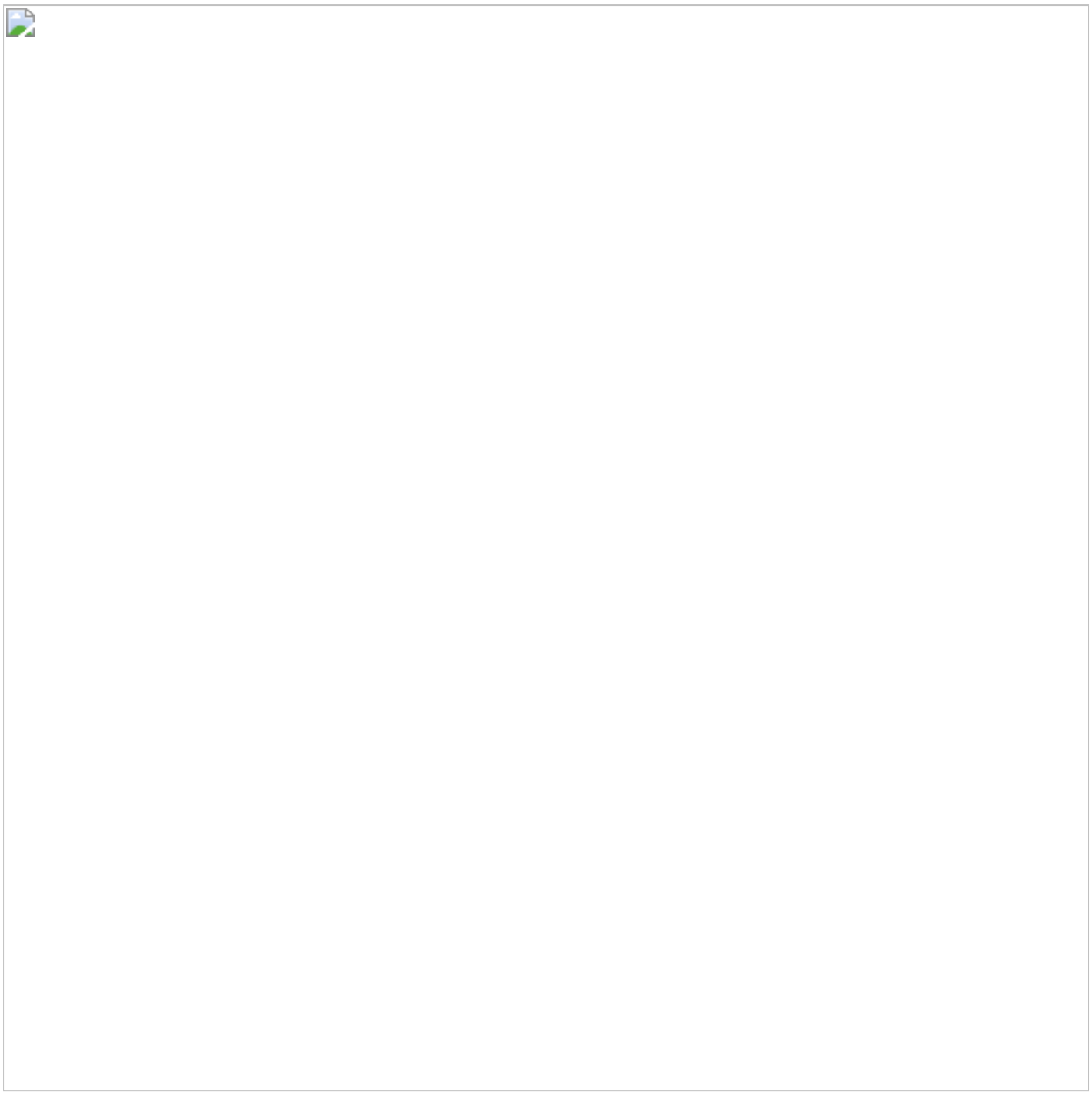


Texturing

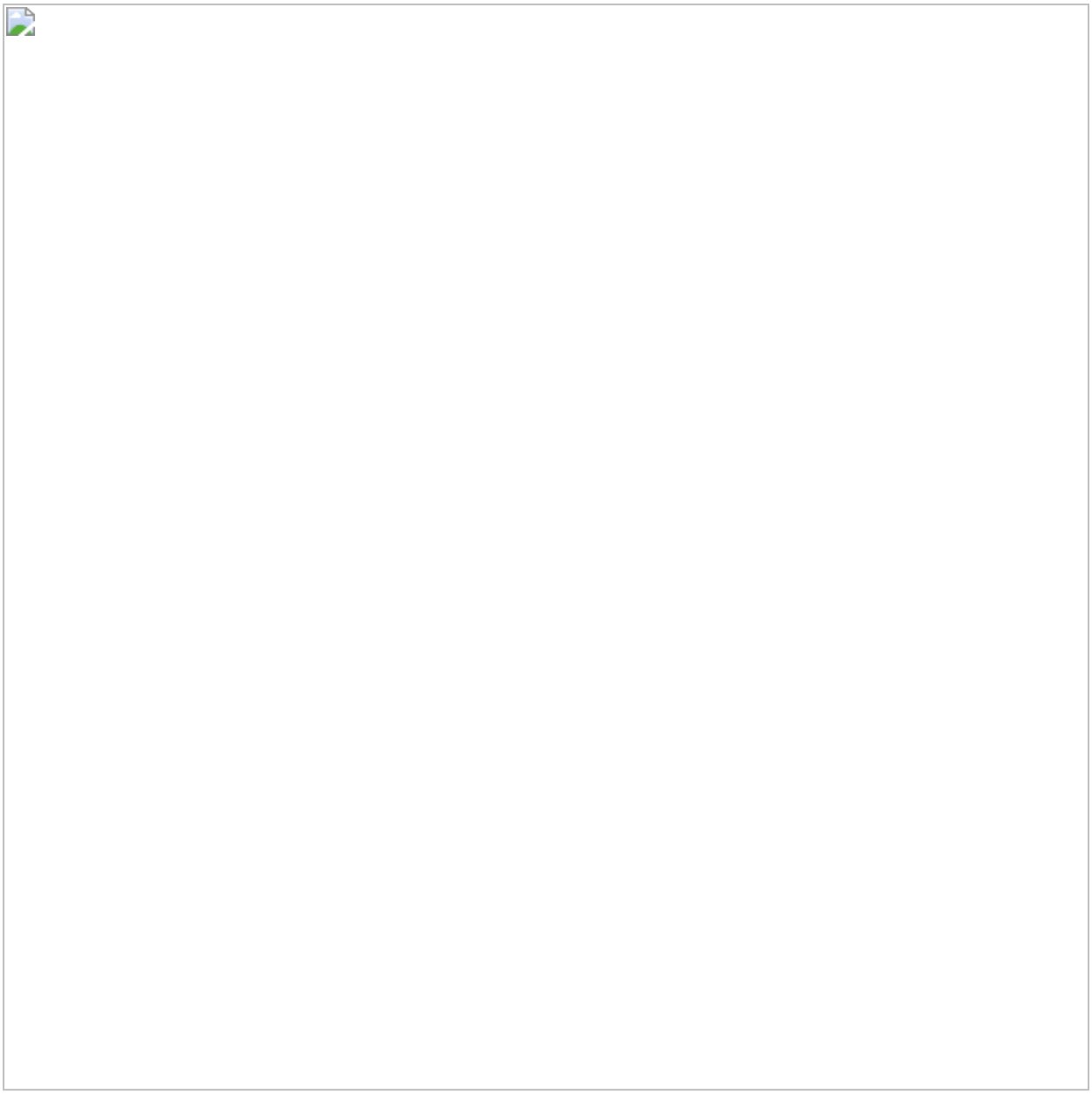
Speaking of texturing, the idea was first to get a black-and-white base before going full color. This was easier for working on the volume and global lighting without having to think about hue and saturation.

Once the black-and-white painting pass was done, we could use some Gradient Maps to apply colors. This tool is super powerful because you can get some early color variations without painting everything.

Here's a step-by-step screen, where you can see the most important parts of the process:



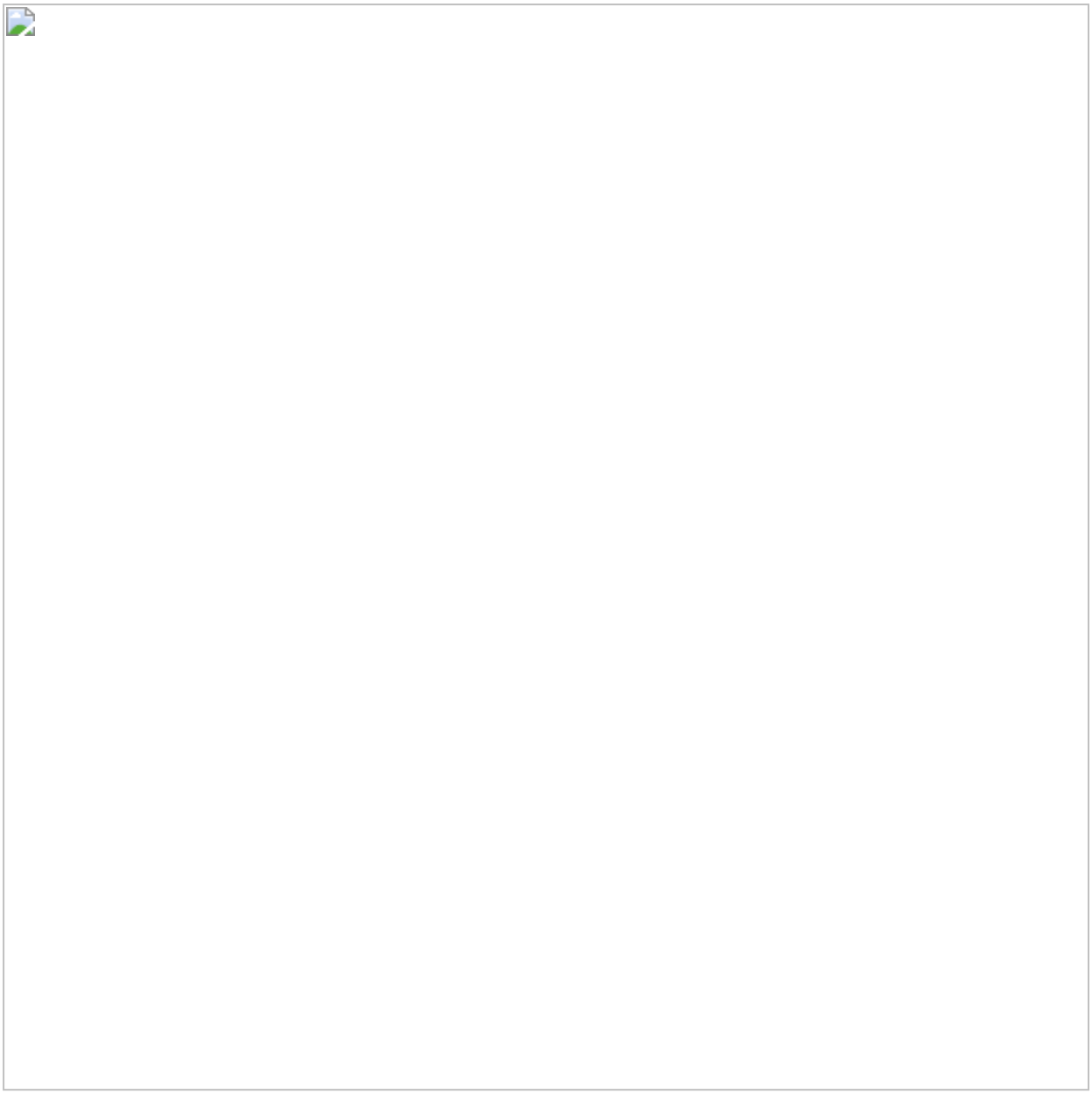
Painting everything can be very long and tedious, sometimes you have to choose where you want to put your effort and details. Here I focused on the head. I added more scales, skin folds, and specular highlights. By doing that, I also created some contrast with the rest of the characters and brought more attention to the head and mouth.



I always try to play with values by creating high contrast, especially on the focal point of the artwork, which in this case was the head and his big dangerous mouth!

Don't forget to take a screenshot and check the values in Photoshop by turning it into a black-and-white image once in a while. This will prevent you from making big mistakes.

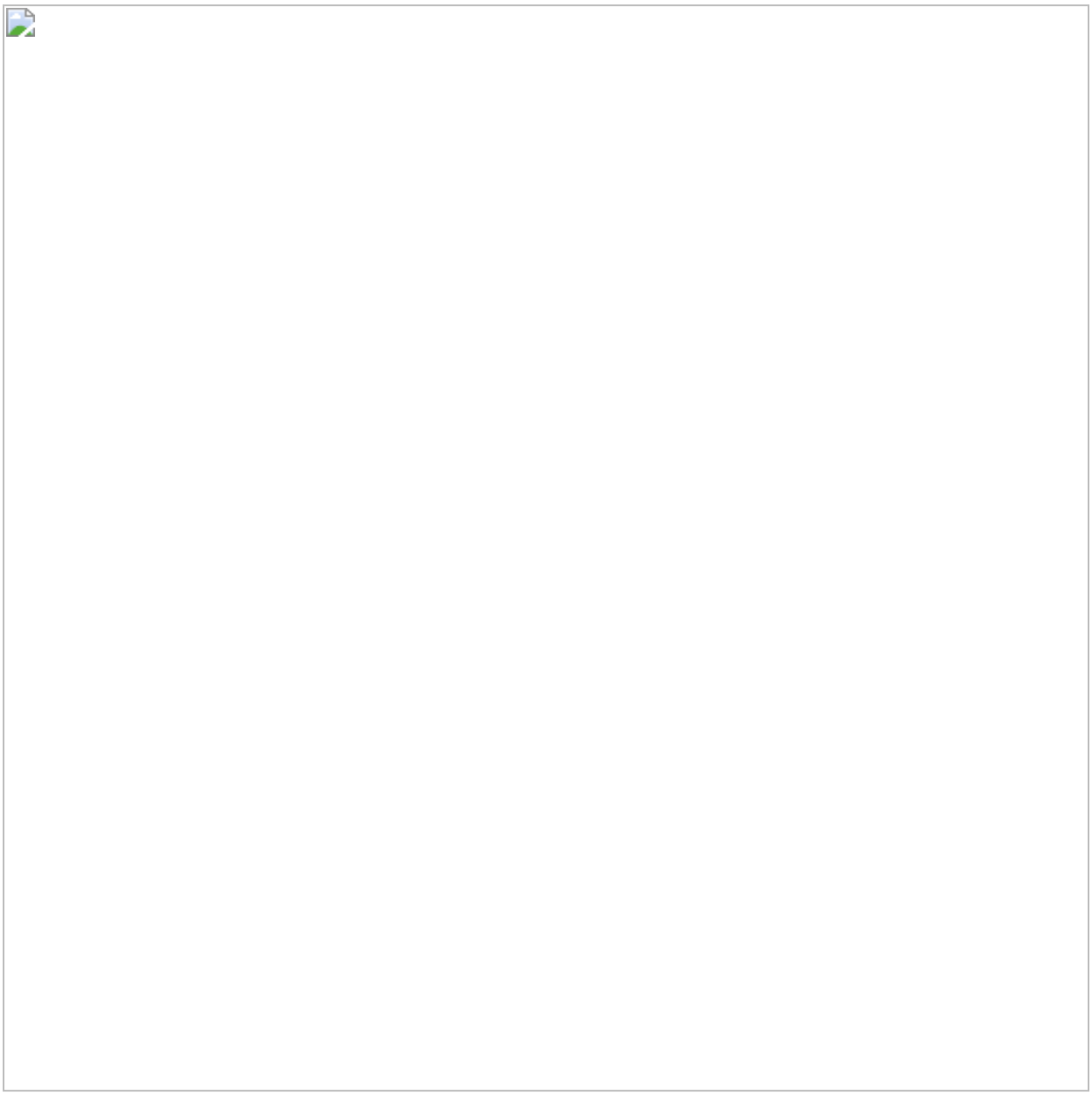
Adding a general gradient coming from the bottom was also super important to keep our eyes focused on the upper part.



Adding some cast shadows helped to bring the character to the next level and make it believable.

Since I was not working with PBR, the cast shadows wouldn't be automatically calculated. Therefore, I had to manually paint them in.

One important tip to remember is to soften the edges of the shadows slightly as this will contribute to a more realistic appearance. If you are interested in understanding the reasons behind this technique, I recommend exploring the various 2D painting resources available on the internet. It's always beneficial to foster curiosity and expand your knowledge in the field.



Finally, here's the full texturing process! As you can see the idea was that I came from big simple lighted volumes, then I added some secondary shape details (the biggest scales, skin folds, scars) then tertiary ones (small scales and specular highlights).

When everything was finally painted, I finished with a color dodge layer to fix one last time the overall global lighting.

Rigging and Animation

The super rig that I used to animate him with was made by my friend Max André! His rig was so nice to animate! Don't hesitate to check his profile for more information!

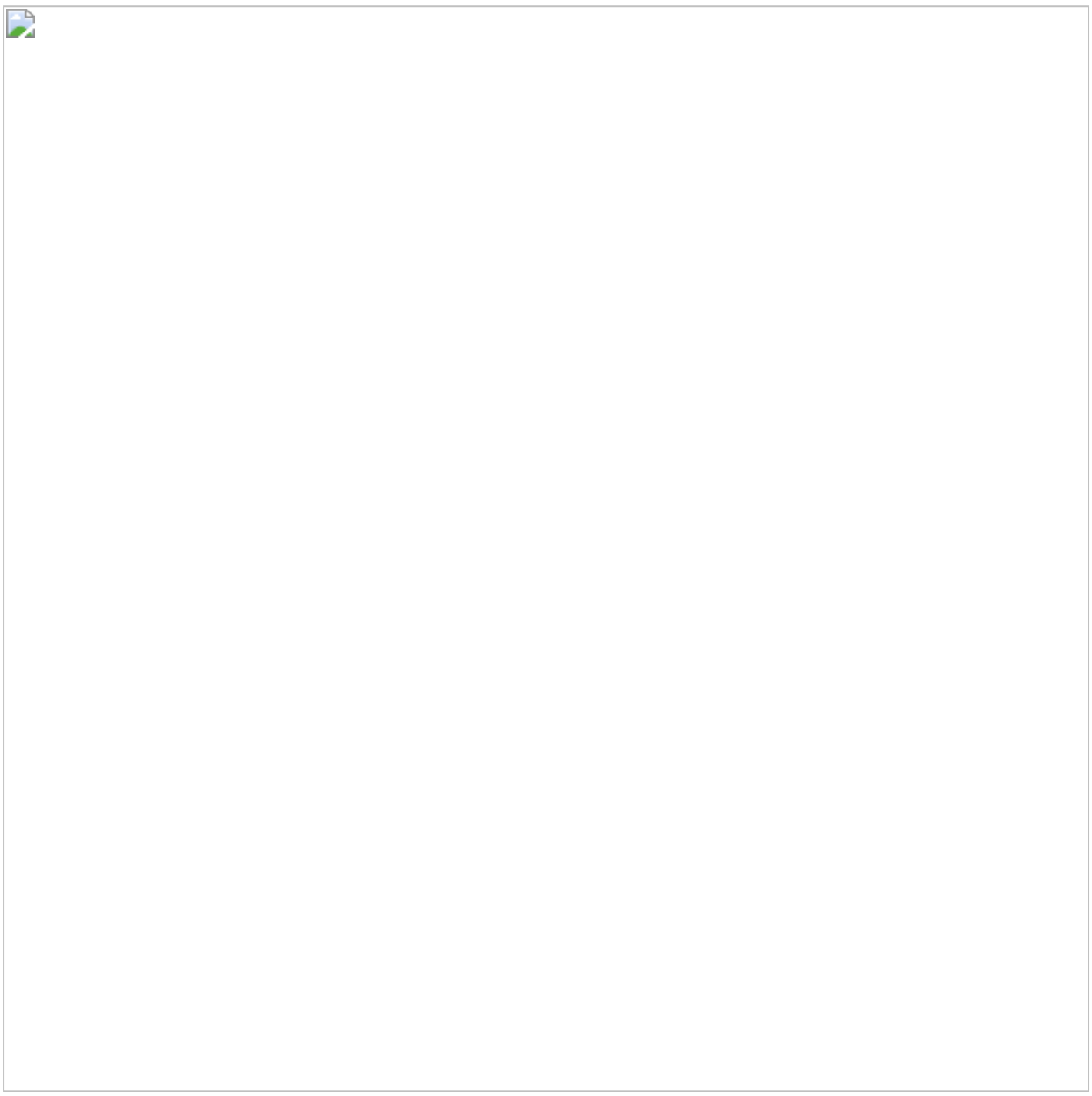
The animation really helps to bring a character to life. This was the first time that I animated one and, I have to say, that was something that thrilled me a lot! This idle animation was quite simple, I created a simple loop of 4 seconds with him breathing. Duplicated it three times to get a 12-second animation. And in the second loop, I implemented a small idle breaker with him turning his head to make it feel less boring.

Rendering

All the renders were made in Marmoset Toolbag 4 (except for the Sketchfab scene).

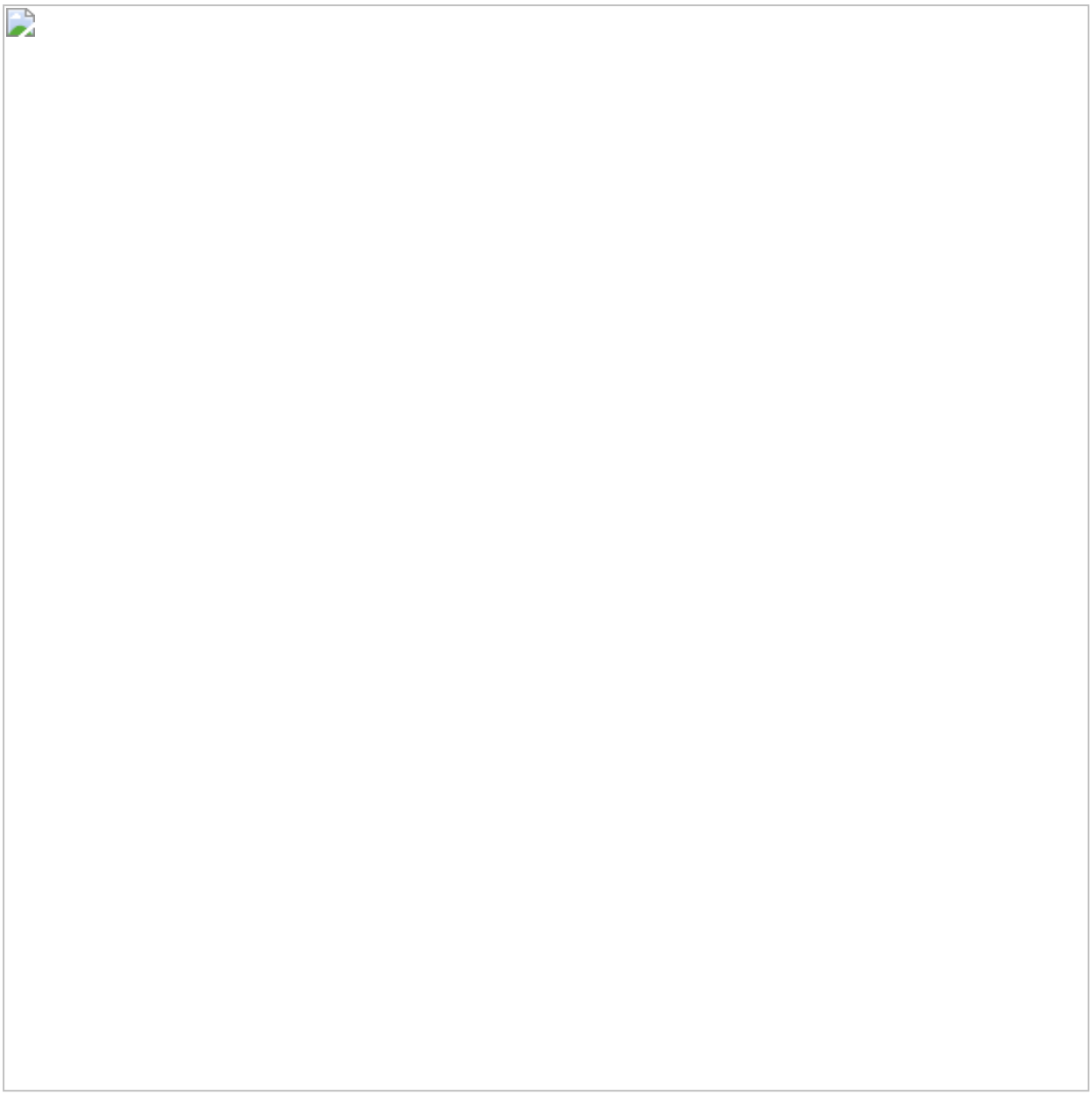
Because this character was unlit, there weren't many settings to tweak. You can see on the screenshot below the materials settings. As well as the post-effects that I played with.

However, a very important post effect that I used is the "Sharpen" one. Sometimes the textures can feel a bit blurry in Marmoset Toolbag, so using the Sharpen slider helps to bring back the details that could have been lost and give better visibility of the character overall.



I also implemented dynamic shadows on the ground with a single top light. But for that, I had to light the ground using a simple material so that the character’s cast shadow would work. Otherwise, when I don't have an environment, I use a shadow catcher.

Finally, I added a bit of Ambient Occlusion in the render settings just to have a dynamic Ambient Occlusion that would follow the animation.



Conclusion

This was an interesting project that allowed me to experiment with a lot of things. Even though I am no Environment Artist or Animator, it was still super fun to make this small scene as well as the character!

Thank you so much for your time and attention to my breakdown! I hope this helps you in some way with your future projects!

Francisque Facon, Character Artist

Interview conducted by Theodore McKenzie

Art: Francisco Rivas

Creating Stylized Violet House in Substance 3D Painter & Unreal Engine

80.lv/articles/creating-a-stylized-violet-house-in-substance-3d-painter-unreal-engine

Victoria Zavhorodnia October 9, 2022

Victoria Zavhorodnia showed the workflow behind the Violet House project, talked about making the vegetation, and explained how she achieved the painterly feeling in the render.

Watch Video At: https://youtu.be/3gfc3agEX-g

Introduction

Hi! My name is Victoria Zavhorodnia, I'm an Environment Artist from Ukraine but now I live in Belgium. I'm working as Art Director on the Town Star game at Gala Games. When I started working in this position, I understood that the feedback stage is very important. It helped me understand how to follow and develop the style of the game. This job increases my artistic skills because this game has an effortless visual style, but here lies the biggest challenge: the more straightforward the task, the harder to make it properly.

My journey in the environment art world started five years ago. I'm a big board and computer games fan and I have always wondered about the game creation process and how to join this process. When I applied for my first job in the industry, I didn't have a

portfolio, so I did some test work, and they liked it. That is how I got my first job (a Level Artist). And I'm very thankful to this company (Artificial Core) for that chance because I found my passion in level art and environment creation.

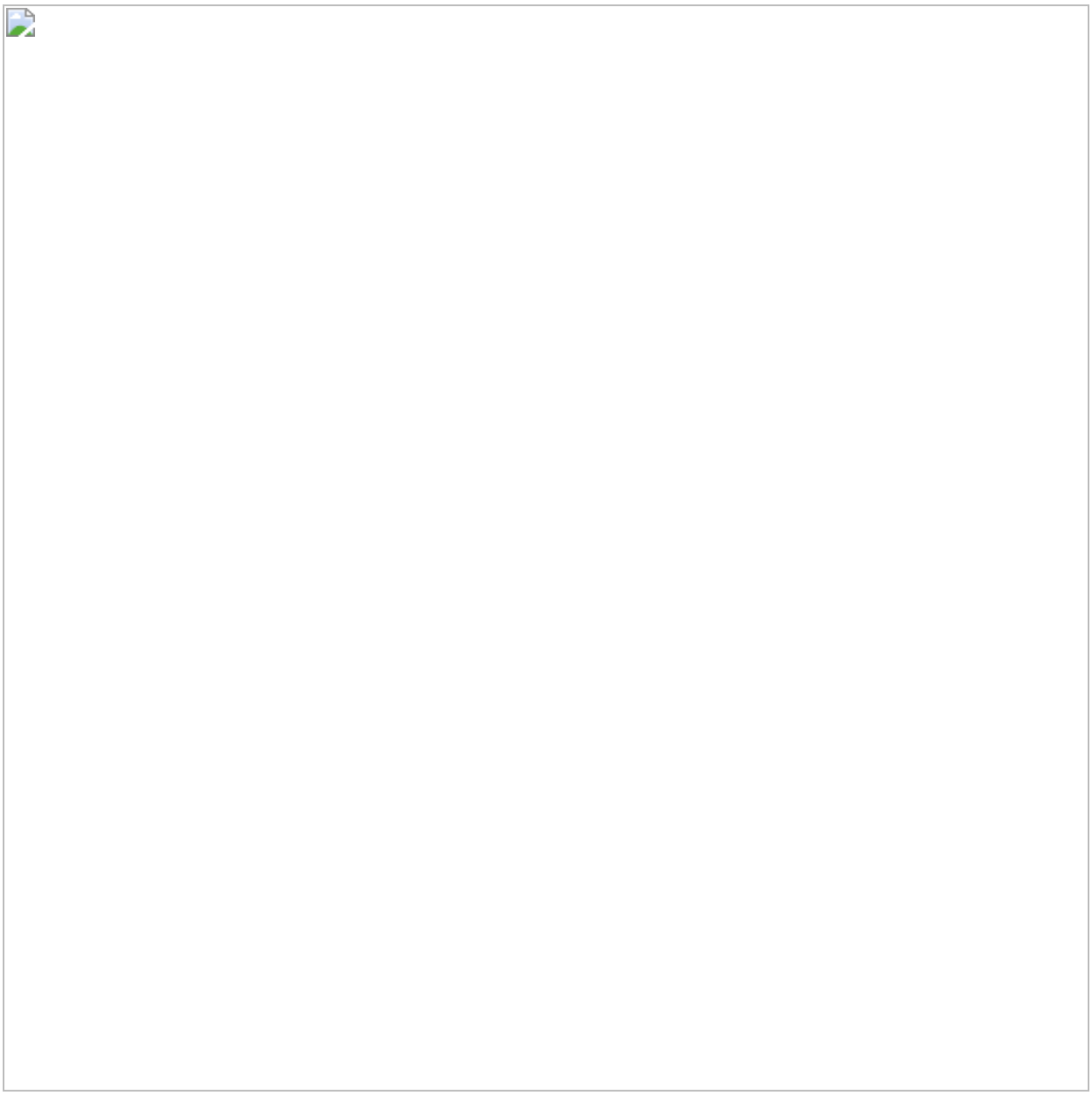
When I started working in the game industry, I discovered different game creation fields: 3D art, concept art, texturing, lighting, etc. Also, I saw fantastic environments created from scratch on ArtStation and thought of doing something by myself. I understood that to learn faster, it's best to do some courses where all information is clearly set out, and I started with 3D, then texturing, Unreal Engine, environment art course, and lighting in UE; I already talked about courses I passed in the previous articles. I am still learning because new tools and ways to do something develop daily.

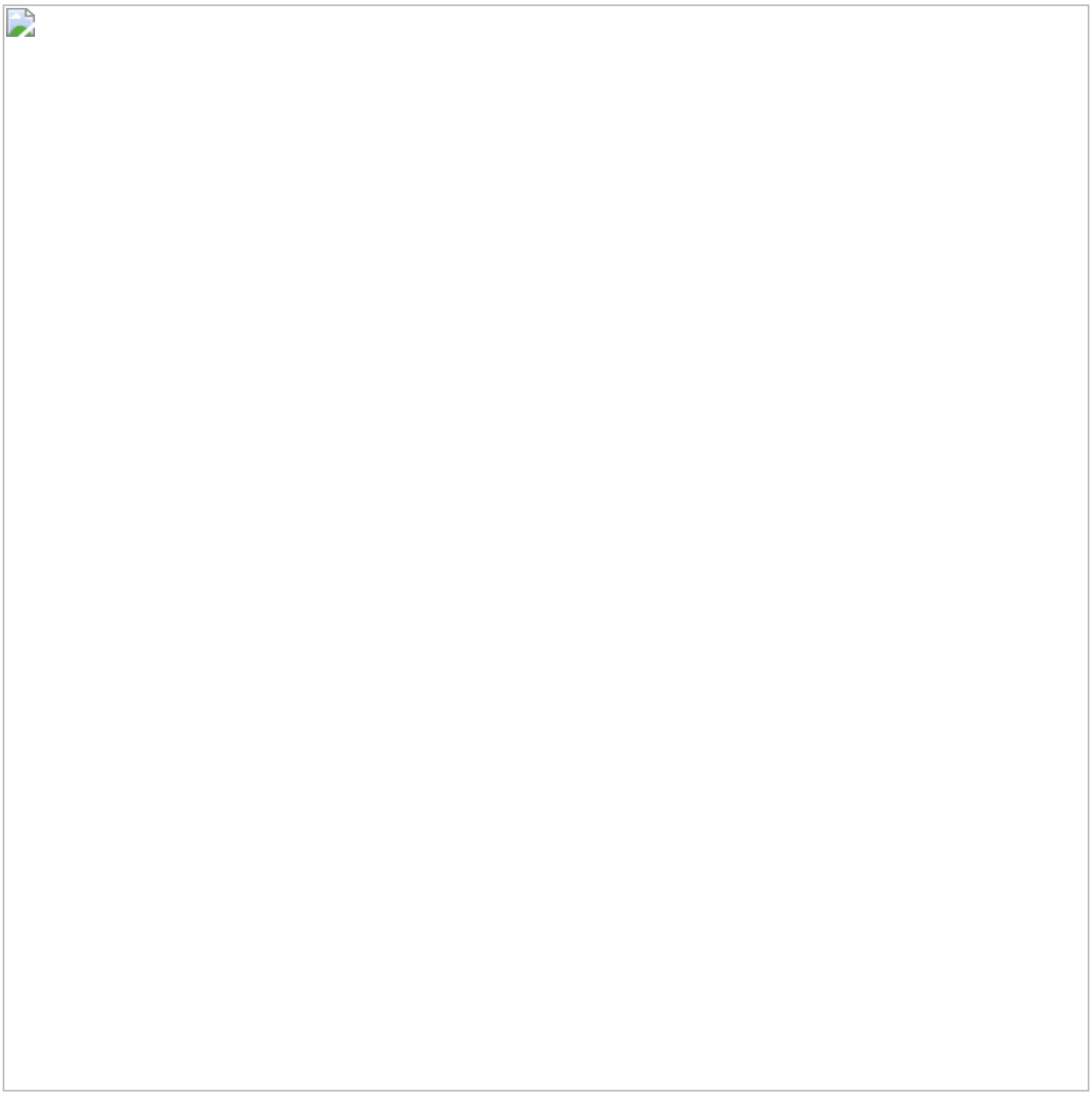
Watch Video At: https://youtu.be/2Lit3JITG2w

The Violet House Project

Stylized art appeals to me more than realistic because it's peaceful, colorful, simple, relaxing, and kind. I will not be the first to say that Studio Ghibli's movies greatly affected my art, as well as anime, comics, and Genshin Impact. I receive a large amount of inspiration from ArtStation, Pinterest, and Twitter. Also, when I see feedback from people about my art, that motivates and inspires me to create new projects.

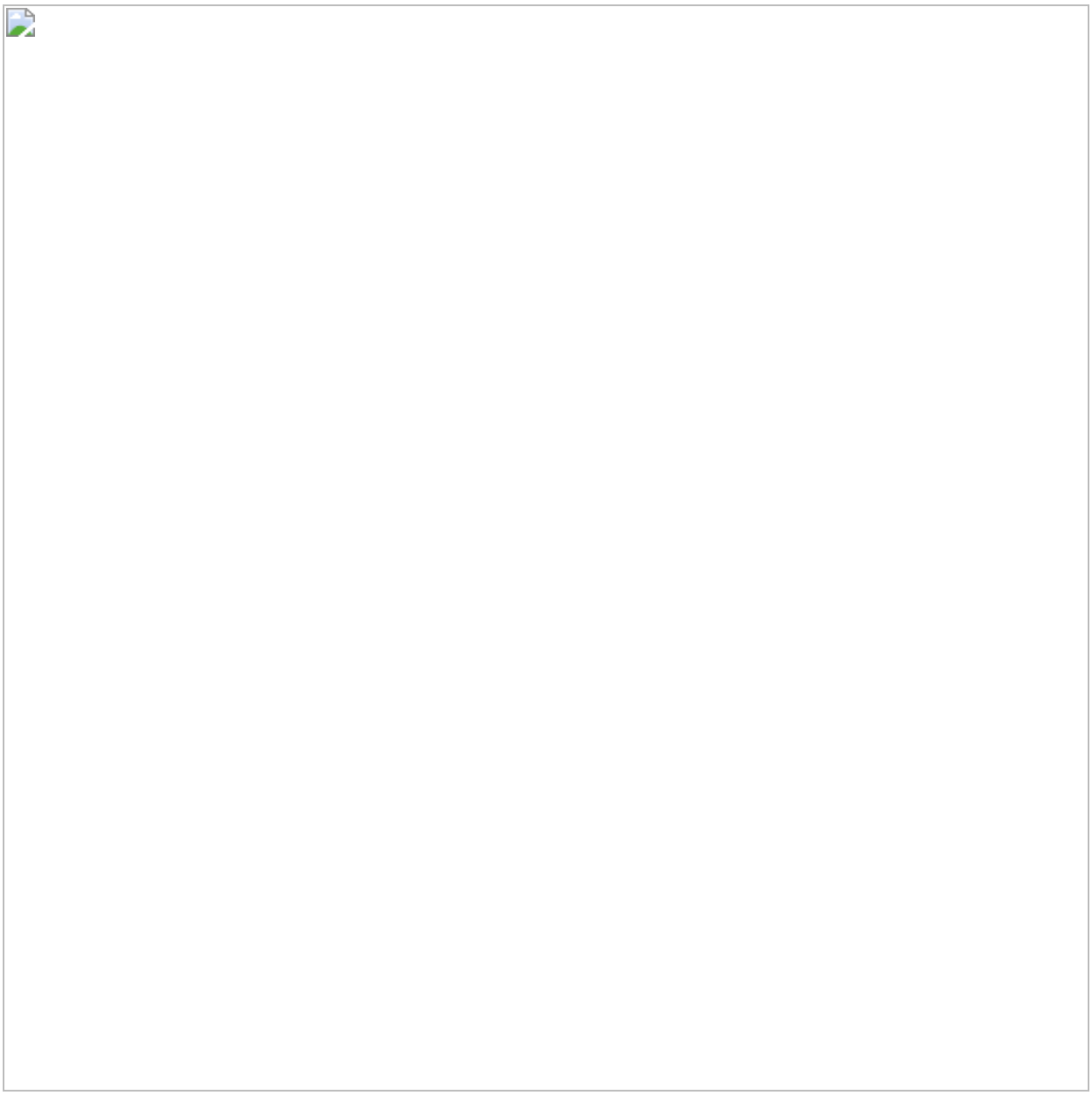
Because I moved to Belgium, I didn't have time to do any art for my portfolio and I wanted to create a simple project to remember all the tools. Whenever I see something interesting on ArtStation, I save it to one of the many collections I have. I often use it when I search for good references. I searched for simple concept art of a building with simple nature around it and found beautiful art by David Merritt. Usually, I search for supporting references for each part of the concept to better understand the textures, structure, etc., but not this time because of the simplicity of the concept.

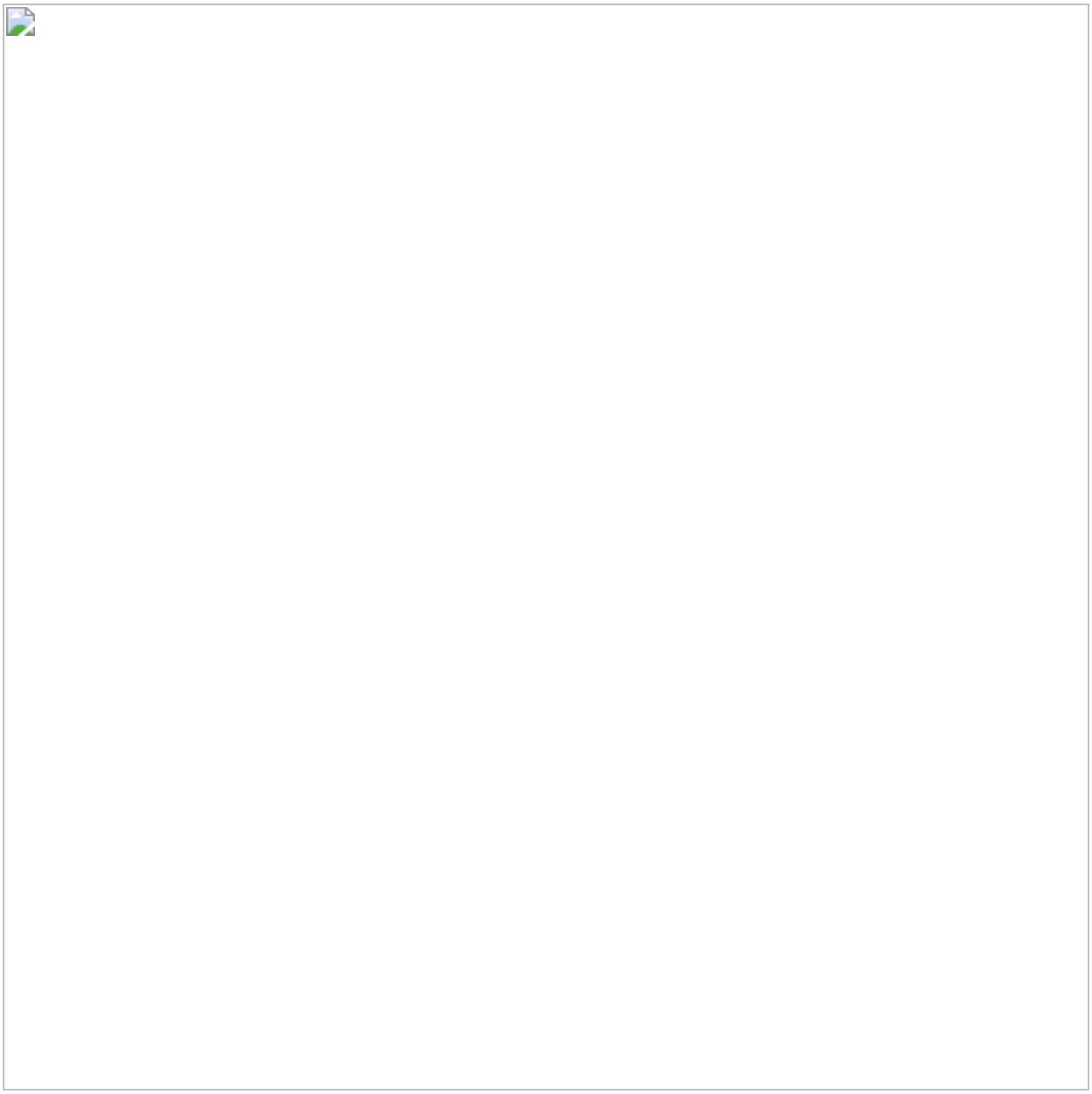




Modeling

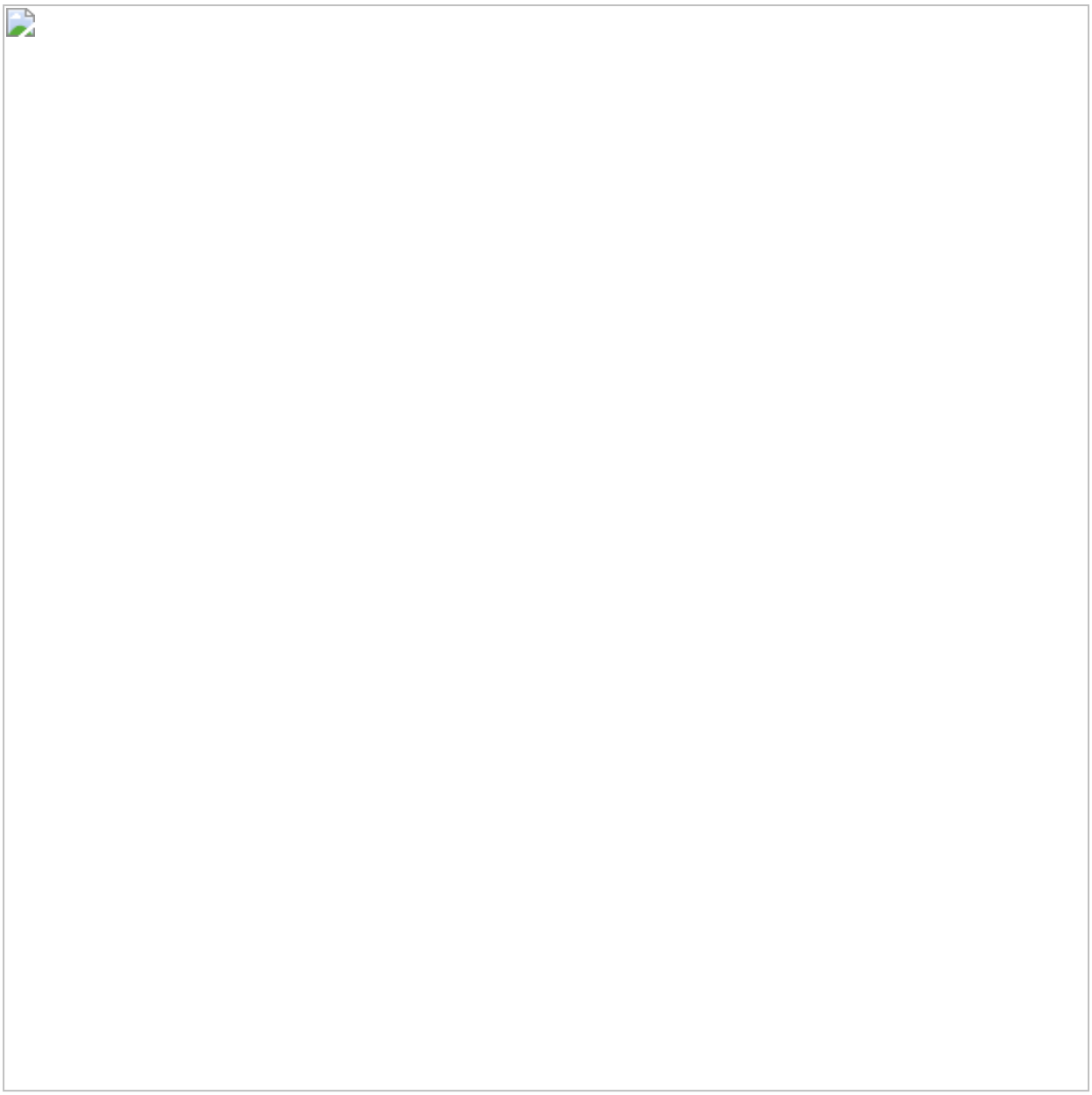
After choosing a concept art, I always start with a simple blockout in Unreal Engine to set up the composition as in the reference and match UE's mannequin proportions because it’s game art, so proportions matter. And when all objects are looking good in the grey box stage, I export this mesh from UE and move it to Maya, where I make a white-box model. Then I import the white-box model to UE and see what this model looks like. This process is iterative until I receive a good-looking low poly/mid poly model.

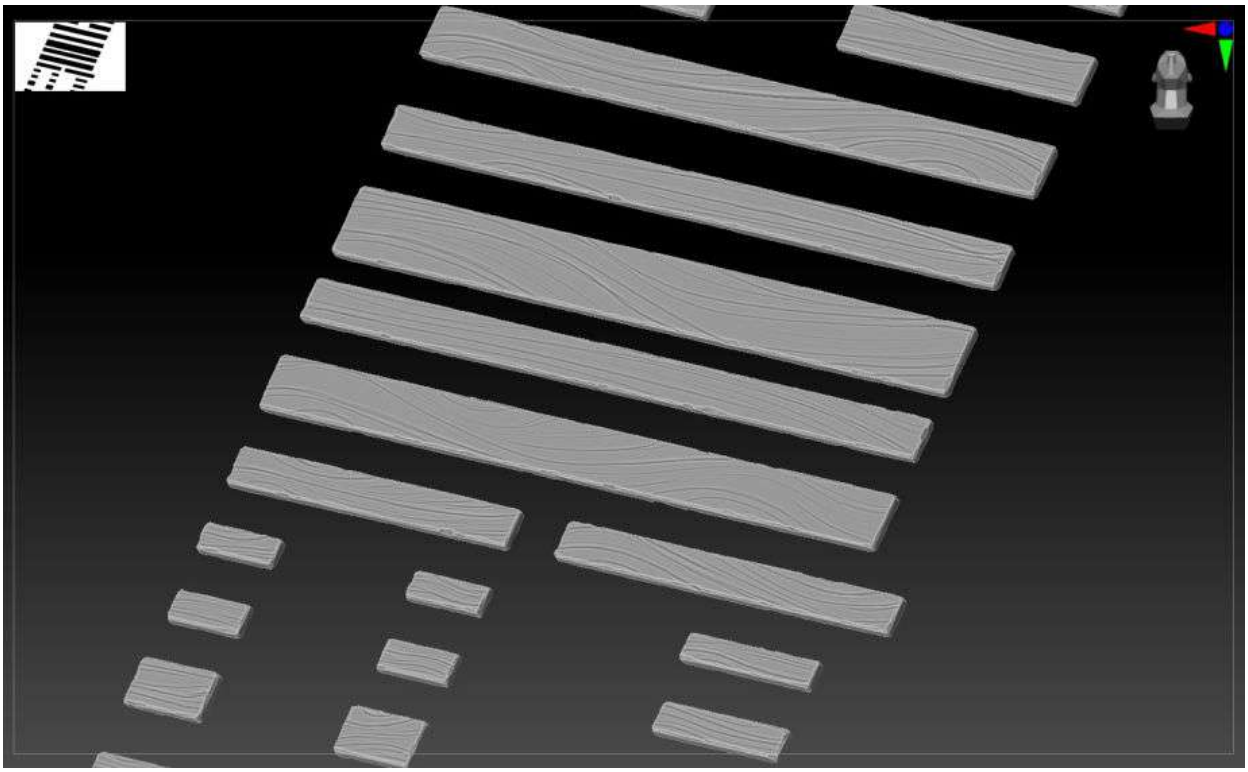


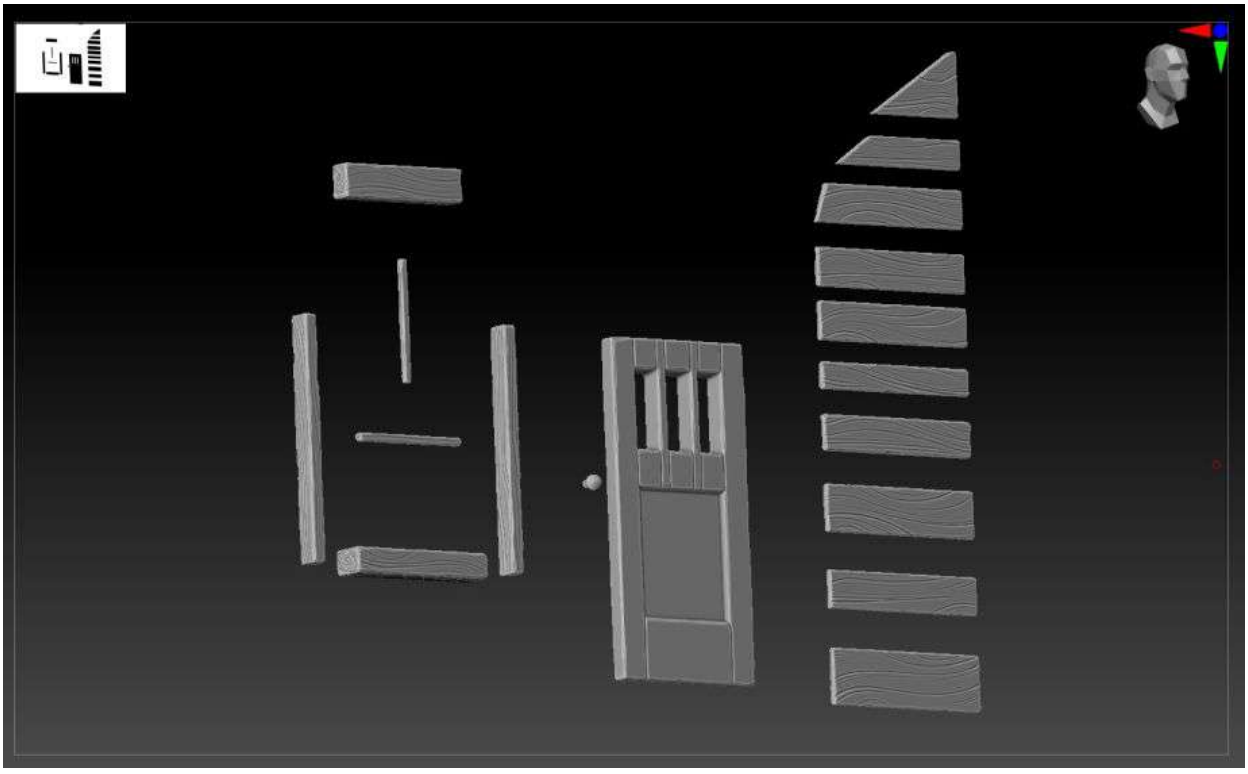


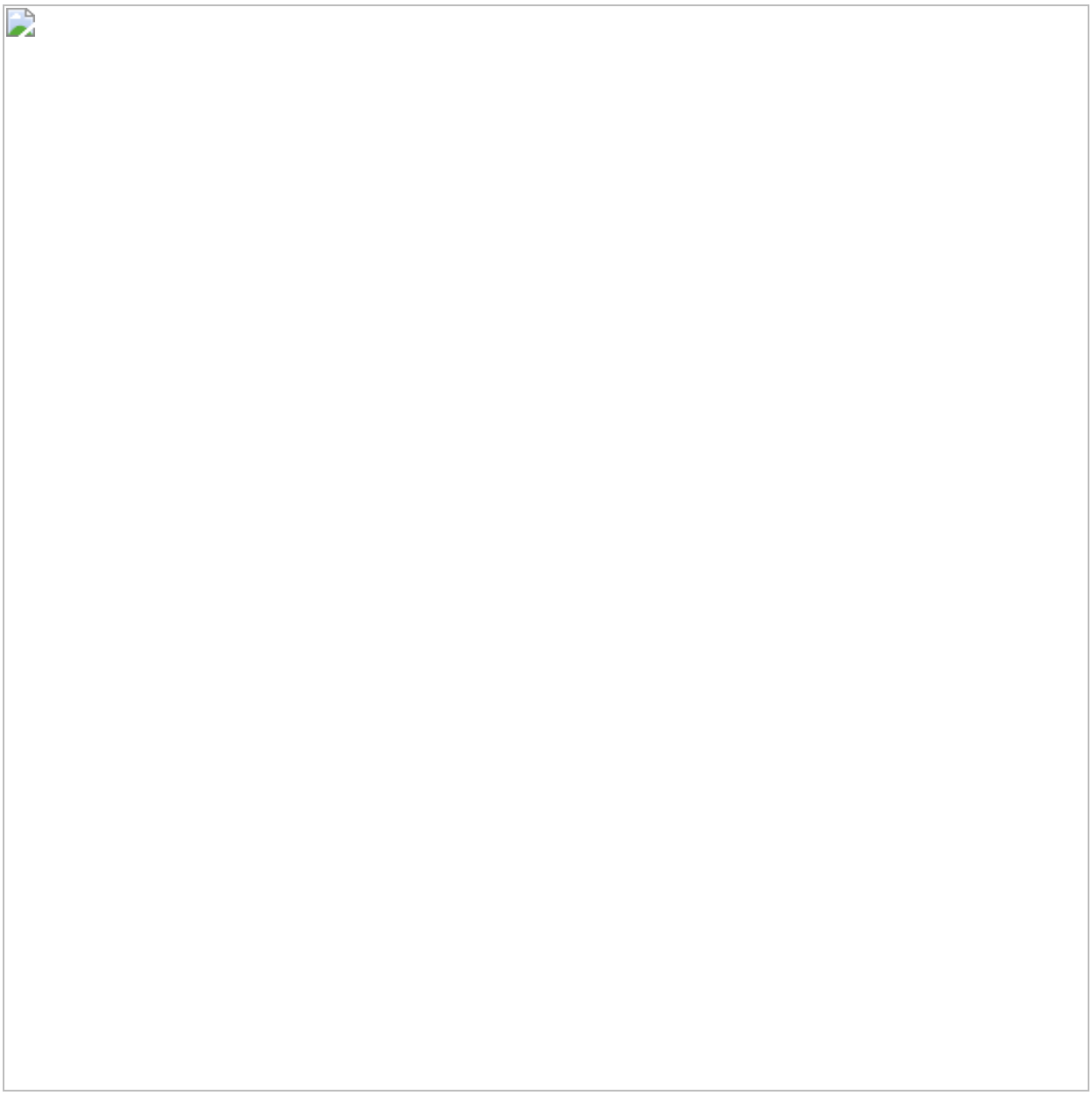
I always imagine a model as a big modular kit and I like to create a few smaller kits and then, from them, recreate the object. For example, if I have a building with wooden planks walls, I will create everything unique and then I start analyzing how many planks I need to have to make repetition not visible. From ten, I will leave only three, sculpt them, bake and recreate this wall only using them. But for this house, I made almost everything unique because my goal for this project was to practice my skills. I have a rule, “Keep it simple,” and I use it in all stages of my work.

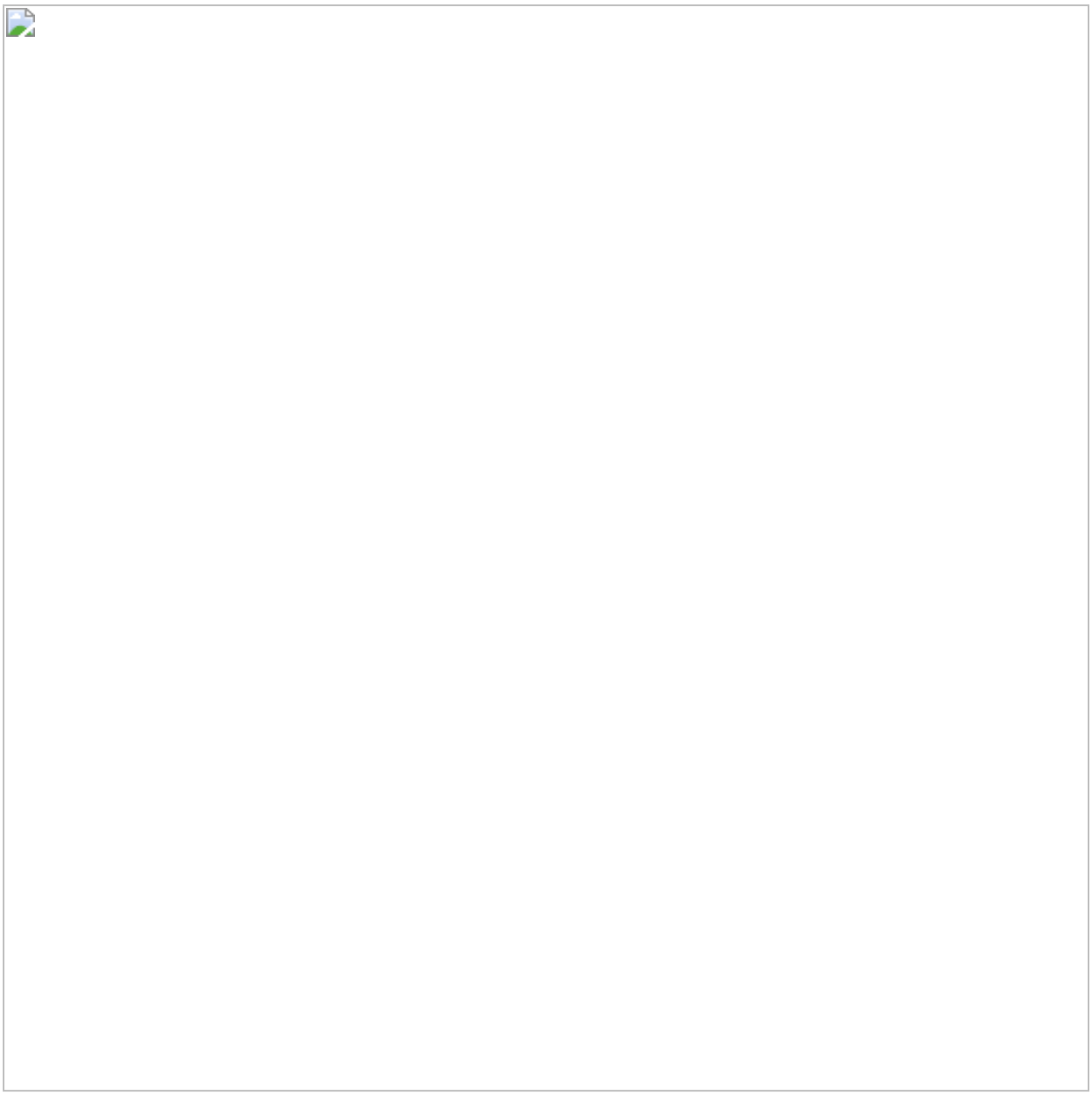
I separated this house into six groups for more effortless sculpting, better texture quality, and an easier texture process (one wall, the second wall, stone parts, windows, door, roof, else). For the roof, I decided to make it geometry entirely, and I used a mid poly pipeline. I didn't sculpt it, I just made a smooth version and baked it to low poly. All other assets I sculpted in ZBrush using only two brushes: Trim Dynamic for borders and Orb Cracks for wood imitation. I bake everything in Marmoset Toolbag.



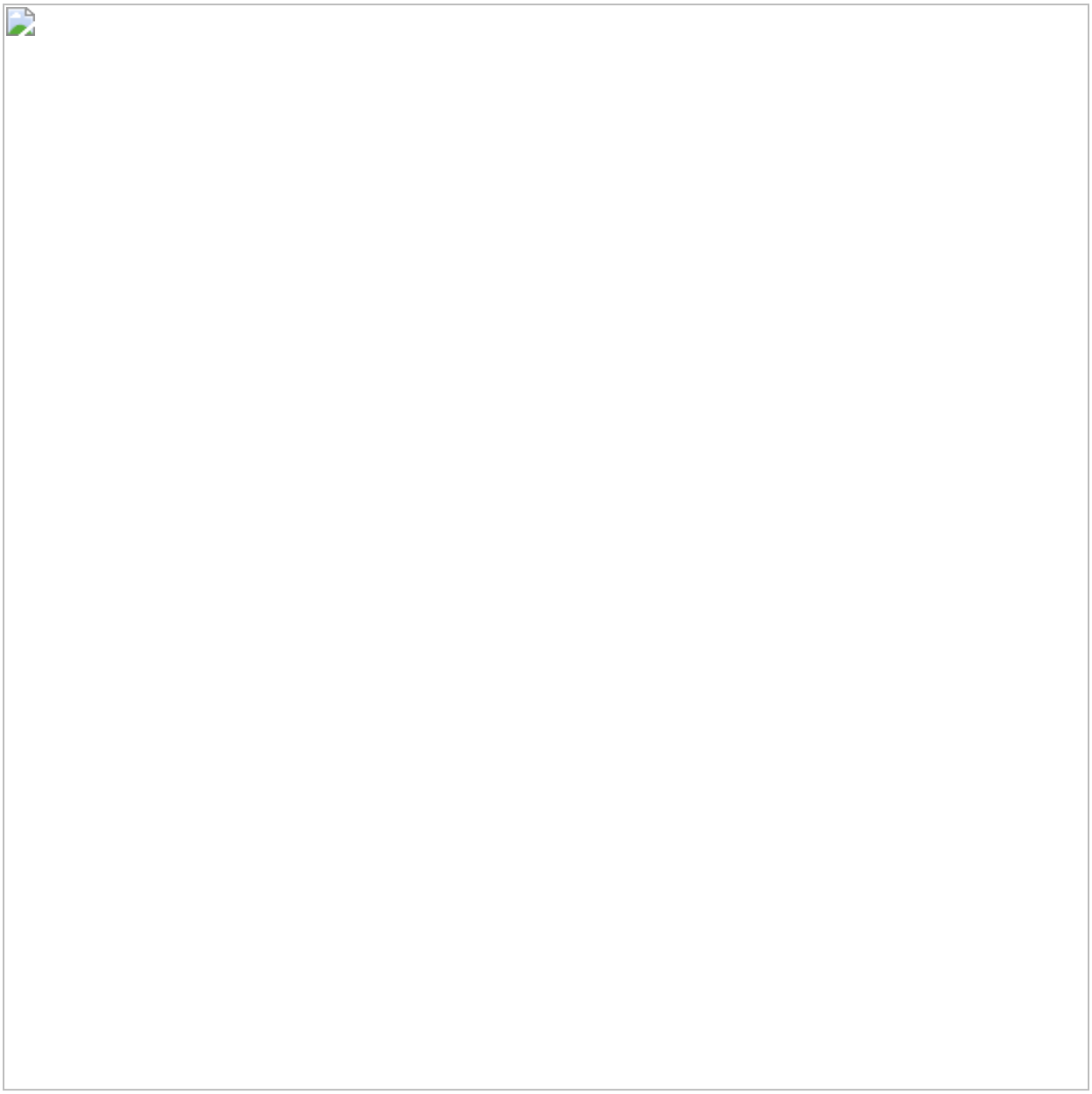




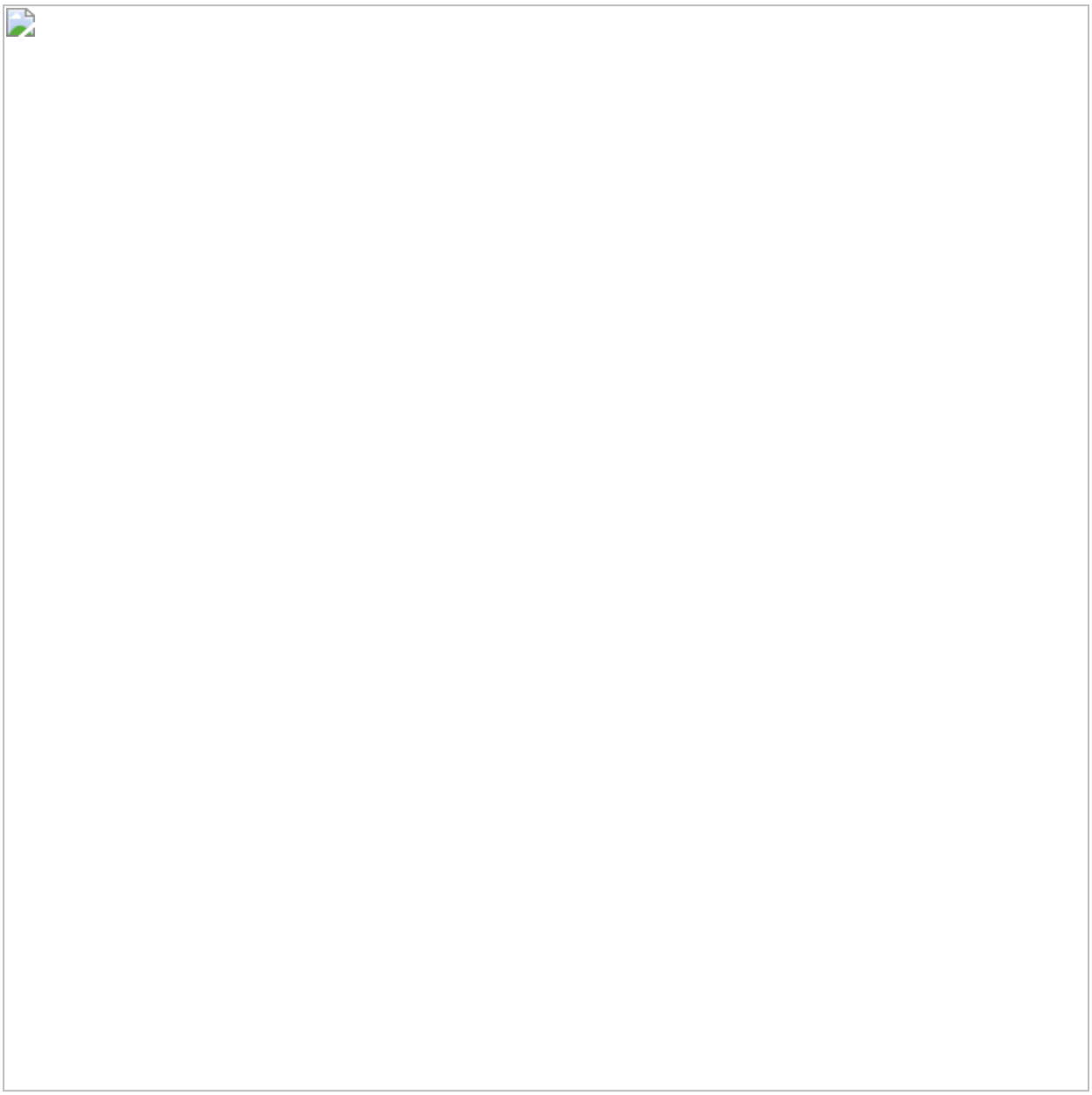


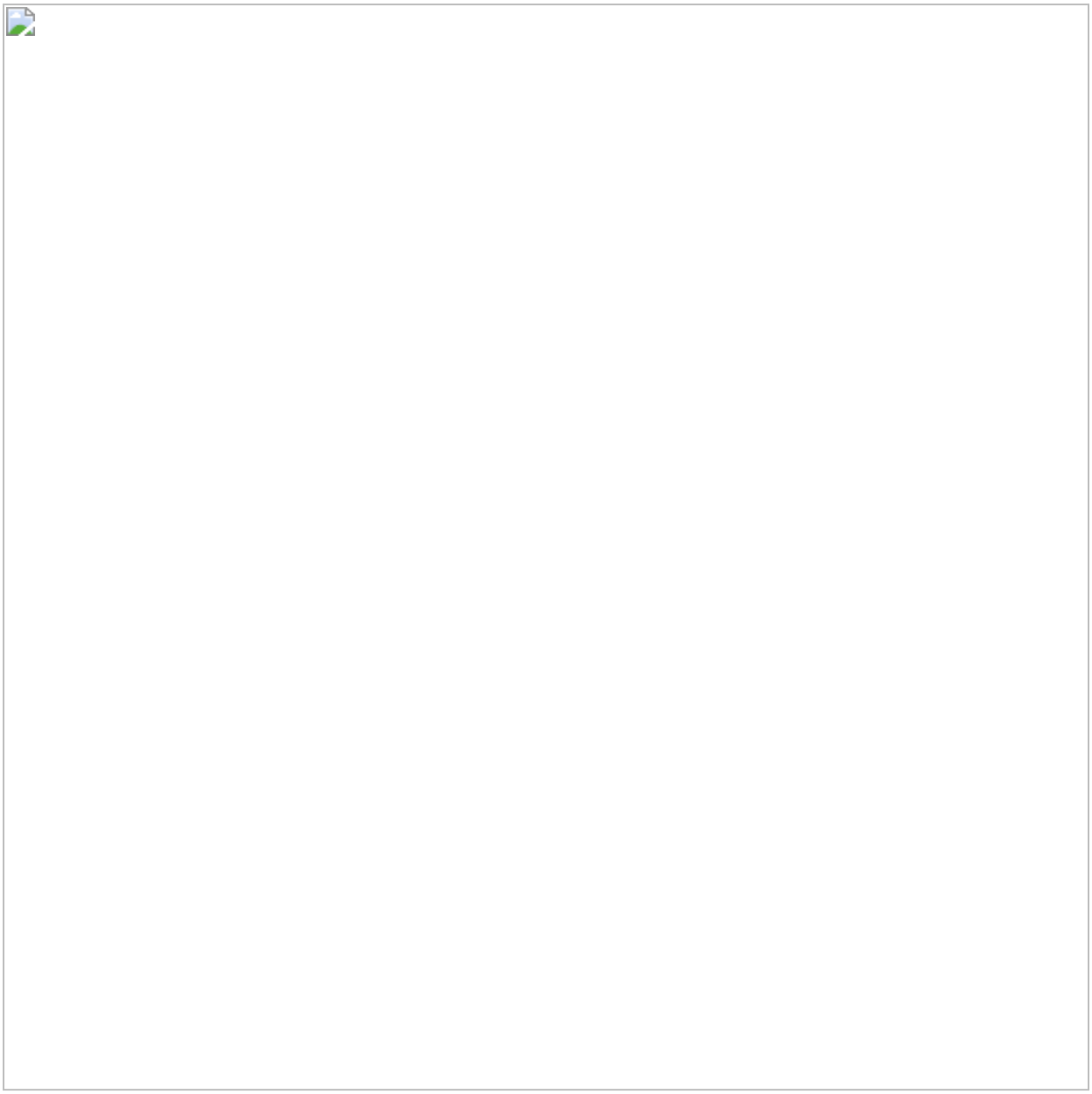


When the main object was made, I started working on vegetation. This concept's biggest challenge was reproducing these paintery trees and vegetation. I already knew the good billboard technic shown by Andre Felipe in the course The Environment Artist’s Survival Kit, but this method did not fit this scene. I started searching for some tutorials on YouTube, and I found another method of creating stylized trees using the TreeIt program by Marpetak Dev, but this method also did not fit. And then, I started researching how I could unite these two approaches into something new that would have similar to the concept art style trees.



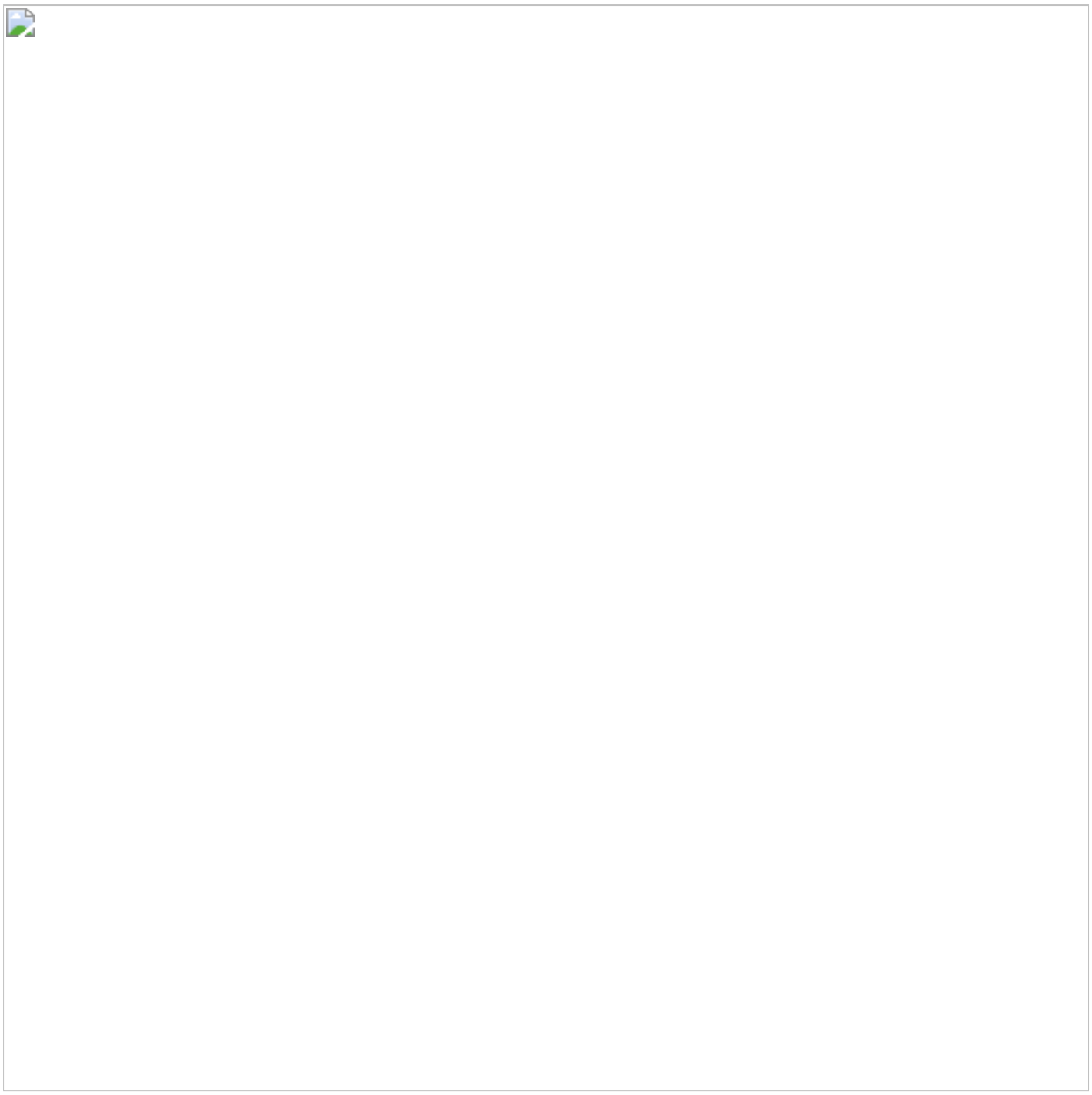
When I published a WIP screenshot with my trees research result, I received massive positive feedback about them and a large number of requests about creating a tutorial on how to make the same. I was very happy because people liked my trees and I created a detailed tutorial where I showed how to create trees like this. This method is quite simple: create a leaf mask in Photoshop, make a tree in TreeIt, and add a billboard-based shader – your tree is ready.



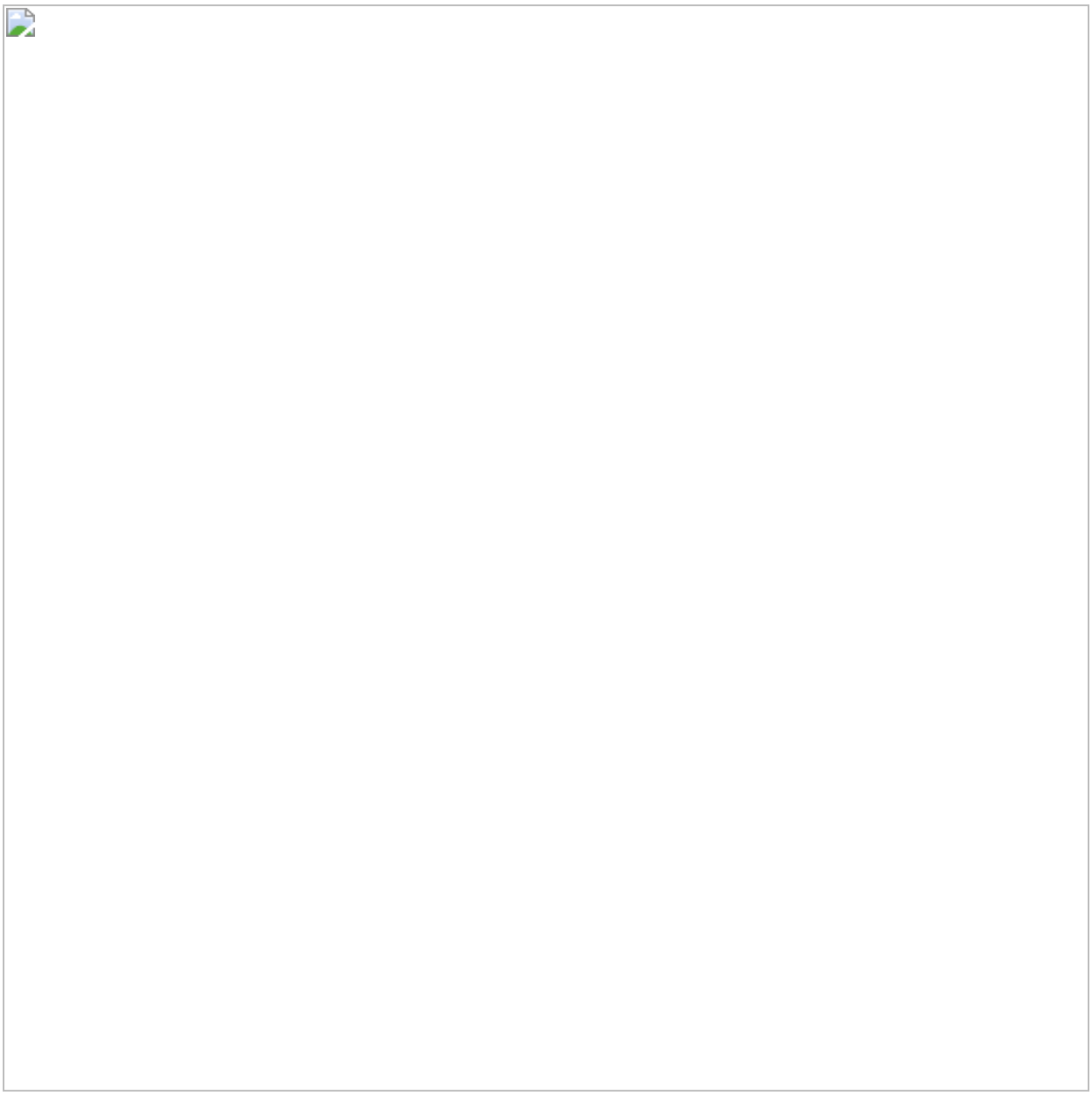


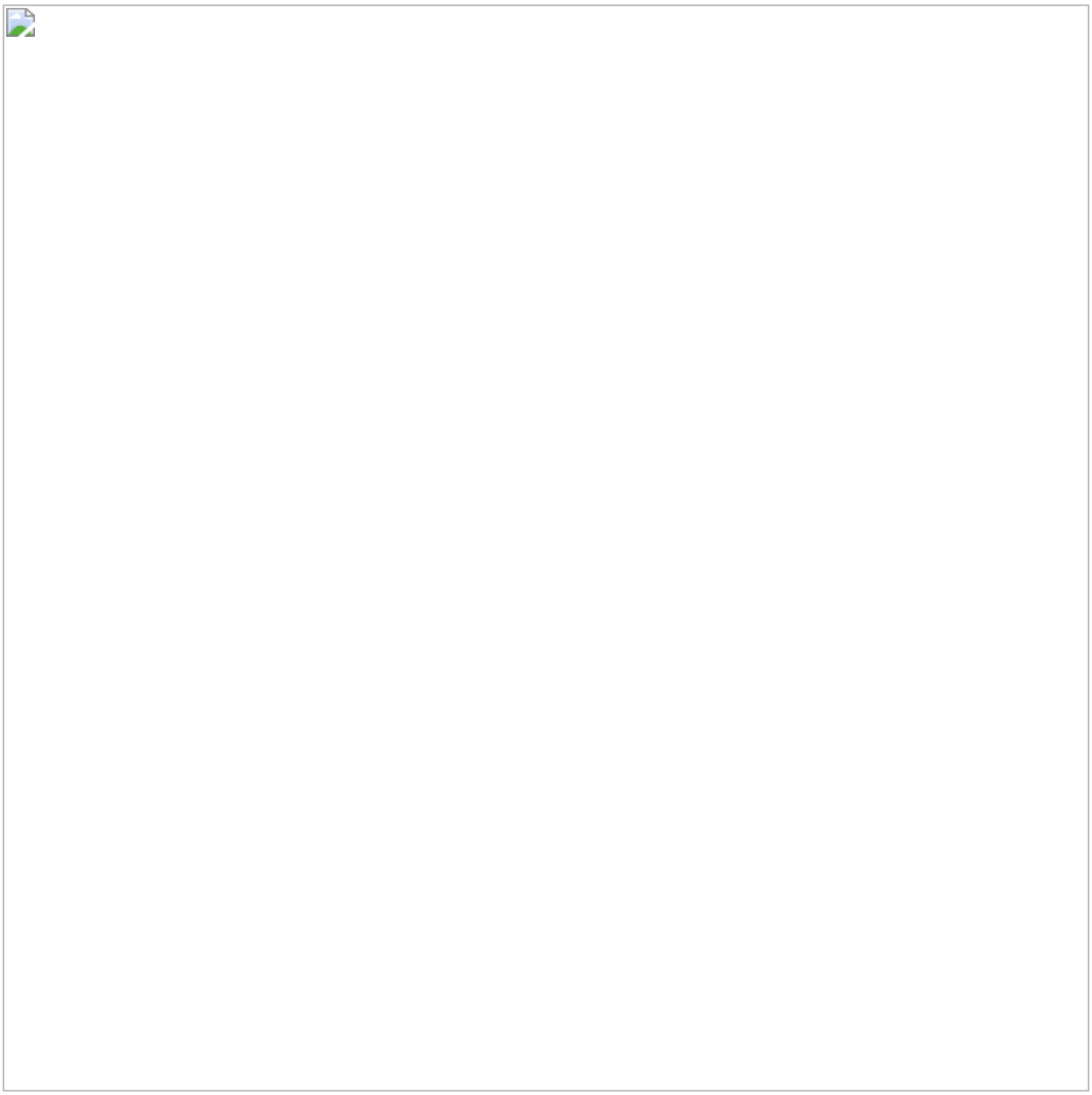
The next important step is grass. I use the RVT base approach and geometry-type grass for all my scenes. I found tutorials from Marpetak Dev about it on YouTube. In this tutorial, he shows how to create geometry-based grass. And in this tutorial, he talks about setting up RVT and about RVT grass material creation.

I created the landscape material in a very simple way. I didn't add any textures, only solid colors. RVT textures are very tricky and often they don't work right, so if something doesn't work, it's better to set up everything one more time from the beginning.



We need flowers! We can't imagine a beautiful sunny grassy field without them. I created the base color and mask texture in Photoshop using the same approach I showed in the tree creation tutorial. I created a plane in Maya and applied material to it with my base color texture, cut the flowers, and made from this part the final flower. For good shading in the engine, we need to remember to set up vertex normals looking up.

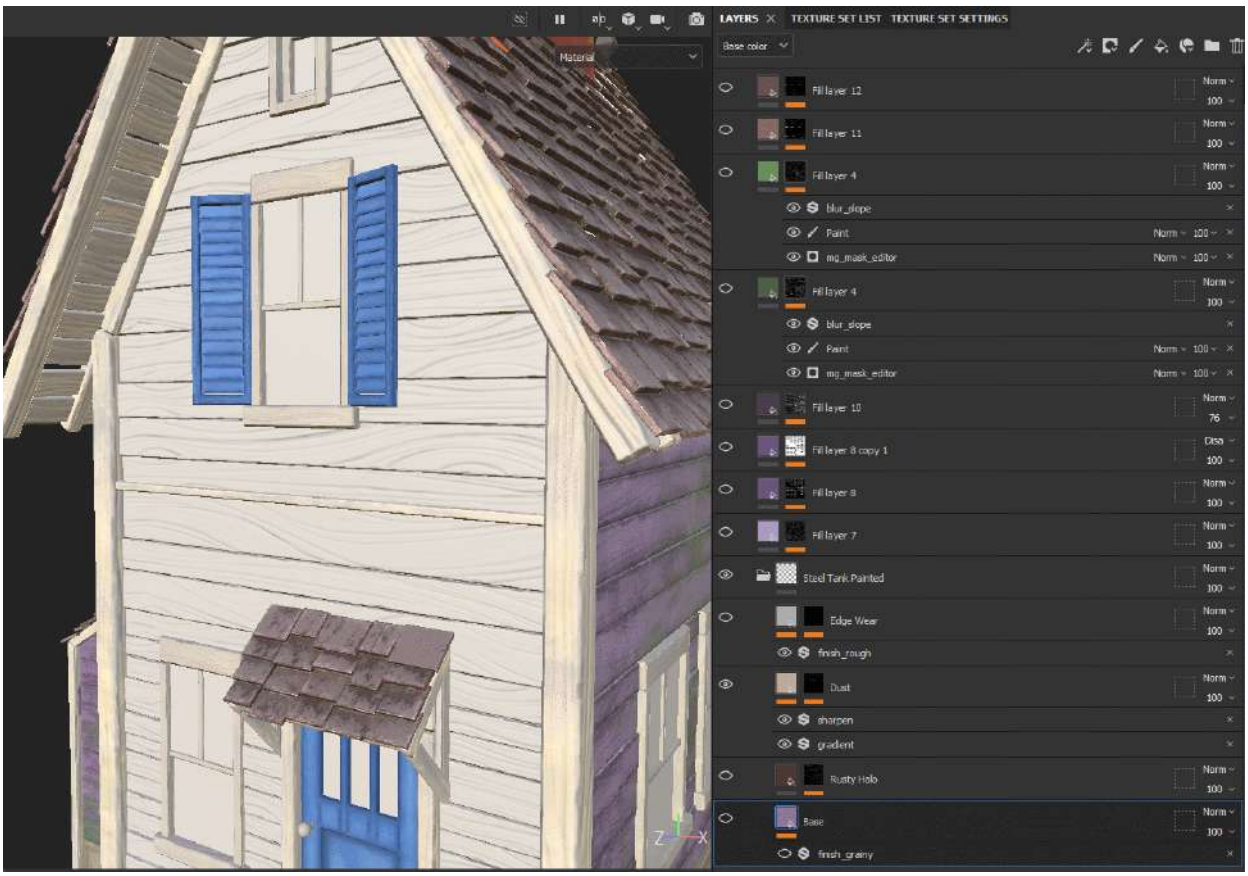


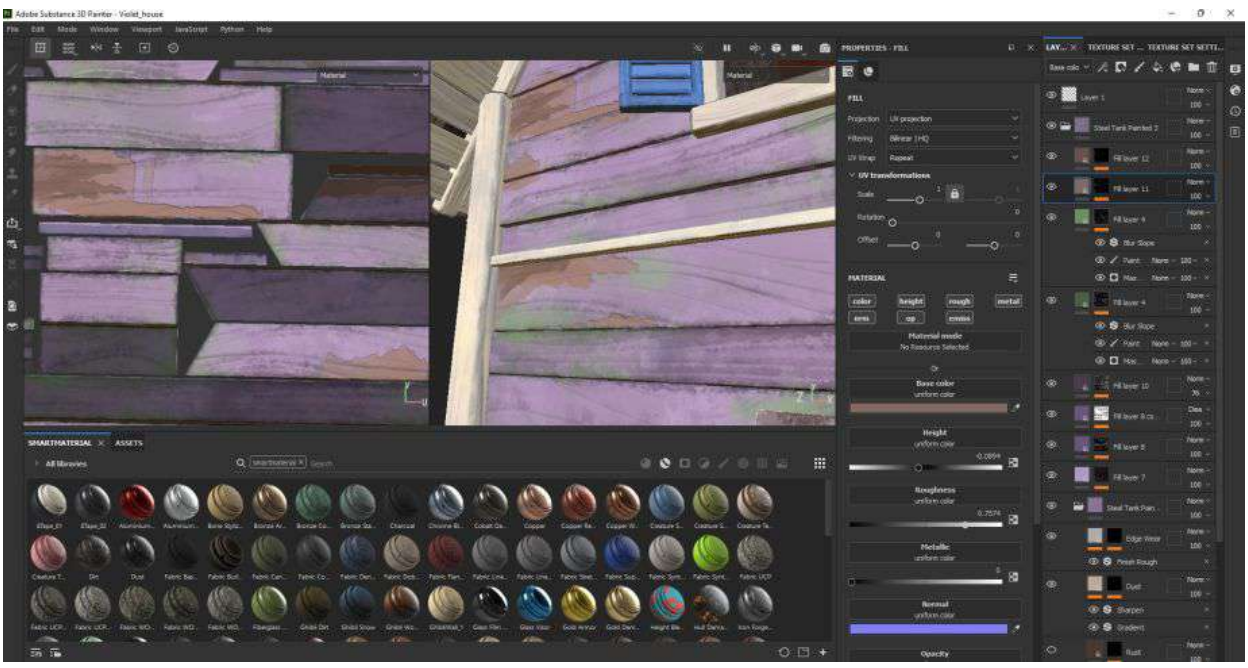


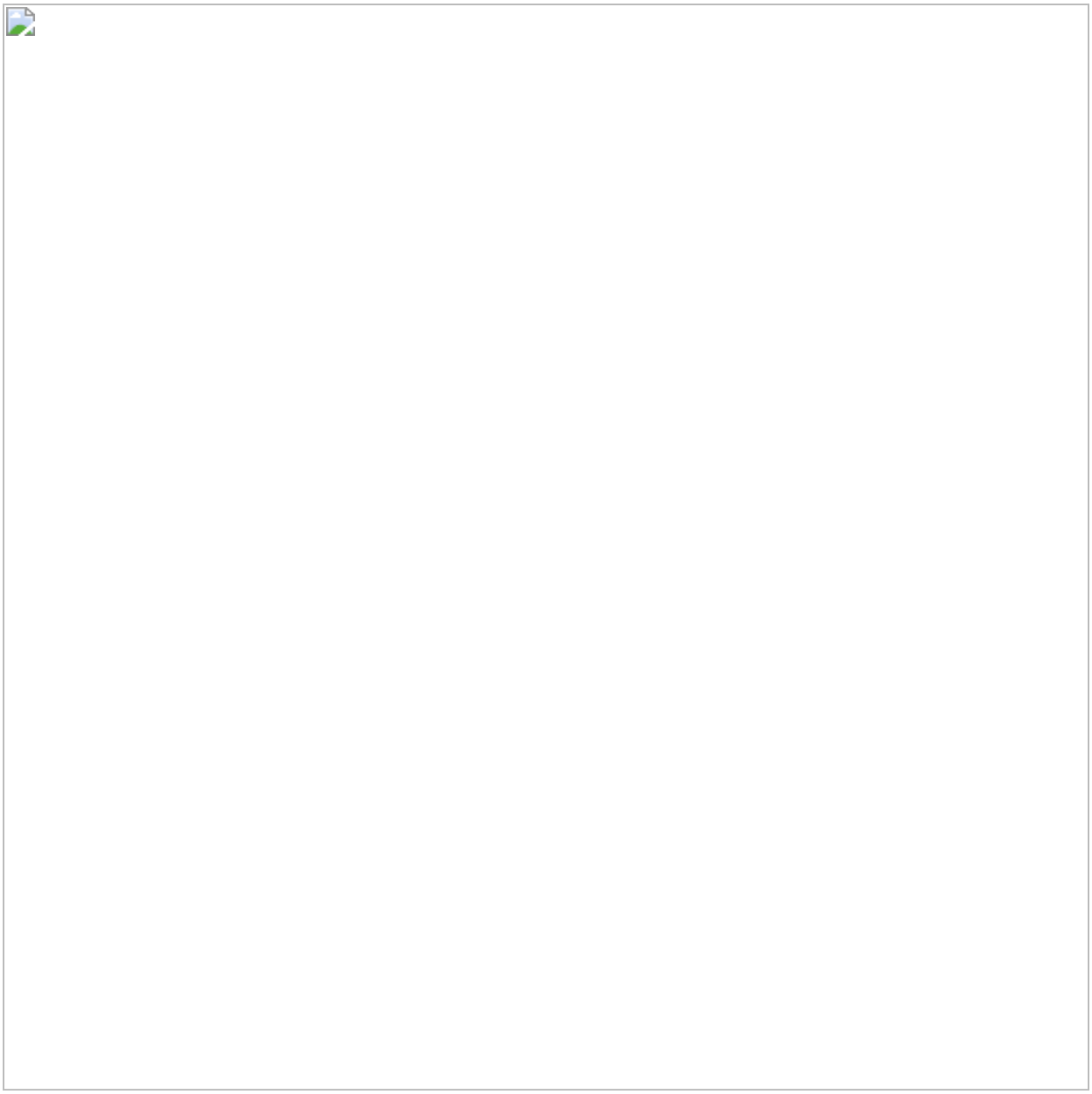
Texturing

I had an exact reference for this work and tried to follow it. From the beginning, I understood that making everything unique is the best decision because this house has some special damage to it. I could add it after by using decals, etc., but I wanted to texture everything in Substance 3D Painter.

I baked normal maps and AO textures for all the house parts in Marmoset Toolbag 4. Then, I imported everything to Substance 3D Painter and bake all other texture maps using previously baked ones. When I texture objects, I use a combination of fill layers with different smart masks. To achieve the painted style, I added a slope blur filter with different intensities.





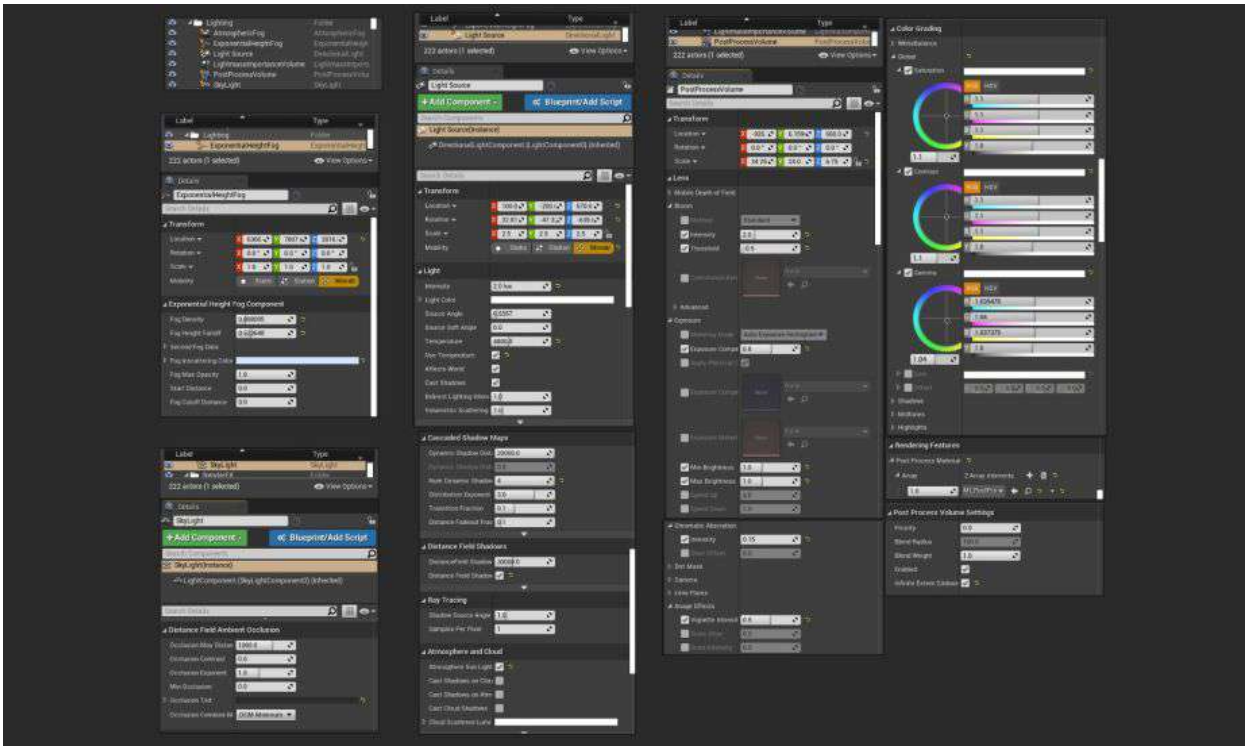


When I finished the scene, I wanted to show how flexible my approach to world creation is. I decided to create the autumn scenario. I googled some autumn color schemes on the internet, which determined my color palette. To choose the house color, I used the color wheel scheme. Because blue is the opposite of orange, it would match perfectly. To change the color of all elements, I created new material instances and tweaked the tinting parameter on them. Because trees have more contrasting colors, I made the grass less saturated and less contrasted. This change helped to separate one from another and add more depth to the scene.


Color
Palettes

Lighting

I started this project in Unreal Engine 4 because I think it's more reliable than UE5 for now and because Lumen doesn't work well with stylized environments as it gives very dark shadows that are hard to fight. I tried to copy the atmosphere and light as in the reference. One of the problems I had was a shadow on the building cast by the leaves. But after I applied my stylized fluffy tree shader on the trees, there was only one hitch left to create the right leaf shape mask. I didn't use any LUT textures here for color correction. I only used Crisp and Detailed post-process material by Dominique Buttiens. You can find it in the ArtStation marketplace.



To make the environment feel alive, I like adding different visual effects. For the summer environment, butterflies flew in other parts of the scene. I used a simple butterfly particle system created by Brandy Jahraus. In the autumn scene, there were flying leaves from the nearest trees. To create this effect, I used a tutorial by Dean Ashford on YouTube. There were also some stylized wind effects behind both. I saw how to make it in another YouTube tutorial by Dean Ashford. To make grass, trees, and bushes move, I added a Simple Grass Wind node to all materials.

Conclusion

The biggest challenge for this project was to reproduce these paintery-looking trees. I'm happy that I did this. And I am so glad that this method is helpful for other artists to create their environments. What's interesting is after I finished half of this scene, I decided to quit and stopped working on this scene. But some time after, I got back to it, and now I have a beautiful portfolio piece.

Never give up. Keep things simple. Enjoy the working process. Ask for feedback. These are the main tips I can share. I hope this article was helpful. I wish you all good luck with your future environments!

Victoria Zavhorodnia, Environment Artist

Interview conducted by Theodore McKenzie

This content is brought to you by 80 Level in collaboration with Unreal Engine. We strive to highlight the best stories in the gamedev and art industries. You can read more **Unreal Engine interviews** with developers here.

80 Level Talent

Your best job hunting tool is here

Keep reading

You may find these articles interesting

This is normal 2: Baking normal maps.

artstation.com/blogs/typhen/BDr6/this-is-normal-2-baking-normal-maps

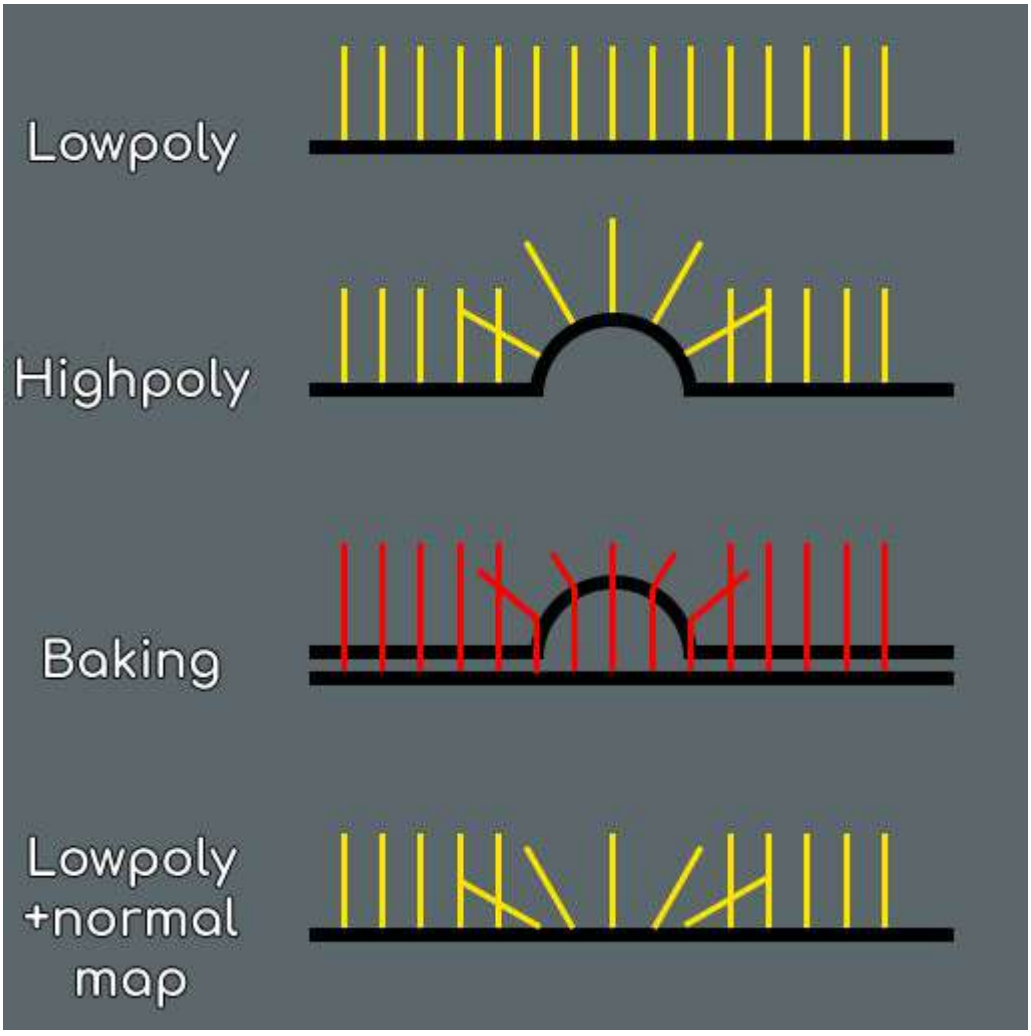
byCarlos Lemos inTutorial• 4 years ago

This is the second part of a tutorial series Im doing about normal mapping. You can find the part one here, but its not required to understand this part.

Baking normal maps

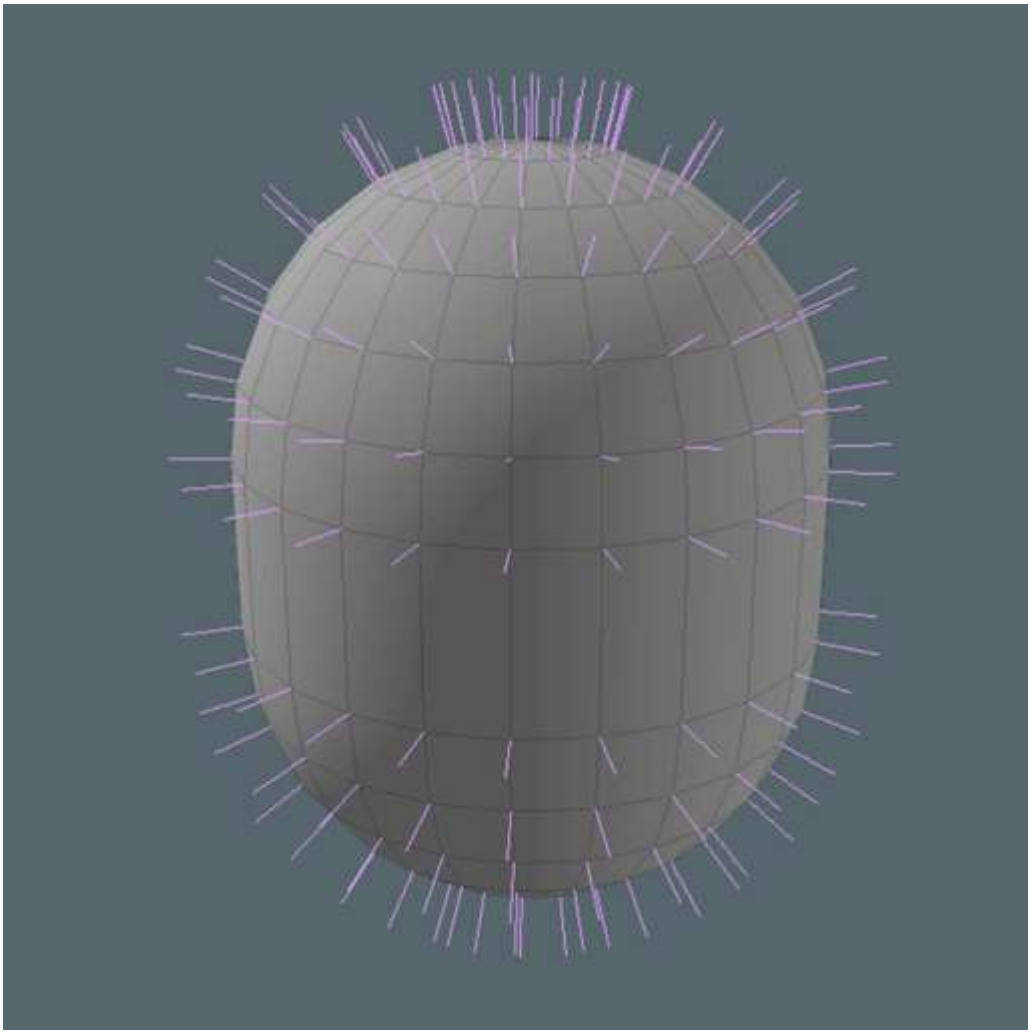
The general idea of baking a normal map is relatively simple: you have a lowpoly with UVs and a highpoly model; and you transfer the normal information from the highpoly to the lowpoly. This way, the lowpoly will bounce light as the highpoly would.

During this process, the baking program will basically cast rays from the lowpoly, following the vertex normals and searching for the higpoly. This is the most important aspect of normal mapping, and most problems people have when working with normal maps are related to this.
If you don´t control the vertex normals of your lowpoly model, you will lose control over your normal map.



In order to control the smoothing of our lowpoly model, we can have split vertex normals (to create hard edges) or averaged vertex normals (to create soft edges).

Turns out that not all 3D programs use the same calculations to average the vertex normals. This means that your lowpoly will look different and have its vertex normals pointing at slightly different locations depending on your 3D program. This isn´t usually a big problem, since these deviations are very small, but can affect how your model looks; and these differences are exagerated when using normal maps, since your normal maps are modifying the lowpoly normals that are changing between applications.



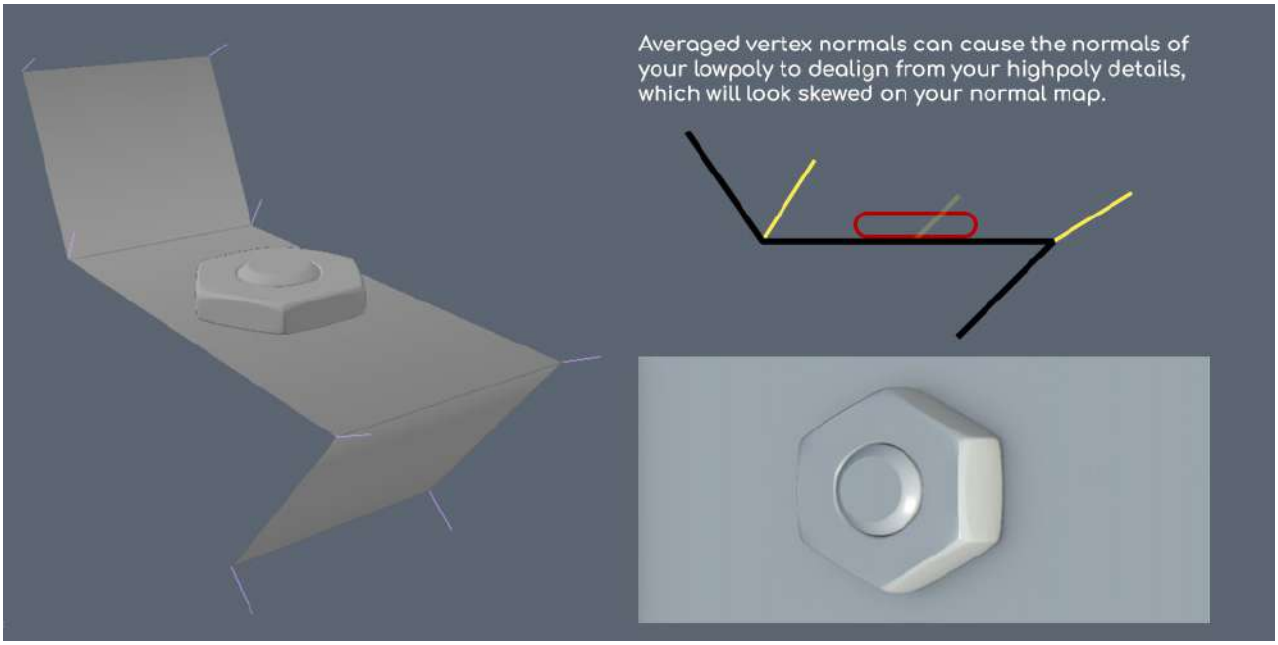
The 3D industry is working to fix this problem, and a standard has recently appeared, called Mikk space. This is a method of calculating vertex normals that all 3D apps could use, so vertex normal don´t change between 3D programs. Keep in mind that not all 3d apps use it yet.

Another way to reduce this is effect is not to rely too much on normal maps when baking. Try to match your lowpoly more closely to the highpoly and use more hard edges on flat surfaces. This way, your normal map won't have to do all the work and these small deviations will be less noticeable.

The skewing problem

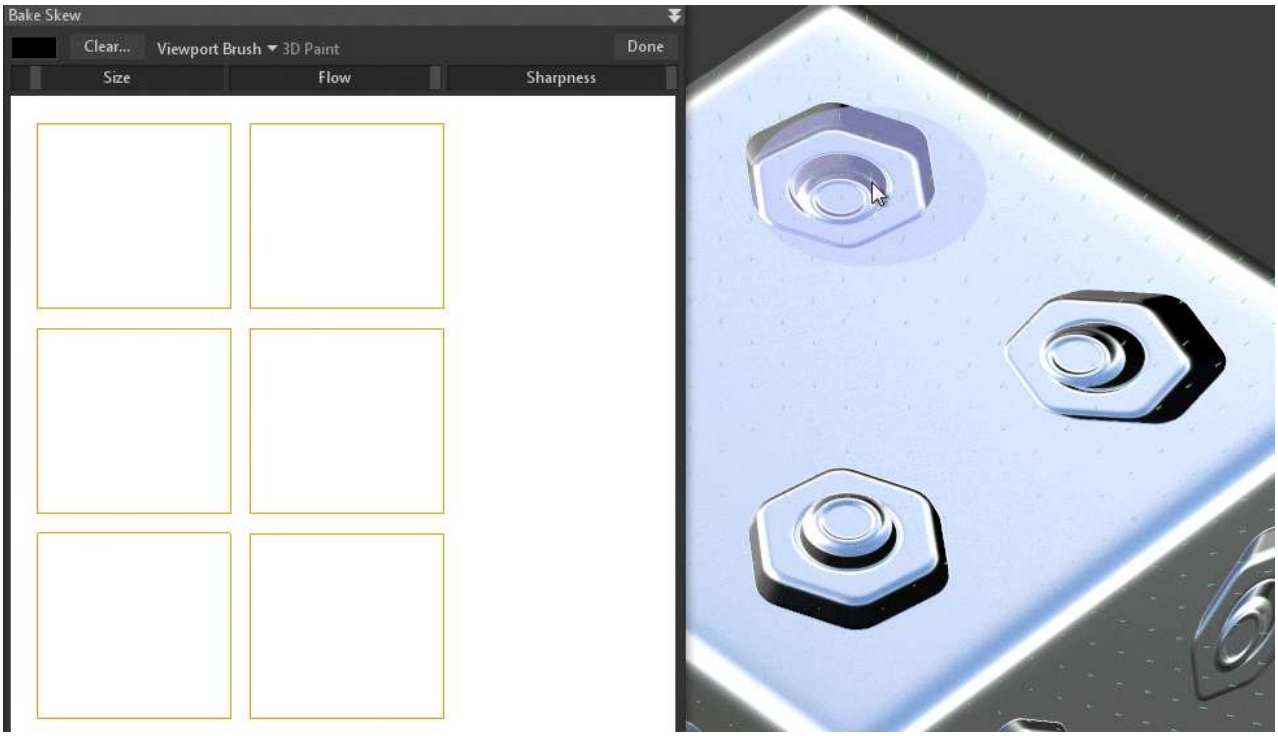
When the computer averages the normal direction of your lowpoly vertex normals, big changes in the angles of your surface can "skew" the lowpoly normals and they won't be perpendicular to the lowpoly surface.

Since the normal map baker uses the lowpoly normal directions when searching for the highpoly details, if these directions are skewed; they will appear skewed on the normal map:

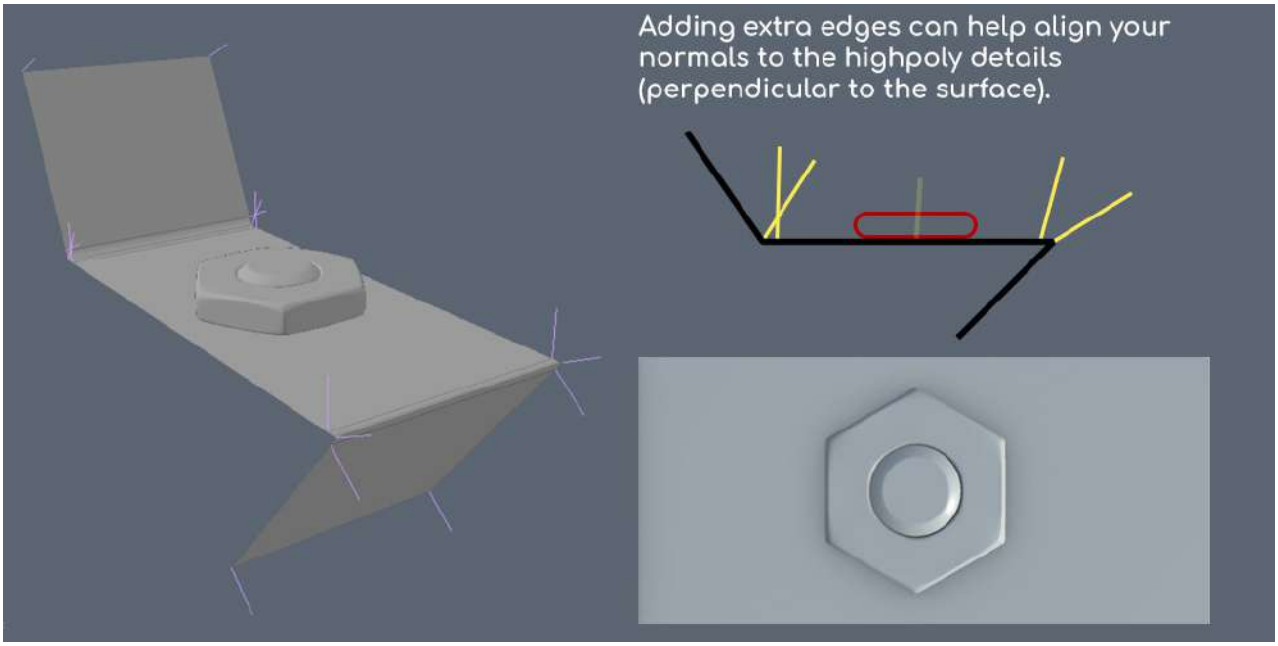


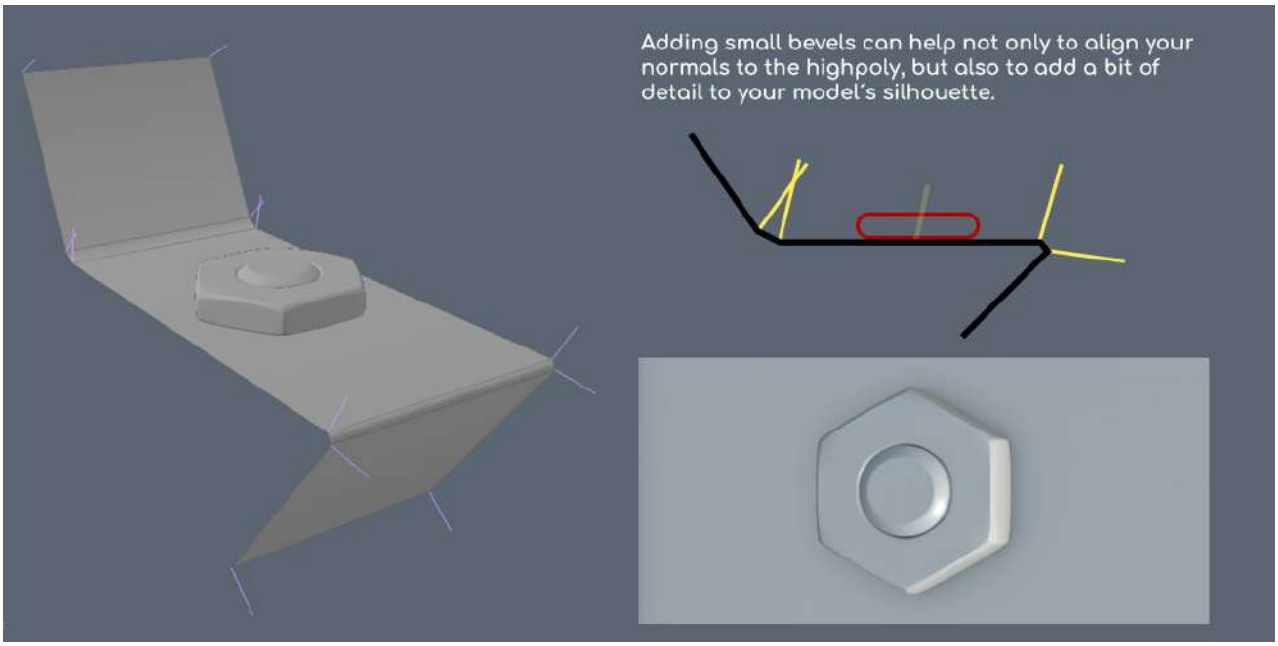
This is a very common problem, and several solutions have been found. There isn't a best solution, it really depends on the geometry.

The first solution for the skewing problem is to rebake the problematic parts by **modifying the lowpoly normals temporarily**, so they are baked without skewing. Marmoset Toolbag has this option. Reddit user Tanagashi kindly explained to me that some programs such as xNormal can tesselate the lowpoly to add new vertices and make the normals perpendicular to the lowpoly surface, bake an object space normal map and then convert it to tangent space using the original lowpoly normals. Using this new normal map, the program can create masks to control where to use the original normal map and the one created from the tesselated lowpoly.

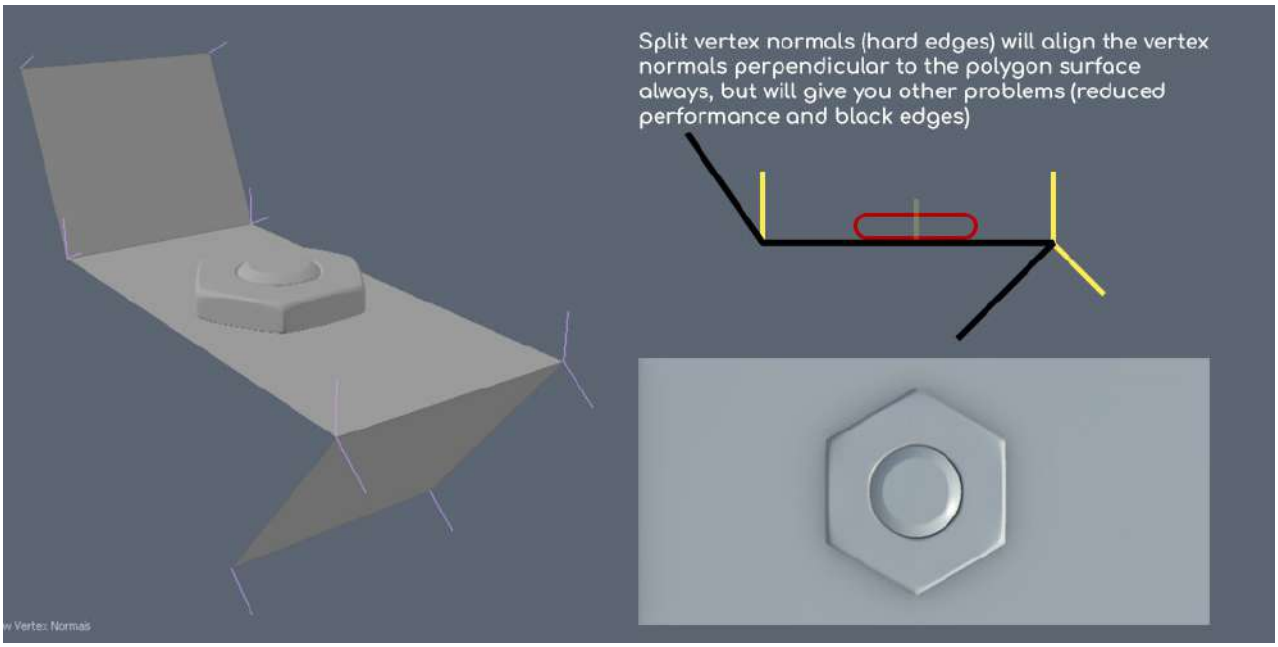


We can also reduce the skewing problem by adding vertices, as one 90º angle can be split into smaller degrees, making the second transition less skewed. This obviously increases your polycount and, since you are adding geometry, I reccomend you use this extra geometry to also add a more interesting silhouette to your model.



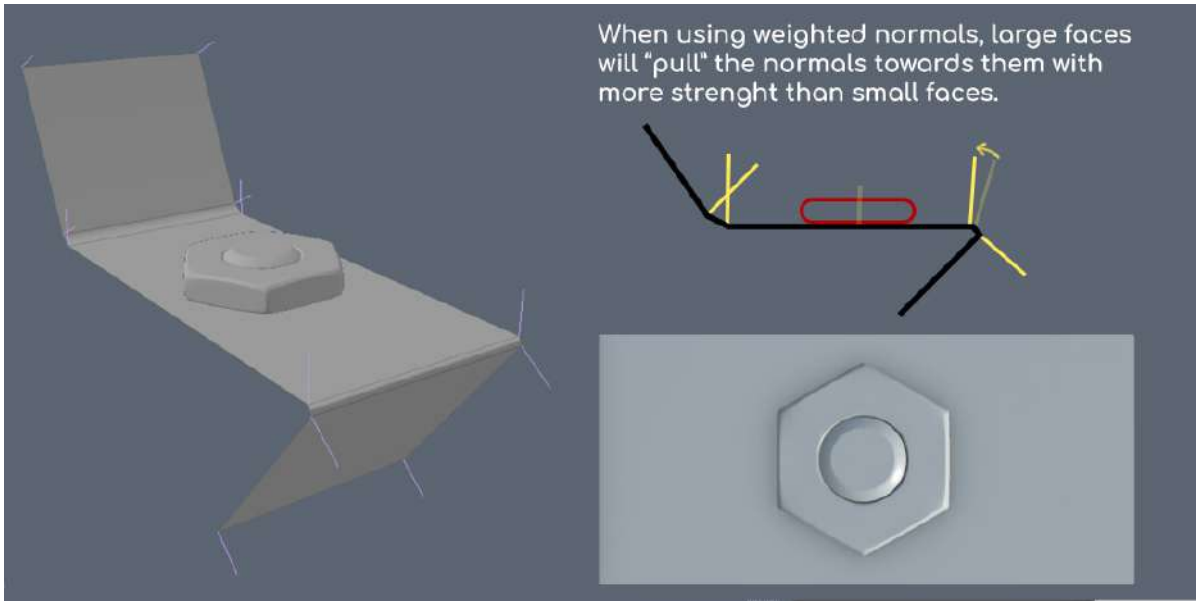


Another solution to the skewing problem is to **split the averaged vertex normals (making the edge hard/use separate smoothing groups):** this way, each vertex will have several normals, each one perpendicular to the lowpoly surface. Keep in mind that, when the 3D program has a split vertex normal; it actually creates a duplicate of the vertex, so this will increase your vertex count and slightly decrease performance. Additionally, hard edges could also give you a "black edge" problem, as we will see later.

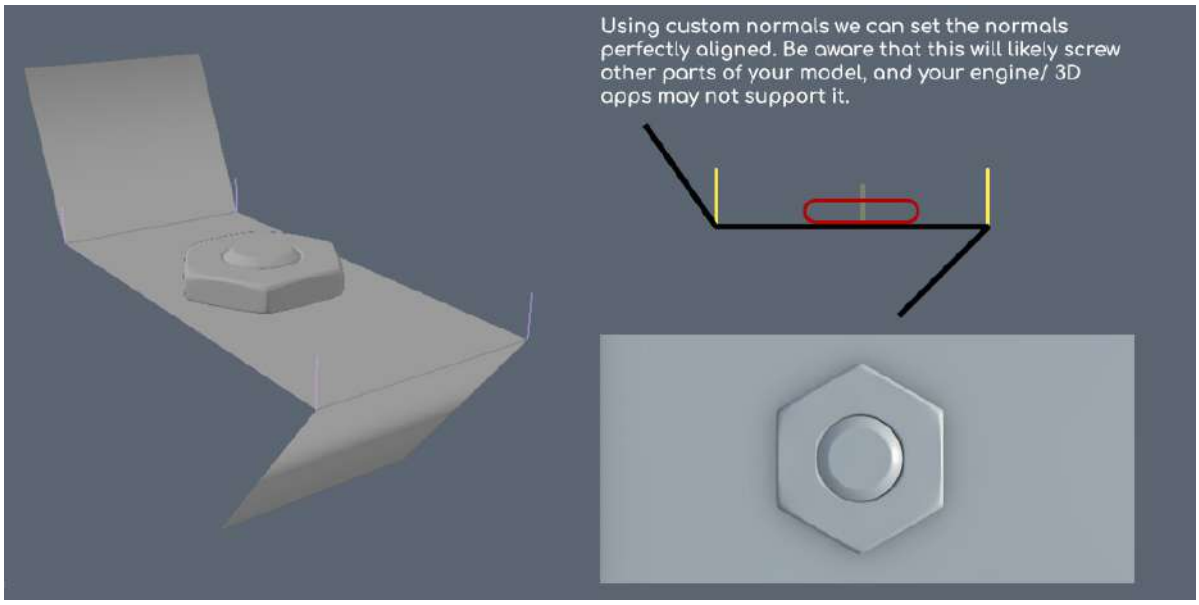


Finally, we can also reduce the skewing problemby **modifying the normals of our models** to bend the normals of our lowpoly so they are perpendicular to the highpoly details. Keep in mind that not all programs allow modified normals (Zbrush only has averaged normals, OBJ and older FBX files don't have custom normal information). There are basically 2 ways of modifying the normals:

- **Weighted normals:** this is an automatic method similar to the average vertex normals. The idea here is that when averaging the vertex normals not all faces will have the same strenght: larger faces will "pull" the vertex normals towards them with more strenght than smaller faces. This way, larger faces (which are usually more important) will have better detail projection. This works specially well with highpoly panels.

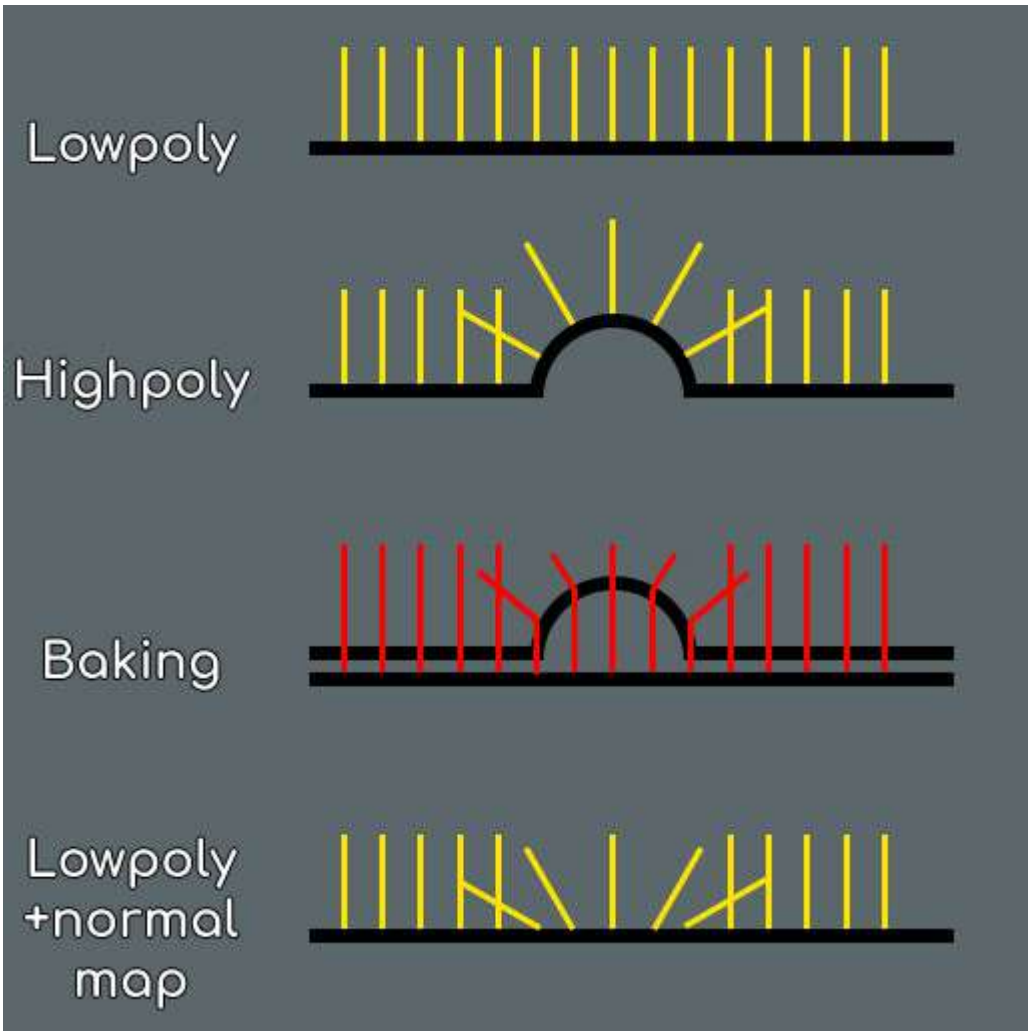


- **Custom normals:** using tools from your 3D software, you can bend your lowpoly normals. This is a relatively new idea and there aren't many standarized tools for this. Keep in mind that bending the normals can create very weird unintended shading on other parts of your model, so this technique is usually combined with bevels. Some people call this technique "midpoly modelling".

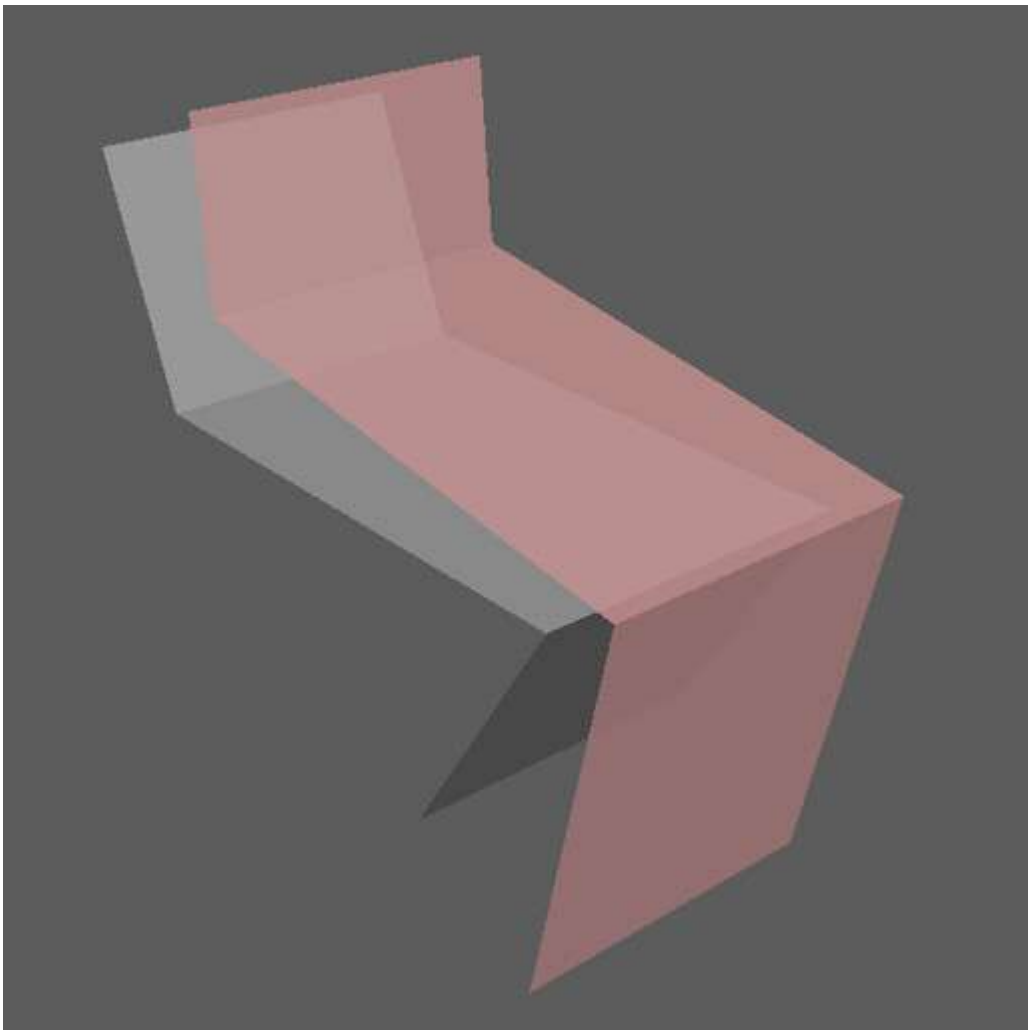


Bake distance

By default,the rays that are cast from the lowpoly surface travel a limited distance, to prevent the lowpoly from recieving normal information from far away parts of the highpoly. This distance is usually called "frontal/rear distance", as the rays can be casted towards the inside, outside of the model, or both. You can see this distance represented in red in the following image:



Some 3D apps (3ds Max for instance) also allow us to use a cage. A cage is a "copy" of our lowpoly model that we can modify so that it encapsulates the highpoly perfectly. And, in some cases (not all), also allow us to change the direction of the rays, without changing your original lowpoly vertex normals. This can help get the best baking extremes and avoid skews, but keep in mind that you are not baking using the normal direction of your vertex normals, but in the end you will use a normal map to modify the actual lowpoly normals, so the result could look strange.



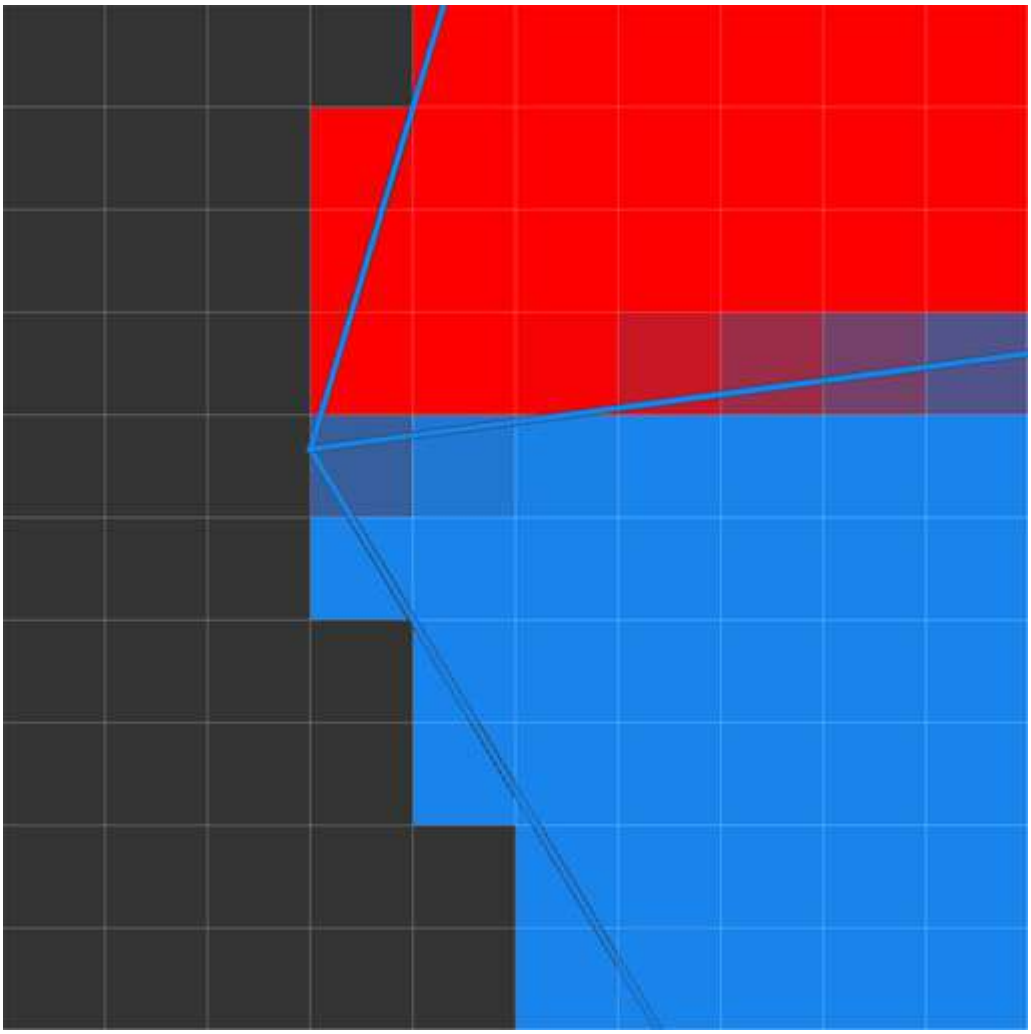
Bake seams

Sometimes, when baking normal maps, we can see some seams:

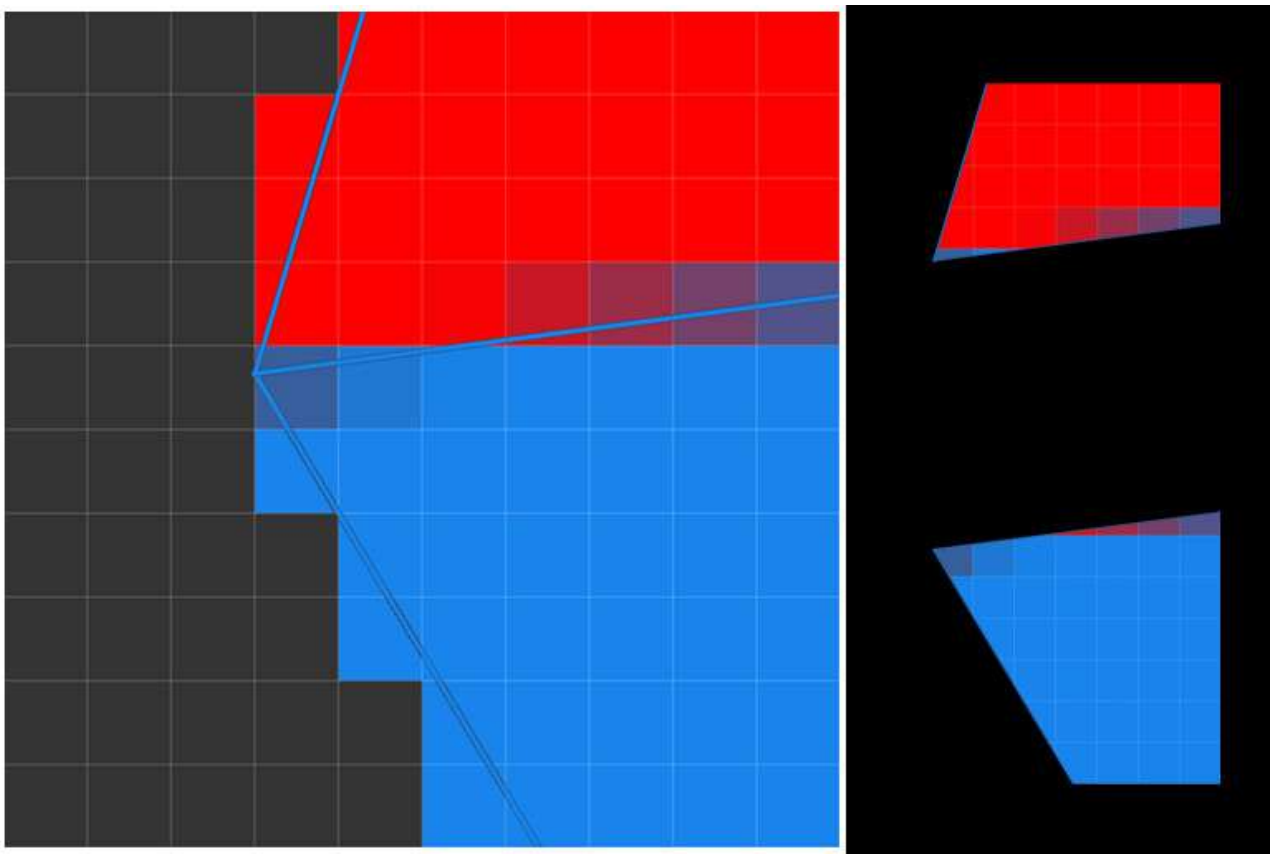
These seams appear when we have hard edges in the middle of a UV island. Why is this happening?

UV values can be very precise: we can have a vertex placed exactly at the coordinates 0.0001203123 U, and 0.340404021 V. However, textures are much less precise: pixels are limited, and they can't be split (we can't have half a pixel).

So, a certain pixel might be exactly at 0.001 and a vertex might fall into 0.00134 inside the UV space. When considering the color of the pixel in the normal map, we have to choose a color that best represents the normal direction of the faces inside it, so we make an average:



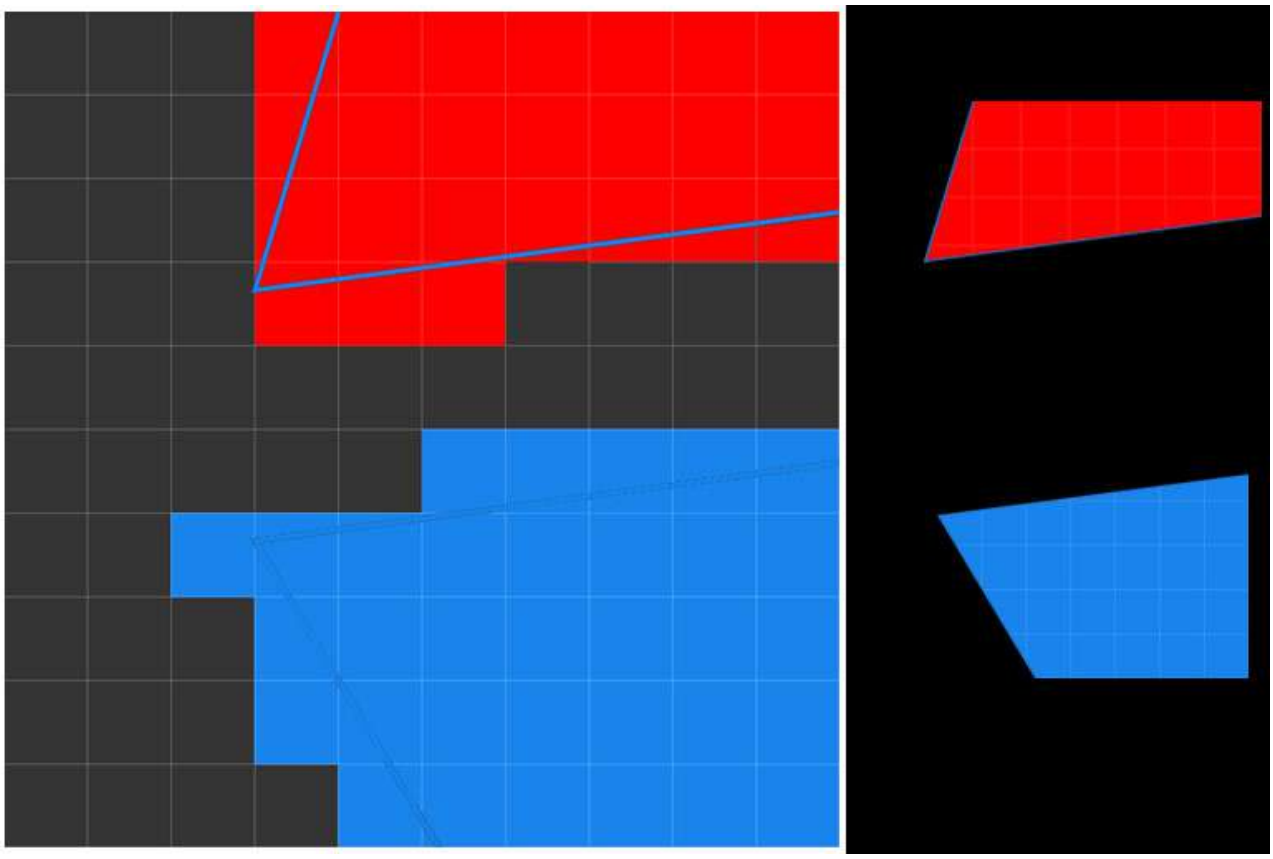
This is true for all baking processes, not just normal maps. However, this is much more noticeable with normal maps when we use hard edges. Normal maps indicate a direction and in the case of a hard edge, we have 2 faces looking in different directions, so they might have very different colors.



At the intersection between UV islands, the islands will compete to fill the pixels with their colors.This means that in the end, we will end up with pixels from island B modifying the colors of pixels from island A and vice-versa.

Islands separated by smooth edges look very similar on the normal map (because their normal directions are averaged), so a small difference in color is not very noticeable. However, islands separated by hard edges usually look very different on the normal map (because vertex normals are split and each face is looking in a different direction), and the difference can be noticeable, generating along the hard edges.

If we split the islands, they won't be competing for the pixel color: each island will be able to modify the entire pixel to fit its normal direction. This is why you should split apart the faces in the UVs when they are connected by a hard edge.



This video might also help you understand this process.

The rule of thumb is very simple: whenever you have a hard edge on your model, separate the faces connected by it in your UVs.

In conclusion:

Once I have my lowpoly model ready and adjusted to the highpoly model as close as possible, I start working on the smoothing before UVs.
I set my smoothing for the lowpoly (if its organic, I start with a completely smooth model, if its hard surface I start with a set angle smooth of 30-60º; and tweak the model smoothing until it looks good).
Once I have a set smoothing for the model, I work on the UVs, making sure that all **hard edges** are split into separate UV islands (to avoid edge seams).
If I have skewing errors, I add additional edges (usually bevels, to keep a more rounded silhouette). This works for most of my models, but I could also fix the skewing errors if I used Marmoset Toolbag for baking, or by using custom/weighted normals.
If there are projection errors, I modify the baking distance/ cage, modify the lopoly/highpoly so they are better fit for baking, or erase the normal maps on certain, really hard parts such as the tip of a cone.

Next up, I´ll be making a troubleshoothing guide for normal map baking and discuss some of the most common problems and solutions. If you are enjoying these tutorials, please comment so I have some feedback, even if its negative. I´m doing this so I can learn and improve, but a complete silence can be discouraging. Thank you for your time, and I hope you are enjoying these!

Part 1: What normal maps are and how they work.

Part 3: Types of normal maps.

Part 4: Normal map troubleshooting guide.

Part 5: Normal map workflows.

PDF file (Gumroad) / PDF file (Artstation)

Tutorial: How Normal Maps Work & Baking Process

80.lv/articles/tutorial-how-normal-maps-work-baking-process

Carlos Lemos March 20, 2020

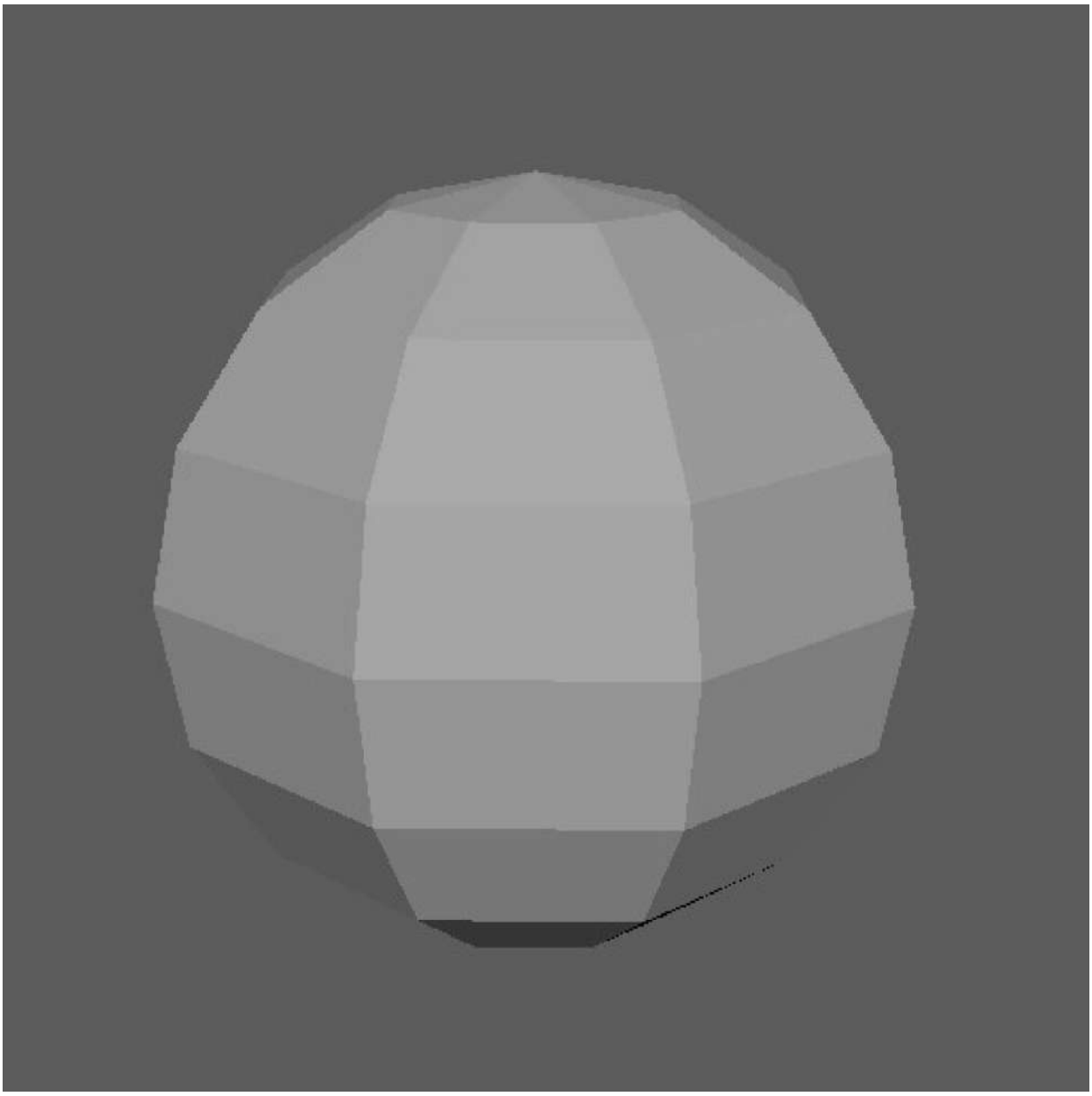
Carlos Lemos has recently finished his 4-part tutorial on normal mapping and kindly allowed us to repost it on 80 Level (originally posted on Artstation). In this article, read about what normal maps are and the process of baking them.

Part 1. What Normal Maps Are and How They Work

Throughout the years, I have been trying to understand normal mapping and the problems that usually appear when working with them.

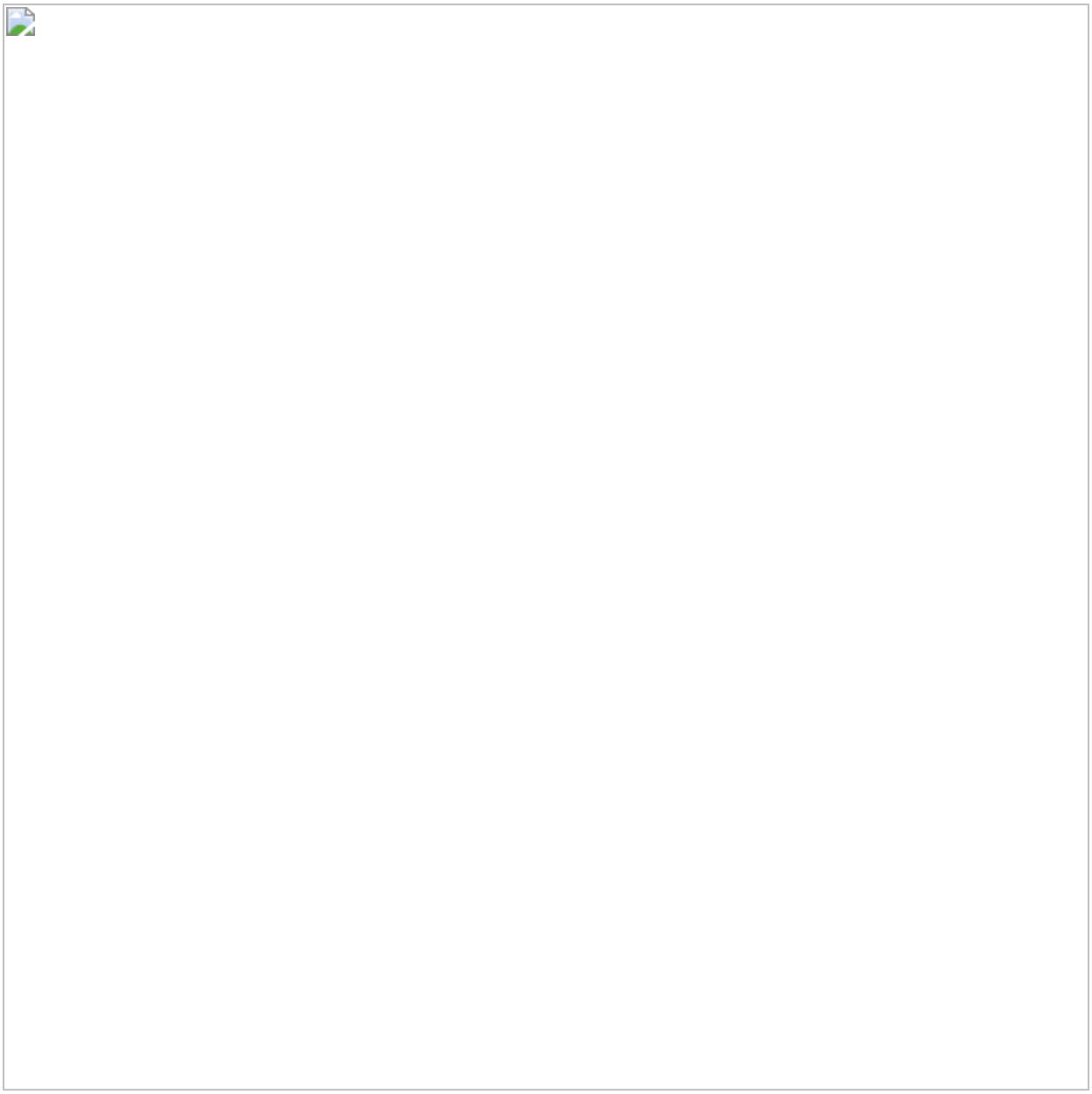
Most explanations I found were usually too technical, incomplete or too hard to understand for my taste, so I decided to give it a try and explain what I gathered so far. I recognize that these explanations may be also incomplete or not 100% accurate, but I want to try anyway.

The first 3D models ever made looked something like this:



We see an obvious limitation: it looked too polygonal.

The first obvious solution was to add more polygons, making the surface more even and smooth, up to a point where the polygons would look like a single, smooth surface. Turns out this needed a huge amount of polygons (especially at that time) in order to make surfaces such as a sphere look smooth.

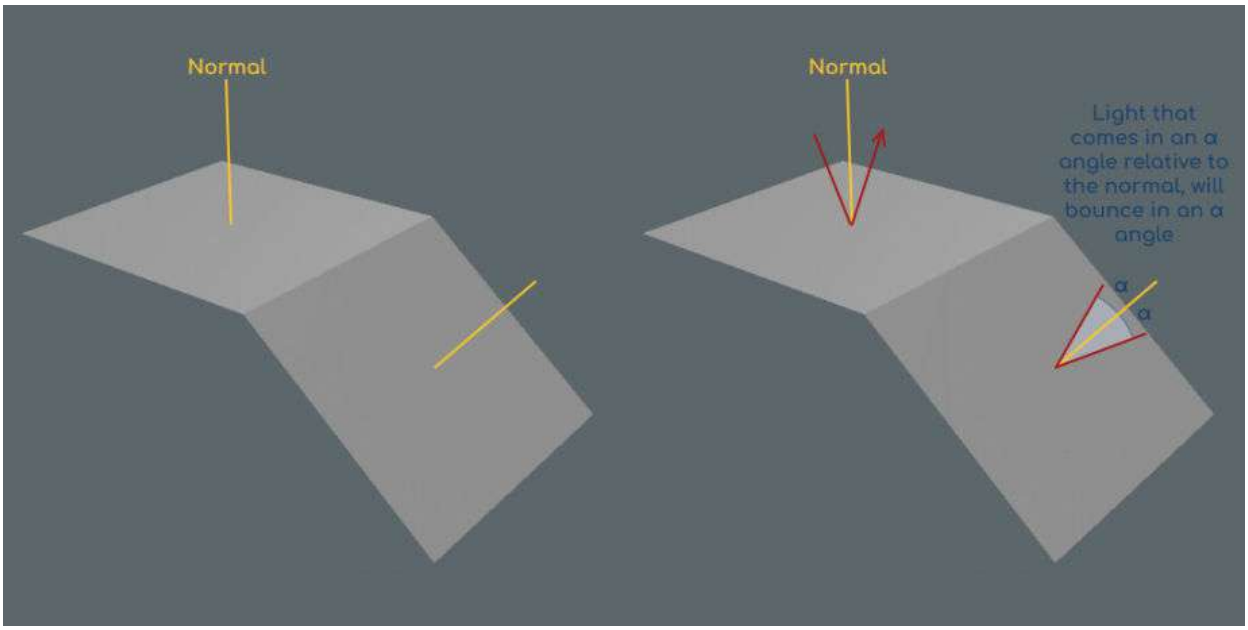


Another solution was needed, and thus normals were invented (not really, but it's easier to explain and understand that way).

Let's draw a line from the center of a polygon, completely perpendicular to its surface. We will call this line a super confusing name: normal.

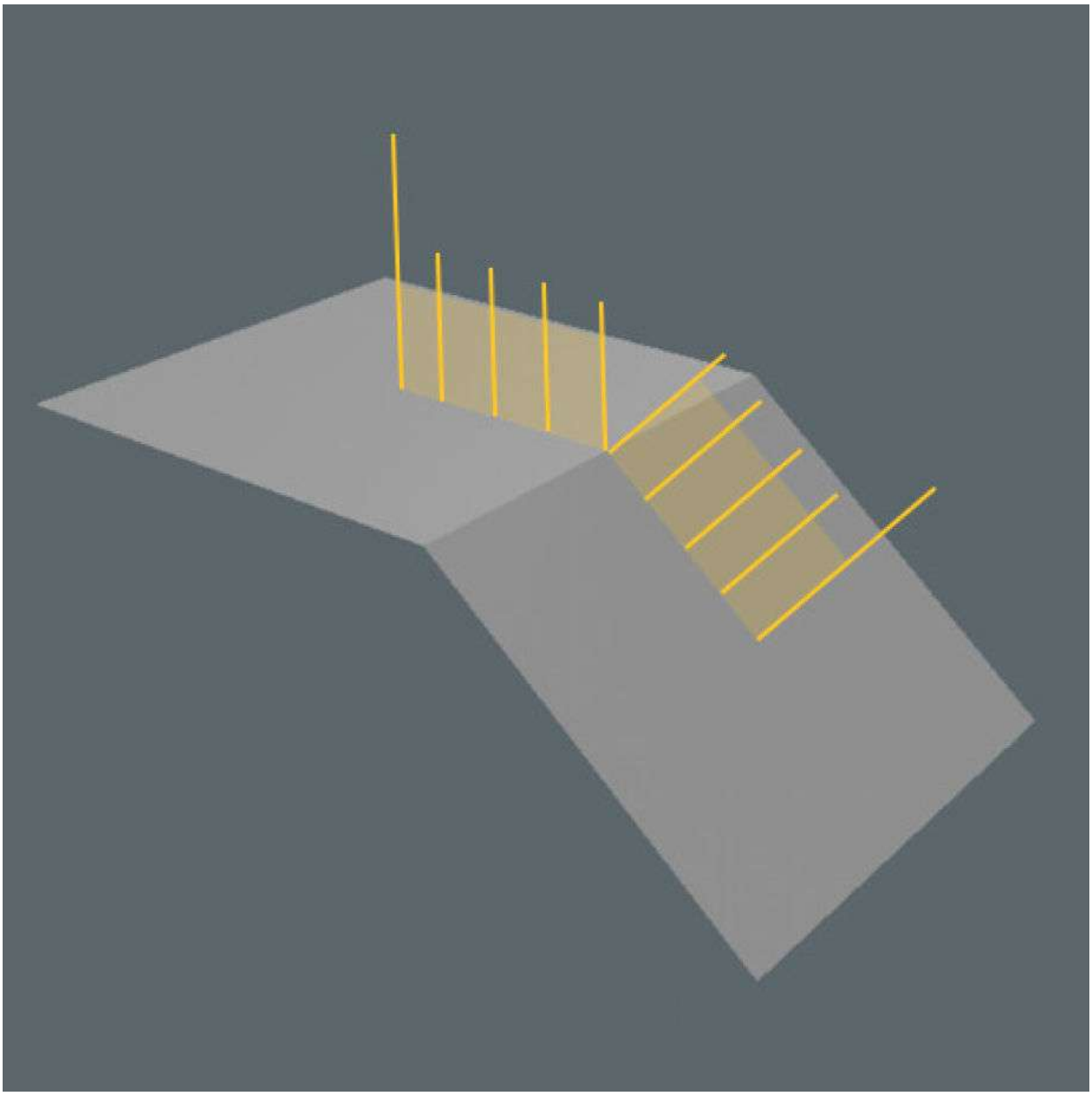
The purpose of this normal is to control where a surface is pointing at so that when light bounces from this surface, it will use this normal to calculate the resulting bounce.

When light hits a polygon, we compare the angle of the light ray to the normal of the polygon. Light will get bounced back using the same angle relative to the normal direction:

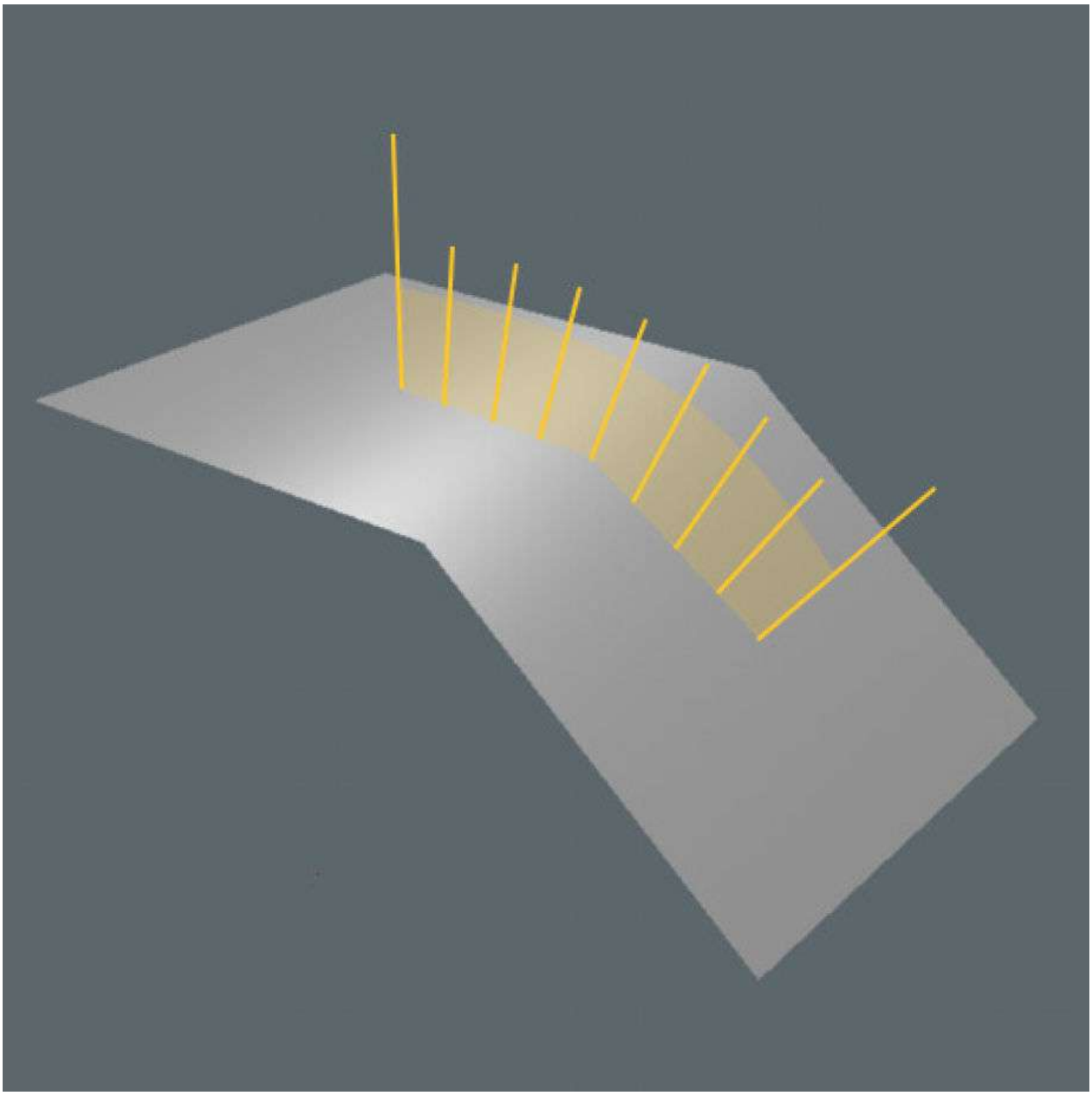


In other words, the light bounce will be symmetrical relative to the polygon normal. This is how most bounces work in the real world.

By default, all polygons bounce light rays completely perpendicularly to their surface (as they would in real life), because polygon normals are, by default, perpendicular to the polygon surface. If there are gaps in the normals, we will see them as separate surfaces, since light will bounce in either one direction or the other.



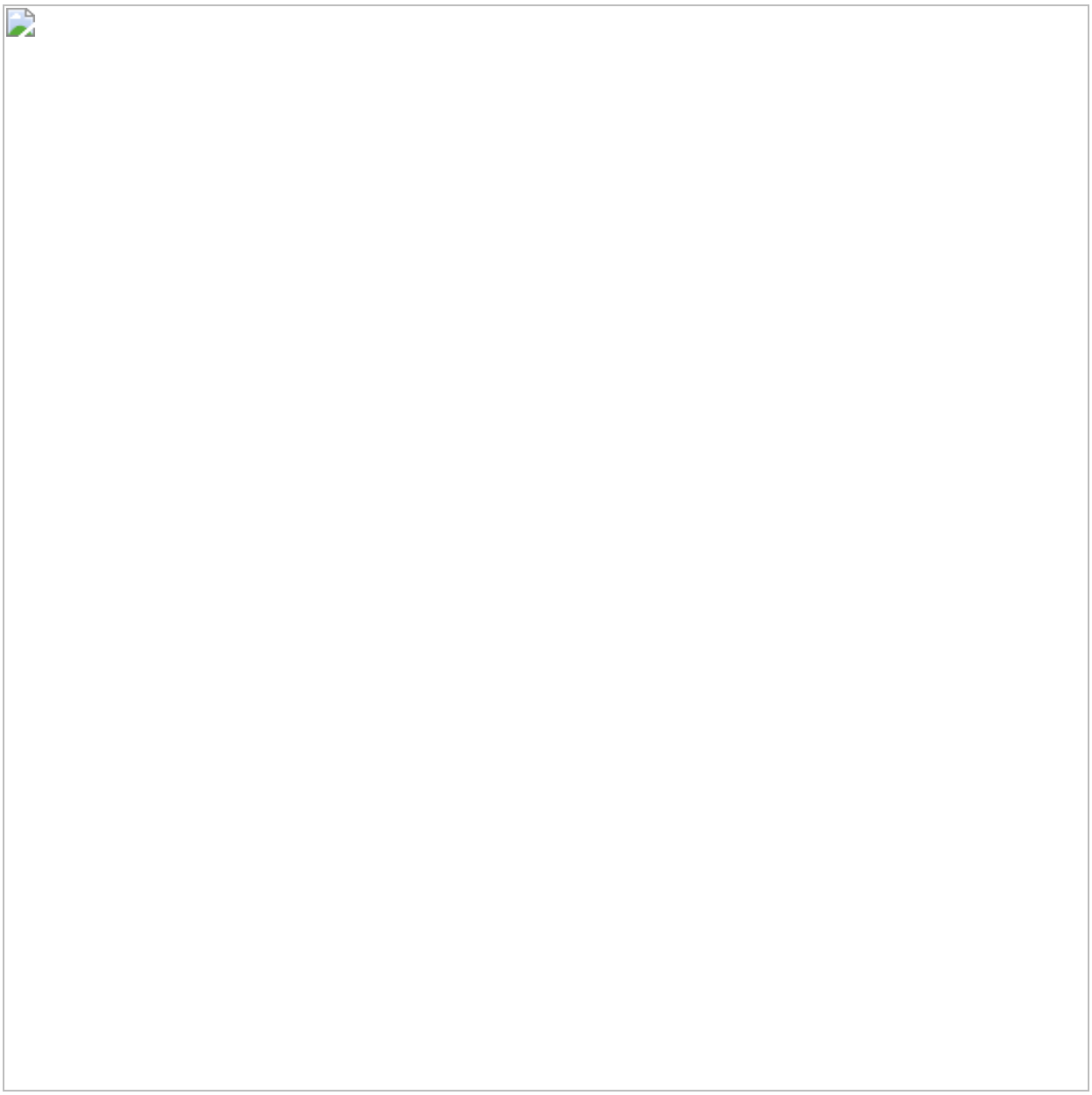
Now, if we have two faces connected, we can tell the computer to smooth the transition between the normal of one polygon and the other so that the normal gradually aligns with the closest polygon normal. This way, when light hits one polygon directly in its center, the light will bounce straight, following the normal direction. But, in between polygons, this normal direction is smoothed, bending how light bounces.



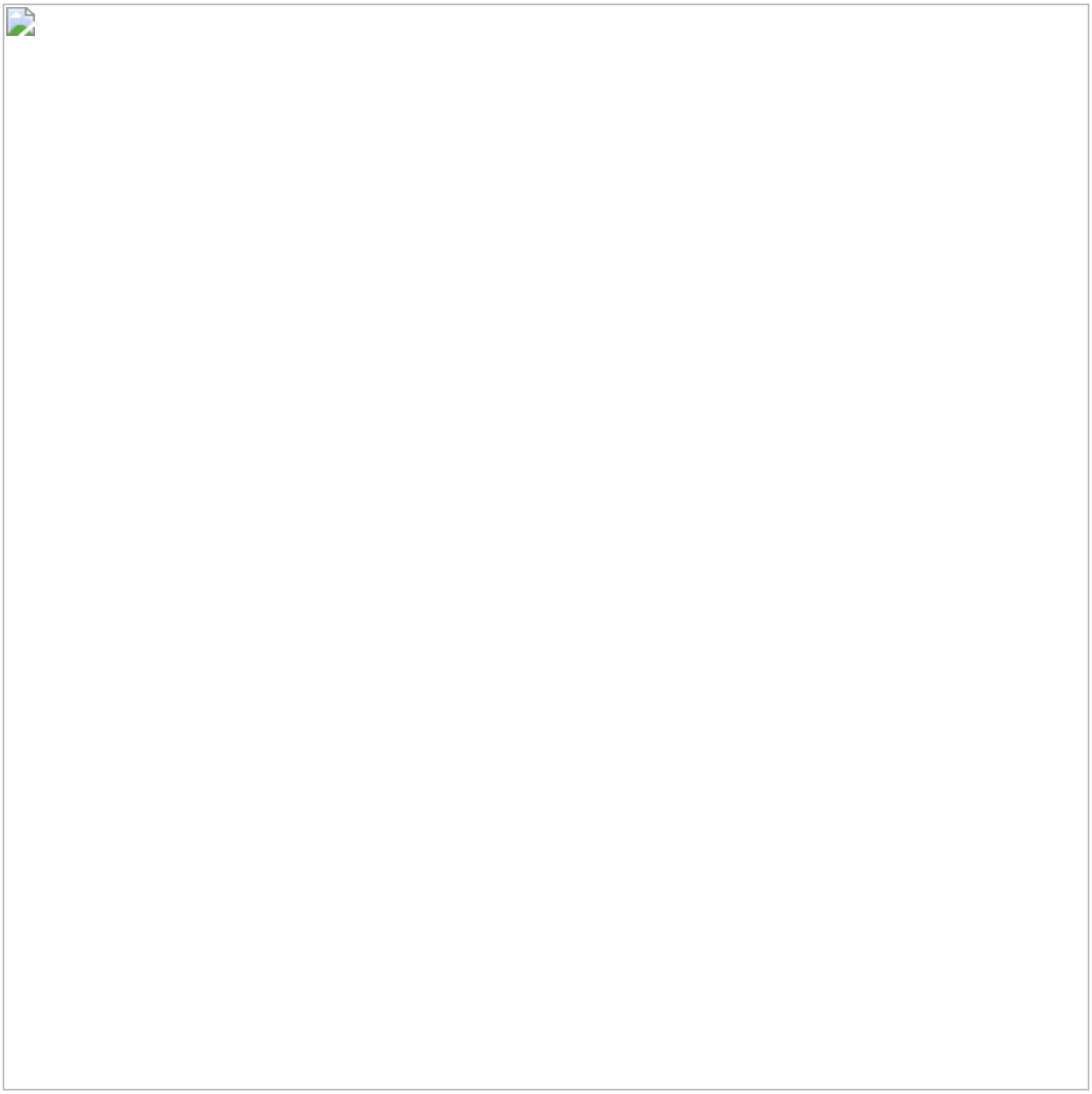
We will perceive the transition as a single surface, as the light will bounce from one polygon to the other in a smooth fashion, and there will be no gaps. Effectively, the light bounces from these polygons smoothly, as it would do if it had a ton of polygons.

This is what we are controlling when we set smoothing groups (3ds Max, Blender) or set edges to be hard or smooth (Modo, Maya): we are telling the program which transitions between faces we want to be smooth and which we want to be hard.

Here's a comparison of the same sphere with hard and smooth transitions, both with 288 polys:



We could potentially set something like a box so that all its vertices would have averaged normals. The 3D software will try to smooth its surface, so it'd look like a single, smooth surface. This makes perfect sense for the 3D program, but it looks very weird because we have something that should have several separate surfaces (each face of the box) but the program is trying to show it as a single, smooth surface.



This is why we usually have a smoothing angle setting in 3D software: if we have 2 connected polygons in an angle greater than this smoothing angle, their transition will be soft, and polygons connected in an angle smaller than the smoothing angle will be hard. This way, extreme angles between surfaces will be shown as different surfaces, as they would in the real world.

So, we used normals to control the transitions between faces in our model, but we can take this a step further.

Since we are changing how light bounces from an object, we can also make a very simple object bounce light as a complex one would. We use a texture to bend the direction of the light that bounces from a 3D object, making it look more complex than it really is.

A real-life example of this would be those holograms that were gifted back in the day with potato chips (at least here in Spain). These are completely flat, but bounce light in a way that's similar to how a 3D object would, making it look more complex than it really is. In

the 3D world these work even better, but still have some limitations (since the surface would still be flat).

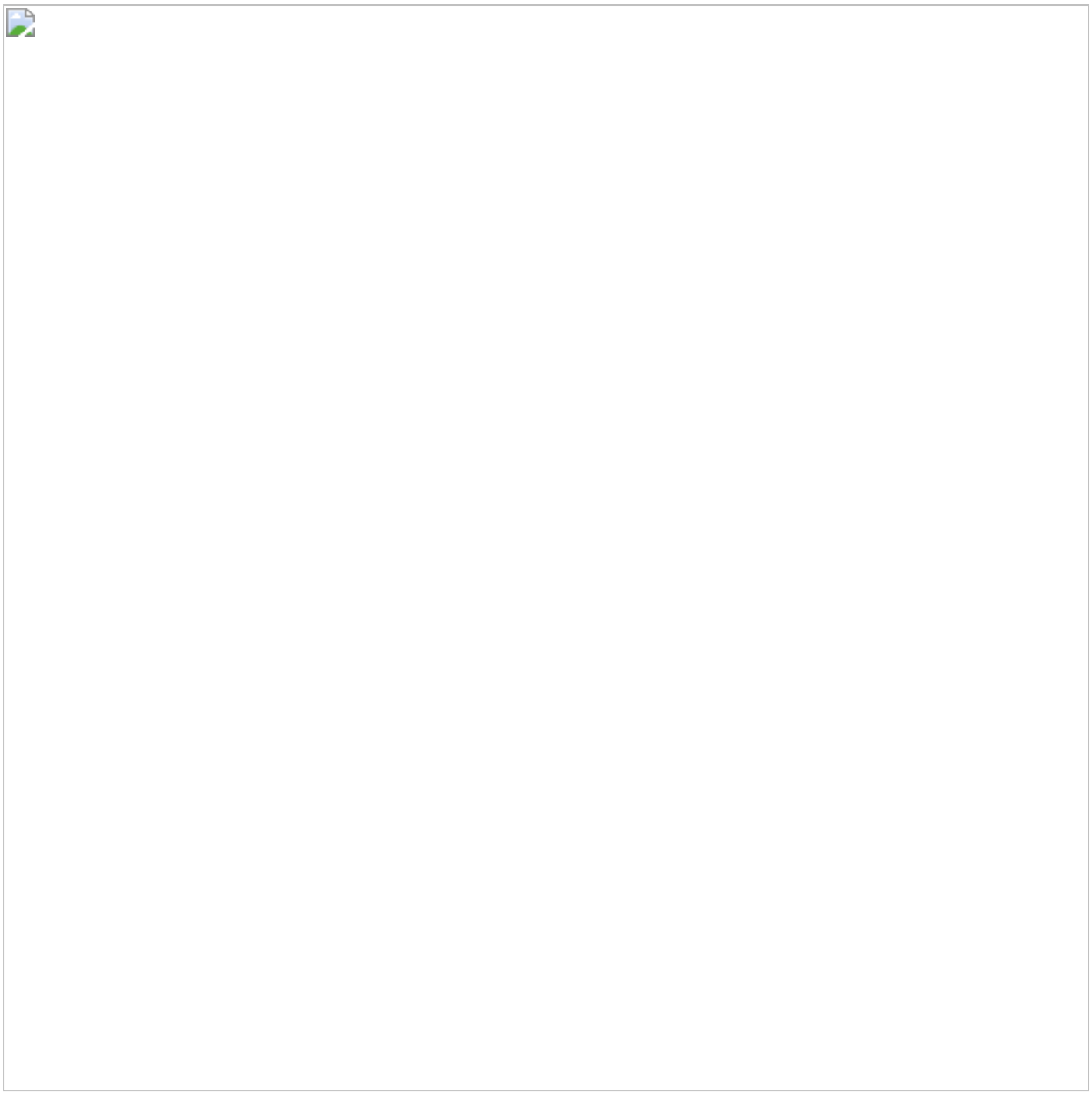
While we do use the normals of the polygons for some other black magic related things, we don't actually control the smoothing of our model surface using the polygon normals. We use the vertex normals to control the smoothing of our normals. This is basically the same idea, but a little bit more complex.

Each vertex can have one or more normals associated. If it has a single normal, we call it an averaged vertex normal and if it has more than one, we call it a split vertex normal.

Let's take two polygons connected by an edge. If the transition between the two faces is smooth (we set it to smooth in Maya/Modo, or they both have the same smoothing group in Max/Blender), each vertex has a single normal, which is the average of the polygon normals (this is why it's called averaged vertex normal).

Important note: up until very recently, each 3D program used its own method of calculating averaged vertex normals, which meant that normal maps calculated in one program might look completely different in another 3D program. I explain more about this in the second part of this tutorial (see below in this article).

If the transition is hard (hard edge or the smoothing groups are different), each vertex has several normals: one for each connected vertex, and aligned with their normals. This leaves a gap in the normals, which looks like 2 different surfaces. This is what we call a split vertex normal.



As you can probably guess, controlling the vertex normals is vital if we want to control our normal maps. Fortunately, we don't really need to directly modify the normals or even see them, but knowing how this works will help us understand why we do things the way we do and know more about the problems we may see.

When baking a normal map, we are basically telling the baking program to modify the direction that the low poly normals follow so that they match the direction of the high-poly model; as a result, the low-poly model is bouncing light as the high-poly would. All this information is stored in a texture called normal map. Let's see an example.

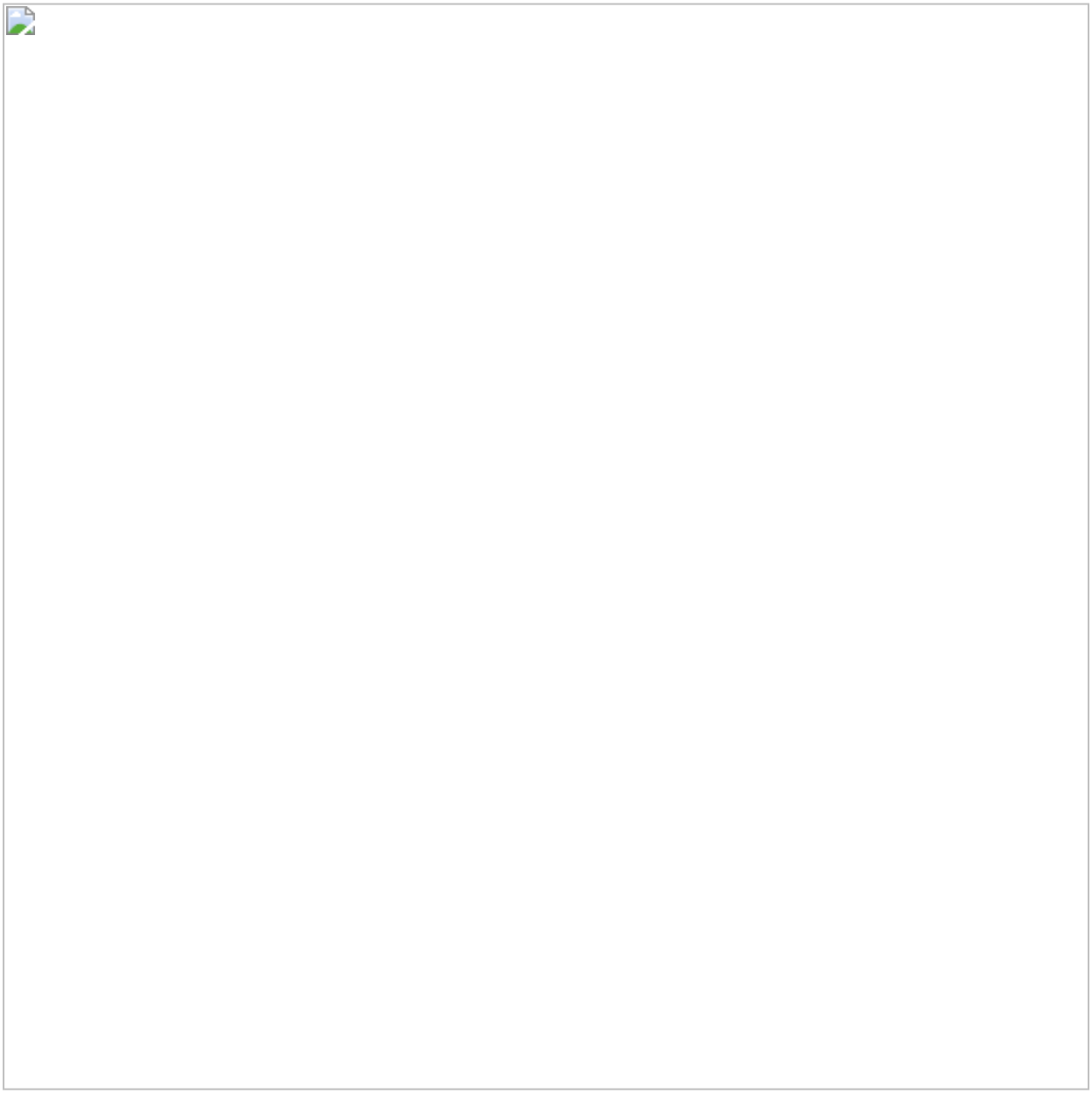
Let's say that we have a low-poly model like this one - a flat plane with 4 vertices and a UV set that our baking program will use to create the normal map.

And it has to receive the normal information from this high-poly model the normals of which are more complicated.

Keep in mind that we are only transferring normal information, so the UVs, material, topology, transformations, etc. are completely irrelevant. Rule of thumb: if your high-poly looks good, it means that its normals are good and should be fine enough for baking.

Our baking program will take the low-poly and cast rays following its normal directions (this is why we need to control the low-poly normals). Those rays have a limited length, to avoid getting normal information from far away faces (usually named bake distance or cage distance). When those rays collide with the high-poly, the baker calculates how to bend those rays so that they follow the same normal direction as the high-poly, and stores that information into a normal map.

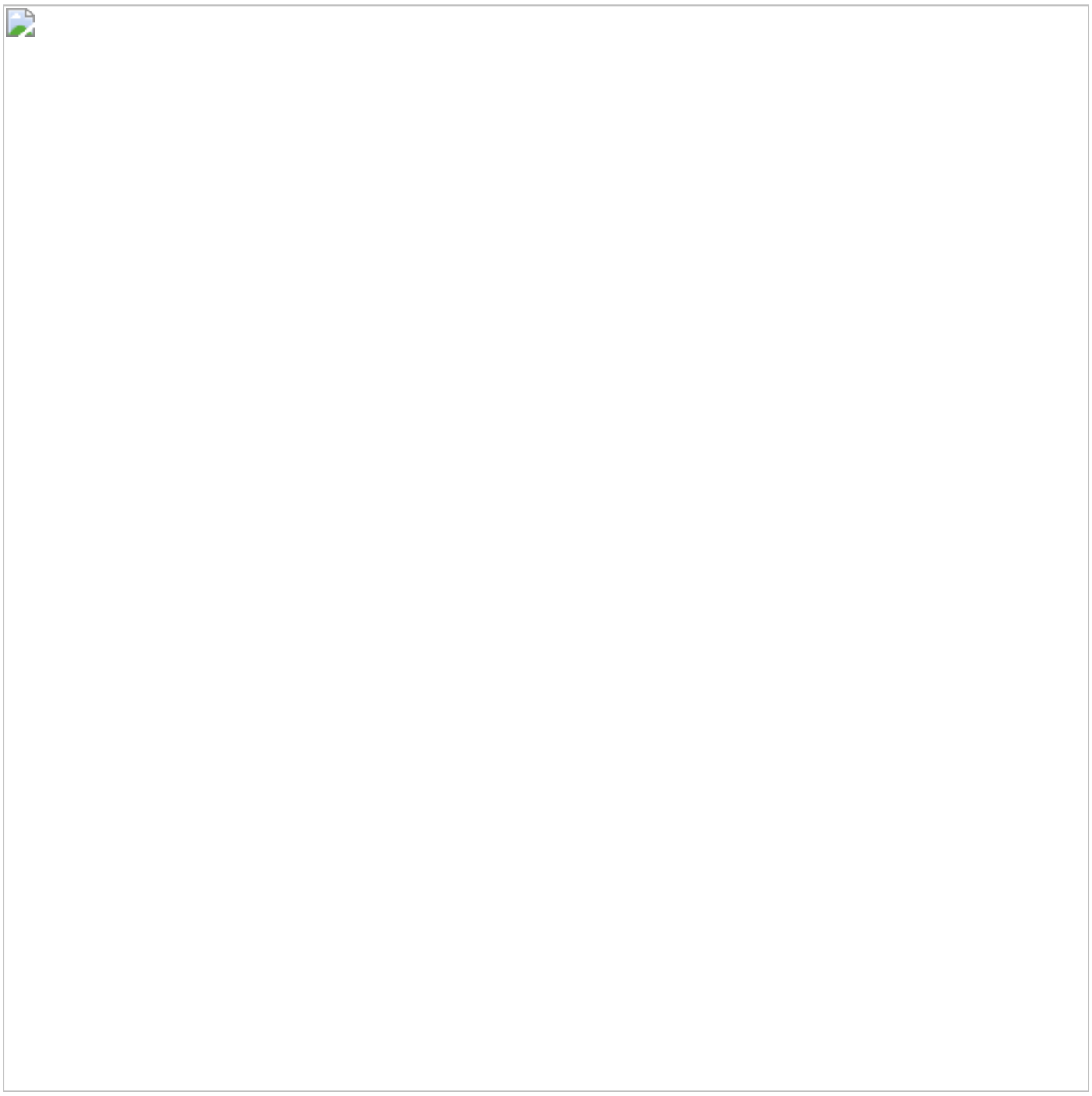
Here's the bake result of this example:



We have a texture that our engine uses to modify the low-poly normals, so light bounces from this low-poly model the way it would if we had the high-poly version. Keep in mind that this is a texture, and can't affect the silhouette of your low-poly (you can't modify how the light bounces from your model if it doesn't hit your model).

It's obvious that normal maps are not regular textures. This is because they don't carry color information, but normal information. This also means that normal maps should not be treated as regular textures and they have special compression and gamma correction settings, as we will see.

You can think of a normal map as a set of 3 greyscale textures, stored in a single image:



- The first image is telling the engine how this model should bounce light when lit from the right side, and it's stored in the red channel of the normal map texture.

- The second image is telling the engine how the model should bounce light when lit from below*, and it's stored in the green channel of the normal map texture.

*Some programs use above instead of below, so we can have "left-handed" and "right-handed" normal maps, and this can cause some problems as we will see later.

- The third image is telling the engine how the model should bounce light when lit from the front, and it's stored in the blue channel of the normal map texture. Since most things look white when lit from the front, normal maps usually look blueish.

When we combine all three images in a single one, we have a normal map. Please keep in mind that this explanation is not 100% correct, but it will hopefully help you understand the information inside a normal map and have a better understanding of what it does.

So, in conclusion:

Normals are vectors that we use to define how light bounces from a surface. They can be used to control the transition between faces (by averaging the normals of connected vertices to make a smooth transition or splitting them to make a hard transition), but they can also be reoriented, to make a low-poly model bounce light the way a much more complex model would.

This information is stored in 3 separate channels of an image, and the 3D program reads it in order to understand which direction the surface of the model should look towards.

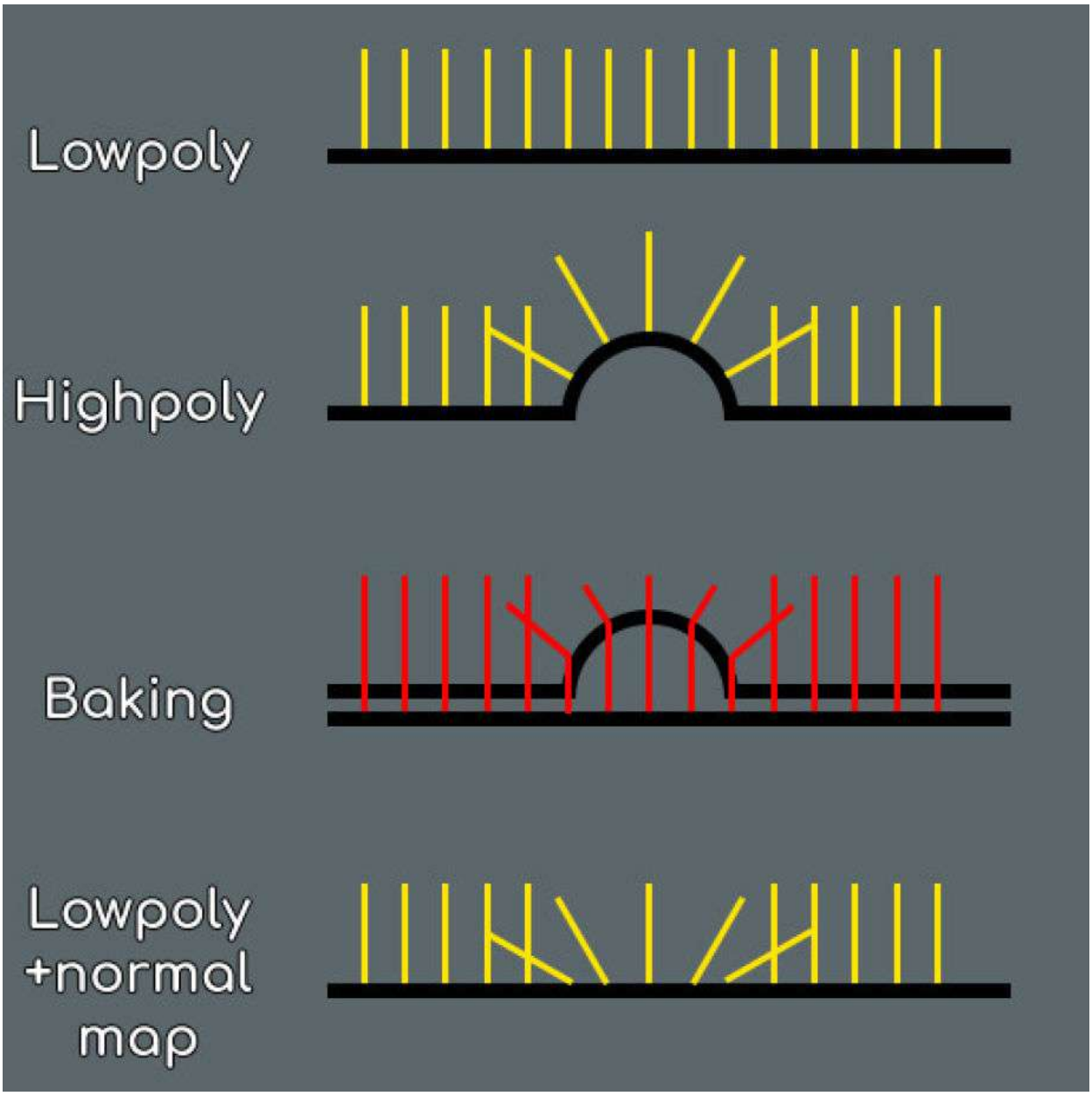
Now that we have a general idea of what normals are, and how a normal map works, let's talk about how we can bake these details from high-poly to low-poly.

Part 2. Baking Normal Maps

The general idea of baking a normal map is relatively simple: you have a low-poly with UVs and a high-poly model, and you transfer the normal information from the high-poly to the low-poly. This way, the low-poly will bounce light as the high-poly would.

During this process, the baking program will basically cast rays from the low-poly, following the vertex normals and searching for the high-poly. This is the most important aspect of normal mapping, and most problems people have when working with normal maps are related to this.

If you don´t control the vertex normals of your low-poly model, you will lose control over your normal map.



Bad Normal Map Correlation

In order to control the smoothing of our low-poly model, we can have split vertex normals (to create hard edges) or averaged vertex normals (to create soft edges).

Turns out that not all 3D programs use the same calculations to average the vertex normals. This means that your low-poly will look different and have its vertex normals pointing at slightly different locations depending on your 3D program. This isn't a big problem usually since these deviations are very small, but it can affect how your model looks like, and these differences are exaggerated when using normal maps since your normal maps are modifying the low-poly normals that are changing between applications.

The 3D industry is working on fixing this problem, and one solution has recently appeared, called Mikk space. This is a method of calculating vertex normals that all 3D apps could use, so vertex normal don't change between 3D programs. Keep in mind that not all 3D apps use it yet.

Another way to reduce this effect is not to rely too much on normal maps when baking. Try to match your low-poly more closely to the high-poly and use more hard edges on flat surfaces. This way, your normal map won't have to do all the work and these small deviations will be less noticeable.

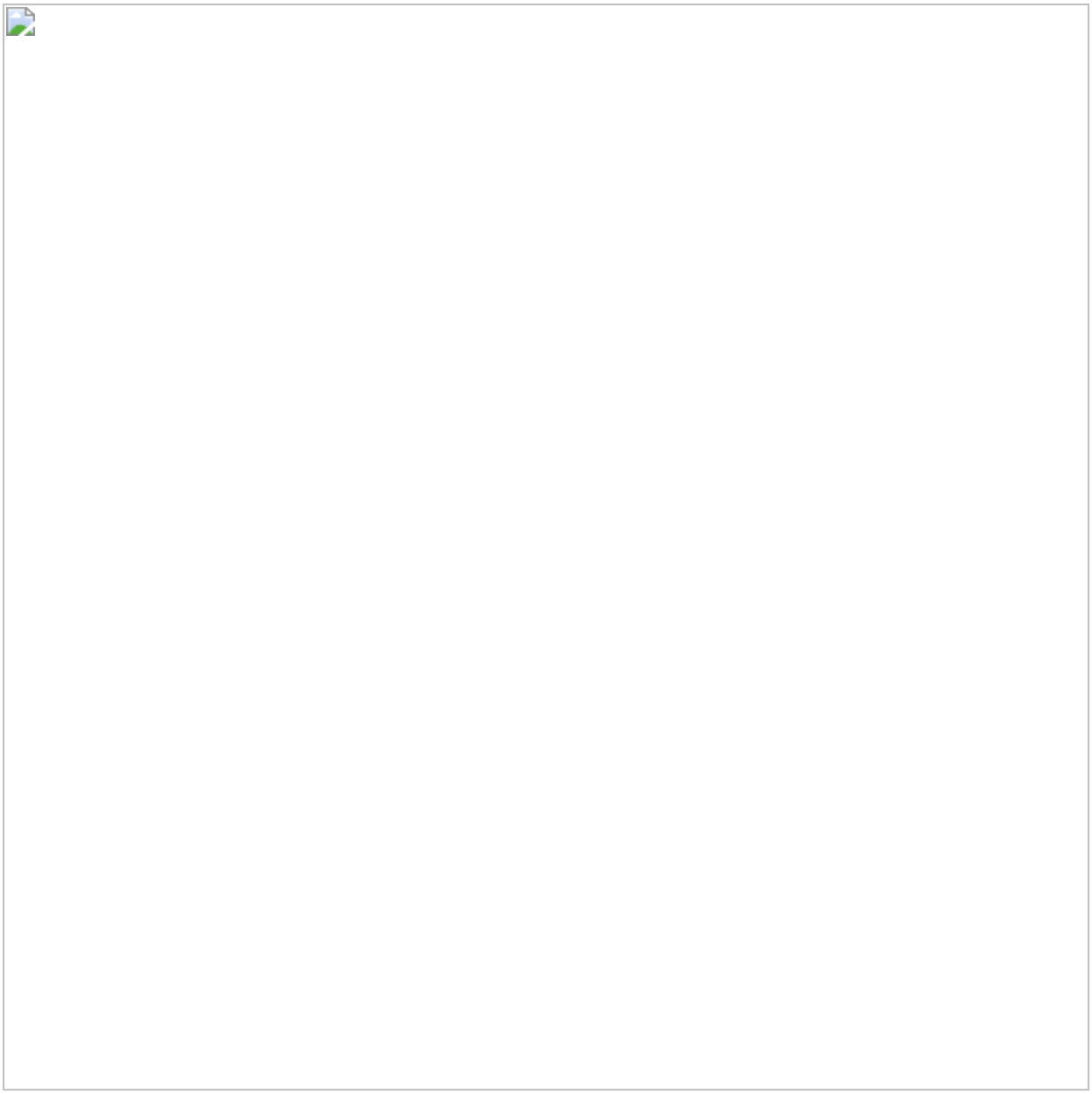
Normal Map Detail Skewing

When the computer averages the normal direction of your low-poly vertex normals, big changes in the angles of your surface can "skew" the low-poly normals and they won't be perpendicular to the low-poly surface.

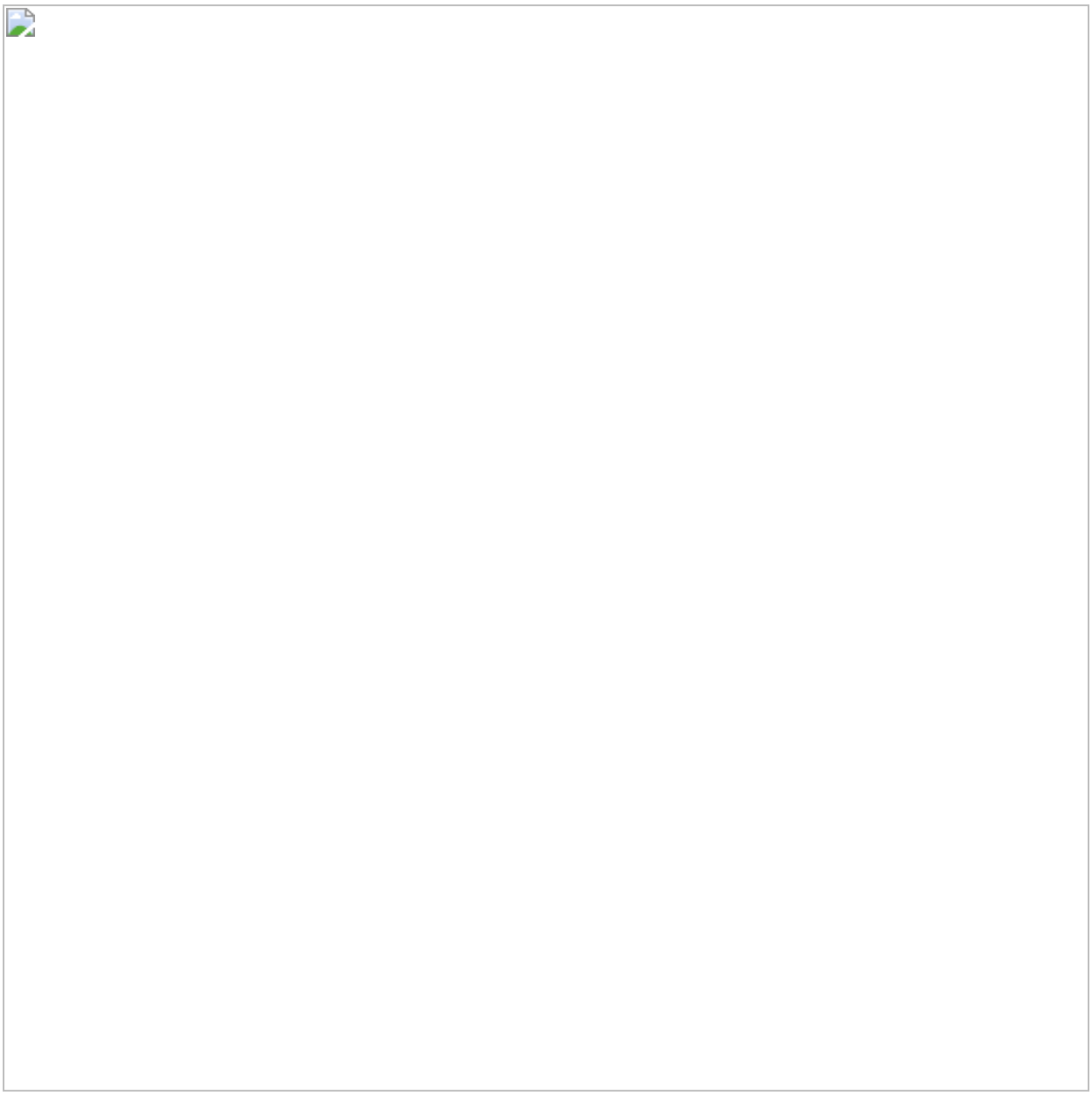
Since the normal map baker uses the low-poly normal directions when searching for the high-poly details, if these directions are skewed they will appear skewed on the normal map:

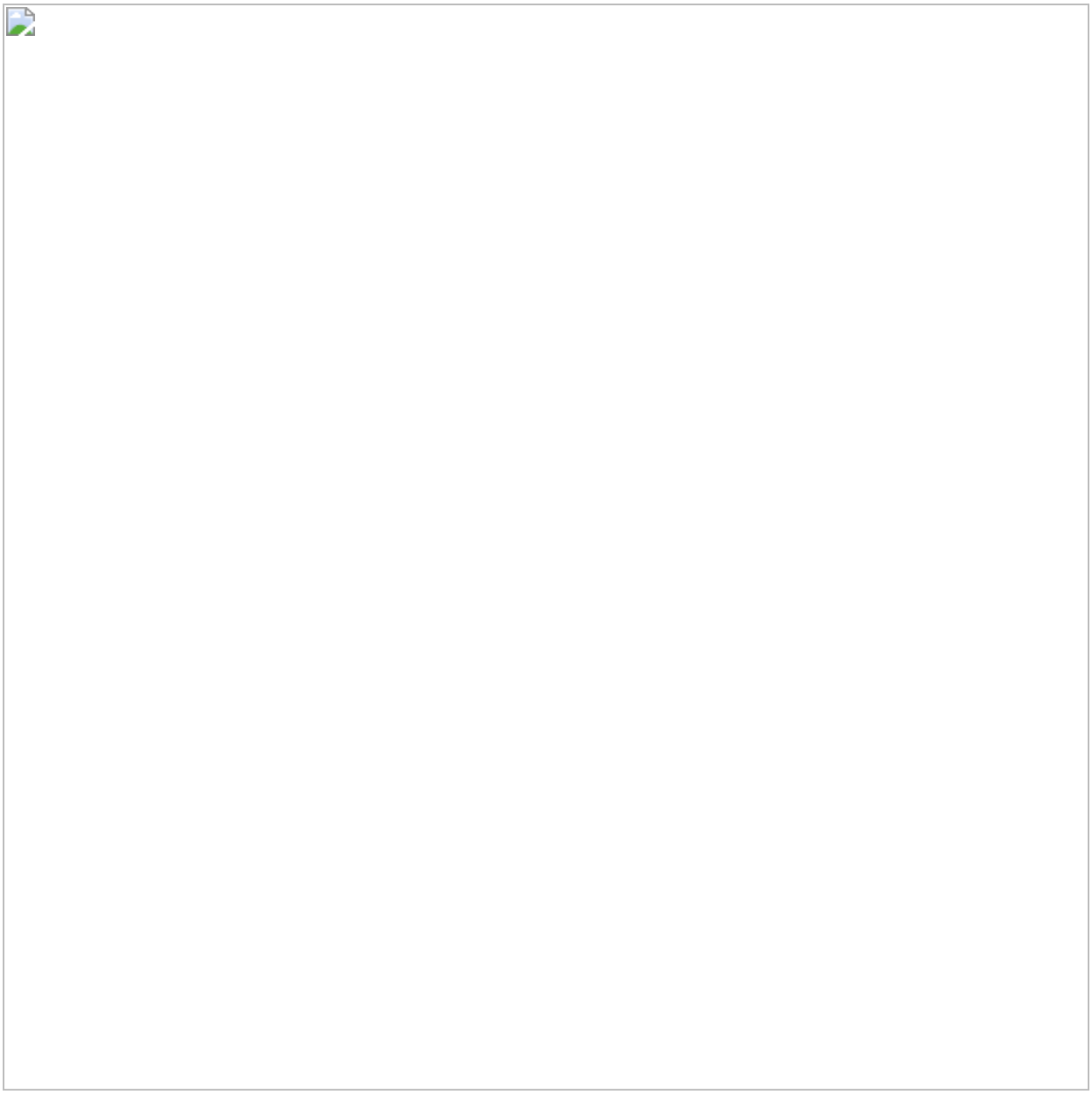
This is a very common problem, and several solutions have been found. There isn't the best one though, it really depends on the geometry.

- Some 3D bakers have the option of rebaking these parts modifying the low-poly normals temporarily, so they are baked without skewing. Marmoset Toolbag has this option. Reddit user Tanagashi kindly explained to me that some programs such as xNormal can tesselate the low-poly to add new vertices and make the normals perpendicular to the low-poly surface, bake an object space normal map and then convert it to tangent space using the original low-poly normals. Using this new normal map, the program can create masks to control where to use the original normal map and the one created from the tesselated low-poly.

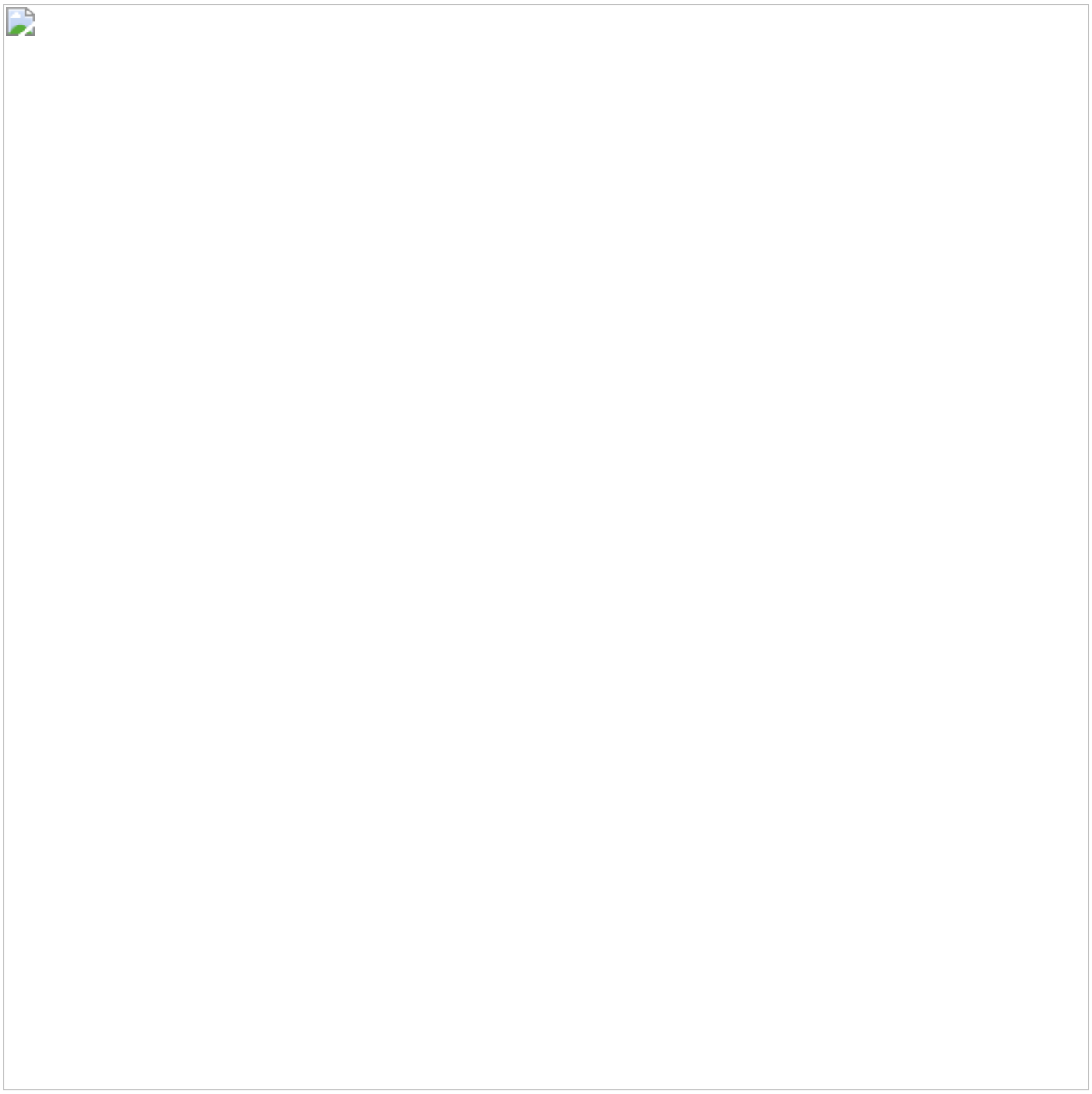


- Adding vertices will make the transition between the vertex normals less skewed, as one 90º angle can be split into smaller degrees, making the second transition less skewed. This obviously increases your polycount and, since you are adding geometry. I recommend you to use this extra geometry to add a more interesting silhouette to your model.



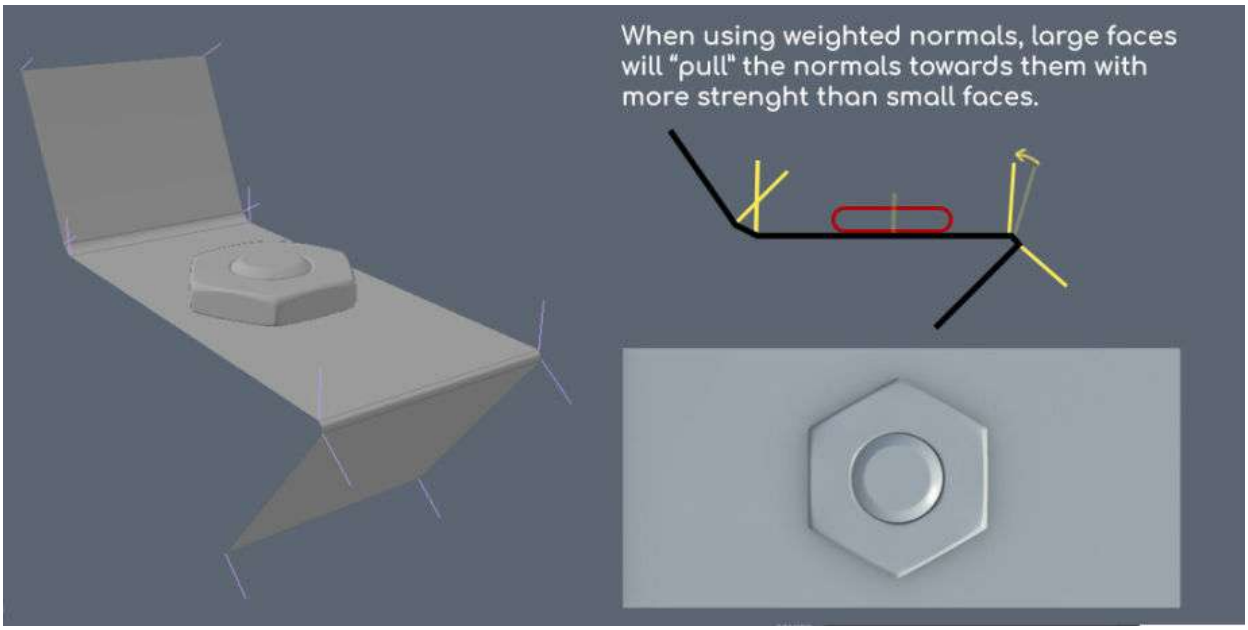


- Split the averaged vertex normals (making the edge hard/use separate smoothing groups): this way, each vertex will have several normals, each one perpendicular to the low-poly surface. Keep in mind that, when the 3D program has a split vertex normal it actually creates a duplicate of the vertex, so this will increase your vertex count and slightly decrease performance. Additionally, hard edges will also give you a "black edge" problem, as we will see later.

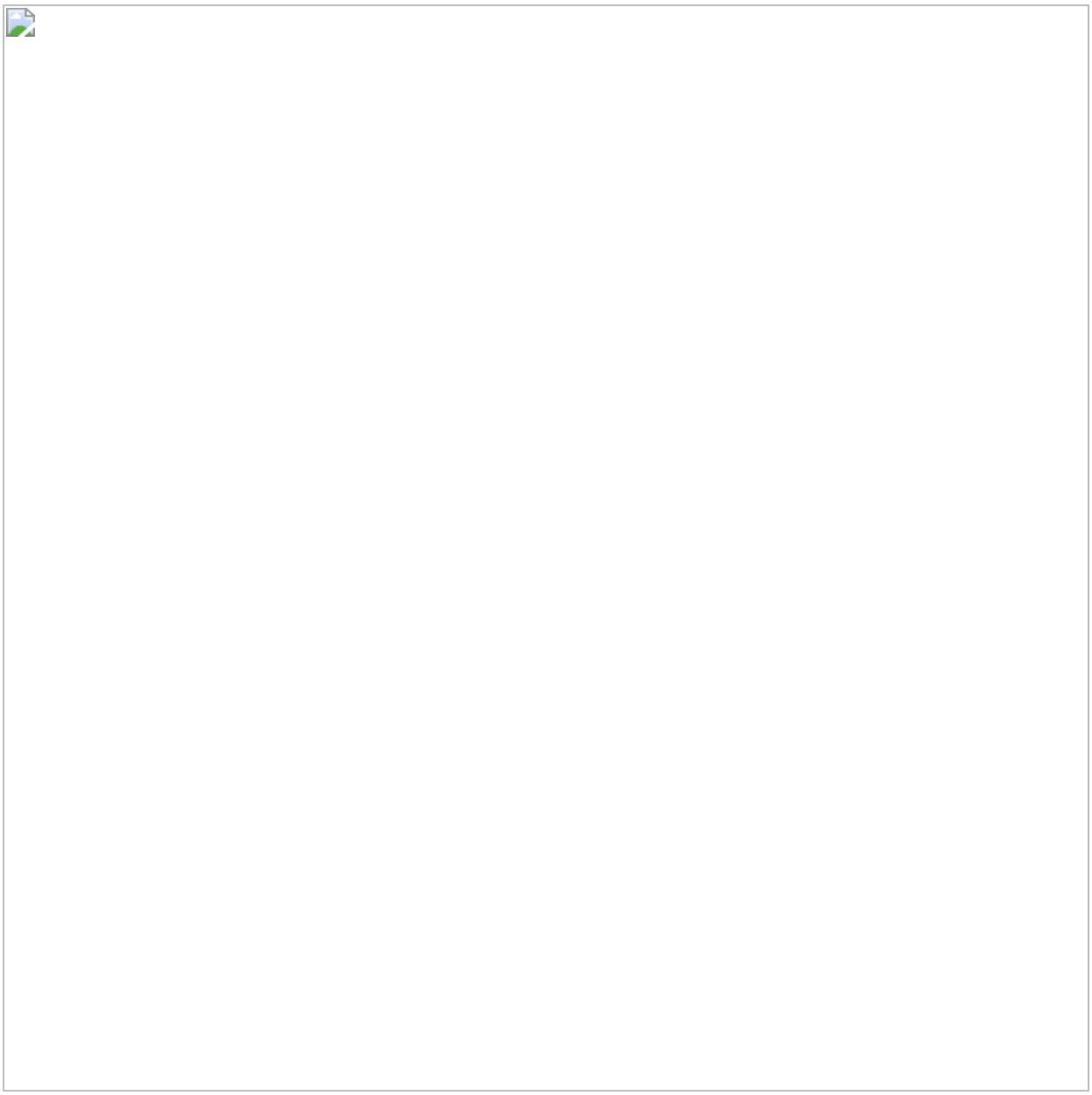


- We can modify the normals of our models to bend the normals of our low-poly so that they are perpendicular to the high-poly details. Keep in mind that not all programs allow modified normals (ZBrush only has averaged normals, OBJ and older FBX files don't have custom normal information). There are basically 2 ways of modifying the normals:

1. Weighted normals: this is an automatic method similar to the average vertex normals. The idea here is that when averaging the vertex normal not all faces will have the same strength: larger faces will "pull" the vertex normals towards them with more strength than smaller faces. This way, larger faces which are usually more important will have better detail projection. This works especially well with high-poly panels.

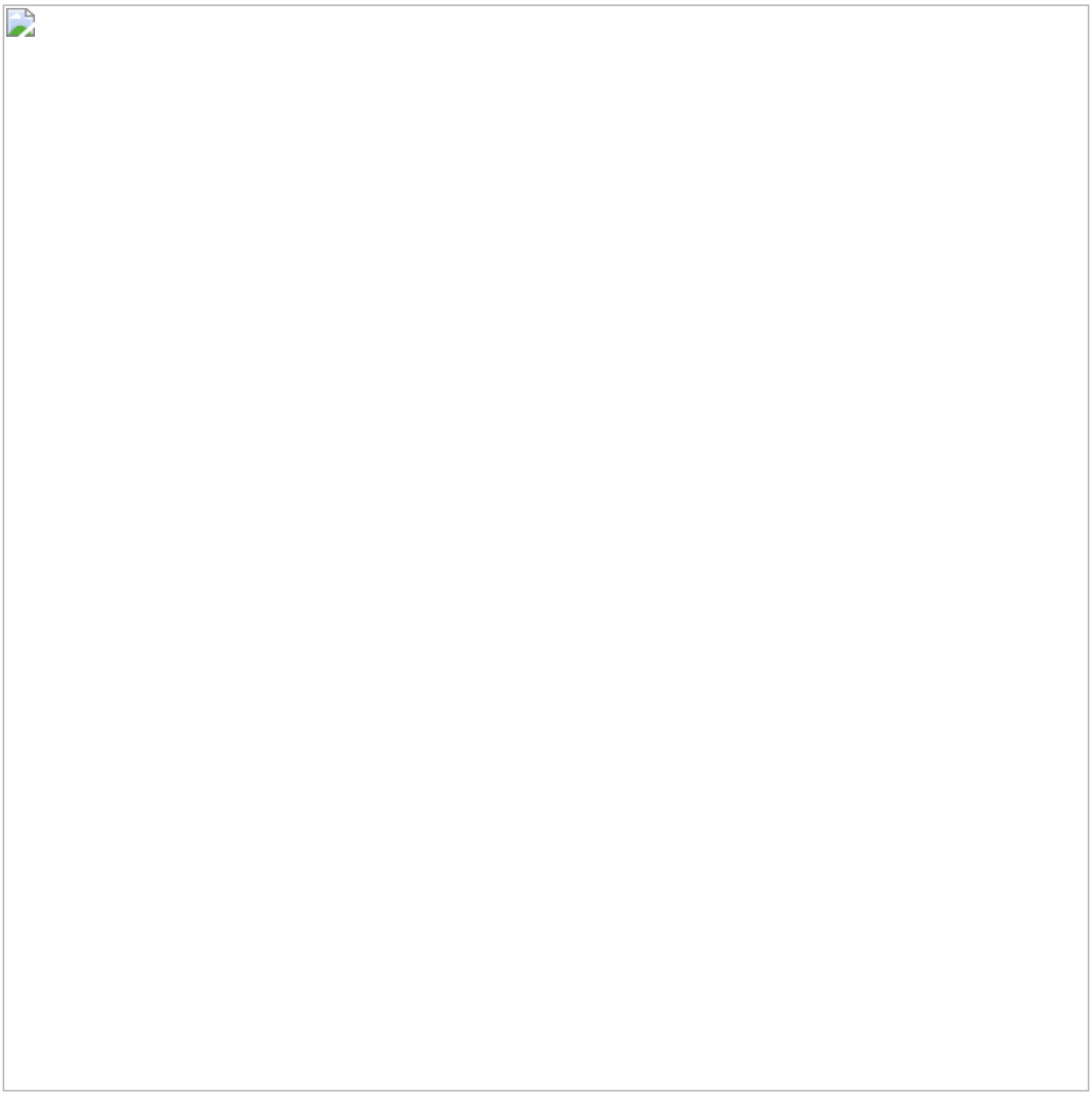


2. Custom normals: using tools of your 3D software, you can bend your low-poly normals. This is a relatively new idea and there aren't standardized tools for this. Keep in mind that bending the normals can create very weird unintended shading on other parts of your model, so this technique is usually combined with bevels. Some people call this technique "mid-poly modeling".

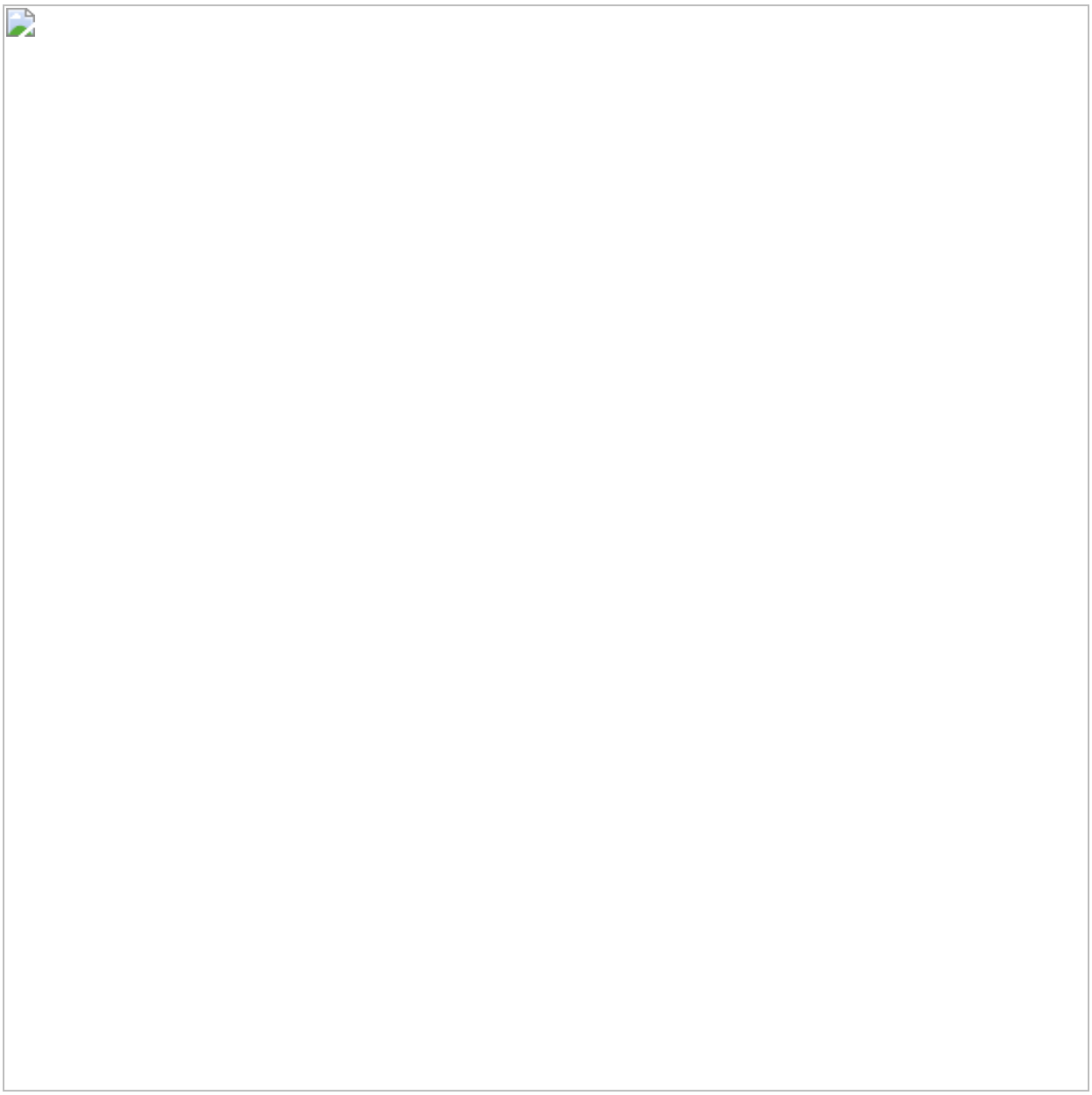


Cage/Baking Distance

By default, the rays that are cast from the low-poly surface travel a limited distance, to prevent the low-poly from receiving normal information from far away parts of the high-poly. This distance is usually called "frontal/rear distance", as the rays can be cast towards the inside, outside of the model, or both. You can see this distance represented in red in the following image:



Some 3D apps (3ds Max, for instance) also allow us to use a cage. A cage is a "copy" of our low-poly model that we can modify so that it encapsulates the high-poly perfectly. And, in some cases (not all), it also allows us to change the direction of the rays, without changing your original low-poly vertex normals. This can help us get the best baking extremes and avoid skews, but keep in mind that though you are not baking using the normal direction of your vertex normals, in the end, you will use a normal map to modify the actual low-poly normals, so the result could look strange.

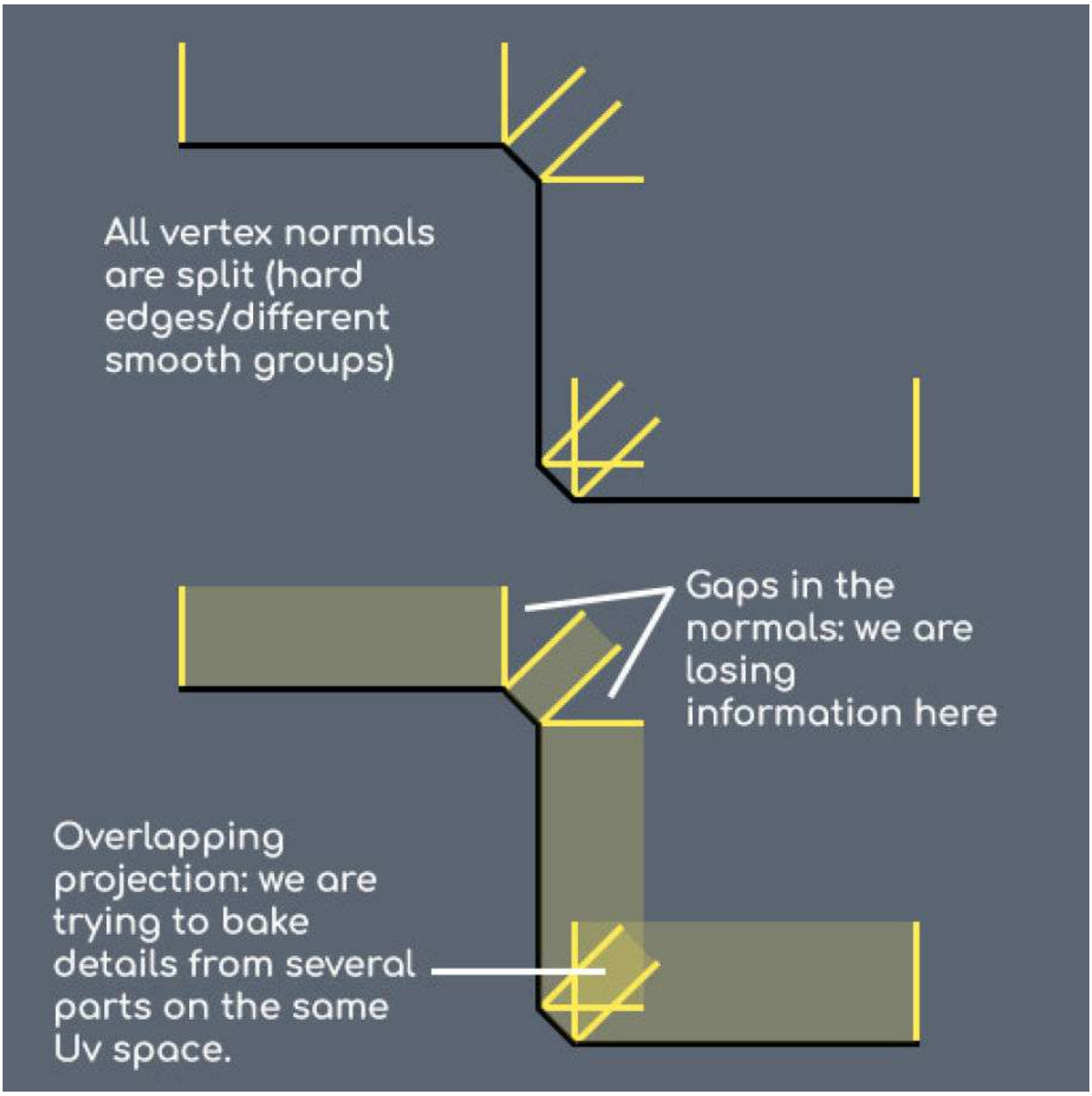


Edge Seams

As we have seen, if you have a model with hard edges (some faces have more than 1 smoothing group, or some edges are set to "hard"), the baking program will split the normal of the vertices between 2 faces. This has good and bad effects.

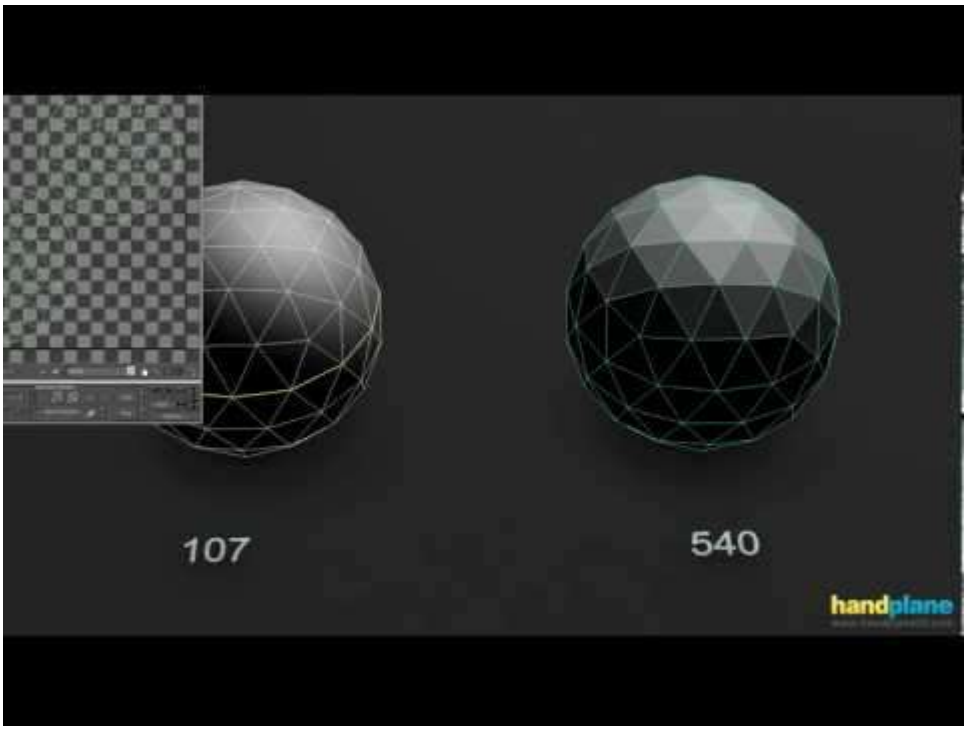
The good effect is that the normals are not averaged, so there are less normal map distortions: the vertex normals are completely perpendicular to the surface of the low-poly. It can also help make your low-poly look better if it has faces on extreme angles, more appropriate for hard edges.

The bad effect is that there is now a gap between the normals, and this could mean that you lose information if your low-poly has a gap in the normals where it can't get high-poly details. Furthermore, some parts of your low-poly projection could be intersecting and will compete for the same UV space. Both effects leave a seam along the edge, that's more or less noticeable depending on your engine.

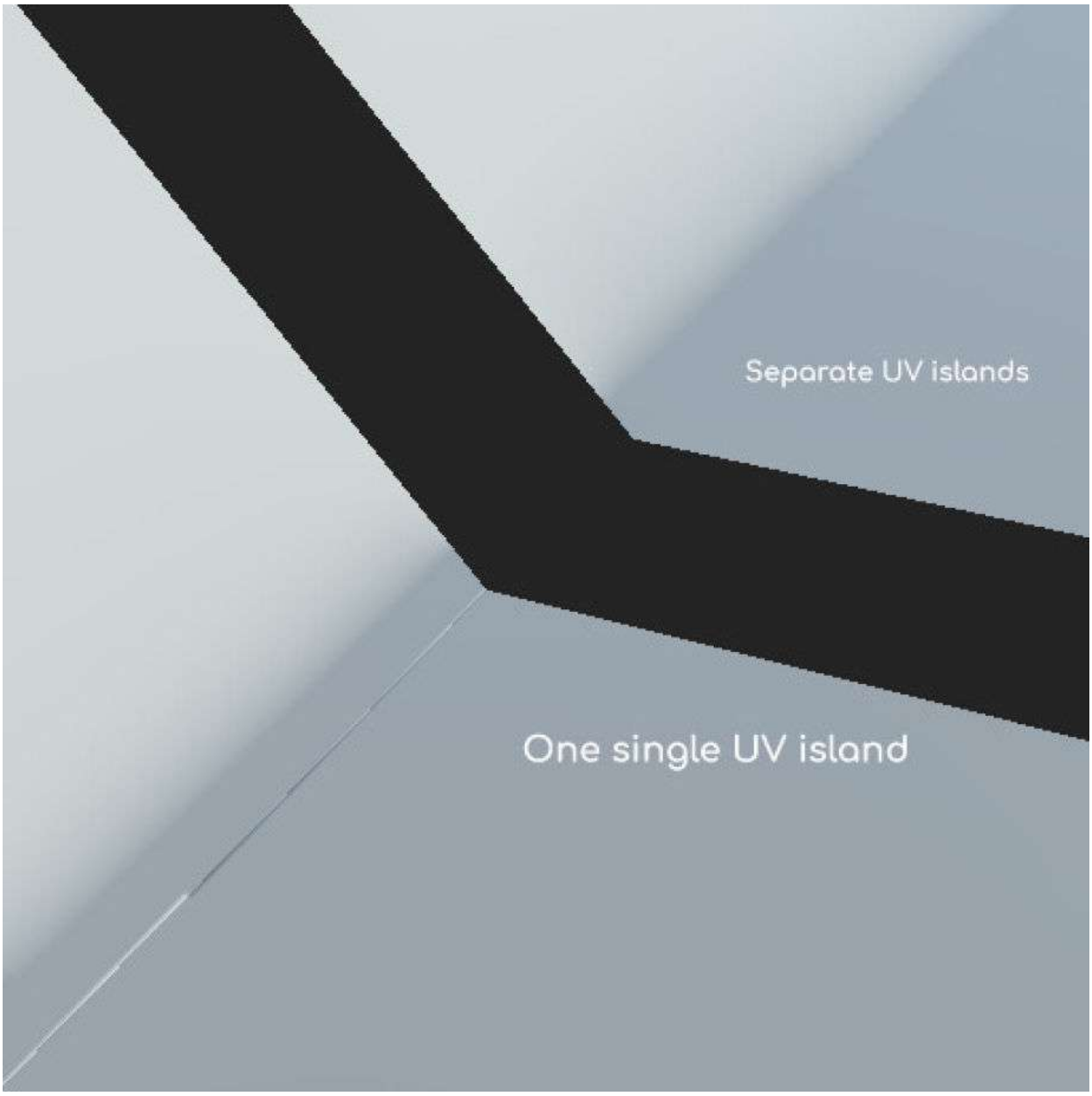


This, however, can be easily avoided using a simple trick: if you have two faces separated by a hard edge, put each face on a different UV island, with some space between them.

If both faces are connected in the UVs, there are drastic changes in color from one face to the other, and color can bleed (because of the processes that occur when rendering faces) which is extremely noticeable in the normal map. By separating the faces in the UVs, the baker can add padding between your faces and avoid this color bleeding. The video below might help you understand this process:



Watch Video At: https://youtu.be/ciXTyOOnBZQ



In conclusion:

Once I have my low-poly model ready and adjusted to the high-poly model as close as possible, I start working on the smoothing before UVs.

I set my smoothing for the low-poly (if it's organic, I start with a completely smooth model; if it's hard-surface I start with an angle smooth set to 30-60º; and tweak the model smoothing until it looks good).

Once I have the smoothing for the model set, I work on the UVs, making sure that all hard edges are split into separate UV islands (to avoid edge seams).

If I have skewing errors, I add additional edges (usually bevels, to keep a more rounded silhouette). This works for most of my models, but I could also fix the skewing errors in Marmoset Toolbag if I used it for baking, or by using custom/weighted normals.

If there are projection errors, I modify the baking distance/cage, modify the low-poly/high-poly so that they are better fit for baking, or erase the normal maps on certain, really hard parts such as the tip of a cone.

In the next part, I'll be making a troubleshooting guide for normal map baking and discuss some of the most common problems and solutions. If you are enjoying this tutorial so far, please comment here or below the posts on ArtStation (Part 1 & Part 2) - I'm looking forward to some feedback, even if it's negative. I'm doing this so that I could learn and improve, but a complete silence can be discouraging. Thank you for your time!

Carlos Lemos, 3D Artist

Keep reading

You may find this article interesting

> Tips On Normal Map Baking For Hair Cards

Join discussion

Comments 5

- very clear and useful. thanks!

  0

  Anonymous user

  ·a year ago·
- This has been quite insightful. Thank you so much ^^

  0

  Anonymous user

  ·a year ago·
- nice~

  0

  Anonymous user

  ·a year ago·

- Really good explanation ma men! And really appreciate all the time and effort you put into this. It helped me to understand why I get some artifacts sometimes. I know why now. So thank you very much!

  0

  Fonseca Bruno

  ·2 years ago·

- Hi! it was a really nice explanation.

  It would be cool to see examples applied in real cases, like baking a character for games.

  0

  Anonymous user

  ·3 years ago·

You might also like

HOW TO PAINT INTUITIVELY

PART 1

ALL YOU NEED TO KNOW

WELCOME

Welcome to my LAB tutorial! I studied and practiced a lot before writing it, I planned to make it as intuitive and didactic as possible. With this tutorial you will be able to optimize your process, in addition choosing the colors correctly to use in your paintings, making it more believable and harmonious.

WHY SHOULD YOU USE LAB?

It is easy to paint **keeping the volume** of the shapes easily changing the colors!

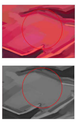
It is great to use while painting as it is possible to change the colors without major changes in tonal values, allowing **color variations(broken colors)** without lose readability. In addition to being intuitive in terms of lighting between cold and warm tones.

You move the slider towards the tone and add the tint, automatically the correct transition happens!

LAB

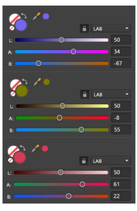
I didn't change slider L and randomly changed slider A and B (slider of colors). In 90% of cases the values will be very similar.

Example of broken colors in my painting:



Note

To check the values, I use a black painted layer with blendmode color above all others.



Advantages

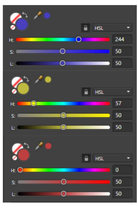
- Intuitive color change when, especially when there is already a base color.
- Minimum value change when changing colors.
- Easy use of broken colors to enrich the painting.

Disadvantages

- It is not possible to measure the saturation of the mathematically equal color in the compared models.

HUE CUBE, HSL, WHEEL

In the HSL/Cube/Wheel model, changing only the slider H(HUE) does not mean that the tonal value of the color will remain the same, as shown in the example of the squares.



Advantages

- Easy to measure the saturation of a color mathematically.
- Easy to choose a base color.

Disadvantages

- In most cases it is difficult to choose the color resulting from the lighted / shaded and cool / warm area, needing to check the painting in black and white more often.

LAB EXPLAINED

DARK — LIGHT

L bar - Change the color value / exposure

+ GREEN + SATURATED — +GRAY - SATURATED — + MAGENTA + SATURATED

A bar - Tint

Adds green or magenta ink.

To add saturation it is to move to the ends of sliders A and/or B. And to remove saturation it is to move them to the middle.

+ BLUE + SATURATED — +GRAY - SATURATED — + YELLOW + SATURATED

B bar - Temperature

When we want to add blue (cool) / yellow (warm) in a tone, just move the bar B, which makes it much easier during painting.

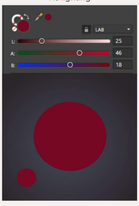
We can see the representation of bars A and B in this graph. Basically we use **coordinates** to get the colors we want. The extremity of the graph has more saturated colors while in the center is desaturated.

So the bar B is used for temperature, as it takes the tone to the part that contains **cold** or **warm** colors.

LET'S SEE EXAMPLES

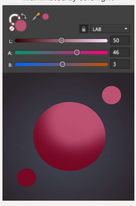
Pay attention to how I changed only the **slider L to illuminate** and the **slider B to change the temperature** of the area that receives warm or cold light.

No lighting



Illuminated by warm light

Illuminated by cold light



We can see that only by temperature it is possible to arrive at an interesting result 90% realist!

In the PART 2, we'll accurately see what real-world lighting looks like for any type of lighting different from blue and yellow and how to make lighting through saturation even more interesting.

End part 1

HOW TO PAINT INTUITIVELY

PART 2

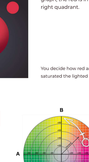
ALL YOU NEED TO KNOW

LAB ADVANCED + COLOR THEORY

When using different warms and cold lights mood, we need to combine slider A and B to achieve the desired shade or light tone.

We will use a red ball, illuminated by several lights for you to see how it works.

1- Illuminated by the same hue



+ saturation in relation to base color.

= hue in relation to base color.

Step by step example:

1 - Base tone.

2 - I increased the **slider L** to make it clearer.

3 - To increase saturation in this case, I will need to bring **slider A** and **B** closer to the extremity simultaneously, keeping the same color and increasing the saturation. As shown in the graph, the red is in the upper right quadrant.

You decide how red and saturated the lighted area will be.

2- Illuminated by analogous color (next)

+ saturation in relation to the base color.

Hue 30% towards the light tone.

The resulting color of the illuminated area is approximately the dash drawn, depends on how strong the light is.

Step by step example:

1 - Base tone.

2 - I increased the **slider L** to make it clearer.

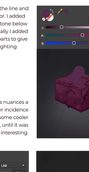
3 - Only **slider B** will give a very nice result, but in the next step, the **slider A** will help with a small change in hue and increased saturation.

4 - I moved **slider B** more than **slider A**, because the ball receives the yellow light.

Note

That increasing saturation and moving the hue towards the light tone are done simultaneously in this case.

3- Illuminated by complementary color (further away)

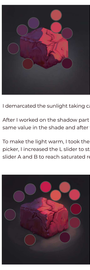


- saturation in relation to base color.

Hue 30% towards the light tone.

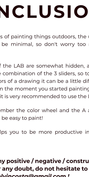
Step by step example:

1 - Base tone.



2 - I increased the **slider L** to make it clearer.

3 - To decrease saturation and go towards the desired tone, you need to move **slider A and B** towards the center, simultaneously going towards the color of the light. As we see in the graph.



I drew on top of these graphics, to show how it would be to light another color with different colors

LET'S PAINT WITH LAB

I made a red brick with a strip of sunlight, below of blue sky very clean. I decided to exaggerate the colors a little using broken colors, to generate more interest and color variation. Note that the brick has cool tones from the sky, and warm tones from the sun.

Before starting to draw and paint, do a brief reference search to find out how the material behaves in terms of texture, cracking, tones in the shade and in the sun.

I made the line to help me know the reliefs and the darkest parts.

I reduced the opacity of the line and left it a more reddish color. I added a slightly cool dark base tone below the line layer to start. Finally I added occlusion in the darker parts to give a volume with diffused lighting from the sky.

I modelled the block's nuances a little more, with a greater incidence of sky light adding some cooler tones through **slider B**, until it was minimally interesting.

I demarcated the sunlight taking care not to cover the slits.

After I worked on the shadow part more. I used different colors but the same value in the shade and after the same process in the sunlight.

To make the light warm, I took the first cold shade used with the color picker, I increased the L slider to stay bright and I greatly increased the slider A and B to reach saturated red.

When we put it in black and white, we see that the modeling and legibility of the parts are working, but it can get better.

Finally, the part that led to more time. I added more brush strokes, more contrast at the edges. I left the spotlight on the top of the block.

CONCLUSION

In most cases of painting things outdoors, the use of slider A will be minimal, so don't worry too much about it.

The colors of the LAB are somewhat hidden, as you will need the combination of the 3 sliders, so to start the base colors of a drawing it can be a little difficult, however from the moment you started painting light and shadow it is very recommended to use the LAB.

Always remember the color wheel and the A and B axes that will be easy to paint!

I hope it helps you to be more productive in your processes.

If there is any positive / negative / constructive feedback or any doubt, do not hesitate to send it to me okelvincosta@gmail.com !

Thanks for reading and for the support!