Materials for games - An overview on creating materials for games

1st Davi Amancio Souza
Universidade Federal De Minas Gerais
Belo Horizonte, Brazil
daviuai25@gmail.com

2nd Rosilane Ribeiro Mota
Universidade Federal De Minas Gerais
Belo Horizonte, Brazil
rosilanerm@gmail.com

***Abstract*—This paper presents an overview on some game texturing concepts and on how creating materials and its textures might be approached, by different means and for different purposes. Even though creating most materials can be done using more commonly talked about workflows in tutorials, specially useful for beginners, this approach is limiting on what can be achieved both artistically and technically. This paper intends on diffusing the existence of these concepts and how they might be kept in mind when creating game assets. The focus will be on high resolution, realistic materials. However, most of the presented points also apply for more stylized work just requiring some adaptation.**

***Index Terms*—texture mapping, UV tiling, game texture, texture repetition, shaders, material creation**

I. Introduction

The balance between a game's materials resolution, performance and storage needs difficult to achieve. Not only that, but there are also requirements in regards to workflow and production that must be taken into account. There are numerous methods of texturing, shading, optimizing the models and their UVs, variations for static assets, deformable assets, characters, environment and other concepts to keep in mind [1]. When faced with so many assets with many different requisites it becomes easy to be overwhelmed, or at least limited in options when the artist is unsure of what tools and workflows can be used.

Knowing what these tools and workflows are and what they can be used for greatly empower artists and developers in creating the assets or to start the research of other similar techniques in order to develop their own. Its not uncommon to see games or portfolio pieces that require a greater resolution texture than what is actually used, for example, an asset might have too obvious texture resolution discrepancies (Fig. 1). It's likely that simply using the needed texture resolution or shader without going for a different workflow might cause the game to have poor performance [2] [3] or have unreasonable storage requirements [4]. These problems can usually be avoided with smart use of a different workflow or art direction, allowing for better use of the requirements and limitations imposed. This paper takes on an analytical stand towards technical and artistic aspects of material creation for games, using a structure akin to the PBR guide's by allegorithmic [5] and Labschutz's work [1], albeit with a more generalized view.

Figure 1. Texture resolution discrepancy can be seen in the character's face and clothes in Fallout 4

II. Methodology

This paper's technical view of the subject is through analyzing how an engine can use textures and shaders in different ways for different purposes and how an artist might approach that. While also talking about some of the artistic aspects, by taking into consideration what kind of techniques would work for one or other asset's needs, according to its nature, such as organic or not and the game's style. First and foremost there will be a brief overview on understanding a subject material(s) requirements on a per case basis, which should be the starting point on creating any material, based on Beddows' explanation of material layers [6]. Then the introduction of some basic principles of 3D texturing and some techniques to approach them. For example, in the section about UV mapping IV-A, besides introducing the notion behind UV mapping different workflows and use cases will be presented, to make it so the artist has a reference on tools to mix and match in practical work. The topics that will be presented are in order

- UV Mapping
- Material layers and slots
- Texture tiling
- Texture picking, projecting, painting and authoring

- Shading
- Material assembly

This paper assumes the use of the PBR rendering workflow, which is commonly used by modern game engines and off-line render engines alike. The PBR, or physically based rendering [5] workflow, takes into consideration real-life properties of surfaces in order to calculate the final output of each pixel on the screen, which makes for a more accurate and easier to deal with workflow in general. Most concepts can be applied to other workflows, such as Specular/Glossy, however.

Practical examples will be presented using softwares such as Allegorithmic's Substance Designer and Substance Painter [7] [8] for material authoring and painting, Quixel's Megascan database [9] for scanned PBR textures and Epic's Unreal Engine 5 [10] as the graphic motor and game engine. They are not all for the same assets or even in the same style, even though there is a main test object, which is a scabbard model from the Megascan database [9], not all exemplified techniques would apply to it.

A. Terminology in material development

The terminology when working with material development, either for games or otherwise is quite vast and it's easy to be overwhelmed by. Besides, some terms seem interchangeable, and even might be in some cases. With that said, they will be clarified for the purpose of streamlining the understanding of this work.

The most basic layer of the material development that will be touched upon in this paper is the shader which, in short, refers to programming on how the pixels on a surface will be calculated [11] [12]. Most game engines and off-line renderers allow for some degree of customization of the shaders themselves. Either allowing the user to alter properties in the usage of textures and other parameters on top of the base shader calculations, as is the case with Unreal Engine [10], or changing how the shader works fundamentally [13]. The later is usually something only a technical artist would work with.

The term material is sometimes mistakenly used interchangeably with the shader, and is actually the final result of the shader when setup, with its parameters and textures. Two vastly different materials can in fact be made using the same shader when these are changed.

Textures are the color and grayscale images used for driving the shader's parameters [5], such as reflectivity, base color, metallicness and others. These can be image files or procedural in nature.

Post processing refers to an image processing effect applied after the rendering of the scene, and can be used to create many stylized looks. A common use case being cel shading [14] in order to achieve a cartoonish look. Post processing won't be discussed at length in this paper, but can be of great importance in the final look of a game and should, therefore, be kept in mind.

III. Identifying and planning materials and shaders

Even before the artist gets on to creating a material or shader for a game (Fig. 3), they need to keep in mind a number of factors, specially when working in a team. Technically speaking, for example, a game might be developed for a platform, such as a smartphone, which can be quite limiting [15], that in turn limits how complex your shader and how large your texture resolutions can be in order to work appropriately for the target systems. While artistically your game might be stylized in a way that will require an outline, which needs to be specified on a shader or post-processing level [14], or additional texture work for internal lines. It might also require specific UV layouts in order to properly use the texture setup. The stylistic and aesthetic choices also dictates how large your textures resolutions need to be in order to properly represent the artistic vision for the assets. Besides the development of the shader and the material themselves, having proper planning on how they are applied makes it possible for the artist to use them most efficiently. Larger surfaces usually require different mapping techniques than smaller ones. Some geometries might themselves benefit from texture repetition with proper modelling and UV mapping [16]. Besides, different types of assets might have different approaches to aspects such as Texel Density [17] for some more information on the subject refer to IV-A.

The method Beddow [6] uses when approaching the analysis of a material and its components is a great way to understand the needs of an asset and its materials as well. Even though in his example the subject is a single material with multiple layers, a similar approach can be used on a single asset with multiple materials, if so required. For example, in the scabbard model exported from the Megascan's[9] library (Fig.2), multiple materials can be perceived:

- a darker, surface-level leather with spotty details;
- a lighter, rougher leather that appears wherever the surface is scratched or branded;
- an even rougher and raw leather at the back;
- the threads going around the frame of the scabbard;
- the metal for the tacks holding the leather straps together.

Figure 2. Scanned scabbard model avaiable on the Megascan [9] library

As mentioned before, knowing the artistic and technical requirements for the project dictates how to approach these

different layers. On a technical level, the artist might treat all of it as an unique material with different parameters for each layer, that would be taken care of, on its majority, by hand painting textures in a software, such as Substance Painter [8], using a scanned material, which is available on this specific asset, or by blended materials on an engine level, method further discussed at Section IV-B. Each with its own advantages and disadvantages, as seen at [18]. On an artistic level, the artist would need to take into account the aesthetic of the game and specifications from the art direction. Including this identification and analysis step of creating a material, a possible workflow is exemplified in Fig. 3.

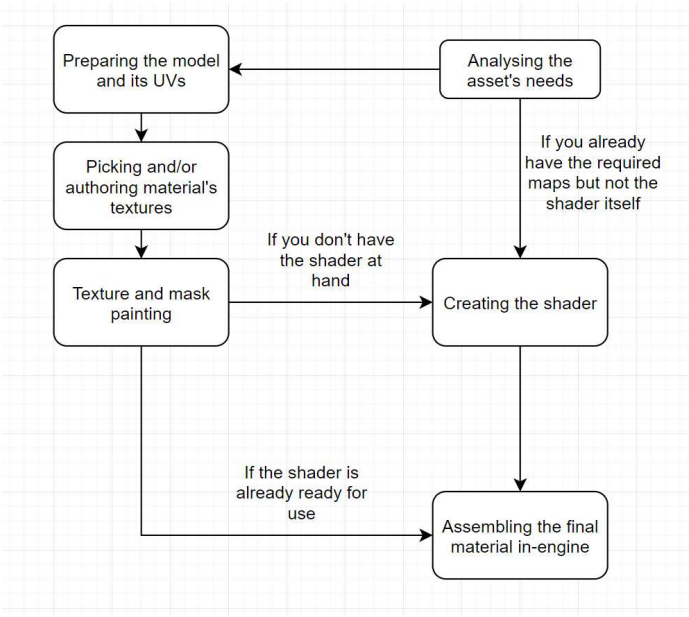

Figure 3. Steps on creating a material

IV. Material creation and technique variations

There are multiple ways in which a 3D artist can make a more efficient use of shaders, textures and UV mapping for each purpose. The decisions made in each step are consequences of the planning phase and overall direction of the game. A brick surface isn't created the same way a character's face is, for example.

A. UV Mapping

It's important to keep in mind some fundamentals on UV mapping when optimizing the UV usage on each asset. It's notable however that the model UV mapping is not always the texturing artist responsibility, but of the modelling artist. That being said, proper communication is needed since the way UVs are planned and laid out greatly affect how materials are created and used.

The first thing usually taken into consideration when mapping a 3D model is to make the most efficient use of UV space possible, so that the texture has the least amount of wasted pixels and the greatest resolution density, that is also known as "Texel Density", "Pixel Density" or just "Texture density" [17]. What is an efficient use of UV space changes according to each model and game needs, you might specifically want a part of your model to have lower density than the rest. A character's shoes don't often need as much resolution as its face, for example. Also, the same asset can have its UV seams created in ways that prioritizes either having less distortion, or fewer seams [19]. That decision is driven by the nature of the material, which should be considered on the analysis of material step. An inorganic pattern with straight lines quickly loses its appeal when the UV is distorted and has misplaced edge cuts, while an organic texture does not usually suffer nearly as much as from this phenomenon. UV mapping also depends on how the textures are created. If the surface uses either preexisting authored or scanned surface's textures, tiled or not, this aspect is far more relevant than when a texture is hand painted or projected. The reason is that the later techniques mostly ignore edge seams, rather projecting maps, placed on a 3D view on to the UV space. In some cases hand painted textures or masks are going to be mixed to create the final material. For example, in Fig. 4 we can see a scabbard from Megascan's library [9] with 3 variations of leather. The rightmost example being just the scanned material from the library and the other two that were created using a combination of painted masks and tiled scanned surface textures. Neither of them uses any textures with a higher resolution than the other, but there are differences in finer details and modularity between them. Whenever these workflows come into play, the artist must take into consideration how each of these textures will play a role in the final asset in order to properly create them.

Figure 4. Comparison between a scanned material and two variations on a workflow that uses masking and tiled textures

An important technique in regards to UV mapping is the use of UV overlapping, which can be used to great effect depending on the type of asset being created. A rather common use case is to take into account model symmetry (Fig. 7) when it requires no texture variation from side to side, which is quite common for mobile games. For example, leveraging the symmetry might lead to doubling the pixel texture density for that portion of the model or even more depending on layout possibilities. That is not, however, the only possible use for UV overlapping,

another reason to use it is in situations with repeated meshes, such as buttons, bolts and others. They can potentially increase the texture density many times over. For example, a character model with the same button multiplied many times over along its clothing can either have a higher resolution in the end, or have these parts of the asset occupying a much smaller portion of the UV space, if they share the same mapping. This technique can even be taken a step further by using another technique, mentioned at IV-B, in which different models that can use the same material, such as the button, might reuse the same material instance in models with similar or identical UV layouts. A variation of this technique takes into account using a texture atlas. For example, in the same UV space one might have some variations of the same textures, or textures of similar nature, for which models with the same, or similar UV layouts are mapped, so that they fit to any one of these textures, making for a more organic result, while still leveraging the power of UV overlapping. That's an usual approach for creating the textures for character hair cards as showed in Fig. 5 and Fig. 6.

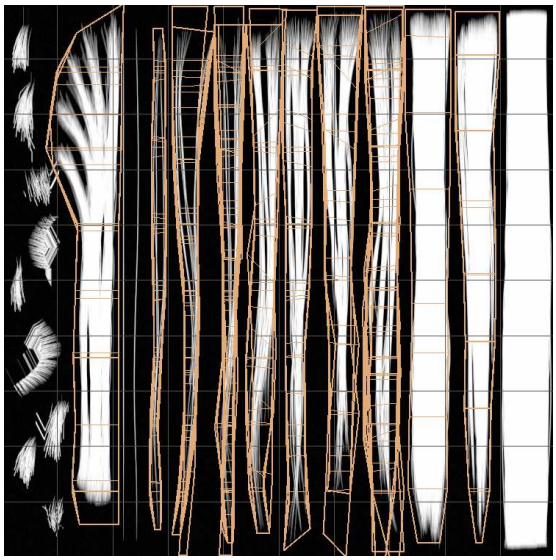

Figure 5. Example of a character's hair texture atlas

Figure 6. Example of a character's hair utilizing overlapping UV's for texturing

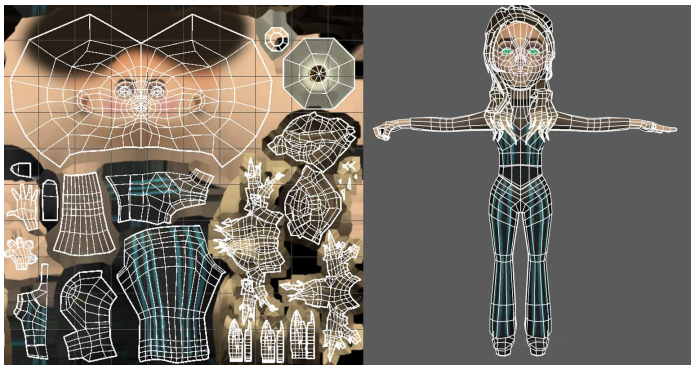

Figure 7. Example of a character using UV symmetry to make a more efficient use of UV space on the clothing

B. Leveraging material layers and material slots

Some materials might require the usage of material slots [20], layers [18] or both. These tools are common to most 3D software even though the implementation or terminology might vary. These are other options to assign different materials besides simply assigning the appropriate values on the textures themselves, either by painting or assigning these values with parameters on the shader. They can, most times, be used interchangeably but whenever applicable material slots are preferable. Due to material layers requiring more instructions and being, therefore, more expensive performance-wise [18].

Each works on a different level, material slots are assigned per-face, while material layers are blended through a function on whatever 3D software the artist is working on, such as Fig.10 in Unreal Engine. Layers are mostly used for materials that share the same faces, as they are blended on a per-pixel level and they work to great effect on scratched surfaces such as the leather on Fig.4, besides having the edge in creating that blending while also working with materials that require any kind of tiling (Section IV-C) or modularity. On the other hand, material slots require a different material for each slot, needing no extra functions or shader calculations in order to

work. For these reasons it has a better performance than layered materials, while still allowing the same degree of flexibility such as allowing tiling of textures.

With that, not only should the artist keep in mind what technique will be used for each asset, but also that they can be mixed and matched. For example, an asset that requires an organic transition between materials on its edge but has an uniform use of a single material otherwise can have 3 material slots. Which would be one for each surface with solid materials and one for the transition, such as Fig.8, taking advantage of the best aspect of each technique.

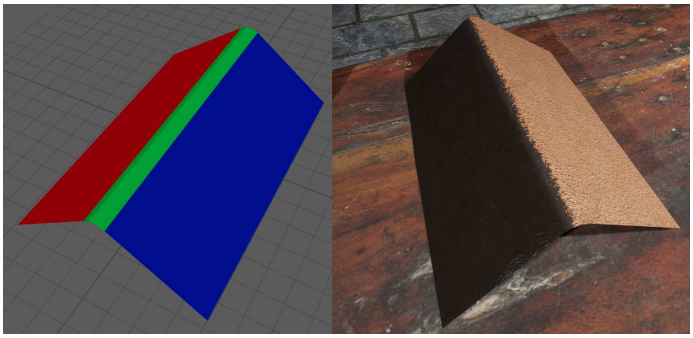

Figure 8. Material slots view (left), with a leather type in red, another in blue and the transition portion in green and resulting asset (right)

C. Shader-level UV manipulation and texture tiling

The UV coordinates for each texture can be manipulated on a shader-level, so that the final texture mapping can be changed accordingly. This principle can be used for multiple reasons, such as creating flipbook animations for effects from textures [21], changing the UV offset over time, moving the texture, rotating the textures, both of which could be used to create inexpensive (performance-wise) texture animations, and even texture distortion, which can be used for many effects, such as flowing lava textures or even fire [22].

The most common usage of UV manipulation is texture tiling, which is a rather basic concept in 3D texturing, referring to the repetition of textures in the UV space according to a parameter. In Unreal Engine [10] for example, the UV coordinates can be multiplied in order to output the final texture tiling. There are many different applications besides the basic one which is to repeat the textures for the material by themselves. An example is detailing on top of preexisting textures, usually to achieve micro detailing in surfaces such as skin [23]. In which case the modelling artist creates a high resolution model with skin detailing that can be projected to create a normal map, but does not have micro details, then, when the final material is assembled, these are layered on top. Also, even if the texture only uses traditional tiling, the tiled textures can be blended in, either on top of each other, with overlay, screen, opacity maps, normal map blending and others, or on a painting software, using layering on a texture level, in a similar fashion to the material layering mentioned in IV-B, with a single texture set [24].

Besides that, there are tiling methods other than traditional tiling that can be used for different purposes, often at a performance trade-off. Two of them are the aperiodic tiling method [25], and texture bombing [26]. They serve similar purposes, both tile textures in ways that doesn't show an obvious pattern, unlike traditional tiling, there are some key differences in the way they should be applied though. Aperiodic tiling requires the tiles to be driven in a way that each texture tile fits into all surrounding tiles, which makes it so that their boundaries still exist and could still be identified if the edges were framed for example. But they don't repeat the same visual over, while still maintaining the patterns within the tiles intact, which works great for caustics[25], hard surfaces and handmade patterns. Texture bombing, on the other hand, samples portions of the texture and blends these samples over each other, in a way that makes the tiling unrecognizable. That makes it unhandy to describe patterns, while working perfectly for surfaces such as leather, granite, or any other surface that has detailing too big for traditional tiling to handle without making the repetition obvious, but still too little to be described on either the base model or it's projected maps.

D. Texture picking, projecting, painting and authoring

The creation and selection of textures can be done in various ways. The most basic of which is choosing them from a texture database, such as Megascans [9], TextureHaven [27] and others. These are a great way to make plausible materials using scanned textures. However, they do not work for all situations, depending on your decisions when analysing the needs of material that approach might not even be an option, for example, when creating stylized materials. Besides that, these textures don't often work by themselves. Depending on the meshes, their UV layouts and texel density, the shader will require additional work in order to properly utilize these textures. A set of textures from a single scanned source might also not be enough to describe the entire surface of an object, requiring layering IV-B or to be used in texture painting.

Texture painting refers to the painting of texture maps and masks, that can be done in 2D or 3D painting softwares such as Substance Painter [8]. In which you paint directly on top of the 3D model, having instant feedback through a real-time render engine. This technique is pretty flexible, being integrated in a multitude of workflows and styles. The artist can just paint on top of the model for greater control on the final output, besides being able to go for a more simplified/stylized look. Besides that, options are given to combine this painting with fill layers using authored or scanned material textures and generators [28] to create procedural masking and colors, both for stylized and realistic looking work.

Material authoring or, sometimes, just "material creation", which might get confusing, so will be referred to as authoring [29] in order to avoid confusion, refering to the creation of material's textures using shapes, attributes, preexisting assets, such as 3D models or textures, and scripting in softwares such as Substance Designer[7]. These can be used by themselves or combined with painting and masking workflows for the final

result. In Fig. 9 we have an example of a marshland texture authored in Substance Designer without the use of any external software. It has a procedural ”wetness” property that allows it to blend from a flooded marshland texture to a dry one. Since this workflow also allows the use of preexisting assets [6], it makes for a greatly flexible and complete material creation workflow, that works for great effect when used with scanned textures, for example.

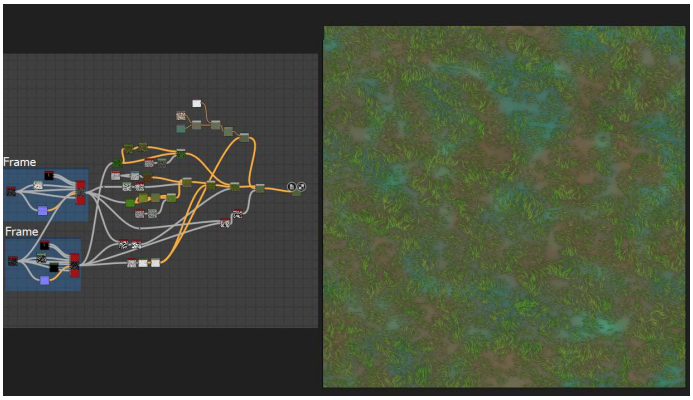

Figure 9. An example of an albedo texture(right) created using substance designer and its graph (left)

E. Shading

The shader will determine how the created/selected textures will affect the final material output. It is at this stage that things like the tiling, mentioned at IV-C and shader-level layering, mentioned at IV-B are implemented. What functions are available out of the box, how much customization is possible, among other particularities depends on the rendering software of choice. As said before, the tests and most of the research for this paper were done in Unreal Engine 5 [10], which offers a great array of tools out-of-the box and a visual scripting system, called material editor [30], requiring no coding from the artist in order to create detailed and efficient materials in a variety of styles.

In the example Fig.10 we have a graph describing a material function that toggles between two tiling techniques IV-C, that's part of the shader. However, that's just a high-level way to modify the shaders in Unreal, which as many other software, allows the user to code their own shaders from scratch and create variations of the existing ones [13]. Whether the artist works with the available tools to create shaders or program them from scratch, an important aspect is to be organized in a way that they can be reused efficiently for more materials whenever possible. The reason is that certain parameters and functions setup or a layering setup might work for multiple materials in the same way. Unreal Engine's hair shader, for example, can be used for vastly different hair styles, colors and models as long as the artist generates the appropriate textures [31], the example in Fig. 6 was assembled using a shader that way.

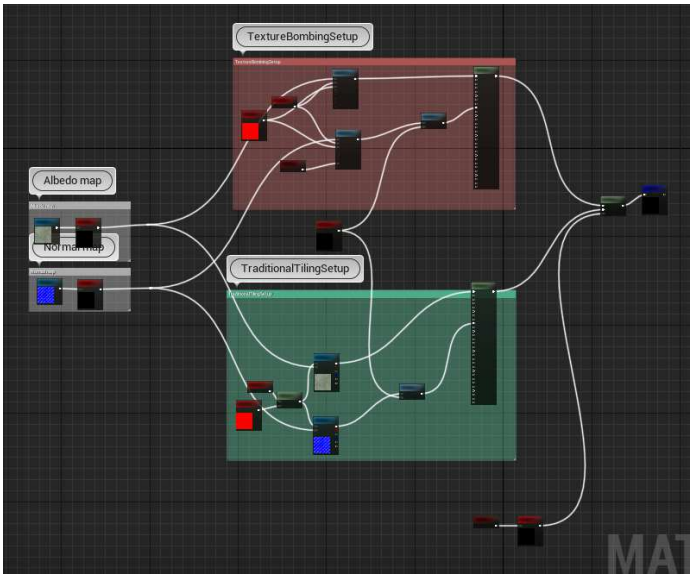

Figure 10. Material layer function overview

F. Material assembly

At last we have the assembly, which is to say, use the shader and the textures to create the final material. This step varies greatly depending on the workflow and assets chosen. There are one of a kind materials, for example, where the assembly and shading are one and the same, since the material isn't supposed to be modular or reusable. When the material is modular and can be used for multiple purposes is when this step is most important, since you can have all the textures and shader needed for a material ready, assemble it and assemble another reutilizing the shader or textures to make more variations of the same asset or to make different assets entirely.

V. Conclusion

Having set out to analyse and study different methods for the creation of materials for games, creating its textures and shaders it became clear how varied and deep this process might be. More than just going for the most detailed and high fidelity materials possible there are other aspects need to be taken into consideration. The artist or the art team working on a game need to have in mind as many of the tools and techniques available in order to achieve the final result, since not all assets are created the same way and a technique that works for one might not even be available as an option for another. Even though art direction, stylistic decisions and art fundamentals weren't discussed much in this work, they are of the utmost importance. These aspects of creating an asset can be hindered by a lack of technical knowledge. Understanding of the techniques herein described opens up much more possibilities than trying to solve every material problem through higher and higher resolution textures every time. Which might make it a bad experience for the player, and even the developers themselves, since working smarter and leveraging some of these methods might make for faster iteration.

Some points were either left untouched or just glossed over. Specially from a performance standpoint, more tests and a more methodical approach would be required in order to test the advantages and disadvantages of similar techniques, such as comparing the different tiling methods. Also, the examples were few and far between. Having a paper or tutorial touching on a specific workflow using the mentioned techniques while both going more in-depth on it and presenting more use cases would greatly help in exploring this paper's topics. And even more so, even taking into consideration that this work's intention is presenting some of the possibilities when working with game-ready materials and textures, there are still many techniques and tools to cover. Of note, also, is the fact that textures also play gameplay and narrative roles, which weren't mentioned, but an artist should look into and could relate with the techniques of this work.

References

[1] M. Labschütz, K. Krösl, M. Aquino, F. Grashäftl, and S. Kohl, "Content creation for a 3d game with maya and unity 3d", *Institute of Computer Graphics and Algorithms, Vienna University of Technology*, vol. 6, p. 124, 2011.

[2] E. Games. (2021). "Performance guidelines for artists and designers — unreal engine documentation", [Online]. Available: https://docs.unrealengine.com/4.26/en-US/TestingAndOptimization/PerformanceAndProfiling/Guidelines/ (visited on 07/20/2021).

[3] Microsoft. (2021). "Textures", [Online]. Available: https://docs.flightsimulator.com/html/Asset_Creation/Textures.html (visited on 07/20/2021).

[4] A. Blizzard. (2021). "Call of duty: Modern warfare system requirements - blizzard support", [Online]. Available: https://us.battle.net/support/en/article/244496/ (visited on 07/17/2021).

[5] W. McDermott, *The pbr guide*. [Online]. Available: https://substance3d.adobe.com/tutorials/courses/the-pbr-guide-part-1.

[6] A. Beddows, *Artstation - why you are struggling to learn substance designer - breaking an idea down*. [Online]. Available: https://www.artstation.com/learning/courses/xQY/why-you-are-struggling-to-learn-substance-designer/chapters/P5pA/breaking-an-idea-down.

[7] Adobe, *Substance designer*, version 2021.1.1, Jun. 13, 2021. [Online]. Available: https://www.substance3d.com/products/substance-designer/.

[8] ——, *Substance painter*, version 2021.1.1, Jun. 13, 2021. [Online]. Available: https://www.substance3d.com/products/substance-designer/.

[9] Quixel. (2021). "Quixel megascans", [Online]. Available: https://quixel.com/megascans/home/ (visited on 06/13/2021).

[10] Epic Games, *Unreal engine 5*, version 2021.1.1, Jun. 13, 2021. [Online]. Available: https://www.substance3d.com/products/substance-designer/.

[11] M. Bailey and S. Cunningham, *Graphics shaders: theory and practice*. AK Peters/CRC Press, 2009.

[12] B. Karis and E. Games, "Real shading in unreal engine 4", *Proc. Physically Based Shading Theory Practice*, vol. 4, no. 3, 2013.

[13] E. Games. (2018). "Shader development — unreal engine documentation", [Online]. Available: https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/Rendering/ShaderDevelopment/ (visited on 01/04/2021).

[14] ——, (2018). "Unreal engine post processing - youtube", [Online]. Available: https://www.youtube.com/watch?v=cQw1CL0xYBE (visited on 01/04/2021).

[15] J.-W. Lee and Y. Kim, "Rendering performance evaluation of 3d games with interior mapping", *Journal of Korea Game Society*, vol. 19, no. 6, pp. 49–60, 2019.

[16] M. Hassan, *Proposed workflow for uv mapping and texture painting*, 2016.

[17] M. Andrieshyn. (2018). "What is texel density and how to master it", [Online]. Available: https://www.youtube.com/watch?v=5e6zvJqVqlA9 (visited on 07/18/2021).

[18] E. Games. (2021). "Material layers — unreal engine documentation", [Online]. Available: https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/Materials/MaterialLayers/ (visited on 07/08/2021).

[19] T. Foundry. (2021). "Working with uv maps", [Online]. Available: https://learn.foundry.com/modo/content/help/pages/uving/working_with_uvmaps.html (visited on 07/17/2021).

[20] E. Games. (2021). "Fbx material pipeline — unreal engine documentation", [Online]. Available: https://docs.unrealengine.com/4.26/en-US/WorkingWithContent/Importing/FBX/Materials/ (visited on 07/08/2021).

[21] ——, (2021). "Render a flipbook animation — unreal engine documentation", [Online]. Available: https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/RenderToTextureTools/5/ (visited on 07/19/2021).

[22] E. G. Simon Trümpler. (2018). "'stylized vfx in rime' by simon trümpler — unreal fest europe 2018 — unreal engine - youtube", [Online]. Available: https://youtu.be/ExD_p3hsV80?t=241 (visited on 07/19/2021).

[23] A. Jorge Jimenez. (2013). "Jorge jimenez – next generation character rendering", [Online]. Available: https://docs.unrealengine.com/4.26/en-US/WorkingWithContent/Importing/FBX/Materials/ (visited on 07/08/2021).

[24] Adobe. (2021). "Layer stack — substance 3d painter", [Online]. Available: https://substance3d.adobe.com/documentation/spdoc/layer-stack-29130767.html (visited on 07/20/2021).

[25] J. Stam, *Aperiodic texture mapping*. Citeseer, 1997.

[26] S. Glanville, "Texture bombing", *GPU Gems: Programming Techniques, Tips, and Tricks for*, vol. 1, 2004.

[27] P. Haven. (2021). "Textures • poly haven", [Online]. Available: https://polyhaven.com/textures (visited on 12/12/2017).

[28] K. Aava. (2021). “Generator — substance 3d painter”, [Online]. Available: https://substance3d.adobe.com/documentation/spdoc/generator-109608968.html (visited on 12/12/2021).

[29] Allegorithmic. (2017). “Substance designer - the ultimate 3d material authoring tool - youtube”, [Online]. Available: https://www.youtube.com/watch?v=vVta%5C_LekQxY/ (visited on 07/12/2021).

[30] E. Games. (2017). “Material editor reference — unreal engine documentation”, [Online]. Available: https://docs.unrealengine.com/4.26/en-US/RenderingAndGraphics/Materials/Editor/ (visited on 05/21/2021).

[31] ——, (2021). “Photorealistic character — unreal engine documentation”, [Online]. Available: https://docs.unrealengine.com/4.26/en-US/Resources/Showcases/PhotorealisticCharacter/ (visited on 07/19/2021).