The New Age of Procedural Texturing

Dr Sébastien Deguy

CEO and Founder, Allegorithmic

Synonyms

VFX : Visual Effects ; HD : High Definition ; UI : User Interface ; UX : User Experience ; CG : Computer Graphics.

Definitions

In the computer graphics and game development worlds, a *texture* is the digital representation of the appearance of a surface ; *Procedural texturing* is the process of using an algorithm in order to generate a texture.

1 Introduction

We believe we have entered a new age for procedural texturing, an age where these techniques are not only interesting to the theoretically concerned, the technically-minded 3D artists who put them into practical use, but to a wider audience. Procedural techniques have indeed, now extended past the most high-end of game development studios and the VFX and animation houses that gave them their first homes. As this new age progresses, the presence and use of procedural techniques for creating textures is only going to grow more entrenched in the years to come.

In this article, we will focus mainly on the reasons why this new age is upon us, and on some of the new techniques and approaches that have emerged in the past few years. The subject of procedural texturing, in itself, is already very known and covered (see in particular [2] for more on this topic), so we will not go into much details about the science behind procedural texturing, rather we'll focus on the exploration of new trends emerging from the visual effect and gaming industries.

2 What is a procedural texture?

A procedural texture is a computer-generated image created using an algorithm (this is where the term *procedural* is derived from: a *procedure* is driving the process), instead of a digital painting or image processing application such as Adobe™ Photoshop™ (see figures 1 and 2).

Procedural texturing involves a two-step process:

- **the "authoring" phase**: During this first phase, procedural textures authors pick and combine the techniques they want to use, set the parameter values that drive these techniques and expose the meaningful parameters allowing to change the resulting textures' look and feel (see figure 3). These parameters can be of any type, from numerical values (the fractal dimension of a fractional Brownian motion generation algorithm for instance, see [1]) to complex multi-array data (an image, i.e. an array of RGB numerical values, can in that respect be treated as a parameter to procedural texturing algorithms, see figure 4). At this authoring stage, the result of the procedural texturing process is not visible yet: the generation phase needs to be activated.

- **the "generation" phase**: During this phase, the program and parameter values defined in the previous phase are interpreted and executed, in order to produce the final output(s) (in the particular case of procedural texturing: textures). The process is *realised*. Two main methods can be witnessed (see [2] for more details):

  - **implicit generation**: in this case the generation algorithms are analytically defined, allowing to be easily called in a random fashion. Usually, rendering engines will ask for a certain (u,v) value to be generated in the parameterization space for the given geometry upon which the procedural textures are applied. With this approach, no bitmap texture is stored in memory, usually at the cost of "expression power".
  - **explicit generation**: here a resolution for the final output textures is set and the whole texture is rasterised, in one and only one step, and "baked out" as a bitmap texture, stored in central or video memory to be accessed by the rendering engine. This explicit techniques allow for a very wide range of output, referred to as " high power of expression".

  In both cases, the generation phase is done by a *generation engine*.

3 Why use procedural texturing?

The motivation for using procedural texturing can be very varied, going from speed of authoring to fast delivery of content. Here are some of the most important reasons why creators should use procedural texturing:

- **Productivity**: the rise of Ultra HD and even higher resolution displays, combined with the increase in project sizes (see games such as Rockstar^{TM}'s Grand Theft Auto^{TM} where the player can freely explore gigantic areas), again combined with the always decreasing (relative to the amount of content produced) production budgets (and margins to creators) of computer graphics and digital assets represent a significant challenge to creative teams. Large volume of assets need to be created as quickly and cheap as possible, at the utmost levels of quality. Here procedural texturing, with its "data amplification" nature (see [2]), can

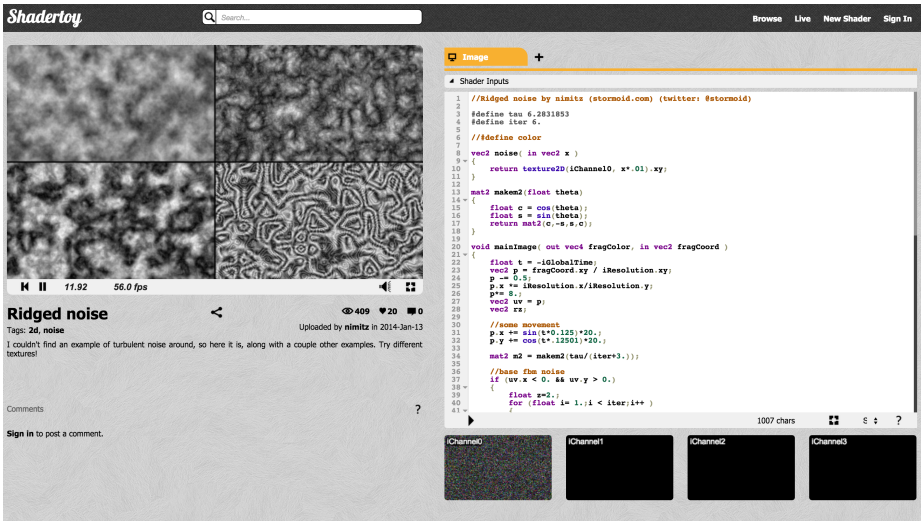


Figure 1: The four textures on the left have been procedurally generated using four different parameter values from the procedure code on the right. Extract from Inigo Quilez's online tool Shadertoy Beta (see [6]).

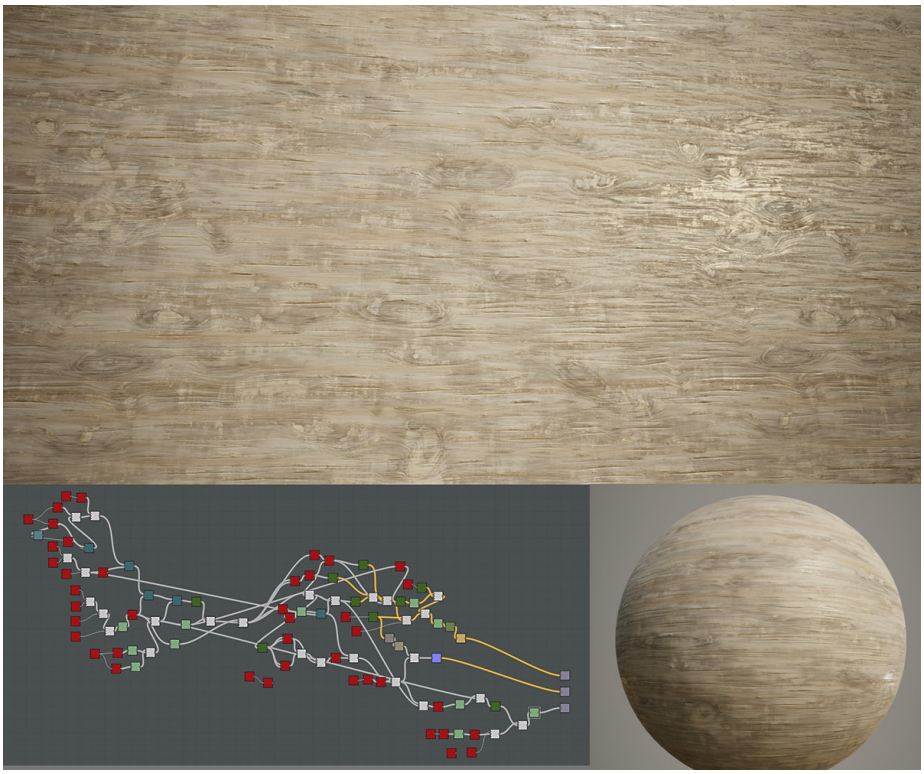
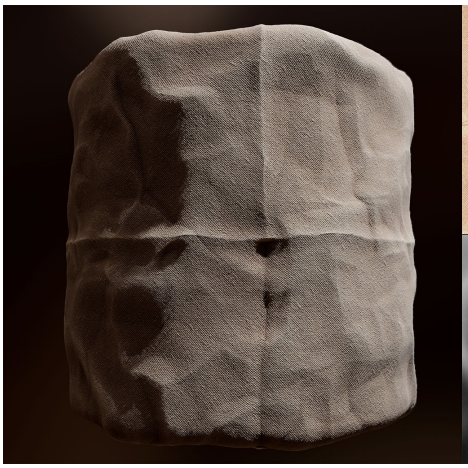
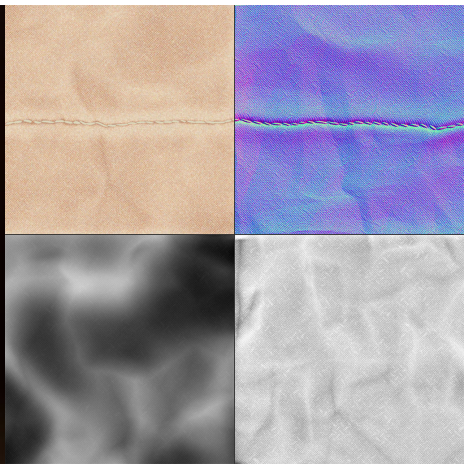


Figure 2: This plywood procedural material is derived from the procedure visually represented as a graph at the bottom left. Tool used: Substance Designer™ by Allegorithmic™. Extract from Hugo Beyer's online portfolio (see [4]).



(a) a procedural fabric material



(b) the various texture maps necessary to the full fabric material

(c) variations of the same fabric procedural material

Figure 3: By playing on some parameter values of the procedural texturing algorithm created to generate this fabric texture, one can obtain variations in the look and feel: fabric type, level of dirt, presence or absence of stitches, etc. Extract from the work of Bradford Smith, see [7].

(a) (b)

(c) (d)

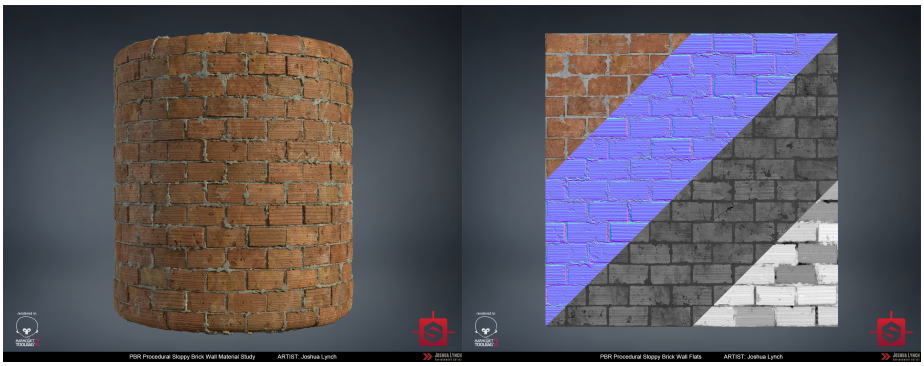
Figure 4: Images can be treated as input parameters of procedural texturing techniques. In these procedural materials, images at the bottom left are used to drive the positioning and shape of the stones in the final material. Extract from the work of Nicolas Wirrmann from Allegorithmic, see [5]. (logo at the bottom left of image (d) is the Polycount logo ("Greentooth"), the reference discussion forum when it comes to technical 3D art, see [18])

definitely help, as once a procedural texture definition is set and approved for one realisation of the process, one can change parameters, and in particular the random seed used when generating the textures (see [2]), to automatically create a theoretically infinite number of other textures. These will look like they are part of the same "family" but will be different when compared pixel by pixel (see [2]). A lot of creative studios now use procedural techniques to produce very large amounts of textures, used either as is, or as a building block to other, more complex textures, using a combination of various creation techniques. The gains are non linear and higher as time passes as creators can hoard techniques and texture definitions in a library of generative, versatile content. Finally, if the art direction for the content changes at some point in the process, it is easier than ever to change one input value to all the procedural textures, then re-generate all your batch via a single command. Here again, the gains in productivity are immense.

- **Complex texture production**: sometimes replicating an effect witnessed in nature or fantasised by an artist is not an easy task. References can be lacking, and in that case, iterations are the key to success in a trial and error process. Not having to paint a new texture for every iteration, or take a picture of an effect so it can be processed and used as a proper texture, can lead to huge time savings. Playing with parameters and simply calling again against the generation engine is a very important time saver (going back to the previous, productivity-centric point), making it a good step towards the production of the desired look. It is also worth noting that the level of complexity that noise generators especially can produce is something that is very hard to replicate by hand. Here again, procedural noise generation and overall texturing techniques can prove very helpful (see for instance figure 5).

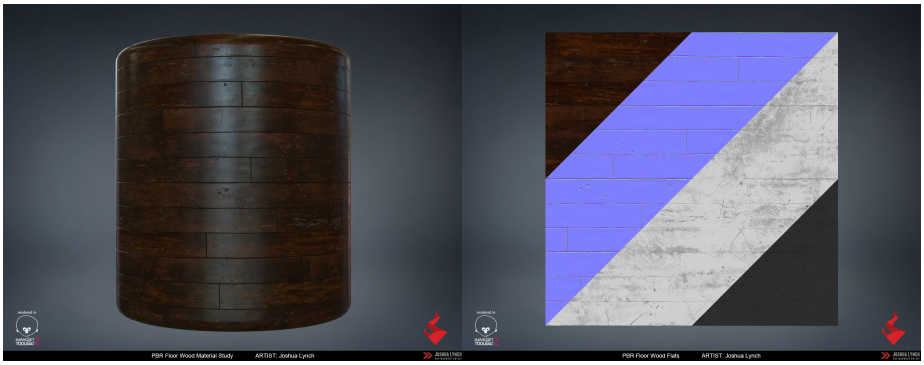
- **Look consistency**: Applying procedural generation techniques in the production of a large amount of textures can also prove very important to ensure the look consistency of the assets. Most of the time, even if not guaranteed, changing parameter values will lead to textures sharing the same visual properties. Also, by controlling the procedural techniques used by creators, one can make sure the outputs of these processes and the processes themselves respect specific properties and constraints respectively, such as "the roughness map only contains values within a certain range", or that "the normal map is only computed after all the computation on the height maps have been done, this latter map being in grayscale 32f format instead of 24b colour for the albedo map", etc. An automated process can definitely help ensure rules are respected and that the assets maintain a consistent look once imported in the game engine.

- **Data size optimisation**: Procedural textures description files, especially when they are of the "purest" form (see Definitions section), can be very compact. Due to the descriptive nature of procedural techniques, and the two-step process (authoring, then generation, see section 2) of procedural texturing, a procedural texture description file will typically weigh a few KiloBytes instead of several MegaBytes or GigaBytes in relation to their bitmap equivalent. This is often referred to as "data amplification", and



(a) brick and mortar procedural material wrapped onto a cylinder

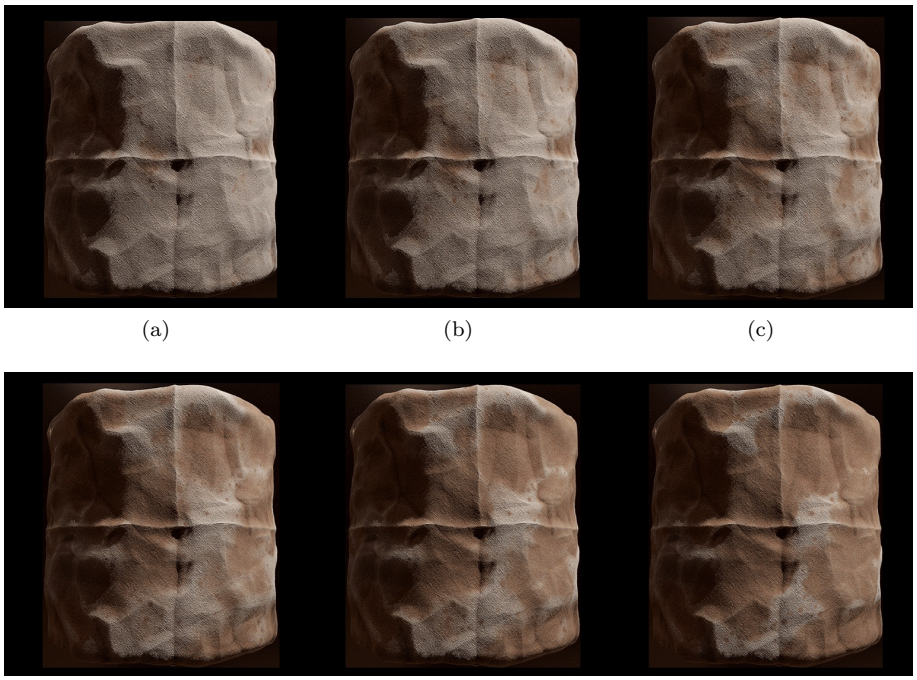
(b) the different textures composing the brick and mortar material



(c) wood panel procedural material wrapped onto a cylinder

(d) the different textures composing the wood panel texture

Figure 5: These textures (all the components of the digital materials presented here, like "albedo", "normal", "roughness" and "metallic" maps) would have been extremely complex and time consuming to produce by hand. Procedural techniques, on the other hand, produce these textures quite easily. Extract from Joshua Lynch's online portfolio, see [3].



(a) (b) (c)

(d) (e) (f)

Figure 6: In these realisations of the same procedural texturing algorithm, defining parameters can make the fabric more wet and dirty. Extract from the work of Bradford Smith, see [7].

can prove useful when the generation process can be done on the "client side": in the case of a mobile or online game for instance, the players will only download a set of very light description files on their local device (the "client"). These files are then generated locally by the generation engine whenever they are required to display (install-, load- or run-time). This generation process is often faster than the download time, making the benefits two-fold: faster download times and lighter network consumption.

- **Animated and customised textures**: Because procedural textures are defined by parameters and generated in a two-step process through a generation engine, and in the case that generation engine is embedded in the playing application (i.e. game or virtual experience for instance), one can modify these parameters' values at run time and call against the engine to regenerate the output textures with a new look and feel to them. This leads to very interesting effects such as animated textures (see figure 6), adaptive textures (more or less dirt after an explosion occurred in the area where they are set for instance) or simply user customization (the idea of letting the player of a game choose parameter values herself to "tweak" the look of the assets to their desire: see figure 7).

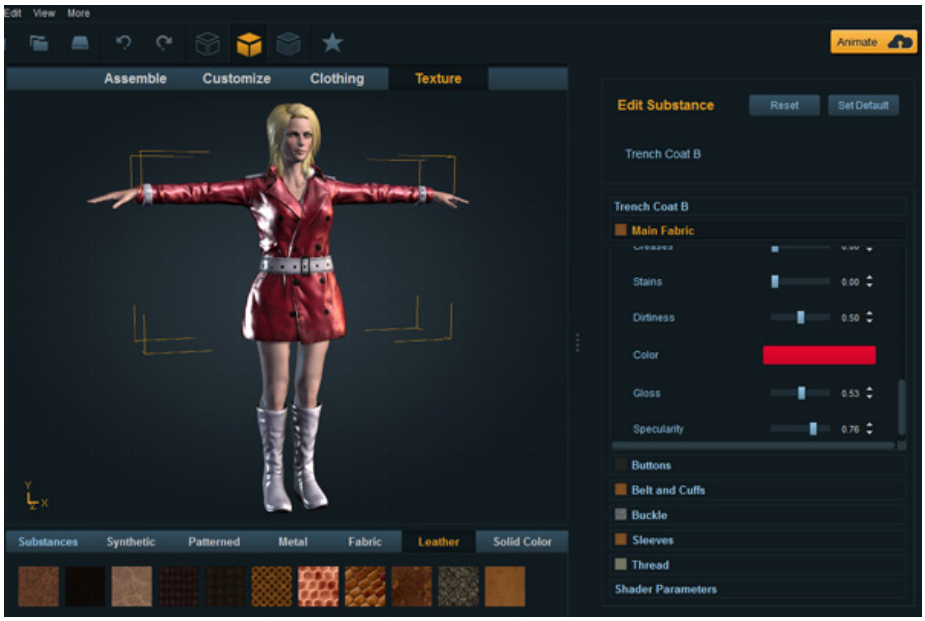


Figure 7: The textures on this virtual character are fully procedural, and can be controlled by the operator via the parameters on the right side of the interface. Extract from Adobe™ Fuse™, see `/https://community.mixamo.com/hc/en-us/articles/203533673-Tutorial-Create-a-Character-with-Fuse`.

4 Misconceptions about procedural texturing

Procedural texturing has suffered in some circles from a bad reputation. Among the main misconceptions, we can cite:

- **Procedural textures look too "mathematical"**: It is true that, when confined to 30+ year old techniques, the outputs one can obtain by using "colorised" noise functions can look fake or the result of a very mathematical process. Just like any other tool, procedural texturing techniques can bias outputs, so it is up to the artist to remove this by enhancing the techniques and / or use of these techniques. Combining various outputs often breaks this "mathematical look" (see figures 8 and 9) and today's production of procedural textures are now operating at the highest quality levels.

- **Procedural textures are only for tilebale textures**: This point also inherits from 30+ years of procedural texturing techniques, the early stage of which found most techniques being used to produce square, tiling textures that would be repeated over surfaces in a way that optimised video memory usage and accesses. But there is no limitation for procedural textures to be evaluated directly onto a 3D surface and / or to take into account the geometry onto which the textures are then wrapped (see figures 10, 11 and 12 and tools like Substance [5]).

- **Procedural texture limit artist's creativity**: Up to now, the avail-

Figure 8: This photorealistic "Sandy Ground" procedural material is the result of the combination of several noises, see figure 9. Extract from the work of Bradford Smith Adobe^{TM} Fuse^{TM}, see [7].

ability of tools and knowledge needed to advance procedural techniques were limited. This converges with an outdated view that believes procedural textures should be only evaluated at runtime, leading to techniques that produce undesired outputs, especially with regards to their artistic value. A lack of user input is indeed frustrating to artists. This aspect is now mostly outdated: see 5 and the numerous counter-examples displayed in this article.

- **Procedural textures are "replacing" artists**: Productivity is one of the main advantages of using procedural texturing. In my observations, most studios that implement procedural texturing techniques enjoy more creation time thanks to the practical use of these techniques. This time is then used to increase the quality of their final textures. Procedural textures are also used as a way to quickly generate a texture draft that can then be reprocessed, or sometimes completely redone, by hand. My view is that procedural texturing will only replace the mere, repetitive texturing tasks with no added value, rather than completely "replace artists": artistic vision is still key in the texture creation process.

- **Procedural textures are slow to compute and should only be implicitly evaluated**: A higher complexity in the procedure necessary to generate textures at the desired graphical level implies longer computation times. Combined with limited, available resources on devices, this factor can limit the spread of the techniques. Also an inherited thinking is that procedural textures are necessarily implicitly evaluated, which dramatically limits the power of expression of the techniques at hand. This point is addressed at length in section 5.

- **Procedural textures are barely used in real production**: All the

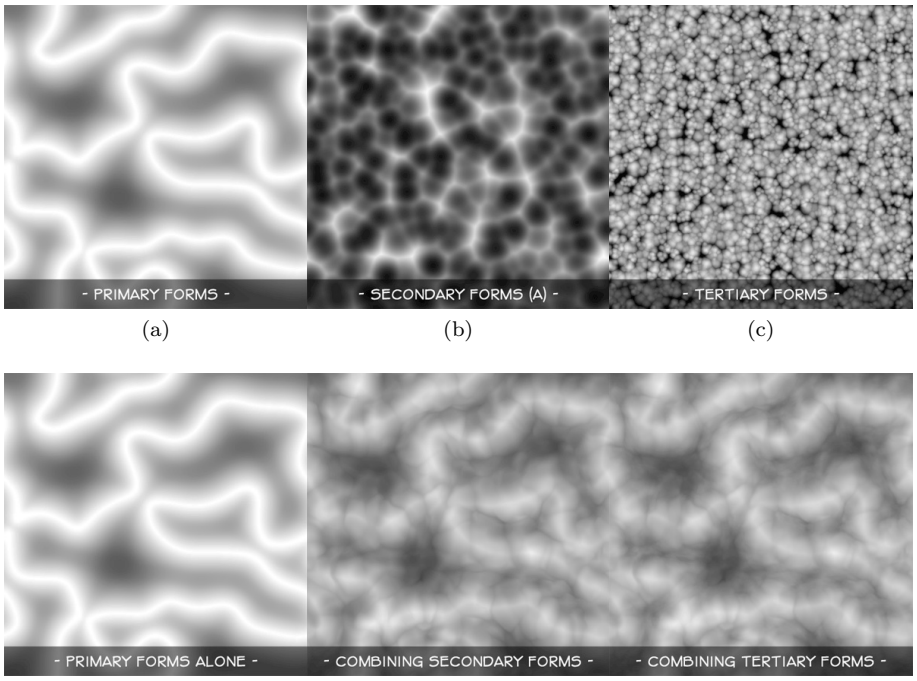
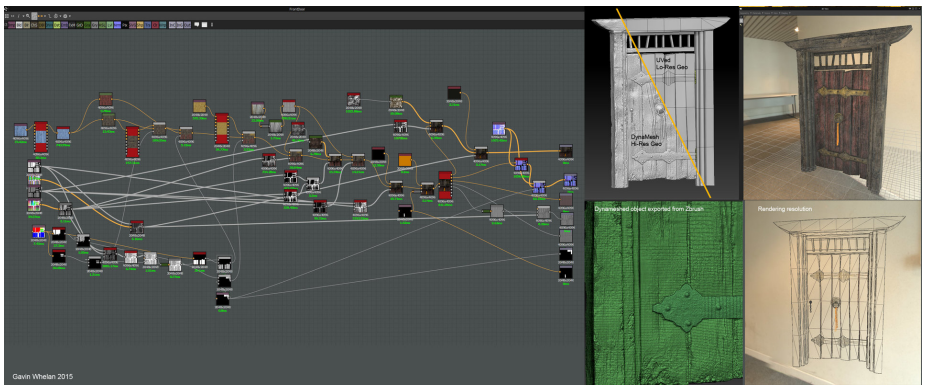


Figure 9: In order to produce the procedural "Sandy Ground" material presented in figure 8 , author Bradford Smith combined three different, procedural noise functions: (d)=(a); (e)=(a)+(b) ; (f)=(a)+(b)+(c). (f) is the final result, utilised as a height (or displacement) map in figure 8. Although each "form" or noise function is the result of a pure mathematical process, the end result is photorealistic. Extract from the work of Bradford Smith, see [7].

Figure 10: This character has been textured using a procedural, non-destructive methodology offered in Allegorithmic™ Substance™ Designer™. Extract from the work of Samuel Compain, see [19].

(a) this scene has been entirely textured using a procedural approach

(b) most of the procedural textures in this scene are not mere tiling textures, but adapted to every, specific asset



(c) the procedural texturing process described in the graph on the left is dedicated to the door mesh on the right

Figure 11: All elements composing this scene have been produced using a procedural, non-destructive methodology offered in Allegorithmic™ Substance™ Designer™. Extract from the work of Gavin Whelan, see [20].

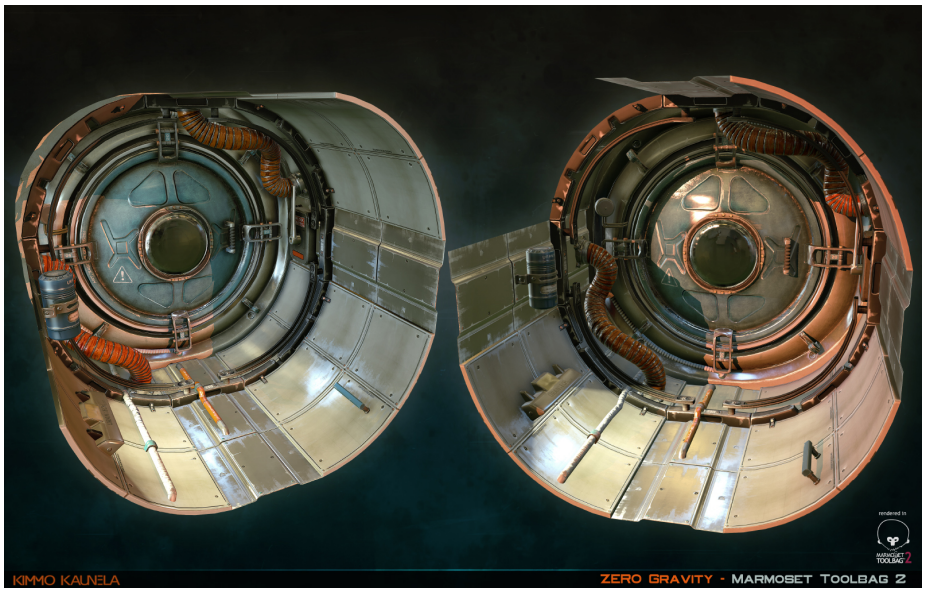


Figure 12: This scene has been entirely textured using Allegorithmic™ Substance™ Designer™, a procedural texturing tool of the latest generation. Note how the mesh properties are taken into account in the process, such as on the edges of metallic parts – material is more aged in these areas. Extract from the work of Kimmo Kaunela, see [17].

limitations and misconceptions above have hindered the spread of procedural techniques in the past. That being said, and this is one of the main inspiration points behind this article, their now pragmatic use (see 5) has helped spread usage in production. At the time this article was published, a very significant percentage (25%+) of AAA games currently being produced make a large use of procedural techniques.

5 The New Age of Procedural Texturing

This New Age is made possible in large part by a combination of two categories of factors: human and technical.

- **Procedural techniques at authoring time**: First artists have realised that *procedural texturing finds maybe its best application case during the authoring phase of the content*, instead of the runtime phase. In fact, procedural texturing in most studios today are used as a tool more than a means, and the output of this procedural texturing process is simply "baked" out as a set of bitmaps. These bitmaps can then be modified using traditional tools and techniques and consumed using traditional techniques by game and rendering engines alike. This way of using procedural techniques and tools eradicates most of the drawbacks that artists and engineers might have experienced by using procedural textures at runtime (see section 4).

(a)

(b)

Figure 13: This scene has been textured using a pragmatic approach in Allegorithmic™ Substance™ Designer™: a mix of procedural and bitmap elements. Extract from the work of Koola, see [21].

- **A pragmatic use**: The large trend one can observe now is the *mix of techniques* used by the most advanced texture artists in the industry: using whatever tool is available and best suited for the task will be the choice of reason, and these tools and processes will be integrated in a holistic process made possible in particular by the "authoring-time" centric creation process described earlier: see for instance figures 13 and 14. This is a trend we will see getting stronger in the years to come as new artists begin to be freed from the misconceptions described in section 4.

The second category of factors is technical:

- **Available computational power**: The amount of computational power available to creators, whether on desktop computers, mobile devices or via cloud-based servers, is now higher than ever and has reached a point where generating content using procedural techniques is a viable option. Computationally expansive techniques, such as iteration-based algorithms can now be put in practice.

Figure 14: This character has been textured using a mix of procedural techniques and hand craft. It is representative of the latest advances in texturing: a pragmatic use of all available techniques at hand, including procedural texturing. Extract from the work of Juan Puerta, see [22].

- **Advances in Computer Science**: Techniques such as machine learning have made very important leaps in the past years and it is to be expected that these approaches, which are based on training a set of algorithms to produce textures according to an inputed set of examples, will lead to new advances in the field.

- **Available, more accessible tools**: Advances in UI and UX practices, combined with the rise of new tools dedicated to procedural texturing, such as the Substance™ tool suite that my company Allegorithmic™ develops of course (see [5] and figures 15 and 16), but others like Quixel Suite™ (see [8]), 3DCoat™ (see [9]) and the always relevant Side Effects Software™ Houdini™ (see [10]) have definitely supplanted the older generation of tools. More importantly, these tools are easy to use and do not require advanced mathematical or computer science and programming knowledge to be able to produce efficiently and professionally procedural textures. Their price points and dedicated business models make them accessible to smaller studios as well as enthusiasts, which is one more move toward the democratisation of procedural texturing. Finally, these tools are not limited or "exclusive" (in their approach) by design, and embrace a novel view of procedural texturing and propose new techniques and approaches (see next section).

6 New techniques and tools

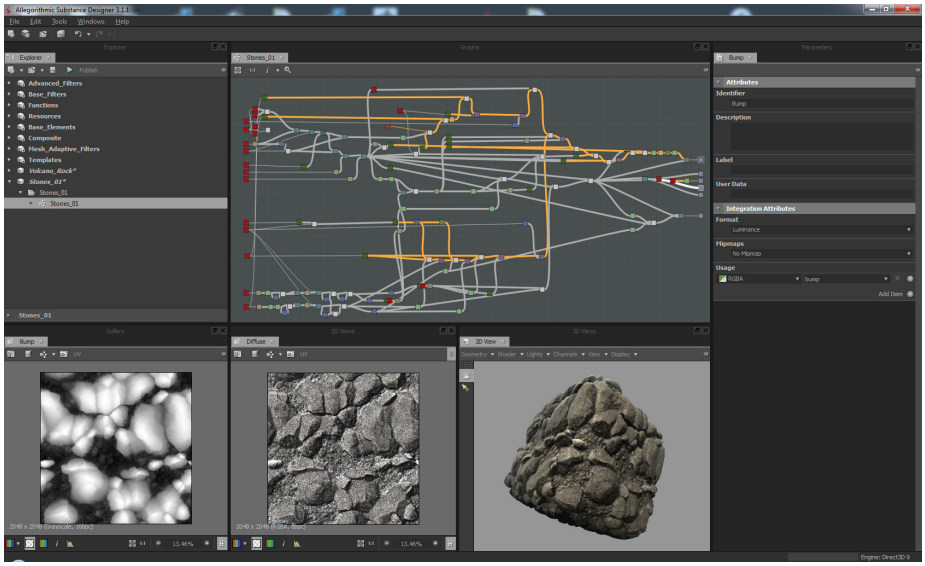
Maybe the most important point made in the past years by the creators of procedural texturing tools is the fact that the process has to be *hybrid*. Artists and developers alike do not want to "take on a side" anymore, whether to go full procedural or full manual. Procedural techniques, as we have seen, can now be utilised practically, and it is up to the artists to choose the technique they want to use for the particular task they have to complete. Modern tools allow for a hybrid approach, proposing procedural techniques that complement traditional hand-painting features, such as Allegorithmic™ Substance Painter™ (see [5] and figure 16).

Other techniques have been largely improved in the recent years, adding to the arsenal of options to the artists. Techniques such as image-based texture synthesis are now widely used and tools like Allegorithmic™ Bitmap2Material™ (see [5] and figure 17) or more recently Artomatix™ (see [13]) are procedural-based tools largely used in production.

Iterative generation techniques (see for instance figure 18) have also been made practical, especially thanks to the advances in computational power. Instead of taking a long time to render (for each iteration to go through), these procedures can now be quasi real-time, allowing artists to use a "trial and error" creative approach while modifying the parameters of the process in use.

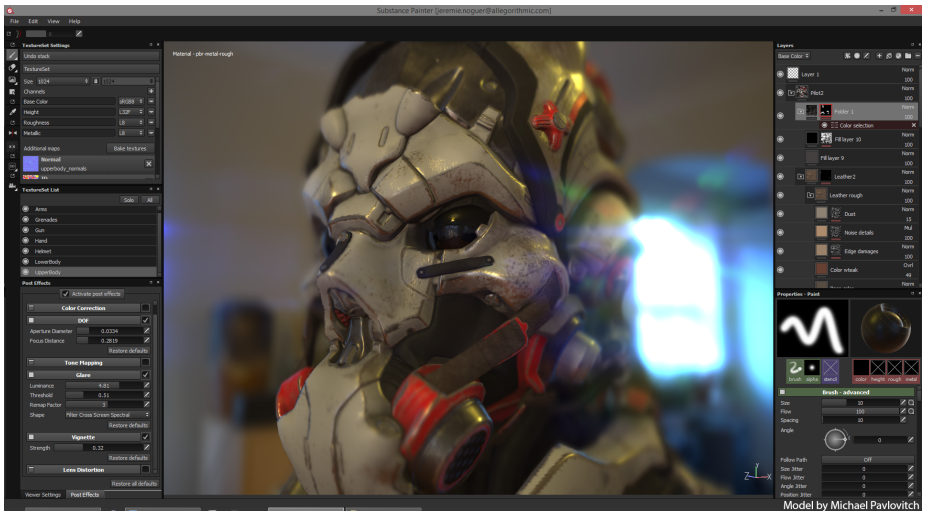
Also worth noting here, and very much aligned with the idea of hybrid approaches mentioned above: Allegorithmic™ recently introduced a procedural brush system in which a particle simulation algorithm is utilised to paint over a surface, while the user controls various parameters that define the behavior of the particles, as well as the position at which the particles are spawned and directed (see figures 19 and 20).

(a)



(b)

Figure 15: Substance Designer™ by Allegorithmic™ is a tool largely (but not exclusively) dedicated to the creation of procedural textures, while still being accessible to every artist thanks to its advances in UI and UX. See [5].



(a)

(b)

Figure 16: Substance Painter™ by Allegorithmic™ makes a large use of procedural texturing techniques on top of more traditional hand-painting features. See [5].

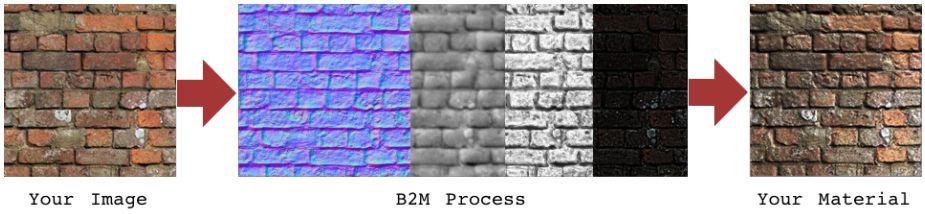
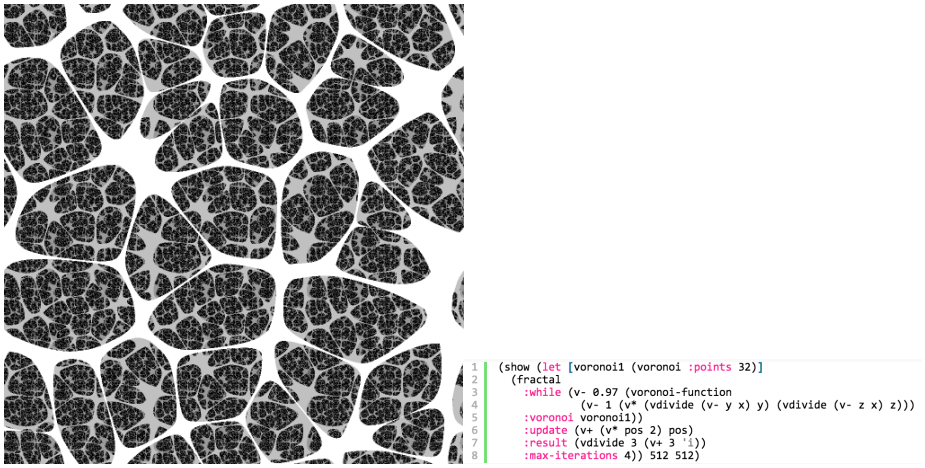


Figure 17: Bitmap2Material^{TM} by Allegorithmic^{TM} uses an image as input to procedurally generate a full digital material. See [5].



(a) "Voronoi fishing net fractal" texture, the result of an iterative process

```
(show (let [voronoi1 (voronoi :points 32)]
  (fractal
    :while (v- 0.97 (voronoi-function
                      (v- 1 (v* (vdivide (v- y x) y) (vdivide (v- z x) z)))
    :voronoi voronoi1))
    :update (v+ (v* pos 2) pos)
    :result (vdivide 3 (v+ 3 'i))
    :max-iterations 4)) 512 512)
```

(b) the generation code for the "Voronoi fishing net" texture

Figure 18: The "Voronoi fishing net fractal" texture on the left is the result of an iterative process. Advances in computational power now allow for such techniques to be used in production. Extract from Creative Clojure, see [14].

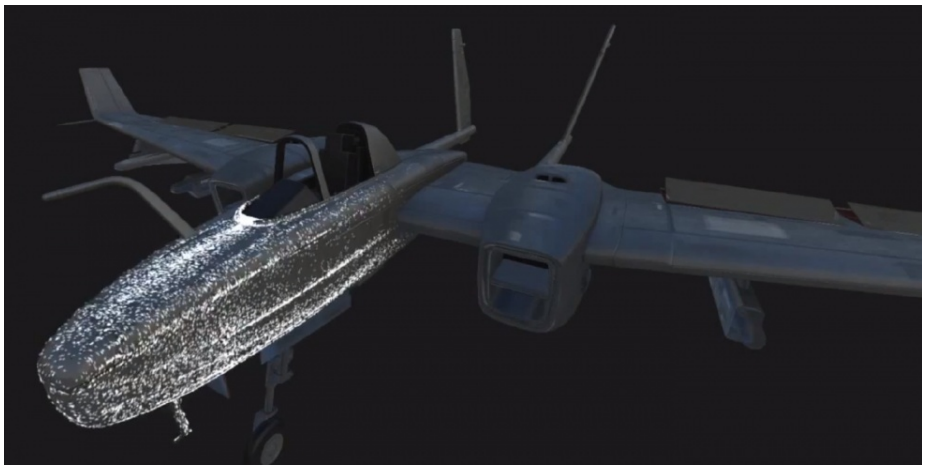


Figure 19: A particle simulation system is being used in Allegorithmic^{TM} Substance Painter^{TM} to give artists new procedural-based texture creation tools. The particles in white are "thrown" at the plane's nose and, following a certain behavior, these particles will flow against the structure in a realistic way and leave dirt traces in the textures wrapped onto the plane, where they are supposed to appear. See [5].

Figure 20: Most of the textures on this character make use of the particle-based procedural brushes available in Allegorithmic^{TM} Substance Painter^{TM}. See [5]. Image by Christophe Desse, model by Olivier Couston (see [16])

Figure 21: The images or textures at the bottom have been "dreamt" by an algorithm developed at Google Research™. Advances in machine learning and neural networks will lead to novel procedural techniques and more effective manipulation of procedural techniques by computers and artists. See [15].

Finally, we can cite an area of research which will lead to more advances in procedural texturing: *machine learning* and the latest developments in recurring neural networks, as illustrated on figure 21 (see [15]). These techniques will help both in producing new procedural techniques, derived from the experience and knowledge of the artists sharing their work and work process at a large scale (large enough so that algorithms can *learn* from it), as well as in the estimation of meaningful parameter spaces for a given procedural technique.

7 Conclusion

In this article We have been shedding some light on what can reasonably be called a new age for procedural texturing. Procedural texturing, in itself, is a technique more than thirty years old that can have a lot of benefits but has been only very lightly used in production compared to traditional techniques. Due to major advances in computational power, to the appearance of dedicated, user-friendly tools and to a pragmatic use of various, available techniques by artists, today's production studios are embracing procedural texturing at a scale that has never been witnessed before. In many ways, this age is only beginning

and we predict we will see a lot more advances in this field in the years to come.

References

[1] Sébastien Deguy, Christophe Debain and Albert Benassi, *Classification of texture images using multi-scale statistical estimators of fractal parameters*, in British Machine Vision Conference, M. Mirmehdi et al., 2000.

[2] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, Steven Worley, *Texturing and Modeling, A Procedural Approach*, AP Professional, ISBN 0-12-228730-4, 2nd edition, 1998.

[3] Joshua Lynch, *Texture Artist*, `http://josh-lynch.com/texture-sheets/`, 2015.

[4] Hugo Beyer, *Texture Artist*, `https://hugobeyer.artstation.com/`.

[5] Allegorithmic, *Substance Tools*, `/http://www.allegorithmic.com`, since 2003.

[6] Inigo Quilez, *Shadertoy*, `https://www.shadertoy.com/`.

[7] Bradford Smith, *Technical Artist*, `http://bradfolio.com/`.

[8] Quixel, *Quixel Suite*, `http://www.quixel.se`.

[9] 3D Coat, *3D Coat*, `http://www.3d-coat.com`.

[10] Side Effects Software, *Houdini*, `http://www.sidefx.com`.

[11] Spiral Graphics, *Genetica*, `http://www.spiralgraphics.biz/genetica.htm`.

[12] Dark Sim, *Dark Tree*, `http://www.darksim.com/html/dt25_description.html`.

[13] Artomatix, *Artomatix*, `http://artomatix.com`.

[14] Creative Clojure, *Creative Clojure*, `https://clojurefun.wordpress.com`.

[15] Google Research, *Inceptionism: Going Deeper into Neural Networks*, `http://googleresearch.blogspot.com/2015/06/inceptionism-going-deeper-into-neural.html`.

[16] Christophe Desse, *Xtrm3D*, `http://www.xtrm3d.com/`.

[17] Kimmo Jaunela, *3D Artist*, `http://kimmokaunela.esy.es/galleria.html`.

[18] Polycount, *Polycount Forum*, `http://www.polycount.com/forum/`.

[19] Samuel Compain, *Character Artist*, `https://www.artstation.com/artist/samuel_compain`.

[20] Gavin Whelan, *CG Artist*, `https://www.artstation.com/artist/gavinwhelan`.

[21] Koola, *CG Artist*, https://www.flickr.com/photos/99646627@N03/.

[22] Juan Puerta, *Character Artist*, https://www.artstation.com/artist/juan_puerta.