UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

JEFFERSON MAGALHÃES PINHEIRO

A Procedural Model for Snake Skin Texture Generation

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. Marcelo Walter

Porto Alegre
November 2017

ACKNOWLEDGEMENTS

We would like to thank Marluci Müller Rebelato, of the Biosciences Institute at Universidade Federal do Rio Grande do Sul, for her advice on snakes biology.

RESUMO

Existem milhares de espécies de serpentes no mundo, muitas com padrões distintos e intricados. Esta diversidade se torna um problema para usuários que precisam criar texturas de pele de serpente para aplicar em modelos 3D, pois a dificuldade em criar estes padrões complexos é considerável. Nós primeiramente propomos uma categorização de padrões de pele de serpentes levando em conta suas características visuais. Então apresentamos um modelo procedural capaz de sintetizar uma vasta gama de textura de padrões de pele de serpentes. O modelo usa processamento de imagem simples (tal como sintetizar bolinhas e listras) bem como autômatos celulares e geradores de ruído para criar texturas realistas para usar em renderizadores modernos. Nossos resultados mostram boa similaridade visual com pele de serpentes reais. As texturas resultantes podem ser usadas não apenas em computação gráfica, mas também em educação sobre serpentes e suas características visuais. Nós também realizamos testes com usuários para avaliar a usabilidade de nossa ferramenta. O escore da Escala de Usabilidade do Sistema foi de 85.8, sugerindo uma ferramenta de texturização altamente efetiva.

Palavras-chave: Computação gráfica, síntese de texturas, geração procedural de texturas, biologia matemática.

ABSTRACT

There are thousands of snake species in the world, many with intricate and distinct skin patterns. This diversity becomes a problem for users who need to create snake skin textures to apply on 3D models, as the difficulty for creating such complex patterns is considerable. We first propose a categorization of snake skin patterns considering their visual characteristics. We then present a procedural model capable of synthesizing a wide range of texture skin patterns from snakes. The model uses simple image processing (such as synthesizing spots and stripes) as well as cellular automata and noise generators to create realistic textures for use in a modern renderer. Our results show good visual similarity with real skin found in snakes. The resulting textures can be used not only for computer graphics texturing, but also in education about snakes and their visual characteristics. We have also performed a user study to assess the usability of our tool. The score from the System Usability Scale was 85.8, suggesting a highly effective texturing tool.

Keywords: Computer graphics. texture synthesis. procedural texture generation. mathematical biology.

LIST OF FIGURES

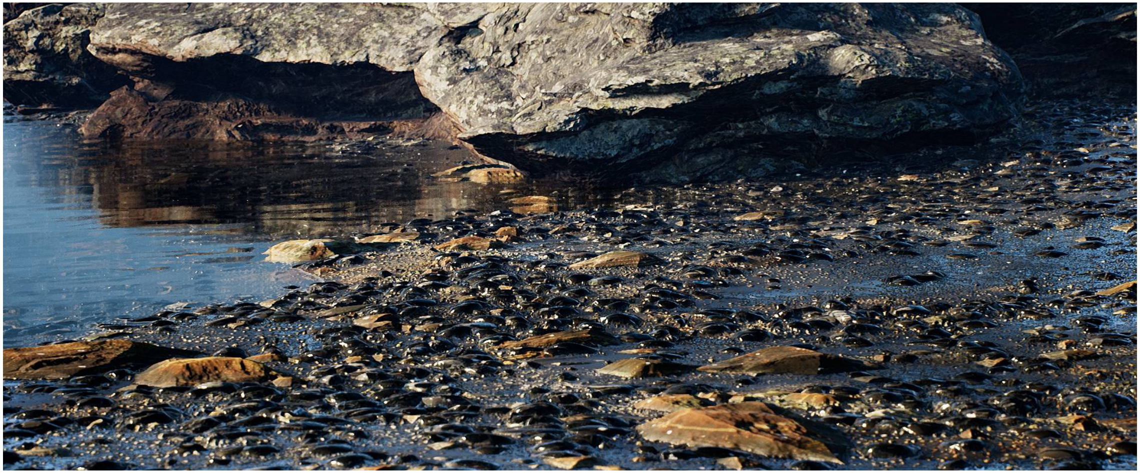
LIST OF TABLES

CONTENTS

1 INTRODUCTION

The quest for realistic and efficient computer rendering drives technical innovation ever forward. It is becoming increasingly difficult to determine whether an image is a photograph from a real location, or a screen shot from a game. Figure 1.1 shows a screen shot taken from a scene rendered in real-time using Unreal Engine 4, a modern game engine.

Figure 1.1 – Real-time render example



Source: kooooola (2015).

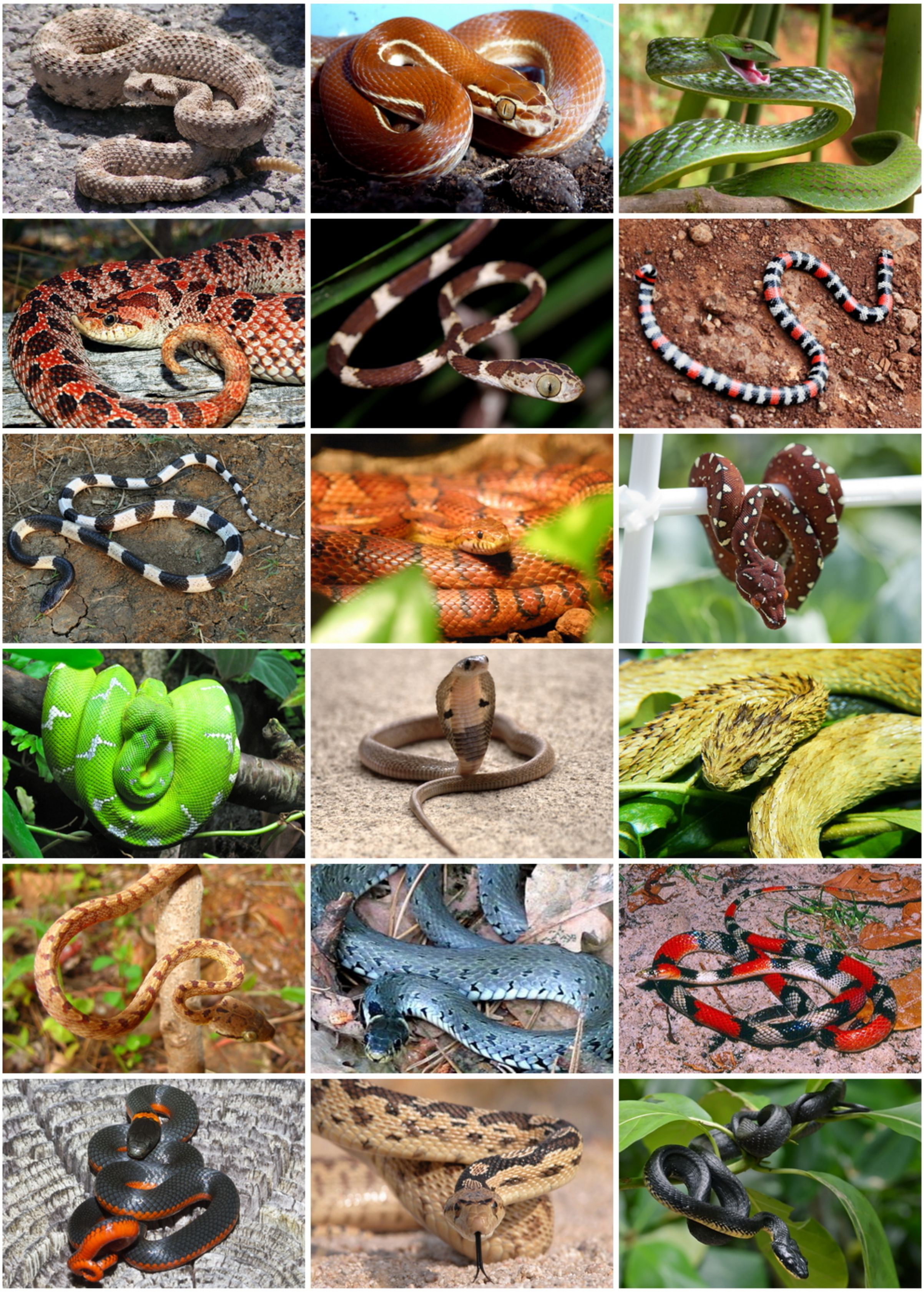
In order to increase visual realism, textures representing the material properties (which are applied to 3D models) require more and more details, which in turn, may take a long time for artists to create, increasing production costs. Thus, it may be impractical to add minute details to every 3D model in a virtual world.

Procedural texturing comes as a possible solution to these problems. Instead of relying on an artist to create textures and add fine details to it, a mathematical approach is taken, and a high resolution texture can be generated from an algorithm. (RHOADES et al., 1992) and (DEGUY, 2015) offer an in-depth discussion on the advantages of this technique.

Procedural texturing is also more flexible. The algorithm that generates textures will likely use random noise generators, such as Perlin Noise (PERLIN, 1985). By changing the random number generator seed, a different pattern may be synthesized, which still follows the same rules. As an example, we could generate infinite variations of zebra stripes patterns with the same algorithm. The standard approach of using a texture created by an artist would require manual labor for every variation. As well as variation, the procedural generator can take several parameters as inputs, enabling to easily modify, for instance, the colors or the size of a pattern.

For this study, we'll focus on snakes (reptiles of the Serpentes suborder), which are numerous in species and in skin pattern variation. Some have only a single plain color, while others have patterns composed of two or more colors, and the geometric pattern is sometimes difficult even to describe. They are present on every continent, except Antarctica and some islands (notably Ireland, Iceland, Greenland, Hawaii and New Zealand) (BAUCHOT, 2006). Some aquatic species live in the Indian and Pacific oceans as well. Figure 1.2 shows some snake species, highlighting their diversity.

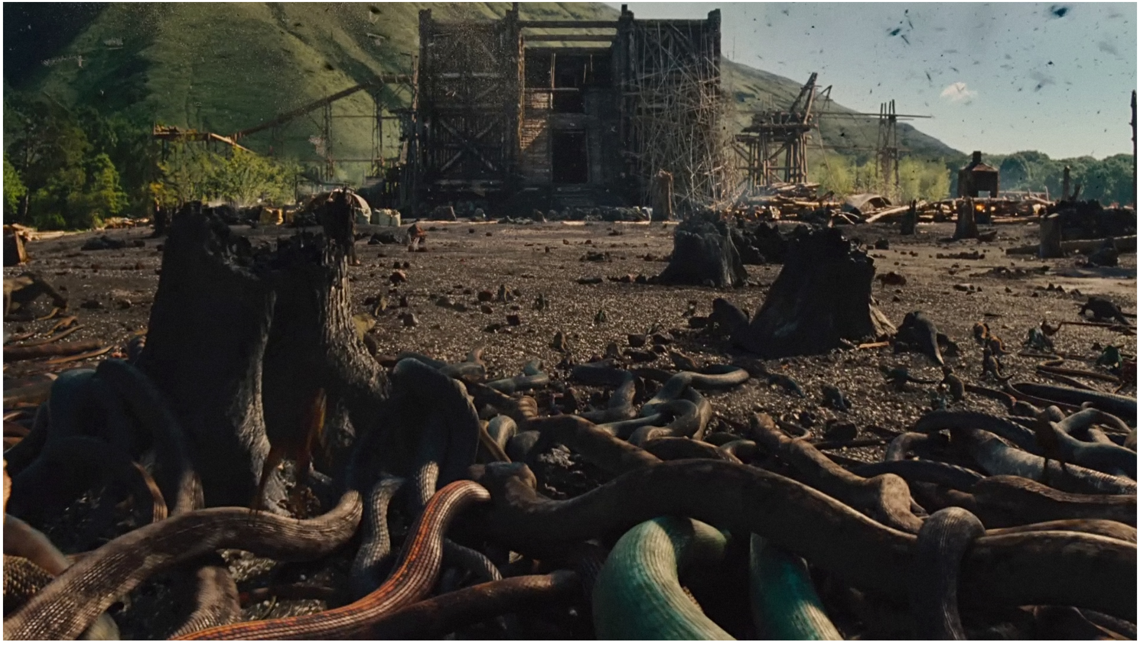
Figure 1.2 – Some snake species.



Source: Commons (2013).

Snakes are also present on various myths, such as the Medusa (greek myth), the Nagas (Indian religions), and the Jörmungandr (monster in Norse mythology). For more details, see (WIKIPEDIA, 2016). For these reasons, snakes are usually depicted in films, video games, or virtual reality simulations that take place in just about any non-urban setting (because snakes are so widespread), and on many mythological/fantasy settings. Figure 1.3 shows a still image taken from the *Noah* (2014) film, depicting computer-rendered snakes.

Figure 1.3 – Computer-rendered snakes in the *Noah* (2014) film.



Source: Regency Enterprises, Protozoa Pictures e Paramount Pictures (2014).

When we consider textures, a complicating factor comes into play: there are over 3600 snake species, according to The Reptile Database (UETZ; FREED; (EDS.), 2016), which means a large variety of colors and patterns. Creating such textures manually, from scratch, may prove to be a very time-consuming task. Extracting textures from photos will be difficult because unwrapped snake skin textures are not easily found, and collecting a specimen for photography could also be a complicating factor.

This work introduces a model to procedurally generate snake skin textures, with the main purpose of using them in computer graphics applications, such as animations and games, but can also be used in education and herpetology study. We also discuss the advantages of this technique over traditional texturing methods. We have implemented a tool for our model and we have also assessed the usability of this tool.

In the next chapter we discuss related work available in literature. In Chapter 3 we present a quick overview on snakes biology, as well as our categorization of snake skin patterns.

In Chapter 4 we present our procedural model, with results shown in Chapter 5. Chapter 6 discusses conclusions of this work, followed by the references and appendices.

Appendix A shows the tool's user test results, with 13 students as test subjects. Appendix B presents a short paper on visualization of snakes information, a work based on this one. Appendix C shows user test results with an herpetologist as test subject. Appendix D shows the parameters used to generate the textures shown in Chapter 5 (the Results). Appendix E shows the cellular automaton descriptor files for the automata generated in Section 4.2.4.

2 RELATED WORK

In this chapter we review the literature related to modeling the skin of snakes and procedural content generation, since our model is a procedural one.

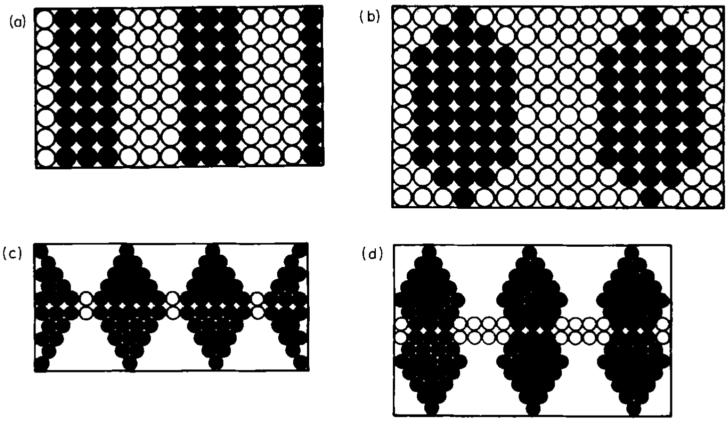
2.1 Snake Skin Patterns

We were unable to find any work in the computer graphics literature that focuses on snakes (or any reptilian) skin texture generation to apply on a 3D model and use on a modern renderer. However, there are some studies in the Mathematical Biology field that we can use in our study.

We'll need two types of generators, one for macro variation (the skin pattern) and other for micro variation (fine details such as surface irregularities). For the macro variation, we'll discuss the works of Murray (MURRAY; MYERSCOUGH, 1991) and Cocho (COCHO; PéREZ-PASCUAL; RIUS, 1987) (COCHO et al., 1987). These authors discuss how snake skin patterns can be generated using a mathematical model, which is precisely the basis for procedural texture generation.

Cocho uses cellular automata, a dynamical system with elements in a finite number of states in a discrete space and time (typically organized as a 2D/3D grid). Figure 2.1 shows some results obtained in Cocho's work. This method is further described in Section 4.2.4, and serves as the basis for the generation of some types of patterns in our method.

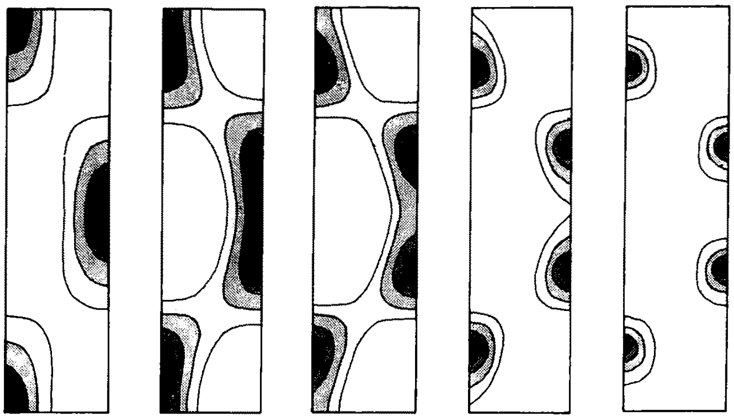
Figure 2.1 – Results obtained from cellular automata on the paper (COCHO et al., 1987).



Source: (COCHO et al., 1987).

Murray uses reaction-diffusion, a mathematical model first proposed by (TURING, 1952), which tries to explain several pattern formation mechanisms, in this case, to model the concentration of different pigments on animal skins, including snakes. The model can simulate how the chemicals (pigments) travel around an animal's body as it grows from the embryo stage to adulthood. Eventually, the system is stabilized (further iterations cause little to no changes), and the system's final state corresponds to the animal's skin pattern. Figure 2.2 shows an example result obtained by the method.

Figure 2.2 – Results obtained with a reaction-diffusion system from the paper (MURRAY; MYERSCOUGH, 1991).



Source: (MURRAY; MYERSCOUGH, 1991).

Both of these authors have managed to reproduce specific snake species' patterns. Cocho reproduced the *Bungarusfasciatus*, *Crotalus viridis*, *Crotalus basiliscus*, *Bothrops neuwiedi*, and the *Crotalus viridis lutosus*; while Murray shows results of the *Pseudonaja modesta*, *Lampropehis getulux catiforniae*, *Bitis atropos atropos*, and also mentions several other species throughout the paper.

One drawback of these models for our goals is that they can only generate textures that are visually too simple to use on many modern applications. For instance, typically only black-and-white images are synthesized, and in the case of Cocho's work, the resolution is extremely low (about a dozen pixels wide). Regardless, these works serve as a good basis for a more elaborate (detailed) texture generation method.

2.2 Procedural Generation

Procedural texture generation usage in games is seeing an increase in recent years (HENDRIKX et al., 2013), considering the advantages discussed in the introduction. The paper (HENDRIKX et al., 2013) defines several categories for procedural texture generators:

- Pseudo-random Number Generators (PRNG), which includes noise generators such as Perlin Noise (PERLIN, 1985);
- Image Filters (IF), which includes convolution filters;
- Spatial Algorithms (SA), which includes tiling and layering, grid subdivision and Voronoi diagrams;
- Modeling and Simulation of Complex Systems (CS), which includes cellular automata, tensor fields and other complex systems.

Each of these types of procedural generators can be used in combination. For example, the result from a complex simulation (CS) such as a cellular automaton can be combined with a spatial algorithm (SA) to tile it in a grid, then a blur image filter (IF) can be applied and the result mixed with a noise generator (PRNG) for extra details. Therefore, we'll take all these categories into consideration when modeling our own procedural model.

As Pseudo-random Number Generators, we'll need some noise generators. Some generators of this class include white noise, Perlin Noise (PERLIN, 1985) (and its improvement, (PERLIN, 2002)), Amortized Noise (PARBERRY, 2014) and Fractal Sum (MANDELBROT; PIGNONI, 1983). (LEWIS, 1989) also offers a review on some methods, and (EBERT et al., 2002) presents many techniques in greater detail.

Image Filters, such as blur, distortion and contrast adjustment are well established areas with known algorithms such as Gaussian blur. Spatial Algorithms such as tiling are usually easy and straightforward to implement. Possible choices for Modeling and Simulation of Complex Systems have been discussed in the previous section.

2.3 Other Works

Other works focus on snakes in computer graphics, but not specifically on texture generation. One of these focuses on snakes (and other materials) iridescence (DHILLON et al., 2014). Another two studies, (PANAGIOTAKIS; TZIRITAS, 2006) and (MILLER, 1988), focus on their movement patterns.

2.4 Discussion

In this chapter we presented existing approaches that deal with snakes skin pattern, however, none of them fulfil our purposes of generating realistic snake skin textures. We also discussed procedural texture generation as well as some random noise generators that will be necessary for our procedural model. In the next chapter we will discuss snakes biology, as well as our categorization of snake skin patterns.

3 SNAKES BIOLOGY

Before working on our model, we studied snakes biology to understand what kind of patterns were present, as well as other visual characteristics such as scales. This chapter presents an overview on snakes biology, and proposes a categorization of snake skin patterns.

3.1 Skin Pattern

Snakes are numerous in species and in skin pattern variation. Some have only a single plain color, while others have patterns composed of two or more colors. The pattern can be as simple as stripes or spots, or it can be so complex that it is even difficult to describe.

We are mostly interested in the snake's visual appearance, that is, pattern and colors. When taxonomists discover a new snake species, they describe, among other things, their appearance. However, there is not an international standard for categorization of snake skin pattern, such as, "if it looks like this then it should be categorized as Spots; if it looks like that then it should be categorized as Horizontal Stripes". Instead, taxonomists verbally describe the species' pattern. For example, in (NISTRI; LANZA, 2006, p.97), the species *Eryx somalicus* is described as follows:

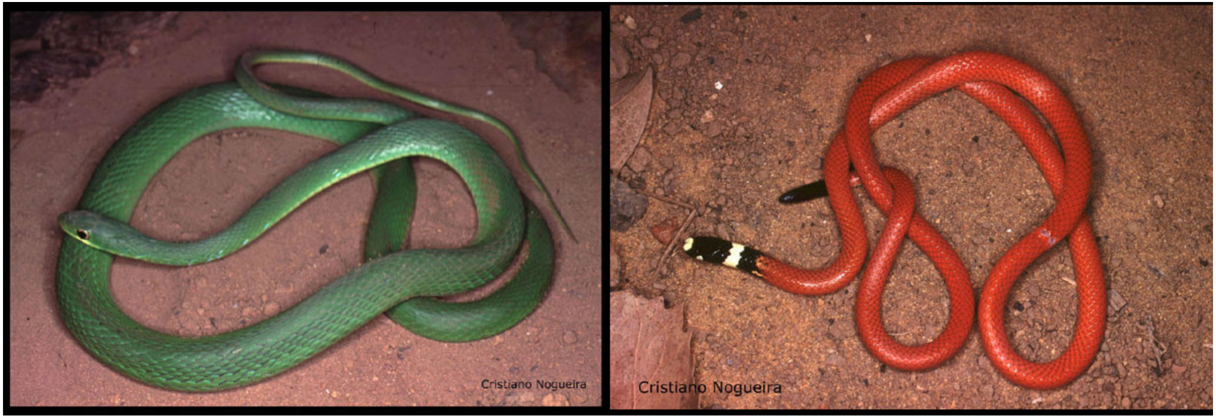
> "Pattern rather variable. Dorsal ground colour light to rather dark brown, crossed by about 30 off-white, sometimes dark-edged, transverse to more or less oblique stripes, each approximately 1-6 scales long; some of the light stripes may fuse with each other forming Y or X shaped marks. Furthermore the back is finely dark-striped longitudinally, since each dark scale is off-white medially. Lower parts of the flanks off-white with dark, irregular small spots and sometimes with a series of brownish black, larger roundish spots, each usually underlying a light dorsal stripe."

As we can see from the example description above, it would be hard to analyze such descriptions in order to assign them a single category such as "Spots" or "Rings". Therefore, the first step in our study was to create a categorization of snakes visual patterns, and assign each snake species to one of these categories by means of a visual assessment of snakes pictures. This will be useful to help determining the pattern synthesis method, and also what types of pattern are mostly present in Nature. Since it is very hard to find images of a snake's ventral side (the underside, or "belly"), and it is typically not visible, we decided to consider only the visible dorsal side ("back" and sides). Also, some species have a significantly different pigmentation color or pattern on their head or tail (such as the *Apostolepis assimilis*, which has no pattern on its body, but its head and tail are black; see Figure 3.1); again we did not take this into consideration. Both topics will be explored in future work.

With the help of an herpetologist, we defined the following six snake pattern categories. Please note that when we say "body", we mean only the dorsal side of a snake. We show two figures for each category, as illustrative examples.

1. **No Pattern**. The snake shows no pigmentation pattern, consisting only of a single solid color. Remember that we do not consider head or tail pigmentation, therefore the *Apostolepis assimilis* is categorized as having no pattern.

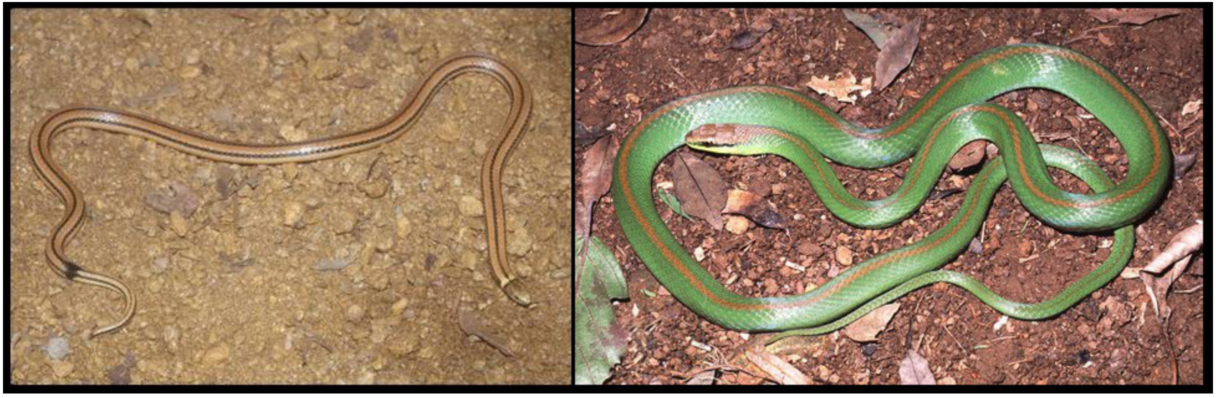
Figure 3.1 – No Pattern example snakes. Left: *Philodryas aestiva*. Right: *Apostolepis assimilis*.



Source: Left: (NOGUEIRA, n.d.b), right: (NOGUEIRA, n.d.a).

2. **Longitudinal Stripes**. The snake shows one or more stripes aligned along its body, that go from neck to tail or from head to tail.

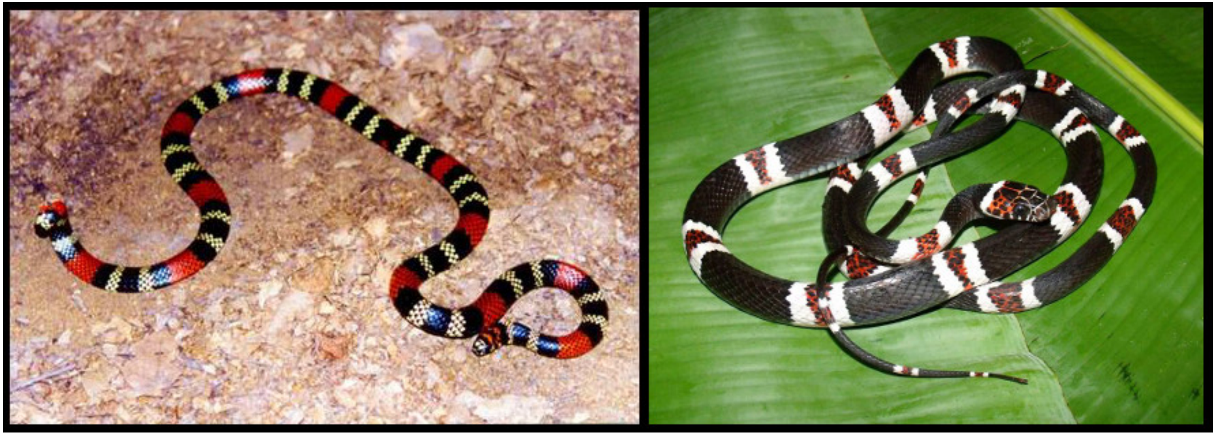
Figure 3.2 – Longitudinal Stripes pattern example snakes. Left: *Phalotris lemniscatus* (s). Right: *Philodryas olfersii*.



Source: Left: (BORGES-MARTINS, 2007b), right: (SAWAYA, n.d.).

3. **Transversal Stripes**. The snake shows many stripes aligned perpendicularly to its body axis, also known as "rings". These rings may be in two alternating colors, or may be in multiple colors. Typical examples include corals and false corals.

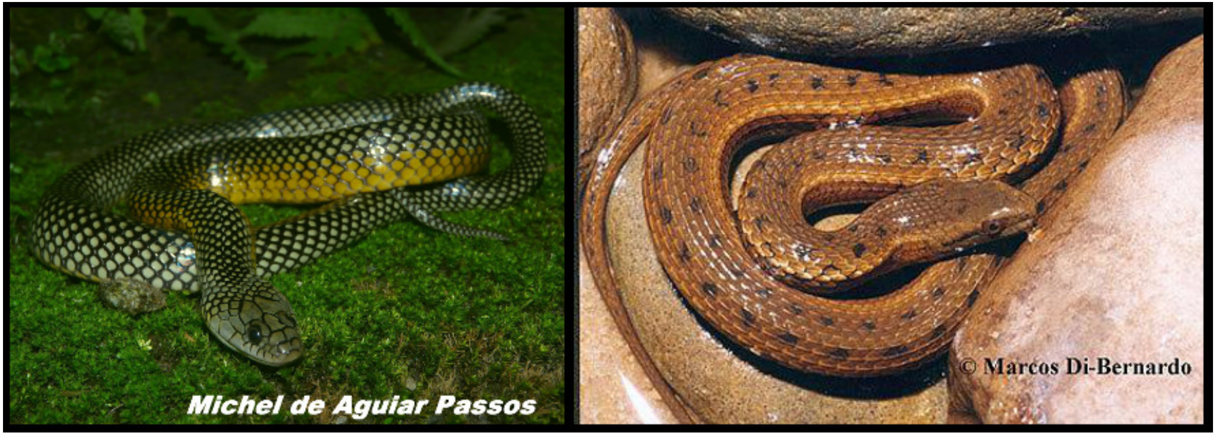
Figure 3.3 – Transversal Stripes pattern example snakes. Left: *Micrurus altirostris* (coral). Right: *Rhinobothryum lentiginosum* (false coral).



Source: Left: (BORGES-MARTINS, 2007a), right: (ALBUQUERQUE, n.d.).

4. **Spots**. The snake shows regular (similar) shapes on its body, often elliptical.

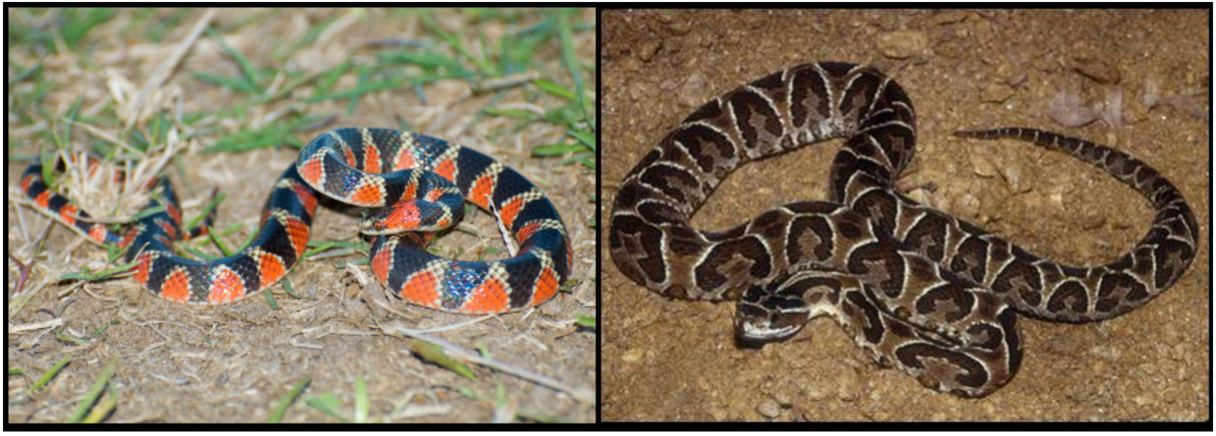
Figure 3.4 – Spots pattern example snakes. Left: *Liophis miliaris*. Right: *Ptychophis flavovirgatus*.



Source: Left: (PASSOS, 2013), right: (DI-BERNARDO, n.d.).

5. **Other Simple Pattern**. Few snakes show a pattern that is easily described and repeats along its body and do not fall on any of the categories above. Examples include the *Bothrops alternatus*, that has many C-shaped patterns along its body (see Figure 3.5).

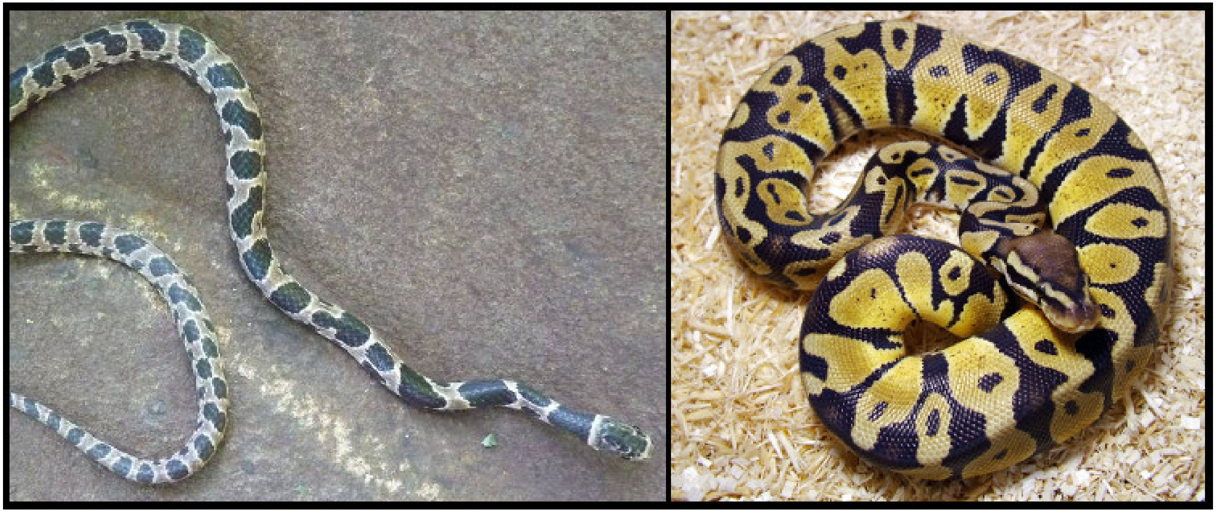
Figure 3.5 – Other Simple Pattern example snakes. Left: *Oxyrhopus rhombifer*. Right: *Bothrops alternatus*.



Source: Left: (TIMM, 2014), right: (BORGES-MARTINS, 2007c).

6. **Complex**. When the pattern is none of the above, it is categorized as "Complex". Typically this includes intricate (or abstract) camouflage patterns, that would be hard to describe verbally, or with numerous different elements on the snakes body.

Figure 3.6 – Complex pattern example snakes. Left: *Sibynomorphus turgidus*. Right: *Python regius*.



Source: Left: (MAY, 2010), right: (CAMPBELL, 2005).

Our goal is to assign each species to one category, so that we can determine the relative proportions between different categories. This was done by means of a visual assessment of each species' photos.

3.2 Scope Limitation

As mentioned in the introduction, there are over 3600 species of snakes, and studying all these species would prove to be tremendous work. Scope had to be limited somehow. We have at Universidade Federal do Rio Grande do Sul experts on snakes, particularly for the species present in the state of Rio Grande do Sul, Brazil. In this state, there are dozens of snake species, at least 81 of which have been identified and cataloged in the book by Abegg and Neto (ABEGG; NETO, 2012). In this book, these 81 species have been described as accurately as possible, however, in text format (instead of a well structured data table). Using the information available in the book, as well as available local expertise, we decided to limit the scope geographically, to the Rio Grande do Sul state.

It is important to note that although we limited the scope to the snakes in this specific region, all snake families are present in it, with the exception of Pythonidae. Therefore, we deemed our subset as a good representation of the whole and use the book as a guide.

3.3 Skin Pattern Analysis Results

In the studied region the most expressive family is *Dipsadidae*, with 59 species (72.8%), followed by *Viperidae* (7 species, 8.6%), *Colubridae* (5 species, 6.1%) and *Elapidae* (5 species, 6.1%). After analyzing pictures of these 81 species, we categorized them as the following pattern types (as discussed on section 3.1):

Table 3.1 – Pattern types distribution.

Pattern	Count	% of Total
No Pattern	19	23.46%
Longitudinal Stripes	19	23.46%
Transversal Stripes	7	8.64%
Spots	8	9.88%
Other Simple Pattern	9	11.11%
Complex Pattern	19	23.46%

From this analysis, we have learned that 76.54% of snake species from the sample region have some type of skin pattern that could benefit from the creation of a procedural model. We also did a visual assessment on a sample of 18 of the 40 snakes from the Pythonidae family (which is not present in the studied region), chosen randomly. 14 of them (77.8%) have complex pattern, 3 (16.7%) have other simple pattern, and 1 (5.5%) has no pattern. This doesn't changes the fact that most snakes have some type of skin pattern.

While working on this analysis, we also collected data on other snake characteristics, such as what they feed on, activity time, habitat, teeth type, and more. With the resulting data table, we created a linked-view visualization and used it to find correlations between certain characteristics. This represents a contribution for both the Information Visualization and Herpetology fields, as new knowledge regarding snakes behavior was revealed. The paper will be published on the information visualization workshop at SIBGRAPI '17 and can be seen on the Appendix B.

3.4 Scales

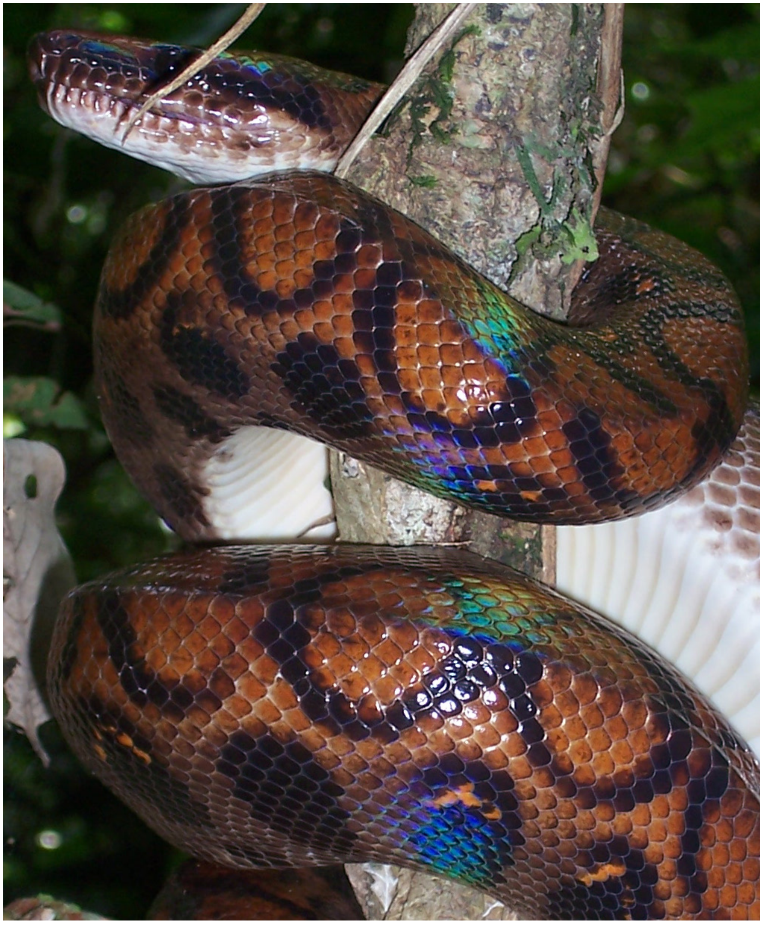
Snakes are reptiles that exhibit scales throughout their entire body. Besides the skin pattern, for completeness our model will also address the modeling of scales. Snakes have scales in many shapes (BAUCHOT, 2006), but they are generally diamond or leaf-shaped. Furthermore, scales may or may not have a longitudinal keel on it. As such, scales are divided into two classes:

- Smooth. The surface is approximately even, or smooth. Many species with smooth scales also exhibit iridescence when lit with a bright light. Figure 3.7 shows an example.

- Keeled. There is a longitudinal keel, or ridge, along the scale. The scale is quite rugged as well. This causes incident light rays to diffuse more throughout the surface, which helps with camouflage. Snakes with keeled scales do not exhibit iridescence, as the surface roughness is fairly high. Figure 3.8 shows an example.

One problem with scales, though, is that their quantity varies from species to species. This is not an issue for the body texture, as it is easy to configure how many scales should be generated vertically and horizontally. However, each species has specific scales layout on their head, sometimes even with specific shapes, such as the “eyelashes” on the eyelash viper (*Bothriechis schlegelii*) (see Figure 3.9). Therefore, this work will focus only on generating textures to apply on a snake's body, not on their head or tail. These topics will be explored in a future work.

Figure 3.7 – Rainbow boa (*Epicrates cenchria*), showcasing smooth scales with iridescence.

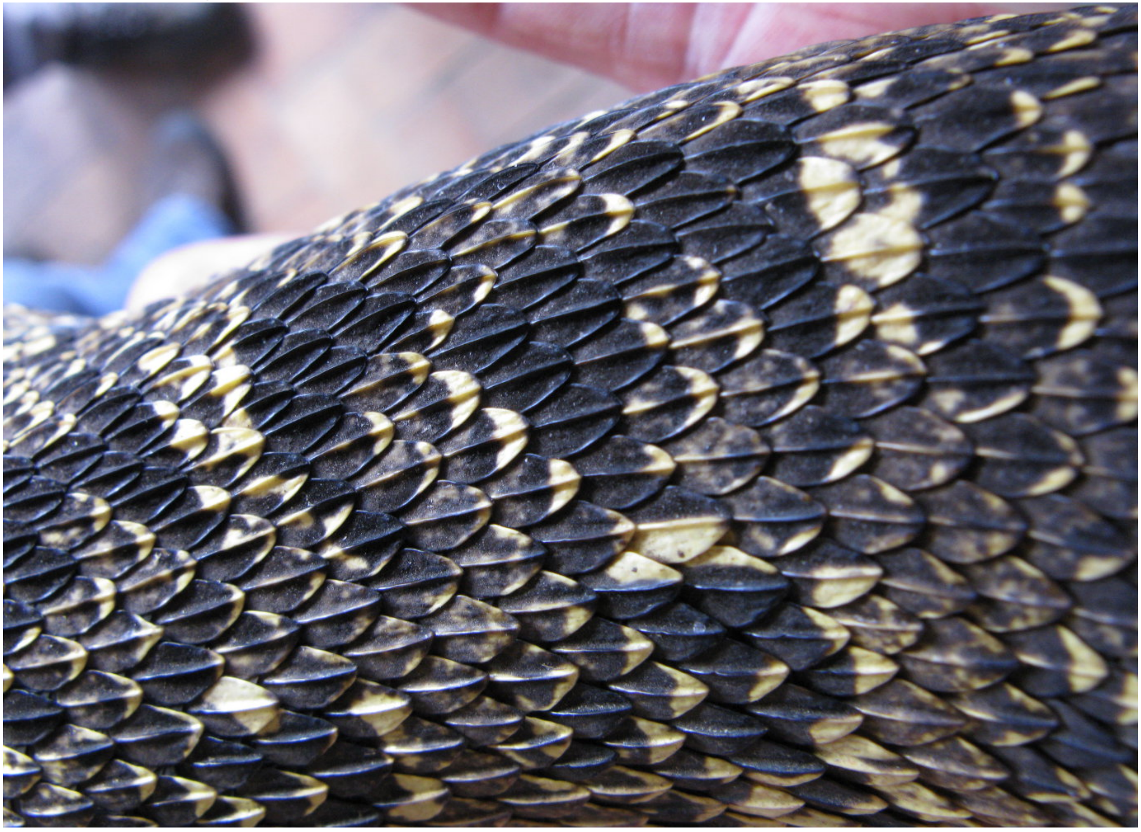


Source: Commons (2009).

3.5 Discussion

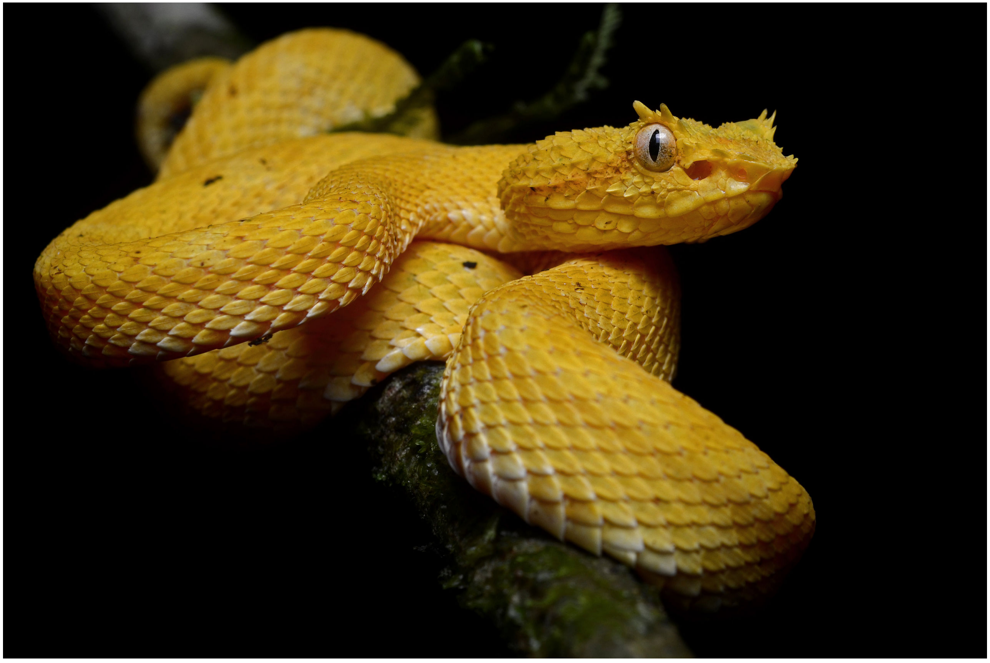
In this chapter, we estimated that 76.54% of snake species would benefit from a texture synthesizer, proposed a categorization according to their skin pattern, and studied snake scales shape. Knowing these characteristics will be useful for implementing the generator, which will be discussed in the next chapter.

Figure 3.8 – Puff Adder (*Bitis arietans*), showcasing keeled scales.



Source: Commons (2012).

Figure 3.9 – Eyelash Viper (*Bothriechis schlegelii*).



Source: Commons (2011).

4 THE PROCEDURAL MODEL

In this chapter, we will discuss our implementation of the texture generator. First, we present a general overview of the model followed by specifics of each pattern type (see Section 3.1 for details on the pattern types), a pattern outliner, and scales generation.

The goal of our procedural model is to generate high resolution textures (up to 4096x4096 pixels) for use in a modern renderer. As such, it should generate at least a color map and a height map. We'll also generate a roughness map, which describes how rough or glossy the surface looks, and is explained further in Section 4.4. The generated textures will be applied to a previously-made 3D mesh and should yield a realistic result. The mesh's UV coordinates should be mapped with the seam at the center of the snake's underside (i.e. the edges of the textures will meet there). Also, since a snake's body length is much longer than its circumference, the UV map should tile several times. It should also be very fast, outputting the resulting textures in less than half a second, allowing for an iterative texture creation process.

4.1 General Overview

Each of the textures are generated sequentially in several steps. The color, height and roughness maps generation are independent and can be parallelized. It should also be noted that pattern generation (which goes to the color map) is independent from scales generation (which goes to the height map).

The color map generation can be summarized as follows:

1. A background color is chosen by the user. This is the color that forms most of the snake's skin color;
2. The user selects a pattern type, and the pattern texture is generated (see Sections 4.2.1 through 4.2.5);
   (a) Optionally, the pattern is outlined (see Section 4.2.6);
   (b) A post-process distortion effect is applied to make the texture look more natural (see Section 4.2.7);
3. The resulting image from Step 2 is drawn over the background color from Step 1.

The height map (scales) generation can be summarized as follows:

1. A single scale is generated (see Section 4.3.1);
2. It is replicated across the texture (see Section 4.3.2);
3. It is cut and stretched on the horizontal corners to form the ventral (underside) scales (see Section 4.3.3).

The roughness map generation is done in one simple step which is explained further on Section 4.4.

Throughout the explanation of each of the textures' generation, we refer to texture size sometimes. All texture sizes are normalized in the [0.0;1.0] range, with [0.0;0.0] being the top-left corner, and [1.0;1.0] the bottom-right corner. The texture generator can synthesize any texture size, as long as there is processing power.

Regarding parameters, some have minimal and maximal values of zero to one because they represent percentages; range for other values (such as blur and distortion intensities) have been defined empirically, by trying different values for the parameters and finding a good value range. It is possible to extrapolate the defined range, however, the result may be inadequate. For example, increasing too much the size of spots can cause them to be so large that they intersect each other. We now proceed to explain how each texture (color, height and roughness) is synthesized.

4.2 Color Map Generation

As explained in the overview, the final texture's base color begins with a single solid color, which forms the "background" of the texture. This is the snake's predominant skin color. The next step is the pattern generation (see Section 3.1 for details on each pattern type). We now proceed to discuss how the pattern colors are generated, for each pattern type, except "No Pattern".

4.2.1 Transversal Stripes Pattern (Rings)

Our model supports up to 4 distinct stripes colors and sizes. For each color, we can define where it starts and ends using a two-dimensional vector parameter, with values ranging from 0.0 to 1.0. For example, choosing [0.2;0.4] for a given color means it would begin to be drawn at 20% of the texture's height up to 40% of its height.

The available input parameters for this pattern type are:

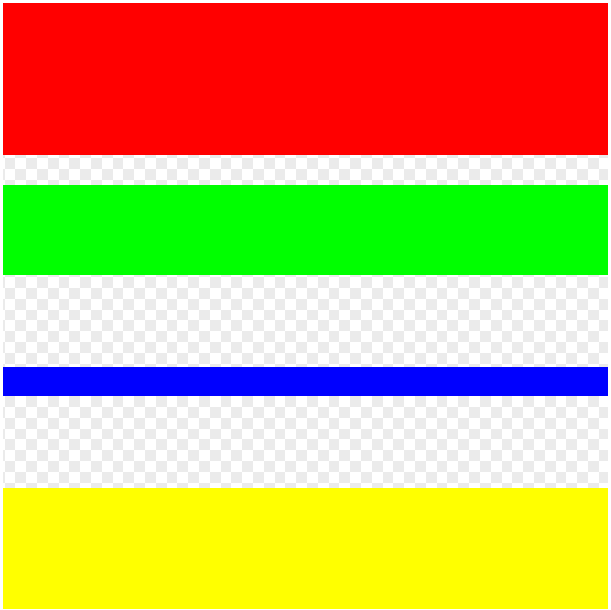
- **Transversal_Color**: array of 4 colors, one for each stripe;
- **Transversal_BeginEnd**: array of 4 vectors of 2 floats each, with values ranging from 0.0 to 1.0.

For instance, if the parameters are those in Table 4.1, the resulting image would be as seen in Figure 4.1. Small gray-and-white squares mean transparency, which will be filled by the background color.

Table 4.1 – Parameters used for example Transversal pattern generation seen in Figure 4.1.

Parameter	**Value**
Transversal_Color1	Red
Transversal_Color2	Green
Transversal_Color3	Blue
Transversal_Color4	Yellow
Transversal_BeginEnd1	[0.0;0.25]
Transversal_BeginEnd2	[0.3;0.45]
Transversal_BeginEnd3	[0.6;0.65]
Transversal_BeginEnd4	[0.8;1.0]

Figure 4.1 – Example of generated Transversal Stripes Pattern.



Source: The author.

We do not know of any snake species that has more than 5 colors in its pattern (the background plus the 4 stripes color), so we did not deem necessary to allow more colors on the transversal pattern, though that could be easily implemented. Some of the most colorful snake are the corals, with 3 colors, typically red, black and yellow.

By setting a color's alpha channel to 0 (transparent), or setting the same value on the begin/end vectors, we can omit one color, in case all 4 aren't needed. Figure 4.2 shows an example where the fourth color is omitted by setting its begin/end vector to [1.0;1.0]. The parameters can be seen in Table 4.2. Notice that the first color is drawn first, so it is occluded by colors 2 and 3; and also that color 4 (blue) won't be drawn, because it begins and ends in the same position.

Table 4.2 – Parameters used for second example Transversal pattern generation seen in Figure 4.2.

Parameter	**Value**
Transversal_Color1	Black
Transversal_Color2	Desaturated yellow
Transversal_Color3	Desaturated yellow
Transversal_Color4	Blue
Transversal_BeginEnd1	[0.1;0.9]
Transversal_BeginEnd2	[0.3;0.4]
Transversal_BeginEnd3	[0.6;0.7]
Transversal_BeginEnd4	[1.0;1.0]

By using a red background color, the result would be that of the coral snake *Micrurus altirostris*, as seen in Figure 5.3.

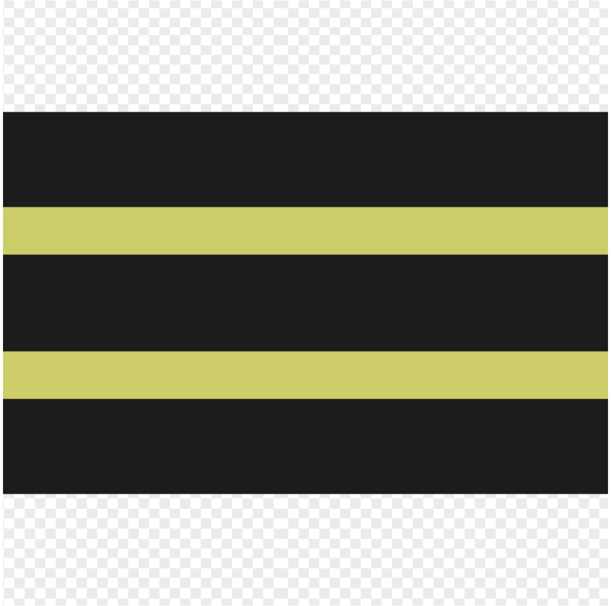
4.2.2 Longitudinal Stripes Pattern

The longitudinal pattern implementation can be described by the following steps.

1. A single white stripe is generated on a black background. These colors are used because blending them will be easier later on;

2. The result from step 1 is optionally repeated (tiled) to form multiple stripes (stripes width will be decreased when tiling 2 or more times);

Figure 4.2 – Second example of generated Transversal Stripes Pattern with the fourth color omitted.



Source: The author.

3. The result from step 2 is optionally squeezed to make the stripes closer to the snake's dorsal (visible) side, as present in certain species (width will be decreased here as well);

4. Stripe and background colors are blended according to user's choice.

Figure 4.3 shows an example of each step's result. The parameters used can be seen in Table 4.3.

The available input parameters for this pattern type are:

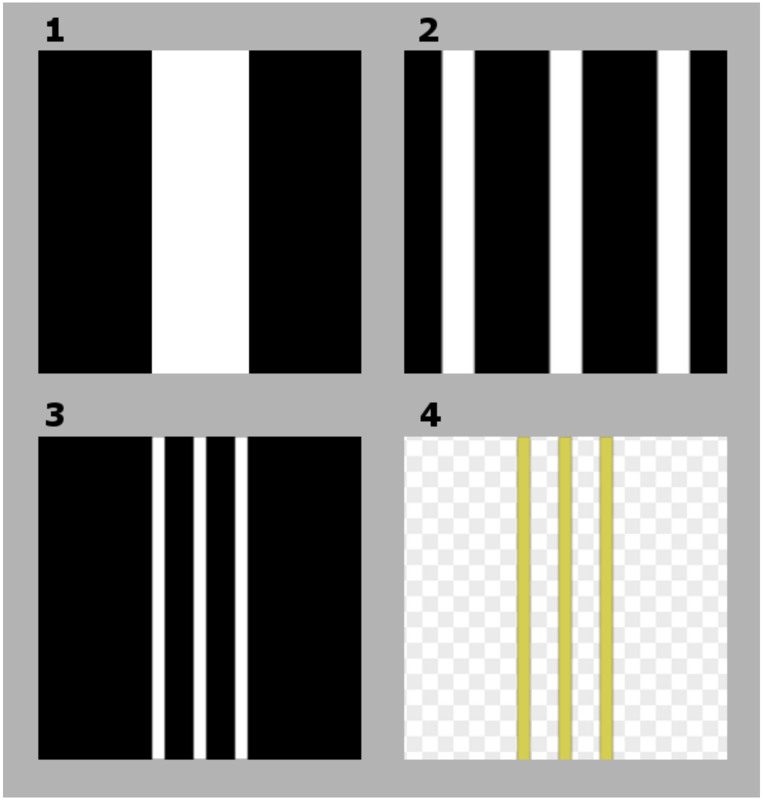
- **Longitudinal_NumberOfStripes**: integer that defines how many stripes will be generated. Suggested range: 1 to 10;

- **Longitudinal_StripesWidth**: float ranging from 0.0 to 1.0 that defines the stripe's width. 0.1 for example means that the stripe occupies 10% of the texture's width, divided by the number of stripes;

- **Longitudinal_StripesColor**: color of the stripes;

- **Longitudinal_Pinch**: float that squeezes the textures closer to the center (dorsal side). At 1.0 there is no change, at 2.0 the texture's width is reduced by half, at 4.0 it is reduced by 1/4th, and so on. Suggested range: 0.0 to 10.0.

Figure 4.3 – Example of generated Longitudinal Stripes Pattern.



Source: The author.

Table 4.3 – Parameters used for example Longitudinal pattern generation.

Parameter	Value
Longitudinal_NumberOfStripes	3
Longitudinal_StripesWidth	0.3
Longitudinal_StripesColor	Yellow
Longitudinal_Pinch	2.6

4.2.3 Spots Pattern

The spots pattern is generated by the following steps:

1. A single circumference is synthesized. It is grayscale with intensity ranging linearly from 1.0 (white) at the center and 0.0 (black) at the edges;

2. Midtones are moved in the image histogram (same as the Levels operation, present in most image manipulation software such as Photoshop) so that the image turns more opaque;

3. The spots' color is applied;

4. It is tiled in a 10x10 grid, and other parameters are then applied to control the various characteristics of the pattern, such as tiling, random position variation, size and random size variation. Here we use the function “Tile Generator” from the software Substance Designer.

The available input parameters for this pattern type are:

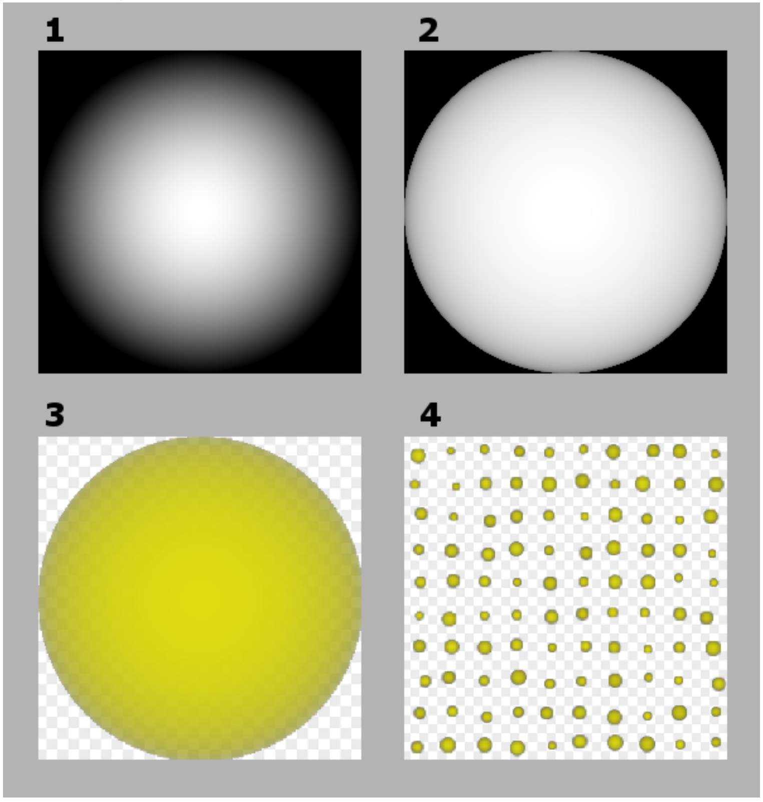
- **Spots_Scale**: float that determines each spot’s overall size. Suggested range: 0.0 to 2.0;
- **Spots_ScaleVariation**: float ranging from 0.0 to 1.0. Adds a random variation to each spot’s size. 0.0 means no scale variation, 0.5 means the size variation ranges randomly from 50% to 100% of the spot’s size;
- **Spots_PositionRandom**: vector of 2 floats ranging from 0.0 to 1.0. Adds a random position on X and Y for each spot. 0.0 means no random displacement and 1.0 means the displacement can invade the next spot’s position;
- **Spots_Offset**: float ranging from 0.0 to 0.5. Offsets all rows after the first. At 0.0 all spots are aligned as a grid; at 0.5 the spots are arranged as a grid rotated 45 degrees;
- **Spots_Color**: color of the spots;
- **Spots_Interstice**: vector of 2 floats ranging from 0.0 to 1.0. Controls each spot’s width and height respectively. 0.0 means the spots will be round and 0.5 means the spots’ width or height will be half of its original radius. Useful to create elliptical spots;
- **Spots_IntersticeRandom**: vector of 2 floats ranging from 0.0 to 1.0. Adds a random variation to each spot’s width and height respectively. 0.0 means no variation and 1.0 means total variation;
- **Spots_Rotation**: float ranging from 0.0 to 360.0. Rotates all spots a number of degrees;
- **Spots_RotationRandom**: float ranging from 0.0 to 180.0. Adds a random variation to each spot’s rotation. 30.0 for example would randomly rotate the spots up to 30 degrees to either side;
- **Spots_Tiling**: At 1.0 there are 10x10 spots in the texture; decreasing this float value will “zoom into” the texture (so that fewer spots are visible) while increasing it will make the texture repeat (so that more spots are visible). Suggested range: 0.1 to 5.0.

Figure 4.4 shows an example of the generation process. The parameters used can be seen in Table 4.4.

Table 4.4 – Parameters used for example Spots pattern generation.

Parameter	Value
Spots_Scale	0.5
Spots_ScaleVariation	0.5
Spots_PositionRandom	[0.2;0.2]
Spots_Offset	0.0
Spots_Color	Yellow
Spots_Interstice	[0.0;0.0]
Spots_IntersticeRandom	[0.0;0.0]
Spots_Rotation	0.0
Spots_RotationRandom	0.0
Spots_Tiling	1.0

Figure 4.4 – Example of generated Spots Pattern.



Source: The author.

Notice that the resulting texture in Figure 4.4 has spots that are slightly different in size (because of the value used in the parameter **Spots_ScaleVariation**), and not precisely aligned in the grid (because of the value in **Spots_PositionRandom**).

It is possible to generate interleaved spots by setting the parameter **Spots_Offset** to 0.5, and elliptical shapes by setting the parameter **Spots_Interstice**. Figure 4.5 shows an example using the parameters seen in Table 4.5. It also has a value of 10.0 in **Spots_RotationRandom**, so the ellipses are randomly rotated up to 10 degrees in either direction.

Table 4.5 – Parameters used for second example Spots pattern generation.

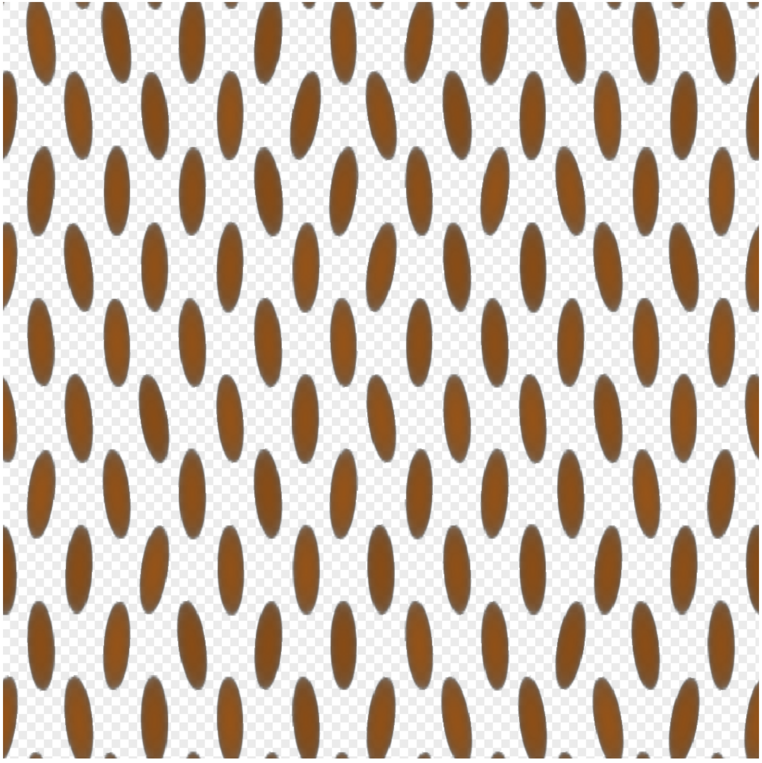
Parameter	**Value**
Spots_Scale	1.2
Spots_ScaleVariation	0.02
Spots_PositionRandom	[0.01;0.01]
Spots_Offset	0.5
Spots_Color	Dark orange
Spots_Interstice	[0.7;0.0]
Spots_IntersticeRandom	[0.0;0.0]
Spots_Rotation	0.0
Spots_RotationRandom	10.0
Spots_Tiling	1.0

4.2.4 Other Simple Pattern

For the simple pattern group, which includes repeating geometrical patterns, we took inspiration from the works of Cocho (COCHO; PéREZ-PASCUAL; RIUS, 1987) (COCHO et al., 1987), who used cellular automata to create simple patterns, including snake skin.

As in (COCHO et al., 1987), we have two types of cellular automata: rectangular and triangular, and both work similarly. The automaton is initialized with a single row of bits, usually 10 to 25 bits, depending on the desired pattern size. This will be the texture's first row: each bit is a pixel, with zeros being white, and ones being black. The more bits the initial row has, the more complex the pattern is. Values can be repeated across the initial row as well as a means of tiling the texture. Then, for a predefined number of loops (called "Epochs"), the next rows are calculated as follows.

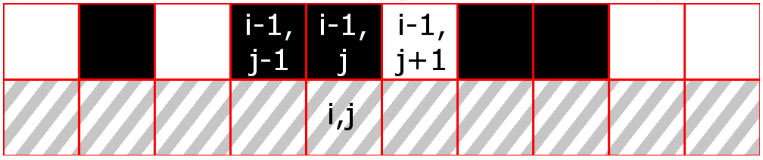
Figure 4.5 – Second example of generated Spots Pattern.



Source: The author.

For rectangular automata, there is a transition rule that is defined by four bits, as in Cocho's work. Assume we are calculating the value for pixel [i,j] in the next row. The algorithm takes the previous row's pixel value, [i-1,j], and sums it with its adjacencies, [i-1,j-1] and [i-1,j+1] (see Figure 4.6). If the result of this sum is 0, the last (rightmost) bit of the transition rule is the new pixel value. If the sum is 1, the 3rd bit is used. If the sum is 2, the 2nd bit is used, and if the sum is 3, the first bit is used. In case there are no adjacent cells (for calculating the left-most or right-most cells), the value from the other end is used (i.e. wrap around).

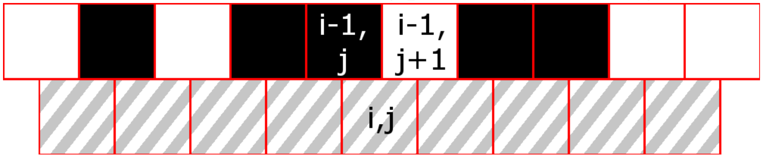
Figure 4.6 – Values used for calculating position i,j in a rectangular automaton.



Source: The author.

The triangular automaton works similarly, however, each odd row is displaced 50% of each element's diameter to the right, and only the two cells above it are used for the sum (see Figure 4.7). Since the maximum sum is 2, the transition rule has only 3 bits (for sums of 0, 1, and 2), following the same logic as above.

Figure 4.7 – Values used for calculating position i,j in a triangular automaton.



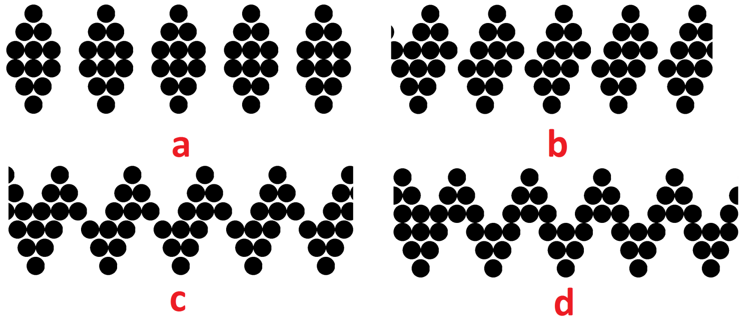
Source: The author.

Once all rows have been calculated (given by the number of epochs), an image is synthesized using the resulting bit matrix. We paint white circles for zeros and black circles for ones. The diameter of these circles is adjustable (50 pixels by default). Afterwards, the resulting image is duplicated above the original and mirrored vertically, to create a symmetrical texture. This mirror duplicate can also be displaced horizontally, to generate interleaving patterns. Figure 4.8 shows an example of the same automaton, with varying values for this mirror's displacement.

We developed a program to generate these patterns. The user defines the cellular automaton parameters in text format, then the program parses this file, calculates the results, and saves the output image in tga format. The program also supports changing the transition rule after a predefined number of loops. The parameters are the following:

- **Initial values**: an array of bits (of any size) that defines the cellular automaton's first row;
- **Automaton type**: can be either rectangular or triangular;
- **Shift Mirror**: integer that controls how much the mirrored image should be displaced. Each increment shifts the mirror image by 1/3 of the circle's diameter. Figure 4.8 shows examples for some values;
- **Epochs**: the number of loops the automaton will calculate (i.e. the number of rows, before duplicating and mirroring);
- **Transition rules**: An array of 3 bits (for triangular automata) or 4 bits (for rectangular automata), followed by an integer that specifies for how many epochs this rule should be applied ('x' for all remaining epochs). Afterwards, the parser expects another transition rule on the next line, containing the transition rule that will be applied after the previous rule was used for its preset number of epochs.

Figure 4.8 – Varying values for the mirror image's displacement. Values for parameter **Shift Mirror**: (a) = 0, (b) = 1, (c) = 2, (d) = 3.



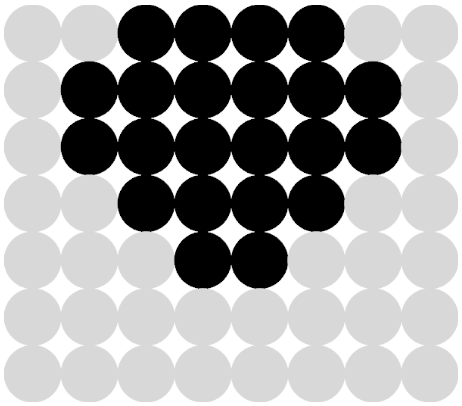
Source: The author.

As an example, using the following input file:

```
0 0 1 1 1 1 0 0 //Initial values
rectangular //Automaton type
0 //Shift mirror
6 //Epochs
1110 1 //First transition rule for 1 epoch
1100 1 //Second transition rule for 1 epoch
1000 x //Third transition rule for remaining epochs
```

will produce the following image (Figure 4.9), before duplicating and mirroring:

Figure 4.9 – Example cellular automaton result. White circles were replaced with gray circles for better visualization.

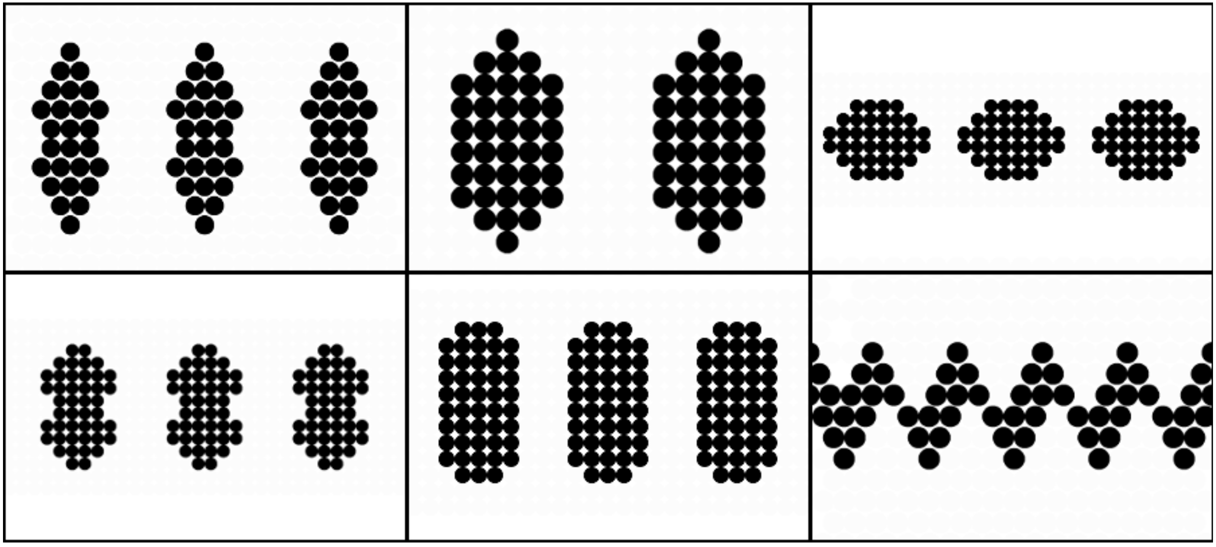


Source: The author.

The first row is simply the initial values input. The second row used the 1110 transition rule, meaning the result will only be 0 if the element above is 0 and its two adjacencies are 0 as well. The third row uses the 1100 rule, which basically duplicates the values. Remaining rows use the 1000 rule, which only results in 1 if the element above and its two adjacencies are all at 1.

Figure 4.10 shows some examples of generated images using this algorithm. All these images represent a specific snake species' pattern. Although simple, cellular automata can generate many patterns visually similar to real snakes. The first two were generated in Cocho's paper; the others were generated using automata of our own design. Each automaton's descriptor file can be seen in Appendix E.

Figure 4.10 – Sample results obtained using cellular automata. From top-left to bottom-right: *Bothrops neuwiedii*, *Crotalus viridis*, *Daboia russelii*, *Eunectes murinus*, *Python molurus* and *Vipera berus*.



Source: The author.

After generating the image, additional steps are taken to remove the circles pattern. The image is blurred and then its contrast is increased. See Figure 4.11 for an example. We also added the option to rotate the pattern by 90 degrees, because some automaton results are best used longitudinally, not transversally.

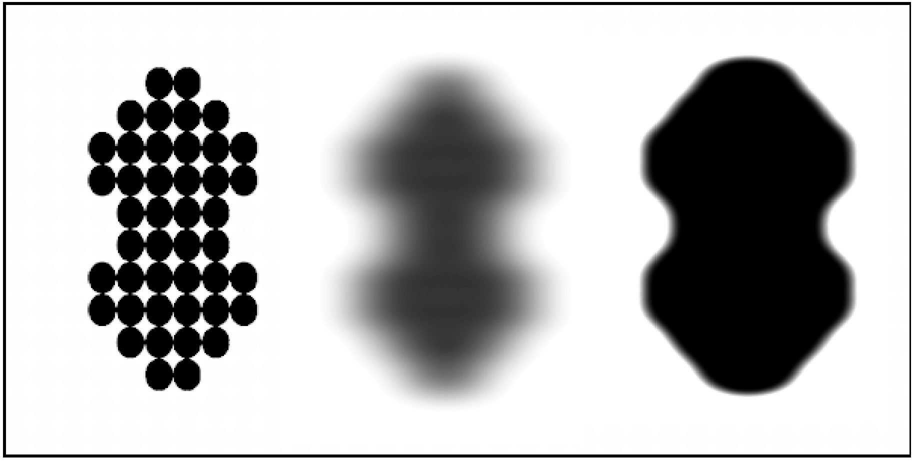
These parameters are:

1. **Pattern_RotatePattern90Degrees**: boolean that controls whether the pattern should be rotated 90 degrees;

2. **Pattern_BlurAmount**: float that controls the intensity of the blur applied on the whole pattern. Suggested value range: 0.0 to 16.0;

3. **Pattern_Contrast**: float ranging from 0.0 to 1.0 that controls the intensity of the contrast applied on the whole pattern.

Figure 4.11 – Cellular automaton's resulting image post process. Left: the generated image. Middle: blurred image. Right: contrast increased.



4.2.5 Complex Pattern

Snakes categorized as having *Complex* pattern present spots of different shapes and sizes, often forming a camouflage pattern as seen in many military vehicles. This pattern's form is apparently random, being hard to even describe it.

We believe that Murray's reaction-diffusion systems would perform well to generate these patterns. Indeed, complex snake skin patterns have been generated, as discussed in his paper (MURRAY; MYERSCOUGH, 1991). However, the computational cost of running a reaction-diffusion simulation is high compared to our other methods and therefore Murray's method was not implemented and is left as a future improvement.

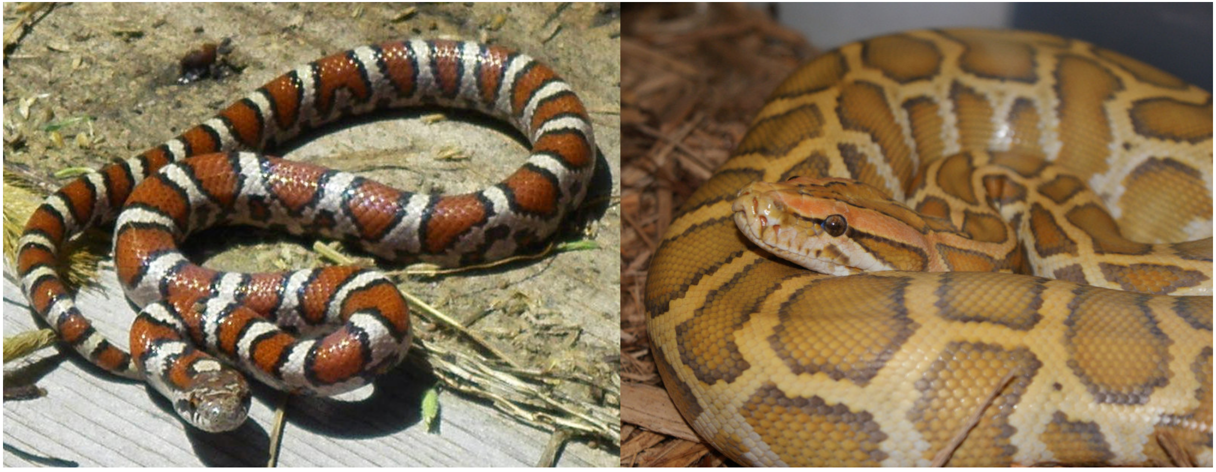
4.2.6 Pattern Outline

Some snakes, such as the eastern milk snake (*Lampropeltis triangulum triangulum*) and the burmese python (*Python bivittatus*) also present a strong color contrast on the borders of their pattern (see Figure 4.12). For this reason, we implemented a pattern outliner.

The outline is created by the following steps:

1. Perform an edge detection on the pattern image;
2. Distort it using the height map from the scales (scales generation is explained on section 4.3). This is done so that the outline doesn't matches perfectly the contour of the pattern, but is adjustable;

Figure 4.12 – Left: *Lampropeltis triangulum triangulum*. Right: *Python bivittatus*.



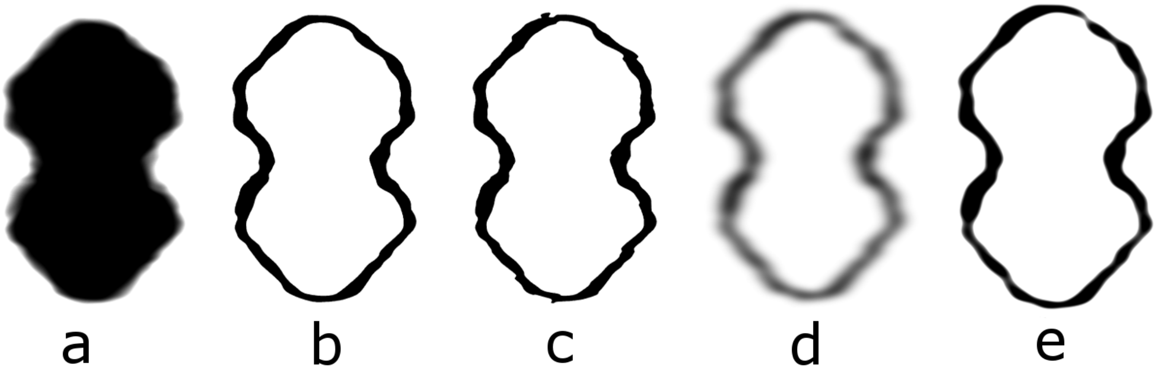
Source: Left: (RADA, 2008), right: (Andrews II, 2010).

3. A blur is applied, and contrast is increased. This is to remove some of the irregularities introduced in step 2. Both the blur and the contrast intensities are adjustable.

The edge detection algorithm used is the one implemented in the software Substance Designer. It performs a blur on the input image, subtracts this blurred image from the input image, and adjusts the levels of the image histogram to increase contrast.

Figure 4.13 shows an example of the outline being created step by step.

Figure 4.13 – Outliner steps. (a) Input image. (b) Edge detection. (c) Distortion. (d) Blur. (e) Contrast.



Source: The author.

The outliner's parameters are:

- **Has_Outline**: boolean. Whether the snake has a pattern outline or not. The next parameters are unused if this is false;
- **Outline_Color**: color of the outline;

- **Outline_Edge_Width**: float that controls the thickness of the outline. Suggested value range: 0.0 to 16.0;

- **Outline_Blur**: float that controls the intensity of the blur. Suggested value range: 0.0 to 16.0.

- **Outline_Edge_Roundness**: Makes hard edges more rounded. It is a float value with suggested value range of 0.0 to 16.0;

- **Outline_Distortion_Intensity**: float that controls the distortion effect applied on the outline. Suggested value range: 0.0 to 10.0.

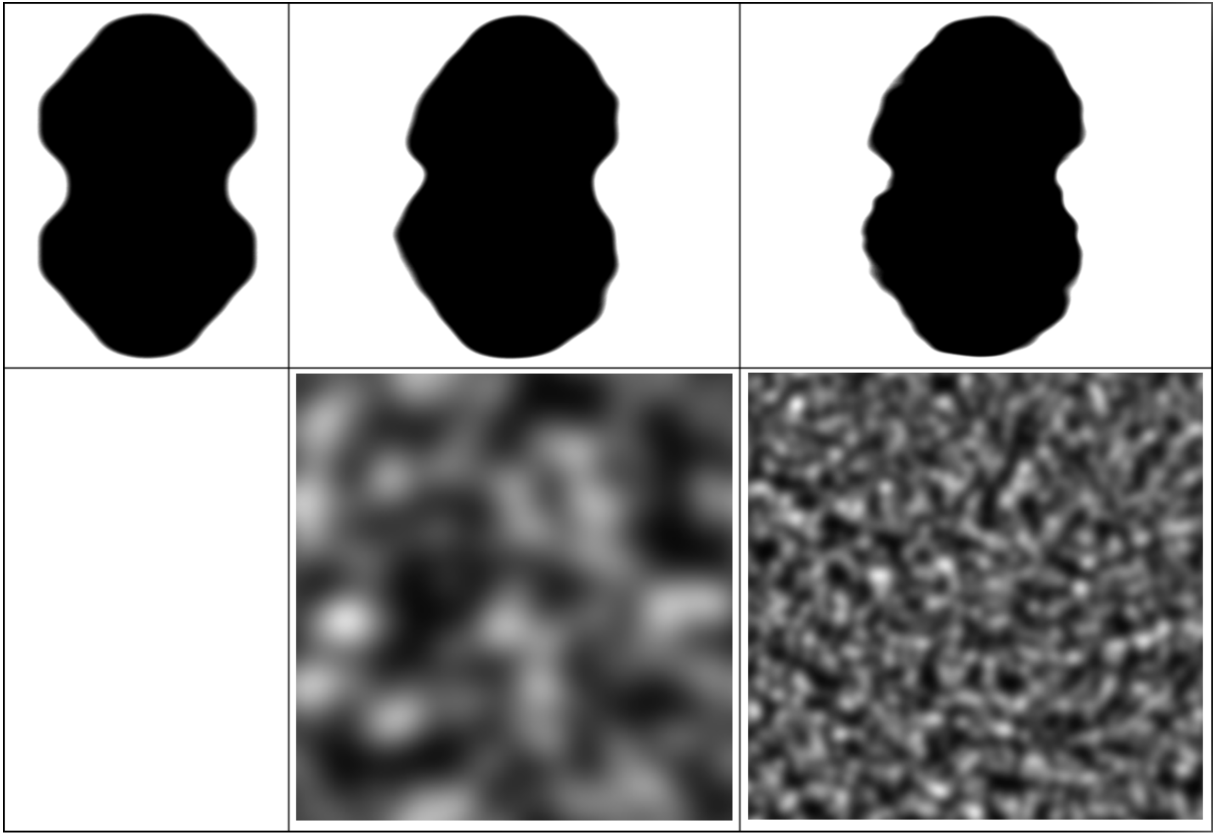
4.2.7 Pattern post-process

One problem up to this point is that the generated pattern is very artificial. Lines are perfectly straight and curves are perfectly round, unlike patterns seen in nature. To counter this, we apply a distortion effect, using two Perlin noises (PERLIN, 2002) of different granularities: one of low frequency to provide a macro variation, and one of high frequency to provide a micro variation. The intensity of these distortions are adjustable by two parameters, **Pattern_Warp_Small** and **Pattern_Warp_Large**. The suggested value range is 0.0 to 1.0 for the former and 0.0 to 10.0 for the latter. See Figure 4.14 for an example. The distortion is made using the function “Warp” from the software Substance Designer.

4.3 Height Map Generation

The height map is a grayscale texture where black means no height offset from the surface and white means high height offset. It will be used to represent scales detail, as well as fine surface irregularities. We begin by explaining how a single scale’s Height map is generated, followed by the how final height map texture is generated.

Figure 4.14 – Left: input image. Middle: low frequency distortion applied (above) and the noise used (below). Right: high frequency distortion applied (above) and the noise used (below).



Source: The author.

4.3.1 Single Scale Generation

Scales generation begins by first synthesizing a single scale, which will later be tiled. We can notice from reference images that most snake scales are leaf-shaped (see Figures 3.7 and 3.8). Also, there are two types of scales, smooth and keeled, as shown in section 3.4.

Scale generation is performed by the following steps. See Figure 4.15 for a visual depiction of each step's result.

a A grayscale circumference is generated. Its intensity is 1.0 at the center and linearly decays to 0.0 at the edges;

b Midtones are adjusted to make the image more solid;

c One copy is created and translated to the left...

d ... to the right...

e ... and blended together by taking the minimal value between these two. This is to generate the scale's leaf shape;

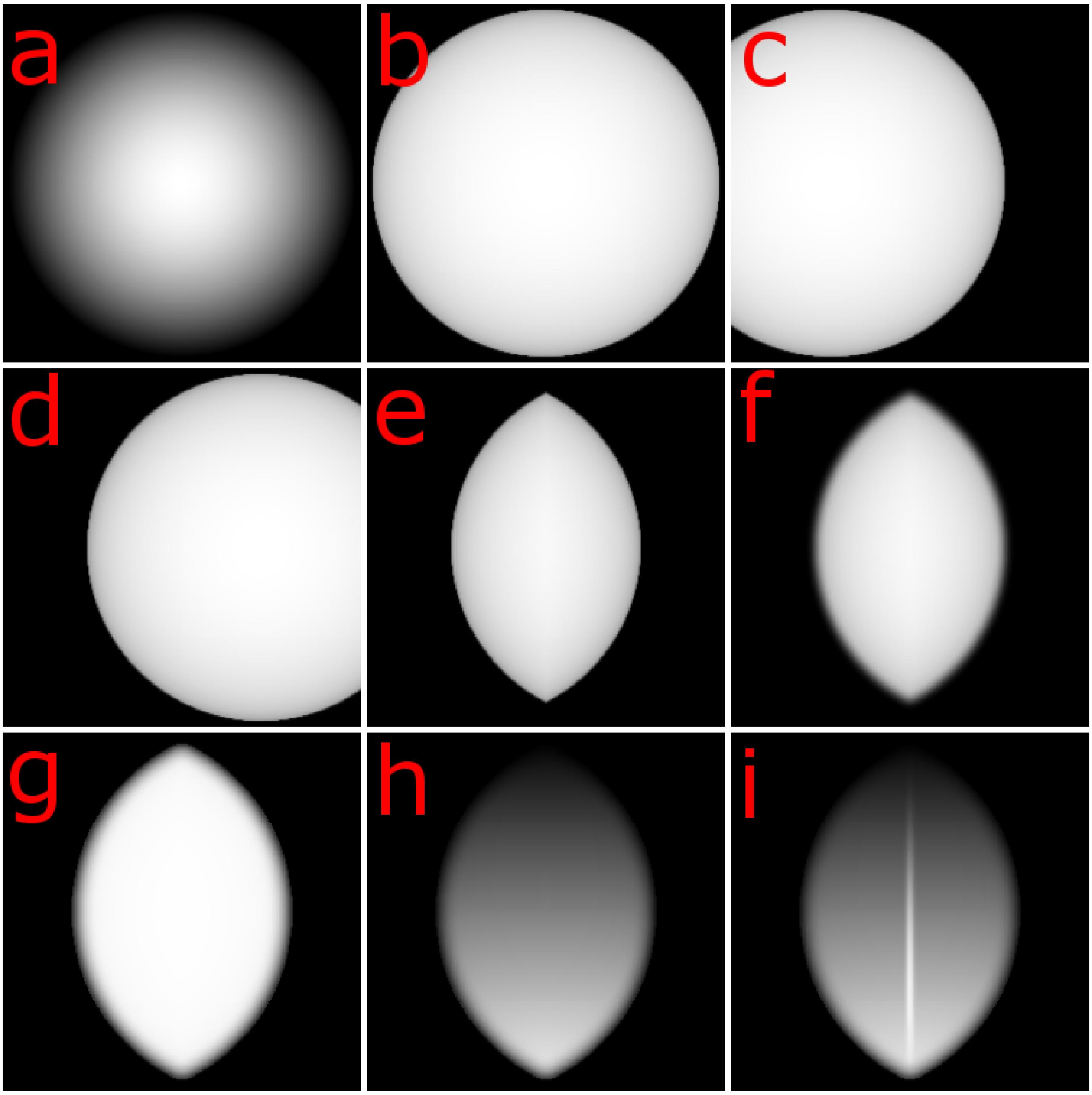
f A blur is applied...

g ... and contrast is increased to make the image smoother, particularly on the top and bottom ends;

h It is multiplied with a linear gradient, ranging from black to white, to create the scale's height map;

i If the scale is keeled, then a thin linear gradient ranging from black-white-black is blended into the center of the scale.

Figure 4.15 – Scale generation steps.



Source: The author.

The available parameters for single scale generation are:

- **ScaleShape_Contrast**: float ranging from 0.0 to 1.0. Controls the midtones (see Figure 4.15.b) on the generated circumference (4.15.a);
- **ScaleShape_Blur**: float that controls the blur applied (see Figure 4.15.f). Suggested value range: 0.0 to 16.0;
- **ScaleShape_Gradient**: float ranging from 0.0 to 1.0. Controls the linear gradient's contrast (see Figure 4.15.h), which is used to form the height map;
- **ScaleShape_IsKeeled**: boolean, controls whether the scale should be keeled or not (see Figure 4.15.i);
- **ScaleShape_KeelThickness**: float ranging from 0.0 to 1.0. Controls the width of the keel. At 0.0 it is so thin it disappears and at 1.0 it occupies the entire scale's width, so suggested values are in the range 0.02 to 0.1.

4.3.2 Scales Tiling

Once the single scale is generated, it is instanced horizontally and vertically a number of times adjustable by parameters (i.e. the number of scales). The last row is drawn first, then the upper rows up to the first, so that top rows occlude bottom rows. It is also possible to add a random variation to size, location and rotation for each duplicated scale, so the result looks more natural. On the left and right ends, we stretch the texture to create ventral (underside) scales. See Figure 4.16.c for an example.

We can also enhance fine details on this texture by using a high-frequency noise. We can notice from reference images that scales' surface has some irregularities, which we model by adding a Fractal Sum noise texture (MANDELBROT; PIGNONI, 1983) to the height map. The intensity should be very low as its purpose is only to add micro details.

Available parameters:

- **Scales_DetailNormal**: float that controls intensity the fractal sum noise, which is used on the scales to add micro detailing. Optimal values are in the range 0.0 to 0.05;
- **Scales_XAmount**: integer that controls how many scale instances will be tiled along the X axis. Suggested value range: 1 to 64;

- **Scales_YAmount**: integer that controls how many scale instances will be tiled along the Y axis. Suggested value range: 1 to 64;
- **Scales_ScaleSize**: float that controls the size of individual scales. Suggested value range: 0.0 to 2.0;
- **Scales_BellyScalesPct**: float ranging from 0.0 to 1.0. Controls where the ventral area begins and ends, as a percentage of total texture width. Suggested value range: 0.1 to 0.2. It is further described in Section 4.3.3.

It should be noted that the parameters **Scales_XAmount** and **Scales_YAmount** control the amount of scales on the texture, so the amount of scales remains unchanged regardless of whether the snake's mesh is thin or thick. Scales will simply be compressed or stretched according to the thickness of the snake's mesh.

4.3.3 Ventral Scales

On their ventral (under) side, snakes have only a single column of scales per row. We model this by stretching the generated Height map on the left and right ends. From the parameter **Scales_BellyScalesPct**, we detect the closest scale's center on the horizontal and stretch it to both ends. Figure 4.16.c shows an example.

4.4 Roughness Map Generation

We also synthesize another texture to describe how glossy or rough the surface looks. This is important because snakes with smooth scales have a glossier appearance than keeled ones (see Section 3.4 for details). The roughness map is a grayscale texture where black means the surface is not rough (i.e., glossy or polished) and white means it is rough (diffuse). In technical terms, low roughness values mean there is more specular reflections than diffuse reflections; while high roughness values mean there is more diffuse reflections than specular reflections. Roughness maps are used in certain game engines that use a physically-based renderer, such as Unreal Engine 4. Other game engines, such as Unity 5 and CryEngine 3, use a Glossiness map instead, which is simply the inverse of the Roughness map. Other rendering solutions such as Marmoset Toolbag and Substance Designer can use either a roughness or a Glossiness map. For more details on physically-based rendering, we recommend (PHARR; JAKOB; HUMPHREYS, 2016).

While we could have used a single solid color for the roughness map, we chose to use the same fractal sum (MANDELBROT; PIGNONI, 1983) noise texture from the height map, to add fine details. Maximum output levels are adjustable as a means to adjust the minimal and maximum surface roughness. Figure 4.16.d shows an example of generated roughness map. Available parameters:

1. **Scales_RoughnessMin**: float ranging from 0.0 to 1.0. Controls the minimum black level on the noise texture, effectively controlling the minimal surface roughness;

2. **Scales_RoughnessMax**: float ranging from 0.0 to 1.0. Controls the maximum white level on the noise texture, effectively controlling the maximal surface roughness;

We suggest low values (between 0.2 and 0.4) for snakes with smooth scales (because the surface is glossier) and high values (between 0.6 and 0.9) for snakes with keeled scales (because the surface is rougher). These values were defined empirically by adjusting and comparing to reference photos. The greater the difference between the minimal and maximal roughness values, the dirtier the surface looks because it will reflect light in different, noticeable shapes (glossy to rough).

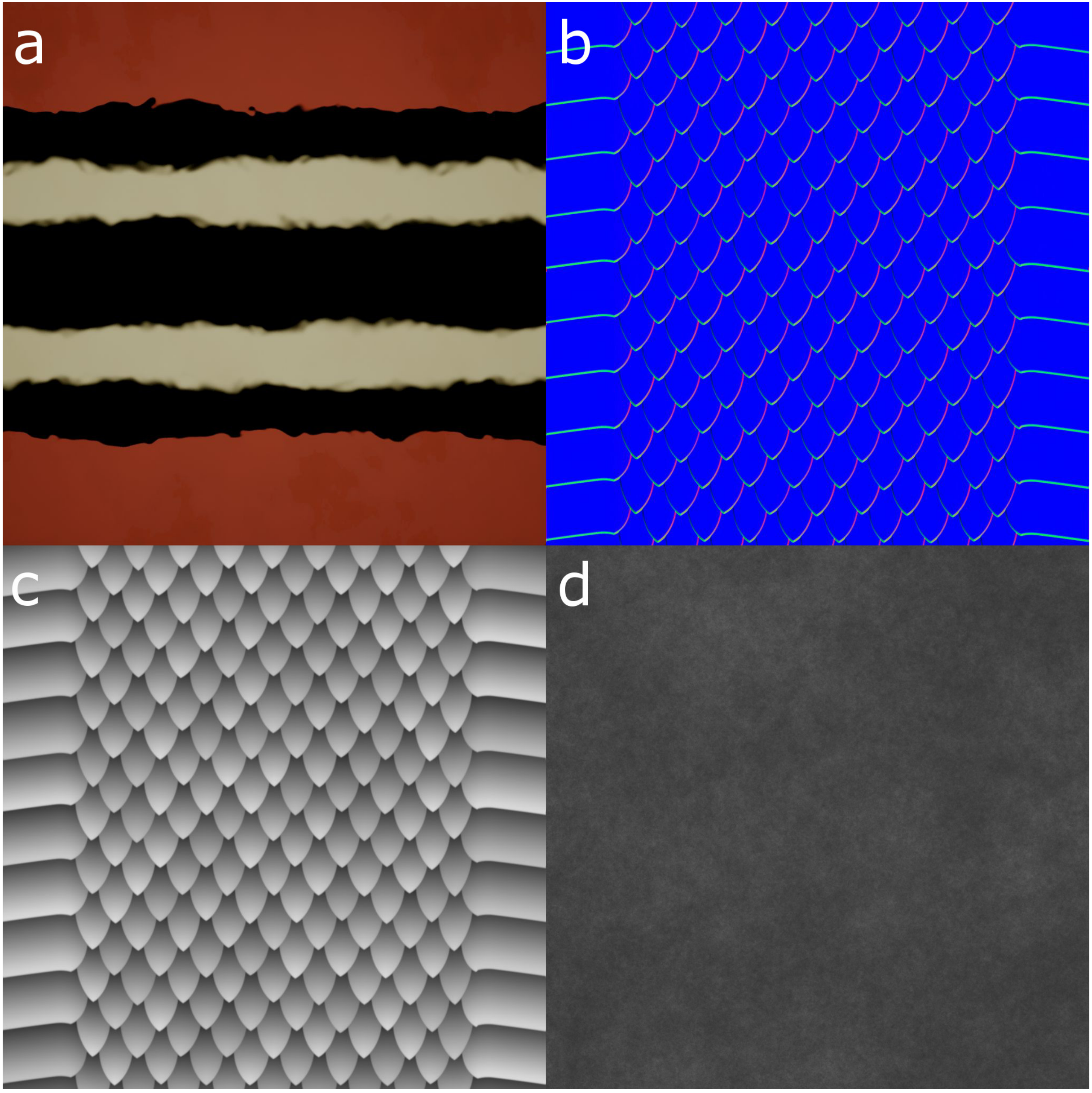
4.5 Discussion

In this chapter, we discussed how our texture generator was implemented. It generates three distinct textures: a color map, where goes the snake's skin color and pattern; a height map, where goes the scales texture; and a roughness map, to describe the surface's roughness. These textures are independent so all three can be generated at once in separate threads.

Figure 4.16 shows all generated textures for the *Micrurus altirostris* snake. It also shows a normal map, which was generated from the height map, and is used by certain renderers to add local shading (it is better explained in Chapter 5).

In the next chapter, we show results obtained from this procedural model.

Figure 4.16 – Generated textures for the *Micrurus altirostris*. (a): Base color. (b): Normals map. (c): Height map. (d): Roughness map.



Source: The author.

5 RESULTS

In this chapter, we show results obtained with our procedural model. First, we present specifics on our implementation of the model, followed by sample snake renders using textures generated by our model. Finally, we show user test results we ran on our implementation.

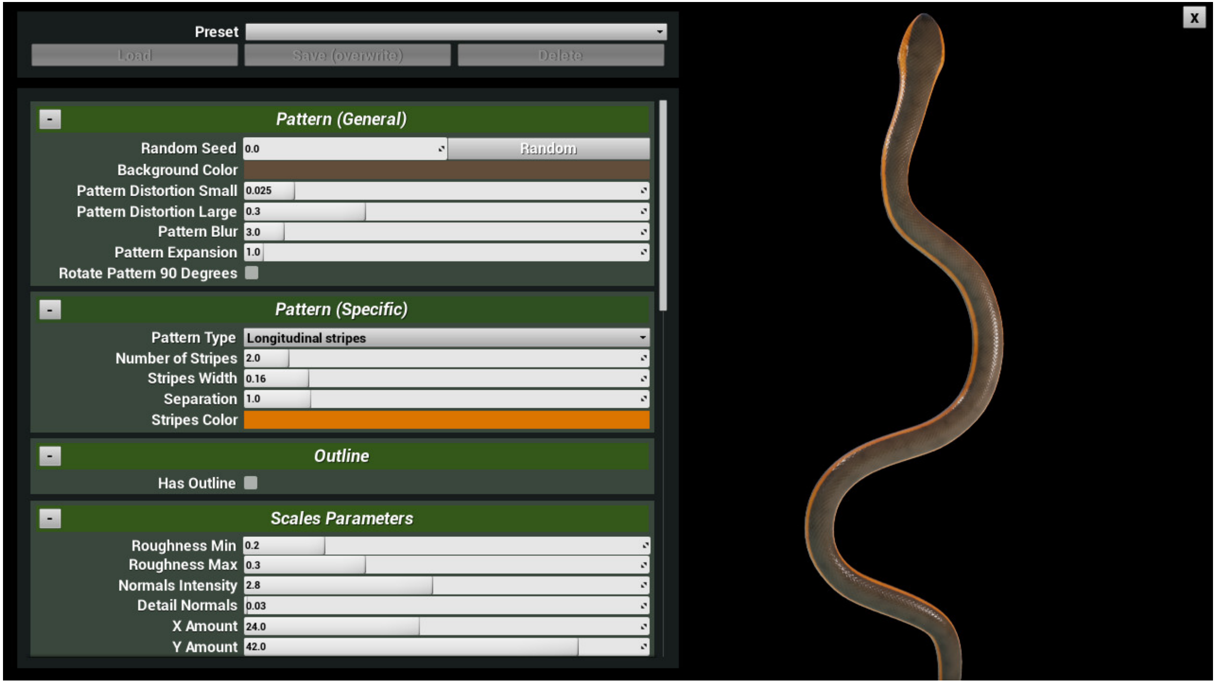
The software Substance Designer version 2017.1.0, by Allegorithmic SAS, was used to implement the texture generation process. It is a node-based procedural texture generator, which performs many image processing routines such as blur, levels adjustment (brightness, contrast, minimal output and maximal output), 2D transformations and distortion; and also has several built-in noise generators, such as Perlin noise (PERLIN, 1985). We chose this software because it is easy to use and has most routines we'll need already implemented, ready for use.

However, work created in Substance Designer can only be edited if the user owns a copy of the software. Therefore, we developed a tool which is basically an interface to change parameters on the procedural generator. We also added other functionalities to the tool:

- Load and save presets (the values from all parameters);
- Display what the final textures look like;
- Display the textures applied on a manually modeled snake 3D mesh;
- Load images from the Internet and display them within the tool. This can be used for the user to see photo references of snake species;
- Export the resulting textures in bitmap format to use them in other tools or renderers.

The software was implemented using Unreal Engine 4, by Epic Games Inc., which is a real-time game engine that uses a physically-based renderer. We chose this game engine because it has integration with Substance Designer, and also because it has a user interface designer and a renderer to display the preview snake mesh. The software's main interface can be seen in Figures 5.1 and 5.2.

Figure 5.1 – The software's interface. On the left is the panel where parameters are adjusted. On the right, a preview is shown on a manually modeled snake 3D mesh.



Preset

Pattern (General)

Random Seed 0.0

Background Color

Pattern Distortion Small 0.025

Pattern Distortion Large 0.3

Pattern Blur 3.0

Pattern Expansion 1.0

Rotate Pattern 90 Degrees □

Pattern (Specific)

Pattern Type Longitudinal stripes

Number of Stripes 2.0

Stripes Width 0.16

Separation 1.0

Stripes Color 🟧

Outline

Has Outline □

Scales Parameters

Roughness Min 0.2

Roughness Max 0.3

Normals Intensity 2.8

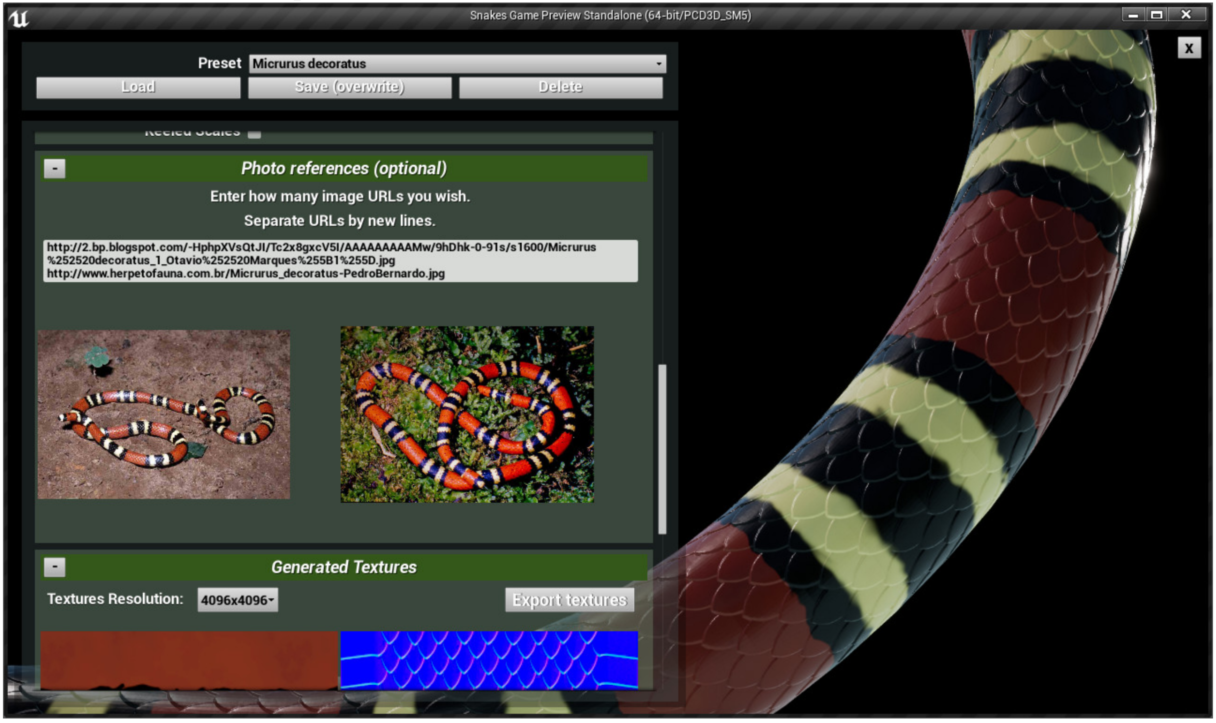
Detail Normals 0.03

X Amount 24.0

Y Amount 42.0

Source: The author.

Figure 5.2 – The software's interface, after creating the *Micrurus decoratus*. The left panel was scrolled down to show the optional photo references section and the generated textures, with option to change their resolution and to export them.



Preset Micrurus decoratus

Keeled Scales □

Photo references (optional)

Enter how many image URLs you wish.

Separate URLs by new lines.

http://2.bp.blogspot.com/-HphpXVsQtJI/Tc2x8gxcV5I/AAAAAAAAAMw/9hDhk-0-91s/s1600/Micrurus%252520decoratus_1_Otavio%252520Marques%255B1%255D.jpg
http://www.herpetofauna.com.br/Micrurus_decoratus-PedroBernardo.jpg

Generated Textures

Textures Resolution: 4096x4096

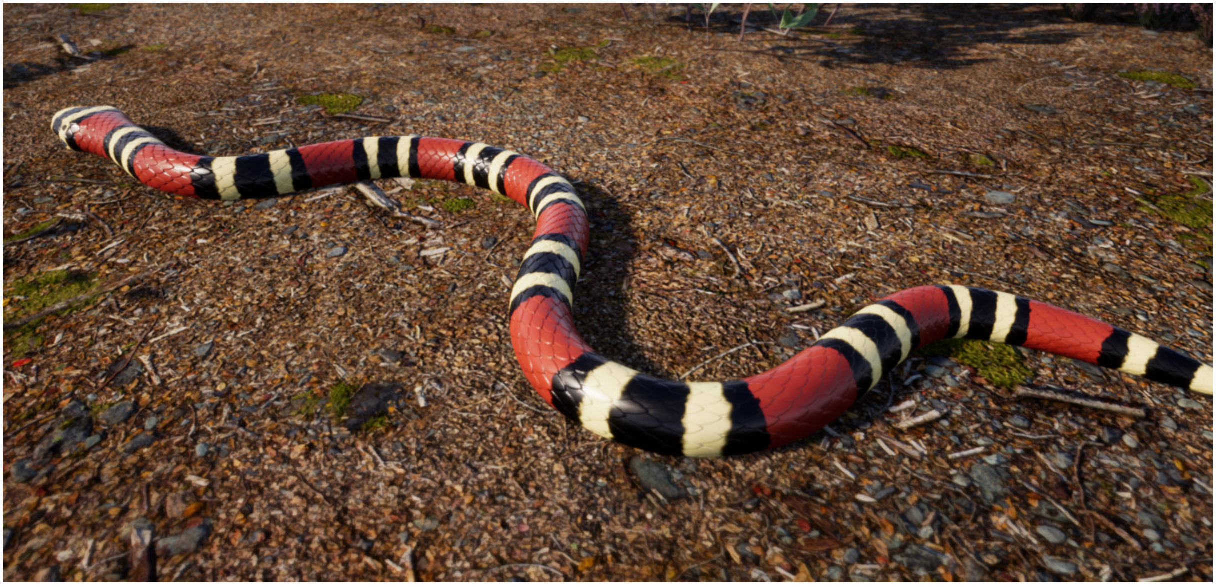
Source: The author.

This game engine uses a normal map texture (not the height map) to add local shading. Using a function available in Substance Designer, the height map generated by the model is converted to a normal map for use by the engine, with its intensity adjustable by the parameter **Normal_Intensity**. The higher the intensity, the higher the contrast on the red and green channels, which control light reflection angles at the surface's tangent and binormal vectors direction respectively.

The cellular automaton was implemented using Matlab, because it is easy to perform matrix operations. The implementation simply shows a dialog box for the user to select a cellular automaton descriptor file (see Section 4.2.4 for details), computes the output, and saves it in the same folder in tga format. Currently, it is not possible to load the tga file directly into the tool to use them for the Other Simple Pattern type; they must be first imported within Unreal Engine 4.

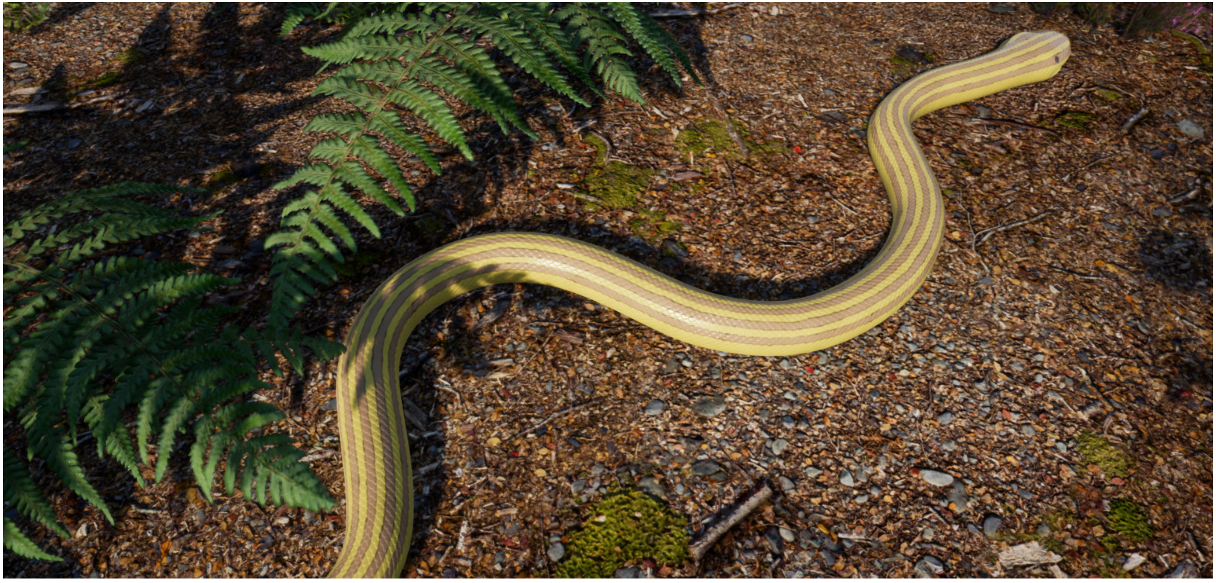
Our model can generate a multitude of snake skin textures. Some examples can be seen in Figures 5.3, 5.4, 5.5 and 5.6. These Figures show snakes with textures synthesized by our generator in an example environment, rendered in real-time in Unreal Engine 4. We chose at least one snake from each pattern type. Background scenario assets (ground, foliage and rocks) were provided by Epic Games, Inc., and the snake 3D mesh was manually modeled using Blender. To generate these samples, we looked at reference photos and reproduced them by manually tweaking parameters, such as pattern type and size, colors, and specific pattern parameters. There are 59 parameters in total.

Figure 5.3 – Generated *Micrurus altirostris* textures rendered on a snake mesh. It uses the Transversal Stripes pattern type.



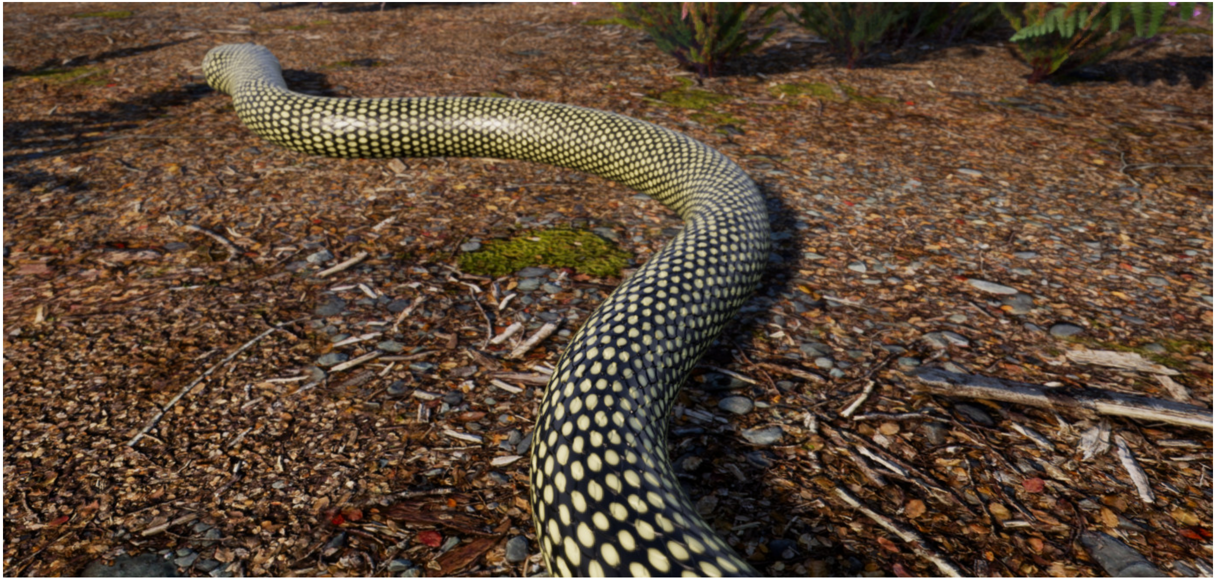
Source: The author.

Figure 5.4 – Generated *Elapomorphus quinquelineatus* textures rendered on a snake mesh. It uses the Longitudinal Stripes pattern type.



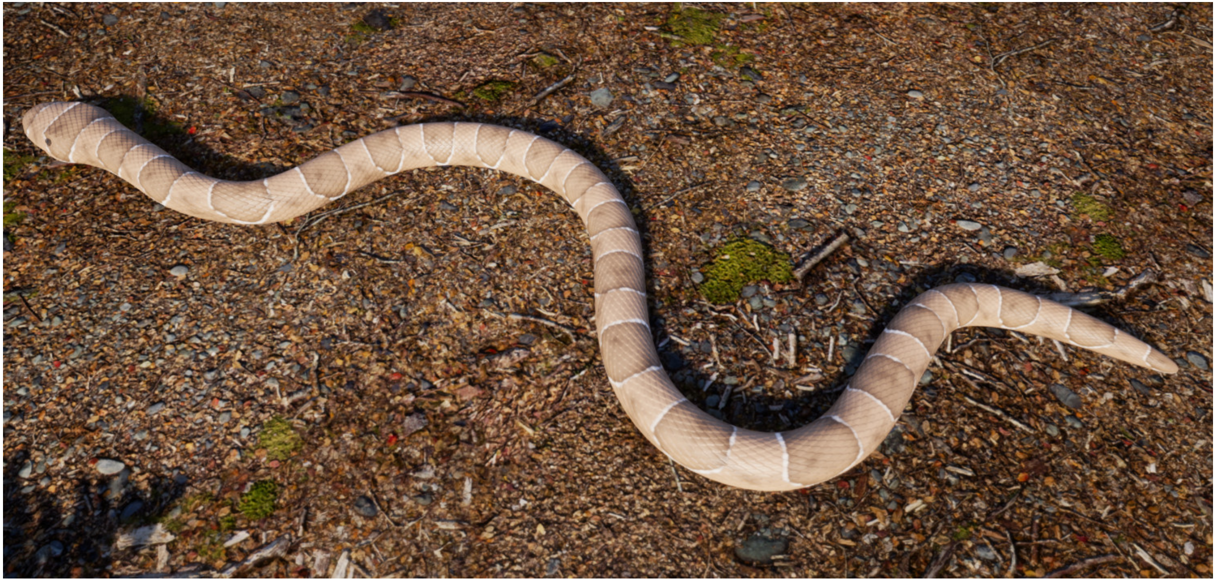
Source: The author.

Figure 5.5 – Generated *Liophis miliaris* textures rendered on a snake mesh. It uses the Spots pattern type.



Source: The author.

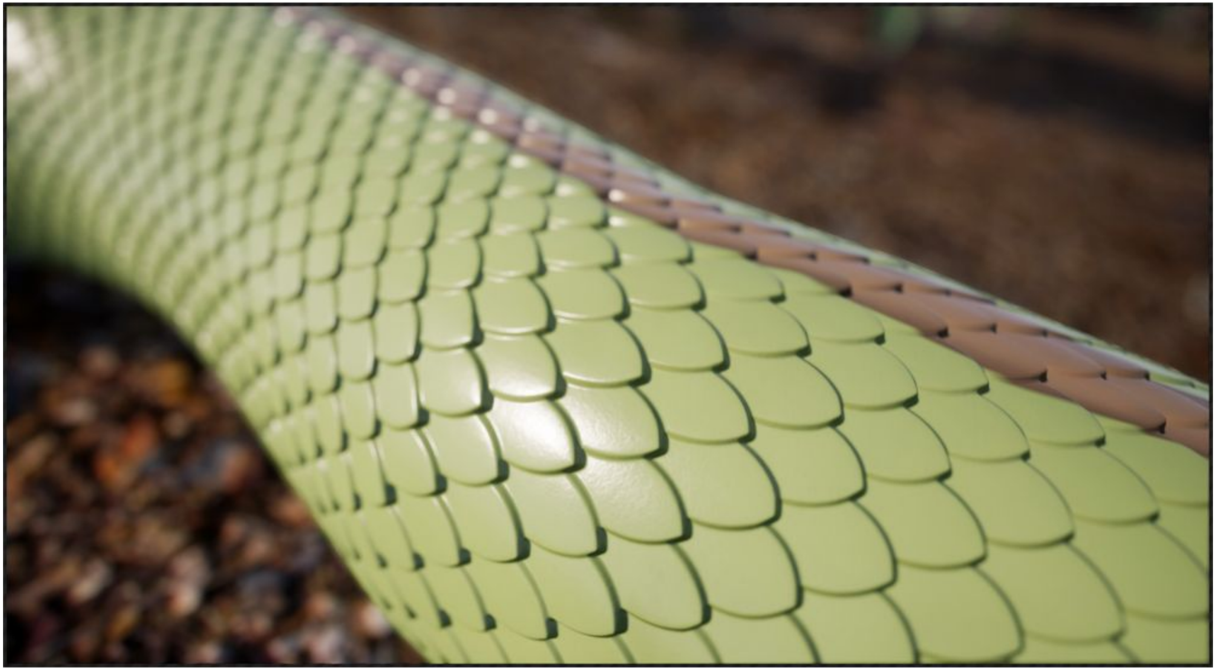
Figure 5.6 – Generated *Crotalus viridis* textures rendered on a snake mesh. It uses the Other Simple Pattern pattern type.



Source: The author.

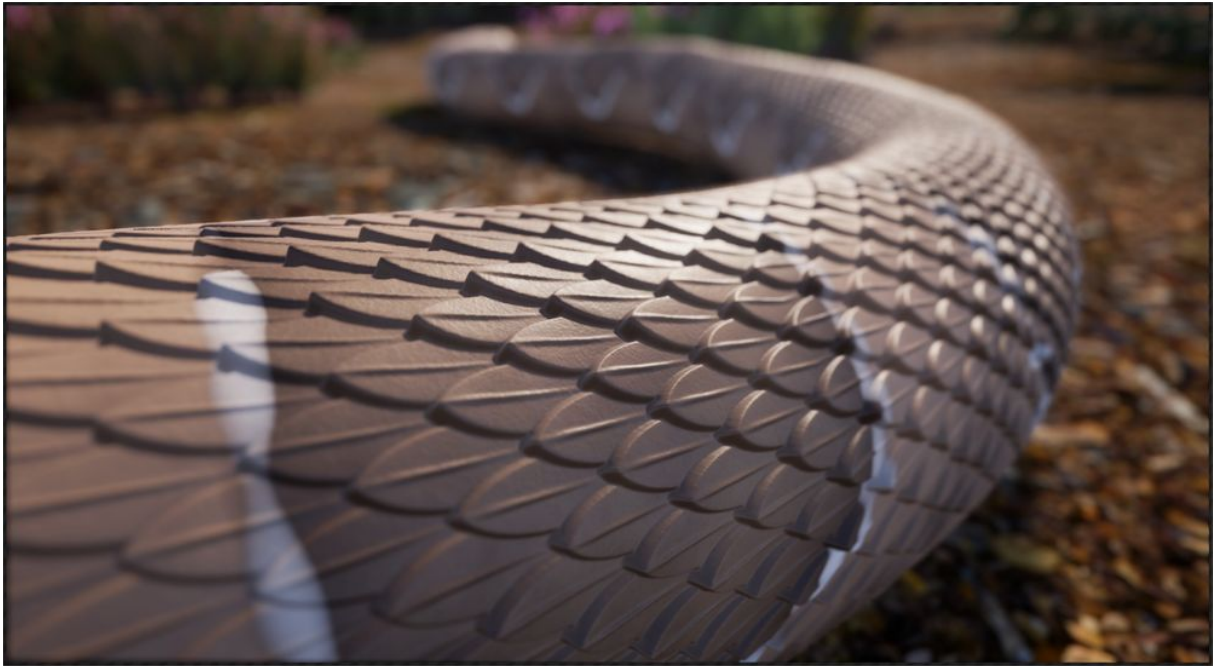
Figure 5.7 better shows smooth scales detail, and Figure 5.8 shows keeled scales detail.

Figure 5.7 – Generated smooth scales.



Source: The author.

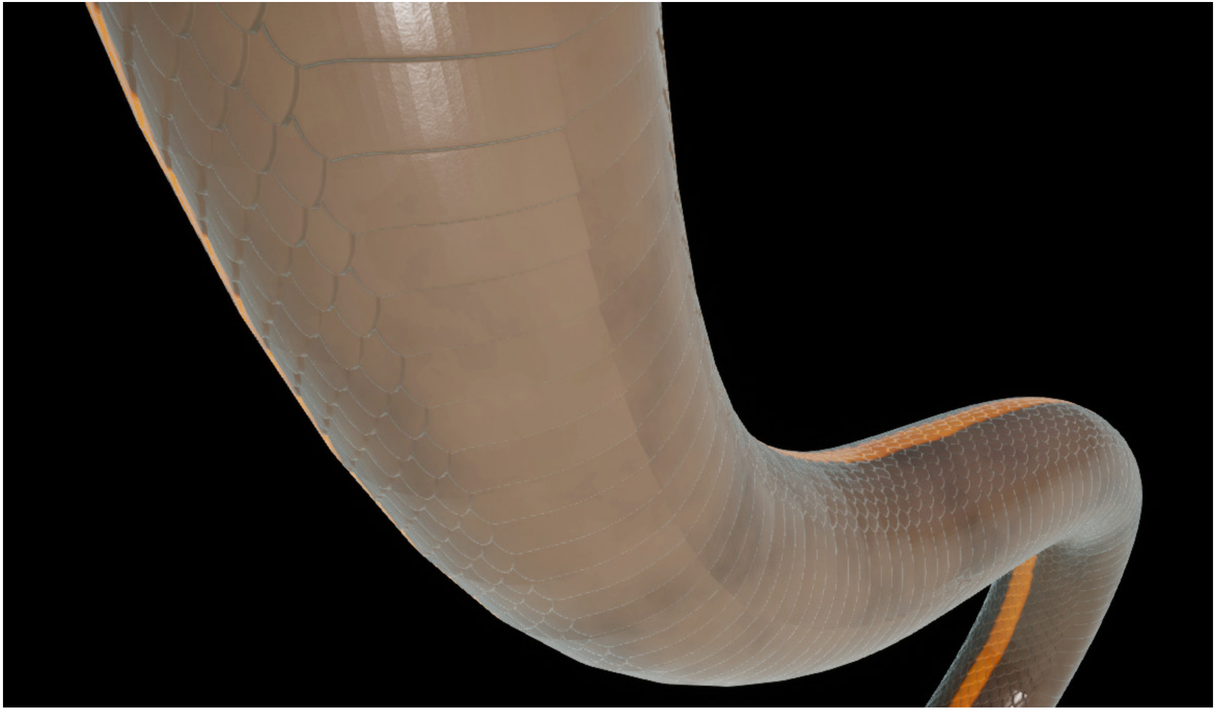
Figure 5.8 – Generated keeled scales.



Source: The author.

Figure 5.9 show scales detail on the snake's underside.

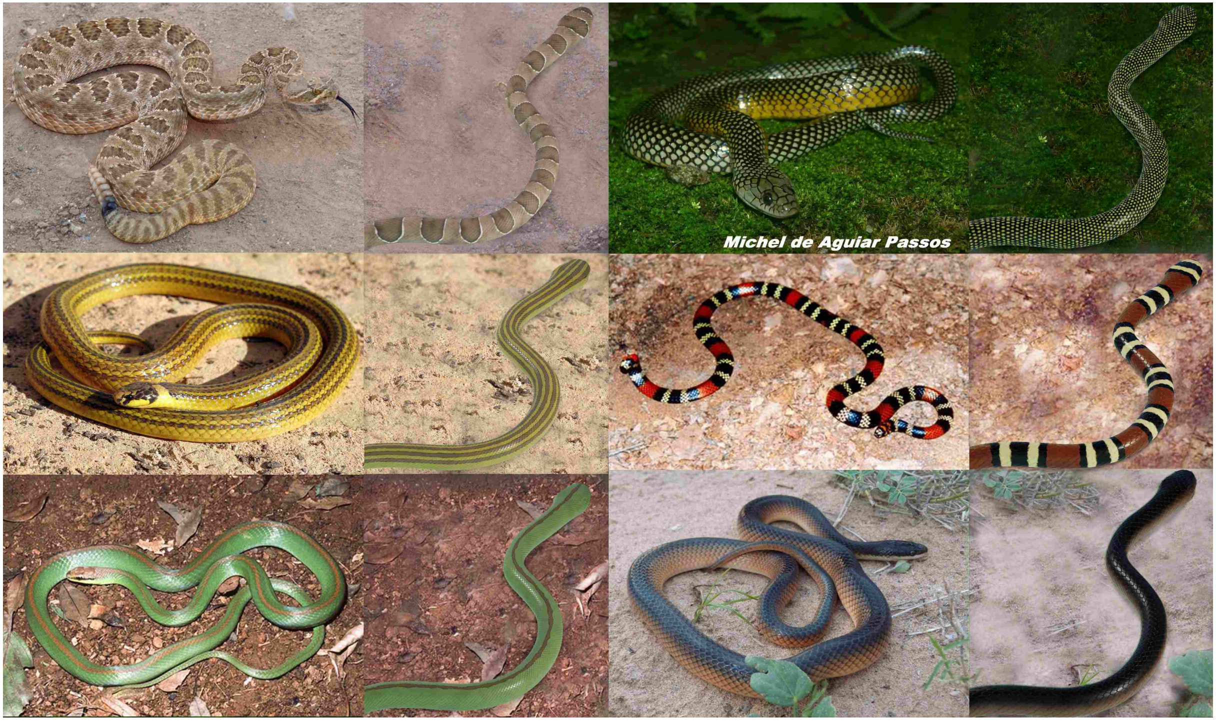
Figure 5.9 – Generated underside scales.



Source: The author.

Figure 5.10 shows a side-by-side comparison of reference photos and generated textures. Background on synthesized images was replaced with the background from the reference photo, using an image editing software, for better comparison.

Figure 5.10 – Photo reference (left) and textures obtained using our texture synthesizer (right), applied on a 3D mesh. In reading order: *Crotalus Viridis*, *Liophis miliaris*, *Elapomorphus quinquelineatus*, *Micrurus altirostris*, *Philodryas olfersii* and *Mussurana bicolor*.

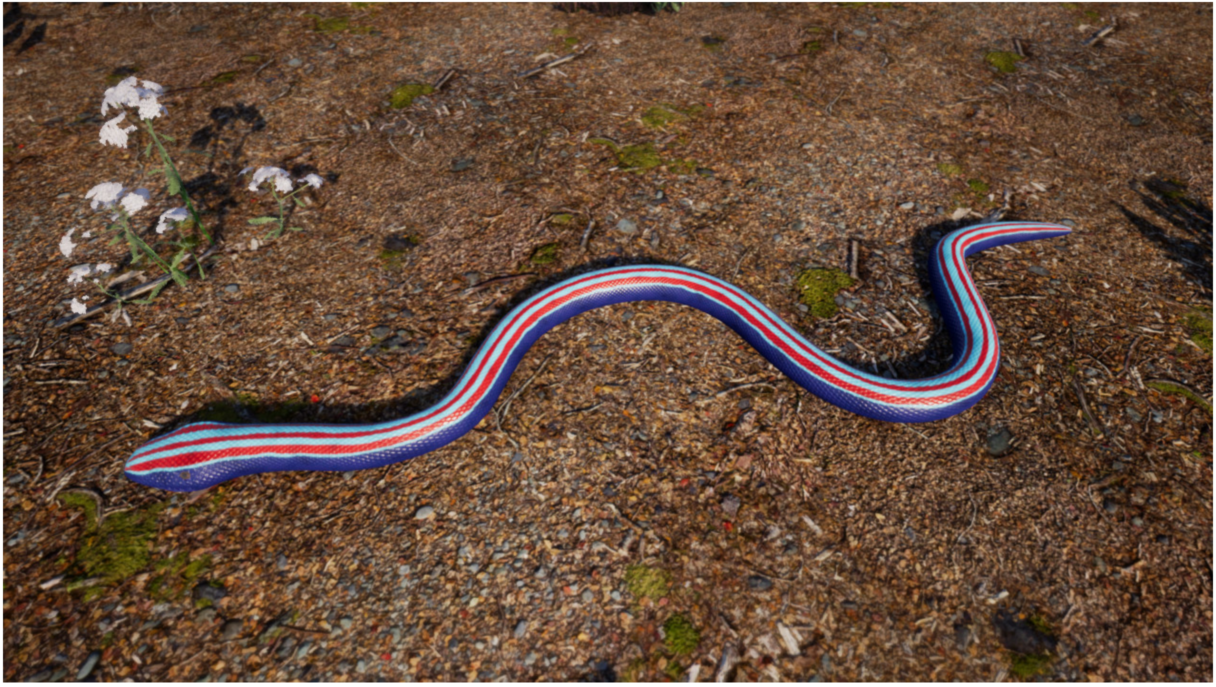


Sources for left images (photographs): *Crotalus Viridis*: (STUART, 2011), *Liophis miliaris*: (PASSOS, 2013), *Elapomorphus quinquelineatus*: (MARTINS, 2011), *Micrurus altirostris*: (BORGES-MARTINS, 2007a), *Mussurana bicolor*: (SMITH, 2015). Source for right images (renders): The author.

We can see that our model is capable of generating textures that are quite similar to their real-world counterparts, as well as generating fine details, as seen in Figures 5.7 and 5.8.

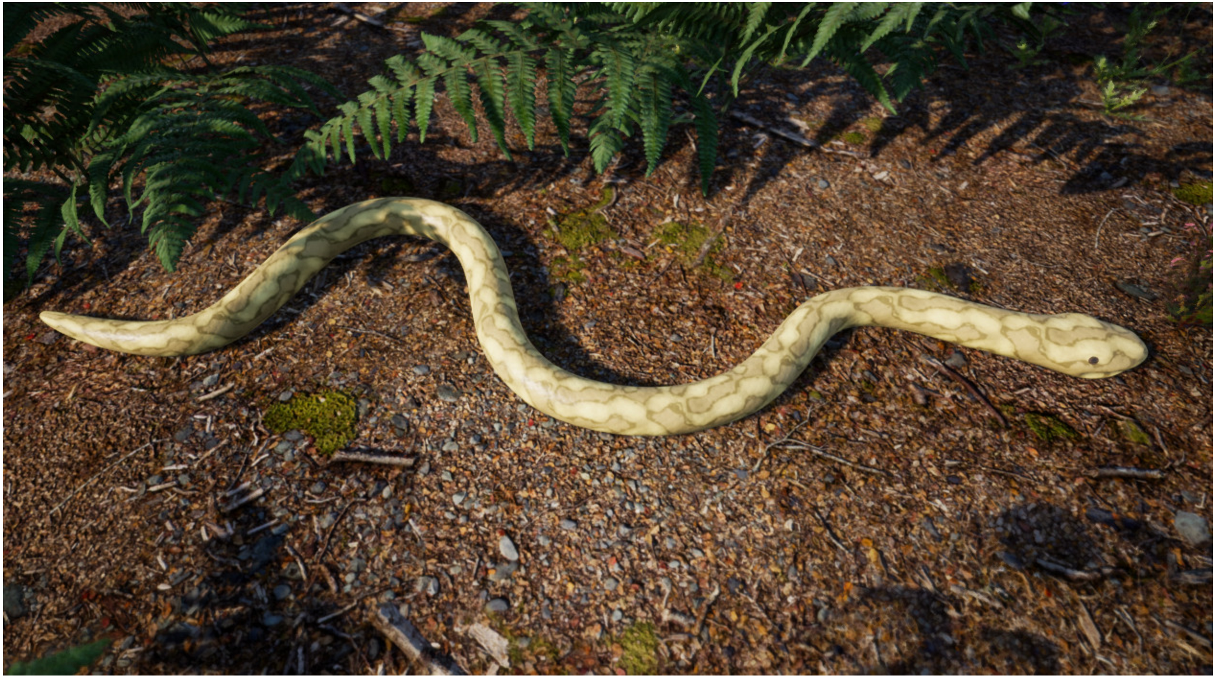
Although we based our study on real-world snakes, it is entirely possible to use our model to create unrealistic, or fantasy, snake textures. Some examples can be seen in Figures 5.11, 5.12, 5.13 and 5.14.

Figure 5.11 – Generated textures for a fictional snake.



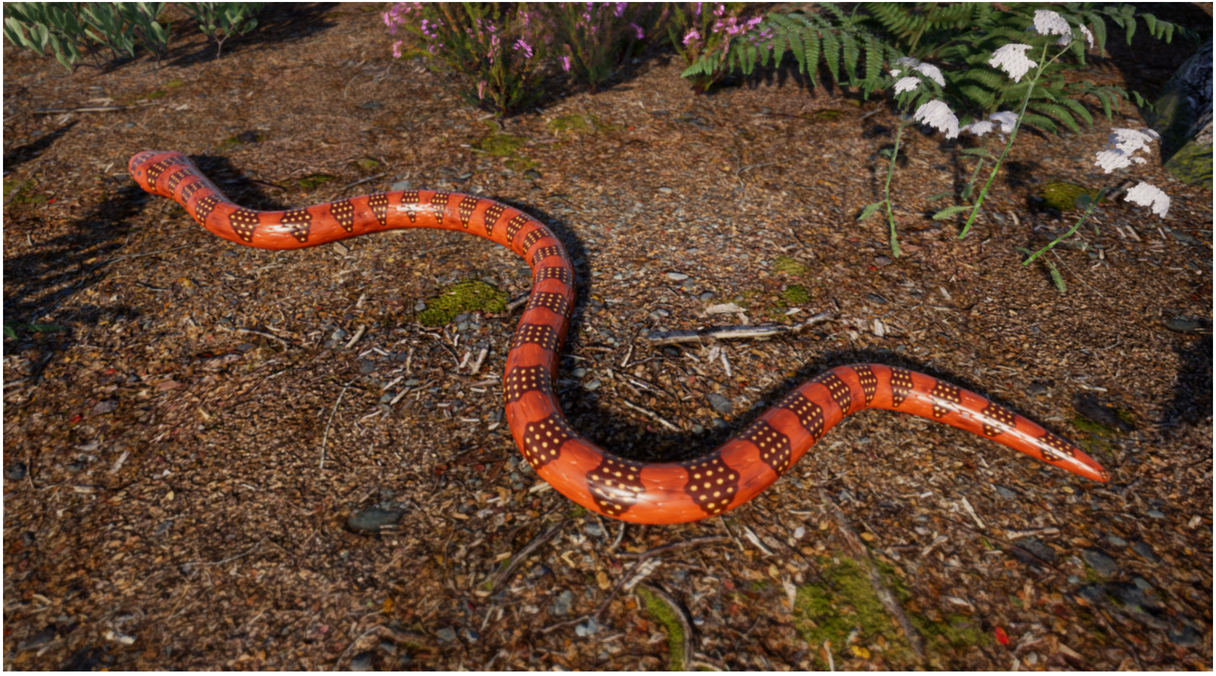
Source: The author.

Figure 5.12 – Generated textures for a fictional snake.



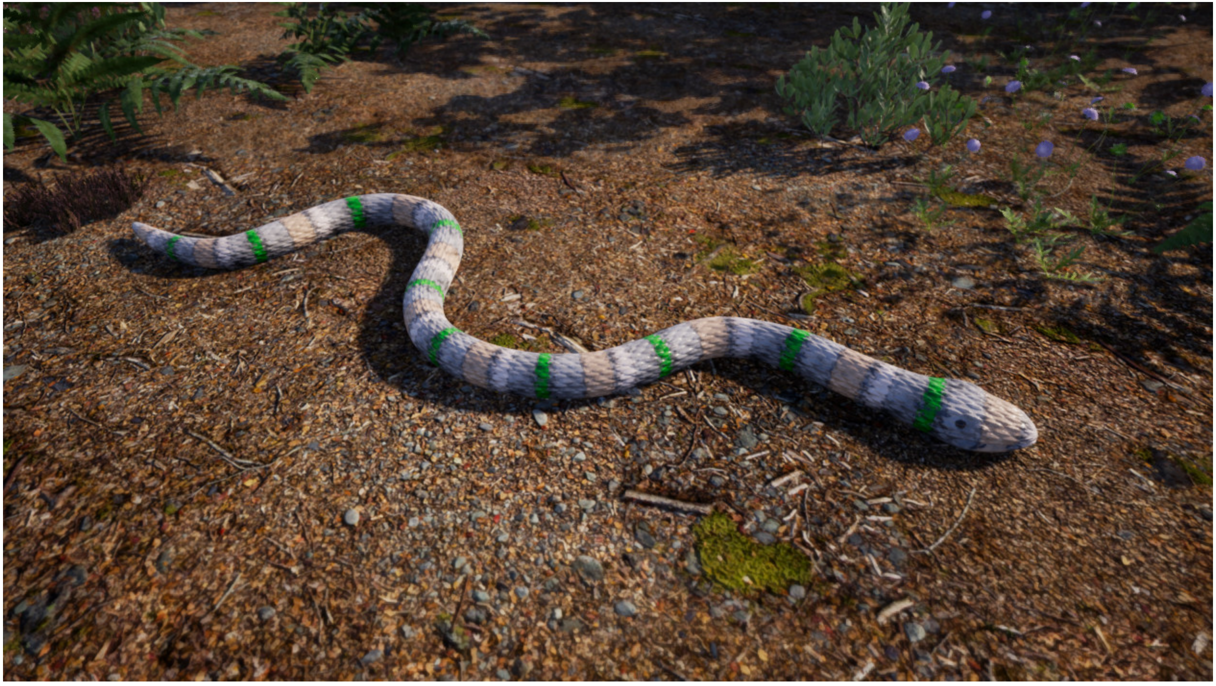
Source: The author.

Figure 5.13 – Generated textures for a fictional snake.



Source: The author.

Figure 5.14 – Generated textures for a fictional snake.



Source: The author.

The exact parameters used to generate all of these textures can be seen in Appendix D.

5.1 Performance

In out test system, equipped with an Intel Core i5-4670K @ 4.1 GHz, 16GB RAM and NVIDIA GeForce GTX 1070, it takes an average of 82.73ms to compute all 4 textures (base color, normal map, height map and roughness map) at 1024x1024 resolution. Thus, the user can change parameters and see the result in real-time, allowing to work iteratively. Table 5.1 shows the average time taken to compute all 4 textures in other resolutions.

Table 5.1 – Average computing times for all 4 textures at different texture resolutions.

Resolution	Average time
256x256	51ms
512x512	56.7ms
1024x1024	82.73ms
2048x2048	163.06ms
4096x4096	554.82ms

5.2 User Tests

In order to identify how useful our tool is for the average designer, as well as its usability, we ran user tests. We selected as subjects students from the 4th semester of an undergraduate Game Design course at UniRitter, located in the city of Porto Alegre, Brazil. 13 subjects participated in the test, most of them between 21 and 26 years old, and most of them males (12 males, 1 female). The test involved 4 tasks:

1. Freely explore the tool;
2. Attempt to reproduce the snake *Elapomorphus quinquelineatus* (see Figure 5.4);
3. Attempt to reproduce any coral snake of their choice;
4. Attempt to reproduce any snake of their choice.

For each task, users had a maximum working time of 7 minutes. We chose this time limit because we take an average time of 3 minutes to create each of these textures, and added 4 minutes for unexperienced users.

Users had to look for photos of these snakes on the internet. The time they had to browse for images was free (it was not counted in each task's time limit). After taking this test, they answered a questionnaire anonymously. The full test results can be seen in Appendix A.

Among the questions, we asked whether they had any prior experience in texturing. 30.8% said they had no prior experience at all, 69.2% said they had some experience, and no one said they had good experience or worked with texturing frequently. However, most of them (90%) were already familiar with physically-based rendering before trying the tool.

We also asked if they were able to reproduce each of the three snakes, with 3 options: "Yes", "Yes, partially" and "No". No user answered "no" for any snake.

- For the first snake (*Elapomorphus quinquelineatus*), 61.5% answered "Yes" against 38.5% for "Yes, partially".
- For the second snake (the coral), 92.3% answered "Yes" against 7.7% for "Yes, partially".
- For the third snake (user's choice), 30.8% answered "Yes" against 69.2% for "Yes, partially".

When asked how easy it was to reproduce each snake, most users reported the *Elapomorphus quinquelineatus* as easy, followed by the coral with mixed results and lastly their chosen snake as the most difficult. It is likely that some users chose a snake with a complex pattern, or one that is not supported by our model.

Thus, we asked if users realized they couldn't reproduce all patterns in the tool (the complex patterns). 30.8% answered "No" (meaning they think all patterns can be reproduced), 38.5% answered "Yes, just a few", 30.8% answered "Yes, several", and 0% answered "Yes, many".

Lastly, we asked the 10 questions from the System Usability Scale (SUS) by John Brooke (BROOKE, 1986). This questionaire is used to assess a system's usability for the end user. It is composed of 10 objective questions regarding usability with options ranging from "Strongly disagree" to "Strongly agree". From the answers, it is possible to calculate a score that ranges from 0 (bad usability) to 100 (good usability).

We only adapted the first question. The original "I think that I would like to use this system frequently" was changed to "I would like to use this tool in the future to create snake textures". We also changed all mentions of "system" to "tool". The SUS test result was 85.8 points on average for the 13 users. This indicates a very good usability, and that users are likely to use the tool in the future to generate snake skin textures.

Part of our work (excluding the cellular automaton implementation) was made available for free at https://share.allegorithmic.com/libraries/3214. Feedback from other users is positive (4 reviews, all with 5 stars) and it was downloaded 64 times as of October 20, 2017 (24 days after it was uploaded).

5.3 Herpetologist User Test

We've shown our texture generator to an herpetologist to gather feedback. Although originally intended to use in the game or animation industry, the herpetologist said the tool is useful to them as well. It could be used to easily create a visual representation of a new snake species: according to her, biologists currently use photos (which aren't always available or of good quality) or manually draw using a vector graphics tool such as CorelDraw. The tool could also be used to communicate a snake's appearance, when it was spotted in the field but couldn't be photographed or captured. The herpetologist tested our tool and answered a questionnaire, which can be seen in Appendix C. The System Usability Scale score was 80.0, however, keep in mind we had only one test subject.

5.4 Discussion

In this chapter, we have shown our results, that have good similarity with real world counterparts. We also ran user tests that suggest that the tool is easy to use and that users are likely to use it in the future for generating snake skin textures. In the next chapter we present our conclusions and directions for future work.

6 CONCLUSIONS

We presented a procedural model that uses a combination of several types of procedural generators to synthesize snake skin textures. We implemented a tool for the end-user to define generation parameters, which was proven to be easy to use. The tool takes as input the many parameters (such as colors and number of stripes) and outputs textures of any resolution, ready to be used in today's renderers. We have shown that the generator is able to synthesize textures with a high degree of correspondence with real-world counterparts.

From the results we obtained and the positive feedback on user tests, we believe our tool proves to be a quite useful texture synthesizer for snake skin patterns. Even though we based our study only on snakes found in the Rio Grande do Sul region, we found it can be used for snakes from other regions as well without any problem (save the complex patterns we are unable to reproduce – mostly present in the Pythonidae family), as well as fictional snakes.

We've also contributed to the Herpetology field with our categorization of snake skin patterns, as well as our visualization which allowed us to discover facts that were previously unknown by herpetologists (see Appendix B).

With respect to limitations, our tool is unable to reproduce textures of snakes categorized as having the "Complex" pattern type, which was not implemented as discussed on section 4.2.5, and snakes that combine multiple pattern types (such as both longitudinal stripes and spots). We also do not generate head and tail textures for two reasons. First, these would be dependent on how the mesh's UV coordinates were mapped. Second, due to the much higher complexity. Scales on the head often have a specific amount for each species, and many different shapes.

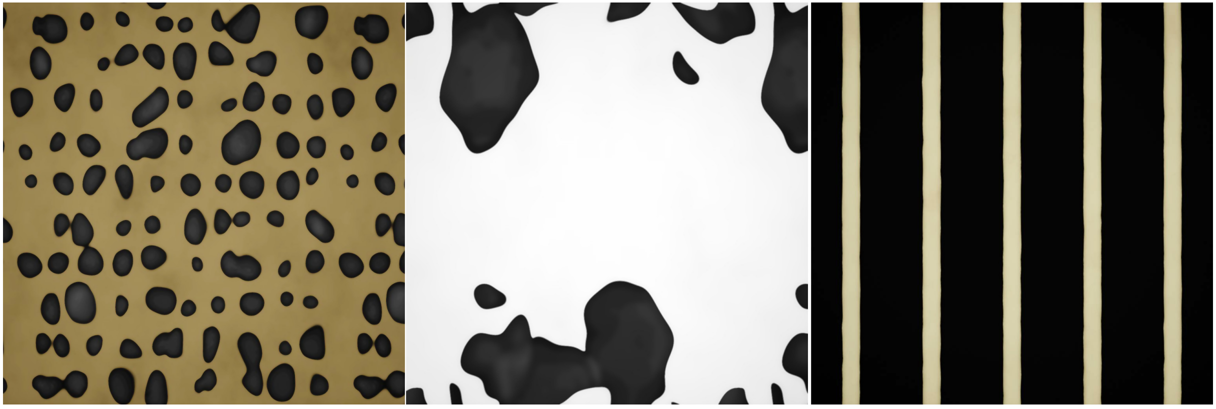
Still, given that our model is unable to reproduce only a fraction of the snakes (23.46% of our samples, see Section 3.3 for details), we believe our approach is adequate for the generation of these textures. In addition, thanks to our categorization of snake skin patterns, our model is extensible, so that the complex patterns can be resolved independently as a future work.

The paper version of this work was submitted to GRAPP 2018 – 13th International Conference on Computer Graphics Theory and Applications, and was accepted as a full paper.

6.1 Future Work

The generated textures can easily be adapted for other scaled reptilians as well, or indeed, any other animal if the normal map and height map aren't used (scales go into these two textures only). Figure 6.1 shows some preliminary results. The first two use the Spots pattern type with high values on random position and scale, and also a high value on the pattern distortion. The last one simply uses the Longitudinal Stripes pattern type. As such, the tool could be explored for usage and improvement outside the proposed domain (snakes).

Figure 6.1 – Base color textures generated using our method for different animals. From left to right: Cheetah (*Acinonyx jubatus*), cow (*Bos taurus*) and five-lined skink (*Plestiodon fasciatus*).



Source: The author.

As mentioned in our limitations above, we would like to expand our tool to take into account the complex patterns, possibly using Murray's reaction-diffusion systems. We would also like to implement a way to combine pattern types in a single texture, and to generate head and tail textures.

With certain modifications, the model could also be implemented as a shader, allowing for infinite resolution and lower memory usage. However, this approach could possibly be too GPU intensive, and will be studied in a future work.

Regarding our tool, we would like to be able to import cellular automata descriptor files directly into the tool (instead of using Matlab and importing the resulting image in Unreal Engine 4), and also allow the user to manually draw a pattern, if the model is unable to synthesize the user's desired pattern.

REFERENCES

ABEGG, A. D.; NETO, O. M. E. *Serpentes do Rio Grande do Sul*. Tapera, RS, Brazil: Livraria e Editora Werlang Ltda., 2012. ISBN 978-85-98202-44-0.

ALBUQUERQUE, S. de. *Falsa-coral (Rhinobothryum lentiginosum*. n.d. [Acesso em: 23 nov. 2017]. Disponível em: <http://www.herpetofauna.com.br/Corais.htm>.

Andrews II, M. J. *Caramel Burmese Python*. 2010. [Acesso em: 23 nov. 2017]. Disponível em: <https://commons.wikimedia.org/w/index.php?curid=9950860>.

BAUCHOT, R. *Snakes: A Natural History*. [S.l.]: Sterling, 2006. ISBN 1402731817.

BORGES-MARTINS, M. *Micrurus altirostris (Cope, 1860)*. 2007. [Acesso em: 23 nov. 2017]. Disponível em: <http://www.ufrgs.br/herpetologia/R%C3%A9pteis/Micrurus%20altirostris.htm>.

BORGES-MARTINS, M. *Phalotris lemniscatus trilineatus (Boulenger, 1889)*. 2007. [Acesso em: 23 nov. 2017]. Disponível em: <http://www.ufrgs.br/herpetologia/R%C3%A9pteis/Phalotris%20lemniscatus%20trilineatus.htm>.

BORGES-MARTINS, M. *Rhinocerophis alternatus (Duméril, Bibron and Duméril, 1854)*. 2007. [Acesso em: 23 nov. 2017]. Disponível em: <http://www.ufrgs.br/herpetologia/R%C3%A9pteis/Rhinocerophis%20alternatus.htm>.

BROOKE, J. System usability scale (sus): a quick-and-dirty method of system evaluation user information. *Reading, UK: Digital Equipment Co Ltd*, 1986.

CAMPBELL, C. *Pastel ball python*. 2005. [Acesso em: 23 nov. 2017]. Disponível em: <https://en.wikipedia.org/wiki/File:Python_Regius_Pastel.jpg>.

COCHO, G.; PéREZ-PASCUAL, R.; RIUS, J. L. Discrete systems, cell-cell interactions and color pattern of animals. i. conflicting dynamics and pattern formation. *Journal of Theoretical Biology*, Elsevier, v. 125, p. 419–435, 1987.

COCHO, G. et al. Discrete systems, cell-cell interactions and color pattern of animals ii. clonal theory and cellular automata. *Journal of Theoretical Biology*, Elsevier, v. 125, p. 437–447, 1987.

COMMONS, W. *Rainbow boa peruvian*. 2009. [Acesso em: 23 nov. 2017]. Disponível em: <https://commons.wikimedia.org/wiki/File:Rainbow_boa_peruvian.jpg>.

COMMONS, W. *KAn eyelash pit viper (Bothriechis schlegelii) at La Selva Biological Station, Sarapiqui, Costa Rica*. 2011. [Acesso em: 23 nov. 2017]. Disponível em: <https://commons.wikimedia.org/wiki/File:Bothriechis_schlegelii_(La_Selva_Biological_Station).jpg>.

COMMONS, W. *Keeled Scales on skin of Puff Adder IMG 0875*. 2012. [Acesso em: 23 nov. 2017]. Disponível em: <https://commons.wikimedia.org/wiki/File:Keeled_Scales_on_skin_of_Puff_Adder_IMG_0875.JPG>.

COMMONS, W. *Snakes Diversity*. 2013. [Acesso em: 23 nov. 2017]. Disponível em: <https://commons.wikimedia.org/wiki/File:Snakes_Diversity.jpg>.

DEGUY, S. The new age of procedural texturing. *Encyclopedia of Computer Graphics and Games*, Springer, p. 1–20, 2015.

DHILLON, D. S. et al. Interactive diffraction from biological nanostructures. In: WILEY ONLINE LIBRARY. *Computer Graphics Forum*. [S.l.], 2014. v. 33, n. 8, p. 177–188.

DI-BERNARDO, M. *Ptychophis flavovirgatus GOMES, 1915*. n.d. [Acesso em: 23 nov. 2017]. Disponível em: <http://reptile-database.reptarium.cz/species?genus=Ptychophis&species=flavovirgatus>.

EBERT, D. S. et al. *Texturing and Modeling: A Procedural Approach*. 3rd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002. ISBN 1558608486.

HENDRIKX, M. et al. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, ACM, v. 9, n. 1, p. 1, 2013.

KOOOOOLA. *UE4 - Substance designer*. 2015. [Acesso em: 23 nov. 2017]. Disponível em: <https://www.flickr.com/photos/99646627@N03/18542824813/>.

LEWIS, J. P. Algorithms for solid noise synthesis. *SIGGRAPH Comput. Graph.*, ACM, New York, NY, USA, v. 23, n. 3, p. 263–270, jul. 1989. ISSN 0097-8930. Disponível em: <http://doi.acm.org/10.1145/74334.74360>.

MANDELBROT, B. B.; PIGNONI, R. *The fractal geometry of nature*. [S.l.]: WH freeman New York, 1983. v. 173.

MARTINS, P. H. *Elapomorphus quinquelineatus. Serra do Cipó, Minas Gerais, Brazil.* 2011. [Acesso em: 23 nov. 2017]. Disponível em: <https://www.flickr.com/photos/pedrohmartins/9173238831/in/photostream/>.

MAY, P. S. *FIGURE 1 - (FPREP393PH) Adult, Encarnación, Departamento Itapúa (Paul Smith May 2010)*. 2010. [Acesso em: 23 nov. 2017]. Disponível em: <http://www.faunaparaguay.com/sibynomorphusturgidus.html>.

MILLER, G. S. P. The motion dynamics of snakes and worms. *SIGGRAPH Comput. Graph.*, ACM, New York, NY, USA, v. 22, n. 4, p. 169–173, jun. 1988. ISSN 0097-8930. Disponível em: <http://doi.acm.org/10.1145/378456.378508>.

MURRAY, J.; MYERSCOUGH, M. Pigmentation pattern formation on snakes. *Journal of Theoretical Biology*, Elsevier, v. 149, p. 339–360, 1991.

NISTRI, A.; LANZA, B. Somali boidae (genus eryx daudin 1803) and pythonidae (genus python daudin 1803) (reptilia serpentes). *Tropical Zoology*, v. 18, n. 1, 2006. ISSN 1970-9528. Disponível em: <http://www.fupress.net/index.php/tropicalzoology/article/view/120>.

NOGUEIRA, C. *Répteis Squamata do Cerrado - Apostolepis assimilis*. n.d. [Acesso em: 23 nov. 2017]. Disponível em: <http://www.ib.usp.br/~crinog/pages/Apostolepis%20assimilis%20site%20CHUNB%2023781_JPG.htm>.

NOGUEIRA, C. *Répteis Squamata do Cerrado - Philodryas aestiva*. n.d. [Acesso em: 23 nov. 2017]. Disponível em: <http://www.ib.usp.br/~crinog/pages/Philodryas%20aestiva%20CHUNB%2023795%20site_JPG.htm>.

PANAGIOTAKIS, C.; TZIRITAS, G. Snake terrestrial locomotion synthesis in 3d virtual environments. *The Visual Computer*, v. 22, n. 8, p. 562–576, 2006. ISSN 1432-2315. Disponível em: <http://dx.doi.org/10.1007/s00371-006-0035-1>.

PARBERRY, I. Amortized noise. *Journal of Computer Graphics Techniques (JCGT)*, v. 3, n. 2, p. 31–47, June 2014. ISSN 2331-7418. Disponível em: <http://jcgt.org/published/0003/02/02/>.

PASSOS, M. de A. *Erythrolamprus miliaris (Linnaeus, 1758)*. 2013. [Acesso em: 23 nov. 2017]. Disponível em: <http://michelpassosherpetolife.blogspot.com.br/2013/03/liophis-miliaris-orinus-griffin-1916.html>.

PERLIN, K. An image synthesizer. *SIGGRAPH Comput. Graph.*, ACM, New York, NY, USA, v. 19, n. 3, p. 287–296, jul. 1985. ISSN 0097-8930. Disponível em: <http://doi.acm.org/10.1145/325165.325247>.

PERLIN, K. Improving noise. In: *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: ACM, 2002. (SIGGRAPH '02), p. 681–682. ISBN 1-58113-521-1. Disponível em: <http://doi.acm.org/10.1145/566570.566636>.

PHARR, M.; JAKOB, W.; HUMPHREYS, G. *Physically based rendering: From theory to implementation*. [S.l.]: Morgan Kaufmann, 2016.

RADA, T. *Lampropeltis triangulum triangulum (Eastern Milksnake)*. 2008. [Acesso em: 23 nov. 2017]. Disponível em: <https://commons.wikimedia.org/wiki/File:Tennessee_milksnake.jpg>.

Regency Enterprises; Protozoa Pictures; Paramount Pictures. *Noah (2014) film*. 2014. [Acesso em: 23 nov. 2017]. Disponível em: <http://www.moviedeskback.com/2013/11/19-hd-screencaps-noah-2014-aronofsky.html>.

RHOADES, J. et al. Real-time procedural textures. In: *Proceedings of the 1992 Symposium on Interactive 3D Graphics*. New York, NY, USA: ACM, 1992. (I3D '92), p. 95–100. ISBN 0-89791-467-8. Disponível em: <http://doi.acm.org/10.1145/147156.147171>.

SAWAYA, R. *Philodryas olfersii (Lichtenstein, 1823)*. n.d. [Acesso em: 23 nov. 2017]. Disponível em: <http://serpentesbrasileiras2.blogspot.com.br/2011/04/philodryas-olfersii-lichtenstein-1823.html>.

SMITH, P. *BICOLOURED MUSSURANA Mussurana bicolor*. 2015. [Acesso em: 23 nov. 2017]. Disponível em: <http://www.faunaparaguay.com/mussuranabicolor.html>.

STUART, J. N. *Prairie Rattlesnake (Crotalus viridis), adult*. 2011. [Acesso em: 23 nov. 2017]. Disponível em: <https://www.flickr.com/photos/stuartwildlife/6089419013/in/photostream/>.

TIMM, C. D. *Falsa-coral (Oxyrhopus rhombifer)*. 2014. [Acesso em: 23 nov. 2017]. Disponível em: <https://www.flickr.com/photos/cdtimm/16053639749/>.

TURING, A. M. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, The Royal Society, v. 237, n. 641, p. 37–72, 1952.

UETZ, P.; FREED, P.; (EDS.), J. H. *The Reptile Database*. 2016. [Acesso em: 23 nov. 2017]. Disponível em: <http://www.reptile-database.org/>.

WIKIPEDIA. *Serpent (symbolism) — Wikipedia, The Free Encyclopedia*. 2016. [Acesso em: 23 nov. 2017]. Disponível em: <https://en.wikipedia.org/w/index.php?title=Serpent_(symbolism)>.

APPENDIX A — USER TEST RESULTS

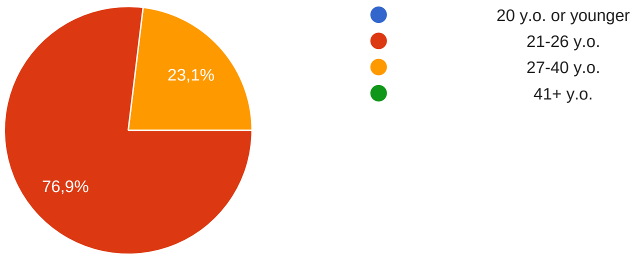
Tool user test results.

Questionnaire – snake skin textures generator

13 answers

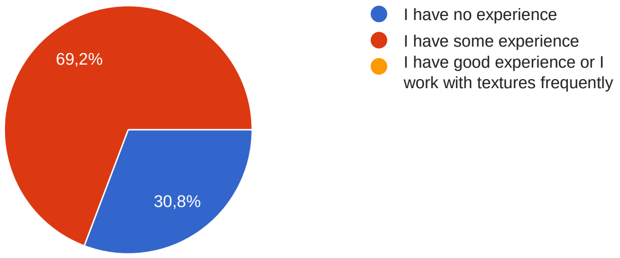
How old are you?

13 answers



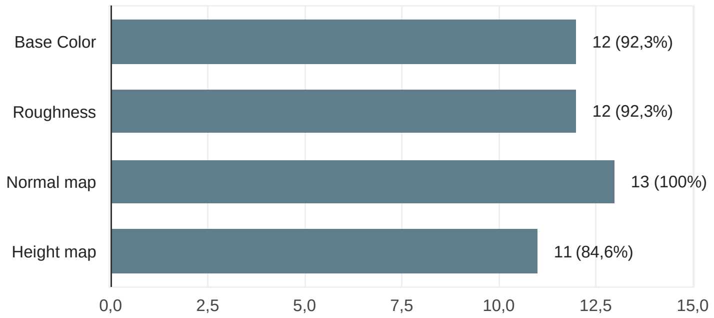
How do you consider yourself as a texture artist?

13 answers



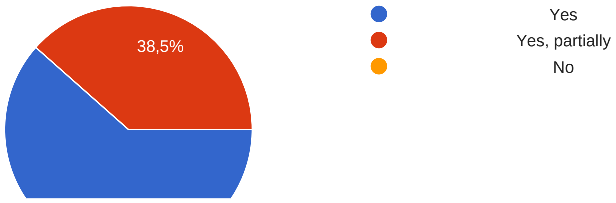
Regarding your familiarity with the PBR (physically based rendering) system, did you knew (before using this tool) what each of these textures represent? (check for "yes")

13 answers



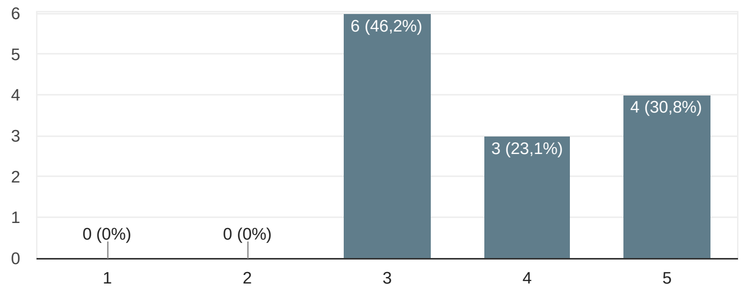
Did you manage to reproduce the textures of the elapomorphus quinquelineatus?

13 answers



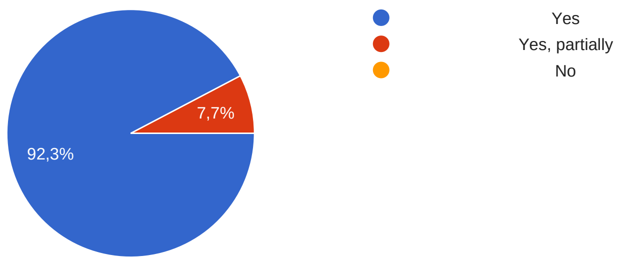
How easy was it to reproduce the elapomorphus quinquelineatus? (1 for very difficult, 5 for very easy)

13 answers



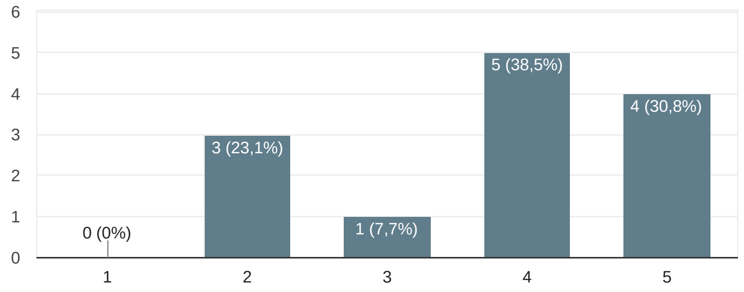
Did you manage to reproduce the coral’s texture?

13 answers



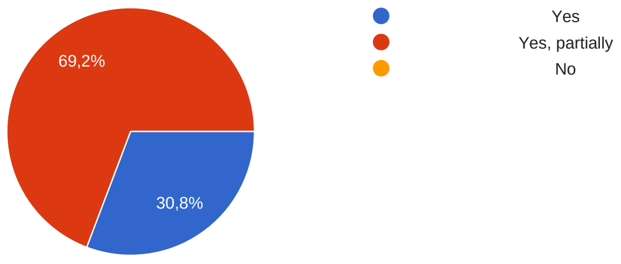
How easy was it to reproduce the coral? (1 for very difficult, 5 for very easy)

13 answers



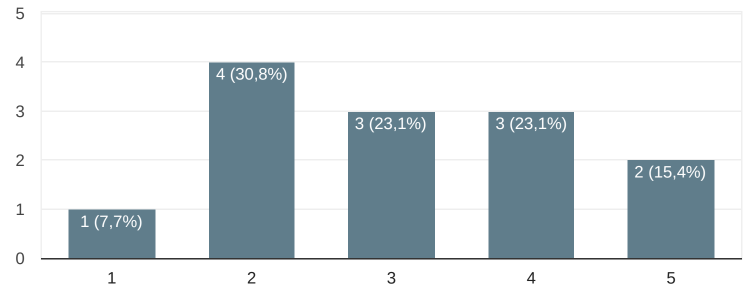
Did you manage to reproduce the snake of your choice?

13 answers



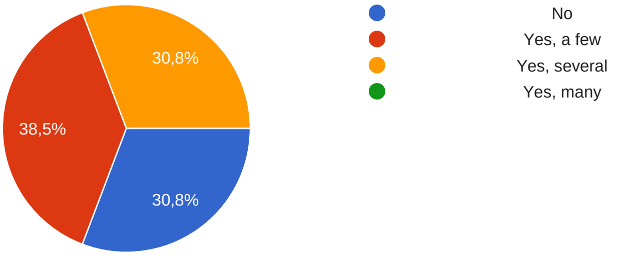
How easy was it to reproduce the snake of your choice? (1 for very difficult, 5 for very easy)

13 answers



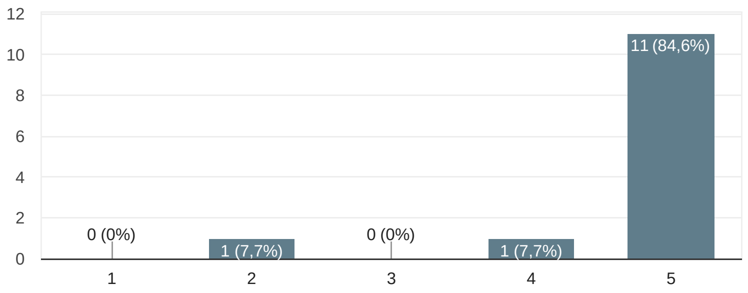
Is there any kind of pattern that is impossible to reproduce with the tool?

13 answers



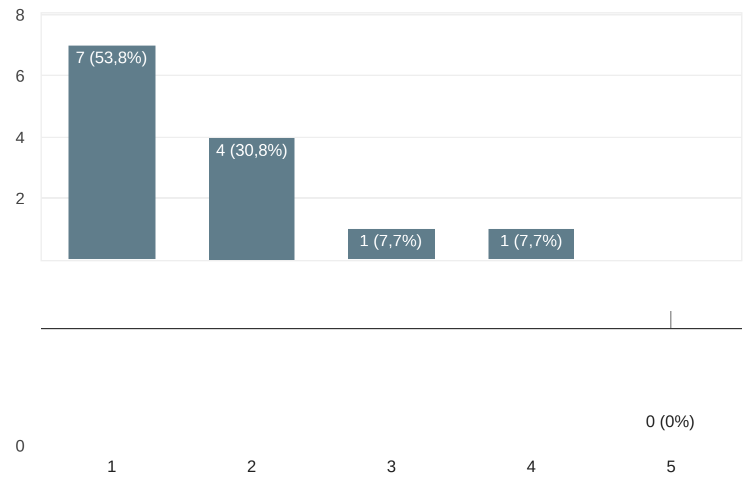
I would like to use this tool in the future to create snake textures. (1 for strongly disagree, 5 for strongly agree)

13 answers



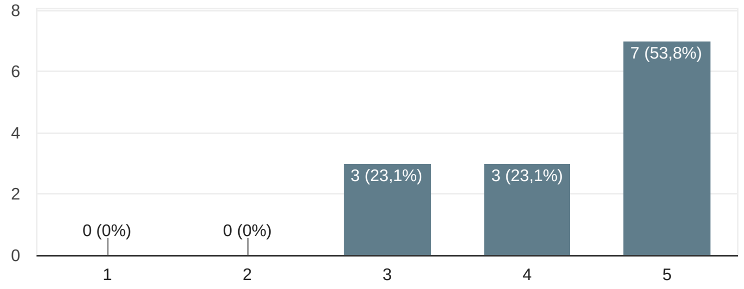
I found the tool unnecessarily complex. (1 for strongly disagree, 5 for strongly agree)

13 answers



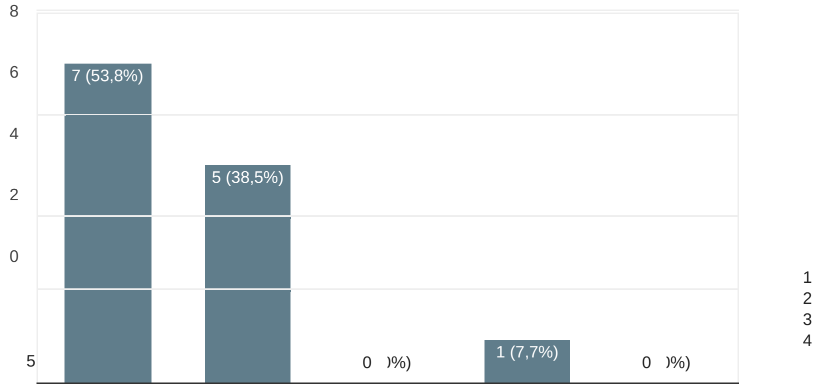
I found the tool easy to use. (1 for strongly disagree, 5 for strongly agree)

13 answers



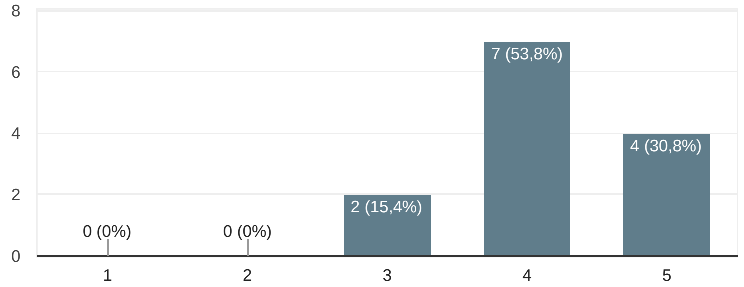
I think I would need assistance from someone more technical to use this tool. (1 for strongly disagree, 5 for strongly agree)

13 answers



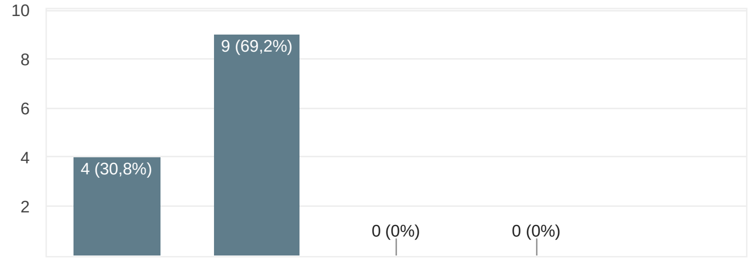
I think the tool's many functionalities are well integrated. (1 for strongly disagree, 5 for strongly agree)

13 answers



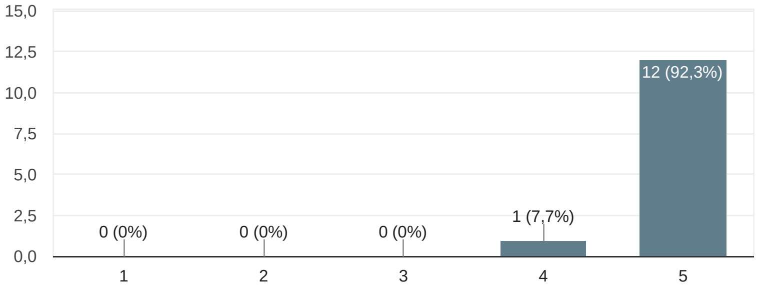
I think the tool had many inconsistencies. (1 for strongly disagree, 5 for strongly agree)

13 answers



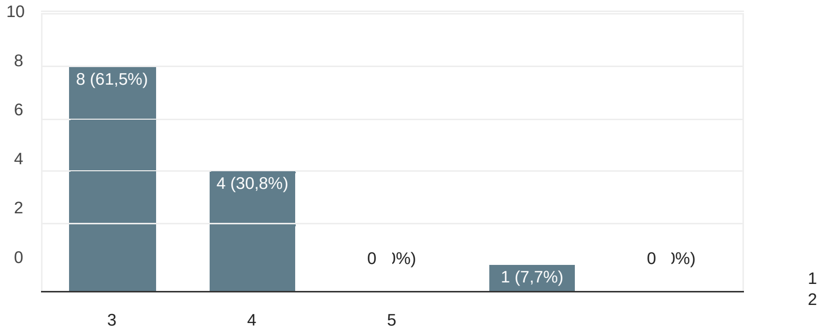
I believe most artists would quickly learn how to use this tool. (1 for strongly disagree, 5 for strongly agree)

13 answers



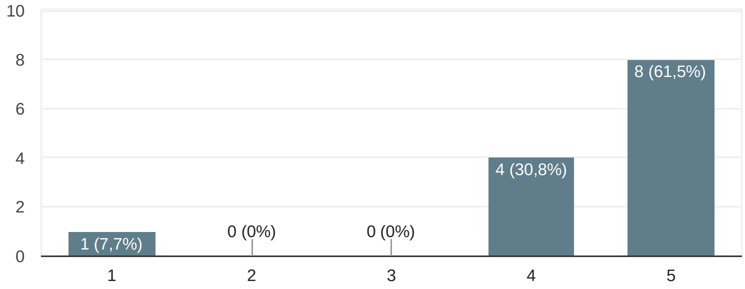
I found the tool confusing to use. (1 for strongly disagree, 5 for strongly agree)

13 answers



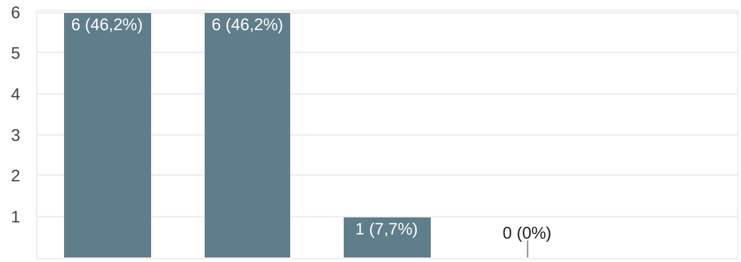
I felt confident when using the tool. (1 for strongly disagree, 5 for strongly agree)

13 answers



I needed to learn many things before being able to use the tool. (1 for strongly disagree, 5 for strongly agree)

13 answers



Any other comments?

9 answers

Very nice!!!

The custom pattern function was apparently bugged and didn’t change the snake’s texture.

I was a bit lost for like 3 minutes at the beginning to learn what functions it had but I found it interesting, I think it will be quite useful.

It would be nice to be able to import (or select) HDRI images to use as skysphere, to change lighting conditions. The standard lighting is fixed and I’m not certain what it would look like under other lighting conditions.

cool

Very nice. I really liked the prototype’s idea and I think it has great potential, it would be nice to expand to other animals besides snakes.

CTRL+Z plz, also button to reset the view. Maybe it was needed to add Spots only on the upper part of the snake.

Only with basic texturing knowledge it is possible to reproduce many textures without any problem because the program is simple and intuitive, even though it has lots of features. GG,well played!

I tried to create a snake that had brown back and yellow belly, there was an orange line separating these parts. I didn’t manage to do it, maybe add an option to change the belly’s color. Also would be nice to be able to change each color in longitudinal pattern type, just like you can with the transversal one.

APPENDIX B — SNAKES INFORMATION VISUALIZATION

Paper on snakes information visualization, that will be published on SIBGRAPI ’17.

Multidimensional Information Visualization on Snakes Characteristics

Jefferson Magalhães Pinheiro
Instituto de Informática, Universidade Federal do Rio Grande do Sul;
Faculdade de Informática, Centro Universitário Ritter dos Reis – UniRitter
Email: jeffersonmpinheiro@gmail.com

***Abstract*—This paper describes how information visualization techniques can be used to find out correlations between certain characteristics on serpents, including their appearance (skin pattern), what it feeds on, dentition type, activity time, habitat, subjugation method and taxonomy. Data are obtained from serpent species present in southern Brazil.**

I. Introduction and Motivation

Snakes (reptiles of the *Serpentes* suborder) are numerous in species. They are present on every continent, except Antarctica and some islands (notably Ireland, Iceland, Greenland, Hawaii and New Zealand) [1]. Some aquatic species live in the Indian and Pacific oceans as well.

Given how numerous and widespread they are, it is natural that snakes are a well-studied subject. The amount of data available about them is huge and easily found – The Reptile Database [2] being a good example. However, we do not know of any work that uses visualization techniques to find out correlations between certain characteristics in snakes.

Some of these characteristics are known by biologists to be related with another. For example, all snakes with solenoglyph dentition are venomous, and all vipers (*Viperidae* family) have keeled scales.

However, certain correlations are not immediately obvious. For example, there are certain snake species (such as the *Calamodontophis paucidens* and the *Tomodon dorsatus*) that are semi-venomous, however, they feed only on small mollusks, which they swallow alive and shouldn't need any venom.

It is also desirable to find out correlations between the snake's skin pattern, its habitat, activity time, what it feeds on, and other variables.

In order to identify these correlations, data visualization methods are desirable.

II. Related work

The Snake Database [3] contains data we could use, however, very few characteristics are catalogued: only common names, taxonomy, toxicity, whether the snake uses the constriction subjugation method, and maximum length.

The Reptile Database's [2] data can also be downloaded; however, it too includes very few data relevant for this study.

We were unable to find any other database on snakes, or any other study that uses visualization techniques for snakes' characteristics.

III. Data

The first step in our visualization is to acquire data. Most resources present information about snakes verbally (generally, books), not on a data table. There are over 3600 snake species catalogued on The Reptile Database (as of July 2017) [2]. It would be tremendous work to assemble data about all these species. Thus, we decided to limit the scope geographically.

In the state of Rio Grande do Sul, southern Brazil, there are dozens of snake species, at least 81 of which have been identified and catalogued in the book *Serpentes do Rio Grande do Sul* [4]. In the aforementioned book, these 81 species have been described as accurately as possible, verbally.

All snake families are present in this region, with the exception of *Pythonidae*. Thus, we have decided to limit the scope to this specific region, using information available from the book.

The first step then is to put all available information from the book into a data table. Because the book's description of each species is verbal, some characteristics had to be simplified, or clustered, so that data analysis is easier.

A. Diet

The first characteristic that had to be clustered was their diet. Verbal description found on the book typically describes what families or orders of animals the snake eats, such as "small lizards, amphisbaenians, birds, and snails". If we included each of these as separate strings, there would be dozens of fields, however, many of them are related. For example, most snakes that eat toads would also eat caecilians, which are both amphibians.

Therefore, we divided diet on the following categories.

- **Amphibians**: Incorporates all amphibians, such as frogs, toads, salamanders and caecilians.
- **Reptiles**: All reptiles have been condensed into this category, except snakes (which have their own category).
- **Fishes**: Self-explanatory.
- **Arthropods**: Self-explanatory.
- **Birds**: Self-explanatory.
- **Eggs**: Bird eggs.
- **Snakes**: This snake species eats other snakes, possibly its own species as well.
- **Rodents**: Small rodents, such as rats.
- **Mammals**: Small mammals, such as rabbits, capybaras, and possibly larger mammals.

- **mollusks**: Typically, snails.
- **Worms**: Typically, earthworms.

Furthermore, these categories have been assigned into two groups: *Large* and *Small*. *Small* animals include amphibians, arthropods, mollusks and worms, while *Large* animals include the opposite. Then, a field was created, that says whether a snake feeds only on *Large* animals, only on *Small* animals, or both *Small and Large* animals.

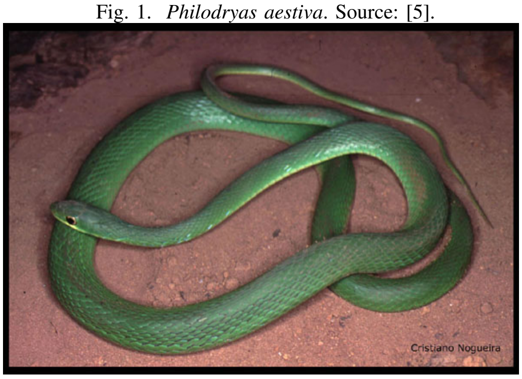
It is also worth mentioning that all snakes are carnivorous.

B. Skin Pattern

With the help of a herpetologist, we have divided the snakes skin pattern in the following five categories. Note that we only considered the dorsal (visible) side of the snake, because it is difficult to find pictures of their ventral side (underside), and also because it typically not visible. We also did not take into consideration the head or tail pigmentation, for simplification purposes.

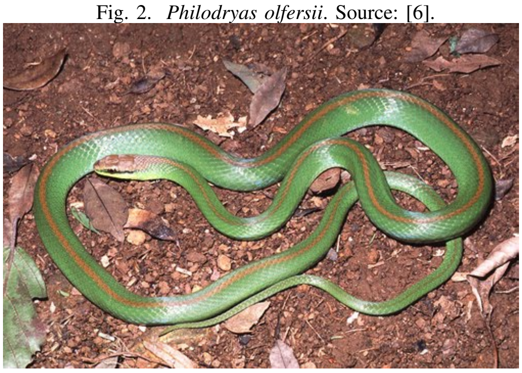
- **No Pattern**. The snake shows no pigmentation pattern, consisting only of a single solid color (see figure 1).

Fig. 1. *Philodryas aestiva*. Source: [5].



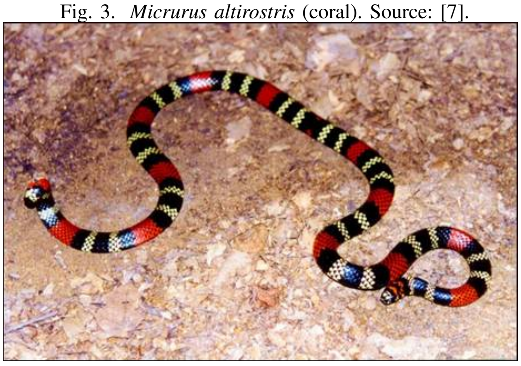
- **Longitudinal Stripes**. The snake shows one or more stripes aligned along its body, that go from neck to tail or from head to tail (see figure 2).

Fig. 2. *Philodryas olfersii*. Source: [6].



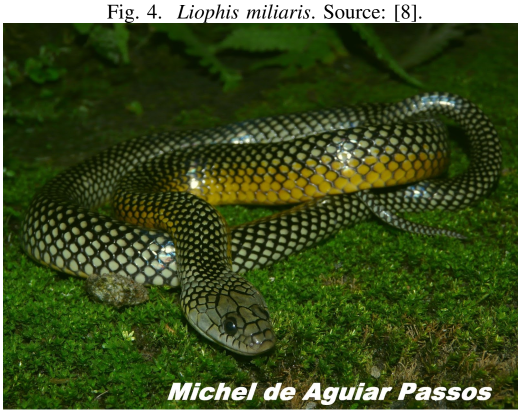
- **Transversal Stripes**. The snake shows many stripes aligned perpendicularly to its body axis, also known as "rings". These rings may be in two alternating colors, or may be in multiple colors. Typical examples include corals and false corals (see figure 3).

Fig. 3. *Micrurus altirostris* (coral). Source: [7].



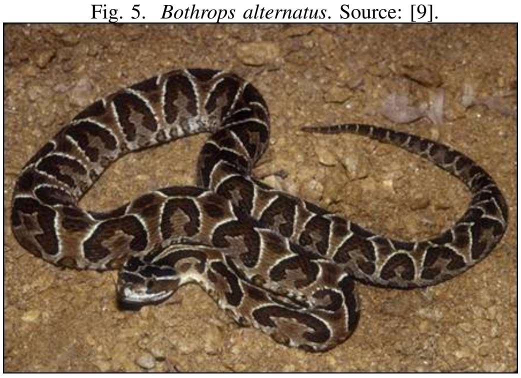
- **Spots**. The snake shows regular (similar) shapes on its body, often elliptical (see figure 4).

Fig. 4. *Liophis miliaris*. Source: [8].



- **Complex**. When the pattern is neither of the above, it is categorized as "Complex". Typically, this includes intricate camouflage patterns, that would be even hard to describe verbally, or numerous different shapes on its body (see figure 5).

Fig. 5. *Bothrops alternatus*. Source: [9].



Then, looking at pictures of each snake species, we assigned a single skin pattern category to it. If a snake presents two skin patterns (for example, when a snake's pattern has both longitudinal stripes *and* spots), we assigned the most visible one.

C. Other data

Other data that have been extracted from the book and added to the data table are:

- **Taxonomy**: Each snake is categorized by its taxonomic family, genus and species.
- **Teeth type**: there are four teeth types available in snakes: aglyph, opistoglyph, proteroglyph, and solenoglyph. Snakes with aglyph dentition are not venomous; opistoglyph and proteroglyph are semi-venomous, and solenoglyph are venomous. Each species has one specific dentition type.
- **Activity time**: period of the day that the snake is active. May be either diurnal, nocturnal, or both.
- **Habitat**: Where the snake lives or is active. May be aquatic (in or near lakes and rivers), arboreal (on trees), fossorial (caverns, under foliage, etc.) or terrestrial (on land). Each species may have multiple habitats assigned to it.
- **Subjugation type**: How the snake subjugates its prey. May be swallow alive (the snake swallows its prey whole and alive, without killing or weakening it first), by constriction, or envenomation. Each species may have multiple subjugation types.
- **Size**: Average or maximum size of an adult snake, in centimeters.

Data that are unavailable in the book were left blank (in the case of numeric entries) or assigned the value "Unknown" (for categorical entries).

A visualization was then created to view all these data at once.

IV. Visualization techniques

All of these data have been added to a single linked-view visualization with multiple charts (see figure 6). We have chosen stacked bar charts for wide characteristics (Feeds On and Habitat), and pie charts for narrow characteristics (all others). Tooltips on mouse over reveals information on what each color represents.

One characteristic was originally wide, Subjugation Type, which could be any combination of Swallow Only, Constriction and/or Envenomation. We made it narrow by creating the following fields: Swallow Alive Only, Constriction Only, Envenomation Only, Swallow Alive or Envenomation, and Constriction or Envenomation. Other combinations were not needed because no snake uses another combination of subjugation method. This was done to make filtering easier.

Clicking on any chart will filter all other charts in the visualization. Thus, by selecting multiple filters, it is possible to find correlations between different characteristics.

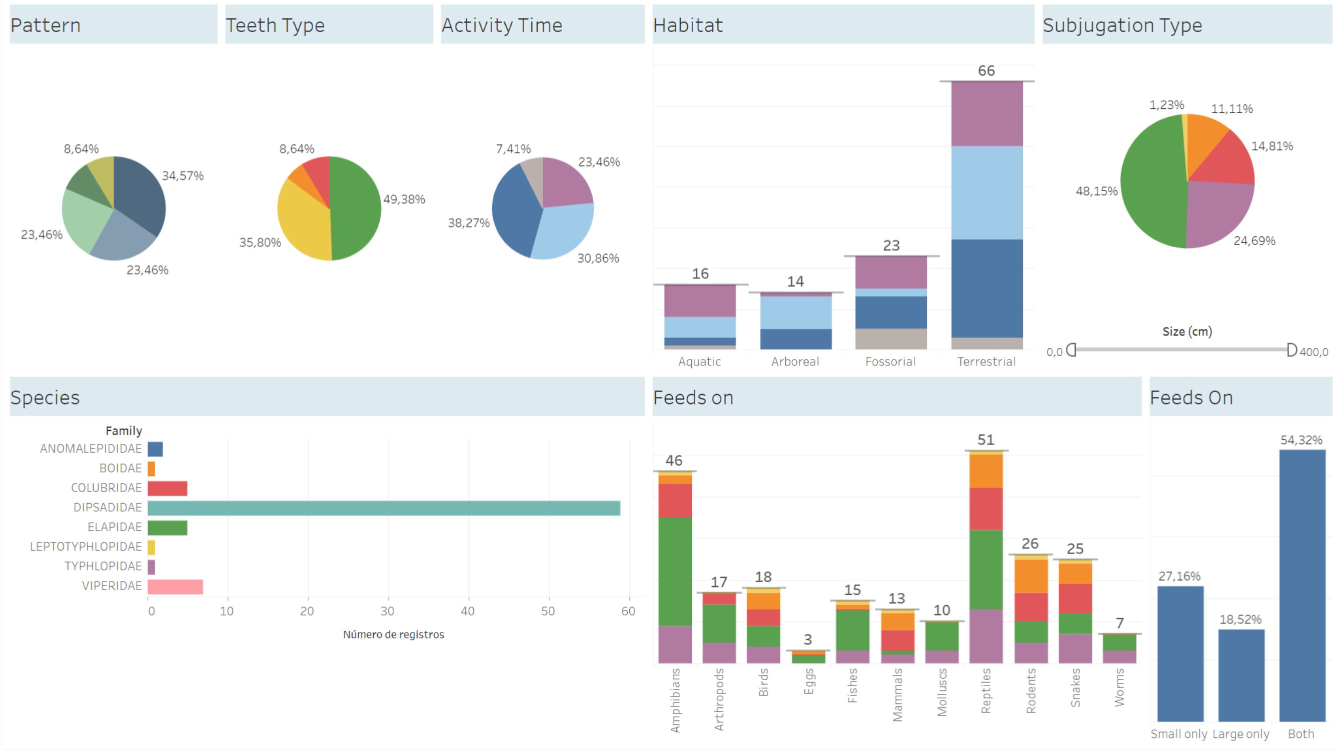
V. Results

Before applying any filters, certain characteristics are evident on Rio Grande do Sul snakes as a whole. Most of them have terrestrial habitat (81.4%), roughly half is non-venomous, roughly half swallow their prey alive, 56% prey on amphibians and 62% on reptiles.

After applying several filters in different combinations, we have come to the following insights. See the references for a snapshot of the corresponding visualization, to see which filters were applied.

1) Most snakes (85.72%) with a Transversal pattern are venomous [10].
2) All snakes with the Spots pattern may swallow their prey alive, and 25% may also use venom [11].
3) All snakes who use venom, constriction, or both to subjugate their prey have either pure nocturnal (72.73%) or diurnal/nocturnal (27.27%) habits, but never purely diurnal [12].
4) Snakes who only swallow their prey alive tend to have more diurnal habits (43.59%) [13] than the others (19.05%) [14].
5) The average size of snakes with both diurnal and nocturnal habits is 87.56 cm, while those with purely diurnal or nocturnal habits measure an average of 111.5 cm – a difference of about 21% (reference: tooltip on chart "Activity Time").
6) Venomous snakes (both proteroglyph and solenoglyph dentition) always use venom to subjugate their prey; they never try constriction or swallow alive [15].
7) 68.97% of snakes with opistoglyph (semi-venomous) dentition may swallow their prey alive as subjugation method, and 31.03% may use constriction. But all of them may also use venom as well [16].
8) 97.5% of snakes with aglyph (non-venomous) dentition swallow their prey alive (and feed on small animals), while 2.5% use constriction (and may feed on large animals as well) [17]. However, this 2.5% represents only a single species – the *Eunectes notaeus*, a large constrictor boa.
9) Snakes who feed on *Large*, or both *Small and Large* animals, tend to be venomous (61.02%) [18]. However, this is another regional characteristic, as there is only one boa species in the region (boas are large and non-venomous snakes).
10) Of the snakes who only swallow their prey alive as subjugation method, 41.03% feed exclusively on *Small* animals, while 58.97% feed on both *Small and Large* animals [19]. Naturally, the average size of these two groups is significantly different: 55.44 cm for the first group, 106.14 cm for the second.
11) All snakes from the *Viperidae* family have complex pattern and exclusively terrestrial habitat; 85.71% have only nocturnal habits (14.29% both nocturnal and diurnal), and all of them prey on rodents (as well as other small and large animals, to a lesser extent) [20].
12) Snakes from the *Dipsadidae* family prey mostly on reptiles and amphibians. In addition, roughly half has aglyph (non-venomous) dentition and half has opisto-

Fig. 6. The visualization.



glyph (semi-venomous) dentition [21].

We asked an herpetologist if any of these correlations were already known. She said insights number 1, 6, 9, 10 11 and 12 were. All others represent new knowledge.

VI. Conclusions

The visualization allowed us to discover information that would be otherwise difficult or even impossible to determine by just looking at the data table. Although the data set is a bit restricting – there are no pythons and only one boa species, for example – we believe much more could be learned if a larger data set was available. Even with the current data set, we are certain much more could be learned by applying different filters.

The herpetologist who worked with us was also excited about the possibilities of this tool, particularly with a larger database.

Of course, we must also take into consideration the reliability of the source data. There are 81 species catalogued in *Serpentes do Rio Grande do Sul*, but we cannot assume that all of these species have been thoroughly studied. It is possible that certain fields have incorrect values. For example, some snake species with proteroglyph or solenoglyph dentition may use the constriction subjugation method after all, this behavior just wasn't observed yet.

For future work, we would like to extend the source data to include snakes from a larger region. Another possibility would be to include several samples from all over the world, so that the analysis represents a global average, not a regional average. Eventually cataloging all snake species in the world would be beneficial as well.

Another update would be to include more data in the data table and visualization, such as skin color(s), relative strength of venom toxicity, which toxins are contained in the venom, conservation status, region where they occur, period of the year that they reproduce and how many baby snakes are spawned, and so on.

The visualization and data table are available for viewing and download at [22]. Tableau version 10.3 was used to create the visualization. Permission is granted to freely use, distribute, modify and expand the visualization, as long as you cite this paper.

Acknowledgments

The author would like to thank Marluci Müller Rebelato, of the Herpetology Institute at Universidade Federal do Rio Grande do Sul, for her advice on snakes biology and feedback provided; and Carla Maria Dal Sasso Freitas, of the Informatics Institute at Universidade Federal do Rio Grande do Sul, for her advice on information visualization.

References

[1] R. Bauchot, *Snakes: A Natural History*. Sterling, 2006.

[2] P. Uetz, P. Freed, and J. H. (eds.). (2016) The reptile database. [Online]. Available: http://www.reptile-database.org/

[3] S. Steinhoff. (2017) The snake database. [Online]. Available: http://snakedatabase.org/

[4] A. D. Abegg and O. M. Entiauspe Neto, *Serpentes do Rio Grande do Sul*. Tapera, RS, Brazil: Livraria e Editora Werlang Ltda., 2012.

[5] C. Nogueira. Répteis squamata do cerrado - philodryas aestiva. [Online]. Available: http://www.ib.usp.br/~crinog/pages/Philodryas%20aestiva%20CHUNB%2023795%20site_JPG.htm

[6] R. Sawaya. Philodryas olfersii (lichtenstein, 1823). [Online]. Available: http://serpentesbrasileiras2.blogspot.com.br/2011/04/philodryas-olfersii-lichtenstein-1823.html

[7] M. Borges-Martins. (2007) Micrurus altirostris (cope, 1860). [Online]. Available: http://www.ufrgs.br/herpetologia/R%C3%A9pteis/Micrurus%20altirostris.htm

[8] M. de Aguiar Passos. (2013) Erythrolamprus miliaris (linnaeus, 1758). [Online]. Available: http://michelpassosherpetolife.blogspot.com.br/2013/03/liophis-miliaris-orinus-griffin-1916.html

[9] M. Borges-Martins. (2007) Rhinocerophis alternatus (duméril, bibron and duméril, 1854). [Online]. Available: http://www.ufrgs.br/herpetologia/R%C3%A9pteis/Rhinocerophis%20alternatus.htm

[10] J. M. Pinheiro. (2017) Rio grande do sul snakes. [Online]. Available: https://public.tableau.com/shared/QJ3TNWPQY

[11] ——. (2017) Rio grande do sul snakes. [Online]. Available: https://public.tableau.com/shared/SX9G933WX

[12] ——. (2017) Rio grande do sul snakes. [Online]. Available: https://public.tableau.com/shared/P9WC84MQ5

[13] ——. (2017) Rio grande do sul snakes. [Online]. Available: https://public.tableau.com/shared/TNSBH6K72

[14] ——. (2017) Rio grande do sul snakes. [Online]. Available: https://public.tableau.com/shared/K3FTSN6MG

[15] ——. (2017) Rio grande do sul snakes. [Online]. Available: https://public.tableau.com/shared/BKPDSB8GG

[16] ——. (2017) Rio grande do sul snakes. [Online]. Available: https://public.tableau.com/shared/MJ82YGBSD

[17] ——. (2017) Rio grande do sul snakes. [Online]. Available: https://public.tableau.com/shared/T7TMF5TN7

[18] ——. (2017) Rio grande do sul snakes. [Online]. Available: https://public.tableau.com/shared/TMMW5NFQR

[19] ——. (2017) Rio grande do sul snakes. [Online]. Available: https://public.tableau.com/shared/D59B496DY

[20] ——. (2017) Rio grande do sul snakes. [Online]. Available: https://public.tableau.com/shared/QC9T9XKK8

[21] ——. (2017) Rio grande do sul snakes. [Online]. Available: https://public.tableau.com/shared/46RRQK9JN

[22] ——. (2017) Rio grande do sul snakes. [Online]. Available: https://public.tableau.com/views/Snakes_0/Dashboard

APPENDIX C — HERPETOLOGIST TEST RESULTS

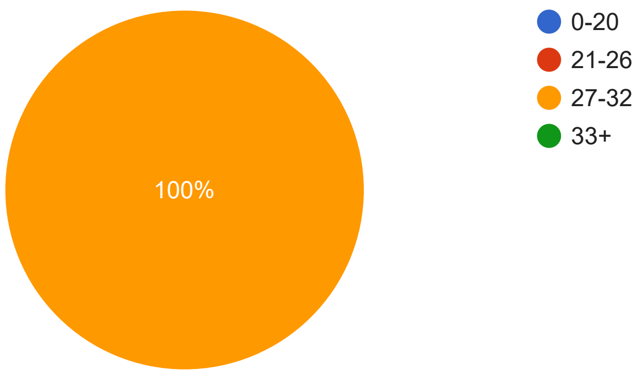
System usability scale and other questions regarding the tool, answered by an herpetologist.

Herpetologist review - snake skin textures generator

1 resposta

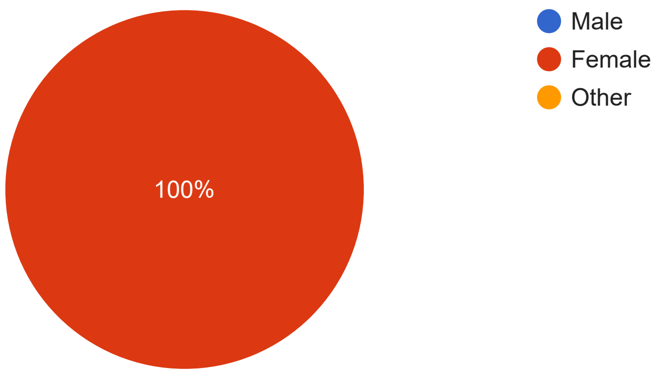
How old are you?

1 resposta



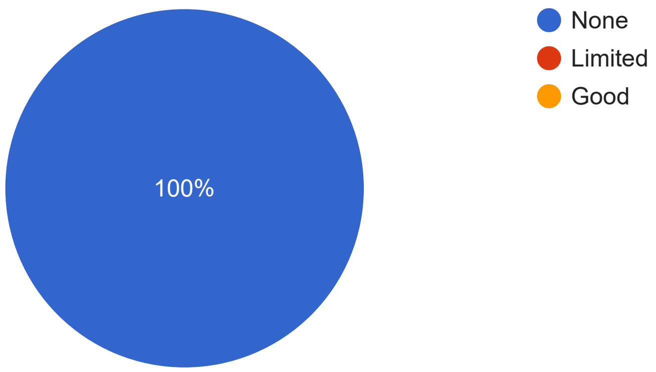
What is your gender?

1 resposta



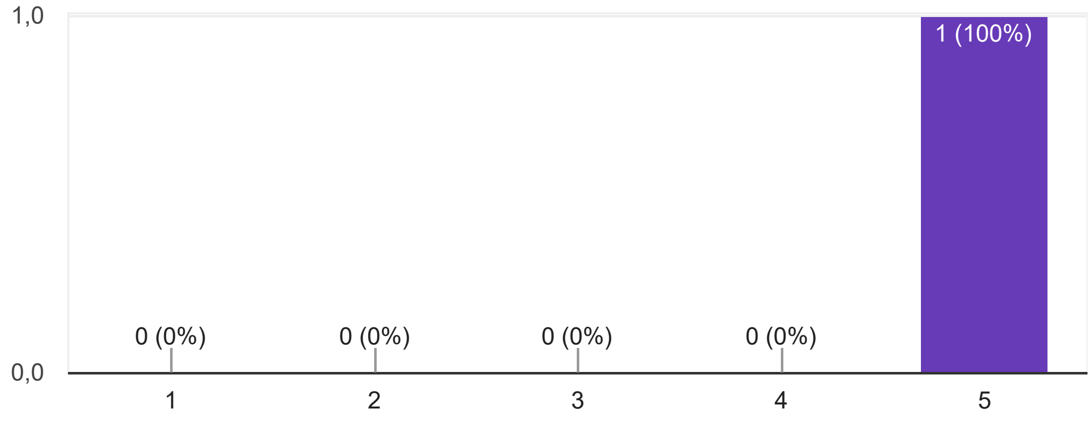
What is your experience with creating textures for computer graphics applications, such as games and computer animations?

1 resposta



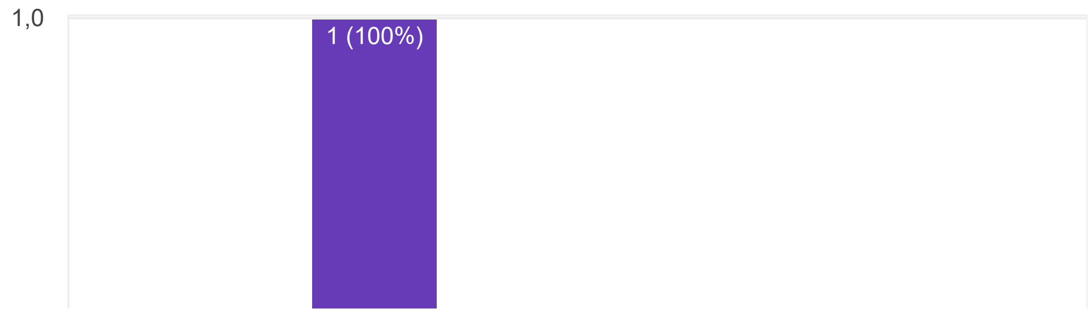
As an herpetologist, I would like to use this tool in the future for uses such as education, research, communicating a snake's appearance with colleagues, and others. (1: strongly disagree, 5: strongly agree)

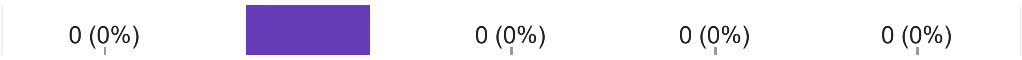
1 resposta



I think that the tool is unnecessarily complex.

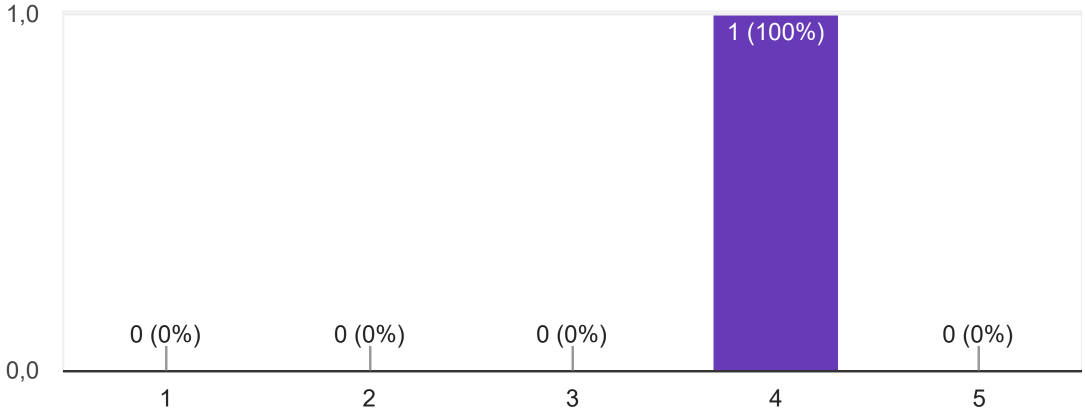
1 resposta





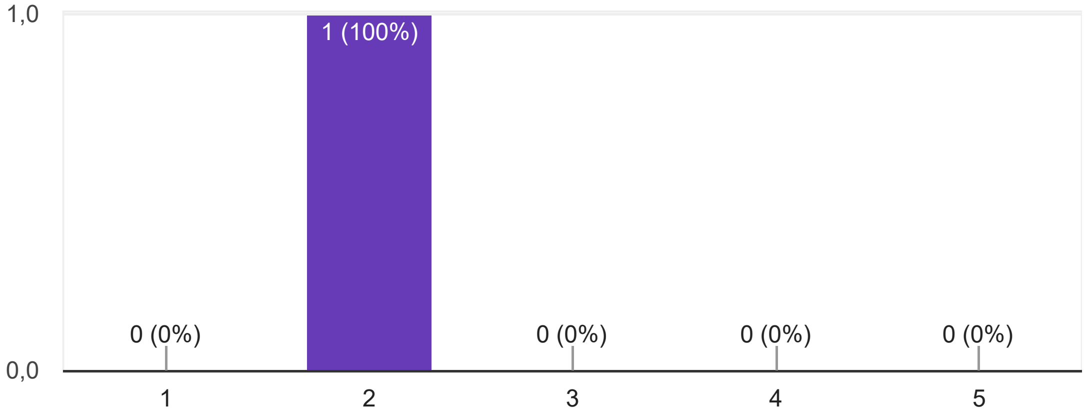
I think that tool is easy to use.

1 resposta



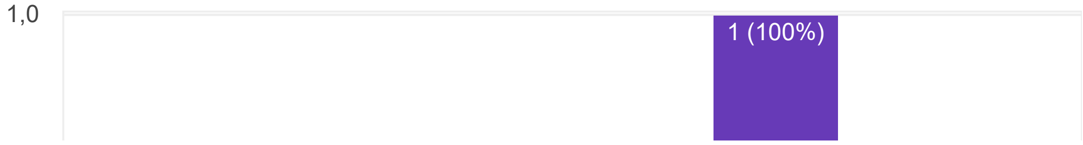
I think I would need assistance from a technician to use this tool.

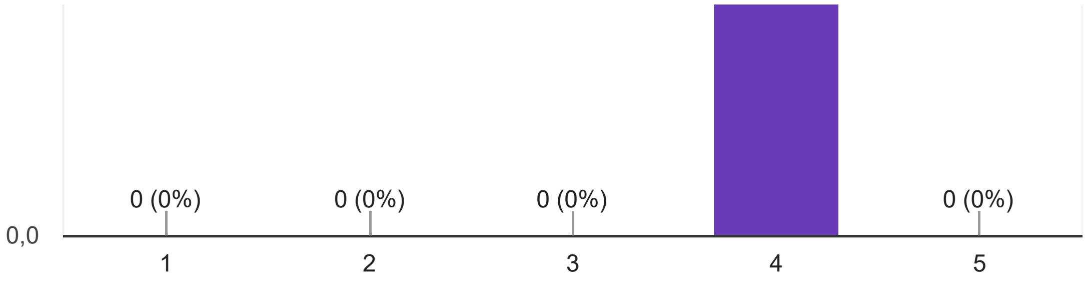
1 resposta



I think the tool's many functionalities are well integrated.

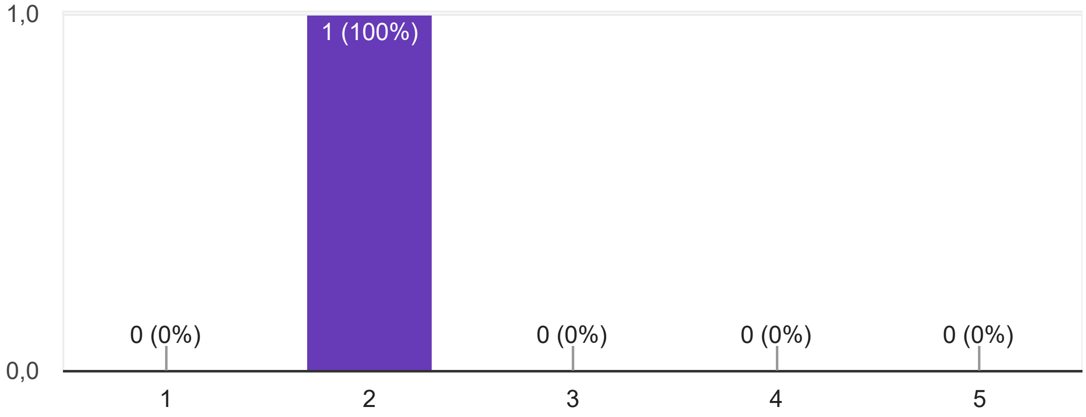
1 resposta





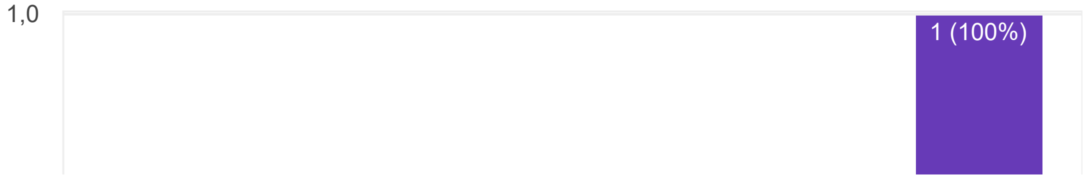
I think the tool has many inconsistencies.

1 resposta



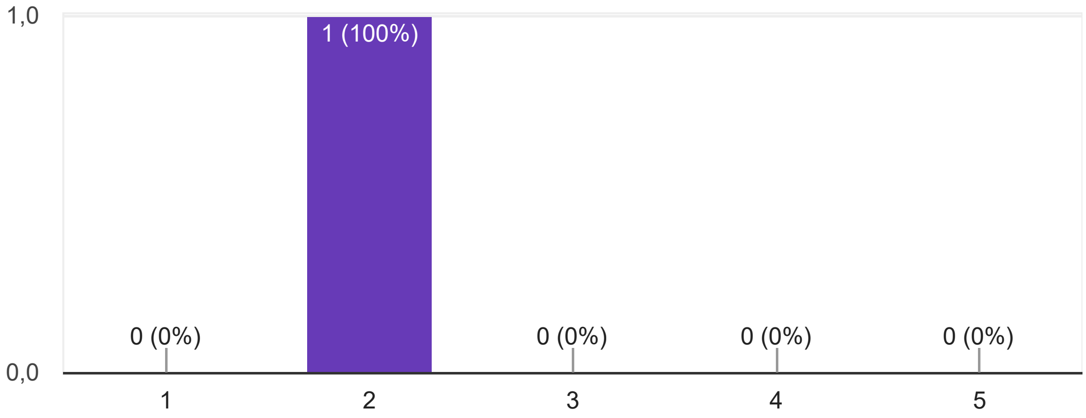
I believe most herpetologists would quickly learn how to use this tool.

1 resposta



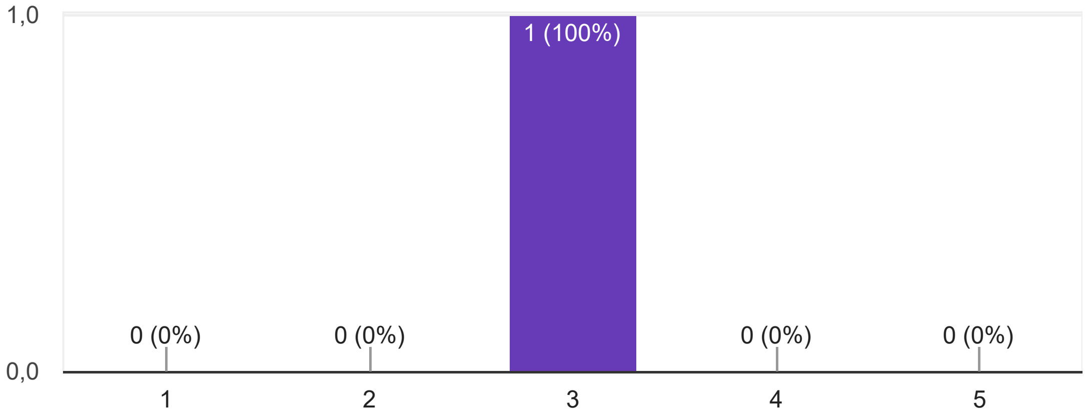
I found the tool confusing to use.

1 resposta



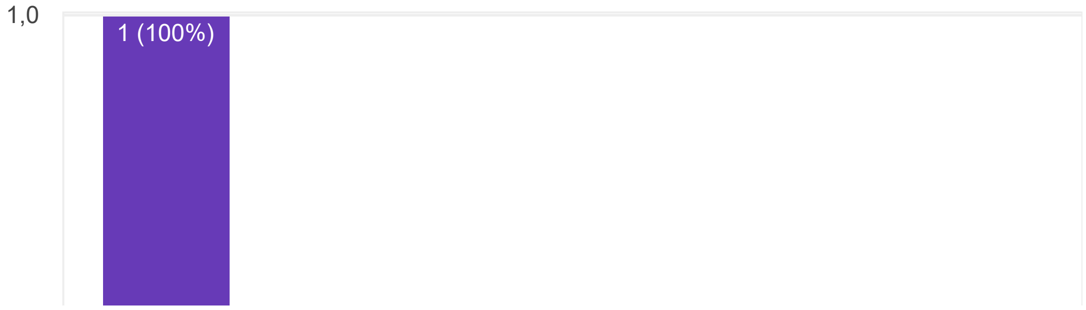
I felt confident when using the tool.

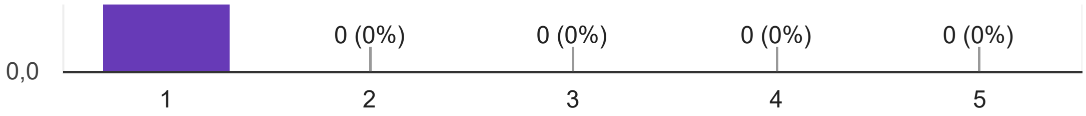
1 resposta



I had to learn many things before being able to use this tool.

1 resposta





Would this tool be useful for your work or research as an herpetologist? How? What tasks would you accomplish with it?

1 resposta

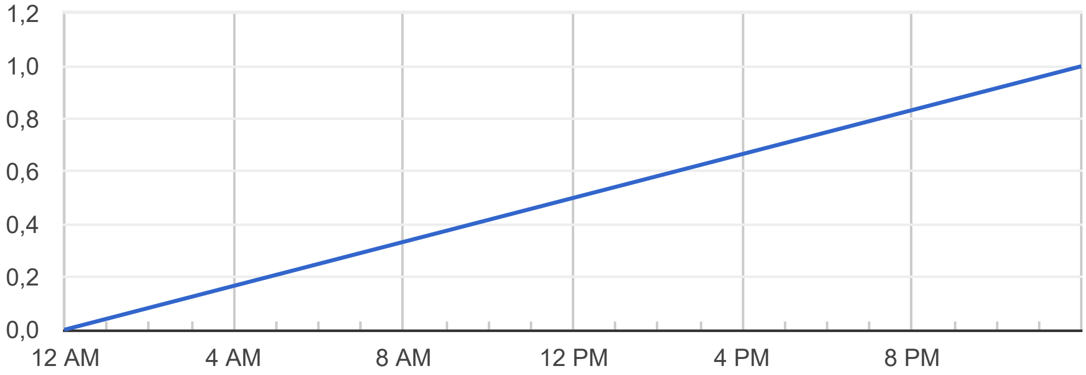
This tool can be useful in taxonomic works of new species description and revision of species complexes, through the graphic representation mainly of colors and patterns that differ between species. In addition, as the refinement of the tool, it may be possible to perform a programming that allowing photoidentification of the same individual captured several times in a population study.

Any comments? (optional)

1 resposta

It is very important that different areas within the sciences communicate, as is the case of computing and biology, because the demand for graphic computing by biologists increases in order to better represent our work.

Número de respostas diárias

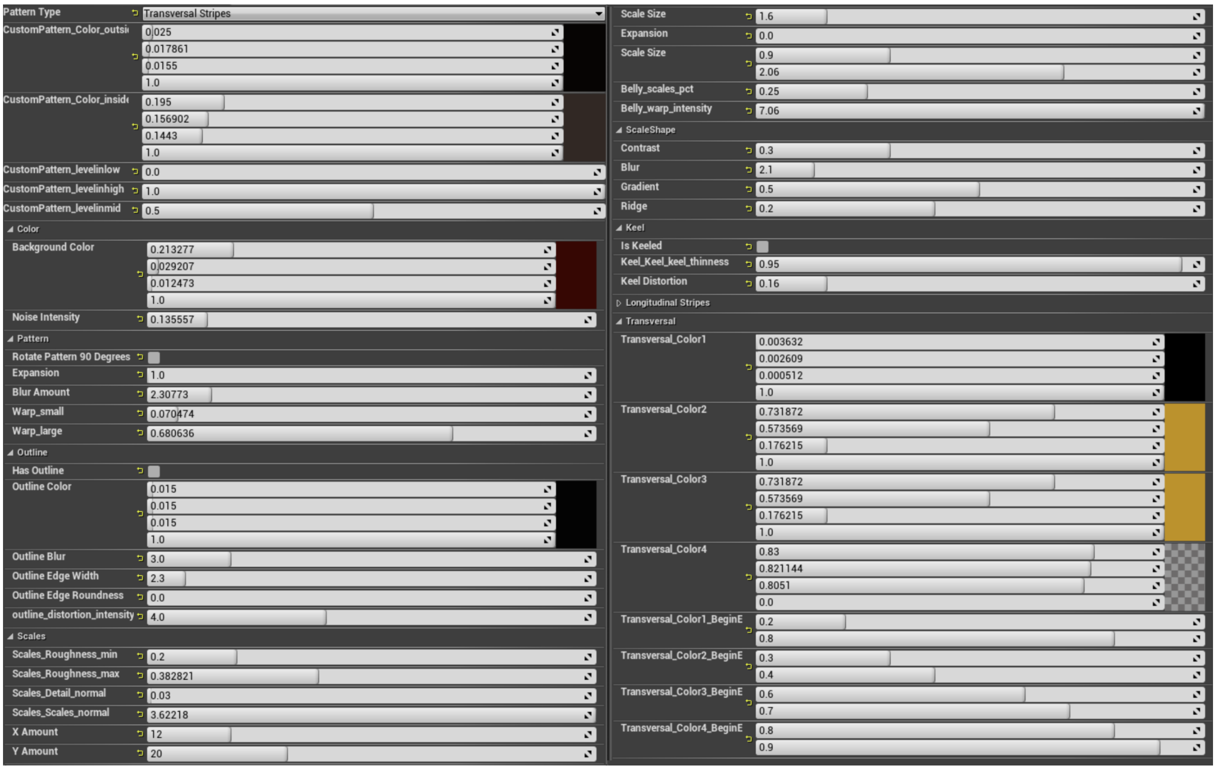


APPENDIX D — PARAMETERS

What follows in this appendix are the parameters used to generate each of the snakes shown in our results. Colors are represented in gamma space float4 RGBA format, with values ranging from 0.0 to 1.0.

D.1 *Micrurus altirostris*

The generated *Micrurus altirostris* textures can be seen in Figure 5.3.

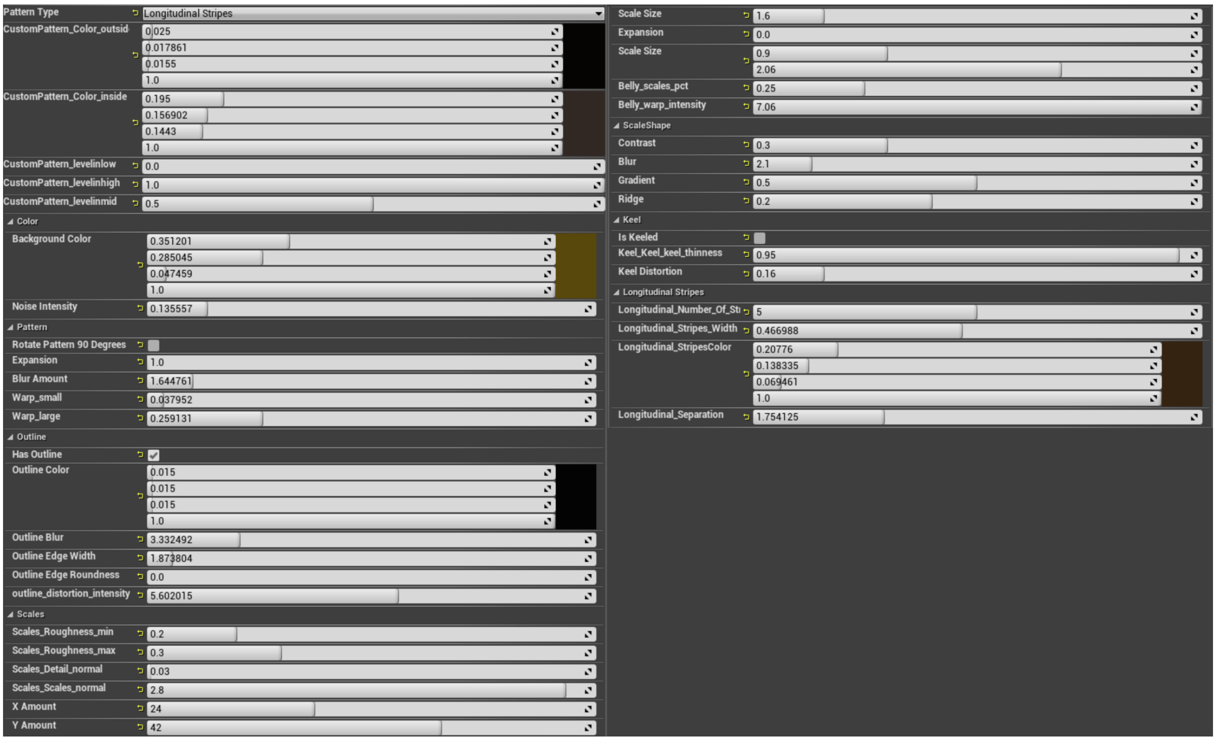


Pattern Type	Transversal Stripes
CustomPattern_Color_outsi	0.025 0.017861 0.0155 1.0
CustomPattern_Color_inside	0.195 0.156902 0.1443 1.0
CustomPattern_levelinlow	0.0
CustomPattern_levelinhigh	1.0
CustomPattern_levelinmid	0.5
Color	
Background Color	0.213277 0.029207 0.012473 1.0
Noise Intensity	0.135557
Pattern	
Rotate Pattern 90 Degrees	□
Expansion	1.0
Blur Amount	2.30773
Warp_small	0.070474
Warp_large	0.680636
Outline	
Has Outline	□
Outline Color	0.015 0.015 0.015 1.0
Outline Blur	3.0
Outline Edge Width	2.3
Outline Edge Roundness	0.0
outline_distortion_intensity	4.0
Scales	
Scales_Roughness_min	0.2
Scales_Roughness_max	0.382821
Scales_Detail_normal	0.03
Scales_Scales_normal	3.62218
X Amount	12
Y Amount	20
Scale Size	1.6
Expansion	0.0
Scale Size	0.9 2.06
Belly_scales_pct	0.25
Belly_warp_intensity	7.06
ScaleShape	
Contrast	0.3
Blur	2.1
Gradient	0.5
Ridge	0.2
Keel	
Is Keeled	□
Keel_Keel_keel_thinness	0.95
Keel Distortion	0.16
Longitudinal Stripes	
Transversal	
Transversal_Color1	0.003632 0.002609 0.000512 1.0
Transversal_Color2	0.731872 0.573569 0.176215 1.0
Transversal_Color3	0.731872 0.573569 0.176215 1.0
Transversal_Color4	0.83 0.821144 0.8051 0.0
Transversal_Color1_BeginE	0.2 0.8
Transversal_Color2_BeginE	0.3 0.4
Transversal_Color3_BeginE	0.6 0.7
Transversal_Color4_BeginE	0.8 0.9

Figure D.1 – Parameters used for the *Micrurus altirostris*.

D.2 *Elapomorphus quinquelineatus*

The generated *Elapomorphus quinquelineatus* textures can be seen in Figure 5.4.



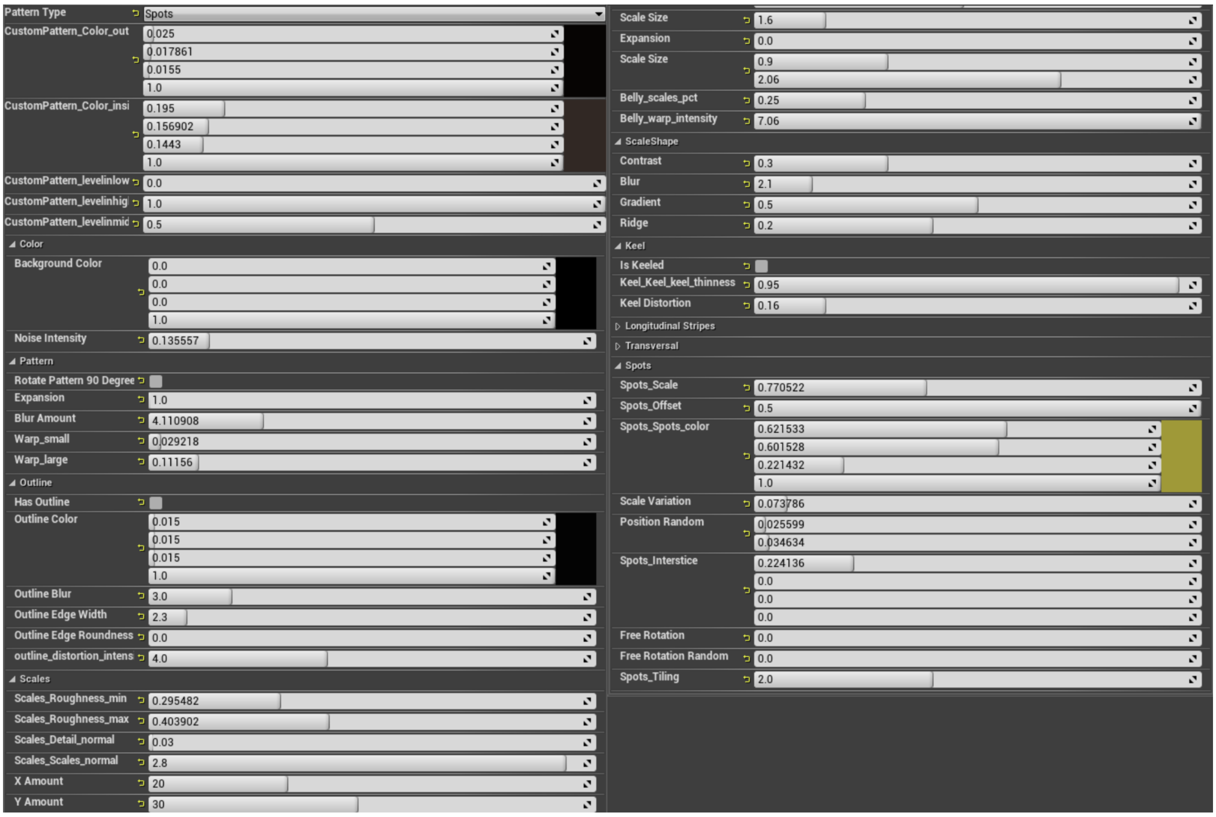
Pattern Type	Longitudinal Stripes
CustomPattern_Color_outsid	0.025 0.017861 0.0155 1.0
CustomPattern_Color_inside	0.195 0.156902 0.1443 1.0
CustomPattern_levelinlow	0.0
CustomPattern_levelinhigh	1.0
CustomPattern_levelinmid	0.5
◢ Color	
Background Color	0.351201 0.285045 0.047459 1.0
Noise Intensity	0.135557
◢ Pattern	
Rotate Pattern 90 Degrees	□
Expansion	1.0
Blur Amount	1.644761
Warp_small	0.037952
Warp_large	0.259131
◢ Outline	
Has Outline	☑
Outline Color	0.015 0.015 0.015 1.0
Outline Blur	3.332492
Outline Edge Width	1.873804
Outline Edge Roundness	0.0
outline_distortion_intensity	5.602015
◢ Scales	
Scales_Roughness_min	0.2
Scales_Roughness_max	0.3
Scales_Detail_normal	0.03
Scales_Scales_normal	2.8
X Amount	24
Y Amount	42

Scale Size	1.6
Expansion	0.0
Scale Size	0.9 2.06
Belly_scales_pct	0.25
Belly_warp_intensity	7.06
◢ ScaleShape	
Contrast	0.3
Blur	2.1
Gradient	0.5
Ridge	0.2
◢ Keel	
Is Keeled	□
Keel_Keel_keel_thinness	0.95
Keel Distortion	0.16
◢ Longitudinal Stripes	
Longitudinal_Number_Of_Sti	5
Longitudinal_Stripes_Width	0.466988
Longitudinal_StripesColor	0.20776 0.138335 0.069461 1.0
Longitudinal_Separation	1.754125

Figure D.2 – Parameters used for the *Elapomorphus quinquelineatus*.

D.3 *Liophis miliaris*

The generated *Liophis miliaris* textures can be seen in Figure 5.5.

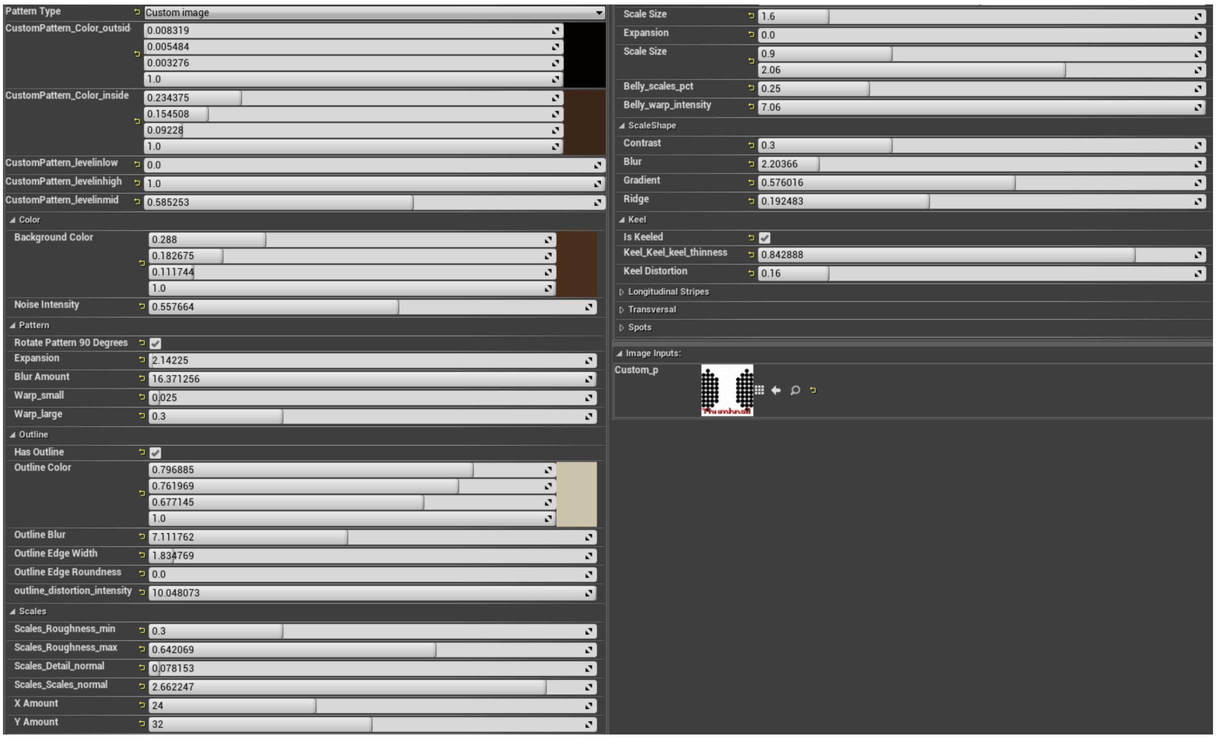


Pattern Type	Spots
CustomPattern_Color_out	0.025 0.017861 0.0155 1.0
CustomPattern_Color_insi	0.195 0.156902 0.1443 1.0
CustomPattern_levelinlow	0.0
CustomPattern_levelinhig	1.0
CustomPattern_levelinmid	0.5
◢ Color	
Background Color	0.0 0.0 0.0 1.0
Noise Intensity	0.135557
◢ Pattern	
Rotate Pattern 90 Degree	□
Expansion	1.0
Blur Amount	4.110908
Warp_small	0.029218
Warp_large	0.11156
◢ Outline	
Has Outline	□
Outline Color	0.015 0.015 0.015 1.0
Outline Blur	3.0
Outline Edge Width	2.3
Outline Edge Roundness	0.0
outline_distortion_intens	4.0
◢ Scales	
Scales_Roughness_min	0.295482
Scales_Roughness_max	0.403902
Scales_Detail_normal	0.03
Scales_Scales_normal	2.8
X Amount	20
Y Amount	30
Scale Size	1.6
Expansion	0.0
Scale Size	0.9 2.06
Belly_scales_pct	0.25
Belly_warp_intensity	7.06
◢ ScaleShape	
Contrast	0.3
Blur	2.1
Gradient	0.5
Ridge	0.2
◢ Keel	
Is Keeled	□
Keel_Keel_keel_thinness	0.95
Keel Distortion	0.16
▷ Longitudinal Stripes	
▷ Transversal	
◢ Spots	
Spots_Scale	0.770522
Spots_Offset	0.5
Spots_Spots_color	0.621533 0.601528 0.221432 1.0
Scale Variation	0.073786
Position Random	0.025599 0.034634
Spots_Interstice	0.224136 0.0 0.0 0.0
Free Rotation	0.0
Free Rotation Random	0.0
Spots_Tiling	2.0

Figure D.3 – Parameters used for the *Liophis miliaris*.

D.4 *Crotalus viridis*

The generated *Crotalus viridis* textures can be seen in Figure 5.6.



Pattern Type	Custom image
CustomPattern_Color_outsid	0.008319 0.005484 0.003276 1.0
CustomPattern_Color_inside	0.234375 0.154508 0.09228 1.0
CustomPattern_levelinlow	0.0
CustomPattern_levelinhigh	1.0
CustomPattern_levelinmid	0.585253
◢ Color	
Background Color	0.288 0.182675 0.111744 1.0
Noise Intensity	0.557664
◢ Pattern	
Rotate Pattern 90 Degrees	☑
Expansion	2.14225
Blur Amount	16.371256
Warp_small	0.025
Warp_large	0.3
◢ Outline	
Has Outline	☑
Outline Color	0.796885 0.761969 0.677145 1.0
Outline Blur	7.111762
Outline Edge Width	1.834769
Outline Edge Roundness	0.0
outline_distortion_intensity	10.048073
◢ Scales	
Scales_Roughness_min	0.3
Scales_Roughness_max	0.642069
Scales_Detail_normal	0.078153
Scales_Scales_normal	2.662247
X Amount	24
Y Amount	32

Scale Size	1.6
Expansion	0.0
Scale Size	0.9 2.06
Belly_scales_pct	0.25
Belly_warp_intensity	7.06
◢ ScaleShape	
Contrast	0.3
Blur	2.20366
Gradient	0.576016
Ridge	0.192483
◢ Keel	
Is Keeled	☑
Keel_Keel_keel_thinness	0.842888
Keel Distortion	0.16
▷ Longitudinal Stripes	
▷ Transversal	
▷ Spots	
◢ Image Inputs:	
Custom_p	Thumbnail

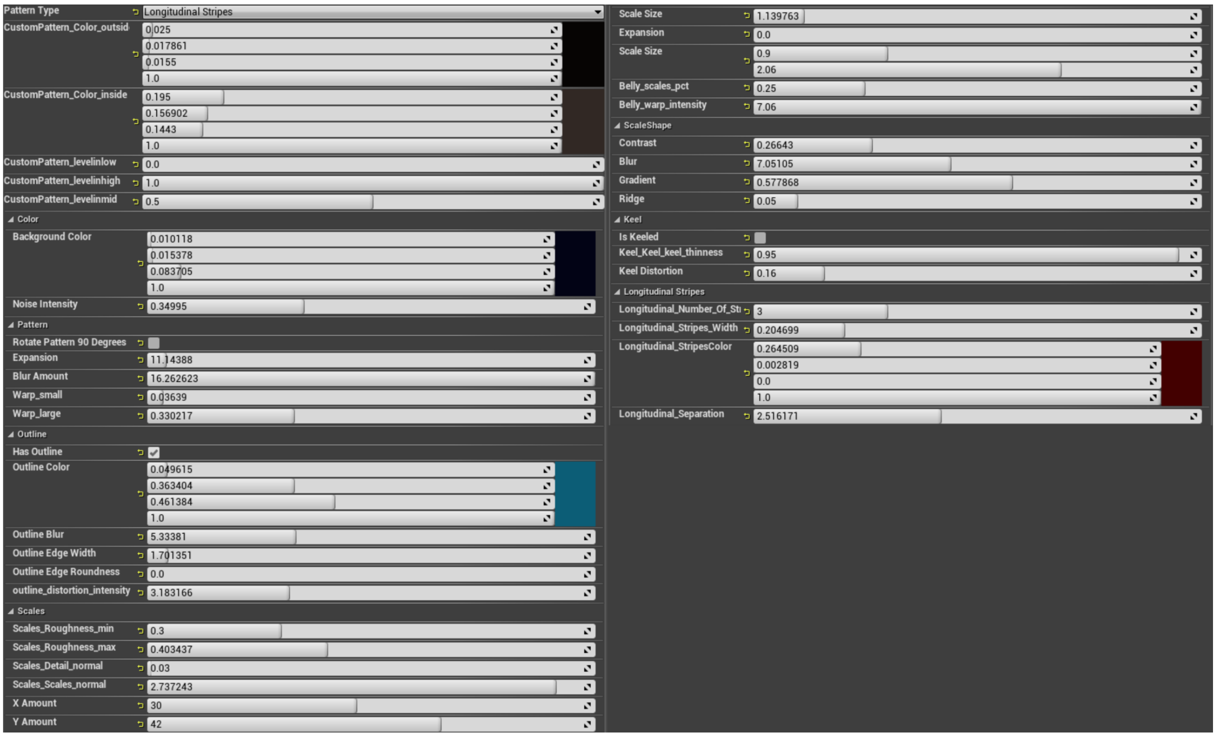
Figure D.4 – Parameters used for the first fictional snake.

Cellular automaton descriptor file used to generate the pattern:

```
0 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0
rectangular
0
5
1100 2
1000 x
```

D.5 *Fictional Snake 1*

The generated Fictional Snake 1 textures can be seen in Figure 5.11.

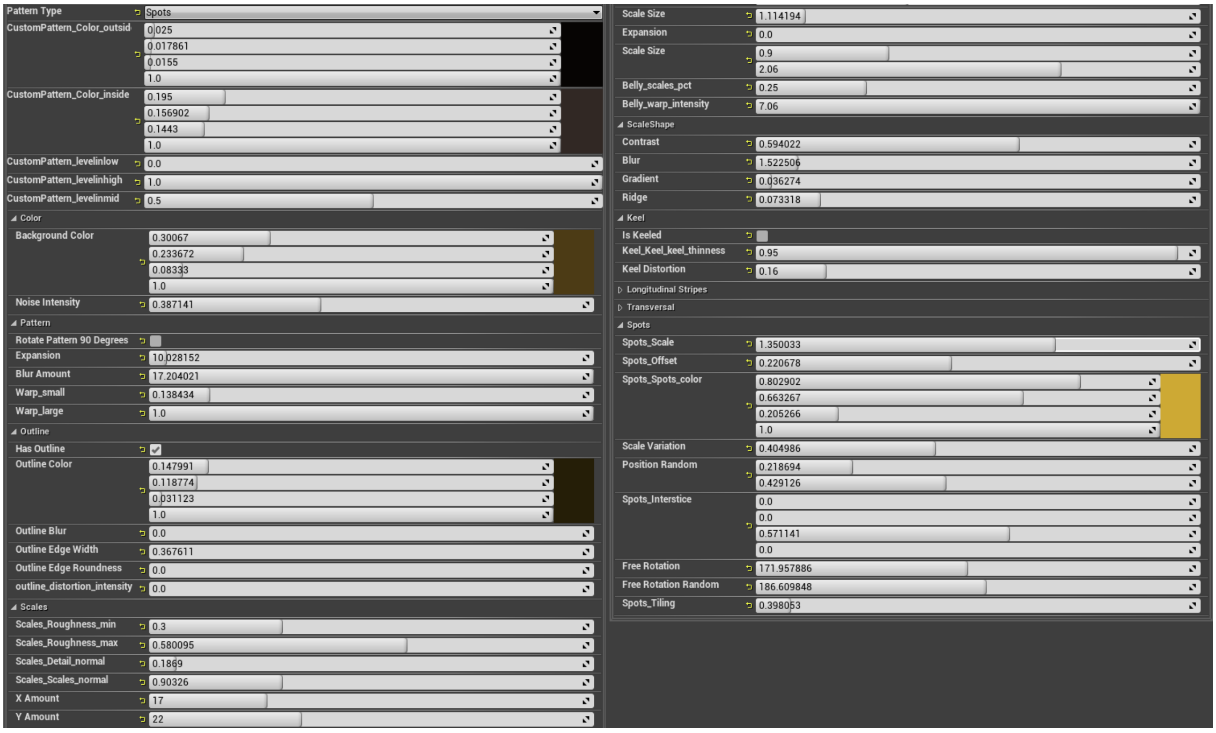


Pattern Type	Longitudinal Stripes
CustomPattern_Color_outsid	0.025 0.017861 0.0155 1.0
CustomPattern_Color_inside	0.195 0.156902 0.1443 1.0
CustomPattern_levelinlow	0.0
CustomPattern_levelinhigh	1.0
CustomPattern_levelinmid	0.5
◢ Color	
Background Color	0.010118 0.015378 0.083705 1.0
Noise Intensity	0.34995
◢ Pattern	
Rotate Pattern 90 Degrees	□
Expansion	11.14388
Blur Amount	16.262623
Warp_small	0.03639
Warp_large	0.330217
◢ Outline	
Has Outline	☑
Outline Color	0.049615 0.363404 0.461384 1.0
Outline Blur	5.33381
Outline Edge Width	1.701351
Outline Edge Roundness	0.0
outline_distortion_intensity	3.183166
◢ Scales	
Scales_Roughness_min	0.3
Scales_Roughness_max	0.403437
Scales_Detail_normal	0.03
Scales_Scales_normal	2.737243
X Amount	30
Y Amount	42
Scale Size	1.139763
Expansion	0.0
Scale Size	0.9 2.06
Belly_scales_pct	0.25
Belly_warp_intensity	7.06
◢ ScaleShape	
Contrast	0.26643
Blur	7.05105
Gradient	0.577868
Ridge	0.05
◢ Keel	
Is Keeled	□
Keel_Keel_keel_thinness	0.95
Keel Distortion	0.16
◢ Longitudinal Stripes	
Longitudinal_Number_Of_Sti	3
Longitudinal_Stripes_Width	0.204699
Longitudinal_StripesColor	0.264509 0.002819 0.0 1.0
Longitudinal_Separation	2.516171

Figure D.5 – Parameters used for the second fictional snake.

D.6 *Fictional Snake 2*

The generated Fictional Snake 2 textures can be seen in Figure 5.12.



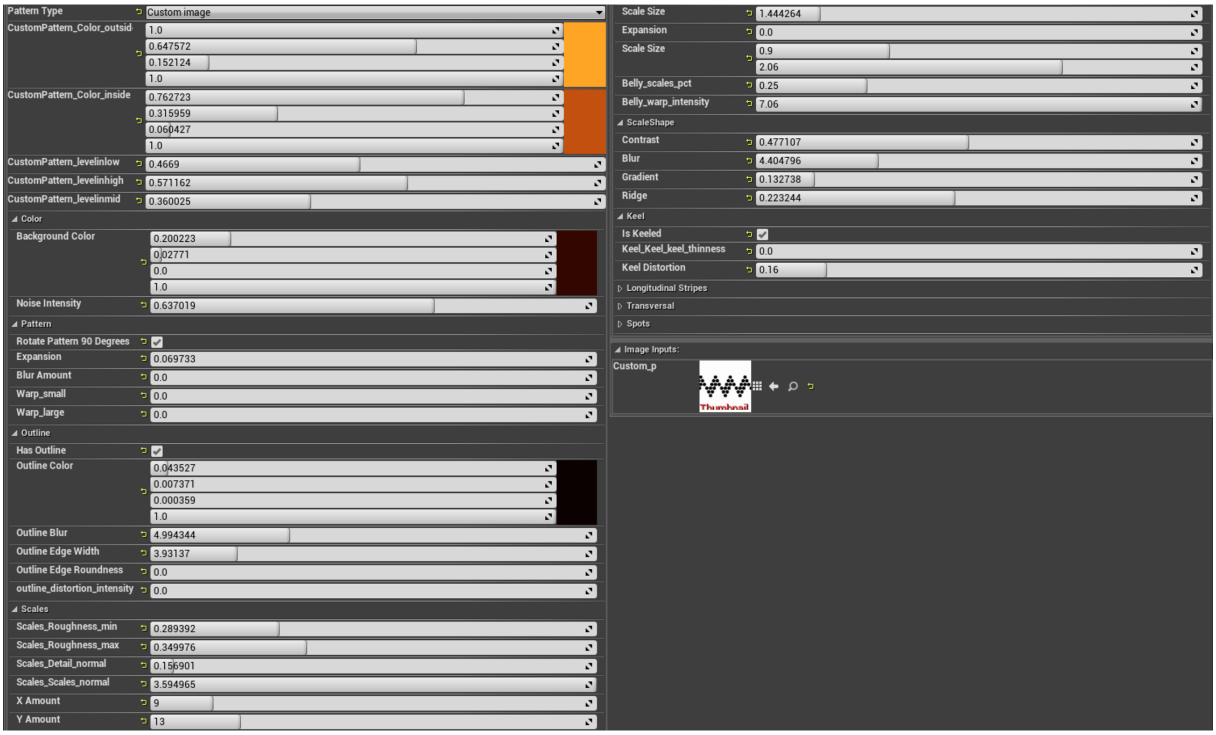
Pattern Type	Spots
CustomPattern_Color_outsid	0.025 0.017861 0.0155 1.0
CustomPattern_Color_inside	0.195 0.156902 0.1443 1.0
CustomPattern_levelinlow	0.0
CustomPattern_levelinhigh	1.0
CustomPattern_levelinmid	0.5
Color	
Background Color	0.30067 0.233672 0.08333 1.0
Noise Intensity	0.387141
Pattern	
Rotate Pattern 90 Degrees	☐
Expansion	10.028152
Blur Amount	17.204021
Warp_small	0.138434
Warp_large	1.0
Outline	
Has Outline	☑
Outline Color	0.147991 0.118774 0.031123 1.0
Outline Blur	0.0
Outline Edge Width	0.367611
Outline Edge Roundness	0.0
outline_distortion_intensity	0.0
Scales	
Scales_Roughness_min	0.3
Scales_Roughness_max	0.580095
Scales_Detail_normal	0.1869
Scales_Scales_normal	0.90326
X Amount	17
Y Amount	22

Scale Size	1.114194
Expansion	0.0
Scale Size	0.9 2.06
Belly_scales_pct	0.25
Belly_warp_intensity	7.06
ScaleShape	
Contrast	0.594022
Blur	1.522506
Gradient	0.036274
Ridge	0.073318
Keel	
Is Keeled	☐
Keel_Keel_keel_thinness	0.95
Keel Distortion	0.16
Longitudinal Stripes	
Transversal	
Spots	
Spots_Scale	1.350033
Spots_Offset	0.220678
Spots_Spots_color	0.802902 0.663267 0.205266 1.0
Scale Variation	0.404986
Position Random	0.218694 0.429126
Spots_Interstice	0.0 0.0 0.571141 0.0
Free Rotation	171.957886
Free Rotation Random	186.609848
Spots_Tiling	0.398053

Figure D.6 – Parameters used for the *Crotalus viridis*.

D.7 *Fictional Snake 3*

The generated Fictional Snake 3 textures can be seen in Figure 5.13.



```
Pattern Type                    Custom image
CustomPattern_Color_outsid      1.0
                                0.647572
                                0.152124
                                1.0
CustomPattern_Color_inside      0.762723
                                0.315959
                                0.060427
                                1.0
CustomPattern_levelinlow        0.4669
CustomPattern_levelinhigh       0.571162
CustomPattern_levelinmid        0.360025
◢ Color
  Background Color              0.200223
                                0.02771
                                0.0
                                1.0
  Noise Intensity               0.637019
◢ Pattern
  Rotate Pattern 90 Degrees     ☑
  Expansion                     0.069733
  Blur Amount                   0.0
  Warp_small                    0.0
  Warp_large                    0.0
◢ Outline
  Has Outline                   ☑
  Outline Color                 0.043527
                                0.007371
                                0.000359
                                1.0
  Outline Blur                  4.994344
  Outline Edge Width            3.93137
  Outline Edge Roundness        0.0
  outline_distortion_intensity  0.0
◢ Scales
  Scales_Roughness_min          0.289392
  Scales_Roughness_max          0.349976
  Scales_Detail_normal          0.156901
  Scales_Scales_normal          3.594965
  X Amount                      9
  Y Amount                      13

Scale Size                      1.444264
Expansion                       0.0
Scale Size                      0.9
                                2.06
Belly_scales_pct                0.25
Belly_warp_intensity            7.06
◢ ScaleShape
Contrast                        0.477107
Blur                            4.404796
Gradient                        0.132738
Ridge                           0.223244
◢ Keel
Is Keeled                       ☑
Keel_Keel_keel_thinness         0.0
Keel Distortion                 0.16
▷ Longitudinal Stripes
▷ Transversal
▷ Spots

◢ Image Inputs:
Custom_p
```

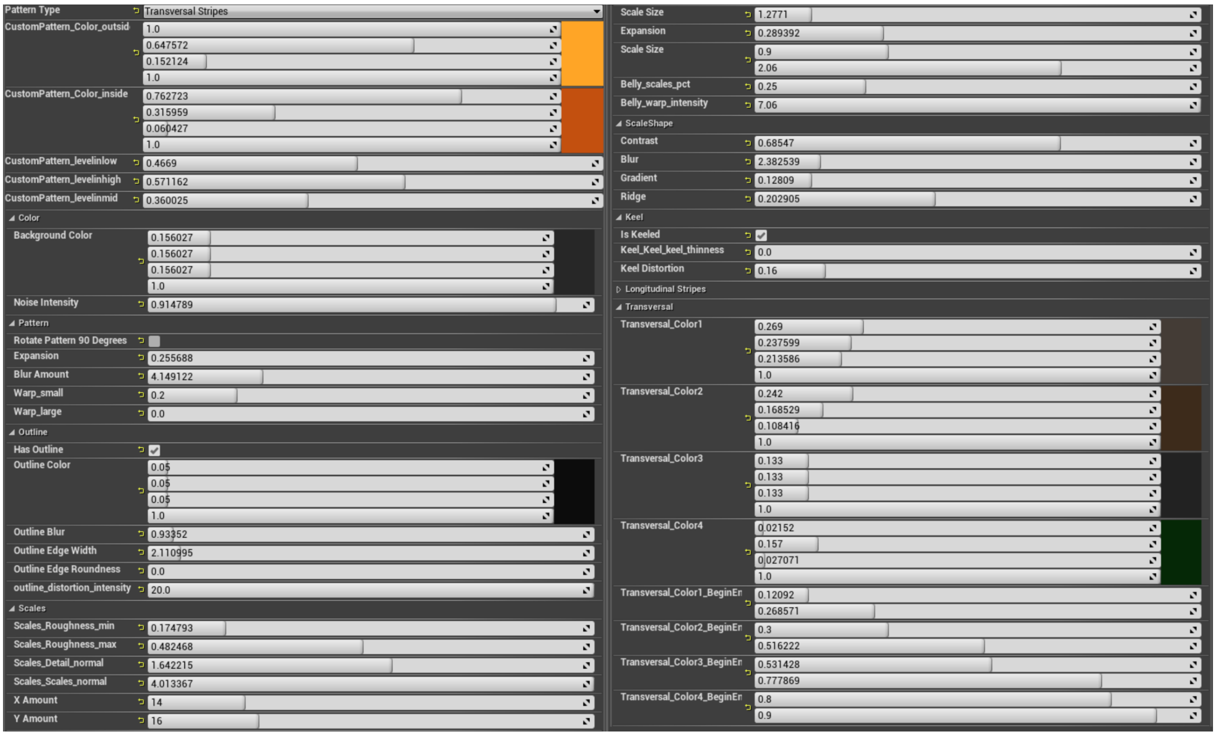
Figure D.7 – Parameters used for the third fictional snake.

Cellular automaton descriptor file used to generate the pattern:

```
1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
triangular
2
6
100 2
000 x
```

D.8 *Fictional Snake 4*

The generated Fictional Snake 4 textures can be seen in Figure 5.14.



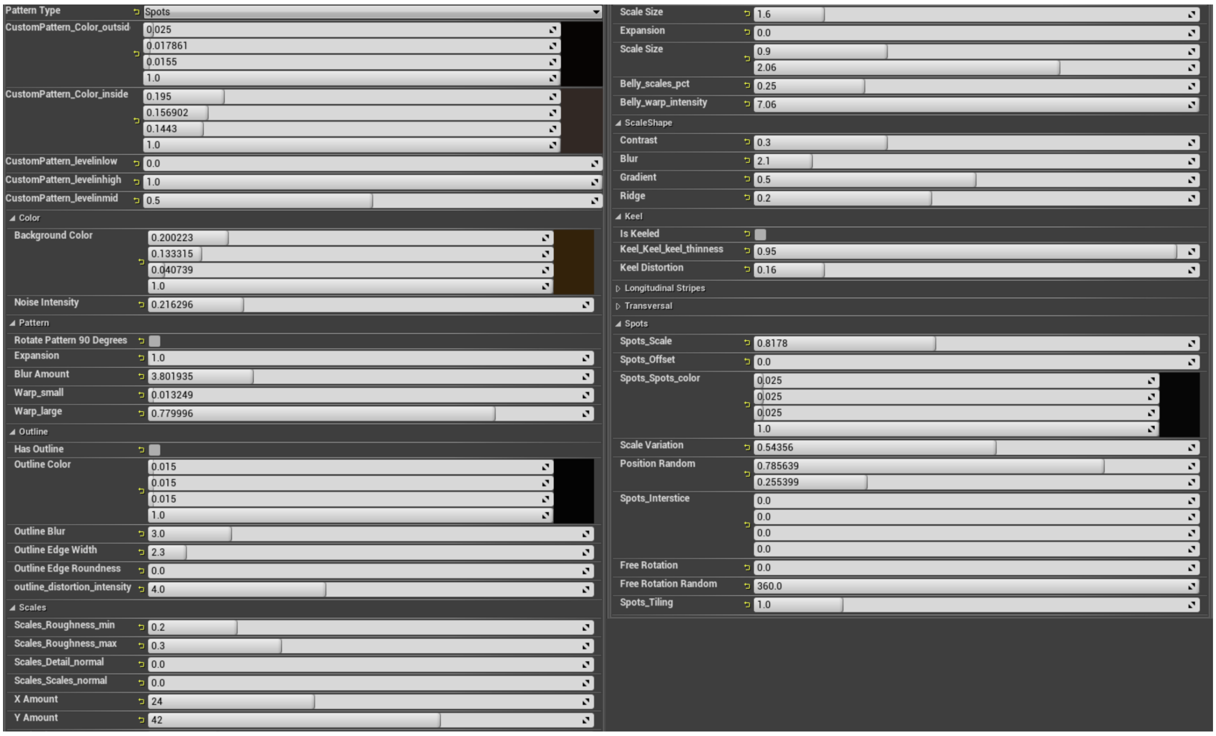
Pattern Type	Transversal Stripes
CustomPattern_Color_outsid	1.0 0.647572 0.152124 1.0
CustomPattern_Color_inside	0.762723 0.315959 0.060427 1.0
CustomPattern_levelinlow	0.4669
CustomPattern_levelinhigh	0.571162
CustomPattern_levelinmid	0.360025
◢ Color	
Background Color	0.156027 0.156027 0.156027 1.0
Noise Intensity	0.914789
◢ Pattern	
Rotate Pattern 90 Degrees	□
Expansion	0.255688
Blur Amount	4.149122
Warp_small	0.2
Warp_large	0.0
◢ Outline	
Has Outline	☑
Outline Color	0.05 0.05 0.05 1.0
Outline Blur	0.93352
Outline Edge Width	2.110995
Outline Edge Roundness	0.0
outline_distortion_intensity	20.0
◢ Scales	
Scales_Roughness_min	0.174793
Scales_Roughness_max	0.482468
Scales_Detail_normal	1.642215
Scales_Scales_normal	4.013367
X Amount	14
Y Amount	16

Scale Size	1.2771
Expansion	0.289392
Scale Size	0.9 2.06
Belly_scales_pct	0.25
Belly_warp_intensity	7.06
◢ ScaleShape	
Contrast	0.68547
Blur	2.382539
Gradient	0.12809
Ridge	0.202905
◢ Keel	
Is Keeled	☑
Keel_Keel_keel_thinness	0.0
Keel Distortion	0.16
▷ Longitudinal Stripes	
◢ Transversal	
Transversal_Color1	0.269 0.237599 0.213586 1.0
Transversal_Color2	0.242 0.168529 0.108416 1.0
Transversal_Color3	0.133 0.133 0.133 1.0
Transversal_Color4	0.02152 0.157 0.027071 1.0
Transversal_Color1_BeginEn	0.12092 0.268571
Transversal_Color2_BeginEn	0.3 0.516222
Transversal_Color3_BeginEn	0.531428 0.777869
Transversal_Color4_BeginEn	0.8 0.9

Figure D.8 – Parameters used for the fourth fictional snake.

D.9 *Cheetah*

The generated Cheetah texture can be seen in Figure 6.1.



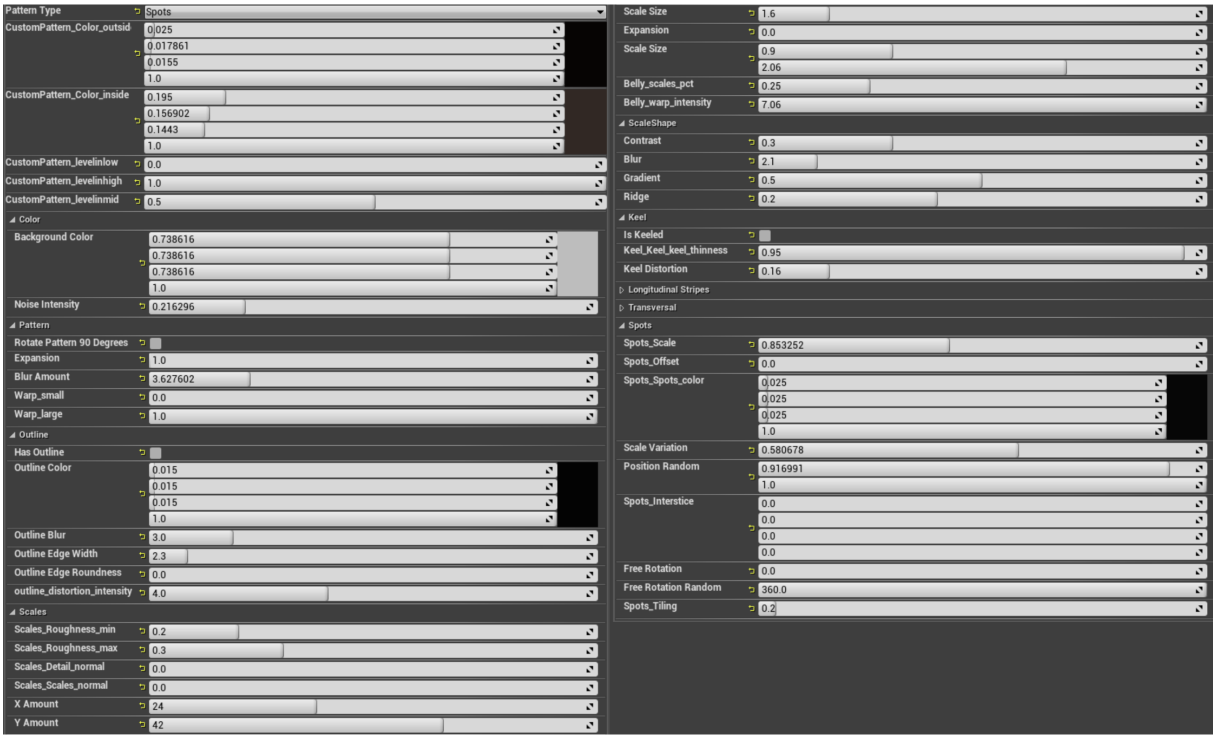
Pattern Type	Spots
CustomPattern_Color_outsid	0.025 0.017861 0.0155 1.0
CustomPattern_Color_inside	0.195 0.156902 0.1443 1.0
CustomPattern_levelinlow	0.0
CustomPattern_levelinhigh	1.0
CustomPattern_levelinmid	0.5
◢ Color	
Background Color	0.200223 0.133315 0.040739 1.0
Noise Intensity	0.216296
◢ Pattern	
Rotate Pattern 90 Degrees	☐
Expansion	1.0
Blur Amount	3.801935
Warp_small	0.013249
Warp_large	0.779996
◢ Outline	
Has Outline	☐
Outline Color	0.015 0.015 0.015 1.0
Outline Blur	3.0
Outline Edge Width	2.3
Outline Edge Roundness	0.0
outline_distortion_intensity	4.0
◢ Scales	
Scales_Roughness_min	0.2
Scales_Roughness_max	0.3
Scales_Detail_normal	0.0
Scales_Scales_normal	0.0
X Amount	24
Y Amount	42

Scale Size	1.6
Expansion	0.0
Scale Size	0.9 2.06
Belly_scales_pct	0.25
Belly_warp_intensity	7.06
◢ ScaleShape	
Contrast	0.3
Blur	2.1
Gradient	0.5
Ridge	0.2
◢ Keel	
Is Keeled	☐
Keel_Keel_keel_thinness	0.95
Keel Distortion	0.16
▷ Longitudinal Stripes	
▷ Transversal	
◢ Spots	
Spots_Scale	0.8178
Spots_Offset	0.0
Spots_Spots_color	0.025 0.025 0.025 1.0
Scale Variation	0.54356
Position Random	0.785639 0.255399
Spots_Interstice	0.0 0.0 0.0 0.0
Free Rotation	0.0
Free Rotation Random	360.0
Spots_Tiling	1.0

Figure D.9 – Parameters used for the Cheetah (*Acinonyx jubatus*).

D.10 *Cow*

The generated Cow texture can be seen in Figure 6.1.



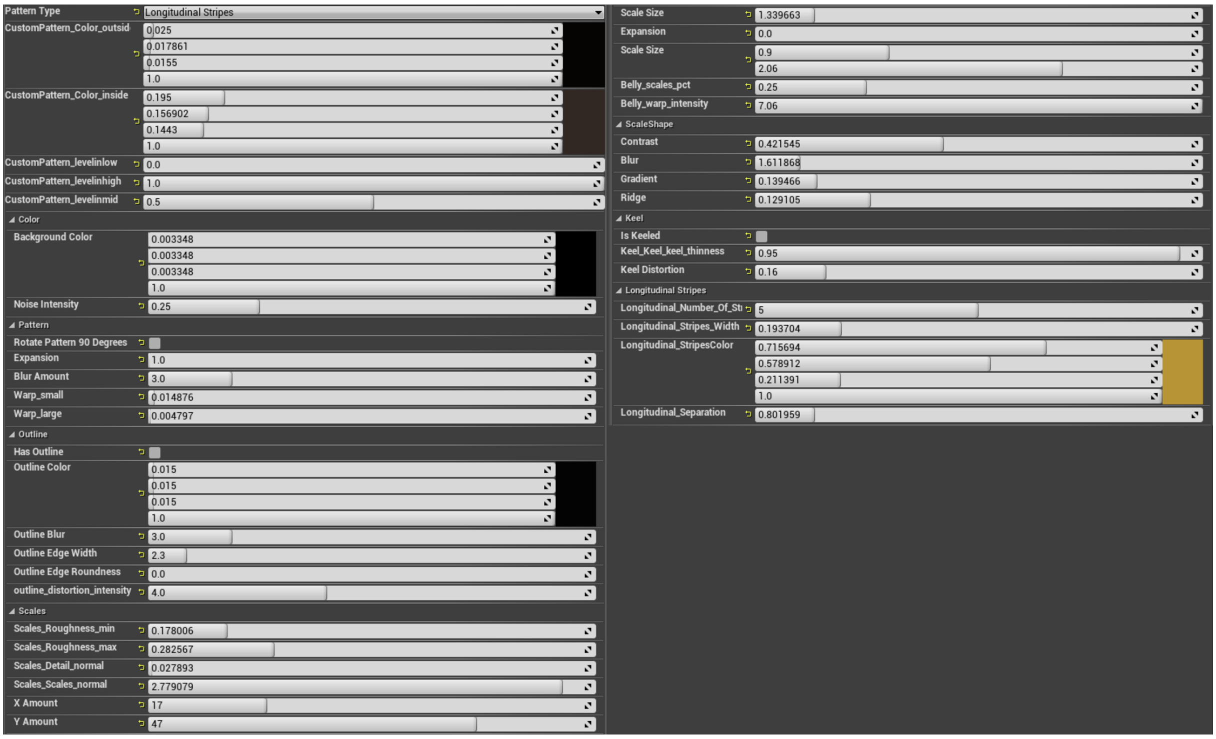
Pattern Type	Spots
CustomPattern_Color_outsid	0.025 0.017861 0.0155 1.0
CustomPattern_Color_inside	0.195 0.156902 0.1443 1.0
CustomPattern_levelinlow	0.0
CustomPattern_levelinhigh	1.0
CustomPattern_levelinmid	0.5
◢ Color	
Background Color	0.738616 0.738616 0.738616 1.0
Noise Intensity	0.216296
◢ Pattern	
Rotate Pattern 90 Degrees	□
Expansion	1.0
Blur Amount	3.627602
Warp_small	0.0
Warp_large	1.0
◢ Outline	
Has Outline	□
Outline Color	0.015 0.015 0.015 1.0
Outline Blur	3.0
Outline Edge Width	2.3
Outline Edge Roundness	0.0
outline_distortion_intensity	4.0
◢ Scales	
Scales_Roughness_min	0.2
Scales_Roughness_max	0.3
Scales_Detail_normal	0.0
Scales_Scales_normal	0.0
X Amount	24
Y Amount	42

Scale Size	1.6
Expansion	0.0
Scale Size	0.9 2.06
Belly_scales_pct	0.25
Belly_warp_intensity	7.06
◢ ScaleShape	
Contrast	0.3
Blur	2.1
Gradient	0.5
Ridge	0.2
◢ Keel	
Is Keeled	□
Keel_Keel_keel_thinness	0.95
Keel Distortion	0.16
▷ Longitudinal Stripes	
▷ Transversal	
◢ Spots	
Spots_Scale	0.853252
Spots_Offset	0.0
Spots_Spots_color	0.025 0.025 0.025 1.0
Scale Variation	0.580678
Position Random	0.916991 1.0
Spots_Interstice	0.0 0.0 0.0 0.0
Free Rotation	0.0
Free Rotation Random	360.0
Spots_Tiling	0.2

Figure D.10 – Parameters used for the cow (*Bos taurus*).

D.11 *Five-lined Skink*

The generated Five-lined Skink texture can be seen in Figure 6.1.



Pattern Type	Longitudinal Stripes
CustomPattern_Color_outsid	0.025 0.017861 0.0155 1.0
CustomPattern_Color_inside	0.195 0.156902 0.1443 1.0
CustomPattern_levelinlow	0.0
CustomPattern_levelinhigh	1.0
CustomPattern_levelinmid	0.5
◢ Color	
Background Color	0.003348 0.003348 0.003348 1.0
Noise Intensity	0.25
◢ Pattern	
Rotate Pattern 90 Degrees	□
Expansion	1.0
Blur Amount	3.0
Warp_small	0.014876
Warp_large	0.004797
◢ Outline	
Has Outline	□
Outline Color	0.015 0.015 0.015 1.0
Outline Blur	3.0
Outline Edge Width	2.3
Outline Edge Roundness	0.0
outline_distortion_intensity	4.0
◢ Scales	
Scales_Roughness_min	0.178006
Scales_Roughness_max	0.282567
Scales_Detail_normal	0.027893
Scales_Scales_normal	2.779079
X Amount	17
Y Amount	47

Scale Size	1.339663
Expansion	0.0
Scale Size	0.9 2.06
Belly_scales_pct	0.25
Belly_warp_intensity	7.06
◢ ScaleShape	
Contrast	0.421545
Blur	1.611868
Gradient	0.139466
Ridge	0.129105
◢ Keel	
Is Keeled	□
Keel_Keel_keel_thinness	0.95
Keel Distortion	0.16
◢ Longitudinal Stripes	
Longitudinal_Number_Of_Sti	5
Longitudinal_Stripes_Width	0.193704
Longitudinal_StripesColor	0.715694 0.578912 0.211391 1.0
Longitudinal_Separation	0.801959

Figure D.11 – Parameters used for the Five-lined Skink (*Plestiodon fasciatus*).

APPENDIX E — CELLULAR AUTOMATA DESCRIPTORS

Descriptor files for each of the cellular automatons shown in Figure 4.10.

Bothrops neuwiedii:

```
0 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0
rectangular
0
5
1100 2
1000 x
```

Crotalus viridis:

```
0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 1 1 1 0 0
triangular
0
6
110 2
100 x
```

Daboia russelii:

```
0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0
rectangular
0
4
1000 2
0000 x
```

Eunectes murinus:

```
0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0
rectangular
0
6
1110 1
1100 1
1000 x
```

Python molurus:

```
0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0
rectangular
0
6
1100 3
1000 1
0000 x
```

Vipera berus:

```
1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1
triangular
3
6
100 2
000 x
```