Noname manuscript No.
(will be inserted by the editor)

Procedural Generation of Materials for Real-Time Rendering

Alessio Bernardi · Davide Gadia · Dario Maggiorini · Claudio Enrico Palazzi · Laura Anna Ripamonti

Received: date / Accepted: date

Abstract The use of Procedural Content Generation techniques in the production of Video Games has seen a large diffusion in these last years. Regarding the procedural generation of Computer Graphics content, several works have been proposed about the automatic construction of complex models and environments, or about the instancing of several copies of a reference model, each with peculiar differences to introduce variety. However, very few works have proposed techniques

Alessio Bernardi
Department of Computer Science
University of Milan
Via Celoria 18
20133 - Milan (Italy)
E-mail: alessio.bernardi@studenti.unimi.it

Davide Gadia
Department of Computer Science
University of Milan
Via Celoria 18
20133 - Milan (Italy)
E-mail: gadia@di.unimi.it

Dario Maggiorini
Department of Computer Science
University of Milan
Via Celoria 18
20133 - Milan (Italy)
E-mail: dario@di.unimi.it

Claudio Enrico Palazzi
Department of Mathematics
University of Padoa
Via Trieste, 63
35131 - Padoa (Italy)
E-mail: cpalazzi@math.unipd.it

Laura Anna Ripamonti
Department of Computer Science
University of Milan
Via Celoria 18
20133 - Milan (Italy)
E-mail: ripamonti@di.unimi.it

for the procedural production of complex materials to be assigned to these generated models. In this paper, we present a method for the automatic generation of realistic layered materials based on the application of a Genetic Algorithm. We show that, with the proposed approach, is possible to generate several instances of a target material (e.g., a car paint, or a rusty metal), maintaining a desired level of closeness to the overall characteristics of the simulated interaction between the light and the surface, but introducing also a controlled amount of differences in the final reproduction of the perceived appearance.

Keywords Procedural Content Generation · Computer Graphics · Layered Material · Genetic Algorithm · Real-Time Rendering

1 Introduction

In the context of video games, Procedural Content Generation (PCG) refers to the automatic creation of contents through the application of algorithms and/or heuristics that are designed specifically for the game under development. The main positive effects of the application of PCG are a reduction of development time and an increase of the randomness of the game content and/or gameplay, thus extending the game replayability and longevity [11].

In these last years, a relevant number of researches have exploited PCG in order to automatically generate game levels [23,37,43], racing tracks [7], or dungeons [4]. More recently, the application of PCG and AI techniques has provided a relevant contribution to several specific aspects of game optimization, such as: impact on Non Player Characters (NPCs) [3, 24, 34, 36, 48], dynamic game balancing [20], and adaptive or personalized content generation [15, 18, 44]. Moreover, the term *Search-Based PCG* [41] has been proposed to refer to PCG techniques based on the application of evolutionary algorithms to game contents. The proposed techniques are mainly used to generate or evolve a game environment [9, 14, 28], the features of a character [17, 30], or for the evolution of the game agents behaviour in order to produce more challenging opponents to the players [1, 26, 27].

In the Computer Graphics (CG) field, PCG has a long and established history. The proposed techniques have addressed mainly the generation of 3D meshes and large virtual worlds to allow a much-reduced intervention by the user without the need of a long process of manual mesh modeling. A well-known example is the creation of plants and trees, which usually exploits the characteristics of fractals and L-systems [35, 47]. Another relevant application is the generation of buildings of different sizes and heights [29, 32, 40], large urban environments with streets, parks, lakes, areas with different population density [10,12,31,39], or other complex structures [19]. These techniques usually rely on a combination of shape grammars and optimization methods, often supported by the use of auxiliary image maps.

However, few works have been proposed on the application of PCG for the generation of the surface appearance of models. In the Procedural Texturing field [13], mathematical equations based also on fractals or noise functions are used to generate 2D images. These images are often applied during the rendering process, and they have indeed a role in the determination of the final visual appearance of a model. However, the procedural textures are usually applied e.g., to determine the base color of an object, or to simulate a complex surface roughness by mimicking

or perturbing the actual vertex normals of the mesh, rather than having a role in the functions computing the actual reflective behaviour of the material.

In this paper, we show how a Genetic Algorithm (GA) can be efficiently applied to an advanced computational model for real-time rendering of realistic layered materials. The main idea is to generate several versions of a target material, by evolving the parameters of the function calculating the interaction between the light and the surface layers. Their effectivenes is then evaluated by considering the perceptual differences with the original material. This approach can be used when several instances of the same model are generated in a scene (e.g., a medieval armour). The materials of each instance must share a common behaviour (e.g., a rusty metal effect), but we want to introduce a certain amount of perceptual differences in each instance, in order to enhance the variety and effectiveness of the generated virtual world.

The paper is structured as follows. In Section 2, we briefly describe the mathematical functions used to represent a realistic material in real-time rendering. In Section 3, we provide an overview of PCG techniques based on the application of evolutionary methods for the generation of materials in Computer Graphics. Section 4 describes the proposed PCG technique. In Section 5, we present the experimental setup used to evaluate the fitness functions considered in the proposed GA. In Section 6, we present some implementation details and discuss the computational performances of the method. Finally, Section 7 draws conclusions from the results and presents future work.

In [6], we already presented some preliminary results of the proposed approach. In this paper, we significantly extend the previous work, by considering:

- a third vector distance (the *Manhattan distance*) in the set of the possible fitness functions
- an additional test material, characterized by the presence of a participating media (the *validation metal depth* material)
- the results of a visual metric (*HDR-VDP-2* [21]), which graphically presents the possible perceived differences between a reference and a target image, in the perceptual evaluation of the generated materials.

2 Computation of materials reflectance in Computer Graphics

The rendering process in CG is based on the simulation of the physical interaction between light and the materials of the surfaces in a scene. For a uniform, not-emitting material, the light reflected by a surface in the direction of the virtual camera is given by the *rendering equation* [33]:

$$L_o(\boldsymbol{\omega_o}) = \int_{\boldsymbol{\omega_i} \in \Omega} f_r(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}) L_i(\boldsymbol{\omega_i})(\boldsymbol{\omega_i} \cdot \mathbf{n}) \partial \boldsymbol{\omega_i} \tag{1}$$

where

- $\mathbf{n}$ is the surface normal
- $\boldsymbol{\omega_i}$ and $\boldsymbol{\omega_o}$ are the direction of incoming light and the direction of reflected light, respectively
- $L_i(\boldsymbol{\omega_i})$ and $L_o(\boldsymbol{\omega_o})$ are the incoming and the reflected radiances, respectively

– $f_r(\boldsymbol{\omega_i}, \boldsymbol{\omega_o})$ is the *Bidirectional Reflectance Distribution Function (BRDF)*, which describes the amount of radiance reflected in the direction $\boldsymbol{\omega_o}$, given the radiance coming from direction $\boldsymbol{\omega_i}$

Due to its complexity, Eq. (1) cannot be solved directly. As a consequence, several approaches were proposed in order to approximate its solution [33]. Among the different proposals suitable for real-time rendering, the methods following the *microfacets* approach [2] are based on the idea that rough surfaces are composed of a large collection of microscopic facets bouncing light in different directions. The light scattering properties of the material are given by the orientation of the collection of microfacets, which is modeled using a statistical distribution. Following this approach, several methods for real-time rendering of complex materials use Eq. (2) [2] to define the BRDF:

$$FGD = f_r(\boldsymbol{\omega_i}, \boldsymbol{\omega_o}) = \frac{F(\boldsymbol{\omega_i} \cdot \mathbf{h})G(\boldsymbol{\omega_i}, \boldsymbol{\omega_o})D(\mathbf{h})}{4(\boldsymbol{\omega_i} \cdot \mathbf{n})(\boldsymbol{\omega_o} \cdot \mathbf{n})} \tag{2}$$

where

– $\mathbf{h}$ is the *half vector* between $\boldsymbol{\omega_i}$ and $\boldsymbol{\omega_o}$, used to approximate the direction of specular reflection
– $F(\boldsymbol{\omega_i} \cdot \mathbf{h})$ is the *Fresnel reflectance term*, which describes the amount of reflection and trasmission of light on a surface
– $G(\boldsymbol{\omega_i}, \boldsymbol{\omega_o})$ is the *Shadowing-Masking Function*, which accounts for self-occlusion effects among the microfacets, by providing the fraction of microfacets visible in both $\boldsymbol{\omega_i}$ and $\boldsymbol{\omega_o}$ directions
– $D(\mathbf{h})$ is the *Normal Distribution Function (NDF)*, which describes the statistical distribution of the orientation of the microfacets normals on a surface

Among the several NDFs proposed in literature, the most used is currently the *GGX distribution* [46]:

$$D = GGX(\mathbf{n}, \mathbf{h}, \alpha) = \frac{\alpha^2}{\pi\Big((\mathbf{n} \cdot \mathbf{h})^2(\alpha^2 - 1) + 1\Big)^2} \tag{3}$$

where α parameter controls the overall *roughness* of the surface.

When light reflection and transmission are both considered in the computational models, the function f_r in Eq. (2) is called *Bidirectional Scattering Distribution Function (BSDF)*, rather than *BRDF*; because the latter describes only the reflectance properties of a material.

3 Related work

To the best of our knowledge, at the time of writing the applications of PCG and evolutionary techniques for the generation of BRDFs or BSDFs have not yet been extensively investigated.

Brady et al. [8] proposed a framework for learning new analytic BRDF models through Genetic Programming (GP). They used, as the initial population of the method, the set of expressions of different BRDFs proposed in literature and, as the target, a trained set of measured materials from a database containing the

reflectance functions of 100 materials freely available for academic and research purposes. At each iteration, tournament selection is used to determine a new population, and one-point crossover is used to generate offsprings. During the pairing process, different symbolic transformations could be applied to the parameters and to the mathematical operations of the two parent BRDF models in order to create a new, more complex reflectance function. The same operations are used in case of random mutation. Then, the fitness of the generated formulations is evaluated considering an error function with respect to the training set of selected measured materials.

A GP approach is used also by Sitthi-Amorn et al. [42], but with a different approach. In this work, an evolutionary method is applied in order to increasingly simplify the source code of a shader, by copying, reordering and deleting its statements and expressions. The goal of the method is to find the optimal compromise between rendering speed and accuracy of the approximation of the light-material interaction. The authors applied an iterative GA which maintains and evaluates a diverse population of shader variants, returning those representing the best optimization tradeoffs. The proposed GA is characterized by tournament selection (with size 16), one-point crossover and random mutation. To evaluate the accuracy of this approach, a per-pixel color difference metric is applied between the images generated using the original shader and the images created using the simplified offsprings.

Masia et al. [22] applied a GA to determine the reflectance properties of the material of an object in a given image. The main goal of the paper was not the acquisition of the parameters of the actual BSDF, but to determine an estimation of the parameters using an image of the object and rendering a virtual scene. In order to reduce the dimensionality of the problem, the authors assumed that parameters like e.g., lighting, or the geometry of the object are known. The parameters of two well-known BRDF models for opaque and translucent materials are used as chromosomes, and an initial population is created assigning random values to these parameters. At each iteration, reproduction is applied using one-point crossover and random mutation. Then, an image is rendered for each of the chromosomes created in each generation. The fitness is then evaluated calculating a per-pixel difference between the target image and the rendered image.

4 Methodology

In this paper, we consider a different approach to the application of evolutionary techniques for the generation of BSDFs. In particular, we apply a GA to evolve the parameters of a recent computational model for the rendering of *layered materials*. From a CG point of view, when several instances of a particular 3D mesh are created to populate a large virtual world, usually some PCG approach is considered in order to introduce variety in the scene and to maintain an adequate level of interest in the users. Procedural techniques are usually applied in order to modify to some degree the features of the meshes (e.g., changing dimensions or applying deformations), to add or remove some details in the setup of a character (e.g., the set of available weapons), or to create or modify the patterns and colors of the textures applied on the meshes. In the proposed method, we generate several versions of a target material, by evolving the parameters of its BSDF. Each generated

material instance will share the overall physical behaviour of the target one, but with a moderate (but perceivable) amount of difference. Moreover, we have also considered how to efficiently integrate our approach in a modern real-time rendering pipeline. Our proposal intrinsically separates the evolutionary stage from the actual rendering: by evolving the parameters of a state-of-the-art computational model, rather than completely creating a new BSDF, we can apply our GA in any pre-rendering stage computed on the CPU (e.g., the loading of a level, or during the procedural generation of the model instances), while we can simply apply the generated parameters to the adopted rendering method on the GPU.

In this section, we stepwise go through our approach, by describing the BSDF model we have adopted for our evolutionary approach and the main parts of the GA: representation, algorithm and fitness function.

4.1 Considered BSDF for layered materials

Layered materials are composed of different layers, each layer offering peculiar reflectance characteristics. Examples of layered materials are car paint, painted wood, rusty metal, etc. Computational models for the simulation of layered materials usually consider separate BRDFs/BSDFs for each layer, and then apply some kind of blending, or other operations, to simulate, at various degrees of approximation, the scattering of light between the different layers [2].

Recently, a work by Belcour [5] has proposed a novel framework for the analysis and computation of light transport within layered materials. The innovative approach of the method is to analyze the BSDF (and its parameters), assigned to each layer, and to infer a statistical description of its light scattering characteristics. Using, as target NDF, the GGX distribution [46] described in Eq. (3), a set of atomic statistical operators is proposed to describe reflection, refraction, volume scattering and volume absorption starting from the *energy*, *mean*, and *variance* of the BSDF at each layer. Moreover, the *Adding-Doubling* method [16] is used to combine the atomic operators defined in each layer. The method can approximate multiple light scatterings inside the material considering also the possible presence of a participating media among the layers. The resulting statistical description, after the combination of the atomic operators, is used to instantiate a single BSDF model approximating the complex and multiple light scatterings within the original layered structure. Table 1 summarizes the atomic operators proposed by the Belcour's method:

- the exponents R and T are used to indicate a reflected or transmitted parameter
- FGD refers to a microfacet-based BSDF as described in Eq. (2)
- α is the roughness of the layer
- s is a roughness scaling factor for the trasmission
- η_{12} is the ratio of the refractive indices
- h is the depth of the layer
- σ_t is the transmittance cross-section
- σ_s is the scattering cross-section
- σ_g accounts for the increase in variance due to the width of the phase function

A detailed analysis of the method and of its parameters is beyond the scope of this paper. As a consequence, we refer to the original paper [5] for more details.

Table 1: The statistical atomic operators of Belcour's method [5], used to approximate the outgoing energy e, mean μ, and variance σ given the incident energy e_i, mean μ_i, and variance σ_i. As in the original paper, we have omitted the square on the variance for better readability. We refer to the original paper [5] for more details.

	energy	**mean**	**variance**
Reflection	$e^R = e_i \times FGD$	$\mu^R = -\mu_i$	$\sigma^R = \sigma_i + f(\alpha)$
Refraction	$e^T = e_i \times (1 - FGD)$	$\mu^T = -\eta_{12}\mu_i$	$\sigma^T = \frac{\sigma_i}{\eta_{12}} + f(s \times \alpha)$
Absorption	$e^T = e_i \cdot e^{-\frac{\sigma_t \cdot h}{\sqrt{1-\lvert\mu_i\rvert^2}}}$	$\mu^T = -\mu_i$	$\sigma^T = \sigma_i$
Scattering	$e^T = e_i \cdot \frac{\sigma_s \cdot h}{\sqrt{1-\lvert\mu_i\rvert^2}} \cdot e^{-\frac{\sigma_t \cdot h}{\sqrt{1-\lvert\mu_i\rvert^2}}}$	$\mu^T = -\mu_i$	$\sigma^T = \sigma_i + \sigma_g$

4.2 Representation

Belcour's method represents an excellent candidate for the application of an evolutionary technique for the automatic generation of advanced materials. The method structure is modular and compact, and it is characterized by a consistent description of its different components, and by a limited and intuitive set of variables. In [5], the author considers only a top and bottom layer for the implementation of the method for real-time rendering. In this configuration, the reflectance characteristics of the bottom layer represent an approximated combination of multiple other scattering phenomena. Moreover, an optional participating media, acting as the third layer, can be considered between the top and bottom layer. In our GA, we follow the same approach. As a consequence, the genotype of our GA is composed of 12 floating point chromosomes:

- η_1: refractive index of the top layer
- η_2: refractive index of the bottom layer
- α_1: roughness of the top layer
- α_2: roughness of the bottom layer
- h: the depth of the partecipating media layer between the top and bottom layers
- σ_s^R, σ_s^G, σ_s^B: scattering cross section values in the R, G and B channels
- σ_a^R, σ_a^G, σ_a^B: absorption values in the R, G and B channels (involved in the computation of σ_t in Table 1)
- g: anisotropic factor (involved in the computation of σ_g in Table 1)

If a layered material has no participating media, then the final appearance is controlled only by η_1, η_2, α_1, and α_2 chromosomes. If a texture is applied to one of the layers, then the roughness value to apply is extracted from the texture; and

α_1 or α_2 are used as weights multiplied to the texel value in order to control the overall roughness effect.

4.2.1 Empirical analysis of the parameters of Belcour's method

In [5], the author presents clearly the limits of the method, and the particular cases which are not adequately simulated. Moreover, the setup for real-time rendering introduces an additional simplification of the scattering phenomena inside a material. With these premises, we have considered that setting the values of the chromosomes in our GA without constraints could lead to unpredictable results in the evolutionary process.

Thus, we have performed an accurate empirical analysis and pre-experimentation stage on the Belcour's method. The pre-experimentation was aimed at determining possible constraints in the value ranges to assign to the different chromosomes in our GA, given the characteristics of different target materials to simulate. Our analysis led to the conclusion that there are differences in the possible value ranges, on the basis of the *family* of the desired material. We call *roughcoat* a material with a smooth bottom layer (a metal or a plastic), covered by a rough top layer (rust, dirt, Venetian plaster, etc). This kind of material requires a high value of roughness in the top layer, and a lower one in the bottom. A *clearcoat* material is used to simulate materials like e.g., car paint, ceramic, lacquered wood. In this case, the roughness value of the top layer must be lower than the value of the bottom layer. Moreover, there are further differences, according to whether the bottom layer is a *conductor* (e.g., metal) or a *dielectric* (e.g., plastic or wood) material. We have summarized these rules and constraints in Table 2. It can be noticed how the combination of material family and type of the bottom layer affects the possible range of values of the chromosomes. A particular attention must be given to the values (and their relative differences) of the refractive indices of the layers.

4.3 Algorithm

Our proposed GA follows a standard approach. First of all, given a target material, an initial population of N individuals of the same material family is created. The values of the chromosomes of each individual are generated randomly, but respecting the constraints listed in Table 2. As shown in the experimental evaluation, described in Section 5, an adequate value for N is between 50 and 100. Each individual is then ranked using a fitness function (described in Section 4.4).

We apply *tournament selection* with tournament size 4 and $p = 1$. The choice of a deterministic tournament leads to the selection of only the best individuals. However, considering only a limited set of participants in the tournament allows to maintain a high level of variety in the selected individuals, which is the desired behaviour of the proposed approach.

Each couple of parents has a 0.9 probability to generate offsprings. We set a very high probability in order to further enhance the variety of the population. If the crossover is applied, we apply a *uniform crossover*. Each chromosome has a 0.25 probability of being swapped among the two parents. With this probability value, at least one chromosome is almost always swapped, but it is highly unlikely

Table 2: Rules and constraints for the values of the material chromosomes.

	Bottom Layer: conductor	**Bottom Layer: dielectric**	**Layer with textured roughness**
roughcoat	$\alpha_1 \geq 0.1$ $\alpha_2 < 0.1$ $1.0 < \eta_1 \leq 2.0$ $\lvert\eta_1 - \eta_2\rvert \in (0.0, 1.5]$[a]	$\alpha_1 \geq 0.1$ $\alpha_2 < 0.01$ $1.0 < \eta_1 \leq 2.0$ $\lvert\eta_1 - \eta_2\rvert \in (0.0, 0.5]$[b] $\eta_2 \geq 1.0$	top layer
clearcoat	$\alpha_1 < 0.1$ $\alpha_2 \geq 0.1$ $1.0 < \eta_1 \leq 2.0$ $\lvert\eta_1 - \eta_2\rvert \in (0.0, 1.5]$[a]	$\alpha_1 < 0.01$ $\alpha_2 \geq 0.01$ $1.0 < \eta_1 \leq 3.0$ $\lvert\eta_1 - \eta_2\rvert \in (0.0, 3.0]$[b] $\eta_2 \geq 1.0$	bottom layer
roughcoat/ clearcoat with participating media	$h \in [0.05, 40.0]$[c] $\sigma_s^R, \sigma_s^G, \sigma_s^B \in [0.0, 1.0]$ $\sigma_s^R, \sigma_s^G, \sigma_s^B \in [0.0, 1.0]$ $\eta_2 < 1.0$[d]	$h \in [0.05, 40.0]$[c] $\sigma_s^R, \sigma_s^G, \sigma_s^B \in [0.0, 1.0]$ $\sigma_s^R, \sigma_s^G, \sigma_s^B \in [0.0, 1.0]$	—

[a] we set η_2 slightly less than η_1 to favor the internal reflection, without the dependance from the attenuation factor of the refractive index.
[b] we set η_2 greater than η_1 to favor light transmission for subsurface scattering.
[c] $h \in [0.0, 1.0]$ favors the color given by scattering cross section (with less absorption). $h \in [2.0, 40.0]$ favors the color given by transmission cross section. With $h = 40.0$: full light absorption, only the top layer is visible.
[d] with $\eta_2 < 1.0$ the effect of the partecipating media on the color is more visible. But increasing the value of η_1, and lowering the value of η_2 such as $\eta_2 \ll 1.0$, the original color of the metal progressively disappears.

to have crossover applied to all the chromosomes at the same time. We apply two steps in the selection of individuals for the crossover operation in order to avoid issues with offsprings not following the constraints of Table 2. In the first step, for each individual, we search in the population the first material with refractive indices suitable to be swapped (i.e., with indices respecting the constraint on $|\eta_1 - \eta_2|$). If this individual is found, then a uniform crossover is applied. In the second step, a matching process is applied sequentially to create couples among the individuals excluded from the first step. For these couples, we apply uniform crossover excluding from the process the chromosomes related to the refractive indices.

The mutation on the generated offsprings can occur with a 0.4 probability. If activated, each chromosome has a 0.1 probability to mutate. If a chromosome is

subject to mutation, a new value is created randomly. The new value must follow the rules and constraints of Table 2.

Considering the goal of the proposed GA, we have set a fixed number of generations as the termination condition of the evolutionary process.

4.4 Fitness functions

The goal of the proposed evolutionary approach is to generate the parameters for a new material that shares the physical behaviour of the target one, but with a perceivable difference. Thus, the fitness function must be chosen in order to measure in a simple but effective way the distance between the target and generated genotypes. However, the function must be flexible enough to allow further refinements of the process, allowing adaptation to different situations and applications (e.g., applying different weights to specific subsets of chromosomes). Thus, we have decided to consider some well-known vector distances between the target material and the generated individual:

- the *Chebyshev distance*

$$d_{ch}(g, tm) = \max_i \{|g[i] - tm[i]|\} \tag{4}$$

- the *Euclidean distance*

$$d_{eu}(g, tm) = \sqrt{\sum_i (g[i] - tm[i])^2} \tag{5}$$

- the *Manhattan distance*

$$d_{ma}(g, tm) = \sum_i |g[i] - tm[i]| \tag{6}$$

where in Eq. (4), (5) and (6) g is an individual, tm is the target material, and i represents each of the chromosomes.

In Section 5, we present an experimental setup aimed at investigating the effect of the distances on the final perceptual differences of the generated materials.

5 Experimental evaluation

In order to evaluate the effect of the considered fitness functions in the selection of the best individuals in the population, we have set up a test scene consisting of a sphere, illuminated by a single light. We have then selected three target materials:

- **validation metal**, a clearcoat metal with no texturing. The values for the chromosomes are $\eta_1 = 1.2$, $\eta_2 = 0.8$, $\alpha_1 = 0.03$, $\alpha_2 = 0.1$
- **validation painted metal**, a roughcoat metal with a roughness texture applied in the top layer. The values for the chromosomes are $\eta_1 = 1.2$, $\eta_2 = 0.8$, $\alpha_1 = 2.5$, $\alpha_2 = 0.099$

– **validation metal depth**, a clearcoat metal with a participating media between the top and bottom layers. No roughness texture is considered, but a texture is used as base color of the sphere. The values for the chromosomes are $\eta_1 = 1.460$, $\eta_2 = 0.8$, $\alpha_1 = 0.05$, $\alpha_2 = 0.45, h = 1.5, \sigma_s^R = 0.1, \sigma_s^G = 1.0, \sigma_s^B = 0.0, \sigma_a^R = 0.0, \sigma_a^G = 1.0, \sigma_a^B = 1.0, g = 0.8$

We have considered two different target materials without the presence of the participating media, because this feature has a relevant effect on the final color of the generated material. As a consequence, the variety in the generated individuals is evidently higher for this class of materials. Without the participating media, any perceptual difference between the original and generated materials is given only by the core chromosomes of the top and bottom layers, which represents a more tricky situation to manage. In any case, due to their characteristics, the considered target materials represent optimal candidates to test the efficacy of the evolutionary approach. Figures 1 and 2 show the test scene with the target materials applied to the sphere.

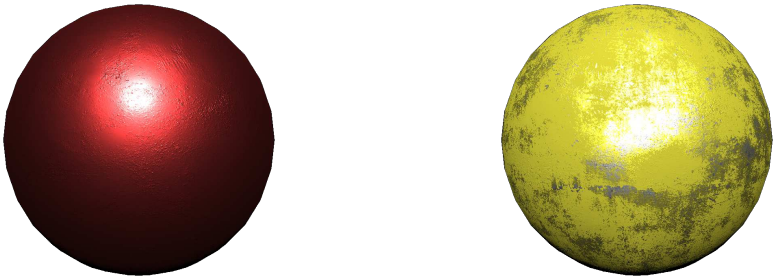
For all the three target materials, we have generated a set of new materials using the proposed GA, applying all the considered fitness functions. For each combination of a target material and applied distance, we have performed 36 runs of the GA, with a population of 50 individuals, and termination after 5 generations. Every 9 runs, to introduce a higher variability, we have changed the seed of the pseudo-random number generator, used in the initialization of the population and in the mutation step. For each run, we have then selected the material with the best fitness (i.e., with minimum distance to the target material). Figures 9-17 show some examples of these generated materials.

We have then applied the *CIEDE2000* $\Delta E_0^* 0$ difference [25] to evaluate the differences between the generated and target materials. $\Delta E_0^* 0$ is a measure proposed in colorimetry for the perceptual difference among two colors. Its value ranges from 0 and 100:

– $\Delta E_0^* 0 \leq 1.0$: color difference is not perceptible by human eyes
– $\Delta E_0^* 0 \in [1.0, 2.0]$: color difference is perceptible through close observation
– $\Delta E_0^* 0 \in [2.0, 10.0]$: color difference is perceptible at a glance
– $\Delta E_0^* 0 \in [10.0, 49.0]$: colors are more similar than opposite
– $\Delta E_0^* 0 \in [49.0, 100.0]$: colors are exact opposite

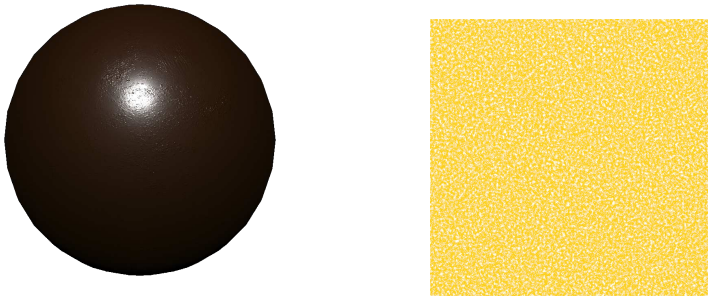
We have applied $\Delta E_0^* 0$ between each pixel of the image rendered using the target material, and the corresponding pixels in each of the images created using the generated materials parameters. Then, we have calculated the mean $\Delta E_0^* 0$ value for each image couple, by averaging the $\Delta E_0^* 0$ values on the single pixels. In the rendered images, we have set the background of the scene as fully transparent, in order to calculate the mean $\Delta E_0^* 0$ only on the colors generated on the sphere. This is an acceptable choice for a numeric measure, where no perceptual tests with real observers are considered. When human subjects are involved, the choice of the background color and/or pattern has been proven to be a delicate choice, because of the contextual nature of human color perception [38].

In Fig. 3, 4, and 5, we have plotted the mean $\Delta E_0^* 0$ values. For **validation metal** and **validation painted metal** materials (Fig. 3 and 4), it can be noticed how the applied parameters (population of 50 individuals, termination after 5 generations) are able to produce at least half of the materials with evident perceptual



(a) **validation metal** (b) **validation painted metal**

Fig. 1: (a): Test scene with **validation metal** material applied to the sphere. The chromosomes of the material are $\eta_1 = 1.2$, $\eta_2 = 0.8$, $\alpha_1 = 0.03$, $\alpha_2 = 0.1$.
(b): Test scene with **validation painted metal** material applied to the sphere. The chromosomes of the material are $\eta_1 = 1.2$, $\eta_2 = 0.8$, $\alpha_1 = 2.5$, $\alpha_2 = 0.099$. In this case, α_1 is used as a weight to be multiplied to the texel value from the roughness texture.



(a) (b)

Fig. 2: (a): Test scene with **validation metal depth** material applied to the sphere . The values for the chromosomes are $\eta_1 = 1.460$, $\eta_2 = 0.8$, $\alpha_1 = 0.05$, $\alpha_2 = 0.45, h = 1.5, \sigma_s^R = 0.1, \sigma_s^G = 1.0, \sigma_s^B = 0.0, \sigma_a^R = 0.0, \sigma_a^G = 1.0, \sigma_a^B = 1.0, g = 0.8$.
(b): The texture used as base color of the sphere in (a). In the **validation metal depth** material, no roughness texture is considered.

differences with the target one, and an adequate number of materials with more subtle differences. This is in line with the intended behaviour: the generation of new materials with an adequate resemblance with the reference, but presenting perceptual differences ranging from low to moderate. With the same parameters, the results obtained for the **validation metal depth** material (Fig. 5) are different. We still have approximately half of the generated materials with an average level of perceptual differences, but, in this case, the remaining individuals are more evidently different than the target material. This is due to the absorption and scattering of the participating media, which have a relevant role in the determination of the final color of the object. Comparing the considered fitness functions, we can

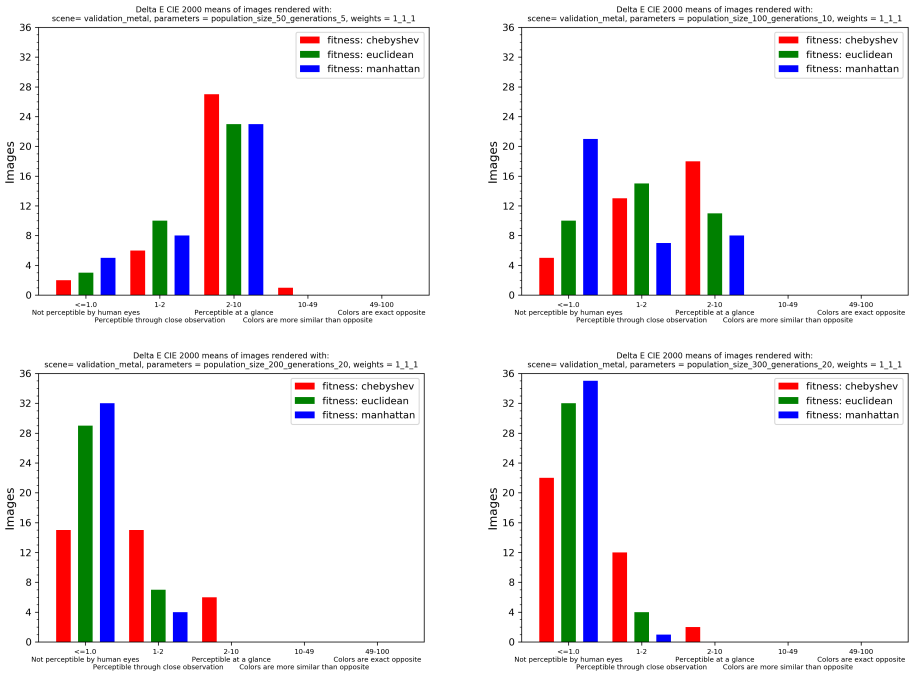


Fig. 3: Comparison of the mean ΔE^*_{00} values using the three distance functions, between the test scene rendered with **validation metal** and the images with the generated materials, increasing the population size and the number of generations.

conclude that, with the initial parameters, the Chebyshev distance tends to generate more individuals centered in the middle range of the $\Delta E^*_0 0$ values. Using the Euclidean and Manhattan distances, we obtain more individuals close to the target one, in case of **validation metal** and **validation painted metal**. Interestingly, we observe the opposite behaviour for the **validation metal depth** material.

In Fig. 3, 4, and 5, we also show the resulting mean $\Delta E^*_0 0$ values repeating the whole experimental setup with increasing values of population size and number of generations. For the **validation metal** and **validation painted metal** materials, the plots confirm that the generations using the Euclidean and Manhattan distances as fitness functions converge faster to materials perceptually undistinguishable from the original, while the Chebyshev distance generates more individuals with an average level of difference. In any case, with a population larger than 100 individuals and with a number of generations higher than 10, the evolutionary approach generates mainly materials identical to these two references, which is not the desired result. Again, we notice a different trend in the case of the **validation metal depth** target material. With a population of 300 individuals, and termination after 20 generations (Fig. 5), we notice how most of the generated individuals have an average value of $\Delta E^*_0 0$. However, in this case the Chebyshev distance seems to perform slightly worse than the other ones. In Fig. 6, we show the resulting mean $\Delta E^*_0 0$ values for **validation metal depth**, setting an initial population

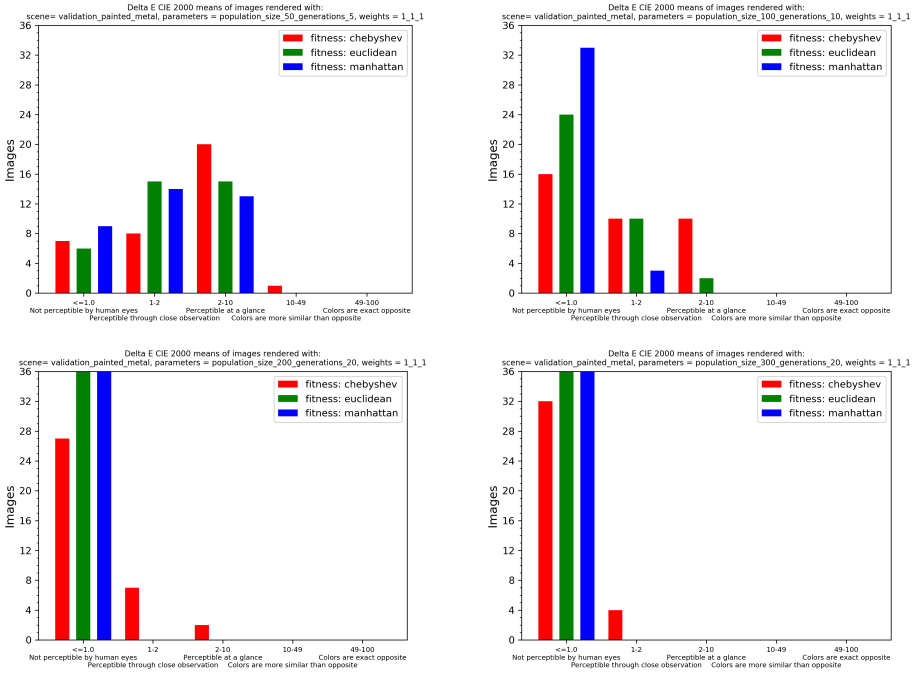


Fig. 4: Comparison of the mean ΔE^*_{00} values using the three distance functions, between the test scene rendered with **validation painted metal** and the images with the generated materials, increasing the population size and the number of generations.

of 500 individuals and termination after 50 generations. The plot is very similar to the results obtained with the other two target materials, but with parameters 10 times smaller: in this case the Chebyshev distance is more equilibrate, while, with the Euclidean and Manhattan distances, the generated materials are too similar to the target ones.

To better understand the perceptual differences of the materials created by the proposed evolutionary approach, we show in Fig. 9-17 some representatives of the generated materials with different levels of $\Delta E^*_0 0$ values, for each target material and for each considered fitness function. We show also a probability map generated by the HDR-VDP-2 visual metric [21]. HDR-VDP-2 compares a reference and a test image, and predicts what is the probability that the differences between the images are visible for an average observer. Thus, each image in the bottom row of Fig. 9-17 represents a per-pixel measure of the perceptual difference between the target materials and the materials generated by the proposed GA. The colors used in the probability maps range from red (high probability) to green (low probability). Even if we have set a transparent background in our rendered images, HDR-VDP-2 performs its evaluation on all the pixels given in input. As a consequence, the probability maps present a color code also in each pixel of the backgrounds. These values are not significative due to the absence of an actual

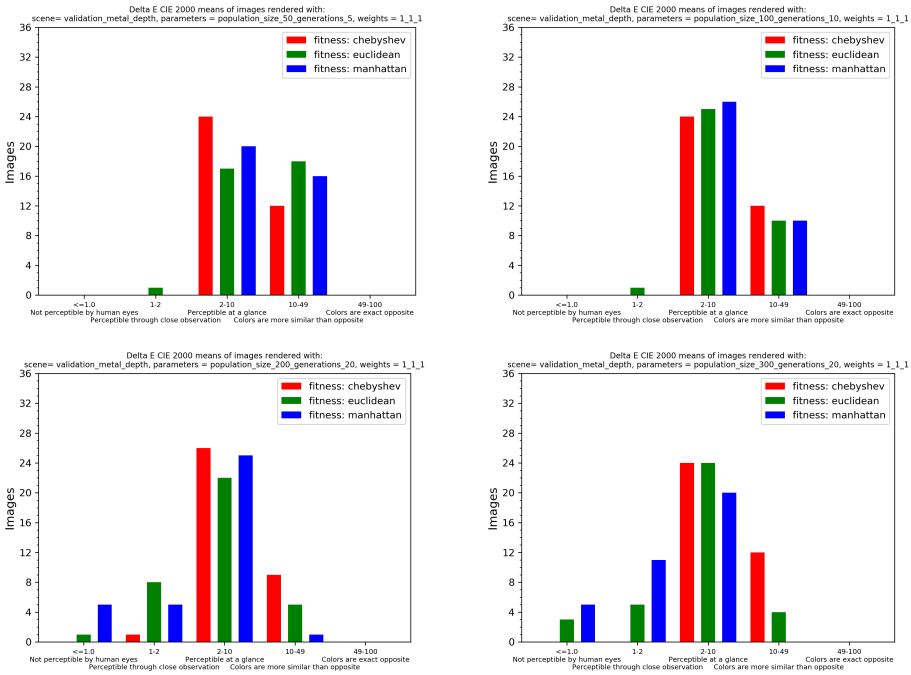


Fig. 5: Comparison of the mean ΔE^*_{00} values using the three distance functions, between the test scene rendered with **validation metal depth** and the images with the generated materials, increasing the population size and the number of generations.

color in the input images and, as a consequence, they must not be considered in the visual evaluation of the differences.

6 Implementation details and performance

The presented approach is well suited to be integrated into a real-time rendering pipeline: in fact, the GA is executed separately from the rendering stage, and it provides the final parameters to be applied to the Belcour's method [5] for the rendering of the final layered material. As a consequence, the evolutionary method can be executed in any stage computed on the CPU prior to the final rendering on GPU. For example, it can be executed during the loading of the meshes of a scene, or it can be integrated into the PCG stage for the generation of several instances of a model. To test our approach, we have developed a simple C++ application based on the recent Vulkan API [45] for real-time rendering. The application exploits a deferred rendering approach [2] for a computationally efficient rendering of complex scenes. The application reads from an external text file:

– the name of the models' files to load

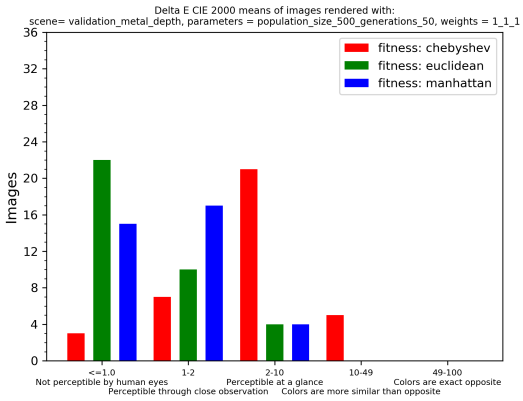


Fig. 6: Comparison of the mean ΔE^*_{00} values for the test scene rendered with **validation metal depth** and the images with the generated materials, with population size 500 and the 50 generations. In this case, the number of materials with less perceivable differences is in line with the results of the other two test materials, but with higher values of the GA parameters.

- the values of population size for the GA
- the number of generations to use as termination
- the chosen fitness function

For testing purposes, a limited number of predefined scene configurations (i.e., a fixed number of instances, with predefined positions in the scene) have been considered. The application, after loading the initial parameters and the meshes from disk, performs the GA on the CPU. Then, the generated parameters are passed as *uniform* variables to a shader computing the Belcour's method on the GPU. The shader for the Vulkan API has been implemented using the information from the Belcour's paper [5] and the provided supplemental material. Figures 7 and 8 show two configurations of the considered test scene for the performance analysis.

On an Intel Quad-Core i7-6700HQ machine with 8 GB DDR4 RAM, equipped with an NVIDIA GeForce GTX 970M with 3GB GDDR5 VRAM, the GA took around 30 ms to generate a material using a population of 100 individuals and 10 generations. The required computational time is dependent on the number of individuals and generations. Since the rendering stage uses parameters generated by the GA without any further computation, the detected performances of the shader implementing the Belcour's method, during real-time rendering and navigation in the test scenes, are in line with the one stated in the original paper.

7 Conclusion and future work

In this paper, we have shown how a Genetic Algorithm (GA) can be applied efficiently to evolve the parameters of a BSDF in order to generate different versions of a target material presenting a moderate amount of perceptual differences. Some results of the proposed approach can be seen in Fig. 7 and 8.

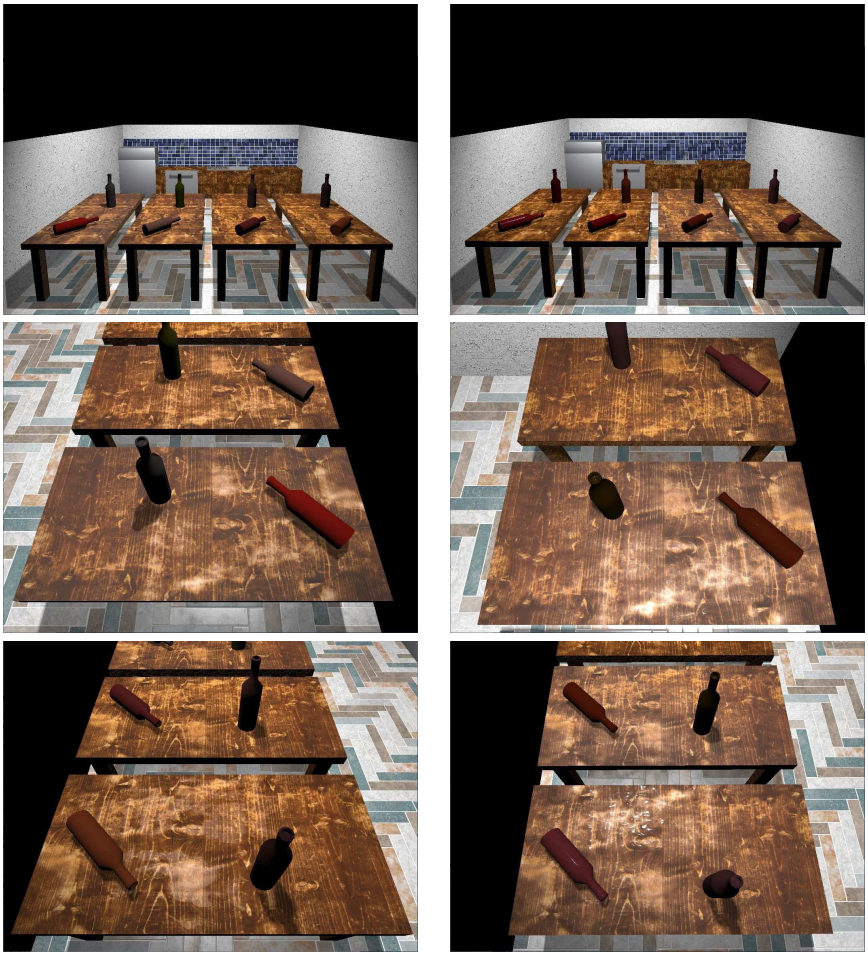


Fig. 7: Different versions of *roughcoat* and *clearcoat* materials generated with the proposed approach and applied to instances of the same model.

The proposed approach is based on a very limited and intuitive set of parameters, whose effect and contribution to the final generation is consistent. By changing the number of individuals and the number of generations of the GA, we can change the amount of perceptual differences in the generated materials. The computational time needed to generate a new material is adequate for a real-time rendering application, even if some trade-off must be considered for complex materials characterized by the presence of participating media. These cases may require higher values (and, as a consequence, longer computational times) of the GA parameters to set the desired level of perceptual difference in the generated results. In the evaluation of the proposed GA, we have seen how the choice of the fitness function leads to different behaviours in the overall GA computation. However, this can be seen as an additional control parameter for the final user,

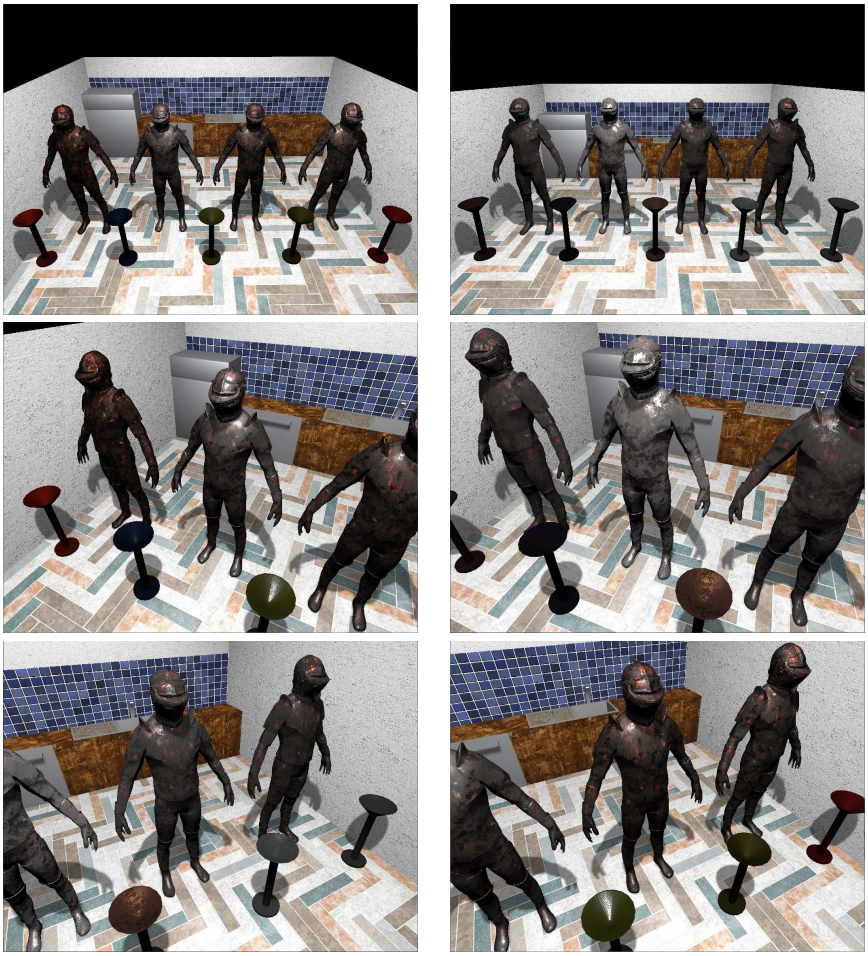


Fig. 8: Different versions of *roughcoat* and *clearcoat* materials generated with the proposed approach and applied to instances of the same model.

who can decide about the generation of materials more or less close to the target one, by selecting a different distance function.

Other methods may be considered for a random generation of instances of a target material, like e.g., using a simple random perturbation of the BSDF parameters. However, this kind of approaches may generate a higher number of unwanted/inadequate samples during the generation process, requiring thus longer time to converge to the desired result. Moreover, it could be difficult, if not impossible, to have control of the final desired amount of perceptual differences to introduce in the generated materials.

In future research, we will consider the effect of other distance functions and we will test the effects of different weights applied to the chromosomes in the fitness functions. This will allow a final user to select the more relevant features to be considered in the selection of the best individuals.

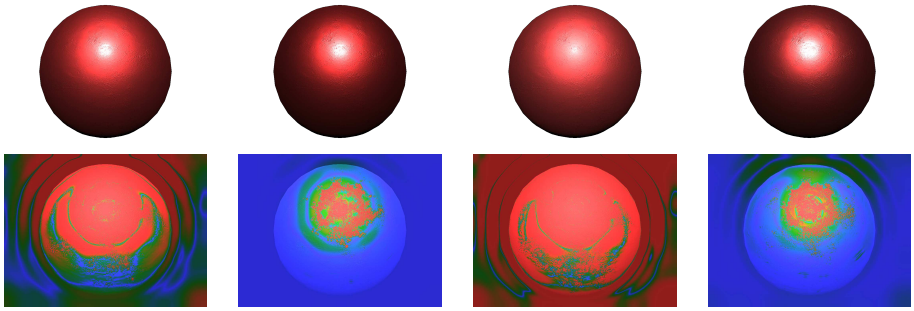


Fig. 9: A subset of the generated materials (population: 50, generations: 5), with target material **validation metal** and Chebyshev distance, and the probability maps generated by HDR-VDP-2 visual metric.

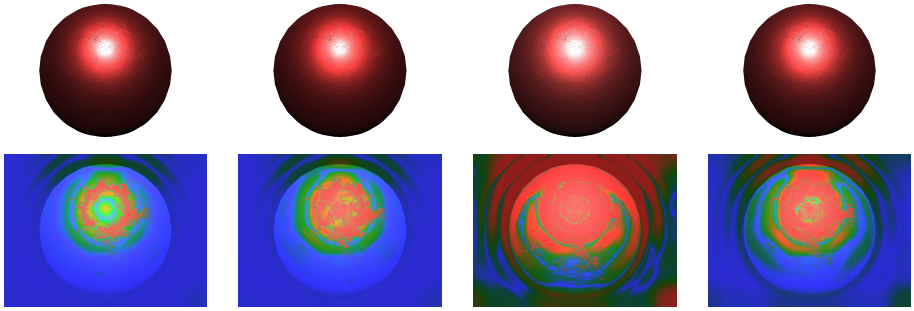


Fig. 10: A subset of the generated materials (population: 50, generations: 5), with target material **validation metal** and Euclidean distance, and the probability maps generated by HDR-VDP-2 visual metric.

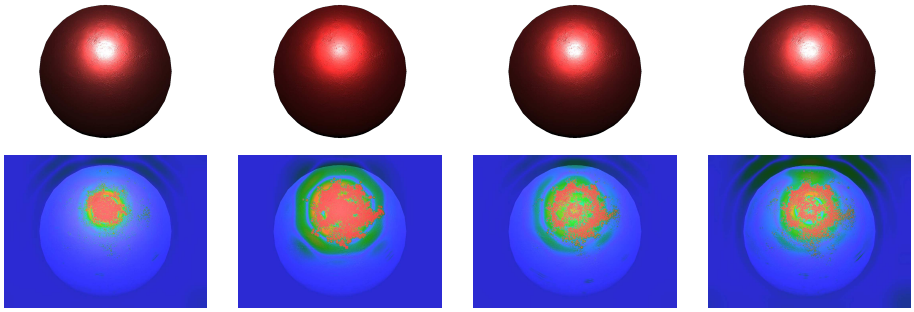


Fig. 11: A subset of the generated materials (population: 50, generations: 5), with target material **validation metal** and Manhattan distance, and the probability maps generated by HDR-VDP-2 visual metric.

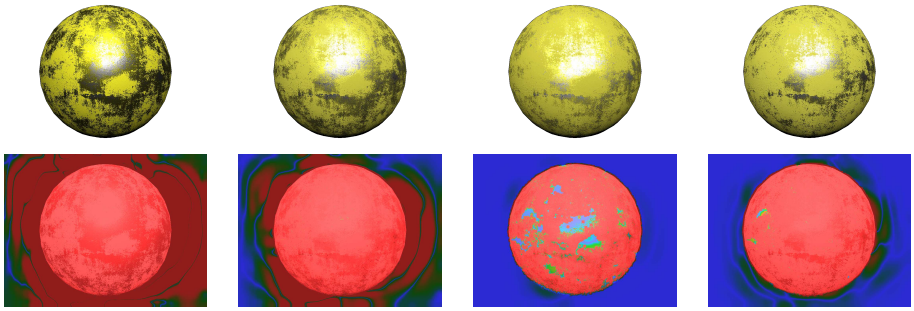


Fig. 12: A subset of the generated materials (population: 50, generations: 5), with target material **validation painted metal** and Chebyshev distance, and the probability maps generated by HDR-VDP-2 visual metric.

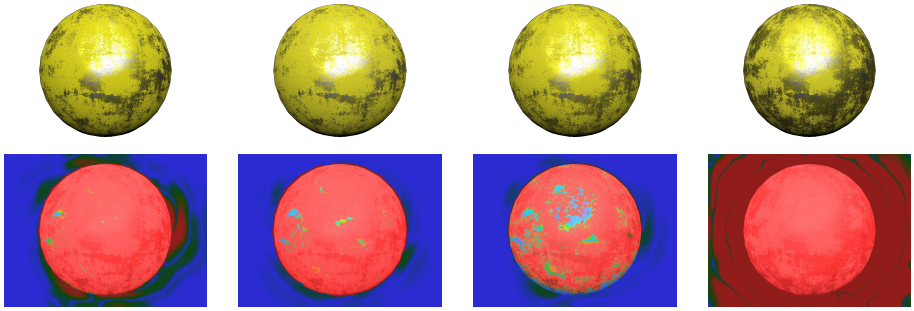


Fig. 13: A subset of the generated materials (population: 50, generations: 5), with target material **validation painted metal** and Euclidean distance, and the probability maps generated by HDR-VDP-2 visual metric.

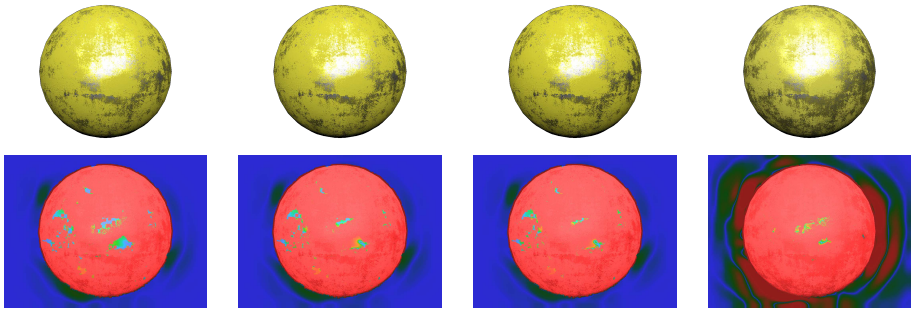


Fig. 14: A subset of the generated materials (population: 50, generations: 5), with target material **validation painted metal** and Manhattan distance, and the probability maps generated by HDR-VDP-2 visual metric.

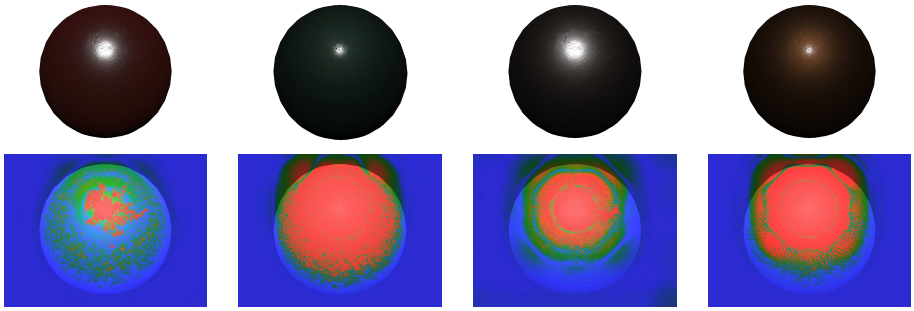


Fig. 15: A subset of the generated materials (population: 50, generations: 5), with target material **validation metal depth** and Chebyshev distance, and the probability maps generated by HDR-VDP-2 visual metric.

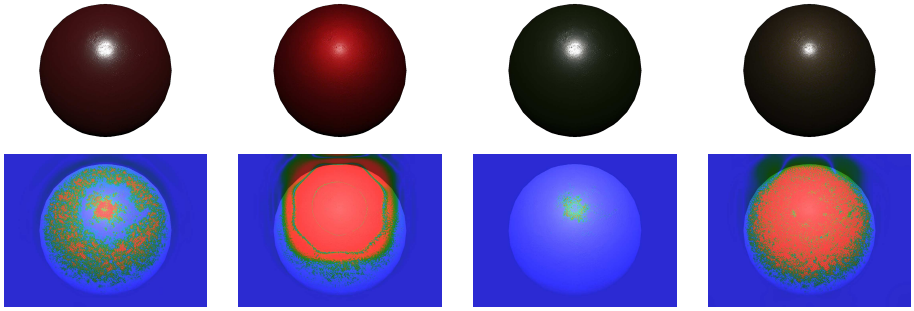


Fig. 16: A subset of the generated materials (population: 50, generations: 5), with target material **validation metal depth** and Euclidean distance, and the probability maps generated by HDR-VDP-2 visual metric.

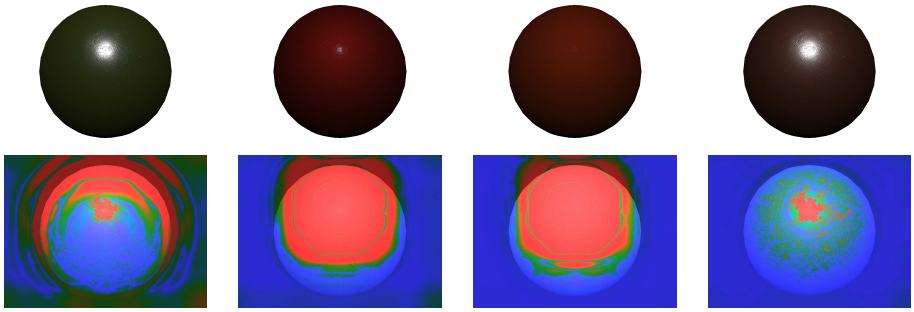


Fig. 17: A subset of the generated materials (population: 50, generations: 5), with target material **validation metal depth** and Manhattan distance, and the probability maps generated by HDR-VDP-2 visual metric.

References

1. Agliata, F., Bertoli, M., Ripamonti, L.A., Maggiorini, D., Gadia, D.: Adding variety in NPCS behaviour using emotional states and genetic algorithms: The Genie project. In: Proceedings of GAME-ON Conference on Games 2019, pp. 45–50 (2019)
2. Akenine-Möller, T., Haines, E., Hoffman, N., Pesce, A., Iwanicki, M., Hillaire, S.: Real-Time Rendering 4th Edition. A K Peters/CRC Press (2018)
3. Andrade, G., Ramalho, G., Santana, H., Corruble, V.: Automatic computer game balancing: A reinforcement learning approach. In: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '05, pp. 1111–1112. ACM, New York, NY, USA (2005)
4. Baldwin, A., Dahlskog, S., Font, J.M., Holmberg, J.: Mixed-initiative procedural generation of dungeons using game design patterns. In: 2017 IEEE Conference on Computational Intelligence and Games (CIG), pp. 25–32 (2017)
5. Belcour, L.: Efficient rendering of layered materials using an atomic decomposition with statistical operators. ACM Trans on Graph **37**(4), 1 (2018)
6. Bernardi, A., Gadia, D., Maggiorini, D., Ripamonti, L.A.: Using a genetic algorithm for the procedural generation of layered materials for real-time rendering. In: Proceedings of GAME-ON Conference on Games 2019, pp. 29–36 (2019)
7. Botta, M., Gautieri, V., Loiacono, D., Lanzi, P.L.: Evolving the optimal racing line in a high-end racing game. In: 2012 IEEE Conference on Computational Intelligence and Games (CIG), pp. 108–115 (2012)
8. Brady, A., Lawrence, J., Peers, P., Weimer, W.: genBRDF: Discovering new analytic BRDFs with Genetic Programming. ACM Trans. Graph. **33**(4), 114:1–114:11 (2014)
9. de Carvalho, L.F.B.S., Neto, H.C.S., Lopes, R.V.V., Paraguaçu, F.: An application of genetic algorithm based on abstract data type for the problem of generation of scenarios for electronic games. In: 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems, vol. 2, pp. 526–530 (2010)
10. Chen, G., Esch, G., Wonka, P., Müller, P., Zhang, E.: Interactive procedural street modeling. ACM Trans. Graph. **27**(3), 103:1–103:10 (2008)
11. Compton, K., Mateas, M.: Procedural level design for platform games. In: Proceedings of the Second AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE'06, pp. 109–111. AAAI Press (2006)
12. De Francesco, A., Ripamonti, L.A., Gadia, D., Maggiorini, D.: A.T.L.A.S.: Automatic Terrain and Labels Assembling Software. In: Proceedings of the 3rd Workshop on Games-Human Interaction (GHItaly19), no. 2480 in CEUR Workshop Proceedings (2019)
13. Ebert, D.S., Musgrave, F.K., Peachey, D., Perlin, K., Worley, S.: Texturing & Modeling: a procedural approach. Morgan Kaufmann (2003)
14. Frade, M., de Vega, F.F., Cotta, C.: Automatic evolution of programs for procedural generation of terrains for video games. Soft Computing **16**(11), 1893–1914 (2012)
15. Galactic Arms Race homepage: (2019). http://galacticarmsrace.blogspot.it
16. Grant, I.P., Hunt, G.E., Flowers, B.H.: Discrete space theory of radiative transfer I. Fundamentals. Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences **313**(1513), 183–197 (1969). DOI 10.1098/rspa.1969.0187
17. Guarneri, A., Maggiorini, D., Ripamonti, L.A., Trubian, M.: GOLEM: Generator of life embedded into MMOs. In: Proceedings of the Twelfth European Conference on the Synthesis and Simulation of Living Systems: Advances in Artificial Life, ECAL, pp. 585–592 (2013)
18. Hastings, E.J., Guha, R.K., Stanley, K.O.: Automatic content generation in the Galactic Arms Race video game. IEEE Transactions on Computational Intelligence and AI in Games **1**(4), 245–263 (2009)
19. Krecklau, L., Kobbelt, L.: Procedural modeling of interconnected structures. Computer Graphics Forum **30**(2), 335–344 (2011). DOI 10.1111/j.1467-8659.2011.01864.x
20. Lee, S., Jung, K.: Dynamic game level design using gaussian mixture model. In: PRICAI 2006: Trends in Artificial Intelligence, pp. 955–959. Springer (2006)
21. Mantiuk, R., Kim, K.J., Rempel, A.G., Heidrich, W.: HDR-VDP-2: a calibrated visual metric for visibility and quality predictions in all luminance conditions. In: ACM Transactions on graphics (TOG), vol. 30, p. 40. ACM (2011)
22. Masia, B., Munoz, A., Tolosa, A., Anson, O., Lopez-Moreno, J., Jimenez, J., Gutierrez, D.: Genetic algorithms for estimation of reflectance parameters. In: Proceedings of the 2009 Spring Conference on Computer Graphics (SCCG09) (2009)

23. Mazza, C., Ripamonti, L.A., Maggiorini, D., Gadia, D.: FUN PLEdGE 2.0: A FUNny Platformers LEvels GEnerator (rhythm based). In: Proceedings of the 12th Biannual Conference on Italian SIGCHI Chapter (CHItaly '17), pp. 22:1–22:9. ACM (2017)
24. Missura, O., Gärtner, T.: Player modeling for intelligent difficulty adjustment. In: Discovery Science: 12th International Conference, pp. 197–211. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
25. Mokrzycki, W., Tatol, M.: Colour difference Delta E - a survey. Machine Graphics and Vision **20**(4), 383–411 (2011)
26. Mora, A.M., Montoya, R., Merelo, J.J., Sánchez, P.G., Castillo, P.Á., Laredo, J.L.J., Martínez, A.I., Espacia, A.: Evolving bot AI in Unreal™. In: Applications of Evolutionary Computation: EvoApplicatons 2010, pp. 171–180. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
27. Mora, A.M., Moreno, M.A., Merelo, J.J., Castillo, P.A., Arenas, M.G., Laredo, J.L.J.: Evolving the cooperative behaviour in Unreal™ bots. In: Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, pp. 241–248 (2010)
28. Mourato, F., dos Santos, M.P., Birra, F.: Automatic level generation for platform videogames using genetic algorithms. In: Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology, ACE '11, pp. 8:1–8:8. ACM (2011)
29. Müller, P., Wonka, P., Haegler, S., Ulmer, A., Van Gool, L.: Procedural modeling of buildings. ACM Trans. Graph. **25**(3), 614–623 (2006)
30. Norton, D., Ripamonti, L.A., Ornaghi, M., Gadia, D., Maggiorini, D.: Monsters of Darwin: A strategic game based on artificial intelligence and genetic algorithms. In: Proceedings of the 1st Workshop on Games-Human Interaction (GHItaly 2017), no. 1956 in CEUR Workshop Proceedings (2017)
31. Parish, Y.I.H., Müller, P.: Procedural modeling of cities. In: Proceedings of SIGGRAPH'01, pp. 301–308. ACM (2001)
32. Pea, J.M., Viedma, J., Muelas, S., LaTorre, A., Pea, L.: Designer-driven 3d buildings generated using variable neighborhood search. In: 2014 IEEE Conference on Computational Intelligence and Games, pp. 1–8 (2014)
33. Pharr, M., Wenzel, J., Humphreys, G.: Physically Based Rendering: From Theory to Implementation 3th Edition. Morgan Kaufmann (2016)
34. Piergigli, D., Ripamonti, L.A., Maggiorini, D., Gadia, D.: Deep reinforcement learning to train agents in a multiplayer first person shooter: some preliminary results. In: Proceedings of IEEE Conference on Games (CoG) 2019, pp. 1–8 (2019)
35. Prusinkiewicz, P., Lindenmayer, A.: The Algorithmic Beauty of Plants. electronic version (2004)
36. Ripamonti, L.A., Gratani, S., Maggiorini, D., Gadia, D., Bujari, A.: Believable group behaviours for NPCs in FPS games. In: Proceedings of IEEE Digital Entertainment, Networked Virtual Environments, and Creative Technology Workshop (DENVECT 2017) (2017)
37. Ripamonti, L.A., Mannalà, M., Gadia, D., Maggiorini, D.: Procedural content generation for platformers: designing and testing FUN PLEdGE. Multimedia Tools and Applications **76**(4), 5001–5050 (2017)
38. Rizzi, A., Bonanomi, C., Gadia, D., Riopi, G.: YACCD2: yet another color constancy database updated. In: Color Imaging XVIII: Displaying, Processing, Hardcopy, and Applications, Proc. of IS&T/SPIE's Symposium on Electronic Imaging, pp. 86520A–86520A–10. SPIE (2013)
39. Scalabrin, M., Ripamonti, L.A., Maggiorini, D., Gadia, D.: Stereoscopy-based procedural generation of virtual environments. In: Proceedings of IS&T's Stereoscopic Displays and Applications XXVII (28^{th} Symposium on Electronic Imaging : Science and Technology), 5, pp. 1–7 (2016)
40. Schwarz, M., Müller, P.: Advanced procedural modeling of architecture. ACM Trans. Graph. **34**(4), 107:1–107:12 (2015)
41. Shaker, N., Togelius, J., Nelson, M.J.: Procedural Content Generation in Games: A Textbook and an Overview of Current Research. Springer (2016)
42. Sitthi-Amorn, P., Modly, N., Weimer, W., Lawrence, J.: Genetic programming for shader simplification. ACM Trans. Graph. **30**(6), 152:1–152:12 (2011)
43. Snodgrass, S., Ontan, S.: Procedural level generation using multi-layer level representations with mdmcs. In: 2017 IEEE Conference on Computational Intelligence and Games (CIG), pp. 280–287 (2017)

44. Togelius, J., Nardi, R.D., Lucas, S.M.: Towards automatic personalised content creation in racing games. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games (2007)
45. Vulkan API homepage: (2019). https://www.khronos.org/vulkan/
46. Walter, B., Marschner, S.R., Li, H., Torrance, K.E.: Microfacet models for refraction through rough surfaces. In: Proceedings of the 18th Eurographics Conference on Rendering Techniques, EGSR'07, pp. 195–206 (2007)
47. Weber, J., Penn, J.: Creation and rendering of realistic trees. In: Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '95, pp. 119–128. ACM (1995)
48. Yannakakis, G.N., Hallam, J.: Real-time game adaptation for optimizing player satisfaction. IEEE Trans. Comput. Intellig. and AI in Games **1**(2), 121–133 (2009)