The Astrophysical Journal, 783:138 (10pp), 2014 March 10 doi:10.1088/0004-637X/783/2/138

INTRODUCING GAMER: A FAST AND ACCURATE METHOD FOR RAY-TRACING GALAXIES USING PROCEDURAL NOISE

N. E. Groeneboom and H. Dahle
Institute of Theoretical Astrophysics, University of Oslo, P.O. Box 1029 Blindern, N-0315 Oslo, Norway; nicolaag@astro.uio.no
Received 2013 April 24; accepted 2014 January 17; published 2014 February 24

ABSTRACT

We developed a novel approach for fast and accurate ray-tracing of galaxies using procedural noise fields. Our method allows for efficient and realistic rendering of synthetic galaxy morphologies, where individual components such as the bulge, disk, stars, and dust can be synthesized in different wavelengths. These components follow empirically motivated overall intensity profiles but contain an additional procedural noise component that gives rise to complex natural patterns that mimic interstellar dust and star-forming regions. These patterns produce more realistic-looking galaxy images than using analytical expressions alone. The method is fully parallelized and creates accurate high- and low- resolution images that can be used, for example, in codes simulating strong and weak gravitational lensing. In addition to having a user-friendly graphical user interface, the C++ software package GAMER is easy to implement into an existing code.

Key words: galaxies: general – gravitational lensing: strong – methods: numerical

Online-only material: color figures

1. INTRODUCTION

The generation of realistic artificial images of large numbers of galaxies is of considerable use for research in extragalactic astronomy and cosmology. The study of gravitational lensing by galaxies, galaxy clusters, and large-scale structure is a particularly relevant example. Ongoing and future large "cosmic shear" surveys such as Pan-STARRS, DES, KIDS, Hyper-Suprime-Cam, LSST, Euclid, and WFIRST require systematic development and testing of precise methods to derive the gravitational lensing shear based on measurements of galaxy shapes. The gravitational lens community has dealt with this challenge by setting up blind tests on increasingly large sets of mock data (e.g., Heymans et al. 2006; Massey et al. 2007; Bridle et al. 2009, 2010; Kitching et al. 2011; Mandelbaum et al. 2013). In most of these simulated data sets, the simulated gravitationally lensed galaxies are of a relatively simple form, being otherwise featureless combinations of de Vaucouleurs–type bulges (de Vaucouleurs 1948) and exponential disks that may or may not be coaligned in terms of centroid position and position angle, each with two-dimensional (2D) projected ellipticities drawn from an empirical distribution. There were several important motivations for using such relatively simple galaxy models, including the typically limited resolution of gravitationally lensed galaxies in seeing-limited ground-based data, and also for keeping the simulations fairly simple with a limited number of variable factors in order to better test and understand the relative performance of the methods being tested.

More realistic source galaxy morphologies were, for example, used in the simulations of Massey et al. (2007) and Meneghetti et al. (2008), who decomposed the morphologies of real galaxies observed with the *Hubble Space Telescope* (*HST*) into a set of orthogonal basis functions called "shapelets" (Refregier 2003; Massey et al. 2004) and produced mock images by generating galaxies with similar shapelet types. Mandelbaum et al. (2012) used actual *HST* images from the Cosmological Evolution Survey (Scoville et al. 2007) to simulate ground-based data from the Sloan Digital Sky Survey. An earlier, similar approach was taken by Bouwens et al. (2006). These approaches are suitable for simulating ground-based data, since the images are being convolved with a seeing-limited point-spread function (PSF) much larger than the *HST* PSF, making the effect of the latter fully removable (Mandelbaum et al. 2013). For space-based data, there is fundamental limitation in the lack of resolution beyond a certain spatial frequency: The optical transfer function (OTF) of diffraction-limited data falls to zero at a finite spatial frequency $k_{\max}$, beyond which we have no information about the structure of galaxies (Kaiser 2000). If we want to accurately simulate the appearance of galaxies that are magnified, such that information contained in some Fourier modes $k > k_{\max}$ are shifted below this cutoff scale, we cannot simply use random images of galaxies obtained from imaging having the same OTF, as we will then have no information on the intrinsic structure of galaxies on some scales that could still be observable post-lensing.

In this paper, we describe a novel approach to generate artificial images of galaxies that can overcome some of the limitations of earlier methods. The main differences from the studies quoted above are the modeling of realistic galaxy substructure to arbitrarily small scales by using procedural noise and the modeling of galaxies as multi-component, three-dimensional (3D) structures. Using different galaxy components with different spectral energy distributions (SEDs) naturally produces color gradients within the simulated galaxies. Color gradients may introduce a significant bias in future high-precision weak gravitational lensing measurements, an effect that has only been explored very recently (Voigt et al. 2012; Semboloni et al. 2013).

Generating galaxies with realistic substructure is also important for making simulations of the statistics of strongly gravitationally lensed galaxies in order to compare predictions of the numbers and properties of strongly gravitationally lensed arcs with observations (for a recent review on the use of arc statistics as a cosmological probe, see Meneghetti et al. 2013).

While gravitational lensing studies provide our main initial motivation for the development of our code, the 3D ray-tracing approach used here would be suitable for generating even more elaborate and realistic high resolution simulated galaxies. For example, this approach would allow the subsequent introduction of spatially varying stellar populations with intrinsic spectra

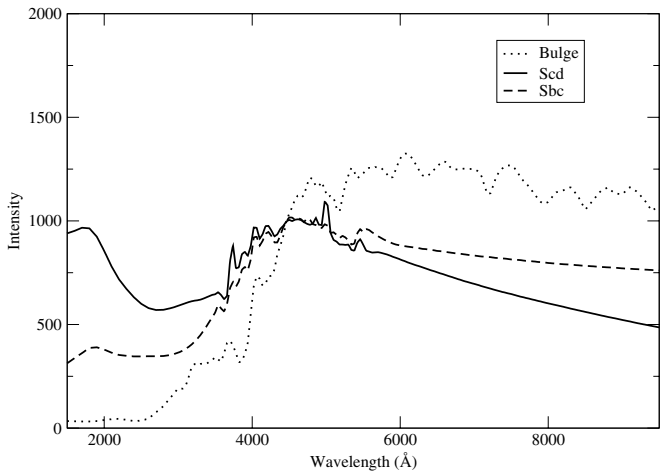

Figure 1. Three spectral energy distributions we have used in our code. The dotted line corresponds to the bulge component, the solid line (Scd) to the thin disk (younger blue stars) component, and the stippled line (Sbc) to the thick disk (older red stars). These are empirical SEDs measured by Coleman et al. (1980).

predicted by stellar population synthesis models (Bruzual & Charlot 2003) that are then ray-traced through a spatially varying dust component, even with the option of allowing for deviations from the mean extinction law along individual lines of sight within each galaxy (Cardelli et al. 1989).

Our aim with this paper is to present the basic methodology and demonstrate its ability to generate realistic artificial images of relatively local, well-resolved galaxies. Subsequent papers will focus on the production of individual galaxy morphologies based on empirical redshift-dependent distributions. This will include quantitative measures of how well the morphologies of our synthetic galaxies can be made to match observed lensed galaxies. The software package "GAMER" (Galaxy Morphology Easy Ray-tracer) will be released during 2014 and will be available on all OS X and UNIX platforms. Please see http://gamer.irio.co.uk for release details, documentation, images, and information.

In Section 2, we define our method and explain the ray-tracing scheme. We continue by giving a summary of the procedural noise methods used in GAMER in Section 2.1, before going through the individual components in Section 2.4. We define the input parameters for the code and describe how the pipeline operates. We also give a brief overview of the GAMER graphical user interface (GUI) in Section 3. In Section 4, we conclude our work and present our plans for future extensions and applications of GAMER.

2. METHODS

GAMER produces images by ray-tracing galaxies and their corresponding components through a galaxy sphere. First, we set up a galaxy object in 3D space, and we continue by defining global galaxy properties such as orientation and nominal arm structure. A galaxy can then be assigned a number of individual galaxy components from four categories: bulge, disk, star-forming regions, and dust. Each of these components are complete with individual parameters, such as arm thickness, azimuthal shift with respect to nominal arm position, height above galactic plane drop-off, exponential disk drop-off, intensity modifier, and procedural noise patterns. When the components are set up, they are assigned an SED, either from a template file or as an analytical expression. It is

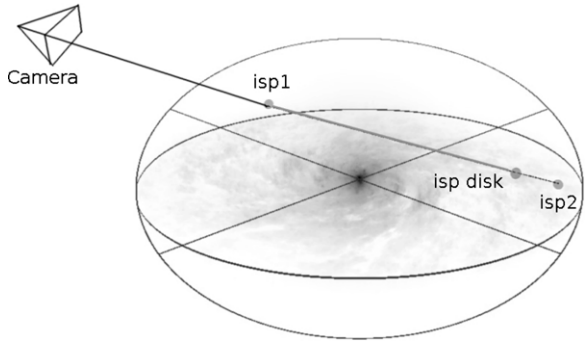

Figure 2. Ray-tracing scheme for a single galaxy. When a ray is cast from the camera, the code checks whether each ray intersects the galaxy sphere. If it does, we calculate the two intersection points on the sphere, together with the intersection points on the disk. The ray-tracing process continues by combining the component intensities from the furthest intersection point to the nearest.

now possible to extract the luminosity from any 3D point within the galaxy sphere by calculating the combined density from the galaxy components.

The process initializes the view matrix by using a camera, target, and up vector. The code projects rays for each pixel from the camera origin through the camera plane and checks whether the galaxy sphere is intersected. If the sphere is intersected, we calculate the two intersection points $\boldsymbol{i}_1, \boldsymbol{i}_2$ together with the intersection with the disk plane $\boldsymbol{i}_p$. The ray-tracing process starts at the back of the sphere at point $\boldsymbol{i}_2$ with direction $\boldsymbol{i}_2 - \boldsymbol{i}_1$. At every point on the line, the light density for each component is calculated and processed before being weighted with the frequency spectrum at the current wavelength. Finally, the pixel value is obtained and inserted into an image object. In Figure 1, we depict the three main SEDs $\sigma(\lambda)$ used in the current version of our code: the bulge, a thin disk with flux dominated by a young stellar population (Scd), and thick disk dominated by an older stellar population (Sbc). When all pixels have been calculated, the file is saved to disk as a FITS file. It is also possible to automate the creation of images at different wavelengths in order to produce color images. The image rendering process is fully parallelized with OpenMPI but can also be used on single-processor computers. A schematic representation of the galaxy sphere intersection process is depicted in Figure 2.

2.1. Procedural Noise

Procedural algorithmic methods for producing realistic-looking fractal-like patterns have been around for more than 30 yr. The most famous of these algorithms is Perlin noise, first developed by Ken Perlin in 1982 for use in the movie "Tron." Perlin noise is a type of pseudo-random gradient noise that mimics a Gaussian field but is extremely fast to compute. An improved version of Perlin noise is called Simplex noise, which has several advantages over standard Perlin noise in terms of scalability and speed.

2.1.1. Perlin Noise—An Overview

An overview of the current state of procedural noise functions can be found in Lagae et al. (2010). Perlin noise (Perlin 2002a, 2002b) is a band-limited repeatable pseudo-random function from $\mathbb{R}^n \rightarrow \mathbb{R}$ that is computationally fast and has properties that make it suitable for creating patterns that mimic nature. Perlin noise is an approximation to Gaussian filtered noise implemented as a pseudo-random spline. The algorithm uses

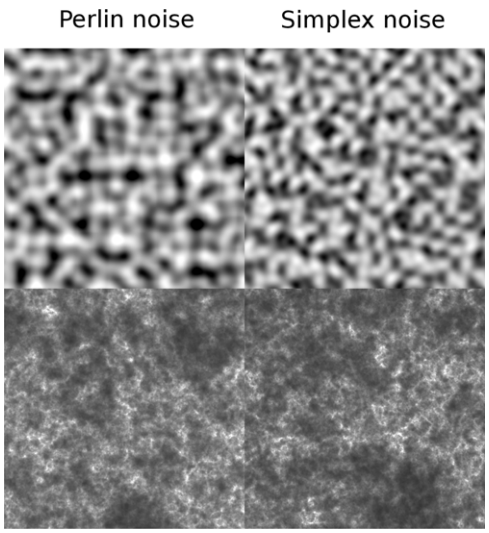

Figure 3. Perlin noise (left) vs. simplex noise (right). Note the straight x-y pattern in the single upper Perlin mode and the less anisotropic simplex mode. Lower part: combining octaves tend to remove these patterns.

a predefined grid of gradients pointing in a random direction, where for any point within a grid, we interpolate the gradients from the four corners. The algorithm is as follows.

1. For a point, obtain the four closest gradients in the grid.
2. For each of the gradients, calculate the dot product between the gradient and a vector defined by the distance between the point and the current grid corner.
3. Instead of linear interpolation $F(t) = t$, use a fade function to produce smoother interpolated values. Perlin noise typically uses the following fade function: $F(t) = t^3(t(6-15)+10)$.
4. Obtain the final value by linearly interpolating between the x-axis, and use this result to linearly interpolate the y-axis.

Simplex noise works in a similar manner but requires fewer computational steps, scales better in higher ($\geqslant$3) dimensions, and has fewer anisotropic properties. Both Perlin and Simplex noise are included in the GAMER package, but any other pseudo-random pseudo-Gaussian field method could easily be implemented by inheriting from the main noise class.

2.1.2. Properties of Procedural Noise

Standard Perlin noise is known for having several anisotropic features (Lagae et al. 2010). Perlin noise also gives rise to a "textile" pattern in the x and y directions, which is evident from the upper left part of Figure 3, depicting 2D Perlin noise in a single octave. The upper right part shows the same node for Simplex noise, with less obvious patterns. However, when combining several octaves, as depicted in the two lower parts, these anisotropic patterns tend to cancel out. In any case, for observations of weak gravitational lensing, these anisotropies would occur on much smaller scales than the typical resolution. In addition, we have calculated the statistical distributions for Perlin and simplex noise with $\sigma = 0.2$ and have plotted the results in Figure 4. Each procedural noise method has distributions that are close to normal but with some intrinsically shifted sigma and mean. Therefore, each time GAMER starts, the distribution of the current procedural noise function is calculated in order to obtain the standard deviation and mean. These values are then used to normalize and offset procedural noise values when creating patterns.

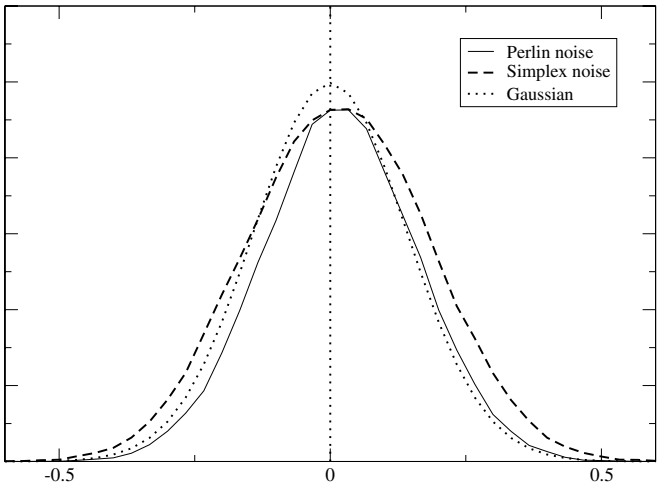

Figure 4. Normalized distributions of Perlin noise and simplex noise compared with a Gaussian distribution. Note how both noise functions produce near-Gaussian but slightly offset distributions.

2.2. Multiple Octaves: Creating Patterns

The different octaves from procedural noise functions can be combined to produce a wide range of nature-like patterns. A generic N-dimensional procedural function $\Phi(\boldsymbol{x}) : \mathbb{R}^N \rightarrow \mathbb{R}$ could be expressed as

$$\Phi(\boldsymbol{x}) = \Theta\left(\sum_k \kappa(k) P(f_s(\boldsymbol{x}, k))\right), \tag{1}$$

where $f_s(\boldsymbol{x}, k)$ describes the frequency scale modifier (e.g., $f_s(\boldsymbol{x}, k) = k\boldsymbol{x}$), $\kappa(k)$ is the octave amplitude (e.g., $\kappa(k) = 1/k$), and $\Theta(x)$ an overall modifier (e.g., $\Theta(x) = x$ or $\Theta(x) = 1/x$). In a sense, $\kappa(k)$ is similar to Fourier coefficients. In its simplest case, summing directly over the octaves with amplitude $1/k$ results in

$$\Phi(\boldsymbol{x}) = \sum_k \frac{1}{k} P(k\boldsymbol{x}), \tag{2}$$

and produces a 2D procedural noise pattern depicted in the lower part of Figure 3. In GAMER, we construct a general procedural noise function

$$\Phi(\boldsymbol{x}, \omega, k_0, f_s, \nu, \alpha) = \left(\sum_{k=1}^{\omega} \Theta((k_0 + f_s k)^{-\nu} (P((k_0 + f_s k)\boldsymbol{x})))\right)^{\alpha}, \tag{3}$$

where ω is the number of octaves, k_0 is the initial frequency, f_s is the frequency scale, ν is the spectral index, α is an exponential factor, and $\Theta(x)$ is a function modifying the end product of the procedural sum, either $\Theta(x) = x$ or $\Theta(x) = 1/x$.

2.3. Procedural Noise in GAMER

The various large-scale components of a galaxy have simple analytical expressions for their intensities as a function of radius r and height above the galaxy plane z. However, while the analytical functions are consistent with the statistical properties of intensity distributions, real-life galaxies exhibit fluctuations in intensities because of overdensitites and underdensities, interstellar dust, pressure waves, and star formation areas. We therefore choose to include a random perturbation to the disk

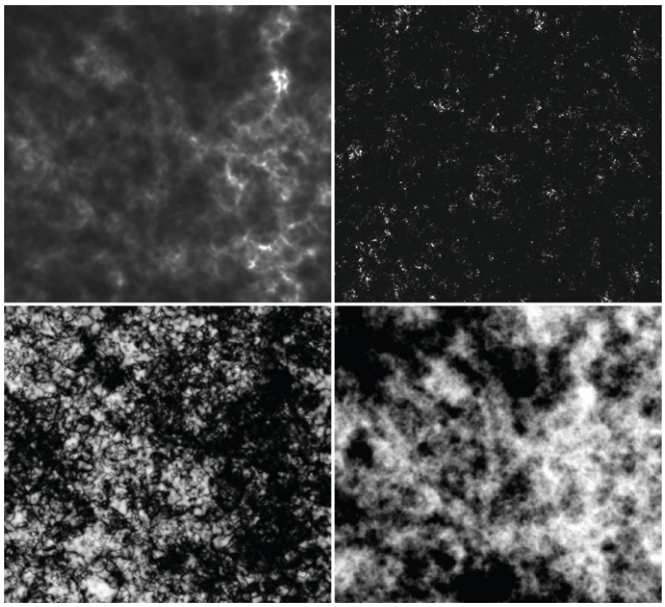

Figure 5. Procedural 2D noise component examples. Upper left: linear sum of procedural noise, showing fluctuations on all scales (Equation (2)). Upper right: combination of procedural noise that mimics clusters of stars (Equation (6)). Lower left: procedural noise that mimics dust clouds. Lower right: procedural noise that depicts disk filament structure (5).

and dust components that mimics these natural patterns. This is implemented through the use of procedural noise.

In Figure 5, we depict various galaxy components used in GAMER. In these four panels, there is no ray-tracing, and only the direct 2D representation of noise patterns are presented. Recall the upper panels of Figure 3, where a single octave is presented with parameters $\omega = f_s = \nu = \alpha = 1$ and $k_0 = 0$:

$$\Phi_{\rm flat}(\boldsymbol{x}) = P(\boldsymbol{x}). \tag{4}$$

In the upper left corner of Figure 5, the raw 2D representation of the disk component is shown, with $k_0 = 0, \omega = 7, \Theta(x) = 1/x$, $f_s = \nu = \alpha = 1.0$, yielding

$$\Phi_{\rm disk} = \left(\sum_{k=1}^{7}\left(\frac{1}{k}(P(k\boldsymbol{x})\right)\right)^{-1}. \tag{5}$$

In the upper right corner of the same figure, we illustrate the raw 2D star patterns with parameters $k_0 = 20, \omega = 7$, $f_s = 3, \nu = 1, \Theta(x) = x$, and $\alpha = 8$:

$$\Phi_{\rm stars} = \left(\sum_{k=1}^{7}\left(\frac{1}{k}(P((20+3k)\boldsymbol{x})\right)\right)^{8}. \tag{6}$$

Similarly, in the lower left corner of Figure 5, dust patterns are created with parameters $k_0 = 0, \omega = 8$, $f_s = 1, \nu = 1, \Theta(x) = 1/x$ and $\alpha = 4$, while the dustlike component of the lower right corner differs only by $\Theta(x) = x$

It is worth noting that some of these parameters (f_s, ν, α) are defined for each galaxy component and are therefore individualized for each galaxy. In addition, ω specifies the detail level of the procedural noise and is defined as a global rendering parameter.

We also wish to stress that our recipes for generating procedural random fields in GAMER are chosen on the basis of visual resemblance with what is observed in real galaxy images. Therefore, the perturbations in the dust and disk components currently superficially resemble dust and star-forming regions, but the similarity is not yet tested with a quantitative measure. However, both the visual resemblance illustrated by the examples in this paper and the versatility of procedural noise functions for generating a broad range of structures are encouraging. In a subsequent paper, we would like to explore the possibility of creating procedural noise models that actually fit with observed data and perform a full likelihood analysis of dust parameters.

2.4. Components

The various galaxy components are treated as individual classes in the C++ code, where each ray-tracing step iterates through the assigned components and modifies the current intensity. If desired, it is straightforward to add new components to the code. In Figure 6, we present an optical image of the grand design spiral galaxy NGC 4321 (Messier 100) together with its simulated counterpart. We also depict each galaxy component as individual images. In this section, we go through each of these components and explain their morphologies.

Each component has a set of user-defined parameters, and in order to limit the number of parameters, the various components might reuse parameters in different ways. For instance, each component contains an intensity multiplier parameter I_m, which is used to scale the overall intensity of the current component. This parameter is used in different ways for various components: in disk and star forming and bulge components, it is used to linearly scale the intensities, while in dust components it is used as the optical depth.

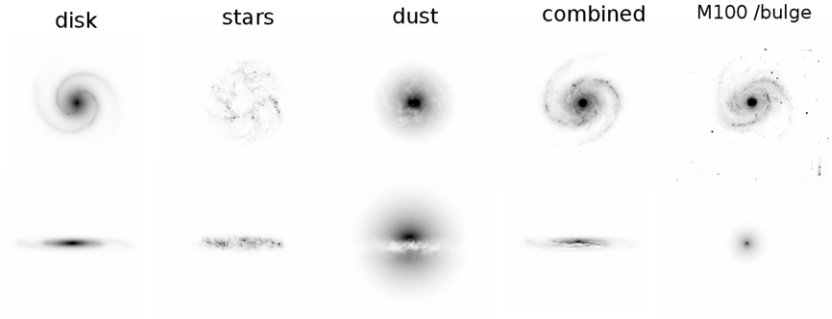

Figure 6. NGC 4321 together with a simulated galaxy, where each component is presented individually and from two different angles. First panel from left: the thin/thick disk that eventually modulates the other components. Second panel: star-forming regions. Third panel: dust component, where we have included a bulge as a light source. Fourth panel: the components combined. Fifth panel, upper: a 1280s *R*-band image of NGC 4321, obtained with the MOSCA camera at the 2.56-m Nordic Optical Telescope. Fifth lower panel: the individual spherically symmetric bulge component.

Table 1
Summary of Parameters in the Code

Type	Name	Values
Rendering	Image size	$\mathbb{N}$
Rendering	λ (wavelength)	$\mathbb{R}$
Rendering	Output	fits, bmp
Rendering	Camera	vec3
Rendering	Target	vec3
Rendering	Up	vec3
Rendering	Ray step length	$\mathbb{R}$
Rendering	ω (procedural noise octaves)	$\mathbb{R}$
Render-list	Position	vec3
Render-list	Orientation	vec3
Render-list	Magnitude	$\mathbb{R}$
Galaxy	Axis	vec3
Galaxy	Number of arms	$\mathbb{Z}$
Galaxy	W_B, W_N	$\mathbb{R}$
Galaxy	Number of components	$\mathbb{Z}$
Component	A_w (arm width)	$\mathbb{R}$
Component	τ (twirl)	$\mathbb{R}$
Component	I_m (Intensity multiplier)	$\mathbb{R}$
Component	Bulge size	$\mathbb{R}$
Component	z_0 (height drop-off)	$\mathbb{R}$
Component	r_h (radial drop-off)	$\mathbb{R}$
Component	δ (azimuthal shift parameter)	$\mathbb{R}$
Component	α (exponential factor)	$\mathbb{R}$
Component	f_s (frequency scale)	$\mathbb{R}$
Component	ν (spectral index)	$\mathbb{R}$

Note. vec3 corresponds to the 3D vector class CVector.

For each ray step in every pixel, all non-bulge galaxy components calculate a combined intermediate intensity value defined as

$$\Psi(\mathbf{p}) = A(A_w, W_N, W_B)D(z_0, r_h)\Phi(f_s, \nu, \alpha, \tau, \delta)\sigma(\lambda), \quad (7)$$

where the parameters $\mathbf{p}$ are

$$\mathbf{p} = \{A_w, W_N, W_B, z_0, r_h, f_s, \nu, \alpha, \tau, \delta, \lambda\}. \quad (8)$$

Here the first expression A is the arm modifier with parameters A_w, W_N, and W_B, defined in Section 2.4.1. The second expression D is the disk intensity modifier, containing the exponential disk drop-off r_h and disk height drop-off z_0, defined in Section 2.4.3. The third term Φ is the procedural noise function from Equation (3), using the current component parameters. While f_s, ν and α are described in Section 2.2, the vortex twirl parameter τ is explained in Section 2.4.4. The δ is the arm shift parameter, explained in Section 2.6, and $\sigma(\lambda)$ is the component-assigned SED value for the current wavelength λ. Note that for dust components, σ is strictly speaking not an SED but rather a dust extinction law.

A list of all parameters can be found in Table 1. In this section, we break down Ψ to its constituencies and explain its usage in the various components.

2.4.1. Spiral Arm Structure

We base our spiral arm component on the work of Ringermacher & Mead (2009), where the authors provide a simple continuous function of two free winding parameters W_B and W_N

$$r(\theta, W_N, W_B) = \frac{1}{\log\left(W_B \tan\left(\frac{\theta}{2W_N}\right)\right)}, \quad (9)$$

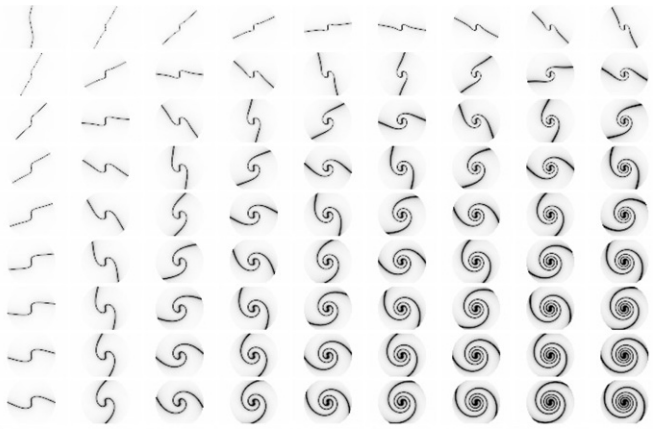

Figure 7. Spiral arm structure for various values of the winding numbers $W_B \in [0.05, 1.0]$ and $W_N \in [1, 10]$ given from Equation (9).

which has been shown to tightly fit spiral galaxies of *Hubble* types ranging from grand design spirals to late-type large barred galaxies. In our code, we invert this function and obtain the angle θ as the function of r:

$$\theta(r, W_B, W_N) = \tan^{-1}\left(\frac{e^{1/r}}{W_B}\right)2W_N. \quad (10)$$

A set of arm structures for various values of winding numbers W_B and W_N can be seen in Figure 7. We implement this function in order to perform two tasks: to modulate the intensity of the disk and thus yield arm structure and to perturb the tangential direction of the procedural noise components (dust and disk) to create the illusion of a vortex twirl. In addition, each galaxy disk component needs to provide a parameter A_w for the width of the arm, ranging from zero (disk only) to a positive value (for instance 10, yielding a narrow arm). The arm modifier A is then defined as

$$A(A_w, W_B, W_N) = \left(1 - \frac{1}{\pi}|\theta(r, W_B, W_N) - \theta'|\right)^{A_w}, \quad (11)$$

where θ is the radial center of the closest arm and θ' the current radial position. Figure 8 shows a spiral galaxy with $W_B = 0.5$ and $W_N = 3$ with $A_w = 0.2$ and $A_w = 1.2$ for the case of wide and thin arm, respectively. The thick and thin disk are set up using the Sbc and Scd SEDs, respectively, of Figure 1.

2.4.2. Bulge Component

The bulge component, being the most straightforward component as it is not modified by procedural noise, is calculated with intensity proportional to the density ρ at radius r given by the approximation of a deprojected de Vaucouleurs profile derived by Mellier & Mathez (1987):

$$I(r) \propto \rho(r) = I_m r_b^{-0.855} \exp\left(-r_b^{1/4}\right). \quad (12)$$

Note that the bulge is treated differently than the rest of the components and is therefore not described by Equation (7). The bulge has three parameters: the bulge intensity (defined as the intensity multiplier parameter) at the center I_m, the bulge size given by a scale r_0 and defined as $r_b = r/r_0$, and a bulge axis (for elliptical galaxies). If the bulge component is the only galaxy component, the galaxy will be elliptical with axes defined by the bulge axis. A galaxy rendered from the side with a thick dust

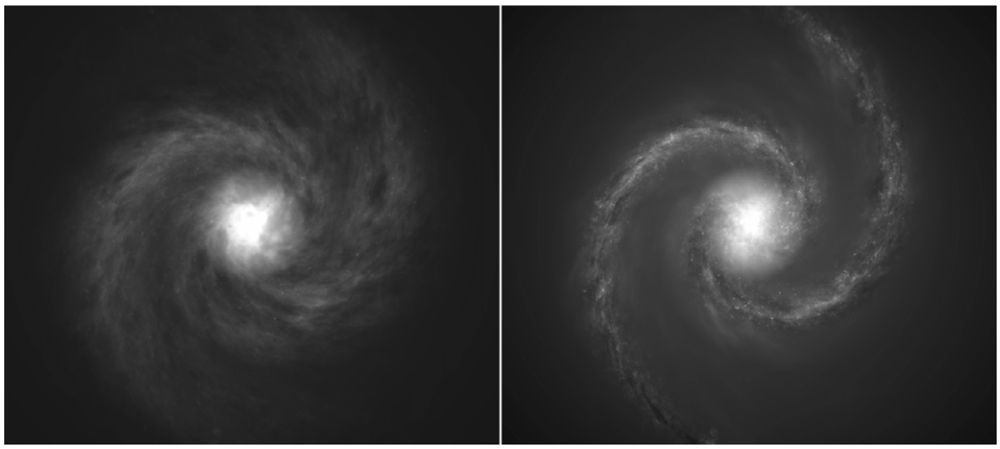

Figure 8. Example of the arm width modifier A_w parameter. The left image displays a spiral arm galaxy with $A_w = 2$, yielding a visible centralized arm structure. The right image depicts the same galaxy but with $A_w = 0.5$, and the arm structure is less visible. Each galaxy component is provided with an individual parameter A_w.

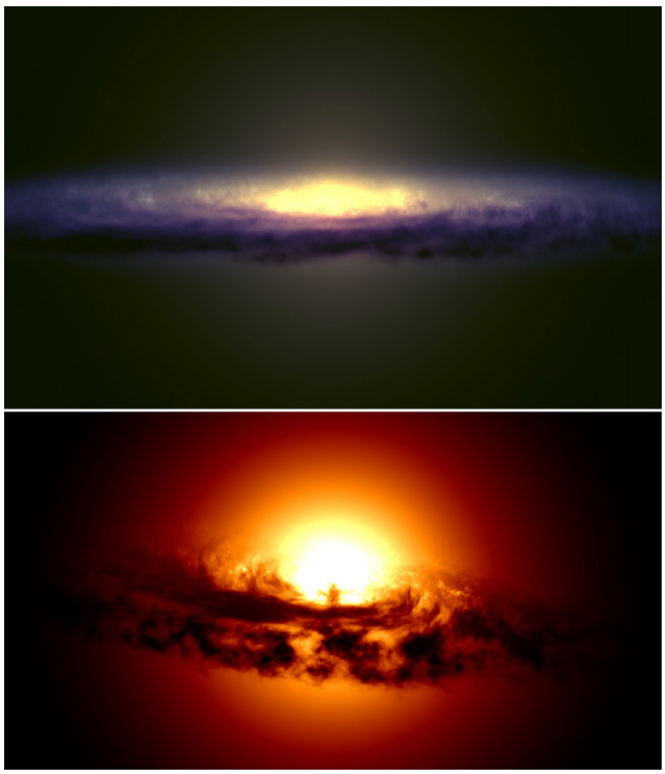

Figure 9. Upper: an armless galaxy with a noticeable dust band, corresponding to a medium-sized disk component together with a thick dust component. The galaxy is rendered almost from the side, where the individual RGB images are ray-traced at wavelengths 4000, 5500, and 7500 Å. Lower: a simulation of an elliptical galaxy with a thick dust band, rendered in the *I*-band (7500 Å).

(A color version of this figure is available in the online journal.)

layer is depicted in Figure 9, where the bulge is visible from both above and below the disk. The image uses the three SEDs from Figure 1. The lower image shows an elliptical galaxy with a thick dust band, with the bulge SED from Figure 1.

2.4.3. Disk and Dust Component

The disk, dust, and star components follow intensity profiles given by the disk intensity modifier (Mo et al. 2010)

$$D(z_0, r_h) = \mathrm{sech}^2\left(\frac{z}{z_0}\right) e^{-r/r_h}, \tag{13}$$

where z is the current height above the galaxy plane, r is the current radial position in the galaxy plane, z_0 is a parameter defining the thickness of the plane, and r_h is another component parameter defining the exponential disk drop-off. For intensity-producing components, intensities are added linearly:

$$I' = I + I_m \Psi. \tag{14}$$

However, dust differs from the remaining components as they decrease intensities through a dust extinction law. When calculating intensities in the sampling scheme, the extinction law is given by

$$I' = I e^{-I_m \Psi}. \tag{15}$$

Note here that the intensity multiplier I_m acts as a linear multiplier when increasing intensities but as an optical depth parameter when concerning dust components. We base our dust extinction law on Cardelli et al. (1989), where we generate a dust-extinction data file with $R_v = 3.1$, a standard value that is used for interstellar medium in the Milky Way. This file is treated in the code just as a regular SED, and as it can be selected as a component parameter, it is automatically included from Equation (7). In the next version of GAMER, the user will be able to generate a wide range of SEDs with user-defined parameters, including R_v.

In addition, the amplitude of the star formation component is proportional to the amplitude of the procedural pattern in the disk component, enabling the effect of star-forming regions tracing the spiral arm pattern (defined by Equation (7)). This effect is clearly visible in Figure 10, where we show a close-up image of a spiral galaxy with dust, stars, bulge, and disk components. In addition, in Figure 11, we display a color image of a spiral galaxy with thick dust bands that absorb mostly blue light. In the leftmost panel of Figure 6, we display the disk component of the simulated NGC 4321 from two angles. In addition, the dust component is shown in the center panel, where we have added a bulge in order to visualize the dust. The star-forming component is presented in the second panel. Finally, the fourth panel depicts the combined image of the NGC 4321 simulation, and the upper-right corner is an optical *R*-band image of NGC 4321 obtained with the Nordic Optical Telescope (NOT). The purpose of this plot is to illustrate qualitative visual similarity, and a future paper will employ algorithms to optimize and quantitatively test the goodness of fit.

Figure 10. Combined RGB image of a spiral galaxy, showing the dust, disk, Bulge, and star components. Note how the twirling of the dust component gives rise to the effect of a central vortex. The image is ray-traced at 4000, 5500, and 7500 Å.

(A color version of this figure is available in the online journal.)

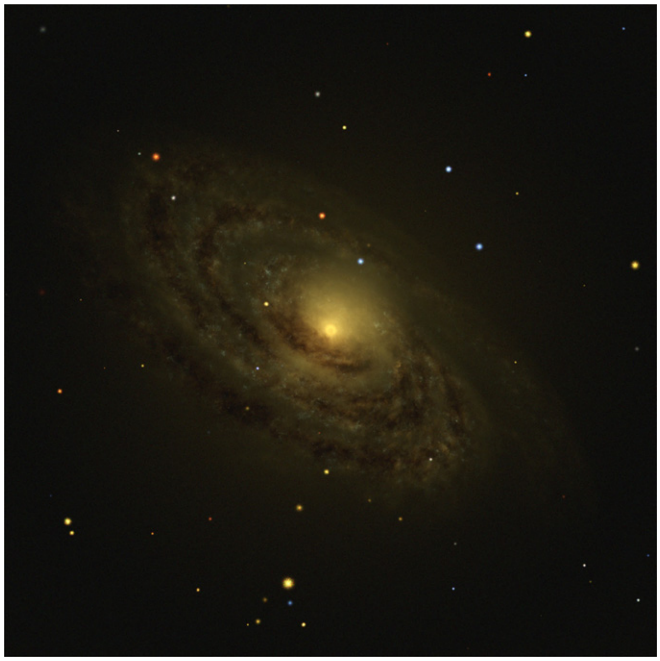

Figure 11. Galaxy with heavy dust components. The image is ray-traced at 4000, 5500, and 7500 Å.

(A color version of this figure is available in the online journal.)

2.4.4. Twirl

In order to produce more realistic-looking patterns, we include a feature that introduces the illusion of a vortex twirl in the component ray-tracer. The procedural noise for each component is tangentially perturbed by the spiral arm expression given by Equation (9) and is scaled with the twirl parameter τ. Therefore, if τ is zero, then there is no tangential perturbation, while a positive τ provides a tangential stretching following the spiral arm structure. In Figure 12, we present two simulated spiral galaxies viewed face-on and edge-on: the right image with $\tau = 0$ in all components and the left with $\tau > 0$. The twirl is also evident in Figure 10, where the star component (small-scale fluctuations) has $\tau = 0.1$, while the dust and disk component has $\tau = 0.5$.

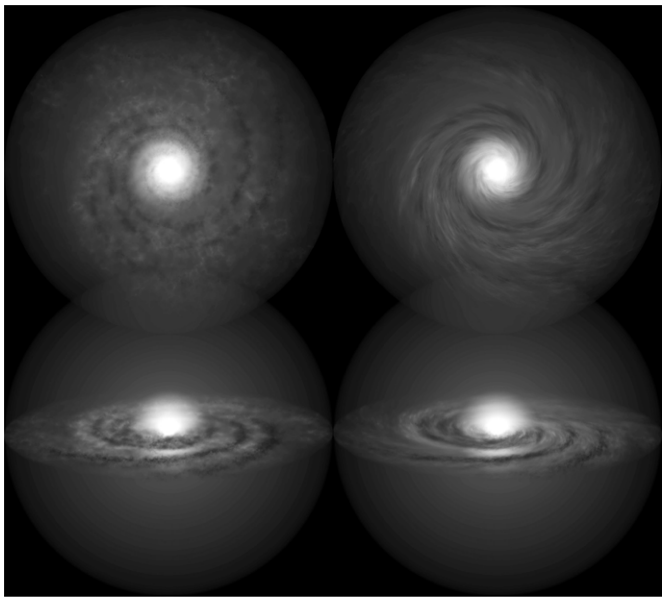

Figure 12. Image depicting the effect of the twirl component parameter (τ). Right panels: a spiral galaxy with $\tau = 0$ in all components, yielding "flat" procedural noise fluctuations that do not follow the spiral structure. Left panels: the same galaxy but with $\tau > 0$. The image is ray-traced at 4000 Å, using SEDs from Figure 1.

2.5. Irregular Galaxies

About a quarter of all galaxies are irregular galaxies, and it is therefore important to be able to account for such galaxy types. We have not yet provided a method to created complete irregular-type galaxies with no visible arm structure, but this will be implemented in the next version of GAMER. However, an easy way to create irregularities in spiral galaxies is to perturb the arm placement. For instance, if a simulated galaxy has two arms, the second arm would regularly be placed with a π angular separation from the first. However, this number can be modified to any value, with examples shown in Figure 13. Here we present a symmetric regular galaxy in the upper left corner, while the upper right corner shows the same galaxy with its arm moved 1.0 radians with respect to π. The lower-left corner displays the galaxy with three arms, while the lower-right image has winding number close to zero, yielding a more irregular morphology.

2.6. Azimuthal Shift

Density wave theory predicts that spiral structures in galaxies are quasistatic, where the arms are regions with 10%–20% greater mass densities than in the rest of the galactic plane. Dust and gas typically have angular rotation rates larger than the pattern speed of the spiral arms. Hence, in a galaxy with a trailing spiral, they will approach a spiral arm from the concave side of the arm and become compressed there. This is in accordance with what we typically observe in spiral galaxies, where the most prominent dust lanes are found on the concave side. The azimuthal offset between the dust lanes and the bright young stars outlining the spiral arms correspond to the movement of the dust and gas during the $\sim$10 Myr it takes for stars to be born

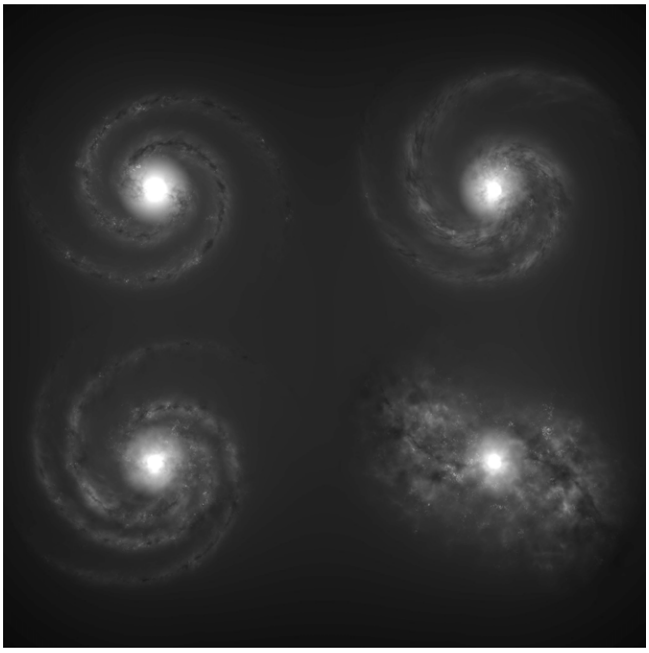

Figure 13. Example of irregularity in spiral galaxy production. The upper left corner shows a symmetric, regular galaxy, while the upper right corner shows the same galaxy with one of the arm positions perturbed. The lower left corner depicts the same galaxy with three arms, while the lower right has symmetric arms but near-zero winding number.

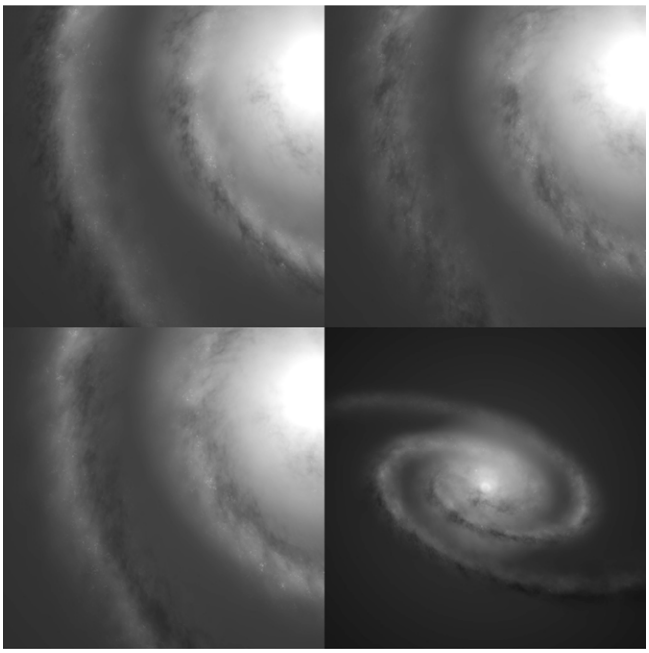

Figure 14. Example of dust component azimuthal shift δ in a spiral galaxy. In the upper left corner, the dust component is shifted by $\delta = -0.3$ radians, in the upper right corner it is shifted by $\delta = 0$ radians, in the lower left corner it is shifted by $\delta = 0.3$ radians, and in the lower right corner the full galaxy with $\delta = -0.3$ radians is displayed.

from the compressed gas. We therefore include an azimuthal shift parameter δ that effectively rotates the current galaxy component by δ radians with respect to the nominal arm. This parameter is depicted in Figure 14.

2.7. Sampling Scheme

The current sampling scheme is linear: galaxy intensities for each component are computed at a linear interval on the ray

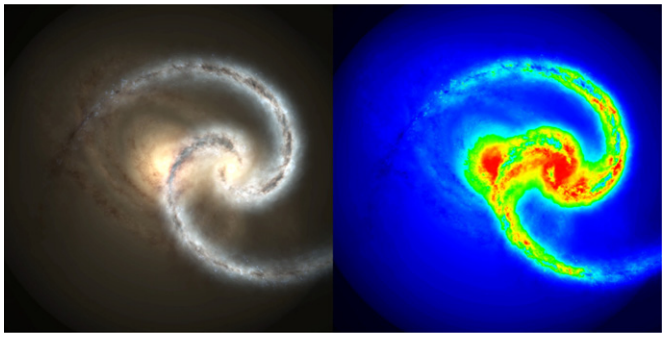

Figure 15. Dust absorption in a foreground spiral galaxy. The right panel displays the same image with artificial colors that enhance the foreground dust absorption.

(A color version of this figure is available in the online journal.)

transversing the galaxy sphere. The most costly step in the sampling scheme is calculating the procedural noise functions, but the code is optimized to exclude calculating Φ for values that are outside a threshold of the disk intensity $D(z_0, r_h)$ from Equation (13). The step length is defined in the global rendering parameters file and can be manipulated through the GAMER Java interface. The actual sampling scheme is defined in a C++ class and can be easily replaced by a more advanced scheme, for example, with varying step lengths or adaptive refinement.

2.8. Rendering Multiple Galaxies

The code currently supports rendering of multiple galaxies, where the renderer loads a render list of galaxies before processing the scene. This render list typically contains the position of the galaxies and their orientations, magnitudes, and morphological types. In Figure 15, we have placed a spiral galaxy in the background together with a smaller barred galaxy in the front. The foreground galaxy has a sizable dust component, and the extinction law of the dust ensures that mostly blue wavelengths are absorbed. Note how the dust patterns become visible when the arms of the foreground galaxy overlap with the background galaxy. In the next version of GAMER, we plan on allowing for galaxies to intersect, along with bending of light rays caused by gravitational lensing.

GAMER should be able to produce simulated high-resolution deep-field images that, for example, mimic existing *HST* or future *JWST* and Euclid images. In Figure 16, we depict such a simulated deep-field image with ∼2000 galaxies. The 1600 × 1600 image took about 8 minutes to render on a MacBook Pro with four cores. However, the galaxies are uniformly distributed, and both the redshift and galaxy morphologies are random. Rendering a correct distribution of galaxy morphologies for a given luminosity will be implemented in the next version of GAMER.

2.9. Code Parameters

The input parameters found in Table 1 can be divided into four parts.

1. Rendering parameters: image size, current wavelength, output type, and file names.
2. Rendering list: galaxy type, 3D position, orientation, size, and absolute magnitude.
3. Galaxy parameters: bulge/disk ratio, axis, number of arms, and number of components.

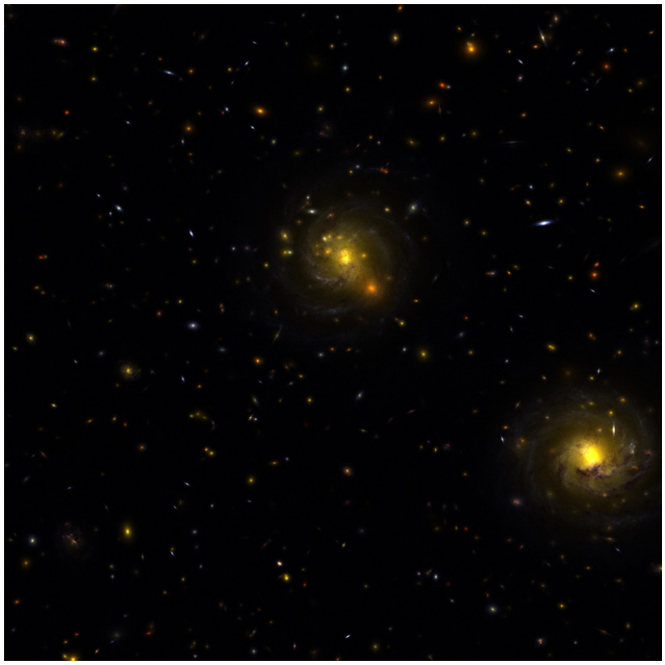

Figure 16. Example of a deep-field image, with ∼2000 galaxies. The galaxies are uniformly distributed, but the galaxy morphology population is random. This represents the next step for GAMER to be able to simulated deep-field images with statistically correct galaxy populations.

(A color version of this figure is available in the online journal.)

4. Galaxy component parameters: arm width modifier, vortex twirl, relative component intensities, bulge size, exponential disk falloff, and disk thickness

The main renderer requires two input parameters at the command line, which correspond to the rendering parameters and the render list. The rendering parameters contain information about the camera and image properties, while the render list contains the individual scene. The galaxy types are represented as individual text files, where the galaxy properties together with the individual component parameters are described.

The code is fully parallelized and uses OpenMPI to distribute the ray-tracing scheme to the individual processors. When all processors have completed their task, the image data is returned to the main processor and the image is assembled.

3. GAMER INTERFACE

GAMER is run as binary command-line files operating on text files. There are two binary files: the GAMER ray-tracer and the GAMER utility tool (gutil). The GAMER utility tool lets the user in command line perform operations on files, such as adding noise, convolving, normalizing, rotating, adding, and gamma-correcting fits files. For ease of use, we have provided a Java GUI that lets the user set up global rendering parameters, define render scenes, and create and modify galaxies including their individual components. The interface contains a preview image of the current scene and also implements post-processing tools such as convolution, noise, and gamma correcting. Figure 17 displays a screenshot of the GAMER GUI software.

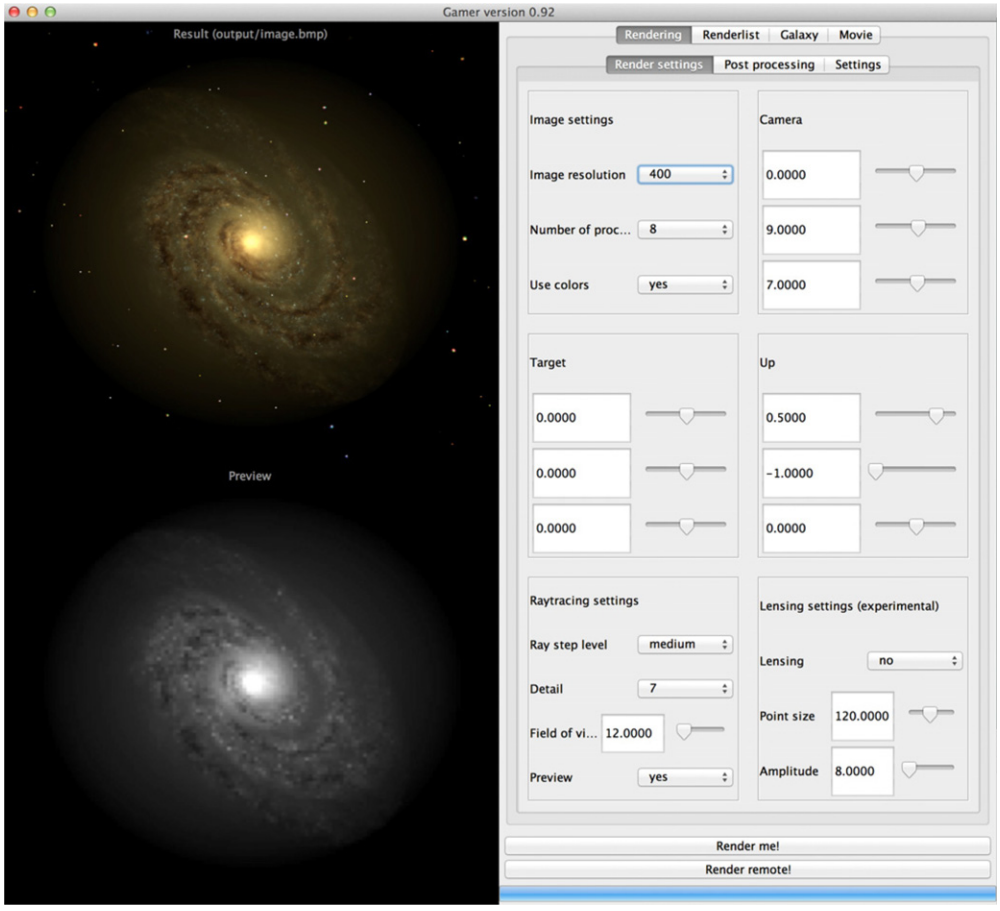

Figure 17. Screenshot of the GAMER GUI package, depicting the main rendering settings.

(A color version of this figure is available in the online journal.)

4. CONCLUSIONS AND FUTURE WORK

We have developed a new framework using procedural random fields to ray-trace artificial images of galaxies with statistically realistic component morphologies. The galaxies consist of several components defined by component parameters, where some can be randomized in order to produce variations in the galaxy population. Each component is coupled to an SED, making it possible to render images at different wavelengths. This naturally produces color gradients in the simulated galaxies, an improvement over most earlier works that have assumed a fixed SED across the angular extent of a simulated galaxy. The code sets up a full 3D viewport, allowing for ray-tracing multiple galaxies in one pass.

The main goal of GAMER is to produce realistic-looking optical deep-field survey images that can be used in gravitational lensing analyses. With the next version of GAMER, we hope to implement the following features.

1. Selecting simulated galaxies from a distribution $N(L, z, type)$ that provides a correct mix of galaxy morphology for a given redshift and luminosity.
2. Allowing for evolution of galaxy sizes, for example, by making the disk radius of galaxies dependent on redshift $r_h = r_h(z)$.
3. A Monte-Carlo Markov Chain (MCMC) or genetic algorithm for best-fitting galaxy parameters with empirical data. See Perret et al. (2011) for a previous application of an MCMC algorithm to fit barred spiral galaxies. This will enable us to study the correlations between different galaxy parameters and their evolution.
4. Support for deflection of light passing through a 3D matter distribution containing galaxies, producing images of background galaxies distorted by weak or strong gravitational lensing effects and the lenses responsible for producing these effects.
5. Using output from cosmological *N*-body simulations to select galaxy positions and luminosities on the basis of simulated locations and halo masses. Furthermore, the merger history of each simulated galaxy halo can be linked to the star formation history of each galaxy (e.g., Cox et al. 2008). We may include such effects for example, by adopting synthetic SEDs from stellar population synthesis models that include the effects of the star bursts caused by the mergers. The merger history may also be linked to the morphological properties, such as boxy or disky isophotes and isophote twists (e.g., Naab & Burkert 2003).
6. A stringent test of the accuracy of the procedural noise patterns in the dust components by comparing statistical properties from our model with detailed observations of Milky Way dust (Miville-Deschênes et al. 2007).

The authors acknowledge financial support from the Research Council of Norway and thank Claudio Llinares, Hans A. Winther, and Theodor Groeneboom for useful discussions. We also thank the anonymous referee for suggesting several useful improvements. Based on observations made with the Nordic Optical Telescope, operated by the Nordic Optical Telescope Scientific Association at the Observatorio del Roque de los Muchachos, La Palma, Spain, of the Instituto de Astrofisica de Canarias.

REFERENCES

Bouwens, R. J., Illingworth, G. D., & Magee, D. K. 2006, in ASP Conf. Ser. 351, Astronomical Data Analysis Software and Systems XV, ed. C. Gabriel, C. Arviset, D. Ponz, & E. Solano (San Francisco, CA: ASP), 145
Bridle, S., Balan, S. T., Bethge, M., et al. 2010, MNRAS, 405, 2044
Bridle, S., Shawe-Taylor, J., Amara, A., et al. 2009, AnApS, 3, 6
Bruzual, G., & Charlot, S. 2003, MNRAS, 344, 1000
Cardelli, J. A., Clayton, G. C., & Mathis, J. S. 1989, ApJ, 345, 245
Coleman, G. D., Wu, C.-C., & Weedman, D. W. 1980, ApJS, 43, 393
Cox, T. J., Jonsson, P., Somerville, R. S., Primack, J. R., & Dekel, A. 2008, MNRAS, 384, 386
de Vaucouleurs, G. 1948, AnAp, 11, 247
Heymans, C., Van Waerbeke, L., Bacon, D., et al. 2006, MNRAS, 368, 1323
Kaiser, N. 2000, ApJ, 537, 555
Kitching, T., Balan, S., Bernstein, G., et al. 2011, AnApS, 5, 2231
Lagae, A., Lefebvre, S., Cook, R., et al. 2010, in Proc. of Eurographics Association, EG 2010—State of the Art Reports, ed. H. Hauser & E. Reinhard (Geneva: Eurographics Association)
Mandelbaum, R., Hirata, C. M., Leauthaud, A., Massey, R. J., & Rhodes, J. 2012, MNRAS, 420, 1518
Mandelbaum, R., Rowe, B., Bosch, J., et al. 2013, arXiv:1308.4982
Massey, R., Heymans, C., Bergé, J., et al. 2007, MNRAS, 376, 13
Massey, R., Refregier, A., Conselice, C. J., David, J., & Bacon, J. 2004, MNRAS, 348, 214
Mellier, Y., & Mathez, G. 1987, A&A, 175, 1
Meneghetti, M., Bartelmann, M., Dahle, H., & Limousin, M. 2013, SSRv, 177, 31
Meneghetti, M., Melchior, P., Grazian, A., et al. 2008, A&A, 482, 403
Miville-Deschênes, M.-A., Lagache, G., Boulanger, F., & Puget, J.-L. 2007, A&A, 469, 595
Mo, H., van den Bosch, F. C., & White, S. 2010, Galaxy Formation and Evolution (Cambridge: Cambridge Univ. Press)
Naab, T., & Burkert, A. 2003, ApJ, 597, 893
Perlin, K. 2002a, ACM Trans. Graph, 21, 3
Perlin, K. 2002b, ACM Trans. Graph, SIGGRAPH 2002, 681
Perret, B., Mazet, V., Collet, C., & Slezaket, E. 2011, Pattern Recognition, 44, 1328
Refregier, A. 2003, MNRAS, 338, 35
Ringermacher, H. I., & Mead, L. R. 2009, MNRAS, 397, 164
Scoville, N., Aussel, H., Brusa, M., et al. 2007, ApJS, 172, 1
Semboloni, E., Hoekstra, H., Huang, Z., et al. 2013, MNRAS, 432, 2385
Voigt, L. M., Bridle, S. L., Amara, A., et al. 2012, MNRAS, 421, 1385