A Scatter Point Tool for Object Placements using Poisson-Disk Sampling

Per Blåwiik, Fredrik Norrstig
Department of Science and Technology
Linköping University
2020–12–17

CONTENTS

LIST OF FIGURES

A Scatter Point Tool for Object Placements using Poisson-Disk Sampling

***Abstract*—To create a compelling scene in a large 3D environment, hundreds of different but similar models are often needed. In this report, a scattering tool aimed to make the process of populating a scene with multiple objects easier is presented. The scatter point distribution is based on Poisson-disk sampling and random deviations in terms of rotation and scaling. Hierarchical dart throwing was used for a fast generation of Poisson-disk sample points, suitable for real time applications. The scatter tool manage to create natural looking patterns using only random based strategies.**

I. Introduction

The task of placing out lots of similar models in a 3D environment is a tedious and time consuming task, and by doing it by hand it can be hard to create natural patterns. The aim for this project was to develop a tool to solve these problems. Inspiration for this work comes from a similar project presented in the paper *The Jungle Book: Art-Directing Procedural Scatters in Rich Environments* by Cieri et al. [1], where a geometry distribution tool for artists was developed. This project is a much smaller version of the tool presented by Cieri et al. [1], with focus on controllable scatter distribution on arbitrary surfaces with automatized size and orientation deviations. For generating the scatter points, two different sampling methods were used: Poisson-disk distribution and random distribution. Specifically, the hierarchical dart throwing algorithm proposed by White et al. [2], was used to generate the Poisson-disk distribution efficiently.

The scatter points are placed on surfaces using ray casting from a sample plane to utilize 2D sampling methods as well as allowing more control of shaping the scatter fields.

In this report, the theory and motivation behind the implementation of the tool is presented and discussed.

II. Method

In this section, the full method used in the scatter point tool is presented in a chronological order, and lastly the art-directing tool parameters are explained. An overview of the scatter point generation process is illustrated in Figure 1.

The idea of the method is to select one or multiple objects or surfaces, set scatter parameters, and generate points on the selected surfaces.

The process of generating sample points on a surface can be solved in different ways: for example by creating samples directly on a procedural texture of the selected surface, generating scatter distributions based on the triangle mesh geometry, or sample points on a 2D plane and use ray casting to place them on the surface intersection points.

For this project, the ray casting from a plane method was chosen so that 2D sampling methods could be utilized for generating the scatter distribution. A positive side effect of this method is that it offers more intuitive control over the scatter field. One example of this is shown in Section III.

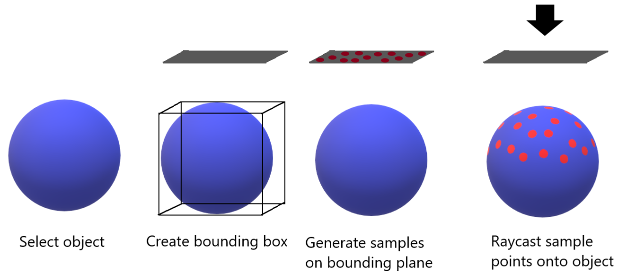

Select object
Create bounding box
Generate samples on bounding plane
Raycast sample points onto object

Fig. 1. An illustration of the scatter point generation process from selecting an object to the final scatter points attached on it's surface.

A. The Bounding Box

When selecting one or multiple surfaces, an *axis aligned bounding box* (AABB) that covers the total shape is calculated. The top surface of the bounding box is then used as a sampling plane to generate the scatter points. This bounding plane is used to guarantee that the sample points completely covers the selected surfaces.

An AABB is defined by two points, one in the minimum corner $P_{min} = (x_{min}, y_{min}, z_{min})$, and one in the maximum corner $P_{max} = (x_{max}, y_{max}, z_{max})$. The minimum and maximum coordinates are given by the minimum and maximum of the selected surfaces.

B. Scatter Point Sampling

The major part of the scatter point tool is to generate sample points. As described in Section II-A, the sample domain is defined by the top plane of the bounding box to make sure the full surface is covered with points.

For this project, two sampling methods were used: Poisson-disk distribution (blue noise), and random distribution (white noise-like). The random distribution simply generates random points inside a sample domain based on resolution and probability distribution. Poisson-disk sampling is useful for evenly distributing points with a controllable point density. Direct comparison between the two approaches is presented in Section III.

1) Poisson-Disk Sampling: A set of Poisson-disk sample points, also referred to as blue noise, is a set of randomly and uniformly distributed points with a minimal distance between them [3]. Poisson-disk distribution patterns are useful in a wide range of computer graphics applications including surface remeshing [4], caustic rendering using photon mapping [5], and plant growth simulation [6]. Figure 2 illustrates the minimum disk radius distance in a Poisson disk set.

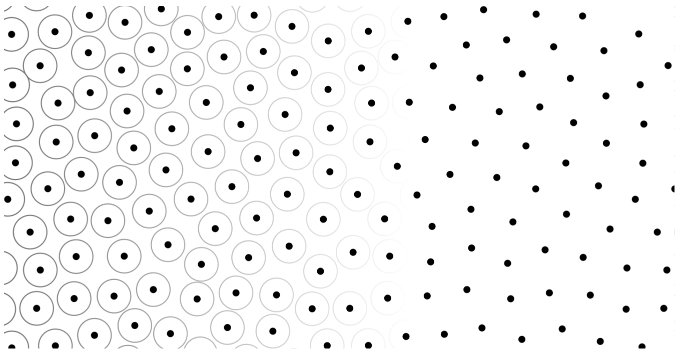


Fig. 2. Points generated with a Poisson-disk sampling method using the minimum distance r between the points. The illustrated circle radii of the points are $r/2$. (Source: wikipedia.org/wiki/File:Poisson_disk_sampling.svg)

2) Hierarchical Dart Throwing: One of the most straightforward approaches for generating Poisson-disk distribution is to use the *dart throwing* algorithm. Dart throwing refers to randomly create a point in the sample domain, verify that the point does not violate the minimum distance requirement, and repeat. This is a computationally slow method [7], especially when generating a maximal point set (a distribution without gaps).

The naive dart throwing method was not suitable for this project since the sampling should be done through an interactive tool, using arbitrary sample domains and disk radii. Instead, the *hierarchical dart throwing* (HDT) algorithm, proposed by White et al. [2], was used. Sample points generated with HDT is considered to be equivalent to naive dart throwing [2] but is a lot faster. The method utilizes quadtree subdivisions of the sample domain as well as an acceleration grid for lookup, to ensure a performance of $O(n)$ in both time and memory on average.

A critical part of the algorithm is the first subdivision of the sampling domain into a list of *active squares*, where each active square is not known to be covered by point radii. The basic idea is to sample points only in active squares and deactivate these squares as much as possible to reduce the sampling area, and thus, reduce rejections and distance lookup.

The length of each base level square b_0 should be as large as possible, to faster reduce the available sampling area, but still smaller than the minimum distance r, to make sure that only one point fits inside a square. White et al. [2] propose setting the base length to

$$b_0 = \frac{w}{\left\lceil w\frac{\sqrt{2}}{r}\right\rceil}, \tag{1}$$

where w is the width of the sampling domain and r is the minimum distance radius.

For each iteration of the HDT algorithm, one of the active squares is randomly selected based on a probability proportional to the active area [2]. If the selected square is not covered by any point radii, a dart is thrown in it and a minimum distance test is performed. If the minimum distance requirement fails, the active square is subdivided into four squares with half the length. All of the four squares that are not covered by point radii are added to an active list based on the subdivision depth. An example of six concurrent steps in the algorithm is illustrated in Figure 3.

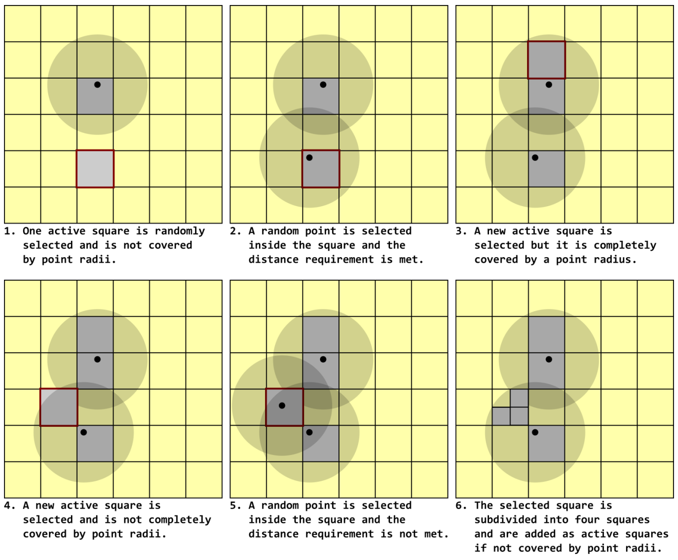

1. One active square is randomly selected and is not covered by point radii.
2. A random point is selected inside the square and the distance requirement is met.
3. A new active square is selected but it is completely covered by a point radius.
4. A new active square is selected and is not completely covered by point radii.
5. A random point is selected inside the square and the distance requirement is not met.
6. The selected square is subdivided into four squares and are added as active squares if not covered by point radii.

Fig. 3. Six concurrent steps in the hierarchical dart throwing algorithm illustrating possible scenarios and the decision making. The yellow squares represent active squares available for selection and sampling. Each loop iteration reduces the active square area, and subsequently, the available sampling area is reduced over time.

Since the minimum distance check is the most frequent operation in the algorithm, it is also the main bottle-neck for computation time. To speed up the lookup process, an acceleration grid system was used for storing all accepted sample points. The idea is to concentrate the minimum distance check in a small neighbour of points instead of checking the entire sampling domain. The details of the acceleration grid is explained in [2].

Another frequent operation is to determine if a selected square is covered by point radii. Instead of measuring the distance from all four corners of the square to a point, White et al. [2] proposes a way of measuring only from the farthest corner. The distance d_{fc} from the farthest corner to a point can be derived by Equation 2,

$$d_{fc}^2 = (|x_c - x_p| + \frac{b}{2})^2 + (|y_c - y_p| + \frac{b}{2})^2 \tag{2}$$

where $P = (x_p, y_p)$ is the point location, $C = (x_c, y_c)$ is the center point of the square, and b is the length of the square.

C. Ray Casting

The method for attaching the scatter points on the selected surfaces is based on simple ray casting. A point on a ray can be defined by Equation 3

$$P(t) = x_o + \vec{d} \cdot t\Delta_t, \tag{3}$$

where P is a point on the ray, x_0 is the ray origin point, $\vec{d}$ is the ray direction, t is the time step, and Δ_t is the step size.

All sample points in the bounding plane, described in Section II-B, are used as ray origins x_0 with the directions $\vec{d} = (0, -1, 0)$, and then the final scatter points are defined by the first intersection points of the surface(s).

D. Orientation Adjustment

When placing a model on a surface it might be desired to use the same orientation as the surface normal. This can be done by applying a series of rotations on the model so that its local coordinate system aligns with the normal of the surface. To get the rotations needed it is easier to think of the problem in reversed, by getting the surface normal vector to align with the models up direction, in this method the models y-axis was used. An illustration of how the rotation angles can be derived is shown in Figure 5. A detailed explanation and implementation of this method is given by M. Kesson [8].

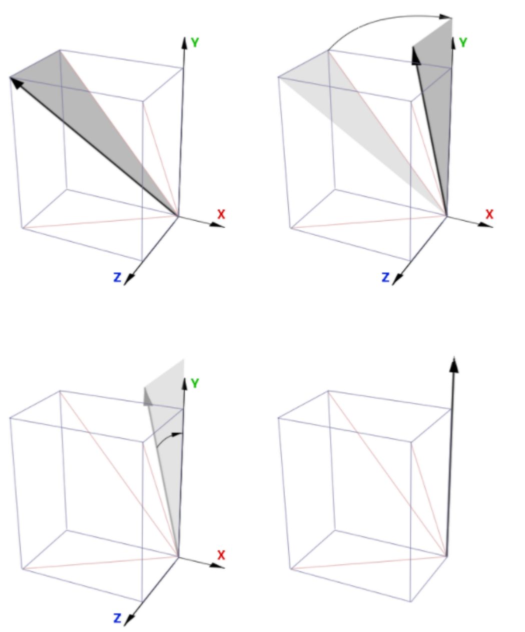


Fig. 4. The rotation around the x- and z-axis to align the vector with the y-axis. (Source:http://www.fundza.com/mel/axis_to_vector/index.html)

How the math works to accomplish these rotation can be found in equation 4-7 with an illustration of the variable names used in Figure 4. In this implementation, vecLength is assumed to be normalized so the magnitude will always be 1.

After calculating the rotations needed, they are applied in the reversed order to the scatter point, first θ_x followed by θ_z, to align them with the surface normal. This process is done for every scatter point.

$$xyLength = \sqrt{x^2 + y^2} \tag{4}$$

$$\theta_z = acos(\frac{y}{xyLength}) \tag{5}$$

$$vecLength = \sqrt{x^2 + y^2 + z^2} \tag{6}$$

$$\theta_x = acos(\frac{xyLength}{vecLength}) \tag{7}$$

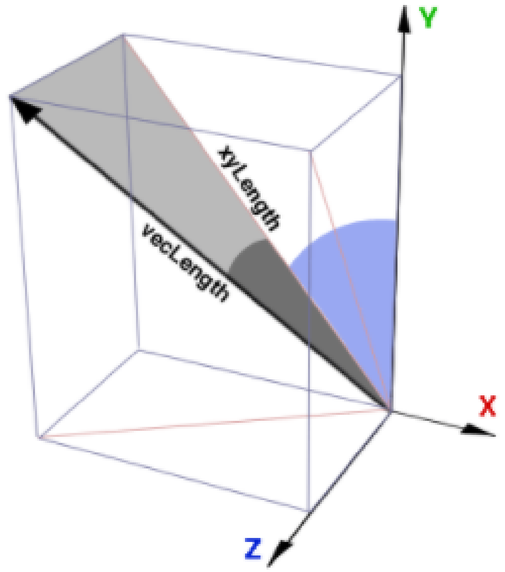


Fig. 5. An image with the two principal angles, the rotation angle around the z-axis θ_z (in blue) and rotation angle around the x-axis θ_x (in grey), and the lengths of the vector and its xy projection. (Source: http://www.fundza.com/mel/axis_to_vector/index.html)

E. Model Placements

Once a group of scatter points has been generated and placed on a surface, the final task is to replace them with one or multiple models. If multiple models are selected they will replace the scatter points based on a random uniform distribution. The models are then transformed based on the scale and rotation of the individual scatter points. Details on the scatter point transforms are explained in Section II-F.

F. Tool Parameters

For the implementation of the scatter tool, different parameters were added to make the scatter customizable. These parameters include options for

- selecting between two different sampling methods and adjusting the point density by setting the disc radius of the Poisson sampling, or, the resolution of the random distribution;
- applying a random rotation around the y-axis based on a specified interval between $[-\pi, \pi]$;
- applying a random scale variation between a specified interval of $[s_{min}, s_{max}]$, $s_{min} <= s_{max}$, where the scalar values are uniformly proportional to the original scale of the models used to replace the scatter points.

When a set of scatter points is generated, all points are placed in the same group. The user can set the colour as well as setting a custom group name for the scatter point group. The group names are used when it is time to replace the scatter points with models.

III. Results

The tool script was implemented for Maya using the python API together with the OpenMaya API.

The GUI for the tool can be seen in Figure 6. The drop down menu at the top is used to select the preferred sampling method for generating the scatter points, either Poisson-disk sampling or random distribution, which are explained in Section II-B. This is followed by setting the disk radius of which the Poisson-disk method uses. Figure 7 shows a direct comparison between the two sampling methods. The Poisson-disk scatter is more evenly distributed than the random distribution scatter, which is prone to have clusters.

Next is the setting to align the scatter point to the surface normal, this is a toggle button because it is not always necessary to align the model with it surface. In Figure 8, the orientations of a set of scatter points has been adjusted to the surface normals of a sphere, as described in Section II-D.

The *Randomized Local Y-Axis Rotation* and *Uniform Scaled Randomization Interval* settings are described in Section II-F. The result of using random rotations combined with random scaling can be seen in Figure 9 where the forest is created with only one tree model. Another example of this combination is also shown in Figure 9, where the group of sand colored rocks is created with one model using varying scaling and rotations.

Figure 10 shows the result of selecting multiple models, in this case three differently coloured flowers, when replacing the scatter points. As displayed, the different models are equally distributed over the field.

Figure 11 shows how the ray casting method can be used for artistic control of creating scatter fields. In this example, two disk planes were selected for the same scatter point group, where the top plane acted as a mask for the lower plane. The result is a crude asteroid field surrounding a bigger body.

IV. Discussion & Future Work

When the surface orientation function was first implemented it was not toggleable, this was later changed because not all types of models need to be adjusted to the normal of the surface. For example, trees mostly grow straight up and stones orients along the surface.

The hierarchical dart throwing algorithm was very efficient in generating large scatter point sets, the main time consuming part proved to be the ray casting process. No effort was made to optimize this part, which could be a future improvement.

The presented tool fulfills the main purpose of being a scatter tool but there are still improvement and features which can be added. An improvement could be to have the noise functions be more procedural/deterministic and add more different noise functions for scatter distribution. Further additions would be to allow the user to change the orientation and position of the sample plane as well as specify the direction used in the ray casting. Another improvement is to implement a brushing tool to highlight on the surface where the scatter points should be placed instead of selecting entire surfaces or faces.

V. Conclusion

A tool for generating scatter points which can be replaced with one or multiple models has been presented. The scatter distribution is done with two different methods, Poisson-disk sampling and random distribution. Poisson-disk distribution has the property, unlike ordinary random distribution, that it guarantees evenly distributed points without clustering. This is particularly useful for creating convincing forests as seen in Figure 9.

The hierarchical dart throwing algorithm can successfully generate a maximal set of Poisson-disk sample points with linear time complexity. This makes the algorithm a suitable choice for an interactive tool.

The method for placing the scatter points on surfaces by using ray casting can be utilized in artistic control of shaping scatter fields.

The presented results show that the tool can produce a natural looking forest environment using random based patterns with only a few unique models.

References

[1] Stefano Cieri, Adriano Muraca, Alexander Schwank, Filippo Preti, and Tony Micilotta. The jungle book ; art-directing procedural scatters in rich environments. (2), 2016.

[2] Kenric B White, David Cline, and Parris K Egbert. Poisson disk point sets by hierarchical dart throwing. *IEEE Symposium on Interactive Ray Tracing Interactive Ray Tracing*, pages 129–132, 2007.

[3] Robert L Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, 5(1):51–72, 1986.

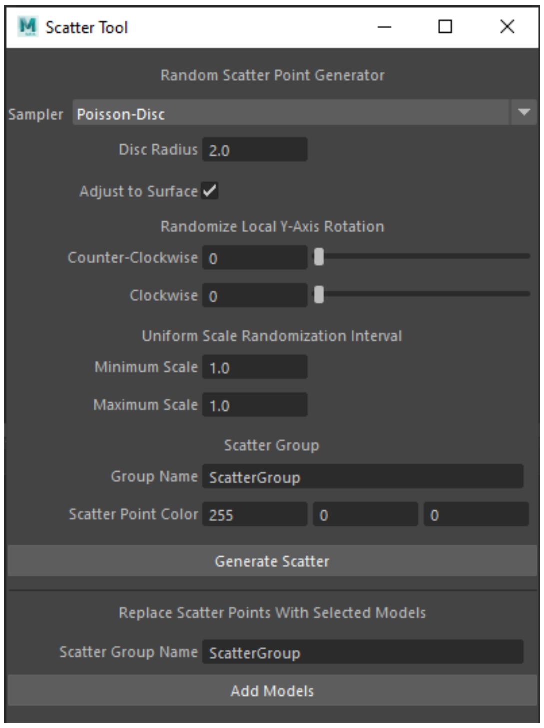
[4] Dong-Ming Yan, Jianwei Guo, Xiaohong Jia, Xiaopeng Zhang, and Peter Wonka. Blue-noise remeshing with farthest point optimization. *Eurographics Symposium on Geometry Processing*, 33(5), 2014.

[5] B Spencer and M W Jones. Progressive photon relaxation. *ACM Transactions on Graphics*, 32(1), 2013.

[6] Dong-Ming Yan, Jianwei Guo, Bin Wang, Xiaopeng Zhang, and Peter Wonka. A survey of blue-noise sampling and its applications. *Journal of computer science and technology*, 30(3), 2015.

[7] David Cline, Stefan Jeschke, Kenric B White, Anshuman Razdan, and Peter Wonka. Dart throwing on surfaces. *Computer Graphics Forum*, 28(4):1217–1226, 2009.

[8] Malcolm Kesson. Align y axis to vector. 2002.



Scatter Tool

Random Scatter Point Generator

Sampler: Poisson-Disc

Disc Radius: 2.0

Adjust to Surface: ✓

Randomize Local Y-Axis Rotation

Counter-Clockwise: 0

Clockwise: 0

Uniform Scale Randomization Interval

Minimum Scale: 1.0

Maximum Scale: 1.0

Scatter Group

Group Name: ScatterGroup

Scatter Point Color: 255 0 0

Generate Scatter

Replace Scatter Points With Selected Models

Scatter Group Name: ScatterGroup

Add Models

Fig. 6. The final look of the tools GUI implemented in Maya.

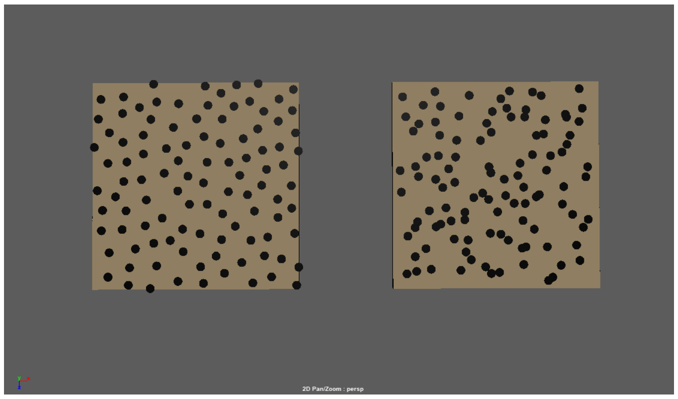


Fig. 7. A comparison between Poisson-disk distributed samples (left) and randomly generated samples (right) using our scatter point tool in Maya. Both sets consists of about 100 sample points each.

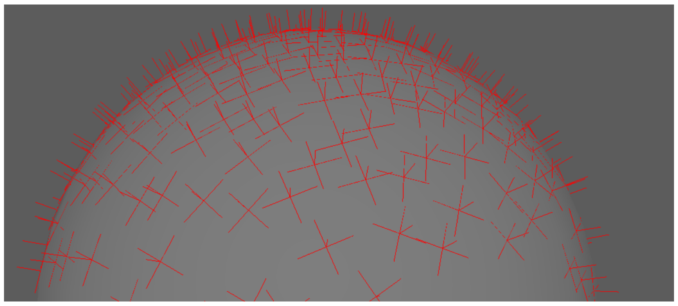


Fig. 8. Poisson-disc distributed scatter points, represented by red *Space Locators*, has been adjusted to the surface of a sphere.

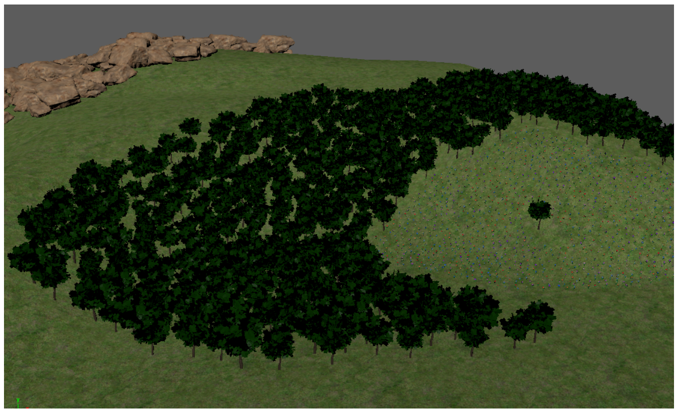


Fig. 9. A scene created in Maya 2020 using our scatter point tool. A single tree and rock model was used in creating the entire forest area, and the rock section. Poisson-disk scatter distribution was generated by selecting groups of individual faces of the ground surface.

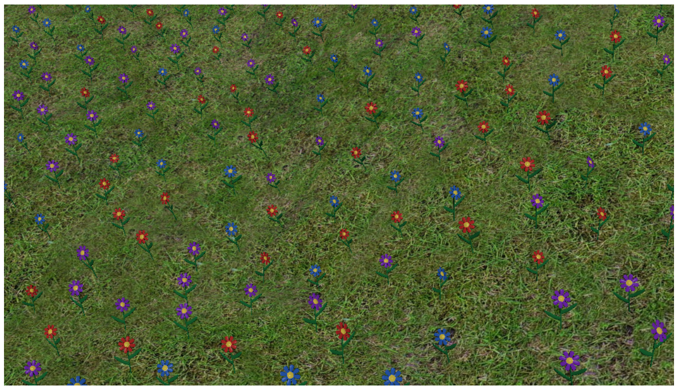


Fig. 10. The result of replacing one set of scatter points with three different flower models. The scatter points were generated using the Poisson-disk distribution with randomized scale and y-axis rotation.

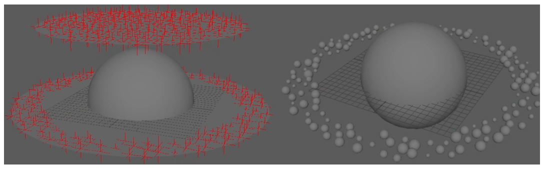


Fig. 11. Example of how the ray casting can be used artistically. The left image shows a group of scatter points covering two disk planes. In the right image, the disks as well as the top points has been removed and the remaining points has been replaced by spheres of varying sizes.