ISSN 2464-4617 (print)
ISSN 2464-4625 (CD)

Computer Science Research Notes
CSRN 2802

Short Papers Proceedings
http://www.WSCG.eu

Real-time visualization of Dynamic Unlimited Objects Instancing

Szymon Jabłoński
Institute of Computer Science
Warsaw University of Technology
ul. Nowowiejska 15/19
00-665 Warsaw, Poland
s.jablonski@ii.pw.edu.pl

Tomasz Martyn
Institute of Computer Science
Warsaw University of Technology
ul. Nowowiejska 15/19
00-665 Warsaw, Poland
martyn@ii.pw.edu.pl

ABSTRACT

In this paper, we propose a novel approach to an efficient rendering of an unlimited number of dynamic and unique 3D objects in real-time. We present an extension to the Holistic Unlimited Object Instancing (UOI) rendering pipeline and the holistic computer graphics paradigm. We called this extension Dynamic Unlimited Object Instancing rendering pipeline. Using Signed Distance Functions (SDF) for the virtual scene representation and the Holistic Scene Dynamics Function, we can control and render an unlimited number of dynamic 3D objects in real-time. In order to solve some issues of the original UOI rendering pipeline, we developed two extensions: first, a collection of holistic Dynamic operators, and, second, the Multipass Depth-Based Ray Marching rendering pipeline. The operators are used to apply affine transformations to an unlimited number of 3D objects and also to animate their materials and other attributes. In order to solve the problem of the uniform object distribution within the scene, we redefined the original definition of the scene SDF component. The virtual scene equation is divided into independent SDF components, which are rendered separately using the Multipass Depth-Based Ray Marching pipeline. Thanks to both extensions, the new version of the Holistic UOI rendering pipeline can handle 3D objects intersections what significantly enhances the realism of SDF scenes. The presented extensions to the UOI rendering pipeline are fully compatible with the Holistic UOI rendering pipeline, SDF and Sparse Voxel Octree (SVO) based algorithms. The only hardware requirement for our approach is the support for multipass rendering with compute shaders or any GPGPU API.

Keywords

Computer graphics, signed distance function, holistic programming paradigm, voxel rendering, sparse voxel octree, instancing, data-based amplification, procedural generation, fractal noises, level of detail

1 INTRODUCTION

Virtual scene geometrical complexity is one of the most common indicators used to evaluate the quality of real-time realistic image synthesis. In order to achieve the desired depth and realism of virtual worlds, the scenes should be composed of high-resolution 3D objects with detailed geometries and materials. Moreover, to provide an appropriate level of immersion of the user in the virtual environment, we have to use suitable efficient rendering techniques and algorithms to process and visualize the scenes in real time.

Over the years, many algorithms for virtual scenes management [Greene95], level of detail control [Lueb02], objects culling [Bittner04], and geometry instancing [Carucci05] have been developed. However, despite the constantly increasing computational power and memory capacity of today's GPUs, still, the main limitation of video game engines is the object space computation complexity [Jab17]. Also, one of the often overlooked factors influencing the quality of synthesized scenes, when regarded from the standpoint of the user's immersive perception of the virtual environment, is the "evolving" complexity of the virtual world that undergoes structural changes over time. We decided to focus mainly on this issue in this paper.

On the other hand, much effort has been devoted to studying alternative representations of geometry for real-time graphics. Signed Distance Functions (SDF), which derive from fractal theory and ray-tracing of quaternion Julia sets [Hart89], have found application in modern video game engines, among others for shadow map generation [Wright15] and font rendering [Green07]. Voxel-based representations, which had been used mainly in offline computer graphics, thanks to developed Sparse Voxel Octree algorithms [Crassin11, Jab16, Domaradzki16] can now be used successfully in real-time graphics. The research we conducted on these two approaches to representing geometry for computer graphics has resulted in the development of the Holistic Unlimited Object Instancing (UOI) rendering pipeline.

The SDF-based representation has been successfully integrated with Sparse Voxel Octree algorithms in the Holistic UOI rendering pipeline [Jab17]. Thanks to the screen-space computation complexity of the SDF and SVO processing algorithms along with the newly proposed *Holistic Graphics Programming* paradigm, this allowed for a significant increase in the complexity of virtual scenes that can be rendered in real-time. By using SDFs for the scene representation integrated with SVOs as a 3D geometry representation, the idea behind object instancing was extended in that it was possible to render in real time actually an unlimited number of 3D objects created by artists.

Nevertheless, the original concept of the Holistic UOI rendering pipeline is limited in several aspects and in this paper we propose solutions to these issues. First of all, we present a novel approach to an efficient rendering of a potentially unlimited number of *dynamic* and *unique* 3D objects in real-time. By extending the original Holistic UOI rendering pipeline with *Dynamic operators* and *Holistic Scene Dynamics Functions*, we are able to add movement and animation of objects' attributes to originally static scenes. Moreover, thanks to redefining the scene SDF component, we can effectively limit the uniform object distribution artifact, which was inherent to the original definition of the component. To express with the name the functionality offered by our extension to the original UOI pipeline, we call it the *Dynamic Unlimited Object Instancing* rendering pipeline.

2 RELATED WORK

In the paper "Unlimited Object Instancing in real-time" [Jab17] the holistic computer graphics programming paradigm was introduced and embodied therein in a novel Holistic UOI rendering pipeline. Thanks to this new holistic approach to expressing scenes for computer graphics and the dedicated rendering pipeline, it was possible to process and render a potentially unlimited number of unique 3D objects in real-time. The main foundation of the presented approach was the integration of SDF and SVO algorithms in a single-pass rendering pipeline.

The original Holistic UOI rendering pipeline was based on four main components. In the context of the topic we tackle in this paper, the most important is the component of *Global operators*. It was used to control the content and complexity of the virtual scenes.

Using a collection of Global operators, it is possible to instantiate an unlimited number of 3D objects, generate and apply object variations, and control the existence of objects in the virtual scene.

Thanks to the SDF-based representation and a holistic approach to control, the memory requirements for the scene description were significantly reduced, making processing an unlimited number of the 3D object for each SDF component possible. However, the original implementation suffers from two problems, which are essential from the standpoint of realistic and immersive rendering of a virtual world.

The first issue is that the rendered worlds were static and the original architecture of the holistic pipeline makes it difficult to introduce any kind of movement to these unlimited but indeed "frozen-in-time" virtual worlds.

The second issue is related to the inherent to SDF instancing, easily noticeable artifact of the uniform distribution of objects populating the scene. In this paper, we provide the solutions to both these problems.

There is a wide selection of literature related to each component used in our Dynamic UOI rendering pipeline. The SDF-based graphics representation derives from a method introduced in the paper [Hart89] for the visualization of quaternion Julia sets. The idea of unbouding volumes presented there was then extended by Hart et al. [Hart94, Hart97] into *sphere tracing*. Given an object represented by an SDF, sphere tracing relies on iteratively traversing a ray from the eye through the projection plane towards the object. If the eye-to-object distance estimation is smaller than a predefined precision value, the ray is considered to hit the object. SDF functions can be used to create highly detailed procedural objects using SDF primitives with boolean operators. Reiner et al. [Reiner11] presented an introduction to an interactive SDF ray marching pipeline with a procedural object generation based on domain operations.

In turn, thanks to the development of SVO algorithms, the high-resolution voxel-based representation can now be used in real-time graphics applications. Due to the screen-space character of the computation complexity of the SVO rendering pipeline, numerous high-resolution 3D objects can be processed in real-time using instancing approach. Cyril Crassin was able to perform visualization of the global illumination using SVO and voxel cone tracing [Crassin11]. There are also a few promising SVO methods for object animation, deformation, and fracturing in real-time [Bau11, Wil13, Domaradzki16]. The SVO-based object representation has also found application in continuous LOD management [Jab16].

For these reasons, the SVO-based representation has been becoming an increasingly serious alternative to polygon-mesh representations and, as such, is a promising candidate to be utilized in the holistic rendering pipeline.

There are also a few interesting papers about procedural generation of infinite cities which are worth mentioning [Greu03, Stein14, Steninb14].

The last group of papers is related to the topic of the procedural generation of geometry by means of the data amplification approach. Since there is a vast literature on fractals and procedural graphics, below we will focus only on the papers most relevant to our work.

Ken Perlin introduced a relatively simple and efficient method for generating a space continuous, pseudo-random noise for computer graphics [Perlin02]. It has been used across the computer graphics applications from terrain generation, objects randomization to special effects. There are also many improvements to Perlin's original idea that can be relatively easily implemented in today's GPUs [Li15].

Deussen et al. [Deussen98] presented a great example of how to exploit the data based amplification approach with geometry instancing in order to create realistic plant ecosystems in non-real-time graphics engines. Due to the limited capacity of GPU memory, real-time procedural content generation is required for creating complex and unique virtual scenes.

3 HOLISTIC UNLIMITED OBJECT INSTANCING

In this section, we present a short summary of the Holistic UOI rendering pipeline which was introduced in [Jab17]. We describe the main idea behind the holistic virtual scene definition, the available features, and the architecture of the Holistic UOI rendering pipeline. In particular, we focus on the issues of the UOI rendering pipeline which we deal with in this paper.

3.1 Holistic UOI rendering pipeline

The main foundation of the Holistic UOI rendering pipeline is the integration of the SDF and SVO algorithms in a single-pass ray marching visualization pipeline [Jab17]. The holistic approach is applied to the virtual scene definition. Rather than representing a virtual scene as a collection of individual objects, the whole scene is perceived and processed in its entirety as a complex object whose geometry is described by a single and (usually) relatively simple equation.

By using this new approach to the scene representation and visualization, which was termed as the *Holistic Graphics Programming*, it is possible to process in real-time as many unique instances of 3D objects as we want. The usage of SDFs allows memory requirements for the scene description to be significantly reduced, making it possible to deal with complex and even unlimited scenes with a low memory capacity. Moreover, thanks to incorporating the SVO representation into the SDF scene description, it is possible to render high-resolution 3D objects created by artists with the usage of e.g. Physically Based Rendering materials [Pharr17].

The features of the UOI rendering pipeline are as follows:

- Real-time processing and rendering of an unlimited number of unique 3D objects in the virtual scene.
- The possibility of visualizing 3D objects created by artists.
- Compatibility with other SDF and SVO based algorithms.
- Holistic content and complexity control with a data amplification method.
- A continuous LOD management of the virtual scene.

3.2 Holistic UOI architecture

Fig. 1 presents the four components the Holistic UOI rendering pipeline.

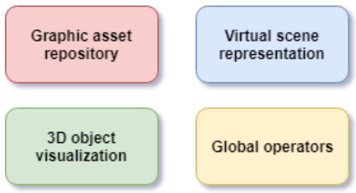


Figure 1: The components of the Holistic UOI rendering pipeline.

In this paper, we mainly focus on developing an extension to the *Global operators* component. The original paper [Jab17] introduced *Transition operators* to apply affine transformations to 3D objects. However, the capabilities of the operators were limited and they didn't take into account the passage of time in the virtual world. In the next section, we discuss two major issues of the original concept of the UOI rendering pipeline.

3.3 Holistic UOI issues

The holistic UOI rendering pipeline offers the possibility to handle an unlimited number of unique 3D objects in the virtual scene in real-time.

The original *Global operators* component gathers various instancing, geometry and material operators. Although there was a class of object transformation operators available, they did not offer satisfactory results. The two main issues were the uniform distribution of objects within the scene and no support for possible objects' intersections.

3.3.1 *Uniform object distribution problem*

The first problem pertains to the construction of the *Instancing operator*—the principal operator of the holistic UOI approach. In order to generate an unlimited number of 3D objects, a modulo function is applied to the scene distance function. The result is that a single scene SDF component, which represents an object, is repeated with a defined interval and, thus, a uniform grid of the object's copies is generated, populating the virtual world.

Fig. 2 presents rendering results of a virtual scene represented by a single scene SDF component with a modulo instancing operator applied.

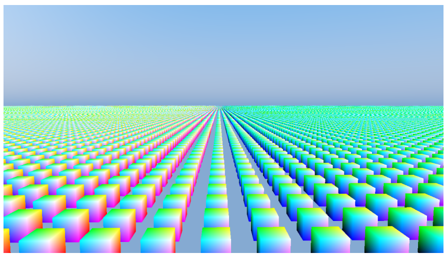


Figure 2: Uniform object distibution problem visible on single SDF component scene with Instancing Operator applied.

The original Holistic UOI rendering pipeline offers a collection of operators that can be used to reduce the visibility of this artifact—for example, the *Existence operator* could be applied to partially overcome this issue. Nevertheless, despite the application of the operator, there are always many vantage points from which the uniform object distribution is still noticeable, as we can see in Fig. 3.

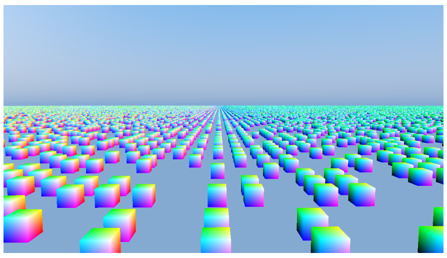


Figure 3: Uniform object distibution problem visible on single SDF component scene with *Existence operators* applied.

3.3.2 *No 3D objects intersection support*

The second problem is related to the integration of SFD and SVO in a single-pass rendering pipeline. The biggest implementation challenge for the Holistic UOI development was dealing with potential object occlusion errors [Jab17]. In the Holistic UOI rendering pipeline occlusion errors were fixed using multiple ray marching iterations. If an occlusion error occured, the grid cell coordinates and the SDF component id were stored. Then, the distance to the grid cell from the previous ray marching iteration was calculated and subtracted from the scene equation in the next iteration of ray marching algorithm. Thanks to that, the distance to the potentially occluded 3D objects could be found.

Although the algorithm is relatively simple and efficient, it also causes a serious problem, because cutting off the previous grid hit by a ray may result in that the 3D objects associated with the cell and potentially intersected by the ray are removed from the scene, too. In order to solve this issue, it is necessary to find the distance of the intersection between the multiple SDF components and apply it to the virtual scene definition [Jab17]. It means that as the result the complexity of the algorithm increases significantly.

In this paper, we propose a different solution to this problem—rather then the original single pass rendering, we make use of the Multipass Depth-Based Ray Marching (Sec. 4.3).

4 DYNAMIC UNLIMITED OBJECT INSTANCING

In this section, we describe the developed extension to the Holistic UOI rendering pipeline which we called *Dynamic Unlimited Object Instancing*. The Dynamic UOI rendering pipeline is based on the following three components:

1. **Holistic Scene Dynamics Function**—a continuous function parameterized by time and used to procedurally generate unique, dynamic variations of 3D objects populating the virtual scene.
2. **Dynamic Operators**—an extension to the original collection of the *Global operators* from the original Holistic UOI rendering pipeline. The dynamic operators are used to apply unique affine transformation and material animation to 3D objects. They utilize the *Holistic Scene Dynamics Function* to calculate dynamic variations per 3D object.
3. **Multipass Depth-Based Ray Marching**—an extension to the Holistic UOI pipeline which is based on a multipass rendering rather than—as it was in the original implementation—a single-pass rendering.

4.1 Holistic Scene Dynamics Function

The first component of the *Dynamic UOI* rendering pipeline is the *Holistic Scene Dynamics Function* (the HSD function for short). Though the function is an integral part of the *Dynamic Operators* component, we decided to define it as an independent component for the following reasons:

First, the HSD function is a perfect example of the implementation of the *Holistic Graphics Programming* paradigm. Instead of controlling each 3D object in the virtual scene independently, we animate the whole scene by means of a relatively simple equation. Secondly, the form of the function strongly depends on the scene content. For example, a different HSD function will be used to control an animation of flying 3D objects and a different one to sway grass under the wind.

In general, the HSD function can be expressed as:

$$f_{HSD} : D \times t \to V \tag{1}$$

where:

V	= a dynamics variation for a given 3D object
D	= the object input data
t	= the current simulation time

For example, a HSD function implemented using Perlin's noise algorithm could generate a color and other material attributes of a 3D object as well as its transformation matrix as an output by using the object's world space position or its grid cell as an input.

A good example of a potential application of the HSD function is the impact of wind. Based on the passing time, the world space position and the SDF component unique id, we could calculate, for example, translation matrices for 3D objects.

It can be expressed as a simple pseudorandom noise generator or by using a more sophisticated method based on, for example, Fractional Brownian Motions [Mandelbrot68] or Vector Fields [Chen11]. In this paper, we use relatively simple HSD functions based on trigonometric functions and continuous noise generators.

4.2 Dynamic Operators

The second component of the Dynamic UOI rendering pipeline is a collection of *Dynamic Operators* that are used to apply changes to the static virtual scene, processed using the rendering pipeline.

Thanks to the Dynamic Operators, 3D objects can be transformed with unique affine transformations per instance and/or have their attributes animated in real-time. The Dynamic Operators extend the *Global operators* collection with the additional time dimension [Jab17].

4.2.1 Dynamic Operators architecture

The processing pipeline of the Dynamic Operators slightly differs from that related to the remaining operators. This is particularly evident in the example of the transformation operators which were applied in the original holistic approach using following processing pipeline [Jab17]:

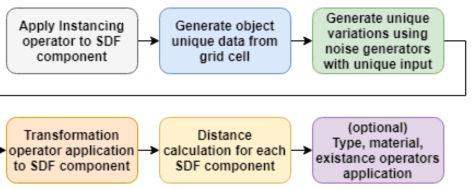

Apply Instancing operator to SDF component
Generate object unique data from grid cell
Generate unique variations using noise generators with unique input
Transformation operator application to SDF component
Distance calculation for each SDF component
(optional) Type, material, existance operators application

Figure 4: Global operators processing pipeline from original Holistic UOI rendering pipeline.

In the case of the Dynamic Operators, we need to perform an additional processing pass at the beginning of the holistic operator's application pipeline. In order to apply dynamic transformations for generated 3D objects, it is required to apply an additional transformation to a position on the ray from the camera to an SDF component at each iteration of the ray marching algorithm. It is required to preserve the correct form of the SDF.

The *Instancing operator* from the Holistic UOI rendering pipeline returns an object instance grid cell vector [Jab17]. The remaining *Global operators* used this value as an input for generating variations. For the Dynamic operators, we need to apply a transformation to the ray before applying *Intancing operator*. It means that we need to calculate the grid cell vector independently from the operator.

The architecture of the pipeline for the newly proposed operators takes the following form:

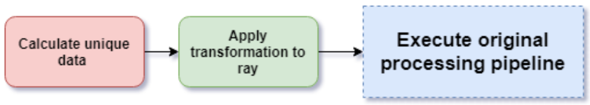

Calculate unique data
Apply transformation to ray
Execute original processing pipeline

Figure 5: Global operators processing pipeline developed for Dynamic UOI rendering pipeline.

Using the SDF-based object representation, the development of additional processing paseses is relatively simple. The cube distance function, which represents base scene SDF component [Jab17], can be extended as:

$$\begin{aligned}
gridCell &= floor((p + interval * 0.5)/interval)\\
variation &= f_{HSD}(time, gridCell)\\
&TrasOp(p)\\
&InstancingOp(p, interval)\\
&RotOp(p)\\
&ScaleOp(p)\\
distance &= length(max(abs(p) - size), 0)
\end{aligned} \tag{2}$$

where:

$gridCell$	= an object instance grid cell
$variation$	= an object instance dynamics variation
f_{HSD}	= the HSD function

$TransOp$	= Translation operator
$InstancingOp$	= Instancing operator
$RotOp$	= Rotation operator
$ScaleOp$	= Scale operator
$time$	= the elapsed simulation time
$interval$	= the repeat interval
$distance$	= the distance from the eye to the object
p	= a point on the ray from the eye to the object
$size$	= the scene SDF component cube size

4.2.2 *Dynamic Operators application results*

In this section, we present results of rendering virtual scenes with Dynamic Operators applied. In the following examples, we used a simple HSD function implemented with the use of the trigonometric functions supported by hardware.

Fig. 6 presents results of rendering a scene represented by a single scene SDF component with dynamic transformation operators applied.

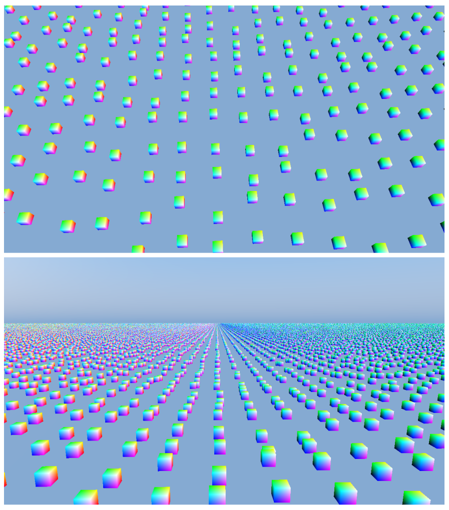


Figure 6: *Dynamic Operators* applied for the virtual scene represented by single SDF component in 2D.

The results show that the dynamic operators effectively solve the issue of the uniform object distribution. Moreover, they exemplify the possibility of the processing and visualization of an unlimited number of unique, dynamic 3D objects in real-time. They are also a good example of an implementation of the holistic programming paradigm accompanied by the data amplification approach.

Although the Dynamic operators are designed mainly as an extension to the *Transformation operators* from the original Holistic UOI, their usage is not limited only to 3D objects affine transformations. They could be also used to animate other objects attributes, e.g. the albedo color or the material's roughness values.

4.3 Multipass Depth-Based Ray Marching

The third component of the Dynamic UOI rendering pipeline is *Multipass Depth-Based Ray Marching* which turns a single-pass rendering pipeline into a multipass rendering pipeline.

The Dynamic UOI rendering pipeline with *Multipass Depth-Based Ray Marching* was developed in order to fulfill the following requirements:

- Support for 3D object intersections.
- Classic triangle rasterization rendering results integration support.
- Optimization and LOD management features for complex scenes.

4.3.1 *Scene SDF component redefinition*

In the original paper [Jab17], a virtual scene was represented using a single distance equation. Thanks to that, the whole rendering was performed in a single-pass. However, the available occlusion error-fixing algorithms do not support intersections between scene SDF components. In order to solve this issue, we decided to redefine the scene SDF function.

In order to render a scene with intersecting scene SDF components, we define each component as an independent virtual scene equation and render using a separate rendering pass. Then, so as to integrate the rendering results, a custom-made depth buffer is used.

The depth buffer is created as a second floating-point render target and utilized along with the ray-marching pipeline to store the minimum distance values obtained from the ray-marching passes. The depth buffer can be treated as another scene SDF component at the next rendering pass.

The integration of depth testing (if necessary for subsequent rendering passes) with the ray marching pipeline is quite simple. We need only to apply an additional check if the current distance traveled by the ray is smaller than the appropriate value in the depth buffer filled in in the previous passes.

4.3.2 *Multipass rendering pipeline*

A good example of the usage of the *Multipass Depth-Based Ray Marching* is rendering an open-world scene we prepared for this paper. The test scene consists of the following elements:

- **Terrain SDF component**—a procedural terrain distance function based on heightmap ray marching.
- **Trees SDF component**—objects of 3D trees created by artists with material and type operators applied. The component utilizes the terrain SDF function in order to snap 3D objects to terrain height by using *Translation operator* and *Existence operator*. Moreover, Dynamic operators are used to implement wind movement.

- **Grass SDF component**—grass objects with user-defined textures. Type and Material operators applied. 3D objects are snapped to the terrain with Translation and Existence operators applied in the same way as the previous component. Dynamic operators are also used to apply wind movement.

Fig. 7 presents outcomes of the subsequent passes of rendering the test scene. A more detailed discussion on the performance results of *Multipass Depth-Based Ray Marching* is given in the next section.

5 RENDERING AND PERFORMANCE RESULTS

All the given timings were obtained on Intel Core i5-8600K CPU with Nvidia GeForce GTX 1060 GPU and the algorithms were implemented using OpenGL 4.6 API with C++17 for Windows 10 64-bit.

We utilized 3D models Stanford Repository models [Stanford11] and other public resources [CGTrader, Sinnaeve] as test objects. In the tested scenes, we used the SDF function based on the online articles by Inigo Quilez [Iniqo08] and Alexander Alekseev [Aleksaeev14].

We prepared three virtual scenes: *Stanford*, *Terrain* and *Ocean*. For the second and third ones, we were using *Multipass Depth-Based Ray Marching* to handle 3D objects intersections and *Dynamic operators* to add movement to our scenes. The content of the *Terrain* scene was described in Sec. 4.3. The *Ocean* scene contains one scene SDF component for a procedural ocean and a second one for a herd of balloons. All the scenes are using the vast collection of *Global operators*, including instancing, type, material, and existence operators along with the newly developed *Dynamic operators* described in Sec. 4.2.

All the 3D objects are represented by SVOs with 10 levels of detail (1024 x 1024 x 1024 voxelization). Each voxel stored a compressed normal vector and texture coordinates.

The obtained rendering times (given in the figures) prove that the developed rendering pipeline is efficient and offers real-time performance. Moreover, the presented images show that the application of the Dynamic UOI rendering pipeline makes it possible to limit the noticeable regularity in object distribution inherent to the original algorithm.

The use of *Multipass Depth-Based Ray Marching* allows for handling 3D object intersections what effectively increases the depth and realism of rendered scenes. It also solves the limitation of the original occlusion error-fixing algorithm. Finally, introducing the newly developed *Holistic Scene Dynamics Function* component extends the holistic programming paradigm with movement and other possible changes to originally static objects.

As a part of the rendering performance tests, we performed an additional comparison test between the compute-shader-based and the pixel-shader-based rendering pipelines. In all performed test the rendering pipeline based on compute shaders was operating much faster. In our opinion, the compute-shader-based variant which offers the full control over the shader invocations is a better choice for a Holistic UOI implementation. Nevertheless, one should be aware that, unlike as in the case of the traditional triangle rasterization pipeline, the compute-shader implementation requires one to pay very close attention to every single line of code and even the number of registers in use.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we presented a novel approach to efficient rendering of an unlimited number of dynamic and unique 3D objects in real-time. Thanks to the developed extensions to the *Holistic Unlimited Object Instancing* based on the *Holistic Graphics Programming* paradigm, we successfully limited issues featuring the original approach.

Using the developed *Dynamic Operators* along with the *Holistic Scene Dynamics Function*, we can limit the artifact of the 3D object uniform distribution. Moreover, the introduction of changes in position and other attributes of the 3D objects populating the scene significantly increase the depth and the level of immersion featuring the virtual words created and rendered with the holistic approach.

Another issue of the original method—no support for 3D object intersections we solved using *Multipass Depth-Based Ray Marching*. The redefinition of the scene SDF component allowed for authoring complex virtual scenes by means of an efficient and relatively simple method. What's more, the implementation of the multipass rendering pipeline made the integration of SDF-based objects in the virtual scene much simpler. Moreover, thanks to incorporating the depth buffer into the ray-marching rendering, a depth-based integration with results obtained with the standard triangle-rasterization pipeline is possible (e.g., for particle effects, skeletal animation or animated, user-controlled 3D objects).

One should also note that *Multipass Depth-Based Ray Marching* makes it possible to define additional rendering optimization features. For example, each rendering pass could use different ray marching parameters (ray iteration number, precision, near/far planes, etc.) or even a different render target resolution.

An obvious step forward is an implementation of a more advanced *Holistic Scene Dynamics Function*.

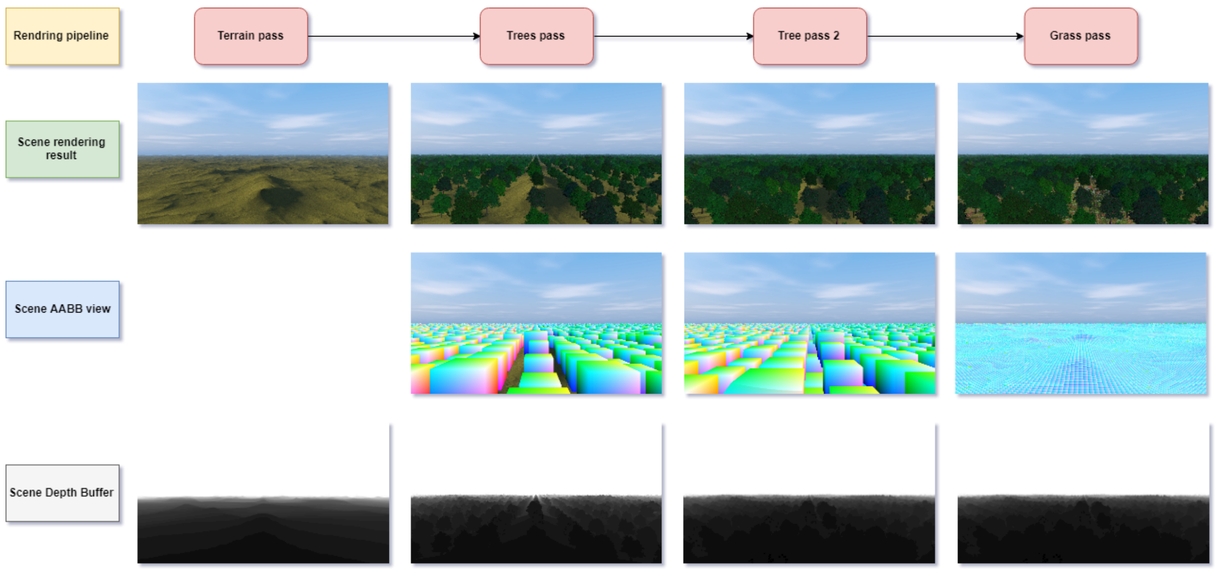
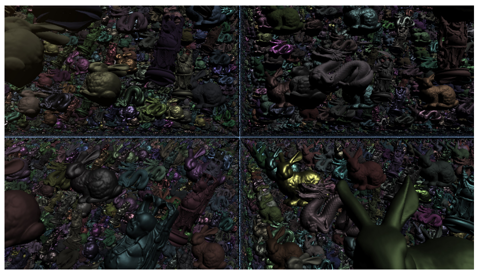
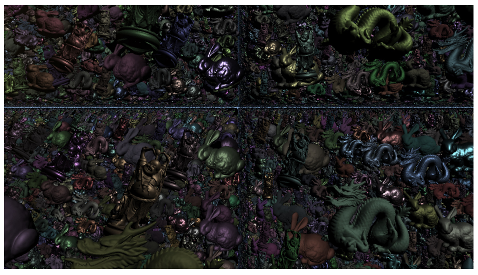


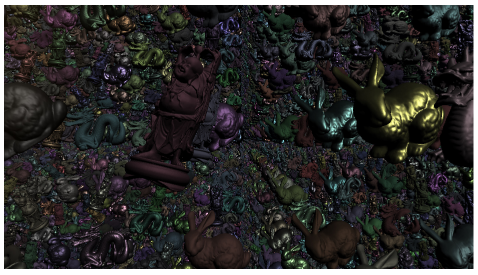
Figure 7: Multipass Depth-Based Ray Marching rendering pipeline application for test scene with multiple scene SDF components.



(a)

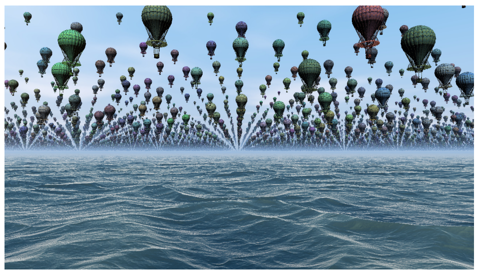


(b)

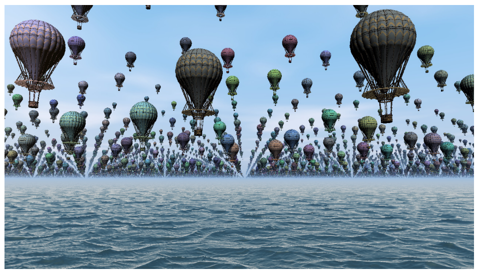


(c)

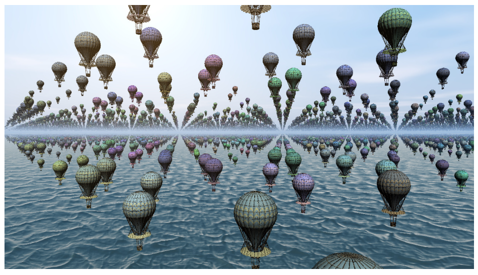
Figure 8: *Stanford* virtual scene with Dynamic operators applied with collection of original UOI operators. 40-50 FPS on Nvidia GeForce GTX 1060.



(a)



(b)



(c)

Figure 9: *Ocean* virtual scene with Multipass Depth-Based Ray Marching and Dynamic operators applied. 50-60 FPS on Nvidia GeForce GTX 1060.

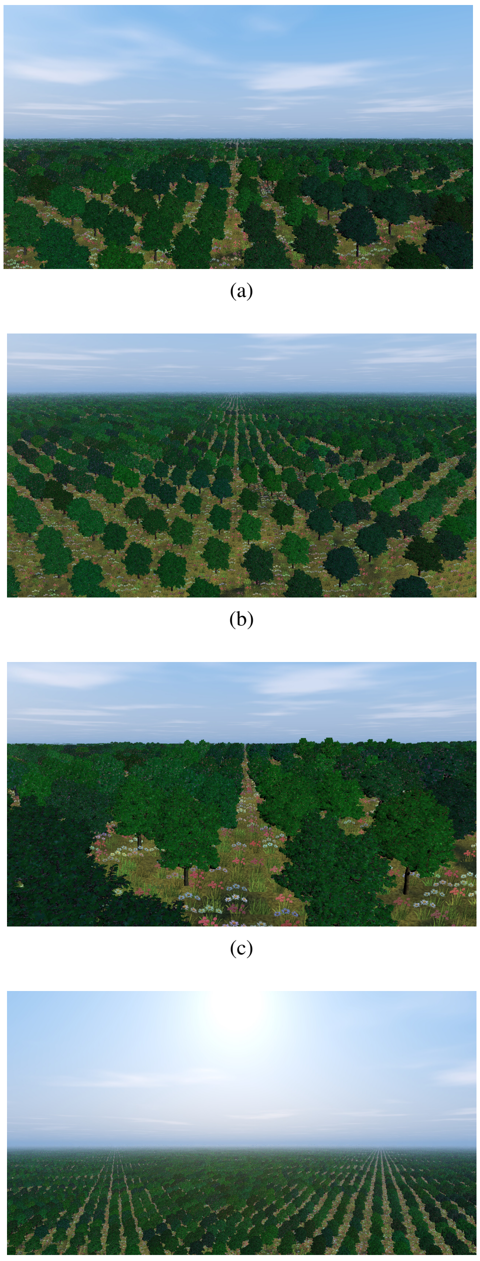

(a)
(b)
(c)
(d)

Figure 10: *Terrain* virtual scene with Multipass Depth-Based Ray Marching and Dynamic operators applied. 50-60 FPS on Nvidia GeForce GTX 1060.

A good idea seems to be the usage of a procedurally generated model of wind for realistic influencing 3D objects such as balloons, grass, etc. In our opinion, a collection of specialized movement functions should become a next important component of the Holistic UOI rendering pipeline. Also, a further optimization and extension to the *Dynamic Operators* should be taken into account in future work.

A further optimization is also possible for the *Multi-pass Depth-based Ray Marching*. For example, the integration with the Hierarchical Z-buffer [Greene93] algorithm instead of simple depth testing seems to be easy to implement. We suspect that it could result in a significant processing performance increase.

Finally, increasing the complexity of virtual scenes which is now possible by using dynamic objects in multiple rendering passes requires an additional research concerning the level of detail management for such a rendering pipeline.

7 REFERENCES

[Aleksaeev14] Seascape, Alekseev, A., https://www.shadertoy.com/user/TDM

[Bau11] Bautembach, D., Animated sparse voxel octrees, Bachelor Thesis, University of Hamburg, 2011.

[Bittner04] Bittner, J., M. Wimmer, H. Piringer, W. Purgathofer, Coherent Hierarchical Culling: Hardware Occlusion Queries Made Useful, Computer Graphics Forum, vol. 23 no. 3, pp. 615-624, 2004.

[Carucci05] Carucci, F., Inside Geometry Instancing, in Matt Pharr, ed., GPU Gems 2, Addison-Wesley, pp. 47-67, 2005.

[CGTrader] CG Trader, 3D models for VR / AR, 3D printing and computer graphics, http://www.cgtrader.com.

[Chen11] Chen, Cheng-Kai and Yan, Shi and Yu, Hongfeng and Max, Nelson and Ma, Kwan-Liu, An Illustrative Visualization Framework for 3D Vector Fields, Comput. Graph. Forum, number 7, vol. 30, pages 1941-1951, 2011.

[Crassin11] Crassin, C., Neyret, F., Sainz, M., Green, S., and Eisemann, E., Interactive indirect illumination using voxel cone tracing, Computer Graphics Forum (Proceedings of Pacific Graphics 2011), vol. 30, no. 7, sep 2011.

[Deussen98] Deussen, O., Hanrahan, P., Lintermann, B., Mesh, R., Pharr, M., Prusinkiewicz, P., Realistic modeling and rendering of plant ecosystems, Proceedings of SIGGRAPH 98, Orlando, Florida, July 19-24, 1998, In Computer Graphics Proceedings, Annual Conference Series, 1998, ACM SIGGRAPH, pages 275-286.

[Domaradzki16] Domaradzki, J., Martyn, T., Fracturing Sparse-Voxel-Octree objects using dynamical Voronoi patterns, Computer Graphics, Visualization and Computer Vision WSCG 2016. Full Papers Proceedings / Pan Zhigeng, Skala Vaclav (red.), Computer Science Research Notes, vol. 2601, 2016, Vaclav Skala - UNION Agency, ISBN 978-80-86943-57-2, pages 37-46.

[Greene93] Greene, Ned and Kass, Michael and Miller, Gavin S. P., Hierarchical Z-buffer visibility, SIGGRAPH In Proceedings, 1993.

[Greene95] Greene, N., Hierarchical Rendering of Complex Environments, Ph.D. Thesis, University of California at Santa Cruz, Report no. UCSC-CRL-95-27, June 1995.

[Green07] Green, C., Improved alpha-tested magnification for vector textures and special effects, Proceeding SIGGRAPH '07 ACM SIGGRAPH 2007 courses, pages 9-18.

[Greu03] Greutner, S., Parker, J., Stewart, N., and Leach, G., Real-time procedural generation of 'pseudo infinite' cities. In Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia (GRAPHITE '03). ACM, New York, NY, USA, 87-ff. DOI=http://dx.doi.org/10.1145/604471.604490

[Hart89] Hart, J., C., Sandin, D., J., Kaufmann, L., H., Ray Tracing Deterministic 3-D Fractals Computer Graphics 23(3), (Proc. SIGGRAPH 89,) July 1989, pages 289-296.

[Hart94] Hart, J., C., Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces, The Visual Computer, Volume 12, pages 527-545.

[Hart97] Hart, J., C., Implicit Representations of Rough Surfaces Computer Graphics forum, Volume 16, Issue 2, June 1997, pages 91-99

[Iniqo08] Quilez, I., Modeling with distance functions, http://iquilezles.org/, 1994-2017.

[Jab16] Jabłoński, Sz., Martyn, T., Real-Time Rendering of Continuous Levels of Detail for Sparse Voxel Octrees, Computer Graphics, Visualization and Computer Vision WSCG 2016. Short Papers Proceedings / Skala Vaclav (red.), Computer Science Research Notes, vol. 2602, 2016, Vaclav Skala - UNION Agency, ISBN 978-80-86943-58-9, pages 79-88.

[Jab17] Jabłoński, Sz., Martyn, T., Unlimited Object Instancing in real time, Computer Graphics, Visualization and Computer Vision WSCG 2017. Short Papers Proceedings / Skala Vaclav (red.), Computer Science Research Notes, vol. 2702, 2017, Vaclav Skala - UNION Agency, ISBN 978-80-86943-50-3, pages 91-100.

[Li15] Li, H., Tou, X., Liu, Y., Jiang, X., A Parallel Algorithm Using Perlin Noise Superposition Method for Terrain Generation Based on CUDA architecture, International Conference on Materials Engineering and Information Technology Applications, 2015.

[Lueb02] Luebke D., Watson B., Cohen, J., D., Reddy, M., and Varshney, A., Level of Detail for 3D Graphics, New York, NY, USA: Elsevier Science Inc., 2002.

[Mandelbrot68] Mandelbrot, B. B. and van Ness, J. W., Fractional Brownian motions, fractional noises and applications, SIAM Review, vol. 10, pages 422-437, 1968.

[Perlin02] Perlin, K., Improving Noise, ACM Transactions on Graphics, vol. 21, pages 681-682, 2002.

[Pharr17] Pharr, Matt and , and Jakob, Wenzel and , and Humphreys, Greg, Physically Based Rendering (Third Edition), Morgan Kaufmann, Boston 2017, ISBN 978-0-12-800645-0.

[Reiner11] Reiner, T., Mückl, G., Dachsbacher, C., Interactive modeling of implicit surfaces using a direct visualization approach with signed distance functions, Computers and Graphics, Volume 35 Issue 3, June, 2011, pages 596-603.

[Sinnaeve] A collection of free CG resources provided by Midge Sinnaeve, https://themantissa.net.

[Stanford11] The Stanford 3D Scanning Repository, Stanford University, 22 Dec 2010, Retrieved 17 July 2011.

[Stein14] Steinberger, M., Kenzel, M., Kainz, B., Mueller, J., Wonka, P., Schmalstieg, D., Parallel Generation of Architecture on the GPU, Eurographics 2014

[Steninb14] Steinberger, M., Kenzel, M., Kainz, B., Wonka, P., Schmalstieg, D., On-the-fly Generation and Rendering of Infinite Cities on the GPU, Eurographics 2014

[Wil13] Willcocks, C. G., Sparse volumetric deformation, Ph.D. dissertation, Durham University, 2013.

[Wright15] Dynamic Occlusion with Signed Distance Fields, Advances in Real-Time Rendering in Games, SIGGRAPH 2015.