Pokemon GO є яскравим прикладом AR технологій. Познайомитись з доповненою реальністю можна за допомогою одного лише смартфона.

Створення мобільного додатку для першокурсника буде спрямовано для спрощення та кращого засвоєння інформації. Програмний продукт має надати можливість швидко орієнтуватись в просторі, сприймати інформацію та саме головне заощаджувати дорогоцінний час.

Для вирішення поставлених проблем було використано наступні інструменти:

- Мови програмування Java
- Середовище розробки Android Studio
- СУБД Firebase

# OPTIMIZED VOLUME RENDERING IN OBJECT SPACE

**Sergey I. Vyatkin[1], Candidate of Technical Sciences, senior scientific researcher of Synthesizing; Alexander N. Romanyuk[2] , Doctor of Technical Sciences, Professor; Oksana V. Romanyuk[2], Candidate of Technical Sciences; Alla V. Denisyuk[2]**
**[1]Institute of Automation and Electrometry**
**[2]Vinnytsia National Technical University**

**Abstract**

In this paper, we propose an algorithm for volumetric rendering in object space (splatting), optimized by culling voxels that are invisible to the observer.

Compared to the standard voxel data storage method, the amount of information has been reduced by the group description of free voxels.

A special data storage format has been developed that allows the user to enter additional voxel attributes required for a particular task.

## 1. Introduction

An alternative to presenting objects with surfaces in a scene is the volume [1], in which the whole scene is a set of voxels – elementary volumes. They are kind of analogous to pixels in 2D graphics. Each voxel is usually realized by a cube or a ball. The main drawback of the volume is its size. The volume with an average resolution of $256^3$ requires storage of about 16 million voxels. To generate an image of a three-dimensional object on the screen, they must all be processed. However, the volume has a number of important advantages: it can represent the inside of the object, not just the outer layer. Rendering and processing do not depend on the complexity or type of the object, but only the resolution of the volume. There are two main methods of volume rendering: rendering in object space or rendering forward rendering (object-space rendering/forward rendering) and image space method (image-space method/backward viewing method).

In the first method, when traversing the entire volume of each voxel is considered as a 3D point, which is converted by the matrix species the transformation, and then projected into the Z-buffer and drawn onto the screen. There are two algorithms for volume traversal. The algorithm BTF (back to front) essentially coincides with the algorithm of Z-buffer is a pre-sorted array of voxels. The BTF algorithm bypasses the volume in the order of decreasing the distance to the observer, and Z-buffer is needed to remove the invisible parts of the object. The algorithm for the FTB (front to back) is basically the same as BTF, but voxels are processed in ascending order of distance. This should be noted, bearing in mind That the z-buffer algorithm can not provide proper rendering of translucent materials, as voxels are displayed on the screen in any order. In the second method, a ray is launched from the observer's view through the volume through each pixel of the image plane. In each resulting intersection of the beam with the volume, the opacity coefficient is calculated, taking into account the influence of the nearest voxels. Along the ray path, the intersection opacity is calculated until it reaches a certain value.

In this paper, we consider an optimized splatting algorithm, to which the following requirements were presented. The ability to store additional voxel attributes (for each application area specific attributes of the model). Storage in the description of the model of the minimum possible amount of information about free voxels (reducing to a minimum the amount of information relating to free voxels). The ability to save the model to a file (the development of the storage format of the model description in the file); the ability to view the internal structure of the model.; to minimize rendering time (optimization of the rendering algorithm to reduce the time of obtaining the image).

The purpose of this work is to develop an algorithm for visualization voxel models on personal, handheld computers, portable PDA, etc.in real time.

The work has a great relevance: the implementation will find application where physically accurate visualization of the model (Geophysics) is important, and there should be an opportunity to view the internal structure of the object (medicine).

## 2. Splatting

The algorithm is based on a method called splatting. The three-dimensional object is conditionally divided into voxels — three-dimensional points, and the flat imprint of each of them on the plane of the screen is called "splat". The basic algorithm includes the following steps. As a result of the convolution of each voxel with a certain kernel function (kernel), we obtain a 3D image of this voxel. The resulting 3D image of voxel is projected onto the screen plane. The energy of the 3D image is divided into pixels that are in the area of its projection. This area, together with the energy of the 3D image, is called voxel's fingerprint. The final image is formed by the integration of energy on all prints. As was said, the algorithm uses a convolution of voxel with a certain function called the kernel to fill the space between projected voxels. The core for each voxel is the same, and does not depend on the direction of view, so it can be calculated at the stage of preliminary data preparation.

Calculating the value of the imprint function during rendering takes a long time, so you should build a table of values before rendering. The most common imprint function is Gaussian because the table of values for it is considered relatively simple. You can increase the resolution to ensure better quality and prevent artifacts.

The method of representing objects as a set of points and using these as rendering primitives has been introduced in [2].

In [3, 4] authors describe a rendering technique called splatting which directly renders opaque and transparent surfaces from point clouds. Volume splatting is extension of surface splatting. A spherical 3D reconstruction kernel centered at each voxel is integrated along one dimension into a 2D- footprint function. Each voxel is projected onto the screen, the 2D footprints are accumulated directly into the image buffer.

Splatting proceeds as follows:

```
for each point P {
project P[k] to screen space;
determine the resampling kernel [k];
splat [s];
}
for each pixel x in the frame buffer {
shade x;
}
```

The resampling kernel is determined by the Jacobian mapping that transforms coordinates of the local surface parameterization to viewport coordinates. This mapping consists of a concatenation of an affine viewing transformation that maps the object to camera space, a perspective projection to screen space, and the viewport mapping to viewport coordinates.

In the viewing transformation there is no nonuniform scaling or shearing. This means we preserve the rotation invariance of basis functions in camera space. Therefore, the Jacobian of this transformation can be written as a uniform scaling matrix with scaling factor. As, since we restrict the viewport mapping to translations and uniform scaling, we can describe the Jacobian with a scaling factor. To compute the Jacobian of the perspective projection, we have to compute the local surface parameterization. After the viewing transformation, objects are given in camera coordinates that can be projected simply by division by the z coordinate. The center of projection is at the origin of camera space and the projection plane is the plane z = 1. We define the parameterization by choosing two orthogonal basis vectors u0 and u1 in the tangent plane. Since basis functions are radially symmetric, the orientation of these vectors is arbitrary. The tangent plane approximation leads to the same inconsistencies of the local parameterizations as in conventional rendering pipelines.

Splatting the resampling kernel, each point is mapped to the position on screen. Then the resampling kernel is centered and is evaluated for each pixel. The contributions of all points are splatted into an accumulation buffer. The projected normals of the points are filtered. In addition, color and normal components, each frame buffer pixel contains the sum of the accumulated contributions of the resampling kernels and camera space z values as well. Because the pixel grid in screen space is regular, the kernel can be evaluated efficiently by forward differencing in a rectangular bounding box and using lookup tables. The depth complexity of a scene is greater than one, thus a mechanism is required that separates the contributions of different surfaces when they are splatted into the frame buffer. Therefore, the z value of the tangent plane is computed at each pixel that is covered by the kernel, which can be done by forward differencing. To determine whether a new contribution belongs to the same surface as is already stored in a pixel, the difference between the new z value and the z value stored in the frame buffer is compared to a threshold. If the difference is smaller than the threshold, the contribution is added to the pixel. Given that it is closer to the eye-point, the data of the frame buffer is replaced by the new contribution. Deferred shading the frame buffer is shaded after all points of a scene have been splatted. This avoids shading invisible points. Instead, each pixel is shaded using the filtered normal. Parameters for the shader are accessed via an index to a table with material properties. This approach provides order independent transparency using a single rendering pass and a fixed amount of frame buffer memory. The general idea is to use a frame buffer that consists of several layers, each containing the data listed. A layer stores a fragment at each pixel. The purpose of a fragment is to collect the contributions of a surface to the pixel. After all points have been splatted, the fragments are blended back-to-front to produce the final pixel color. Splatted into a pixel is processed in three steps: 1. Accumulate or Separate decision. Using a z threshold all fragments of the pixel are checked to see if they contain data of the same surface as the new contribution. If this is the case, the contribution is added to the fragment and we are done. Otherwise, the new contribution is treated as a separate surface and a temporary fragment is initialized with its data. 2. New fragment insertion. If the number of fragments including the temporary fragment is smaller than the limit l, the temporary fragment is copied into a free slot in the frame buffer and we are done. 3. Fragment merging. If the above is not true, then two fragments have to be merged. Before merging, the fragments have to be shaded. When fragments are merged, some information is inevitably lost and visual artifacts may occur. These effects are minimized by using an appropriate merging strategy.

## 3. OPTIMIZED SPLATTING

Traditionally, a three-dimensional graph requires a lot of memory, since each voxel is stored with some data associated with it. In addition, for high-quality images in the algorithm the splatting filter, requiring large computations. In work [5], the method of volume visualization in which only those voxels, which belong to object surface, are processed is described. This significantly reduces the number of calculations. Another variant of splicing optimization – hierarchical splatting is described in [6], which uses a pyramidal data structure to represent the volume with multiple resolutions.

## 3.1 Description of the model

In this paper, we propose a way to describe a model that requires much less memory to store data than the standard approach. This method will allow you to store less information about free voxels. The model is represented as segments along the z-axis.

## 3.2 Storage model

Each voxel, in addition to the basic attributes (coordinates), stores additional data specific to a particular application.

Storing the model in the file is as follows: 1. for each additional voxel attribute, specify the number of bytes required to store it. 2. Each segment looks like the figure below. 1.

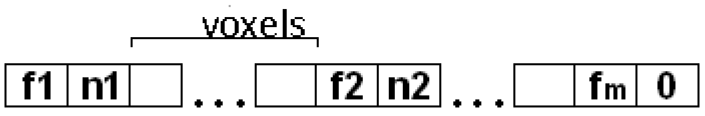

**Figure 1.** Description of the segment in the file

where fi is the number of free contiguous voxels, and ni is the occupied voxels, after ni follows ni of voxels, including the additional attributes. The description of the segment ends when ni equals zero.

Description of the segments that do not have occupied voxels in memory is not stored. The segments are sequentially starting from the first, forming a single sequence of bytes.

3. In a separate array for each segment, the byte number is stored in the sequence described in the previous paragraph, from which it begins. If the segment does not contain occupied voxels, the array corresponds to one.

When working with a model, only the sequence described in step 2 and the array described in step three are stored in memory.

This method of describing and storing the model allows significantly reducing the number of vacant voxish, and, consequently, filing size. At the same time, the high speed of working with data in the application is maintained, with the possibility of using additional voxel attributes.

### 3.3 Optimized rendering

The disadvantage of the original splatter algorithm is that it passes all the points of the model to obtain an image. However, to get the image quality is not worse than in the original algorithm, we only need to use a small amount of voxels. These voxels are closer to the screen and overlap the core radius of the voxels that are behind. The optimization algorithm consists of the following steps:

1. Create a buffer plane the size of the screen plane.
2. Passing through the whole model, we project voxel onto the buffer plane taking into account the action of the core radius. Fill the pixels of the plane in this radius value voxel by Z, if this value is less than the current pixel value. Otherwise, we will keep the same if the value of voxel by Z is greater than the current pixel value.
3. To apply the splatter algorithm, we select only those voxels whose projection on the plane at least partially exists.

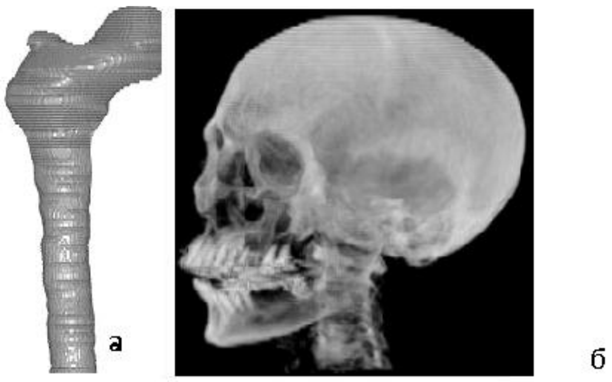

**Figure 2.** Models of the bone and skull

## 4. CONCLUSION

In this paper, we propose a method of visualization of volumetric data using the developed storage format models voxel-based approach. As a rendering algorithm was chosen splatting, optimized by rejecting invisible voxels.

Compared to the standard voxel-based storage method, the amount of information about free voxels was reduced by a group description of free voxels. The bill description format allows the user to enter additional attributes that are required to solve a specific task.

The application was implemented using the proposed method, as well as its testing, the results of which prove the fulfillment of the requirements: the size of the files was reduced by 20-25 times, the rendering time was also reduced by 5-6 times.

### References

1. O. N Romanyuk, S. I., Vyatkin, S. G Antoshchuk,. (2019). 3D Vector Fields Visualization Using Graphics Processing Units”. Research and Modeling of

Information Processes and Technologies. Herald of Advanced Information Technology; Vol. 2 No. 3: 2019, pp. 173-182.
2. Романюк О. Н. Високопродуктивні методи та засоби зафарбовування тривимірних графічних об’єктів. Монографія. / О. Н. Романюк, А. В .Чорний. ─Вінниця: УНІВЕСУМ-Вінниця ─2006. ─190 с.
3. S. Rusinkiewicz, M. Levoy. QSplat: A Multiresolution Point Rendering System for Large Meshes. In Computer Graphics, SIGGRAPH 2000 Proceedings, Los Angeles, CA, July 2000, P. 343–352.
4. H. Pfister, M. Zwicker, J. Baar, M. Gross. Surfels: Surface Elements as Rendering Primitives. In Computer Graphics, SIGGRAPH 2000 Proceedings, Los Angeles, CA, July 2000. P. 335–342.
5. L. D. Sobierajski, D. Cohen, A. Kaufman, R. Yagel, D. Acker. A Fast Display Method for Volumetric Data. *The Visual Computer,* 10(2):116-124, 1993.
6. D. Laur, P. Hanrahan. Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering. *Computer Graphics,* 25(4):285-288, July 1991.

## ФОРМУВАННЯ КОМПЕТЕНТНОСТІ ФАХІВЦІВ З ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ В ПРОЦЕСІ ЗАСТОСУВАННЯ ІНФОРМАЦІЙНОГО РЕСУРСУ

**А.М.Гафіяк, к.е.н., доцент**
**Національний університет «Полтавська політехніка імені Юрія Кондратюка»**

Тема інформатизації як ресурсу сучасного розвитку суспільства належить до тем, які визначають собою формування інформаційного простору епохи. Інформатизація – сукупність взаємопов’язаних політичних, соціально-економічних, організаційних, науково-технічних, виробничих процесів використання інформаційних і комунікаційних технологій (ІКТ) в усіх сферах соціально-політичного, соціально-економічного, культурного життя суспільства, спрямованих на автоматизацію виробничих процесів і процесів управління, на забезпечення прав юридичних та фізичних осіб отримувати, зберігати та поширювати інформацію (Концепція Національної програми інформатизації). Інформаційні ресурси – це окремі документи і окремі масиви документів в інформаційних системах (бібліотеках, архівах, фондах, банках даних, інших інформаційних системах). Інформаційна система – це організаційно упорядкована сукупність документів (масивів документів) та інформаційних технологій, у тому числі з використанням засобів обчислювальної техніки та зв’язку, які реалізують інформаційні процеси. Інформаційні якості – це згущення інформації, які заключаються в рішенні, що відображає управлінську ситуацію, задачу, проблему і включає потенціал управлінських впливів, покликаних перевести їх на новий рівень. Інформаційні