УДК 004.42 DOi 10.30837/ bi.2023.1(99).14

Eduard Sheliemietiev[1], Yuriy Novikov[2], Oleksii Nazarov[3], Nataliia Nazarova[4]

[1] ХНУРЕ, м. Харків, Україна, eduard.sheliemietiev@nure.ua

[2] ХНУРЕ, м. Харків, Україна, yuriy.novikov@nure.ua, ORCID iD: 0000-0003-1910-3256

[3] ХНУРЕ, м. Харків, Україна, oleksii.nazarov1@nure.ua, ORCID iD: 0000-0001-8682-5000

[4] ХНУРЕ, м. Харків, Україна, nataliia.nazarova@nure.ua, ORCID iD: 0009-0007-7816-7088

INVESTIGATE METHODS FOR SMOOTH TRANSITIONS BETWEEN LEVELS OF DETAIL FOR EFFECTIVE VISUALIZATION OF 3D SPACES

The object of study is 3D-rendering. The subject of study is providing smooth transitions between LOD models. The purpose of the work is to increase rendering performance of large 3D-scenes considering smooth transitions between models with different levels of detail. The study examines existing algorithms for enabling smooth transitions between levels of detail considering computational complexity and visual appeal of said methods.

3D-RENDERING, ALPHA-BLENDING, COMPUTER GRAPHICS, GEOMORPHING, LOD, NOISE-BLENDING, POPPING EFFECT

Шелємєтьєв Е.О., Новіков Ю.С., Назаров О.С., Назарова Н.В. Дослідження методів плавних переходів між рівнями деталізації для ефективної візуалізації 3D-просторів. Об'єктом дослідження є 3D-рендеринг. Предметом дослідження є процес забезпечення плавного переходу між LOD моделями. Метою роботи є підвищення ефективності візуалізації великих 3D сцен із урахуванням плавності переходу між моделями з різним рівнем деталізації. В ході роботи виконується дослідження існуючих алгоритмів плавного переходу між рівнями деталізації з точки зору обчислювальної складності та візуального вигляду. Результати дослідження дозволяють прийняти рішення про застосування того чи іншого методу плавного переходу з урахуванням їх сильних та слабких сторін.

3D-РЕНДЕРИНГ, LOD, АЛЬФА-ЗМІШУВАННЯ, ГЕОМОРФІНГ, ЕФЕКТ ПОППІНГУ, КОМП'ЮТЕРНА ГРАФІКА, ШУМИ

Introduction

Levels of Detail (LOD) play an important role in improving the performance of 3D graphics applications by striking a balance between frame processing speed and user experience. The essence of this optimization method is that the further away from the virtual camera, the simpler the 3D models are drawn. This allows you to focus computing resources on objects that are more important to the user.

Since the camera position is often dynamic, some objects need to be detailed directly in the user's field of view. The problem arises of how to smoothly transition between different levels of detail — replacing simplified models with highly detailed ones and vice versa.

This paper explores methods for smoothing the transition between levels of detail to mitigate the notorious "popping" effect. This effect occurs when the transition between LODs results in abrupt and noticeable changes in the level of detail, disrupting the visual coherence of the scene.

By studying and comparing different approaches, this research aims to provide valuable insights into effective strategies for preventing or minimizing the popping effect, which will ultimately help to balance performance and optimal visual user experience.

Thus, the goal of the work is to improve the efficiency of the visualization of large 3D scenes, taking into account the smooth transition between models with different levels of detail.

The subject of the research is 3D visualization (rendering).

The subject of the study is the process of ensuring a smooth transition between LOD models.

To achieve this goal, it is necessary to solve the following tasks

– To analyze existing methods and algorithms for eliminating the popping effect when transitioning between levels of detail;

– Study the factors that affect the efficiency of drawing distant objects;

– Implement and compare methods for smooth transition between levels of detail;

– Draw conclusions about the appropriateness of using a particular approach.

Based on the results obtained, it will be possible to draw conclusions about

– Which algorithm should be used to ensure the highest performance;

– Which algorithm provides the smoothest transition;

– To what extent it is generally advisable to use a smooth transition between levels of detail compared to discrete LODs.

This work is relevant now and will be in the future as 3D visualization becomes more widespread and has many applications in the modern world.

1. Subject area analysis

Today, 3D visualization has many applications: video games, graphic design, visual effects, virtual reality, cinematography, visualization, virtual engineering, etc. These fields

require 3D graphics to look good (realistic or stylized) and to be efficient. Often these two requirements are in conflict, and application specialists must find a compromise between the quality of the image and the cost of generating it. Real-time 3D rendering is particularly challenging from this perspective when the cost of producing an image is expressed in terms of frames per second (FPS).

Often, a very effective way to increase the speed of visualization is to use 3D models with different levels of detail. The essence of this method is to draw simpler geometry and use faster shaders the further the object is from the camera.

For example, instead of rendering every detail of a very distant glass skyscraper, you can draw a stretched cube with a pre-created texture. With this approach, it's important to balance the amount of detail with the distance to the viewer, as you can either lose the effect of immersion in the 3D world or get a very long frame generation time.

When working with LOD, you should also pay attention to the smooth transition between levels of detail as the camera approaches the object. For example, if you decide to draw distant trees using a square with a texture stretched over it (the so-called billboard), when the virtual camera quickly approaches the forest, you can see how the 2D image of the tree suddenly turns into a 3D model (popping effect). This phenomenon has a negative impact on the user experience, as it reveals the falsity of the virtual world.

Several methods are used to eliminate the popping effect: LOD blending (smooth blending of two adjacent levels of detail using transparency or noise) and geomorphing (creating additional intermediate states of the model by bringing the geometry of one level of detail closer to another).

Although there are a number of methods available today, each approach has its advantages and disadvantages, and it is often necessary to find a balance that satisfies the requirements for performance and appearance.

2. Work Relevance

This work has significant relevance in the context of 3D graphics and virtual environments. Its results solve a critical problem that directly affects user experience and application performance.

Modern applications and 3D models strive for ever-increasing levels of detail. In the past, computers could draw a relatively small number of 3D polygons, but in recent years, computer-generated images (CGI) have become indistinguishable from real photographs. This is largely due to the use of highly detailed models.

And although modern hardware is capable of drawing large numbers of polygons, computing resources are not unlimited (especially in the context of real-time 3D rendering and on mobile platforms), so software developers still have to make compromises. LOD is still a must-have method for maintaining the illusion of continuity and optimal performance.

Users moving in virtual space often experience the "popping" effect, which negatively affects immersion. Choosing the wrong method to smoothly transition between levels of detail destroys the user experience, as the viewer's attention is directed not to what the designer or developer intended, but to the 3D object that is transitioning to a different level of quality. By exploring different methods to prevent this phenomenon, this paper aims to offer practical solutions that can be applied in a wide range of applications: from video games to architectural modeling and virtual reality.

In summary, the relevance of the research lies in its ability to improve the user experience in a variety of applications. Whether it is games, simulations, or resource-constrained platforms, the results of this research can positively impact image accuracy, performance, and user experience.

3. Problem statement

In accordance with the identified problems, we describe the task for research practice:

– Analyze existing methods and algorithms for eliminating the popping effect when switching between levels of detail;

– To study the factors influencing the efficiency of rendering distant objects;

– Perform the software implementation of the studied algorithms;

– Draw a conclusion about the appropriateness of applying a particular approach.

Based on the results obtained, it will be possible to draw conclusions about

– Which algorithm should be used to ensure the highest performance;

– Which algorithm provides the smoothest transition;

– To what extent it is generally advisable to use a smooth transition between levels of detail compared to discrete LODs.

Thus, the study will allow us to choose one or another algorithm to ensure a smooth transition between levels of detail, taking into account their strengths and weaknesses.

4. Justification of research methods and stages

The object of research is 3D visualization (rendering).

The subject of the research is the process of ensuring a smooth transition between LOD models.

The goal of the research is to improve the efficiency of the visualization of large 3D scenes, taking into account the smooth transition between models with different levels of detail.

The empirical scientific method "experiment" is used in this research: during a series of experiments the parameters of the selected algorithms for ensuring smooth transition between levels of detail (the value of the popping effect and performance) are determined.

Then the theoretical method of "analysis" will be used to decompose the obtained results separately from each other. As a result, it will be decided which algorithm is more suitable in a given situation and for what reasons.

This research consists of the following stages

1) Preparation of the study — definition of the purpose, specification of the subject and object of the study, formulation of the hypothesis and research methodology;

2) Experimental research and data processing — preparing and conducting experiments, processing the obtained data;

3) Analysis of research results — drawing conclusions about the feasibility of using certain algorithms for smooth transition between levels of detail;

4) Evaluation of the application of the research results.

5. Development of formal mathematical model of subject area

The study will compare the algorithms for ensuring a smooth transition between LOD models by two factors:

– the time of image generation;

– the importance of the popping effect.

Each algorithm studied is characterized by the following gray box mathematical model (see Figure 1): it is an expression of the complexity of two adjacent levels of detail between which a smooth transition is made (x1 and x2) and the distance of the object from the camera (d), as well as a random error ξ that we cannot control.

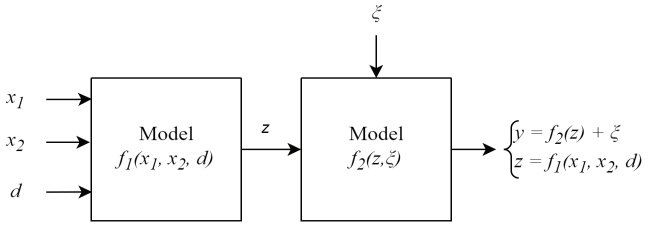

Fig. 1. Mathematical Model of the Gray Box for the Smooth Transition between Levels of Detail Algorithm

The function f1(x1,x2,d) is a controlled parameter that indicates the method of smooth transition between the levels of detail under study. The result of the function is the parameter z, which denotes a set of instructions to the GPU to draw 2 adjacent levels of detail at a given distance from the camera.

The function f2(z,ξ) denotes the environment that performs the drawing (the operating system and the GPU). This environment is uncontrolled and a black box. All we can do with it is pass input commands and data, and the output is the generated image.

Note that to simplify further calculations of the complexity of the x1 and x2 detail levels, the number of polygons in the model (triangles that form the object's edge) is taken without regard to the position of their normal relative to the camera (when using the back-face culling technique, polygons that are not visible from the camera, such as the back of a house, are cut off and do not participate in further steps of the graphical pipeline).

In the experiment comparing image generation times, y is the time (in milliseconds) to generate an image, and ξ is a random measurement error.

In the experiment to evaluate the significance of the popping effect, y is the difference between the reference image (the original high-poly model) and the image generated by the smooth transition algorithm between detail levels.

In this experiment, all random values (noise textures, etc.) are pre-generated, so the result of the experiment is completely deterministic (determined by x1, x2, d), and the error value ξ is zero.

After conducting the experiments, it will be necessary to conclude how well the algorithms meet certain criteria and to give examples of situations in which it is advisable to use a certain method of smooth transition between levels of detail.

6. Experimental methodology

To perform the experiments, we will create a software that consists of the following functional modules:

1) a module to measure the performance of the algorithms when displaying a large number of 3D models at different distances from the camera;

2) a module for measuring the popping effect when comparing the performance of the method under study with a reference (high-poly model).

Algorithms for smooth transition between the levels of detail to be studied:

– Alpha mixing;

– Mixing with noise;

– Geomorphing.

These methods are compared with rendering:

– The original high-poly model with no levels of detail;

– discrete levels of detail without smooth transition.

The experiment to measure the performance of the smooth transition algorithms will measure the time required to render a large number of models (50, 250, 500, 1,000, and 2,000 models) at different distances from the camera. The number of objects should be large enough because modern GPUs process large 3D scenes very quickly.

The objects are located along the x-axis in the range from the beginning of the camera's visible range to the distance required to activate the lowest level of detail at the same interval, which is equal to (1):

$$dx = (w - camera_x) / n, \quad (1)$$

where dx is the interval at which objects are placed in the scene, n is the number of objects, $camera_x$ is the camera coordinate, and w is the distance of the minimum level of detail.

Accordingly, the camera is oriented along the positive x-axis.

Also, occlusion culling should be turned off because the models overlap each other (if occlusion culling is turned on, the objects in front will overlap the models behind, and the latter will not be drawn).

As mentioned earlier, this experiment has an error ξ, so the time measurement should be performed several times (we assume that 3600 times is sufficient) and the average value should be taken.

Since the experiment is performed on a modern multiprocessing operating system, we assume that the time measurement errors are caused by other processes on the computer, so the average time is the time of the algorithm.

The magnitude of the popping effect is measured as follows:

1) The 3D model under study is placed on the scene at the minimum distance from the camera, so that the maximum level of detail is initially displayed;

2) the object is moved at a constant distance along the camera's line of sight, passing through all states of detail levels;

3) after each movement, the resulting image is stored;

4) after reaching the lowest level of detail, the object is gradually brought back in the same way;

5) Steps 1-4 are performed for the reference model and the specified algorithm, and the absolute difference between the pixel values of the images is found. The resulting image has only one channel, which is interpreted as black (value 0) and white (value 1). Since the input images are multichannel (RGB), the difference is the arithmetic mean of all channels;

6) for each difference image, the root mean square error (RMSE) is calculated [1].

As a result, we get a single number — how much the image generated by a particular method differs from the reference.

The image obtained in step 4 can be useful for visually assessing the difference between the algorithms.

As a result of the experiment you will find

– The fastest and slowest algorithms;

– which algorithm has the most pronounced popping effect.

Based on the results, it will be possible to draw conclusions about

– Which algorithm should be used to ensure the highest performance;

– Which algorithm provides the smoothest transition;

– the general advisability of using a smooth transition between levels of detail versus discrete LODs.

7. Nature of Experimental Errors and Uncertainty

The measurement error is present only when performing an experiment to evaluate the performance of algorithms for smooth transition between levels of detail. Despite the fact that the same set of data is passed to the program, the processing time is different each time.

This is due to the fact that the application is not running in a separate, isolated environment: many other processes and threads are running in parallel on the same device.

Hardware and software caching, code interpretation, and JIT compilation must also be taken into account. Because of this, the first experiment usually takes a little longer. Therefore, in practice, before measuring the execution time of a program, a so-called "warm-up" is performed (preferably with as many conditional transitions as possible). This increases the chance that subsequent experiments will have approximately the same runtime.

When calculating the size of the popping effect, operations are performed on floating point numbers (addition and division operations). It is known that this introduces an error in the resulting amount, but its relative value is very small and does not matter when comparing large numbers. Therefore, to simplify the experiment, its existence is allowed.

8. Alpha blending

One of the easiest methods to implement in software is the smooth transition method called alpha blending (LOD blending).

The essence of this method is to simultaneously display two levels of detail that are mixed using the alpha channel (transparency channel) according to the transition coefficient. This coefficient can take values from zero (the beginning of the transition) to one (the end). Conventionally, the method can be represented by the following formula (2):

$$\alpha_{result} = (1 - k) * \alpha_{prev} + k * \alpha_{next}, \quad (2)$$

where α_{result} — is the transparency of the resulting pixel on the screen, α_{prev} — is the transparency of the pixel of the previous detail level (where the transition started), α_{next} — is the transparency of the next (target) detail level, k is the transition coefficient.

An alpha channel value of one corresponds to a completely opaque pixel, and a value of zero corresponds to a completely transparent (hidden) pixel.

As the virtual camera moves and reaches the distance at which the transition should take place, the old model gradually fades out and a new model gradually appears in its place. When the transparency of the old model reaches zero, the old model is not drawn to save resources.

Visually, the principle of alpha blending is shown in Figure 2.

Fig. 2. Principle of alpha blending, transitioning from a low quality model to a high quality model

When implementing alpha blending, it is important to make the transition within a small distance range. For example, if you need to make a transition at a distance of 1m, the start and end of the transition should be 0.95m and 1.05m respectively.

Alpha blending can be combined with billboards because the virtual structure of the model is not important for this method. In the context of LOD optimization, billboards are usually used to improve the efficiency of visualizing the farthest objects (the lowest quality level).

Billboards are two-dimensional planes or sets of connected polygons that always face the camera, simulating a three-dimensional object. This technique is used to more efficiently represent distant or small objects in a scene, reduce computational load, and improve performance.

The advantage of billboards is that only two triangles are needed to represent a plane. Often, multiple planes are combined to create the illusion of a large object in space (tree leaves, bushes, clouds, etc.) [2].

This method of smooth transition has two major drawbacks.

First, visualizing two models at the same time is computationally intensive. The main reason for using detail levels is to reduce the number of polygons drawn simultaneously to speed up the creation of the image on the screen, but the transition in alpha blending requires drawing both models. As a result, this method can sometimes be detrimental to performance.

One way to deal with this drawback is to limit the number of objects that can transition between levels of detail at the same time. This helps to avoid jumps in the number of drawing function calls, which guarantees a more stable number of frames per second. Keep in mind that delaying the transition creates a delay that can be detrimental to the user experience.

Second, alpha blending is very noticeable to the viewer at close range: in certain situations, the model can look like a translucent ghost.

Alpha blending works very well at a distance from the camera, when adjacent layers of detail have a small number of polygons and the visual effect of blending is not noticeable enough.

9. Blending with noise

The smooth transition between levels of detail using noise is very similar to alpha blending.

However, in this method, the transparency of each pixel is a discrete value: either zero or one. In other words, this algorithm uses either fully transparent or fully opaque pixels to create a smooth transition.

Thus, the decision whether to display a particular pixel with x and y coordinates of each model is made based on the numerical value of a random variable (noise): if the value of the transition coefficient k exceeds the threshold set by the noise function, we use a pixel from the next level of detail, otherwise — the previous one (3):

$$\alpha_{result} = \alpha_{next}, \text{ if } k > f_{noise}(x, y); \quad \alpha_{result} = \alpha_{prev}, \textit{otherwise}, \tag{3}$$

where $f_{\text{noise}}(x, y)$ — is a noise function that returns a value between zero and one.

In computer graphics, noise is a pseudo-random value (one-dimensional or multidimensional) used to add detail to computer-generated images. Noise is very easy to compute, and its applications are almost limitless: from cloud and particulate visualization to ocean wave and tornado simulation [3].

For better performance, noise is pre-computed and stored as textures. This format is very convenient for GPUs as they are specialized in sampling data from textures.

So for a smooth transition, we need a two-dimensional noise value in the range of zero to one, stored in advance as a two-dimensional texture. Its size doesn't need to be large: the maximum size is the resolution of the generated image. Even if you compute a texture that is too small, it can be re-rendered, and this will minimize the user experience.

Noise texture sampling can be done at the processing stage instead of separately for each model to be drawn. This can speed up the algorithm and avoid noise repetition when objects are too close together.

The most popular noise functions include the following [4]:

a) white noise — returns pseudo-random values even for input values that are very close to each other;

b) gradient noise — interpolates white noise values and returns close noise values for close parameters;

c) pearlin noise — a subtype of gradient noise in which visual details have the same size; it is used to make computer graphics more realistic;

d) multilayer noise — uses a combination of gradient noise with different levels of scale and weight; allows you to get noise that has both high-frequency details and low-frequency details;

e) voronoi noise — returns a noise value that looks like a set of cells (or distances between cells).

A separate type of noise is dithering [5]. It is used to give a random variable the desired stylistic appearance. An example of a smooth transition using dithering is shown in Figure 3.

Fig. 3. Principle of noise blending, transition from low-quality model to high-quality model

The figure shows a useful feature of noise-based blending — at a certain value of the smooth transition threshold, only one model can be drawn at a time. This helps to reduce the load on the GPU. Of course, this trick is not suitable for all types of models and environments.

Another advantage of this method over alpha blending is that the designer has full control over the appearance of the transition: you only need to replace the noise texture, and you can make certain parts of the model transition faster than others; you can set the transition direction (from bottom to top, from edges to center, etc.).

Noise-based transitions between levels of detail have the same drawbacks as alpha blending: the need to draw two models at the same time and the obviousness of the transition when viewed up close.

In practice, however, the noise-based transition is more efficient in terms of performance [6].

The fact is that the most common way of working with translucent models is much slower than drawing opaque objects. Semi-transparent geometry often requires a separate graphics pipeline step from opaque objects, where the

depth test is disabled and transparent polygons are sorted from back to front [7].

The lack of a depth test causes every pixel to be redrawn (overdraw), which is very detrimental to performance as the GPU wastes time on a pixel that is not visible to the user anyway.

Sorting can be quite expensive when the number of model triangles is large, which is exactly the case when alpha blending is used (especially for models of levels of detail close to the virtual camera).

In practice, noise blending is often used for vegetation, which is essentially a reflection of random variables in nature [8].

10. Geomorphing

Geomorphing is another method of smooth transition between levels of detail. Its essence is to approximate a 3D model to create intermediate transition states.

The main operations of geomorphing are vertex splitting (new vertices are added to the model as the quality of the model increases) and edge collapse (some vertices are removed as the quality of the model decreases) [9].

During the transition, not only the number of vertices is changed, but also their position and other additional attributes: normal, color, texture coordinates, etc. (see Figure 4), which prevents popping.

Fig. 4. The principle of geomorphing, going from a low quality model to a high quality model

Most vertex attributes are linearly interpolated. Normals, which are 3-dimensional vectors with a length of one unit, are modified using directional interpolation to preserve their length.

When implementing geomorphing in software, it is important to consider the situation when vertex splitting or edge convolution is performed during an existing transition. In other words, vertices should be able to perform animations of transitions that overlap in time.

One way to optimize the geomorphing method is to perform a smooth transition only for the visible part of the model (view-dependent LOD control) [10]. This technique is useful for large models (both in size and number of vertices). The entire mesh is divided into parts (clusters) that have their own memory buffers. The decision of which cluster to draw on the screen is based on an optimization technique called frustum culling.

The term "frustum" refers to a pyramid-shaped viewport that covers the visible area of a 3D scene [11]. This truncated pyramid is obtained by projecting the camera perspective onto the near and far planes. Frustum culling selectively renders only those objects that fall within this truncated area, discarding those that are out of view.

By eliminating invisible objects early in the visualization pipeline, we significantly reduce the computational load and increase overall performance.

During the object removal process, each model in the scene is checked on a slice to determine if it intersects or lies outside the field of view. Several algorithms are used to quickly determine if an object is potentially visible, such as checking with spheres or Axis-Aligned Bounding Boxes (AABB). If an object is completely invisible, it is excluded from the rendering process, eliminating unnecessary geometry, lighting, and shadow calculations.

This optimization is especially useful in scenes with a large number of objects, allowing you to focus rendering resources on rendering only the models that are visible to the camera.

The main difficulty in dividing the model into clusters is ensuring a consistent and synchronized transition for vertices that are on the boundary of two clusters. If this implementation detail is overlooked, there may be breaks along the edges of the mesh parts, causing a "popping" effect in some cases.

The easiest way to implement this is to have the transition performed entirely on the CPU. A list of vertices that perform a smooth transition can be stored in RAM, and their attributes need to be updated every frame. However, it should be noted that this approach is not optimal for a large number of vertices, since after updating the data in CPU memory, it must be sent to the GPU, which can cause some delay [12].

Another approach is to store the high and low quality models in GPU memory. For a smooth transition, a weight parameter (from zero to one) is used to interpolate vertex parameters. Models of different quality can be preloaded with priority. Obviously, this method requires more video memory, but it significantly reduces the amount of data the CPU sends to the graphics card for each frame.

The advantage of the geomorphing method is that only one model is drawn at a time when switching between levels of detail. Also, a smooth change in the model is less noticeable to the viewer at any distance.

The disadvantages of this method are the complexity of the implementation and the need to pay special attention to the placement of the 3D model in memory. Thus, geomorphing becomes the key factor that determines how 3D rendering is performed.

11. Software implementation

The program for the study is developed using the C# programming language, the .NET 6 platform, and the MonoGame video game development framework (DirectX-based drawing).

Each of the studied smooth transition methods is represented by a separate class implementing the common ILodTransition interface. Each class receives models of detail levels, the degree of transition, the object transformation matrix, and a reference to the graphics pipeline for drawing.

Smooth transitions between detail levels are performed depending on the distance to the camera.

Alpha blending is implemented using a separate step for translucent objects and double-pass rendering. Because the last step of the Output-Merger (OM) graphics pipeline is rendering, all pixels with a depth value less than the depth of the z-buffer are not rendered at all. For this reason, semitransparent objects are usually drawn separately from opaque objects and then added to the final image with depth.

Double-pass rendering helps reduce the popping effect on certain models. The depth buffer does not work correctly when drawing translucent polygons of the same model, and sometimes transparent triangles can be seen through each other — resulting in regions of the model having different alpha values. When mixing levels of detail, it is important that all polygons of both models have the same transparency level, which is equal to one. If the transparency is less than one, the models are transparent. If the transparency is greater than one, the RGB color is usually different from the original model color, which is very noticeable to the user.

Therefore, each level of detail is drawn twice during the transition (a total of 4 draw calls). The first draw should only capture the depth value. Any changes in the RGB channels should be ignored. At this stage, a very simplified pixel shader is used that returns a simple color. This is done to avoid wasting time calculating the light and color of the pixels. In the second stage of drawing the model, the values of the RGBA channels are recorded based on the mixing factor (formula 4.1). Before that, you must disable writing to the z-buffer and leave it read-only. Before drawing the next model, the depth buffer must be cleared, otherwise the pixels of the second model will be discarded due to the depth remaining in memory:

```
private void DrawTransparentModelDoublePass(
 LodLevel lod, GraphicsDevice graphicsDevice, float alpha)
{
 graphicsDevice.Clear(
  ClearOptions.DepthBuffer, Color.Transparent, 1f, 0);
  graphicsDevice.DepthStencilState = DepthStencilState.
Default;
 graphicsDevice.BlendState = this.blendDepthOnly;
 this.mainMaterial.AlphaPass.Apply();
 foreach (var part in lod.Mesh.MeshParts) {
  graphicsDevice.SetVertexBuffer(part.VertexBuffer);
  graphicsDevice.Indices = part.IndexBuffer;
  graphicsDevice.DrawIndexedPrimitives(
   PrimitiveType.TriangleList,
   part.VertexOffset, part.StartIndex, part.PrimitiveCount);
 }
 using var bs = new BlendState {
  ColorSourceBlend = Blend.BlendFactor,
  ColorDestinationBlend = Blend.InverseBlendFactor,
  AlphaSourceBlend = Blend.BlendFactor,
  AlphaDestinationBlend = Blend.One,
  BlendFactor = new Color(alpha, alpha, alpha, alpha),
 };
 graphicsDevice.BlendState = bs;
graphicsDevice.DepthStencilState=DepthStencilState.Depth
Read;
 this.mainMaterial.MainPass.Apply();
 foreach (var part in lod.Mesh.MeshParts) {
  graphicsDevice.SetVertexBuffer(part.VertexBuffer);
  graphicsDevice.Indices = part.IndexBuffer;
  graphicsDevice.DrawIndexedPrimitives(
   PrimitiveType.TriangleList, part.VertexOffset,
   part.StartIndex, part.PrimitiveCount);
 }
}
```

The noise-based smooth transition implementation uses the built-in HLSL clip function to discard a given pixel from further processing. Such a pixel is not written to either the color texture or the depth buffer. This greatly simplifies image rendering, since you only need to draw models once, and you don't need a separate step for transparent objects. The rules by which each pixel is gradually cut off or appears are loaded as a 16x16 black and white texture.

Smooth transition based on geomorphing is implemented on the GPU. This helps to reduce the memory bus load, since the pre-prepared smooth transition mesh is loaded into video memory only once, and further interpolation between vertex attributes is performed in the vertex shader using the progress variable:

```
MainVertexShaderOutput GeomorphVS(
 in GeomorphVertexShaderInput input) {
    MainVertexShaderOutput output;
    float3 avgPos = lerp(
     input.StartPosition, input.EndPosition, Progress);
    float3 normal = lerp(
     input.StartNormal, input.EndNormal, Progress);
    output.Position = mul(float4(avgPos, 1.0),
                WorldViewProjection);
    output.Normal = normalize(normal);
    return output;
 }
```

The geomorphic feature set is built by finding the closest vertices between high- and low-quality models. Since 3D models often have vertices with the same positions but different additional attributes (normal, UV coordinates, etc.), a simple distance between vertex coordinates is not sufficient: the distance metric must also take into account normal vectors. If you ignore this property, triangles may have incorrect illumination and color during the transition, which negatively affects the smoothness of the transition.

The geomorphic mesh is stored in the cache using the Least Recently Used (LRU) strategy. When a transition is generated based on two models, it is added to the cache,

which allows it to be reused both during the next drawing call and during the same frame.

Creating a mesh is a slow operation for models with a large number of vertices. The slowest part is the calculation of the nearest vertices. Unfortunately, even with the use of parallelization, the delay in generation is quite noticeable. In practice, it is suggested to pre-calculate all the nearest vertices at the compilation stage of the program and just read them from the file. Another method is to create the model in the background thread and display it as soon as it is ready. In the software implementation under study, the delay is not critical, but it is taken into account when conducting experiments.

Analyzing smooth transition metrics

Let's measure the performance of discrete and smooth transition algorithms. The computer on which the measurements are performed has the following characteristics:

– Windows 10;
– Intel(R) Core(TM) i5-8250U 1.60GHz CPU;
– 8 GB OF RAM;
– NVIDIA GeForce MX250 GPU;
– 2 GB VRAM.

The resolution of the drawing window is 800x600 pixels.

The average time for drawing 3D space as a function of the number of models is shown in Figure 5.

It can be seen that the slowest algorithm for a relatively small number of objects (50-500 models) is alpha blending. However, the running time of this method is linear over the range studied, making it the fastest method for smooth transitions when visualizing 1000-2000 models.

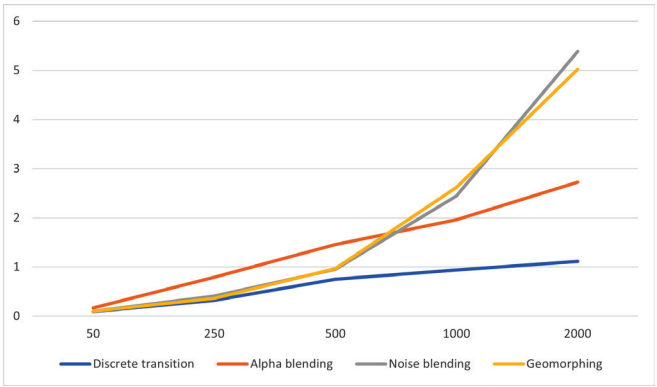

Fig. 5. Comparison of drawing times for smooth transition ethods

Geomorphing and noise blending have approximately the same runtime, which increases exponentially with the number of objects. Using the Windows Task Manager, running these methods on a large amount of data will load the GPU to 100% and the CPU to about 8%. It can be assumed that for a given hardware and software configuration, the GPU is the bottleneck for these methods.

The discrete transition between detail levels is the fastest. Let's analyze the magnitude of the popping effect for the studied algorithms for transitioning between levels of detail (see Figure 6).

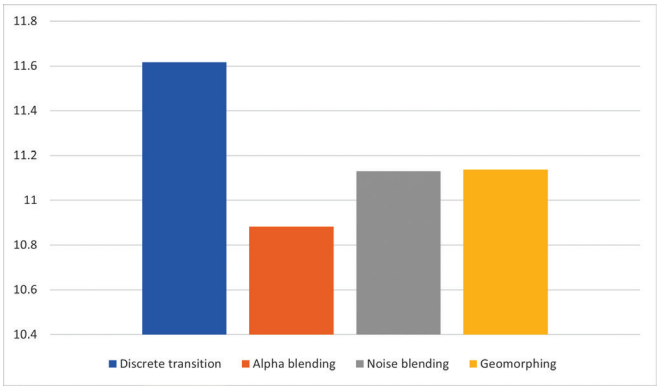

Fig. 6. Comparison of Popping Effect Levels

The figure shows that for the Stanford rabbit model under study, alpha blending is the best method for smooth transition. Noise-based transition and geomorphing have approximately the same amount of popping. It should be noted, however, that all of the algorithms studied are significantly smoother than discrete transitions.

Based on the results of the study, the following conclusions can be drawn about the feasibility of using smooth transition methods for the 3D model under study:

a) Alpha blending, although it gives the best visual appearance, is the slowest method in 3D spaces with small and medium number of objects. Alpha blending is the most efficient transition method when you need to display a transition for many objects at once;

b) Noise-based transition has an average level of performance, does not require much computing power for preparation, but is acceptable for drawing a limited number of models at the same time;

c) Geomorphing has about the same performance as noise-based blending, but it has one drawback: it requires the creation of a 3D mesh for a smooth transition.

Further research

The following areas of activity can be identified for further research on smooth transition methods:

– to study the work of the algorithms on a larger number of 3D models of different categories (architecture, landscape, vegetation, transportation, interiors, people, etc.);

– determine the effectiveness of each algorithm as a function of distance from the viewer;

– extend the metrics for evaluating the effect of popping (distances in CIELAB color space, graphical representation of popping, consideration of the statistical significance of the measurement);

– explore the possibility of automatically recommending transition methods depending on the type of model and the number of objects in 3D space to obtain the most effective values of the factors under study;

– to investigate the use of additional visual effects in combination with levels of detail (textures, normals, reflections, backlighting, ray tracing, etc.).

Conclusions

During the research internship, the students studied methods for smooth transitions between levels of detail (LOD) for three-dimensional graphics.

The focus was on finding a balance between performance and good-looking transitions. In particular, the research focused on eliminating the "popping" effect that disrupts visual consistency during transitions between LODs. A comprehensive analysis of existing methods and algorithms provided valuable information on strategies to prevent and minimize this effect.

It was found that alpha blending is the easiest method to implement for smooth transitions and reduces the popping effect the most, but is the slowest for a small number of models.

Noise blending has an average level of performance and a sufficient level of visual consistency in the image. This method also allows the most creative freedom in rendering objects and does not require any model preparation.

Geomorphing has the same performance and image consistency characteristics, but it is relatively difficult to implement and requires extensive calculations to prepare a 3D model for blending.

In practice, the results of this study are expected to help select algorithms that offer an optimal balance between computational efficiency and good looks, taking into account different circumstances. As 3D visualization continues to grow in popularity in a variety of applications, this work remains relevant and provides insights that can be used in future developments in the field.

References

[1] Jim Frost. Root Mean Square Error (RMSE). Statistics by Jim. URL: https://statisticsbyjim.com/regression/ root-mean-square-error-rmse/

[2] Anton L. Fuhrmann, Eike Umlauf, Stephan Mantler. Extreme Model Simplification for Forest Rendering. ResearchGate. URL: https://www.researchgate.net/ publication/221314842_ Extreme_Model_Simplification_ for_Forest_Rendering

[3] State of the Art in Procedural Noise Functions. [A. Lagae, S. Lefebvre, R. Cook, DeRose, G. Drettakis, D.S. Ebert, J.P. Lewis, K. Perlin, M. Zwicker]. URL: https://graphics.cs.kuleuven.be/publications/LLCDDELPZ10STARPNF/

[4] Noise Functions. Ronja's Shader Tutorials. URL: https://www.ronja-tutorials.com/noise.html

[5] The Importance of Dithering Technique Revisited with Biomedical Images – A Survey. [Liu Yue, P. Ganesan, B.S. Sathish, C. Manikandan, A. Niranjan, V. Elamaran, Ahmed Faeq Hussein]. ResearchGate. URL: https://www.researchgate.net/publication/329763540_The_Importance_of_Dithering_Technique_Revisited_With_Biomedical_Images-A_Survey

[6] Nithin Pranesh. Smoother LOD Transitions in Cesium for Unreal with Dithered Opacity Masking. 20.10.2022. Cesium. URL: https://cesium.com/blog/ 2022/10/20/smoother-lod-transitions-in-cesium-for-unreal/

[7] Transparency (or Translucency) Rendering. Nvidia Developer. URL: https://developer.nvidia.com/content/ transparency-or-translucency-rendering

[8] Benny Onrust, Rafael Bidarr, Robert Rooseboom, Johan van de Koppel. Procedural generation and interactive web visualization of natural environments. The 20th International Conference. ResearchGate. URL: https://www.researchgate.net/publication/ 300490331_Procedural_generation_and_interactive_ web_visualization_of_natural_environments

[9] Hugues Hoppe. Smooth View-Dependent Level-of-Detail Control and its Application to Terrain Rendering. Microsoft Research. URL: https://hhoppe.com/svdlod.pdf

[10] Pedro V. Sander, Jason L. Mitchell. Progressive Buffers: View-dependent Geometry and Texture LOD Rendering. Advanced Real-Time Rendering in 3D Graphics and Games. URL: https://advances. realtimerendering.com/s2006/ Chapter1-Out-of-Core_ Rendering_of_Large_Meshes_with_Progressive_Buffers.pdf

[11] Eun-Seok Lee, Byeong-Seok Shin. Vertex Chunk-Based Object Culling Method for Rreal-Time Rendering in Metaverse. 09.07.2023. MDPI. URL: https://www.mdpi.com/2079-9292/12/12/2601

[12] Data Transfer Matters for GPU Computing. [Yusuke Fujii, Takuya Azumi, Nobuhiko Nishio, Shinpei Kato, Masato Edahiro]. ResearchGate. URL: https://www.researchgate.net/publication/269197419_Data_Transfer_Matters_for_GPU_Computing

The article was delivered to editorial stuff on the 26.05.2023