Improving computer-generated images – methods for realistic depiction

Boyana Nedkova Ivanova, Rumen Rusev

Boyana Nedkova Ivanova, University of Ruse, Bulgaria
Rumen Rusev, University of Ruse, Bulgaria

Abstract.

What is a realistic image? The realistic image provides information for the object that is close to the physical/ real object.
One of the most challenging problems in computer graphics is to generate images that appear realistic. Predicting human perception of visual realism has many applications - simulations, design, entertainment, advertisement, and education. Presenting objects with high level of visual realism will help people to understand in details more quantity of information. There are some techniques that are applied to objects and scenes to make them look more realistic, to make them look 3D (Three dimensional).
The purpose of this paper is to present a systematic review of existing techniques for making objects and scenes appear realistic - which are the different stages of modelling a scene, which are the existing methods to make a "real" object.
In this paper, you will find a classification of software tools that are used for modelling realistic images. Visual artists in entertainment, automotive, architecture, advertising, and web design industries utilize these tools to create lifelike scenes for use in media or client engagement purposes.

Keywords: images, techniques, methods, rendering, realism, object, software

1. Introduction

Realism in Computer Graphics is never ending quest. In a paper written by Roy Hall, is discussed that there are two fundamental reasons for continuing to seek realism – simulation and emulation. Simulation tries to accurately model physical behaviour while emulation or illusion tries to provide the impression of realism by empirically approximating what we observe. [1]
Nowadays, most industries use realistic images to present their services, products. People rely on imagery to share information, learn about new ideas and educate themselves on things that interest them. The key is to use high-quality images, photos and illustrations.
The old saying, “a picture is worth a thousand words,” comes true when an image helps tell your story. Whether you sell a complex idea, technical product or a simple service that requires almost no explanation, imagery helps explain why your company’s offering is the best choice.
In Computer Graphics exists techniques that make the objects in a scene look realistic.
The aim of this paper is to describe techniques such as depth buffering, lightning, fogging, anti-alising and texture mapping – some of the most popular methods for improving image quality. It will be presented a systematic review of software tools using these methods in practise.

Figure 1: Scene with no rendering; Scene with applied render techniques

2. Methods

2.1. Z-buffer algorithm:

The wide availability of Z-buffer has sparked an explosion in the number of applications of the algorithm whose origins lie in hidden surface elimination. The full screen Z-buffer algorithm has become standard in Computer Graphics. This method is developed by Cutmull. It is an image-space approach (implemented in screen coordinate system). The basic idea is to test the Z-depth of each surface to determine the closest visible surface.

The depth buffer is used to determine which portions of objects are visible within the scene. When two objects cover the same *x* and *y* positions but have different *z* values, the depth buffer ensures that only the closer object is visible.

The Z-buffer algorithm is best viewed as operating in a three-dimensional screen space. Each pixel is associated with a two dimensional screen coordinate (xs, ys) together with a depth of Z-value interpolated from the vertex eye space z-depth at the same time as shading values are interpolated and rasterization is performed. The Z-buffer is initialized to the depth of the far clipping plane. For each pixel, it is compared its interpolated depth with the depth already stored in the Z-buffer (xs, ys). If the value is less than the stored value then the pixel is nearer the viewer than previously encountered pixels, in which case the current shading value is written to the screen memory and the current depth placed in the Z-buffer.

The Z-buffer algorithm imposes no constraints on database organization, and, in its simplest form can be driven on a polygon-by-polygon basis, with polygons being presented to the rendering process in any order. Polygons appear on the screen in the order in which they are extracted from database. Clearly some polygons will appear then disappear and this highlights a major inefficiency of the algorithm, in that shading calculations are performed on (hidden) pixels that are subsequently overwritten. Because the algorithm deals with one polygon at a time, it imposes no constraints on scene complexity and this is another reason for its popularity that has seen a steady increase in modelling resolution or object complexity.

The *z* value specifies the distance from the fragment to the plane of the eye. The relationship between distance and *z* is linear in an orthographic projection, but not in a perspective one. To achieve the best depth buffer precision, the near plane should be moved as far from the eye as possible without touching the object of interest (which would cause part or all of it to be clipped away). The position of the near clipping plane has no effect on the projection of the *x* and *y* coordinates, so moving it has only a minimal effect on the image. As a result, readjusting the near plane dynamically shouldn't cause noticeable artifacts while animating. On

the other hand, allowing the near clip plane to be closer to the eye than to the object will result in loss of depth buffer precision.[3]

In this method each surface is processed separately one pixel position at a time across the surface. The depth values for a pixel are compared and the closest smallestZ surface determines the color to be displayed in the frame buffer.

It is applied very efficiently on surfaces of polygon. Surfaces can be processed in any order. To override the closer polygons from the far ones, two buffers named **frame buffer** and **depth buffer,** are used.

Depth buffer is used to store depth values for x, y position, as surfaces are processed 0≤depth≤1. The **frame buffer** is used to store the intensity value of color value at each position x, y.

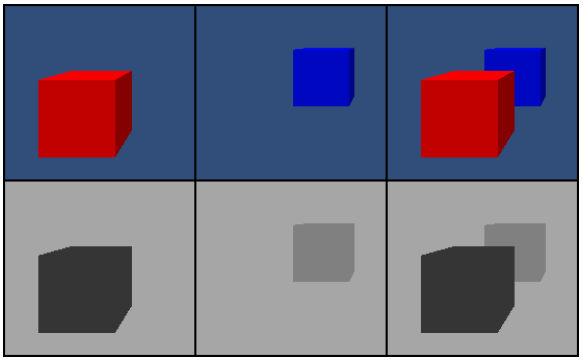
The z-coordinates are usually normalized to the range [0, 1]. The 0 value for z-coordinate indicates back clipping pane and 1 value for z-coordinates indicates front clipping pane.

Algorithm

Step-1 − Set the buffer values −
Depthbuffer x,y = 0
Framebuffer x,y = background color
Step-2 − Process each polygon One at a time
For each projected x,y pixel position of a polygon, calculate depth z.
If Z > depthbuffer x,y
Compute surface color,
set depthbuffer x,y = z,
framebuffer x,y = surfacecolor x,y

The advantages of this method are that it is simple to use; it can be implemented easily in object or image space; it can be executed quickly, even with many polygons. And the disadvantages are that it takes a lot of memory and it can't do transparent surfaces without additional code.

Figure 2: Example with applied Z-buffer algorithm



Source: jakobknudsen.wordpress.com

A survey published in 2001, describes **some of the key applications of the Z-buffer** algorithm from the fields of rendering, modelling and vision in a common notation in order to help users make better use of this resource. In a first place, it is **image compositing** - if depth and viewpoint conditions are respected, the various objects may be rendered in separate pairs of frame and Z-buffers and then combined into the final image, eliminating hidden surfaces. The

second application is **shadow mapping** - the most popular alternative use of the Z-buffer algorithm is the fast generation of shadows from spotlights or parallel projectors. This shadow generation technique was first introduced by Williams in 1978 and its variations are still widely used in rendering software and real-time applications, like computer games. The main advantages of Z-buffer-based shadows are the simplicity and generality of the Generating shadows with the Z-buffer algorithm, its speed as well as the ability to produce soft shadows. The third application is **CSG (Computer Solid Geometry) rendering** – it is a very common method of modelling complex 3D objects from simple primitives by performing Boolean operations on their volumes. Z-buffer methods utilise the standard Z-buffer algorithm implemented in conventional graphics hardware in order to render a CSG hierarchy using multiple passes of clipping (stencil test) and depth sorting (depth test) operations. The technique uses one Z-buffer (surface Z-buffer) to store the visible surface of each primitive and another one (output Z-buffer) to compose in correct depth-order the partial results of the surface Z-buffers. A stencil buffer is also necessary for the clipping of the primitives in the surface Z-buffers. Each time a primitive is compared with the surface Z-buffer, the stencil test is configured so that the stencil buffer holds the number of surfaces in front of the Z-buffer already stored. Another application is **object reconstruction** - this application is one of the rare cases where rendering is not used just for visualization, but actively participates in the solution of a computer vision problem. In this procedure, object fragments have to be tested one against another for complementary matching in order to be glued together later. The last described application of Z-buffer is **symmetry detection** - symmetry of three-dimensional objects is a valuable property that is of use in a wide range of applications, e.g. object recognition and reconstruction from views, or data compression. Reflectional and rotational symmetry of a 3D object can be measured using a modified version of the traditional depth buffer, where all depth values are stored, instead of the minimum ones. For every pixel (x, y), the modified depth buffer holds all corresponding depth values, in ascending order.[4]

2.2. Lighting and Shading:

Lighting is one of the most important considerations for realistic 3D graphics. The goal is to simulate light sources and the way that the light that they emit interacts with objects in the scene. What the human eye (or virtual camera) sees is a result of light coming off of an object or other light source and striking receptors in the eye. In order to understand and model this process, it is necessary to understand different light sources and the ways that different materials reflect those light sources.

The light in an environment comes from a light source such as a lamp or the sun. In fact, a lamp and the sun are examples of **two essentially different kinds of light source: a point light and a directional light**. A point light source is located at a point in 3D space, and it emits light in all directions from that point. For a directional light, all the light comes from the same direction, so that the rays of light are parallel. The sun is considered to be a directional light source since it is so far away that light rays from the sun are essentially parallel when they get to the Earth.

Computer graphicc technology has ability to produce a realistic looking three dimensional object on a two dimensional output device like computer screen or printed paper. This ability is achieved by rendering methods in which shading is applied for more realistic rendering. Shading uses to compute the amount and color of the light that emitted from every point of the surface.

The resulting shaded image can be basically depended on the following entities:

- ✓ The light source. The intensity, color, shape, direction and distance of the light source are considered, and it also can be point source or a large source, such as a window or a light fixture.2.
- ✓ The surface of the object. The object can be from very shiny to from smooth to rough, and from bright to dark. It can have several colors can be opaque, transparent or translucent.3.
- ✓ The environment. Objects seen in empty space, without any background to reflect light on them, look harsh. Imagine a spaceship in deep space, away from any reflecting planets. Those parts of the ship illuminated by direct starlight are very bright, while parts that are in the shade are completely dark. The result is the ship mostly looked in black and white, with few grays or colors. A realistic shading model should therefore consider light reflection from other objects and from nearby walls.

The object is illuminated by light which is to follow rays of light from light-emitting on the object surfaces called light sources.

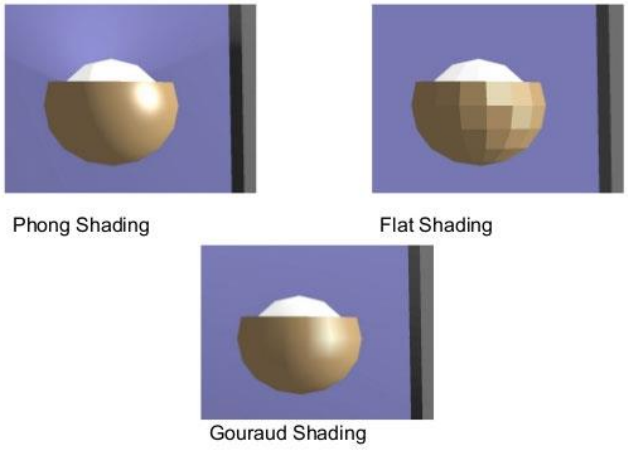
- ➢ ***Ambient light*** - this light is non-directional light source in which the light is come from all direction. The intensity is not affected by anything else such as position or orientation.
- ➢ ***Point light*** - the light source that does not give equal amount of light in all direction in which the object will gain brighter when it is closer to the light. The intensity of the light source is depended on the distance. The angle function is affected to light ray. It is characterized by color, intensity, location and fall off function.
- ➢ ***Directional light*** - this kind of light source is produced a light source from infinite distance from the scene. All of the light rays illuminate in a form of a single parallel direction and with equal intensity everywhere. It is characterized by color, intensity and direction.
- ➢ ***Spotlight*** - the light is radiated in a cone with more light in the center of the cone. This light is attached by the primary axis of direction with a restricted on it. It is characterized as a point light, an axis of direction, a radius about that axis and possibly a radial falloff function.[5]

Shading Models:

- ➢ *Flat Shading* (per polygon) - this is the most simple and efficient way to specify color for an object. It defines a single color for a face. Implementations of it may vary, but the main idea is that we use only one surface normal per polygon. The color itself is uniform (unchanging) on that polygon.
- ➢ *Gouraud Shading* (per vertex) - this was invented as an improvement to allow for more smooth transitions of the color on round objects. Main idea is that there is a different normal per vertex and the color is calculated in the vertex shader. That color is then interpolated over the polygon. Because there are less vertices then there are fragments, then calculating the color per vertex and interpolating it, is more efficient than calculating it per fragment. This approach handles badly materials that have a specular reflection. This reflection might occur inside the polygon, but not on any of the vertices. This shading would not show it, if it does not happen to be on the vertex.
- ➢ *Phong Shading* (per fragment) - this was another improvement in order to account for the specular reflection. Main idea is that the normal from the vertices is interpolated. Color is calculated per fragment, taking into account the interpolated normal. For

an approximation of a sphere, this is quite ideal, because the interpolated normals would be exactly those that a perfect sphere would have. Because the color is calculated based on the normals, it will be calculated as if it were a perfect sphere.

Figure 3: Examples with applied shading models



Source: slideshare.net

Lighting Models:

➢ **Lambert Lighting Model –** it is given a direction from where the light is coming from and some sort of a surface. Diffuse surfaces have the property to diffuse reflected light around. Light will enter the surface, bounce around there and then exit in a random direction. Or you can think that an atom will absorb the photon and after some time emits another photon to a different direction. Because of the diffusion of light, it does not matter at which angle we look at the material. All that matters is on which angle the light actually reaches the material.

This is because the amount of light reaching one surface unit will be higher if the light is coming from a more perpendicular angle. If the light is coming from a grazing angle, then the same amount of light will cover a much larger area and thus a surface unit will receive less light. The amount of light received and thus reflected is directly related to the angle between the surface and the light direction.

This is the place where will need a surface normal. The cosine of the angle between the surface normal and direction towards light will directly give us the illumination percentage. If the cosine is 1, there is a 90-degree angle, and a surface unit will receive 100% light. If the angle is 60°, the cosine will be 0.5 and a surface unit will receive 50% light. If the angle is 45°, we have around 71% (as can be observed from the illustration above).

In order to determine the actual color, the most important things which should be known are what is the color of the light emitted from the light source and what colors are reflected from the surface.

In computer graphics the colors are defined by three channels: red, green and blue and respectively 3 terms for each of those channels for both the light source and the surface material. The final color computed in the fragment shader would be like this:

Red=MdiffuseRLdiffuseR·max(normal·light,0)red=MdiffuseR·LdiffuseR·max(normal·light,0)
Green=MdiffuseG·LdiffuseG·max(normal·light,0)green=MdiffuseG·LdiffuseG·max(normal·light,0)
Blue=MdiffuseB·LdiffuseB·max(normal·light,0)blue=MdiffuseB·LdiffuseB·max(normal·light,0)

Here the value MdiffuseR would be the percentage of red light this material will reflect. The value LdiffuseR will be the percentage of red light the light source emits.

In reality, there are very few surfaces that are almost completely diffusely reflective. Examples would include the surface of the Moon, chalk, matte paper.

➢ **Ambient Light** - the non-illuminated sides of it are totally in darkness. This is not the case, there is almost always some light coming from every direction. That is because light will reflect from nearby surfaces, bounce around and reaches the area not directly illuminated.

This is called *indirect illumination* and is one of the things that global illumination techniques try to do accurately, which may be computationally quite expensive. In simple model it is added an ambient term to the reflected intensity. That way the non-illuminated areas will not be totally black. Although this means that the other parts will become a bit more illuminated also. This might cause the intensity to be above 1. Because of that the values are usually clamped to be in the range [0, 1], [0, 1]. The clamp operation makes the values above maximum be the maximum, values below minimum to be the minimum. Often times if you assign a value higher than 1 to the pixel's color in code, the output will still be the same as if it were 1.

Red=MambientR·LambientR+MdiffuseR·LdiffuseR·max(normal·light,0)
Green=MambientG·LambientG+MdiffuseG·LdiffuseG·max(normal·light,0)
Blue=MambientB·LambientB+MdiffuseB·LdiffuseB·max(normal·light,0)

➢ **Phong Lighting Model** - there are very few surfaces that are only diffusely reflective. Most surfaces have an amount of specular reflection. This means that some part of the incoming light will reflect in the reflected direction. If the viewer is looking from that direction, the area of the surface will seem lighter than the rest of the surface. Some part of the light is absorbed by the surface and reflected in an arbitrary angle. Another part is directly reflected from the surface. If the directly reflected part would be 100%, it is called a perfect mirror.

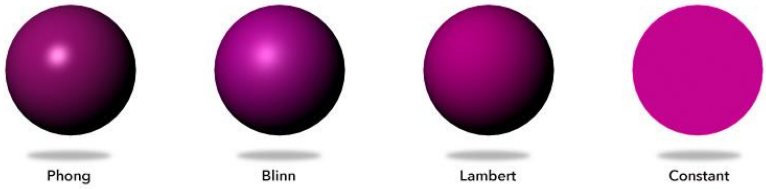
Phong's lighting model. It includes the ambient term, Lambertian / diffuse term and the Phong's specular term. There are 2 colors for each of those terms: color of the light and the material. So there are 6 colors. There are also 4 vectors that indicate directions towards: surface normal; light; reflected light; viewer. Those vectors should all be normalized

Red=MambR·LambR+MdiffR·LdiffR·max(normal·light,0)+MspecR·LspecR·(max(refl·viewer,0))
Green=MambG·LambG+MdiffG·LdiffG·max(normal·light,0)+MspecG·LspecG·(max(refl·viewer,0))shininess
Blue=MambB·LambB+MdiffB·LdiffB·max(normal·light,0)+MspecB·LspecB·(max(refl·viewer,0))shininess [6]

➢ **Blinn-Phong** - Phong lighting is a great and very efficient approximation of lighting, but its specular reflections break down in certain conditions, specifically when the shininess property is low resulting in a large (rough) specular area. In 1977 the Blinn-Phong shading model was introduced by James F. Blinn as an extension to the Phong shading. The Blinn-Phong model is largely similar but approaches the specular model slightly different which as a result overcomes our problem. Instead of relying on a reflection vector it is using a halfway vector that is a unit vector exactly halfway between the view direction and the light direction. The closer this halfway vector aligns with the surface's normal vector, the higher the specular contribution. When the view direction is perfectly aligned with the (now imaginary) reflection vector, the halfway vector aligns perfectly with the normal vector. The closer the view direction is to the original reflection direction, the stronger the specular highlight. Another subtle difference between Phong and Blinn-Phong shading is that the angle between the halfway vector and the surface normal is often shorter than the angle between the view and reflection

vector. As a result, to get visuals similar to Phong shading the specular shininess exponent has to be set a bit higher.

Figure 4: Examples with applied lighting models



Source: docs.viromedia.com

2.3. Alpha Blending:

Alpha compositing is the process of **combining an image with a background to create the appearance of partial transparency**. The combining operation takes advantage of an alpha channel, which basically determines how much of a source pixel's color information covers a destination pixel's color information.

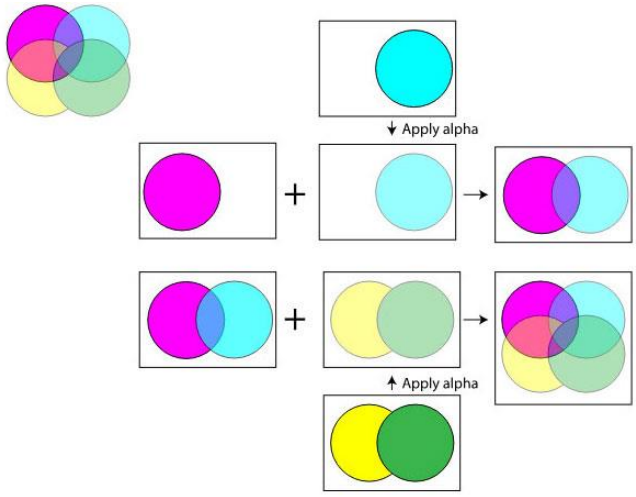
The alpha idea been used to composite billions of pixels (if not more) to create images for print, video, film, and probably every other application of computer graphics. Alpha is obviously incredibly useful for compositing images. In addition to the red, green, and blue components of each color, there is an additional optional fourth component, referred to as the color's "alpha." Alpha means transparency and is particularly useful when you want to draw elements that appear partially see-through on top of one another. The alpha values for an image are sometimes referred to collectively as the "alpha channel" of an image.

Alpha compositing uses the alpha values, or channel (bit mask) to represent the coverage of each pixel. The alpha channel is often said to represent the 'opacity'. This coverage information is used to control the compositing of colors. In other words, alpha compositing (also known as alpha blending) is the process of layering multiple images, with the alpha value for a pixel in a given layer indicating what fraction of the colors from lower layers are seen through the color at the given level. Simple alpha compositing, composites each object onto the background image using a simplistic formula that has the effect of overlaying the object over the background. Where the objects overlap and coverage is not complete the color of the background may show through the object that has just been rendered.

The alpha channel is a color component that represents the degree of transparency or opacity of a color i.e., the red, green and blue channels. It is used to determine how a pixel is rendered when blended with another. It controls the transparency or opacity of a color.

When a color (source) is blended with another color (background), e.g., when an image is overlaid onto another image, the alpha value of the source color is used to determine the resulting color. If the alpha value is opaque, the source color overwrites the destination color; if transparent, the source color is invisible, allowing the background color to show through. If the value is in between, the resulting color has a varying degree of transparency/opacity, which creates a translucent effect. Typically, the higher the value of an alpha channel sample, the more opaque that pixel is (some file formats work the other way around, though, so calling it the transparency channel is fine). The alpha channel is typically represented by the letter A (ex.,"RGBA"). The alpha channel is used primarily in alpha blending and alpha compositing. [7]

Figure 5: Examples with applied alpha blending



Source: cabanier.github.io

2.4. Antialiasing:

Antialiasing is a technique used in computer graphics to remove the aliasing effect. The aliasing effect is the appearance of jagged edges or "jaggies" in a rasterized image (an image rendered using pixels). The problem of jagged edges technically occurs due to distortion of the image when scan conversion is done with sampling at a low frequency, which is also known as Undersampling. Aliasing occurs when real-world objects which comprise of smooth, continuous curves are rasterized using pixels.

Methods of Antialiasing (AA):

- **Using high-resolution display:** One way to reduce aliasing effect and increase sampling rate is to simply display objects at a higher resolution. Using high resolution, the jaggies become so small that they become indistinguishable by the human eye. Hence, jagged edges get blurred out and edges appear smooth.

- **Post filtering (Supersampling):** in this method, we are increasing the sampling resolution by treating the screen as if it's made of a much more fine grid, due to which the effective pixel size is reduced. But the screen resolution remains the same. Now, intensity from each subpixel is calculated and average intensity of the pixel is found from the average of intensities of subpixels. Thus we do sampling at higher resolution and display the image at lower resolution or resolution of the screen, hence this technique is called supersampling. This method is also known as post filtration as this procedure is done after generating the rasterized image.

- **Practical applications:**

In gaming, SSAA (Supersample Antialiasing) or FSAA (full-scene antialiasing) is used to create best image quality. It is often called the pure AA and hence is very slow and has a very high computational cost. This technique was widely used in early days when better AA techniques were not available. Different modes of SSAA available are: 2X, 4X, 8X, etc.

denoting that sampling is done x times (more than) the current resolution. A better style of AA is MSAA (multisampling Antialiasing) which is a faster and approximate style of supersampling. Better and sophisticated supersampling techniques are developed by graphics card companies like CSAA by NVIDIA and CFAA by AMD [8]

- **Pre-filtering (Area Sampling):** in area sampling, pixel intensities are calculated proportional to areas of overlap of each pixel with objects to be displayed. Here pixel color is computed based on the overlap of scene's objects with a pixel area. **For example:** Suppose, a line passes through two pixels. The pixel covering bigger portion(90%) of line displays 90% intensity while less area(10%) covering pixel displays 10-15% intensity. If pixel area overlaps with different color areas, then the final pixel color is taken as an average of colors of the overlap area. This method is also known as pre-filtering as this procedure is done BEFORE generating the rasterized image. It's done using some graphics primitive algorithms.

- **Pixel phasing:** it's a technique to remove aliasing. Here pixel positions are shifted to nearly approximate positions near object geometry. Some systems allow the size of individual pixels to be adjusted for distributing intensities which is helpful in pixel phasing.

Figure 5: Examples with applied anti-aliasing

Source: www.photokaboom.com

2.4. Texture Mapping:

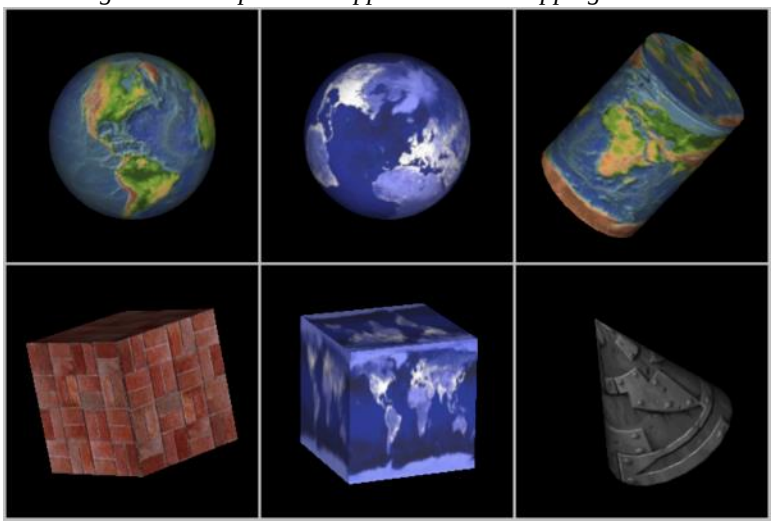
Rich visual detail in computer-generated imagery has always been a great challenge in computer graphics. Texture mapping, pioneered by Catmull in 1974, has been one of the most successful such techniques with a vast number of applications. A paper, published 2007, summarizes approaches of texture mapping:

- *Bump mapping* - Bump mapping is an approach to simulate complex surfaces by perturbing the normal vector of a surface on a per-pixel basis. This yields good results due to the fact that the main effect of surface irregularities on the perceived intensities is due to their effect on the surface normal.
- *Displacement Mapping* - displacement mapping actually changes the geometry of the object. Additional vertices are set onto the surfaces of the model and shifted to create the heights contained in the texture. This approach naturally allows self shadowing and self occlusion and also modifies the silhouette of the object. However, due to the fact that the object

that is rendered is actually a lot more complex, this approach does suffer from poor performance. Thus, displacement mapping is not so much a way to simulate complex objects, but an approach to reduce the memory requirements to store a complex model.

- *Parallax mapping* - it does not work by modifying the surface normals, but rather affects the way a texture is mapped to the surface. Thus, it is possible to use this approach in conjunction with bump mapping so the lighting is affected as well. From the regular texture position on the surface, the height displacement is calculated and transformed back onto the surface by multiplying it with the tangent of the angle between the texture axis and the visual axis. The texel at this new position is then used for texturing the pixel.
- *Relief mapping* - warps portions of the texture map, but this time using a raycaster. The raycaster calculates the intersection point with the bumpy surface using a binary search between the original intersection point of the unmodified surface and the ray's intersection with the surface offset by the maximum difference in the heightfield preceded by a linear search in order to avoid finding the first intersection (might happen if a search point is outside the height field surface but has already intersected the surface before) This intersection point contains the coordinate of the texel to use as well as the depth information (from the observer). To add self shadowing to relief textures, another ray has to be cast towards each light source to check for occlusion with other bumps. By properly refreshing the z-buffer with the modified z values, relief surfaces can even be correctly interpenetrated.[9]

Figure6: Examples with applied texture mapping



Source: http://math.hws.edu

3. Results

In this section, it is going to be presented a classification of software tools for rendering. The solutions that are included in this classification has the major features of best rendering tools:

- Surface modelling – users can work on 3D models and can align, modify and transform the geometry of objects.
- Easy render setup – users can create templates to make a complex scene that can be reused in future.
- Mesh modelling – allows users make 3D model that consists of polygons.
- Solid Technology – allows users to create complex objects; each body is presented as a single object;

.

Table 1: Rendering Software

Rendering Software	Version	Devices Supported
Blender	Free	Windows, Mac, Linux, Web-based
Autodesk Maya	Paid/ 1 620 dollars	Windows, Mac, Linux, Web-based
Aqsis	Free	Web-based
LuxeCoreRender	Free	Web- based
Mandelbulb 3D	Free	Windows, Mac, Linux
Kerkythea	Free	Windows, Mac
Wing 3D	Free	Windows, Mac, Linux
Appleseed	Free	Windows, Mac, Linux
Arnold	Free trial of 30 days/Paid	Windows, Mac, Linux, Web-based
Lumion	Free trial of 14 days/Paid	Windows, Mac

Blender is the free and open source 3D creation suite. It supports the entirety of the 3D pipeline—modeling, rigging, animation, simulation, rendering, compositing and motion tracking, even video editing and game creation. Advanced users employ Blender's API for Python scripting to customize the application and write specialized tools; often these are included in Blender's future releases. Blender is well suited to individuals and small studios who benefit from its unified pipeline and responsive development process. Blender is cross-platform and runs equally well on Linux, Windows, and Macintosh computers. Its interface uses OpenGL to provide a consistent experience. [10]

Maya is the premier application for creating compelling 3D digital content, including models, animation, visual effects, games, and simulations. Its main features are: creating models. Polygons, Non-Uniform Rational B-Splines (NURBS), and subdivision surfaces are different object types with different ways of modeling. Each has its own strengths, and different artists prefer working with different types, create polygons let you model a surface by building up and reshaping a number of simple surface facets; create NURBS and curving surfaces with high-level control; subdivision surfaces let you edit surfaces at a high level with minimum overhead data, while still letting you work with subsections of the surface as if they were made from polygons; add dynamics, fluids, and other simulated effects. Maya includes a comprehensive suite of tools for simulating real world effects such as fire, explosions, fluids, hair and fur, the physics of colliding objects, and more; add lighting, shading, and rendering.[11]

Aqsis is free 3D rendering software that adheres to the RenderMan interface standard defined by Pixar Animation Studios. This cross-platform solution consists of many components. Some of its features are: programmable shading, Motion Blur, High-quality filtering and texture filtering Render arbitrary user data for post-processing control , NURBS (Non-uniform rational basis spline), Dynamics and effects, Lighting and modeling, Pipeline, Shading and texturing, CSG (Constructive Solid Geometry)[12]

LuxCoreRender is the open source 3D rendering software that is built on physically based equations (models the flow of light in the real world) that helps in forming the transportation of light. This free 3D rendering software supports high-dynamic-range (HDR) rendering. The highlights of the tool are: Physically-based rendering, Biased and unbiased rendering, Render algorithm, Materials and textures, Lighting, light groups, and volumes. [13]

Mandelbulb 3D is the free 3D rendering software created for 3D fractal (never ending pattern) imaging. The app creates dozens of nonlinear equations into fractal objects that you will find surprising. Lighting, color, specularity, depth-of-field, shadow-and glow-effects all are included in the 3D rendering environment. It allows the user to have excellent control over the imaging effects. This cross-platform desktop computer graphics application is used for calculating and rendering complex fractals in three dimensions. The core function of this software is to provide fractal images. It also imports and exports some form of 3D data. [14]

Kerkythea is a freeware software that can produce high quality renders without spending a cent on software licensing. Kerkythea is using physically accurate materials and lights, aiming for the best quality rendering in the most efficient timeframe, with target to simplify the task of quality rendering by providing the necessary tools to automate scene setup, such as staging using the GL real-time viewer, material editor, general/render settings, editors, etc., under a common interface. [15]

Wings 3D is a 3D rendering solution, you will find a wide range of modeling tools, built-in AutoUV mapping facility, and customizable interface. The overview of its key features includes: Context sensitive interface, Configurable interface and hotkeys, Wide range of Selection and Mesh tools, UV mapper, Vertex Colors, Materials, and Lights. [16]

appleseed is the free and open source 3D rendering software that is designed for animation and visual effects. Even individual users and small studios can get a complete and fully open rendering package from this app.

There are different types of rendering modes of this app – single-pass rendering, multi-pass rendering, progressive rendering, time-limited progressive rendering, interactive rendering, scene editing during interactive rendering, spectral rendering, and RGB (red, green, and blue) rendering. [17]

Arnold is the advanced Monte Carlo ray tracing renderer (renders three-dimensional scenes) that is built for providing the feature-length animation and visual effects. It quickly and easily renders complex scenes to the artists. An efficient raytraced curve primitive makes Arnold the perfect choice for rendering fur and hair using very little memory. Accurate 3D motion blur correctly interacts with shadows, volumes, indirect lighting, reflection or refraction. The volumetric rendering system is based on proprietary importance sampling algorithms and can render effects such as smoke, clouds, fog, pyroclastic flow or fire. [18]

Lumion is the fast 3D rendering software that renders beautiful images and videos. Quickly the users can render breathtaking images related to residential buildings, landscapes, urban spaces, or interiors. This visualization software is apt for architects and designers. From all major CAD design software programs like Revit, ArchiCAD, and Sketchup, this app offers flawless importing. [19]

4. Conclusion

Rendering programs will complete the photo-realistic look of your scene. Here the light, colour and texture will be manipulated to a fine degree to animate your visual to its best. Materials will be accurately scanned and fabric textures replicated. Reflective surfaces such as glass, metal and mirror often lengthen the process.

As a conclusion, it is important to produce high-quality computer-generated images because of the benefits they provide: full customization, visualize changes, precision, reduction of costs, visual communication, and relevant and competitive products.

With 3D rendering software, the users can interact with their models from any angle and can input precise measurements of their 3D models. **Computer generated imagery (CGI)** can easily offer variations of the initial product, allowing brand visual consistency across a broad range of multimedia. From colour and lighting to different viewing angles, these variations come in fast and at a lower cost. CGI is a versatile technology that can be used to deliver content across multiple media platforms like photography, animated walkarounds, video, Augmented Reality, Virtual Reality, Mixed Reality.

References (TNR 14pt., bold)

(Book style - Author, year. *Title (in italics).* Publisher, location of publisher.**)**

[1] David F. Rogers, Rae A. Earnshow, 1990, *Computer Graphics Techniques: Theory and Practice*, Springer-Verlag.

[2] Chen, W. K. (1993). *Linear Networks and Systems*, Belmont, CA: Wadsworth, pp. 123-135.

[3] Watt A, Watt M, 1992, *Advanced Animation and Rendering Techniques*, Addison-Wesley

[4] Theoharis, Georgios Papaioannou, Evaggelia-Aggeliki Karabassi, (2001), *"The magic of Z-buffer: A survey"*, Department of Informatics, University of Athens, Panepistimioupolis, Ilisia, Athens, Greece

[5] Burhan Saleh, "Computer Graphics Fundamental: Lighting and Shading", Department of Computer Engineering Çukurova University Adana, Turkey

[6] Shading and Lighting – Available: https://cglearn.codelight.eu/pub/computer-graphics/shading-and-lighting#material-shading-models-1

[7] S.Maji, A.Nath, 2015, *"Scope and Issues in Alpha Compositing Technology"*, International Journal of Innovative Research in Advanced Engineering(IJORAE), issue 12, volume 2

[8] F. Liberatore, J. Longazo, S. Pettinati, D. Weise, "*Comparison of Anti-Aliasing Techniques of Real-Time Applications*", Department of Computer Science USC Viterbi School of Engineering Los Angeles, CA

[9] Stephan Pajer,2006, "*Modern Texture Mapping in Computer Graphics*"

[10] Blender official Site, Available: https://www.blender.org/

[11] Autodesk official Website, Available: https://knowledge.autodesk.com/support/maya

[12] Aqsis official website, Available: https://www.aqsis.org/documentation/user_manual/index.html

[13] LuxCoreRender official website, Available: https://luxcorerender.org/

[14] Mandelbulb 3D official website, Available: https://www.mandelbulb.com/

[15] Kelkythea official website, Available: https://www.kerkythea.net/cms/
[16] Wings 3D official website, Available: http://www.wings3d.com/
[17] Appleseed official website , Available: https://appleseedhq.net/
[18] Arnold official website, Available: https://www.arnoldrenderer.com/
[19] Lumion official website, Available: https://lumion.com/