COM6503
01/24

Making objects and scenes look less pristine in Computer Graphics

Prajwal Patil

I declare that this work is my own.

Author: Prajwal Patil, pmpatil1@sheffield.ac.uk

Contents

Introduction

Although there has been significant progress in Computer Graphics' ability to mimic reality, natural clarity of digital images frequently makes it difficult to create scenarios that are completely convincing. The scenes and objects where textures are very smooth doesn't really convey the impression of realism.

Texture mapping is one prominent method of dealing with the inherent perfection of digital images. This technique is quite helpful in adding flaws to scenes and objects, making the portrayal more genuine and lifelike. Applying a 2D image as a texture to a 3D surface is known as texture mapping. By using this technique, fine details can be added to digital model's normally immaculate and smooth surfaces, such as surface patterns, colours, and imperfections.

This report discusses texture mapping along with some other techniques we can use to make objects and scenes in computer graphics look less pristine, crystalline, and perfect. It discusses all the methods one-by-one, along with few illustrations to provide visual examples and enhance the understanding of techniques. Finally, the conclusion will summarise the findings and emphasize the importance of the methodologies mentioned in generating more realistic computer-generated scenes.

Techniques/Methods to make objects and scenes look less pristine:

Modelling dirtiness:

As discussed by Becket W et al. [1], "Dirtiness is any deterioration of some preconceived, idealized notion of perfection corresponding visually to the result of human and natural processes". The suggested modelling method guides and adjusts a comprehensive, localised simulation of blemish instances using a rule-based methodology. There are two main parts in the process: modelling blemish instances, which require developing methods to describe localised conception of imperfections and the second step involves creating guidelines for the placement of blemishes with the intention of managing their distribution and local simulation. This involves considering various parameters of the object such as shape, composition, object use and the existence of nearby objects for the simulation process. The overall goal of the method is to offer a structured and user-friendly framework for accurately adding imperfections to simulation.

Example: To make a stain such as a coffee ring or a drip mark, we can utilise the Fractal Brownian Motion, a technique that includes creating circular pattern with shifting colours.

Stains can be made by varying specific statistical parameters while accounting for the surface's composition. A straightforward guideline can be used to determine which kind of stain to apply, drip stains are usually located close to the coffee pouring location, whereas coffee rings are typically found around the margins of surfaces or eating areas.

Blemishes can be produced with basic Gaussian or random distribution functions and 2D fractal techniques. For more intricate modelling, the high-level approach has two techniques:

1. Rule-guided aggregation: Diffusion-limited aggregation simulates the motion of particles in a sticky environment to create a structure resembling a tree. Different patterns such as rust or complex stains can be produced by substituting moving particles with aggregated growth and changing the sticking probability to a rule-based growth probability.
2. 2D fractal subdivision: When properly post-processed, arrays of values produced by 2D fractal algorithms produce blemishes with fractal boundaries or densities. The array's fractal dimensions can be used to map filled regions in an image, comparing surface smoothness or modify the ranges to simulate rings. However, when interpreting the fractal dimensions for imperfections, the creasing associated with subdivision is less noticeable than when generating fractal mountains.

Using Texture mapping technique to model dirtiness:

To model imperfections without using computation costly methods like ray-tracing, we can make use of customised texture maps based on Cook's shade trees. These maps alter properties like diffuse and specular on object's surface when used with the 'from' and 'to' attribute arguments. Materials are fetched from the database during rendering, and texture

maps with different depths are applied. We can utilise a texture map tracking approach that averages light behaviour and colours to portray thin coatings. To modify the attribute distribution, one map with a unique number is frequently sufficient.

Another simple texture mapping technique is to simply put another layer of texture on top of the pre-existing texture on the object's surface. While it provides a rapid method for adjusting appearances, potential discrepancies, and a lack of seamless integration between two textures are its disadvantages. This method may not account for three-dimensional features, leading to distortion, and lacks the sophistication for more advanced techniques like procedural texture generation or shaders. These restrictions may affect the final rendering's realism and coherence in complex circumstances.

Figure 1a illustrates the intact/more pristine looking cube object. Figure 1b depicts the same cube object with scratch marks. To add scratch marks on the cube object, we used texture mapping to simply put the scratch marks on the existing cube object texture, hence providing the illusion of imperfection.

Figure 1a: Cube object with intact textures.

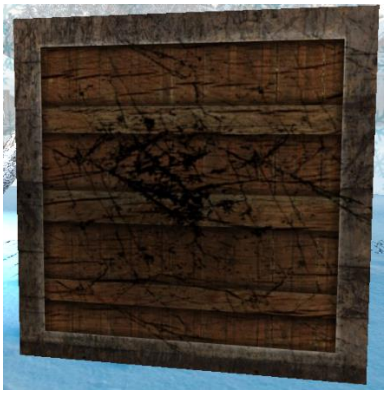

Figure 1b: Cube object scratch marks.

Figure 2a depicts the Alien model with intact textures. Figure 2b shows the same model after texture mapping the snow on its surface. To produce this imperfect version of the alien model, a simple texture mapping approach of applying a snow image texture to the alien's pre-existing body texture is utilized.

Figure 2a: Alien model with intact textures.

Figure 2b: Alien model with snow on its surface.

Figure 3a shows the spotlight model with no imperfections. Plain texture is applied on its surface to render this smooth wooden textured look.

Figure 3b portrays the same spotlight model with dirt and scratches on its surface. Again, same as above, simple texture mapping approach is applied to render this less pristine version of the spotlight.

Figure 3a: Spotlight model with Intact textures.

Figure 3b: Spotlight model with imperfections.

Imperfection models:

To make objects look realistic and less pristine, we can create a model of imperfections. Following are two ways of achieving this:

1. We experiment with imperfection effects based on visual appeal.
2. Observing real objects for modelling distribution of flaws.

Below are some types of imperfect modelling approaches:

1. Scratches: Mérillou et al. [4] describes a novel technique for simulating scratches on surfaces that combines texture mapping and BRDFs (In computer graphics, the term

“Bidirectional Reflectance Distribution Factor” (BRDF) defines the relationship between the incident light direction, the outgoing view direction, and the surface normal to describe how light reflects off a surface). The proposed method uses a theoretical scratch micro-geometry derived from physical data from precise findings. The method uses a texture map to define scratch trajectories on object surfaces and then chooses the optimal BRDF for representing scratches. To replicate the geometry of the scratches in rendering, we must comprehend their actual geometry. Basically, visible global trajectory and invisible profile (measured with a device) are the two components of scratch geometry that are taken into consideration. A surface measurement tool is employed to quantify the micro-geometric fluctuation present in various things. Using these results, a generic scratch model is developed in which the interaction between scratches and light is influenced by the normal vectors of various zones. We employ 2D texture mapping with two colours - one for the unscratched and another for the scratches - to apply scratches on the items.
Rather than the object’s colours, the map modifies BRDF. Using distinct BRDFs for every scratch zone, the scratched BRDF is computed using the scratch parameters. To ascertain the direction of the scratch, we employ the texture map. Next, the scratch BRDF is calculated for a realistic rendering, considering the angles of view, light, and scratches.
Figure 4a illustrates a cube object with scratched surface. In blender, using a sequence of nodes, we created a shader for this object. A roughness image texture is used, and it is divided with another value using a mathematical process. This calculated value is subsequently used in the bevel shader. The dot product of the normal from this bevel and geometry node is calculated at the same time. The dot product is then used as a factor in a mix shader, which combines the colours white and black. Finally, the output of this mix shader is applied to the principled BSDF’s base colour, resulting in the rendered appearance of the item. Here, imperfect object modelling with texture mapping is used to render this imperfect cube.
Figure 4b depicts the scratches on a real-world object. It is clear from comparing the pictures in Figure 4a and Figure 4b that the blender tool has been successful in making items more realistic, reflecting their real-world equivalents. Moreover, the object’s scratches have slightly altered its geometry, giving it a less pristine appearance.

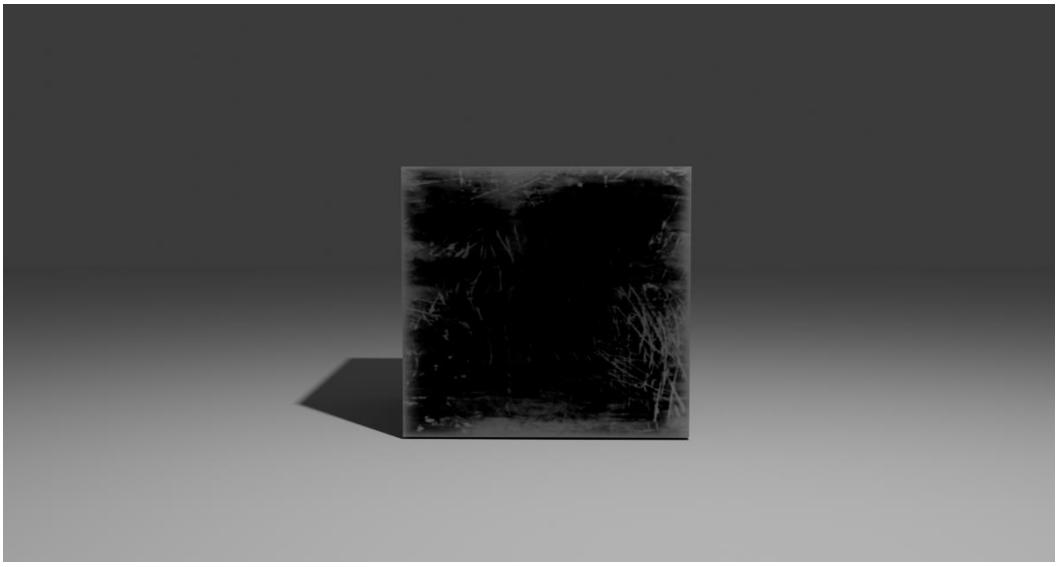

Figure 4a: Scratched cube object made in blender tool.

Figure 4b: Scratched object in real world.

2. Smudges: Fluid patches are represented as asymmetrical linked patterns. Like fingerprints, smudges use “fractal dimension” to create seamless transitions.
3. Stains: Stains use a linking boundary pattern for smoothness.
4. Mould: The mould is represented by random dot clusters.
5. Rust: Rust is rule-based, more prevalent at edges and in previously rusted places. Figure 5a portrays a sphere object with rusting effect on its surface. A shader with a noise texture is used. The factor of this noise texture is used in several ways, including connecting it to a colour ramp and using the generated colours (colour of the sphere) as a factor in a mix shader. Furthermore, the noise texture’s factor was sent to the height input of a bump node, altering the object’s surface. Another colour ramp selected the colour for the rust effect, which was then utilised as the foundation colour in a principled BSDF, which was driven by the noise texture. A mix

shader was used to blend the output of this BSDF and another Principled BSDF. The bump node altered the Principled BSDF's normal input, which contributed to the final appearance of the object.
Figure 5b shows a rusted object in real world. As we can observe, the object's geometry has changed a bit due to the rust on its surface. We can observe the same with the virtually rendered rusted object in Figure 5a. Rusting has somewhat altered the sphere's geometry.

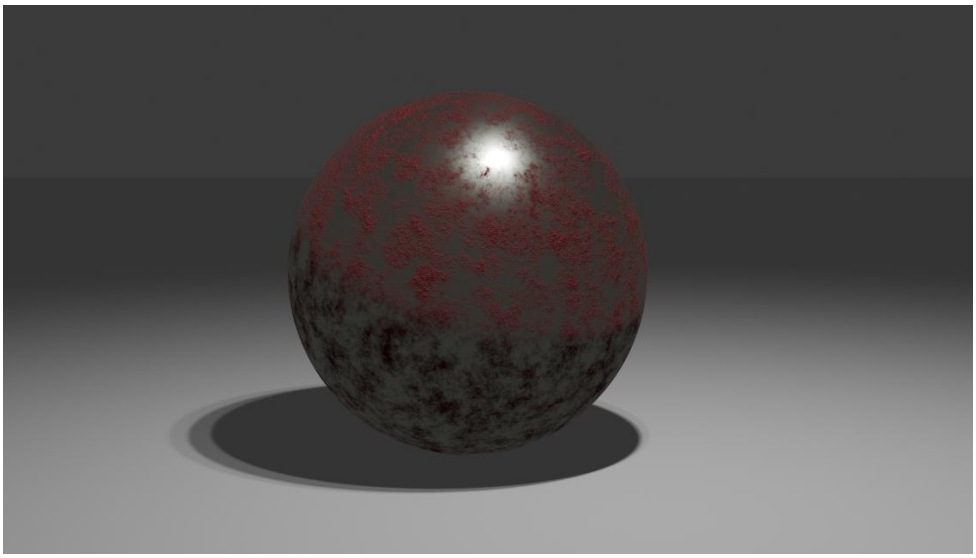

Figure 5a: A rusted object made in blender tool.

Figure 5b: Rusted object in real world

For realism, every approach considers the way imperfections interact with light.

Time-varying weathering:

Unlike some methods, the time-varying weathering simulates realistic less pristine objects and scenes without requiring any user interaction or assistance.

When things are exposed to the weather over time, they change in both colour and shape, this is called time-varying weathering.

Instead of placing pre-made textures on objects that show how something might change over time which takes an artist a considerate amount of time to create, a different technique can be applied.

Bellini et al. [2] discusses a technique in which the computer analyses the aging patterns of an existing texture, the computer creates dynamic weathered textures. The key factor is a "weathering age-map" that shows how weathered each pixel is. The age map is computed by analysing patches according to their uniqueness. The process allows for the gradual addition to features like stains and cracks. Repeated patterns remain during de-weathering, but the contents of the pixel change. An 'age-map' is produced to illustrate the extent of weathering in various regions. Using this map, it creates photos of the texture's evolution throughout the time, including weathering and de-weathering processes. The method is adaptable and works with a range of textures. Two steps involved in making the intact textures: a basic template is created, and then the original patches are transferred. Regular patterns are found, and two approaches are used for template generation: structured and stochastic. Using an age map, the de-weathered method combines the input texture with its intact form. The general structure is preserved while the texture is gradually altered by the weathering process, which adds randomness for realism. The performance of the approach depends on the input texture's generally undamaged pixels, which is a disadvantage.

Modelling lichens:

Lichens form on surfaces like rocks, trees, or soil, it frequently takes the form of a crusty of leafy growth.

According to B. Desbenoit et al. [5], for modelling and simulating the Lichens on objects in computer graphics, the Diffusion Limited Aggregation (DLA) has many disadvantages, including the need for time and memory and the modelling of lichens in perfectly uniform patterns with equally sized lobes-features that are typically absent from the nature.

Another method proposed is to simulate the growth of patterns of 3D lichens by using real-world image data. Lichen atlases, which are sets of mesh models with textures that depict various lichen species are used.

We can simulate the seeding of lichens by dispersing spores over objects and simulating how environmental elements like wind and rain affect their propagation. A unique Open Diffusion Limited Aggregation (Open DLA) model that takes environmental variables and space competition into account is used to regulate the growth of lichens. By modifying biological parameters, sensitivity to light and moisture and environmental features through texture maps, we can manipulate lichen patterns.

Textured mesh models are created from the particles that are distributed over the surface of colonised objects by instantiating template lichen models stored in the lichen atlas.

There are two methods for lichen seeding:

1. Wind and water flow simulations: Considering variables like wind and rain, we disperse spores at random throughout the chosen regions. Based on the properties of the substrate and the surrounding environment, spores adhere.
2. Painting: Spores can be directly "painted" into a surface in place of complex simulations. The "paint" colour provides exact control over spore density.

Lichen Propagation:

Lichen propagation occurs in two stages:

1. Environmental features: We look at how sunlight and moisture affect surfaces. Indirect sunlight causes lichen to develop more slowly than direct sunlight. We assess light for things that were detected during seeding and are probably going to be colonized using two lighting maps.
2. Lichen growth model: We use an Open DLA model to scatter particles on surfaces that form fractal patterns. Lichens compete for suitable sites and cease growing near favourable ones.

The Open DLA:

1. The expansion is controlled by a probability function that combines environmental conditions and cluster attributes.
2. Local particle density is considered in theoretical aggregation.
3. The environmental influence considers the amount of moisture and light at each particle's position.

We employ Monte Carlo ray tracing to streamline intricate processes like lighting assessment. The user defines the moisture with a 3D painting tool, simulating both wet and dry regions in the scene.

Modelling lichen shapes:

Using the input images as a guide, we create three-dimensional lichen shapes that are organised in a lichen atlas to replicate the fine detail of lichen pattern, textures, and shapes.

The atlas saves memory while covering large areas with a diversity of lichens.

In the Lichen atlas, models are grouped based on relative appearance probability, size, and orientation.

The lichen propagation algorithm makes use of these models based in the type, age, and orientation of the lichen.

Modelling paint cracking and peeling:

Over time, weathering can cause paint and other coating to peel and crack.

A simplified approach to simulating and rendering these effects in computer graphics is discussed by Eric Paquette et al. [6].

The physical properties of paint, such as its tension and strength, are essential to this procedure. The features help in locating surface cracks in the best possible way.

To access local surface properties, cracks based on simulated tensile stress are generated using a 2D grid. The simplicity of the crack propagation algorithm enables user control and efficient computation.

Accurate visual information is added by curling without altering the underlying item's shape. We use a customisable approximation for curling in the rendering step and use the results of the simulation to construct peeling geometry along crack paths. By assembling a mesh of micro-polygons with user-specified sizes, smooth shading is achieved. The additional geometry is seamlessly integrated into the base surface, requiring no modifications. The rendering of base and paint colours inside and outside of the cracks is made easier by the 2D grid layout.

Figure 6 shows the simulation of paint peeling effect rendered using blender tool. Here, we used a blender shader to simulate the paint flaking effect on an object. Two noise textures with different roughness parameters were used to affect the inner surface colour and roughness of the Principled BSDF. The depth of the peeling effect was decided by the second noise texture using colour ramps, colour mixes and the overall shader setup. A bump node was also employed to improve the realism by adding a normal map to the Principled BSDF. The ultimate output of a colour multiply node became the shader's base colour.

It's clear from comparing figures 6a and 6b that blender's portrayal of the paint peeling effect is quite similar to the real thing. This achievement is made possible through advanced technology.

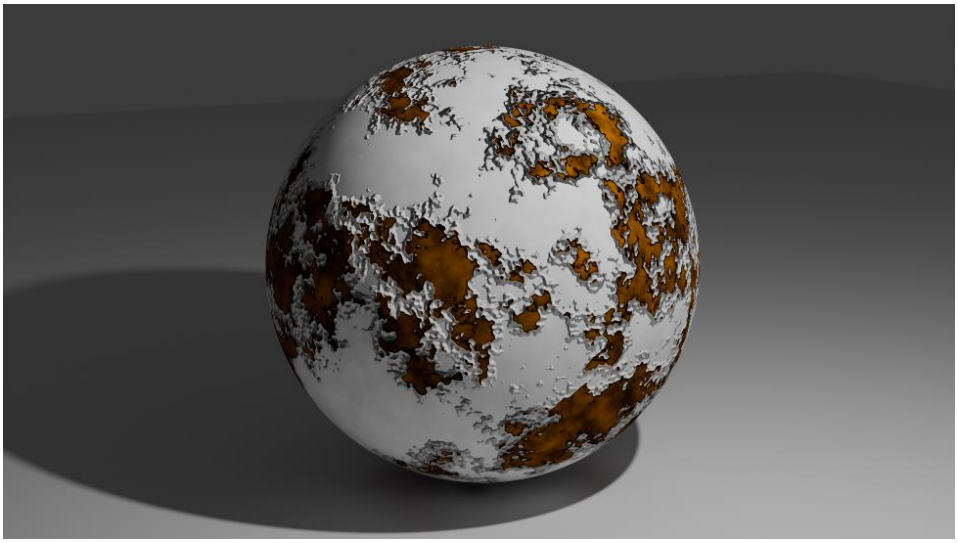

Figure 6a: A sphere object made in blender to simulate paint peeling effect on the surface of the sphere.

Figure 6b: Paint peeling in real world.

Conclusion

In Conclusion, this report explores various methods in computer graphics to overcome challenges in creating realistic scenes by incorporating imperfections. It highlights the effectiveness of texture mapping and other approaches, providing real-world examples and applications.

The report includes modelling techniques which can be used to render paint peeling effect, other method such as time-varying weathering, imperfection models, lichen modelling and dirtiness modelling. It is highlighted that texture mapping is effective for adding imperfections without the computational load of ray-tracing. The effective incorporation of rust, scratches, stains, and weathering is demonstrated through visual examples.

The complexities of paint peeling simulation and lichen modelling are explored, showcasing the importance of comprehending physical properties and utilising cutting-edge methods.

In comparison between virtually rendered and real-world objects, the report underlines the success of computer graphics techniques, particularly blender in capturing imperfections for true and lifelike scenes.

The approaches covered improve the production of realistic computer-generated scenes by bridging the gap between digital accuracy and natural imperfections.

References

1. Becket W, Badler NI. Imperfection for realistic image synthesis. Journal of Visualization and Computer Animation 1990;1(1):26–32. https://doi.org/10.1002/vis.4340010108

2. Bellini, R., Kleiman, Y. and Cohen-Or, D., 2016. Time-varying weathering in texture space. ACM Transactions on Graphics (TOG), 35(4), pp.1-11. https://doi.org/10.1145/2897824.2925891

3. Mérillou, S. and Ghazanfarpour, D., 2008. A survey of aging and weathering phenomena in computer graphics. Computers & Graphics, 32(2), pp.159-174. https://doi.org/10.1016/j.cag.2008.01.003

4. Mérillou, S., Dischler, J. & Ghazanfarpour, D. Surface scratches: measuring, modeling and rendering. Visual Comp 17, 30–45 (2001). https://doi.org/10.1007/s003710000093

5. B. Desbenoit, E. Galin and S. Akkouche, "Simulating and modeling lichen growth", Comput. Graph. Forum, vol. 23, no. 3, pp. 341-350, Sep. 2004. https://doi.org/10.1111/j.1467-8659.2004.00765.x

6. Eric Paquette, Pierre Poulin, George Drettakis. The Simulation of Paint Cracking and Peeling. Proceedings of Graphics Interface, May 2002, Calgary, Canada. pp.10. ffinria-00606725f. https://inria.hal.science/inria-00606725/document