Real-Time Cloud Simulation and Rendering

Ana Filipa da Cunha Rebelo
pg53624@alunos.uminho.pt

***Abstract*—This paper provides a comprehensive review of modern techniques for real-time cloud simulation and rendering, focusing on their application in computer graphics, movies, and flight simulation. We examine several modeling approaches, including particle systems and fluid dynamics, as well as rendering techniques such as volumetric rendering and fractal noise. Additionally, we discuss methods for animating clouds and their implementation in video games and flight simulators, highlighting key advancements and challenges in achieving realistic cloud representations.**

***Index Terms*—cloud simulation, real-time rendering, fluid dynamics, volumetric rendering, fractal noise, animation**

I. Introduction

Clouds play a vital role in shaping the visual narrative of virtual environments, yet their accurate portrayal poses significant computational challenges. This paper offers a comprehensive review of modern approaches to cloud rendering, exploring their evolution and impact on real-time computer graphics applications. The accurate simulation and rendering of clouds are crucial for creating realistic and immersive environments in various fields, including movies, video games, and flight simulation. Recent advancements in techniques such as particle systems, fluid dynamics, and fractal noise have significantly improved the visual quality and realism of cloud representations.

II. Modeling

Procedural modeling stands as a robust method for crafting intricate shapes and structures through algorithms rather than manual labor or object scanning. This approach grants a broad spectrum of control over the shape, structure, and look of diverse cloud formations. Techniques like L-Systems and reaction-diffusion systems serve as potent instruments in generating lifelike clouds within computer graphics. Through the establishment of mathematical rules and algorithms, researchers simulate the complex dynamics of clouds, producing captivating visual effects applicable across various domains, including film, animation, video games, and scientific visualization.

The L-system is a type of formal grammar as well as description for the structure of various natural objects on the basis of the growth process of the plant, or an algorithm that allows the expression. It is also used for creating a self-similarity shapes and fractal geometry shapes such as Iterated Function System (IFS) in addition to natural objects. The most important task of the clouds modeling using the L-system is to define the production rules and its parameters in accordance with the form of clouds. These two elements used to decide direction of spreading clouds, and the shape, and the density. Therefore, the first thing to do is to determine the shape of clouds to generate. Also, rendering time is another parameter to be considered .[1]

Reaction-diffusion is a process in which two or more chemicals diffuse at unequal rates over a surface and react with one another to form stable patterns such as spots and stripes. It was achieved by introducing a method that uses a sequence of processes to produce a variety of textures, and a technique for adapting these reaction-diffusion textures to different polyhedral surfaces, simulating the process directly on a mesh. This approach avoids the challenges of assigning texture coordinates. The mesh is created by distributing points evenly and determining their neighborhood relationship across Voronoi regions. Textures can be rendered directly from the mesh and used to create realistic relief on surfaces. [2]

III. Simulation

Techniques for simulating real-time clouds in computer graphics have evolved significantly in recent years. These methods aim to create visually compelling and dynamic cloud formations that respond realistically to changes in lighting, atmosphere, and environmental conditions.

A. Particle Systems

Particle systems model an object as a cloud of primitive particles that define its volume. A particle system is a collection of many minute particles that together represent a fuzzy object. Over a period of time, particles are generated into a system, move and change from within the system, and die from the system. To compute each frame in a motion sequence, the following sequence ofsteps is performed: (1) new particles are generated into the system, (2) each new particle is assigned its individual attributes, (3) any particles that have existed within the system past their prescribed lifetime are extinguished, (4) the remaining particles are moved and transformed according to their dynamic attributes, and finally (5) an image of the living particles is rendered in a frame buffer.

The resulting model is able to represent motion, changes of form, and dynamics that are not possible with classical surface-based representations. The particles can easily be motion blurred, and therefore do not exhibit temporal aliasing or strobing. Stochastic processes are used to generate and control the many particles within a particle system.the particle system can be programmed to execute any set of instructions at each step.

This type of technique is indeed challenging to implement in cloud simulation for several reasons. Firstly, the shape

and form of clouds are inherently complex, influenced by numerous factors such as wind direction, temperature, terrain, and humidity. While cloud models in atmospheric science may seem conceptually simple, they often rely on computationally intensive partial differential equations, adding to the complexity. Secondly, clouds possess the unique property of casting shadows on themselves, which significantly contributes to their appearance and realism. Simulating this self-shadowing phenomenon accurately is crucial for achieving convincing cloud renderings. Lastly, due to the intricate nature of cloud formations, a large number of particles are often required to model them accurately. This necessitates the development of efficient rendering algorithms capable of handling the substantial computational load associated with rendering such dense particle systems.[3]

B. Fluid Dynamics

Fluid simulation is effective in creating realistic clouds because clouds are the visualization of atmospheric fluid. Physics-based cloud dynamics are closely linked with fluid dynamics, because the laws from fluid dynamics (namely the Navier- Stokes equations) can be used to accurately describe the flow of air masses in the atmosphere. All approaches to modeling the dynamics of clouds im- plement some method to solve these equations. Since fluid dynamics and the Navier-Stokes equations play such an integral part in the simulation of clouds, some works that were important in the development of fluid solvers are presented first, although these works are generally not focused on clouds, but comparable phenomena such as smoke, fire, or liquids. They model fluid flow using a set of non-linear partial dif- ferential equations in terms of velocity and pressure.[4]

IV. Rendering

A. Volumetric Rendering

Volume rendering refers to a set of techniques that display a three-dimensional dataset as a two-dimensional image . The techniques can mainly be divided into two categories: indirect volume rendering and direct volume rendering (DVR) . Indirect volume rendering is based on a polygonal surface representation of the volumes, which typically uses the equipotential surface of volume data to generate triangular facets. While direct volume rendering is more or less a direct mapping of the volume equation and is implemented as volume raycasting. It is of high efficiency because it does not rely on intermediate geometric polygonal elements. The basic idea of DVR algorithms is volume raycasting, which cast rays through the volume data and accumulates the color value and the opacity value of each ray. The rays are in parallel to each other, in which case an orthographic projection would be advocating. Since raycasting is easy to implement in accelerated graphics cards that have greatly enhance the application ability of volume rendering.[5] n its basic form, the volumetric ray projection algorithm comprises four steps: Ray casting. For each pixel of the final image, a ray of vision is fired (“launched”) through the volume. At this stage, it is useful to consider the volume being touched and enclosed within a bounding primitive, a simple geometric object – usually a cuboid – that is used to intersect the viewing radius and the volume. Sampling. Along the part of the viewing radius that is within the volume, equidistant sampling points or samples are selected. In general, the volume is not aligned with the viewing radius and sampling points will often be located between voxels. Therefore, it is necessary to interpolate sample values from surrounding voxels (commonly using trilinear interpolation). Shading. For each sampling point, a transfer function retrieves an RGBA material color and a gradient of illumination values is calculated. The gradient represents the orientation of local surfaces within the volume. The samples are then shaded (i.e., colored and illuminated) according to the orientation of the surface and the location of the light source in the scene. Composition. After all sample points have been shaded, they are composited along the viewing radius, resulting in the final color value for the pixel currently being processed. The composition is derived directly from the rendering equation and is similar to mixing acetate sheets on an overhead projector. It can work backwards, that is, the calculation starts with the sample furthest from the observer and ends with the one closest to the observer. This workflow direction ensures that masked parts of the volume do not affect the resulting pixel. The front-to-back order could be more computationally efficient, as the ray’s residual energy decreases as the ray moves away from the camera; therefore, the contribution to the render integral is decreasing, so a more aggressive speed/quality trade-off can be applied (increasing distances between samples along the radius is one such speed/quality trade-off).[6]

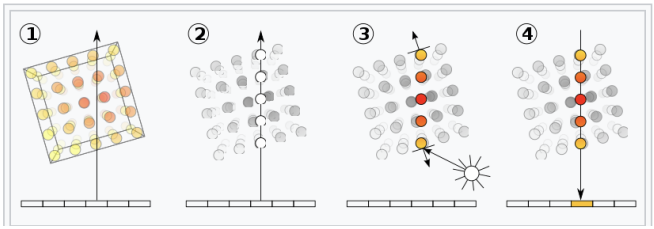

Fig. 1. The Four basic steps of volume ray casting:(1):Ray Casting,(2):Sampling,(3):Shading,(4):Compositing.

B. Fractal Noise

Fractal noise algorithms play a crucial role in generating realistic clouds in computer graphics. These algorithms create complex, natural-looking patterns that mimic the irregular and detailed structures observed in real-world clouds. fractal noise is a type of noise function that is generated by combining multiple layers of simpler noise functions, typically Perlin noise or Simplex noise. Each octave adds finer details to the overall pattern, creating a more realistic and complex texture.

Perlin’s method produces spatially coherent noise which can produce visually appealing clouds when the parameters are tuned correctly. The most useful parameters are the

number of harmonics and the weight applied when summing the harmonics together. Higher harmonics produce a finer level of detail than lower ones. When multiple harmonics are summed, noise with high and low frequencies is produced. When used for clouds, pure Perlin noise tends to produce 'overcast' type clouds; a domain where the water density is greater than 0 at almost every point. To produce more realistic clouds with definite edges and open space between clumps of cloud an exponential function is applied to the pure Perlin noise.[7]

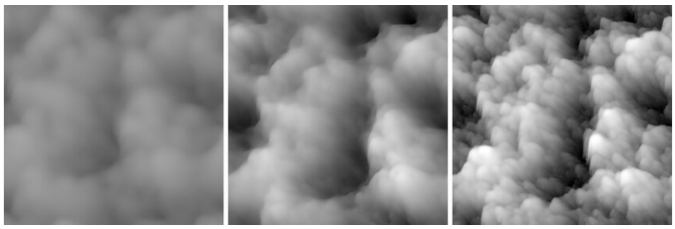

Fig. 2. Clouds created using Perlin Noise

Simplex noise is the result of an n-dimensional noise function comparable to Perlin noise ("classic" noise) but with fewer directional artifacts, in higher dimensions, and a lower computational overhead. Ken Perlin designed the algorithm in 2001to address the limitations of his classic noise function, especially in higher dimensions. [8]
In general simplex is a figure which has minimum vertices in a given space, can't be represented in space with lesser number of dimensions, and can be replicated to pitch the entire space. In two-dimensional space it is a triangle, in three-dimensional space it is a tetrahedron. It is hard to imagine what it would look like in four-dimensional space, but at least it is guaranteed that this figure would have only 5 vertices instead of 16 vertices in a case of 4D-hypercube.[9]

V. Animation

In many dynamic scenes, the requirement is not merely to model the clouds, but also to animate them. Cloud animation is a complex task, which entails dealing with the parcel and environment physics and setting the right initial conditions that leading to cloud formation.There are two broad types of methods employed in CG to generate cloud animations: procedural and physics-based. [10]

A. Procedural

Procedural methods give the illusion of animating clouds without the direct use of underlying physics. This is made possible by modeling the basic behavior of the physics field through noise, texture or obtaining cues from images, videos or other sources. Noise-based Gardner [11] achieved animation of the modeled clouds by varying the modeled mathematical texturing parameters in textured plane/ellipsoids with time. Ebert and Parent [12] propose two different methods, surface and volume-based for animating the modeled texture gases. The surface-based animation entails positioning planes at the scene boundaries with evolving parameters, which provide transparency and motion to the containing fluid. In order to animate three-dimensional volume, each point in the fluid is moved along a set path to generate swirling gases. Similarly, Max et al. [13] employ 3D textures to render clouds which are advected by the wind flow. Sakas and Westermann [14] present a functional approach for the visual simulation of cloud and fire where turbulence is generated using time-varying fractals. Much like with cloud modeling, its evolution is handled by controlled variation of the density as a function of the implicit function and time in [15]. In [16], Neyret animates the procedural textures by advecting them as a fluid. This method relies on a low-resolution grid fluid simulation where density and texture are advected using the simulation. Similar to mip-mapping, the texture is decomposed into several layers to pick up the layers most suited to the local deformation. Schpok et al. [17] animate the obtained clouds with a two-level approach. While macro-level animation governs the general direction of the cloud movement, the micro-level handles effects like wisps straying away, cloud appearing or disappearing at the edges through pre-computed volumetric texture noise. Grudzinski and Debowski [18] generate particle-based clouds with a probabilistic cloud generation function and a few global variables. The particles are assigned parameters like velocity, lifespan, maximum height, etc., as they are created, and this helps create desired trajectories that resemble a particular cloud type (stratus, cumulus, etc.). Kusumoto et al. [19] animate cumulus clouds on multiple prespecified target fields for keyframe animation. The simulation contains controlled heat sources to adjust the amount of heat generated on the ground, thereby influencing the buoyancy forces. The effect of wind is incorporated by shifting all the grid velocities by a user-specified wind velocity. The procedurally modeled cloudscapes by Webank et al. [20] are also animated procedurally through the process of morphing between two selected models or density fields. An optimal transport function helps to establish correspondences between the best pair matches in the source and target cloudscapes. The animated primitives are created from the interpolation of the identified pairs such that they follow the shortest trajectory.

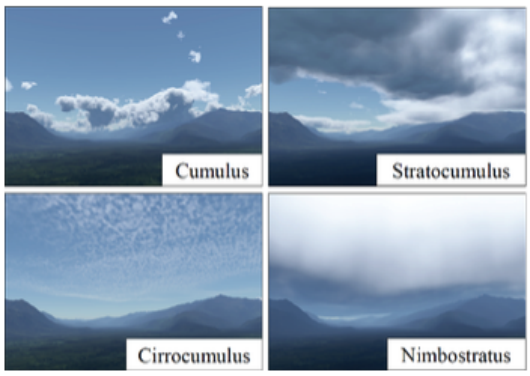

Fig. 3. A wide variety of clouds are modeled and animated procedu- rally by Webank et al

B. Physics-based

Stam [21] came up with the first unconditionally stable, grid- based model to produce complex fluid-like flows while still taking large time steps. This work has been the basis of a large number of grid-based fluid simulations in the last 2 decades, including for the clouds. Additionally, the scientific literature in meteorology and atmospheric science has been a key reference to the researchers working with clouds in the field of CG. Grid-based methods and their variations were for long the only choice for simulating the micro-level cloud physics. A grid-based solver is proposed by Overby et al. in [22] for the interactive cloud simulation. Harris et al. [23] implemented an interactive cloud simulation on the NVIDIA programmable graphics hardware. A 3D grid-based simulation is structured as layers of 2D textures for easy computation on the GPU, allowing gradual evolution of clouds in calm skies. Miyazaki et al. [24] used coupled map lattice (CML) to model and simulate various types of clouds based on the atmospheric fluid dynamics. CML is an extension of the cellular automata and uses a 3D grid and the traditional Navier–Stokes equations for simulation. The user specifies the type of cloud desired together with the initial and bound- ary conditions, grid resolution and other variables. All the variables are stored in the grid cells and are implemented through grid-based operations. Mizuno et al. [25] model volcanic clouds using modified physical laws that assume these clouds to be composed of two materials, magma and air. In this formulation, the evolu- tion of the velocity field is given by Navier–Stokes equation, leading to the evolution of the mass of magma and entrained air separately. Mizuno et al. [26] simulate volcanic clouds by obtaining the pressure formulation in Navier–Stokes equations from coupled map lattice (CML) method. As against defining the mass evolution of matter over time, formulate volcanic cloud density evolution as an equation. In [27], Dobashi and Yamamoto develop a framework for animating clouds surrounding the earth as seen from space. The motion of clouds is controlled by physics based not just on the Navier–Stokes equations, but also on the cloud thermodynamics and the Coriolis force. The input to their physics is a pressure map supplied by the user, specifying regions of high and low atmospheric pressure on the planet's surface.

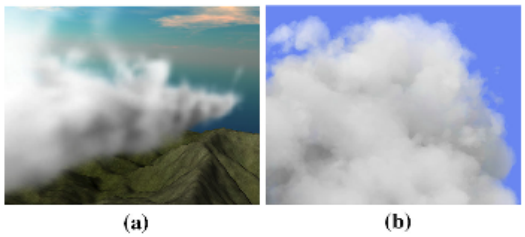

Fig. 4. a: A frame of the GPU-based cloud simulation and rendering achieved by Harris et al. and b: developing cumulus cloud modeled with two-level modeling and rendering in Schpok et al.

VI. Movies

With the advancement of technology, movie production has become increasingly complex and sophisticated, requiring the use of a plethora of computer graphics techniques, such as 3D modeling, animation, visual effects, lighting, and materials.

A. Bolt

In "Bolt," the team at Disney Animation Studios leveraged point clouds for interactive lighting of effects. A point cloud is a discrete set of data points in space. The points may represent a 3D shape or object. Each point position has its set of Cartesian coordinates (X, Y, Z).Point clouds are generally produced by 3D scanners or by photogrammetry software, which measure many points on the external surfaces of objects around them. As the output of 3D scanning processes, point clouds are used for many purposes, including to create 3D computer-aided design (CAD) or geographic information systems (GIS) models for manufactured parts, for metrology and quality inspection, and for a multitude of visualizing, animating, rendering, and mass customization applications.[28]
Typically, RenderMan point clouds and the "indirectdiffuse" calculation will generate colored bounce light from one surface to another. We have taken that basic idea and expanded it as a primary light source for a majority of environmental effects that we would create. In addition to having effects like fire illuminating hard surfaces, we integrated the lighting calculation into our volume smoke pipeline to also receive light from point clouds.
a. All effects surface, volume and sprite shaders written with the "bake3d" call in order to generate point clouds from any effect. Point clouds also exportable directly from Houdini.
b. Point clouds filtered in order to combine or reduce point counts to a manageable size using ptmerge and custom point cloud tools.
c. All effects surface, volume and sprite shaders written with the "indirectdiffuse" call to receive illumination from the generated point clouds.
d. Surfaces in our standard pipeline would receive our point clouds and use ptfilter to combine and pre-calculate the illumination for use on surfaces.

In order to fully integrate point clouds into our effects pipeline, a number of useful tools were created.
a. Houdini point cloud reader/writer. Point clouds generated from any source could be brought into Houdini to be visualized, manipulated and exported. Point clouds could also be generated directly from Houdini without rendering.
b. Point cloud filter. Heavy point clouds could be filtered down in point count with this utility. A time filter option would mix in a percentage of before/after frames to avoid time aliasing due to culling random points.
c. Maya point cloud visualizer. A visualizer inside of Maya displayed animated point clouds and any attributes. This was extremely useful for both effects and lighting artists. Illuminating smoke from animated and organic light sources is extremely difficult using standard light types.
By using point clouds generated from the actual effects

provides interactive lighting, which is easily adjustable. In order to use the “indirectdiffuse” call on volumes and sprites, it is necessary that there is no falloff of intensity based on the surface normal or I vector. Setting the “distribution” to “uniform” will enable this behavior. Since a volume does not have surface normals, we would pass an arbitrary normal and it’s inverted direction for each shaded point. We would then sum the lighting contribution from both directions. This would make sure that you were receiving the “indirectdiffuse” calculation from all directions for each volume point.[29]

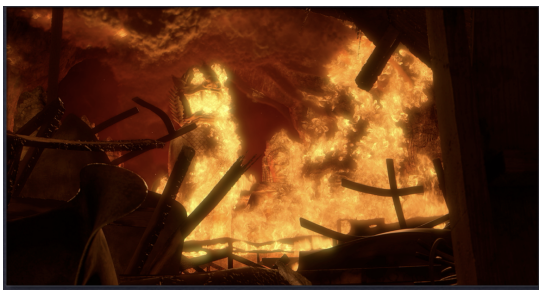

Fig. 5. Illumination of Surfaces and Volumes

Fig. 6. Volume smoke pass illuminated only by point clouds

VII. Computer Games

Over the past few decades, video games have undergone significant advancements in graphics technology, with a particular focus on creating realistic and immersive environments. One of the most important elements of any game world is the sky and its accompanying clouds. In the early days of video games, clouds were often hand-painted textures mapped onto a cube, which created the illusion of a larger game world. However, as technology has progressed, new techniques have emerged that have allowed game developers to add life and realism to game environments.[30]

A. Flow-Mapping Technique

This technique involves using a texture called a flow map to add movement to a box. The directional information stored in the flow map creates a dynamic, realistic storm that brings the game world to life.[30]

Fig. 7. The Stormy sky from the Apex Legends map ’Storm Point’

B. 3D Mesh Technique

Another technique that has gained popularity in recent years is the creation of cloud shapes as 3D meshes. These meshes are then run through filters to achieve a painterly-looking sky. This process is highly performance-intensive, so the clouds are calculated off-screen on a buffer and then composited with everything else in the scene to create one cohesive sky.[30]

Fig. 8. Clouds in Sea of Thieves

C. Real-Time Volumetric Clouds

With current technology, game studios are able to create real-time volumetric clouds using 3D volume textures that are ray-marched to represent cloud layers. This allows for highly flexible clouds that can take on any shape or form, adding a new level of realism to games like Horizon Zero Dawn. The implementation of real-time volumetric clouds requires significant computational power, but the result is a game world with stunning visual fidelity and immersion.[30]

Fig. 9. High-fidelity volumetric clouds in Horizon: Zero Dawn

VIII. FLIGHT SIMULATION

In flight simulation, clouds should have realistic looking, various shapes, and appropriate distribution across the sky of simulation world. More over, simulation application should also achieve these requirements as much as possible in real time. So, a method that can fast and efficiently render large amount clouds at large scale which comparable with the flight simulation environment is required. At this scale, some physically based methods are not applicable because of the tremendous computation cost. Fortunately, in flight simulation, physically plausible is not the issue of clouds modeling and rendering, people can use much cheaper technology such as procedure method and particle sprites system. Wang promote a simple, fast method simulating clouds to vary large scale which has been integrated into Flight Simulator 2004, a flight simulation computer game released by Microsoft. Equipped with this method, realistic clouds can be simulated in real time and thus especially suited to flight simulation applications. [31]

A. Generating cloud distribution

In Wang's paper, clouds are modeled by a set of alph-ablended textured sprites. The sprites face the camera during rendering and together comprise a 3-dimensional volume (boxes in that paper). Cloud sprites are rendered back-to-front based on distance to the camera. In that paper, a authoring tool were described as a script plug-in to 3dmax, which allow artists interactively creating clouds by placing sufficient boxes in the scene and then the plug-in populate sprites in each boxes randomly. With artist's effort in modeling work, very realistic looking clouds with diverse shapes can be created. However, in some situations that artists work are not available, or player want to change the predefined cloud system with certain distribution and shape once in a while. These above situations require that the cloud modeling system allow users generate their own realistic clouds system with reasonable randomness automatically. The algorithm generates clouds distribution in three steps as follows. Firstly, several cloud templates were prepared. Secondly, based on the cloud templates, a cloud distribution generator calculate a cloud distribution of cloud blocks, in which each cloud block is a subset of a certain cloud template. And lastly, a random generator work on each boxes of the cloud system that generate cloud sprites within that box. When this step is done, the resulting cloud system can be used in rendering immediately. Here the cloud template is a group of several boxes within which a number of cloud sprites would be generated. Each box is placed in some offset from the center of the cloud template and has its own size property. More over, each box is assigned a type, indicating which kind of cloud sprites it mainly contains. These boxes together indicate what shape would the cloud look like derived from this template. So these cloud templates can be considered as representations of different types of clouds and can be used as building blocks to generating clouds with certain variety in shape.

The algorithm can use these cloud templates to produce similar clouds block (cloud block, instant of cloud template) with slightly different shapes. Each cloud block is a subset of boxes of a certain cloud template. The cloud block is generated by coping boxes from cloud template. To explain this idea, let us look a cloud template as a ring of boxes from the data structure point of view, and the coping is start from random position in the ring and stop after walking forward or backward random steps, i.e. copy random number of boxes from random position in the ring structure. The second step is to generate cloud distribution using the templates. The generation of cloud distribution is carried out across the entire simulation field. Assuming the center of simulation is at the origin of world coordinate system. We first figure out how many cloud blocks needed to generate. And then for each cloud block, we determine its position,correspond cloud template and the subset to be copied in the template. In the third step we generate cloud sprites within each box of cloud blocks generated in the second step. In this step, we can simply use standard random function to generate certain number of cloud sprites.[31]

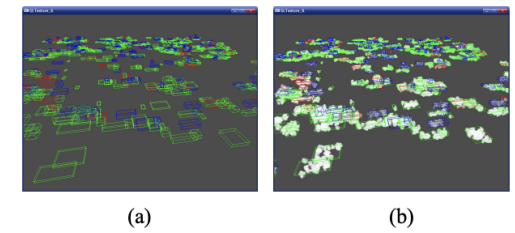

Fig. 10. Generation of cloud distribution

B. Rendering of Clouds

The clouds are render by approximating sky light and sunlight with simple calculation. As wang's method does, specifying five cloud color levels according to height in the sky and associated each height level a RGBA color. This color is considered as cloud ambient color which is the result of sky light passing through the cloud and scattered by the particles within the cloud. [31]

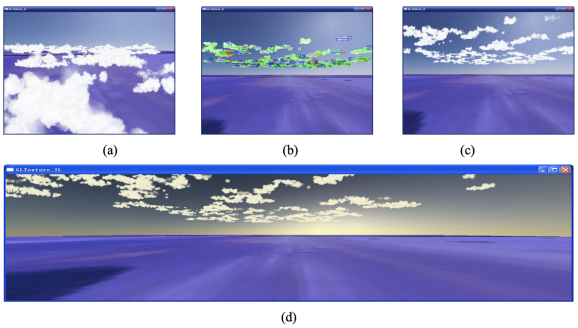

Fig. 11. Result of clouds rendering. (a) Boxes of cloud block and sprites they contain; (b) A closed view to clouds; (c) Look up to clouds from a far lower viewpoint; and (d) Dawn at 4:30AM

IX. Realistic Cloud Visualization from Weather Data

Since the beginning of numerical weather prediction the visualization of meteorological data has always been an important task. Numerical weather models calculate the future atmospheric conditions on a three-dimensional grid at discrete points in time. The data produced has to be visualized to make its four-dimensional structure transparent. But the best weather forecast is of little use if it is not communicated to the end user in an appropriate way. One important visual clue for the perception of weather is given by clouds. Although many sophisticated visualization systems for meteorological data were developed in the past, only few techniques for visualizing 3D clouds based on weather forecast data are known. In particular, the transformation of weather model data into naturalistic pictures of the sky is an open problem.

Numerical weather prediction (NWP) models are processed several times a day. They consistently provide new forecasts which are then subject to an analysis and a further post-processing for the presentation to a wider audience. However, due to the complexity of the data it cannot be presented directly. Rather, the different abstract parameter fields the data consists of have to be combined and interpreted to yield a forecast of the weather as it is experienced by the observer on the ground. In TV weather presentations, for example, usually simplified, 2D visualizations of model data can be seen. The weather forecast itself is often presented verbally and visualized by weather symbols. Obviously, much information that is contained in the NWP model data is lost during this process.[32]

A. Weather Model Data

Different types of NWP models are in operational use world-wide, two main types can be distinguished: Global models, which operate on a relatively coarse grid but whose simulation covers the whole world, and limited area models, which operate on grids with higher resolutions but are restricted to certain parts of the world. Limited- area models usually cover domains of the size of Central Europe. In this work we focus on limited area models. The state of the atmosphere is described in a NWP model by several two and three dimensional parameter fields. These parameters are defined on a grid and represent the overall conditions inside the grid cells surrounding the grid points. Clouds are represented by parameters such as cloud water content which describes the amount of liquid water inside a grid cell, or cloud coverage which models the percentage of the grid cell being covered by clouds. Despite the ever higher grid resolutions of today's weather models the cloud structure is not directly represented. For the visualization this means that the data represents only the overall structure of the clouds. Their actual visual appearance is not modeled. In the horizontal dimensions the NWP models form a regular latitude-longitude grid which is rotated such that the region of interest lies near the equator. The horizontal resolution of the grid is about seven kilometers. In the vertical dimension the grid is irregular although the number of levels is constant. The lowest grid level follows the terrain while the higher grid levels show constant heights. The grid consists of 35 levels with the highest level at about 23 kilometers and with a vertical resolution varying between 30 meters near the ground and about one kilometer in the upper levels. Obviously, the small-scale geometry of clouds, as observed in nature, cannot be represented sufficiently by such a grid.[32]

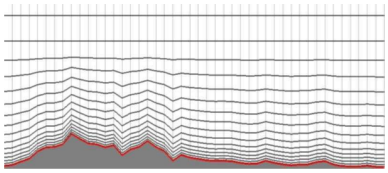

Fig. 12. Vertical structure of the numerical data grid

B. Cloud Modeling

For the realistic visualization of clouds we have to introduce details which are not encoded in the NWP data. This is accomplished on basis of the visual cloud types determined during the cloud classification. The overall cloud structure specified by the data is conserved by guaranteeing that the coverage of the particle volume inside a grid cell complies to the cloud coverage given by the data. The cloud modeling consists of placing particles ("metaballs") in the scene and specifying appropriate parameters (like size, density and texture) for each particle. Basically, two different types of grid cells may be distinguished: Totally covered and partially covered grid cells. While the totally covered grid cells have "just" to be filled up with metaballs, the particle distribution inside the partially covered grid cells is critical because these particles may be directly visible. Since a purely random particle distribution inside these grid cells does not yield visually pleasing results, a method for the placement of the particles has to be devised. The overall rendering performance highly depends on the number of particles. The obvious strategy for reducing the number of particles is to use large metaballs. Unfortunately, the grid cells of the NWP grid have a much higher resolution in the vertical than in the horizontal dimension. For clouds with a small vertical extension this means that the column has to be filled with small metaballs not exceeding the vertical boundaries. The method chosen here to cope with this dilemma is to use flattened particles representing an oval volume.The use of flattened particles rep-resenting oval shapes allows to reduce the number of particles necessary to fill thin layers. The oval shapes are achieved by simulating the tilting of the particle's rendering billboard which would usually be perpendicular to the viewing rays. The degree of tilting depends on the position of the view point and thus has to be evaluated for each particle during the rendering step. The tilting is simulated by simply scaling the height of the billboard which is perpendicular to the viewing ray.[32]

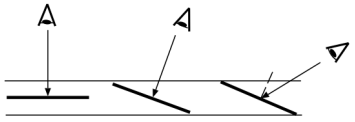

Fig. 13. View point dependent tilting of billboards.

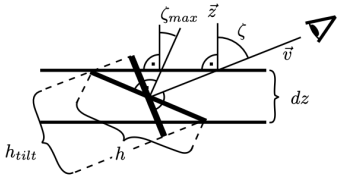

Fig. 14. Projection of the tilted billboard onto the rendering billboard.

X. Conclusion

This article explored the diverse and sophisticated techniques used in real-time cloud simulation and rendering, highlighting their applications in films, video games and flight simulators. Various methods were examined, such as particle systems and fluid dynamics for modeling, volumetric rendering and fractal noise for rendering, and procedural and physics-based approaches for animation. Each technique contributes in a unique way to the creation of realistic and dynamic cloudscapes, improving the visual and immersive quality of digital environments. Looking to the future, ongoing challenges in this field include improving computational efficiency, achieving higher levels of visual fidelity, and seamlessly integrating these techniques with other elements of virtual environments. By continuing to refine these techniques and explore new approaches, the industry can expect even more realistic and captivating virtual experiences. The intersection of technology and creativity will drive the next generation of immersive environments, where clouds are not just a background element, but a dynamic and integral part of the virtual world.

References

[1]S.Kang,K.C.Park,andK.-I.Kim,"Real-timecloudmodelingandren- dering approach based on l-system for flight simulation," International Journal of Multimedia and Ubiquitous Engineering, vol. 10, no. 6, 2015. [2]https://faculty.cc.gatech.edu/ turk/my_papers/ reaction_diffusion.pdf

[3]https://dl.acm.org/doi/pdf/10.1145/280811.280996
[4]https://d-nb.info/1108555179/34
[5]https://www.sciencedirect.com/topics/computer-science/volume-rendering

[6]https://en.wikipedia.org/wiki/Volume_ray_casting
[7]https://citeseerx.ist.psu.edu/document?repid=rep1type=pdf doi=09da15943f482bae552c32889fe54205802abc2e
[8]https://en.wikipedia.org/wiki/Simplex_noise
[9]https://www.graphicon.org/html/2008/proceedings /English/S8/Paper_3.pdf

[10]https://link.springer.com/article/10.1007/s00371-020-01953-y [11]Gardner, G.Y.: Visual simulation of clouds. ACM SIGGRAPH Comput. Graph. 19(3), 297–304 (1985) [12]Ebert, D.S., Parent, R.E.: Rendering and animation of gaseous phenomena by combining fast volume and scanline A—buffer techniques. ACM SIGGRAPH Comput. Graph. 24(4), 357–366 (1990) [13]Max, N.L., Crawfis, R., Williams, D.: Visualizing wind velocities by advecting cloud textures. In: Proceedings Visualization, pp.179–184 (1992) [14]Sakas, G., Westermann, R.: A functional approach to the visual simulation of gaseous turbulence. Comput. Graph. Forum 11(3), 107–117 (1992) [15]Ebert, D.S.: Volumetric modeling with implicit functions: a cloud is born. In: ACM SIGGRAPH Visual Proceedings: The Art and Interdisciplinary Programs of SIGGRAPH, vol. 147 (1997) [16]Neyret, F.: Advected textures. In: Proceedings of the ACM SIG- GRAPH/Eurographics Symposium on Computer Animation, pp. 147–153 (2003) [17]Schpok, J., Simons, J., Ebert, D.S., Hansen, C.: A real-time cloud modeling, rendering, and animation system. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 160–166 (2003) [18]Grudzinski, J., De bowski, A.: Clouds and atmospheric phenomena simulation in real-time 3D graphics. In: Computer Vision/Computer Graphics Collaboration Techniques: 4th International Conference, MIRAGE. Rocquencourt, pp. 117–127 (2009) [19]Kusumoto, K., Dobashi, Y., Yamamoto, T.: Keyframe control of cumulus cloud simulation, Art. 4, ACM SIGGRAPH Asia Posters (2012) [20]Webanck, A., Cortial, Y., Guérin, E., Galin, E.: Procedural cloud- scapes. Comput. Graph. Forum 37(2), 431–442 (2018) [21]Stam, J.: Stable fluids. In: Proceedings of the 26th Annual Con- ference on Computer Graphics and Interactive Techniques, ACM SIGGRAPH, pp. 121–128 (1999) [22]Overby, D., Melek, Z., Keyser, J.: Interactive physically-based cloud simulation. In: Proceedings of 10th Pacific Conference on Computer Graphics and Applications, pp. 469–470 (2002) [23]Harris, M.J., Baxter, W.V., Scheuermann, T., Lastra, A.: Simula- tion of cloud dynamics on graphics hardware. In: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware, pp. 92–101 (2003) [24]Miyazaki, R., Yoshida, S., Dobashi, Y., Nishita, T.: A method for modeling clouds based on atmospheric fluid dynamics. In: Proceedings of the 9th Pacific Conference on Computer Graphics and Applications, pp. 363–372 (2001) [25]Mizuno, R., Dobashi, Y., Chen, B.Y., Nishita, T.: Physics moti- vated modeling of volcanic clouds as a two fluids model. In: Proceedings 11th Pacific Conference on Computer Graphics and Applications, pp. 440–444 (2003) [26]Mizuno, R., Dobashi, Y., Nishita, T.: Modeling of volcanic clouds using CML. J. Inf. Sci. Eng. 20, 219–232 (2004) [27]Dobashi,Y.,Yamamoto,T.:Acontrollablemethodforanimation of earth-scale clouds. In: Proceedings of CASA (2006) [28]https://en.wikipedia.org/wiki/Point_cloud
[29]https://web.archive.org/web/20170809053200id_/http://webstaff.itn.liu.
TNCG13/Siggraph09/content/talks/355-mayeda.pdf
[30]https://www.linkedin.com/pulse/how-

clouds-work-video-games-sarthak-sachdeva
[31]https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=5455192
[32]https://www.academia.edu/4134384/LARGE_SCALE_REALISTIC_CLOUD_VISUALIZATION_BASED_ON_WEATHER_FORECAS