# THE DEVELOPMENT OF 3D COMPUTER **GRAPHICS IN THE 21ST CENTURY**

Asst. Prof. Burak DERVİŞOĞLU

ARTİKEL AKADEMİ: 467

*The Development of 3D Computer Graphics In the 21st Century*

Asst. Prof. Burak DERVİŞOĞLU
İstanbul Topkapı University
https://orcid.org/0000-0002-5638-1848

ISBN 978-625-5025-07-4

1st Edition: November - 2026

Publisher Certificate No: 19708

Cover and Book Design: Elif Esra Erzen
elifesraerzen@topkapi.edu.tr
PRINTING: Uzunist Dijital Matbaa Anonim Şirketi
Akçaburgaz Mah.1584.Sk.No:21 / Esenyurt - İSTANBUL

Certification No.: 68922

©Karadeniz Kitap - 2026

Except for quotations to be made in accordance with academic ethical rules and short quotations to be made for promotional purposes, printing, publication, copying, reproduction or distribution, in whole or in part, by electronic, mechanical or photocopying, cannot be made without written permission.

KARADENİZ KİTAP LTD. ŞTİ.
Koşuyolu Mah. Mehmet Akfan Sok. No:67/3 Kadıköy-İstanbul
0530 076 94 90

mail: destek@artikelakademi.com
www.artikelakademi.com

# THE DEVELOPMENT OF 3D COMPUTER **GRAPHICS IN THE 21ST CENTURY**

Asst. Prof. Burak DERVİŞOĞLU

artikel akademi

# CONTENTS

**Preface** ..... **7**
**Introduction** ..... **9**
Subject and Scope ..... 9
Key Concepts ..... 9
Structure of the Book ..... 11

**1. The Historical Development of Three-Dimensional Computer Graphics** ..... **12**
1.1 The Pioneering Years: The Birth of Interactive Graphics ..... 12
1.2 The 1970s: Shading, Texture and Curved Surfaces ..... 13
1.3 The 1980s: Towards the Physics of Light ..... 14
1.4 The 1990s: Industrialization ..... 16
1.5 The State of the Field at the Turn of the Century ..... 19

**2. The Transformation of Hardware and Software Technologies in the 21st Century** ..... **21**
2.1 From Fixed Function to a Programmable Graphics Pipeline ..... 21
2.2 Opening the GPU to General-Purpose Computing ..... 22
2.3 The Road to Real-Time Ray Tracing ..... 23
2.4 Game Consoles, Mobile Devices and the Cloud ..... 25
2.5 Virtual and Augmented Reality Hardware ..... 26
2.6 The Software Ecosystem: Engines, Production Tools and Open Standards ..... 26
2.7 Assessment ..... 29

**3. Modeling, Animation and Rendering** ..... **30**
3.1 Modeling ..... 31
3.2 Animation ..... 33
3.3 Rendering ..... 36

**4. Three-Dimensional Graphics in Film, Digital Games and Design ..... 41**
4.1 Film ..... 41
4.2 Digital Games ..... 45
4.3 Design ..... 48
4.4 Assessment ..... 51

**5. Artificial Intelligence and New Production Technologies ..... 52**
5.1 Deep Learning Fundamentals and Their Encounter with Graphics ... 52
5.2 Generative Models ..... 53
5.3 Neural Scene Representations ..... 54
5.4 Generative 3D Content ..... 58
5.5 New Capture and Production Technologies ..... 60
5.6 Limitations of Learning-Based Methods ..... 61
5.7 Assessment ..... 62

**6. The Future of Three-Dimensional Computer Graphics ..... 63**
6.1 Convergence in Rendering ..... 63
6.2 A Plurality of Representations ..... 64
6.3 The Transformation of Content Creation ..... 65
6.4 Spatial Computing and Immersive Environments ..... 66
6.5 Simulation, Digital Twins and Scientific Use ..... 67
6.6 Ethics, Law and Society ..... 68
6.7 Education and the Profession ..... 70
6.8 Open Research Problems ..... 70
6.9 Assessment ..... 71

**Conclusion ..... 72**
**Glossary ..... 75**
**References ..... 78**

# Preface

Fifty years ago, the idea that a picture could be computed inside a machine, starting from nothing more than a scene described in numbers, occupied a handful of research laboratories. Today the same operation runs dozens of times a second on the phone in our pocket, and we meet its results on the cinema screen, on game consoles, in architects' presentations, in museums, in surgical planning and in the digital twins of factories. This book is about one of the most intense periods of that transformation: the first quarter of the twenty-first century.

Three-dimensional computer graphics is often presented either as a set of software skills or as a branch of applied mathematics. Both views are true as far as they go, but neither explains the field on its own. 3D graphics is also a medium of visual expression, and many of its central questions are ones that painters, photographers, cinematographers and designers have been asking for centuries: how light falls on a surface, how a material reveals its character, how form and composition guide the eye, and how movement conveys weight and intention. When computer animators in the 1980s showed how squash and stretch, anticipation and timing, principles developed in traditional cel animation, could be carried over to the computer, they were acknowledging that a machine can calculate an image but cannot, by itself, make it expressive. This dialogue between computation and craft runs through the whole book.

3D graphics is, at the same time, a cultural and industrial field shaped by the interplay of hardware manufacturers, software developers, artists, designers, studios and researchers. To understand why an algorithm caught on in a given year, one often has to ask which graphics card appeared that year, which film was forced to solve which problem, which design practice needed which tool, or which game engine adopted which licensing model. Technical progress has, in turn, changed what artists and designers are able to imagine. Physically based rendering gave lighting artists a predictable common language; real-time engines turned the film set and the architectural review into interactive spaces; parametric and generative tools allowed designers to describe the rules behind a form rather than the form itself. Meanwhile, productions that deliberately

set photorealism aside, borrowing the look of comic book printing, watercolor or hand-drawn animation, have reminded us that realism is only one stylistic choice among many. This book sets out to place technical developments within this wider artistic, cultural and industrial context.

The book is organized in six chapters. The first summarizes the historical background from the 1960s to 2000, because much of what happened in the twenty-first century amounts to theoretical foundations laid in that period finally becoming practical. The second chapter examines the transformation of hardware and software, from the arrival of programmable graphics processors to real-time ray tracing. The third looks at the three core stages of a 3D production pipeline: modeling, animation and rendering. The fourth is devoted to applications in film, digital games and design, ranging from visual effects, digital humans and stylized animation to architecture, digital fabrication and cultural heritage. The fifth discusses the subject that has changed most rapidly over the past decade, artificial intelligence, together with the new production technologies that have come with it and the questions generative tools raise about authorship and creative work. The sixth chapter draws on current trends to offer some cautious thoughts about the future of the field, including what it may mean for the education and profession of artists and designers.

The book is written for undergraduate and graduate students in computer graphics, digital art and design, for professionals in visual communication design, animation, game design, architecture and industrial design, and for general readers curious about how the images around them are made. It does not require advanced mathematics; instead of equations, it favors intuitive explanations of the underlying concepts. Readers who wish to go further will find that the works listed in the references provide a detailed map for doing so.

Any book about a fast-moving field accepts that it will be partly out of date on the day it is published. For this reason, the text tries to follow lasting ideas rather than short-lived product names. Questions such as how light travels through a scene, how a surface should be represented and how a character can be made to move convincingly will remain relevant whatever the tools happen to be, and so will the artist's judgment about what an image should say. If the book helps readers, whether they come to the field from engineering or from art and design, to develop their own thinking about these questions, it will have achieved its aim.

# Introduction

In its simplest definition, three-dimensional (3D) computer graphics is the business of producing a two-dimensional image from a numerical description of objects, light sources and a viewpoint defined in a virtual space. This definition contains three separate questions. How should the shape of objects be represented? How do those shapes change over time? And how is the resulting scene turned into an image? The standard textbooks of the field treat these three questions under the headings of modeling, animation and rendering (Hughes et al., 2014; Marschner & Shirley, 2021). The chapters that follow adopt the same division, but their aim is to show how these three activities have converged in the twenty-first century and, in places, become intertwined.

## Subject and Scope

The subject of this book is the transformation that 3D computer graphics has undergone from 2000 to the present. This period stands apart in the history of the field for three reasons. First, once graphics processing units became programmable, real-time graphics broke free of predefined fixed functions and entered an environment in which researchers could run their own lighting models directly on the hardware (Lindholm et al., 2001; Mark et al., 2003). Second, the film industry moved to physically based path tracing, a method it had avoided for decades as impractical, so that the simulation of light in visual effects and animated features largely came together within a single theoretical framework (Christensen & Jarosz, 2016; Keller et al., 2015). Third, the rise of deep learning did more than speed up existing methods: by raising the possibility that the scene itself could be represented by a neural network, it called some of the field's basic assumptions into question (Tewari et al., 2020; Xie et al., 2022).

## Key Concepts

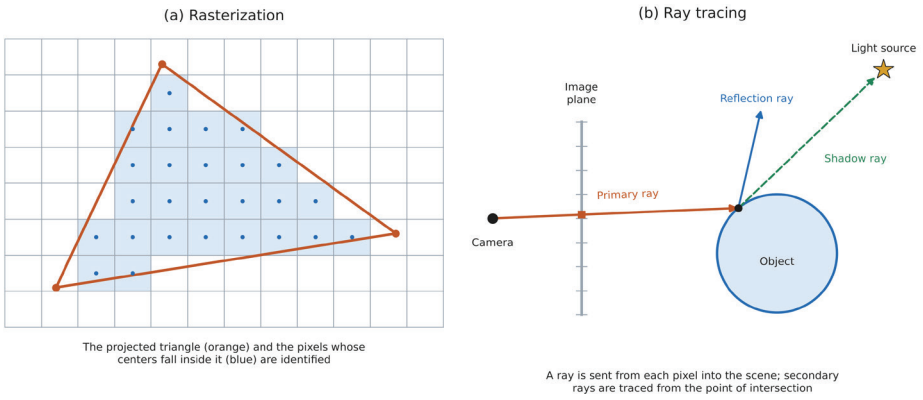
It is useful to define at the outset a few concepts that will recur throughout the book. A 3D scene usually consists of geometry, materials, lights and a camera. Geometry defines the shape of objects and is most commonly represented by polygon meshes, made up of vertices and the faces that connect them (Botsch et al., 2010). A material determines how a surface responds to the light

that falls on it; whether a surface looks glossy or matte, transparent or opaque depends on its material definition. Light sources supply energy to the scene, and the camera determines which part of that energy contributes to the image.

Rendering is the process of computing the color of every pixel from these components, and historically it has developed around two main approaches. In the first, known as scan conversion or rasterization, geometry is projected onto the screen and the pixels it covers are determined. Because this method can be implemented very efficiently in hardware, it became the foundation of real-time graphics (Akenine-Möller et al., 2018). In the second, ray tracing, rays are sent from the camera into the scene and their interactions with surfaces are followed; phenomena such as reflection, refraction and shadows arise naturally within this framework (Whitted, 1980). The basic difference between the two approaches is shown schematically in Figure 1. Path tracing, a generalization of ray tracing based on probabilistic sampling, makes it possible to compute the full circulation of light in a scene, that is, global illumination (Kajiya, 1986).

**Figure 1**

*A Schematic Comparison of Rasterization and Ray Tracing*




*Note.* On the left, in rasterization, a triangle is projected onto the screen and the pixels whose centers fall inside it are identified. On the right, in ray tracing, a ray is sent from each pixel into the scene; from the point of intersection, a shadow ray is traced toward the light source and, on reflective surfaces, a reflection ray is traced as well.

Another distinction is the one between real-time and offline rendering. In real-time applications such as a digital game, each frame must be produced within a thirtieth or a sixtieth of a second. In offline applications such as an

animated film, computing a single frame may take hours. The difference is not merely one of speed; it has led the two areas to develop different algorithms, different data structures and different production cultures. In recent years, the convergence of the tools used in these two worlds has made new forms of production, such as virtual production, possible.

## Structure of the Book

Chapter 1 summarizes the historical development of the field, from the interactive drawing systems of the early 1960s to the first feature-length computer-animated films of the late 1990s. Its purpose is not to write a complete history but to set out the conceptual foundations on which twenty-first-century developments rest. Chapter 2 examines the emergence of programmable graphics processing units, their opening up to general-purpose computing, hardware support for real-time ray tracing and the software ecosystem that grew up alongside them. Chapter 3 deals with methodological developments in modeling, animation and rendering. Chapter 4 discusses how these methods are used in film, digital games and design, and how the needs of these fields have in turn steered the technology. Chapter 5 covers methods based on deep learning, neural scene representations, generative models, and the capture and manufacturing technologies that have developed alongside them. Chapter 6 assesses the opportunities and problems facing the field in light of current trends. The Conclusion brings together the main arguments of the book.

The central argument of the book is this: the development of 3D computer graphics in the twenty-first century should be read not as a series of unconnected inventions but as the combined working of three tendencies. These are the move toward physical accuracy, the move toward real time and the move toward data-driven methods. Physical accuracy has made images more convincing and production more predictable; real-time performance has made interaction and iterative work possible; and data-driven methods have automated tasks that are difficult or expensive to do by hand. The chapters that follow try to show how each of these tendencies has taken concrete form in each subfield.

## 1. THE HISTORICAL DEVELOPMENT OF THREE-DIMENSIONAL COMPUTER GRAPHICS

The rapid progress of 3D graphics in the twenty-first century is often presented as the product of new inventions. Yet the theoretical foundations of the methods in use today were laid largely in the forty years between 1960 and 2000. Ray tracing was described in the late 1960s; subdivision surfaces and texture mapping in the 1970s; the rendering equation and path tracing in the 1980s; and image-based lighting in the 1990s. Many of these methods were far too expensive for the computers of their day, and they spread only once the hardware had caught up. This chapter outlines the historical background needed to make sense of the developments discussed in the rest of the book.

### 1.1 The Pioneering Years: The Birth of Interactive Graphics

The origin of computer graphics is usually traced to Sketchpad, the system Ivan Sutherland developed at the Massachusetts Institute of Technology. Sutherland (1963) described a system in which the user could draw directly on the screen with a light pen, define geometric constraints between the shapes drawn and create copies of a master shape. Although Sketchpad was a two-dimensional system, it gave concrete form to the idea that people and computers could communicate through a visual language, and it laid the conceptual groundwork for computer-aided design. Ideas such as the hierarchical definition of objects and the creation of instances from a master definition live on in today's scene description formats.

The head-mounted display that Sutherland introduced a few years later was a direct forerunner of 3D graphics. The system tracked the user's head movements and updated the perspective of virtual objects, drawn as wireframes, accordingly (Sutherland, 1968). Because of the hardware of the time, the device could show only simple shapes made of lines; nevertheless, it is regarded as the starting point of research in virtual and augmented reality. It is striking that an idea which would become a consumer product half a century later was put forward in the first decade of the field.

In the same years, researchers began to ask how shaded images of solid

objects could be produced. Appel (1968) proposed determining visible surfaces by casting rays from the viewpoint into the scene and computing shadows by sending a second ray toward the light source. This approach, known as ray casting, was the first form of an idea that would become recursive ray tracing a decade later. Because computing a single image took a very long time on the computers of the period, the method long remained a matter of academic interest.

### 1.2 The 1970s: Shading, Texture and Curved Surfaces

The 1970s was the decade in which the basic building blocks of 3D graphics were defined one after another. Much of this work was carried out at the University of Utah, and the researchers trained there went on to shape the direction of the field in both academia and industry.

The first problem was how to remove the faceted appearance of models built from polygons. Gouraud (1971) proposed computing the lighting only at the vertices and interpolating the colors linearly across the interior of each polygon. The method was computationally very cheap and created the impression of a curved surface, but it could not capture specular highlights correctly. To overcome this, Phong (1975) proposed interpolating the surface normal instead of the color and computing the lighting separately for each pixel; he also defined an empirical lighting model that divides reflection into ambient, diffuse and specular components. Thanks to its simplicity, this model remained one of the basic lighting models of real-time graphics for decades. Blinn (1977) proposed using the halfway vector to compute the specular term, making the model both more efficient and more consistent with physical observation; in the same paper, he also introduced reflection models based on microfacet theory to computer graphics.

The second problem was how to add surface detail without complicating the geometry. The idea of wrapping an image around a surface emerged in this period, and Blinn (1978) took it a step further by proposing that the mapped image should perturb the surface normal rather than the color. Known as bump mapping, this method makes a flat surface look rough or wrinkled and is the direct ancestor of today's normal mapping techniques. In the same year, Williams (1978) introduced shadow mapping, which renders a depth image of the scene from the point of view of the light source and determines shadows by comparison with that image. With countless refinements, shadow mapping has remained one of the most widely used shadowing methods in real-time applica-

tions for more than forty years.

The third problem was the representation of curved surfaces. Parametric surfaces used in the automotive and aerospace industries, in particular non-uniform rational B-spline surfaces, offered precise control over shape (Piegl & Tiller, 1997). However, because these surfaces are built from rectangular patches, they made it difficult to model complex branching shapes such as a character. Catmull and Clark (1978) proposed a set of rules by which a coarse polygon mesh of arbitrary topology could be refined repeatedly to arrive at a smooth surface. This approach, known as subdivision surfaces, was used relatively little for twenty years, but by the late 1990s it had become a common modeling method in character animation.

### 1.3 The 1980s: Towards the Physics of Light

In the 1980s, the center of gravity of research shifted to the question of how to make images more realistic. The decade opened with Whitted's work. Whitted (1980) proposed generating new reflected and refracted rays at the point where a ray strikes a surface and continuing this process recursively. Mirrors, glass objects and sharp shadows could thus be obtained with a single consistent algorithm. The published images were striking by the standards of the time, but the computational cost was very high and the method could handle only perfect reflection and refraction.

Cook and Torrance (1982) took a decisive step regarding the appearance of materials. The authors proposed a model that treats a surface as a collection of tiny mirror-like facets, too small to be seen, and links the color of the reflected light to the physical properties of the material. The model could explain why metals look different from plastics and, unlike earlier empirical models, it rested on physical principles such as the conservation of energy. Thirty years later, under the name of physically based shading, microfacet theory would become the shared material language of both the film and the games industries.

Indirect illumination, the lighting of surfaces by light that has already bounced off other surfaces, was another fundamental problem of the period. Goral and colleagues adapted the radiosity method, borrowed from heat transfer engineering, to computer graphics and solved the exchange of light between diffusely reflecting surfaces as a system of linear equations (Goral et al., 1984). For the first time, the method could convincingly reproduce subtle effects such as the faint tint picked up by a white surface next to a colored wall, and it was used for many years, particularly in architectural visualization.

The work that brought these separate efforts under a single theoretical roof belongs to Kajiya. Kajiya (1986) set out an integral equation stating that the light leaving a surface point in a given direction equals the light emitted by that point plus the light arriving from all directions and scattered into that direction. Known as the rendering equation, this expression showed that many earlier rendering methods could be seen as different approximations of the same equation. In the same paper, Kajiya also proposed solving the equation with the Monte Carlo method, that is, by averaging randomly sampled light paths. The method, called path tracing, was complete in theory, but the number of samples needed to remove the noise from the resulting images was far beyond the computers of the day. It would take roughly a quarter of a century for the film industry to adopt path tracing (Christensen & Jarosz, 2016).

The 1980s also saw new approaches to modeling natural phenomena. Reeves (1983) proposed modeling phenomena without a well-defined surface, such as fire, smoke and explosions, as collections of many particles, each with its own position, velocity and lifetime, and used the method to create the wall of fire that sweeps across the surface of a planet in *Star Trek II: The Wrath of Khan* (1982). Particle systems remain one of the basic tools of visual effects today. Perlin (1985) introduced a noise function that allows natural-looking textures to be generated by mathematical functions rather than image files, together with a shading language that made use of it. The idea of procedural texturing made it possible to produce countless materials, such as marble, clouds and wood, without painting them by hand. Reynolds (1987), for his part, showed that the collective motion of flocks of birds and schools of fish could be simulated by a model in which each individual follows three simple rules, looking only at its nearest neighbors. By demonstrating that complex collective behavior can emerge from simple local rules, this work proved influential well beyond computer graphics.

There were also lasting contributions in image processing and data structures. Williams (1983) introduced a pyramid structure that stores progressively smaller prefiltered copies of a texture and removes the shimmering seen on distant surfaces; this structure is now a standard feature of graphics hardware. Porter and Duff (1984) defined an algebra that adds an opacity value to the color information of each pixel and uses it to combine separately rendered image layers. Known as digital compositing, this operation is one of the fundamental operations of visual effects work in which live-action footage and computer-generated imagery are brought together. In medical imaging, Lorensen and Cline (1987) developed an algorithm that extracts an isosurface from a three-di-

mensional grid of measurements as a triangle mesh, while Levoy (1988) and Drebin et al. (1988) put forward methods for rendering volumetric data directly, without extracting an intermediate surface. The optical foundations of volume rendering were later systematically compiled by Max (1995). This body of work is the source of the volumetric rendering approach on which neural radiance fields would build thirty years later.

In animation, two distinct orientations took shape. Lasseter (1987) showed how principles that traditional cel animators had developed over decades, such as squash and stretch, anticipation and timing, could be applied to computer animation, and demonstrated the artistic possibilities of the computer as an animation tool. In contrast, Terzopoulos and colleagues proposed models based on elasticity theory that derive the motion of deformable objects such as cloth and rubber from the laws of physics (Terzopoulos et al., 1987). Witkin and Kass (1988) tried to bring these two orientations closer together by proposing that motion satisfying the animator's goals while obeying the laws of physics be computed as an optimization problem. The tension between artistic control and physical simulation remains at the heart of animation research today.

### 1.4 The 1990s: Industrialization

The 1990s was the decade in which 3D graphics left the research laboratory and became a mass industry. This transformation took place in three areas at once: the development of dedicated graphics workstations, the spread of computer-generated imagery in film, and the arrival of 3D graphics on home computers and game consoles.

On the hardware side, workstations from Silicon Graphics set the standard of the period. Akeley (1993) documented in detail the architecture of a graphics system that supported texture mapping and antialiasing in real time. The programming interface developed alongside these systems was released as an open standard under the name OpenGL in 1992 and became one of the common languages of 3D graphics programming. These workstations were extremely expensive, but the architecture that matured on them was carried over in the second half of the decade to inexpensive 3D accelerator cards for personal computers. The arrival in 1999 of the first consumer cards that also handled vertex transformation and lighting, and the marketing of one of them, the NVIDIA GeForce 256, as a graphics processing unit, marks the starting point of the transformation discussed in the next chapter.

In film, the use of computer-generated imagery in the 1980s was limited to

a few short sequences. In the early 1990s, the liquid-metal character of *Terminator 2: Judgment Day* (1991) and, soon after, the dinosaurs composited with live action in *Jurassic Park* (1993) permanently changed the expectations of audiences and producers alike. Prince (2012) has examined in detail the relationship that digital visual effects formed with photographic realism in this period, arguing that computer-generated imagery did not undermine cinema's claim to reality but instead established a new form of perceptual realism. The release in 1995 of *Toy Story*, the first feature-length film produced entirely by computer, led to the acceptance of 3D animation as a cinematic form in its own right. Placing these developments in a broader framework, Manovich (2001) argued that digital cinema had become a particular kind of animation, one that uses live-action footage as just one of its many raw materials.

During this period, the needs of productions directly shaped the research agenda. In work carried out for Pixar's 1997 short film *Geri's Game*, DeRose and colleagues showed how subdivision surfaces could be used in an animation studio's production pipeline and defined the extensions needed to control sharp edges and creases (DeRose et al., 1998). Following this work, subdivision surfaces gradually replaced parametric patches in character modeling. Baraff and Witkin (1998) developed a numerical method that allows large time steps to be taken stably in cloth simulation, making garment animation usable in production. Stam (1999), for his part, proposed an unconditionally stable solution of the fluid equations, opening the way for physically based simulation of phenomena such as smoke and water within acceptable times.

In geometry processing, making scanned or highly detailed models manageable became an important research topic. Hoppe (1996) proposed storing a polygon mesh as a coarse base mesh together with a sequence of operations that gradually restores the detail, while Garland and Heckbert (1997) developed a method that rapidly reduces the number of faces in a mesh while preserving its shape as far as possible. Known collectively as level-of-detail management, these techniques allow distant objects to be drawn with fewer polygons and became indispensable components of real-time applications.

In the second half of the decade, the idea of using measurements taken from the real world directly in rendering came to the fore. Levoy and Hanrahan (1996) showed that by storing many photographs of a scene in a regular structure, images could be generated from new viewpoints without any geometric information. This representation, called the light field, was among the first comprehensive solutions to the problem of novel view synthesis. Debevec and Malik (1997) developed a method for recovering the true light intensities in a scene from pho-

tographs taken at different exposure times, and Debevec (1998) showed that these high dynamic range images could be used to light virtual objects. Known as image-based lighting, this approach became one of the common methods for placing computer-generated objects convincingly into live-action scenes. Blanz and Vetter (1999) derived a statistical model of shape and texture from a large number of scanned faces and showed that a 3D face model could be obtained from a single photograph. These studies were early examples of the data-driven approach that would prove decisive in the twenty-first century.

Research on global illumination also made significant progress. Ward (1994) presented a rendering system that put physical accuracy first, intended for use in architectural lighting design. Jensen (1996) proposed photon mapping, which stores photons emitted from light sources in the scene and uses this information during rendering; the method proved especially successful at computing caustics, the bright patterns formed when light is focused by transparent objects. Veach and Guibas (1997) developed a sampling method that improves the efficiency of path tracing under difficult lighting conditions by exploring the neighborhood of an important light path once it has been found. These studies completed the theoretical foundations of physically based rendering, although widespread use in production would have to wait another decade.

Another defining development of the decade was the move of digital games into three dimensions. At the start of the decade, most games were drawn with two-dimensional sprites; the first popular titles to give an impression of depth, such as *Wolfenstein 3D* (1992) and *Doom* (1993), actually relied on methods that worked on a two-dimensional map and drew walls as vertical strips. By the middle of the decade, fully polygonal environments and characters, as in *Quake* (1996), had become common in arcades, on home consoles and on personal computers. This transition was not merely a visual change; it redefined basic problems of game design such as camera control, level design and character movement. To reach acceptable frame rates on limited hardware, game developers made heavy use of techniques such as visibility determination, level-of-detail management and precomputed lighting, and this experience strongly influenced the architectural principles of today's game engines (Gregory, 2018). 3D accelerator cards for personal computers, such as the 3dfx Voodoo released in 1996, also reached a mass market largely thanks to the demand created by games, and over time they became able to compete with expensive workstations.

### 1.5 The State of the Field at the Turn of the Century

By the year 2000, 3D computer graphics appeared to be split into two separate worlds. In film, rasterization architectures based on micropolygons, capable of handling complex scenes with limited memory, were dominant; ray tracing was used selectively only for reflection and refraction effects where it was deemed necessary, and indirect illumination was mostly faked with additional lights placed by hand by artists (Christensen & Jarosz, 2016). In games, fixed-function hardware allowed only a few predefined lighting and texturing operations, and developers achieved visual richness largely through precomputed light maps. The two worlds used different tools, different data formats and different job descriptions.

The decisive factor behind this picture was computing power. Moore (1965/1998) predicted that the number of components that could be placed on an integrated circuit would double at regular intervals, and this prediction served for decades as the roadmap of the semiconductor industry. The fate of many methods in graphics was tied to this curve as well: whether a technique known in theory was practical was often decided by the state of the hardware in a given year. Table 1 summarizes the milestones discussed in this chapter.

**Table 1**

*Selected Milestones in Three-Dimensional Computer Graphics, 1963–1999*

| Year | Development | Source |
|---|---|---|
| 1963 | Interactive drawing and constraint-based design | Sutherland (1963) |
| 1968 | Head-tracked display; ray casting | Sutherland (1968); Appel (1968) |
| 1971–1975 | Smooth shading and an empirical lighting model | Gouraud (1971); Phong (1975) |
| 1978 | Bump mapping, shadow mapping, subdivision surfaces | Blinn (1978); Williams (1978); Catmull and Clark (1978) |
| 1980 | Recursive ray tracing | Whitted (1980) |
| 1982 | Physically based microfacet reflection model | Cook and Torrance (1982) |
| 1984 | Radiosity; digital compositing | Goral et al. (1984); Porter and Duff (1984) |
| 1986 | The rendering equation and path tracing | Kajiya (1986) |

| Year | Development | Source |
|---|---|---|
| 1987 | Flocking behavior, isosurface extraction, principles of animation | Reynolds (1987); Lorensen and Cline (1987); Lasseter (1987) |
| 1996–1998 | Light fields and image-based lighting | Levoy and Hanrahan (1996); Debevec (1998) |
| 1998–1999 | Subdivision surfaces in production; stable fluid simulation | DeRose et al. (1998); Stam (1999) |

What was missing at the turn of the century was the computing power to put this accumulated knowledge into practice and a common framework to bring the two worlds closer together. The computing power would come from programmable graphics processing units, and the common framework from the adoption of physically based rendering in both fields. The next chapter turns to the hardware and software side of this transformation.

## 2. THE TRANSFORMATION OF HARDWARE AND SOFTWARE TECHNOLOGIES IN THE 21ST CENTURY

The history of 3D graphics is a history of mutual dependence between algorithms and the machines that run them. When a method becomes widespread is often determined not by when it was invented but by when hardware capable of running it in a reasonable time appears. This relationship was especially visible in the first quarter of the twenty-first century. The graphics processing unit turned from a specialized circuit performing a fixed function into a general-purpose parallel processor capable of running arbitrary programs, a transformation that deeply affected not only graphics but also scientific computing and artificial intelligence. This chapter looks first at the changes in hardware and then at the software layers that grew up on top of that hardware.

### 2.1 From Fixed Function to a Programmable Graphics Pipeline

A graphics card from around 2000 consisted of a fixed processing pipeline that transformed geometry onto the screen, broke triangles down into pixels and applied a texture and simple lighting to each pixel. Developers could switch parts of this pipeline on and off, but they could not change what happened inside it. As a result, the games of the period bore the marks of the same lighting model, whichever artist had made them.

The step that changed this was the ability to replace certain stages of the pipeline with small programs written by the user. Lindholm and colleagues presented the design of a vertex processing unit that runs for every vertex and can transform data such as position, normal and color in arbitrary ways (Lindholm et al., 2001). The same flexibility soon reached the pixel level, where programs running for each pixel allowed artists and programmers to define their own lighting models. These programs, called shaders, were at first written with a limited instruction set close to machine language. Mark et al. (2003) introduced a system that allowed shaders to be written in a high-level language resembling C, and similar languages soon became part of every major programming interface. High-level shading languages took hardware programming out of the hands of a small group of specialists and led to a rapid increase in the visual

variety of real-time graphics.

The first fruits of programmability were effects that had previously been seen only in offline rendering being brought into real time. Within a few years, per-pixel lighting, normal mapping, high dynamic range rendering and tone mapping had become commonplace in games. The tone mapping operator developed by Reinhard and colleagues, inspired by photographic practice (Reinhard et al., 2002), became one of the most widely used methods for convincingly displaying scenes with a wide range of brightness on displays with a limited range. Sloan and colleagues proposed a method in which an object's response to light arriving from its surroundings is precomputed and stored in compressed form, so that global illumination effects such as soft shadows and interreflections could be obtained in real time under changing lighting (Sloan et al., 2002). The idea of striking a balance between precomputation and runtime flexibility became one of the main lines of real-time global illumination research over the following two decades (Ritschel et al., 2012).

Toward the middle of the decade, the separately designed vertex and pixel units of the pipeline gave way to a unified architecture made up of many identical general-purpose processing cores. Stages that allowed geometry to be tessellated in hardware were then added, along with compute shaders that allow general computation outside the graphics pipeline. Akenine-Möller et al. (2018) provide a comprehensive account of the present form of each of these stages and of the techniques built on them.

### 2.2 Opening the GPU to General-Purpose Computing

When programmable graphics hardware appeared, researchers realized that it could be used not only to produce images but for any computation with a parallel structure. In the first attempts, scientific problems were recast as texture and pixel operations; in other words, the graphics interface was put to uses for which it had not been designed. Buck et al. (2004) proposed a programming model that hid graphics concepts from the programmer and described data as streams and computation as kernel functions applied to each element of those streams. A few years later this idea became a commercial programming environment. Nickolls et al. (2008) explained the basic abstractions of this model, in which thousands of threads are organized in hierarchical groups, while Owens et al. (2008) surveyed the state of GPU computing at the time, its successful areas of application and the difficulties encountered.

This development had two important consequences for graphics. The first

was direct: work previously carried out on the central processing unit, such as physics simulation, volumetric data processing and ray tracing, moved to the GPU. The second was indirect but more profound. The ImageNet study presented by Krizhevsky and colleagues in 2012 trained a large convolutional neural network on GPUs, outperformed earlier methods in image classification by a clear margin, and became one of the emblems of the rise of deep learning. The text cited in this book is the later version of that study published in *Communications of the ACM* in 2017 (Krizhevsky et al., 2017). This breakthrough in deep learning was made possible by hardware designed for playing games becoming the basic infrastructure of artificial intelligence research. As Chapter 5 will show, a few years later these methods returned to computer graphics and transformed the field from within. This cycle, in which graphics hardware fed artificial intelligence and artificial intelligence in turn fed graphics, is one of the most remarkable interactions in the history of twenty-first-century computing.

### 2.3 The Road to Real-Time Ray Tracing

Using ray tracing in real-time graphics had long been one of the field's deferred goals. Rasterization is very efficient at finding which pixels a triangle covers, but it cannot directly answer the question of whether one point is visible from another. Because shadows, reflections and indirect illumination depend on exactly this question, each of these effects had to be obtained by a separate approximation in rasterization-based systems. Techniques such as shadow maps, screen-space reflections, ambient occlusion and precomputed light probes each had their own artifacts and edge cases.

During this period, researchers developed intermediate solutions that brought indirect illumination closer to real time. Crassin and colleagues proposed a method that converts the scene into a hierarchical voxel structure and gathers indirect light by tracing widening cones through this structure (Crassin et al., 2011). The method could approximate indirect illumination in dynamic scenes at interactive rates. Even so, such solutions lacked the generality of ray tracing.

Efforts to accelerate ray tracing itself proceeded along two lines. On the CPU side, Wald and colleagues presented a highly optimized library of ray tracing kernels that exploits the wide vector instructions of modern processors (Wald et al., 2014). The library was designed so that it could be plugged into different renderers as their ray–geometry intersection layer. On the GPU side, Parker et al. (2010) introduced a general-purpose ray tracing engine in which

stages such as ray generation, intersection and shading could be programmed by the user. This engine showed that the shader concept of the rasterization pipeline could be adapted to ray tracing, and a similar structure of programmable stages was later adopted in the ray tracing support added to mainstream programming interfaces.

The turning point came in 2018. That year, the first consumer graphics cards with dedicated circuitry for testing intersections between rays and acceleration structures reached the market with NVIDIA's GeForce RTX family, and ray tracing support was added to widely used programming interfaces such as DirectX. The collection edited by Haines and Akenine-Möller (2019) documents the first ways in which this new hardware was used and the practical problems encountered. Despite hardware support, the number of rays that could be traced per pixel was very limited: whereas a film frame might use hundreds or thousands of samples per pixel, the real-time budget was often restricted to one or a few. What made real-time ray tracing practical, therefore, was as much the reconstruction algorithms that accompanied the hardware as the hardware itself.

The first generation of these algorithms relied on filtering the noisy image across time and space. Schied and colleagues developed a filter that uses information accumulated from previous frames and a per-pixel estimate of variance to obtain stable results from path-traced images rendered with a single sample per pixel (Schied et al., 2017). In the same year, Chaitanya and colleagues showed that this reconstruction task could be taught to a recurrent neural network (Chaitanya et al., 2017). The second generation focused instead on choosing samples more intelligently. Bitterli and colleagues proposed a resampling method that reuses good light samples found in neighboring pixels and previous frames; as a result, direct illumination in scenes containing millions of dynamic light sources could be computed at interactive rates with only a few rays per pixel (Bitterli et al., 2020). This method and its descendants later found their way into the real-time ray tracing libraries that hardware vendors provide to developers.

Another dimension of reconstruction concerns resolution. Temporal antialiasing slightly jitters the sample positions in each frame and combines the results with previous frames, approaching multisample quality with a single sample per pixel; Yang et al. (2020) examine this family of methods and their typical artifacts in detail. Taking the same idea one step further, an image can be rendered at low resolution and then upscaled using temporal information. Xiao and colleagues showed that this upscaling could be performed by a learned neural network and that detail could be largely preserved even when the pixel

count was increased sixteenfold (Xiao et al., 2020). Similar approaches have become commercial upscaling solutions offered by graphics card vendors and game engines and are available as an option in many current games. In these solutions, the image is rendered at an internal resolution below the output resolution and then upscaled. Real-time rendering thus now includes, alongside the approach in which every pixel is computed directly, approaches in which the image is estimated from sparse and noisy measurements.

### 2.4 Game Consoles, Mobile Devices and the Cloud

In parallel with developments in personal computers, each generation of game consoles has set the level that 3D graphics reaches on a mass scale. The fixed hardware of a console allows developers to learn a particular machine over many years and push it to its limits; for this reason, console games have been one of the important proving grounds in which many real-time rendering techniques have matured. Gregory (2018) discusses in detail how console and PC hardware have shaped game engine design, from memory management to the distribution of work across multiple cores. The fact that the PlayStation 5 and the Xbox Series X and Series S, released in 2020, offered hardware-accelerated ray tracing and high-speed solid-state storage as standard brought the techniques described in the previous section to a much wider audience of players.

The spread of smartphones has brought 3D graphics to billions of devices. Because of power and thermal limits, mobile GPUs rely on architectural choices that differ from those of their desktop counterparts, the best known of which is tile-based rendering. From the mid-2010s, low-overhead programming interfaces of a new generation, such as Vulkan, Metal and Direct3D 12, spread across both desktop and mobile platforms. What these interfaces have in common is that they reduce the hidden work done by the driver, hand control to the application and make better use of multicore processors.

The idea that rendering need not take place on the user's device was also put into practice during this period. In remote rendering, the scene is processed on a powerful server and the result is sent to the user as a compressed video stream. In their comprehensive survey of such systems, Shi and Hsu (2015) identify interaction latency as the main challenge and note that methods such as reprojecting the image on the client side have been developed to reduce it. Cloud gaming services and some browser-based 3D design tools are examples of this approach.

### 2.5 Virtual and Augmented Reality Hardware

Although the idea of head-mounted displays dates back to the earliest years of the field, practical products on a consumer scale appeared only in the mid-2010s. The reason for this delay is the severity of the requirements for a convincing virtual reality experience: a separate, high-resolution image for each eye, a high and stable refresh rate, and the shortest possible latency between head movement and image update. The required refresh rate is not a single universal value: different products may use 72, 90 or more frames per second, and in some systems the frames produced by the application are reprojected according to the latest head pose to match the refresh rate of the display. It should also be kept in mind that in such systems the refresh rate of the display and the rate at which the application produces new frames are not the same thing. Likewise, the numerical limits given for acceptable latency are not universal thresholds but design targets that depend on the product and on the conditions of measurement. Koulieris et al. (2019) provide a comprehensive review of the optical design of near-eye displays, tracking technologies and the constraints that the human visual system imposes on these devices.

These demanding requirements brought rendering methods that exploit the properties of human perception to the fore. The human eye has high acuity only in a narrow region around the point of gaze; in peripheral vision, the perception of detail falls off rapidly. Guenter and colleagues developed a system that uses eye-tracking data to render only the region being looked at in full resolution and the rest at progressively lower resolution, showing that the computational load could be reduced severalfold under conditions in which users did not notice the difference (Guenter et al., 2012). Patney et al. (2016) found that simply lowering the resolution in the periphery could produce a sensation of tunnel vision and proposed a contrast-preserving filtering approach. Kaplanyan and colleagues showed that a convincing image for peripheral vision could be reconstructed by a generative neural network from a sparse image in which only a small fraction of the pixels had been computed (Kaplanyan et al., 2019). This approach, known as foveated rendering, can be used in practice on headsets equipped with eye-tracking hardware.

### 2.6 The Software Ecosystem: Engines, Production Tools and Open Standards

The transformation in hardware would have had only a limited effect with-

out the software layers that made it accessible. Three tendencies stood out in these layers during the twenty-first century: game engines turned into general-purpose real-time production environments, digital content creation tools matured and became more accessible, and open standards for exchanging data between studios spread.

A game engine is a software framework that brings together the common infrastructure components of a game, such as rendering, physics, audio, animation and asset management (Gregory, 2018). At the beginning of the century, most large studios developed their own engines, while commercial engines were sold to a limited number of companies for high license fees. In the mid-2010s, leading commercial engines were opened up to everyone under free or revenue-sharing models, which radically changed access to these tools for independent developers, students and industries outside games. In the same period, engines adopted physically based material models, so that the material definitions used in film and game production converged. More recently, for example with Unreal Engine 5, released in 2022, commercial engines gained virtualized geometry systems that allow geometry with very high polygon counts to be used without hand-made levels of detail, together with dynamic global illumination solutions that require no precomputation. These capabilities have made it easier to use game engines in fields ranging from architectural visualization to automotive design and from television broadcasting to film production.

On the content creation side, specialized software for modeling, animation, digital sculpting, texture painting and effects simulation has matured. One of the most notable developments of the period is that open-source tools have reached a level at which they can be used in professional production. The best-known example, the open-source package Blender, offers individual artists and small studios a comprehensive production environment, with a path-tracing renderer and a real-time preview engine. A similar trend can be seen in research; for example, the open-source mesh processing tool presented by Cignoni and colleagues is one of the tools most frequently used to clean, repair and simplify scanned data (Cignoni et al., 2008). Open-source tools have not only lowered costs but also shortened the distance between research and practice by making methods more reproducible.

The third tendency is the shift of data exchange formats toward open standards. On a large film production, hundreds of artists, dozens of different software packages and often more than one studio work together, and collaboration on this scale is extremely difficult without common data formats. Many formats and libraries that studios had developed for their own needs were released as

open source during this period, chief among them OpenEXR for high dynamic range images, Alembic for cached animated geometry, OpenVDB for sparse volumetric data, OpenSubdiv for the efficient evaluation of subdivision surfaces and OpenColorIO for the consistent management of color transformations. The hierarchical sparse volume data structure introduced by Museth (2013) forms the basis of OpenVDB: developed at an animation studio for the efficient storage of smoke, cloud and liquid simulations, it became a widely used volume format, supported by many software packages, after its release as open source.

At the level of the scene, two standards serving different needs have come to the fore. Universal Scene Description, released as open source by Pixar in 2016 and now known as OpenUSD, describes an entire scene in layers; its layered composition structure, which allows different artists and departments to work on the same scene without disturbing one another's work, was designed for large-scale production workflows. glTF, developed by the Khronos Group, is by contrast a lean transmission format, especially since its 2.0 version released in 2017, focused on the efficient delivery of 3D assets over networks and their rapid loading in applications. In short, OpenUSD targets scene composition within the production pipeline, while glTF targets the distribution of finished assets. These standards have begun to be used not only in the entertainment industry but also in industrial digital twin applications.

### *2.6.1 From Shading Languages to Material Systems*

Another important change in the software layer is that the definition of materials has passed from programmers to artists. In the early years of programmable hardware, every new surface appearance required shader code written by hand by a programmer. Over time, engines and production software began to offer interfaces in which materials are defined visually by connecting nodes, so that artists could create original materials without writing code by combining texture samples, mathematical operations and procedural noise functions (Perlin, 1985). Behind the scenes, these node graphs are automatically translated into shader code suitable for the target hardware.

The adoption of physically based material models simplified this process further. Instead of choosing a separate lighting model for each material, artists now prepare texture maps that specify a small number of intuitive parameters such as base color, roughness and metalness. To make these maps portable between different engines and renderers, shared material description formats such as MaterialX and definitions such as glTF's physically based material model

have been developed. As a result, it has become easier for a material created once to look similar in different environments, although exact equivalence often still requires additional adjustment because of the particular differences of each renderer. The fact that texture painting software works directly on the 3D model and with this set of parameters has made material creation a distinct specialization.

Another prerequisite for materials to look consistent across environments is color management. Physically based computations must be carried out in a linear color space, textures and images must be correctly converted into that space, and the result must be converted for the target display or print medium only at the final stage. If one of these steps is skipped, the same material can look markedly different in different software packages. The film industry uses shared color encoding systems and open-source color transformation libraries of the kind mentioned above for this purpose.

### 2.7 Assessment

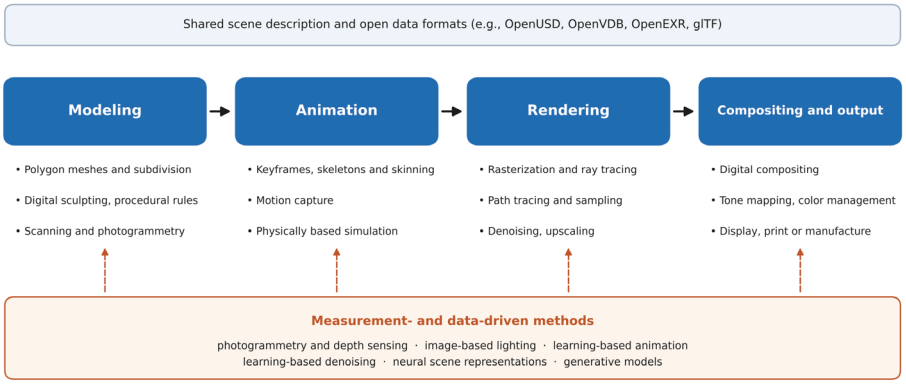
The common denominator of this twenty-five-year transformation is that computation has become increasingly flexible and increasingly parallel. The fixed-function pipeline gave way to programmable stages, programmable stages to general-purpose computing, and general-purpose computing to hybrid architectures enriched with units specialized for ray tracing and neural network inference. On the software side, access to tools has widened, and infrastructure that studios once developed behind closed doors has evolved into standards that function as a common good. The change has also been reflected in education: topics once considered advanced, such as shader programming and ray tracing, are now standard chapters in introductory textbooks (Marschner & Shirley, 2021).

## 3. MODELING, ANIMATION AND RENDERING

Whatever industry it is built for, a 3D production pipeline performs three basic tasks: it defines shapes, sets those shapes in motion and turns the resulting scene into an image. These three tasks correspond to the traditional subfields of the discipline, namely modeling, animation and rendering. In the twenty-first century, a similar change of direction can be seen in all three. A way of working in which the artist specifies every detail by hand has steadily given way to one supported by measurement, simulation and rule-based methods. Models have come to be derived from photographs and scans, motion to be captured from performers or computed from the laws of physics, and light to be distributed by physical simulation rather than by the artist's intuition. Figure 2 summarizes these three stages and the families of methods that stand out at each. The chapter examines this change under three headings.

**Figure 2**

*The Main Stages of a 3D Production Pipeline*




*Note.* The examples beneath the boxes show the main methods discussed in this book for each stage. The upper band represents the shared scene description and open data formats that connect the stages; the lower band represents the measurement-based and data-driven methods that have gained ground at every stage.

## 3.1 Modeling

### *3.1.1 Surface Representations*

How a shape is stored in the computer determines everything that can be done with it. The most common representation, the polygon mesh, describes a surface as a structure of vertices, edges and faces. Botsch et al. (2010) provide a comprehensive treatment of the data structures used for polygon meshes and of the operations defined on them, such as smoothing, parameterization, remeshing, simplification and deformation. The strengths of polygon meshes are their simplicity and their direct compatibility with graphics hardware; their weakness is that smooth surfaces can only be approximated.

In engineering applications that require precise control over shape, parametric surfaces have remained in use. The non-uniform rational B-spline surfaces described in detail by Piegl and Tiller (1997) allow a car body or an aircraft wing to be defined within manufacturing tolerances and are the basic representation in computer-aided design software. In the entertainment industry, by contrast, subdivision surfaces have become a widely used representation. The rules defined by Catmull and Clark (1978) make a coarse control mesh converge in the limit to a smooth surface; combined with the crease control added by DeRose et al. (1998), this representation lets artists create complex organic shapes with only a small number of control points. The addition of tessellation stages to graphics hardware has made it possible to use subdivision surfaces in real-time applications as well.

Digital sculpting tools, which spread at the beginning of the century, profoundly changed modeling practice. Rather than moving individual vertices, the artist shapes a surface made of millions of polygons with brush-like tools, much as one might work clay. The resulting highly detailed model is too heavy to animate directly, so the detail is transferred onto a lighter mesh in the form of normal and displacement maps. This division of labor between high-resolution sculpting and a low-resolution production mesh long remained one of the common methods for creating characters and environments. The simplification methods developed by Hoppe (1996) and by Garland and Heckbert (1997) were used in this process to derive levels of detail automatically. More recently, the virtualized geometry systems added to some game engines have moved the same idea to runtime, selecting detail on the fly according to the density of pixels on the screen.

Implicit representations, which express a surface not directly but as a particular value of a function defined at every point in space, have also gained im-

portance. Signed distance fields and volumetric grids are far better suited than polygon meshes to storing shapes whose topology changes, such as a splashing liquid or dispersing smoke. Museth (2013) showed that very high resolution volumes can be stored and processed efficiently with a hierarchical data structure that uses memory only in the occupied regions of space. When a renderable polygon mesh has to be obtained from an implicit representation, the algorithm of Lorensen and Cline (1987) and its descendants remain in use.

### *3.1.2 Procedural Modeling*

As virtual worlds have grown in scale, modeling every object by hand has become impossible. Procedural modeling rests on the idea of defining content not directly but through the rules that generate it. Parish and Müller (2001) presented a system that generates road networks from input maps such as population density and terrain elevation, then subdivides the resulting blocks into lots and fills them with buildings. P. Müller et al. (2006) extended this work with a shape grammar that divides a building mass into floors, floors into facade elements and facade elements into details such as windows and doors, showing that a small number of rules can yield great architectural variety. In later years this approach also became the basis of commercial tools for generating scenes at the scale of a city.

Smelik et al. (2014) surveyed procedural methods developed for terrain, vegetation, waterways, road networks, cities and buildings, and stressed that the main difficulty in the field is control: rules are good at producing variety, but it is often far from intuitive which parameter a designer should change, and how, to obtain a particular result. This difficulty led to tools in which rules are defined through node-based visual interfaces and the designer can steer the result interactively. Today, procedural methods are used less as a replacement for hand modeling than as a complementary layer: the artist shapes the key elements of a scene by hand, and rules fill in the rest.

### *3.1.3 Capturing Models from the Real World*

The third, and increasingly important, route to modeling is the digitization of existing objects and spaces through measurement. In their review of image-based 3D modeling methods, Remondino and El-Hakim (2006) noted that photogrammetry had advantages over laser scanning in cost and portability, but that a fully automatic and reliable workflow did not yet exist at the time. In the years that followed, this gap was largely closed by progress in computer vision.

Snavely and colleagues showed that camera positions and a sparse 3D point cloud could be recovered automatically from unordered collections of photographs gathered from the internet and taken with different cameras at different times (Snavely et al., 2006). The robustness and scalability of this method, known as structure from motion, were improved by subsequent work, and the general-purpose system presented by Schönberger and Frahm (2016) became a widely used reference tool in research and practice. Szeliski (2022) explains in detail the methods on which this process relies, such as feature matching, camera calibration and dense reconstruction.

As photogrammetry became accessible, it was also more readily adopted outside the entertainment industry. Westoby et al. (2012) showed that terrain models obtained with a consumer-grade camera could reach an accuracy comparable to terrestrial laser scanning, and their study was among those that helped spread the method in the geosciences. In fields such as archaeology, cultural heritage documentation, construction and mining, photogrammetry is now a frequently used measurement tool. In game and film production, rocks, tree trunks, building surfaces and even actors' faces are modeled by scanning them from photographs, and libraries of scanned materials and objects are becoming important inputs to production pipelines.

Cheaper depth sensors have had a similar effect. Newcombe and colleagues showed that data from a consumer-grade depth camera could be fused into a volumetric model in real time while the position of the camera was tracked simultaneously (Newcombe et al., 2011). Being able to scan a room in minutes with a handheld device put 3D capture within reach of non-specialist users. Some smartphones and tablets now include similar depth sensors.

### 3.2 Animation

#### *3.2.1 Keyframes, Skeletons and Skin*

The basic method of computer animation is keyframing, in which the animator defines poses at particular moments and the frames in between are computed by interpolation. Parent (2012) gives a systematic account of methods ranging from interpolation curves to inverse kinematics and from hierarchical modeling to behavioral animation. The technique itself has changed little over the years; the traditional principles described by Lasseter (1987) still form the basis of animation training today. What has changed is the complexity of the character rigs that animators work with and the fact that these rigs can be evaluated at interactive rates.

To make a character animatable, an articulated skeleton is first placed inside it, and each vertex of the surface is then bound to one or more bones with weights. In the simplest binding method, the position of a vertex is computed as the weighted average of the transformations of the bones it is attached to. This method is fast, but where joints bend or twist strongly it leads to loss of volume and to a twisting artifact known as the candy-wrapper effect. Lewis et al. (2000) showed that these defects could be removed by interpolating, in pose space, shapes that the artist had corrected by hand in particular poses. Kavan et al. (2008), in turn, developed a method based on blending transformations with dual quaternions that requires no additional effort from the artist and largely prevents the loss of volume. Placing the skeleton and assigning the weights long remained manual work; Baran and Popović (2007) took an important step toward automating the process by showing that a generic skeleton could be fitted automatically to a given character mesh and that the binding weights could be computed by simulating heat diffusion.

Facial animation is a specialty of its own, because the human face moves through the subtle interplay of many muscles and viewers are extremely sensitive to the slightest inconsistency in a face. A common approach in industry is to model the shape of the face separately for particular expressions and to obtain any expression as a weighted sum of these shapes. Lewis et al. (2014) examined in depth the mathematical structure, construction methods and limitations of these models, known as blendshapes. In high-budget productions, the facial model of a single character may consist of hundreds of blendshapes.

### *3.2.2 Motion Capture*

Producing realistic human motion with keyframes is both difficult and time-consuming. Motion capture overcomes this by measuring a performer's movement and transferring it to a digital character. The systems that were common at the beginning of the century relied on tracking reflective markers attached to the performer's suit with many infrared cameras. These systems were highly accurate but required a dedicated studio and expensive equipment. Moeslund et al. (2006) surveyed research on markerless motion capture based solely on video and showed that, at the time, these methods still faced significant difficulties in uncontrolled conditions.

The situation changed with depth sensors and machine learning. Shotton and colleagues trained a classifier that assigns each pixel of a single depth image to a body part and succeeded in estimating joint positions in real time on

consumer hardware, without the need for tracking over time (Shotton et al., 2011). Having entered millions of homes, this system showed that body movement could be captured without wearable equipment. In the years that followed, deep learning also brought major progress in estimating human pose from ordinary camera images. The system developed by Cao et al. (2021) can detect the body, hand and facial keypoints of several people in an ordinary image in real time, in two dimensions. These two-dimensional estimates, however, are not in themselves a 3D motion record; turning them into 3D motion requires additional steps, such as triangulating points obtained from several cameras or fitting a 3D body model to the estimates. The statistical human body model that Loper et al. (2015) learned from thousands of body scans is one of the target models frequently used in such fitting; because it is compatible with standard skeleton-based skinning, the model can be used together with existing production software. These developments have reduced the need for markers or dedicated depth sensors in some applications, but in productions that require high accuracy they have not fully replaced marker-based, multi-camera systems.

Reusing captured motion is a separate research topic. In a game, a character has to respond instantly to the player's input, so a long pre-recorded motion clip cannot simply be played back as it is. Kovar et al. (2002) proposed a method that automatically finds the points at which clips in a motion database can transition into one another and organizes them into a graph. By searching for paths on this graph, continuous motion can be generated in the desired direction and at the desired speed. The method can be counted among the conceptual forerunners of the motion matching techniques that later came into use in the games industry, which search the database every frame for the most suitable continuation.

Crowd scenes are a special case of motion generation. On a battlefield where thousands of soldiers clash, or in a stadium full of spectators, animating each individual by hand is impossible. The line of behavioral animation that began with Reynolds's (1987) flocking model developed in the early 2000s into agent-based crowd systems, in which each individual perceives its surroundings, makes its own decisions and selects an appropriate motion clip. In these systems, the director steers not individuals but rules and the overall flow, and the resulting collective behavior emerges from the interaction of individual decisions. The same approach is used in games to simulate pedestrians and traffic, and in architecture to analyze evacuation and circulation.

### *3.2.3 Physically Based Simulation*

Animating phenomena such as clothing, hair, water, smoke, fire and destruction by hand is impossible in practice; these elements are produced by numerical simulation of the laws of physics. In fluid simulation, Stam's (1999) stable solver was taken as a starting point, and Fedkiw and colleagues added a correction that restores the small-scale vortices damped out by the numerical method, obtaining detailed smoke motion even on coarse grids (Fedkiw et al., 2001). Bridson (2015) explains in detail the theory and practice of grid-based and particle-based fluid solvers, from tracking the water surface to interaction with solid bodies. From the mid-2000s, these methods became common tools for ocean, flood and explosion scenes.

For rigid and deformable bodies, two different sets of requirements emerged. Film puts physical accuracy and detail first, whereas games need methods that remain stable under all conditions and run very fast. M. Müller et al. (2007) proposed an approach that works directly on positions rather than on forces and velocities and satisfies constraints iteratively. Although physically approximate, the approach is stable and easy to control, and it has become one of the methods most frequently used in games for simulating cloth, ropes and soft bodies. Macklin et al. (2014) extended the same framework to unify liquids, gases, rigid bodies and cloth under a single particle representation, so that interactions between different materials could be handled naturally.

On the film side, methods from continuum mechanics have been adapted to simulate complex materials. Stomakhin and colleagues adapted the material point method, which combines particles with a background grid, to the snow scenes of *Frozen* (2013), capturing the characteristic behavior of snow, such as compression, fracture and clumping, with a single model (Stomakhin et al., 2013). The method was later adapted to other materials as well. In cloth simulation, the implicit integration approach of Baraff and Witkin (1998), with countless refinements, became one of the approaches on which many production systems rely.

## 3.3 Rendering

### *3.3.1 Physically Based Rendering*

One of the most prominent developments in twenty-first-century rendering is the widespread adoption of the physically based approach in both offline and real-time production. The essence of this approach is to simulate the circula-

tion of light in a scene in accordance with the laws of physics and to describe materials with models that obey principles such as the conservation of energy. Pharr et al. (2023) present the theory of this approach together with the source code of a complete renderer, and their book serves as a basic reference in both academia and industry.

The benefit of the physically based approach in production is not only that images look more realistic. With earlier methods, each material was tuned for a particular lighting condition, and whenever the lighting of a scene changed, the materials had to be reworked. A material that obeys physical principles, by contrast, behaves far more consistently under different lighting conditions; this reduces the artist's workload and makes it easier for assets prepared by different people to sit together harmoniously in the same scene. The microfacet model of Cook and Torrance (1982), reformulated in terms of intuitive parameters such as roughness and metalness, has become the basis of common material definitions in both film and game production.

The simulation of light that penetrates beneath the surface also became practical in this period. In materials such as skin, marble, milk and wax, light enters the surface, scatters inside the material and exits at another point; when this phenomenon is ignored, such materials look hard and lifeless. Jensen et al. (2001) proposed a model that approximates this scattering efficiently. The model played an important role in achieving convincing human skin for digital characters and was widely used in film production.

Lighting itself has also become measurement-based. Debevec and Malik's (1997) method for capturing high dynamic range images, together with Debevec's (1998) idea of using these images as light sources, became one of the common practices of visual effects production from the early 2000s. On set, a full spherical high dynamic range photograph of the surroundings is taken from the position of the scene; this image is wrapped around the virtual object and used as a source of both illumination and reflections. In this way, the digital object largely inherits the direction, color and intensity of the light on the real set. Working with high dynamic range data requires the image to be compressed into a limited range of brightness before it is displayed; the operator of Reinhard and colleagues (2002), inspired by the zone system of photography, is among the best-known methods developed for this purpose. The arrival of high dynamic range displays, however, is changing the role of this compression step and requires production pipelines to be organized so as to preserve wide ranges of color and brightness.

### 3.3.2 Path Tracing and Sampling

Path tracing as defined by Kajiya (1986) is conceptually simple: for each pixel, random light paths are traced from the camera and the energy they carry is averaged. The difficulty of the method is noise, a natural consequence of random sampling. Halving the noise requires four times as many samples, so much of the research has been directed at obtaining less noise with the same number of samples, that is, at choosing samples more intelligently. Techniques such as multiple importance sampling, which come from the same line of research as the work of Veach and Guibas (1997), are among the core components of many production renderers, while more complex methods such as bidirectional path tracing have remained an option for particular lighting conditions.

Rendering participating media such as smoke, fog, clouds and fire is a separate challenge, because in these media light is scattered and absorbed not at particular points, as on surfaces, but throughout the volume. Novák et al. (2018) surveyed the Monte Carlo methods developed for volumetric light transport; these methods make it possible to treat volumes within the same path tracing framework as surfaces. Effects such as the soft glow created by multiple scattering inside clouds can be achieved more convincingly with this unified approach.

### 3.3.3 Denoising and Reconstruction

The second factor that made path tracing practical in production was progress in denoising. The idea is simple: instead of taking samples until the image has fully converged, a noisy image is rendered with fewer samples and the remaining noise is removed with a filter. Zwicker et al. (2015) reviewed methods that distribute samples adaptively across the image and then reconstruct it, showing that auxiliary information such as surface normals, depth and texture color plays an important role in guiding the filter. Bako et al. (2017) proposed that the filter kernels be predicted for each pixel by a convolutional neural network rather than designed by hand, and showed that a network trained on the frames of one animated film could generalize to other productions. Learning-based denoising was soon adopted in many production renderers, and, as described in the previous chapter, similar ideas became one of the prerequisites of real-time ray tracing.

### 3.3.4 Comparing the Two Worlds

Table 2 offers a qualitative comparison of the typical characteristics of offline and real-time rendering around 2000 and in the 2020s. The table is based

not on a measurement of all practice in the industry but on selected production and research examples documented in the sources. As the table shows, the gap between the two areas has not disappeared, but its nature has changed. At the beginning of the century, the two areas used different algorithms and different material models; in the 2020s they largely share the same physical framework, and the difference between them stems mainly from the number of samples that can be afforded per pixel and, consequently, from how much of the work falls to reconstruction.

**Table 2**

*A Qualitative Comparison of Offline and Real-Time Rendering Across Two Periods*

| Feature | Offline (c. 2000) | Offline (2020s) | Real time (c. 2000) | Real time (2020s) |
|---|---|---|---|---|
| Core algorithm | Micropolygon rasterization; selective ray tracing | Largely path tracing | Fixed-function rasterization | Mainly rasterization; selective or hybrid use of hardware-accelerated ray tracing |
| Indirect illumination | Precomputation and hand-placed fill lights | Simulated light transport | Precomputed light maps | Precomputation combined with dynamic methods |
| Material model | Empirical, scene-specific | Physically based | Fixed empirical model | Mostly physically based |
| Frame time | Minutes to hours | Minutes to hours | About 16–33 ms | About 8–33 ms (depending on target frame rate) |
| Denoising and reconstruction | Not used or limited | Classical and learning-based filters | Not used | Temporal accumulation, spatiotemporal filters and learning-based upscaling |

*Note.* This is a qualitative comparison based on selected sources; the 2020s columns reflect typical approaches documented in the sources, not all practice in the industry. Sources: Akenine-Möller et al. (2018); Bako et al. (2017); Bitterli et al. (2020); Christensen and Jarosz (2016); Christensen et al. (2018); Keller et al. (2015); Schied et al. (2017); Xiao et al. (2020).

This convergence has had concrete consequences in the application areas discussed in the next chapter. The fact that a material prepared for film can look much the same in a game engine is one of the technical prerequisites for forms of production, such as virtual production, that bring the tools of the two worlds together.

# 4. THREE-DIMENSIONAL GRAPHICS IN FILM, DIGITAL GAMES AND DESIGN

The hardware and methods discussed in the previous chapters did not develop in isolation but in response to the needs of particular fields of production. A film studio's need to light millions of strands of hair, a game studio's need to draw an open world sixty times a second, or an architectural practice's wish to walk a client through a building that does not yet exist have steered the research agenda at least as much as theoretical curiosity. This chapter considers the three major areas in which 3D graphics is applied, namely film, digital games and design, and examines the particular needs of each, the solutions each has adopted and the ways in which those solutions have spread to the other areas.

## 4.1 Film

### *4.1.1 From Visual Effects to Digital Cinema*

At the beginning of the century, computer-generated imagery in film was still regarded as a special effect, a solution for shots that could not be filmed any other way. Twenty-five years later, digital intervention has become routine in high-budget productions. Operations such as replacing skies, extending sets, multiplying crowds, de-aging actors and swapping stunt performers' faces are routine practices that audiences often do not notice. The handbook edited by Okun and Zwerman (2020) documents how every stage of this process, from previsualization to final compositing, is carried out in the industry.

This spread has also fed theoretical debates about the nature of cinema. Manovich (2001) argued that with digital tools the photographic record lost its privileged position and that cinema was moving closer to a form of animation in which every frame can be painted. Prince (2012) offers a view that balances this position: digital visual effects are, for the most part, used not to abandon reality but to build perceptually convincing images by imitating the physical behavior of light, lenses and motion. Indeed, as this chapter will show, the main direction of technical development in film has been to make physical simulation ever more complete.

### *4.1.2 The Path Tracing Revolution*

In the first decade of the century, film renderers relied on an architecture that processed the scene by splitting it into small pieces, which allowed them to handle very complex geometry with limited memory. Global illumination had no natural place in this architecture, and reflections and soft shadows required separate precomputation passes. Christensen and Jarosz (2016) describe in detail the complexity of this arrangement and the move of studios to path tracing over the course of a decade. As the authors present it, the transition was the result not of a single invention but of several developments coming together, among them the growth in processor core counts and memory, progress in ray tracing acceleration structures and the maturing of sampling techniques.

The most compelling argument for the transition concerned production rather than art. Keller et al. (2015) emphasize that simplicity was decisive in the adoption of path tracing: because a single algorithm in principle covers all lighting effects, artists no longer have to manage precomputation passes, fix light leaks or place fake fill lights, and can light a scene as they would a real set. In addition, because path tracing converges progressively, the artist can see a noisy but complete preview within seconds. In production environments where artists' time is more valuable than computing time, this makes the method, more expensive to compute but simpler to use, economically sensible as well.

A series of papers published in the same journal issue in 2018 documented the architectures of the leading production renderers and made visible the common ground the industry had reached. Georgiev et al. (2018) explained the design principles of a renderer that relies on optimized unidirectional path tracing rather than complex algorithms and puts predictability first. Christensen et al. (2018) described how a rasterization-based system with a thirty-year history was transformed into a path tracing architecture. Burley et al. (2018) introduced an architecture, first used on *Big Hero 6* (2014), in which large numbers of rays are sorted and processed in batches, allowing very large and complex scenes to be handled efficiently. Fascione et al. (2018) presented a design that represents light by wavelengths rather than three color components and separates shading from light transport. Although these systems differ in their details, all of them share the same theoretical framework, the Monte Carlo solution of the rendering equation. A method seen thirty years earlier as a theoretical curiosity had become the common foundation of the major production renderers.

### *4.1.3 Digital Humans and the Uncanny Valley*

The hardest test for computer graphics in film has been the creation of a convincing human face. People are expert readers of faces; the slightest inconsistency in skin tone, eye movement or the timing of an expression is perceived as disturbing. This phenomenon has long been explained with the concept of the uncanny valley, put forward by the roboticist Masahiro Mori in 1970: as an artificial being comes to resemble a human, our affinity for it increases, but when the resemblance is almost complete, small imperfections produce a sudden feeling of revulsion (Mori et al., 1970/2012). Reactions to some animated films that tried to use realistic human characters in the first decade of the century brought the concept into wide use.

Measurement-based methods of face capture and rendering have made an important contribution to reducing the uncanny effect and increasing photorealism in particular examples of digital humans. Debevec and colleagues showed that the reflectance properties of a face could be recorded completely by photographing it under light arriving from many different directions, and that with these data the face could be re-rendered under any lighting (Debevec et al., 2000). In later years this capture setup was developed further, making it possible to obtain pore-level geometry and layered skin reflectance maps. Starting from a scan of this kind, Alexander et al. (2010) prepared the digital face of an actress, placed it over her live-action performance and showed that a photorealistic digital actor could be achieved. The subsurface scattering model of Jensen et al. (2001) and performance capture systems that record facial expressions from the actor are among the other components of this progress. Even so, individual successes do not mean that the problem of the uncanny valley has been overcome for all digital humans and under all viewing conditions; particularly for faces in motion and for prolonged viewing, the perceptual result can only be assessed through user studies.

Performance capture became widely known through the character of Gollum in *The Lord of the Rings: The Two Towers* (2002), and by the end of the decade it had matured in productions such as *Avatar* (2009), in which digital characters played leading roles. In this method, the performer's body movement and facial expression are recorded together and transferred to the digital character; animators then adapt the data to the character's different anatomy and correct it by hand where necessary. The method blurred the line between acting and animation and raised the question of whom a digital character's per-

formance belongs to. In recent years, the same techniques have been used to de-age actors and to bring deceased actors back to the screen, applications that have raised the ethical debates discussed in Chapter 6.

### *4.1.4 Animated Features and the Search for Style*

Since the beginning of the century, films produced entirely by computer have largely replaced hand-drawn animation, particularly in US feature animation. In the first two decades, the visual language of these films was defined by increasing physical realism: each new production offered more convincing hair, cloth, water or snow than the one before. The material point method mentioned in the previous chapter, developed for the snow scenes of *Frozen* (2013), shows that some simulation techniques arose directly from the needs of particular films (Stomakhin et al., 2013). Cartoonish character design combined with surfaces and lighting approaching photographic realism became the defining style of the period.

From the late 2010s, a marked reaction against this trend developed. Some productions used 3D tools to create images resembling comic book printing, watercolor painting or hand-drawn animation. One of the most frequently cited examples of this trend, *Spider-Man: Into the Spider-Verse* (2018), combined two-dimensional elements such as halftone patterns and color offsets reminiscent of comic book printing with 3D scenes, and in some sequences deliberately animated movement at a lower frame rate. The attention such productions received was a reminder that physical realism is not a necessity but one stylistic option among many. Technically, this diversity does not rest on a single approach: some productions combine a physically based foundation with stylization applied afterward, while others turn to custom lighting and material models, to non-photorealistic rendering methods, or to hybrid use of two- and three-dimensional techniques.

### *4.1.5 Virtual Production*

The rising quality of real-time rendering has led to practices that change the filmmaking process itself. Virtual production is a broad approach that brings real and digital production processes together in a continuum stretching from preproduction through the shoot to postproduction. It can include previsualization, virtual location scouting, the use of virtual cameras, performance capture and real-time compositing during the shoot; green-screen shoots that offer a live preview can also fall within this framework. Its most prominent form in

recent years, however, has been in-camera visual effects. In this practice, actors are filmed in front of large light-emitting diode (LED) walls on which the virtual environment is displayed in real time. The position of the camera is tracked and the image on the wall is updated instantly according to the camera's viewpoint, so that the background appears with the correct perspective shift. This setup has several important advantages: the actors and the director can see the environment they are in, the light emitted by the wall illuminates the actors and the set, reflections on glossy surfaces form largely correctly, and part of the image is already complete during the shoot. On the other hand, because of limitations such as the resolution of the wall, its color range and its viewing angle relative to the camera, some shots may still need correction in postproduction.

Virtual production is the point at which several developments traced in this book intersect. The adoption of physically based materials by game engines has allowed assets prepared for film to look convincing in real time. Debevec's (1998) idea of image-based lighting has found a physical counterpart in the virtual environment lighting the real actor. Camera tracking, for its part, draws on tracking technologies that matured in motion capture and virtual reality. In-camera visual effects using LED walls attracted wide attention with the production of the television series *The Mandalorian*, which began airing in 2019, and have since been used in other series and films. By moving part of the visual effects work from postproduction into preproduction, this practice is affecting production schedules and the division of labor between teams.

## 4.2 Digital Games

### *4.2.1 The Constraints of Real Time*

Digital games are one of the areas in which 3D graphics reaches the widest audiences. What distinguishes games from film is interaction: the image cannot be determined in advance, and every frame is produced anew according to the player's current input. This implies a strict time budget of a few milliseconds per frame, which has to be shared among rendering, physics, artificial intelligence, audio and networking. Gregory (2018) explains in detail how a modern game engine organizes these subsystems and how the game loop is structured.

Under this constraint, game graphics long developed as an art of approximation. Most of the techniques documented by Akenine-Möller et al. (2018) aim to produce a cheap imitation of an expensive physical phenomenon that looks close enough. Their success depends on knowing, through experience, which errors human perception is sensitive to and which it ignores. Low-frequency

errors in indirect illumination, for example, often go unnoticed, whereas flickering at the edges of shadows or sudden changes during movement are immediately apparent.

In the first fifteen years of the century, game graphics went through several distinct stages. With the arrival of programmable shaders, per-pixel lighting and normal mapping became widespread, and characters and surfaces gained considerably in detail. Deferred shading architectures then allowed the number of lights in a scene to be increased independently of geometric complexity. In the mid-2010s, the industry moved rapidly to physically based material models, a transition driven by a rationale similar to that of the shift in film: materials that behave more consistently under different lighting conditions make production easier. After 2018, hardware-accelerated ray tracing began to be used first for individual effects such as reflections and shadows (Haines & Akenine-Möller, 2019); in the 2020s, some high-budget games offered optional rendering modes based largely on path tracing. The resampling and denoising approaches on which such modes rely were discussed in Chapter 2.

### *4.2.2 Content Creation and Procedural Methods*

The price of rising graphical quality has been rising content production costs. With each new hardware generation, the level of detail players expect has increased, and content creation has accordingly become a major item of labor and cost in game development. In their survey of procedural content generation for games, Hendrikx et al. (2013) identified this cost pressure as the main motivation and organized the content that can be generated into a layered taxonomy ranging from textures and vegetation to buildings, maps, quests and entire game worlds.

Procedural generation serves two different purposes in games. The first is to speed up the artist's work during development: terrain is shaped by erosion simulation, forests are distributed according to rules, buildings are derived from grammars, and the artist corrects the result by hand. The second is to generate content while the game is running, so that each player encounters a different world and the game can, in principle, offer endless variety. Togelius et al. (2011) classified approaches in which content is searched for according to a fitness function and optimized with evolutionary methods, while Summerville et al. (2018) examined the use of machine learning models that learn from existing game content to generate new content. The main challenge of runtime generation is quality assurance: the designer must be confident that content they

have never seen will be playable, balanced and engaging.

The second response to the cost of content has been scanning. Photogrammetry, discussed in the previous chapter, began to be used in many game productions from the mid-2010s. Large libraries created by scanning real rocks, trees and surfaces have reduced the need for studios to produce every asset from scratch. The very high polygon counts of scanned data initially required manual simplification; in some workflows, virtualized geometry systems have eased this step as well.

### *4.2.3 Character Animation*

Character animation in games faces a different problem from that in film: the character must respond instantly and convincingly to unforeseen player input and environmental conditions. The traditional solution is to link clips prepared by animators through a state machine. But as the number of actions a character can perform grows, the number of states and transitions increases unmanageably. The motion graphs of Kovar et al. (2002) were an early answer to this problem; in industry, from the mid-2010s, the motion matching method, which searches a large motion database every frame for the pose best suited to the current state and the desired trajectory, began to be used (Holden et al., 2020).

Neural networks have opened up new possibilities here as well. Holden et al. (2017) showed that a neural network whose weights vary with the phase of the gait cycle could animate a character walking, running, jumping and crouching over rough terrain in real time while using very little memory. Starke et al. (2019) extended this approach to cover actions that require interaction with the environment, such as sitting down, carrying objects and passing through doors. Holden et al. (2020) proposed a method that combines the controllability of motion matching with the low memory requirements of neural networks, replacing the database search with networks trained to imitate it. The fact that some of this work came directly from the research divisions of game studios shows how porous the boundary between industry and academia is in this area.

### *4.2.4 Game Technology Beyond Games*

As game engines have become accessible and capable, they have come to be used outside games. Architectural visualization, automotive design reviews, military and medical training simulations, broadcast graphics and the virtual production discussed in the previous section are among these uses. The reason for this spread is not only image quality: game engines offer interaction, phys-

ics, networked multi-user operation and support for virtual reality out of the box. Applications that once each required separate and expensive specialized software can now be developed on the same general-purpose tool. As a result, game development skills are finding demand in industries outside games as well.

## 4.3 Design

### *4.3.1 Architecture and Building Information Modeling*

In the design disciplines, the role of 3D graphics differs from its role in entertainment: the aim is not only to produce an image but to create the description of a real object or building that will be made. In her theoretical examination of the place of digital tools in the design process, Oxman (2006) argued that the computer had ceased to be merely a tool for speeding up drawing and had become the very medium in which form is generated, evaluated and transformed, and within this framework she distinguished models such as formation, generative design and performance-based design. The spread of parametric design tools is one of the visible results of this transformation: instead of drawing a single form, the designer defines a network of relationships that produces countless alternatives when its parameters are changed.

In the building sector, one of the most important developments of the period has been building information modeling (BIM). Sacks et al. (2018) define this approach not merely as a method of 3D drawing but as a process in which all the components of a building are defined, together with their geometric and non-geometric properties, in a single consistent model through which all stakeholders collaborate, from design to operation. In such a model, a wall is not just a surface but an object whose material, cost, thermal performance and construction sequence are known. Operations such as clash detection, quantity take-off and energy analysis can be derived directly from the model.

On the visualization side, physically based rendering has changed the quality of architectural presentation. While the physically accurate lighting simulation pioneered by Ward (1994) continues to be used for daylight analysis, real-time engines allow clients to move freely inside a building that has not yet been built. In product visualization, computer-generated images are increasingly used in catalogs and advertising; rather than photographing every color and trim option of a product one by one, producing them from its digital model is, in many cases, a more flexible approach.

### *4.3.2 From Digital Model to Physical Object*

The conversion of a 3D model into a physical object has gained a new dimension with the development of additive manufacturing technologies. Gibson et al. (2021) classify these technologies, in which an object is built up by adding material layer by layer, into process families ranging from material extrusion to powder bed fusion, and emphasize that the common starting point of all of them is the digital 3D model. Ngo et al. (2018) assess the advantages of the method, such as freedom in producing complex geometries, suitability for customized production and low material waste, together with its difficulties, such as the directional dependence of mechanical properties and limits on size. Additive manufacturing has made it possible to realize lattice structures and organic forms obtained through topology optimization that cannot be produced by conventional methods, which in turn has increased the importance of geometry processing in design tools. Operations such as mesh repair and watertightness checking, covered by Botsch et al. (2010), are routine steps in preparing a model for printing.

### *4.3.3 Digital Twins*

The useful life of the digital model in design has also grown longer. The concept of the digital twin envisages that the virtual counterpart of a physical product or system is kept up to date not only during design but throughout manufacture and use, fed by data coming from the physical asset. Grieves and Vickers (2017) trace the origins of the concept to product lifecycle management and argue that a digital twin can help detect unforeseen behavior in complex systems in the virtual environment before the physical product exists. In their survey of industrial applications of the concept, Tao et al. (2019) identify modeling, data fusion and the interaction between physical and virtual space as the main research topics.

The relationship between digital twins and 3D graphics goes beyond visualization. A factory-scale twin requires mechanical designs, robot programs, logistics and building models to be brought together in a single scene, which is one of the main reasons for the adoption in industry of the open scene description standards discussed in Chapter 2. Physical simulation and real-time rendering are used in these environments to test robots and sensing systems virtually.

### *4.3.4 Cultural Heritage and Medicine*

Alongside design and engineering, two fields in which 3D graphics is used

for documentation and analysis deserve particular mention. In the preservation of cultural heritage, recording buildings, excavation sites and museum objects in three dimensions has become widespread. As Remondino and El-Hakim (2006) note, the aim in this field is to produce not merely a visually convincing model but a document with measurable geometric accuracy, which is why photogrammetry and laser scanning are often used together. The raw data consist of millions of points and must be cleaned, aligned, merged and simplified before they become a usable model; software such as the open-source tool developed by Cignoni et al. (2008) was created to meet needs of this kind. 3D records serve as a reference in the restoration of works damaged by natural disasters or conflict, and they widen access through virtual museums and immersive exhibitions.

In medicine, 3D visualization is an important tool for making sense of the slice data produced by computed tomography and magnetic resonance imaging. The isosurface extraction algorithm of Lorensen and Cline (1987) and the direct volume rendering method of Levoy (1988) lie at the root of visualization techniques in this field. In the twenty-first century, the integration of visualization with planning and treatment is also being explored: models derived from a patient's scan data can be used in surgical planning, and patient-specific guides and implants can be produced by additive manufacturing (Ngo et al., 2018), while aligning the model with the patient through augmented reality during surgery is an active area of research and development. How widespread these applications are, and what clinical benefit they bring, must be assessed through clinical validation studies, separately from how convincing the visual output may be. In this context, what matters is not the beauty of the image but its accuracy and the absence of delay.

### *4.3.5 Immersive Environments*

Virtual and augmented reality have found a place in design as tools for evaluation and collaboration. Azuma (1997) defined augmented reality as systems that combine the real and the virtual, are interactive in real time and are registered in three dimensions, and this definition remains valid today. Slater and Sanchez-Vives (2016) emphasize that the distinctive feature of immersive virtual reality is the illusion of being there that it creates in the user, and they examine applications of this feature ranging from education and psychological therapy to cultural heritage and industrial design. When assessing the scale of an architectural space or the ergonomics of a car interior, immersive viewing at

full scale provides information that an image on a screen cannot.

The prediction that these technologies will develop into persistent, shared and interconnected virtual environments has been widely discussed in recent years. Mystakidis (2022) defines such environments as a multi-user, persistent construct that merges physical reality with digital virtuality, and notes that the concept rests on the convergence of virtual, augmented and mixed reality technologies. Although it is uncertain to what extent this prediction will come true, the infrastructure it requires, namely large-scale 3D content creation, real-time rendering and interoperable data formats, is a direct extension of the developments discussed in this book.

### 4.4 Assessment

The three application areas started from different requirements but have converged markedly over time. Film prioritized physical accuracy, games prioritized speed and design prioritized precision; in the 2020s, all three draw on physically based materials, scanned assets, similar data formats and, increasingly, similar real-time engines. This convergence has made it easier to transfer a solution developed in one area to the others: open data formats developed by film studios have found a place in game and design tools, game engines on film sets, and the parametric approach of the design world in the procedural tools of both entertainment fields. The layer added to this common ground over the past decade, and one that affects all three areas at once, is artificial intelligence, the subject of the next chapter.

## 5. ARTIFICIAL INTELLIGENCE AND NEW PRODUCTION TECHNOLOGIES

For a long time, computer graphics and artificial intelligence developed largely independently of each other. Graphics researchers tried to produce images from a numerical description of the world, while computer vision researchers tried to extract a description of the world from images; these two problems, one running forward and the other in reverse, were for the most part tackled by different communities. From the mid-2010s onward, this separation disappeared quickly. Deep neural networks were first used to speed up individual steps of the rendering pipeline, then took a place in the representation of the scene itself, and finally came to play a role in generating content directly. This chapter takes up these three stages in turn and discusses the capture and manufacturing technologies that developed alongside them.

### 5.1 Deep Learning Fundamentals and Their Encounter with Graphics

Deep learning is a family of methods in which the mapping from input to output is expressed by a multilayer neural network whose parameters are learned from example data. LeCun et al. (2015) emphasize that the distinguishing feature of these methods is their ability to learn increasingly abstract representations from raw data on their own, without hand-designed features. The textbook by Goodfellow et al. (2016) offers a comprehensive account of the foundations of the field, from convolutional and recurrent networks to optimization methods. Three factors were decisive in making these methods work in practice: the emergence of large labeled datasets, improvements in training methods and, as described in Chapter 2, the computing power provided by graphics processing units. The combination of these three factors was plainly visible in the ImageNet work mentioned in Chapter 2 and first presented in 2012 (Krizhevsky et al., 2017).

The first contributions of deep learning to graphics took the form of replacing expensive steps in the existing pipeline with learned approximations. The most successful example is denoising, discussed in earlier chapters. The methods developed by Bako et al. (2017) for film production and by Chaitanya

et al. (2017) for interactive applications substantially reduced the number of samples that path tracing requires. In the same period, neural networks also began to be used in character animation and in upscaling (Holden et al., 2017; Xiao et al., 2020). What these applications have in common is that the neural network is a component placed inside the traditional pipeline: the scene is still described with polygons and textures, the image is still produced by simulating light transport, and the network takes over only one step of the process.

### 5.2 Generative Models

The second wave came with models capable of producing new images. Generative adversarial networks (GANs) were introduced in the approach Goodfellow and colleagues presented in 2014, in which two networks are trained against each other: one produces fake samples, while the other tries to tell real from fake. The text cited here is the version of this line of research published in *Communications of the ACM* in 2020 (Goodfellow et al., 2020). As a result of this contest, the generator learns to produce new samples that resemble the distribution of the training data. The method developed rapidly in the years that followed; as one example of these developments, Karras et al. (2019) used an architecture that allows features at different scales of the generated image to be controlled separately, producing high-resolution human faces that are hard to distinguish from real photographs. Photorealistic portraits of people who do not exist were among the first examples to show a wide audience what generative models could do.

From the standpoint of graphics, generative adversarial networks matter above all in their conditional forms. Isola et al. (2017) proposed a general framework for translating one type of image into another, showing that transformations as varied as edge drawings to photographs, label maps to street scenes and daytime to nighttime images could all be learned with the same architecture, using nothing more than paired example images. This approach suggested that rendering itself might be treated as a translation problem: a simply rasterized image of a rough 3D scene could be given as input and a photorealistic output learned from it.

Among the most widespread approaches to generative modeling today are diffusion models (Po et al., 2024). Ho et al. (2020) showed that a model which learns to reverse the process of adding noise to an image step by step, that is, one that starts from pure noise and cleans it up iteratively, can produce high-quality images. It is generally accepted that these models can be trained more stably

than adversarial networks and cover the data distribution more broadly; their weakness is that generation requires a large number of steps (Po et al., 2024). Rombach et al. (2022) substantially lowered the computational cost by running the diffusion process in a previously learned, compressed latent space rather than in pixel space, and showed that conditioning inputs such as text could be supplied to the model flexibly through cross-attention layers. Once text-to-image systems could run on consumer hardware, these models quickly reached a very large number of users.

In 3D graphics production, these models first found a place in supporting two-dimensional tasks. These include rapidly multiplying concept art, producing texture and material maps, generating environment images and extending existing images. The real challenge, however, is for generative models to produce 3D content directly, and this cannot be separated from developments in representing the scene with neural networks.

## 5.3 Neural Scene Representations

### *5.3.1 Learned Implicit Surfaces*

Chapter 3 explained that a shape can be stored as a polygon mesh, a parametric surface or a volumetric grid. Deep learning added a radically different option to this list: a shape can be stored in the weights of a neural network that takes the coordinates of a point in space as input and returns some property of that point as output. Park et al. (2019) proposed a network that learns a continuous function giving the signed distance from a point to the nearest surface; by adding a latent code to the network, they showed that a single model could represent an entire class of shapes and complete shapes from missing or noisy data. In the same year, Mescheder et al. (2019) proposed defining the surface as the decision boundary of a classifier that predicts whether a point lies inside or outside an object. In both approaches the representation is not tied to a fixed output grid; the surface can be extracted by querying the network at different densities. The geometric detail that can be obtained, however, is limited by the capacity of the network, the training data and the accuracy of the learned representation; sampling more densely does not create detail the model has not learned.

These studies are part of a broader line of research on processing 3D data with neural networks. Because images are stored on a regular pixel grid, they are a natural fit for convolutional networks; 3D data has no such single natural structure. Qi et al. (2017) introduced a network architecture that takes an unordered set of points directly as input and produces results independent of

the order of the points, providing a basic building block for the classification and segmentation of point clouds. Xie et al. (2022) gathered fields represented by coordinate-based neural networks under the heading of neural fields, and showed that this approach is used to represent quantities as diverse as shape, appearance, motion and physical properties.

### *5.3.2 Neural Radiance Fields*

The application of neural representations that attracted the widest attention emerged in novel view synthesis. The problem is this: given a certain number of photographs of a scene, can we compute how the scene would look from a viewpoint from which no photograph was taken? Levoy and Hanrahan's (1996) light fields were an early answer to this question, but they required very dense sampling. Lombardi et al. (2019) showed that a volumetric representation of dynamic scenes could be learned from multi-camera recordings and rendered with differentiable ray marching, achieving good results on hair, smoke and semi-transparent materials, where surface-based reconstruction struggles.

Mildenhall et al. (2020) gave this idea a form of striking simplicity. In their method, called a neural radiance field (NeRF), the scene is represented by a single multilayer network that takes a position in space and a viewing direction as input and returns the volume density at that point and the color emitted in that direction. The color of a pixel is computed by sampling the network along the ray passing through that pixel and combining the results according to the classical rules of volume rendering (Max, 1995). Because this whole process is differentiable, the network's weights can be optimized simply by comparison with photographs whose camera positions are known; no 3D supervision is needed. The results reached a level of realism, including glossy reflections and fine geometric detail, that surpassed the novel view synthesis methods of the time. The camera positions themselves are obtained using the structure-from-motion systems described in Chapter 3 (Schönberger & Frahm, 2016).

The original form of the method had important limitations: training a single scene took hours or even days, rendering a single image took seconds, and the method worked only on static scenes. In the years that followed, each of these limitations became the subject of intensive research. Barron et al. (2021) eliminated the aliasing artifacts that appear when a scene is viewed from different distances by sampling along cones rather than rays. T. Müller et al. (2022) cut training time from hours to seconds by replacing a large network with a very small one combined with learnable feature vectors stored in a multiresolution

hash table. Tewari et al. (2022) surveyed a large body of work on dynamic scenes, relighting, editability and generalization, showing how much the field had expanded within a few years.

### *5.3.3 3D Gaussian Splatting*

Interestingly, the most effective answer to the speed problem of neural radiance fields came from a method that moved away from neural networks. Kerbl et al. (2023), in a method they called 3D Gaussian splatting, proposed representing the scene as the sum of a large number of 3D Gaussian distributions, each carrying a position, a shape, an opacity and a view-dependent color. These primitives are projected onto the screen, sorted by depth and blended; for this step the authors developed a fast, visibility-aware, tile-based renderer that runs on the graphics processing unit. The idea of splatting point-like primitives onto the screen was not new; twenty years earlier, Zwicker et al. (2001) had developed a similar approach for high-quality rendering of point-based surfaces. What was new was that the parameters of these primitives are optimized from photographs through a differentiable rendering pipeline, and that primitives are adaptively added and pruned during training. The result is image quality comparable to the best neural radiance field methods at real-time frame rates and high resolution.

The method's impact was very rapid. Because the representation is explicit, meaning that the position and properties of each primitive are directly accessible, it is in principle easy to select primitives, delete them and combine them with other scenes. Even so, game engines do not support this representation natively as they do polygonal assets; a suitable renderer or plug-in that performs the sorting and blending has to be added to the engine. Nor do editing, animation and relighting follow automatically from the representation being explicit; they require separate methods and are active research topics. Still, the fact that scenes captured from photographs could be rendered in real time in this way meant that the method was soon taken up in experimental and commercial applications through browser-based viewers and engine plug-ins. Subsequent work has sought to improve the geometric accuracy of the representation, for example by reducing volumetric Gaussian primitives to oriented two-dimensional disks (Huang et al., 2024). The method also carries an important conceptual lesson: the success of recent years stems not only from neural networks as such, but from making the rendering process differentiable and optimizing the representation directly from images.

Table 3 compares the main forms of 3D representation discussed in this book.

**Table 3**

*A Comparison of the Main 3D Scene Representations*

| Representation | Strengths | Limitations | Key source |
|---|---|---|---|
| Polygon mesh | Direct compatibility with hardware; mature editing and animation tools | Approximation of smooth surfaces; difficulty with changes in topology | Botsch et al. (2010) |
| Subdivision surface | Smooth organic shapes from few control points | Very fine detail requires additional displacement | Catmull and Clark (1978) |
| Sparse volume grid | Phenomena with changing topology; smoke and liquids | Memory requirements; resolution limits on sharp surfaces | Museth (2013) |
| Learned implicit surface | Not tied to a fixed output grid; compact representation of shape classes | Detail limited by network capacity and training data; query cost; hard to edit directly | Park et al. (2019) |
| Neural radiance field | High realism from photographs; view-dependent appearance | Slow training and rendering in its original form (accelerated by later methods); hard to edit and relight | Mildenhall et al. (2020); T. Müller et al. (2022) |
| 3D Gaussian primitives | Real-time rendering; explicit, directly accessible representation | Requires a dedicated renderer; high memory use; no exact surface geometry; editing tools not yet mature | Kerbl et al. (2023) |

*Note.* Strengths and limitations apply to each representation in the form described in the cited sources; later work has reduced some of the limitations. The mature tool support available for polygon meshes does not yet have an equivalent for the newer representations. Compiled by the author from the cited sources.

### *5.3.4 Differentiable Rendering*

The common foundation of the methods mentioned in the previous sections is differentiable rendering. A conventional renderer computes the function that maps scene parameters to an image; a differentiable renderer also computes how small changes in the scene parameters affect pixel values, that is, the sensitivity of the image to those parameters. Gradient-based optimization uses this information to update the scene parameters step by step so that they match the target images more closely; since more than one scene may be consistent with a given image, the result is usually guided by additional assumptions. The difficulty stems from the discontinuities inherent in rasterization: whether or not a triangle covers a pixel is a binary question. The derivative of this visibility function is zero away from the boundary and undefined in the classical sense at the visibility boundary itself, so a gradient usable for optimization cannot be obtained directly. Liu et al. (2019) obtained usable gradients by defining each triangle's contribution to a pixel as a soft probabilistic transition rather than a hard edge, and showed that mesh reconstruction from a single image could be learned with two-dimensional supervision alone. Laine et al. (2020), for their part, developed a library that offers rasterization, interpolation, texture sampling and antialiasing as separate differentiable building blocks, handles the gradients arising from visibility boundaries through the antialiasing step and takes advantage of the speed of graphics hardware. Tewari et al. (2020) classified this kind of combination of classical graphics knowledge with learned components under the name neural rendering, and observed that it constitutes a new field bringing together physical knowledge and learning from data.

## 5.4 Generative 3D Content

The main obstacle to carrying the success of two-dimensional generative models over into three dimensions is data. While two-dimensional models can be trained on billions of image–text pairs, the number of high-quality, labeled 3D models is smaller by orders of magnitude. Researchers have therefore looked for ways to draw on the knowledge contained in two-dimensional models in order to generate 3D content.

One approach in this direction is to build a 3D structure into the generator network. Chan et al. (2022) designed a generative adversarial network that combines an efficient scene representation, made up of features stored on three orthogonal planes, with volume rendering; although the network was trained only on a dataset of individual photographs, it could produce faces that looked

consistent from different angles and whose geometry could be extracted. Another approach is to use a pretrained two-dimensional model as a kind of critic. Poole et al. (2023) proposed rendering a randomly initialized neural radiance field from different angles and updating it in the direction that a text-to-image diffusion model would judge to fit a given text description. In this way, 3D objects could be obtained from a text description alone, without any 3D training data. Lin et al. (2023) both accelerated this approach and raised its level of detail through a two-stage process in which a coarse representation is optimized first and a high-resolution textured mesh afterwards. The shared weakness of these methods is that every new object requires a long optimization process. From 2024 onward, reconstruction models trained on large-scale object datasets that need no separate optimization for each input came to the fore; Hong et al. (2024) presented one such model, which predicts a 3D model of an object from a single image in about five seconds.

The digitization of the human body is an area where generative methods are particularly important. Saito et al. (2019) proposed a pixel-aligned implicit function that infers the detailed 3D geometry and texture of a clothed person from a single photograph, including the back of the person, which is not visible in the image. In face modeling, Blanz and Vetter's (1999) statistical model served for more than twenty years as a common basis for reconstructing faces from a single image; Egger et al. (2020) reviewed in depth the development of this family of models and its integration with deep learning. Thies et al. (2016) showed that the facial expressions in an ordinary video recording could be replaced in real time with those of another person. As much as this work was a technical achievement, it was also an early sign of the social concerns surrounding the production of fake videos; the subject is taken up again in Chapter 6.

Motion generation has gone through a similar transformation. Peng et al. (2018) showed that a character inside a physics simulation could learn, through reinforcement learning, to imitate captured motion clips. In this method the character does not play back the motion recording directly; the learned control policy produces target orientations for each joint, and proportional–derivative (PD) controllers compute from these targets the torques to be applied at the joints. The resulting controller can respond to external disturbances, such as being pushed, and to changing task goals; even highly dynamic movements such as backflips and spin kicks could be obtained in this way. It should be noted that the method is not a musculoskeletal model controlling anatomical muscle activations, but a physics simulation operating at the level of the joints.

This approach can be seen as a data-driven realization of the goal that Witkin and Kass (1988) had articulated thirty years earlier: combining artistic intent with physical plausibility.

The current state of generative 3D methods needs to be assessed with care. The models they produce often have irregular topology, texture layouts unsuited to production and materials with the lighting baked in; they contain neither an animatable skeleton nor clean edge flow. For now, therefore, such outputs find use in work such as background objects, quick previsualization and concept development, while in the production of hero assets they give the artist a starting point rather than replacing the artist's work. Even so, the pace of development suggests that these limitations are not permanent.

### 5.5 New Capture and Production Technologies

Alongside artificial intelligence methods, the technologies that bring the real world into the digital domain and return digital models to the physical world have also matured. These technologies often work hand in hand with learning-based methods.

On the capture side, the photogrammetry and depth sensing discussed in Chapter 3 have combined with neural reconstruction methods to form a new workflow. It has become possible to obtain a navigable 3D scene in a relatively short time from a short video of a place shot on a smartphone (Kerbl et al., 2023; T. Müller et al., 2022). The real-time dense reconstruction approach of Newcombe et al. (2011) is among the important forerunners of the methods that augmented reality devices use to map their surroundings. In the practice known as volumetric video, a performance is recorded with many cameras, yielding a moving 3D recording around which the viewer can walk freely (Lombardi et al., 2019). Markerless methods have made it possible to estimate two-dimensional body, hand and face keypoints from ordinary camera images (Cao et al., 2021); turning these estimates into 3D motion, however, requires further steps such as multiple cameras or fitting to a 3D body model, as explained in Chapter 3.

On the production side, two developments stand out. The first is virtual production, described in Chapter 4, in which real-time rendering has ceased to be a post-production process and has become part of the shoot itself. The second is additive manufacturing. Because a digital model can be turned directly into a physical object, modeling tools have had to define shape not only for the image but also for manufacture (Gibson et al., 2021). In the approach known

as generative design, the designer does not draw the shape; instead, they define the loads, connection points, material and manufacturing constraints, and the software generates many candidate shapes that satisfy these conditions. The organic and lattice-like structures that result can often be produced only by additive methods (Ngo et al., 2018). This cycle of scanning, digital editing and printing is used in applications such as patient-specific implants in medicine and the completion of missing pieces in cultural heritage.

### 5.6 Limitations of Learning-Based Methods

The successes described in this chapter should not overshadow the particular limitations of learning-based methods. The first is dependence on data. A neural network performs well on inputs that resemble the situations it saw in its training data; outside the training distribution its behavior can become unpredictable. A denoiser trained on frames from one animated film may produce unexpected artifacts in a production with a very different visual style. With classical algorithms, by contrast, the ways in which they fail are usually known and can be bounded theoretically.

The second limitation is controllability. In a production setting, an artist wants to be able to determine a particular detail of the result precisely. In a physically based renderer, it is clear what will happen when the intensity of a light is doubled; in a generative model, a small change to the text description can produce an entirely different result. For this reason, systems used in production usually place learned components within a classical framework that can be controlled explicitly (Tewari et al., 2020).

The third limitation concerns consistency. In conventional 3D production, a character looking the same from one shot to the next and an object being geometrically consistent from every angle are directly supported by the shared scene description, since every frame is derived from the same asset definition. Even so, artifacts such as temporal flicker, sampling noise, aliasing at shadow edges and transitions between levels of detail also occur in classical production and need to be controlled separately. In generative models that work in image space, the consistency of object identity and geometry must, in addition, be ensured separately. This is one of the main reasons for the interest in generative models that explicitly incorporate 3D structure (Chan et al., 2022).

Finally, computational cost and reproducibility should be mentioned. Training large models can require computing resources on a scale that most research groups and studios cannot reach, which carries the risk that the direction of

the field will be set by a small number of large organizations. Where training data and model weights are not shared, it also becomes harder to verify results independently. In this respect, the long-standing tradition of open publication and open source in the computer graphics community stands out as something worth preserving.

### 5.7 Assessment

Within a decade, artificial intelligence has gone from being an auxiliary tool in 3D graphics to being one of the field's core methods. This transformation can be summarized in three stages. In the first, neural networks took over individual steps such as denoising and upscaling. In the second, the scene itself became a learned representation that could be optimized directly from photographs. In the third, content began to be generated from high-level inputs such as text or images. These stages did not replace one another but were layered on top of one another; all three may be found together in a production pipeline today.

In their article assessing the place of generative artificial intelligence in artistic production, Epstein et al. (2023) argue that, much like the arrival of photography or of digital editing software, these tools will transform creative practice rather than destroy it, but that they raise serious questions about authorship, copyright, the value of labor and the media ecosystem. From the perspective of 3D graphics, these questions are far from theoretical. The next and final chapter discusses the future of the field, taking up these questions alongside the technical trends.

## 6. THE FUTURE OF THREE-DIMENSIONAL COMPUTER GRAPHICS

Predictions about the future of technology often say more about the concerns of the time in which they are made than about the future itself. The history of computer graphics also counsels caution here: a quarter of a century had to pass after path tracing was defined in theory before it was used in production, whereas neural radiance fields began to appear in commercial applications within a few years of being introduced. Rather than making firm prophecies, therefore, this chapter aims to discuss where the trends traced in the previous chapters are heading, which problems remain unsolved and which non-technical factors may shape the direction of the field.

### 6.1 Convergence in Rendering

One of the main trends emphasized throughout this book is the convergence of offline and real-time rendering. There are strong reasons to expect this trend to continue. Real-time path tracing is now practical on high-end hardware, and every advance in resampling and denoising methods allows more complex lighting to be handled on the same hardware (Bitterli et al., 2020). In the coming years, games can be expected to abandon precomputed lighting, and the many special-purpose techniques that imitate it, in favor of a single unified light transport solver. The simplification that cinema went through a decade ago may thus be repeated in real-time production (Christensen & Jarosz, 2016).

Even so, it would be an oversimplification to claim that the two areas will merge completely. Film production has to deal with extremely large amounts of geometry, texture and volume data, as well as complex participating media, in a single frame (Burley et al., 2018; Novák et al., 2018); this scale can be expected to remain beyond real-time budgets for the foreseeable future. A more realistic expectation is that the same scene description, the same materials and the same family of algorithms will be used in both contexts with different quality settings. The artist will work in a real-time preview, and the final frames will be obtained by running the same solver with a higher sample count. For studios that have had to maintain two separate production pipelines, this would be a

significant gain.

The second dimension of convergence is that an ever larger part of the image is inferred rather than computed. In many real-time applications today, the image that reaches the screen can be reconstructed by learned models from a low-resolution, noisy measurement (Xiao et al., 2020). At the far end of this trend may lie systems in which a conventional renderer provides only rough guidance about the structure of the scene, while the final appearance is produced by a learned model (Tewari et al., 2022). The main obstacles to such an approach are temporal consistency and controllability: a generative model may produce a convincing image in every frame, but keeping those images consistent from frame to frame without flicker, and allowing the artist to determine a particular detail precisely, are separate problems. Hybrid systems that combine the theoretical guarantees of physically based rendering (Pharr et al., 2023) with the flexibility of learned models appear to be the most likely resolution of this tension.

### 6.2 A Plurality of Representations

Polygon meshes have long been the dominant representation, especially in real-time entertainment applications; as seen in Chapters 1 and 3, parametric surfaces in engineering, subdivision surfaces in character modeling and volumetric representations in scientific visualization and effects production have existed alongside them. The developments discussed in Chapter 5 are adding new options, optimized from data, to this plural landscape. Gaussian primitives for scenes captured from photographs, sparse volumes for phenomena with changing topology, learned implicit functions for compact encoding of shape classes and parametric surfaces for precision engineering are now used side by side (Kerbl et al., 2023; Xie et al., 2022). Rather than converging on a single representation, the production pipelines of the future are more likely to evolve toward a plural structure in which the representation suited to each task is chosen and reliable conversions are made between representations.

This plurality raises several open questions. The first is editability. The modeling, sculpting, UV unwrapping and rigging tools developed for polygon meshes over fifty years do not yet have mature counterparts for the new representations. Selecting an object in a captured scene, moving it, changing its shape or redefining its material are active research topics. The second is relighting. Representations optimized from photographs usually bake the lighting present at the time of capture into their appearance, whereas in production an

asset must be usable under different lighting conditions. Separating geometry, materials and lighting from one another, that is, inverse rendering, will remain one of the most important applications of differentiable methods (Laine et al., 2020). The third is standardization. The integration of the new representations with file formats, compression methods and scene description frameworks is still at an early stage; these representations will need to go through the same process of open standardization described in Chapter 2.

The representation of motion is on the threshold of a similar transformation. Rigs based on skeletons and blend shapes (Lewis et al., 2014) hold their place because they are controllable, but learned motion models are taking on more and more of the work. Combining characters that learn to move within a physics simulation (Peng et al., 2018) with approaches that compress large motion databases into learned models (Holden et al., 2020) could allow game characters to respond convincingly to unexpected situations without being tied to prepared clips. The fact that this approach shares methods with control research in robotics suggests that the exchange of knowledge between the two fields will grow.

### 6.3 The Transformation of Content Creation

3D content creation remains one of the most labor-intensive links in the chain. The work of the artists who create models, textures, animations and environments is a major cost item in big-budget game and film productions. The economic impact of artificial intelligence methods can be expected to be felt clearly at this point. The quality problems of today's generated models were noted in the previous chapter; but the ground that two-dimensional generative models covered in a few years (Ho et al., 2020; Rombach et al., 2022), and the rapid progress in adapting these models to 3D generation (Hong et al., 2024; Po et al., 2024), suggest that a similar development is likely in 3D production as well.

The question of how this progress will change the artist's role is more complicated than a simple story of substitution. The history of procedural methods can serve as a guide here. Rule-based generation has been able to produce cities, forests and terrain automatically for more than twenty years; yet, as Smelik et al. (2014) emphasize, designer control has remained decisive in these tools, and part of the artist's work has shifted from modeling individual objects to designing rules and to selecting and correcting results. As Summerville et al. (2018) point out, in learning-based generation too the central problem is not

generation itself but ensuring that the generated content serves the design goals. For generative models to enter production pipelines, there will need to be interfaces through which the artist can steer the result in detail, outputs that can be edited in existing tools and results that are predictable.

The more likely scenario is a change in the distribution of labor. Repetitive and technical tasks such as background assets, variations, levels of detail, UV unwrapping and rigging may be largely automated, while creative direction, stylistic consistency and quality control become even more important. This shift may make it easier for small teams to produce work on a larger scale, but it may also have consequences such as fewer entry-level jobs in the profession. The broadening of access to production tools can be read as the continuation of a long process that began when the engines and open-source software described in Chapter 2 became widely accessible.

### 6.4 Spatial Computing and Immersive Environments

Despite a history of half a century, head-mounted displays are still a technology in development. As Koulieris et al. (2019) note, fundamental problems such as the mismatch between focus and the vergence of the eyes, a narrow field of view, the weight of the device and a resolution that falls short of human visual acuity have been only partly solved. Work continues on varifocal and light field displays, thinner optical designs and high-density microdisplays. In terms of rendering, these devices are extremely demanding; foveated rendering and learning-based reconstruction (Kaplanyan et al., 2019) appear to be the main ways of closing the gap between the number of pixels required and the computing power of a portable device.

If these devices become widespread, the place of 3D graphics in society may change fundamentally. Today, 3D content is largely experienced through a two-dimensional screen, as if looking through a window. Once augmented reality in the sense defined by Azuma (1997) enters everyday use, virtual objects will be perceived as part of the physical environment, and this will require content to adapt instantly to the lighting, geometry and occlusion relationships of the real world. Reconstructing the environment in real time, estimating its lighting and rendering virtual objects accordingly require all of the capture and rendering methods discussed in this book to work at the same time, within the power budget of a portable device.

Persistent and shared virtual environments (Mystakidis, 2022) have attracted intense interest in recent years, but it remains unclear to what extent

the commercial expectations surrounding them will be realized. Whatever the commercial fate of those expectations, the underlying technical requirements are real, and their solutions will be useful in other areas as well: many users seeing the same scene consistently, content that can move between different platforms and very large scenes streamed over the network. The research reviewed by Slater and Sanchez-Vives (2016) shows that the potential value of immersive environments in fields such as education and health is being studied independently of entertainment.

The question of where computation takes place also becomes important in this context. A pair of lightweight glasses cannot be expected to perform high-quality path tracing, so split architectures in which part of the rendering is offloaded to a nearby device or to the cloud are on the agenda. The latency problem identified by Shi and Hsu (2015) is even more critical here, because an image that responds late to head movement causes discomfort. Reprojecting the image sent by the server on the device according to the latest head position, and filling in missing regions with learned methods, are among the possible solutions to this problem.

### 6.5 Simulation, Digital Twins and Scientific Use

In the coming period, the use of 3D graphics technologies outside entertainment may grow faster than their use within it. The spread of the digital twin approach (Grieves & Vickers, 2017; Tao et al., 2019) is increasing the need for continuously updated 3D models of factories, cities, energy infrastructure and buildings. Combining the detailed building data provided by building information modeling (Sacks et al., 2018) with as-built data from photogrammetry and laser scanning makes it possible to maintain a digital counterpart of the built environment that extends from the design stage to operation.

An important development in this area is the use of 3D simulation environments to train artificial intelligence systems. Training autonomous vehicles, robots and perception systems in the real world is expensive, slow and sometimes dangerous. Virtual environments that produce photorealistic images and behave in physically consistent ways can supply these systems with large amounts of labeled data. The cycle mentioned in Chapter 2 thus comes full circle once more: artificial intelligence improves graphics, and graphics trains artificial intelligence. This use brings requirements different from the standards of the entertainment industry; what matters here is not that the image looks pleasing but that it matches, statistically, what the sensor will see in the real world.

Scientific visualization and medicine have been important areas of application since the field began, and they will remain so. The fact that volume rendering methods originated in medical imaging (Levoy, 1988) and, thirty years later, formed the basis of neural radiance fields is a fine example of the unexpected connections between application areas and basic research. The application of neural fields to inverse problems such as reconstruction in medical imaging (Xie et al., 2022) has also opened a new channel through which graphics research can contribute to scientific measurement.

### 6.6 Ethics, Law and Society

The expansion of technical possibilities has made concrete problems that were once regarded as theoretical. Foremost among these is fake content. The ability to imitate a real person's face and voice convincingly (Karras et al., 2019; Thies et al., 2016) undermines public trust in visual evidence and opens the door to abuses ranging from fraud to political disinformation and from non-consensual intimate imagery to character assassination. The solution to the problem cannot be purely technical; even so, provenance standards that make it possible to verify the source of content, the labeling of generated content and methods for detecting fake content are active topics of research and standardization. Binding regulations have also begun to emerge in this area. The European Union's Artificial Intelligence Act, adopted in 2024, requires that synthetic audio, image and video outputs generated by AI systems be marked in a machine-readable format, and that deepfake content which presents real persons, objects or events as authentic be disclosed as artificially generated or manipulated; these transparency obligations are set to apply from August 2026 (European Parliament & Council of the European Union, 2024). This regulation is specific to European Union law; other countries address the subject through different legal frameworks. It is also important for graphics researchers to discuss the possible misuses of the methods they develop at the time of publication.

The second problem area is the right to one's digital likeness. The digital human technologies discussed in Chapter 4 make it technically possible to use an actor's scanned body and captured performance in new scenes without the actor's participation. Practices such as scanning background performers once and using them indefinitely, or bringing deceased artists back to the screen, raise questions about the scope of consent, compensation and the post-mortem status of personality rights. In some countries and industry agreements, these

questions have begun to be addressed through concrete rules. For example, the 2023 television and theatrical contracts of SAG-AFTRA, the union representing performers in the United States, require that a producer creating and using a digital replica of a performer in the course of employment obtain the performer's consent limited to specific projects and pay for its use, and they do not permit blanket consent of a general nature (SAG-AFTRA, n.d.). Contractual provisions of this kind apply only to the industry and country covered by the agreement; the legal framework for personality rights differs from country to country and continues to evolve.

The third problem area is the legal status of the training data and outputs of generative models. These models are often trained on large datasets collected without the explicit permission of the people who created the material. Epstein et al. (2023) note that questions such as recognition for the work of artists who contribute to training data, whether generated outputs can benefit from copyright protection and the consequences of imitating a particular artist's style remain unanswered. In some countries, answers to these questions have begun to be given at the administrative and judicial level. In a report published in January 2025, the U.S. Copyright Office stated that, with current technology, prompts alone do not give a person sufficient control over the output and therefore do not on their own make the person who wrote the prompt its author; on the other hand, human-authored expressive elements perceptible in the output, and the creative selection, coordination or arrangement of material, can be protected (U.S. Copyright Office, 2025). This assessment is an administrative view specific to the United States; other countries address the subject within their own legislation. The situation is similar for 3D content: the legal and ethical position of models that learn from online model libraries and game assets remains unclear in many respects.

Fourth, the environmental cost should be mentioned. It has been emphasized that training large neural network models can consume a considerable amount of energy and that this consumption should be measured and reported in research (Strubell et al., 2019); the hours spent computing a single film frame, or the power consumption of large numbers of gaming devices, can be considered part of the same discussion. In this discussion it should not be overlooked that transistor density, performance, energy efficiency and total energy consumption are separate measures. The slowing of the growth in density that Moore (1965/1998) predicted, and of the power scaling that accompanied it, means that performance gains can no longer be expected from the shrinking of transistors alone, and brings to the fore hardware architectures specialized

for particular tasks (Hennessy & Patterson, 2019). Many of the reconstruction methods discussed in this book aim to achieve a given visual quality with less computation. Although these methods may reduce the energy used per frame, their effect on total energy consumption depends on how they are used and should not be generalized without measurement.

### 6.7 Education and the Profession

The rapid change in the field also puts pressure on education. Twenty years ago, training a 3D artist rested largely on mastering a particular piece of software. In an environment where tools change quickly and repetitive tasks are automated, it is becoming increasingly clear that the knowledge of lasting value consists of foundations independent of any tool: the behavior of light and materials, an understanding of form and composition, the principles of motion and the ability to build a narrative. The detailed production processes documented by Okun and Zwerman (2020) likewise suggest that, alongside knowledge of tools, knowledge of workflows and communication within the team matter in this field.

The traditional division between technical and artistic skills is also weakening. Artists who can write shaders, designers who work with procedural tools and programmers who understand what artists need play important roles in production teams, building bridges between different specialties. As described in Chapter 4, the use of game engines across different industries has widened the range of settings in which these skills apply. As generative tools spread, it can be expected that the ability to evaluate a result, steer it and make it part of a whole will become more important than the ability to produce it. For education programs to adapt to this change, they will need to give students both solid foundations and the habit of continuous learning.

### 6.8 Open Research Problems

In addition to the topics touched on in the previous sections, a few technical problems facing the field deserve separate mention. The first is consistency across scales. The appearance of a fabric arises from the interaction of individual fibers with light, and that of a lock of hair from the interaction of individual strands; viewed from a distance, however, computing these details one by one is both unnecessary and expensive. Representations that preserve appearance in the transition from near to far, that is, that correctly summarize micro-scale structure at the macro scale, are still an active research topic. Generalizing to geometry, materials and volumes the idea of prefiltering that Williams (1983)

proposed for textures lies at the heart of this problem.

The second problem is the integrated simulation of human characters. Skin, hair, clothing and muscle movement are today mostly handled by separate systems, and the interaction between them is tuned by hand. Once statistical body models (Loper et al., 2015), face models (Egger et al., 2020) and physics simulation are brought together in a single coherent framework, the production of digital humans will become considerably simpler. In real-time applications, convincing facial expression and eye contact are critical issues that will determine the quality of remote communication in immersive environments.

The third problem is the gap between capture and creation. Scenes captured from photographs are extremely realistic but can only reproduce what already exists; scenes made by artists can be edited without limit, but reaching the same level of realism takes great effort. Once a captured scene can be edited, relit and animated as freely as a scene modeled by an artist, the distinction between these two modes of production will largely disappear.

The fourth problem is evaluation. There is no objective answer to the question of how good an image or an animation is. Pixel-level error metrics agree with human perception only in part, and learned perceptual metrics carry their own biases. With generative methods, the problem is even more pronounced, because there is no single correct result to compare against. User studies are reliable but expensive. Developing metrics that take the properties of the human visual system into account (Koulieris et al., 2019) matters both for making sound comparisons in research and for optimizing perception-based methods such as foveated rendering.

### 6.9 Assessment

The trends discussed in this chapter point in one direction: 3D graphics is ceasing to be a type of content produced by specialists with dedicated tools and is becoming a general-purpose infrastructure that extends from capturing the real world to building virtual environments, and from scientific measurement to industrial design. The speed and shape of this transformation will depend as much on economic, legal and social factors as on technical progress. That a method is possible does not mean it will be adopted, and its adoption does not guarantee that it will be used well. What will shape the future of the field is, on the one hand, work on the technical problems that remain unsolved and, on the other, collective decisions about for whose benefit and under what rules these powerful tools will be used.

## CONCLUSION

This book has examined the development of 3D computer graphics in the first quarter of the twenty-first century under the headings of hardware, methods, applications and artificial intelligence. In conclusion, the findings set out across the chapters can be gathered under a few general observations.

The first observation concerns continuity. Most of the methods that became widespread in the twenty-first century had been defined in theory in the twentieth. The rendering equation and path tracing were set out in 1986 (Kajiya, 1986), the microfacet-based physical reflectance model in 1982 (Cook & Torrance, 1982) and subdivision surfaces in 1978 (Catmull & Clark, 1978), while the optical models of volume rendering had been systematically compiled by the mid-1990s (Max, 1995). What these methods needed in order to spread in production was not so much a new theory as sufficient computing power and the engineering expertise that made that power usable. This observation matters because it reminds us that ideas set aside today as too expensive or impractical may prove decisive tomorrow. The fact that neural radiance fields rest on volume rendering, and 3D Gaussian primitives on point splatting methods from twenty years earlier, are the most recent examples of this continuity.

The second observation is the decisive role of hardware. The transformation of the graphics processing unit from a fixed-function accelerator into a programmable and then a general-purpose parallel processor is one of the most important developments of the period (Nickolls et al., 2008; Owens et al., 2008). This transformation made possible not only real-time graphics but also the rise of deep learning; deep learning, in turn, came back to graphics and changed the field's basic methods, from denoising to scene representation. This mutual reinforcement between hardware and algorithms is an indispensable key to understanding how the field has developed.

The third observation is convergence. At the turn of the century, film, games and design used different tools, different data formats and different algorithms. In the 2020s, all three increasingly share physically based materials, similar light transport methods, scanned assets and open data standards (Akenine-Möller et al., 2018; Pharr et al., 2023). Virtual production, the use of game engines in architecture and engineering, and digital twin applications are concrete results

of this convergence. Convergence has eased the flow of knowledge and labor between fields and has sped up the transfer of a solution found in one field to the others.

The fourth observation concerns a change of methodological direction. Manual production has increasingly given way to measurement, simulation and learning. Models are reconstructed from photographs (Schönberger & Frahm, 2016), motion is captured from performers or learned within physics simulations (Peng et al., 2018), light is computed through physical simulation and content can increasingly be created with generative models (Hong et al., 2024; Poole et al., 2023). This change has not done away with the artist's role, but it has transformed it: from a craftsperson who specifies every detail by hand toward a director who steers, selects and corrects measurement, simulation and generative tools.

The fifth and final observation is the widening of the field's boundaries. 3D graphics is no longer merely a technique for producing images. Combined with computer vision, it has become a means of bringing the real world into the digital domain; combined with additive manufacturing, a means of turning digital shapes into physical objects; and combined with simulation, a means of training artificial intelligence systems. This expansion has also increased the field's social responsibility. Issues such as fake content, rights to one's digital likeness, the legal status of training data and energy consumption are problems that need to be addressed at the center of technical research, not at its margins (Epstein et al., 2023).

In light of these observations, the argument set out at the beginning of the book can be restated. The development of 3D computer graphics in the twenty-first century is the product of the combined working of three tendencies: toward physical accuracy, toward real time and toward data-driven methods. Physical accuracy has provided a common language, real-time performance has made that language interactive and data-driven methods have made both accessible. The tools that have emerged where these three tendencies meet seem almost limitless when compared with the effort it took, half a century ago, to rotate a wireframe cube in a laboratory.

Yet the basic questions of the field have not changed. How should a shape be represented, how can motion be made convincing, and how does light travel through a scene? Each generation has given new answers to these questions with its own tools, and each answer has raised new questions. Some of the answers that hold as this book is being written may already be out of date by the

time it reaches the reader. What endures are the questions themselves and the shared effort of the researchers, engineers and artists who keep asking them. The future of 3D computer graphics will depend on that effort being carried forward with a sense of responsibility as well as technical skill.

## GLOSSARY

**3D Gaussian splatting**: a method that represents a scene with a large number of 3D Gaussian distributions and renders them in real time by projecting them onto the screen.

**Additive manufacturing**: the production of an object from a digital model by adding material layer by layer; also known as 3D printing.

**Augmented reality (AR)**: systems in which virtual content is combined with a real environment in real time and registered with it in three dimensions.

**Blendshape**: in facial animation, a shape representing a particular expression; weighted sums of blendshapes produce new expressions.

**Building information modeling (BIM)**: a process in which the components of a building are defined, with their geometric and non-geometric properties, in a single model.

**Bump mapping**: a texturing technique that creates an impression of roughness by perturbing the surface normal without changing the geometry.

**Denoising**: estimating a clean image from a noisy image produced with a small number of samples.

**Differentiable rendering**: rendering in which the sensitivity of the image to the scene parameters is also computed, so that it can be used in optimization.

**Diffusion model**: a family of generative models that produce new samples by learning to remove noise step by step, starting from pure noise.

**Digital twin**: a virtual counterpart of a physical product or system that is kept up to date with data coming from it.

**Foveated rendering**: producing the image at full detail only in the region the user is looking at, and at lower detail in the periphery.

**Generative adversarial network (GAN)**: a family of generative models in which two networks, a generator and a discriminator, are trained against each other.

**Global illumination**: taking into account not only direct light but also the indi-

rect light reflected between surfaces.

**Graphics processing unit (GPU)**: a processor made up of a large number of cores, designed for graphics and parallel computation.

**Image-based lighting**: the use of high-dynamic-range images measured in a real environment to illuminate virtual objects.

**Keyframe**: a frame in which the animator directly sets the pose of an object or character, with the in-between frames computed by interpolation.

**Light field**: a representation that stores the distribution of light in a scene by position and direction in a regular structure made up of many images.

**Motion capture**: measuring a performer's movement and transferring it to a digital character.

**Motion matching**: an animation method based on selecting, at every frame, the continuation from a large motion database that best fits the current situation.

**Neural radiance field (NeRF)**: a method that represents a scene with a neural network returning density and color for a given position and viewing direction.

**Path tracing**: a form of ray tracing that solves the rendering equation by the Monte Carlo method, through random sampling of light paths.

**Physically based rendering (PBR)**: an approach to rendering in which light transport and materials are simulated with models consistent with the principles of physics.

**Polygon mesh**: a representation that defines a surface as a structure made up of vertices, edges and faces.

**Procedural modeling**: defining content not directly but through the rules and algorithms that generate it.

**Rasterization**: a rendering method based on projecting geometric primitives onto the screen and determining the pixels they cover.

**Ray tracing**: a rendering method based on following the interactions with surfaces of rays sent from the camera into the scene.

**Rendering**: computing a two-dimensional image from a numerical description of a scene.

**Rendering equation**: an integral equation that expresses the light leaving a surface point in a given direction as the sum of emitted and scattered light.

**Shader**: a small program that runs on the graphics processing unit for elements such as vertices, pixels or rays.

**Shadow mapping**: determining shadows using a depth image rendered from the point of view of the light source.

**Signed distance field (SDF)**: a function that gives, for every point in space, its distance to the nearest surface, with opposite signs inside and outside.

**Skinning**: binding the vertices of a character's surface to the bones of a skeleton with weights.

**Structure from motion (SfM)**: jointly estimating camera positions and scene geometry from photographs taken from different viewpoints.

**Subdivision surface**: a smooth surface obtained by iteratively refining a coarse polygon mesh according to specific rules.

**Uncanny valley**: the phenomenon whereby artificial figures that closely, but not completely, resemble humans cause discomfort in viewers.

**Virtual production**: carrying out physical and digital production processes together, from preproduction to postproduction, using real-time tools.

## REFERENCES

Akeley, K. (1993). RealityEngine graphics. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 109–116). ACM. https://doi.org/10.1145/166117.166131

Akenine-Möller, T., Haines, E., Hoffman, N., Pesce, A., Iwanicki, M., & Hillaire, S. (2018). *Real-time rendering* (4th ed.). A K Peters/CRC Press. https://doi.org/10.1201/b22086

Alexander, O., Rogers, M., Lambeth, W., Chiang, J.-Y., Ma, W.-C., Wang, C.-C., & Debevec, P. (2010). The Digital Emily project: Achieving a photorealistic digital actor. *IEEE Computer Graphics and Applications, 30*(4), 20–31. https://doi.org/10.1109/MCG.2010.65

Appel, A. (1968). Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference* (pp. 37–45). ACM. https://doi.org/10.1145/1468075.1468082

Azuma, R. T. (1997). A survey of augmented reality. *Presence: Teleoperators and Virtual Environments, 6*(4), 355–385. https://doi.org/10.1162/pres.1997.6.4.355

Bako, S., Vogels, T., McWilliams, B., Meyer, M., Novák, J., Harvill, A., Sen, P., DeRose, T., & Rousselle, F. (2017). Kernel-predicting convolutional networks for denoising Monte Carlo renderings. *ACM Transactions on Graphics, 36*(4), Article 97. https://doi.org/10.1145/3072959.3073708

Baraff, D., & Witkin, A. (1998). Large steps in cloth simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 43–54). ACM. https://doi.org/10.1145/280814.280821

Baran, I., & Popović, J. (2007). Automatic rigging and animation of 3D characters. *ACM Transactions on Graphics, 26*(3), Article 72. https://doi.org/10.1145/1276377.1276467

Barron, J. T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., & Srinivasan, P. P. (2021). Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 5835–5844). IEEE. https://doi.org/10.1109/ICCV48922.2021.00580

Bitterli, B., Wyman, C., Pharr, M., Shirley, P., Lefohn, A., & Jarosz, W. (2020). Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics, 39*(4), Article 148. https://doi.org/10.1145/3386569.3392481

Blanz, V., & Vetter, T. (1999). A morphable model for the synthesis of 3D faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 187–194). ACM. https://doi.org/10.1145/311535.311556

Blinn, J. F. (1977). Models of light reflection for computer synthesized pictures. *ACM SIGGRAPH Computer Graphics, 11*(2), 192–198. https://doi.org/10.1145/965141.563893

Blinn, J. F. (1978). Simulation of wrinkled surfaces. *ACM SIGGRAPH Computer Graphics, 12*(3), 286–292. https://doi.org/10.1145/965139.507101

Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., & Lévy, B. (2010). *Polygon mesh processing*. A K Peters/CRC Press. https://doi.org/10.1201/b10688

Bridson, R. (2015). *Fluid simulation for computer graphics* (2nd ed.). A K Peters/CRC Press. https://doi.org/10.1201/9781315266008

Buck, I., Foley, T., Horn, D., Sugerman, J., Fatahalian, K., Houston, M., & Hanrahan, P. (2004). Brook for GPUs: Stream computing on graphics hardware. *ACM Transactions on Graphics, 23*(3), 777–786. https://doi.org/10.1145/1015706.1015800

Burley, B., Adler, D., Chiang, M. J.-Y., Driskill, H., Habel, R., Kelly, P., Kutz, P., Li, Y. K., & Teece, D. (2018). The design and evolution of Disney's Hyperion renderer. *ACM Transactions on Graphics, 37*(3), Article 33. https://doi.org/10.1145/3182159

Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E., & Sheikh, Y. (2021). OpenPose: Realtime multi-person 2D pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 43*(1), 172–186. https://doi.org/10.1109/TPAMI.2019.2929257

Catmull, E., & Clark, J. (1978). Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design, 10*(6), 350–355. https://doi.org/10.1016/0010-4485(78)90110-0

Chaitanya, C. R. A., Kaplanyan, A. S., Schied, C., Salvi, M., Lefohn, A., Nowrouzezahrai, D., & Aila, T. (2017). Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics, 36*(4), Article 98. https://doi.org/10.1145/3072959.3073601

Chan, E. R., Lin, C. Z., Chan, M. A., Nagano, K., Pan, B., De Mello, S., Gallo, O.,

Guibas, L., Tremblay, J., Khamis, S., Karras, T., & Wetzstein, G. (2022). Efficient geometry-aware 3D generative adversarial networks. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 16102–16112). IEEE. https://doi.org/10.1109/CVPR52688.2022.01565

Christensen, P., Fong, J., Shade, J., Wooten, W., Schubert, B., Kensler, A., Friedman, S., Kilpatrick, C., Ramshaw, C., Bannister, M., Rayner, B., Brouillat, J., & Liani, M. (2018). RenderMan: An advanced path-tracing architecture for movie rendering. *ACM Transactions on Graphics, 37*(3), Article 30. https://doi.org/10.1145/3182162

Christensen, P. H., & Jarosz, W. (2016). The path to path-traced movies. *Foundations and Trends in Computer Graphics and Vision, 10*(2), 103–175. https://doi.org/10.1561/0600000073

Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., & Ranzuglia, G. (2008). MeshLab: An open-source mesh processing tool. In *Eurographics Italian Chapter Conference* (pp. 129–136). The Eurographics Association. https://doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136

Cook, R. L., & Torrance, K. E. (1982). A reflectance model for computer graphics. *ACM Transactions on Graphics, 1*(1), 7–24. https://doi.org/10.1145/357290.357293

Crassin, C., Neyret, F., Sainz, M., Green, S., & Eisemann, E. (2011). Interactive indirect illumination using voxel cone tracing. *Computer Graphics Forum, 30*(7), 1921–1930. https://doi.org/10.1111/j.1467-8659.2011.02063.x

Debevec, P. (1998). Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 189–198). ACM. https://doi.org/10.1145/280814.280864

Debevec, P., Hawkins, T., Tchou, C., Duiker, H.-P., Sarokin, W., & Sagar, M. (2000). Acquiring the reflectance field of a human face. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 145–156). ACM. https://doi.org/10.1145/344779.344855

Debevec, P. E., & Malik, J. (1997). Recovering high dynamic range radiance maps from photographs. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 369–378). ACM. https://doi.org/10.1145/258734.258884

DeRose, T., Kass, M., & Truong, T. (1998). Subdivision surfaces in character animation. In *Proceedings of the 25th Annual Conference on Comput-*

*er Graphics and Interactive Techniques* (pp. 85–94). ACM. https://doi.org/10.1145/280814.280826

Drebin, R. A., Carpenter, L., & Hanrahan, P. (1988). Volume rendering. *ACM SIGGRAPH Computer Graphics, 22*(4), 65–74. https://doi.org/10.1145/378456.378484

Egger, B., Smith, W. A. P., Tewari, A., Wuhrer, S., Zollhöfer, M., Beeler, T., Bernard, F., Bolkart, T., Kortylewski, A., Romdhani, S., Theobalt, C., Blanz, V., & Vetter, T. (2020). 3D morphable face models—Past, present, and future. *ACM Transactions on Graphics, 39*(5), 1–38. https://doi.org/10.1145/3395208

Epstein, Z., Hertzmann, A., Akten, M., Farid, H., Fjeld, J., Frank, M. R., Groh, M., Herman, L., Leach, N., Mahari, R., Pentland, A., Russakovsky, O., Schroeder, H., & Smith, A. (2023). Art and the science of generative AI. *Science, 380*(6650), 1110–1111. https://doi.org/10.1126/science.adh4451

European Parliament & Council of the European Union. (2024). *Regulation (EU) 2024/1689 of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)*. Official Journal of the European Union, L series, 12 July 2024. http://data.europa.eu/eli/reg/2024/1689/oj

Fascione, L., Hanika, J., Leone, M., Droske, M., Schwarzhaupt, J., Davidovič, T., Weidlich, A., & Meng, J. (2018). Manuka: A batch-shading architecture for spectral path tracing in movie production. *ACM Transactions on Graphics, 37*(3), Article 31. https://doi.org/10.1145/3182161

Fedkiw, R., Stam, J., & Jensen, H. W. (2001). Visual simulation of smoke. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 15–22). ACM. https://doi.org/10.1145/383259.383260

Garland, M., & Heckbert, P. S. (1997). Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 209–216). ACM. https://doi.org/10.1145/258734.258849

Georgiev, I., Ize, T., Farnsworth, M., Montoya-Vozmediano, R., King, A., Van Lommel, B., Jimenez, A., Anson, O., Ogaki, S., Johnston, E., Herubel, A., Russell, D., Servant, F., & Fajardo, M. (2018). Arnold: A brute-force production path tracer. *ACM Transactions on Graphics, 37*(3), Article 32. https://doi.org/10.1145/3182160

Gibson, I., Rosen, D., Stucker, B., & Khorasani, M. (2021). *Additive manufacturing technologies* (3rd ed.). Springer. https://doi.org/10.1007/978-3-030-56127-7

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S.,

Courville, A., & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM, 63*(11), 139–144. https://doi.org/10.1145/3422622

Goral, C. M., Torrance, K. E., Greenberg, D. P., & Battaile, B. (1984). Modeling the interaction of light between diffuse surfaces. *ACM SIGGRAPH Computer Graphics, 18*(3), 213–222. https://doi.org/10.1145/964965.808601

Gouraud, H. (1971). Continuous shading of curved surfaces. *IEEE Transactions on Computers, C-20*(6), 623–629. https://doi.org/10.1109/T-C.1971.223313

Gregory, J. (2018). *Game engine architecture* (3rd ed.). A K Peters/CRC Press. https://doi.org/10.1201/9781315267845

Grieves, M., & Vickers, J. (2017). Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In F.-J. Kahlen, S. Flumerfelt, & A. Alves (Eds.), *Transdisciplinary perspectives on complex systems* (pp. 85–113). Springer. https://doi.org/10.1007/978-3-319-38756-7_4

Guenter, B., Finch, M., Drucker, S., Tan, D., & Snyder, J. (2012). Foveated 3D graphics. *ACM Transactions on Graphics, 31*(6), 1–10. https://doi.org/10.1145/2366145.2366183

Haines, E., & Akenine-Möller, T. (Eds.). (2019). *Ray tracing gems: High-quality and real-time rendering with DXR and other APIs*. Apress. https://doi.org/10.1007/978-1-4842-4427-2

Hendrikx, M., Meijer, S., Van Der Velden, J., & Iosup, A. (2013). Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications, 9*(1), 1–22. https://doi.org/10.1145/2422956.2422957

Hennessy, J. L., & Patterson, D. A. (2019). A new golden age for computer architecture. *Communications of the ACM, 62*(2), 48–60. https://doi.org/10.1145/3282307

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems 33* (pp. 6840–6851). Curran Associates. https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html

Holden, D., Kanoun, O., Perepichka, M., & Popa, T. (2020). Learned motion matching. *ACM Transactions on Graphics, 39*(4). https://doi.org/10.1145/3386569.3392440

Holden, D., Komura, T., & Saito, J. (2017). Phase-functioned neural networks for character control. *ACM Transactions on Graphics, 36*(4), Article 42. https://doi.org/10.1145/3072959.3073663

Hong, Y., Zhang, K., Gu, J., Bi, S., Zhou, Y., Liu, D., Liu, F., Sunkavalli, K., Bui, T., & Tan, H. (2024). LRM: Large reconstruction model for single image to 3D. In *The Twelfth International Conference on Learning Representations (ICLR 2024)*. https://mlanthology.org/iclr/2024/hong2024iclr-lrm/

Hoppe, H. (1996). Progressive meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (pp. 99–108). ACM. https://doi.org/10.1145/237170.237216

Huang, B., Yu, Z., Chen, A., Geiger, A., & Gao, S. (2024). 2D Gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 Conference Papers* (pp. 1–11). ACM. https://doi.org/10.1145/3641519.3657428

Hughes, J. F., van Dam, A., McGuire, M., Sklar, D. F., Foley, J. D., Feiner, S. K., & Akeley, K. (2014). *Computer graphics: Principles and practice* (3rd ed.). Addison-Wesley.

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 5967–5976). IEEE. https://doi.org/10.1109/CVPR.2017.632

Jensen, H. W. (1996). Global illumination using photon maps. In X. Pueyo & P. Schröder (Eds.), *Rendering techniques '96* (pp. 21–30). Springer. https://doi.org/10.1007/978-3-7091-7484-5_3

Jensen, H. W., Marschner, S. R., Levoy, M., & Hanrahan, P. (2001). A practical model for subsurface light transport. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 511–518). ACM. https://doi.org/10.1145/383259.383319

Kajiya, J. T. (1986). The rendering equation. *ACM SIGGRAPH Computer Graphics, 20*(4), 143–150. https://doi.org/10.1145/15886.15902

Kaplanyan, A. S., Sochenov, A., Leimkühler, T., Okunev, M., Goodall, T., & Rufo, G. (2019). DeepFovea: Neural reconstruction for foveated rendering and video compression using learned statistics of natural videos. *ACM Transactions on Graphics, 38*(6), 1–13. https://doi.org/10.1145/3355089.3356557

Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4396–4405). IEEE. https://doi.org/10.1109/CVPR.2019.00453

Kavan, L., Collins, S., Žára, J., & O'Sullivan, C. (2008). Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics, 27*(4),

1–23. https://doi.org/10.1145/1409625.1409627

Keller, A., Fascione, L., Fajardo, M., Georgiev, I., Christensen, P., Hanika, J., Eisenacher, C., & Nichols, G. (2015). The path tracing revolution in the movie industry. In *ACM SIGGRAPH 2015 Courses* (pp. 1–7). ACM. https://doi.org/10.1145/2776880.2792699

Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023). 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics, 42*(4), Article 139. https://doi.org/10.1145/3592433

Koulieris, G. A., Akşit, K., Stengel, M., Mantiuk, R. K., Mania, K., & Richardt, C. (2019). Near-eye display and tracking technologies for virtual and augmented reality. *Computer Graphics Forum, 38*(2), 493–519. https://doi.org/10.1111/cgf.13654

Kovar, L., Gleicher, M., & Pighin, F. (2002). Motion graphs. *ACM Transactions on Graphics, 21*(3), 473–482. https://doi.org/10.1145/566654.566605

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM, 60*(6), 84–90. https://doi.org/10.1145/3065386

Laine, S., Hellsten, J., Karras, T., Seol, Y., Lehtinen, J., & Aila, T. (2020). Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics, 39*(6), Article 194. https://doi.org/10.1145/3414685.3417861

Lasseter, J. (1987). Principles of traditional animation applied to 3D computer animation. *ACM SIGGRAPH Computer Graphics, 21*(4), 35–44. https://doi.org/10.1145/37402.37407

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444. https://doi.org/10.1038/nature14539

Levoy, M. (1988). Display of surfaces from volume data. *IEEE Computer Graphics and Applications, 8*(3), 29–37. https://doi.org/10.1109/38.511

Levoy, M., & Hanrahan, P. (1996). Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (pp. 31–42). ACM. https://doi.org/10.1145/237170.237199

Lewis, J. P., Anjyo, K., Rhee, T., Zhang, M., Pighin, F., & Deng, Z. (2014). Practice and theory of blendshape facial models. In *Eurographics 2014 – State of the Art Reports* (pp. 199–218). The Eurographics Association. https://doi.org/10.2312/egst.20141042

Lewis, J. P., Cordner, M., & Fong, N. (2000). Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceed-*

*ings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 165–172). ACM. https://doi.org/10.1145/344779.344862

Lin, C.-H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.-Y., & Lin, T.-Y. (2023). Magic3D: High-resolution text-to-3D content creation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 300–309). IEEE. https://doi.org/10.1109/CVPR52729.2023.00037

Lindholm, E., Kilgard, M. J., & Moreton, H. (2001). A user-programmable vertex engine. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 149–158). ACM. https://doi.org/10.1145/383259.383274

Liu, S., Chen, W., Li, T., & Li, H. (2019). Soft rasterizer: A differentiable renderer for image-based 3D reasoning. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 7707–7716). IEEE. https://doi.org/10.1109/ICCV.2019.00780

Lombardi, S., Simon, T., Saragih, J., Schwartz, G., Lehrmann, A., & Sheikh, Y. (2019). Neural volumes: Learning dynamic renderable volumes from images. *ACM Transactions on Graphics, 38*(4), Article 65. https://doi.org/10.1145/3306346.3323020

Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., & Black, M. J. (2015). SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics, 34*(6), 1–16. https://doi.org/10.1145/2816795.2818013

Lorensen, W. E., & Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics, 21*(4), 163–169. https://doi.org/10.1145/37402.37422

Macklin, M., Müller, M., Chentanez, N., & Kim, T.-Y. (2014). Unified particle physics for real-time applications. *ACM Transactions on Graphics, 33*(4), 1–12. https://doi.org/10.1145/2601097.2601152

Manovich, L. (2001). *The language of new media*. MIT Press.

Mark, W. R., Glanville, R. S., Akeley, K., & Kilgard, M. J. (2003). Cg: A system for programming graphics hardware in a C-like language. *ACM Transactions on Graphics, 22*(3), 896–907. https://doi.org/10.1145/882262.882362

Marschner, S., & Shirley, P. (2021). *Fundamentals of computer graphics* (5th ed.). A K Peters/CRC Press. https://doi.org/10.1201/9781003050339

Max, N. (1995). Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics, 1*(2), 99–108. https://doi.

org/10.1109/2945.468400

Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., & Geiger, A. (2019). Occupancy networks: Learning 3D reconstruction in function space. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4455–4465). IEEE. https://doi.org/10.1109/CVPR.2019.00459

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2020). NeRF: Representing scenes as neural radiance fields for view synthesis. In A. Vedaldi, H. Bischof, T. Brox, & J.-M. Frahm (Eds.), *Computer vision – ECCV 2020* (pp. 405–421). Springer. https://doi.org/10.1007/978-3-030-58452-8_24

Moeslund, T. B., Hilton, A., & Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding, 104*(2–3), 90–126. https://doi.org/10.1016/j.cviu.2006.08.002

Moore, G. E. (1998). Cramming more components onto integrated circuits. *Proceedings of the IEEE, 86*(1), 82–85. https://doi.org/10.1109/JPROC.1998.658762 (Original work published 1965)

Mori, M., MacDorman, K. F., & Kageki, N. (2012). The uncanny valley [From the field]. *IEEE Robotics & Automation Magazine, 19*(2), 98–100. https://doi.org/10.1109/MRA.2012.2192811 (Original work published 1970)

Müller, M., Heidelberger, B., Hennix, M., & Ratcliff, J. (2007). Position based dynamics. *Journal of Visual Communication and Image Representation, 18*(2), 109–118. https://doi.org/10.1016/j.jvcir.2007.01.005

Müller, P., Wonka, P., Haegler, S., Ulmer, A., & Van Gool, L. (2006). Procedural modeling of buildings. *ACM Transactions on Graphics, 25*(3), 614–623. https://doi.org/10.1145/1141911.1141931

Müller, T., Evans, A., Schied, C., & Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics, 41*(4), Article 102. https://doi.org/10.1145/3528223.3530127

Museth, K. (2013). VDB: High-resolution sparse volumes with dynamic topology. *ACM Transactions on Graphics, 32*(3), Article 27. https://doi.org/10.1145/2487228.2487235

Mystakidis, S. (2022). Metaverse. *Encyclopedia, 2*(1), 486–497. https://doi.org/10.3390/encyclopedia2010031

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., & Fitzgibbon, A. (2011). KinectFusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE Interna-*

*tional Symposium on Mixed and Augmented Reality* (pp. 127–136). IEEE. https://doi.org/10.1109/ISMAR.2011.6092378

Ngo, T. D., Kashani, A., Imbalzano, G., Nguyen, K. T. Q., & Hui, D. (2018). Additive manufacturing (3D printing): A review of materials, methods, applications and challenges. *Composites Part B: Engineering, 143*, 172–196. https://doi.org/10.1016/j.compositesb.2018.02.012

Nickolls, J., Buck, I., Garland, M., & Skadron, K. (2008). Scalable parallel programming with CUDA. *ACM Queue, 6*(2), 40–53. https://doi.org/10.1145/1365490.1365500

Novák, J., Georgiev, I., Hanika, J., & Jarosz, W. (2018). Monte Carlo methods for volumetric light transport simulation. *Computer Graphics Forum, 37*(2), 551–576. https://doi.org/10.1111/cgf.13383

Okun, J. A., & Zwerman, S. (Eds.). (2020). *The VES handbook of visual effects: Industry standard VFX practices and procedures* (3rd ed.). Routledge. https://doi.org/10.4324/9781351009409

Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., & Phillips, J. C. (2008). GPU computing. *Proceedings of the IEEE, 96*(5), 879–899. https://doi.org/10.1109/JPROC.2008.917757

Oxman, R. (2006). Theory and design in the first digital age. *Design Studies, 27*(3), 229–265. https://doi.org/10.1016/j.destud.2005.11.002

Parent, R. (2012). *Computer animation: Algorithms and techniques* (3rd ed.). Morgan Kaufmann.

Parish, Y. I. H., & Müller, P. (2001). Procedural modeling of cities. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 301–308). ACM. https://doi.org/10.1145/383259.383292

Park, J. J., Florence, P., Straub, J., Newcombe, R., & Lovegrove, S. (2019). DeepSDF: Learning continuous signed distance functions for shape representation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 165–174). IEEE. https://doi.org/10.1109/CVPR.2019.00025

Parker, S. G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., McAllister, D., McGuire, M., Morley, K., Robison, A., & Stich, M. (2010). OptiX: A general purpose ray tracing engine. *ACM Transactions on Graphics, 29*(4), Article 66. https://doi.org/10.1145/1778765.1778803

Patney, A., Salvi, M., Kim, J., Kaplanyan, A., Wyman, C., Benty, N., Luebke, D., & Lefohn, A. (2016). Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics, 35*(6), 1–12. https://doi.

org/10.1145/2980179.2980246

Peng, X. B., Abbeel, P., Levine, S., & van de Panne, M. (2018). DeepMimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics, 37*(4), Article 143. https://doi.org/10.1145/3197517.3201311

Perlin, K. (1985). An image synthesizer. *ACM SIGGRAPH Computer Graphics, 19*(3), 287–296. https://doi.org/10.1145/325165.325247

Pharr, M., Jakob, W., & Humphreys, G. (2023). *Physically based rendering: From theory to implementation* (4th ed.). MIT Press.

Phong, B. T. (1975). Illumination for computer generated pictures. *Communications of the ACM, 18*(6), 311–317. https://doi.org/10.1145/360825.360839

Piegl, L., & Tiller, W. (1997). *The NURBS book* (2nd ed.). Springer. https://doi.org/10.1007/978-3-642-59223-2

Po, R., Yifan, W., Golyanik, V., Aberman, K., Barron, J. T., Bermano, A., Chan, E., Dekel, T., Holynski, A., Kanazawa, A., Liu, C. K., Liu, L., Mildenhall, B., Nießner, M., Ommer, B., Theobalt, C., Wonka, P., & Wetzstein, G. (2024). State of the art on diffusion models for visual computing. *Computer Graphics Forum, 43*(2). https://doi.org/10.1111/cgf.15063

Poole, B., Jain, A., Barron, J. T., & Mildenhall, B. (2023). DreamFusion: Text-to-3D using 2D diffusion. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*. https://openreview.net/forum?id=FjNys5c7VyY

Porter, T., & Duff, T. (1984). Compositing digital images. *ACM SIGGRAPH Computer Graphics, 18*(3), 253–259. https://doi.org/10.1145/964965.808606

Prince, S. (2012). *Digital visual effects in cinema: The seduction of reality*. Rutgers University Press.

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). PointNet: Deep learning on point sets for 3D classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 77–85). IEEE. https://doi.org/10.1109/CVPR.2017.16

Reeves, W. T. (1983). Particle systems—A technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics, 2*(2), 91–108. https://doi.org/10.1145/357318.357320

Reinhard, E., Stark, M., Shirley, P., & Ferwerda, J. (2002). Photographic tone reproduction for digital images. *ACM Transactions on Graphics, 21*(3), 267–276. https://doi.org/10.1145/566654.566575

Remondino, F., & El-Hakim, S. (2006). Image-based 3D modelling: A review. *The Photogrammetric Record, 21*(115), 269–291. https://doi.org/10.1111/j.1477-9730.2006.00383.x

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics, 21*(4), 25–34. https://doi.org/10.1145/37402.37406

Ritschel, T., Dachsbacher, C., Grosch, T., & Kautz, J. (2012). The state of the art in interactive global illumination. *Computer Graphics Forum, 31*(1), 160–188. https://doi.org/10.1111/j.1467-8659.2012.02093.x

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 10674–10685). IEEE. https://doi.org/10.1109/CVPR52688.2022.01042

Sacks, R., Eastman, C., Lee, G., & Teicholz, P. (2018). *BIM handbook: A guide to building information modeling for owners, designers, engineers, contractors, and facility managers* (3rd ed.). Wiley. https://doi.org/10.1002/9781119287568

SAG-AFTRA. (n.d.). *Digital replicas 101: What you need to know about the 2023 TV/theatrical contracts* [Fact sheet]. https://www.sagaftra.org/sites/default/files/sa_documents/DigitalReplicas.pdf

Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., & Li, H. (2019). PIFu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 2304–2314). IEEE. https://doi.org/10.1109/ICCV.2019.00239

Schied, C., Kaplanyan, A., Wyman, C., Patney, A., Chaitanya, C. R. A., Burgess, J., Liu, S., Dachsbacher, C., Lefohn, A., & Salvi, M. (2017). Spatiotemporal variance-guided filtering: Real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics* (Article 2). ACM. https://doi.org/10.1145/3105762.3105770

Schönberger, J. L., & Frahm, J.-M. (2016). Structure-from-motion revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4104–4113). IEEE. https://doi.org/10.1109/CVPR.2016.445

Shi, S., & Hsu, C.-H. (2015). A survey of interactive remote rendering systems. *ACM Computing Surveys, 47*(4), 1–29. https://doi.org/10.1145/2719921

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., & Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *CVPR 2011* (pp. 1297–1304). IEEE. https://doi.

org/10.1109/CVPR.2011.5995316

Slater, M., & Sanchez-Vives, M. V. (2016). Enhancing our lives with immersive virtual reality. *Frontiers in Robotics and AI, 3*, Article 74. https://doi.org/10.3389/frobt.2016.00074

Sloan, P.-P., Kautz, J., & Snyder, J. (2002). Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics, 21*(3), 527–536. https://doi.org/10.1145/566654.566612

Smelik, R. M., Tutenel, T., Bidarra, R., & Benes, B. (2014). A survey on procedural modelling for virtual worlds. *Computer Graphics Forum, 33*(6), 31–50. https://doi.org/10.1111/cgf.12276

Snavely, N., Seitz, S. M., & Szeliski, R. (2006). Photo tourism: Exploring photo collections in 3D. *ACM Transactions on Graphics, 25*(3), 835–846. https://doi.org/10.1145/1141911.1141964

Stam, J. (1999). Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 121–128). ACM. https://doi.org/10.1145/311535.311548

Starke, S., Zhang, H., Komura, T., & Saito, J. (2019). Neural state machine for character-scene interactions. *ACM Transactions on Graphics, 38*(6), 1–14. https://doi.org/10.1145/3355089.3356505

Stomakhin, A., Schroeder, C., Chai, L., Teran, J., & Selle, A. (2013). A material point method for snow simulation. *ACM Transactions on Graphics, 32*(4), Article 102. https://doi.org/10.1145/2461912.2461948

Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 3645–3650). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1355

Summerville, A., Snodgrass, S., Guzdial, M., Holmgård, C., Hoover, A. K., Isaksen, A., Nealen, A., & Togelius, J. (2018). Procedural content generation via machine learning (PCGML). *IEEE Transactions on Games, 10*(3), 257–270. https://doi.org/10.1109/TG.2018.2846639

Sutherland, I. E. (1963). Sketchpad: A man-machine graphical communication system. In *Proceedings of the May 21–23, 1963, Spring Joint Computer Conference* (pp. 329–346). ACM. https://doi.org/10.1145/1461551.1461591

Sutherland, I. E. (1968). A head-mounted three dimensional display. In *Proceedings of the December 9–11, 1968, Fall Joint Computer Conference, Part I* (pp. 757–764). ACM. https://doi.org/10.1145/1476589.1476686

Szeliski, R. (2022). *Computer vision: Algorithms and applications* (2nd ed.). Springer. https://doi.org/10.1007/978-3-030-34372-9

Tao, F., Zhang, H., Liu, A., & Nee, A. Y. C. (2019). Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics, 15*(4), 2405–2415. https://doi.org/10.1109/TII.2018.2873186

Terzopoulos, D., Platt, J., Barr, A., & Fleischer, K. (1987). Elastically deformable models. *ACM SIGGRAPH Computer Graphics, 21*(4), 205–214. https://doi.org/10.1145/37402.37427

Tewari, A., Fried, O., Thies, J., Sitzmann, V., Lombardi, S., Sunkavalli, K., Martin-Brualla, R., Simon, T., Saragih, J., Nießner, M., Pandey, R., Fanello, S., Wetzstein, G., Zhu, J.-Y., Theobalt, C., Agrawala, M., Shechtman, E., Goldman, D. B., & Zollhöfer, M. (2020). State of the art on neural rendering. *Computer Graphics Forum, 39*(2), 701–727. https://doi.org/10.1111/cgf.14022

Tewari, A., Thies, J., Mildenhall, B., Srinivasan, P., Tretschk, E., Wang, Y., Lassner, C., Sitzmann, V., Martin-Brualla, R., Lombardi, S., Simon, T., Theobalt, C., Nießner, M., Barron, J. T., Wetzstein, G., Zollhöfer, M., & Golyanik, V. (2022). Advances in neural rendering. *Computer Graphics Forum, 41*(2), 703–735. https://doi.org/10.1111/cgf.14507

Thies, J., Zollhöfer, M., Stamminger, M., Theobalt, C., & Nießner, M. (2016). Face2Face: Real-time face capture and reenactment of RGB videos. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 2387–2395). IEEE. https://doi.org/10.1109/CVPR.2016.262

Togelius, J., Yannakakis, G. N., Stanley, K. O., & Browne, C. (2011). Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games, 3*(3), 172–186. https://doi.org/10.1109/TCIAIG.2011.2148116

U.S. Copyright Office. (2025). *Copyright and artificial intelligence, Part 2: Copyrightability*. https://www.copyright.gov/ai/Copyright-and-Artificial-Intelligence-Part-2-Copyrightability-Report.pdf

Veach, E., & Guibas, L. J. (1997). Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 65–76). ACM. https://doi.org/10.1145/258734.258775

Wald, I., Woop, S., Benthin, C., Johnson, G. S., & Ernst, M. (2014). Embree: A kernel framework for efficient CPU ray tracing. *ACM Transactions on Graphics, 33*(4), 1–8. https://doi.org/10.1145/2601097.2601199

Ward, G. J. (1994). The RADIANCE lighting simulation and rendering system. In *Pro-*

*ceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (pp. 459–472). ACM. https://doi.org/10.1145/192161.192286

Westoby, M. J., Brasington, J., Glasser, N. F., Hambrey, M. J., & Reynolds, J. M. (2012). 'Structure-from-Motion' photogrammetry: A low-cost, effective tool for geoscience applications. *Geomorphology, 179*, 300–314. https://doi.org/10.1016/j.geomorph.2012.08.021

Whitted, T. (1980). An improved illumination model for shaded display. *Communications of the ACM, 23*(6), 343–349. https://doi.org/10.1145/358876.358882

Williams, L. (1978). Casting curved shadows on curved surfaces. *ACM SIGGRAPH Computer Graphics, 12*(3), 270–274. https://doi.org/10.1145/965139.807402

Williams, L. (1983). Pyramidal parametrics. *ACM SIGGRAPH Computer Graphics, 17*(3), 1–11. https://doi.org/10.1145/964967.801126

Witkin, A., & Kass, M. (1988). Spacetime constraints. *ACM SIGGRAPH Computer Graphics, 22*(4), 159–168. https://doi.org/10.1145/378456.378507

Xiao, L., Nouri, S., Chapman, M., Fix, A., Lanman, D., & Kaplanyan, A. (2020). Neural supersampling for real-time rendering. *ACM Transactions on Graphics, 39*(4). https://doi.org/10.1145/3386569.3392376

Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., Tombari, F., Tompkin, J., Sitzmann, V., & Sridhar, S. (2022). Neural fields in visual computing and beyond. *Computer Graphics Forum, 41*(2), 641–676. https://doi.org/10.1111/cgf.14505

Yang, L., Liu, S., & Salvi, M. (2020). A survey of temporal antialiasing techniques. *Computer Graphics Forum, 39*(2), 607–621. https://doi.org/10.1111/cgf.14018

Zwicker, M., Jarosz, W., Lehtinen, J., Moon, B., Ramamoorthi, R., Rousselle, F., Sen, P., Soler, C., & Yoon, S.-E. (2015). Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. *Computer Graphics Forum, 34*(2), 667–681. https://doi.org/10.1111/cgf.12592

Zwicker, M., Pfister, H., van Baar, J., & Gross, M. (2001). Surface splatting. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 371–378). ACM. https://doi.org/10.1145/383259.383300