CG Q&A

Aditya Raj

February, 2026

Contents

1 Background of Computer Graphics **3**
- 1.1 Purpose of Computer Graphics . . . 3
- 1.2 First Computer Graphics System . . . 3
- 1.3 First Fully Computer-Animated Film . . . 3
- 1.4 Uncanny Valley . . . 4

2 Main Steps in Computer Graphics **4**
- 2.1 Abstract Steps to Create Graphics . . . 4
- 2.2 Graphics Pipeline Stages . . . 5

3 Requirements in Computer Graphics **6**
- 3.1 Interactive vs. Non-Interactive . . . 6
- 3.2 Requirements for Realistic Rendering (Interactive vs. Offline) . . . 6
- 3.3 Real-Time vs. Offline Rendering . . . 7
- 3.4 Hard vs. Soft Real-Time . . . 7

1 Background of Computer Graphics

1.1 Purpose of Computer Graphics

Computer graphics serves several key purposes:

- **Visualization:**
  Makes complex data and concepts easier to understand (charts, diagrams, simulations, medical imaging).
- **Communication:** Conveys ideas more effectively than text alone (presentations, user interfaces, instructional materials).
- **Entertainment:** Produces content for movies, video games, animation, and virtual experiences.
- **Design and modeling:** Supports CAD and 3D modeling to design, test, and refine products before physical production.
- **Simulation and training:** Creates realistic environments for training (flight simulators, surgical practice).
- **User interaction:** Enables intuitive interfaces for interacting with digital systems.

In short, it *bridges digital information and human perception*, making technology more accessible, useful, and engaging.

1.2 First Computer Graphics System

The first computer graphics on a digital computer system were created on the **Whirlwind computer (MIT, 1951)**. It displayed *vector graphics* and used an early input device similar to a *light pen*, allowing direct interaction with the screen.

Whirlwind is therefore one of the earliest examples of **interactive computer graphics** on a true digital computing system.

1.3 First Fully Computer-Animated Film

The first fully computer-animated feature film is ***Toy Story* (1995)**, released by Pixar and Disney. While there were earlier uses of **computer-generated imagery (CGI)** in films (e.g., *Tron* (1982) and *Young Sherlock Holmes* (1985)), *Toy Story* was the first feature-length film created *entirely* using computer animation. It marked a major milestone in animation history and helped pave the way for modern CGI films.

1.4 Uncanny Valley

The **uncanny valley** is the unsettling feeling people experience when a character looks *almost—but not quite—human*. In computer graphics, this happens when a digital character is highly realistic (face, skin, movement) but still has subtle imperfections that feel unnatural. Small discrepancies (slightly off expressions, unnatural eye motion, mismatched proportions) can trigger discomfort or unease.

The term was introduced by Japanese roboticist **Masahiro Mori (1970)**. He observed that as robots or avatars become more human-like, emotional affinity increases—until a point where they look nearly human but still fall short. At that point, the observer's response drops into discomfort, forming a *"valley"* in a graph of human likeness vs. emotional response.

This is especially relevant in **film**, **games**, and **virtual environments**. For example, *The Polar Express* (2004), *Final Fantasy: The Spirits Within* (2001), and the 2019 *Cats* adaptation were criticized for lifelike yet unsettling characters. Game designers often avoid hyper-realistic humans and use stylized designs to maintain immersion.

To reduce the uncanny valley, designers focus on **consistency** across facial expressions, motion, voice, and proportions so that all cues align with realistic human behavior. Others deliberately choose *stylization* instead of photorealism to create more relatable characters.

2 Main Steps in Computer Graphics

2.1 Abstract Steps to Create Graphics

The main abstract steps in creating a computer graphic are:

- **Modeling:** Defining the objects and components of a scene in terms of shape, size, color, texture, and other parameters. This involves creating a digital representation of the 3D or 2D elements using primitives or detailed geometry.
- **Scene setup:** Arranging virtual objects, lights, cameras, and other entities in a virtual environment. This includes positioning and orienting elements to create a coherent composition (and keyframing for animation when needed).
- **Rendering:** Generating the final 2D image (or animation) from the prepared scene by applying lighting, shading, textures, and effects using algorithms such as ray tracing or scanline rendering.

These stages form the core of the graphics pipeline, which transforms abstract scene data into a visual image displayed on a screen.

2.2 Graphics Pipeline Stages

The **graphics pipeline** is a sequence of stages that transforms **3D scene data** into a **2D image** for display. It operates on primitives (vertices, triangles) using **GPUs** and graphics APIs (e.g., *OpenGL*, *Vulkan*) for real-time rendering.

Main processing steps and computations:

1. **Vertex Processing**
    - **Computations:** Transform vertices from local object space to screen space using 4×4 transformation matrices (modeling, viewing, projection).
    - **Key operations:** Per-vertex calculations such as position, normal, and color transformations.
    - **Programmable:** Vertex shaders allow custom logic (e.g., procedural deformation, animation).
2. **Tessellation** *(optional)*
    - **Computations:** Subdivide coarse geometry into finer meshes using Tessellation Control Shaders (TCS) and Tessellation Evaluation Shaders (TES).
    - **Purpose:** Dynamic level-of-detail and smooth surfaces (e.g., subdivision surfaces).
3. **Geometry Processing** *(optional)*
    - **Computations:** Modify or generate new primitives (e.g., convert points to triangles, create particles).
    - **Programmable:** Geometry shaders process whole primitives and can output new ones.
4. **Primitive Assembly & Clipping**
    - **Computations:** Group vertices into primitives (triangles, lines, points), then clip primitives outside the view frustum.
    - **Perspective division:** Maps coordinates to normalized device coordinates (NDC).
5. **Rasterization**
    - **Computations:** Convert primitives into fragments (potential pixels).
    - **Key tasks:** Determine pixel coverage and interpolate vertex attributes (color, texture coordinates, depth) across the primitive.
6. **Fragment Processing**

- **Computations:** Compute final pixel color per fragment using fragment shaders.
- **Operations:** Apply lighting, texturing, shadows, reflections, and material effects.
- **Highly parallel:** Each fragment is processed independently.

7. **Per-Sample Operations**
   - **Computations:** Final decisions before writing to the framebuffer.
   - **Key operations:**
     - Depth (Z) testing (via Z-buffering) to resolve visibility.
     - Stencil testing for masking.
     - Blending for transparency.
     - Logical operations and write masks.

This pipeline evolved from *fixed-function* hardware (1990s) to *programmable* stages (post-2001), enabling complex visual effects. Modern GPUs extend this with **ray tracing** (e.g., NVIDIA RTX) and **AI-accelerated denoising**, blending rasterization with newer rendering paradigms.

3 Requirements in Computer Graphics

3.1 Interactive vs. Non-Interactive

Interactive computer graphics involve **real-time** user engagement and manipulation of visual content. Users interact through input devices (keyboard, mouse, touchscreen, VR controllers), and the system responds dynamically. Examples include video games, VR simulations, flight simulators, CAD systems, and interactive data visualizations.

Non-interactive computer graphics are **pre-generated** and static, with no real-time user control. The content is fixed and displayed as-is, and the user can only view the output. Examples include digital images, movies, slide shows, and static website graphics.

Key difference: *user interaction*. Interactive graphics respond immediately to user input, while non-interactive graphics are fixed once rendered.

3.2 Requirements for Realistic Rendering (Interactive vs. Offline)

Realism in non-interactive (offline) computer graphics

For non-interactive computer graphics—such as pre-rendered movie scenes or high-quality visualizations—**image quality** is prioritized over speed. Key requirements include high computational power, advanced rendering algorithms, and

accurate physical modeling. This enables techniques like **ray tracing**, **global illumination**, **physically based rendering (PBR)**, subsurface scattering, and photon mapping. The **rendering equation** (Kajiya, 1986) is foundational for modeling real-world lighting. Offline pipelines also rely on shaders (HLSL/GLSL) for effects like bump mapping, normal mapping, and reflections.

Realism in interactive (real-time) computer graphics

In interactive systems (games, simulations, VR), **real-time performance** is the priority. The main challenge is balancing realism with fast rendering and low latency. Requirements include an efficient graphics pipeline, GPU hardware acceleration, and optimized algorithms. Most real-time systems rely on **rasterization** for speed, supported by APIs such as OpenGL, DirectX, and Vulkan. Techniques like depth buffering, level-of-detail (LOD), normal mapping, and shadow mapping help maintain realism with smooth frame rates.

Shared requirements

Both domains rely on geometric primitives (lines, polygons, curves), transformations (translation, rotation, scaling), clipping, texture mapping, and correct color handling. The key difference is the trade-off: offline systems maximize realism; interactive systems maximize responsiveness.

3.3 Real-Time vs. Offline Rendering

Real-time computer graphics generates and displays frames fast enough to create the illusion of motion and support immediate interaction. Typical targets are 30–60 FPS (or higher). Real-time rendering uses GPU-accelerated rasterization and performance-friendly techniques, sometimes sacrificing photorealism to maintain speed.

Offline (non-real-time) rendering prioritizes maximum quality and realism and may take minutes to hours per frame. It can use computationally expensive techniques such as global illumination, ray tracing, and path tracing, and is common in feature films and high-end visualization.

Summary:

- **Real-time:** Speed first; interactivity second $\rightarrow$ used in games and VR.
- **Offline:** Quality first; speed second $\rightarrow$ used in movies and high-end VFX.

3.4 Hard vs. Soft Real-Time

Hard real-time in computer graphics means missing a deadline (e.g., failing to render a frame within a strict time limit) is unacceptable and can cause unsafe or catastrophic outcomes. Examples include safety-critical simulators or medical/aerospace systems.

Soft real-time means missing a deadline degrades quality (e.g., dropped frames, stutter) but the system continues operating. This is typical in games and multimedia playback.

Summary:

- **Hard real-time:** Deadline misses are unacceptable (e.g., safety-critical medical/aerospace systems).
- **Soft real-time:** Deadline misses degrade quality but the system continues (e.g., games, video playback).