Game engine

A **game engine** is a software-development environment designed for people to build video games. Developers use game engines to construct games for consoles, mobile devices, and personal computers. The core functionality typically provided by a game engine includes a rendering engine ("renderer") for 2D or 3D graphics, a physics engine or collision detection (and collision response), sound, scripting, animation, artificial intelligence, networking, streaming, memory management, threading, localization support, scene graph, and may include video support for cinematics. Implementers often economize on the process of game development by reusing/adapting, in large part, the same game engine to produce different games[1] or to aid in porting games to multiple platforms.

Creating a racing game in Blender Game Engine

Contents

Purpose

In many cases, game engines provide a suite of visual development tools in addition to reusable software components. These tools are generally provided in an integrated development environment to enable simplified, rapid development of games in a data-driven manner. Game engine developers attempt to "pre-invent the wheel" by developing robust software suites which include many elements a game developer may need to build a game. Most game engine suites provide facilities that ease development, such as graphics, sound, physics and AI functions. These game engines are sometimes called "middleware" because, as with the business sense of the term, they provide a flexible and reusable software platform which provides all the core functionality needed, right out of the box, to develop a game application while reducing costs, complexities, and time-to-market — all critical factors in the highly competitive video game industry.[2] As of 2001, Gamebryo, JMonkeyEngine and RenderWare were such widely used middleware programs.[3]

Like other types of middleware, game engines usually provide platform abstraction, allowing the same game to be run on various platforms including game consoles and personal computers with few, if any, changes made to the game source code. Often, game engines are designed with a component-based architecture that allows specific systems in the engine to be replaced or extended with more specialized (and often more expensive) game middleware components. Some game engines are designed as a series of loosely connected game middleware components that can be selectively combined to create a custom engine, instead of the more common approach of extending or customizing a flexible integrated product. However extensibility is achieved, it remains a high priority for game engines due to the wide variety of uses for which they are applied. Despite the specificity of the name, game engines are often used for other kinds of interactive applications with real-time graphical needs such as marketing demos, architectural visualizations, training simulations, and modeling environments.[4]

Some game engines only provide real-time 3D rendering capabilities instead of the wide range of functionality needed by games. These engines rely upon the game developer to implement the rest of this functionality or assemble it from other game middleware components. These types of engines are generally referred to as a "graphics engine", "rendering engine", or "3D engine" instead of the more encompassing term "game engine". This terminology is inconsistently used as many full-featured 3D game engines are referred to simply as "3D engines". A few examples of graphics engines are: Crystal Space, Genesis3D, Irrlicht, OGRE, RealmForge, Truevision3D, and Vision Engine. Modern game or graphics engines generally provide a scene graph, which is an object-oriented representation of the 3D game world which often simplifies game design and can be used for more efficient rendering of vast virtual worlds.

As technology ages, the components of an engine may become outdated or insufficient for the requirements of a given project. Since the complexity of programming an entirely new engine may result in unwanted delays (or necessitate that the project be completely restarted), a development team may elect to update their existing engine with newer functionality or components.

Components

Such a framework is composed of a multitude of very different components.

Main game program

The actual game logic has to be implemented by some algorithms. It is distinct from any rendering, sound or input work.

Rendering engine

The rendering engine generates animated 3D graphics by any of a number of methods (rasterization, ray-tracing etc.).

Instead of being programmed and compiled to be executed on the CPU or GPU directly, most often rendering engines are built upon one or multiple rendering application programming interfaces (APIs), such as Direct3D, OpenGL, or Vulkan which provide a software abstraction of the graphics processing unit (GPU). Low-level libraries such as DirectX, Simple DirectMedia Layer (SDL), and OpenGL are also commonly used in games as they provide hardware-independent access to other computer hardware such as input devices (mouse, keyboard, and joystick), network cards, and sound cards. Before hardware-accelerated 3D graphics, software renderers had been used. Software rendering is still used in some modeling tools or for still-rendered images when visual accuracy is valued over real-time performance (frames-per-second) or when the computer hardware does not meet needs such as shader support.

With the advent of hardware accelerated physics processing, various physics APIs such as PAL and the physics extensions of COLLADA became available to provide a software abstraction of the physics processing unit of different middleware providers and console platforms.

Game engines can be written in any programming language like C++, C or Java, though each language is structurally different and may provide different levels of access to specific functions.

Audio engine

The audio engine is the component which consists of algorithms related to the loading, modifying and output of sound through the client's speaker system. At a minimum it must be able to load, decompress and play sound files. More advanced audio engines can calculate and produce such things as Doppler effects, echoes, pitch/amplitude adjustments, oscillation, etc. It can perform calculations on the CPU, or on a dedicated ASIC. Abstraction APIs, such as OpenAL, SDL audio, XAudio 2, Web Audio, etc. are available.

Physics engine

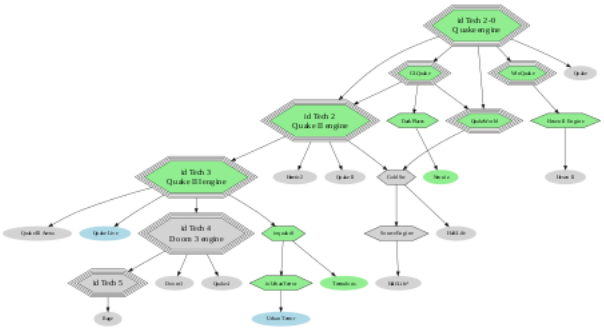
The physics engine is responsible for emulating the laws of physics realistically within the application. Specifically, it provides a set of functions for simulating physical forces and collisions, acting on the various objects within the game at run time.

Artificial intelligence

The AI is usually outsourced from the main game program into a special module to be designed and written by software engineers with specialist knowledge. Most games will implement very different AI systems, and thus, AI is considered to be specific to the particular game for which it is created. Many modern game engines come packaged with search algorithms such as A-star and subroutines for baking level geometry into a Navmesh which can help speed up the process of scripting AI behavior.

History

Before game engines, games were typically written as singular entities: a game for the Atari 2600, for example, had to be designed from the bottom up to make optimal use of the display hardware—this core display routine is today called the kernel by retro developers. Other platforms had more leeway, but even when the display was not a concern, memory constraints usually sabotaged attempts to create the data-heavy design that an engine needs. Even on more accommodating platforms, very little could be reused between games. The rapid advance of arcade hardware—which was the leading edge of the market at the time—meant that most of the code would have to be thrown out afterwards anyway, as later generations of games would use completely different game designs that took advantage of extra resources. Thus most game designs through the 1980s were designed through a hard-coded ruleset with a small number of levels and graphics data. Since the golden age of arcade video games, it became common for video game companies to develop in-house game engines for use with first party software.



Some game engines experience an evolution over time and develop a family tree, like for instance id's Quake engine which resulted in the id Tech family

While third-party game engines were not common up until the rise of 3D computer graphics in the 1990s, there were several 2D game creation systems produced in the 1980s for independent video game development. These include *Pinball Construction Set* (1983), ASCII's *War Game Construction Kit* (1983),[5] *Thunder Force Construction* (1984),[6] *Adventure Construction Set* (1984), Garry Kitchen's GameMaker (1985), *Wargame Construction Set* (1986), *Shoot-'Em-Up Construction Kit* (1987), *Arcade Game Construction Kit* (1988), and most popularly ASCII's *RPG Maker* engines from 1998 onwards. Klik & Play (1994) is another legacy offering that's still available.

The term "game engine" arose in the mid-1990s, especially in connection with 3D games such as first-person shooters (FPS). (*See also:* first-person shooter engine.) Such was the popularity of Id Software's *Doom* and *Quake* games that, rather than work from scratch, other developers licensed the core portions of the software and designed their own graphics, characters, weapons and levels—the "game content" or "game assets". Separation of game-specific rules and data from basic concepts like collision detection and game entity meant that teams could grow and specialize.

Later games, such as id Software's *Quake III Arena* and Epic Games's 1998 *Unreal* were designed with this approach in mind, with the engine and content developed separately. The practice of licensing such technology has proved to be a useful auxiliary revenue stream for some game developers, as one license for a high-end commercial game engine can range from US$10,000 to millions of dollars, and the number of licensees can reach several dozen companies, as seen with the Unreal Engine. At the very least, reusable engines make developing game sequels faster and easier, which is a valuable advantage in the competitive video game industry. While there was a strong rivalry between Epic and id around 2000, since then Epic's Unreal Engine has been far more popular than id Tech 4 and its successor id Tech 5.[7]

Modern game engines are some of the most complex applications written, often featuring dozens of finely tuned systems interacting to ensure a precisely controlled user experience. The continued evolution of game engines has created a strong separation between rendering, scripting, artwork, and level design. It is now common, for example, for a typical game development team to have several times as many artists as actual programmers.[8]

First-person shooter games remain the predominant users of third-party game engines, but they are now also being used in other genres. For example, the role-playing video game *The Elder Scrolls III: Morrowind* and the MMORPG *Dark Age of Camelot* are based on the Gamebryo engine, and the MMORPG *Lineage II* is based on the Unreal Engine. Game engines are used for games originally developed for home consoles as well; for example, the RenderWare engine is used in the *Grand Theft Auto* and *Burnout* franchises.

Threading is taking on more importance due to modern multi-core systems (e.g. Cell) and increased demands in realism. Typical threads involve rendering, streaming, audio, and physics. Racing games have typically been at the forefront of threading with the physics engine running in a separate thread long before other core subsystems were moved, partly because rendering and related tasks need updating at only 30–60 Hz. For example, on PlayStation 3, physics ran in *Need For Speed* at 100 Hz versus *Forza Motorsport 2* at 360 Hz.

Although the term was first used in the 1990s, there are a few earlier systems in the 1980s that are also considered to be game engines, such as Sierra's Adventure Game Interpreter (AGI) and SCI systems, LucasArts' SCUMM system and Incentive Software's Freescape engine. Unlike most modern game engines, these game engines were never used in any third-party products (except for the SCUMM system which was licensed to and used by Humongous Entertainment).

As game engine technology matures and becomes more user-friendly, the application of game engines has broadened in scope. They are now being used for serious games: visualization, training, medical, and military simulation applications, with the CryEngine being one example.[9] To facilitate this accessibility, new hardware platforms are now being targeted by game engines, including mobile phones (e.g. Android phones, iPhone) and web browsers (e.g. WebGL, Shockwave, Flash, Trinigy's WebVision, Silverlight, Unity Web Player, O3D and pure DHTML).[10]

Additionally, more game engines are being built upon higher level languages such as Java and C#/.NET (e.g. TorqueX, and Visual3D.NET), Python (Panda3D), or Lua Script (Leadwerks). As most 3D rich games are now mostly GPU-limited (i.e. limited by the power of the graphics card), the potential slowdown due to translation overheads of higher level languages becomes negligible, while the productivity gains offered by these languages work to the game engine developers' benefit.[11] These recent trends are being propelled by companies such as

Microsoft to support Indie game development. Microsoft developed XNA as the SDK of choice for all video games released on Xbox and related products. This includes the Xbox Live Indie Games [12] channel designed specifically for smaller developers who don't have the extensive resources necessary to box games for sale on retail shelves. It is becoming easier and cheaper than ever to develop game engines for platforms that support managed frameworks.[13]

Game engines as an industry

Producers of game engines decide how they allow users to utilize their products. Just as gaming is an industry, so are the engines they are built off of. The major game engines come at varying prices, whether it be in the form of subscription fees or license payments. [14]

One of the major game engines used to create several notable games such as Fortnite, PlayerUnknown's Battlegrounds, and Life is Strange 2, the Unreal Engine 4 adopted a free-to-use structure with a royalty on all game sales using this engine.[15] Although the differences among the different game engines blur as they are built upon during the game creation process, different game developers may either be too used to a system to change, or attracted by the huge benefits of such engines regardless of pay-walls.

Another game engine currently bringing in a notable income would be the Unity engine, utilizing a similar pay module to the aforementioned Unreal Engine. [16] This engine is the one behind games such as Rust, Subnautica, and Life is Strange Before the Storm.

Among the other game engines available, Unreal Engine and Unity are often compared and considered competitors on the business side of game engineering. [17]

Game middleware

In the broader sense of the term, game engines themselves can be described as middleware. In the context of video games, however, the term "middleware" is often used to refer to subsystems of functionality within a game engine. Some game middleware does only one thing but does it more convincingly or more efficiently than general purpose middleware. For example, *SpeedTree* was used to render the realistic trees and vegetation in the role-playing video game *The Elder Scrolls IV: Oblivion*[18] and *Fork Particle* was used to simulate and render real time particle system visual effects or particle effects in Sid Meier's Civilization V.[19]

The four most widely used middleware packages[20] that provide subsystems of functionality include RAD Game Tools' Bink, Firelight FMOD, Havok, and Scaleform GFx. RAD Game Tools develops Bink for basic video rendering, along with Miles audio, and Granny 3D rendering. Firelight FMOD is a low cost robust audio library and toolset. Havok provides a robust physics simulation system, along with a suite of animation and behavior applications. *Scaleform* provides GFx for high performance Flash UI and high-quality video playback, and an Input Method Editor (IME) add-on for in-game Asian chat support.

Other middleware is used for performance optimisation - for example 'Simplygon' helps to optimise and generate level of detail meshes, and 'Umbra' adds occlusion culling optimisations to 3d graphics.

Some middleware contains full source code, others just provide an API reference for a compiled binary library.

Some middleware programs can be licensed either way, usually for a higher fee for full source code.

First-person shooter engines

A subset of game engines are 3D first-person shooter (FPS) game engines. Groundbreaking development in terms of visual quality is done in order to get FPS games to its current standard. The level of visual details emphasized in these games have become increasingly precise, something that engines focused on flight and driving simulators and real-time strategy (RTS) games don't contain.

The development of the FPS graphic engines that appear in games can be characterized by a steady increase in technologies, with some breakthroughs. Attempts at defining distinct generations lead to arbitrary choices of what constitutes a highly modified version of an "old engine" and what is a brand-new engine.[21]

The classification is complicated as game engines blend old and new technologies. Features that were considered advanced in a new game one year become the expected standard the next year. Games with a mix of older generation and newer feature are the norm.

See also

- List of game engines
- List of first-person shooter engines
- 3D computer graphics
- List of game middleware
- Authoring system

References

1. "What is a Game Engine?" (http://www.gamecareerguide.com/features/529/what_is_a_game_.php). GameCareerGuide.com. Retrieved 2013-11-24.
2. Cowan, Danny. "Joystiq" (http://www.gamedaily.com/articles/features/my-turn-the-real-cost-of-middleware/71334/?biz=1). Gamedaily.com. Retrieved 2013-11-24.
3. "Rise of Middleware" (http://www.develop-online.net/features/13/Rise-of-Middleware-20). Develop-online.net. 2007-07-06. Retrieved 2011-01-17.
4. Report on Use of Middleware in Games (http://adlcommunity.net/file.php/36/GrooveFiles/Games%20Madison/report%20Middleware.pdf) Archived (https://web.archive.org/web/20131017044354/http://adlcommunity.net/file.php/36/GrooveFiles/Games%20Madison/report%20Middleware.pdf) October 17, 2013, at the Wayback Machine
5. "War Game Construction Kit" (https://translate.google.com/translate?hl=en&sl=ja&tl=en&u=http%3A%2F%2Fretropc.net%2Ffm-7%2Fmuseum%2Fsofthouse%2Fascii%2F000701300.html). *Oh!FM*. Archived from the original (http://retropc.net/fm-7/museum/softhouse/ascii/000701300.html) on 3 September 2012. Retrieved 3 September 2012.

6. "Thunder Force Construction" (https://translate.google.com/translate?sl=ja&tl=en&js=n&prev=_t&hl=en&ie=UTF-8&layout=2&eotf=1&u=http%3A%2F%2Fretropc.net%2Ffm-7%2Fmuseum%2Fsofthouse%2Ftecnosoft%2F330602301.html). *Oh!FM*. Archived from the original (http://retropc.net/fm-7/museum/softhouse/tecnosoft/330602301.html) on 1 September 2012. Retrieved 1 September 2012.
7. Bramwell, Tom (2007-08-09). "id Tech 5 Interview • Page 1 • Interviews •" (http://www.eurogamer.net/articles/id-tech-5-interview). Eurogamer.net. Retrieved 2013-11-24.
8. "Game Development Team Composition Study - Changes over time" (http://web.cs.wpi.edu/~id111x/c05/slides/intro.ppt). Retrieved 2011-01-17.
9. "Video Games Starting to Get Serious" (http://www.gazette.net/stories/083107/businew11739_32356.shtml). Gazette.net. 2007-08-31. Retrieved 2011-01-17.
10. "Gaming: Mobile and Wireless Trends for 2008" (https://web.archive.org/web/20110108073502/http://www.m-trends.org/2008/01/mobile-and-wireless-trends-for-2008.html). M-trends.org. Archived from the original (http://www.m-trends.org/2008/01/mobile-and-wireless-trends-for-2008.html) on 2011-01-08. Retrieved 2011-01-17.
11. *3D Game Engine Programming (book)* (https://books.google.com/books?id=-vifhqAi0SEC&pg=PA338&lpg=PA338&dq=gpu-limited+games&source=bl&ots=ETONUQuLV7&sig=L1gSNaYCZh2_TQBkg6wcPLCEobw&hl=en&sa=X&oi=book_result&resnum=6&ct=result#PPA337,M1). Books.google.com. Retrieved 2011-01-17.
12. "xboxlivecommunitygames.org" (http://www.xboxlivecommunitygames.org/). xboxlivecommunitygames.org. Retrieved 2013-11-24.
13. "Microsoft to Enable User-Created XBox 360 Games" (http://www.gamasutra.com/php-bin/news_index.php?story=10458). Retrieved 2017-05-05.
14. "The 10 Best Video Game Engines | 2018 Edition" (https://www.gamedesigning.org/career/video-game-engines/). *The Ultimate Resource for Video Game Design*. 2017-03-11. Retrieved 2019-05-15.
15. Savage, Phil (2015-03-02). "Unreal Engine 4 is now free" (https://www.pcgamer.com/unreal-engine-4-is-now-free/). *PC Gamer*. Retrieved 2019-05-15.
16. "The Two Engines Driving the $120B Gaming Industry Forward" (https://www.cbinsights.com/research/game-engines-growth-expert-intelligence/). *CB Insights Research*. 2018-09-20. Retrieved 2019-05-15.
17. "Unity vs Unreal: Ultimate Game Engine Showdown" (https://www.gamedesigning.org/engines/unity-vs-unreal/). *The Ultimate Resource for Video Game Design*. 2019-02-11. Retrieved 2019-05-15.
18. "Gamusutra Product Review of Top Vegetation Middleware" (http://www.gamasutra.com/view/feature/2797/product_review_speedtree_rt__.php). Gamasutra.com. 2003-10-01. Retrieved 2011-01-17.
19. "Firaxis Using Fork Particle Toolset For Civ V's Visual Effects" (http://www.gamasutra.com/view/news/30820/Firaxis_Using_Fork_Particle_Toolset_For_Civ_Vs_Visual_Effects.php). Gamasutra. 2010-10-06. Retrieved 2013-11-24.
20. "Gamasutra Engine and Middleware Technology Survey" (http://www.gamasutra.com/blogs/MarkDeLoura/20090316/903/The_Engine_Survey_Technology_Results.php). Gamasutra.com. 2009-05-08. Retrieved 2011-01-17.
21. Hauteville, Cédric (2011-04-02). "Technical Game Design: Aim systems in First Person Shooters" (http://technicalgamedesign.blogspot.com/2011/04/aim-systems-in-first-person-shooters.html). *Technical Game Design*. Retrieved 2019-03-13.
