Frequently asked questions

Q. What are the differences between a **3D Engine**, a **3D Language** and a **3D Authoring System**?

A. A **3D engine** is a library of 3D graphics functions. Many 3D engines are available on the Internet, some are free, some are between $100 and $300,000 for commercial use. Those engines require programming with an external delevopment system, like Microsoft Visual C++. Programming a game around a 3D engine offers a maximum of flexibility, but requires, of course, also a maximum of experience, work and time to be invested before something's moving in your game.

An easier approach is offered by some special **3D languages**. You are programming your game with a simple scripting language (mostly BASIC) that is especially designed for 3D games. Such languages don't offer the performance and flexibility of a real 3D engine, but avoid through mostly predefined functions a lot of problems related to real programming.

Even easier is working with an **authoring system**, which has a visual interface for very fast 'clicking together' a game prototype. Of course, only simple games can be created without any programming. Therefore authoring systems normally also provide a scripting language for partially or fully programming or customizing the game. This scripting language is often based on C++ or Basic, but is simplified for ease of use and shortness of code. With an authoring system, even ambitious game projects can be realized this way with a tenth of the code and a tenth of the time that would be needed to program a game around a 3D engine. Many major game companies begin to use authoring systems like 3D GameStudio for their game development.

Some authoring systems are specialized on **FPS** (first or third person shooters) and offer no or only limited scripting possibilities. Although even the FPS that can be created with those systems are not exaclty the state of the art, they are a good alternative if you want to create simple games without scripting at all. Most authoring systems however can be used for creating every type of game or 3D application.

Q. What is a **Renderer**?

A. A core part of a 3D engine which actually paints the 3D objects on the screen. There are mainly three rendering methods: **Software**, **DirectX**, or **OpenGl**. A Software renderer does not need 3D hardware at all. A **DirectX** renderer uses 3D hardware through Microsoft's DirectX library, which is supposed to be installed on each PC - so almost every 3D engine contains a DirectX renderer. The most recent DirectX Version is DX9. Newer DirectX versions support more features, but are not available on every PC. An **OpenGl** renderer uses 3D hardware through the OpenGl graphics library, which is available for most 3D cards. On old 3D cards, OpenGl normally renders a little faster, while on newer 3D cards DirectX is one step ahead. DirectX and OpenGl each contain a software renderer too, which is however for testing purposes only and runs too slow for realtime games.

Q. What is a **Culling System**, and what is a **LOD System**?

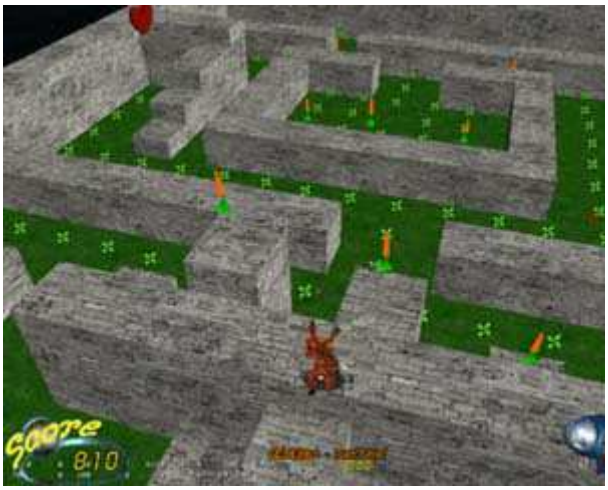
A. A culling system renders only the parts of a game level that are not covered by walls or other objects. Most commercial 3D engines use some sort of culling system, mostly either a **BSP tree** or a **Portal** algorithm. A BSP tree system is more effective than a portal system, but the editor also needs more time for the precalculation of the level. Some simple engines for amateur purposes don't have a culling system. Without culling, all objects in sight range contribute to the rendering time, regardless whether they are visible or hidden. This means the bigger the level is, and the more objects are placed in it, the slower and 'choppier' the game would run. With a culling system, the rendering speed is almost independent of the level size and number of objects, at least for indoor levels. BSP tree culling allows games to run with a decent frame rate even on old PCs.

A LOD system also increases the frame rate in outdoor levels. It automatically switches to 'simpler' shapes of objects when they are far away from the camera, thus reducing the overall number of polygons drawn per frame.

Q. What is **Shadow Mapping**?

A. Also called **Lightmapping** - a system for creating realistic lights and shadows without frame rate penalties. A shadow mapping compiler allows you to place an unlimited number of static light sources in the level, and then precalculates the light flow and static shadows for each surface. Most today's commercial 3D games use shadow mapping. As you see in the images below, smooth lights and shadows greatly add to the atmosphere of a game (images from an unnamed GameStudio game by Rémi Valantin).

Game with Shadow Mapping



Game without Shadow Mapping

Q. What is a **Particle System**?

A. An effect generator that creates a huge number of small particles for special effects like smoke, fire, or explosions. Well-made particle effects look better than pre-rendered animation, and thus particle generators are used in all newer games and game consoles. For creating realistic particle effects, the particle generator must be able to move thousands of particles without frame rate reduction. Some particle systems are also able to create light beams and tracer paths (images from the GameStudio game 'Glider' by REVOgames).

Particle effect

Beam effect

Q. What is a **Physics Engine**?

A. A physics engine calculates movement, rotation and collision response of rigid body objects by applying realistic physics to them. It's not absolutely necessary to use a physics engine for a game - simple 'Newtonian' physics, like acceleration and deceleration, can also be programmed or scripted to a certain extent. However, the programming possibilities end when the game requires objects colliding, rolling, gliding or ricocheting in a complex way - like in a racing or in a bowling game. A

physics engine uses object properties like momentum, torque or elasticity, and constraints like ball joints, wheels or hinges for simulating rigid body behavior. This does not only give much more realistic results, but is also much easier to handle for the developer than writing a behavior script.

Physics engines can be purchased from different manufactorers. Some authoring systems have an integrated physics engine (a video with some tests of GameStudio's physics engine available here). Be aware: some systems have 'physics' listed in their features list, but just mean 'scripting' or 'collision detection'.

Q. What is a **Client/Server Network System**?

A. There are two basic types of network systems for multiplayer games: **peer-to-peer** and **client/server**. In a peer-to-peer systems all connected PCs are equal. Each PC runs its own copy of the game and exchanges update messages with all other PCs. In a client/server system, one PC is the server which hosts and runs the game. All other PCs are clients which only update their player character and receive messages only from the server. To avoid the need for an extra PC, normally the server is also a client at the same time.

Peer-to-peer systems were used for the first multiplayer games, but are rarely used today due to the many disadvantages of this system. Peer-to-peer games don't run synchronously, so for instance a bullet could hit its target on one PC and miss on another PC (consistency problem). Also, due to the lot of cross traffic between all connected PCs you can not have many players in the game. Online games that require more than 4 players normally use the client/server system. A **zone system** is an advanced client/server system with multiple servers. The game world is divided into several levels (zones) that are controlled by separate servers. Zone systems are used for massive online multiplayer games with an unlimited number of players.

Peer-to-Peer .... max. 4-5 players
Client/Server.... max. 25-50 players
Zone .... unlimited, depends on number of servers

Q. What are the main differences between 3D GameStudio and other 3D game development systems or 3D languages?

A. Flexibility, robustness, ease of use, and graphics quality. The best way to see the difference is comparing GameStudio screenshots (Gallery) and demos (Download) with screenshots and demos made with other 3D game systems.

			Authoring Systems											
	3DGM	RF	Rad	Quest2	A6std	A6ext	A6com	A6pro	Jamagic	DBPro	Blitz	TV3D	Halflife	Jet3D
Price (US $)	49	79	99	749	49	89	199	899	99	99	99	150		free
Publish restrictions	$$	$$	none	none	Splash	Splash	Splash	none	Splash	none	none	$$	$$	$$
Renderer	DX7	DX6	DX6	DX8	DX8	DX8	DX8	DX8	DX6,OG	DX8	DX7	DX8	DX7	DX6
Culling system	--	BSP	--	--	BSP	BSP	BSP	BSP	--	BSP	--	BSP	BSP	BSP
LOD system	--	--	--	yes	--	yes	yes	yes	--	yes	yes	--	--	--
Terrain	yes	--	--	--	--	yes	yes	yes	--	yes	yes	yes	--	--
Shadow mapping	--	yes	--	--	yes	yes	yes	yes	--	**	yes	**	yes	yes
Dynamic shadows	--	yes	yes	yes	--	--	yes	yes	yes	yes	--	yes	--	yes
Shaders	--	--	--	yes	--	--	--	--	--	yes	--	yes	--	--
Particle generator	--	yes	--	yes	yes	yes	yes	yes	--	yes	--	yes	yes	yes
Beam generator	--	--	--	--	--	--	yes	yes	--	--	--	--	--	--
Bones animation	--	yes	--	yes	--	--	--	--	--	yes	yes	--	yes	yes
Expandable (plugin)	--	--	--	yes	yes	yes	yes	yes	yes	yes	yes	yes	--	--
Network system	--	--	p-p	--	--	--	c/s	zone	p-p	c/s	c/s	c/s	c/s	--
Physics engine	--	--	--	yes	--	--	yes	yes	--	--	--	--	--	--
Level editor	--	yes	--	yes	yes	yes	yes	yes	--	--	yes	--	yes	yes
Model editor	--	--	--	yes	yes	yes	yes	yes	--	--	--	--	--	--
Script editor	--	--	--	yes	yes	yes	yes	yes	yes	yes	yes	--	--	--
Script compiler	--	--	?	--	yes	yes	yes	yes	--	yes	yes	--	--	--
Script debugger	--	--	--	--	yes	yes	yes	yes	yes	yes	--	--	--	--
Script syntax	--	C	own	Chart	C	C	C	C	Jscript	Basic	Basic	Basic	--	--

** - Shadow mapping can be displayed, but not generated without external tools.
$$ - Publishing or selling games requires further fees or fulfilling special conditions.
Chart based on manufacturer's specifications as of **July 2003**, if available - no responsibility is accepted for the correctness. If you find something incorrect, or if you want a certain engine to be included, or if you are a manufacturer and don't want your engine compared here, please notify webmaster@conitec.net.

Q. Is the network code and the game engine capable of handling large scale multiplayer gaming, as in 2000+ players? Or would it require lots of modification?

A. Theoretically an unlimited number of players can be handled. The real limit depends on the bandwidth needed for multiplayer communication, and the number of servers used. The faster and less predictable the players move, the more bandwidth is needed. A5 uses a very fast client/server system with a dead reckoning algorithm, but even so, in a worst case scenario (single server online air combat game) the limit on an average Internet modem connection can be only around 25 players. In a large scale online game however where the players are mainly walking or chatting, and the game world is split into several zones, much more players - several 1000 - can be handled.

Q. How often is the software updated to support emerging hardware and technologies or new graphical advances, and are the updates free to previous owners?

A. Major new features are added permanently in frequent free updates. You can follow the daily implementations of new features and the schedule for future features on the **beta** and **forecast** links on our forum. New engine generations (**A3 -> A4 -> A5 -> A6**) are released every two or three years and are not free.

Q. How bug-free is the software, and how stable are the games produced with it?

A. We have reason to assume that GameStudio is one of the most stablest and robustest game creation tools. However, no software can claim to be bug-free - just for the reason that absence of bugs can't be proven. We are going to great efforts to make GameStudio as stable and bug free as possible. Our company is **ISO 9001 certified**. Before a new version is released, it is tested for several months by our team of around 100 beta testers - all of them experienced game developers. After that, it is uploaded to our forum for a public beta test, which normally involves over thousand testers. Only then it becomes officially available on our download page.

Does this effort pay off? We think so. A user wrote us:***"I would like to put in something that's very important to our company regarding Gamestudio... We've now sold close to a thousand of our Kitten Girls that were created with this engine. Out of those thousand orders, there has not been one bug report. Not one. The only problem that has been reported so far is due to people not having the proper version of DirectX installed, and once they install it the product runs beautifully. Not one bug report - that's definitely a first, and this is the first product we've made with GameStudio. That tells me something about stability.***"

Q. Is it true that I must learn programming for creating a game?

A. You can build simple games, like 3D shooters, without any programming. However the more 'unique' your game idea, and the more 'special effects' you want, the more you'll need to describe the desired effects by scripts. The script language is a simplified version of the C/C++ programming language - therefore it's great for learning computer programming. If you've never programmed before, by working through the 6-day scripting tutorial you'll pick it up fast.

Q. Why C instead of BASIC?

A. BASIC is a first generation computer language, while C/C++ is a modern third generation object oriented language. For this reason, the C/C++ language is used for almost all professional software programming. Thus with C-Script you've done the first step towards professional programming. It is not harder to learn than BASIC. If you already know BASIC, it will take a day or two to get used to the C syntax. But once you've learned it, you'll notice that a C-Script program is more logically structured and easier to read and understand than a BASIC program. C-Script can be used for huge or complex projects that would be a nightmare - if not impossible - to program with BASIC.

Q. What is the difference between C-Script, 'real' C/C++, and Javascript?

A. C-Script is similar in syntax to Javascript or C/C++, but does without some language elements that are either rarely used, or are difficult to understand for a beginner. The main differences:

- In C/C++ you can define structs and classes. C-Script also supports structs, but only for predefined objects like entities or panels. You can not define your own structs and classes in C-Script.
- C/C++ has many variable types that represent numbers. C-Script has only one numeric variable type, an all-purpose combination of float and integer.
- Javascript uses the same 'var' type for numbers and text strings. In C-Script, as well as in C/C++, numbers and text are different types, and are handled with different functions.

Q. What is better: programming my game in C-Script, or programming it in 'real' C++ using the DLL plugin interface?

A. It depends. C-Script is much easier to learn and use than C++, but almost as fast, error-proof and shorter in code size. It does without all complicated stuff normally required for programming. The management of multitasking, object interaction, and a central "game loop" is automatically handled by the engine. Therefore C-Script is the best choice for a beginner - or for learning C++. 'Real' C++ on the other hand gives you more programming power and flexibility, and lets you add parts to the engine that aren't possible through scripts - like additional renderers for new types of 3D objects. It all depends on your capabilities and preferences.

Q. What was the first commercial game made with 3D GameStudio?

A. The first commercial games were the **3D Hunting** series by **MacMillan**. They were developed shortly after the GameStudio/A4 release by the end of 1999.

Q. What was the game with the most distributed copies so far?

A. Great Clips Racing, an advertising game with 1.5 million distributed CDs, developed with GameStudio/A5 by **Digital Content** in May 2002. The developer wrote us: ***"I am doing this all with the commercial edition of your software. Compared to what I paid for it ($150) and what I am getting from it, I am very happy."***

Q. What was the biggest project ever done with 3D GameStudio?

A. The biggest project so far (and probably **the largest 3D world created ever**) was the 3D reconstruction of the German city Quedlinburg for the "Planet of Visions" hall at the World Exhibition 2000. On a huge video screen, visitors could walk through the medieval center, and enter buildings, colonnades, towers, and cathedrals.

3D GameStudio was used for this ambitious and spectacular project, which was developed for the EXPO 2000 by Procon GmbH and the German Foundation for Protection of Historic Monuments (Stiftung Denkmalschutz). The Quedlinburg old town was created anew on the PC, using over 5000 photographs of historic facades and building interiors. Over 400 detailed buildings were reconstructed. The virtual city covered a walkable area of around two square miles. It was divided into several zones which ran on 600 MHz PCs with GeForce2 cards.