**Unit 70**

LO1 - Understanding the purpose of game engines

Title: Understand the functions of components of game engines



*By – Joseph Roper*

Content

Contents

[Introduction 9](#_Toc509997325)

[Purpose of game engines 9](#_Toc509997326)

[The History of Game Engines 10](#_Toc509997327)

[2D Game Engines 10](#_Toc509997328)

[3D Game Engines 10](#_Toc509997329)

[Mobile Game Engine 11](#_Toc509997330)

[Game Engines Advancement 11](#_Toc509997331)

[Construct 2 Tutorial 11](#_Toc509997332)

[Graphic Rendering 11](#_Toc509997333)

[Culling Methods 12](#_Toc509997334)

[Rendering Techniques 12](#_Toc509997335)

[Lighting 12](#_Toc509997336)

[Textures 13](#_Toc509997337)

[Fogging/Shadowing 13](#_Toc509997338)

[Depth Testing 14](#_Toc509997339)

[Anti-Aliasing 14](#_Toc509997340)

[Vertex & Pixel Shaders 15](#_Toc509997341)

[Animation Systems 15](#_Toc509997342)

[Path-Based 15](#_Toc509997343)

[Inverse Kinematics 16](#_Toc509997344)

[Forward Kinematics 16](#_Toc509997345)

[Particle Systems 16](#_Toc509997346)

[Systems 17](#_Toc509997347)

[Effects 17](#_Toc509997348)

[Networking 17](#_Toc509997349)

[Artificial Intelligence 18](#_Toc509997350)

[AI Agents (Bots, Non-player Characters) 18](#_Toc509997351)

[World Navigation (Pathfinding, Obstacle Avoidance) 18](#_Toc509997352)

[Behaviours 18](#_Toc509997353)

[Neural Nets 18](#_Toc509997354)

[Fuzzy Logic 19](#_Toc509997355)

[Middleware 19](#_Toc509997356)

[API - Application Program interface 19](#_Toc509997357)

[DirectX & OpenGL 19](#_Toc509997358)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name:** |  | **My own definition.** | **RESEARCHED DEFINITION (provide short internet researched definition and URL link)** | **DESCRIBE THE RELEVANCE OF THE RESEARCHED TERM TO YOUR OWN PRODUCTION PRACTICE?** | **IMAGE SUPPORT (Provide an image and/or video**  **link of said term being used in a game)** |
| **VIDEO GAMES / VIDEO GAME TESTING** | Demo | My Definition: A demo is a small version of a game which can be played through to intrigue players to buy the full game. | Internets Definition: A game demo is a freely distributed piece of an upcoming or recently released video game. Demos are typically released by the game's publisher to help consumers get a feel of the game before deciding whether to buy the full version.  <https://en.wikipedia.org/wiki/Game_demo> | This is relevant to my work as I will be making a demo version of my game to hand out to people to play. | [Image result for game demo](https://wholesgame.com/opinions/worth-time-money-pick-game-demos/) |
| Beta | My Definition: Beta is an incomplete version of a game that can be played and tested by the public, betas are usually created so a company can find information about bugs in the game and to also gain fans. | Internets Definition: A beta test refers to the distribution of pre-release game software to a select group of people so that they can test the game in their own homes. The beta version of a game is as close to perfect as the company can make it, but any bugs, glitches or other issues discovered in beta testing will usually be addressed before the game's official release.  <https://www.techopedia.com/definition/27136/beta-test-gaming> | This is relevant to my work as I will be making a beta version of my game to be played by the public so I can find out about any bugs within my game which I can fix. | [Image result for game beta](https://tomclancy-thedivision.ubisoft.com/game/en-gb/news/detail.aspx?c=tcm:154-237962-16&ct=tcm:148-76770-32) |
| Alpha | My Definition: Alpha is an incomplete version of a game that can only be played and tested by the company and developers, alphas are usually created so a company can find information about bugs. | Internets Definition: A very early version of a software product that may not contain all of the features that are planned for the final version. Typically, [software](https://www.webopedia.com/TERM/S/software.html) goes through two stages of testing before it is considered finished. The first stage, called alpha testing, is often performed only by users within the organization developing the software. The second stage, called [beta testing](https://www.webopedia.com/TERM/B/beta_test.html) , generally involves a limited number of external users.  <https://www.webopedia.com/TERM/A/alpha_version.html> | This is relevant to my work as while working on my game I will be using the alpha version to make improvements. | The video is hyperlinked onto the picture. |
| Debug | My Definition: Debug means to get rid of/fix glitches and problems in a game. This is usually done before a game is released but recently developers will release the game and add an update to fix all the bugs on day 1 of the games release. | Internets Definition: Debugging is the routine process of locating and removing computer program bugs, errors or abnormalities, which is methodically handled by software programmers via debugging tools. Debugging checks, detects and corrects errors or bugs to allow proper program operation according to set specifications. Debugging is also known as debug.  <https://www.techopedia.com/definition/16373/debugging> | This is relevant to my work as I will be debugging every game I make all until it is playable. | The video is hyperlinked onto the picture.  In this picture and video it shows that a debug mode was released for the game so if a glitch/bug causes you to get stuck in the game you can switch into debug mode and move the character around until you get to a safe area where you can continue to play the game. |
| Bug | My Definition: A bug in the video game industry means a glitch/ problem in a game which will cause annoyance for a player in their play through of a game. Some bugs can even make the game unplayable as it will stop a player from continuing on in the story. | Internets Definition: In IT, a bug refers to an error, fault or flaw in any computer program or a hardware system. A bug produces unexpected results or causes a system to behave unexpectedly. In short it is any behaviour or result that a program or system gets but it was not designed to do.  <https://www.techopedia.com/definition/3758/bug> | This is relevant to my work as in games I make there will most likely be a lot of bugs which I will have to fix. | The video is hyperlinked onto the picture. |
| Collision | My Definition: This is when two objects in a game collide with one another, this is used a lot when developing a game as you can set an event to happen when they touch each other. For example, you can make it so if a sprite (the player) collides with a tiled background/object (Lava) you can restart the layout basically meaning if the player touches lava the game will restart meaning the player died. | Internets Definition: Collision detection is the computational problem of detecting the intersection of two or more objects. While collision detection is most often associated with its use in [video games](https://en.wikipedia.org/wiki/Video_game) and other [physical simulations](https://en.wikipedia.org/wiki/Computer_simulations), it also has applications in [robotics](https://en.wikipedia.org/wiki/Robotics). In addition to determining whether two objects have collided, collision detection systems may also calculate *time of impact* (TOI), and report a *contact manifold* (the set of intersecting points).[[1]](https://en.wikipedia.org/wiki/Collision_detection#cite_note-1) [Collision response](https://en.wikipedia.org/wiki/Collision_response) deals with simulating what happens when a collision is detected (see [physics engine](https://en.wikipedia.org/wiki/Physics_engine), [ragdoll physics](https://en.wikipedia.org/wiki/Ragdoll_physics)). Solving collision detection problems requires extensive use of concepts from [linear algebra](https://en.wikipedia.org/wiki/Linear_algebra) and [computational geometry](https://en.wikipedia.org/wiki/Computational_geometry).  <https://en.wikipedia.org/wiki/Collision_detection> | This is relevant to my work as I will need to use this in virtually every game I make as collision detection is used when a player is hit in a game. | The video is hyperlinked onto the picture.  In this video it shows that when Mario the sprite collides with lava the object he takes damage. |
| Sprite | My Definition: A 2D or 3D character or object integrated in a digital graphic, it is created with bit maps or pixel art and is designed to be a part of a larger scene such as a world in a game. A sprite can be a static image or an animated graphic such as an isometric sprite. In terms of a video game an example of a sprite is any object in a game, it can range from Mario which is a 2D bit map to the flag pole/castle in Mario which is still a sprite but it just does something different. | Internets Definition: A sprite is a [bitmap](https://techterms.com/definition/bitmap) graphic that is designed to be part of a larger scene. It can either be a static image or an animated graphic. Examples of sprites include objects in 2D video games, [icons](https://techterms.com/definition/icon) that are part of an application [user interface](https://techterms.com/definition/user_interface), and small images published on websites.  <https://techterms.com/definition/sprite> | This is relevant to my work as I will definitely have to use sprites in every game I make as they are very useful. | [Image result for mario flag pole](http://www.nintendo.com/games/detail/super-mario-odyssey-switch)[Image result for mario sprite](http://www.cracked.com/blog/chatting-with-mario-during-a-game-of-super-mario-brothers/) |
| Scene | My Definition: A scene, also referred to as a cut scene is used a lot in video games to usually develop the story of a game and introduce new plot points. A scene is a non – interactive cut scene that displays a video with text or speech. | Internets Definition: A cut scene or event scene (sometimes in-game cinematic or in-game movie) is a sequence in a [video game](https://en.wikipedia.org/wiki/Video_game) that is not interactive, breaking up the [gameplay](https://en.wikipedia.org/wiki/Gameplay). Such scenes could be used to show conversations between characters, to the player, set the mood, reward the player, introduce new gameplay elements, show the effects of a player's actions, create emotional connections, improve [pacing](https://en.wikipedia.org/wiki/Pace_(speed)) or foreshadow future events.  <https://en.wikipedia.org/wiki/Cutscene> | This is relevant to my work as I will use many cut scenes in my games to make my games more interesting as it will have more variety while also providing the user with the information needed to understand what is happening with the story. |  |
| Frames | Frames are a sequence of images that are used to display an animation. | Internets Definition: In video and animation, frames are individual pictures in a sequence of images. For example, a Flash movie you see on the Web may play 12 frames per second, creating the appearance of motion. Most video is shot at 24 or 30 frames per second, or FPS. FPS is often measured in 3D games as a way of checking how fast the graphics processor of a computer is.  <https://techterms.com/definition/frame> | This is relevant to my work as I will be using frames in my games as they are very useful. I could use them to display the players life as 3 hearts |  |
| Event | My Definition: An event is a sequence of code scripted to trigger when a player performs a certain action such as Link pressing “a” to unlock a chest the jingle will play (From the Legend of Zelda game franchise) | Internets Definition: In [video games](https://en.wikipedia.org/wiki/Video_games), a scripted sequence/event is a pre-defined series of events that occur when triggered by player location or actions that play out in the games engine.  <https://en.wikipedia.org/wiki/Scripted_sequence> | This is relevant to my work as I will need to have events in my game to be able to start cut scenes. |  |
| Action | My Definition: An action is a bit of code that is run in a video game. For example, if I put on jump (when the player jumps) play sound file Jump.wav, this means when the player jumps a sound will play the action is the “play sound” bit.  The event is on jump the action is play sound | In events, **actions** do something, such as create an object or go to another layout. They appear to the right of the event.  <https://www.scirra.com/manual/77/actions> | This is relevant to my work as I will use a lot of actions in my game in order to add sound to movements in the game. |  |

# Purpose of game engines

Game engines are key to making a game able to run and work in order to be played, they allow a developer to create 2D or 3D games. Also, Game engines allow the user/developer to use collision detection, artificial intelligence, graphic rendering, sound and physics. As usually game engines will have the code and functions already included making it easier and more accessible to make a game.

Collision detection is a big part of video games as it allows the user to punch and kick and even walk because without it the player would just fall through the map/floor because the game engine wouldn’t detect the player’s body touching the floor.

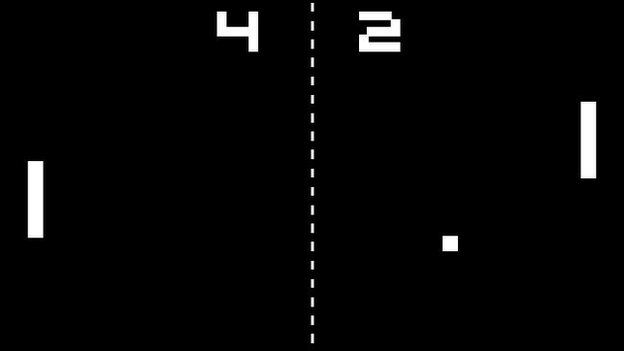
AI stands for artificial intelligence and is used to control every NPC (Non Playable Character - Bot) in a game. AI has rapidly developed since it was first introduced as it is now almost human like although this is because now a lot of money is being used to develop AI for specific games.

Graphic rendering is used in game engines to display objects in a game such as the player or a car. So without the rendering of graphics you wouldn’t be able to see anything while playing a game making the game virtually unplayable.

Sound in games are as important as the visuals of a game it is a necessary feature you must have to enjoy a game to its full potential. Sound in games is a necessary feature as it immerses the player into the movement, action and storytelling of a game by bringing it to life. Sound can convey a character’s emotions which makes it easier to understand what is happening and amplifies it giving the player a more enjoyable experience. Music in video games allow the player to truly be captivated as it sets the mood, the themes and emotions of a game. Having music in games brings the storytelling as it gives emotion and depth to the characters. And these sounds can all be added by using game engines.

Physics is now used within games to simulate realistic effects such as the speed and fall of a bullet, these effects are added into the game by programmers/coders which means that they can adjust the strength of certain physics for example the gravity within a game decreasing it to simulate the gravity on the moon.

Without these functions game engines have, games wouldn’t be games as you wouldn’t be able to make Pong because without collision detection the ball would just pass through the paddle.

[](http://www.bbc.co.uk/news/technology-33005297)

# 2D Game Engines

2D game engines are used to create 2D games across a range of platforms but wouldn’t include touch mechanics for these game, these would be purely for button games. To add touch for mobile games, you would have to use a mobile game engine. These types of games can be found on consoles, mobile devices and on the internet. These game engines will always incorporate flat images (sprites) due to the style of the engine. Everything in 2D always tends to be fairly simple such as the controls will limit to a few keyboard controls or mouse movement. If there are any AI characters in the game, they tend to be easy and the design just revolves around being a simple game.

An example of a 2D engine would be the Cocos2D game engine which created a variety of games, including the infamous Facebook game Farmville. The game engine itself follows a typical standard game engine but it is an open source game engine. This means that source codes and tools are freely available to anyone who wishes to create a game.

# 3D Game Engines

3D game engines are the most complicated game engines currently available at the moment. With the evolution of technology ever expanding, anything is possible within the limitations of a 3D environment. By far the most complicated of engines to use and this is down to the amount of factors there are for a single action. For example, you might be importing a mesh into the 3D space. That might be simple on a 2D engine where you only have to worry about 2 dimensions but for 3D, you have to worry about more dimensions. This is just one of the factors that differs the two engines.

The assets tend to require a lot more talent and skill to pull off and most professional companies will only take on staff with experience of creating the assets, such as the 3D animation of a character or the model of a Ferrari. A lot more time is also spent creating a 3D game, due to all the factors and assets a game needs in a 3D game.

An example of a game that uses a 3D game engine is TERA, a MMORPG game that was created using Unreal 3. Unreal Development Kit, developed by Unreal Games, is a games engine that has the tools to create a FPS game but can be adapted to create other games, such as RPGs.

# Mobile Game Engine

Mobile Game Engines are new to the scene in the game engine industry. What makes mobile game engines unique is that they incorporate other ways to control a character other than using a keyboard and mouse for example motion controls use the gyroscope in most phones to sense whether the player is tilting the phone left to turn left and vice versa. Mobile games are not the most graphical intensive games making them easier than other games to create however the hard bit of coding any mobile game is the controls as touch and tilt controls are fairly new to the industry meaning there are not really any pre-written scripts for a game engine to use to make things easier for the developer.

# Game Engines Advancement

[](https://www.youtube.com/watch?v=I6dB0Iw96eY)Game engines were unknown to the industry before 1983 when the first 2D game engine was made, used mostly to help develop old computer games like the classic pinball machine and plenty of others. Before game engines initial release every single game had to be coded from scratch leading to plenty glitches which slowed down the whole process of developing the game.

However, now game engines have advanced to the point now where you can make a full game without putting too much effort or at least not as much as before game engines, for example with the game engine Construct 2 you can assign your character with a “Platformer” variable making it so your character can jump and move around with the arrow keys. Previously this would have taken weeks without a game engine but now it can be done in a matter of seconds.

Game engines make everything more accessible to game designers allowing them to create their very own games with ease as features/functions are usually already included in game engines such as environments and characters in the form of packages.

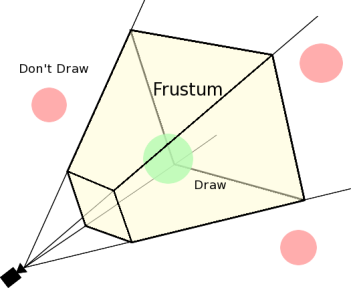
# Construct 2 Tutorial

I have included a PowerPoint Presentation with my work that goes over the basics of Construct 2 focusing more on how to add sounds to show how easy this game engine makes adding sound to a game for a developer.

# Graphic Rendering

## Culling Methods

Every serious 3D game engine uses the culling method to render graphical objects in a game. The frustum culling method saves processing power by only loading/rendering objects the player can see, their point of view. It also only renders the first face of the object not the back which saves processing power as well.

[](https://leonardogamestudies.wordpress.com/year-2/unit-70-game-engines/assignment-1-task-2-components-of-engines/)

## Rendering Techniques

The computer uses the motherboard to help render games for example when you click to shoot in a FPS (First Person Shooter) that process will be sent to the north bridge on a mother board to be rendered. However, if you click too many times and it over powers the north bridge it will be sent to the south bridge to collect until the north bridge is free again to render it. If a game is not rendered properly various glitches could happen such as screen tearing and delayed sound effects.

[](https://www.youtube.com/watch?v=W56cZC8Dk44)

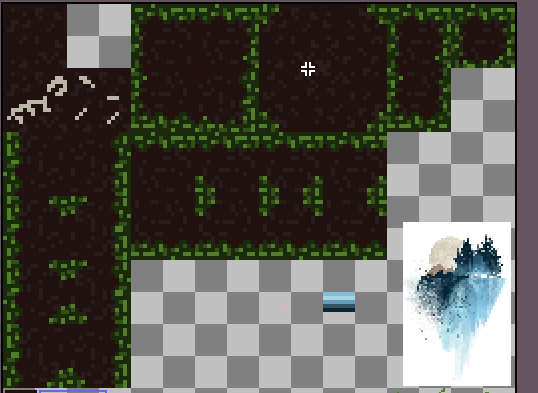
## Lighting

Lighting in video games can set the mood and theme of a game, and the game engine allows the developer to change this. In most game engines the developer can place lights around the world map of a game. With the light function you can control the radius, direction and intensity/brightness of a light, this means you can make a corridor scary or not by making the brightness of the lights lower than usual creating shadows in the game causing the corridor to be creepier than usual to the player. The picture down below is from a horror game, the first image is from the horror game and for the second one I used the brightness tool in Word to increase the brightness to show that lighting can affect the fear of a player.

[](https://videogamesuncovered.com/features/why-p-t-could-be-the-best-horror-game-of-2014/)

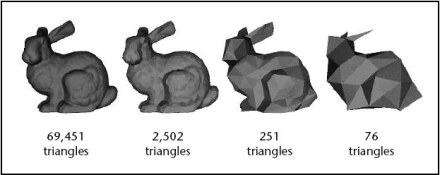
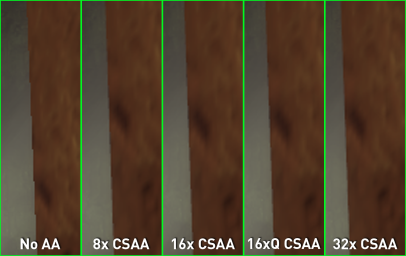
## Textures

Textures help the performance of the system while loading up a level in a game because textures make it so the graphics card only has to render one object rather than hundreds of the same. For example in a 2D game there will be an image of grass to be used on the floor this is a texture, the texture will then be duplicated across the floor to make it look like a forest or garden. However the computer will only have to render the original texture making the game run smoothly.



## Fogging/Shadowing

This technique also helps with increasing the performance of the system running the game, it basically softens the colours of the game by adding fog which makes the game system think there are more white or black pixels which are easier to load, which helps cover up the rendering process for the game. It helps as it disrupts the player’s visions of the in game models for characters and objects do the computer doesn’t have to fully load up the model as the player can’t really see it. Shadowing is the same technique as fogging just reversed as it makes more of the pixels black which the game system can easily render.



## Depth Testing

Depth testing is a rendering technique used to determine which objects are in front of one another at the same pixel location in the image so that the computer doesn’t have to render the concealed object. For example if there was a stack of boxes used in a game the computer using pixel depth testing will know that some boxes overlap so the computer won’t have to render the overlapped image to save processing power.

[](https://www.youtube.com/watch?v=DO5wsspiyEA)

## Anti-Aliasing

Anti-Aliasing is a technique used to make the game look appealing to the audience it does this by scanning all objects and characters detecting rough jagged edges and smoothing them out making the game look good. This technique makes games look infinitely better, however it comes at a price as it is very demanding on a computer’s hardware meaning only high spec computers can handle anti-aliasing. Anti-aliasing works by adding more pixels in between the jagged edges to smooth them out however this means the computer loads up more pixels than needed making the game lag and act slowly.

[](https://graphicdesign.stackexchange.com/questions/41069/correct-anti-aliasing-color-when-moving-to-a-new-background)

## Vertex & Pixel Shaders

Vertex shaders transform the attributes of an in game objects vertices such as; texture, colour, direction and position. Vertex shaders can quickly tweak a character model while a game is being rendered to make the game look more appealing. Although vertex shaders have many useless they are mainly used to render things like water in games as when the water ripples that is because the vertex shader is constantly changing the height of the vertices to create that simulation.

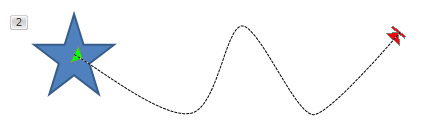
Pixel shaders is a technique that computers will use to make characters and objects look better, as this is the technique that defines what the colour of each pixel should be while also working out how to change the pixels to represent light hitting the object or a shadow casting over it.

# Animation Systems

## Path-Based

A commonly used animation system is a path based system, this is when animated objects in a game such as a door will have a path to follow rather than having each frame of it opening being hand drawn.

A path based system is an animation technique used to make animating a certain movement easier for the user, for example if you wanted to anime a ball bouncing up and down you would usually have to hand draw every single frame whereas if you use the path-based technique prominent in most animating software the computer will fill in the blanks itself so all you have to do is draw the ball then tell the computer where you want it to go by setting key frames of the balls original position then where you want its end point to be and it will just repeat the drawing along the path you made making an animation.

[](https://www.indezine.com/products/powerpoint/learn/animationsandtransitions/motion-paths-edit-points-ppt2010.html)

## Inverse Kinematics

Inverse Kinematics is a very useful animation technique used by professionals as it saves a lot of time and energy while animating, too describe how this animation technique works I will first have to explain the properties a human model will have in a animation software. A human model will have a skeleton inside it which the animator can edit by having the ability to move each body part individually, put together with inverse kinematics when the animator moves an arm it will also slightly move the other joints connected to the arm making a fluid and realistic motion.

[](https://www.youtube.com/watch?v=LDDjS4JGv6c)

## Forward Kinematics

Forward kinematics is basically a manual version of inverse kinematics as with this technique you have to move each joint yourself in order to animate the character, not only that but if you were to animate a hand you would have to move the arm before the hand as this goes along with the hierarchy system. Some animators prefer this over inverse kinematics as they feel like they have more control as they can fully manipulate the movements of the character.

## Particle Systems

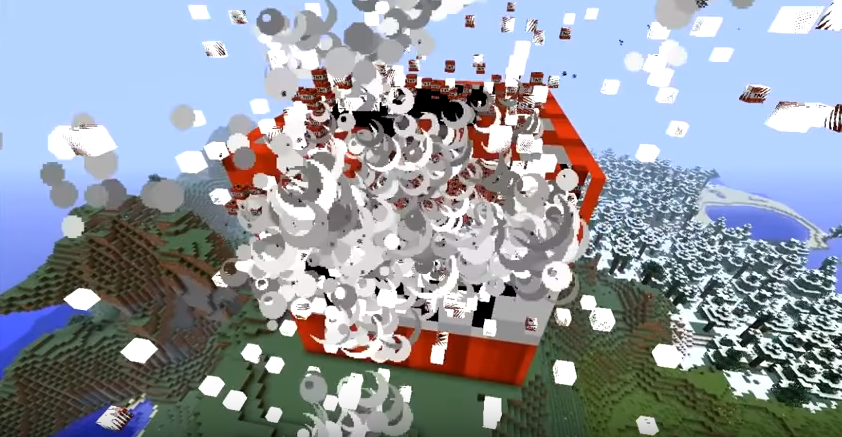
A particle system is a rendering technique used to produce in game assets such as fire, snow or rain. Fire in games is made up of a large number of tiny sprites in fires case wavering. The particle system will use an emitter acting as a source to produce all the sprites/particles to make up the fire. The particles/sprites produced by the emitter will contain information such as; colour, lifespan, size and velocity all so the particles fit in with the background and what they are supposed to be for example if they are meant to be simulating rain they will have to have the right colour and lifespan so that when they hit the ground they disappear making it look natural.

[](https://www.resetera.com/threads/water-rain-makes-games-look-better.8088/page-2)

# Systems

## Effects

Effects in video game engines can range from visual effects to in game audio and they are used to improve the players experience in the game however they do require processing power, every video game engine will allow the user to input these. An example of a visual effect that most developers use is an explosion for example a bomb in a game however when this visual effect is used it takes a lot of processing power to render, especially when a lot of bombs are places in a game it will most likely lag because the computer can’t render it in time.

[](https://www.youtube.com/watch?v=om5TwLETieA)

<https://www.youtube.com/watch?v=om5TwLETieA>

## Networking

Networking can make or break a game as networking in video games is usually used to allow online multiplayer available for games. Which means that if a multiplayer game has bad networking it will make the game highly unenjoyable as if a game is constantly buffering as the servers running the game can’t handle the input from all the players no one would play that game as you couldn’t play it properly. Although usually a game with good online multiplayer from networking would be more enjoyable for the player than a single player game as they can play with their friends.

Online game can also be ruined by the players by taking advantage of the networking code for example with the game Counter-Strike: Global Offensive used to be really popular but then too many people started hacking the game to their advantage making the game unfair, resulting in many people leaving matches and eventually not playing it in general. This is mainly the players fault but it is also the developers as they didn’t create a secure network which people couldn’t hack in to and exploit. As you can see from the picture down below this player is using a cheat commonly known as wall hacking which is where you can see enemy players through walls which is unrealistic meaning that the cheater can know where all of the enemy players are at any time making the game unfair.



# Artificial Intelligence

## AI Agents (Bots, Non-player Characters)

AI since it was first released has been progressively getting better especially within video games, AI stands for artificial intelligence and it is used to basically make non-playable characters (NPC) act more like humans making this important within games to make them more realistic.

## World Navigation (Pathfinding, Obstacle Avoidance)

One big flaw with NPC's AI is that they constantly run into walls not acting like a human would and that is why the pathfinding system was made. A pathfinding system will place waypoints that the NPC's will follow as well as having multiple routes put in place by the developer for the AI to follow to the waypoints with colliding with door's and other objects.[](http://acsanmebabb.hatenablog.com/entry/2017/05/31/065225)

## Behaviours

Behaviours are another subtle thing that makes NPC's seem more realistic for example in the game Grand Theft Auto 5 (GTA5) the police will have certain behaviours towards the player depending on where they are based in the map. As if you play as Franklin a black male in the game and go to the Los Santos Police station the cops will automatically start shooting at you for no reason other than being racist to make the game seem more realistic by giving the police depending on where they are located being more aggressive to the multiple playable characters.

<https://www.youtube.com/watch?v=ZXpfsYiG8Dc>

## Neural Nets

Neural nets are a massive leap into improving artificial intelligence as it basically makes the AI learn from constantly repeating an action and getting progressively better at it just like humans. For my example imagine a simple maze with a start and a finish, at the start 10 small triangles will spawn and go in a random direction and the triangle that gets the furthest, the code will take that triangles data and input it into all the others and the process will repeat until a triangle gets to the finish. A more detailed and advanced version of neural networks used in a fairly recent game is Super Smash Bros for the Wii U with the introduction of the Amiibos, Smash Bros is a fighting game where you can buy an Amiibo which is a personalised NPC which you can fight against and train it to become a better fighter.

[](https://www.amazon.co.uk/Fox-No-6-amiibo-Nintendo-Wii/dp/B00N8PBS0A)

## Fuzzy Logic

Fuzzy logic is used to further develop AI by making a NPC's decisions seem more human/realistic, this is done by adding more options for an NPC's response other than yes or no such as in the Halo game series a lower level/weaker enemy will have a small amount of aggression towards the player for example a grunt will sometimes rather than shoot, if it’s on its own it will run away and call for help. Whereas, if it is a stronger enemy it will have a lot of aggression towards the player by following and targeting them.

# Middleware

## API - Application Program interface

An API basically the middle man that helps two programs communicate with each other. An example of this is with the widely popular online video game store called Steam. In Steam you can buy a game but you don’t actually buy the game you buy a product key that lets you install the full game off Steams server. Steam is the API as when you buy a game the product key makes it so that your computer can communicate to the game for it to download to your computer.

## DirectX & OpenGL

DirectX is a built in system that contains a collection of API’s for the computer to use to communicate to certain programs for example; audio, video and games direct sound. DirectX comes free with Windows and can only be used if your operating system is windows and because DirectX is so good and beats all the other competition most computers run Windows as their operating system rather than something like Linux that uses OpenGL instead of DirectX.

OpenGL stands for Open Graphics Library and is a free graphics API used in operating systems such as Linux, it helps the computer access GPU (Graphics Processing Unit) from this OpenGL will be able to access its built in graphical functions such as; antialiasing, texture mapping, pixel operations and atmospheric effects like fogging and smoke. OpenGL is also open source meaning people will be able to change the code of the API to help improve it although sometimes people will change it to try and hack into the computers operating system or even games.

[](https://mirazmac.info/read/microsoft-directx-all-versions-free-download.html)[](https://www.anandtech.com/show/9506/opengl-siggraph-2015-opengl-es-32-opengl-extensions-announced)