Concepts and features of game engines essay

A

A

Introduction

This report will cover almost everything consumers and entry level developers may need to know or be interested to know about game engines, how they work and what they're used for. This report will cover topics such as the history of game engines, different forms of engines (across multiple platforms), how game engines can be used by the community to create interesting things such as mods, plugins and standalone games.

Not only that but I will cover the purposes of using a game engine, the various components involved in the game engine, how they are used and how they have evolved over the past 2-3 decades into what they are today.

I will also look at what the future of game engines with technologies such as Artificial Intelligence and Graphics Rendering.

The history of game engines1989 – Space Rogue – Origin Systems

The Space Rogue engine, created by Origin Systems, was released in 1989. This was a 3D engine and featured Texture Mapping. An addition that makes surfaces look more realistic or 3D, surfaces that would otherwise be flat, undetailed and dull.

1993 – Doom – ID Tech

The Doom engine was created by ID Tech in 1993. This engine features a hybrid world which consists of 2D sprites in a 3D world. 2D sprites are characters in the game that are 2D and look the same regardless of what angle you are looking at them from. If you walk around them to what should be the back, the characters will rotate to face you.

1994 – Storm Keep – Storm Keep

The Storm Keep engine was created by Storm Keep in 1994. This is a 3D engine in which the key feature was Motion Capture. This is where a person in real life wears a special ‘ suit’ which is hooked up to a computer which tracks movements and then applies them to a 3D model which can then be added to a game. This was used in Storm Keep, but not very well. Since the technology’s appearance in Storm Keep, it has been vastly improved and better versions of this can be found in newer games such as FIFA.

1995 – Quake Engine – ID Tech

The Quake Engine was created by ID Tech in 1995. This 3D engine featured multiple important features which can be found in most of the games developed today. These include advanced lighting, for example, shadows and Culling. Culling is the way a game engine only renders what’s in sight. For example, if you have a large wall in front of you and behind the wall is some mountains – provided the mountains are covered up by this wall, the character would not be able to see the mountains behind the wall. If the wall was removed and nothing else was blocking the view of the mountains, the mountains would be rendered.

1997 – Renderware – Epic Game

The Renderware engine was created by Epic Game and released in 1997. This 3D engine was used in the development of hundreds of PS2 games and could be used across multiple platforms.

1998 – Unreal – Unreal

The Unreal engine was created by Unreal and released in 1998. The first game to be released on this engine was Unreal, followed by Unreal Tournament. This engine was designed for the development of first person shooters, but was later used to develop roleplaying games. This engine also features a map editor.

2001 – Cryengine – Crytek

The Cryengine was created by Crytek and released in 2001. This 3D engine featured pixel shaders, instead of vertex shaders. Pixel shaders result in more detailed textures because instead of taking relatively big portions of a texture and applying colours, like with vertex shaders, pixel shaders split these vertexes into individual pixels and texture each of them individually. As a result of the extra detail pixel shaders provide over vertex shaders, games that run the Cryengine require the user to have an insanely powerful PC to render the world and for the game to be playable.

2006 – Frostbite – Electronic Arts

The Frostbite engine was created by Electronic Arts (or EA) in 2006. This 3D game engine was used in making the Battlefield games and Dragon Age.

One of the key features of this engine is the destructible environment. This means that developers have to tools to give players the ability to use explosives or rocket launchers to blow holes in walls which they can then walk through, or collapse buildings with tanks like in Battlefield 4.

2D engines

The purpose of a 2D game engine is to quickly and easily enable a developer to make a 2D engine without having to recode the core elements along the way. Examples include:

Scripting languages; Artificial intelligence, or bots; Controls; Physics; Among others.

Features of 2D game engines include:

Scripting languages – flash; Visual script – dragging and dropping; Artificial intelligence; Physics; Sounds; Automatic scrolling; Built-in controls.

Examples of 2D game engines include:

Gamemaker; Construct 2D; Unity 2D; Scratch.

2D engines are designed for the development of simple games with basic features such as sprites. This is different to a 3D or mobile game engine as a 2D engine is *not* being designed with the idea of having the most advanced and capable features or with the idea of developing a simple and portable game which is optimised for smartphones and tablets.

In a similar way to certain platforms not being suitable for certain games, certain game engines are not suitable for the development of certain games. For example, it is very easy for a developer to make a 2D game on Gamemaker (2D engine), but the same game would be very difficult to make on UDK (3D engine). However, it is easy to make a 3D game on UDK (3D engine), but it would be difficult to make the same game on Gamemaker (2D engine).

3D engines

The purpose of a 3D engine is to quickly and easily enable a developer to make a 3D game without having to recode the core elements along the way. Examples include:

Scripting languages; Artificial intelligence, or bots; Controls; Physics; Auto scrolling; XYZ coordinates; Among others.

Features of 3D engines include:

Scripting languages – flash; Visual script – dragging and dropping; Artificial intelligence; Physics; Sounds; Automatic Scrolling; Controls; XYZ coordinates; Mesh (i. e. trees, bushes, etc).

Examples of 3D game engines include:

Rockstar Advanced Game Engine (RAGE); Unreal Development Kit (UDK); Quake Engine; Space Rogue; Renderware; Storm KeepFrostbite; Source; Unity; Doom.

My favourite game engine of all of them is the Frostbite engine. This is the engine that powers the Battlefield series. I love the way that the destruction component is carried out in this engine. I love the fact that you can collapse buildings with your handheld rocket launcher or hop into a tank and destroy the support beams of a skyscraper. It’s also great when you’re looking for the edge in a first person shooter, you can grab your explosives and blow a massive hole in a wall which you could then use to snipe out of. I think the destructible environment is definitely one of my favourite features.

A 3D game engine is designed to develop a game which has advanced features, one which requires 3D models, terrain, enhanced realism and increasingly a number of other things; such as, artificial intelligence, destruction, physics and sound.

3D engines are different to 2D and mobile engines as a 3D engine is not being designed to be lightweight, simple or portable. 3D engines are designed for developing games which require powerful hardware to run its games. Games developed using a 3D game engine are likely unsuitable for devices such as laptops, smartphones and tablets as they have insufficient graphics cards and inadequate cooling solutions.

Mobile engines

The purpose of a mobile engine is to quickly and easily enable a developer to make a mobile game without having to recode the core elements along the way. Examples include:

Scripting languages; Artificial intelligence, or npcs/bots; Physics; Controls; Automatic scrolling.

Features of a mobile game engine include:

Scripting languages; Artificial intelligence, or npcs/bots; Physics; Controls; Automatic Scrolling; Sounds; Meshes.

Examples of mobile game engines include:

All Binary Platform Engine; Android Box 2D.

Mobile engines are designed for making games which are lightweight, portable and don't require a large amount of processing power to run smoothly. This game engine is designed to make games for platforms such as smartphones and tablets.

This type of mobile engine is not suitable for developing 2D or 3D games. This is because these types of games require an engine designed with more features, better optimisations, in some cases, a level editor and more powerful computers. For this reason it would be very difficult for a developer to make a 2D or 3D game using this engine. Similarly, it would be very difficult for a developer to make a mobile game on a more powerful and unsuitable game engine such as Gamemaker (2D engine) and UDK (3D engine).

Game mods

A game mod is a way in which a 3rd party developer can modify a game. This may be by removing, modifying or adding features or gameplay elements. This is usually done with the hope of enhancing a game, making it more fun or easier to play. Notable examples of game mods include:

Gary’s ModGary’s Mod is a popular sandbox physics game built on the Source engine, originally designed as a game mod for the Valve game, Half Life 2. In 2006, due to the popularity of Gary’s Mod it was then developed into its own standalone game which is available to download today on Steam. Gary’s Mod has no objectives, but gives the player a sandbox where players can do as they wish, free of restrictions. DayZ ModThe DayZ Mod was originally developed as a free extension to the popular game Arma 2. The DayZ Mod was a massively multiplayer online post-apocalyptic zombie game. The idea behind it was that players must survive the harsh original world by gathering supplies, ranging from food and clothing, to military equipment such as assault rifles and bullet proof vests. These supplies could then be used to fend off against zombies or other threats to life. The mod was so popular that in 2013, the developers released a full standalone version which was better optimised and used a new game engine altogether. The Stanley ParableThe Stanley Parable is a game mod built on the Source engine. The mod was designed as an interactive story modification. The mod features no combat or action-based sequences, but instead the player is able to guide Stanley through a dreamlike environment. The Stanley Parable is available to buy on Steam. Due to the success of game modifications, such as The Stanley Parable, game developers have begun sponsoring teams of game modders in order to promote the development of standalone game modifications for their games. Purposes of game engines

A game engine is the core developer tool used in developing computer games. This software is used to more quickly and efficiently enable a developer to create a game for individual or multiple platforms without having to recode core elements and game functionality.

For example, the physics component of a game engine is responsible for providing the developer with the tools, or in other words, a reference library of game code which is used to implement physics and other components into their game. So, if the developer wanted to implement a gravity effect after a player jumps and reaches certain height – pulling the player back down, the developer could make reference to a chunk of code in the reference library with a single line of code, in a similar way to how a developer would use an API; therefore allowing the developer to implement the physics element without having to unnecessarily recode, in this case, gravity and jumping.

Game engines make game development much faster and more efficient as less code is required to carry out a particular function and less time is spent retyping the same code repeatedly. Utilising a game engine can also result in improved game performance on users' PCs, as less processing power, memory and hard drive space is required to download, install and run the game code.

Game engines are often wrongly confused for being the computer game itself. Game engines are used to make games development easier for programmers and to be adaptable to suit the needs of the developer and the game being developed.

A simple explanation of a game engine would be to imagine the setup of a car and its engine. The car body is what you can physically see and the engine is what enables it the car to drive. In this analogy, the computer game is a car's body and the game engine is a car's engine.

In a game, the game engine is responsible for ' driving' many of the game's core components, such as rendering what you see on screen and implementing other technologies including physics, collision detection and artificial intelligence – the most important component being the rendering facility in 2D and 3D engines. A developer will use the design of the game engine (the car engine) to build up the body of the car around it (the game outputted to a display). Car engines can be moved and adapted to suit different car bodies, in a similar way to game engines.

There are good examples of game engines, but there are also bad ones. Game engines which are well designed can be truly impressive – those who fail to impress, can be quite pathetic.

An example of a good and modern game engine would be the Frostbite engine, the Source engine or the RAGE engine. Examples of how poor quality game engines were used to make poor quality games are clearly present in the games, Big Rigs and Sonic Generations.

In the Frostbite, Source and RAGE engine, many of the top tier games available to buy in 2015 and beyond, have been and will be developed using these engines; including, Battlefield 4, Left 4 Dead 2, Team Fortress 2 and Grand Theft Auto 5. Components including graphics rendering, physics, destruction, artificial intelligence and collision detection were properly implemented which has resulted in these games being hugely popular and fun to play.

In Big Rigs and Sonic Generations, the gameplay experience was, for the most part, less than satisfactory. Collision detection and physics being some of the notable examples of where the worst these games have to offer are put on show. Things such as trucks driving up an almost 90 degree mountain and trucks being able to drive through lamp posts and fences (as if they aren’t there) or Sonic (the hedgehog) ‘ swimming’ through walls, windows and doors.

This is a screenshot of bad collision detection in Sonic Generations.

Source:. 14/10/15.

Sonic can be clearly seen ‘ swimming’ through a window and if you continue to watch the video you will see he is able to walk through walls w/o the game doing anything at all to stop this.

This is a screenshot of a truck driving up a cliff face (90 degree angle) very easily and at speed.

Source:. 14/10/15.

What can be seen in the video and this screenshot of Big Rigs is completely unrealistic and would not happen in real life. This is an example of poorly designed physics and collision detection in an engine.

Components featured the game engine, Frostbite 3, include:

Controls; Physics; Sound; Collision detection; Artificial Intelligence; Destruction; Graphics Rendering; Animation; Cinematics; Visual Effects; Scripting languages – dragging and dropping; Culling; And more.

I would use this game engine in relation to the components listed above as they provide me, the developer, with the ability to create a destructible environment; in which I could use a rocket launcher to blast a whole into a wall which mine and other characters could then walk through. Not only that, but I would also be able to create an environment that looks stunning – one with immersive cinematic and visual effects, which are complemented by the impressive sound options offered by this game engine.

The development of game engines over the years has aided the development of games significantly. Early developments of the game engine component, Destruction, that can be seen in this video: are made to look rubbish in comparison to the incredible destructible environment depicted in this video of Battlefield 4 gameplay:

This is a screenshot of the destruction component from the videos above in Red Faction:

This is example of the destruction component from the videos above in Battlefield 4:

The first picture, which depicts an early form of the now very well developed game engine component destruction, shows a player with a rocket launcher firing rockets at a wall and blowing a clean whole in the wall – with almost no debris left after the explosion. Now, look at the screenshot of the Battlefield

4 gameplay. This shows a tower which has just been hit from the back with a missile collapsing entirely and falling over before smashing into many large pieces of debris. The improvements overtime of how destruction actually works in games is a clear example of how game engines have developed overtime and more specifically the destruction components, to the clear advantage of 2D, 3D and mobile games, such as above.

Graphics rendering

Graphics rendering is the way in which a game engine loads the environment around you. It does this by taking a ‘ wire-frame model’ and applying different components to the individual vertexes. Textures which have had these components applied to them are known as 3D models. These components include:

Shade; Colour; Texture; Reflection; Shadow; And more.

We can make a ‘ wire frame model’ 3D using software such as Blender, Cinema 4D and Autodesk Maya. We can then import them into game engines like UDK and import them into a game. This software is very expensive, but there are good deals available, if you’re a student.

Graphics rendering is used in almost every (good) game available today. There are good examples of graphics rendering, but there are also bad ones. Examples of good and bad include Batman Arkam Knight and Sonic Generations, respectively.

This is a screenshot of Batman’s cape in the game, Batman Arkam Knight:

Source:. 14/10/15.

This is a screenshot of Sonic in the game, Sonic Generations:

Source:. 14/10/15.

The graphics rendering that takes place in Batman Arkam Knight is on a whole new level in comparison to the rendering that takes place in Sonic Generations.

The screenshot of Batman Arkam Knight shows Batman’s cape. This cape is made up of approximately 20 thousand vertexes, in which graphics rendering components are applied to every pixel individually – a technology known as pixel shaders. This game is also known to run at a buttery-smooth 60 frames per second. This is incredible considering the immense amount of graphics performance required to render 20 thousand vertexes and then each of the pixels individually, 60 times per second. Most likely, one of the only reasons this is practical to run is because of the optimisation for certain hardware the game developers will have performed.

Sonic, on the other hand, is made up of approximately 200 vertexes and runs at a significantly less smooth 30 frames per second. This means that the game which is already significantly less detailed, only runs at 30 frames per second. That means that the graphics processor only has to render 200 vertexes, 30 times a second. In truth, this may sound like a lot, but in comparison to Batman Arkam Knight – it doesn’t even scratch the surface.

One of the reasons it is so complicated designed a game and optimising it to run with computer hardware properly at reasonable frame rates is because, most of the time, especially in sandbox games you don’t know what the player is going to do until they do it. So the game has constantly got to render what is in the players’ view and nothing else. If the game rendered everything all at once, most computers would not be able to run it due to the immense amount of processing power required and if they could, it most likely wouldn’t be a very enjoyable or playable experience.

Collision detection

Collision detection is the component of a game engine which recognises two objects have interacted with each other when they collide and, in most cases, providing a response.

An example of this is a player in a first person short being shot in the head which causes the player to die. In some instances, the corpse will have a ragdoll effect (the body will flop to the ground in a random way. Another good example of collision detection is a player that is underwater only having 10 seconds from the time they entered until they drown. In summary, collision detection is interaction and response.

This is a screenshot of bad collision detection in Sonic Generations.

Source:. 14/10/15.

In the screenshot, Sonic can be clearly seen ‘ swimming’ through a window and if you continue to watch the video you will see he is able to walk through walls w/o the game doing anything at all to stop this. This is an example of bad collision detection.

This is a screenshot of good collision detection in Fifa 16.

Source:. 15/10/15.

The screenshot above shows a Football player in Fifa 16 who is about to kick a ball. This is an example of good collision detection. This is because as soon as the player's foot makes contact with the ball, the foot doesn't go through the ball it stops on the surface and the ball goes flying across the pitch. This is how collision detection should and does work in Fifa 16.

This is, however, not the case in Sonic Generations. When Sonic walks hit the building in Sonic Generations, he should hit the wall and have a sore face – not go ' swimming' through as he can be seen doing in the screenshot and videos above.

Poor collision detection not only makes the game look broken and unfinished but it can sometimes make the game almost impossible to play. A simple example of collision detection would be to consider the game, Operation. This is when you have to extract bones from the character on your operating table without touching the sides of his insides which would result in a buzzer sound.

Artificial intelligence (AI)

Artificial intelligence is the component of a game engine which enables developers to make non-playable characters mimic real-life humans or animals in a game. This can be done by using scripting languages and artificial intelligence components of a game engine to implement pathfinders. A pathfinder is the way an npc is made to go from point A to point B, walk back and so on.

In Metal Gear Solid 5, artificial intelligence is used to make npcs follow predefined paths and mimic real life humans. The npcs continuously follow their paths on repeat until they detect the presence of a character that should be there.

An interesting thing players can do to disguise themselves (and trick the npcs) is by wearing a cardboard box with a picture of a saluting soldier on the side of it and lie still whilst the npcs looking or investigating them. The npc then sees the image of the solider on the side of the box and is made to believe the picture to be a real soldier in the game, before returning to the pathfinder it’s been set. If the npc does investigate and find that the player is there when they shouldn’t be, the npc will engage in combat with the player, usually until the player or npc is dead.

This screenshot shows a soldier on a cardboard box being used as a disguise as the enemy.

Source:. 15/10/15.

Other examples of ways the npcs can be tricked is by putting a picture of a naked lady on the side of the box. This causes the npc to come running towards the box – believing that the picture is a real lady. At which point, the player can then sneak past or kill the soldier whilst he is distracted. This an example of good collision detection in a game.

This screenshot shows a naked lady on the side of a cardboard box being used as a distraction.

Source:. 15/10/15.

An example of bad artificial intelligence is in Mario Kart 64. When you are in a race, once your character is a certain distance ahead of ‘ bots’ you’re playing against, they automatically get a speed boost to catch up with you. This doesn’t make sense, it is unfair and is a clear example of how artificial intelligence can be very poorly implemented and not very well thought out in games

This screenshot shows a Mario Kart game where the player is playing against bots and is in first place. He has just completed a full lap around the bots and as soon as he reaches them, they get a speed boost and suddenly they are all travelling much faster him.

Source:. 15/10/15.

When you compare artificial intelligence in Mario Kart 64 and Metal Gear Solid 5. Metal Gear Solid’s artificial intelligence is on a completely different level to Mario Kart’s. In Metal Gear Solid, the AI is well thought out, it makes sense and it performs as advertised. In Mario Kart 64, as soon as you’re winning the race by so much the game compensates for the poor driving performance of the bots and gives them a completely unrealistic and quite frankly, ridiculous speed boost advantage.

An early example of artificial intelligence and collision detection is Pacman. In Pacman, the ghosts move around in random directions. If the ghosts go within a collision detection box with pacman inside it, the ghosts are then able to ‘ see’ Pacman and then head towards him for a meal.

The screenshot below shows a pacman being chased by a ghost in Pacman.

Source:. 15/10/15.

The future of artificial intelligence in game engines is coming to be a reality very fast. Developments of the new engine, Snowdrop will almost certainly bring new gameplay elements and functionality to the table. “ The Snowdrop engine is being built around a node-based system, which affects everything from rendering to Artificial Intelligence, mission scripting and the user interface.”

Source:. 15/10/15.

The future of artificial intelligence in real life could be right around the corner. Already we are seeing new developments such as ‘ mind clones’ are shown below. These are robots which have the thoughts, feelings and memories of real humans stored on their hard drives. Scientists are trying to develop the ways in which these robots can interpret this information and use it.

This is a screenshot of a video of a real life ‘ mind clone’.

Source:. 15/10/15.

Sound

Sound is a key part of almost every game ever created. Sound provides music that can be played in the background, it provides the ‘ scary’ music “ for situations fraught with danger and excitement, the music would be fast-paced and present a sense of immediacy in the situations, making the player tense up and ready themselves for a battle.”

Source:. 07/10/2015.

When the situation is less tense or dangerous, slower, more relaxing music will begin to play. This transition between the speed and change in tone or beat of the music present could be used to the tactical advantage of the player, as it alerts them to the fact that the situation or mood is changing.

If you want to talk about how sound enhances the game, think about a horror game or first person shooter game on a battlefield without any sound whatsoever. It wouldn’t work. You wouldn’t have the same sense of danger or urgency in a horror game without the ‘ scary’ and fast paced sound that is present in these sorts of games.

It is as equally important to have the right sort of sound in the appropriate genre of video game. For example, you most likely wouldn’t have positive, upbeat and relaxing music in a horror game. You would have slow, fast or loud music in a horror game that makes you tense up, sense the urgency and send chills down your spine when you see something you didn’t expect.

Jump scares wouldn’t be nearly as scary without the music. They’d just be annoying or give you a little scare. Whereas if you do have music, not only is it a shock to your eyes, but if you haven't seen it before, the sound makes you panic and react quite badly.

This is a screenshot of a video of someone who played the ' Scary Maze Game' and got a massive scare after seeing this on screen and hearing a very loud roar.

Source:. 15/10/15.

If you wanted to consider another example of how sound enhances the game, look at Fifa 16. Imagine you're in the middle of the field and you're wearing the appropriate headset, you can hear the crowds cheering loudly from all sides due to a technology call 3D sound. This can be truly immersive for the player. The sound is so convincing and immersive that you feel like you're actually there. You wouldn't have this if there were no sound in the game – all you would be doing is pointlessly kicking a virtual ball around a virtual field. It wouldn't be as enjoyable or much point without very important sound.

There are many different types of sound. Sound has evolved significantly over time. It started with a basic form of sound called Mono sound. This is sound which will only come out of one channel (one speaker). This form does work, however, If you're listening to a game in Mono sound (therefore, on one side only) and you're playing on Metal Gear Solid 5. If you have a guy walking behind you, you won't be able to tell what direction he's coming from. This is when Mono sound evolved into Stereo.

Stereo sound put simply is Mono sound that uses two channels (two speakers). This works too, but further developments have been made that have made sound in computer games even more immersive. This includes Surround Sound 5. 1 and 7. 1.

Surround Sound 5. 1 and 7. 1 is sound which uses 6 speakers and 8 speakers, respectively. Disadvantages of this are that you must be in the centre of all the speakers to experience the full effect and benefits of Surround Sound. This setup is not always practical due to the many cables that will trail on the floor, under the sofa, under the floorboards, in the walls, etc. Wireless surround sound speakers are an option, but then you must consider a TV that is compatible and account for the latency you may experience between data transfer from the TV to each of the speakers.

Surround Sound does work for the most part, however, a development of this was made which lead to a new and interesting technology called 3D sound.

3D sound is surround sound with spacial awareness. Meaning that you can more easily to pinpoint which direction a sound is coming from as well as being able to visualise how far away it is. You could hear a character walk from left to right behind you without you being able to see that the character is there.

Even though 3D sound is still very good, some of the same impracticalities present in Surround Sound are present with 3D sound, such as the cables and the need to somewhat be in the centre of the speakers to experience the benefits of 3D sound. Therefore, 3D sound is evolving into a newer form called Violet 3D.

Violet 3D is sound that projects sound in a similar way to a lightbulb. This means that there are large listening angles (unlike surround sound) and sound will go in all directions – instead of just bouncing off the walls. Speakers adjust themselves based on the sound and your location in the room.

Computers are digital, meaning everything is made up of 1s and 0s. Sound projected using the technologies above is digital, which means that we need a piece of hardware called a sound card to convert sound from digital to analogue so that we can understand it. Sound cards can either be a dedicated piece of hardware or be built into the processor as an on-board sound.

Physics

Physics is the component of a game engine which is responsible for controlling how realistic the game is in comparison to real life or simply how players, characters, etc interact with the world.

An example of physics in a game is Gravity. So, when a player jumps and they reach a certain height defined by the developer, gravity will pull them back down. Not only that but you could have it so that if a player is standing in water, sand, mud or quicksand, they will sink. This could be slow or quick and it will depend on the substance the player is stood in to determine how easy it should be to get out.

Other examples include Bullet Drop. This is where as a bullet travels through the air, it will gradually be pulled towards the ground. The other of physics in a game engine being Cars crumbling after they crash into walls, fences, etc.

An early example of physics in a game is in Mario. When Mario jumps onto the flag, he is able to ride it gently down, instead of just falling through the flag.

The screenshot shows Mario riding a flag down the flag pole gently. This is an example of physics.

Source:. 15/10/15.

Now, if we were to compare Mario, which can be seen above, to something like BeamNG Drive, the difference is incredible.

The screenshot below shows two cars in BeamNG Drive crashed into each other.

Source:. 15/10/15.

Where we are now with physics in games as can be seen in BeamNG Drive is far from what it was in the Mario days, where the most advanced physics addition to the game was that you could ride a flag down a flag pole.

In today's games, if you drive a car, truck or something else off a cliff, it will fall differently every single time. The way vehicles, players, animals, etc interact with the world is different every time, but the rules never change. Whereas, if you look at physics in Mario, nothing ever changes. If you jump, you can only jump so high; if you fall, you fall the same way every single time. Improvements in physics technology in game engines and the tools available to developers has made these advancements in games possible.

Conclusion

In conclusion, many things have improved since Space Rogue, the first game engine. Today's game engines are really on a completely different level. Almost every component of the game engine from graphics – sound and collision detection and artificial intelligence – physics. Everything has been improved or made more enjoyable in some way or another.

Examples of how game engines have advanced wouldn't be complete without Graphics rendering has advanced from being a couple of pixels or a couple of vertexes to thousands of vertexes with hundreds of thousands of pixels to be rendered 60 times per second.

Aside from graphics, collision detection continues to become more and more advanced as the weeks go by. More and more game engines are being developed or improved to handle collision detection better all the time. All you have to do is look at how much collision detection has changed from Pacman (where the ghosts would head towards the pacman when they detect its presence inside a collision box) to where it is now where games can detect you've gone underwater and that they know you've drowned when you stay under there for so long or the fact that you can drive cars into walls and they'll crumple and break up into pieces, as is the case in BeamNG Drive.

Artificial Intelligence is becoming more and more realistic all the time. Metal Gear Solid 5 is a prime example of how AI has significantly improved since Pacman. Developments of the new Snowdrop game engine by Ubisoft comes with the promise of even better and more realistic AI.

Sound and physics continues to be more and more crucial in game engines, both having seen many new technologies and developments over the years. I have no doubt that we will begin to see even more interesting developments in the coming months and years in all the component areas mentioned in this document and the development of new ones.