GAME ENGINES
Definition

A game engine is “a piece of software designed for the creation and development of video games”. It offers a number of components and/or tools which can be (re)used to make game development process more easier.
Functionalities

• Rendering engine (2D/ 3D)
• Scene graph.
• Collision detection (and response)
• Physics engine.
• Scripting .
• Animation.
• Artificial intelligence.
• Sound.
• Networking.
• Streaming.
• Memory management.
• Threading.
Advantages

1. Less Code
2. More productivity = less time!
3. Usability (easy to use)
4. Compatibility with design programs (integration)
5. More tools, more options
6. Plugins and libraries.
7. Cross platform
The target hardware layer on which the game will run. A game engine may be capable of running on top of a number of hardware layers.
Typically some means of communicating with the underlying hardware is needed. This can be accomplished using either device drivers or through APIs exposed by the OS.

In some cases, e.g. PCs, the OS strongly controls execution of the game engine (which is one executing process amongst many).

Most game engines use 3\textsuperscript{rd} party SDKs to provide access to:

- Ready made data-structures and algorithms (e.g. STL or Boost in C++)
- Graphics hardware (e.g. DirectX or OpenGL)
- Collision detection and Physics (e.g. Havok, PhysX or ODE)
Platform Independence

Game engines which can run across multiple platforms often provide an abstraction layer hiding platform specific differences from higher-level layers.

The platform independence layer assures that standard data types, libraries, fundamental APIs, etc. offer consistent behaviour across the different supported platforms.

Supported abstractions can include:

- Atomic Data Types
- Collections and Iterators
- File System
- Network Transport Layer
- Threading Library
- Graphics Wrappers
In order to function game engines typically depend upon underlying utility and support classes. Depending upon the game engine, this might include:

- Memory Allocation
- Math Library
- Debug Printing and Logging
- Localization Services
- Profiling / Stats Gathering
- Random Number Generator
- Assertions
- Parsers
- Movie Player
- Network Transport Layer
- RTTI / Reflection & Serialization
- Asynchronous File I/O
The resource manager provides a unified means of accessing the different types of game asset and other engine input data, including but not necessarily limited to model meshes, textures, fonts, skeletal animation data, game/world maps, etc.
Rendering Engine

The rendering engine within a game engine tends to be a large and complex component which is often decomposed into separate layers.
Renderer

This component encapsulates all the raw rendering facilities of the engine, providing a means of rendering a collection of geometric primitives as quickly as possible.

Supported geometric primitives may include triangle meshes, line lists, point lists, particles, etc.

This component typically accesses the graphics device interface (possibly through a defined platform independence layer) and acts to configure the hardware state and game shaders using some defined material system and dynamic lighting system.
It is the job of the scene graph to limit the number of geometric primitives sent to the renderer based on some form of visibility determination.

Typically some form of world spatial decomposition (bsp tree, bounding volume hierarchy, etc.) is employed to build a set of potentially visible objects.
Visual Effects

The rendering engine might provide support for the following types of visual effect:
• Particle effects
• Dynamic shadows
• Light/environmental mapping
• Full-screen post render effects (e.g. HDR, FSAA, colour correction, etc.).
A front end component provides a means of rendering 2D graphics (possibly overlaid on top of a 3D scene), providing:

- Game front end and menus
- In game Head-up-display (HUD)
- In game graphical user interface (e.g. for inventory screens, etc.)
- Debugging and other tools (e.g. console)
Collisions and Physics

Whilst collision detection and a real-time dynamics simulator can be separate entities, they are often found together given the interdependence of physics on collision detection. Given the complexity of both areas, most engines rely upon a 3rd party SDK (e.g. Havok, etc.).

A collision detection system is often of use within a wide range of non-physics related component.
Most current game engines support skeletal animation, permitting a mesh to be posed using a system of bones. Poses are typically interpolated or combined to produce a palette of matrices used to render the mesh parts (a process known as skinning).
Human Interface Devices

Games often support a number of different forms of player interface device, including:

- Keyboard and Mouse
- Gamepad
- Specialised Controllers (e.g. Wii Fit Board, dance pads, steering wheels).

It is often the role of this component to abstract the mapping of physical controls and logical game functions. The component may also embed support for detecting multiple simultaneous button presses and/or input sequences, etc.
Audio

The sophistication of audio engines between different game engines differs considerably. More advanced audio engines offer wide support for different input sound formats, including streaming, and output speaker arrangements.

Additionally, different forms of playback support, including audio effects, may also be supported.
Networking

Game engines typically provide support permitting multiple computers/consoles to be linked together in some form of mutli-player game. This is a different environment to massively multiplayer online games (MMOGs) where thousands of players are typically managed using a server farm.
Gameplay Foundations

Gameplay refers to the actions which take place within the game. Gameplay is either implemented using the same language as that used to create the underlying game engine, or through a higher-level scripting language, or some combination of both.

The gameplay foundations layer offers utilities of use to gameplay formation and a means of readily accessing some of the lower-level engine components.
Gameplay Foundations

Game World

The gameplay foundations layer may define the game world structure in terms of both static (e.g. background geometry) and dynamic objects (e.g. NPCs, rigid bodies subject to physics).

Event / Messaging System

In order to permit game objects to communicate and interact an event based messaging system is often employed.
Scripting System

A scripting system may be employed in order to provide more rapid, accessible and convenient development of gameplay rules and content.

Typically using an interpreted scripting language avoids the need to rebuild the code base given a gameplay change.

High-Level Game Flow System

High-level control mechanisms, such as finite-state machines, etc. may be embedded within the gameplay foundations layer and made available to the game specific sub-systems.
GAME ENGINES ANALYSES AND COMPARISON
Key Points

• Usability (UI, how easy it was to learn and develop with)

• Functionality (What exactly the engine can do)

• Price Point (speaks for itself)
Construct

- Less coding unless sometime u needn’t.
- Used for make a prototypes for large games and to create a simple games.
- 2d support.
- Free to use.
- Rating :7/10
- https://www.scirra.com/construct2
Game Salad

Same features of Construct2.

http://gamesalad.com/
Game Maker Studio

• Easy to learn, easy to develop.
• Need to learn new language GML (Game maker language)
• Support just 2d.
• Using for create simple 2d games.
• Price: free limited recourses and paid licenses.
• Rating 8/10
• https://www.yoyogames.com/studio
Flash Professional

• Not specialized gaming tool.
• Support just 2d.
• More coding.
• Good in animation.
• Needs a plugin to run.
• Free trial, paid.
• Rating: 6/10
• http://www.adobe.com/ie/products/flash.html
UNREAL ENGINE

• Fantastic render engine.
• Support 3d game art.
• Used for creating large 3d games.
• Need high specification computer to render.
• Price: Just paid version.
• Rating: 8/10
• https://www.unrealengine.com/what-is-unreal-engine-4
UNITY 3d

• Easy to learn.
• Easy to develop.
• Great documentation and community.
• 3d/2d Support.
• Good render engine.
• Integrated in design programs (Maya, Photoshop)
• Price: Free, trial and paid pro.

• Rated by: 9/10
<table>
<thead>
<tr>
<th>Feature</th>
<th>Unity 3d</th>
<th>Unreal Engine 4</th>
<th>GameMaker</th>
<th>Construct 2</th>
<th>Flash Pro</th>
<th>Game Salad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full 3d support</td>
<td>✔</td>
<td>✔</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>Full 2d support</td>
<td>✔</td>
<td>❌</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Free Version</td>
<td>✔</td>
<td>✔</td>
<td>❌</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Paid Version</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>PC Export</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Playstation Export</td>
<td>✔</td>
<td>✔</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>Xbox Export</td>
<td>✔</td>
<td>✔</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>Nintendo Export</td>
<td>✔</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>IOS Export</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Android Export</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>WP8 Export</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Mac Export</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>HTML5 Export</td>
<td>❌</td>
<td>❌</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

Available on Mac OS X | ✔       | ✔       | ✔       | ❌       | ✔       | ✔       | ✔       |
Available on Windows    | ✔       | ✔       | ✔       | ✔       | ✔       | ✔       | ✔       |
Made with Unity

http://unity3d.com/showcase/gallery/games