# 1. Game Engine Architecture

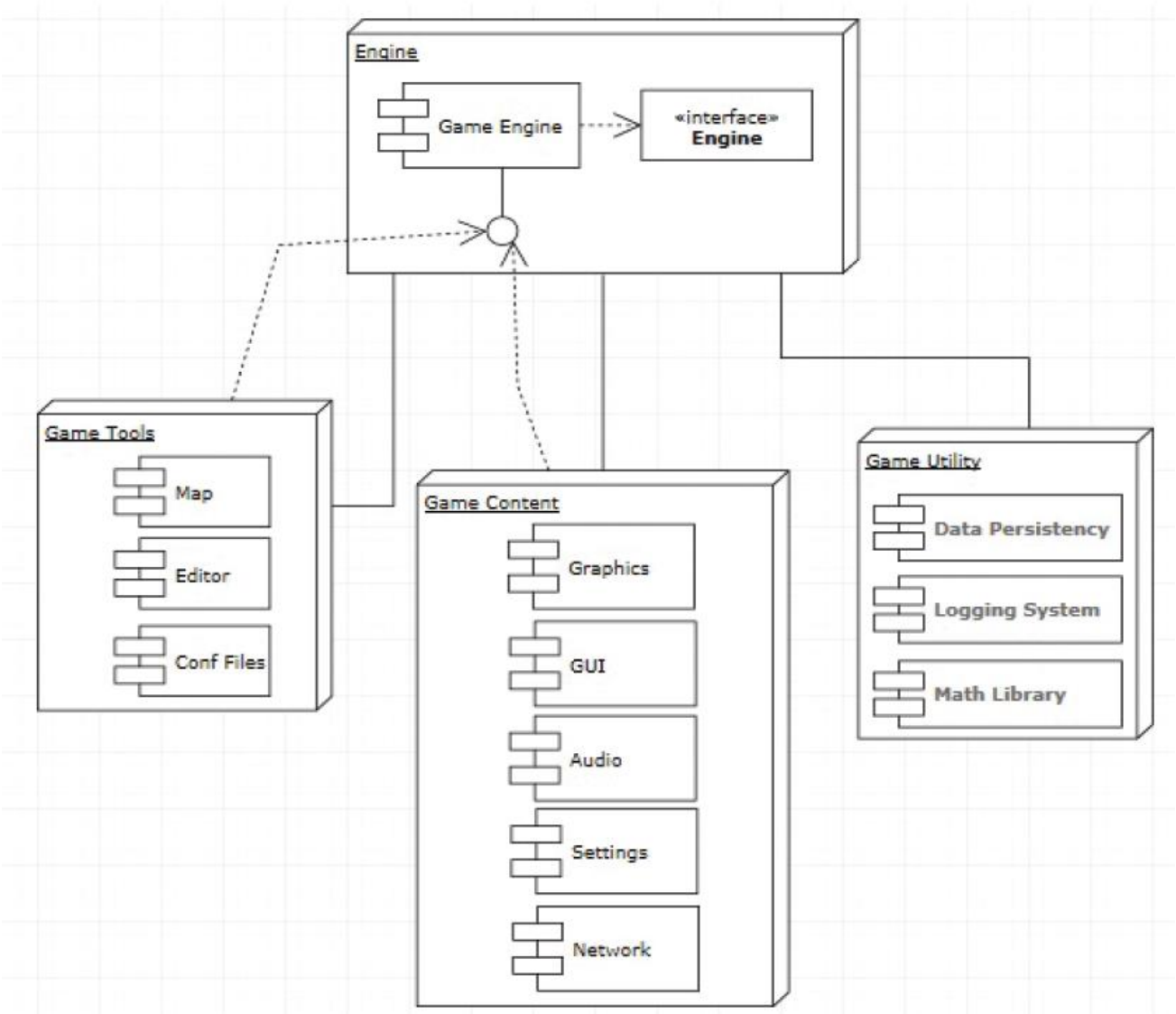
Game Player Experience Design

Prof. H. Kang

# Introduction

Game engine?

- Game engines are a new way to develop high-quality games easily and rapidly without needing intensive programming skills and computational resources.
- 3D game engines help game companies reduce their cost, time, and manpower, since game developers can use the available functionalities of the engine.



# Introduction

Game engine?

| Game engine | Language | Supported platforms | 2D | 3D | License/price |
|---|---|---|---|---|---|
| Unity 3D | C# and JavaScript | **Desktop**: Windows, Mac OS, Linux<br>**Mobile**: Android, IOS, | No | Yes | Free for Indie version |
| Jmonkey Engine | Java, Python, and JavaScript | **Desktop**: Windows, Mac OS, Linux<br>**Mobile**: Android | No | Yes | Free |
| Marmalade | C/C++ | **Desktop**: Windows, Linux,<br>**Mobile**: Android, and IOS | — | Yes | Commercial |
| LibGDX | Java | **Desktop**: Windows, Mac OS, Linux<br>**Mobile**: Android, IOS, Blackberry, HTML5 | Yes | Yes | Free |
| Panda3D | Python and C++ | **Desktop**: Windows and Mac OS | No | Yes | Free/open source |
| Blender | C/C++, Python | **Desktop**: Windows, Mac OS, Linux | No | Yes | Free |
| Unreal engine | C++, VisualScripting | **Desktop**: Windows, Mac OS, Linux, HTML5<br>**Mobile**: NA | | | Free/open source |
| CryEngine | C/ C++ | **Desktop**: Windows, and Mac OS<br>**Mobile**: Android, and IOS, | No | Yes | Unspecified |
| Crystal Space | C/C++, Python | **Desktop**: Windows, Mac OS, Linux<br>**Mobile**: NA | — | Yes | Free |
| CopperCube | JavaScript, Flash, WebGL | **Desktop**: Windows, Mac OS<br>**Mobile**: Android | Yes | Yes | Free |
| Leadwerks | C++, C#, VB.Net, Python | **Desktop**: Windows<br>**Mobile**: NA | | | $99.95 |

# Introduction

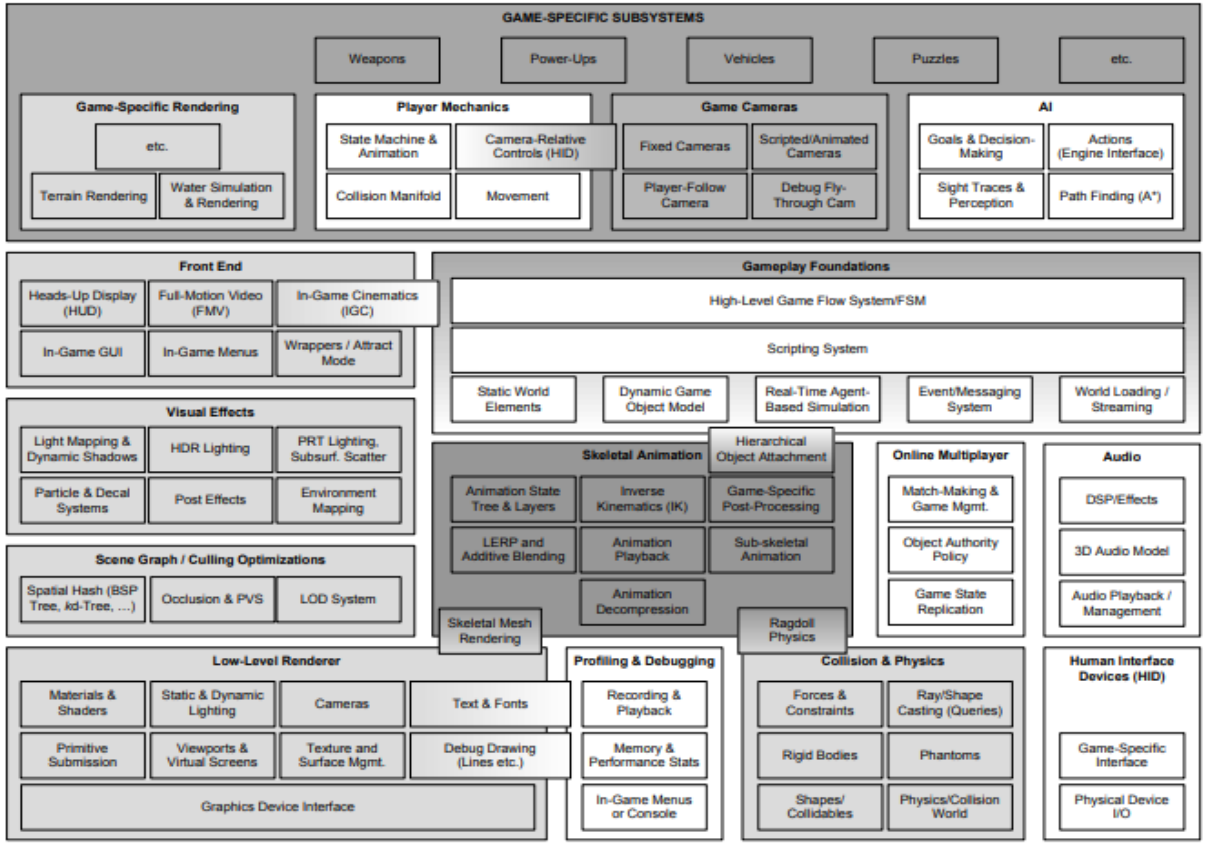
Game engine?

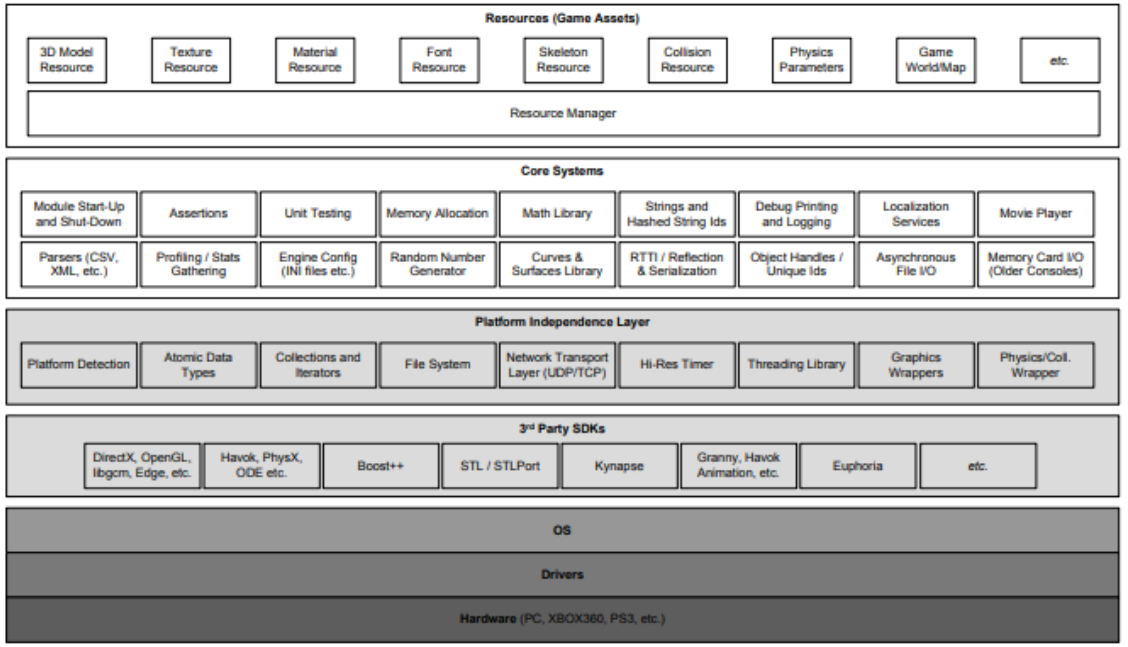
| Game engine | Language | Supported platforms | 2D | 3D | License/price |
|---|---|---|---|---|---|
| Raydium | C/C++ | **Desktop:** Windows, MacOS, Linux<br>**Mobile:** IOS | No | Yes | Free |
| SunBurn | C# | **Desktop:** Windows<br>**Mobile:** Windows Phone | No | Yes | $ 150 |
| UDK | C/C++ | **Desktop:** Windows<br>**Mobile:** iOS | No | Yes | $99.00 |
| Corona SDK 61 | Lua | **Desktop:** Windows, Mac OS<br>**Mobile:** iOS; Android; Windows Phone | Yes | No | Free |
| 3D Game Studio | C++, C# | **Desktop:** Windows<br>**Mobile:** NA | Yes | Yes | Free for the basic version |
| Unigine | C/C++ | **Desktop:** Windows, and Linux<br>**Mobile:** iOS; Android | No | Yes | $ 25,000 |
| Torque3D | C++, TorqueScript | **Desktop:** Windows, Mac OS, and Linux<br>**Mobile:** NA | — | Yes | Free/open source |
| ShiVA | C/C++, Lua | **Desktop:** Windows, Mac OS, and Linux<br>**Mobile:** iOS; Android; Windows Phone, and Blackberry | — | Yes | Free for personal use |
| Irrlicht | C++, C#, VB.Net | **Desktop:** Windows, Mac OS, and Linux | Yes | Yes | Free |

# Runtime Engine Architecture

A game engine generally consists of a tool suite and a runtime component.

- Following figure shows all of the major runtime components that make up a typical 3D game engine.
- Like all software systems, game engines are built in ‘layers’.
- Normally upper layers depend on lower layers, but not vice versa.

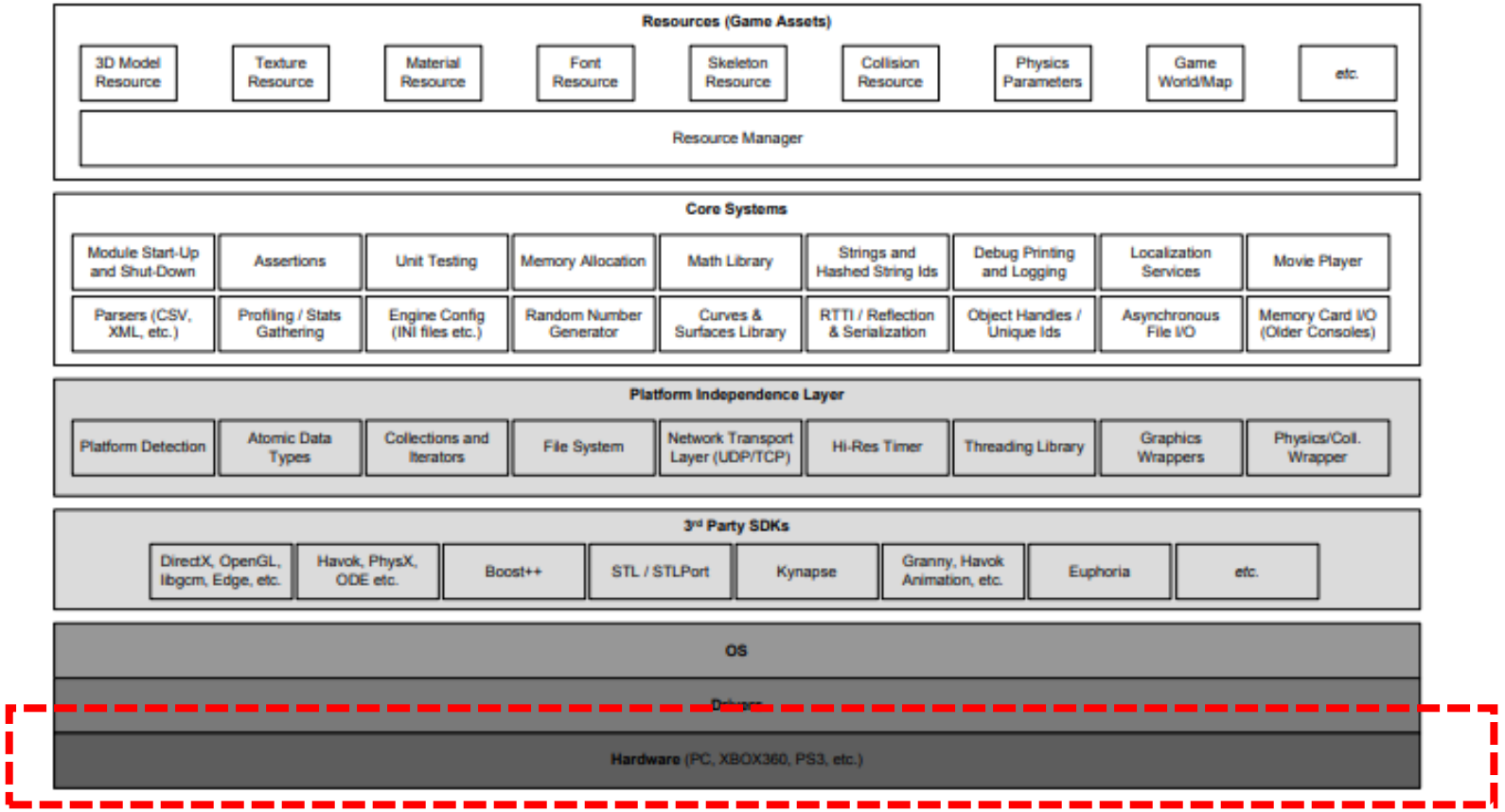


# Runtime Engine Architecture - Hardware

Target hardware

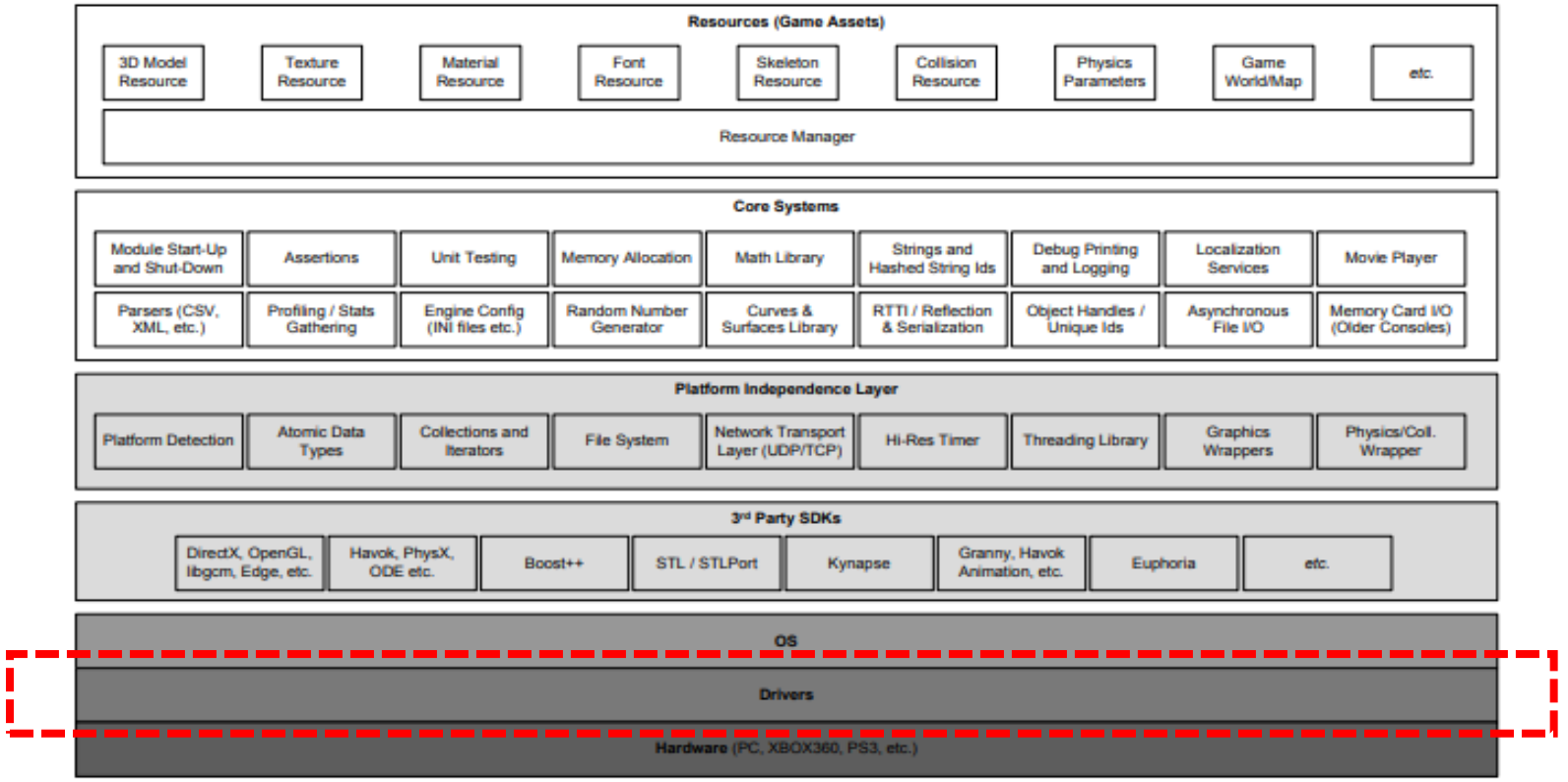
- The target hardware layer represents the computer system or console on which the game will run.
- Typical platforms include Microsoft Windows, Linux, and MacOS-based PCs.
- Mobile platforms include Apple iPhone, iPad, Android smartphones and tablets.



# Runtime Engine Architecture - Drivers

Device drivers

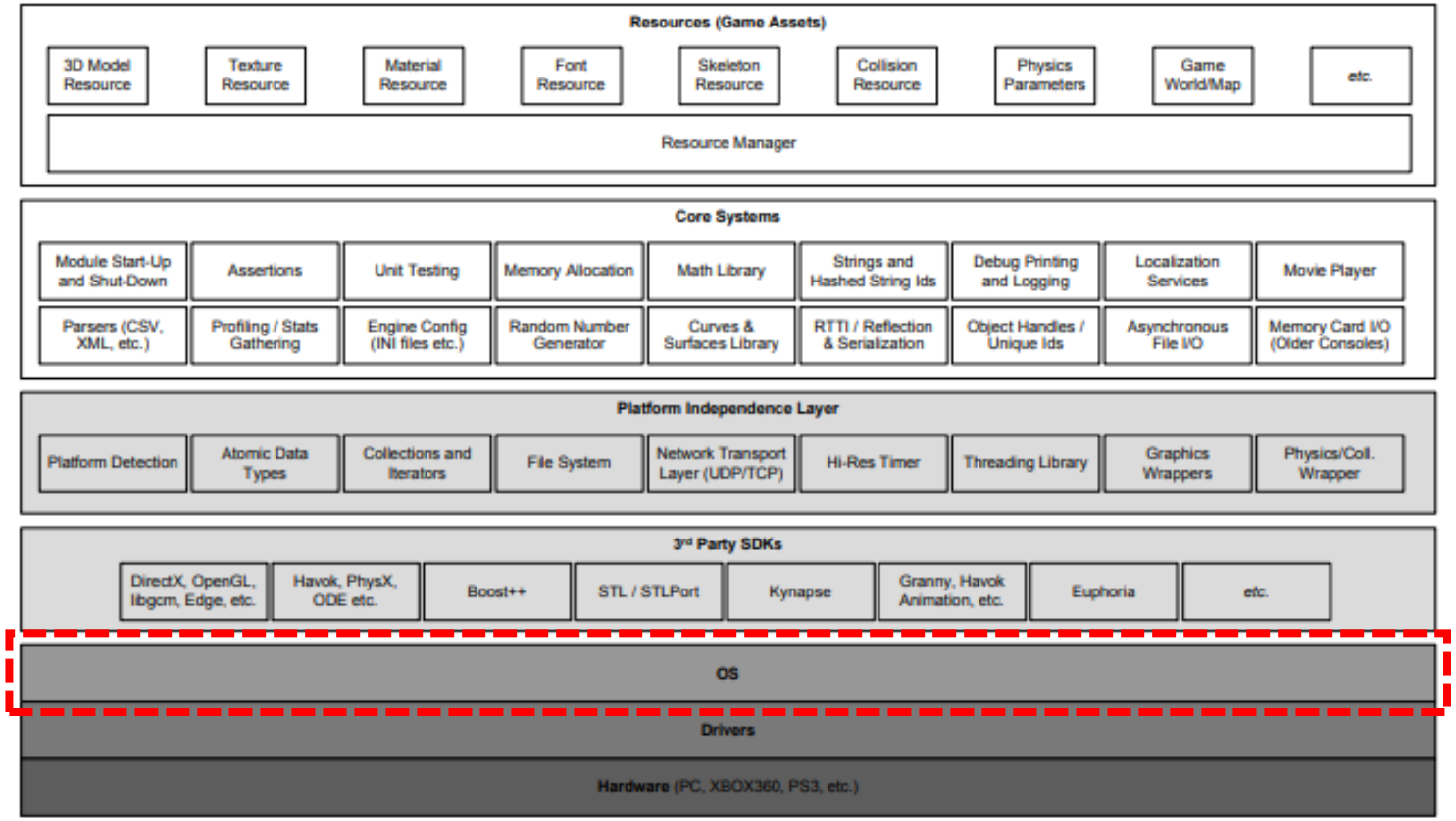
- Device drivers are low-level software components provided by the operating system or hardware vendor.
- Drivers manage hardware resources and shield the operating system and upper engine layers from the details of communicating with the myriad variants of hardware devices available.



# Runtime Engine Architecture - OS

Operating system

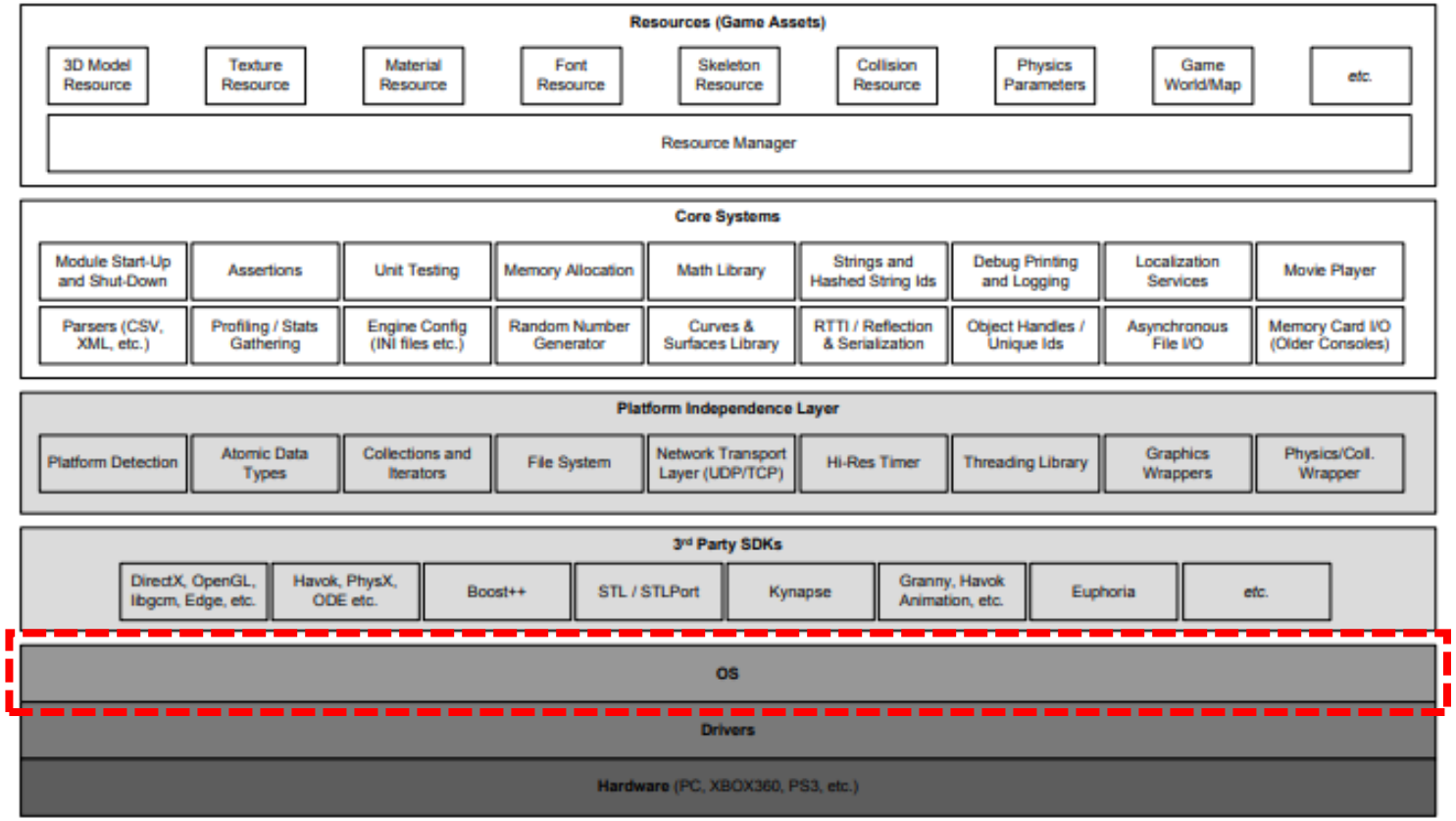
- The operating system (OS) is running all the time.
- It orchestrates the execution of multiple programs on a single computer, one of which is your game.
- Operating system like Microsoft Windows employ a time-sliced approach to sharing the hardware with multiple running programs.
- This mean that a PC game can never assume it has full control of the hardware - it must play nice with other programs in the system.



# Runtime Engine Architecture - OS

Operating system

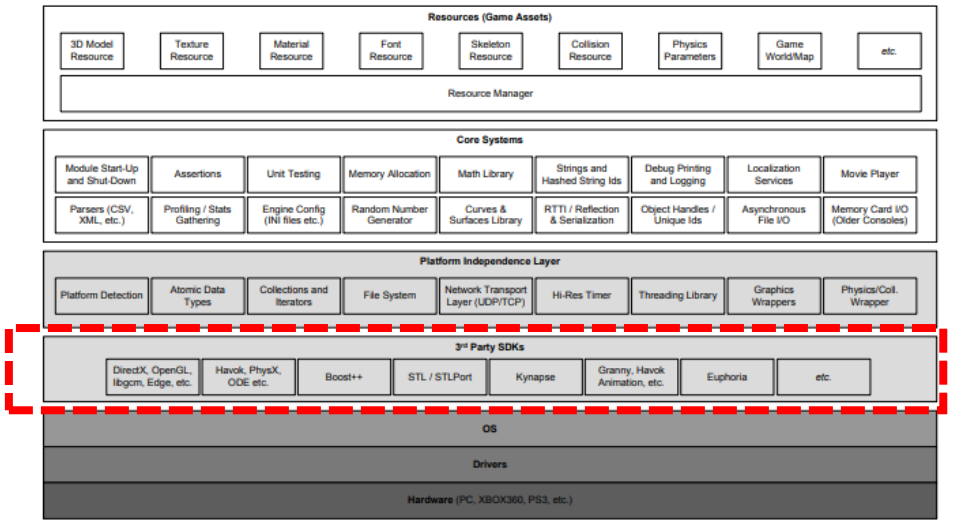
- On a console, the operating systems is often just a thin library layer that is compiled directly into your game executable. The game typically ‘owns’ the entire machine.
- However, the Xbox and Playstation series can interrupt the execution of your game, or take over certain system resources.

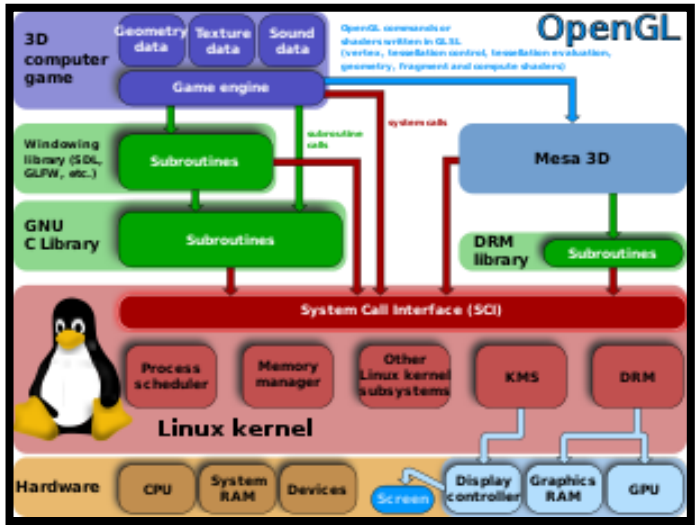


# Runtime Engine Architecture - 3rd party SDKs

Third-Party SDKs and Middleware

- Most game engines leverage a number of third-part software development kits (SDKs) and middleware.
- *Graphics*
  - OpenGL: this is a cross-language API for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering.
  - libgcm: this is a low-level direct interface to the playStation 3's RSX graphics hardware.




# Runtime Engine Architecture - 3rd party SDKs

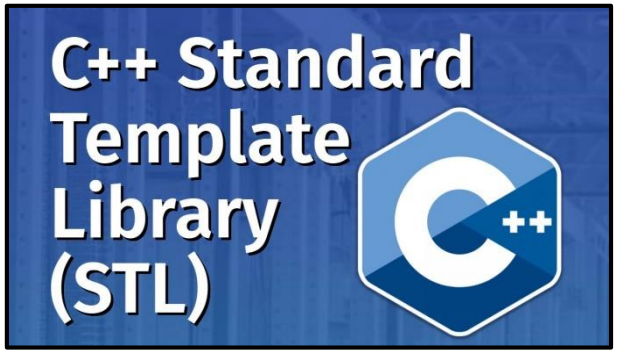
Third-Party SDKs and Middleware

- *Graphics*
  - DirectX: this is Microsoft's 3D graphics SDK and primary rival to OpenGL.
  - Vulkan: A low-overhead, cross-platform API. This aims to provide a considerably lower-level API for the application than the older APIs (OpenGL and Direct3D 11).


# Runtime Engine Architecture - 3rd party SDKs

Third-Party SDKs and Middleware

- *Data structures and algorithms*
  - STL: this is a the c++ standard template library provides a wealth of code and algorithms for managing data structures, strings, and stream-based I/O.
  - Boost++: this is a set of libraries for the C++ programming language that provides support for tasks and structures such as linear algebra, pseduorandom number generation, multithreading, image processing, etc.
  - Loki: Loki is a powerful generic programming template library.

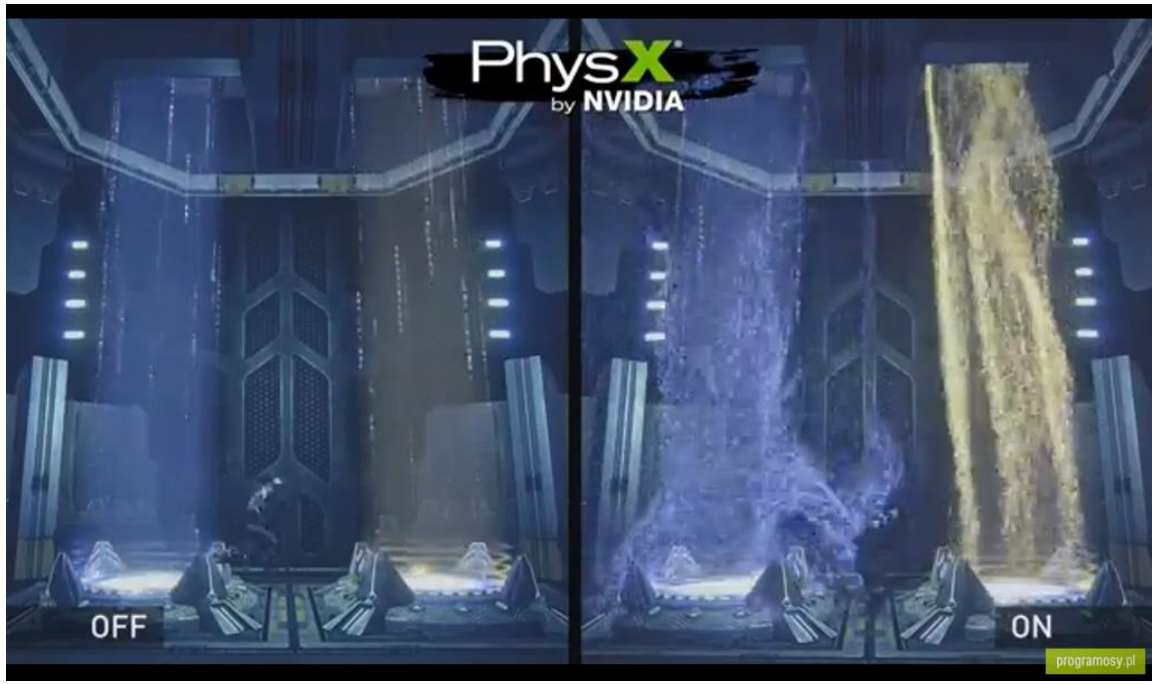



# Runtime Engine Architecture - 3rd party SDKs

Third-Party SDKs and Middleware

- *Collision and physics*
  - Havok: this is a popular industrial-strength physics and collision engine.
  - PhysX: this is another popular industrial-strength physics and collision engine provided by NVIDIA.


Havok engine[1]




PhysX[2]


IGN


NVIDIA
PHYSX

# Runtime Engine Architecture - 3rd party SDKs

Third-Party SDKs and Middleware

- *Character animation*
  - Granny: This includes robust 3D model and animation exporters.
  - Spine: Spine is 2D skeletal animation software for game.
  - Havok animation: Havok company creates a complimentary animation SDK, which makes bridging the physics animation gap much easier than it ever has been.


Spine engine example[spine]


Havok animation[3]

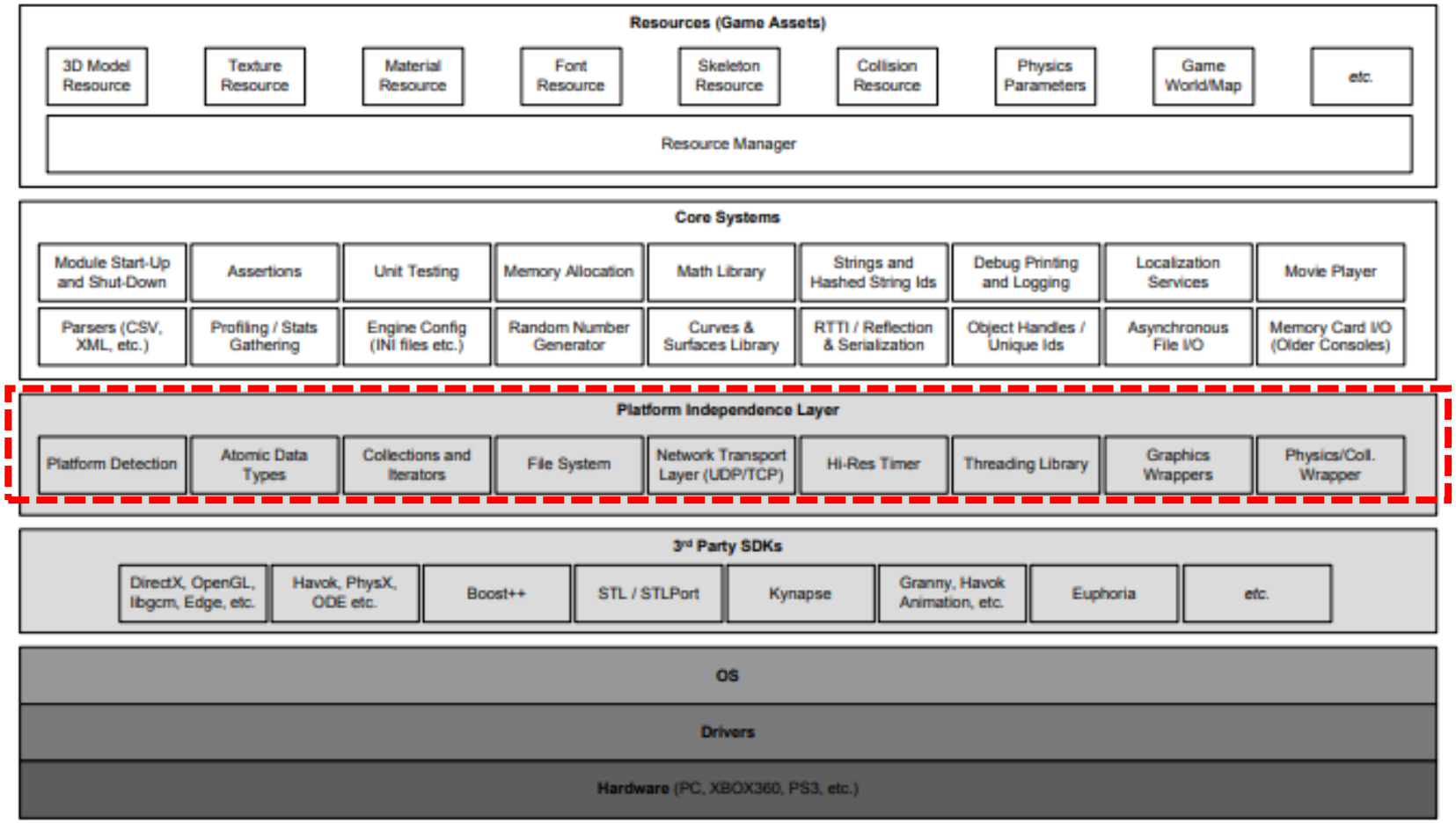
Debug display

Loading...

# Runtime Engine Architecture - platform independence layer

Most game engines are required to be capable of running on more than one hardware platform.

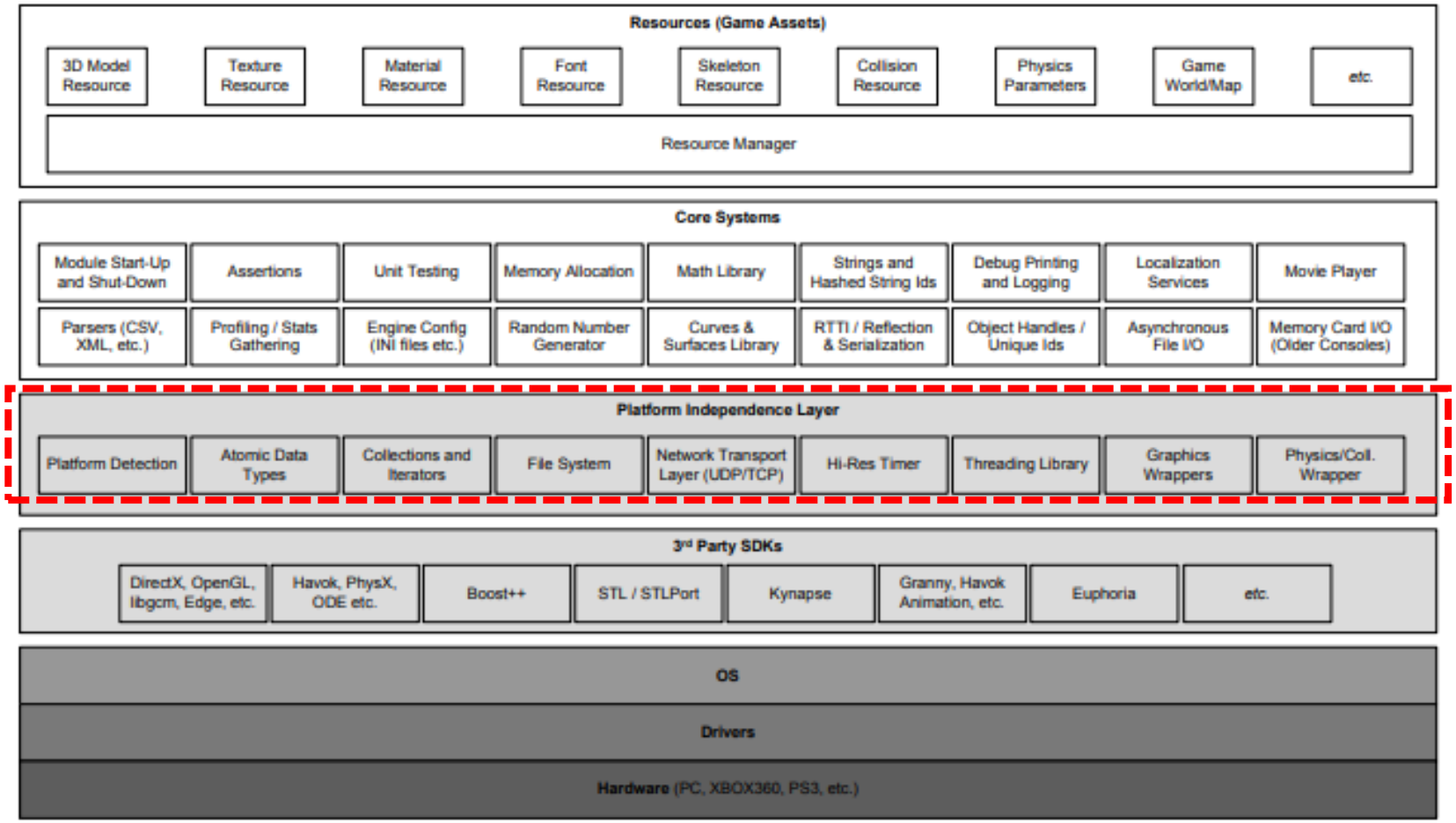
- Typically, most game studios target their games at a wide variety of platforms because it exposes their games to the largest possible market.
- Therefore, most game engines are architected with a platform independence layer.



# Runtime Engine Architecture - platform independence layer

Most game engines are required to be capable of running on more than one hardware platform.

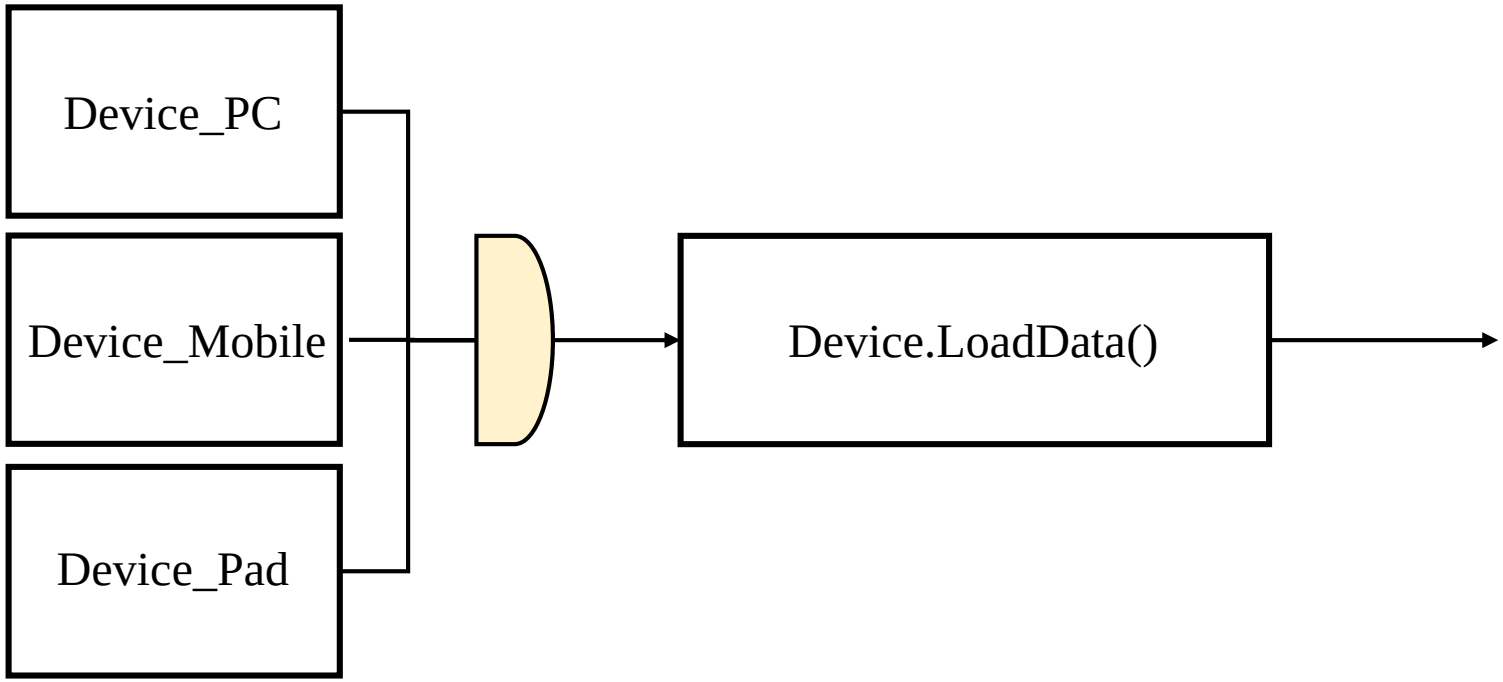
- This layer shields the rest of the engine from the majority of knowledge of the underlying platform.
- This layer ensures consistent behavior across all hardware platforms.



# Runtime Engine Architecture - platform independence layer

Delegation

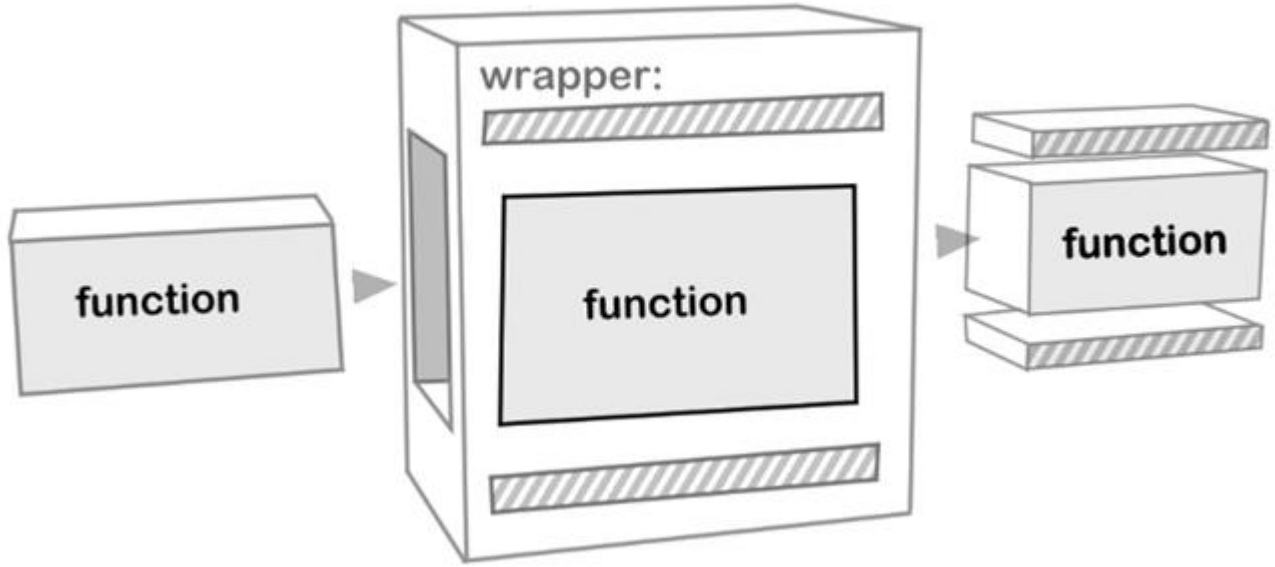
- In object-oriented programming, delegation refers to evaluating a member of one object (the receiver) in the context of another original object (the sender).
- Delegation can be done explicitly, by passing the sending object to the receiving object.
- Implicit delegation can be done by the member lookup rules of the language, which requires language support for the feature (see kotlin).



# Runtime Engine Architecture - platform independence layer

Wrapper function

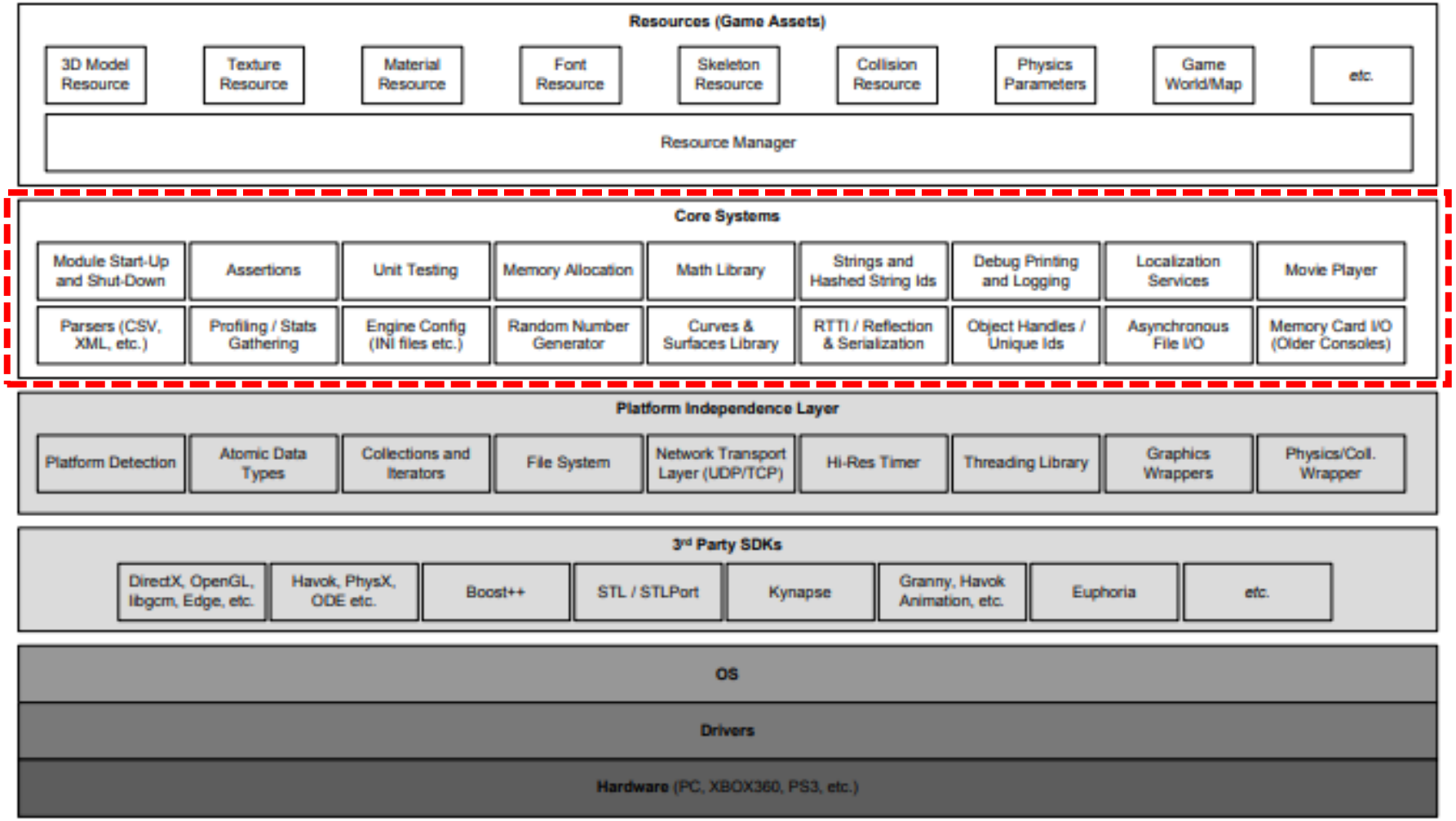
- Wrapper functions are a means of delegation.
- Many library functions act as interfaces for abstraction of system calls.
- By wrapping the most commonly used standard C library functions, operating system calls, and other foundational application programming interfaces (APIs), the platform independence layer ensures consistent behavior across all hardware platforms.



# Runtime Engine Architecture - core systems

Core systems

- Complex game engine requires a grab bag of useful software utilities.
- These are categorized under the label ‘core systems.’



# Runtime Engine Architecture - core systems

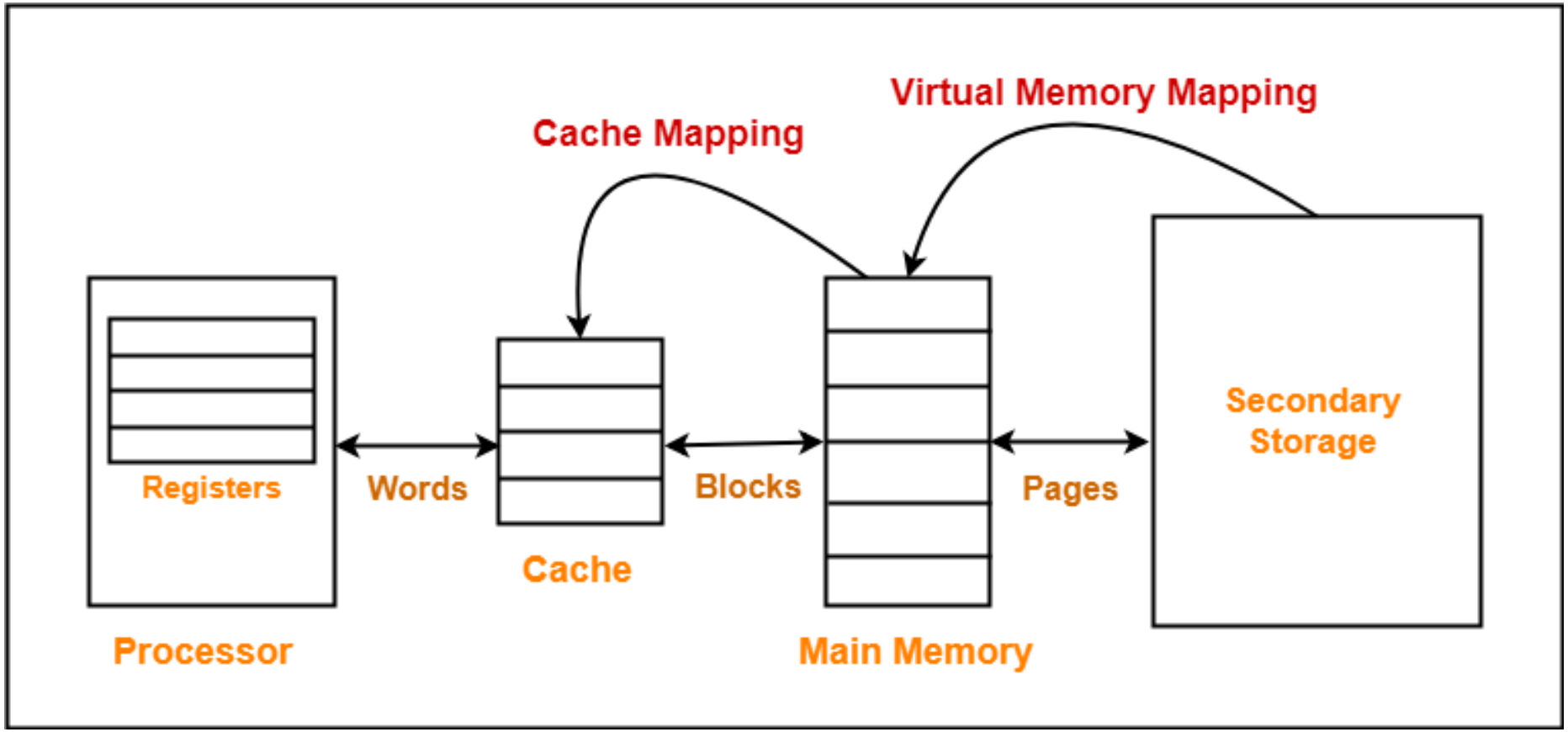
- *Assertions*
  - Assertions are lines of error-checking code that are inserted to catch logical mistakes and violations of the programmer's original assumptions.

```c
/* assert example */
#include <stdio.h>      /* printf */
#include <assert.h>     /* assert */

void print_number(int* myInt) {
  assert (myInt!=NULL);
  printf ("%d\n",*myInt);
}

int main ()
{
  int a=10;
  int * b = NULL;
  int * c = NULL;

  b=&a;

  print_number (b);
  print_number (c);

  return 0;
}
```

C++ assert example

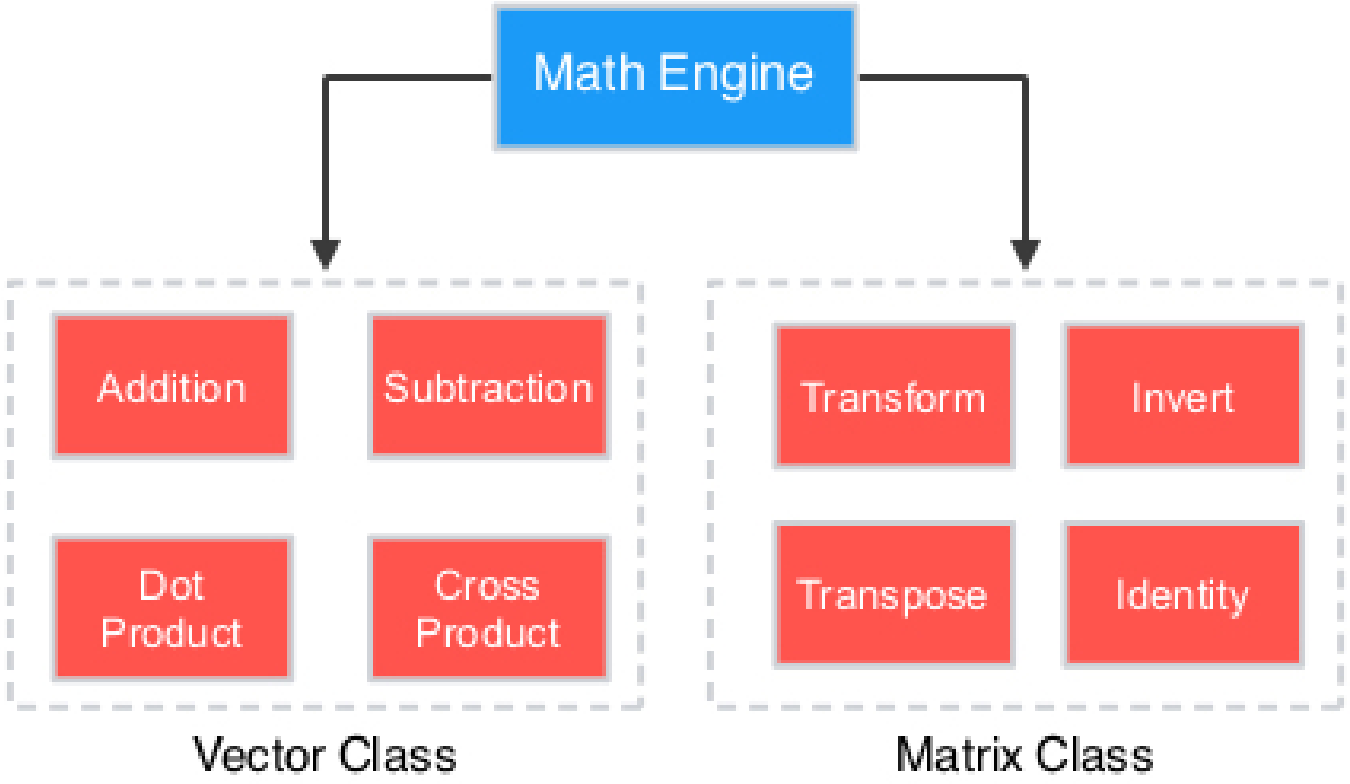
# Runtime Engine Architecture - core systems

- *Memory management*
  - Virtually every game engine implements its own custom memory allocation system to ensure high-speed allocations and deallocations and to limit the negative effects of memory fragmentation.



# Runtime Engine Architecture - core systems

- *Math library*
  - Games are highly mathematics-intensive.
  - As such, every game engine has at least one math libraries.
  - These libraries provide facilities for vector and matrix math, quaternion rotations, trigonometry, geometric operations with lines, rays, spheres, frusta, etc.



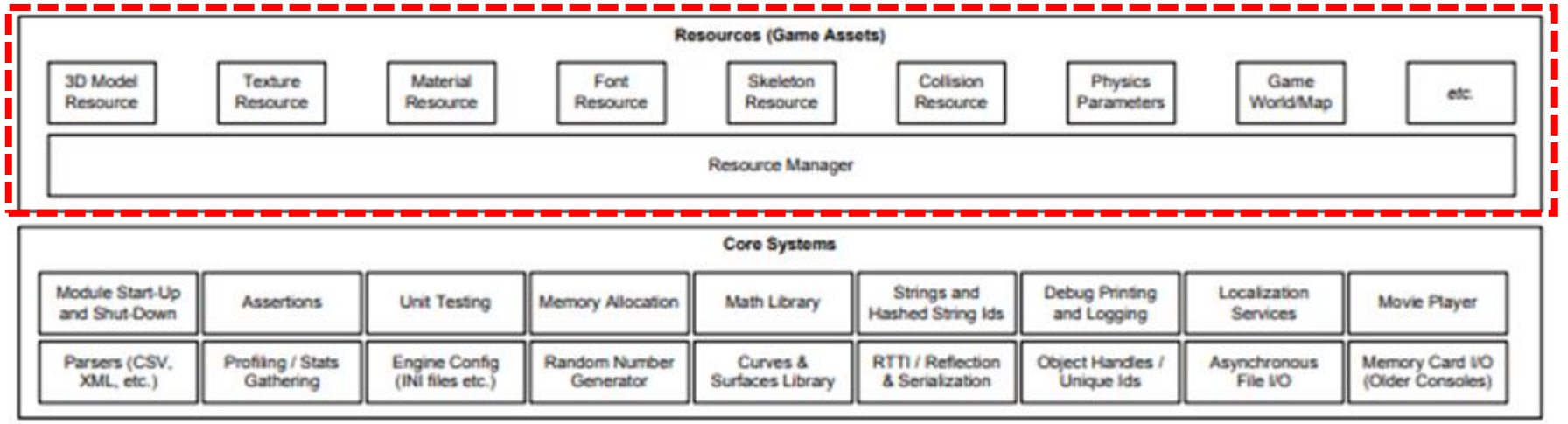
# Runtime Engine Architecture - core systems

- *Custom data structures and algorithms*
  - Game engine often requires a suite of tools for managing fundamental data structures (linked lists, dynamic arrays, binary trees, hash maps, etc.) and algorithms (search, sort, etc.) is usually required.
  - These are often hand coded to minimize or eliminate dynamic memory allocation and to ensure optimal runtime performance on the target platform.

# Runtime Engine Architecture - resource manager

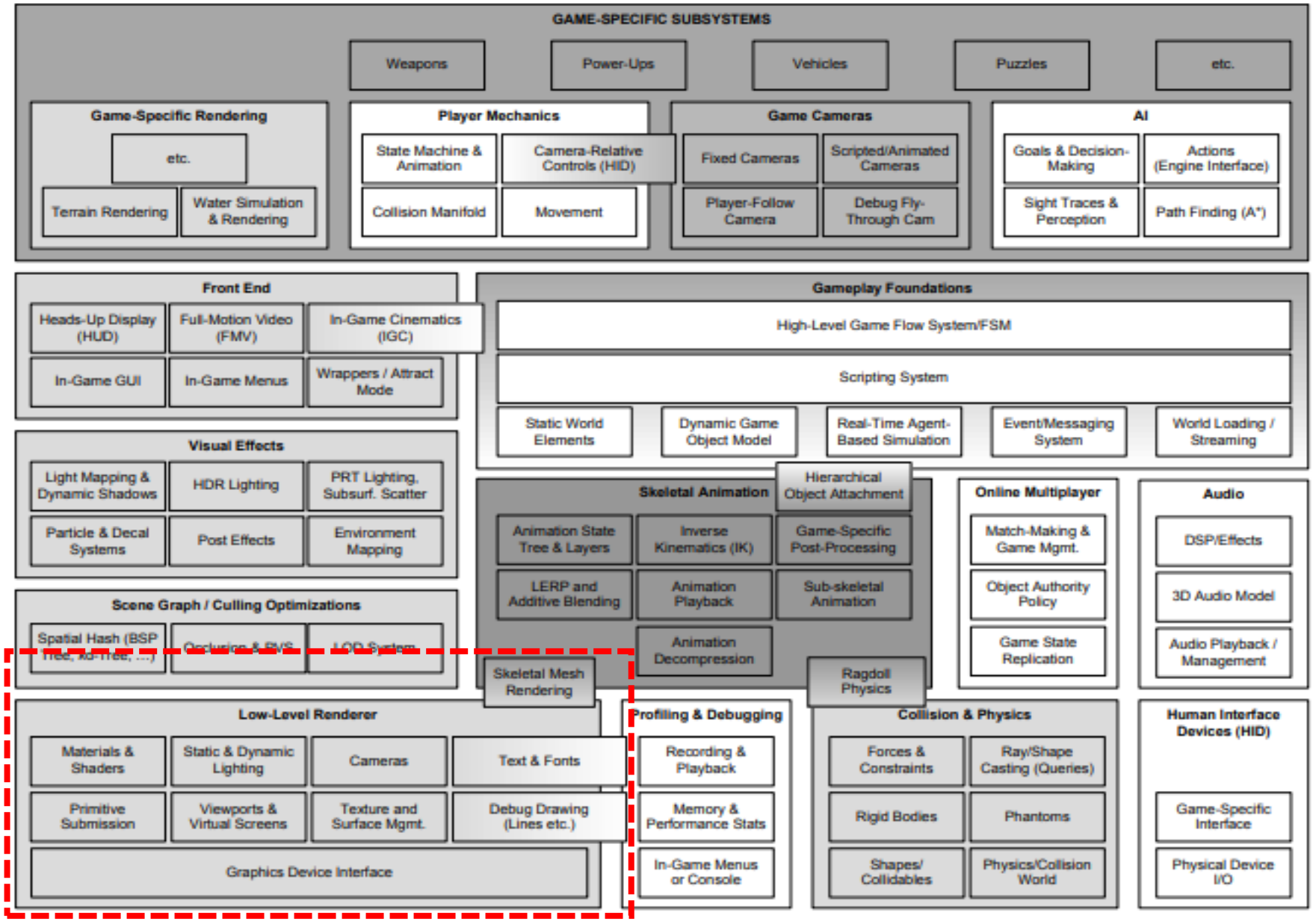
Resource manager

- The resource manager provides a unified interface (or suite of interfaces) for accessing any and all types of game assets and other engine input data.
- Some engines (Unreal's packages, OGRE's Resource-Manager class) do this in a highly centralized and consistent manner.
- Other engines take an ad hoc approach, often leaving it up to the game programmer to directly access raw files on disk or within compressed archives.



# Runtime Engine Architecture - resource manager

# Runtime Engine Architecture - rendering engine



# Runtime Engine Architecture - rendering engine

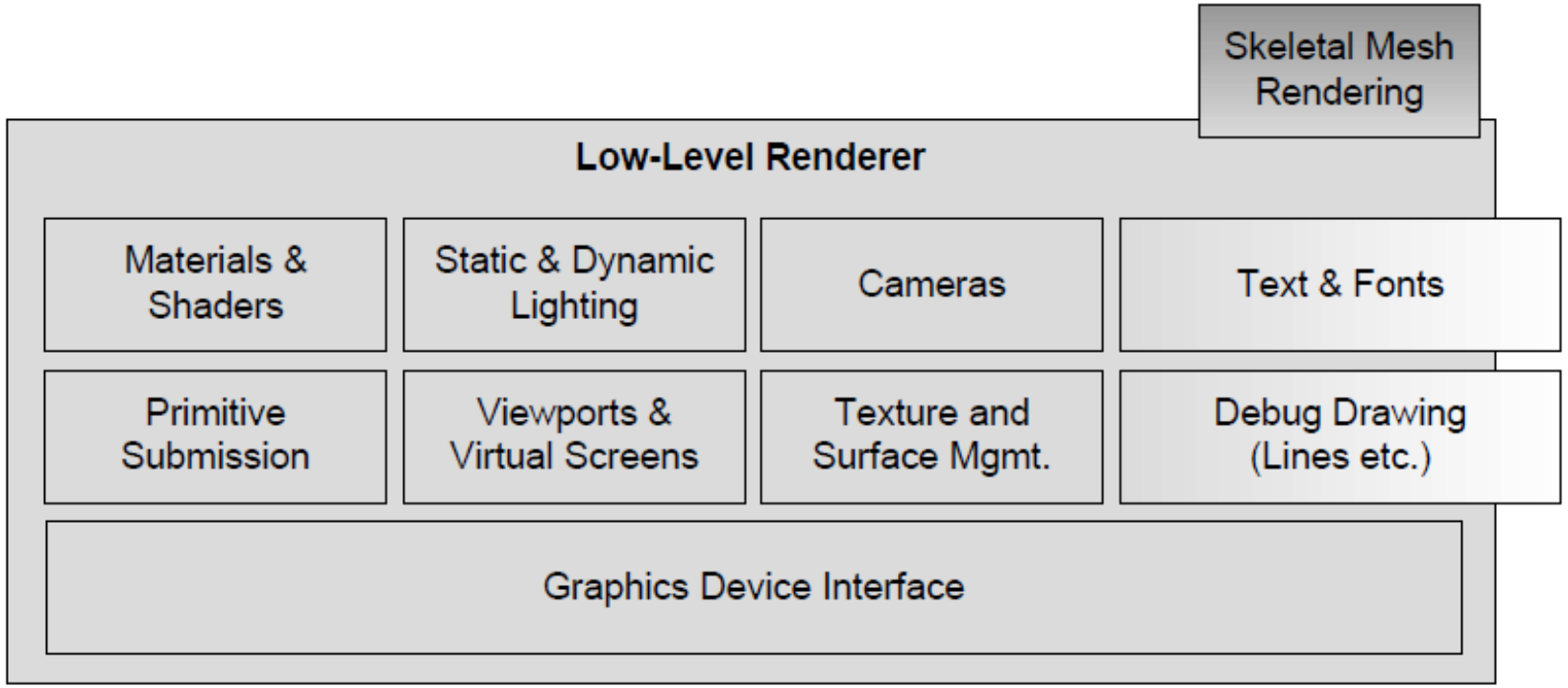
Drawing things on the screen

- The rendering engine is one of the largest and most complex components of any game engine.
- This draws text, images, and meshes on the screen.

# Runtime Engine Architecture - rendering engine

Low-level renderer

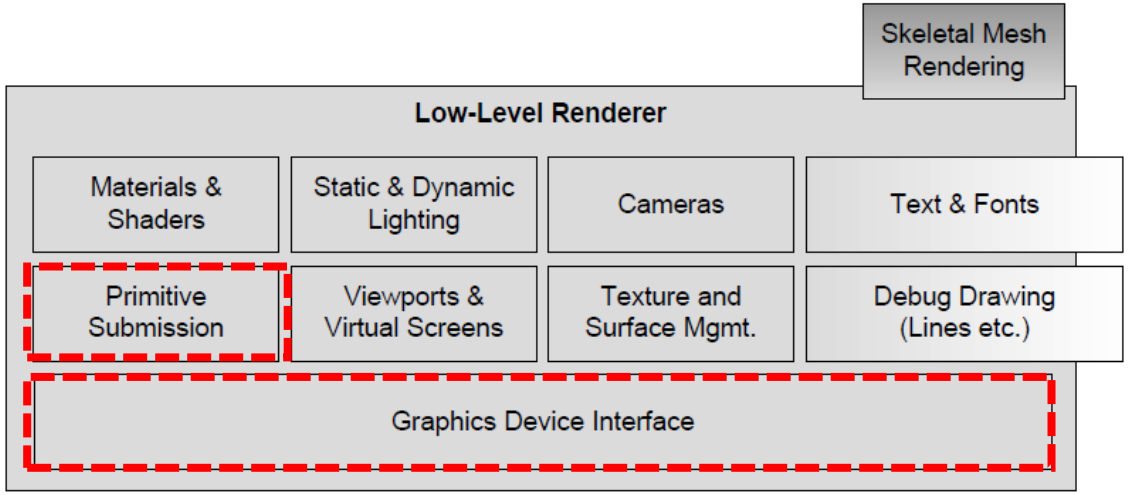
- The low-level renderer encompasses all of the raw rendering facilities of the engine.
- The design is focused on rendering a collection of geometric primitives as quickly and richly as possible, without much regard for which portions of a scene may be visible.



# Runtime Engine Architecture - rendering engine

Low-level renderer

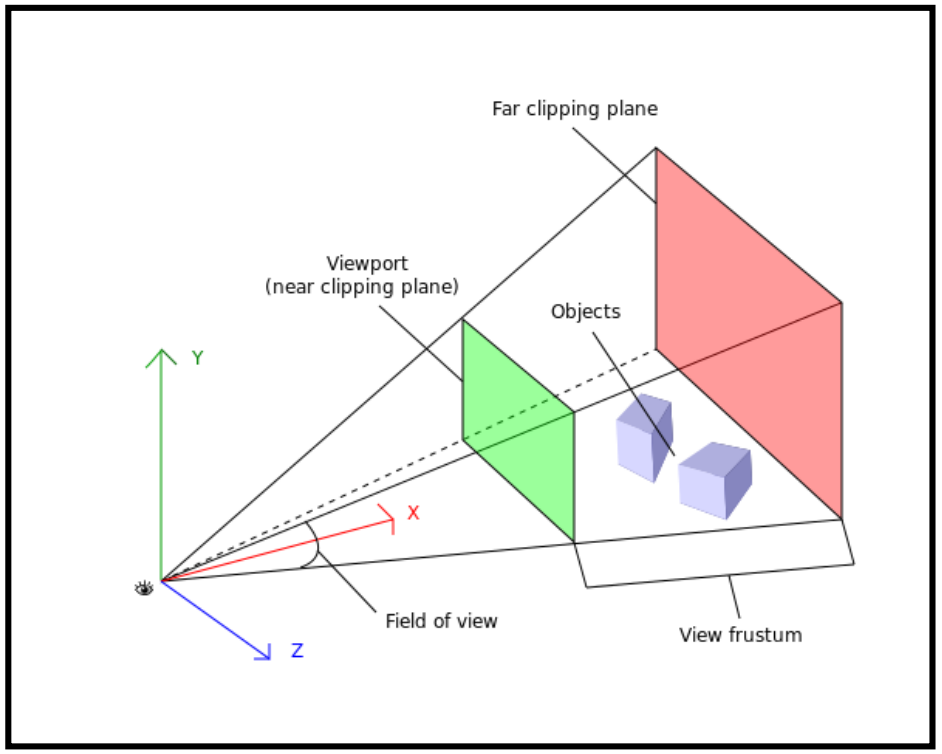
- *Graphics Device Interface*
  - This handles graphics SDKs, such as DirectX and OpenGL, which require a reasonable amount of code to be written just to enumerate the available graphics devices, initialize them, set up render surfaces and so on.
  - For a PC game engine, you also need code to integrate your renderer with the Windows message loop or keyboard polling loop.
- *Geometric primitives submission*
  - This sometimes called render packets, such as meshes, line lists, point lists, particles, terrain patches, text strings, etc.

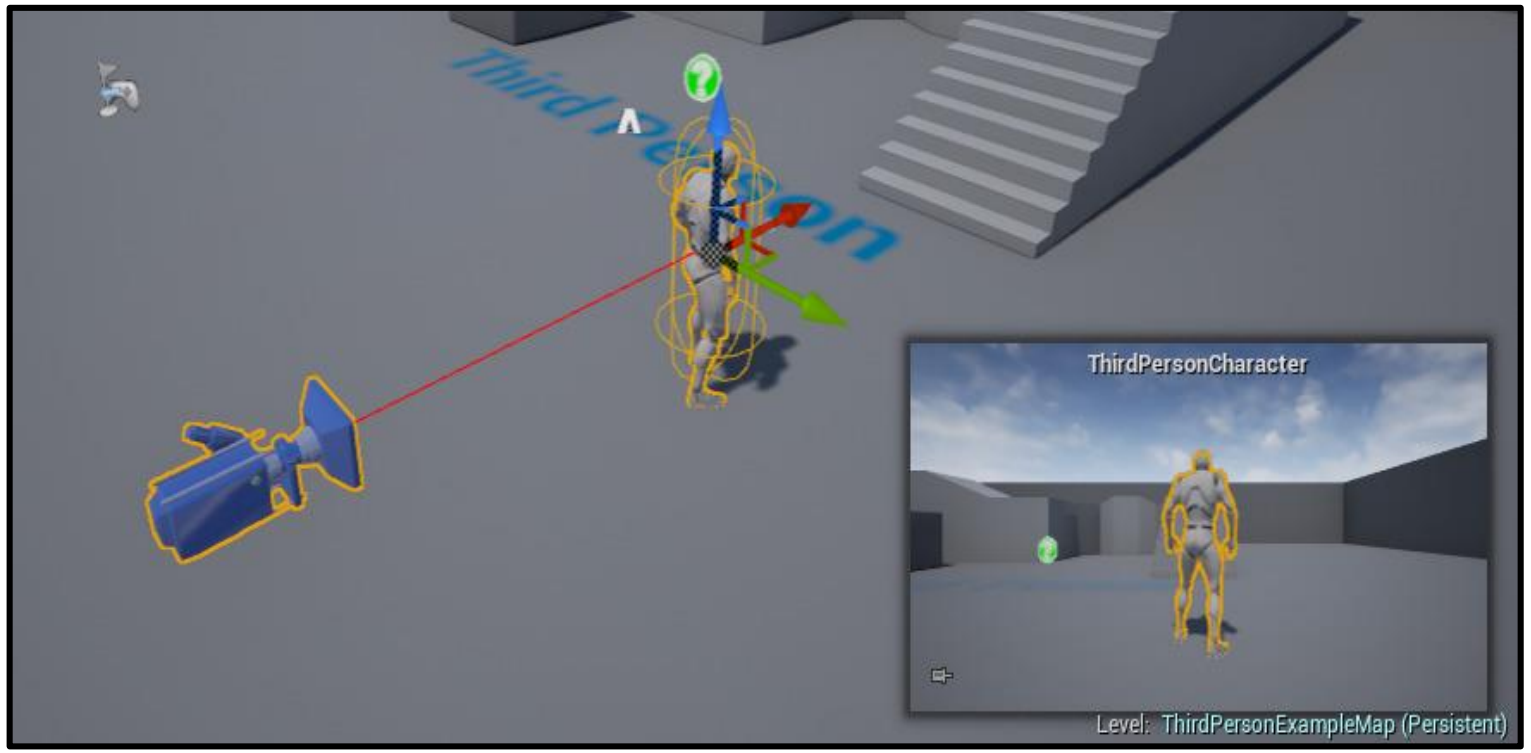


# Runtime Engine Architecture - rendering engine

Low-level renderer

- *Viewports & virtual screens*
  - Renderer usually provides a viewport abstraction with an associated camera-to-world matrix and 3D projection parameters, such as field of view and the location of the near and far clip planes.

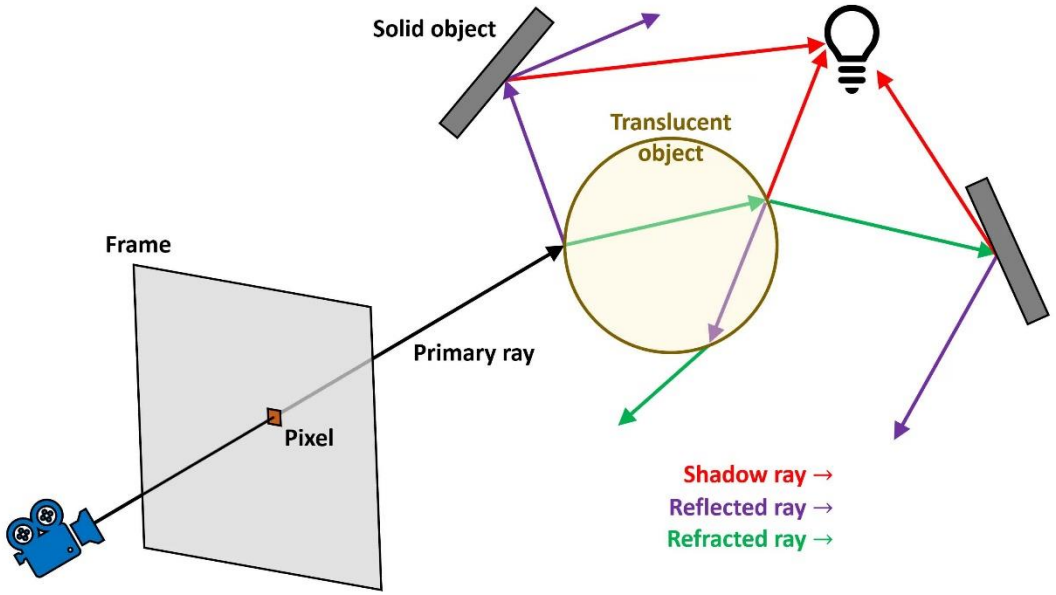


# Runtime Engine Architecture - rendering engine

Low-level renderer

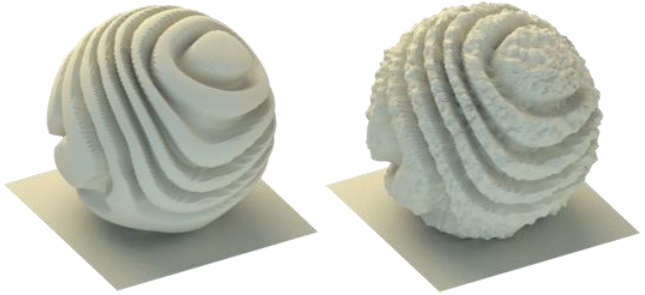
- *Static & dynamic lighting*
  - Each submitted primitive is associated with a material and is affected by *n* lights.

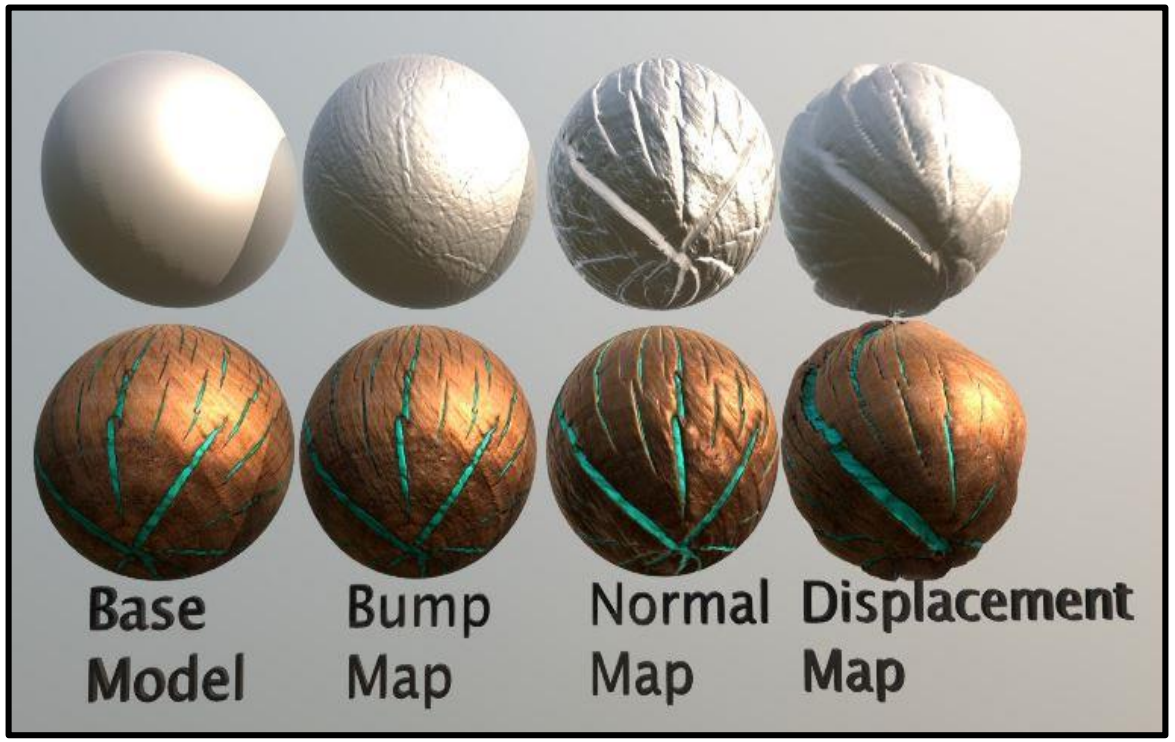


# Runtime Engine Architecture - rendering engine

Low-level renderer

- *Texture & surface management*
  - A material defines the optical properties of an object (color, dull, shinty, etc.)
  - A texture is a pattern or image to paint the object.



# Reference


[1] https://www.youtube.com/watch?v=QnKNliV7gwU&ab_channel=IGN
[2] https://www.youtube.com/watch?v=K1rotbzekf0&ab_channel=NvidiaGameWorks
[3] https://www.youtube.com/watch?v=bmXepYhTwU0&ab_channel=HavokAnimation
[4] https://www.youtube.com/watch?v=QK3oi3Ikbxk&ab_channel=MiguelCepero
[5] https://youtu.be/Z4JfYquO6w8?list=PLwMiBtF6WzsqC7_cJmD26ts0YDbtPCCfe
[6] https://www.youtube.com/watch?v=0JFYt8kGYhM
[7] https://www.guru99.com/difference-compiler-vs-interpreter.html
[8] https://www.youtube.com/watch?v=U5MTIh_KyBc&ab_channel=AIandGames
[bill] https://www.youtube.com/watch?v=hx8m9w4YyNo&ab_channel=TooEazyCG
[rag] https://www.youtube.com/watch?v=ZEL9vS7CeeI&ab_channel=GRANDOS
[path] https://www.youtube.com/watch?v=-bdFEaNeZMM&ab_channel=Ye%27Gaung
[spine] https://note.com/appai/n/nd0c523299c1a