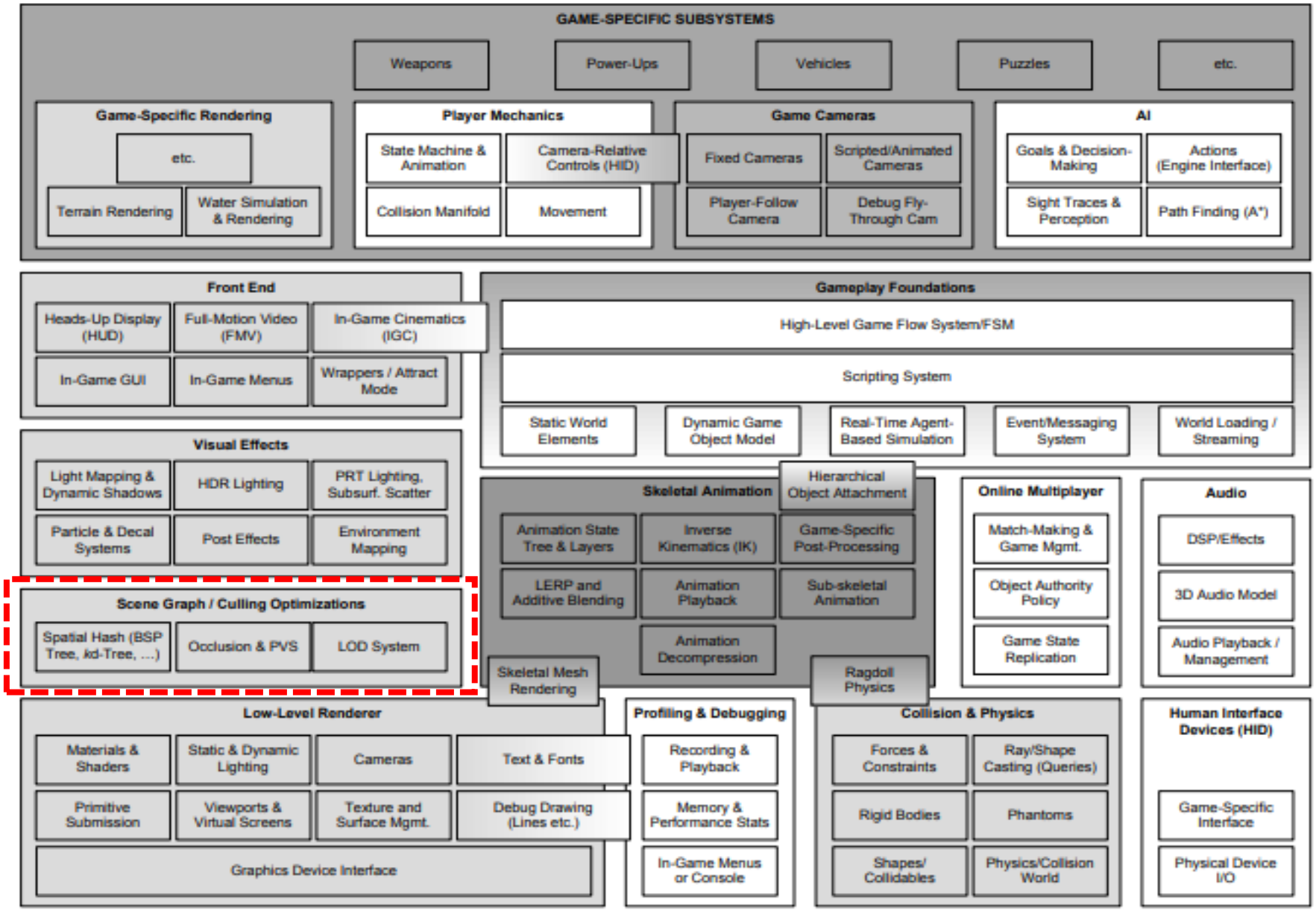
2. Game Engine Architecture 2

Game Player Experience Design

Prof. H. Kang

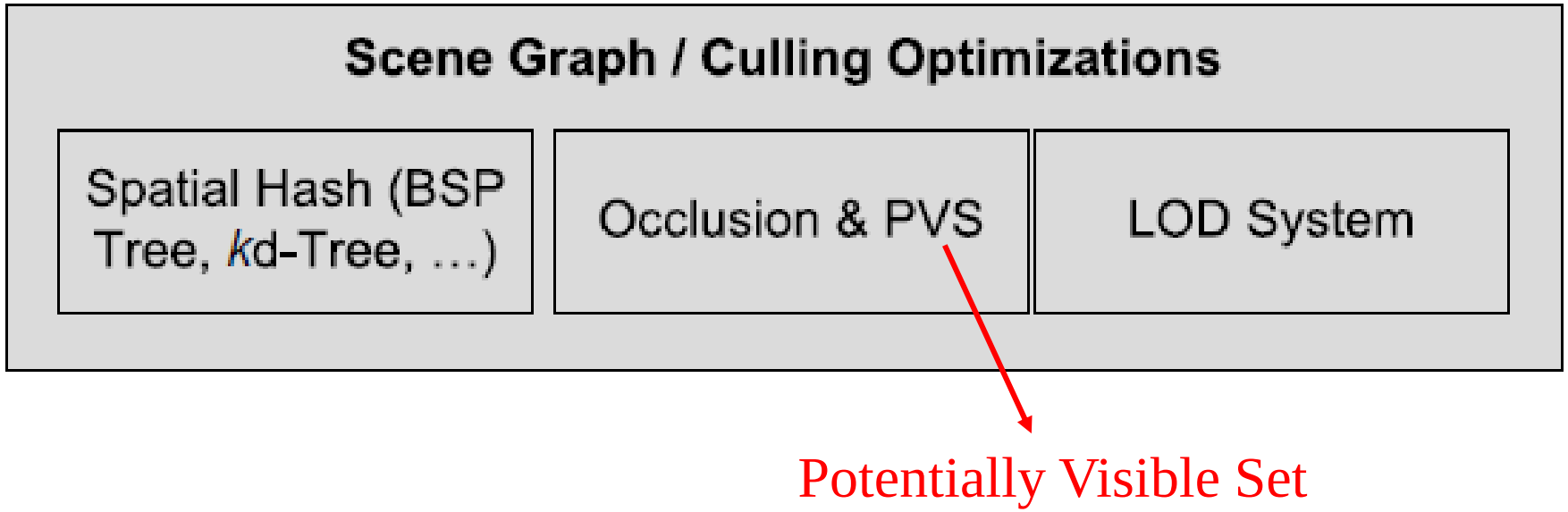
Runtime Engine Architecture - scene optimizations



Runtime Engine Architecture - scene optimizations

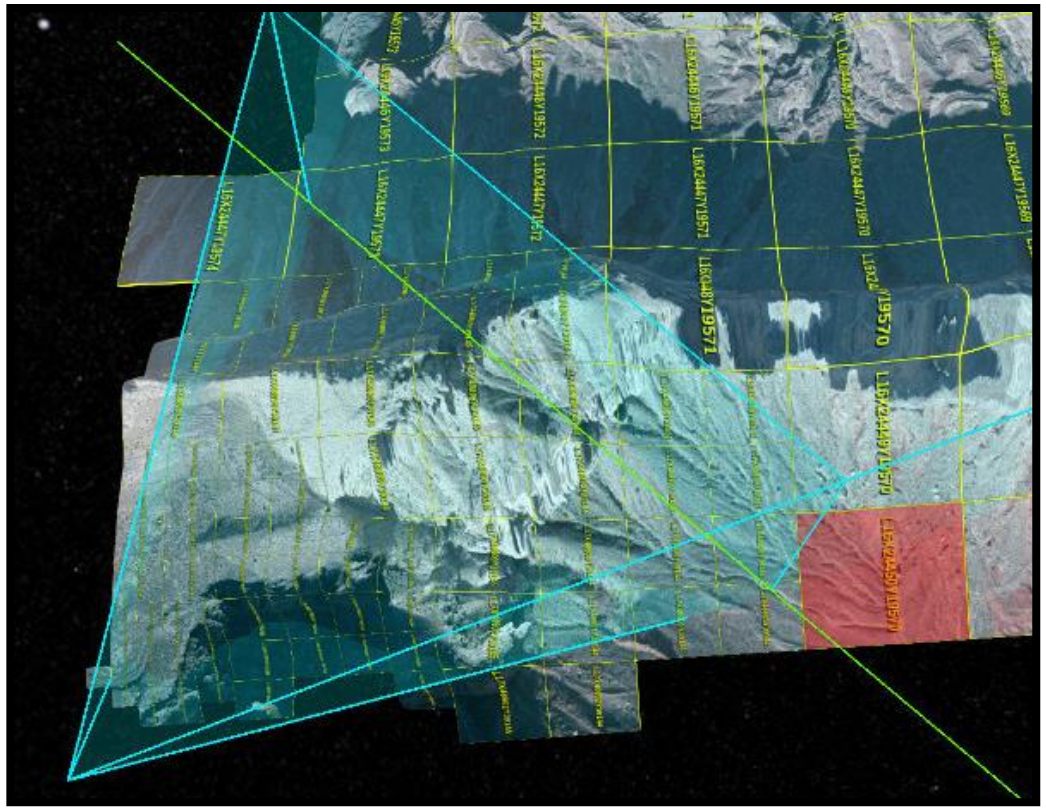
Scene graph/culling optimizations

- The low-level renderer draws all of the geometry submitted to it, without much regard for whether or not that geometry is actually visible.
- A higher-level component is usually needed in order to limit the number of primitives submitted for rendering, based on some form of visibility determination.

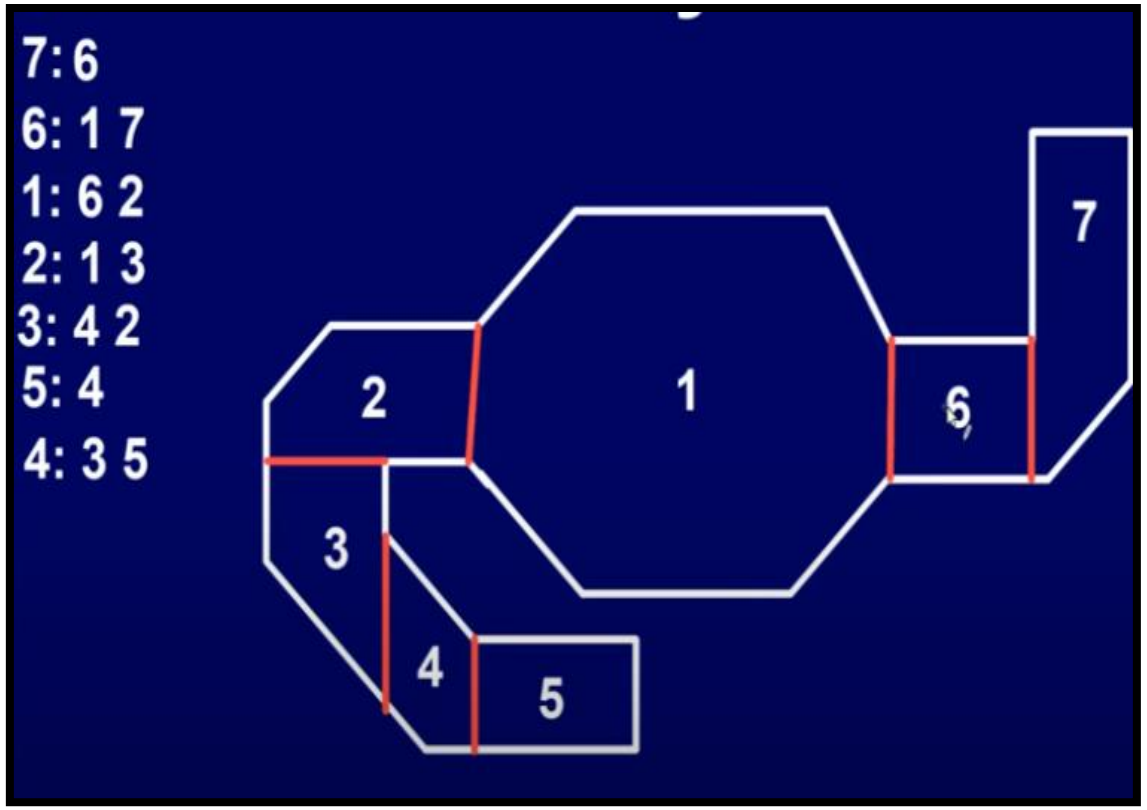


Runtime Engine Architecture - scene optimizations

- *Frustum cull & spatial subdivision*
  - Frustum culling removes objects that the camera cannot see.
  - Spatial subdivision improves rendering efficiency by allowing the potentially visible set (PVS) of objects to be determined very quickly.



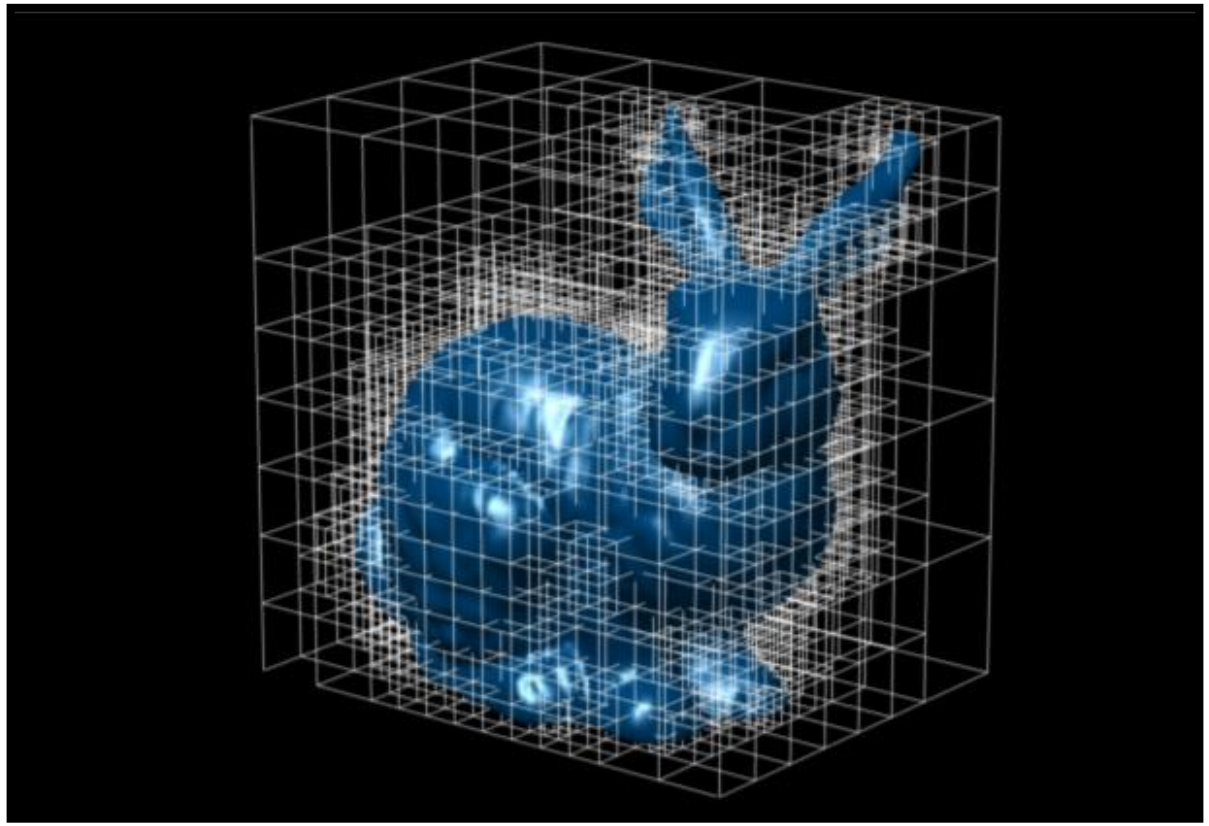
view-frustum culling



spatial subdivision

Runtime Engine Architecture - scene optimizations

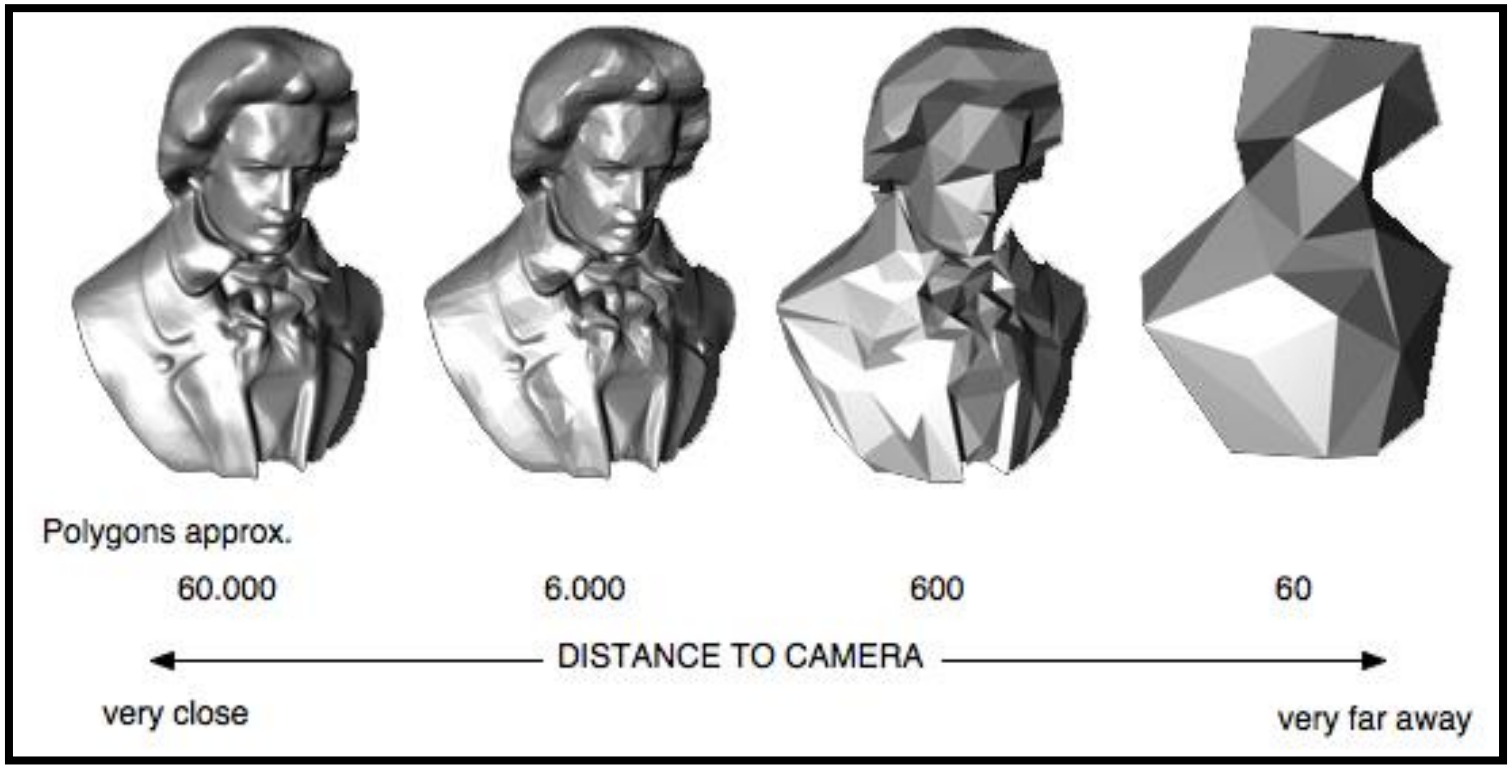
- *Frustum cull & spatial subdivision*
  - Spatial subdivision can take many forms, including a binary space partitioning tree, a quadtree, an octree, a *kd*-tree or a sphere hierarchy.



spatial subdivision

Runtime Engine Architecture - scene optimizations

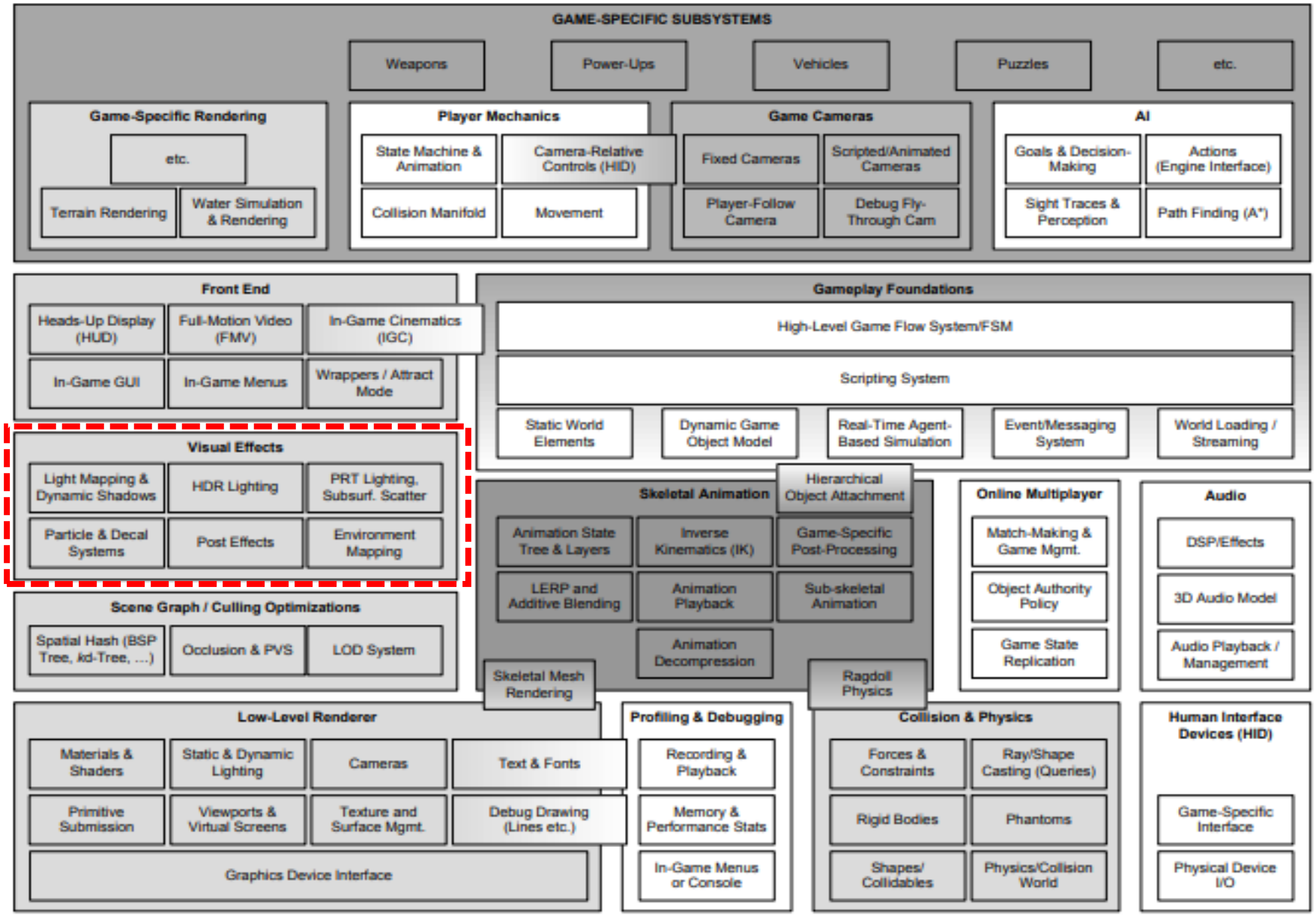
- *Level of detail*
  - Level of detail (LOD) refers to the complexity of a 3D model representation.
  - LOD can be decreased as the model moves away from the viewer or according to other metrics such as object importance, viewpoint-relative speed or position.
  - LOD techniques increase the efficiency of rendering by decreasing the workload on graphics pipeline stages.



Runtime Engine Architecture - scene optimizations

level of detail[4]

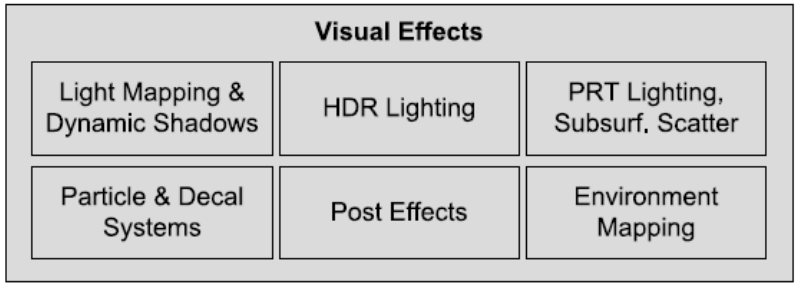
Runtime Engine Architecture - visual effects



Runtime Engine Architecture - visual effects

Visual Effects

- Modern game engines support a wide range of visual effects as follows:
  - Particle systems (for smoke, fire, water splashes, etc.)
  - Decal systems (for bullet holes, foot prints, etc.)
  - Light mapping and environment mapping
  - Dynamic shadows
  - Full-screen post effect, applied after the 3D scene has been rendered to an off-screen buffer.
    - high dynamic range (HDR) tone mapping and bloom
    - full-screen anti-aliasing (FSAA)
    - color correction and color-shift effects, including bleach bypass, saturation and desaturation effect, etc.

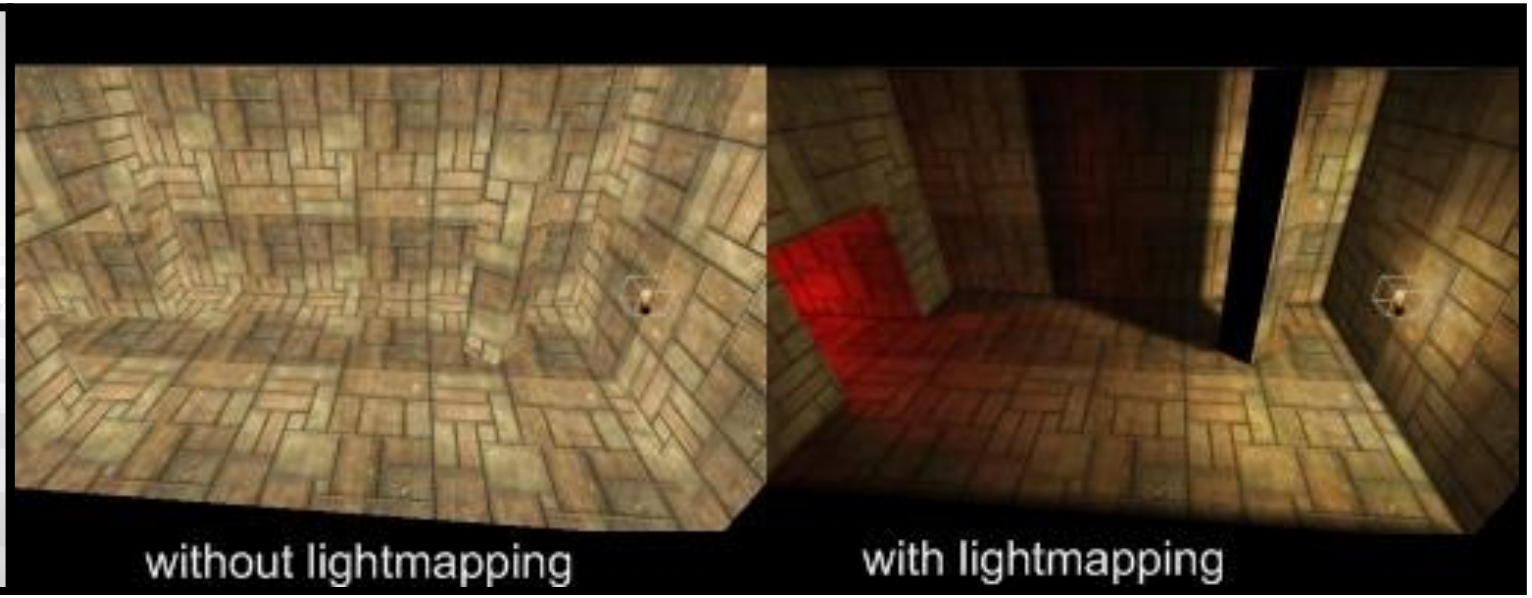


Runtime Engine Architecture - visual effects

Visual Effects

- Particles and decal systems are usually distinct components of the rendering engine and act as inputs to the low-level renderer.
- Light mapping, environment mapping and shadows are usually handled internally within the rendering engine proper.

decal systems

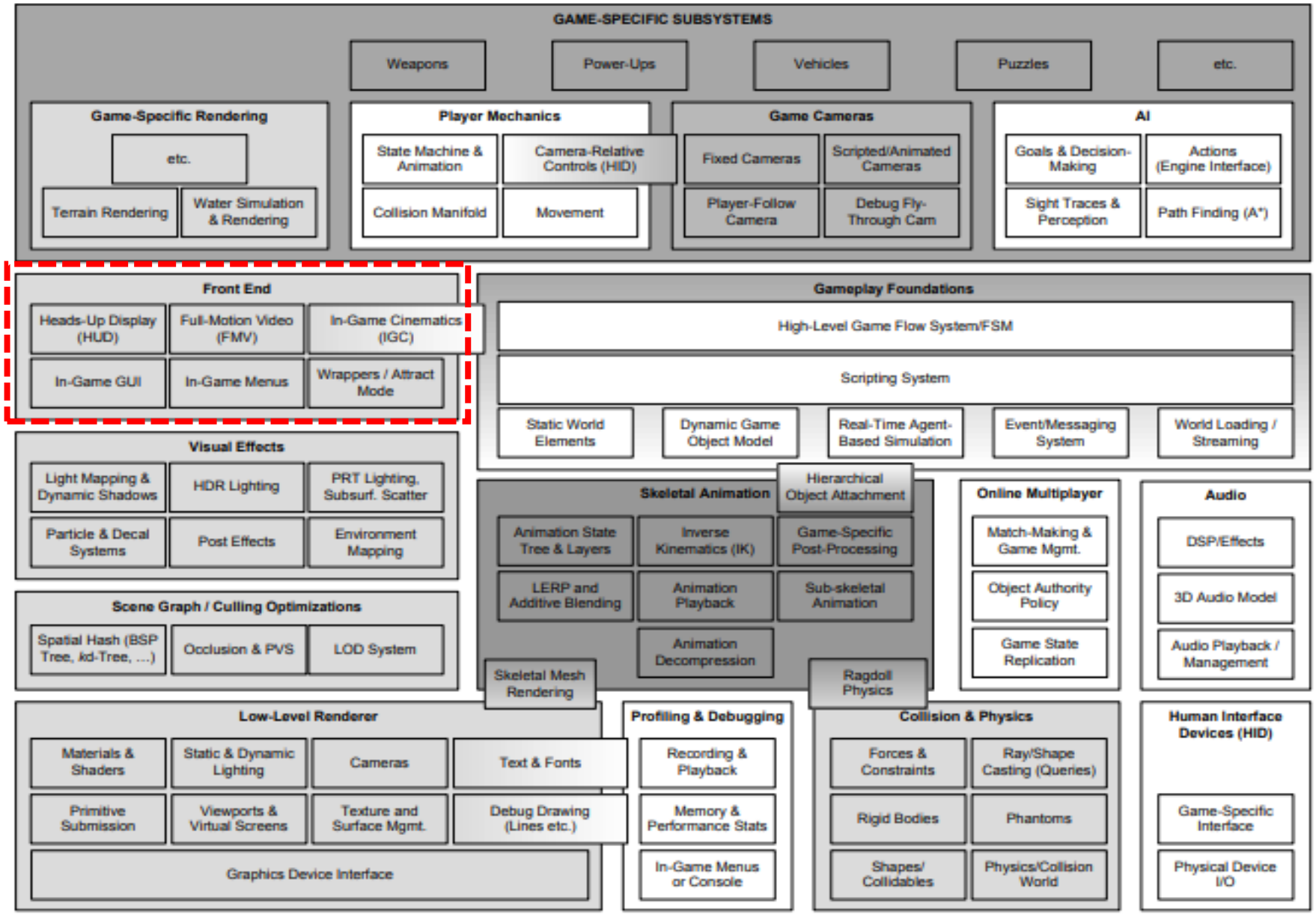


light mapping

Runtime Engine Architecture - visual effects

Unreal Engine particle system demo[4]

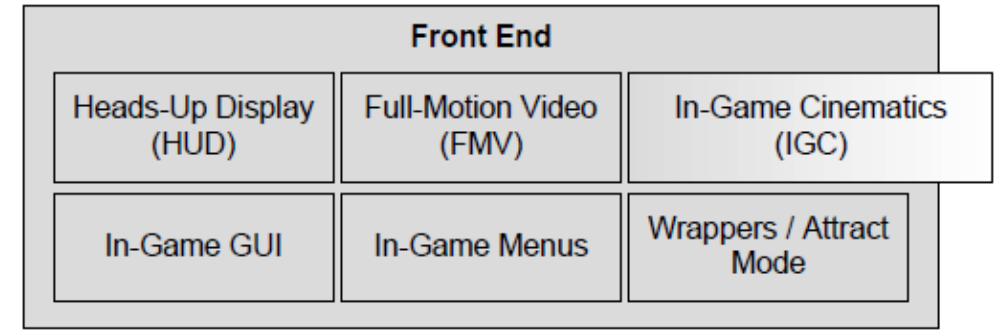
Runtime Engine Architecture - front end



Runtime Engine Architecture - front end

Front end

- Most games employ some kind of 2D graphics overlaid on the 3D scene for various purposes:
  - The game's head-up display (HUD)
  - In-game menus, a console or other development tools
  - In-game graphical user interface (GUI), allowing the player to manipulate his or her character's inventory, configure unites for battle or perform other complex in-game tasks.

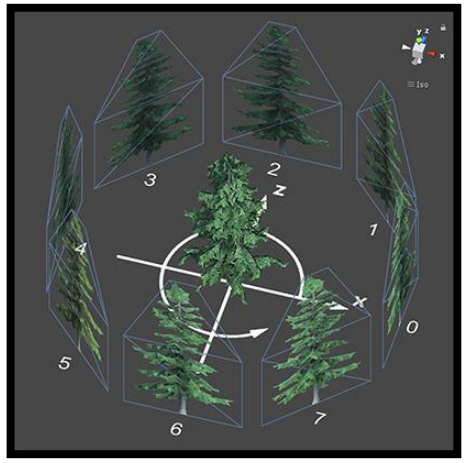


In-game GUI

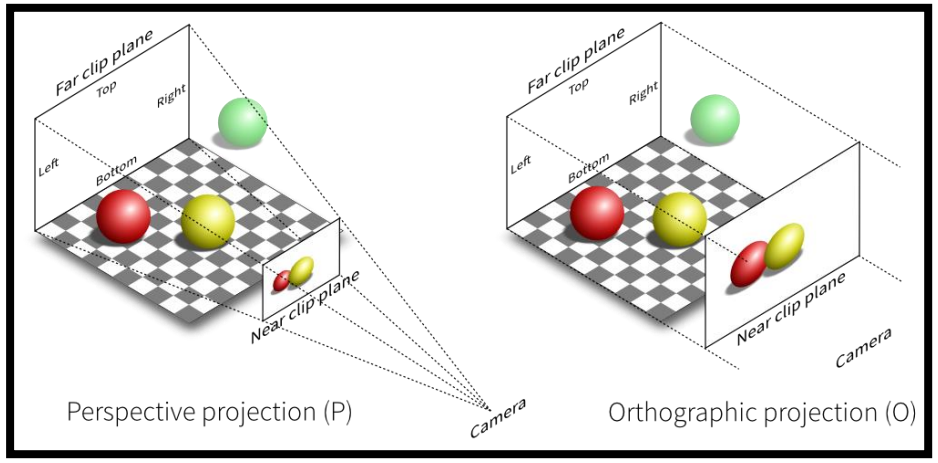
Runtime Engine Architecture - front end

Front end

- These are usually implemented by drawing textured quads (pairs of triangles) with an orthographic projection.
- Or they may be rendered in full 3D, with the quads bill-boarded so they always face the camera.
- The full-motion video (FMV) system and in-game cinematics (IGC) are also included in this layer.
  - These system is responsible for playing movies or cinematic sequences that have been recorded earlier.



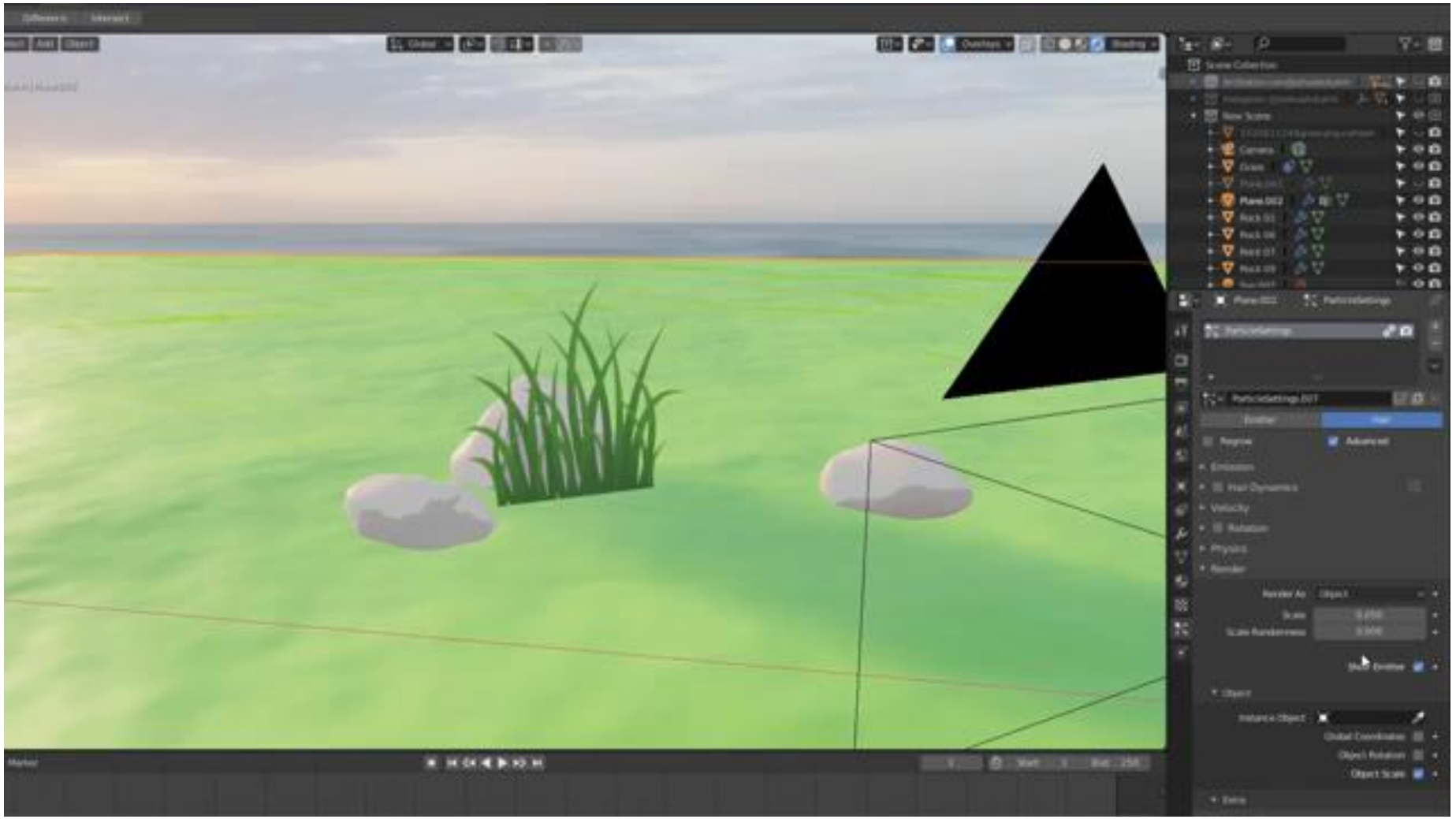
billboard



perspective vs. orthographic

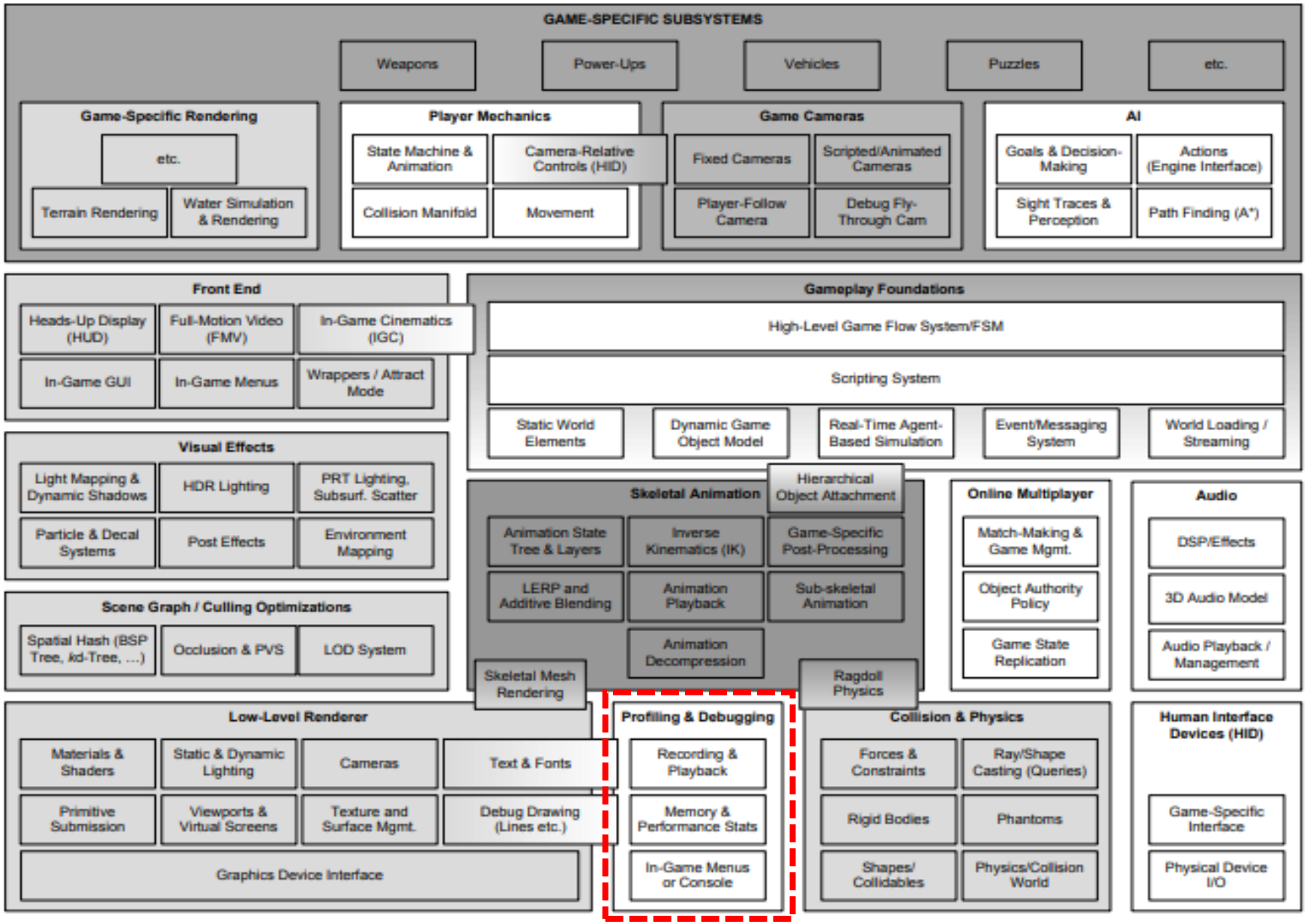
Runtime Engine Architecture - front end

Front end



billboard[bill]

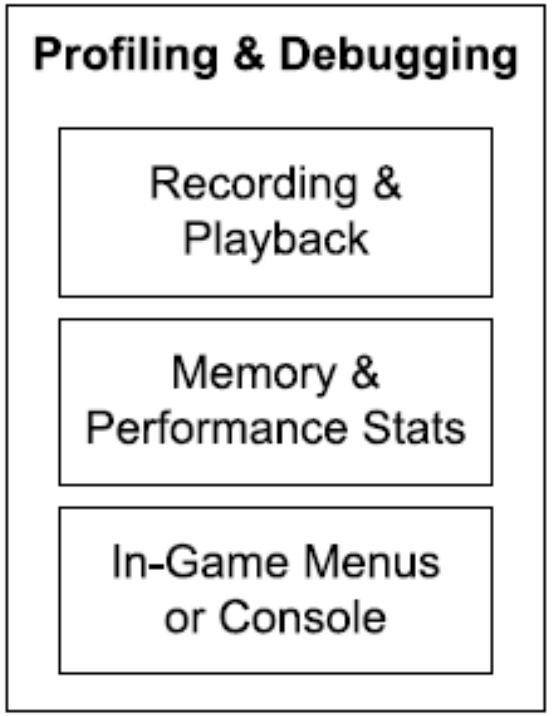
Runtime Engine Architecture - Profiling and Debugging



Runtime Engine Architecture - Profiling and Debugging

Tools for profiling and debugging

- Game engineers often need to profile the performance of their games.
- Memory resources are scarce, so developers make memory analysis tools as well.



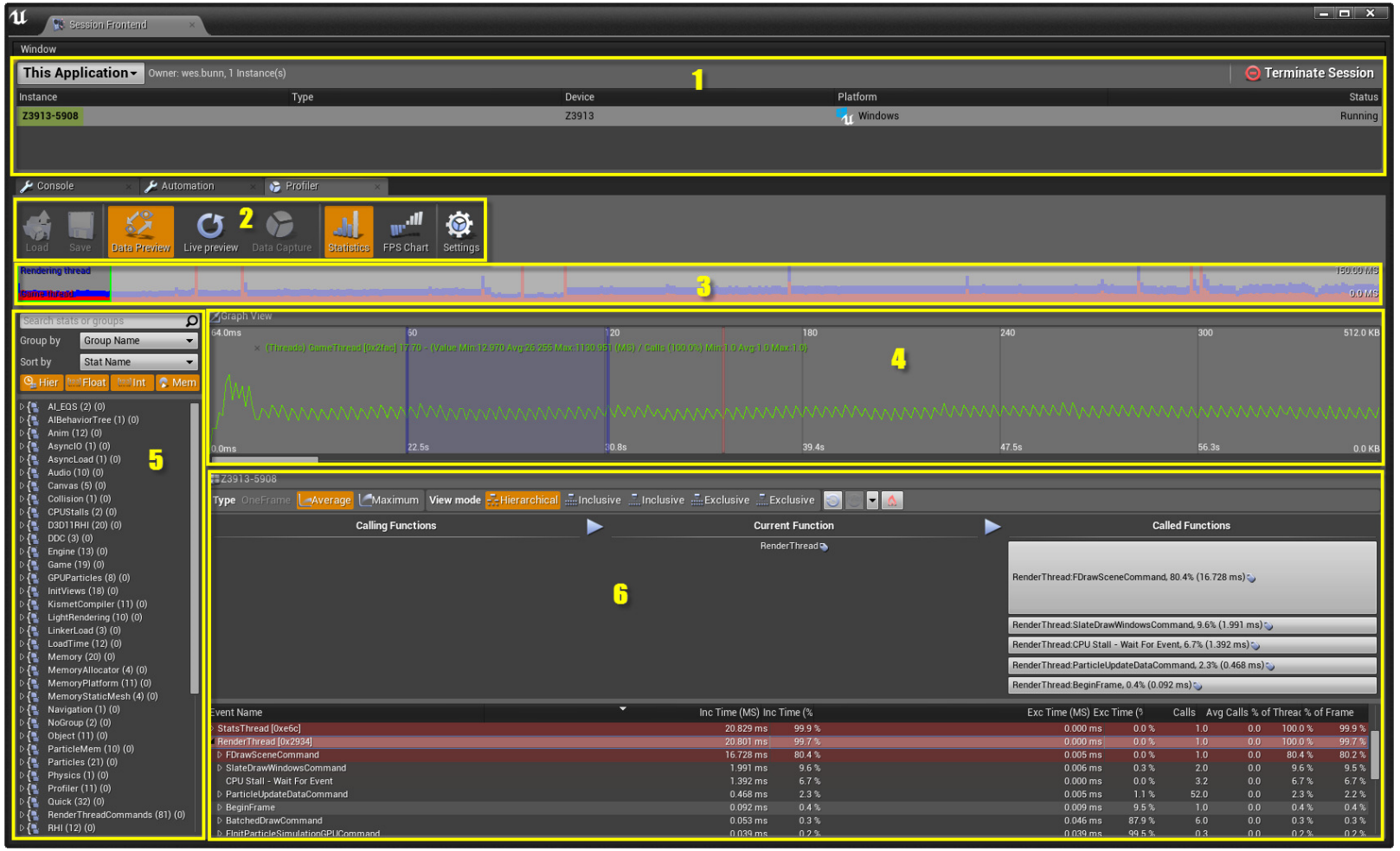
Runtime Engine Architecture - Profiling and Debugging

Tools for profiling and debugging

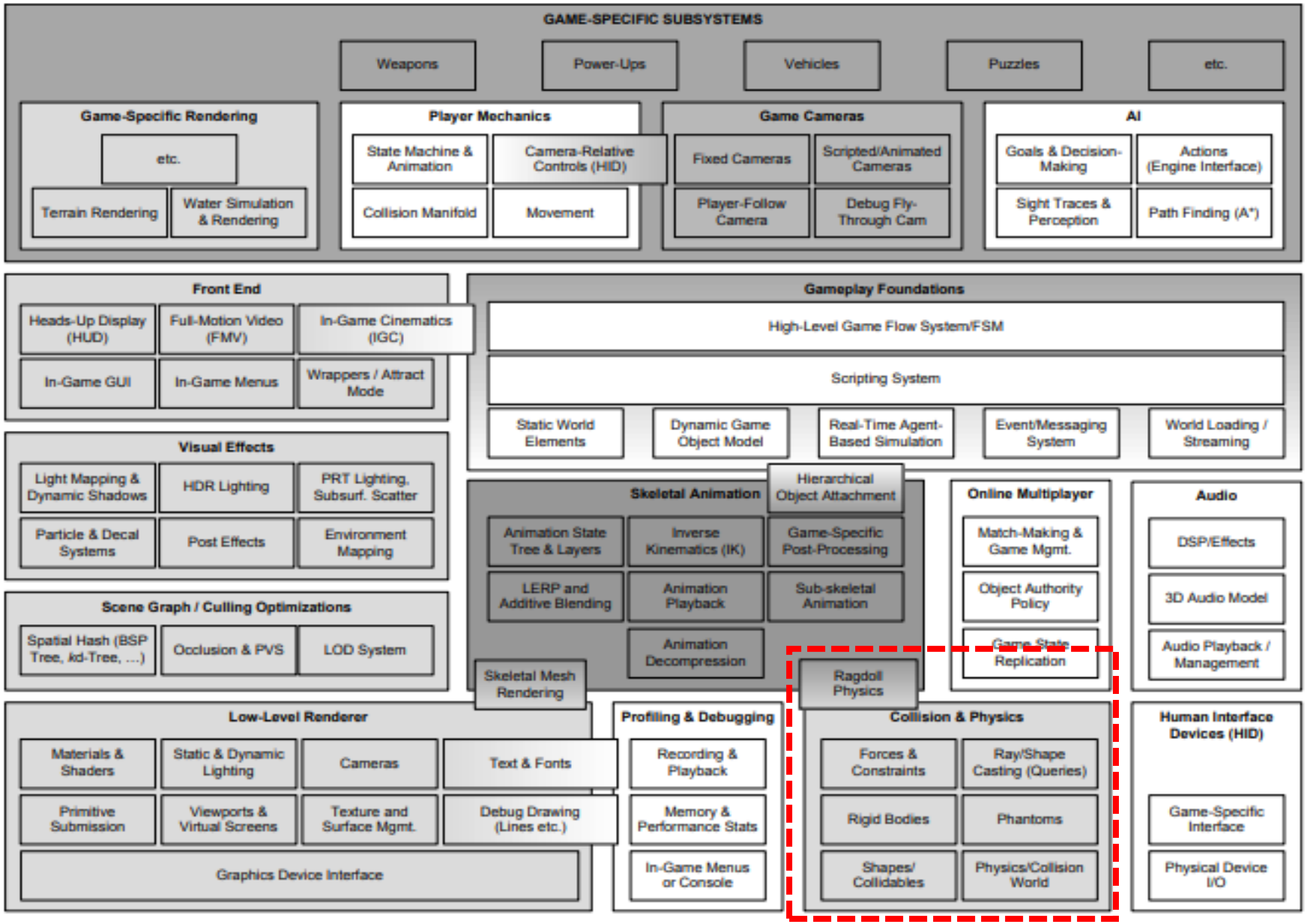
- To support these purposes, game engine encompasses tools including followings:
  - A mechanism for manually instrumenting the code,
    so that specific sections of code can be timed.
  - A facility for displaying the profiling statistics on-screen while the game is running.
  - A facility for dumping performance stats to a test file or to an Excel spreadsheet.
  - A facility for determining how much memory is being used by the engine,
    and by each subsystem, including various on-screen displays.
  - The ability to dump memory usage, high water mark and leakage stats when the game terminates during gameplay.
  - Tools that allow debug print statements to be peppered throughout the code, along with an ability to turn on or off different categories of debug output and control the level of verbosity of the output.
  - The ability to record game events and then play them back.

Runtime Engine Architecture - Profiling and Debugging

- Unreal engine also has profiler
  1. Connected session and session information
  2. Main toolbar
  3. Data graph (full)
  4. Data graph (local)
  5. Filter and presets
  6. Event graph



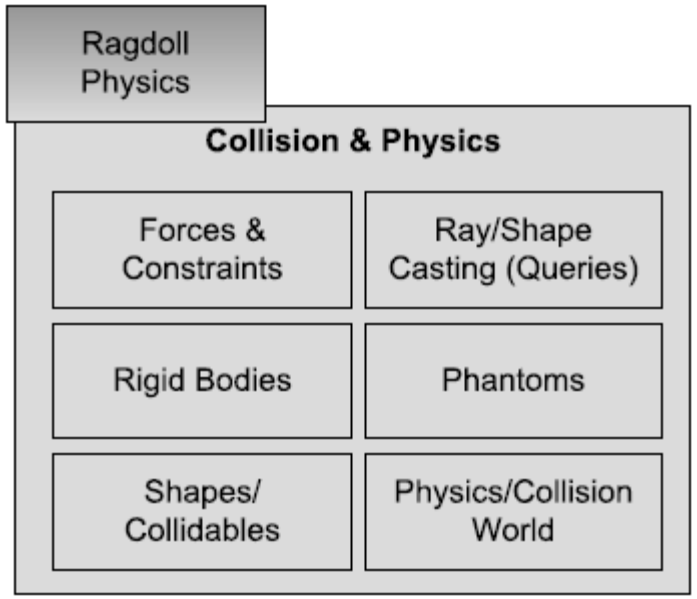
Runtime Engine Architecture - Collision & Physics



Runtime Engine Architecture - Collision & Physics

Collision and physics

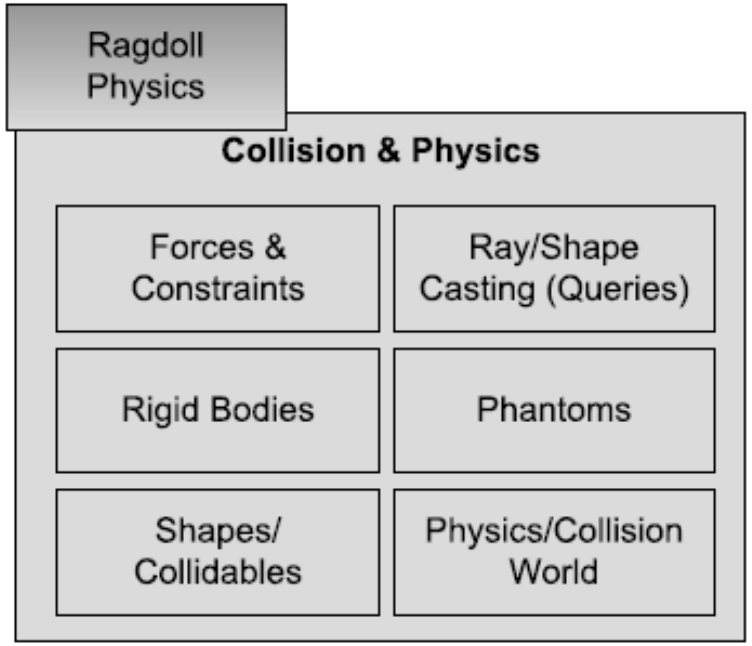
- Collision detection is important for every game.
- In the game industry, rigid body dynamics is important because we are usually only concerned with the motion (kinematics) of rigid bodies and the forces and torques (dynamics) that cause this motion to occur.
- Collision and physics are usually quite tightly coupled. This is because when collisions are detected, they are almost always resolved as part of the physics integration and constraint satisfaction logic.



Runtime Engine Architecture - Collision & Physics

Collision and physics

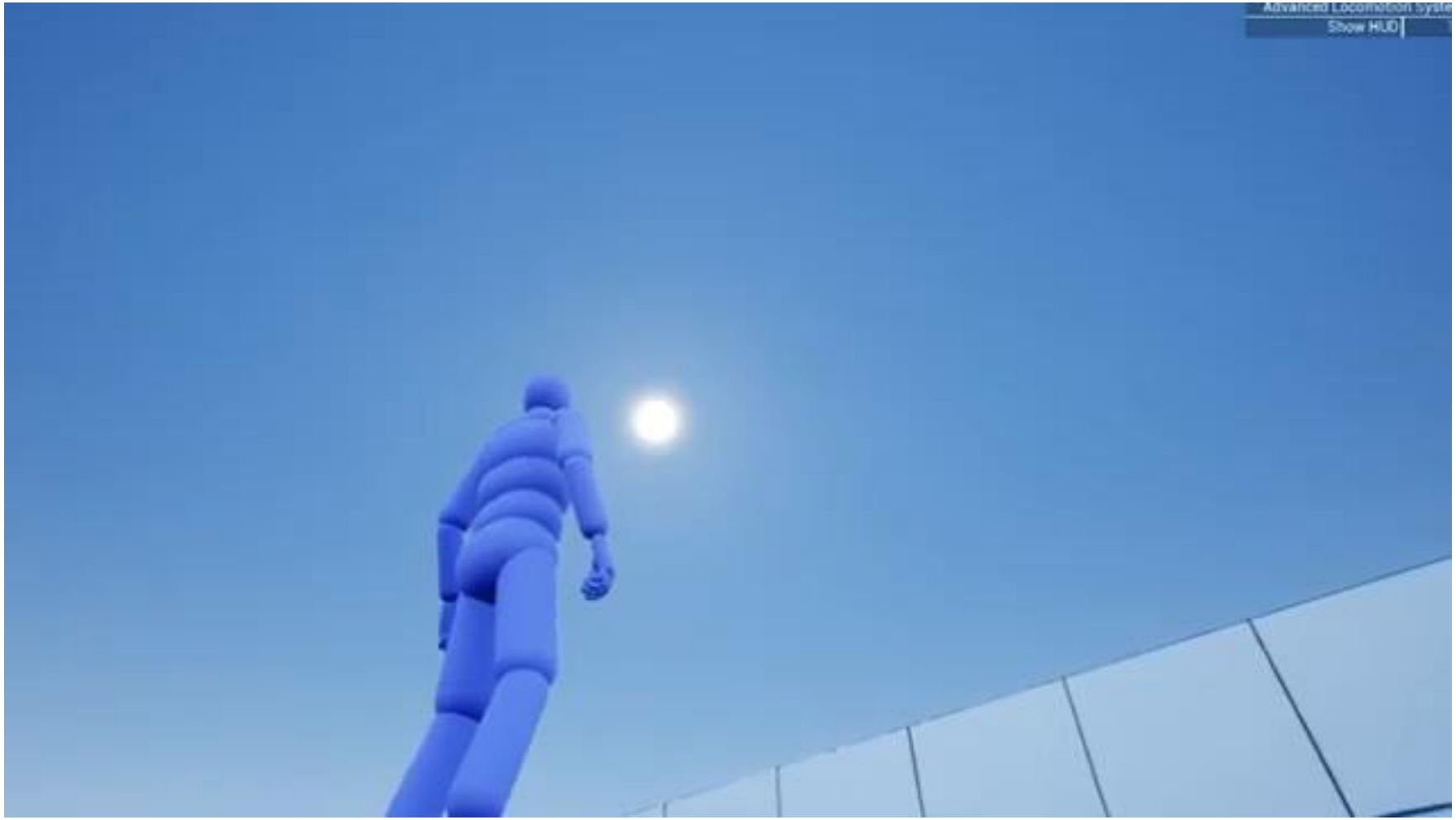
- Ragdoll physics
  - A ragdoll is a collection of multiple rigid bodies tied together by a system of constraints that restrict how the bones may move relative to each other.
  - For example, when the character dies, the body begins to collapse to the ground, honoring restrictions on each of the joints' motion.



ragdoll physics

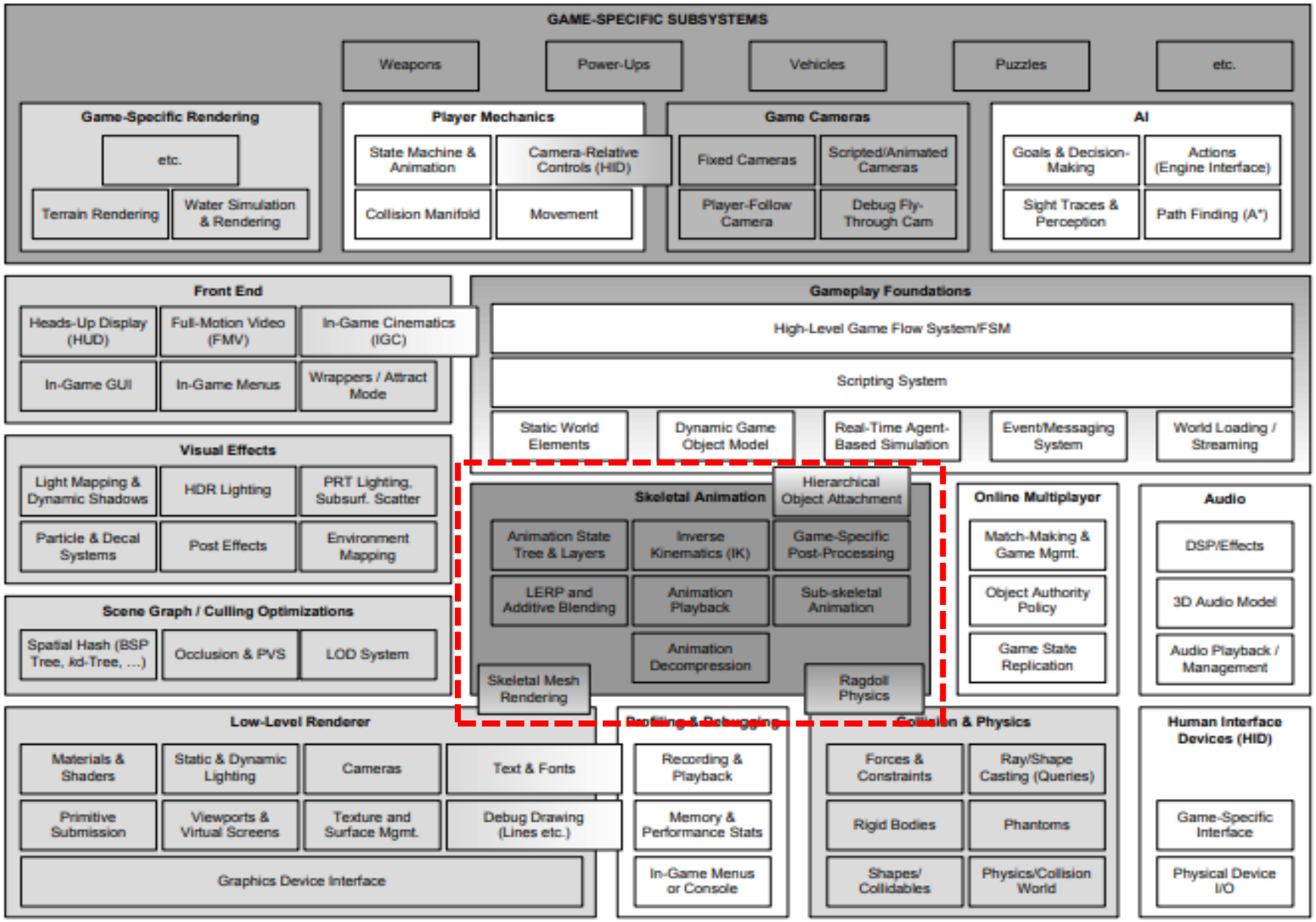
Runtime Engine Architecture - Collision & Physics

Collision and physics



ragdoll in Unreal 5[rag]

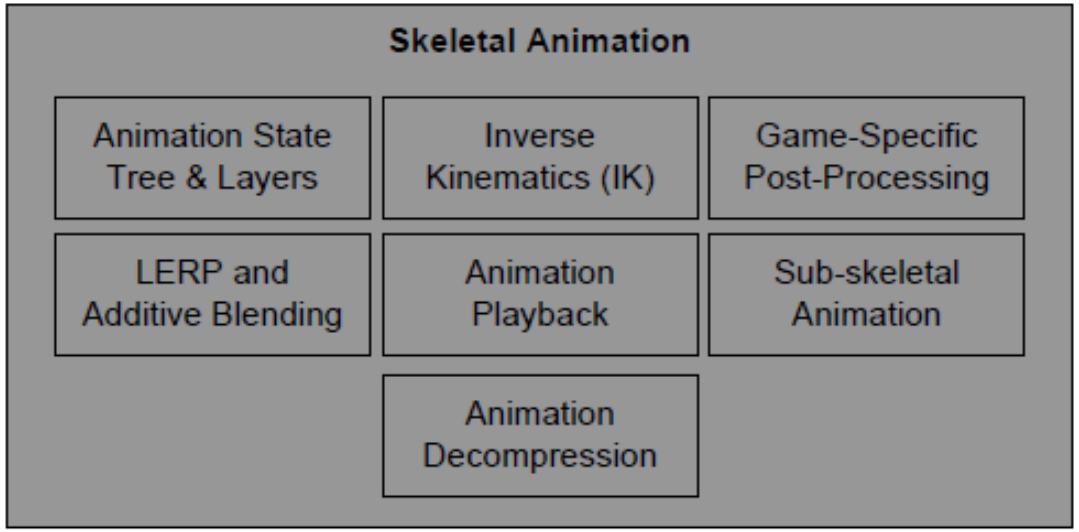
Runtime Engine Architecture - Skeletal Animation

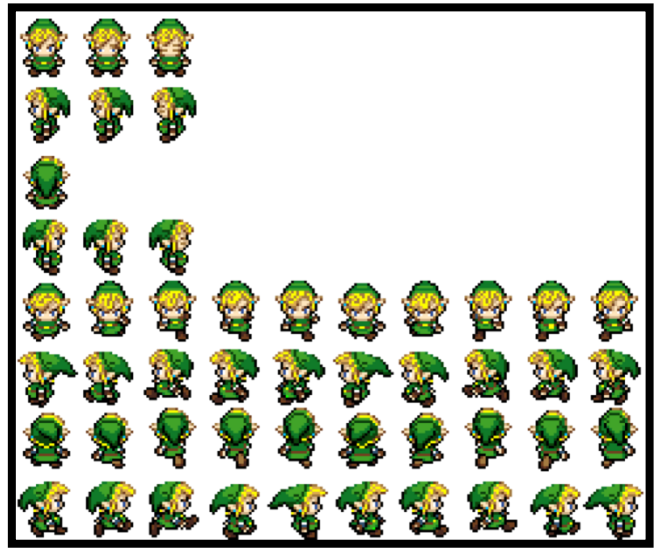


Runtime Engine Architecture - Skeletal Animation

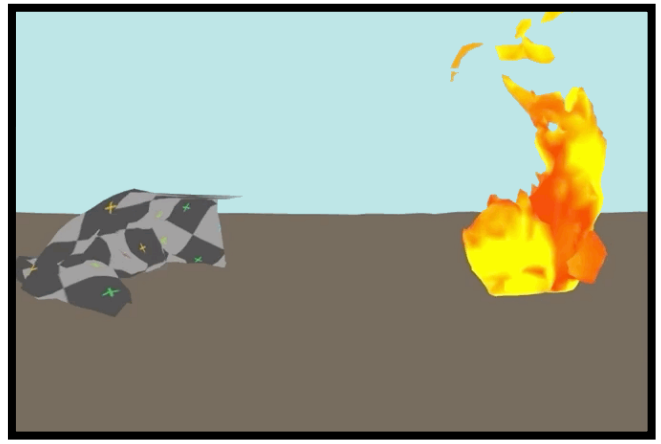
Skeletal Animation

- There are five basic types of animation used in games:
  - Sprite/texture animation
  - Rigid body hierarchy animation
  - Skeletal animation
  - Vertex animation
  - Morph targets





sprite animation

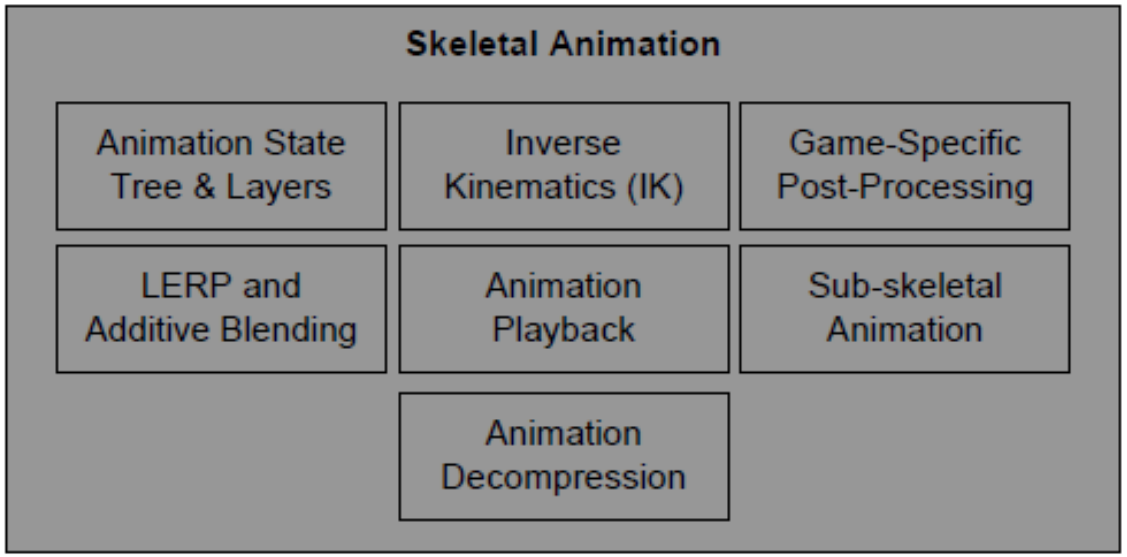


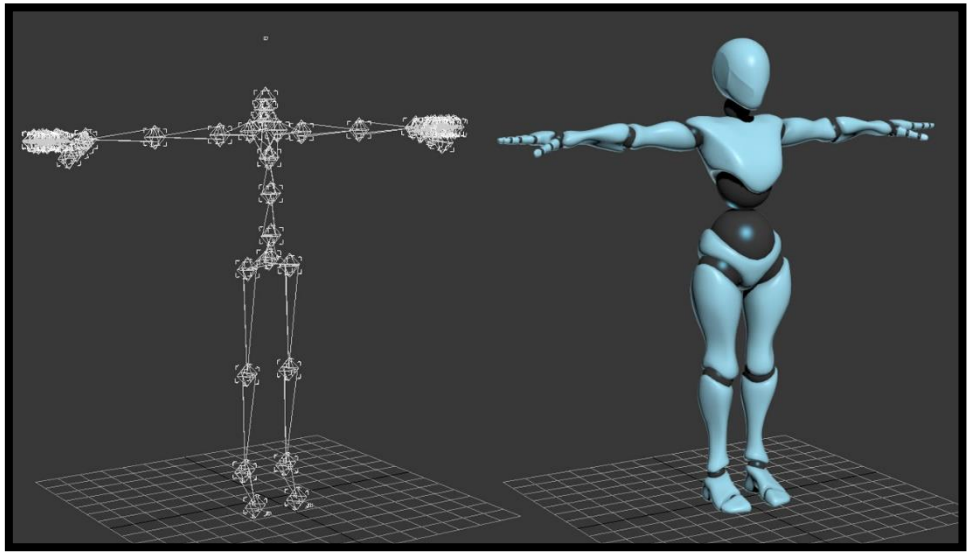
vertex animation

Runtime Engine Architecture - Skeletal Animation

Skeletal Animation

- Skeletal animation is the most prevalent animation method in games although morph targets and vertex animation can be used.
  - The animation system produces a pose for every bone in the skeleton, and then these poses are passed to the rendering engine as a palette of matrices.
  - The renderer transforms each vertex by the matrix of matrices in the palette, in order to generate a final blended vertex position.

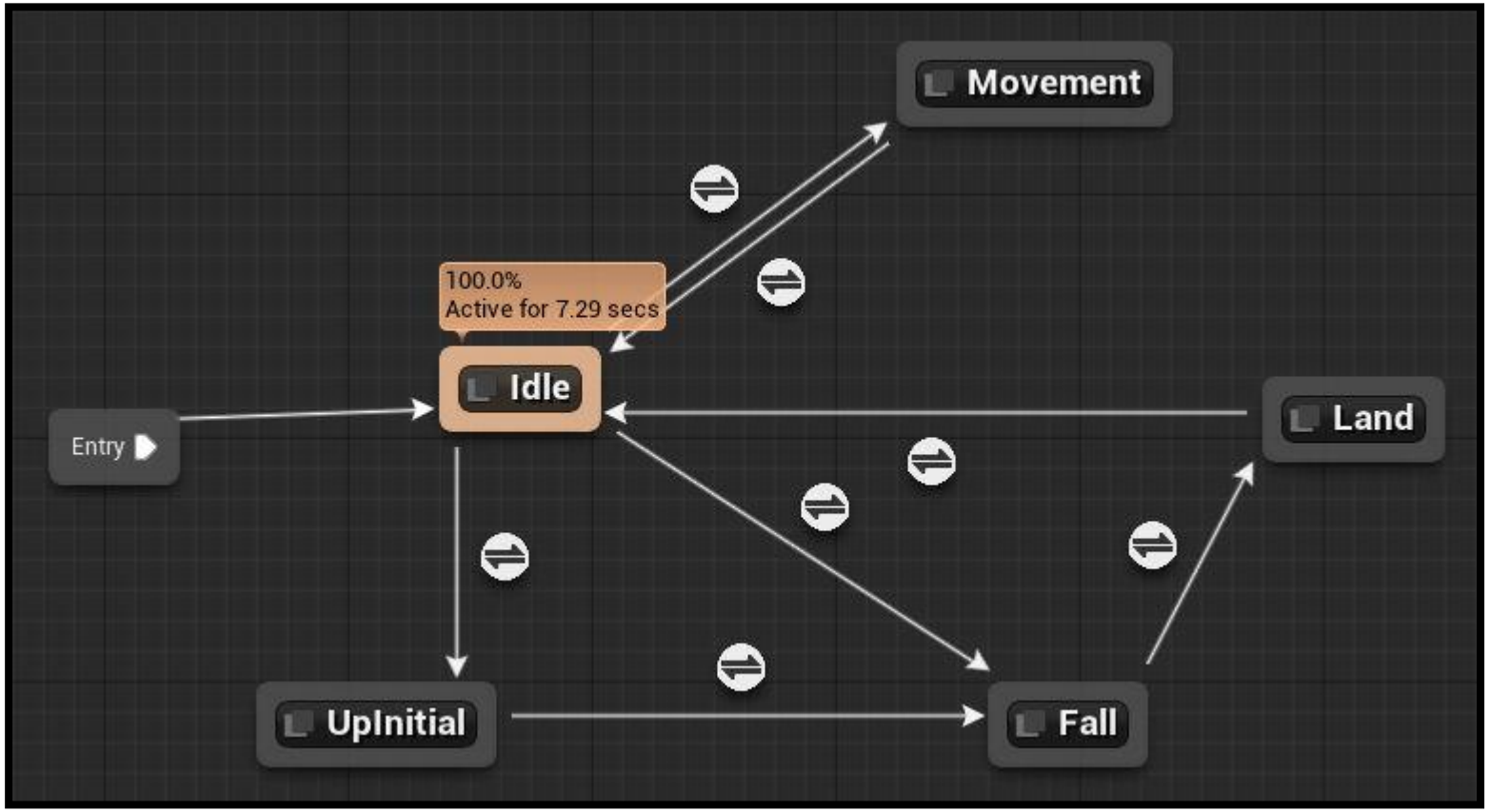


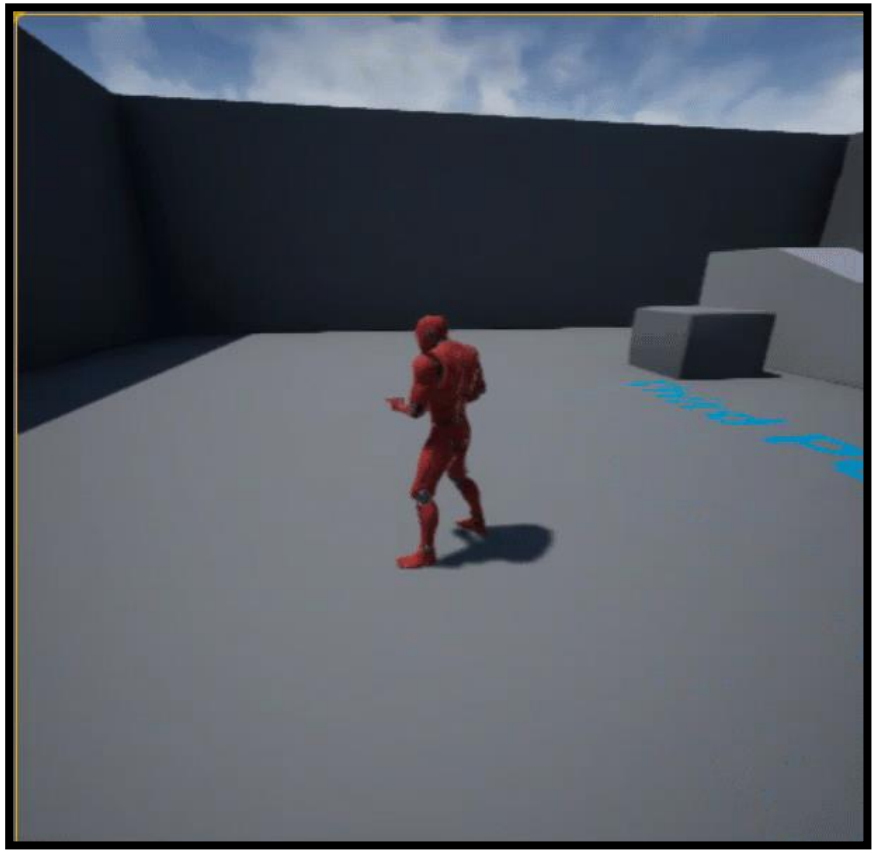


Runtime Engine Architecture - Skeletal Animation

Skeletal Animation

- *Animation state tree & layers*

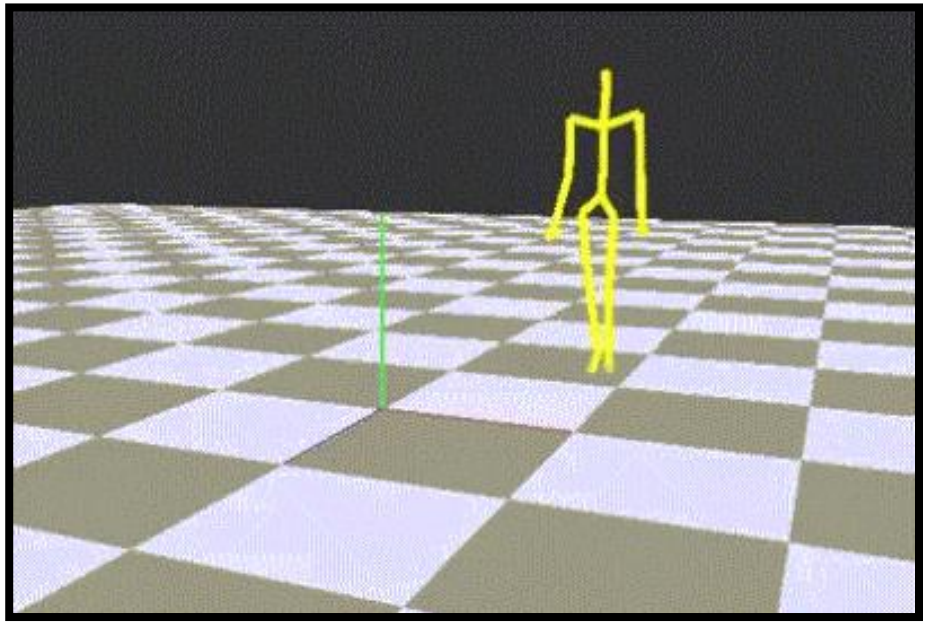


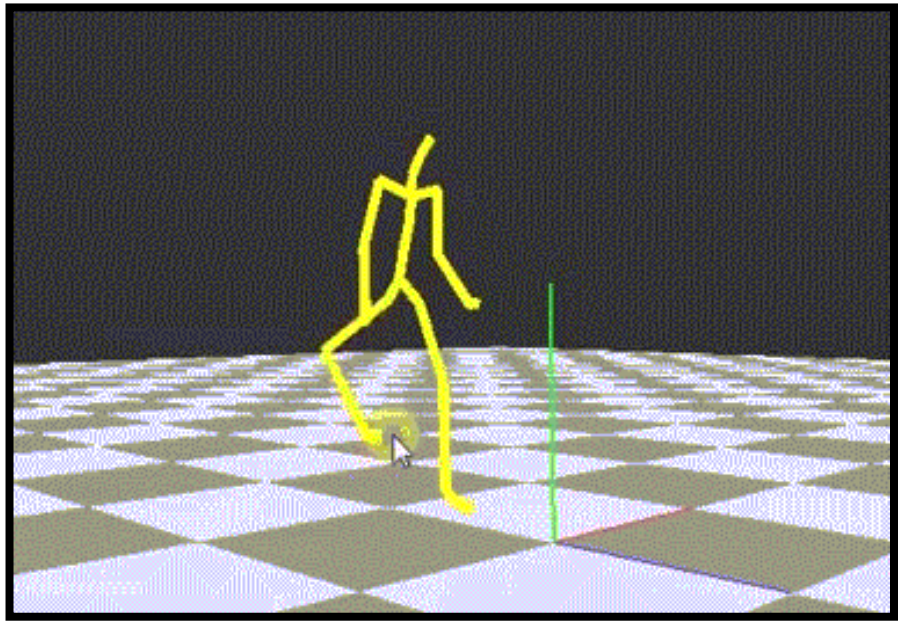


Runtime Engine Architecture - Skeletal Animation

Skeletal Animation

- *Inverse kinematics*
  - Inverse kinematics is the mathematical process of calculating the variable joint parameters needed to place the end of a kinematic chain in a given position and orientation relative to the start of the chain.





forward kinematics vs. inverse kinematics

Runtime Engine Architecture - Skeletal Animation

Skeletal Animation

- *Additive animation*
  - Additive animation is the process of combining animations using difference clips.
  - For example, we can create ‘reload while walking’ animation by adding ‘reload’ and ‘walking’ animations.

additive animation

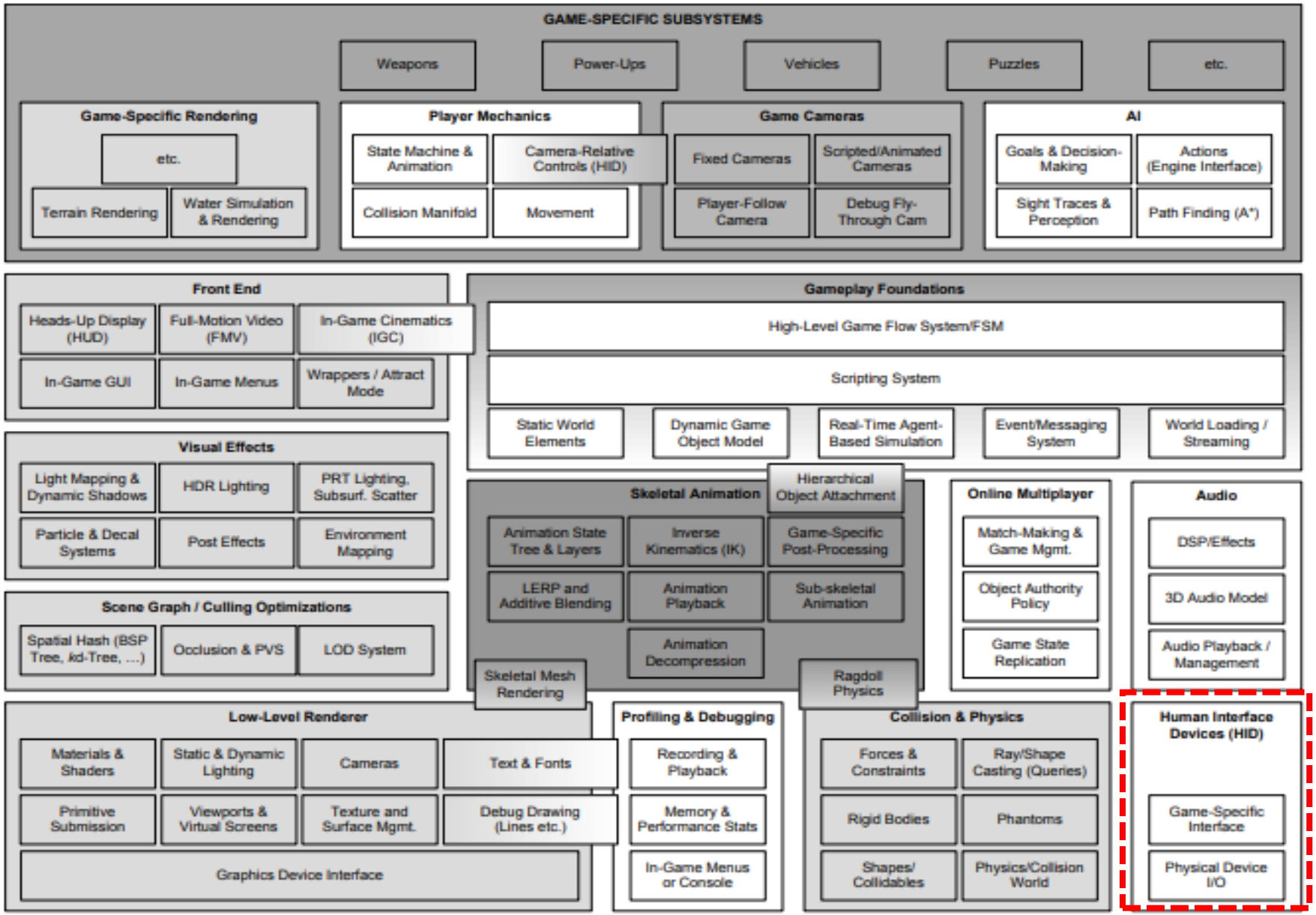
Runtime Engine Architecture - Skeletal Animation

Skeletal Animation

- *Additive animation*

additive animation[6]

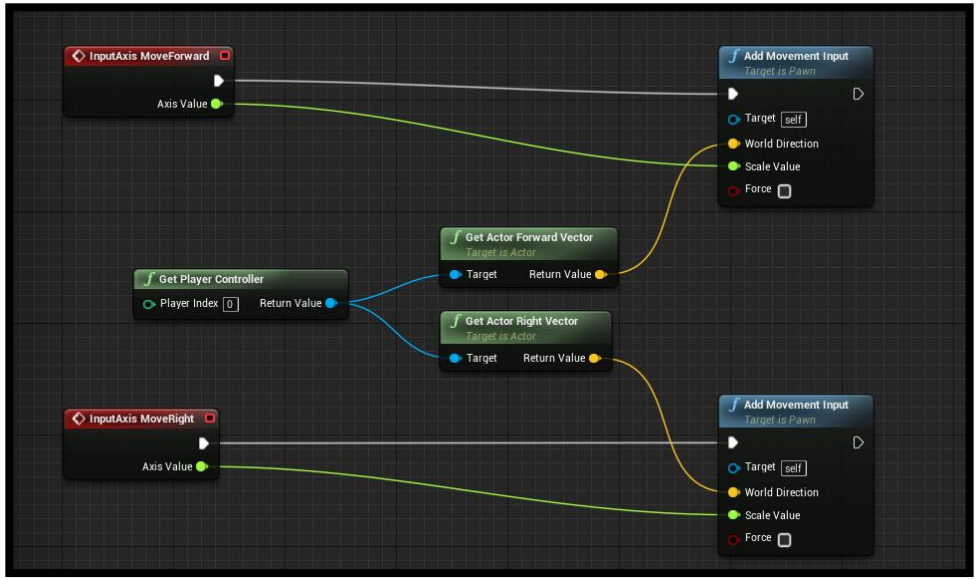
Runtime Engine Architecture - HID



Runtime Engine Architecture - HID

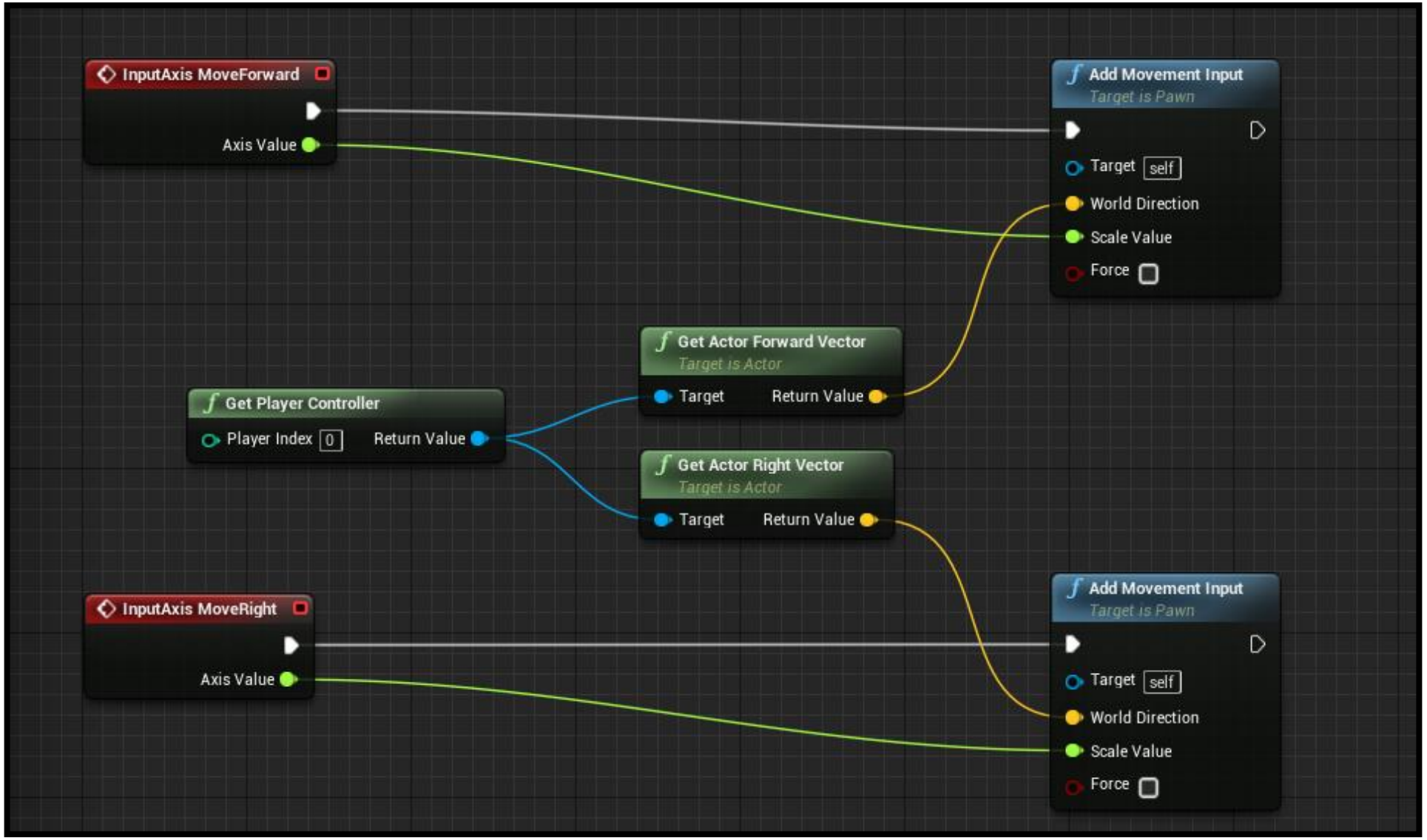
Human interface devices (HID)

- Every game needs to process input from the player, obtained from various human interface devices (HIDs) including:
  - The keyboard and mouse
  - A joypad
  - Other specialized game controllers, like steering wheels, fishing rods, dance pads, the wiimote, etc.
- The HID messages the raw data coming from the hardware.

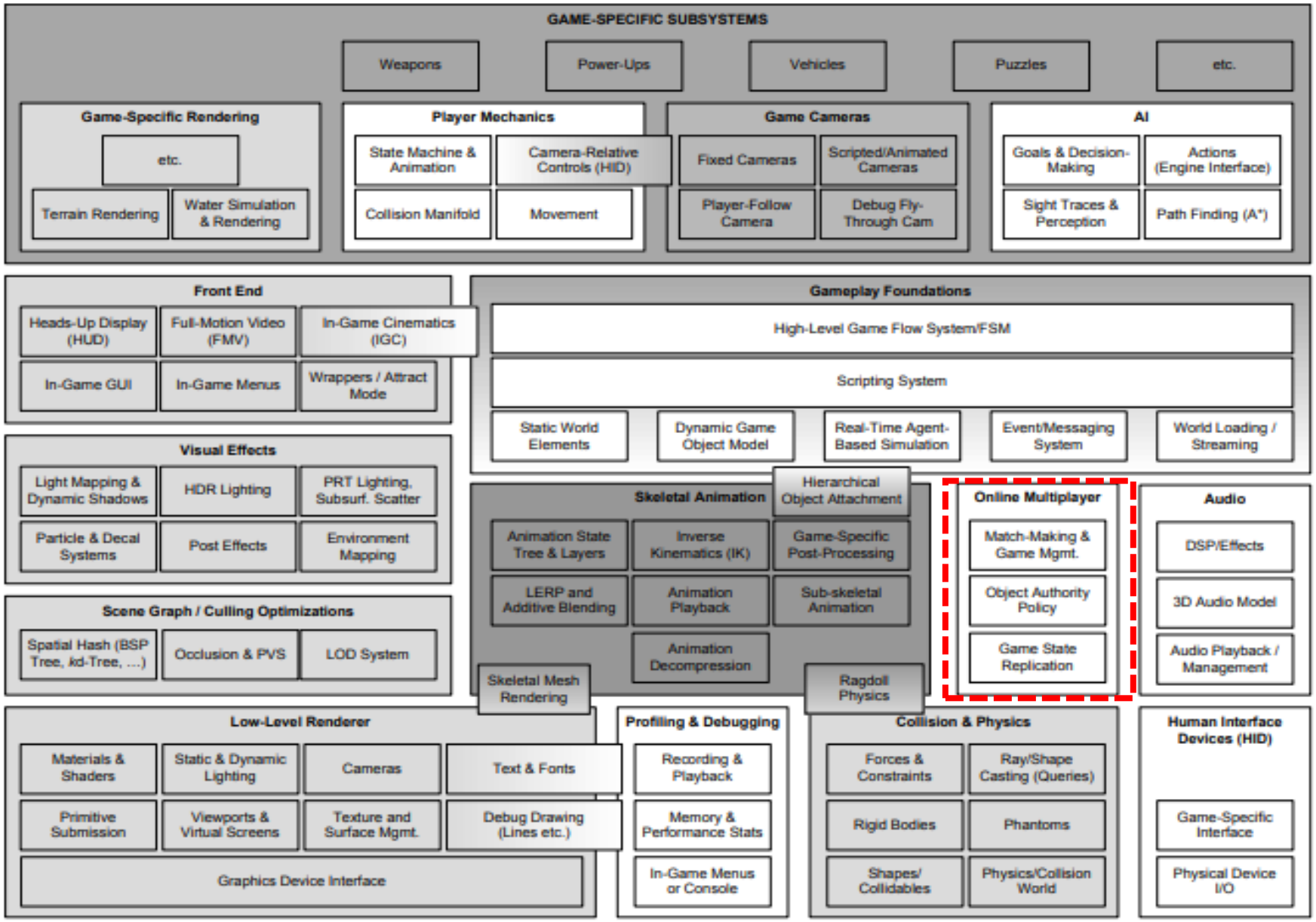


Runtime Engine Architecture - HID

Human interface devices (HID)



Runtime Engine Architecture - Online Multiplayer



Runtime Engine Architecture - Online Multiplayer

Online multiplayer & network

- Many games permit multiple human players to play within a single virtual world.
  - Single-screen multiplayer: two or more human interface devices connected to a single machine. Multiple player characters inhabit a single virtual world.
  - Split-screen multiplayer: Multiple player characters inhabit a single virtual world, with multiple HIDs attached to a single game machine, but each with its own camera, and the screen is divided into sections so that each player can view their character.

single-screen multiplayer

split-screen multiplayer

networked multiplayer

Runtime Engine Architecture - Online Multiplayer

Online multiplayer & network

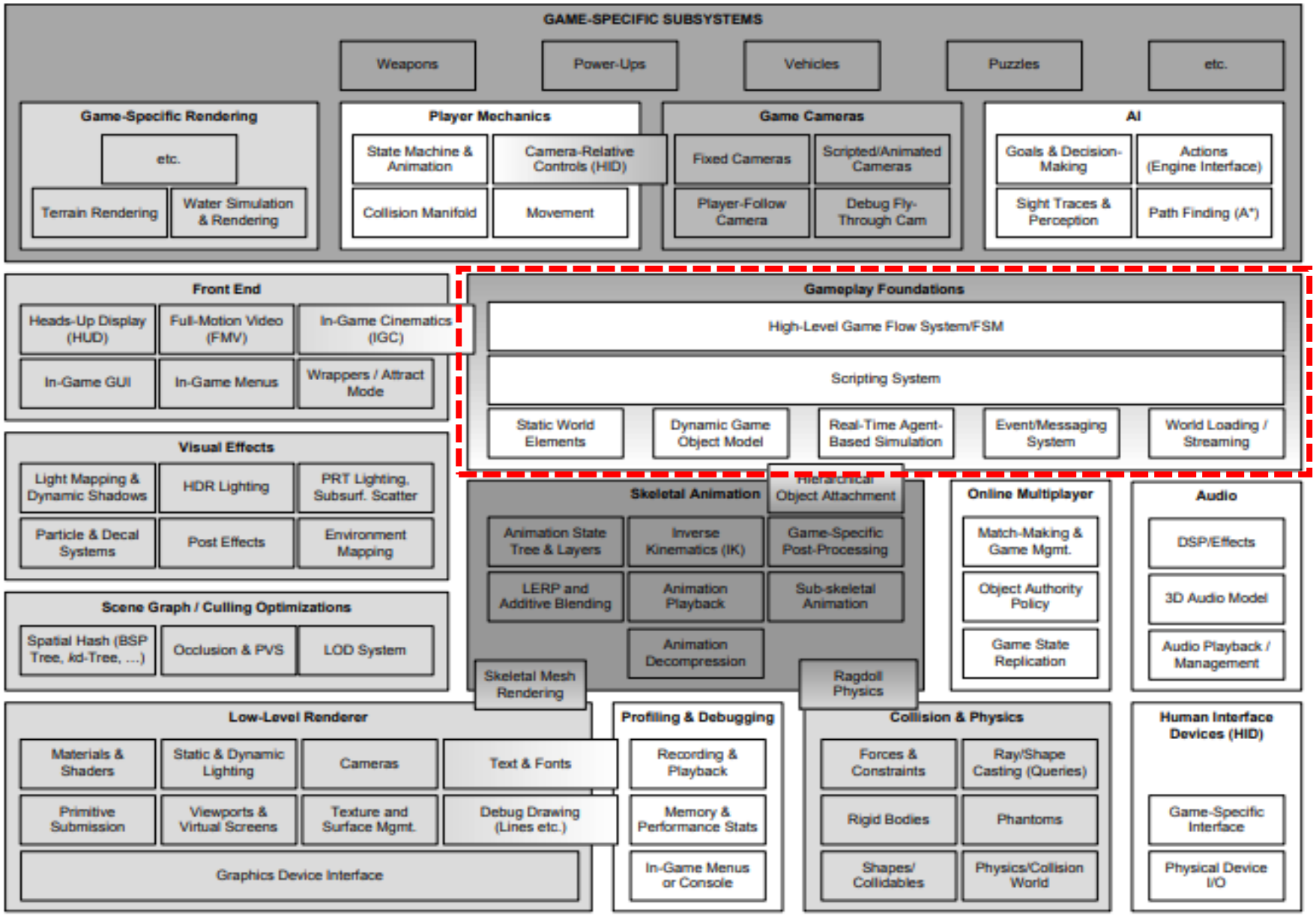
- Many games permit multiple human players to play within a single virtual world.
  - Networked multiplayer: Multiple computers or consoles are networked together, with each machine hosting one of the players.

single-screen multiplayer

split-screen multiplayer

networked multiplayer

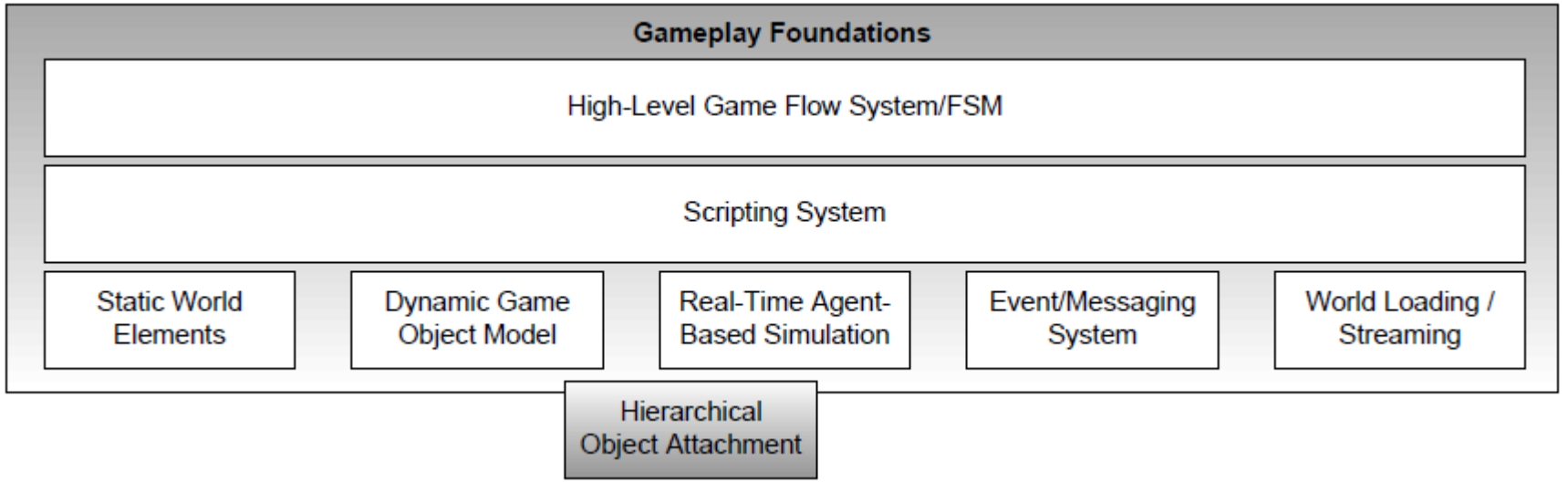
Runtime Engine Architecture - Gameplay Foundation



Runtime Engine Architecture - Gameplay Foundation

Gameplay foundation systems

- The term ‘gameplay’ refers to the action that takes place in the game, the rules that govern the virtual world in which the game takes place, the abilities of the player character (known as player mechanics) and of the other characters and objects in the world, and the goals and objectives of the player.
- Gameplay is typically implemented either in the native language in which the rest of the engine is written or in a high-level scripting language.
- To bridge the gap between the gameplay code and the low-level engine systems, most game engines introduce a layer called ‘gameplay foundations’.



Runtime Engine Architecture - Gameplay Foundation

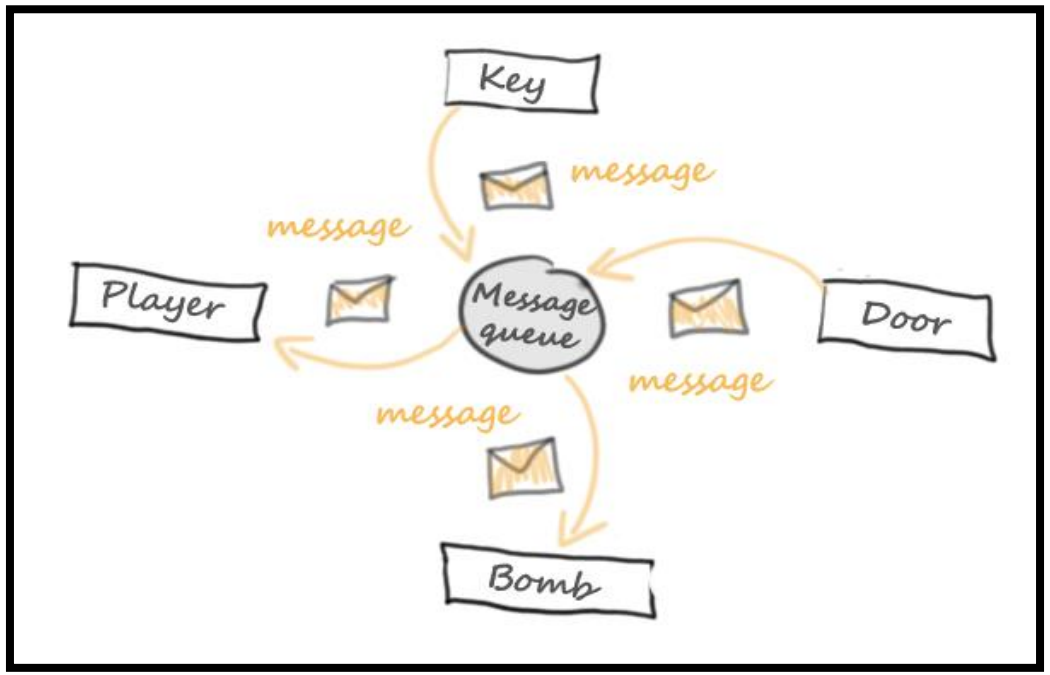
Gameplay foundation systems

- Game worlds and objects models
  - Game world contains both static and dynamic elements.
  - The contents of the world are usually modeled in an object-oriented manner.
- Typical types of game objects include:
  - Static background geometry, like buildings, roads, terrain, etc.
  - Dynamic rigid bodies, such as rocks, chairs, etc.
  - Player Characters
  - Non-player characters
  - Weapons
  - Projectiles
  - Lights
  - Cameras
  - …

Runtime Engine Architecture - Gameplay Foundation

Gameplay foundation systems

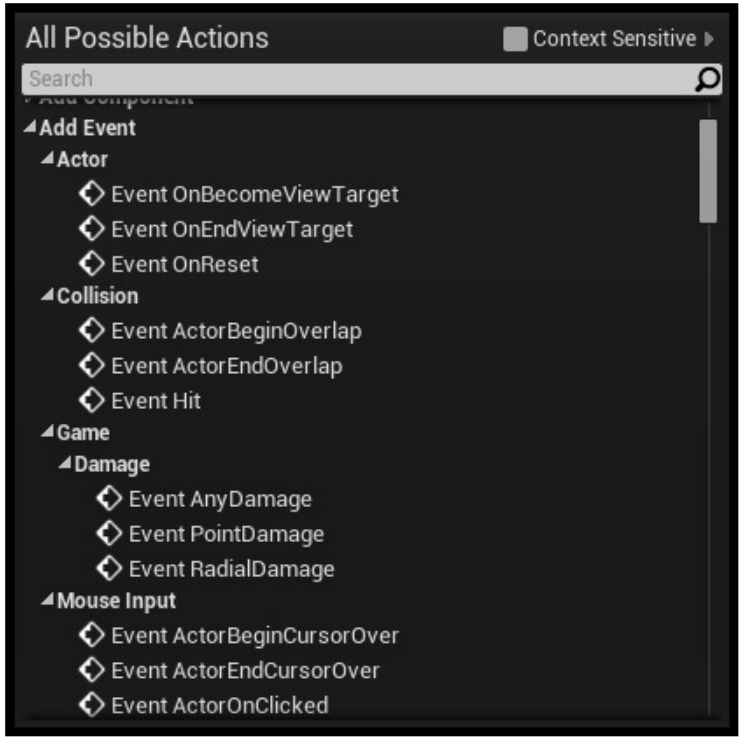
- Event system
  - Game objects need to communicate with one another.
  - In an event-driven system, the sender creates a little data structure called an event or message, containing the message's type and any argument data that are to be sent.
  - The event is passed to the receiver object by calling its *event handler function*.
  - Events can be stored in a queue for handling at some future time.

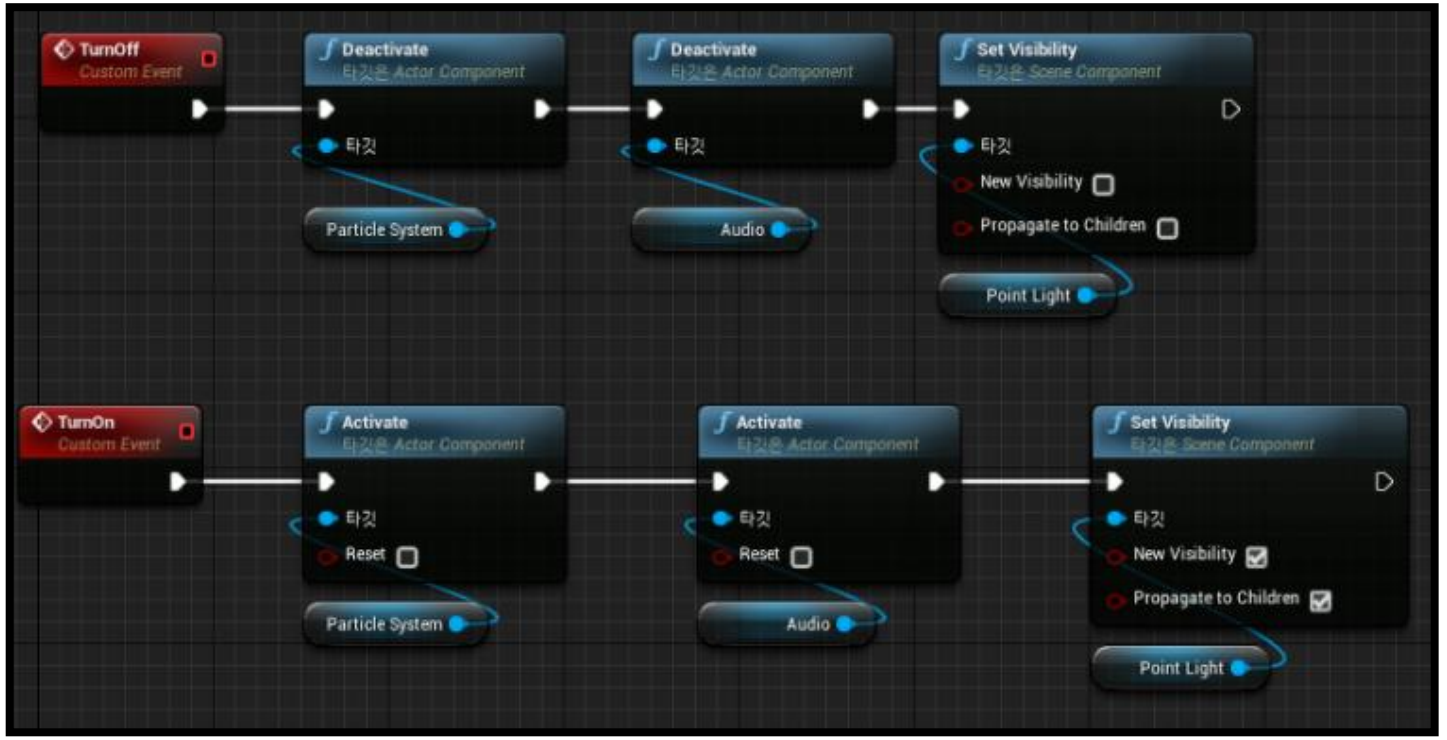


Runtime Engine Architecture - Gameplay Foundation

Gameplay foundation systems

- Event system in Unreal engine
  - Events are called from gameplay code to begin execution of an individual network within the EventGraph.
  - They enable Blueprints to perform a series of actions in response to certain events that occur within the game, such as when the game starts, when a level resets, or when a player takes damage.

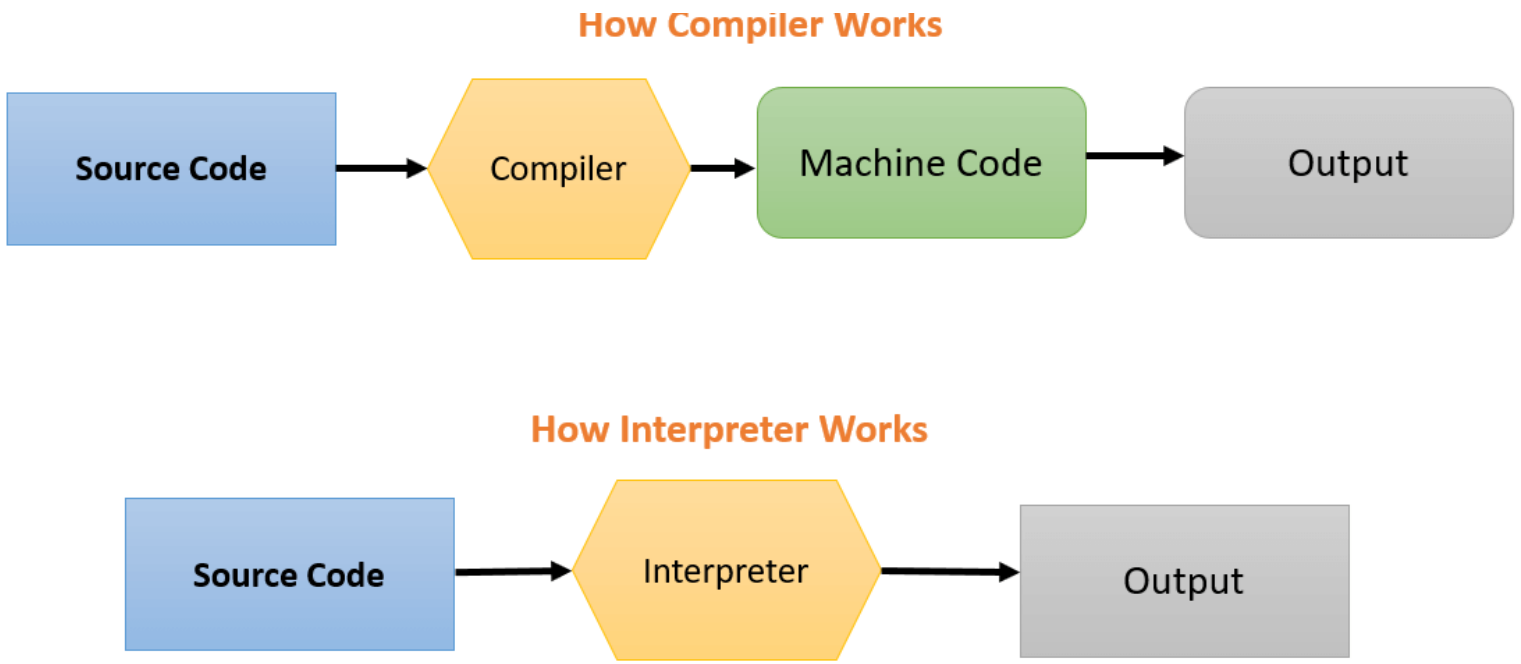




Runtime Engine Architecture – Gameplay Foundation

Gameplay foundation systems

- Scripting system
  - A script language is a programming language for a special run-time environment that automates the execution of tasks.
  - Scripting languages are often *interpreted*, rather than *compiled*.

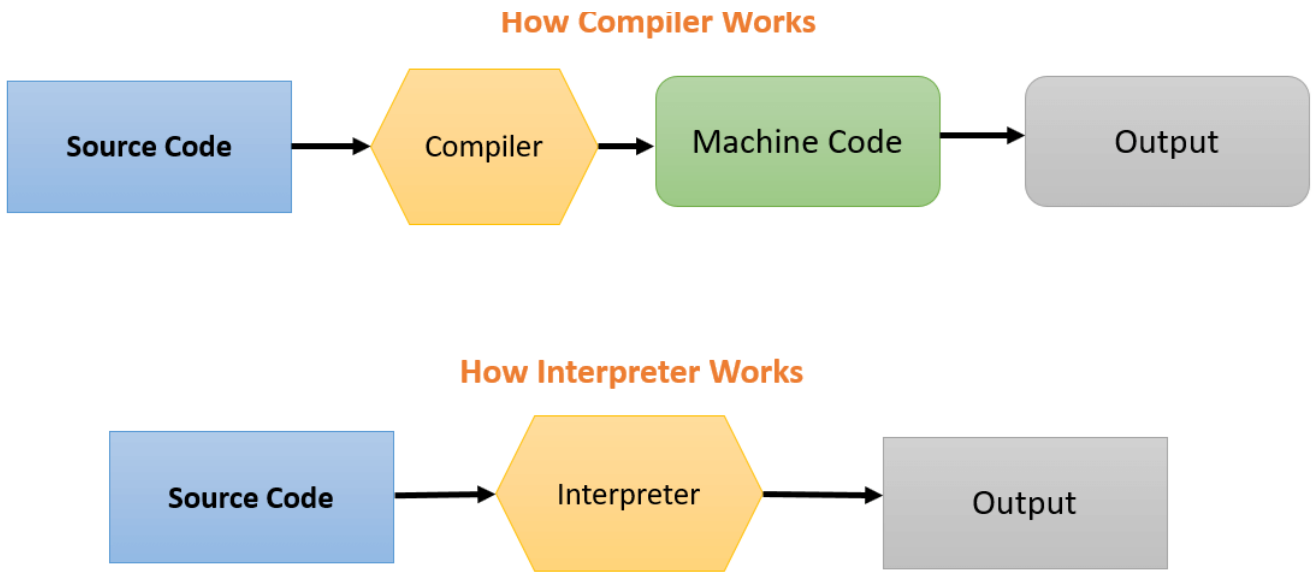


compile vs. interpret [7]

Runtime Engine Architecture – Gameplay Foundation

Gameplay foundation systems

- Scripting system
  - Many game engines employ a scripting language in order to make development of game-specific gameplay rules and content easier and more rapid.
  - Without a scripting language, you must recompile and relink your game executable every time a change is made to the logic or data structures used in the engine.
  - But when a scripting language is integrated into your engine, changes to game logic and data can be made by modifying and reloading the script code.



compile vs. interpret [7]

Runtime Engine Architecture – Gameplay Foundation

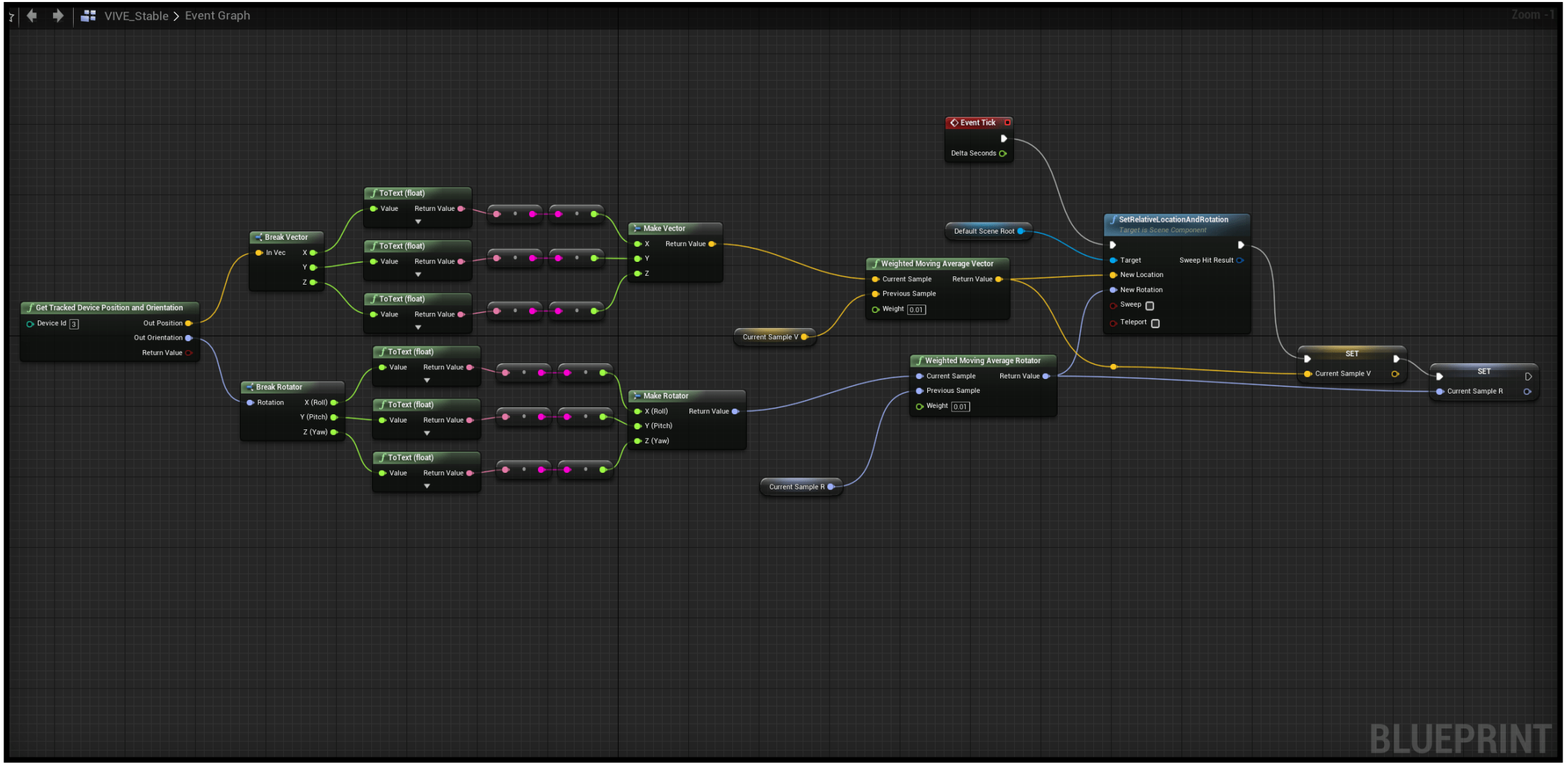
Gameplay foundation systems

- Blueprints Visual Scripting system of Unreal Engine
  - Unreal Engine provides the blueprints visual scripting system based on the concept of using a node-based interface to create gameplay elements from within Unreal Editor.
  - This is used to define object-oriented (OO) classes or objects in the engine.
  - Blueprints provide a visual approach to a scripting language and a execution flow.

Runtime Engine Architecture – Gameplay Foundation

Gameplay foundation systems

- Blueprints Visual Scripting system of Unreal Engine



Runtime Engine Architecture – Gameplay Foundation

Gameplay foundation systems

- Artificial Intelligence Foundations
  - Artificial intelligence has fallen squarely into the realm of game-specific software (it was usually not considered part of the game engine per se).
  - A company called *Kynogon* developed a middleware SDK named *Kynapse*, which provided much of the low-level technology required to build commercially viable game AI.
  - This technology was purchased by *Autodesk* and has been superseded by a totally redesigned AI middleware package called *Gameware Navigation*, designed by the same engineering team that invented *Kynapse*.

Runtime Engine Architecture – Gameplay Foundation

Gameplay foundation systems

- Artificial Intelligence Foundations
  - This SDK provides low-level AI building blocks:
    - Nav mesh generation
    - Path finding
    - Static and dynamic object avoidance
    - Identification of vulnerabilities within a play space
    - Well-defined interface between AI and animation
    - ...

Runtime Engine Architecture – Gameplay Foundation

Gameplay foundation systems

- Artificial Intelligence Foundations

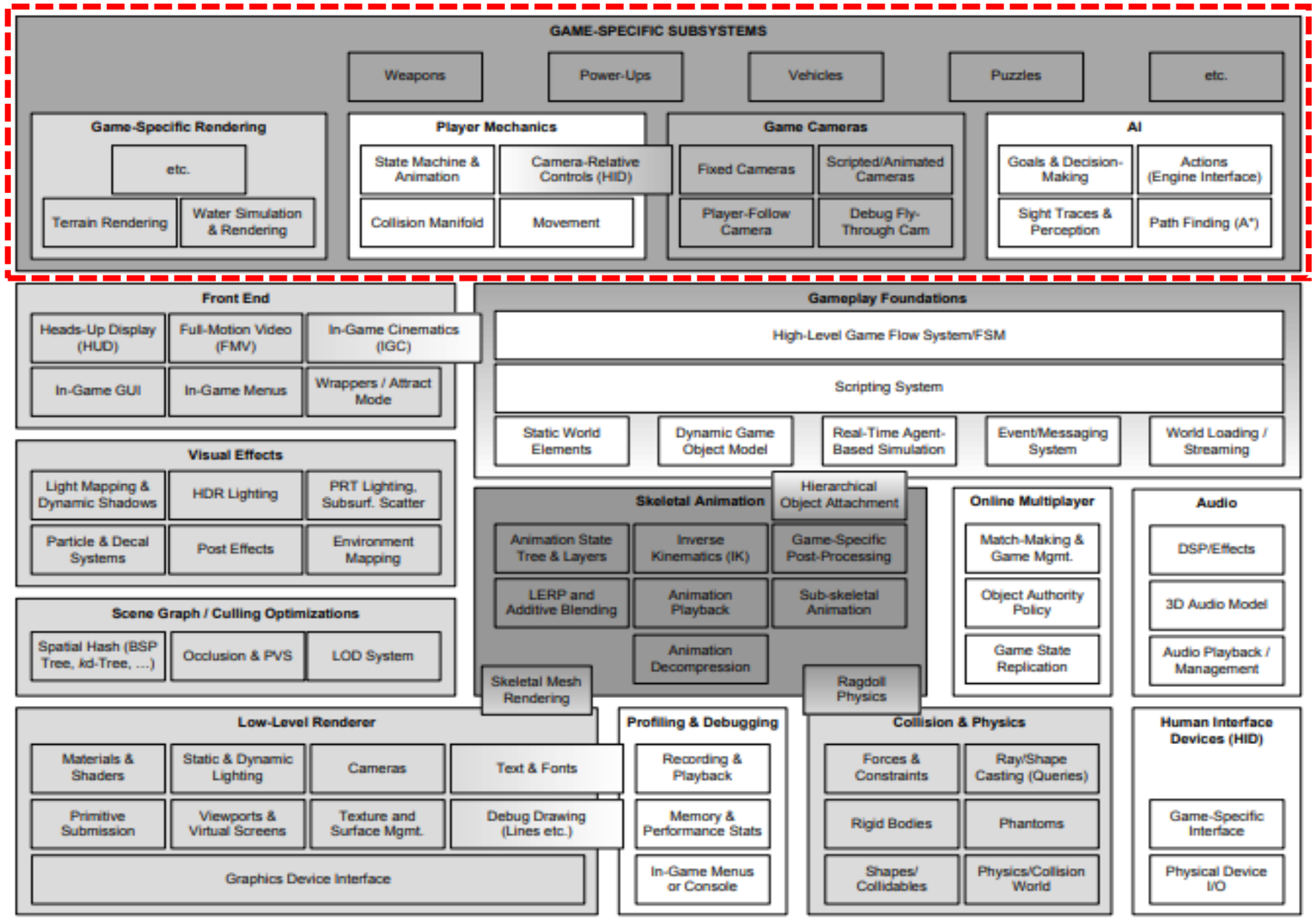
Runtime Engine Architecture – Gameplay Foundation

Gameplay foundation systems

- Artificial Intelligence Foundations

navigation mesh & path finding[8]

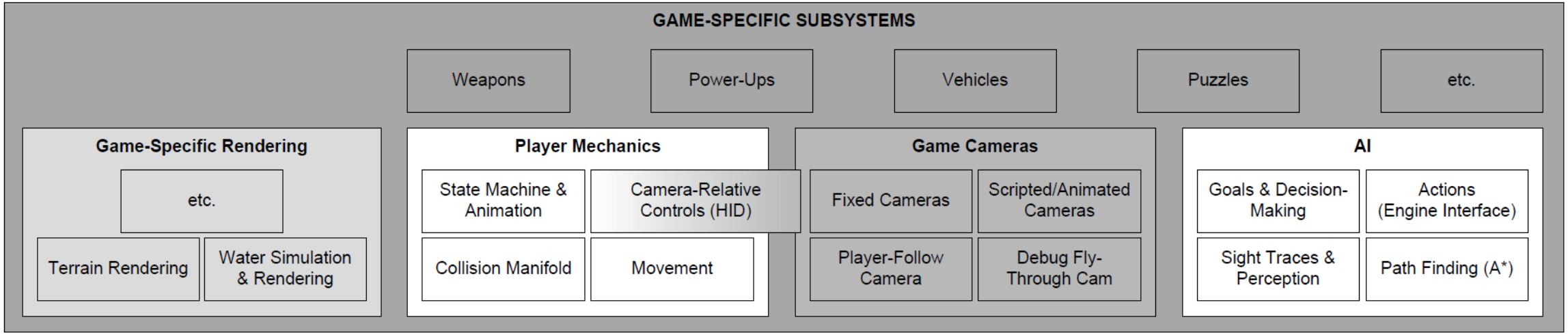
Runtime Engine Architecture - Subsystems



Runtime Engine Architecture - Subsystems

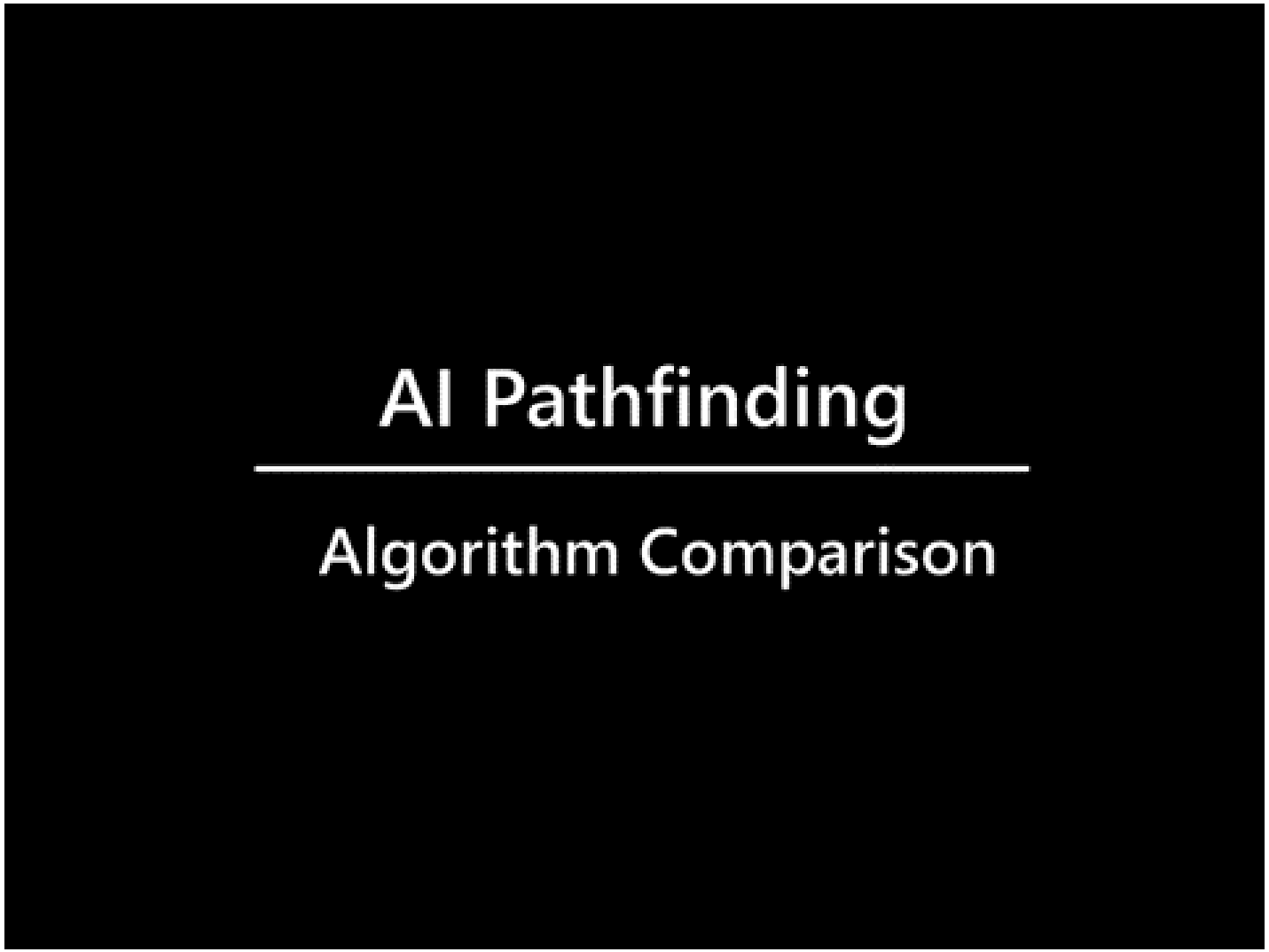
Game-specific Subsystems

- On top of the gameplay foundation layer and the other low-level engine components, gameplay programmers and designers cooperate to implement the features of the game itself.



Runtime Engine Architecture - Subsystems

Game-specific Subsystems



A* pathfinding[path]

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