Definition of Engine

The term **game engine** implies a whole ***complex of application software modules that provide graphic visualization, sound, movement of objects and characters, their actions in accordance with scripts, as well as network interaction, compliance with physical effects and laws, and much more. Simply put, these modules bring three-dimensional worlds to life***.

The term first appeared in the mid-nineties of the last century. It was associated with games in the first-person shooter genre, and more specifically with *Doom*, the most popular one at that time. The source code of *Doom* was well-thought-out, with clearly marked out main components: 3D graphics system, sounds, collision calculations, scripts, etc. Thus, instead of writing their own code, other programmers re-used the *Doom* creators work: they made some changes in the code, changed graphics and appearance of weapons, designed new levels, adjusted the rules and released new games with the same code from *Doom* as the basis.

The engine of a game is its core, the basic software on which all other components of the game are built. It's software code that can be used to create variations of the game, add-ons to it, or even other games with their own worlds.

In the 1990s, more game engines with free access began to appear. These engines allowed both third-party developers and common users to try writing their own games. As time passed, engines became more and more advanced and voluminous in terms of program code.

Using a ready-made game engine saves developers time by providing typical code elements that can be used to build a base for the game. This allows developers to focus on implementing ideas, developing graphics and sound, refining mechanics, and adding new features.

Thus, from a developer's point of view, an engine is a software platform on which applications can be developed, not just games. Engines are actively used in the development of applications with virtual and augmented reality, including guides, reference books, encyclopedias, and more.

Typical project structure

When you start developing a 3D application, you create a project. A project contains **content** (models, textures, scenes, settings, sounds, etc.), program code or **logic** (what breathes life into the content and makes it perform the task at hand), and **metadata** about the application (target platform, architecture, programming language used, and other additional information).

Main stages of project development

The general workflow for developing 3D applications such as computer games, simulators, trainers, and other virtual environments is as follows:

1. Collecting and analyzing data about the objects and processes of the system and requirements.
2. Creating a technical assignment or design document based on the collected data. This document provides a detailed description of the application to be used in the development process (concept, scheme, interface, graphics, virtual environment, and other relevant details).

The iterative cycle, with stages 3-4 executed in parallel:

3. Creating content (3D models, textures, animations, assembling and customizing 3D scenes, materials, lighting, physics).
4. Implementing the application logic (writing and debugging code).
5. At the final development stages, testing of the project is performed along with improvements and performance optimization.
6. Final build and release.

During this course, we will discuss the main stages of 3D application development in more detail as we work on our own projects.