

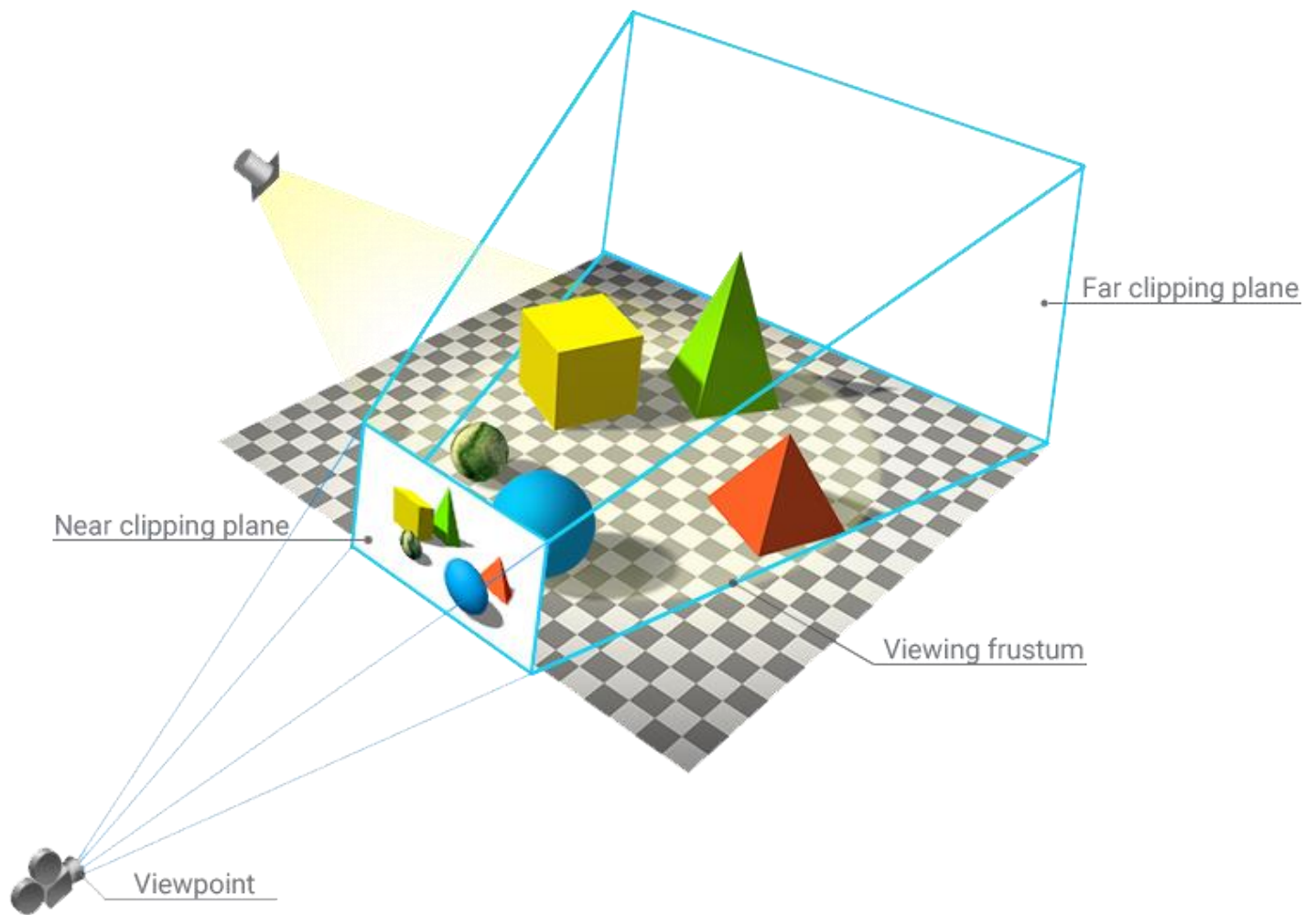
Cameras

In This Article:

- 1.
- 2.
- 3.

To see the virtual environment, we require virtual "eyes" — cameras. Cameras actually define what will be visible in the final image, the angle and distance from which it will be viewed. Virtual cameras come with a number of parameters, including the angle of view (FOV), near and far clipping planes, and the camera position and orientation. By combining the clipping planes with the camera's view angle, a visibility pyramid (or *frustum*) is created, and only objects within this pyramid are rendered as they are within the field of view. Objects outside this pyramid are not rendered as they are out of sight.

For visual representation, UNIGINE uses a standard perspective projection by default. If necessary, you can switch to the orthogonal projection (it can be used for 2D and isometric visualization). You can also set the life-like camera properties such as focal length and frame size (field of view is calculated automatically).



The process of displaying the virtual world involves the following three entities:

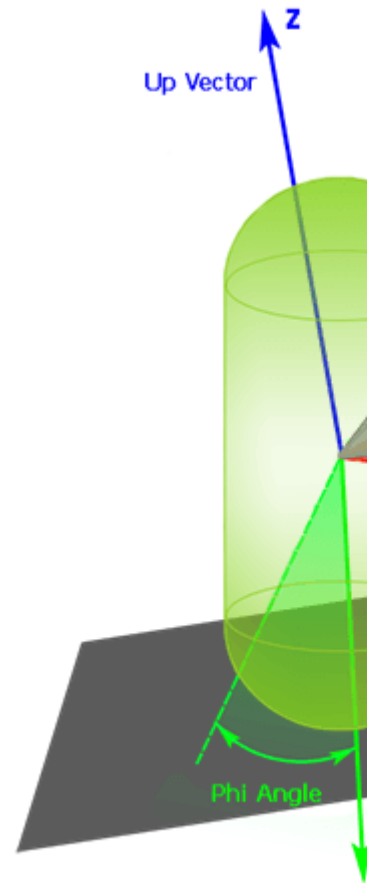
- **Camera**, a structure containing two matrices: view and projection. This structure allows setting the camera parameters: FOV, near and far clipping planes, and various masks and post-processing materials. The camera then passes the parameters to the viewport that draws the image that is "seen" by the camera. The camera is bound to the character that controls the camera's position.
- **Viewport** receives the camera parameters and renders the received image on the screen. In addition, it provides all functions of the main renderer, for example cube maps rendering, stereo rendering, and panoramic rendering.
- **Player** is a node controlled via the input devices (keyboard, mouse, joystick). It has a camera assigned. Once a player has changed its position, its internal camera's view matrix is changed as well.

Camera Types

UNIGINE features several types of players that can be used for various reasons (they have additional parameters depending on their intended use):

Player Dummy is a simple viewport into the world that has no physical properties and cannot collide with objects.

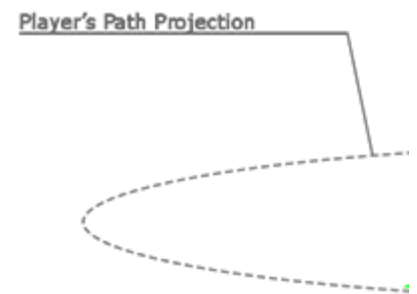
Player Actor is a player with a physical body (having mass, etc.) approximated by a capsule that can only walk on the ground (it can be used in shooters, for example).



Player Persecutor is a free flying camera without a physical body that follows the target node at a specified distance (can be used, for example, for a third-person view in racing). It can collide with other objects, but cannot push or interact with them.

An

Ta




Player Spectator is a free flying camera without a physical body used to create a spectator mode. It can collide with objects but cannot push or interact with them.

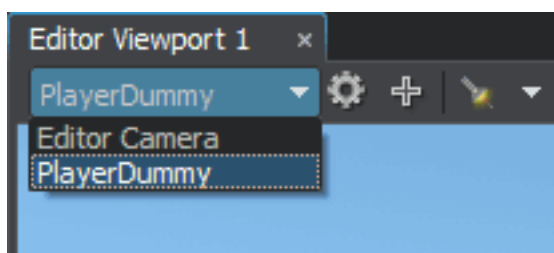


Adding and Configuring a Camera in the Editor

You can have as many cameras as you want in a scene. Moving them around, switching between them, and using various post-processing effects can solve almost any cinematic task.

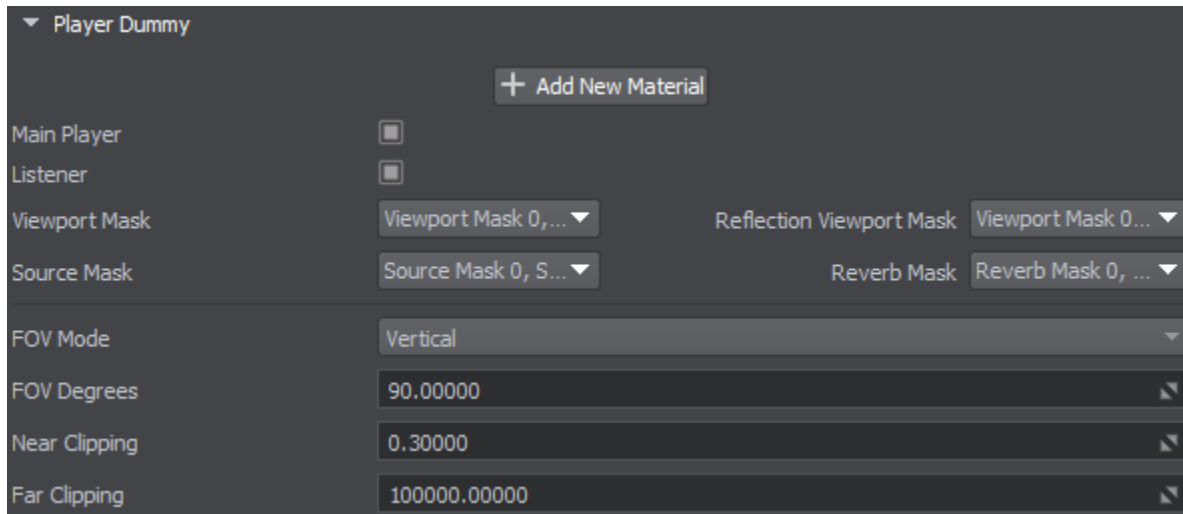
To add a new camera to the current scene, do the following:

1. Orient the currently used camera as required and configure its parameters (its position, rotation, and settings will be copied to a new camera).
2. In the *Camera* panel, click . A new PlayerDummy camera with the specified settings will be added to the list of available cameras.



3. Adjust the camera settings, if required.

You can also add a new camera via the *Create* menu (*Create* → *Camera* → Required camera type) and then set the desired settings in the *Parameters* window.



You can define for any camera whether it will be used for listening to sounds (*Listener*) and whether it will be set by default when the application is launched (*Main Player*). You can also define a set of masks that allow you to additionally control what will be heard and seen through this camera and what will be ignored (sound sources, reverberation, objects, and reflections). We'll deal with masks in more detail later.

Additional Camera Effects

In addition to the general settings, different post-materials can be assigned to cameras (for example, to simulate a color light filter or thermal vision), including custom ones. Moreover, you can control various standard camera effects such as depth of field, lens contamination, flare, and more via the *Settings* window.