In the Camera Prefabs folder are a few Camera Rigs, explained below.

To use them, simply drop one into your scene, and set the target field on the root of the camera rig.

If your player is tagged "Player", you don't even have to assign the target, as by default the cameras will automatically target the first object they find tagged "Player". You can uncheck this option if you don't want it.

After adding one to your scene, you will also probably want to delete the default "Main Camera" that was in your scene already, since the camera rigs contain their own cameras.

The "Free Look" camera is designed to follow a target's position, while allowing the user to rotate the angle of the camera with the mouse (or a touch gesture).

The "Multipurpose" camera is designed to follow a target's position and rotation (or direction of movement) and is useful for many game situations.

With the "Free Look" and "Multipurpose" camera, the rig is designed so the root object of the rig should always move towards the target's position. The camera offset is specified by the height (the Y value) of the "Pivot" object, and the forward offset of the final "Camera" object.

A typical set up for this would be to have the Y positional value of the Pivot set to 2 (for 2 meters above the target), and the Z positional value of the Camera set to -6 (for 6 meters behind the pivot).

If you wanted to adjust the camera to be to one side of the target (eg, for a closer over-shoulder third person view), you should adjust the X position value of the Pivot object.

Camera Rig <- position will move towards target.

Pivot <- adjust Y position for height, X position for horizontal offset

Camera <- adjust Z position for distance away from target

Both the "Free Look" and the "Multipurpose" camera also use the ProtectCameraFromWallClip script, which is intended to stop the camera from passing through items of scenery, as can happen if a character or vehicle targeted by the camera is backed up against a wall.

The ProtectCameraFromWallClip script examines the distance between the Pivot and the Camera, and attempts to preserve this where possible, but draws the camera in closer to the pivot when colliders are detected. For this reason, if you're using the wall clip script, you can't modify the local offset of the Camera at runtime, because it's being set every frame by the clip protection script.

The "CCTV Camera" is a little different, being a single GameObject with no hierarchy. Since it doesn't move to follow a target (it only rotates), it doesn't need the wall clip script. Assigning the target works exactly the same as the other Cameras however, and it will also auto target any object tagged "player" unless told otherwise.

The CCTV camera uses the "LookAtTarget" script, which provides functionality above and beyond Unity's basic "Transform.LookAt" function. It works in local space relative to the object's parent, and allows constraints on the look angle which work relative to the object's starting rotation. Possible uses could be: A view out of the side window of a moving car, A turret which should aim at a target from a moving spaceship, or - as it's used in this case - a CCTV camera which can be placed anywhere and can pan towards a target.