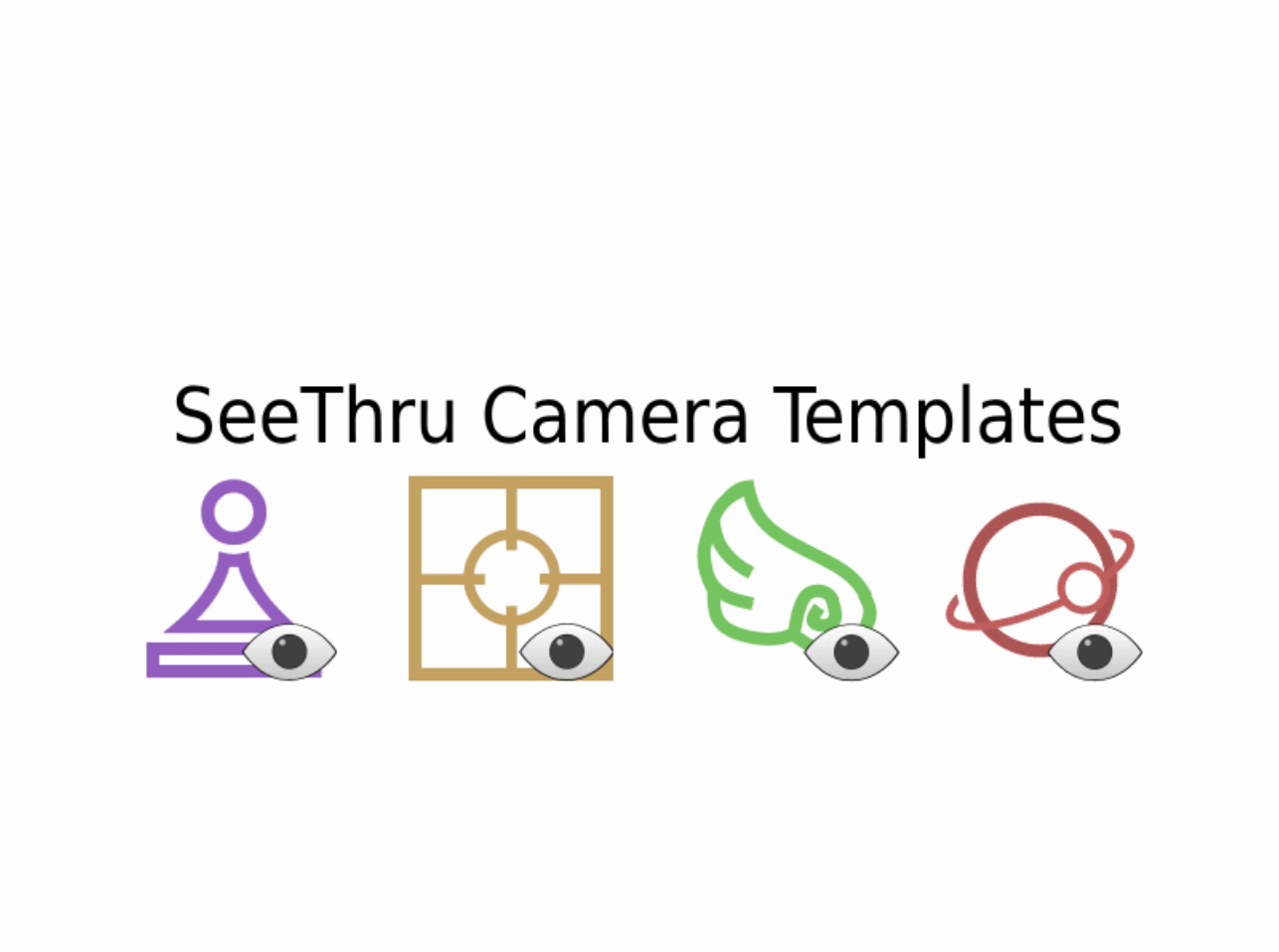

SeeThru Camera Templates

Table of Contents

Introduction

Welcome to the SeeThru Camera Template documentation. Here you will find information about the purpose of the templates as well as documentation for the separate cameras.

Wiki: https://github.com/dynamicrealities/seethru-documentation/wiki

Homepage: http://www.dynamicrealities.net

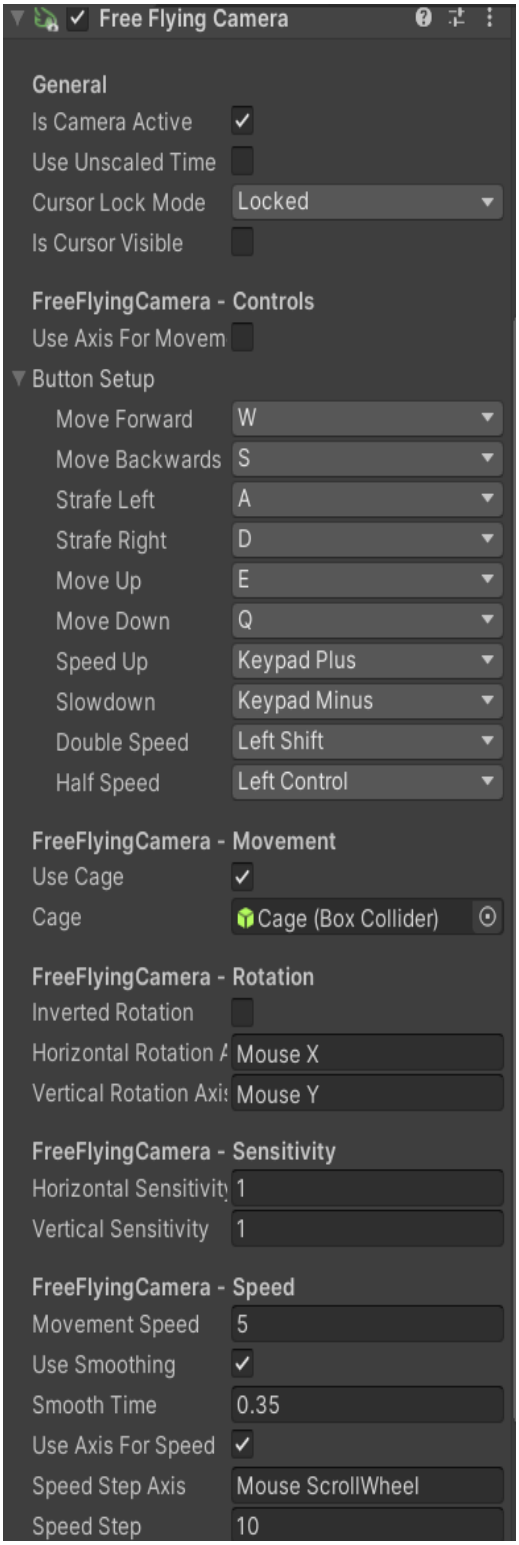
Setup

The setup is easy! Every SeeThru component can be placed on any GameObject which will then automatically include a camera component as well. It will then just work out of the box!

However, as these are camera templates, a starting-point, it's highly encouraged to jump into the code and extend/modify the components so they fit your game.

Free Flying Camera

The Free Flying Camera is designed to be the type of camera you'd use for level editors, spectator views and similar needs. The camera can be controlled freely in 3D space and also limited to only work within a specific area as well.



Settings

All of the SeeThru Cameras come with some general settings that are always located at the top. These include:

•IsCameraActive: Whether the camera should be affected by input or not.

•UseUnscaledTime: Whether to use unscaled DeltaTime or not. All of the cameras make use of Unity's `Time.deltaTime` variable to tie Camera movement to the framerate. Setting this to true will allow for having the camera move independently of `Time.deltaTime` so it always moves at full speed.
•CursorLockMode - This variable decides how Unity will lock the mouse pointer to the screen (or not).

•IsCursorVisible - Whether the mouse cursor should be visible on screen or not.

Controls

In the controls section you will find the options related to how you control the camera.

•UseAxisForMovement: Setting this to true will change the view so that ButtonSetup is no longer visible and is replaced with two fields used for Unity axis names. This should allow for hooking up a controller or joypad and set up its Axis' in the Unity editor for use with the camera.

•ButtonSetup - If UseAxisForMovement is false, then this field will be visible. It lists all the buttons that can be used to control this camera.

Movement

In the movement section you will find the options related to how the camera moves.

•UseCage: If this is true, a new field will be shown called Cage. It takes a box collider as parameter. If a collider is provided then the camera can only ever move when it's inside of the cage. If it tries to move outside of the cage, it will be stopped before it can do so.

•Cage: This variable is visible if UseCage is set to true. It defines the bounds within which the camera can move if desired. Otherwise the camera can move wherever it wants without restrictions.

Rotation

In the rotation section you will find the options related to how the camera rotates.

•InvertedRotation: Setting this to true will invert the rotation values it receives.

•HorizontalRotationAxis: The axis read to rotate the camera horizontally.

•VerticalRotationAxis: The axis read to rotate the camera vertically.

Sensitivity

In the sensitivity section you will find the options related to the sensitivity values applied to the rotation axis.

•HorizontalSensitivity: The multiplication factor to use for the horizontal rotation sensitivity.

•VerticalSensitivty: The multiplication factor to use for the vertical rotation sensitivity.

Speed

In the speed section you will find the options related to how fast the camera moves.

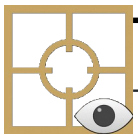
•MovementSpeed: How many unity units per second the camera moves.

•UseSmoothing: If this is set to true then the camera will smoothly decelerate to 0 once it stops receiving input. It also reveals the SmoothTime field.

•SmoothTime: If UseSmoothing is set to true, then this variable will decide how long it takes the camera to smoothly decelerate to 0 once input ceases.

•UseAxisForSpeed: If this is set to true then the camera will read an axis when the user decides to change their speed instead of reading the buttons provided in the ButtonsSetup. It also reveals the SpeedStepAxis field.

•SpeedStep: How many unity units per second the cameras speed is adjusted by when increasing/decreasing the speed.

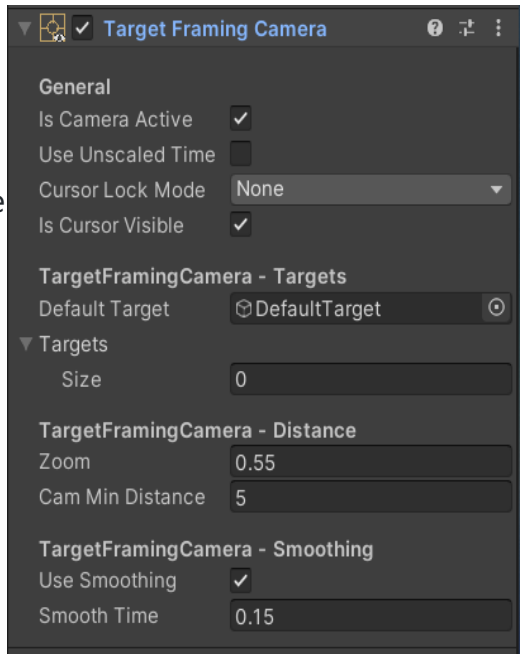


Target Framing Camera

The Target Framing Camera is designed to be the type of camera you'd use for 2.5D views, top-down games with multiple targets and similar needs. The camera will have a default position and will then try to fit in all of its targets within the camera frame while staying centered.

Settings

All of the SeeThru Cameras come with some general settings that are always located at the top. These include:



•IsCameraActive: Whether the camera should be affected by input or not.

•UseUnscaledTime: Whether to use unscaled DeltaTime or not. All of the cameras make use of Unity's `Time.deltaTime` variable to tie Camera movement to the framerate. Setting this to true will allow for having the camera move independently of `Time.deltaTime` so it always moves at full speed.

•CursorLockMode - This variable decides how Unity will lock the mouse pointer to the screen (or not).

•IsCursorVisible - Whether the mouse cursor should be visible on screen or not.

Targets

In the targets section you will find the options related to the targets that the Camera will look at.

•DefaultTarget: If the camera has no target to look at, it will look at this target.

•Targets: The list of targets for the camera to observe. Can either be set programmatically or manually through the drag-and-drop functionality of Unity.

Distance

In the distance section you will find the options related to the distance parameters for the camera.

•Zoom: The Zoom factor of the camera. The higher it is, the more the camera will zoom out.

•CamMinDistance: The minimum distance in Unity units the camera should stay away from its targets. This is especially useful for the games where the camera might only have a single target most of the time.

Smoothing

In the smoothing section you will find the options related to the cameras smoothing.

•UseSmoothing: If this is set to true then the camera will smoothly decelerate to 0 once it stops receiving input. It also reveals the SmoothTime field.

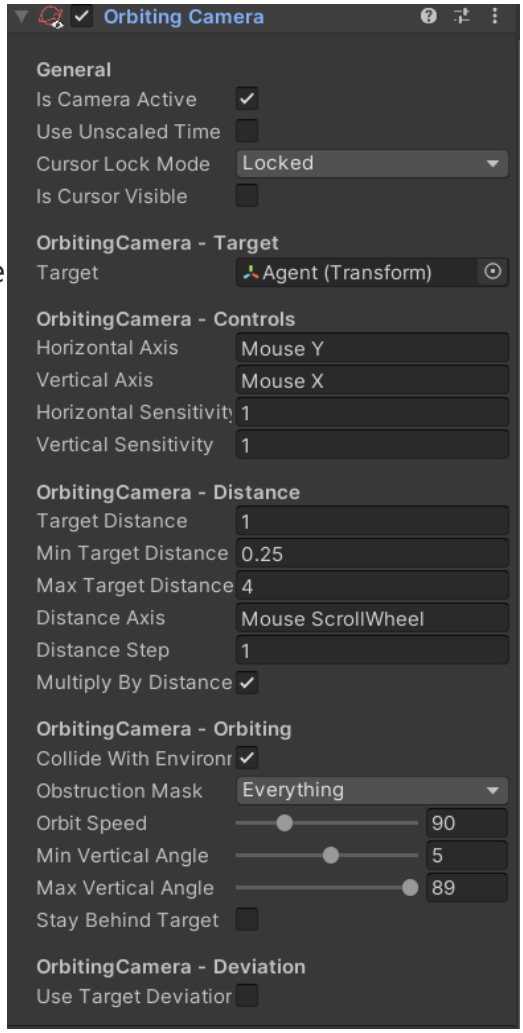
•SmoothTime: If UseSmoothing is set to true, then this variable will decide how long it takes the camera to smoothly decelerate to 0 once input ceases.

Orbiting Camera

The Orbiting Camera is designed to be the type of camera you'd use in tandem with character controllers, object trackers and similar needs. The camera will always have a target that it needs to follow and will then let the user rotate around the object, always keeping it centered.

Settings

All of the SeeThru Cameras come with some general settings that are always located at the top. These include:



- IsCameraActive: Whether the camera should be affected by input or not.
- UseUnscaledTime: Whether to use unscaled DeltaTime or not. All of the cameras make use of Unity's `Time.deltaTime` variable to tie Camera movement to the framerate. Setting this to true will allow for having the camera move independently of `Time.deltaTime` so it always moves at full speed.
- CursorLockMode - This variable decides how Unity will lock the mouse pointer to the screen (or not).
- IsCursorVisible - Whether the mouse cursor should be visible on screen or not.

Target

This section only contains a single field related to what the Camera is supposed to fixate on.

- Target: the target that the camera will fixate on. This is a required field.

Controls

In the controls section you will find the options related to how the camera controls.

•HorizontalAxis: What axis the camera will use for horizontal rotation.

•VerticalAxis: What axis the camera will use for vertical rotation.

•HorizontalSensitivity: How sensitive the horizontal rotation axis is.

•VerticalSensitivity: How sensitive the vertical rotation axis is.

Distance

In the distance section you will find the options related to how the camera distances itself from its target.

•TargetDistance: The cameras current distance in Unity units. The distance will always be clamped between MinTargetDistance and MaxTargetDistance.

•MinTargetDistance: The minimum distance in Unity units the camera can have to its target.

•MaxTargetdistance: The maximum distance in Unity units the Camera can have to its target.

•DistanceAxis: Which axis to read to change the cameras distance to the target.

•DistanceStep: How many Unity units the camera should move away from its target per step.

•MultiplyByDistanceAxis: If this is set to true, it means that the Distance step will be multiplied by the axis delta value, so that the distance can grow or shrink, relative to how big the axis delta is. This is especially useful for controller setups as joysticks have many degrees of values between 0 and 1 rather than digital input from a keyboard that has values 0 or 1.

Orbiting

In the orbiting section you will find the options related to how the camera orbits around its target.

•CollideWithEnvironment: If this is set to true, then the camera will attempt to change the distance it has to the target such that no geometry will ever between it and its target. Setting this to true will also reveal the ObstructionMask field.

•ObstructionMask: If CollideWithEnvironment is set to true this field is shown. It decides what collision layers the camera should check for collision.

•OrbitSpeed: How many degrees per second the camera can turn around its target.

•MinVerticalAngle: The minimum Vertical angle the camera can reach before it's stopped.

•MaxVerticalAngle: The maximum Vertical angle the camera can reach before it's stopped.

•StayBehindTarget: If this is set to true the camera will try to turn itself so that it is behind the target when it passes the camera. If it's false, the camera will stay fixed in one rotation and position at all times.

Deviation

In the deviation section you will find the options related to whether the camera will allow the target to be off-center or not.

•UseTargetDeviation: If this is set to true, it will allow the target to be slightly off-center before the camera will start following again. If it's set to false the camera will stay in a fixed position relative to target. Setting it to true also shows the fields

• TargetDeviation and TargetCentering.

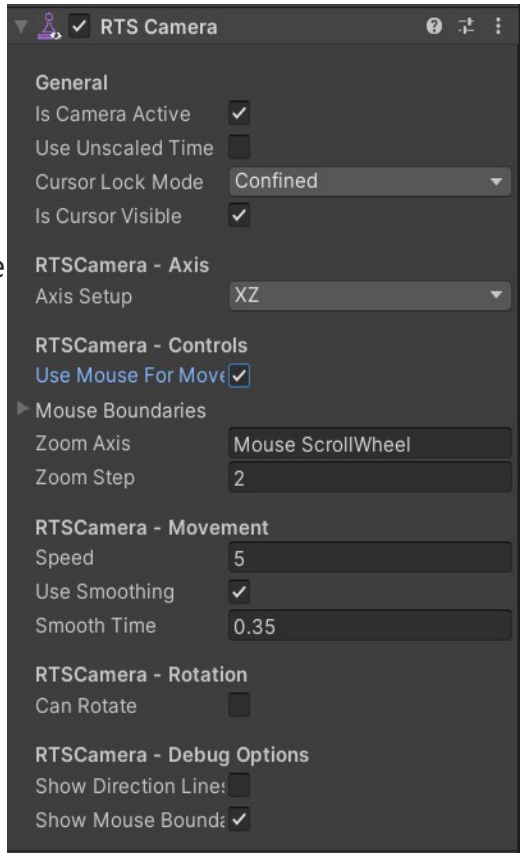
•TargetDeviation: This field is visible if UseTargetDeviation is set to true. It specifies how many Unity units the target can be off-center before the camera will start following the target again at a relatively fixed position.

•TargetCentering: This field is visible if UseTargetDeviation is set to true. The value specifies how far behind the camera can be relative to its target while following.

RTS Camera

The RTS Camera is designed to be the type of camera you'd use for top-down games with multiple targets, real-time strategy, adventure games and similar needs. The camera will have a default height and rotation and will then keep those constant while moving the camera along the chosen horizontal plane.

Settings



All of the SeeThru Cameras come with some general settings that are always located at the top. These include:

•IsCameraActive: Whether the camera should be affected by input or not.

•UseUnscaledTime: Whether to use unscaled DeltaTime or not. All of the cameras make use of Unity's `Time.deltaTime` variable to tie Camera movement to the framerate. Setting this to true will allow for having the camera move independently of `Time.deltaTime` so it always moves at full speed.

•CursorLockMode - This variable decides how Unity will lock the mouse pointer to the screen (or not).

•IsCursorVisible - Whether the mouse cursor should be visible on screen or not.

Axis

In the axis section you will find the options related to how the camera will calculate its horizontal plane.

•AxisSetup: This field decides what is considered the horizontal plane for the camera to move on. The two axis chosen will act as "forward" and "right" while the axis not

shown in the chosen axis setup will act as "up". This means that the horizontal plane can be defined arbitrarily if needed.

Controls

In the controls section you will find the options related to how the camera will be controlled.

•UseMouseForMovement: If this is set to true it will use the edges of the screen as trigger areas for the mouse to move into. If the mouse do move into these trigger areas, it'll make the screen move in that direction. If it's set to false, it will make use of buttons to achieve the same thing. When setting this to true it reveals the MouseBoundaries and ZoomAxis fields. When it's false it reveals the ButtonSetup field.

•ButtonSetup: If UseMouseForMovement is set to false this field is shown. It allows you to set the keys you want to use to control this camera, rather than using the mouse.

•ZoomStep: How far the camera will zoom in or out with each step.

Movement

In the movement section you will find the options related to how the camera will move.

•Speed: The cameras movement speed in Unity units per second.

•UseSmoothing: If this is set to true then the camera will smoothly decelerate to 0 once it stops receiving input. It also reveals the SmoothTime field.

•SmoothTime: If UseSmoothing is set to true, then this variable will decide how long it takes the camera to smoothly decelerate to 0 once input ceases.

Rotation

In the rotation section you will find the options related to how the camera will rotate, and if it even can rotate in the first place.

•CanRotate: If this is set to true then the camera can rotate around its up axis. Setting this to true will show the RotationStep, RotatateClockwise, RotateCounterClockwise and RotationLerpTime fields.

•RotationStep: How many degrees the camera will rotate around its up axis.

•RotateClockwise: What button to use to trigger a clockwise rotation.

•RotateCounterClockwise: What button to use to trigger a counter-clockwise rotation.

•RotationLerpTime: How long it should take the camera to do the rotation.

Debug Options

In the debug options section you will find the options that can whelp debug some of the RTS Camera settings.

•ShowMouseBoundaries: If UseMouseForMovement is set to true then this debug option is available. Setting it to true will draw a rectangle on screen which represents the trigger areas described in the Controls section of this wiki. The lines can be moved by adjusting the values in the MouseBoundaries field.

•ShowDirectionLines: Setting this to true will reveal four colored lines which showcases the computed horizontal plane of the camera. Red is forwards, Green is backwards, Red is Left and Yellow is Right.