

RTS Starter Kit

The Camera

Scene Configuration

In order to achieve scene configuration for the RTSCamera you need to add a “Scene Bounds” component. It’s job is to keep track of the scenery limits (boundaries) defined by the collider at “Bounds Collider”. Also here we have the “Dragging Collider” which is a collider with a mayor size that will be used as base for the clicks detection needed by our camera scroll feature.

Max distance it’s used by the raycast and it’s a value relative to the scenery size.

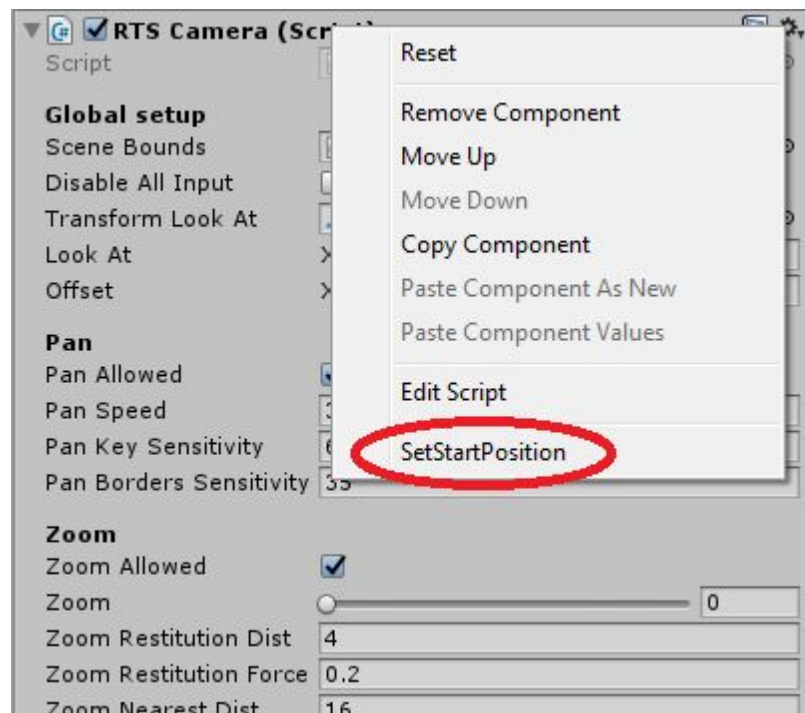
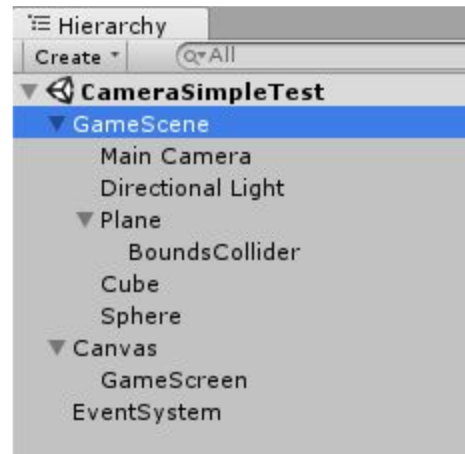
In our “CameraSimpleTest” scene. we find the “Scene Bounds” component at the root of the scene and as child we find the camera with the RTSCamera script and the planes that serves as floor to our game (with the respective boundaries collider).

Also we can find as child of the UI Canvas our “RTS Camera UI” component that serves as cursor screen limits detector and dragging for scroll detector.

Feel free to explore this scene to understand the simplicity of requirements to see the camera working.

Additionally, in the “CameraTerrainTest” scene we found a similar scene but with a unity’s terrain to see how you can get it to work in a so simple way.

Finally a hidden feature: Right click on the RTS Camera script header will bring the context menu with an additional option “SetStartPosition” that sets up the transform of the camera according with the current RTSCamera configuration.



RTSCamera

Here i'll try to cover the RTSCamera component. First of all must be attached next to the Unity's Camera component (as a requirement). At first, this component can be a little overwhelming, but each property allows a proper configuration according to your game scale and how much polish you want for the camera. Almost every property in the editor has a helpful tooltip that explains its behaviour, feel free to explore each one of them. So, in this document will only cover some special hints to know of the different sections of the component.

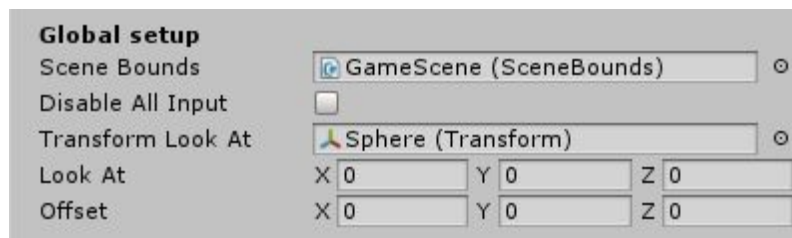
Global setup

At first we need to reference the "Scene Bounds" property to allow this system to know the boundaries of our game and the position of the plane treated as base for our scenery where to scroll our camera.

Disable All Input if you want to control the camera from any other means and don't want to be disturbed by the user input (for example when you run a cinematic).

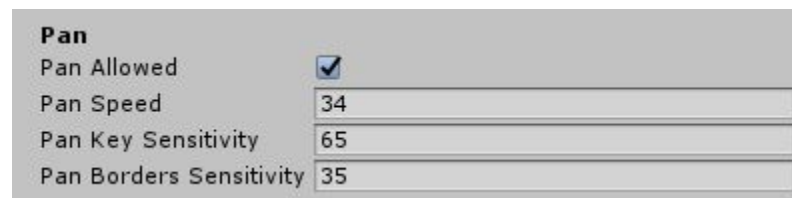
Transform Look At will be used as starting position for our camera placing the lookAt in that "transform.position" coordinates.

Offset can be used to set a permanent offset for the LookAt position, very helpful if you have units of different sizes, just change the y component of the offset to a value relative of your biggest unit in the scene.



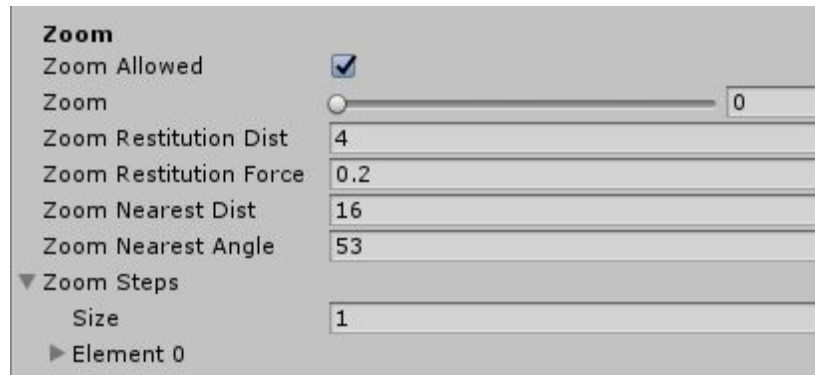
Pan

Enables or disables the input (uses "Horizontal" and "Vertical" axis defined in "Edit/Project Settings/Input"). How fast the camera moves from its current position to the position defined in lookAt is defined through "Pan Speed".



Zoom

Enables or disables the input (uses “Mouse ScrollWheel” axis defined in “Edit/Project Settings/Input”). Set Zoom steps to Zero if you don’t want any zoom at all.



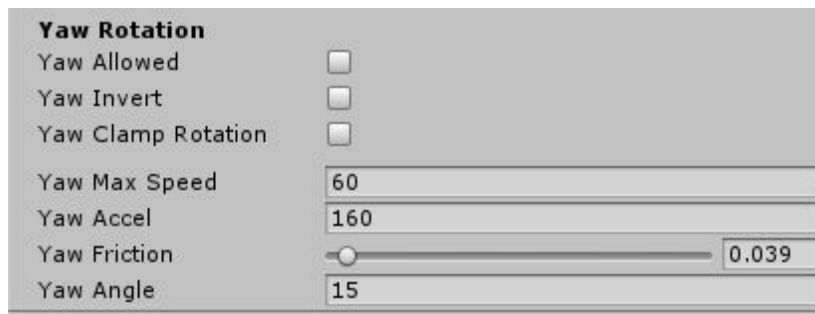
The screenshot shows the 'Zoom' settings panel. It includes a 'Zoom Allowed' checkbox which is checked. Below it is a 'Zoom' slider set to 0. Other settings include 'Zoom Restitution Dist' (4), 'Zoom Restitution Force' (0.2), 'Zoom Nearest Dist' (16), 'Zoom Nearest Angle' (53), 'Zoom Steps' (expanded to show 'Size' at 1 and 'Element 0' at 0).

Zoom	
Zoom Allowed	<input checked="" type="checkbox"/>
Zoom	0
Zoom Restitution Dist	4
Zoom Restitution Force	0.2
Zoom Nearest Dist	16
Zoom Nearest Angle	53
Zoom Steps	
Size	1
Element 0	0

Yaw Rotation

Enables or disables the yaw rotation input (uses “CameraYaw” axis defined in “Edit/Project Settings/Input”). Able to perform clamp limits with a smooth rubberbanding effect (enable “Yaw Clamp Rotation” to show additional related options).

Also it’s possible to change speed, acceleration and friction for this rotation.



The screenshot shows the 'Yaw Rotation' settings panel. It includes checkboxes for 'Yaw Allowed', 'Yaw Invert', and 'Yaw Clamp Rotation', all of which are unchecked. Below these are input fields for 'Yaw Max Speed' (60), 'Yaw Accel' (160), 'Yaw Friction' (0.039, shown as a slider), and 'Yaw Angle' (15).

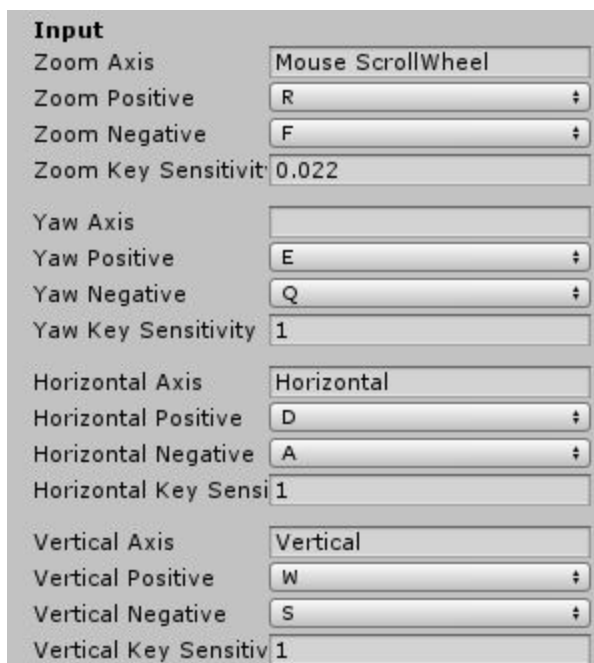
Yaw Rotation	
Yaw Allowed	<input type="checkbox"/>
Yaw Invert	<input type="checkbox"/>
Yaw Clamp Rotation	<input type="checkbox"/>
Yaw Max Speed	60
Yaw Accel	160
Yaw Friction	0.039
Yaw Angle	15

Input

Here it’s the input configuration for all the previous modules.

The component uses the axis name to take the input for the different modules. In case it’s not defined (As it is in YawAxis) it will take the positive and negative keys to control the module (applying the sensitivity factor).

The sensitivity factor only will be used for the positive and negative key codes if they are used. If you want to change the sensitivity for the axis input you must go to the Input settings in the unity editor (“Edit/Project Settings/Input”).

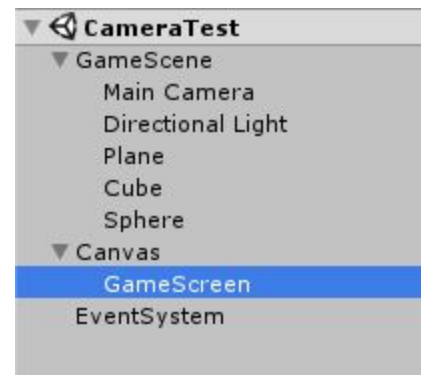


The screenshot shows the 'Input' settings panel. It is organized into sections for Zoom, Yaw, Horizontal, and Vertical axes. Each section has fields for the axis name, positive/negative keys, and key sensitivity.

Input	
Zoom Axis	Mouse ScrollWheel
Zoom Positive	R
Zoom Negative	F
Zoom Key Sensitivity	0.022
Yaw Axis	
Yaw Positive	E
Yaw Negative	Q
Yaw Key Sensitivity	1
Horizontal Axis	Horizontal
Horizontal Positive	D
Horizontal Negative	A
Horizontal Key Sensitivity	1
Vertical Axis	Vertical
Vertical Positive	W
Vertical Negative	S
Vertical Key Sensitivity	1

UI layer

Keeps track of the user input (assuring that only works when mouse cursor are in correct focus of the game screen area) and derive that actions to the RTSCamera system. In this case the job of the RTS Camera UI is to listen to the mouse drag operation with middle button to perform a camera pan effect. And also keeps track of the cursor position to detect when it's at the borders of the screen to achieve the second kind of camera pan (used in all the real time strategy games).

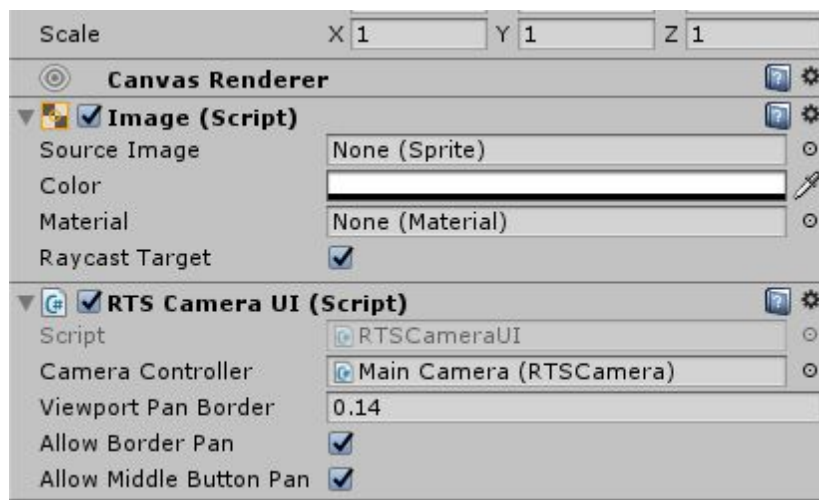


Some basic setup

Below the Canvas we have the panel that controls the focus over our game screen. Here we have two components, an Image (which has the job of send all events related to the cursor drag operation that will be intercepted by the RTSCameraUI component) The second one is the RTSCameraUI component. Here we can setup the viewport pan border that is a factor of the minor value between the current screen size. Meaning, it will be used in the next formula to calculate the size of the borders:

$$\text{border} = \text{Mathf.Min}(\text{Screen.width}, \text{Screen.height}) * \text{viewportPanBorder}$$

With the other two properties we can enable or disable the both main features.



(Components attached to the "GameScreen" GameObject)