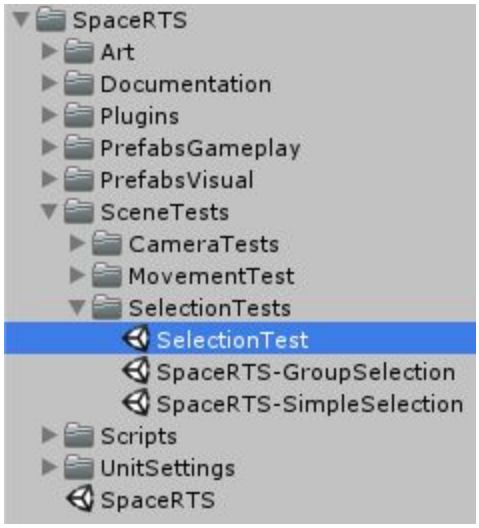
RTS Starter Kit

Units Selection

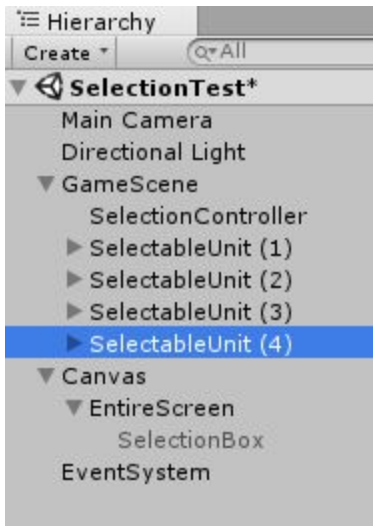
Introduction



In this document, we are going to explain how it's builded, how to configure and how are the communication between the Selection system, the UI to handle the input and the units that we need to be selectables.

In order to do the explanation more visual and simple we are going to analyze the scene called "SelectionTest" (included in the package).

The scene has the minimal GameObjects and components to serve as tutorial for selecting units without any special criteria more that doing the selection with a simple left click from mouse pointer or dragging a selection box to select multiple units.

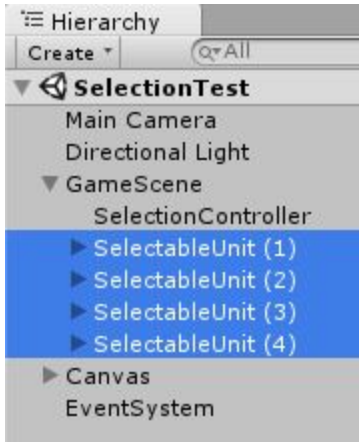


It's also possible to select multiple units by holding the left ctrl key and clicking the units with left click. (like any traditional rts like game). To unselect the units simple do a left click over an empty space.

The scene it's configured with a simple camera without any special behaviour. A "GameScene" holding the GameScene component serving as central point to the whole scene. Inside we find some childs. The first one is the Selection Controller that holds the SelectionSystem and Selection Input components (Explained both some sections below in this document) and the next GameObjects (called SelectableUnit (x)) holds the units that we want to be selectables.

Finally we have the canvas, holding a simple transparent image extended in the entire screen to handle the ui input (clicks and selection drag) that later will be derived to the SelectionInput and finally the SelectionSystem.

Units Configuration

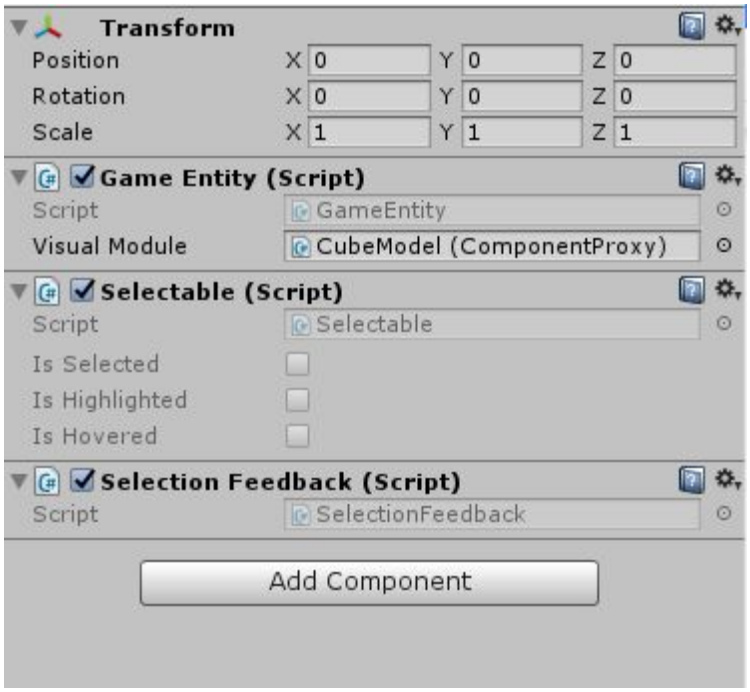


Here we cover how to correctly setup the units for achieve its hover, highlight and selection. in our “SelectionTest” scene we can find out four units able to be selected (highlighted in the image from the right). if we select one then we can see how it’s configured.

Of course, first of all we have the GameEntity component which defines our VisualModule and also required for the correct configuration of our game unit. but next we have two components that matters to us. The “Selectable” component and the “Selection Feedback” component.

Selectable

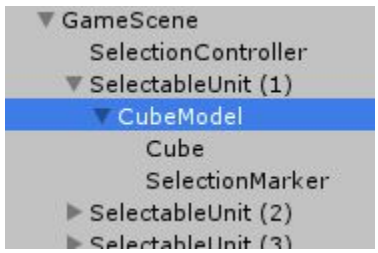
Stores the status of our units if they are selected, hovered and highlighted, but also informs (dispatchs a message) through the MessageSink in our GameEntity to anyone that want to hear about any changes to any of that statuses. (in this case will be heard it from the “SelectionFeedback” component.



Selection Feedback

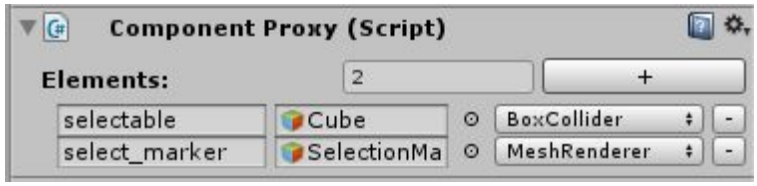
Registers it as observer of the message “SelectableChanged” and proceed to give some feedback in the visual module after any change in the selection and highlight status in the Selectable Component.

Visual Module

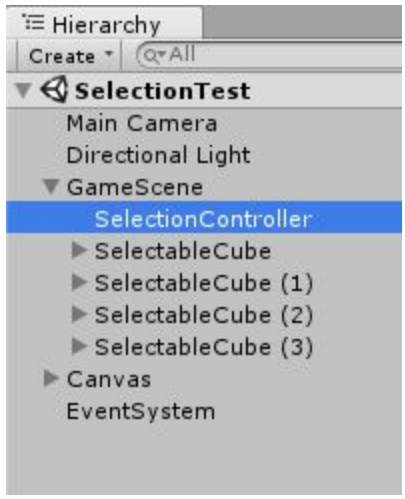


Another important subject to achieve a proper configuration is the visual module of the unit located as child of our GameEntity and having in its root the “ComponentProxy” component.
Here we have two important elements defined: the “selectable” which is of type Collider and will be used by the Selectable component to define the bounding area to click and select the unit and the “select_marker” element that is a

MeshRenderer in this case and will be used by the “SelectionFeedback” component to change visually the selection marker.



Selection Controllers



We have two components that allows the units selection: The *SelectionSystem* and the *SelectionInput.* Both components can be located in our SelectionTest scene as a child of the GameScene GameObject in a GameObject called “SelectionController” (better explained in the right image).

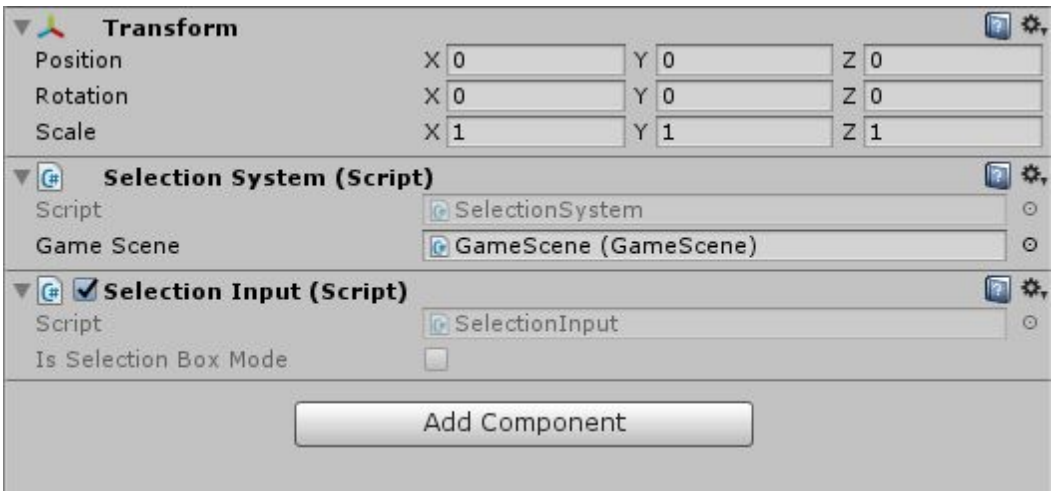
Selection System

Does the most basic job: keeps track of the currently selected, hovered and highlighted Selectables in our scene. What units wants to be selected? doesn’t care to it. That’s the job of another. This components just keeps track, and can Select, Hover, or Highlight any Selectable in out Scene.

Requires a reference to the GameScene but generally will be automatically setted up if it’s located as a child of it.

Selection Input

Is the nexus between the ui input and the SelectionSystem. Requires to be attached with a SelectionSystem component. Process the input provided (clicks, selection box, etc) and determines which Selectables now needs to be hovered, highlighted and selected. Once that Selectables are established, calls the correct methods in the SelectionSystem to do the magic.



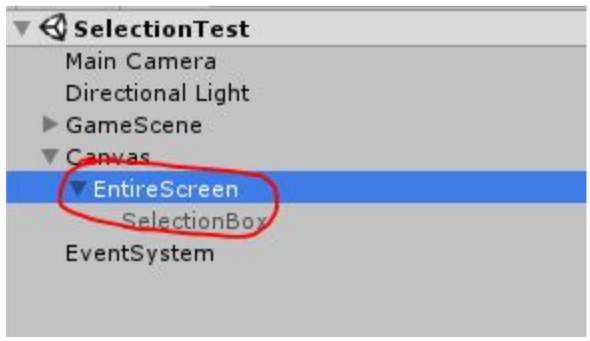
(Here is how it looks the SelectionController GameObject)

UI layer

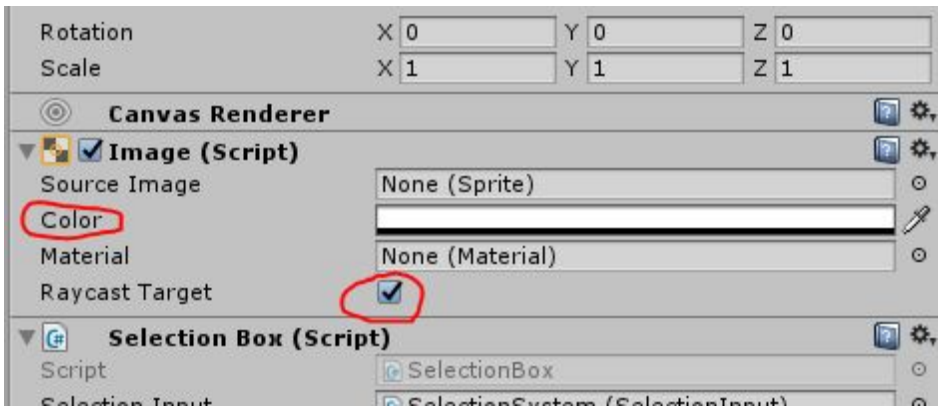
Keeps track of the user input (assuring that only works when in correct focus) and derive that actions to the selection input system (i'm talking about the clicks and selection box dragging operation).

Some basic setup

Below the Canvas we have the panel that controls the focus over our game screen. In this case, handles the selection box and clicks over the screen that later becomes in the unit selection.

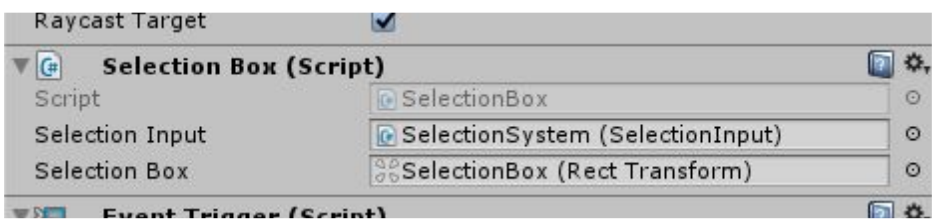


After selecting this screen panel we realize that is just an image with a color with alpha=0. Why is that? because the image allow us to raycast mouse pointer events. we are unable to hear the pointer events without the image component, and in order to avoid the entire white screen we need to set the alpha value to 0.



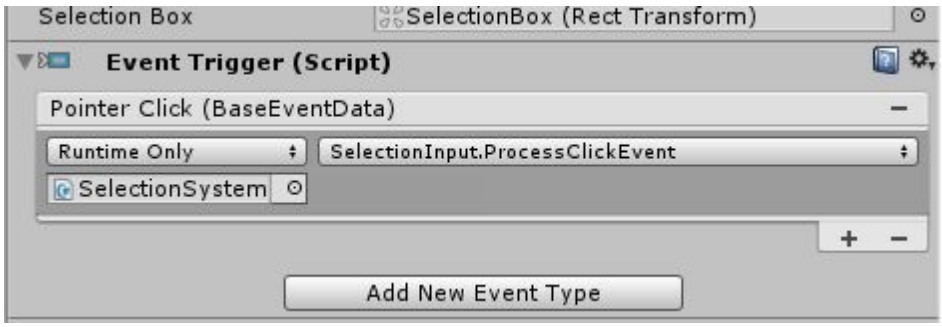
Selection Box

The SelectionBox Component handles all the operations related to the drag over the screen to select multiple units. Requires to setup two references, when the drag operation is detected we need to show the selection box feedback over the screen, this is achieved by changing the bounds of the public RectTransform field called "Selection Box". Also we need to inform to the selection input about this bounds coordinates in order to allow that system to calculate which GameEntities are inside to achieve that group selection. so, we need the reference to the "Selection Input" public field setted accordly.



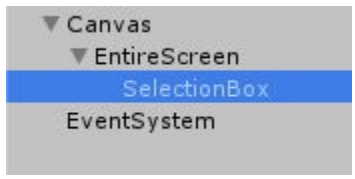
Selection Clicks

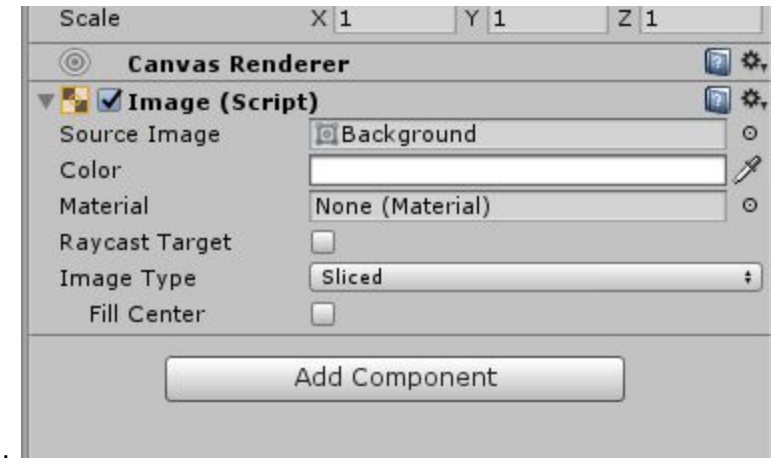
Another component you can find in the example is an EventTrigger component who has the job of derive the click event provided by the image component to the Selection Input, specifically to the method called ProcessClickEvent. Take into account that the SelectionInput doesn't "hear" any input at all by its own means, just listen from other sources the input, process it and translates it to the selection system.



Hint note

When i've been working in the scene i've experienced some problems with the input, for some reason the events aren't reaching my SelectionBox component and the clicks also aren't been triggered. After some digging i've managed to find the problem: all the pointer events are been captured by the selection box image (the one drawing the selection rectangle). so the solution was simple, removing the raycast target check the problem was solved (and makes sense, that image shouldn't been capturing any input at all)





(make sure to remove that raycast target check)