Insane Systems RTS Selection Asset Guide

Content

Features

- Simple to use and implement in your game, minimum code required
- Selection and unselection mechanics, suitable for RTS or other game genres, which have selection like this
- Single unit selection by click
- Multi-selection by holding LMB and dragging mouse
- Customizable UI multi-selection rectangle effect
- Extendible code with interfaces and events, which will help to minimize needed additional code to extend selection system
- Selection effect example included

Before you start

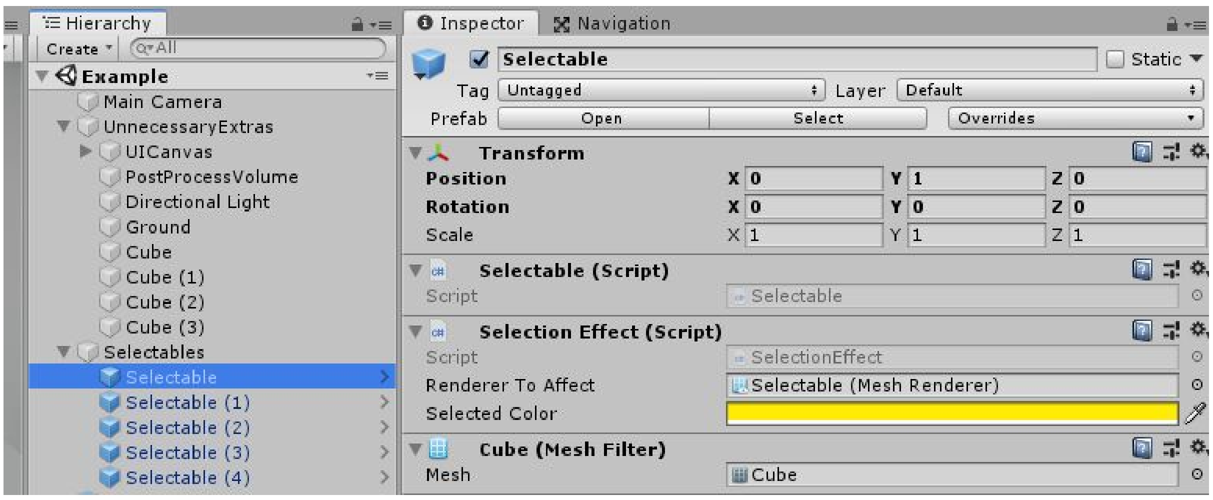
Asset works as it is, it doesn't require any code changes.

All you need to do - add **SelectionSystem** prefab to your scenes and add **Selectable** component to the objects you want to allow be selected by player, and after it all should work. If you need custom selection effect or add game logic, asset code is designed to allow you simply connect your scripts.

Read partitions below to get more info about it.

Quick Start

First of all, open **Example** scene and check one of **Selectables** objects and **SelectionSystem** prefab objects list. It will give you some info about all asset components.



Hierarchy
Create ▾ All
- Example
  - Main Camera
  - UnnecessaryExtras
    - UICanvas
    - PostProcessVolume
    - Directional Light
    - Ground
    - Cube
    - Cube (1)
    - Cube (2)
    - Cube (3)
  - Selectables
    - Selectable
    - Selectable (1)
    - Selectable (2)
    - Selectable (3)
    - Selectable (4)

Inspector | Navigation

✓ Selectable			Static
Tag	Untagged	Layer	Default
Prefab	Open	Select	Overrides

Transform

Position	X 0	Y 1	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Selectable (Script)

Script	Selectable

Selection Effect (Script)

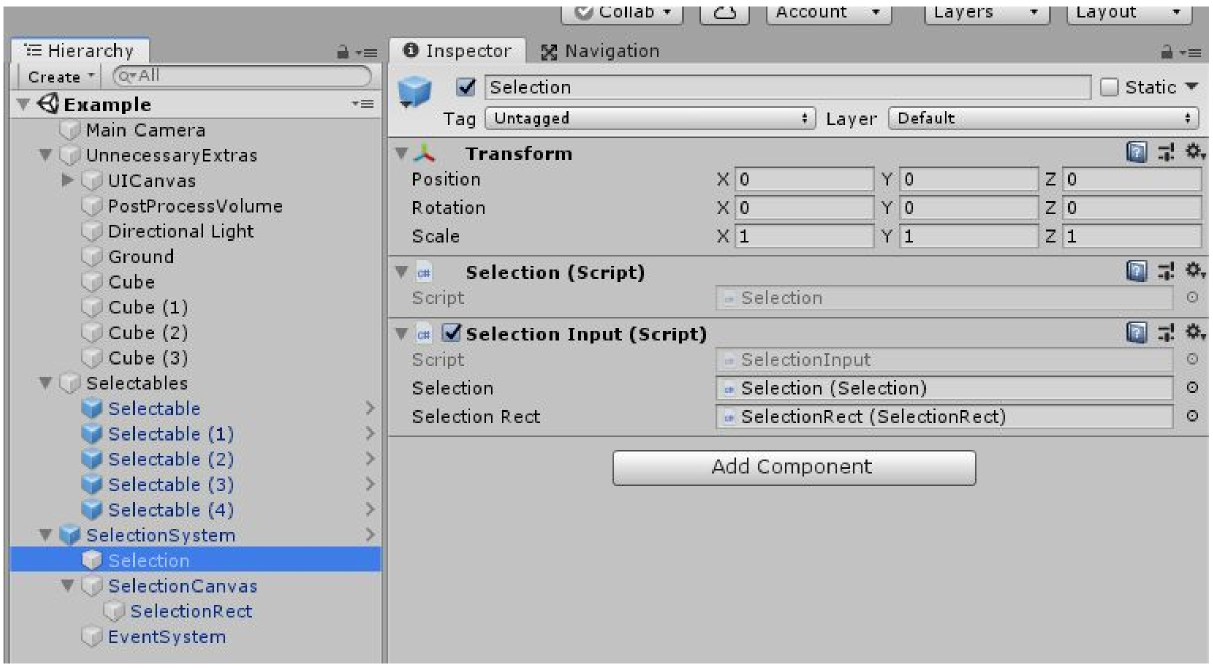
Script	SelectionEffect
Renderer To Affect	Selectable (Mesh Renderer)
Selected Color	🟨

Cube (Mesh Filter)

Mesh	Cube

Selectable component represents any object player can select. In RTS, add it to your units.

Selection Effect is example component, which will colorize selected unit. You can use it or make your own. If you remove it, nothing will broke. You can use it code to implement your own same way.

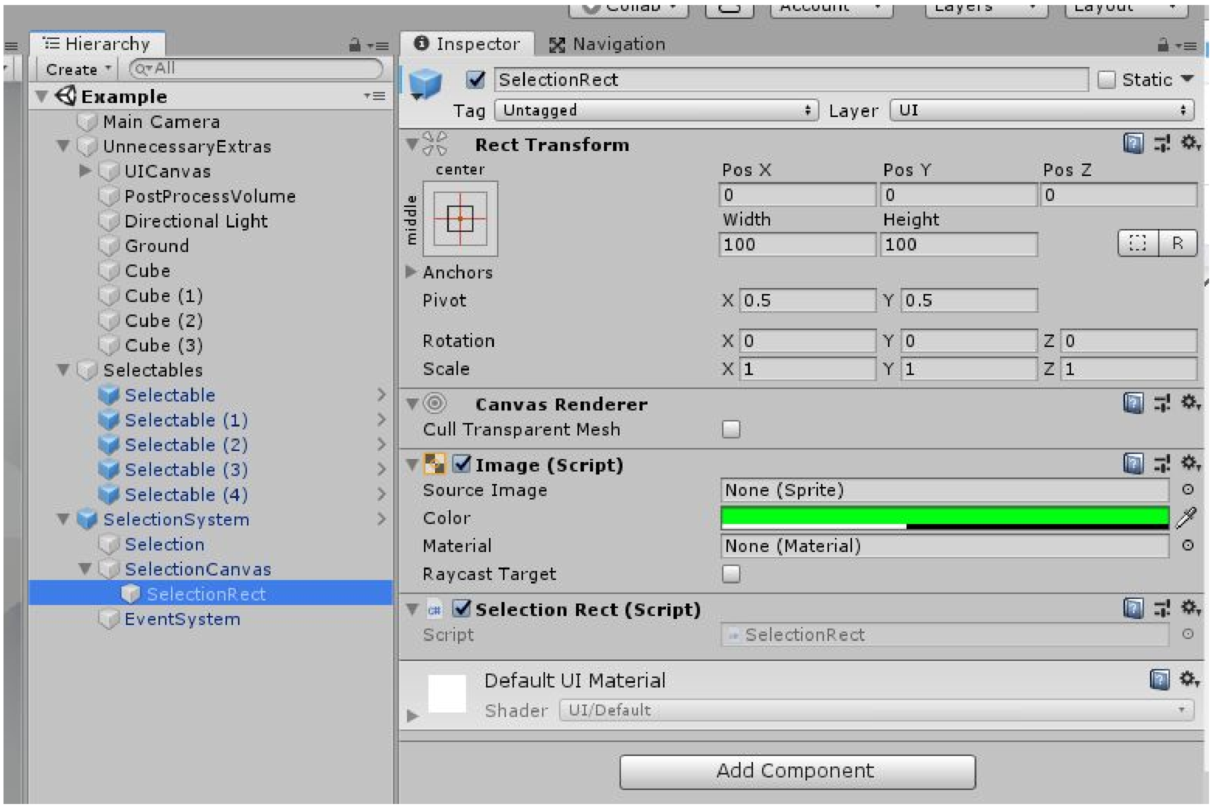


```
Example
  Main Camera
  UnnecessaryExtras
    UICanvas
    PostProcessVolume
    Directional Light
    Ground
    Cube
    Cube (1)
    Cube (2)
    Cube (3)
  Selectables
    Selectable
    Selectable (1)
    Selectable (2)
    Selectable (3)
    Selectable (4)
  SelectionSystem
    Selection
    SelectionCanvas
      SelectionRect
    EventSystem
```

```
Selection                                   Static
Tag  Untagged                    Layer  Default

Transform
Position      X 0     Y 0     Z 0
Rotation      X 0     Y 0     Z 0
Scale         X 1     Y 1     Z 1

Selection (Script)
Script          Selection

Selection Input (Script)
Script          SelectionInput
Selection       Selection (Selection)
Selection Rect  SelectionRect (SelectionRect)
```

Selection is a core component, which calculates all selection. It is driven by Selection Input component in our template.

Selection Input reads player input (mouse button press and release, mouse position), and sends it to the Selection class to make all selection calculations. If some Selectable is in selection area, it will be selected, etc.



Hierarchy

- Example
  - Main Camera
  - UnnecessaryExtras
    - UICanvas
    - PostProcessVolume
    - Directional Light
    - Ground
    - Cube
    - Cube (1)
    - Cube (2)
    - Cube (3)
  - Selectables
    - Selectable
    - Selectable (1)
    - Selectable (2)
    - Selectable (3)
    - Selectable (4)
  - SelectionSystem
    - Selection
    - SelectionCanvas
      - SelectionRect
    - EventSystem

Inspector

SelectionRect — Static: unchecked
Tag: Untagged — Layer: UI

Rect Transform (center / middle)

	Pos X: 0	Pos Y: 0	Pos Z: 0
	Width: 100	Height: 100	
Anchors			
Pivot	X 0.5	Y 0.5	
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Canvas Renderer
Cull Transparent Mesh: unchecked

Image (Script)
Source Image: None (Sprite)
Color: 🟩
Material: None (Material)
Raycast Target: unchecked

Selection Rect (Script)
Script: SelectionRect

Default UI Material
Shader: UI/Default

Add Component

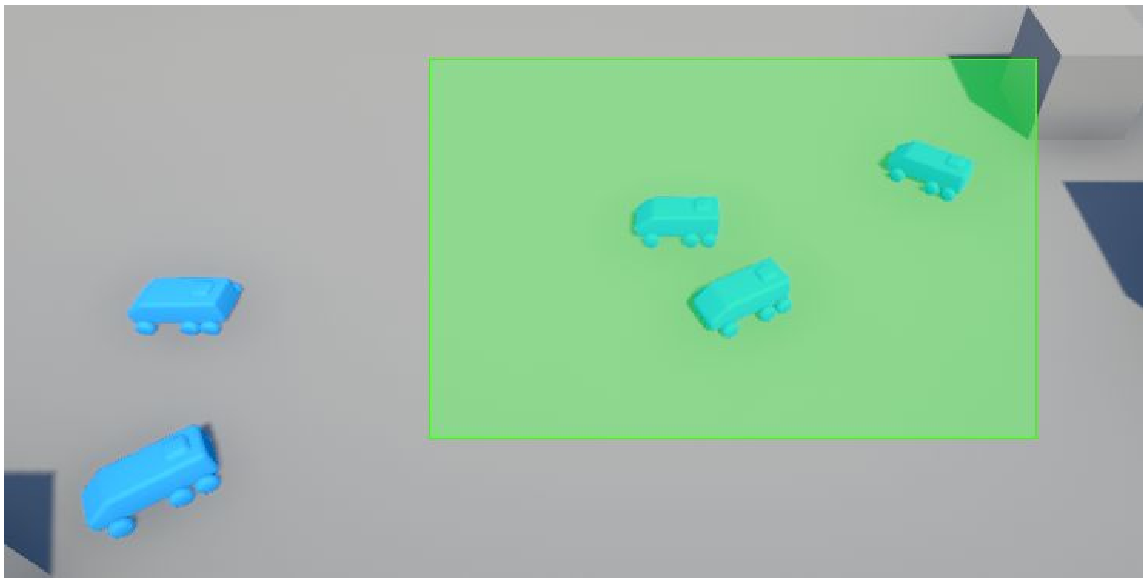
SelectionCanvas object contains multi-selection UI rectangle effect. You don't need to do any changes with this canvas like different screens scale etc, it is works as it is.

Selection Rect component handles multi-selection UI rectangle effect behaviour.

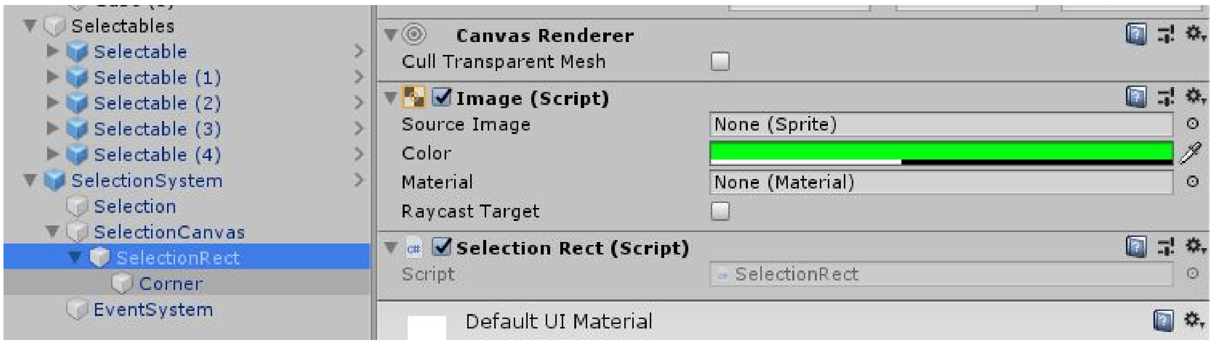
EventSystem is default Unity thing, that just reads whole player input. If you're have your own on scene, you can disable this one.

Customization of UI selection rectangle

It is about this effect:



You can edit Image component parameters of **SelectionRect** object in **SelectionCanvas**:

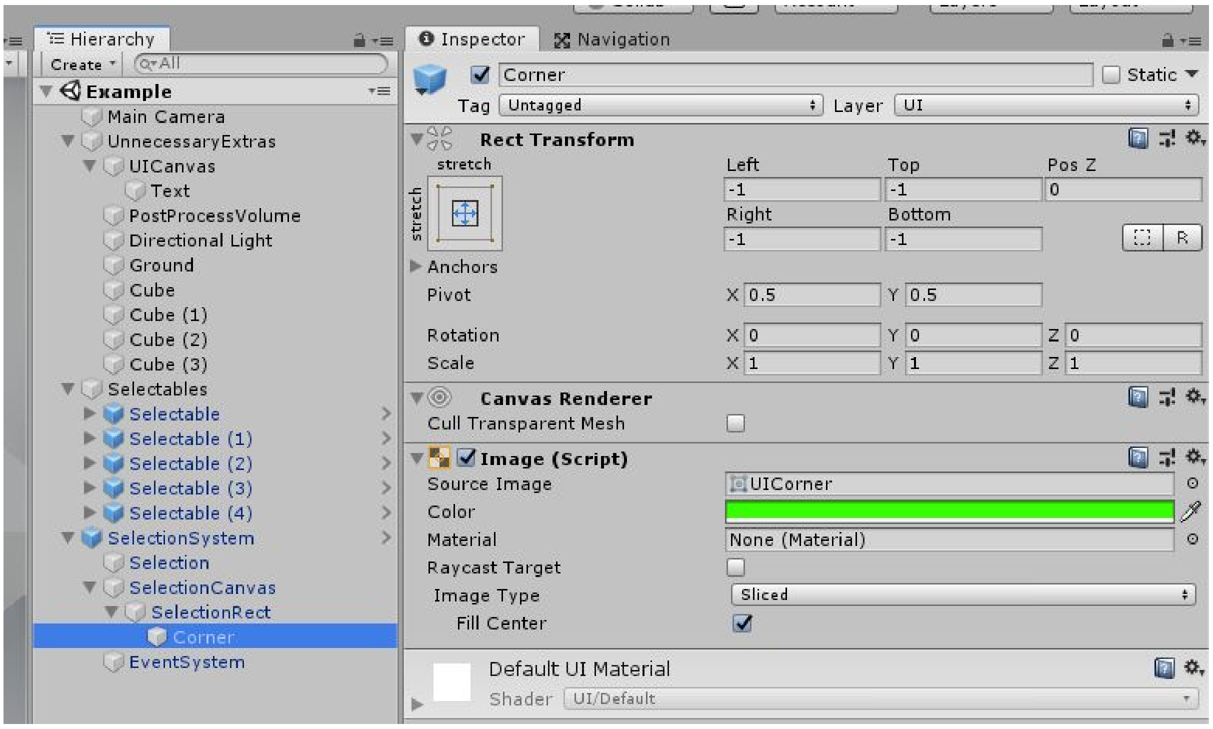


Selectables
Selectable
Selectable (1)
Selectable (2)
Selectable (3)
Selectable (4)
SelectionSystem
Selection
SelectionCanvas
SelectionRect
Corner
EventSystem

Canvas Renderer
Cull Transparent Mesh
Image (Script)
Source Image: None (Sprite)
Color
Material: None (Material)
Raycast Target
Selection Rect (Script)
Script: SelectionRect
Default UI Material

Note that by default **SelectionRect** is disabled - so, to edit it, enable it and do needed changes.

You also can add in childs additional effects, for example, you can see child named **Corner** - it adds simple corner for selection, also have Image component with color and filled image value etc.


```
Hierarchy
Create ▾  Q▾All
▼ Example
    Main Camera
  ▼ UnnecessaryExtras
    ▼ UICanvas
        Text
      PostProcessVolume
      Directional Light
      Ground
      Cube
      Cube (1)
      Cube (2)
      Cube (3)
  ▼ Selectables
    ▶ Selectable
    ▶ Selectable (1)
    ▶ Selectable (2)
    ▶ Selectable (3)
    ▶ Selectable (4)
  ▼ SelectionSystem
      Selection
    ▼ SelectionCanvas
      ▼ SelectionRect
          Corner
      EventSystem
```

```
Inspector   Navigation
[✓] Corner                                   [ ] Static ▾
Tag Untagged          Layer UI

▼ Rect Transform
stretch          Left    Top     Pos Z
                 -1      -1      0
                 Right   Bottom
                 -1      -1
▶ Anchors
Pivot            X 0.5   Y 0.5
Rotation         X 0     Y 0     Z 0
Scale            X 1     Y 1     Z 1

▼ Canvas Renderer
Cull Transparent Mesh   [ ]

▼ [✓] Image (Script)
Source Image     UICorner
Color            🟩
Material         None (Material)
Raycast Target   [ ]
Image Type       Sliced
  Fill Center    [✓]

Default UI Material
Shader  UI/Default
```

You also can add shaders to it using **Material** property or any other things you need for your selection rectangle effect.

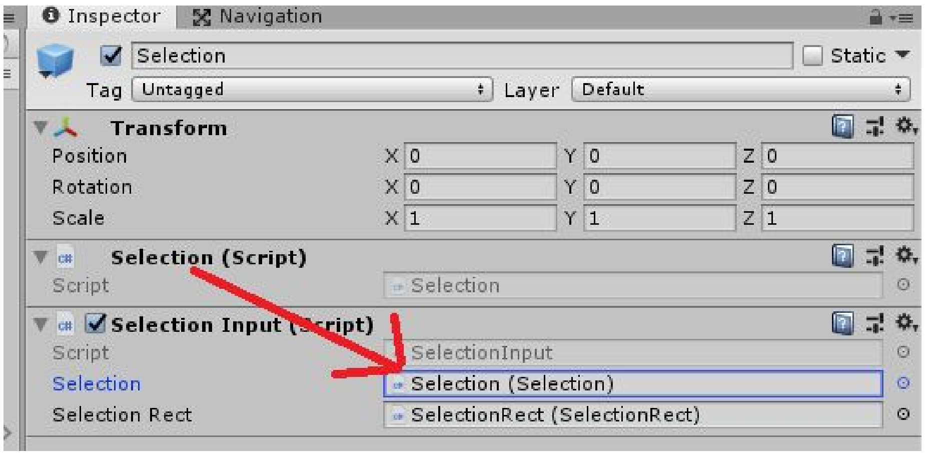
Note: Don’t forget to disable **Raycast target** toggle on any UI components you’re adding here. Otherwise it can block some clicks input on other game elements.

Programming

You do not need a lot of code to implement your custom selection. This partition will help with it.

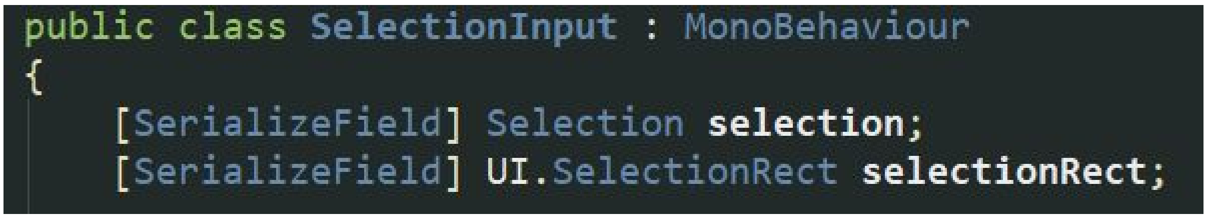
Get access to the Selection script

To do this, you first need access to the Selection class. You can use **FindObjectOfType<Selection>()** in **Start** of your script, or just add link to the **Selection** script like this:



```
Inspector   Navigation
☑ Selection                                   ☐ Static
Tag  Untagged                 Layer  Default
▼ Transform
  Position      X 0      Y 0      Z 0
  Rotation      X 0      Y 0      Z 0
  Scale         X 1      Y 1      Z 1
▼ Selection (Script)
  Script              Selection
▼ ☑ Selection Input (Script)
  Script              SelectionInput
  Selection           Selection (Selection)
  Selection Rect      SelectionRect (SelectionRect)
```

Code of this variable:



```
public class SelectionInput : MonoBehaviour
{
    [SerializeField] Selection selection;
    [SerializeField] UI.SelectionRect selectionRect;
```

Get current selected units

You will need to get selected units for your ordering system, for example. Call **selection.GetAllSelected()** and you'll receive List of ISelectable. Now you can do with it anything you need. You can use **for loop** to handle every selected object. ISelectable interface has **GetTransform** method, you can use it to "extract" game object from this interface.

Selection events

You can use events to add some custom behaviour on selection actions. Selection class have next events:

1. **OnSelectionStarted** - when selection request was received
2. **OnSelectionFinished** - when selection actions was fully finished.

Selectable events

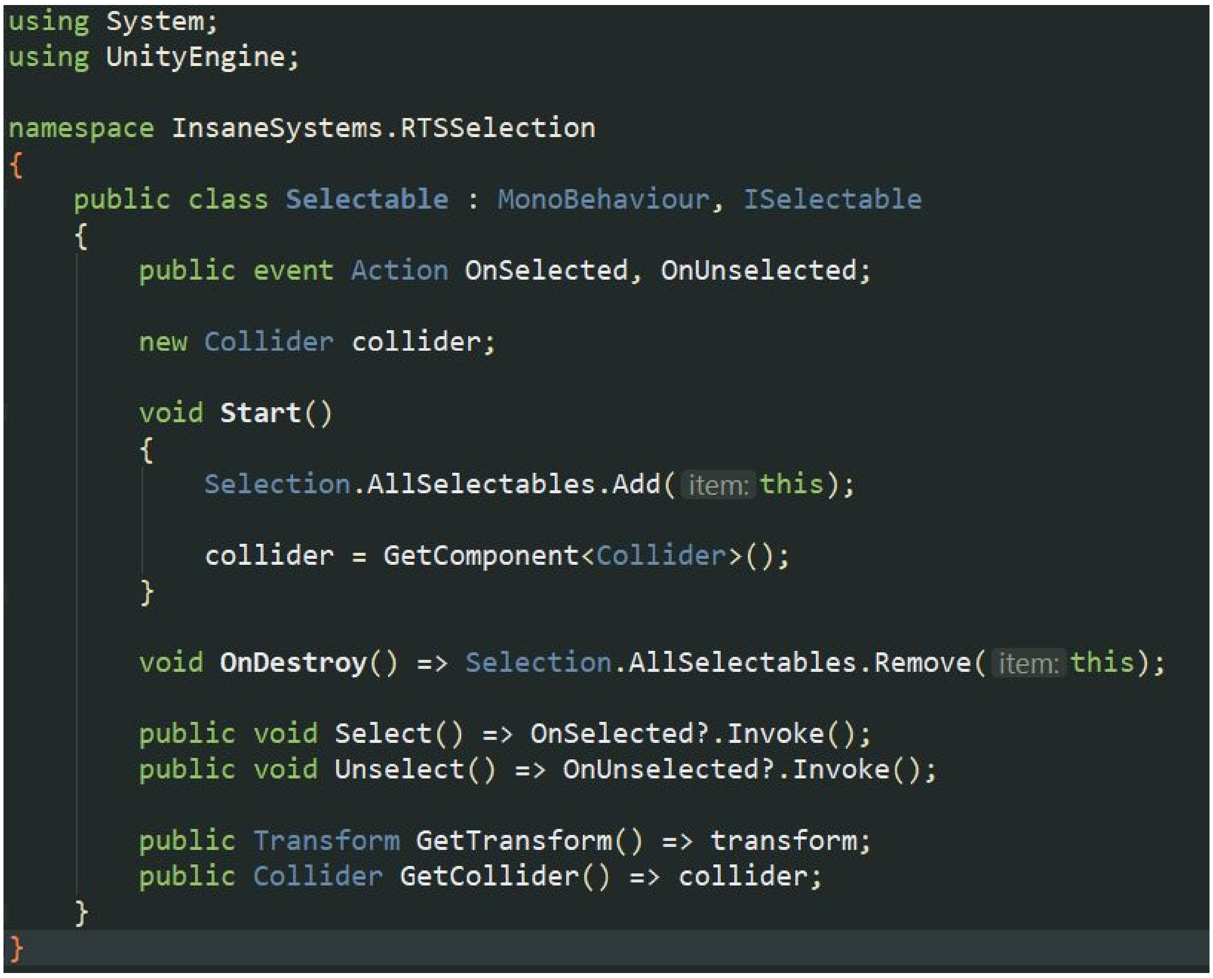
Use it to attach your effects, enable new logic, etc.

1. **OnSelected** - when Selectable was selected

2. **OnUnselected** - when Selectable was unselected

Creating custom Selectable script

You can derive from ISelectable interface and implement your own Selectable script with custom logic.



```
using System;
using UnityEngine;

namespace InsaneSystems.RTSSelection
{
    public class Selectable : MonoBehaviour, ISelectable
    {
        public event Action OnSelected, OnUnselected;

        new Collider collider;

        void Start()
        {
            Selection.AllSelectables.Add( item: this);

            collider = GetComponent<Collider>();
        }

        void OnDestroy() => Selection.AllSelectables.Remove( item: this);

        public void Select() => OnSelected?.Invoke();
        public void Unselect() => OnUnselected?.Invoke();

        public Transform GetTransform() => transform;
        public Collider GetCollider() => collider;
    }
}
```

Here you can see default example implementation. ISelectable interface necessary methods and events:

- **Select** - being called when object selected
- **Unselect** - same for unselection
- **GetTransform** - used for “extract” transform from interface (remember that interface is not MonoBehaviour, so without method like this you cannot use any gameObject or transform on it, etc)

- **GetCollider** - used for “extract” collider from interface. It is used in calculations. In interface realization it is better to cache collider for better performance. In example Selectable it already done.
- **OnSelected** - event which should be called when object selected, don’t forget to call it from Select method. Or, if you do not use events, just ignore it.
- **OnUnselected** - event which should be called when object unselected, don’t forget to call it from Unselect method. Or, if you do not use events, just ignore it.

So you’ll need implement all these methods and events in your own class. You can copy default example class **Selectable** code in your own, you also can extend **Select** and **Unselect** methods with custom logic, but you also can attach your custom logic to the events **OnSelected** and **OnUnselected**.

Online documentation version

You can read it here:
https://docs.google.com/document/d/1mu1PS94EoQTS3XEzYbYrvyMdctmROzcHmwy3KvhykEQ/edit?usp=sharing

Contacts

You can ask your questions or send your suggestions to us. ☺

Our **Discord** server: **https://discord.com/invite/7UUqQhU**

To contact us use email **godlikeaurora@gmail.com**