Creator for

Realistic Car Controller Pro

Thank you for purchasing and using Creator for Realistic Car Controller Pro

Content

Importing and Installing

Let's get started by importing the package to the project. Please use **Unity 2021.3.2f1** or later versions for the best compatibility. Importer will ask you which assets should be imported to the project, be sure to select all assets. After importing the package, editor window will search for Realistic Car Controller Pro in the project. If it can't find it, it will ask you to import the latest version of Realistic Car Controller Pro into the project.

Creator won't work without Realistic Car Controller Pro, project will still compile but you won't be able to use the system. Creator supports all versions of Realistic Car Controller from **V1.37**. It's recommended to use the latest version.

When Creator finds the Realistic Car Controller Pro in the project, you are ready to go. Go ahead and play the demo scenes first to understand how the system works. Demo scenes are in the scenes folder. You can test all features of the system and be sure to check everything.

TextMeshPro

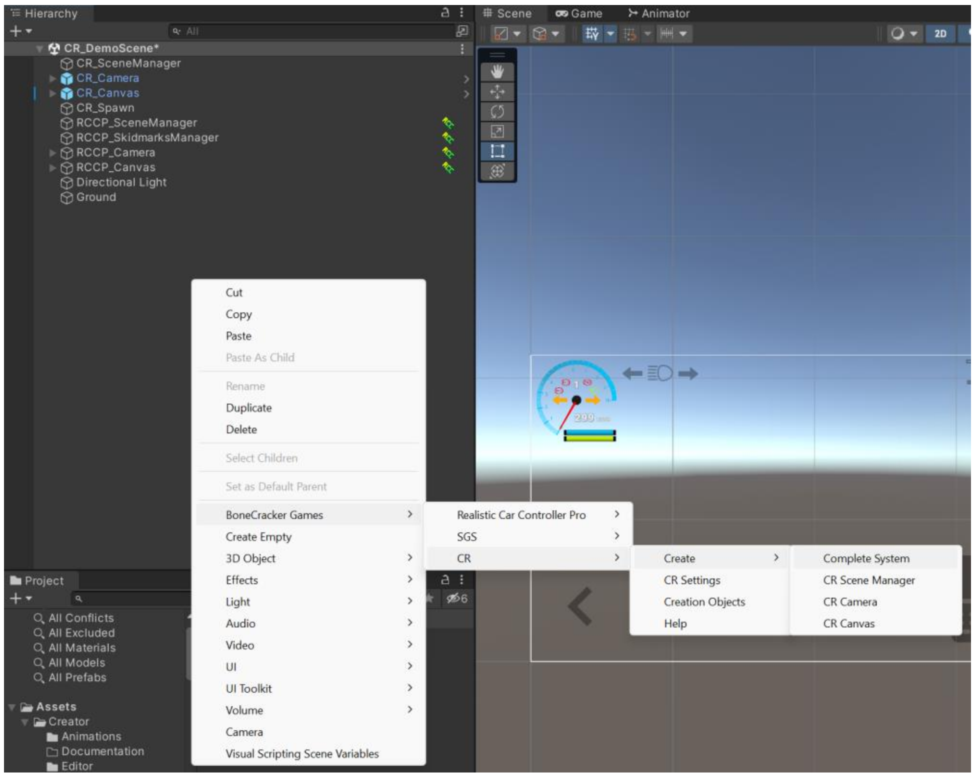
This project and RCCP are using **TextMeshPro**. Editor will ask you to install it when you open the demo scene, please install it. You can also install other resources of the **TextMeshPro** as well.

Overview

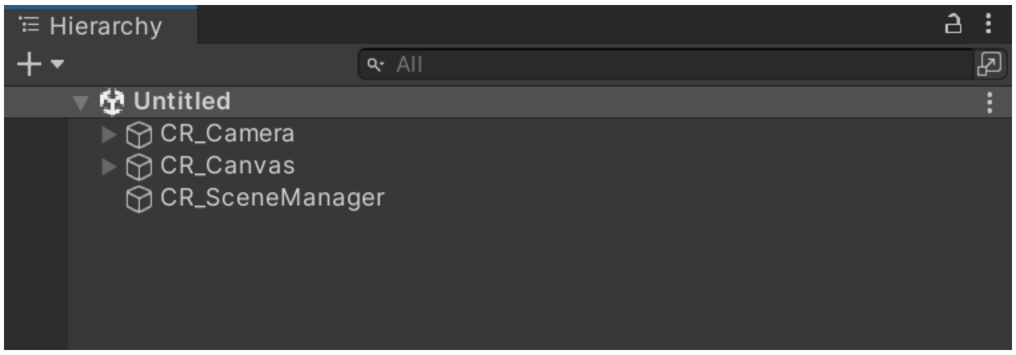
Let's take a look how the system works basically. **CR_SceneManager** is typically brain of the system. Scene must have this manager. It has a few settings for the system, you might want to check them out. System is very simple. When the player interacts with the system, **CR_SceneManager** will be used. All UI buttons are using this manager. Creation objects have **CR_Object** script. And the creation camera has **CR_Camera** script. Scenes must have these managers, that's all.

How to Create a New System for Creator

Creating and using the system in a new scene is very simple. CR has automated installation. Right click to the hierarchy panel and create the main menu system by **Tools → BCG → CR → Create Complete System**. This will add all necessary systems to the scene. Of course, it will be using the default settings and resources. We'll edit and change them later. Once you create the system, play and test the system to ensure everything is working fine in this new scene.

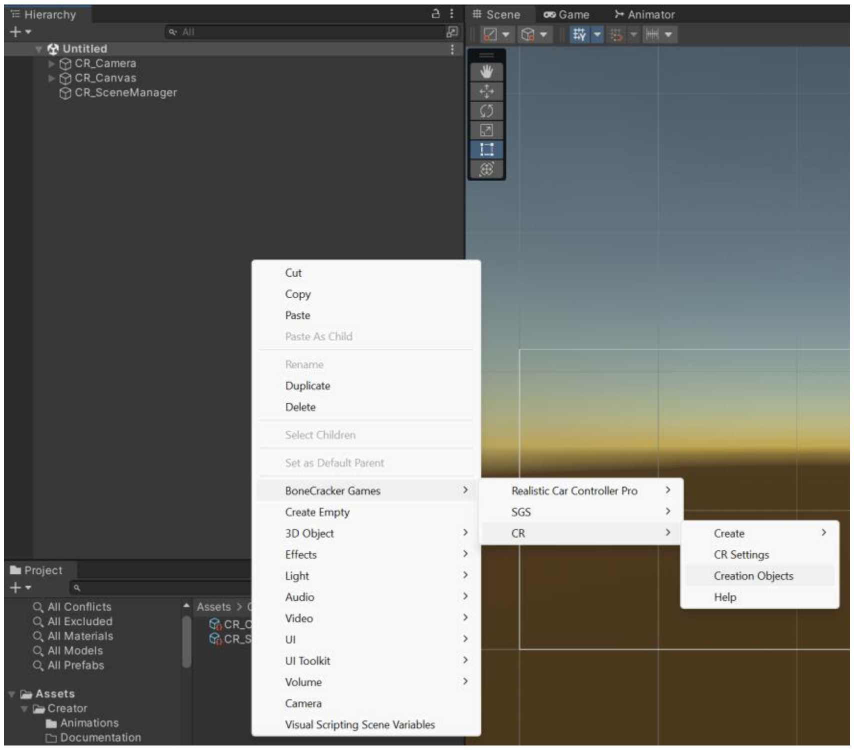


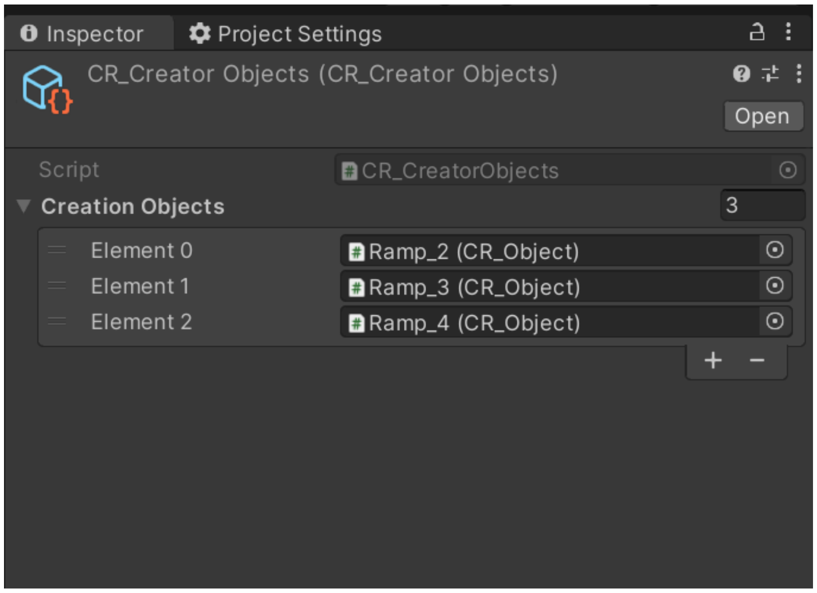
CR system requires these (All of them will be created and initialized automatically);



Creation Objects (Scriptable objects)

The system is using the demo prefabs currently. You can change them from **Tools → BCG → CR → Creation Objects**. It has three basic prefabs. All creation objects must have **CR_Object** script. UI buttons for the creation object selection will be using index values.





CR_Creator Objects (CR_Creator Objects)

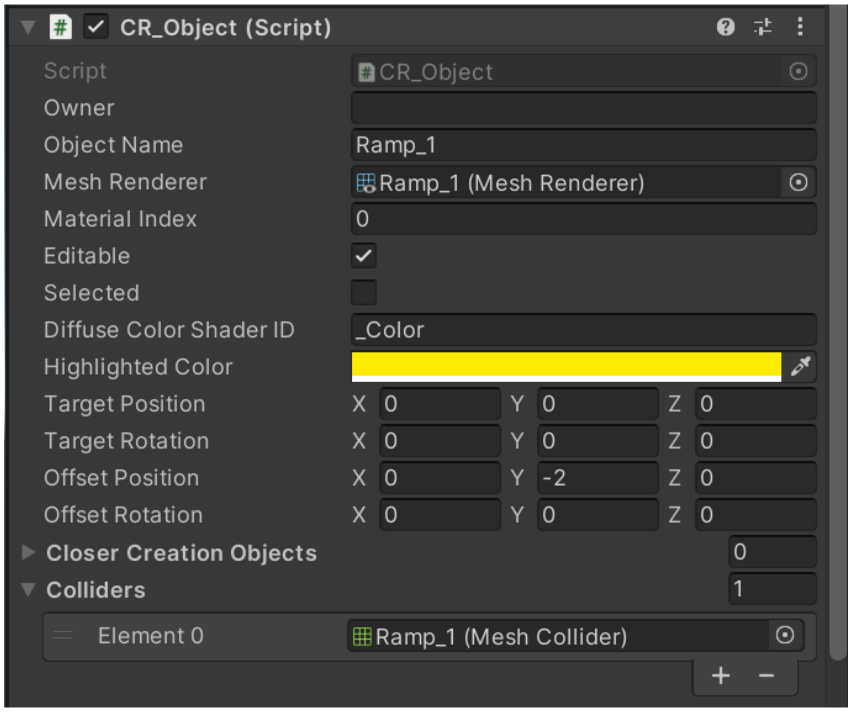
Script: CR_CreatorObjects

Creation Objects: 3

Element 0	Ramp_2 (CR_Object)
Element 1	Ramp_3 (CR_Object)
Element 2	Ramp_4 (CR_Object)

CR_Object

All creation objects must have this component. Position and rotation of the creation object will be processed through this script. Each object must have a unique name, don't use the same name for multiple creation objects. You can change other settings such as highlight color, target shader keyword, and target material index as well. When player selects is, object's renderer will be highlighted with the stated color.



CR_Object (Script)

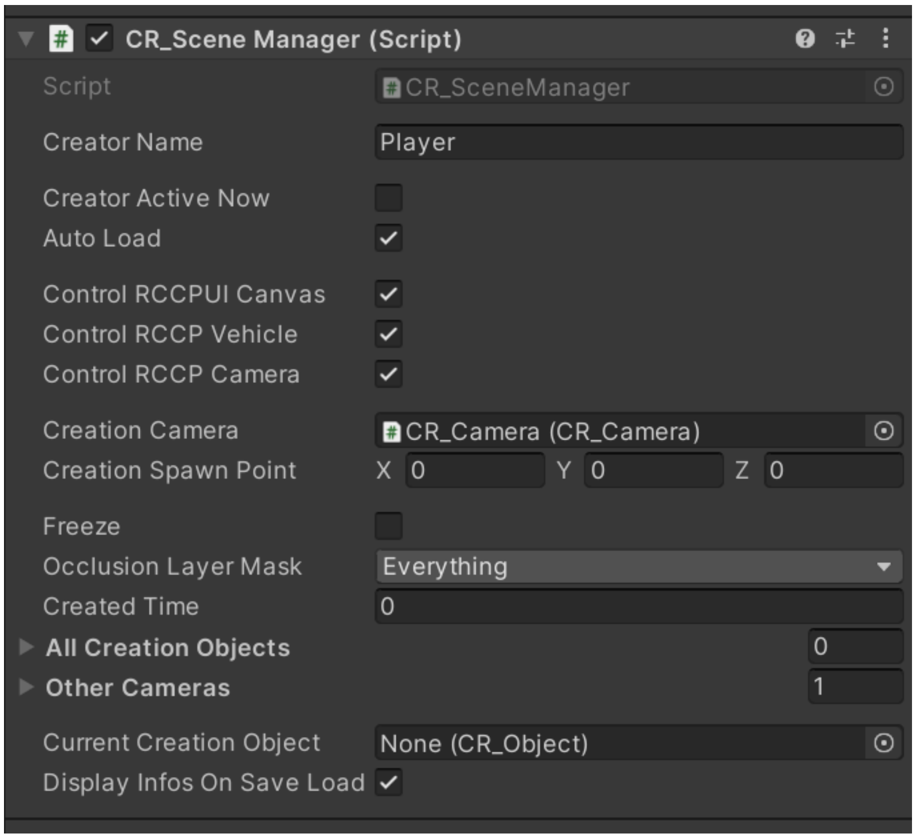
Script	CR_Object		
Owner			
Object Name	Ramp_1		
Mesh Renderer	Ramp_1 (Mesh Renderer)		
Material Index	0		
Editable	✓		
Selected			
Diffuse Color Shader ID	_Color		
Highlighted Color	🟨		
Target Position	X 0	Y 0	Z 0
Target Rotation	X 0	Y 0	Z 0
Offset Position	X 0	Y -2	Z 0
Offset Rotation	X 0	Y 0	Z 0
Closer Creation Objects	0		
Colliders	1		
Element 0	Ramp_1 (Mesh Collider)		

CR_SceneManager

CR_ SceneManager will be used to manage the creation objects. When player selects a creation object, **CR_SceneManager** will spawn it in the scene and operates it. System is using an external camera controller, which is **CR_Camera**. This external camera will be used only if player is customizing the scene. When system is not in use, the external camera will be disabled.

CR_SceneManager has a bool named **creatorActiveNow**. When player wants to use the system, this bool will be enabled. This bool also enables and disables few gameobjects in the scene to prevent unwanted situations. When this bool is set to true, an external camera named **CR_Camera** will be used, and all other cameras need to be disabled. Because we don't want to use multiple cameras at the same time. It also enables / disables the **RCCP Camera**, **RCCP Canvas**, and **RCCP vehicle** in the scene. **CR_SceneManager** has settings for them, they are enabled by default.

CR_SceneManager is a singleton class which you can get access to the instance by using **CR_SceneManager.Instance**. There are many useful public methods in this manager, you might want to check them out.



CR_Scene Manager (Script)

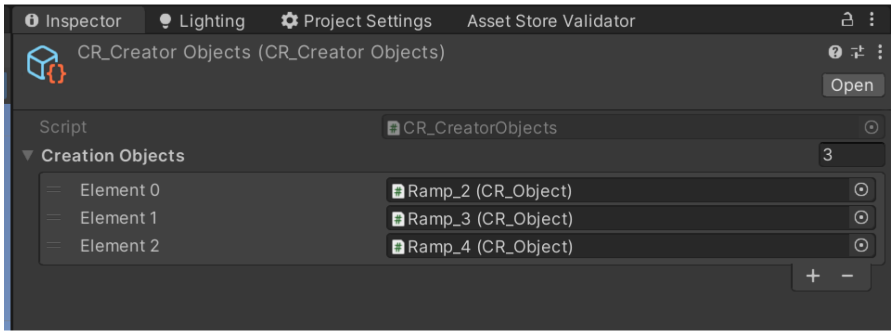
Script	CR_SceneManager
Creator Name	Player
Creator Active Now	☐
Auto Load	✓
Control RCCPUI Canvas	✓
Control RCCP Vehicle	✓
Control RCCP Camera	✓
Creation Camera	CR_Camera (CR_Camera)
Creation Spawn Point	X 0 Y 0 Z 0
Freeze	☐
Occlusion Layer Mask	Everything
Created Time	0
All Creation Objects	0
Other Cameras	1
Current Creation Object	None (CR_Object)
Display Infos On Save Load	✓

Saving, Loading, Clearing, and Restoring the Scene

Saving, loading, restoring, and cleaning the scene customization data have been managed by the **CR_SceneManager**. It has a custom class named **CR_SaveData**. This class contains saved position and rotation values of the objects. **CR_ SceneManager** has **Save**(), **Load**(), **Restore**(), and **Clean**() methods. Also **CR_SceneManager** has auto load option.

How to Add New Creation Objects to the List and UI Canvas

All creation objects must have **CR_Object** script. First, create a prefab version of your gameobject. You can do this by simply drag and drop the gameobject from the scene to the project. Once you create the prefab, be sure it has **CR_Object** script. Now you can add your prefab to the creation list. Open the list from **Tools → BCG → CR → Creation Objects** and create a new field for the object. Assign your prefab here, and you're almost done.



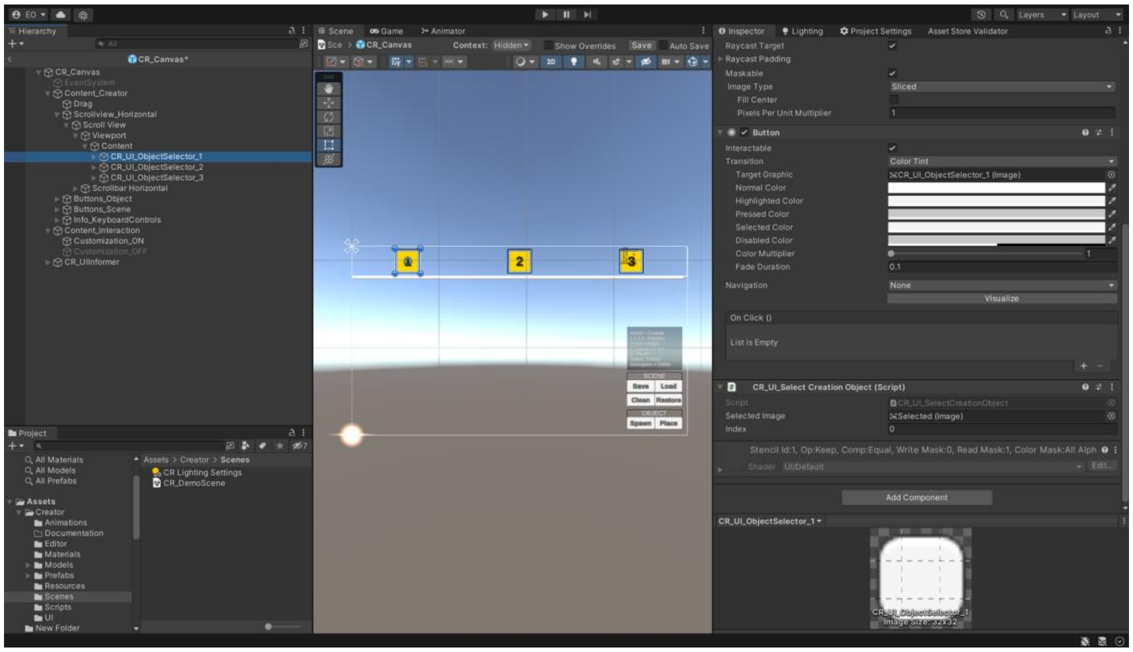
CR_Creator Objects (CR_Creator Objects)

Script: CR_CreatorObjects

Creation Objects: 3

Element 0	Ramp_2 (CR_Object)
Element 1	Ramp_3 (CR_Object)
Element 2	Ramp_4 (CR_Object)

CR UI Canvas has three buttons for the demo objects. Each button has **CR_UI_SelectCreationObject** script. This script has a target index value. Duplicate the last button and rename it. After that, change the index number on the script. And you're good to go.



CR_Canvas
- EventSystem
- Content_Creator
  - Drag
  - Scrollview_Horizontal
    - Scroll View
      - Viewport
        - Content
          - CR_UI_ObjectSelector_1
          - CR_UI_ObjectSelector_2
          - CR_UI_ObjectSelector_3
      - Scrollbar Horizontal
  - Buttons_Object
  - Buttons_Scene
  - Info_KeyboardControls
- Content_Interaction
  - Customization_ON
  - Customization_OFF
- CR_UIInformer

Button: Interactable ✓; Transition: Color Tint; Target Graphic: CR_UI_ObjectSelector_1 (Image); Color Multiplier: 1; Fade Duration: 0.1; Navigation: None; On Click (): List is Empty

CR_UI_Select Creation Object (Script)

Script	CR_UI_SelectCreationObject
Selected Image	Selected (Image)
Index	0

Using the CR_SceneManager in Your Scripts

CR_SceneManager is a singleton class. You can directly access to the instance by;

CR_SceneManager.*Instance;*

All methods have detailed comments in the script. These methods have been listed below. Methods in the image may be changed in later versions, so please check the script.

Examples;

// Saves the scene.

CR_SceneManager.Instance.Save();

// Loads the scene.

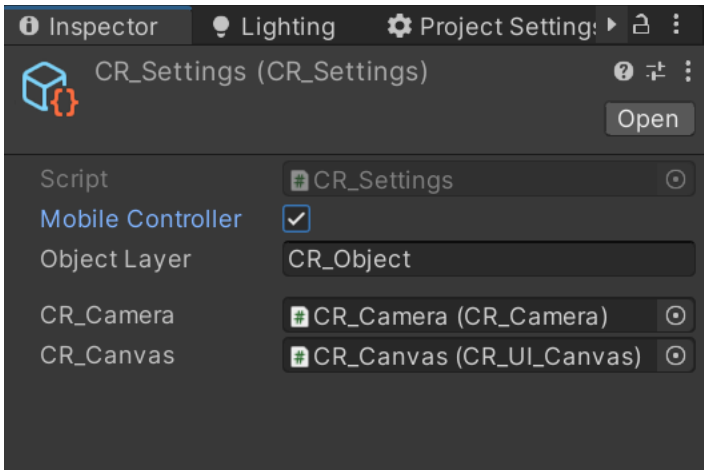
CR_SceneManager.Instance.Load();

// Places the current creation object.

CR_SceneManager.Instance.PlaceCreationObject();

Mobile Controllers

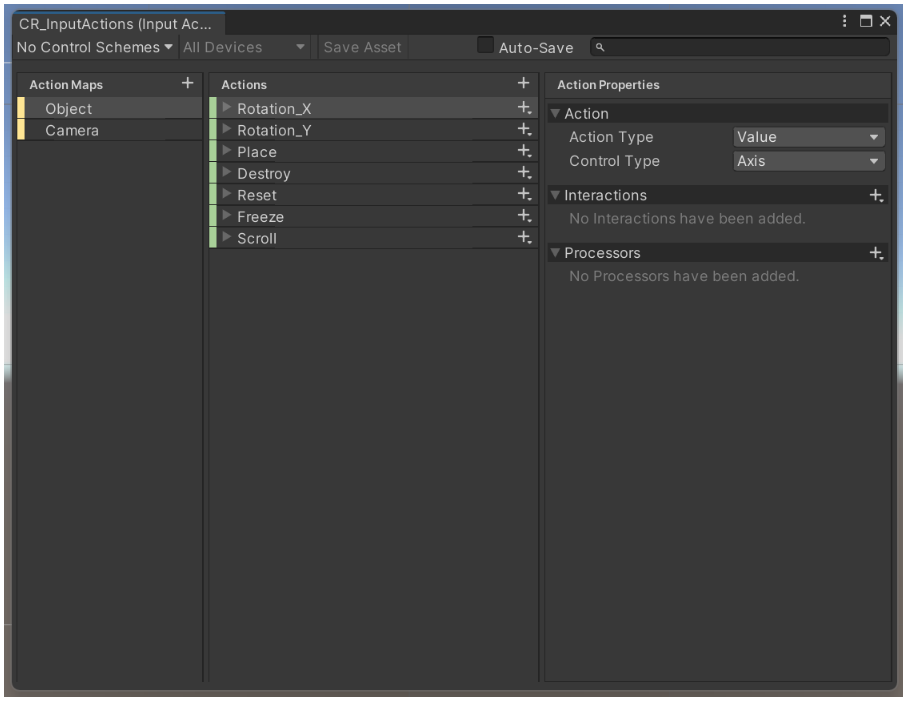
You can enable the mobile controllers from Tools → BCG → CR → Settings. After enabling the mobile controllers, joysticks will be in use. Players are able to move the camera and operate the current creation object with mobile controllers.



Script: CR_Settings
Mobile Controller: ✓
Object Layer: CR_Object
CR_Camera: CR_Camera (CR_Camera)
CR_Canvas: CR_Canvas (CR_UI_Canvas)

Inputs

All inputs have been processed by CR_InputManager. This manager receives all inputs from the connected and supported devices through the new input system. You can change the controller scheme from CR_InputActions which can be found in the resources folder. Don't change name of the inputs, otherwise generated script will fail to compile. You can change, edit, or add new inputs to the system.



Action Maps: Object, Camera

Actions: Rotation_X, Rotation_Y, Place, Destroy, Reset, Freeze, Scroll

Action
Action Type: Value
Control Type: Axis

Interactions
No Interactions have been added.

Processors
No Processors have been added.

Contact

Please include your invoice number while contacting me. I usually respond within a day. I may not respond on the weekend.

Email: bonecrackergames@gmail.com