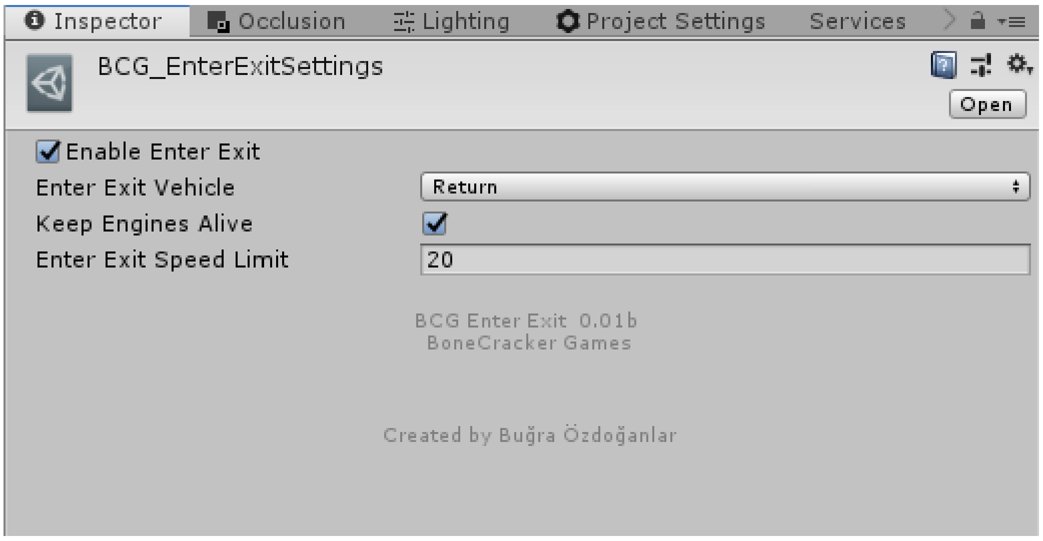
Enter-Exit System (BCG Shared Assets)

Supports all BCG vehicle controllers now. Enables / disables character controllers, registers / deregisters vehicles, and observes all events.

First, you must import "**BCG Shared Assets**" into your project to use enter / exit system. It will add necessary scripts and resources to use your FPS or TPS player for enter / exit vehicles. Package is in the "**Addon Packages\For BCG Shared Assets (Enter-Exit)**" folder. You can also import an integration package from the welcome window as well. **Tools → BCG → RCC → Welcome Window → Addons**.

There will be two new scenes imported to the "**Demo Scenes**" folder. One for FPS, and the other one for TPS controllers. FPS and TPS controllers are using different cameras. Usually, the camera of the FPS character is in the character gameobject. But TPS characters have unattached and independent cameras, and they are not parented to the character. Therefore, **BCG_EnterExitPlayer.cs** script has two options for use with FPS or TPS. If TPS is chosen, you must select the TPS camera in the scene.

You can edit some settings from (**Tools → BCG → Shared Assets → Edit Settings**).



BCG_EnterExitSettings

Enable Enter Exit	✓
Enter Exit Vehicle	Return
Keep Engines Alive	✓
Enter Exit Speed Limit	20

BCG Enter Exit 0.01b
BoneCracker Games

Created by Buğra Özdoğanlar

How The System Works (Simple)

All enter exit vehicles must have “**BCG_EnterExitVehicle**” component. And our **FPS / TPS** character must have “**BCG_EnterExitPlayer**” component. **BCG_SceneManager** (Will be created automatically in the scene) will be observing all vehicles and your character in the scene. If vehicle has no driver, ***canControl** of the vehicle will be disabled, and not registered as player vehicle. If our character has entered any vehicle, **BCG_SceneManager** will register the target vehicle as player vehicle, enables the **canControl** of the vehicle. At that moment, our character will be parented to the vehicle and disabled entirely. Now, **BCG_EnterExitVehicle** has a ****driver** now. If player gets out from the vehicle, **BCG_SceneManager** will be activating character controller, and transport it to “**Get Out Pos**” of the vehicle. And lastly, deregister the vehicle and disable ***canControl** of the vehicle. Now, the vehicle is not controllable, not a player vehicle.

* = **canControl** bool of the **RCC_CarControllerV4**. If it’s enabled, the vehicle is controllable.

** = **driver** variable of the **BCG_EnterExitVehicle**.

How To Make It Work

All you must do is, add necessary scripts to the vehicles and to your character player. Get out position of the vehicles are created automatically in script if you don’t select it in **BCG_EnterExitVehicle**. You can replace get out positions.

For FPS characters, it will depend on your camera of the FPS player. For TPS player, it depends on your actual TPS Player, not camera.

Character Controllers must have BCG_EnterExitPlayer

Vehicles must have BCG_EnterExitVehicle

BCG_EnterExitManager will be created automatically in your scene.
BCG_EnterExitManager will be managing all the process.

How The System Works (Detailed)

Please read simple version first if you haven't read. Let's get it straight with more detailed version.

1- **BCG_EnterExitVehicle** – must be attached to all vehicles.
2- **BCG_EnterExitPlayer** – must be attached to the character.
3- **BCG_EnterExitManager** – will be created automatically.

All the system is based on events. **BCG_EnterExitPlayer** is firing an event when it's spawned, destroyed, gets in a vehicle, or gets out of the vehicle. **BCG_SceneManager** will be listening to this event as well. So, the manager knows your all your character events. All events are explained below.

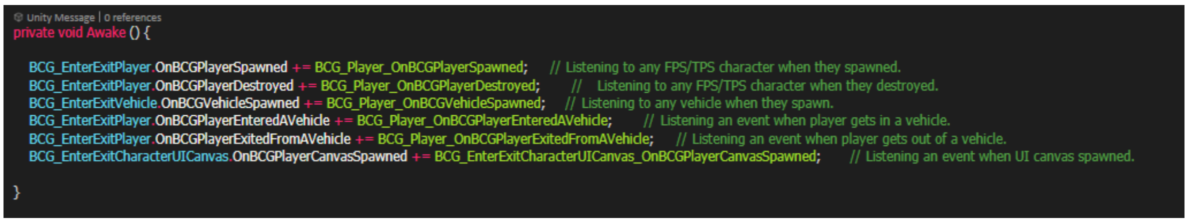
public delegate void **onBCGPlayerSpawned**(BCG_EnterExitPlayer player);

public delegate void **onBCGPlayerDestroyed**(BCG_EnterExitPlayer player);

public delegate void **onBCGPlayerEnteredAVehicle**(BCG_EnterExitPlayer player, BCG_EnterExitVehicle vehicle);

public delegate void **onBCGPlayerExitedFromAVehicle**(BCG_EnterExitPlayer player, BCG_EnterExitVehicle vehicle);

You can listen to any events on the **BCG_EnterExitPlayer** **or** **BCG_EnterExitVehicle**. For example.



```
Unity Message | 0 references
private void Awake () {

    BCG_EnterExitPlayer.OnBCGPlayerSpawned += BCG_Player_OnBCGPlayerSpawned;    // Listening to any FPS/TPS character when they spawned.
    BCG_EnterExitPlayer.OnBCGPlayerDestroyed += BCG_Player_OnBCGPlayerDestroyed;    //   Listening to any FPS/TPS character when they destroyed.
    BCG_EnterExitVehicle.OnBCGVehicleSpawned += BCG_Player_OnBCGVehicleSpawned;    // Listening to any vehicle when they spawn.
    BCG_EnterExitPlayer.OnBCGPlayerEnteredAVehicle += BCG_Player_OnBCGPlayerEnteredAVehicle;    // Listening an event when player gets in a vehicle.
    BCG_EnterExitPlayer.OnBCGPlayerExitedFromAVehicle += BCG_Player_OnBCGPlayerExitedFromAVehicle;    // Listening an event when player gets out of a vehicle.
    BCG_EnterExitCharacterUICanvas.OnBCGPlayerCanvasSpawned += BCG_EnterExitCharacterUICanvas_OnBCGPlayerCanvasSpawned;    // Listening an event when UI canvas spawned.

}
```

Nice color scheme, isn't it? 😊 Anyways, **BCG_EnterExitManager** will be using **GetIn()** or **GetOut()** methods by listening to these events. **GetIn()** method requires "**targetVehicle**". **GetOut()** method doesn't require any parameters.

When our character standing next to the vehicle, get in text will be enabled by **BCG_EnterExitPlayer**. If player pushes the interaction button, **OnBCGPlayerEnteredAVehicle** event will be fired with targetVehicle data. As you now, this event has been listening by **BCG_SceneManager**. So, **BCG_SceneManager** will be firing **GetIn(targetVehicle)** method to do the rest.