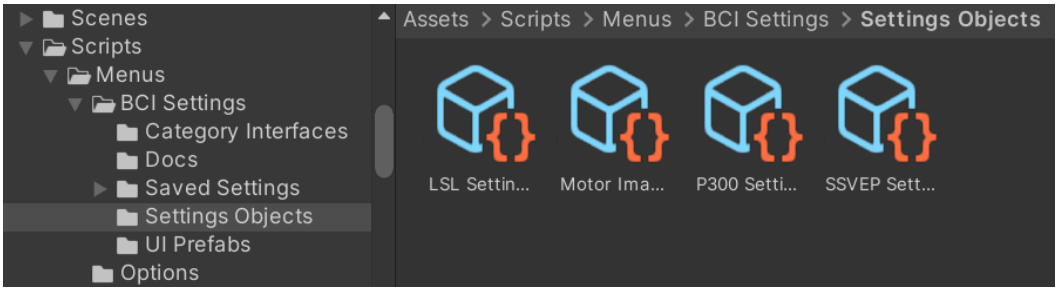
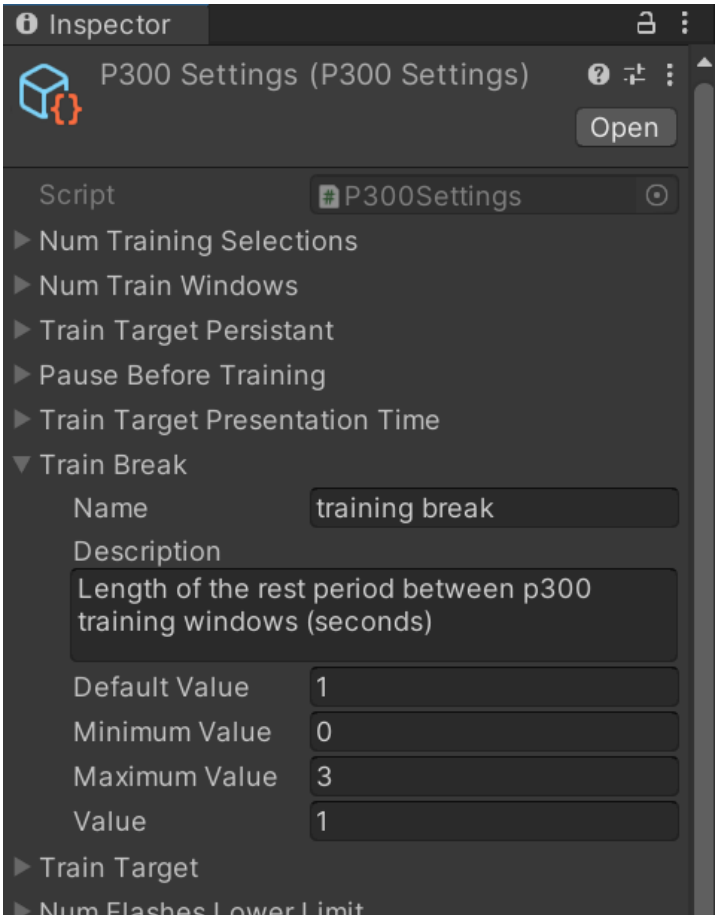
How to Add, Remove, and Edit BCI Settings

The system handling BCI settings exists to provide a human readable settings file and remain functional as the list of settings it saves is altered. This document aims to provide much more detail than necessary, but please reach out with any questions.

Editing Extant Settings (click the objects, change the stuff)

Settings are organized into categories according to their intended “destination”. Settings that affect the P300 Controller Behavior are in P300 settings and so on. Each category has a scriptable object in:
Assets/Scripts/Menus/BCI Settings/Settings Objects.





From these scriptable objects you can alter the display name, default value and suggested range of each setting.

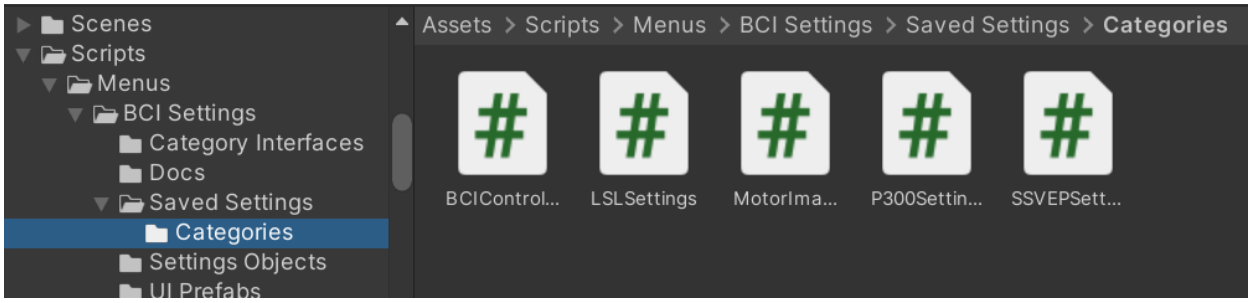
The default value, range, and description can be altered seamlessly and will be immediately reflected by the settings file itself the next time it is saved.

Changing the display name will cause some warning statements to be printed (lost connection to saved value) but will also work seamlessly in practice.

The current value of the setting is only displayed as an in engine utility and should really just be ignored.

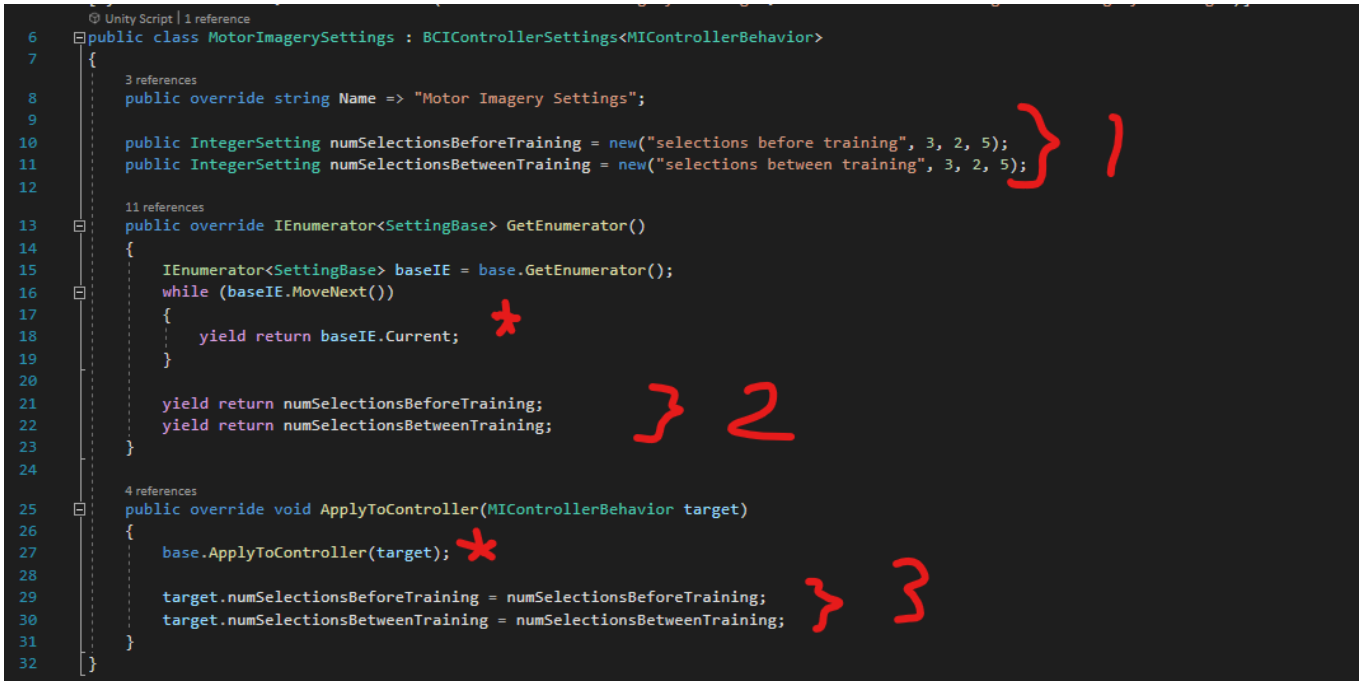
Adding or Removing Settings

To add or remove settings, the definitions of the category must be edited in code. These definitions can all be found in:
Assets/Scripts/Menus/BCI Settings/Saved Settings/Categories



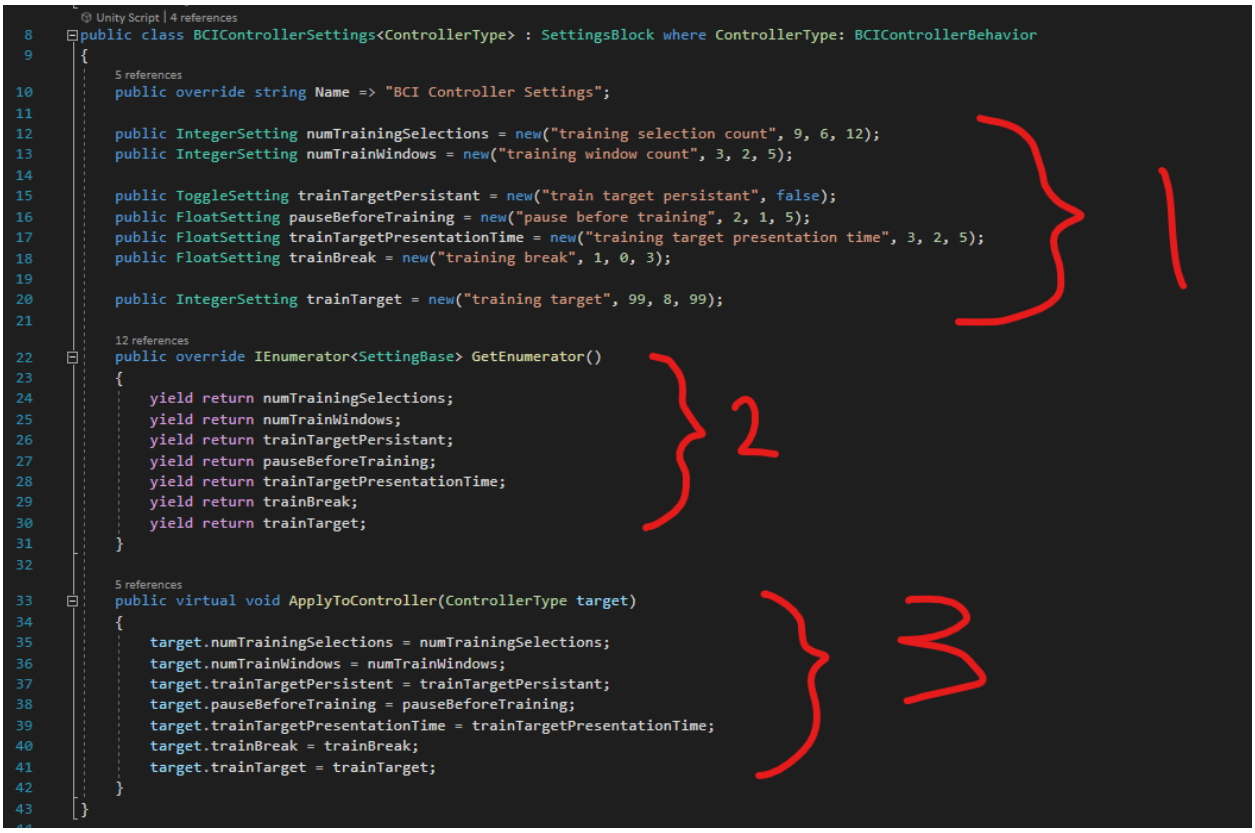
To add or remove a setting, there are 3 steps:

- Add or remove member variable
    - The constructor is completely unnecessary, just preference
- Add or remove the variable from the enumerator
- Add the application functionality



* These sections include settings from the base class

Each of the functional settings categories (all but LSL), inherit from a base BCIControllerSettings class that shares training attributes across all relevant categories. Edit this base class to add or remove settings from this shared pool.



```
Unity Script | 4 references
public class BCIControllerSettings<ControllerType> : SettingsBlock where ControllerType: BCIControllerBehavior
{
    5 references
    public override string Name => "BCI Controller Settings";

    public IntegerSetting numTrainingSelections = new("training selection count", 9, 6, 12);
    public IntegerSetting numTrainWindows = new("training window count", 3, 2, 5);

    public ToggleSetting trainTargetPersistant = new("train target persistant", false);
    public FloatSetting pauseBeforeTraining = new("pause before training", 2, 1, 5);
    public FloatSetting trainTargetPresentationTime = new("training target presentation time", 3, 2, 5);
    public FloatSetting trainBreak = new("training break", 1, 0, 3);

    public IntegerSetting trainTarget = new("training target", 99, 8, 99);

    12 references
    public override IEnumerator<SettingBase> GetEnumerator()
    {
        yield return numTrainingSelections;
        yield return numTrainWindows;
        yield return trainTargetPersistant;
        yield return pauseBeforeTraining;
        yield return trainTargetPresentationTime;
        yield return trainBreak;
        yield return trainTarget;
    }

    5 references
    public virtual void ApplyToController(ControllerType target)
    {
        target.numTrainingSelections = numTrainingSelections;
        target.numTrainWindows = numTrainWindows;
        target.trainTargetPersistent = trainTargetPersistant;
        target.pauseBeforeTraining = pauseBeforeTraining;
        target.trainTargetPresentationTime = trainTargetPresentationTime;
        target.trainBreak = trainBreak;
        target.trainTarget = trainTarget;
    }
}
```

Adding or Editing Categories

This should be pretty straightforward to figure out, but you might be better off reaching out to bridge the step of connecting it to the in game menu.