C# API

All Space Graphics Toolkit features can be accessed from C#. Most features have the same names as seen in the inspector. For example:

```
var myThruster = new GameObject("My Thruster").AddComponent<SgtThruster>();
```

or

```
var myThruster = SgtThruster.CreateThruster(myLayer, myParent);

myThruster.FlameSprite = myFlameSprite;
```

This is the same as Right Click -> Space Graphics Toolkit -> Thruster, and dragging and dropping a new FlameSprite sprite.

However, sometimes you need to call a method to manually update the component. For example:

```
var myRing = new GameObject("My Ring").AddComponent<SgtRing>();
```

or

```
var myRing = SgtRing.CreateRing(myLayer, myParent);

myRing.Color = Color.red;

myRing.UpdateMaterial();
```

The UpdateMaterial call at the end is automatically called when making changes in the inspector, however, from C# you must manually call this. If this was made automatic from code then updating more than one setting at a time would waste a lot of CPU, or require checking for changes in Update. The easiest way to see what method call is required to update a component is by looking at the inspector code for the setting. For example:

```
DrawDefault("Color", ref updateMaterial);
```

This is the line of code that draws the Color setting for the SgtRing component, as you can see it requires the updateMaterial to be called after modification. You can see the inspector code for each component at the top its C# file inside the UNITY_EDITOR block.