VRC OWO System World Integration.

In this documentation we will teach you the Advanced Sensation details of the OWO World Integration (O.W.I)
This includes:

- Sensations Basics
- Sensations Multi-Event
- Setting Muscles

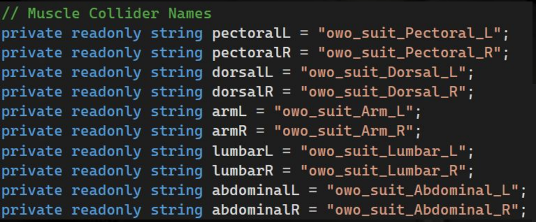
(If you have not gone over the Basic Setup please see the documentation on our Github page or in the /Assets/O.W.I/Documents folder in your project after importing the VRC_O.W.I.unitypackage)

In this documentation we use O.W.I as an abbreviation of the OWO World Integration name to make it easier to read and less repetitive

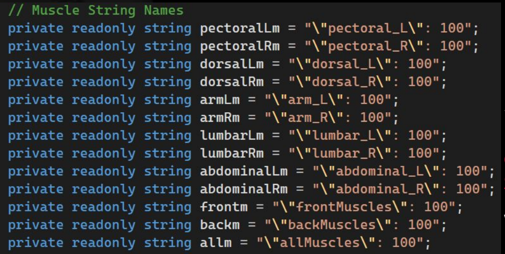
VRC OWO System World integration.

Sensation basics:

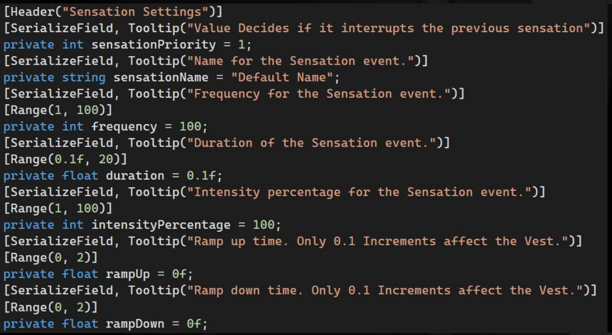
(For after you already have a VRChat Project and Installed the O.W.I package)



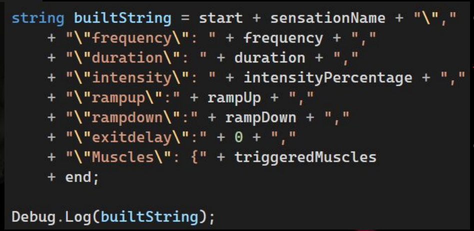
```
// Muscle Collider Names
private readonly string pectoralL = "owo_suit_Pectoral_L";
private readonly string pectoralR = "owo_suit_Pectoral_R";
private readonly string dorsalL = "owo_suit_Dorsal_L";
private readonly string dorsalR = "owo_suit_Dorsal_R";
private readonly string armL = "owo_suit_Arm_L";
private readonly string armR = "owo_suit_Arm_R";
private readonly string lumbarL = "owo_suit_Lumbar_L";
private readonly string lumbarR = "owo_suit_Lumbar_R";
private readonly string abdominalL = "owo_suit_Abdominal_L";
private readonly string abdominalR = "owo_suit_Abdominal_R";
```



```
// Muscle String Names
private readonly string pectoralLm = "\"pectoral_L\": 100";
private readonly string pectoralRm = "\"pectoral_R\": 100";
private readonly string dorsalLm = "\"dorsal_L\": 100";
private readonly string dorsalRm = "\"dorsal_R\": 100";
private readonly string armLm = "\"arm_L\": 100";
private readonly string armRm = "\"arm_R\": 100";
private readonly string lumbarLm = "\"lumbar_L\": 100";
private readonly string lumbarRm = "\"lumbar_R\": 100";
private readonly string abdominalLm = "\"abdominal_L\": 100";
private readonly string abdominalRm = "\"abdominal_R\": 100";
private readonly string frontm = "\"frontMuscles\": 100";
private readonly string backm = "\"backMuscles\": 100";
private readonly string allm = "\"allMuscles\": 100";
```



```
[Header("Sensation Settings")]
[SerializeField, Tooltip("Value Decides if it interrupts the previous sensation")]
private int sensationPriority = 1;
[SerializeField, Tooltip("Name for the Sensation event.")]
private string sensationName = "Default Name";
[SerializeField, Tooltip("Frequency for the Sensation event.")]
[Range(1, 100)]
private int frequency = 100;
[SerializeField, Tooltip("Duration of the Sensation event.")]
[Range(0.1f, 20)]
private float duration = 0.1f;
[SerializeField, Tooltip("Intensity percentage for the Sensation event.")]
[Range(1, 100)]
private int intensityPercentage = 100;
[SerializeField, Tooltip("Ramp up time. Only 0.1 Increments affect the Vest.")]
[Range(0, 2)]
private float rampUp = 0f;
[SerializeField, Tooltip("Ramp down time. Only 0.1 Increments affect the Vest.")]
[Range(0, 2)]
private float rampDown = 0f;
```



```
string builtString = start + sensationName + "\","
    + "\"frequency\": " + frequency + ","
    + "\"duration\": " + duration + ","
    + "\"intensity\": " + intensityPercentage + ","
    + "\"rampup\":" + rampUp + ","
    + "\"rampdown\":" + rampDown + ","
    + "\"exitdelay\":" + 0 + ","
    + "\"Muscles\": {" + triggeredMuscles
    + end;

Debug.Log(builtString);
```

Requirements:
VRChat World Project
VRC_O.W.I.unitypackage Installed

First we will take a look at how we're handling Sensations in the prebuilt scripts.

(Images are from a script that plays a single sensation for the muscles that were collided during the collision timer it can be found at **Assets\O.W.I\Scripts\O.W.IUdon Scripts\OWIMicroSensationCreator.cs**)

We're using collision or trigger events that check for the Name of the O.W.I Avatar objects so we know where we are contacting.

(You can see the Names that we are using in the images on the left, if you are using the same approach you can use anything for these as long as they match the correct object in the world.)

We're then using that logic to select which muscle to send in the communication string, which contains the name of the muscle,then the specific muscle intensity value.

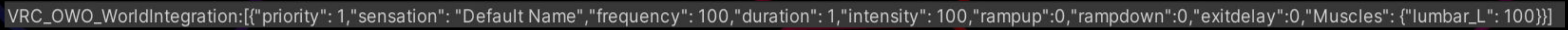
(Muscle names cannot be changed, specific muscle intensity can be set to anything between 1 and 100, to set that specific muscle to be lower than the overall sensation, image on the left shows a static value of 100 for all muscles.)

After you have the Muscles from the Collision then the Sensation values come into play as that is what make the suit actually feel.

(In the Image to the left containing them you can seen ranges set these are the safe values that the OWO System can accept.)

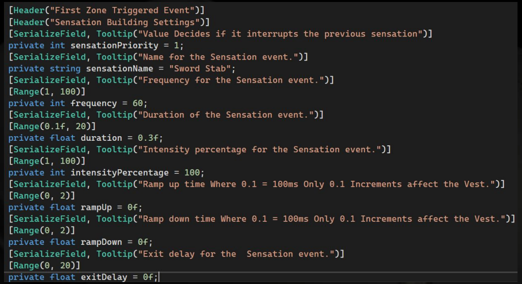
Finally you have how we are handling this information to send it out to the external exe.

(This is done by sending our built string to the debug log so the external exe can read and parse it a proper single sensation string output image is shown at the bottom.)

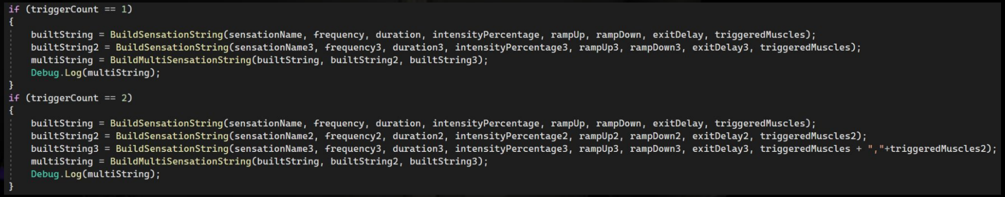


```
VRC_OWO_WorldIntegration:[{"priority": 1,"sensation": "Default Name","frequency": 100,"duration": 1,"intensity": 100,"rampup":0,"rampdown":0,"exitdelay":0,"Muscles": {"lumbar_L": 100}}]
```

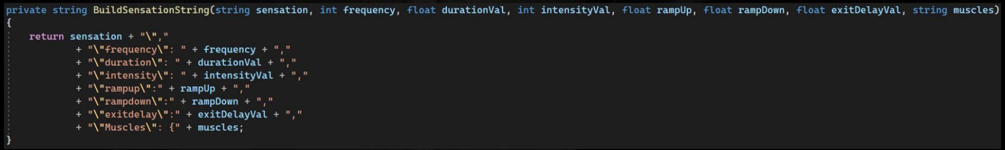
VRC OWO System World integration.



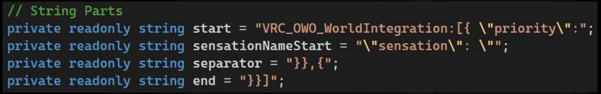
```
[Header("First Zone Triggered Event")]
[Header("Sensation Building Settings")]
[SerializeField, Tooltip("Value Decides if it interrupts the previous sensation")]
private int sensationPriority = 1;
[SerializeField, Tooltip("Name for the Sensation event.")]
private string sensationName = "Sword Stab";
[SerializeField, Tooltip("Frequency for the Sensation event.")]
[Range(1, 100)]
private int frequency = 60;
[SerializeField, Tooltip("Duration of the Sensation event.")]
[Range(0.1f, 20)]
private float duration = 0.3f;
[SerializeField, Tooltip("Intensity percentage for the Sensation event.")]
[Range(1, 100)]
private int intensityPercentage = 100;
[SerializeField, Tooltip("Ramp up time Where 0.1 = 100ms Only 0.1 Increments affect the Vest.")]
[Range(0, 2)]
private float rampUp = 0f;
[SerializeField, Tooltip("Ramp down time Where 0.1 = 100ms Only 0.1 Increments affect the Vest.")]
[Range(0, 2)]
private float rampDown = 0f;
[SerializeField, Tooltip("Exit delay for the  Sensation event.")]
[Range(0, 20)]
private float exitDelay = 0f;
```



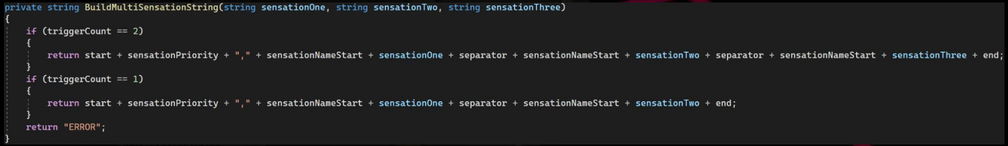
```
if (triggerCount == 1)
{
    builtString = BuildSensationString(sensationName, frequency, duration, intensityPercentage, rampUp, rampDown, exitDelay, triggeredMuscles);
    builtString2 = BuildSensationString(sensationName3, frequency3, duration3, intensityPercentage3, rampUp3, rampDown3, exitDelay3, triggeredMuscles);
    multiString = BuildMultiSensationString(builtString, builtString2, builtString3);
    Debug.Log(multiString);
}
if (triggerCount == 2)
{
    builtString = BuildSensationString(sensationName, frequency, duration, intensityPercentage, rampUp, rampDown, exitDelay, triggeredMuscles);
    builtString2 = BuildSensationString(sensationName2, frequency2, duration2, intensityPercentage2, rampUp2, rampDown2, exitDelay2, triggeredMuscles2);
    builtString3 = BuildSensationString(sensationName3, frequency3, duration3, intensityPercentage3, rampUp3, rampDown3, exitDelay3, triggeredMuscles + ","+triggeredMuscles2);
    multiString = BuildMultiSensationString(builtString, builtString2, builtString3);
    Debug.Log(multiString);
}
```



```
private string BuildSensationString(string sensation, int frequency, float durationVal, int intensityVal, float rampUp, float rampDown, float exitDelayVal, string muscles)
{
    return sensation + "\","
            + "\"frequency\": " + frequency + ","
            + "\"duration\": " + durationVal + ","
            + "\"intensity\": " + intensityVal + ","
            + "\"rampup\":" + rampUp + ","
            + "\"rampdown\":" + rampDown + ","
            + "\"exitdelay\":" + exitDelayVal + ","
            + "\"Muscles\": {" + muscles;
}
```



```
// String Parts
private readonly string start = "VRC_OWO_WorldIntegration:[{ \"priority\":";
private readonly string sensationNameStart = "\"sensation\": \"";
private readonly string separator = "}},{";
private readonly string end = "}}]";
```



```
private string BuildMultiSensationString(string sensationOne, string sensationTwo, string sensationThree)
{
    if (triggerCount == 2)
    {
        return start + sensationPriority + "," + sensationNameStart + sensationOne + separator + sensationNameStart + sensationTwo + separator + sensationNameStart + sensationThree + end;
    }
    if (triggerCount == 1)
    {
        return start + sensationPriority + "," + sensationNameStart + sensationOne + separator + sensationNameStart + sensationTwo + end;
    }
    return "ERROR";
}
```

Sensations Multi-Event:

(For use linking two individual sensations together.)

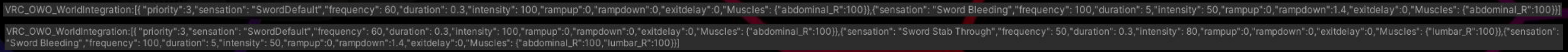
We learned about sending a single sensation on the last page, now we'll move onto multiple linked sensations.
(The script we will be pulling images from can be found at **Assets/O.W.I/Scripts/O.W.IUdon Scripts/OWISword.cs** this script is designed for either two or three sensations chained together based on the number of colliders hit to simulate being stabbed and then bleeding)

The overall structure of a linked sensation is similar to what we saw before, but now we can use an additional setting called Exit Delay which will give us time between the previous and next sensation.
(We will have multiple sets of sensation settings, the naming sense is personal preference)

We start by building our base strings based on a collider trigger count, then they are sent together to the "Multi Sensation String" method, to be put together in our final format, and finally sent to the debug log.
(Examples of the debug output can be seen at the bottom of the page, on the second example it is still a single line it is just to long to shown in the unity debug as one.)

These are the methods we have employed, you can use your own, as long as the output is correct for the external exe to read.

The important parts to note are the string parts that we are using to make sure that our output proper adheres to the proper format.
(If you are getting "Received corrupted Data" in the console log of the external exe when testing a built world that means that your debug log output has an error in it.)

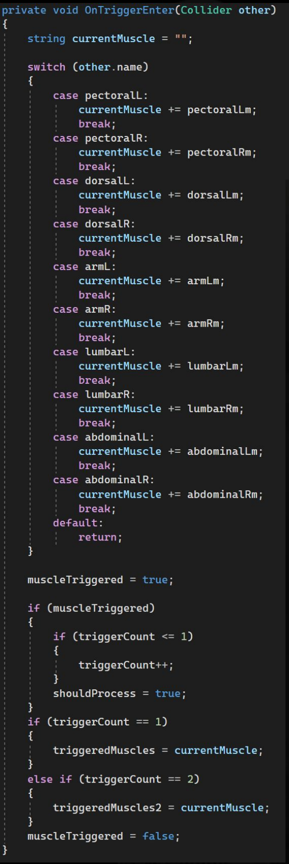


```
VRC_OWO_WorldIntegration:[{ "priority":3,"sensation": "SwordDefault","frequency": 60,"duration": 0.3,"intensity": 100,"rampup":0,"rampdown":0,"exitdelay":0,"Muscles": {"abdominal_R":100}},{"sensation": "Sword Bleeding","frequency": 100,"duration": 5,"intensity": 50,"rampup":0,"rampdown":1.4,"exitdelay":0,"Muscles": {"abdominal_R":100}}]
```

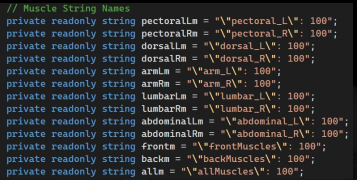
```
VRC_OWO_WorldIntegration:[{ "priority":3,"sensation": "SwordDefault","frequency": 60,"duration": 0.3,"intensity": 100,"rampup":0,"rampdown":0,"exitdelay":0,"Muscles": {"abdominal_R":100}},{"sensation": "Sword Stab Through","frequency": 50,"duration": 0.3,"intensity": 80,"rampup":0,"rampdown":0,"exitdelay":0,"Muscles": {"lumbar_R":100}},{"sensation":
"Sword Bleeding","frequency": 100,"duration": 5,"intensity": 50,"rampup":0,"rampdown":1.4,"exitdelay":0,"Muscles": {"abdominal_R":100,"lumbar_R":100}}]
```

VRC OWO System World integration.

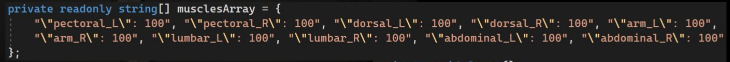
Setting Muscles:



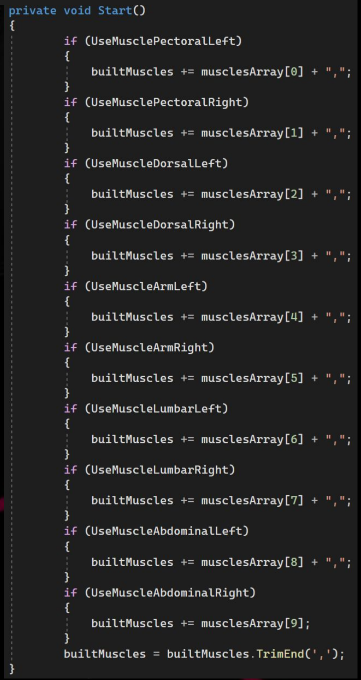
```csharp
private void OnTriggerEnter(Collider other)
{
    string currentMuscle = "";

    switch (other.name)
    {
        case pectoralL:
            currentMuscle += pectoralLm;
            break;
        case pectoralR:
            currentMuscle += pectoralRm;
            break;
        case dorsalL:
            currentMuscle += dorsalLm;
            break;
        case dorsalR:
            currentMuscle += dorsalRm;
            break;
        case armL:
            currentMuscle += armLm;
            break;
        case armR:
            currentMuscle += armRm;
            break;
        case lumbarL:
            currentMuscle += lumbarLm;
            break;
        case lumbarR:
            currentMuscle += lumbarRm;
            break;
        case abdominalL:
            currentMuscle += abdominalLm;
            break;
        case abdominalR:
            currentMuscle += abdominalRm;
            break;
        default:
            return;
    }

    muscleTriggered = true;

    if (muscleTriggered)
    {
        if (triggerCount <= 1)
        {
            triggerCount++;
        }
        shouldProcess = true;
    }
    if (triggerCount == 1)
    {
        triggeredMuscles = currentMuscle;
    }
    else if (triggerCount == 2)
    {
        triggeredMuscles2 = currentMuscle;
    }
    muscleTriggered = false;
}
```



```csharp
// Muscle String Names
private readonly string pectoralLm = "\"pectoral_L\": 100";
private readonly string pectoralRm = "\"pectoral_R\": 100";
private readonly string dorsalLm = "\"dorsal_L\": 100";
private readonly string dorsalRm = "\"dorsal_R\": 100";
private readonly string armLm = "\"arm_L\": 100";
private readonly string armRm = "\"arm_R\": 100";
private readonly string lumbarLm = "\"lumbar_L\": 100";
private readonly string lumbarRm = "\"lumbar_R\": 100";
private readonly string abdominalLm = "\"abdominal_L\": 100";
private readonly string abdominalRm = "\"abdominal_R\": 100";
private readonly string frontm = "\"frontMuscles\": 100";
private readonly string backm = "\"backMuscles\": 100";
private readonly string allm = "\"allMuscles\": 100";
```



```csharp
private readonly string[] musclesArray = {
    "\"pectoral_L\": 100", "\"pectoral_R\": 100", "\"dorsal_L\": 100", "\"dorsal_R\": 100", "\"arm_L\": 100",
    "\"arm_R\": 100", "\"lumbar_L\": 100", "\"lumbar_R\": 100", "\"abdominal_L\": 100", "\"abdominal_R\": 100"
};
```



```csharp
private void Start()
{
        if (UseMusclePectoralLeft)
        {
            builtMuscles += musclesArray[0] + ",";
        }
        if (UseMusclePectoralRight)
        {
            builtMuscles += musclesArray[1] + ",";
        }
        if (UseMuscleDorsalLeft)
        {
            builtMuscles += musclesArray[2] + ",";
        }
        if (UseMuscleDorsalRight)
        {
            builtMuscles += musclesArray[3] + ",";
        }
        if (UseMuscleArmLeft)
        {
            builtMuscles += musclesArray[4] + ",";
        }
        if (UseMuscleArmRight)
        {
            builtMuscles += musclesArray[5] + ",";
        }
        if (UseMuscleLumbarLeft)
        {
            builtMuscles += musclesArray[6] + ",";
        }
        if (UseMuscleLumbarRight)
        {
            builtMuscles += musclesArray[7] + ",";
        }
        if (UseMuscleAbdominalLeft)
        {
            builtMuscles += musclesArray[8] + ",";
        }
        if (UseMuscleAbdominalRight)
        {
            builtMuscles += musclesArray[9];
        }
        builtMuscles = builtMuscles.TrimEnd(',');
}
```

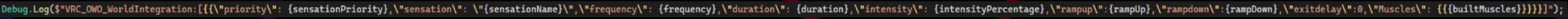
We've covered how to make sensations in either single or multi form, now let's take a look two ways to handle muscles, these are just two examples that we use already in the scripts you can handle it how you see fit as long as the format is correct.

(The OnTriggerEnter example used here is from the same script as the last page **Assets/O.W.I/Scripts/O.W.IUdon Scripts/OWISword.cs**, while the start method is from **Assets/O.W.I/Scripts/O.W.IUdon Scripts/OWIGlobalSensation.cs**)

In the OnTriggerEnter method image we are using a switch case to match the collider names to the O.W.I Avatar objects then storing them based on trigger count so we can handle them like we saw on the last page.

In the Start method image we are using bools to decide which muscles from the array we have set to add to a string variable that we will use in the output.

(The debug message at the bottom of the page shows how we are using this case.)



```csharp
Debug.Log($"VRC_OWO_WorldIntegration:[{{\"priority\": {sensationPriority},\"sensation\": \"{sensationName}\",\"frequency\": {frequency},\"duration\": {duration},\"intensity\": {intensityPercentage},\"rampup\":{rampUp},\"rampdown\":{rampDown},\"exitdelay\":0,\"Muscles\": {{{builtMuscles}}}}}]");
```

VRC OWO System World Integration. Credits

Creators:

RevoForge
SonoVR

Document writer / VRC Contributor:

BassBoostedDuck

Testers:

areno127	kiffin
minthnir	realmasterlink
mutethecyberwolf	nanoade
_nicedicer	mrdings
spokeek	baymaxxxxx
ahrivr	spike_felion
bigglesworth43	mcsolo
himynameskyle	.pepie
kasri	rumrobot

Feel free to contact us on the OWO discord server if you have any issues or bugs.