Intro

There is a new event system in place to build interactivity onto your custom players and environments. Here's a super-brief overview of how to build these events in such a way as to work in VRChat.

Note: Some knowledge of Unity is required for these features to work. In order for custom animations and animation events to be hooked up, you may need to understand the unity animation controller. Unity has excellent online documentation to learn these features and I don't have the bandwidth to duplicate those directions here. Instead, follow this link…

https://docs.unity3d.com/Documentation/Manual/

So you want an interactive VRChat!

General Interaction:

Any interactive element in VRChat needs to have a VRC_EventHandler script on it. This is the list of named events that this object will respond to, and what each event does. Add one to the "Size" element in the event list and a new event is created. Each event has a name that you'll use to trigger the event, an "EventType" (what happens when the event is triggered) and a few parameters. There is also a NetworkId field. Just don't touch that one. It should populate itself with a big number that uniquely identifies this object within a scene.

Event Types:

MeshVisibility:

This will enable or disable a mesh object that is part of this scene element.

Parameters:

ParameterObject – the game object that contains the mesh renderer.

ParameterBool – whether the mesh should be visible after this event or not

Requirements: Only works on skinned mesh renderer and mesh renderer objects that exist under the VRC_EventHandler in the object. The Game Object holding the Mesh must be a child of the Game Object holding the event handler.

AnimationFloat:

This will set a float variable in the game object's animation controller. This can be used to play custom animations or change animation behaviour.

Parameters:

ParameterString: the name of the animation controller float variable

ParameterFloat: the new value of the variable.

Requirements: The Animator Component must be on the same GameObject with the EventHandler

AnimationBool:

This will set a boolean variable in the game object's animation controller. This can be used to play custom animations or change animation behaviour.

Parameters:

ParameterString: the name of the animation controller bool variable

ParameterBool: the new value of the variable.

Requirements: The Animator Component must be on the same GameObject with the EventHandler

AnimationTrigger:

This will trigger a change in the game object's animation controller. This can be used to play custom animations or change animation behaviour.

Parameters:

ParameterString: the name of the animation controller trigger

Requirements: The Animator Component must be on the same GameObject with the EventHandler

AudioTrigger:

This will cause an AudioSource under this game object to play it's sound.

Parameters:

ParameterObject: the game object that contains the AudioSource object

Requirements: Only works on AudioSource objects that exist under the VRC_EventHandler in the object.

PlayAnimation:

This will cause an animation to play using the Unity legacy animation system. This is handy for animations you create in the unity editor on environment objects such as doors or elevators

Parameters:

ParameterString: the name of the animation to play

Requirements: The Animation (Not Animator) Component must be on the same GameObject with the EventHandler

SetParticlePlaying:

This will cause a particle effect to run on the destination object's ParticleSystem.

Parameters:

ParameterBool: Whether the animation should play or stop.

True: call Play on the ParticleSystem

False: call Stop on the ParticleSystem

Requirements: The ParticleSystem Component must be a child of the GameObject with the EventHandler

SendMessage:

This will send a general message to another game object that is a child of this game object. Specific messages are handled by other SDK scripts that are required to handle these messages.

Parameters:

ParameterString: the name of the message. This translates to a unity function call on the destination object.

Requirements: The Message must be passed to an object that is a child of the Game Object with the message handler.

Teleport Player:

This will send a player to a specific location. It can currently only be triggered by Use and Collision event types, as these events have a clear instigator to their action.

Parameters:

ParameterObject: A child of the object that contains the Event Handler. This object is placed at the location and orientation that the player should appear in.

Requirements: The object where the player should appear should have the Y axis pointed upwards in the normal fassion. Otherwise the player might get messed up on teleport.

RunConsoleCommand:

This will send a console command to the player that generates a specific event. This message can be any string that the console command system can parse, such as "/SetAvatar …" or "/Pad". This is very useful for making room exits, or mannequins that a user may click on to adopt that avatar as their own.

Parameters:

ParameterString: The full console command to be executed on the event instigator's system.

Requirements: Only the Use and Collision events may cause console commands to run, and console commands have no effect unless run in the real VRChat environment.

Add Player Mods:

This allows you to grant specific players certain modifications, such as a jumping and walk/run speeds using an attached VRC_PlayerMods script. Mods granted via this event will overwrite any equivalent mods applied by the room.

Parameters:

None

Requirements: The VRC_PlayerMods script must be on the same game object as the EventHandler. Also make sure the "Room Player Mods" flag on the VRC_PlayerMods script is unchecked (set to false). Note: It is okay to have multiple VRC_PlayerMods scripts in a scene, but only one should have the "Room Player Mods" flag set to true.

Remove Player Mods:

Removes any player mods granted to a player via an Add Player Mod event. If there are any player mods associated with the entire room, these will be reapplied to the player.

Parameters:

None

Message Recievers:

Here is a list of the supplied scripts that can be used with the SendMessage event. As a reminder, the object receiving the message must be a child of the object with the Event Handler.

VRC_SceneSmoothShift:

This script receives a message called "Shift" which causes it to move between two predesigned positions. This is good for sliding or rotating doors, elevators or sliding platforms.

Interactive Environments:

Add an element to your scene that you would like to be interactive. For this sample, I will talk about adding an interactive radio. If you have a radio model, go ahead and use it, otherwise just drop in a cube that's about the right size. If you use your own model, make sure you add a collider so that the radio can be interacted with.

1. Add the mandatory VRC_EventHandler script.
2. Add the VRC_UseEvents script. Default values on this script will allow you to use the radio from 2 meters away by looking at it and pressing the E key. By default this will call an event called "Use"

3. Add an AudioSource object to your cube and add any audio file to it. Disable “PlayOnAwake”.
4. Go back to your VRC_EventHandler.
5. Increase the size of the event list to 1
6. Name your first event “Use” to match up with the event generated by the VRC_UseEvents script.
7. Select the “AudioTrigger” event type.
8. Drag the game object containing the AudioSource object onto the “ParameterObject” field in the event.
9. This element should then begin playing the sound effect for everyone in the room when someone activates the radio.

Besides “Use” there are other scripts you could use in your environment, such as the VRC_TimedEvents (triggered every so often) or VRC_CollisionTriggerEvents (triggered when you enter or exit a specific area).

VRC_UseEvents

@description - Allows users to fire events by pressing the interact button while looking at the VRC_UseEvent game object. User must be a minimum of 2m away from the object to work.

@properties

Show Use Text – If true, will create a text object with the text “Use” at the position marked by the Use Text Placement property.

Event Name – Should always be “Use”.

Use Text Placement - The transform where you want the “Use” text to appear around the game object. Can leave empty if you don’t want the word “Use” to appear when a user looks at the VRC_UseEvents game object.

Controlling your Avatar:

The other thing many people are interested in is having greater control of their avatar.

Events are added to an avatar in a similar way to environments, but I’ll highlight other methods of triggering them.

Timed events still work. This could allow you to have your character blink occasionally by creating a blink animation and causing a timed event to trigger it.

Beyond that we also have VRC_KeyEvents which will generate events based upon your keyboard input. For example you could have your avatar play a specific animation whenever you press the “J” key.

One of the most powerful features though is the animation event system. If you have a set of custom animations for your avatar you can add events to those animations in unity. To make these events work you'll need to send all your animation events to the VrcAnimationEvent method in the event handler. The only parameter your animation event will have is the name of the event in your event list which is generated. This should allow you to add audio, such as foot steps to your avatar animation, or to add props during specific animations. You could have a sword appear in your hand when you play the "unsheathe" animation for example.

Player Modifications:

A recent addition to the SDK allows you to grant player modifications to every player in the room or to individual players using the event system. Current modifications include player speeds, jump and voice (toggle to talk/talk distance).

You'll be using the VRC_PlayerMods script to do this. The script's "Room Player Mod?" flag should be set to true if you want the mods applied to every player who joins the room – there should only be one VRC_PlayerMod component w/ this flag set per scene. The "Add Mod" button is used to add a specific mod. Once a mod is added, you can adjust its parameter values.

Room Player Modifications

To add player mods to every player who joins your room, add the VRC_PlayerMods script to the root object of your custom room and check the "Room Player Mods?" checkbox (set it to true). Next, click on "Add Mod" in the VRC_PlayerMod Inspector pane. A window will pop up prompting you to choose a mod to add. You can have as many mods in a room as you'd like. Once you've added a mod to the VRC_PlayerMod Inspector pane, feel free to edit the parameters on each mod.

Individual Player Modifications

It is also possible to apply player mods to specific players in a room, using the normal event system. Please take a look at the Add Player Mods and Remove Player Mods event type docs above. Note: It is okay to have multiple VRC_PlayerMods scripts in a scene, but only one should have the "Room Player Mods" flag set to true.

VRC_Station

A station is a game object that a user can interact with in some way. A basic example would be a chair. By using the VRC_Station script, you can give users the ability to sit in a chair.

To create a station simply add the VRC_Station component to the station game object

a. Check the "Should Immobilize Player" box if the user shouldn't be able to move when they use the station

b. Add your custom animator controller to the Animator Controller parameter. This is the controller that will be applied when a player uses your station

c. Add a transform to the Station Player Location parameter if you want to move the player to a specific position/rotation when they use your event

That's it! You now have a functioning station. You can add a little bit more functionality if you want.

1. **Add your own Use Interactive collider.** By default, a box collider trigger is added to the game object with the VRC_Station script attached. This collider is what's used to determine where the user can look into order to interact with the station. To override this, simply add your own collider on the game object containing the VRC_Station script. Best to make the collider a trigger, but it's not necessary.

2**. Add the text "Use" above the station when a player looks at it**. By default, the VRC_Station adds a VRC_UseEvents component to the station gameobject at runtime w/ the default settings. The "Use" text is turned off by default. In order to override this, simply add your own VRC_UseEvents script to the VRC_Station gameobject and check the "Show Use Text" checkbox and add a transform to the Use Text Placement field.

That's it! In order to test, you must run inside the VRChat application.