

PIVOTE

Introduction to PIVOTE

11 Feb 09

This document provides an introduction to the PIVOTE system for creating immersive training and education exercises in Second Life, other virtual worlds.

PIVOTE offers 5 main advantages over creating a training exercise “natively” within a virtual world (e.g. In Second Life using Linden Scripting Language):

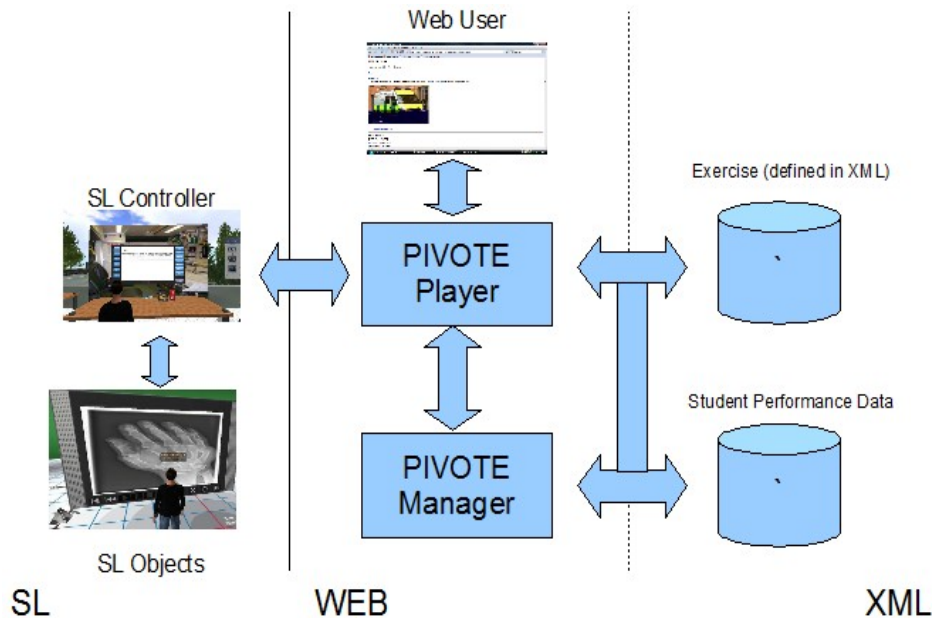
- Since structure and content is defined and maintained on the web, tutors do not need to worry about the detail of a virtual worlds building and programming, but can instead concentrate on the learning exercise
- Since the structure and content are on the web the exercise could potentially be played in multiple virtual worlds (Second Life, OpenSim, Forterra, Wonderland etc) at the same time – protecting content from changes in virtual world fashion
- The exercises can be played from the web if no virtual world is available (and useful for development and testing as well)
- The same exercise can be played in different flavours – e.g. With lots of hints and guidance when being used for teaching, but no hints or guidance when being used for assessment
- Objects can be readily shared between exercises, and even between institutions

PIVOTE was originally developed as part of the UK Government JISC funded PREVIEW project, and is now open-source under a GPL v3 licence.

[Note: Whilst much of the description below refers to Second Life this is only because this was the first target virtual world for PIVOTE. We, and hopefully others, will be working on developing PIVOTE controller interfaces for other virtual worlds.]

1. Overview

An outline of the PIVOTE system is shown in Figure 1.

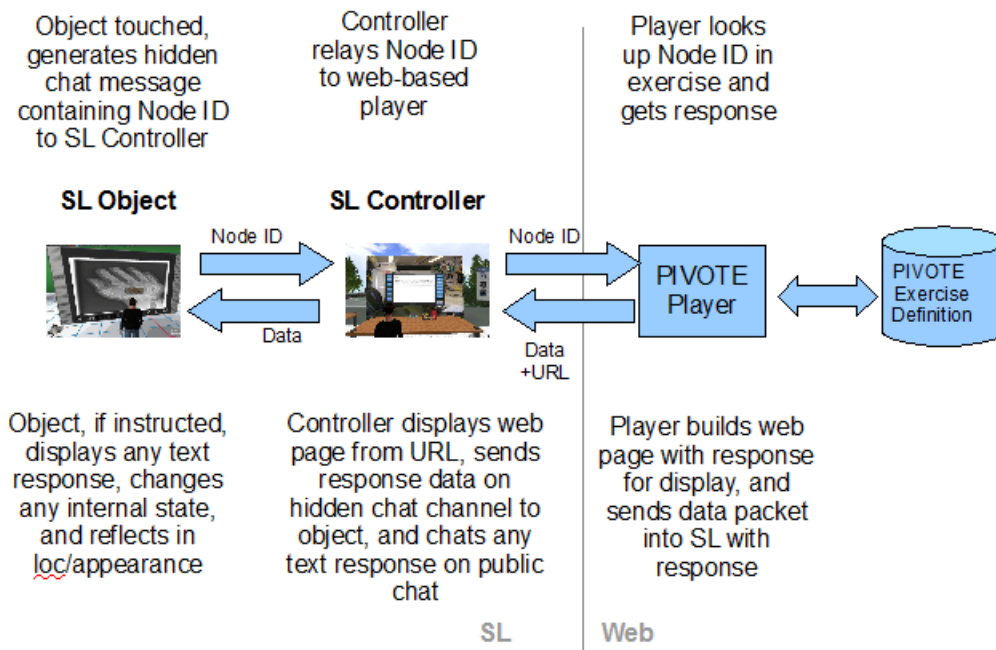


The main components are:

- Objects in Second Life running PIVOTE scripts
- The Second Life PIVOTE controller (a HUD remote is also available)
- The web based PIVOTE player
- The web based PIVOTE manager – including an editor and performance data viewer

It is assumed that you have access to the web based systems and that these have already been installed and configured. For guidance on installation see the separate Installing PIVOTE document.

The main way that the system works is by linking actions on objects in SL, to descriptions of results and further options stored on the web in the PIVOTE system, but displayed back in SL. In PIVOTE each action on an SL object is linked to an “activity node” in PIVOTE. So for instance touching an object called “mouth” in Second Life might result in the user in SL being told about the state of the mouth, and even being shown an image of it, and then being offered options to learn more about the mouth or to touch something else. All that should ideally exist in SL is objects with node identities, with all the content and logic of the exercise coming from the PIVOTE system on the web, as in Figure 2.



2. Planning the Exercise

Annex A describes the approach to planning a PIVOTE exercise. The key needs are to:

- Map the exercise in terms of the lowest level tasks called “activity nodes”
- Assign a meaningful ID to each activity node
- Identify the information that each node will provide, including text, images, audio, web, video etc
- Decide any rules that will be needed where a node may show different data at different times
- Decide any counters needed in order to implement the rules

3. Creating the SL Environment

This is probably the simplest part of the process, just build in SL the environment in which the exercise will take place. This may be a classroom, city street, jungle, lab, spaceship etc. Currently each “exercise” must have 20m chat separation from any other PIVOTE exercise, so use your sim space accordingly. Also each exercise must take place within 10m of the controller (although this could be increased to a whole sim by some custom coding in SL).

4. Creating the SL Objects

You will need to create or obtain each object which plays an active role in the exercise – i.e. touching it will activate a node. Every active object will require a PIVOTE script (standard scripts can be downloaded from the PIVOTE web site or obtained in-world). Objects (or rather scripts in objects) are of two types:

- Simple scripts, where touching the object causes the object description (where the node ID is stored) to be read and that node to be activated)
- Complex scripts, where touching the object (or another action – e.g. Chat) may result in a further SL dialog before a node ID is sent.

A very simple script showing the minimal code needed to link an object into PIVOTE is at Annex B.

Although creating the objects will take time you should find 2 benefits:

- Once the objects for a particular exercise are created it is very easy to create variations on the scenario without creating new objects
- With sensible node naming you can use the objects from different exercises to create a whole new exercise

For ease of distribution you can place all objects in a box in SL and configure the SL Controller to make this box available to users.

5. Creating the Exercise Files

The key activity is in creating the files which drive the PIVOTE exercise. Luckily the PIVOTE editor can completely hide the underlying code (XML) and file structure (see below) from you, allowing you to create complex exercises just by filling out a set of web based forms.

Key to the structure of a PIVOTE exercise is the 3-tiered approach to information:

- Activity Nodes which define the flow through the exercise in terms of the lowest level activities in the exercise
- Data Availability Nodes which define what information is available at each node (but only hold pointers to the information, not the information itself)
- Asset nodes (both text and media – including images, movies and sound) which are called on by the Data Availability Nodes for each Activity Node.

By having this separation between structure (the Activity Nodes) and content (the Asset Nodes) we can easily create variations on an exercise with maximal re-use, and we can create different flows through an exercise for different students or different situations (training or assessment).

We also have available a set of counters (effectively variables) which we can change during the exercise, and a set of rules which can vary flow or content based on counter values and previous nodes visited.

The PIVOTE Editor is described in a separate document. As well as the forms based editor it also allows you to edit the underlying XML of an exercise – although this isn't normally necessary (and you need to be familiar with the underlying Medbiquitous Virtual Patient standard (www.medbiq.org) if you do).

6. Configuring the Controller

The controller requires minimal configuration, but can be significantly customised. All configuration is done by an SL notecard. Details of parameters are at Annex C. You can configure each controller that you use the same, or each differently.

Since the controller uses the parcel media stream in SL you need to ensure that either:

- The Controller is owned by the same avatar as the parcel, or deeded to Group if the land is group owned (although this can stop you from editing the controller)
- You use the free media relay, and the media relay is either owned by the same avatar as the parcel, or deeded to Group if the land is group owned

The controller screen must also be set to the same media texture as the parcel – it may be able to do this automatically, otherwise you'll need to set it manually.

Since the controller is COPY-MOD-TRANS you can edit the controller to suit your own training environment.

There is also a HUD version of the controller which provides slaves of the screen and the controls so students do not need to stay near the controller during an exercise (although in an ideal exercise the students need never see the controller or HUD as all interaction should be through objects in the environment).

7. Testing the System

You should now be able to test the system. We recommend that testing is done as follows:

- First test using the web interface. This will let you test the logic of the exercise without worrying about SL objects and issues
- Then test from SL using the SL objects

Once you have completed your own testing you should then get colleagues to test, and then “tame” students, before finally using in a live environment. At each stage you should refine and improve your exercise and objects.

8. Performance Tracking

When a user undertakes an exercise, all of their performance data is tracked and stored on the system. This includes:

- Name of avatar that started exercise
- Any tutor ID
- Controller name
- Location of controller
- Name of exercise
- Date/time of exercise start
- For each node visited the time that node was entered
- The final value of any counters

Tutors can export and delete student records as required.

Record lists can be filtered by exercise/date/tutor ID/location.

9. Further Information

For more information (and videos) on PIVOTE please visit the web site at www.pivot.info.

Planning The Exercise

Before using any of the systems you will need to plan your exercise. PIVOTE works on the basis of activity nodes. These are the smallest unit of task/activity that you will be asking the student to do – and in SL terms relate to an action on an object. Typical nodes might be:

- Open door
- Switch on device
- Touch something

The initial node map is likely to have a fair degree of structure as you think through the exercise in a logical way. It may also be that the exercise breaks down into a number of sub-tasks, which in turn comprise individual activities. Note that if the same action is done more than once, but at separate stages in the process it counts AS THE SAME NODE in PIVOTE, but you need to decide what condition you will track in order that the node knows to behave differently at each time (this might be a real condition such as whether device is open or closed, or an artificial condition such as whether a certain node has already been visited).

You should also identify which nodes link to which others. In the final implementation such structure may well be removed (the real world doesn't have lines from one activity to another), but you will need to identify dependencies, and the other links can be useful for more training (rather than testing) orientated sessions.

At the end of this exercise you should have a node network that shows the activity nodes you need, and the linkages between them, as at Figure 2.



Each Node is assigned an ID. Whilst this ID could be anything, it is often best if it reflects the object and action which triggers it – for instance mouth_touch.

For each node you will then need to decide what information (called assets) is provided to the user when the node is activated. This will usually include some text which is “chatted” and displayed to the user, but could also include:

- still image
- audio or video
- link to web page
- object movement
- object action (eg changing shape, disappearing etc)

Where the node can be triggered at different times to give different results then you need to identify all the assets, and what the conditions are that will be used to select each. These conditions (other than previous nodes being visited) must be tracked by “counters”, and the rules themselves are stored in a the “data availability node” attached to each activity node. The data availability node links an activity node to a set of response assets.

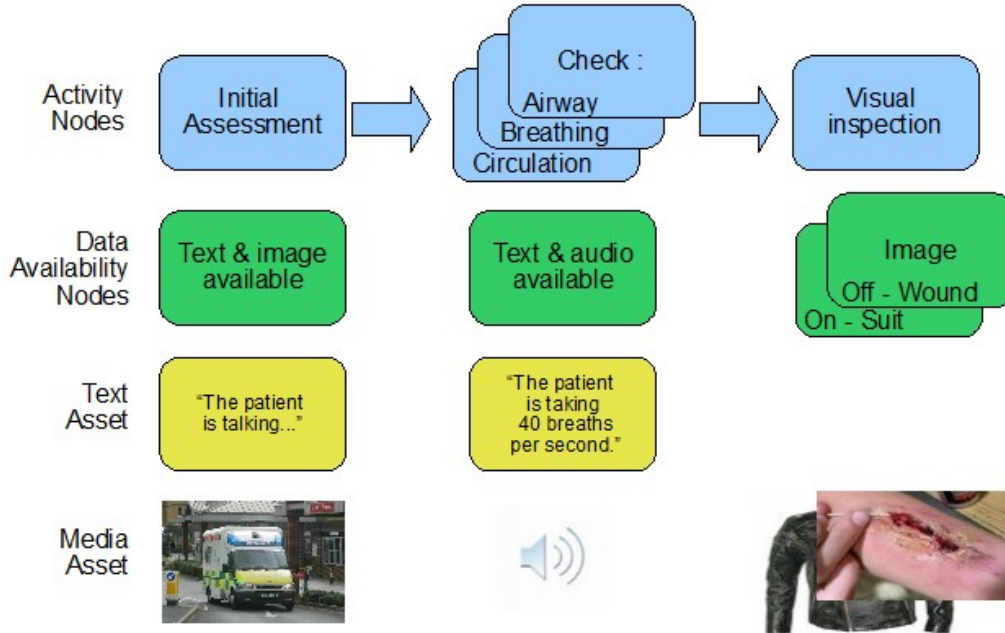
At this stage it is useful to create a simple table like that below, which identifies each node and its related information. This will make the creation process in the editor a lot simpler (particularly since the editor has a wizard which lets you create a whole Activity Node – Availability Node – Asset Node set in one keypress.

SL Object Trigger	Activity Node	DAM Node	Asset Nodes
[Case Defaults]	Counters: Clean Hands: 0 (false) Alarm Sounding: 0 (false) Catheter Blocked: 1 (true)		
Start Button	Start	Start	Outside the Training Centre You are given a brief description of the task. It is morning and you are due to start your shift.
Locker or wear outfit with attachment	Get Changed	getchanged	You change into your uniform
Wash basin	Wash Hands Set Clean hands counter	washhands	You wash your hands
[redirect]	Remember	rememberwash	Remember to wash your hands first
Volume detect on door	Enter Ward Conditional – Redirect to remember to Wash Hands if Wash Hands not visited	enterward	You have entered the ward
Click on nurse primatar at nurses station	Briefing Clear Clean hands counter	handover	You learn that patient in Bay 1 has called twice but night nurse has not had time to investigate. You are told that the is on a morphine drip. Audio: Recording of breifing Image: Image of handover notes

PIVOTE supports multiple linking so any activity node can link to one or more of any data availability nodes, which can in turn link to one or more of any response asset.

This should give you enough information to start creating your PIVOTE exercise, although it is likely to be an iterative process with more nodes and assets being required as you develop and test the exercise.

The PIVOTE node hierarchy is shown below:



PIVOTE LSL OBJECT SCRIPT

Place this script in any object to make it trigger a node in a PIVOTE exercise.

For live deployment you need to check that *pivotech* is the same as that being used by the controller, and whether a channel prefix has been assigned.

For this script the node ID is taken from the object's Description field – this means you can use this same script in multiple objects without making any changes to it.

It is also good practice to include some housekeeping – for instance having the object die or return to a set location when the controller is reset or an exercise starts or ends.

```
// set the channel used to talk to the controller
integer gPivoteCh = 687686;

default
{
    state_entry()
    {
        // nothing to set up
    }
    touch_start(integer i)
    {
        // just say the object description as its node name when touched
        // this could be llRegionSay for greater range, but you'd then
        // need to use channel prefixes to separate multiple PIVOTE
        // exercises on the same sim
        llSay(gPivoteCh, "node="+llGetObjectDesc());
    }
}
```

SL CONTROLLER PARAMETERS

The values that you need to set/review in the Controller notecard are:

Variable	Description	Example/Default	Comment
gExerciseList	List of exercises that the Player will show	about_dogs,about_cats	If only one then will start immediately on that if START button pressed If blank then lists all exercises available on the PIVOTE server
gQuickStart	Whether to show single exercise option	True (default)	If True then starts exercise when START pressed and only one exercise. If False then first shows option screen showing single exercise
gIdleURL	Texture/URL of image to show when Controller is waiting for start to be pressed	http://www.test.com/start.jpg	If doesn't start with http it is interpreted as a texture name in SL which must be in the object
gServiceURL	URL of PIVOTE server	http://www.test.com/pivote.pl	This may be a public or private PIVOTE server
gServiceUID	User ID for PIVOTE server	fred	Public servers may require a user ID and password
gServicePwd	Password for PIVOTE server	pass123	Public servers may require a user ID and password
gShowHUD	Whether to enable/show HUD button	True or false	
gShowObjects	Whether to enable/show Objects button	True or false	
gObjectBox	Name of box given to user when Objects button pressed	paramedic_kit	Must be an object in the Controller inventory
gUserLock	Whether to lock device to avatar who presses start	True or false	If True then only user who presses the Start button for an exercise can use the system until another user presses start
gPIVOTEChannel	Channel used for device chat	-123	MUST also be set in the objects
gPIVOTEPrefix	Prefix appended to any device chat	Ex1:	Used to allow multiple exercises/controllers to operate within chat range of each other
gTutorGroupID	Tutor/tutor group ID stored with each session	Fredblogs_yr7a	This is stored in the user tracking files to aid retrieval later
gControllerName	Name of controller	Station1	This is stored in the user tracking files to aid retrieval later
gCustomMVP	Any initial value for a counter to over-ride than in the XML, or to rename an exercise	<name>paramedic</name><exercise>paramedic_gen</exercise><counter><name>givehints</name><value>1</value>	Eg used to switch an exercise from "learn" to "test"