# Meshing About – A Mesh Alpha Memory Object

Prepared by Whimsical Aristocrat.
Contact me at: whimsical.aristocrat@gmail.com
Blog: scallivanting.wordpress.com

## Table of Contents

## Introduction:

This is a short paper outlining a system for storing and retrieving the alpha information for use with clothing items, and clothing items combined into outfits, for users of mesh bodies in second life.

I have no interest in making money or maintaining a business in second life, so I am putting this solution in the public domain in the hope that the mesh body makers will pick up the ideas and use them to provide us with a much better user experience when using their products.

The subject matter is clothing and mesh bodies, however it is necessarily technical, and includes elements of (simple) mathematics, primitive object parameters and scripting. Still – I hope I have presented it clearly enough that readers can follow the discussion and understand the solution I am putting forward.

I have also included a graphical section just to give some visual sense of how this works, since it actually sounds more complicated than it really is when you put it into words (maybe I'm not so good at words!).

I have created a simple demonstration system in-world, which will hold the full set of scripts and objects required ini the memory, and a very simplified version of a body HUD, and I am very happy to provide these to anyone who is interested in taking this solution further.

## The Problem:

To reliably store, and then reload, the alpha masks information used by a mesh body for a particular item of clothing.

## The Solution:

Add an object to each clothing item (or alternatively, each outfit – see later discussion) that stores the alpha memory for the clothing item. Provide a means of creating or modifying the alpha memory for the clothing item using the mesh body HUD. Provide a means of retrieving the alpha memory for the clothing item back into the mesh body HUD.

## Some terms:

I have defined a few terms just so it is clear what I am talking about. I am sure the developers of each mesh body and HUD system have their own terms for these things – hopefully these definitions will help keep the descriptions here clear.

*Mesh Body* – is the Mesh body we receive from the vendor.

*Mesh Body HUD* – is the HUD we receive from the vendor to control the body.

*Alpha Segment* – is a single section of the mesh body that can be made transparent (or not) under control of the Mesh Body HUD.

*Alpha Map* – is the complete collection of all of the Alpha Segments and their states, represented as information.

*HUD Alpha Display* – is the picture or collection of buttons, which display the current alpha map, typically as a jigsaw puzzle of segments making up the shape of the body.

*Alpha Memory* – is the term I am using to describe the object that stores the alpha information for an item of clothing. In the world of mesh bodies, the Alpha Memory serves the same role as the Alpha Layer (those little white shirts in our mesh item clothing folders) for classic avatars.

*Item Mask* – is the term I am using for a mask that represents the maximum number of alpha segments that could be required by a particular item of clothing. For example, a crop top shirt should not normally be expected to set an alpha bit in a leg. The item mask is used in the alpha memory to the alpha data before sending it to the HUD, so that the HUD only receives sensible information. To facilitate this, a process for saving item masks is defined – see later.

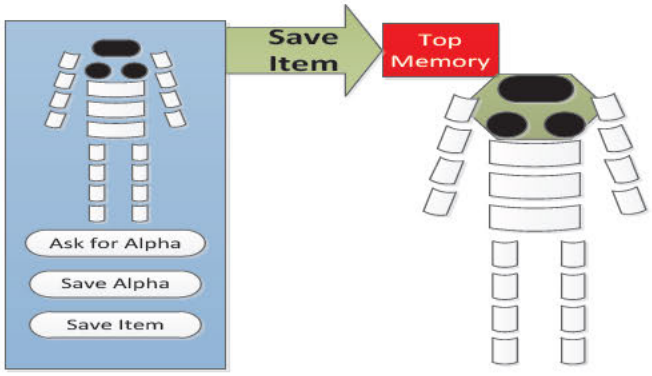
*Alpha Mask* – is the term I am using for a mask that represents the actual alpha pattern (segments turned on or off) for the clothing item. Often we don't want or need all possible segments on, for example the tops of legs under skirts, so long as they were not poking through.

Because we might want to tweak the alpha while wearing the item, we could decide to save the alpha information – even though we are wearing 3 or 4 items of clothing with alpha memories attached. Doing so will store the full update alpha values for all parts of the body in all of the attached memory's. This is why we have a stored Item Mask as well, so that we can return to the HUD only those alpha segments associated with out clothing item.

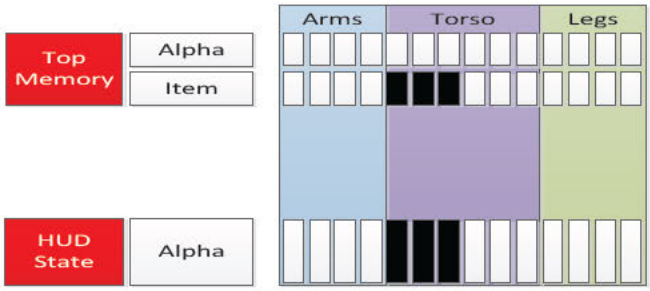
## Thinking in Pictures

Hopefully this will make it easier to follow – pictures are definitely worth those thousand words! (Adding words as well). These are the worst-case processes. A best-case process is a single alpha memory saved with an outfit.

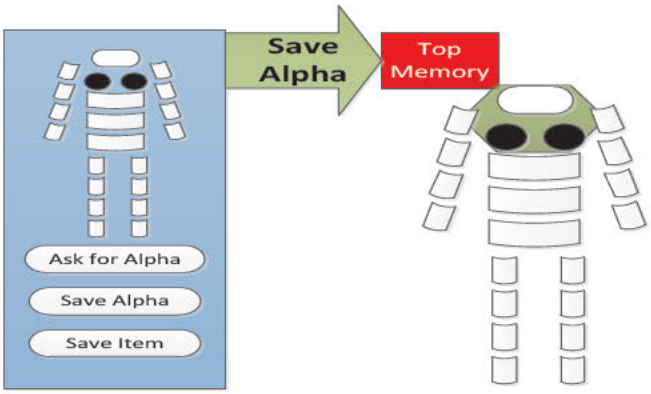
### Making Memories – Items and Alphas for Alice's Top



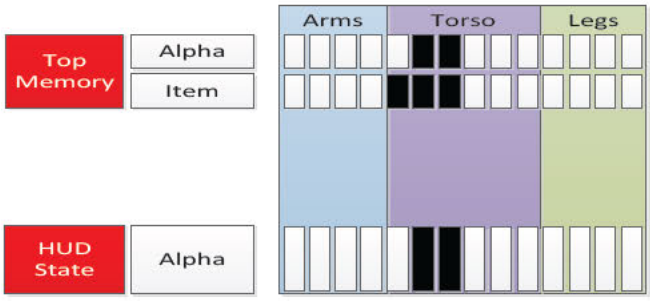

Alice is making an outfit – first she wears a Top, and also she adds the Alpha memory she wants to make for her Top.




Alice uses the body HUD to select all the alpha segments that are under her top, and then she presses "Save Item" on the HUD. The HUD sends the item pattern to Alice's Top memory, which saves it.

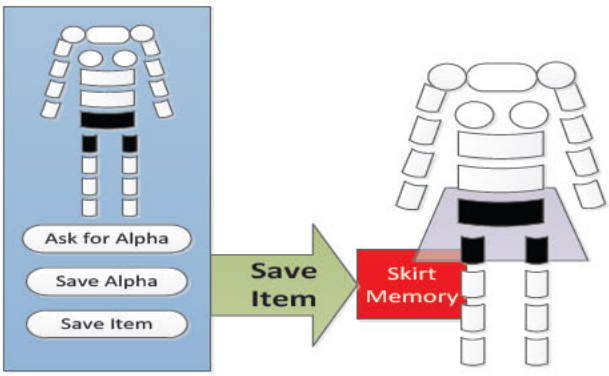



Alice then notices that she does not need the neck area alpha as the top isn't showing through, and it looks better without, so she unclicks that segment on the HUD, and selects "Save Alpha".

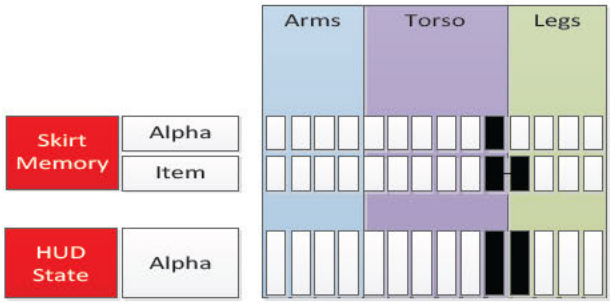



The HUD sends the Alpha information to her Top's memory, so her memory now looks like this.

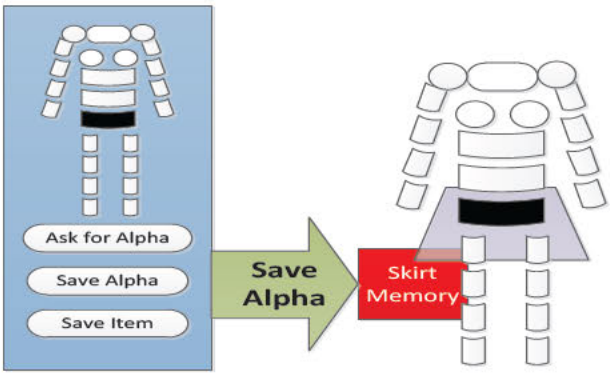
## Making Memories – Items and Alphas for Alice's Skirt




Next, Alice wants to make an alpha for her Skirt, so she repeats the procedure, after taking off her top (She can only save one item at a time) Alice wears her skirt and the skirt memory object from her skirt folder.

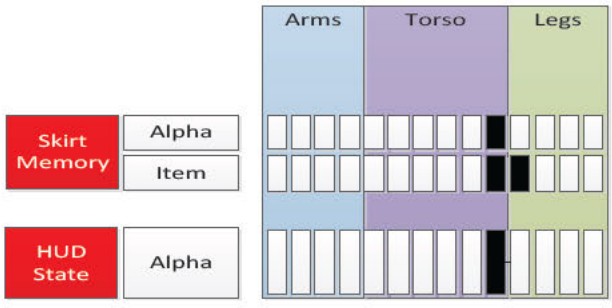



Alice sets all the parts that are under her skirt, and then selects the "Save Item" button to send the item pattern to her Skirt Memory.




Next, Alice clicks the tops of her legs to turn off that alpha. It's a flared skirt and even though they are under the skirt, they never show through.

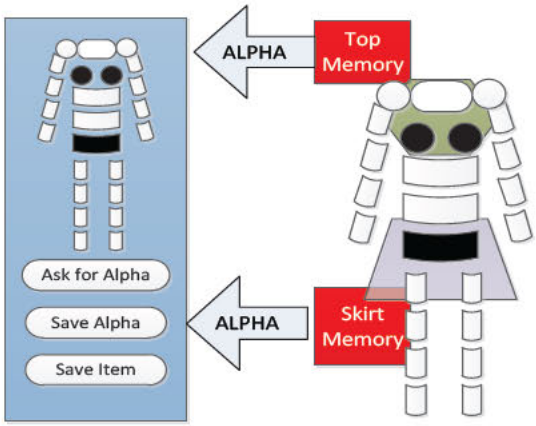
She saves the alpha pattern to her skirt memory by pressing "Save Alpha".




Her skirt memory now holds an item pattern and an alpha pattern, and the body HUD has the alpha pattern.

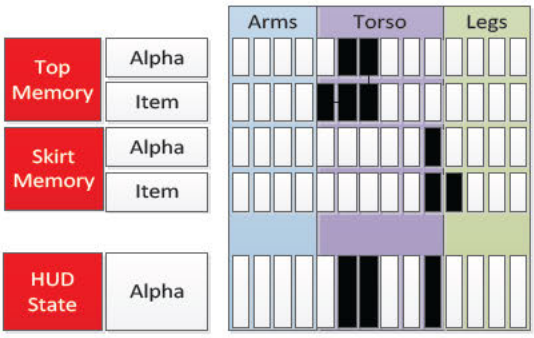
Alice removes that outfit, and the two alpha memories, for her top and her skirt, stay in the folders with those items of clothing.

## Wearing an Outfit – Memories get attached




Later, Alice wants to wear the new outfit. She selects "Show all" to turn off all the alpha bits in her body HUD.

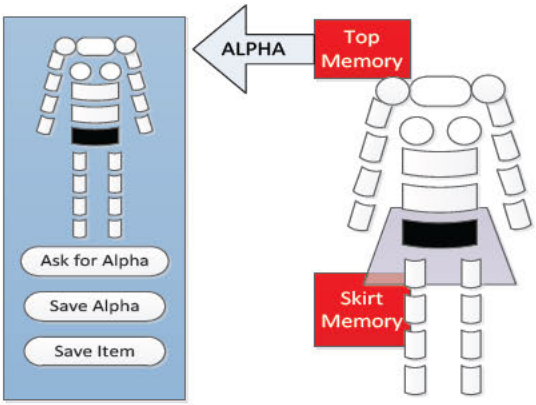
Alice now puts her top and her skirt back on, and she adds the top and skirt memory items as well.




When each memory item is attached, it sends the alpha pattern for the area within its item pattern.

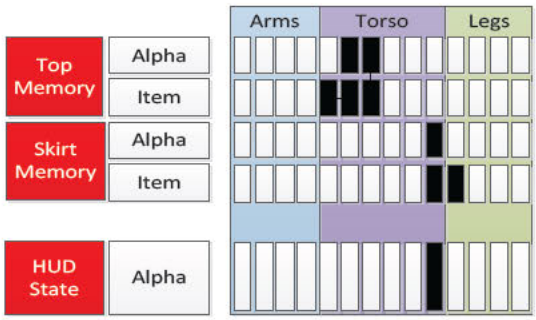
The HUD turns the alpha parts on for her automatically when it receives the alpha information from the clothing memories.

### And Detached!




Later, Alice is trying on other tops. She detaches her top and detaches the top memory.

When the top memory see's that it is detaching, it again sends the alpha pattern to the HUD, and tells it that it is detaching now.

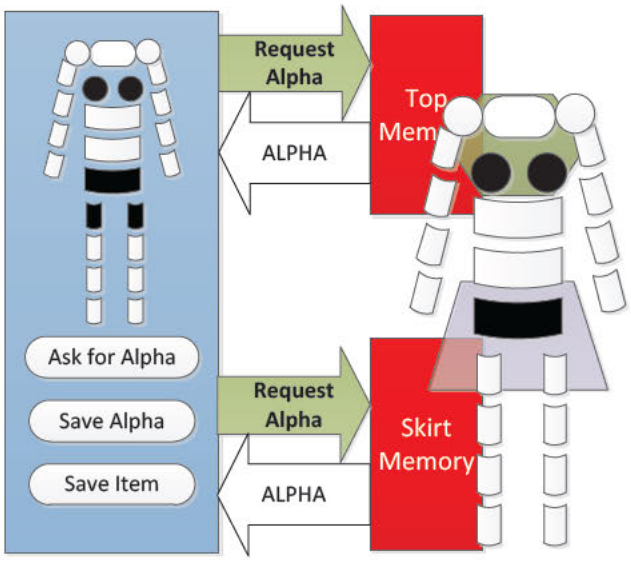



The HUD cleverly turns off only the same alpha bits that the alpha memory told it are for the top that is now detached.

The top memory is not changed. And the HUD still has the skirt alpha pattern intact.

### That's weird – I better get an Update!

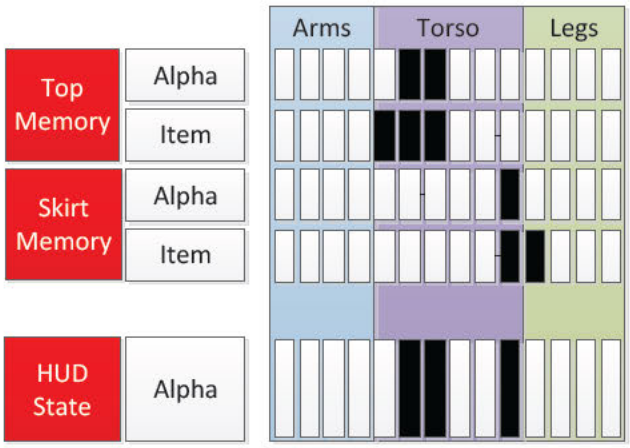
So – there may be cases where two items overlapped, and a detaching top turns off the alpha that the skirt wanted to be left on. Or maybe the Sim crashed and the alpha patterns look like a zebra crossing!




In this case, Alice can press the "Ask for Alpha" button, and the HUD will get the Alpha info from any attached alpha memories.

It is possible that a HUD designer just does that after it receives a detach message from an alpha memory, saving Alice the bother.

Either approach will work.




The Item mask in the memory tells it which bits of alpha to send in each case. This allows a save alpha to be done at any time after the item has been made, without risk that the top alpha will ever try to set the skirt alpha bits.

That's required, because the top might be used with different skirts or pants that have different alpha needs. So by creating the item masks in the memory, we deal with that by ensuring each item of clothing only sends its own alpha areas to the HUD, even is it ends up storing other alpha information as well.

## Design Goals

To be practical and useful, the Alpha Memory must have certain characteristics that address the following issues (there may be more – I'd love to hear your thoughts).

1. *Clothing items should only send alpha bits to the mesh body HUD for areas of the body, which that clothing item covers.*

So – imagine your outfit contains

a. A Short sleeved shirt
b. A skirt
c. A pair of prim gloves

Each of these clothing items will have an Alpha Memory (AM) in the clothing folder, and the user will wear the AM when they wear the clothing item. This is a similar idea to wearing of alpha layers supplied with clothing for classic avatars.

So each item of clothing, via its AM, sends to the mesh body the pattern of the alpha mask to use within the boundaries of the portion of the mesh body the clothing item covers.

The mesh body will take each of these masks and apply them, such that if ANY item of clothing's AM sets a part of the body alpha map then that part will set in the HUD Alpha Display.

This allows the AM for a clothing item to be used in multiple outfits, or for a single AM to be created that is specific to a complete outfit (and so be stored in the outfits folder rather than the clothing folder).

2. *The design should be simple, flexible and low impact (scripts and attachments).*

There are 38 attachments allowed per avatar (including HUDS, clothes and gadgets) – so the design should not add an unnecessarily high number of attachments to the avatar. Similarly, each running script in an attached object takes simulator memory (16k minimum unless the script writer requests a lower amount), and each running script potentially adds to lag.

The discussion above, where the idea of allowing a user to create their own AM for a complete outfit (rather than wearing a separate AM for each clothing item), using the existing facilities of the Mesh Body HUD, addresses this issue.

3. *The Alpha Memory should only respond to my mesh body HUD, and the mesh body HUD should only talk to my Alpha Memory.*

An obvious one – but I don't want my friends to disappear when I wear my full body cat suit!

This is basic scripting and the ability to discriminate between messages from different users is an in-built part of LSL message script functions, so it's not a biggie.

4. *An avatar using the Alpha Memory system should be able to easily create an Alpha Memory object for an outfit.*

I believe this ability should be supported both manually (by editing the AM) and be supported by a simple user control on the AM itself, visible when rezzed.

By making this easy to do, the AM will help achieve the objectives of issue 2 above.

## Design of the Alpha Memory

The complete Alpha Memory system includes

1. The *Alpha Memory Object*, which provides read/write storage for alpha information.

2. Some additional capabilities in the *Mesh Body HUD* so that it knows how to read, write, and make use of the alpha information stored in the Alpha Memory.

### Alpha Memory Object

The following is a complete design, along with discussion, for the construction of an Alpha Memory object that can be used with a suitably enhanced mesh body HUD. Please see the next section for a description of the mesh body HUD enhancements that will be required.

- Stores a 128-bit **item mask** that defines the boundaries of the clothing item in terms of the mesh body alpha segments. See the sizing discussion below.

- Stores a 128-bit **alpha mask** that defines the state of each alpha segment. See the sizing discussion below.

- Provides a means for **receiving the alpha mask** from the mesh body HUD.

The alpha mask is received as a 32-character string holding a 128-bit hexadecimal value.

The Alpha Memory will then update its description by over-writing the alpha mask section of its description string with the received 32 characters. Note that the received mask is the same regardless of which clothing item – the HUD only sends its complete mask.

- Provides a means of **sending the alpha mask** to the mesh body HUD.

  When the Mesh Body HUD asks for alpha information by sending a message to the Alpha Memory objects, the alpha memory will read its description text, obtaining the item mask and the alpha mask.

  The Alpha memory then performs a logical AND function on the mask values, to create a single 128-bit mask that indicates the stored alpha mask settings within the region of the body defined by the item mask.

  That is, it creates a mask value by setting a bit to 1 only where the same bit is a 1 in both the item and alpha masks, and leaves all other bits set to 0.

  This derived alpha mask is then converted to a hexadecimal character string of 32 characters, and returned to the mesh body HUD.

- Provides a means for **receiving the item mask** from the mesh body HUD.

  This last item is mostly going to be useful as a way of easily creating the item mask, by simply using the facilities of the mesh body HUD, however it would not strictly need to be supported for the alpha memory system to be useable.

  The item mask is received as a 32-character string holding a 128-bit hexadecimal value.

  The Alpha Memory will then update its description by over-writing the ***item mask*** section of its description string with the received 32 characters.

  I envisage the user (or the clothing designer) will create the item mask by turning on (making transparent) all of the alpha segments that are completely within (covered by) the clothing item when it is worn. They will use the mesh body HUD facility to do this, and then select a button on the mesh body HUD "Create Item Mask" which will send the information to the Alpha Memory object in a message.

  The user need to ONLY wear the Alpha Memory object for the particular item.

## Messages between the HUD and the Alpha Memory

The following messages are supported in the Alpha memory and can be used by the HUD. Each message has message type and a message data. The message type advises the message receiver of what action is taking place, and the message body is (usually) the alpha mask data.

### Memory Attach

On attach of the alpha memory, it will send the alpha mask to the HUD in a "attach" message.

### Memory Detach

On detach of the alpha memory, it will send the alpha mask in a "detach" message.

### Request Alpha

Sent by the HUD, this message has no alpha data, and is used to trigger a response from the alpha memory.

### Request Alpha Response

Sent by the alpha memory when it receives a Request Alpha message. It contains the alpha mask.

### Save Alpha

Sent by the HUD, and containing the current HUD alpha map. The alpha memory receives this message and saves the data in the alpha mask memory.

### Save Item

Sent by the HUD, and containing the current HUD alpha map. The alpha memory receives this message and saves the data in the item mask memory.

## Sizing the Item and Alpha Masks

The Alpha Memory will use the only bit of storage space it has read and write access to: Its own description, which is a 127-byte UTF character string.

While the description can hold 127 bytes as individual characters (so, 127 X 8 = 1016 bits) the highest information density we can achieve is 4 bits per byte, by storing hexadecimal digits as string characters (0123456789abcdef).

So the true limit to how much information we can store at a practical level is 127 X 4 = 508 bits.

Because we need to store both the item mask (which could be for the whole alpha map, or partial) and the alpha mask, and these need to be the same size, then our available alpha memory is reduced by ½, so 508/2 = 254 bits can be stored (in either item or alpha mask).

No current mesh body has anywhere near 254 alpha segments.

There is a bit of a balance between segment size and usability in the alpha segments on a mesh body HUD. The user has to be able to select a segment as a button and making them too small makes that difficult. The user also demands as many segments as possible in critical areas such as necklines and hem lines, so that they can successfully create alphas for as broad a range of mesh clothes as possible. So far the number of segments has been increasing, with the highest number of segments I am aware of (I have not seen all bodies yet however) being the Maitreya Lara, with 95 segments.

Also, we are in a computing space here – while it is not a strict limit, it is going to be incredibly easier if we work in powers of 2, since we are essentially manipulating binary data (segments are on, or off) in alpha masks. So I propose we set an upper limit of 128 alpha segments.

128 alpha segments can be represented as a binary pattern using 128/4 = 32 hexadecimal characters. So my ***proposal for Alpha memory capacity*** is 256 bits, represented as:

a). 128 bits of item mask written as 32 characters in a hexadecimal numeric format.

Plus

b). 128 bits of alpha mask written as 32 characters in a hexadecimal numeric format.

## Mesh Body HUD modifications

The Alpha Memory system needs to interact with the mesh body HUD for a particular body, so the mesh body HUD will need to add support for the Alpha memory system.

The following functions are defined for the mesh body HUD enhancements:

1. The Mesh body HUD must have some storage of the current state of the alpha segments (I assume this already exists).

2. The mesh body HUD must have a means of consistently identifying each segment with an identifier. (I assume this is already the case)

3. The mesh body HUD will hold an ***alpha list*** of 128 items, where each item contains the segment identifier for one alpha segment.

4. Listens for alpha memory "Attach" messages containing the alpha pattern of the attaching alpha memory, and uses the alpha pattern to turn on only those alpha segments (in addition to any already set).

5. Listens for alpha memory "Detach" messages containing the alpha pattern, and uses the alpha data to turn off only the indicated alpha segments.

6. *Provide a "Save Alpha" button* that, when pressed:

    a. Uses the alpha list to determine the state of each alpha segment, creating a 128 item ***state list*** (holds 128 1 & 0 values),

    b. Converts the state list to a 32-character string representing a hexadecimal number that is the value of the 128 bit binary state list.

    c. Sends the 32-character string as a message on a defined channel as a "Save" message to the Alpha Memory.

7. *Provide a "Load Alpha" button* that, when pressed:

    d. Sends a message on a defined channel to request alpha information from Alpha Memory objects.

    e. Starts a listener for the response message.

    f. Creates an empty (all off) alpha map (the "***working alpha***")

    g. For each received message

        i. Receives the 32 character ***item alpha mask***, and converts it to a binary (bit mask) in a list.

        ii. Performs a logical OR between the working alpha and the item alpha, so that the working alpha will have one bit turned on for every "ON" bit in any of the received messages.

            Working alpha = (working alpha OR item alpha mask)

    h. Updates the HUD alpha mask to reflect the working alpha mask.

8. Provide a “Create Item Mask” button that, when pressed performs exactly the same function as the Save Alpha button, but sends it as a “Create” message to the Alpha Memory.

## Some useful references

There is obviously more that is needed, but these provide some of the Number / string / List converting

http://wiki.secondlife.com/wiki/BinaryDecimalConverter

http://wiki.secondlife.com/wiki/Base2Dec

http://wiki.secondlife.com/wiki/ParseString2List