```
////////////////////////////////////////////////////////////////////
//
// A script to change textures on a prim by voice command or menu.
// The menu-based code is derived from the SimpleDialogMenuSystem
// script by Omei Qunhua
// see http://wiki.secondlife.com/wiki/SimpleDialogMenuSystem
//
// Currently listens on the channel set by the LISTENER_CHANNEL
// variable. To listen on a different channel, change this value.
// Future versions may use a touch menu or notecard to change this.
//
// All commands are in the format of "command <param>" where the
// command and param are separated by a space.
// Currently param is a single string, but could be extended in the
// future to a comma-separated list using llParseString2List
// or llCSV2List
//
/// @author Rebecca Ashbourne
/// @version 1.0
/// @copyright 2017 Rebecca Ashbourne
//
// This script is free software: you can redistribute it and/or
// modify it under the terms of the GNU Lesser General Public License
// as published by the Free Software Foundation, either version 2.1
// of the License, or (at your option) any later version.
//
// This script is distributed in the hope that it will be useful, but
// WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
// See the GNU Lesser General Public License
// at http://www.gnu.org/licenses/ for more details.
//
// TODO list:
// * Add an access list rather than restricting to owner-only.
//   Maybe via notecard.
// * Add ability to set the listener channel via a notecard.
//
////////////////////////////////////////////////////////////////////

// Settings
integer LISTENER_CHANNEL = 45; // change this to listen on a different channel
integer face = ALL_SIDES; // initial / default value.

// Constants
list ordinals = ["0","1","2","3","4","5","6","7","8","9"];
list aliases =  [ "default", "blank", "transparent", "media" ];

// Global variables
integer listenerHandle;
string faceName;

list       gListFullNames;   // List of inventory textures
list       gListBriefNames;  // List of abbreviated texture names for dialog buttons
```

```
integer    gPage;            // Current dialog page number (counting from zero)
integer    gMaxPage;         // Highest page number (counting from zero)
integer    gChan;            // Channel used for dialog communications.
key        gUser;            // Current user accessing the dialogs


////////////////////////////////////////////////////////////////////////

/// Gets the UUID of a default texture from its alias name.
/// @return the UUID if @name is a valid alias, or else an empty string
string getAlias(string name)
{
    if (name == "default" || name == "plywood" || name == "TEXTURE_PLYWOOD")
        return TEXTURE_PLYWOOD;
    else if (name == "blank" || name == "TEXTURE_BLANK")
        return TEXTURE_BLANK;
    else if (name == "transparent" || name == "TEXTURE_TRANSPARENT")
        return TEXTURE_TRANSPARENT;
    else if (name == "media" || name == "TEXTURE_MEDIA")
        return TEXTURE_MEDIA;
    else
        return "";
}

/// Reports a string to the owner(s)
report(string message)
{
    key owner = llGetOwner();

    // single owner that the region still has a handle for
    if (llKey2Name(owner))
    {
        llOwnerSay(message);
    }
    // group owned, must send the message publicly
    else if (llList2Key(llGetObjectDetails(llGetKey(), [OBJECT_GROUP]), 0) == owner)
    {
        llWhisper(0, "/me : " + message);
    }
    // single owner, not present, send them an IM
    else
    {
        llInstantMessage(owner, message);
    }
}

/// Parses a string into a command and params, which are separated by a space.
/// @return a list whose 1st element is the command and whose 2nd element is the params string, which may be empty.
list parseCommandString(string str)
{
    list newList;
```

```
    integer idx = llSubStringIndex(str, " ");
    if (~idx)
        newList = [ llGetSubString(str, 0, idx-1), llStringTrim(llGetSubString(str, idx+1, -1), STRING_TRIM) ];
    else
        newList = [ str, "" ];

    return newList;
}

/// Checks that the param string is not empty. If it is then it emits an error message and returns FALSE
/// @return TRUE or FALSE
integer hasParam(string str)
{
    integer result = (str != "");
    if (!result)
        report("Missing parameter");
    return result;
}

/// Checks that the user is authorised. If they are not, then emits an error message and returns FALSE
/// @return TRUE or FALSE
integer userAuthorised(key user)
{
    key owner = llGetOwner();

    // TODO: Test for group-owned

    if (user != owner)
    {
        llInstantMessage(user, "Access denied");
        report("Unauthorised molestation from " + (string)llKey2Name(user));
        return FALSE;
    }

    return TRUE;
}

/// Gets the positive index represented by the string, otherwise returns -1
/// @returns a valid positive integer, or else -1
/// @note Does not support an explicit leading plus sign. Does not support hexadecimal.
integer getIndex(string str)
{
    integer idx = -1;
    string c = llGetSubString(str,0,0); // first character
    if (~llListFindList(ordinals,[c]))
        idx = (integer)str;
    return idx;
}

/// Sets the texture by name.
/// Allows a limited set of alises too for system textures such as default (plywood), white, media, and transparent.
/// @param name a texture in the inventory of the prim this script is in or a UUID of a texture
/// @return TRUE if the name was valid or else FALSE
```

```
/// @note a texture whose name is the same as an alias will override the alias.
integer setByName(string name)
{
    string texture;
    integer type = llGetInventoryType(name);

    if (type == INVENTORY_TEXTURE) // It's a valid texture in the same object as this script. It can be used.
        texture = name;
    else
        texture = getAlias(name);

    if (texture != "")
    {
        report("Setting texture of face " + faceName + " to '" + texture +"'");
        llSetTexture(texture, face);
        return TRUE;
    }

    report("Invalid texture name. Try saying 'status' to get all textures.");
    return FALSE;
}

/// Sets the texture by index
/// @param idx the index of the texture, corresponding to textures in the prim this script is in, sorted alphabetically.
setByIndex(integer idx)
{
    // NOTE: We could bounds check idx against the number of textures, but the call to llGetInventoryType
    //       will effectively do this for us.

    string texture = llGetInventoryName(INVENTORY_TEXTURE, idx);
    integer type = llGetInventoryType(texture);
    if (type == INVENTORY_TEXTURE)
    {
        report("Setting texture of face " + faceName + " to '" + texture +"'");
        llSetTexture(texture, face);
    }
    else
    {
        report("Index " + (string)idx +" does not correspond to a texture. Try saying 'status' to get all textures.");
    }
}

/// Sets the face value.
/// @return TRUE if the face was valid or else FALSE
integer setFace(integer idx)
{
    if (idx >= ALL_SIDES && idx < llGetNumberOfSides())
    {
        face = idx;
        if (idx == ALL_SIDES)
            faceName = "ALL_SIDES";
        else
            faceName = (string)idx;
```

```
        return TRUE;
    }
    else
    {
        return FALSE;
    }
}

/// Compact function to put buttons in "correct" human-readable order
list orderButtons(list buttons)
{
    // Button Order
    //  9   10   11
    //  6    7    8
    //  3    4    5
    //  0    1    2

    return llList2List(buttons, -3, -1) + llList2List(buttons, -6, -4)
         + llList2List(buttons, -9, -7) + llList2List(buttons, -12, -10);
}

/// Builds the dialog menu
buildDialogPage(key user)
{
    integer totalChoices = (gListBriefNames != [] ); // get length of texture list

    // set up scrolling buttons if needed
    list buttons = [ "<<", "-", ">>" ];

    integer choicesPerPage = 9;
    if (totalChoices < 13)
    {
        buttons = [];
        choicesPerPage = 12;
    }

    // Compute number of menu pages that will be available
    gMaxPage = (totalChoices - 1) / choicesPerPage;

    // Build a dialog menu for current page for given user
    integer start = choicesPerPage * gPage;       // starting offset into action list for current page

    // 'start + choicesPerPage -1' might point beyond the end of the list, but LSL stops
    // at the list end, without throwing a wobbly
    buttons = llList2List(gListBriefNames, start, start + choicesPerPage - 1) + buttons;

    // Now raise the dialog
    llDialog(user, "\nPage " + (string) (gPage+1) + " of " + (string) (gMaxPage + 1) + "\n\nChoose an item",
             orderButtons(buttons), gChan);
    llSetTimerEvent(30);  // If no response in time, return to 'ready' state
}

////////////////////////////////////////////////////////////////////
```

```
default
{
    state_entry()
    {
        setFace(face);
        state ready;
    }
}

state ready
{
    state_entry()
    {
        listenerHandle = llListen(LISTENER_CHANNEL, "", llGetOwner(), "");
        report("State: ready");
    }

    touch_end(integer total_number)
    {
        gUser = llDetectedKey(0);

        if (!userAuthorised(gUser))
            return;

        gChan = 0x80000000 | (integer)("0x" + (string) llGetKey()); // Compute a negative communications channel based on prim UUID
        gListFullNames = [];
        gListBriefNames = [];

        integer n = 0;
        integer nTextures = llGetInventoryNumber(INVENTORY_TEXTURE);
        do
        {
            string name = llGetInventoryName(INVENTORY_TEXTURE, n);
            gListFullNames += name;
            gListBriefNames += llGetSubString(name, 0, 23);
        }
        while (++n < nTextures);

        n = 0;
        integer nAliases = llGetListLength(aliases);
        do
        {
            string name = llList2String(aliases, n);
            gListFullNames += name;
            gListBriefNames += ("(" + name + ")");
        }
        while (++n < nAliases);

        // Changing state sets the application to a busy condition while one user is selecting from the dialogs
        // In the event of multiple 'simultaneous' touches, only one user will get a dialog
        state busy;
    }
```

```
/// Respond to a voice command
listen(integer channel, string name, key id, string message)
{
    if (!userAuthorised(id))
        return;

    // All commands are in the format of "command <params>" where the command and params are separated by a space.
    list cmdLine = parseCommandString(message);
    string command = llList2String(cmdLine, 0);
    string param = llList2String(cmdLine, 1);

    if (command == "status") // shows all options and also the current texture
    {
        report("Listening on channel " + (string)LISTENER_CHANNEL + " with listener id " + (string)listenerHandle);
        report("Current texture: " + llGetTexture(face));
        report("Face: " + faceName);
        report("Available textures:");

        integer n = 0;
        integer nTextures = llGetInventoryNumber(INVENTORY_TEXTURE);
        do
        {
            string textureName = llGetInventoryName(INVENTORY_TEXTURE, n);
            report((string)n + ": " + textureName);
        }
        while (++n < nTextures);
    }
    else if (command == "name" || command == "set") // set the texture by name
    {
        if (param == "") // get instead of set
        {
            report("Face " + faceName + " is currently set to '" + (string)llGetTexture(face) + "'");
            return;
        }

        if (!setByName(param))
            report("'" + param + "' is not a valid texture. Try saying 'status' to get all textures.");
    }
    else if (command == "index" || command == "idx") // set the texture by index
    {
        if (!hasParam(param))
            return;

        integer idx = getIndex(param);
        if (idx != -1)
            setByIndex(idx);
        else
            report("'" + param + "' does not correspond to valid index. Try saying 'status' to get all textures.");
    }
    else if (command == "face")
    {
        if (param == "") // show the current face
```

```
            {
                report("Face is currently " + faceName);
                return;
            }

            // Note: Due to the way getIndex() works, the side effect is that any invalid value will be interpreted as ALL_SIDES
            if (setFace(getIndex(param)))
                report("Face set to " + faceName);
            else
                report("Invalid face '" + param + "'");
        }
        else
        {
            // If a command isn't recognised, treat it as a texture name and try loading that by name.
            // Only if that fails then report an invalid command or texture to the user.
            if (!~setByName(message))
                report("'" + message +"' is not a valid command or texture.");
        }
    }

} // ready

state busy
{
    state_entry()
    {
        report("State: busy");
        llListen(gChan, "", gUser, "");  // This listener will be used throughout this state
        gPage = 0;
        buildDialogPage(gUser);          // Show  Page 0 dialog to current user
    }

    listen (integer chan, string name, key id, string msg)
    {
        if (msg == "<<" || msg == ">>")                  // Page change ...
        {
            if (msg == "<<")        --gPage;             // Page back
            if (msg == ">>")        ++gPage;             // Page forward
            if (gPage < 0)          gPage = gMaxPage;    // cycle around pages
            if (gPage > gMaxPage)   gPage = 0;
            buildDialogPage(id);
            return;
        }

        if (msg != " " && msg != "-")
        {
            // User has selected a texture from the menu
            integer idx = llListFindList(gListBriefNames, [msg]);
            string name = llList2String(gListFullNames, idx);

            setByName(name);
        }
        state ready;
```

```
    }

    timer()
    {
        state ready;
    }

    state_exit()
    {
        llSetTimerEvent(0); // kill the timer
    }

} // busy
```