**Scripts and Web Service Codes**

Based on Functions :

***Greet\_Visitor* ( )**

Webservice

using System;

using System.Collections.Generic;

using System.Web;

using System.Web.Services;

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

public class Service : System.Web.Services.WebService

{

public Service () {

//Uncomment the following line if using designed components

//InitializeComponent();

}

[WebMethod]

public string Greetings(string avatarAtDoor) {

return "Welcome " + avatarAtDoor + " Please enter the password in nearby chat.";

}

}

***Authorize\_Visitor( )***

Webservice

using System;

using System.Collections.Generic;

using System.Web;

using System.Web.Services;

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

public class Service : System.Web.Services.WebService

{

public Service () {

//Uncomment the following line if using designed components

//InitializeComponent();

}

[WebMethod]

public int AuthorizingAvatar(string AvatarAtDoor, string password) {

if (AvatarAtDoor == "Vidnc89" && password == "enterDoor")

return 1;

else

return 0;

}

}

***Greeting and Authentication***

LSL Script

string greetingURL= "http://vhost2245.site1.compute.ihost.com/WEBsite5/GreetingAtTheDoor/Service.asmx/Greetings?avatarAtDoor=";

string authorizationURL="http://vhost2245.site1.compute.ihost.com/WEBsite5/AuthorizingAvatar/Service.asmx/AuthorizingAvatar?avatarAtDoor=";

// The standard HTTP success response code

integer HTTP\_OK = 200;

key greeting\_request\_id;

key authorization\_request\_id;

string avatarAtDoor; // The password

integer listen\_handle;

string getGreeting(string body)

{

string greeting = "";

string startStr = "<string xmlns=\"http://tempuri.org/\">";

string endStr = "</string>";

integer startIndex = llSubStringIndex(body, startStr);

integer endIndex = llSubStringIndex(body, endStr);

if (startIndex != -1 && endIndex != -1 && endIndex > startIndex)

{

greeting = llGetSubString(body, startIndex + llStringLength(startStr), endIndex - 1);

}

return greeting;

}

string getAuthorization(string body)

{

string authorized;

string startStr = "<int xmlns=\"http://tempuri.org/\">";

string endStr = "</int>";

integer startIndex = llSubStringIndex(body, startStr);

integer endIndex = llSubStringIndex(body, endStr);

if (startIndex != -1 && endIndex != -1 && endIndex > startIndex)

{

authorized = llGetSubString(body, startIndex + llStringLength(startStr), endIndex - 1);

}

return authorized;

}

default

{

touch\_start(integer total\_number)

{

llPlaySound("c3b71052-ec38-ca47-0dc1-4bd8c84667e3", 20.0);

avatarAtDoor = llDetectedName(0);

greeting\_request\_id = llHTTPRequest(greetingURL + avatarAtDoor, [HTTP\_METHOD,"GET"], "");

listen\_handle = llListen(0, "", llDetectedKey(0), "");

}

listen( integer channel, string name, key id, string message )

{

llListenRemove(listen\_handle);

authorization\_request\_id = llHTTPRequest(authorizationURL + avatarAtDoor + "&password=" + message, [HTTP\_METHOD,"GET"], "");

}

http\_response(key id, integer status, list metadata, string body)

{

// check if the page was successfully fetched

if (status == HTTP\_OK)

{

if (id == greeting\_request\_id)

{

string greeting = getGreeting(body);

llSay(0, greeting);

}

else if (id == authorization\_request\_id)

{

string authorized = getAuthorization(body);

if ( authorized == "1")

{

llSay(0, "Authorized, opening the door...");

llWhisper(7654, "OPEN\_DOOR");

}

else

llSay(0, "Not Authorized to enter the door");

}

}

else

{

llSay(0, "Couldn't fetch page, http status: " +

(string)status);

}

}

}

***Greeting and Authentication : Open/Close Door***

LSL Script

float delay = 10.0; // time to wait before automatically closing door

vector delta = <0.0, 1.0, 0.0>; // amount to move door when we open it

vector closed\_position; // original position of the door when closed

// Processing for the script when it first starts up

default {

// What we do when we first enter this state

state\_entry() {

closed\_position = llGetPos(); // Save position when door is closed

state closed; // Move to the closed state

}

}

// Processing for the script when it is in the closed state

state closed {

// What we do when we first enter this state

state\_entry() {

llSetPos(closed\_position); // Move door to closed position

llListen(7654, "", NULL\_KEY, "");

}

listen( integer channel, string name, key id, string message )

{

if (message == "OPEN\_DOOR")

state open;

}

timer()

{

llSetTimerEvent(0.0); // Set the timer to 0.0 to turn it off

}

}

// Processing for the script when it is in the open state

state open {

// What we do when we first enter this state

state\_entry() {

llSetPos(closed\_position + delta); // Move door to open position

llSetTimerEvent(delay); // Set the timer to automatically close it

}

// What we do when the door is clicked ("touched") with the mouse

touch\_start(integer total\_number) {

state closed; // Move to the closed state

}

timer()

{

llSetTimerEvent(0.0); // Set the timer to 0.0 to turn it off

state closed; // Move to the closed state

}

}

***Parser/Live Feed :***

LSL Script

// Get the first link in the first item in an RSS feed and open it in

// the user's web browser when the object is touched

//

// This script is very naive and not very robust. It expects only

// one user to be clicking on it at a time.

// URL for the RSS feed

string sourceURL= "http://blog.secondlife.com/feed/";

// The standard HTTP success response code

integer HTTP\_OK = 200;

// Key of the resident that clicked on the object. Saved in the

// touch\_start event so that it can be used in the http\_response event.

key requestingResident = NULL\_KEY;

// Parse an HTTP response that should be an RSS feed and return the

// first link from the first item, or "" if the link could not be found

//

// body is the response from an HTTP request. Since HTTP requests are

// currently limited to 2048 characters, and since this is unknown data

// from the outside world, the body may or may not actually contain the

// information we want.

//

string getFirstItemLink(string body)

{

// Find the start of the "items" in the RSS feed

string startItems = "<item>";

if (llSubStringIndex(body, startItems) == -1)

{

return "";

}

// get the string from the beginning of the first item to the end

// of the body

string items =

llGetSubString(body, llSubStringIndex(body, startItems), -1);

// find the first link in the items

string startLink = "<link>";

string endLink = "</link>";

integer startIndex = llSubStringIndex(items, startLink);

integer endIndex = llSubStringIndex(items, endLink);

// make sure startLink and endLink were found, and startLink is

// before end link

if (startIndex != -1 && endIndex != -1 && endIndex > startIndex)

{

string link =

llGetSubString(

items,

startIndex +llStringLength(startLink),

endIndex - 1);

// Got it!

return link;

}

else

{

return "";

}

}

default

{

touch\_start(integer total\_number)

{

// Save the requestor

requestingResident = llDetectedKey(0);

llSay(0, "Fetching: " + sourceURL);

// Get the RSS feed

llHTTPRequest(sourceURL, [], "");

}

http\_response(key id, integer status, list metadata, string body)

{

// check if the page was successfully fetched

if (status == HTTP\_OK)

{

string link = getFirstItemLink(body);

if (link != "")

{

// Got the link, load it in the user's browser

llLoadURL(requestingResident, "", link);

}

else

{

// Got a response, but didn't get the link

llSay(0, "Could not find link");

}

}

else

{

llSay(0, "Couldn't fetch page, http status: " +

(string)status);

}

// reset this variable

requestingResident = NULL\_KEY;

}

}

***Email Notification***

LSL Script

string emailAddress = "vid4ever@gmail.com";

string emailHeader = "You're registered with SmartChildCare!";

default

{

touch\_start(integer num\_detected)

{

llSay(PUBLIC\_CHANNEL, "Sending eMail report now, this will take ~20 seconds.");

key id = llDetectedKey(0);

string name = llDetectedName(0);

llEmail(emailAddress, emailHeader,

"Registered by: '" + name + "' (" + (string)id + ").");

llSay(PUBLIC\_CHANNEL, "Email has been sent.");

}

}

***Fire Display on Pan***

LSL Script

default

{

touch(integer touch\_number)

{

llParticleSystem([

PSYS\_PART\_MAX\_AGE,2.000,

PSYS\_PART\_FLAGS,PSYS\_PART\_EMISSIVE\_MASK|PSYS\_PART\_INTERP\_COLOR\_MASK|PSYS\_PART\_INTERP\_SCALE\_MASK,

PSYS\_PART\_START\_COLOR, <1.00000, 0.40000, 0.00000>,

PSYS\_PART\_END\_COLOR, <0.10000, 0.00000, 0.00000>,

PSYS\_PART\_START\_SCALE,<0.50000, 0.50000, 1.00000>,

PSYS\_PART\_END\_SCALE,<0.75000, 0.75000, 1.00000>,

PSYS\_SRC\_PATTERN, 10,

PSYS\_SRC\_BURST\_RATE,0.000,

PSYS\_SRC\_ACCEL, <0.00000, 0.00000, 0.40000>,

PSYS\_SRC\_BURST\_PART\_COUNT,15,

PSYS\_SRC\_BURST\_RADIUS,0.050,

PSYS\_SRC\_BURST\_SPEED\_MIN,0.000,

PSYS\_SRC\_BURST\_SPEED\_MAX,0.000,

PSYS\_SRC\_ANGLE\_BEGIN,3.000,

PSYS\_SRC\_ANGLE\_END,0.000,

PSYS\_SRC\_OMEGA, <0.00000, 0.00000, 0.00000>,

PSYS\_SRC\_MAX\_AGE, 0.000,

PSYS\_SRC\_TEXTURE, "",

PSYS\_PART\_START\_ALPHA, 1.000,

PSYS\_PART\_END\_ALPHA, 0.050

]);

llWhisper(6543, "FIRE\_STARTED");

}

}

***Fire Detection***

LSL Script

vector redColor = <1.0, 0.0, 0.0>;

vector whiteColor = <1.0, 1.0, 1.0>;

default

{

state\_entry()

{

llSetColor(whiteColor, ALL\_SIDES);

llListen(6543, "", NULL\_KEY, "");

}

listen(integer channel, string name, key id, string message)

{

if (message == "FIRE\_STARTED")

{

llParticleSystem

([

PSYS\_PART\_FLAGS,

PSYS\_PART\_INTERP\_COLOR\_MASK|

PSYS\_PART\_FOLLOW\_SRC\_MASK|

PSYS\_PART\_EMISSIVE\_MASK, PSYS\_SRC\_PATTERN,

PSYS\_SRC\_PATTERN\_ANGLE\_CONE,

PSYS\_SRC\_INNERANGLE, 0.0,

PSYS\_SRC\_OUTERANGLE, 0.1,

PSYS\_SRC\_BURST\_SPEED\_MIN, 0.0,

PSYS\_SRC\_BURST\_SPEED\_MAX, 0.0,

PSYS\_SRC\_BURST\_RADIUS, 0.0,

PSYS\_SRC\_BURST\_PART\_COUNT, 5,

PSYS\_SRC\_BURST\_RATE, 1.0,

PSYS\_PART\_MAX\_AGE, 1.0,

PSYS\_PART\_START\_SCALE, <1,1,1>,

PSYS\_PART\_START\_COLOR, <1,0.2,0.2>,

PSYS\_PART\_END\_COLOR, <1,0,0>,

PSYS\_PART\_START\_ALPHA, 1.0,

PSYS\_PART\_END\_ALPHA, 0.0

]);

llTargetOmega(<1.0,0.0,0.0>, 4, 1);

llSetColor(redColor, ALL\_SIDES);

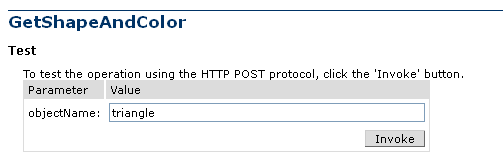
}

}

}

***Teach\_Child ( )***

Web Service for Shape and color of an object





using System;

using System.Collections.Generic;

using System.Web;

using System.Web.Services;

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

public class Service : System.Web.Services.WebService

{

public Service () {

//Uncomment the following line if using designed components

//InitializeComponent();

}

[WebMethod]

public string GetShapeAndColor( string objectName ) {

if (objectName == "square")

{

return "This is a BROWN SQUARE";

}

else if (objectName == "circle")

{

return "This is a PINK CIRCLE";

}

else if (objectName == "triangle")

{

return "This is a RED TRIANGLE";

}

else if (objectName == "rectangle")

{

return "This is a GREEN RECTANGLE";

}

else

{

return "What object is this????";

}

}

}

***Fire\_Alarm ( ) :***

Improvised for alarm and light feature.

Code for fire on the pot in kitchen

integer counter;

default

{

state\_entry()

{

counter = 0;

}

touch(integer touch\_number)

{

llParticleSystem([

PSYS\_PART\_MAX\_AGE,2.000, PSYS\_PART\_FLAGS,PSYS\_PART\_EMISSIVE\_MASK|PSYS\_PART\_INTERP\_COLOR\_MASK|PSYS\_PART\_INTERP\_SCALE\_MASK,

PSYS\_PART\_START\_COLOR, <1.00000, 0.40000, 0.00000>,

PSYS\_PART\_END\_COLOR, <0.10000, 0.00000, 0.00000>,

PSYS\_PART\_START\_SCALE,<0.50000, 0.50000, 1.00000>,

PSYS\_PART\_END\_SCALE,<0.75000, 0.75000, 1.00000>,

PSYS\_SRC\_PATTERN, 10,

PSYS\_SRC\_BURST\_RATE,0.000,

PSYS\_SRC\_ACCEL, <0.00000, 0.00000, 0.40000>,

PSYS\_SRC\_BURST\_PART\_COUNT,15,

PSYS\_SRC\_BURST\_RADIUS,0.050,

PSYS\_SRC\_BURST\_SPEED\_MIN,0.000,

PSYS\_SRC\_BURST\_SPEED\_MAX,0.000,

PSYS\_SRC\_ANGLE\_BEGIN,3.000,

PSYS\_SRC\_ANGLE\_END,0.000,

PSYS\_SRC\_OMEGA, <0.00000, 0.00000, 0.00000>,

PSYS\_SRC\_MAX\_AGE, 0.000,

PSYS\_SRC\_TEXTURE, "",

PSYS\_PART\_START\_ALPHA, 1.000,

PSYS\_PART\_END\_ALPHA, 0.050 ]);

llSetTimerEvent(5.0);

}

timer()

{

if ( counter == 0)

{

llWhisper(6543, "FIRE\_STARTED");

llSetTimerEvent(120.0);

counter = counter + 1;

}

else

{

if ( counter == 1)

{

llParticleSystem([]);

llSetTimerEvent(0.0);

counter = 0;

}

}

}

}

Code for fire alarm flashing red light

vector redColor = <1.0, 0.0, 0.0>;

vector whiteColor = <1.0, 1.0, 1.0>;

default

{

state\_entry()

{

llSetColor(whiteColor, ALL\_SIDES);

llListen(6543, "", NULL\_KEY, "");

}

listen(integer channel, string name, key id, string message)

{

if (message == "FIRE\_STARTED")

{

llParticleSystem

([

PSYS\_PART\_FLAGS,

PSYS\_PART\_INTERP\_COLOR\_MASK|

PSYS\_PART\_FOLLOW\_SRC\_MASK|

PSYS\_PART\_EMISSIVE\_MASK, PSYS\_SRC\_PATTERN,

PSYS\_SRC\_PATTERN\_ANGLE\_CONE,

PSYS\_SRC\_INNERANGLE, 0.0,

PSYS\_SRC\_OUTERANGLE, 0.1,

PSYS\_SRC\_BURST\_SPEED\_MIN, 0.0,

PSYS\_SRC\_BURST\_SPEED\_MAX, 0.0,

PSYS\_SRC\_BURST\_RADIUS, 0.0,

PSYS\_SRC\_BURST\_PART\_COUNT, 5,

PSYS\_SRC\_BURST\_RATE, 1.0,

PSYS\_PART\_MAX\_AGE, 1.0,

PSYS\_PART\_START\_SCALE, <1,1,1>,

PSYS\_PART\_START\_COLOR, <1,0.2,0.2>,

PSYS\_PART\_END\_COLOR, <1,0,0>,

PSYS\_PART\_START\_ALPHA, 1.0,

PSYS\_PART\_END\_ALPHA, 0.0

]);

llTargetOmega(<1.0,0.0,0.0>, 4, 1);

llSetColor(redColor, ALL\_SIDES);

llSetTimerEvent(120.0);

}

}

timer()

{

llParticleSystem([]);

llTargetOmega(<1.0,0.0,0.0>, 4, 0);

llSetColor(whiteColor, ALL\_SIDES);

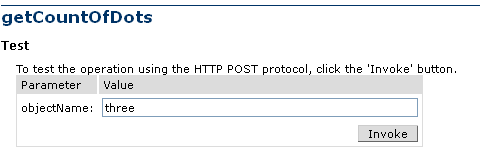
llSetTimerEvent(0.0);

}

}

***Teach Child ()*** : For Counting Number of Symbols

Web Service





using System;

using System.Collections.Generic;

using System.Web;

using System.Web.Services;

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

public class Service : System.Web.Services.WebService

{

public Service () {

//Uncomment the following line if using designed components

//InitializeComponent();

}

[WebMethod]

public string getCountOfDots(string objectName) {

if (objectName == "one")

{

return "There is ONE(1) dot";

}

else if (objectName == "two")

{

return "There are TWO(2) dots";

}

else if (objectName == "three")

{

return "There are THREE(3) dots";

}

else

{

return "There are no dots to count here!";

}

}

}

***Exit Lights***

vector greenColor = <0.0, 1.0, 0.0>;

vector whiteColor = <1.0, 1.0, 1.0>;

default

{

state\_entry()

{

llSetColor(whiteColor, ALL\_SIDES);

llListen(6543, "", NULL\_KEY, "");

}

listen(integer channel, string name, key id, string message)

{

if (message == "FIRE\_STARTED" || message == "FLASH ARROW 1")

{

llSetColor(greenColor, ALL\_SIDES);

llSetTimerEvent(2.0);

}

}

timer()

{

llSetColor(whiteColor, ALL\_SIDES);

llWhisper(4444, "FLASH ARROW 2");

llSetTimerEvent(0.0);

}

}

***Wall Change Color***

LSL Script

default

{

state\_entry()

{

llSetTimerEvent(1728000.0); // every 2 seconds

}

timer()

{

llSetColor(<llFrand(1728000.0),llFrand(1728000.0),llFrand(1728000.0)>, ALL\_SIDES);

}

}

***Browser-SL interaction***

***Code for “upload” button click action***

Webservice

using System;

using System.Collections.Generic;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

public partial class \_Default : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void UploadButton\_Click(object sender, EventArgs e)

{

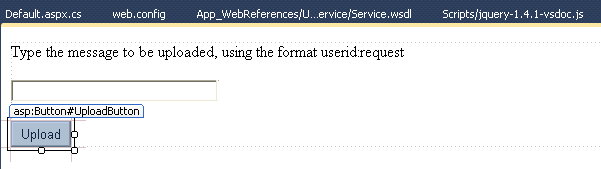
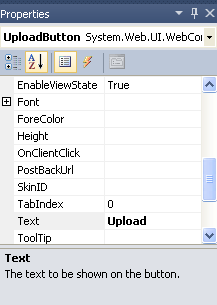
UploadWebService.Service uploadWebService = new UploadWebService.Service();

uploadWebService.Credentials = System.Net.CredentialCache.DefaultCredentials;

uploadWebService.uploadMessage(MessageTextBox.Text);

}

}

***Upload***

web service

using System;

using System.Collections.Generic;

using System.Web;

using System.Web.Services;

using System.IO;

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

public class Service : System.Web.Services.WebService

{

public Service () {

//Uncomment the following line if using designed components

//InitializeComponent();

}

[WebMethod]

public string uploadMessage(string message)

{

FileStream fs = new FileStream(Server.MapPath("message.txt"), FileMode.Append);

StreamWriter streamWriter = new StreamWriter(fs);

streamWriter.BaseStream.Seek(0, SeekOrigin.End);

streamWriter.WriteLine(message);

streamWriter.Flush();

streamWriter.Close();

fs.Close();

return "Login OK";

}

[WebMethod]

public string downLoadMessage()

{

string oldValue = string.Empty, newValue = string.Empty;

using (StreamReader read = new StreamReader(Server.MapPath("message.txt"), true))

{

do

{

newValue = read.ReadLine();

oldValue = newValue != null ? newValue : oldValue;

} while (newValue != null);

}

FileStream fs = new FileStream(Server.MapPath("message.txt"), FileMode.Append);

StreamWriter streamWriter = new StreamWriter(fs);

streamWriter.BaseStream.Seek(0, SeekOrigin.End);

streamWriter.WriteLine("NO\_MESSAGE");

streamWriter.Flush();

streamWriter.Close();

fs.Close();

return oldValue;

}

}

***Download ( )***

web service

using System;

using System.Collections.Generic;

using System.Web;

using System.Web.Services;

using System.IO;

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

public class Service : System.Web.Services.WebService

{

public Service () {

//Uncomment the following line if using designed components

//InitializeComponent();

}

[WebMethod]

public string downLoadMessage()

{

string oldValue = string.Empty, newValue = string.Empty;

using (StreamReader read = new StreamReader(Server.MapPath("message.txt"), true))

{

do

{

newValue = read.ReadLine();

oldValue = newValue != null ? newValue : oldValue;

} while (newValue != null);

}

return oldValue;

}

}

***Download message***

web service

using System;

using System.Collections.Generic;

using System.Web;

using System.Web.Services;

using System.IO;

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

public class Service : System.Web.Services.WebService

{

public Service () {

//Uncomment the following line if using designed components

//InitializeComponent();

}

[WebMethod]

public string downLoadMessage()

{

string oldValue = string.Empty, newValue = string.Empty;

using (StreamReader read = new StreamReader(Server.MapPath("message.txt"), true))

{

do

{

newValue = read.ReadLine();

oldValue = newValue != null ? newValue : oldValue;

} while (newValue != null);

}

return oldValue;

}

}

Email received from secondlife



