Dialog script auto generation:

Procedure:

1. Read the graph description store in a data structure. In the data structure, each node is associated with (1) a sentence that is to be displayed to the target and (2) the bottons to transit to the next state (graph edge). The display may be using dialog box or text box. Dialog box allows different buttons to go to the next state, text box only uses “Submit” to go to the next state.
   1. Grade edge is translated to “state\_button”: for dialog box, all buttons are recorded as they are. For text box, the input file mark the edge as “Submit”, which is translated into a list of empty button.
2. Script generation is broken into the following steps
   1. gen\_header() -- some fixed comments, copyright, etc
   2. gen\_fixed\_variables(): generate fixed variables including:
      1. dialog\_target: who is to see the display, this information is obtained from clicking the object with the script
      2. state\_name: the name of each state as in the input graph file.
      3. timer\_count: each state record the number of timeouts. The script reacts differently with the number of timeouts, the first 2 will redisplay the box, afterthat at the third timeout the script is reset to the default idle state.
      4. Dialog\_box\_interact\_interval = 20: dialog timeout time, set to 20 seconds.
      5. state\_control\_channel = 10001: the channel that the script can take commands, this must be changed for every script in the same island
      6. local\_dialog\_channel = 11001: the channel that the dialog/text box uses. This must be unique for all scripts in the the same island.
   3. gen\_msg\_button (): for each state generated, store the message and buttons values in variables.

In graph file:

s0 [label="Now You are going to teach a labl on Acids and Bases to the students. Do you want to go through the lab guide first?"];

s0 -> s1 [label="Yes"];

s0 -> default [label="No"];

in the generated lsl file:

string s0\_msg = "Now You are going to teach a labl on Acids and Bases to the

students. Do you want to go through the lab guide first?";

list s0\_button = ["Yes", "No"];

in graph file:

s2 [label="A learning objective informs studens what they are expected to achieve

after the instruction. Can you think of 1 or 2 objective for this lab session?"];

s2 -> s3 [label="Submit"];

in the generated lsl file:

string s2\_msg = "A learning objective informs studens what they are expected to

achieve after the instruction. Can you think of 1 or 2 objective for this lab session?";

list s2\_button = [];

d. gen\_utility\_routine(): some hard-coded routines to be used in the code: pay attention to the common channel registered in register\_common\_channel and how the llTextBox and llDialog are called based on the value in the button list ([ ] for llTextBox and non-empty list for llDialog.

reset\_script() {

dialog\_target = NULL\_KEY;

state\_name = "default";

}

register\_common\_channel()

{

llListen(state\_control\_channel, "", NULL\_KEY, "");

llListen(local\_dialog\_channel, "", NULL\_KEY, "");

}

register\_common\_channel\_timer(integer t)

{

llListen(state\_control\_channel, "", NULL\_KEY, "");

llListen(local\_dialog\_channel, "", NULL\_KEY, "");

llSetTimerEvent(t);

}

common\_state\_entry(string n, string s, list l, integer t)

{

state\_name = n; timer\_count = 0;

if (l == [])

llTextBox(dialog\_target, s, local\_dialog\_channel);

else llDialog(dialog\_target, s, l, local\_dialog\_channel);

register\_common\_channel\_timer(t);

}

dialog\_with\_timer(string msg, list button, integer t)

{

llSetTimerEvent(t);

if (button == [])

llTextBox(dialog\_target, msg, local\_dialog\_channel);

else llDialog(dialog\_target, msg, button, local\_dialog\_channel);

}

dialog\_with\_timer\_count(string msg, list button, integer t)

{

timer\_count ++;

if (timer\_count >= 3) {llSetTimerEvent(0); llSay(0, "timeout too many times, b

ack to initial (idle) state"); state default;}

llSetTimerEvent(t);

if (button == [])

llTextBox(dialog\_target, msg, local\_dialog\_channel);

else llDialog(dialog\_target, msg, button, local\_dialog\_channel);

}

e. gen\_state\_control\_routine(): This routine allows the script to go to a certain state by issuing a command to the state\_control\_channel. Here is the command convention:

*-start* --- go to the first state after the default

*-reset* --- go to the default state

*-gotostate:name* --- goto state with name *name*

the generated routine would look like the following:

if (c == state\_control\_channel) {

dialog\_target = ID;

if (msg == "-start") {

llSetTimerEvent(0); state s0;

} else if ((msg == "-reset") || (msg == "-restart")) {

reset\_script();

llSetTimerEvent(0);

state default;

} else if (msg == "-gotostate:default") {

llSetTimerEvent(0);

state default;

} else if (msg == "-gotostate:s0") {

llSetTimerEvent(0);

state s0;

} ........

f. gen\_state(): this routine generate the program for each state. the program differentiate the default state from the rest. It is assumed that the default state does nothing, but when the object is touched, the script starts by going to the next state. Both default state and the all other state can take the command from the state\_control\_channel.

The generated default state is as follows:

default {

state\_entry() {register\_common\_channel\_timer(0);}

touch\_start(integer num\_detected) {

dialog\_target = llDetectedKey(0);

state s0;

}

listen(integer c, string n, key ID, string msg) {

process\_state\_control\_msg(c, n, ID, msg);

}

}

The generated other state would like the following:

state s0 {

state\_entry() {common\_state\_entry("s0", s0\_msg, s0\_button, dialog\_box\_interact\_interval);}

touch\_start(integer num\_detected) {dialog\_with\_timer(s0\_msg, s0\_button, dialog\_box\_interact\_interval);}

timer() {dialog\_with\_timer\_count(s0\_msg, s0\_button, dialog\_box\_interact\_interval);}

listen(integer c, string n, key ID, string msg) {

llSetTimerEvent(dialog\_box\_interact\_interval);

if (c == local\_dialog\_channel) {

if (msg == "Yes") { llSetTimerEvent(0); state s1;}

if (msg == "No") { llSetTimerEvent(0); state default;}

} else {process\_state\_control\_msg(c, n, ID, msg);}

}

}