

Folding Tool

by Bestmomo Lagan

Version 1.0

08/11/2016

Contents

1. Introduction.....	3
1.1 Basic principles.....	3
1.2 Constitution of the objects.....	5
1.3 Scripts.....	5
1.3.1 Setup.....	6
1.3.2 Production.....	6
2. First setup !.....	7
2.1 Initialization.....	7
2.2 Setting axis.....	8
2.3 Rotations.....	12
2.4 Configuration notecard.....	14
2.5 Action !.....	17
2.6 Details of the values of the note.....	18
3. Cocotte.....	19
3.1 Object.....	19
3.2 Axis.....	20
3.3 Setup.....	22
3.3.1 Les axes.....	22
3.3.2 Rotations.....	28
3.3.3 Notecards.....	33
3.4 Action.....	34

1. Introduction

Folding Tool is a tool primarily designed to achieve in Second Life dynamic origami. The first idea comes from a series of french discussions on the JOL forum. I then showed how to make a script to create a cocotte. With the reactions to this example I have concluded that the necessary level LSL was too high to make easily accessible this type of construction.

So I had a long reflection about a simple tool to use, but to make complex origami. I conducted many trials before determining the most obvious way for me to reach a satisfactory result.

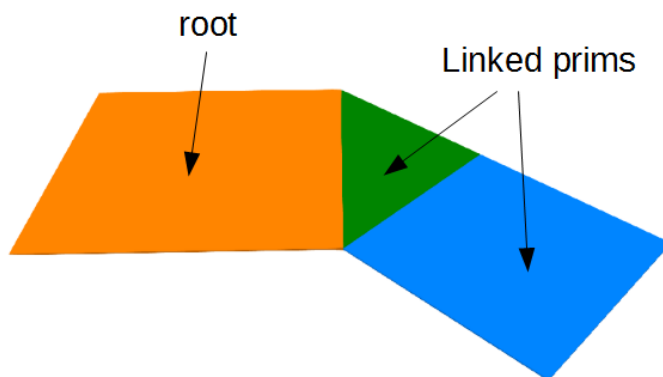
This document is a presentation, a guide and a training manual to discover the tool, understand its principles and manage its operations.

1.1 Basic principles

The tool is based on a number of basic principles you must know :

- the tool must be implemented on an object consisting of linked primitives, it is not possible to use separate primitives,
- the primitive root should serve as a fixed reference,
- rotations are operated by axes (later we'll see how they are generated),
- an axis has a base primitive which it is attached,
- an axis has dependent primitives it brings about in its rotation (and thus consequently dependent axes),
- a notecard contains all the information to go from one state to another (so we can have multiple notecards for several successive states).

These basic principles are important to understand so I'll illustrate with a simple example.



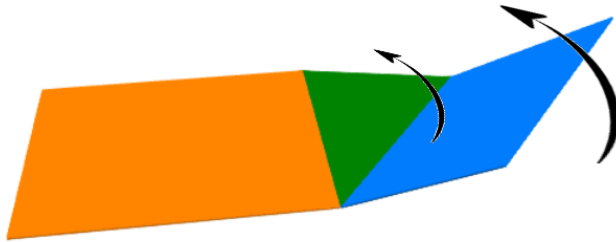
In this object there are three primitives:

- root (orange)
- two linked primitive (green and blue)

It is therefore an object that conforms to the use of Folding Tool.

The root will obviously still and we will move the two linked primitives respecting their connection, they will therefore remain contiguous.

The tool creates a configuration to move from one state to another. With the above example we can for example move the two primitives to reach this state :



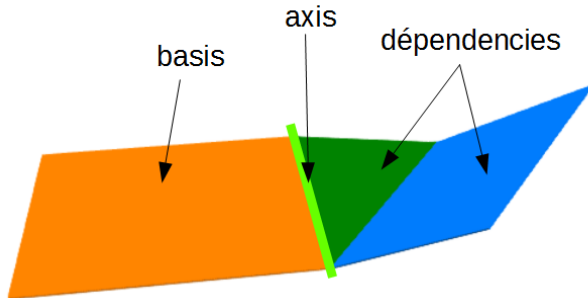
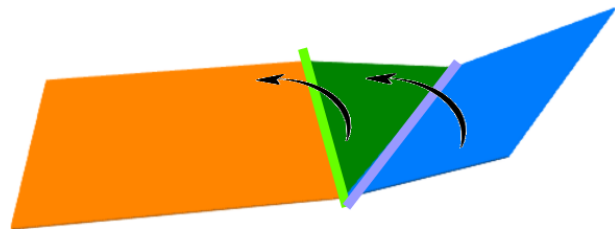
We can see that the two linked primitives have rotated (in reality it is a combination of translation and rotation).

The tool allows for only rotate relative to axes and it is sufficient for the intended purpose.

To illustrate this let see where the axes of rotation are located in this case.

This is what we can see there. The light green axis represents the axis of rotation of the primitive green and light blue axis that of the blue primitive.

For the system to work must firstly axes are secured to a primitive and secondly they have dependent primitives driven by their rotation (and therefore the solidarity axes of these primitive).

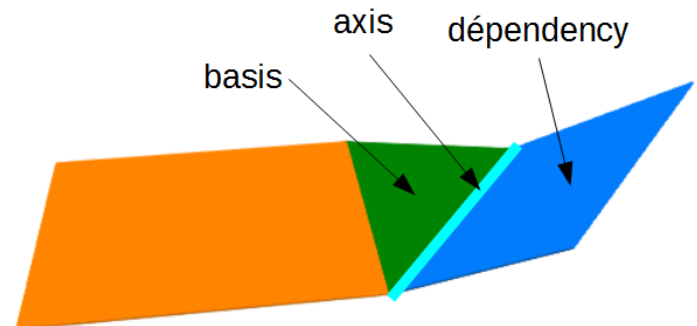


Consider for example the green axis in the illustration against, It has as for base (the original which it is attached) the orange primitive (root in this case) and has 2 dependent primitives.

If the base primitive moves, the axis must follow (in this case it will not move because it is the root). If the axis rotates dependent primitives must follow suit.

Turning now to the second axis (light blue) shows that the base is linked primitive green and it has only one primitive dependence, the linked blue. If the base moves the axis follow, this is what allows the linking movements.

When we will use the tool we'll have to determine the necessary axes, their primitive base and their dependencies. You just have to adjust the rotations for each axis.



1.2 Constitution of the objects

The objects must be made of any linked primitives (classic, sculpts, meshes) but the tool is programmed to recognize the classic primitives :

- box type with or without cutting
- prism type

In both cases there will be an automatic recognition of borders and ease of positioning axes. In all other cases you'll have to adjust the axes either with the menu or manually with the viewer editor.

I suggest you focus on automatically recognized primitives for your convenience !

This is not really a constraint because you have a lot of possibilities:



On the other hand I assumed that uses flattened since you want to make origami but the tool can work with no flattened primitives and can serve other uses.

1.3 Scripts

The tool has eight scripts, 7 for the object:

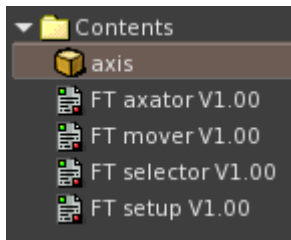
- **setup** : performs setup,
- **selector** : allows the selection of primitives,
- **notify** : reads configuration notecards and transmits information to the mover,
- **mover** : applies movement to primitive from information notecards,
- **player** : supports mover from commands generated by command,
- **axator** : manages axes for setup,
- **command** : interface with users.

And a script for the axes:

- **axis** : responsible for local management of axis,

1.3.1 Setup

In situation of setup we will use these scripts :



- **setup**
- **selector**
- **mover**
- **axator**

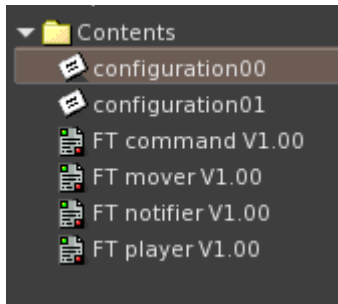
And also **axis** object.

The ultimate goal is to generate a notecard that will include all the information to pass from one state to another.

For several notecard (for several states) we will perform multiple setup.

1.3.2 Production

In situation of production we will use these scripts :



- **notifier**
- **mover**
- **player**
- **command**

And of course one or more numbered configuration notecards : 00 (so **configuration00**), 01 (**configuration01**), 02...

2. First setup !

For our first setup we will perform with the simple object of the previous chapter.

The object is included in the distribution folder as the **training01**. It consists of two conventional boxes and a prism.

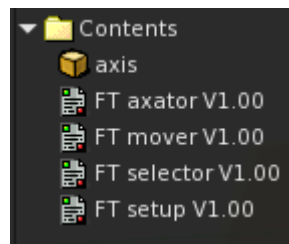
We'll create a setup to switch from start state to end state :



2.1 Initialization

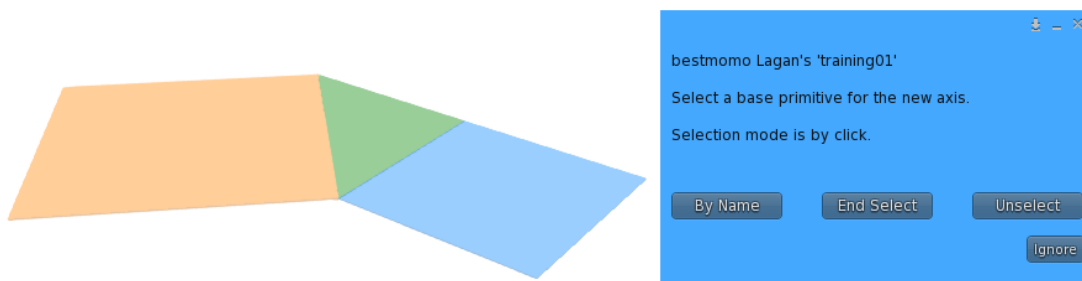
To init setup just put in the object these 4 scripts :

- **setup**
- **selector**
- **mover**
- **axator**



And axis object,

Primitives become transparent and a menu appears:



If accidentally you lose the menu you can select the object and reset scripts with the viewer menu. On the other hand if you accidentally lose it along the way just type /11select in public chat to get it back.

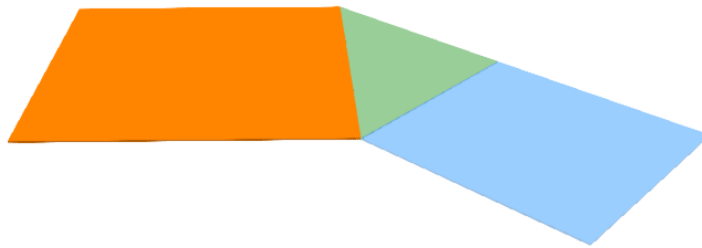
2.2 Setting axis

You are initially in a position of selecting a primitive, that's why they all became clear. That we will select will lose its transparency and we can easily locate it.

If you look at the menu it is reported: "Select a primary basis for the new axis", ie asked to select the basic primitive for the new axis, it'll be our first axis.

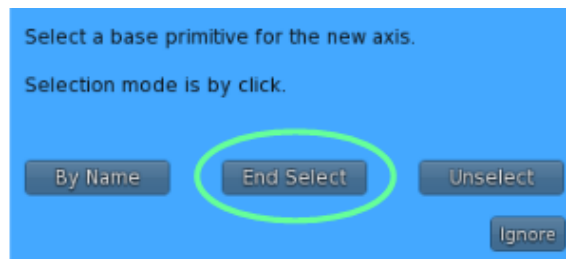
We know we will have to set up two axis, we can do it in any order because the system will rank at the end to meet the dynamic hierarchy.

Select the orange primitive (root) by clicking on it, it becomes opaque:

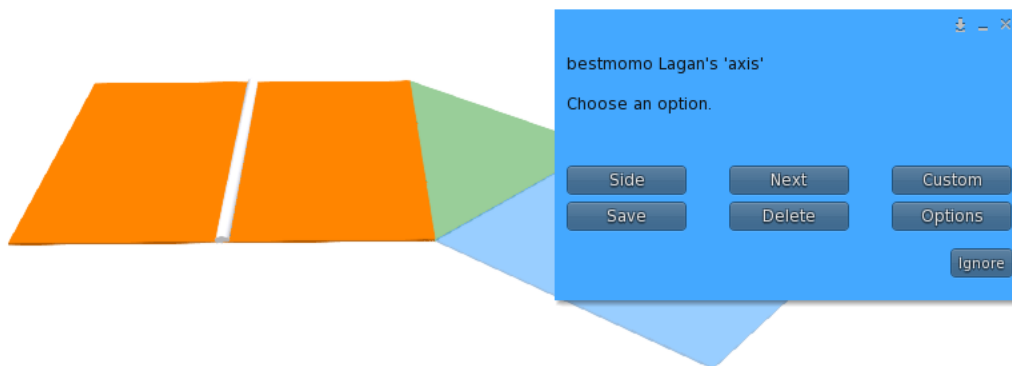


If you make a mistake simply continue to click, the last remaining primitive is selected.

To confirm the selection click on **End Select** :



An axis appears at the center of the basis primitive aligned with its axis X and the menu changes:



There are several options:

Side : send the axis on one side,

Next : change the positioning of the axis (if it's in the center center it lines up on the other axis, so X or Y, and if it is on an edge it goes to the next edge and so on) ,

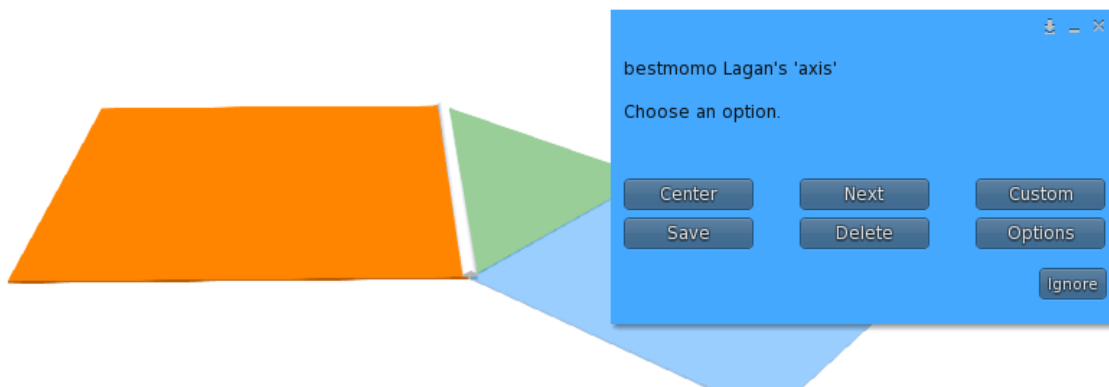
Custom : if you have the primitive is not recognized displays a menu positioning,

Save : save the axis in its position,

Delete : delete the axis permanently (if you made a mistake),

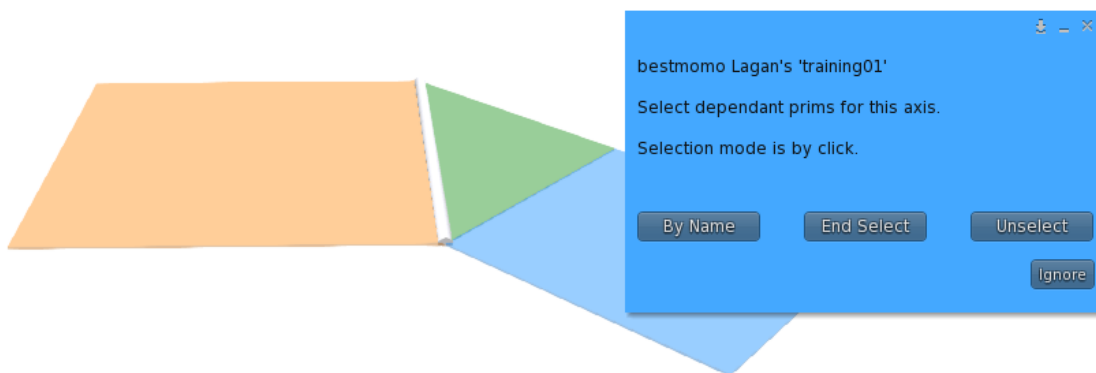
Options : opens the menu to change the size and color of the axis, to differentiate it when there are many.

In our case we will click on Side because we want the axis on one edge:



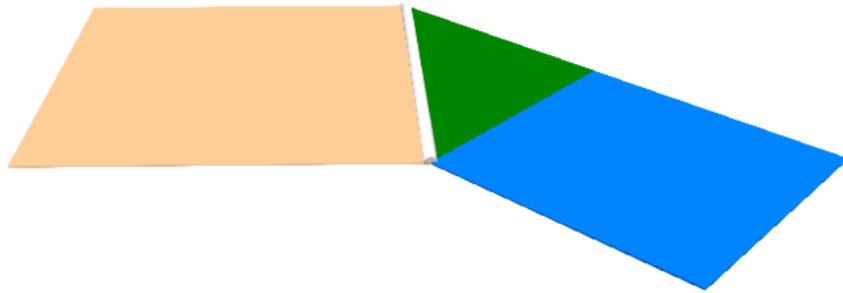
The axis runs directly on the good edge, if it were not the case it would be enough to click **Next** to be on the right edge. Notice the change in the text of the first menu button (now **Center** if you want to return the axis in the center of the primitive).

As the axis is perfectly positioned you click **Save** to save it:

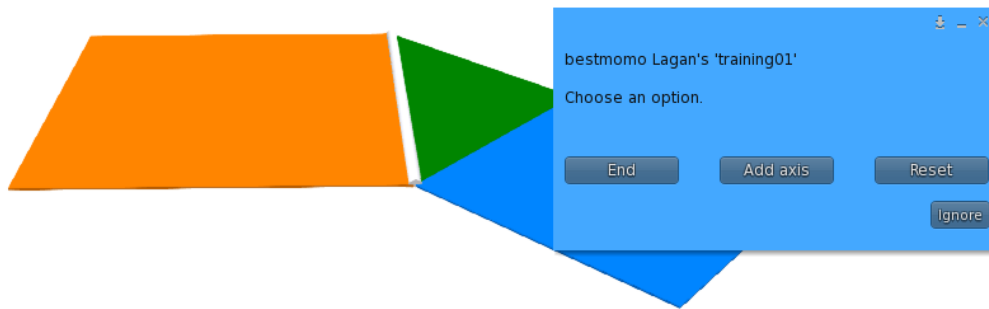


The axis remains in its position and we get again the selection menu but this time to select the dependent primitives for this axis. We know it must be related both primitives (green and blue).

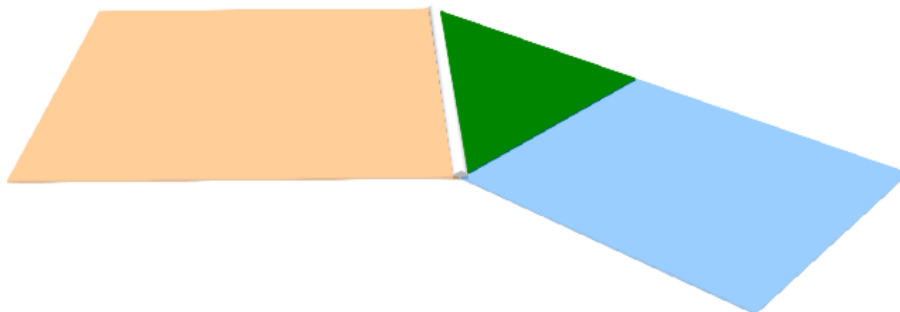
So we will click successively on these two primitives to select them :



The click on **End Select** :



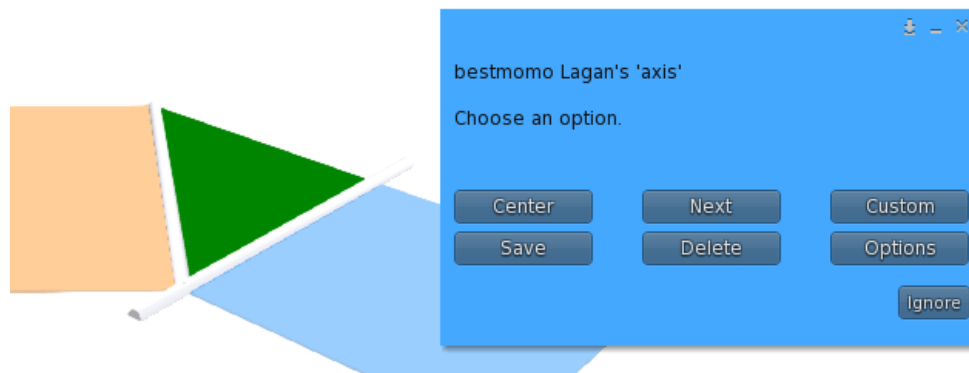
We get the menu to add a new axis. We click on **Add axis** and this time on the green primitive:



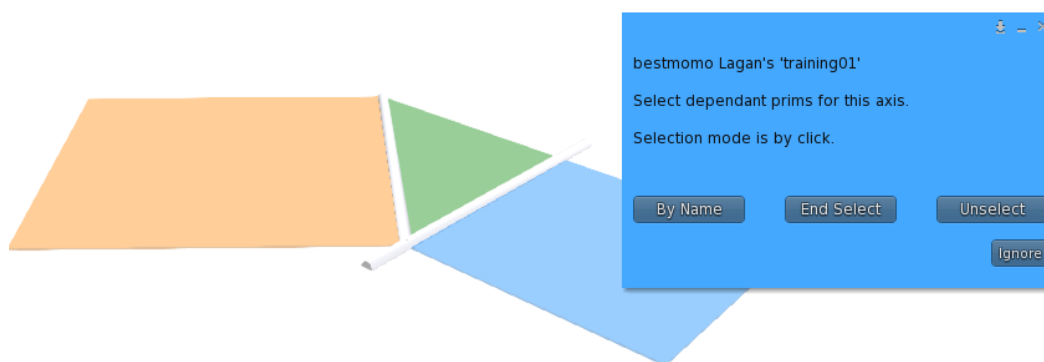
Green primitive becomes opaque, click **End Select** :



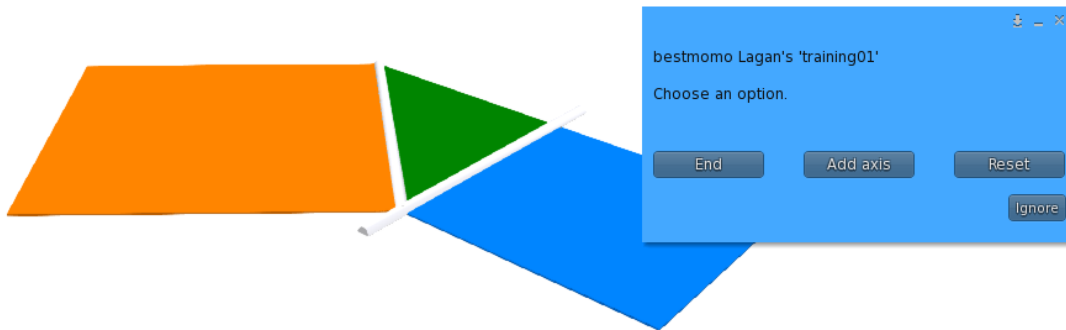
The new axis appears in the center of the green primitive. Click on **Side** to send to the side:



Then click **Save** pour save axis :



Now must select dependencies for this axis. This is only the blue primitive, so you click it to select it and then choose **End Select** :



We now have our two axis, we click on **End** because we have finished this part of the setup.

You get these messages in public chat :

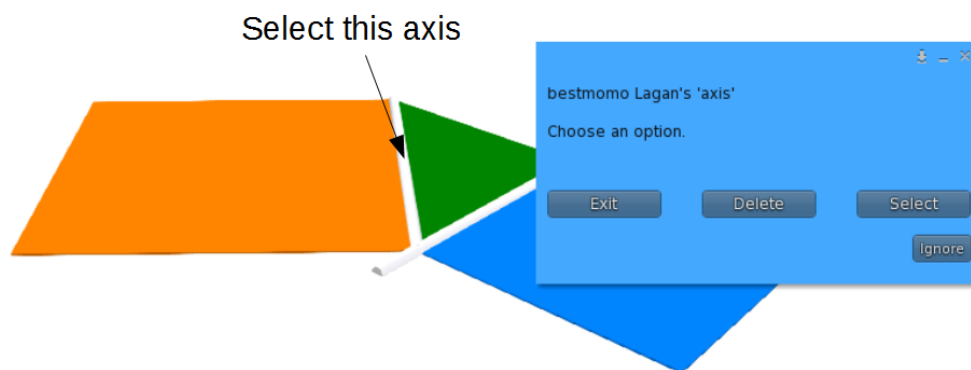
[05:03] training01: Ordering axis. Please wait...

[05:03] training01: Select an axis to set rotation

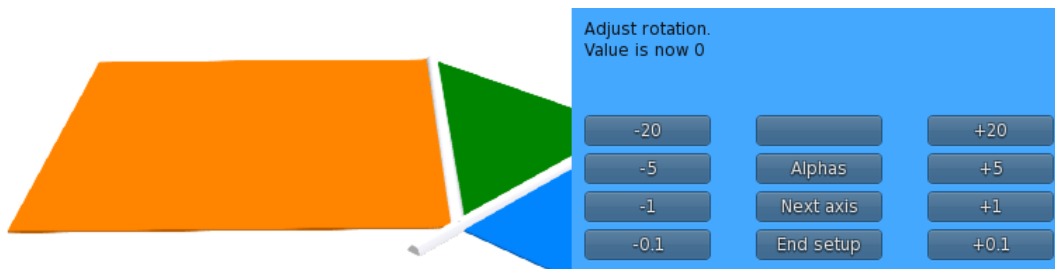
The part of implementation of the axes is now complete, we will move on to setting rotations.

2.3 Rotations

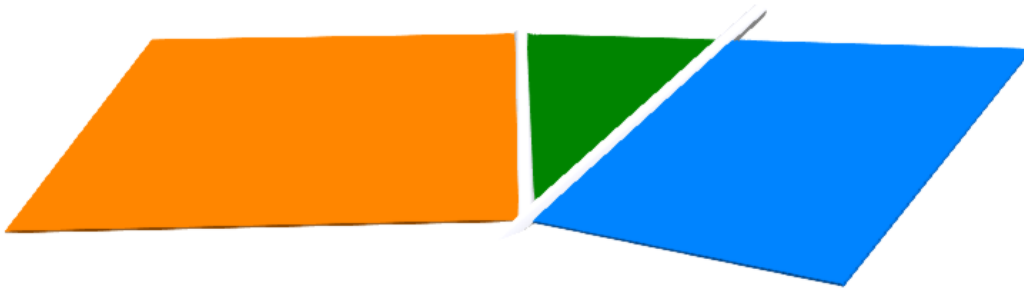
Click on this axis (with orange primitive as basis) :



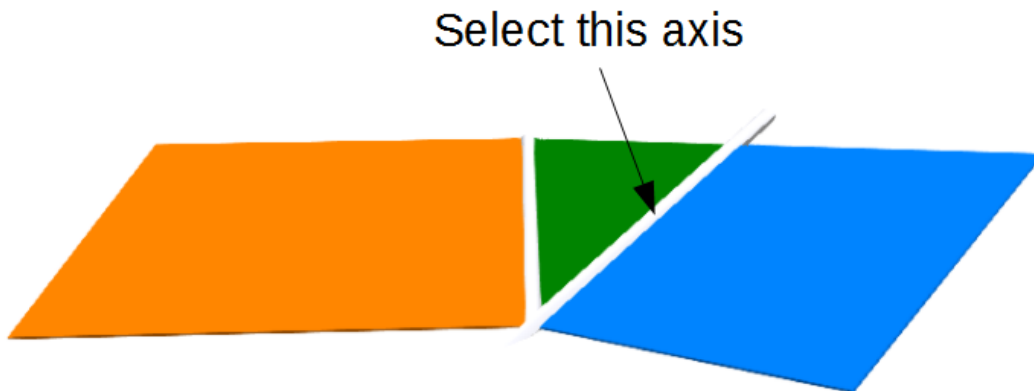
In menu click on **Select** because you want to select this axis. You get the menu to adjust rotation :



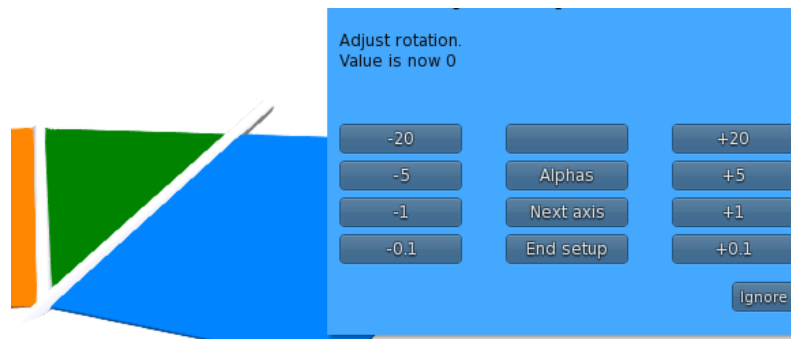
Click on **-20** button to set a negative rotation of 20 degrees :



In menu click on **Next axis** and then click on second axis :



In menu click on **Select**, you get again the rotation menu but for the second axis :

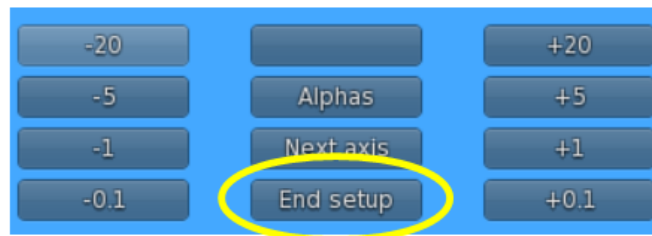


Click on **+20** button to set a positive rotation of 20 degrees :

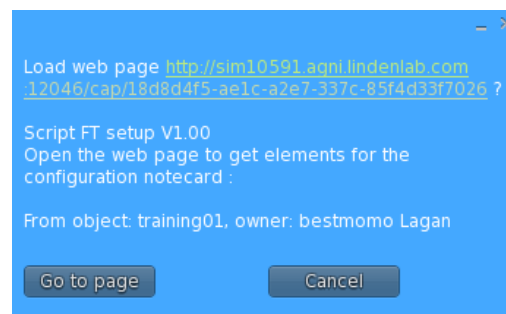


2.4 Configuration notecard

As we finished adjusting the rotation choose the button **End setup** :



You get a new menu :

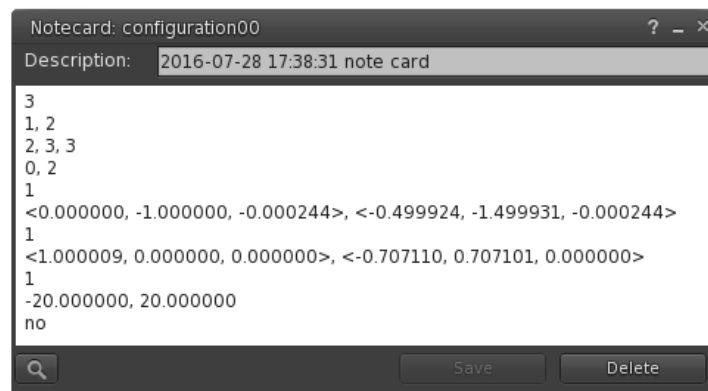


Choose **Go to page**. Normally you open your browser (internal or external to the viewer) with this text:

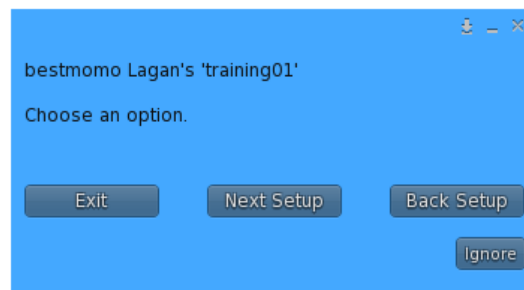
Copy the lines bellow in a notecard called "configuration" with number 00, or 01...

```
3
1, 2
2, 3, 3
0, 2
1
<0.000000, -1.000000, -0.000244>, <-0.499924, -1.499931, -0.000244>
1
<1.000009, 0.000000, 0.000000>, <-0.707110, 0.707101, 0.000000>
1
-20.000000, 20.000000
no
```

Create a note **configuration00** with a copy of the values :



The setup is now complete and this menu is obtained :

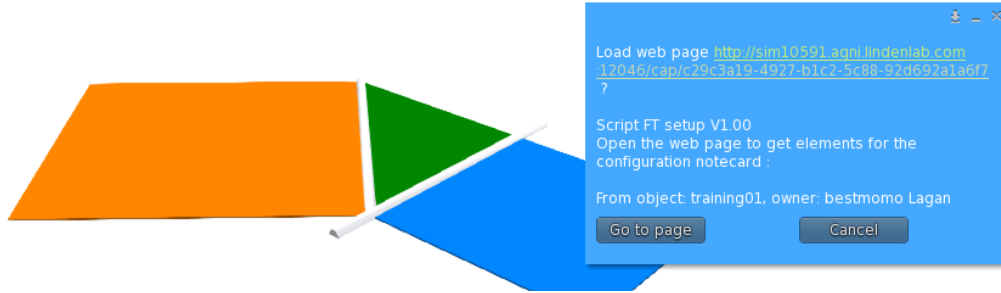


You can choose :

- **Exit** : everything ends,
- **Next Setup** : we leave for a new setup from this state retaining the axes,
- **Back Setup** : it creates the setup back to the starting state.

Choose **Back Setup** to get the inverse setup.

The object returns to its original state, as well as axes and you get the end setup menu :

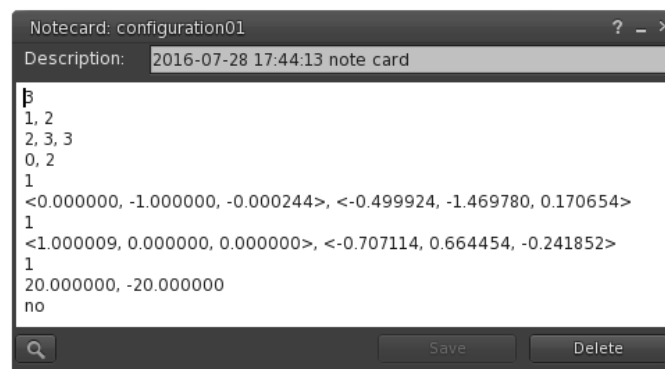


Choose **Go to page** to get values for the new notecard :

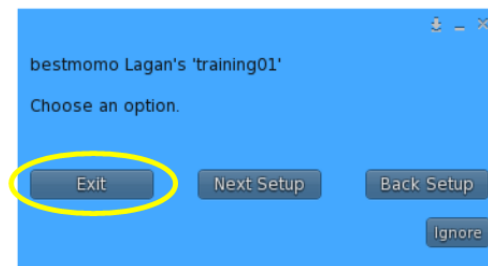
Copy the lines bellow in a notecard called "configuration" with number 00, or 01...

```
3
1, 2
2, 3, 3
0, 2
1
<0.000000, -1.000000, -0.000244>, <-0.499924, -1.469780, 0.170654>
1
<1.000009, 0.000000, 0.000000>, <-0.707114, 0.664454, -0.241852>
1
20.000000, -20.000000
no
```

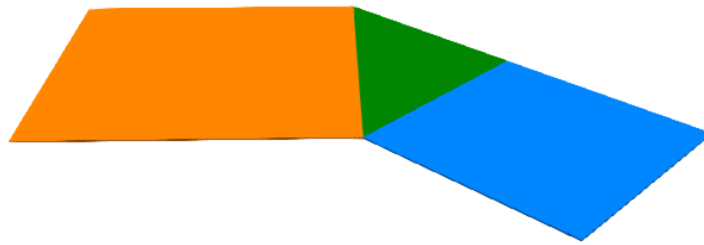
Create a note **configuration01** with these values :



Now choose **Exit** :



We finished the setup and axis disappear.



2.5 Action !

We now have everything we need to run our object !

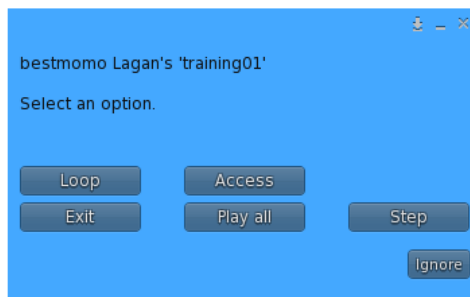
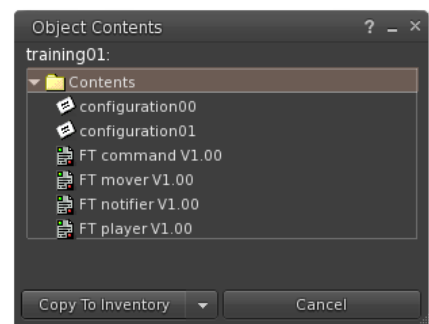
In the object remove the setup scripts (you can keep **mover** that is identical) and put these scripts:

- **notifier**
- **mover**
- **player**
- **command**

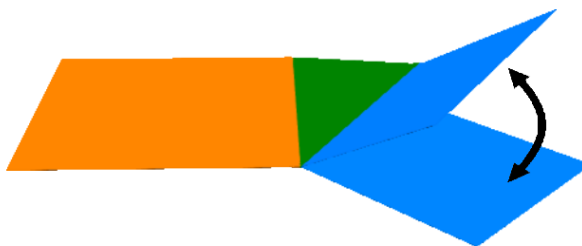
Add the two notes :

- **configuration00**
- **configuration01**

Click the object, you get a menu :



Choose **Play all**. The object gradually go to to the end state and then returns to the starting state :



What happened ?

The system read the note and applied **configuration00** rotations, it then read the note and applied **configuration01** rotations. As we planned a return trip the result is the one expected.

If instead of choosing **Play all** you click **Loop** then the movement is repeated indefinitely until you click again to get the menu and you choose **Stop**. The stop is performed when returning to the starting position.

Another possibility is **Step**, in which case it only reads a note and apply rotations. So use twice **Step** to do the full movement.

The menu also includes an **Access** option to restrict access to the menu to group or owner.

2.6 Details of the values of the note

There is only one value you have to change, the first line. You may notice that for the two notes the value is always 3. This is the time for movement and the default is 3 seconds.

You can change this value, even with a point number like 1.7 for example.

You do not normally have to change the other values, if you do it it's at your own risk! For information I give you the summary meaning:

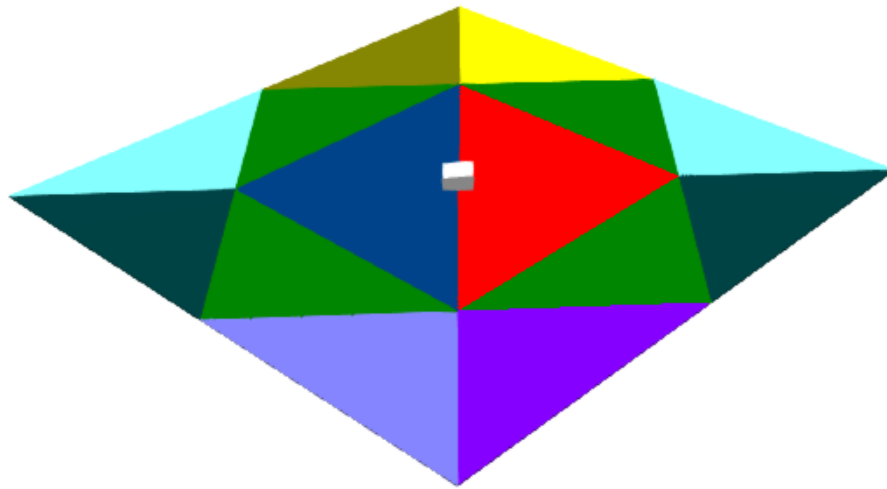
- **line 2** : numbers of basic primitives axis, in our case it is 1, 2
- **line 3** : numbers of dependent primitives
- **line 4** : index numbers of dependent primitives, in our case it starts at 0 for the first axis, and 2 for the second
- **line 6** : local axis positions
- **line 8** : axis vectors
- **line 10** : arrival rotations for axis

3. Cocotte

Now we have seen a simple example of how the tool works let's do something a little more ambitious : the cocotte ! A video is available here : <https://www.youtube.com/watch?v=fg9yiGWv308>.

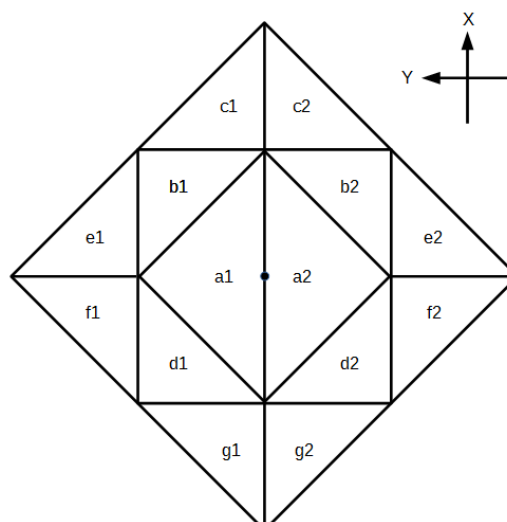
3.1 Object

You can find in the distribution folder the object **training02** :



This is the base object to achieve the cocotte. It consists of 15 conventional primitives. It has a root center primitive that will serve as fixed reference.

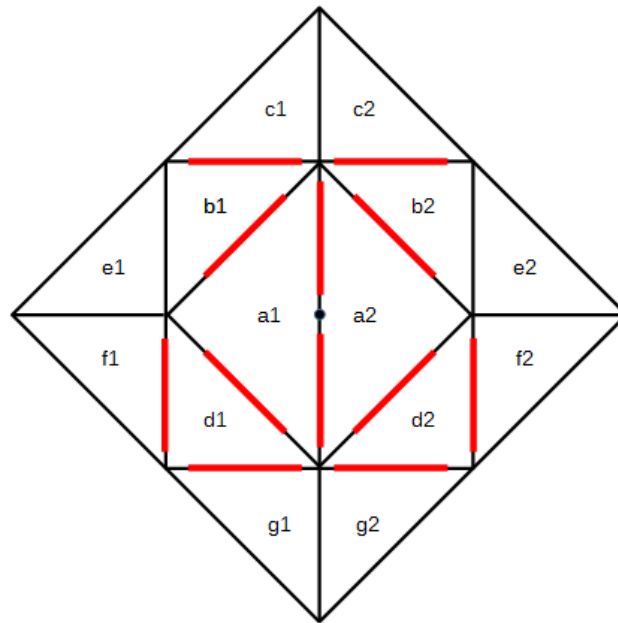
For setup requirements primitives have all been named so :



We may need these names for selections. Sometimes it becomes difficult to click on a primitive, it is easier to use its name. It is not essential for the cocotte but it allows us to better manage the representation of movement.

3.2 Axis

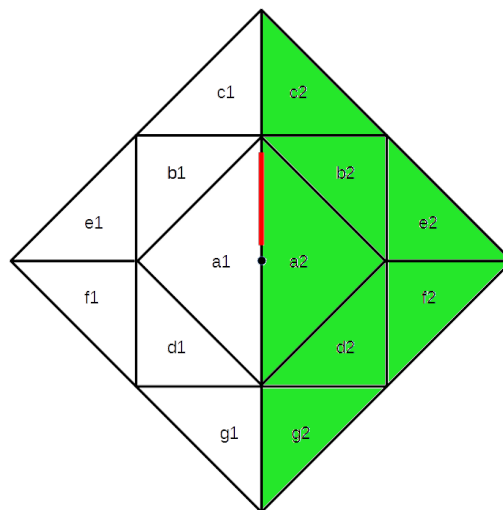
To achieve the movement it is going to need 12 axis. Here they are shown schematically:



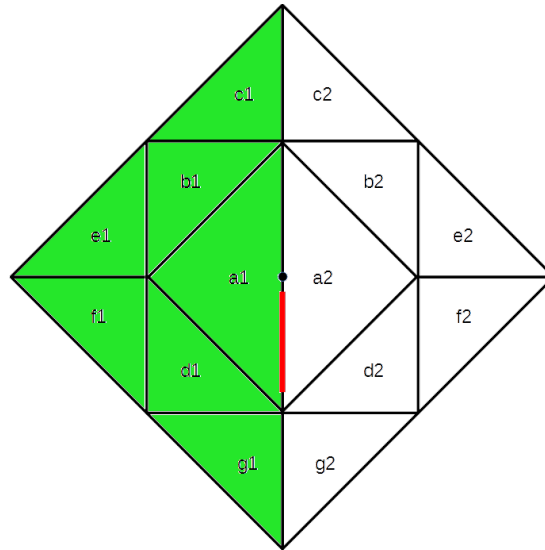
Note that you need two central axis.

For any project you will need and determine the necessary axis but also their dependencies.

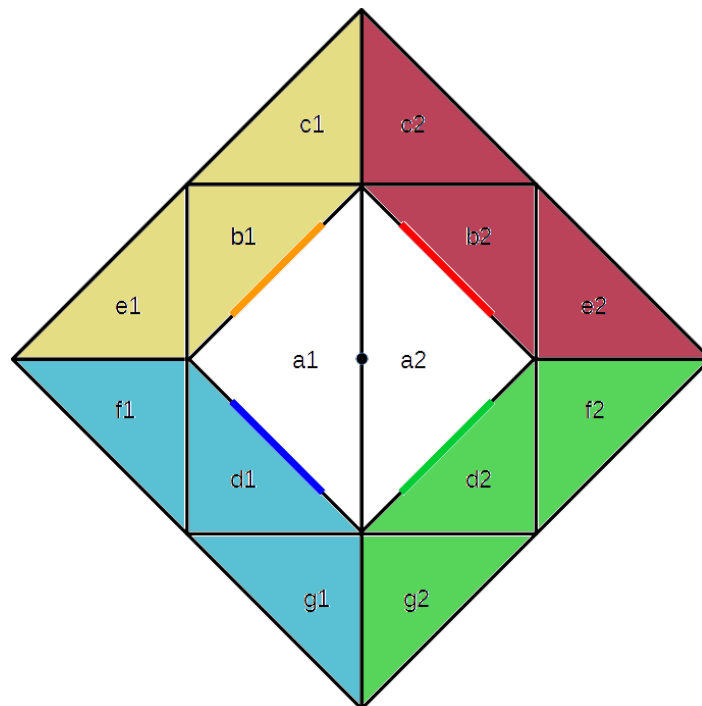
Let's see the dependencies for the axis of the cocotte :



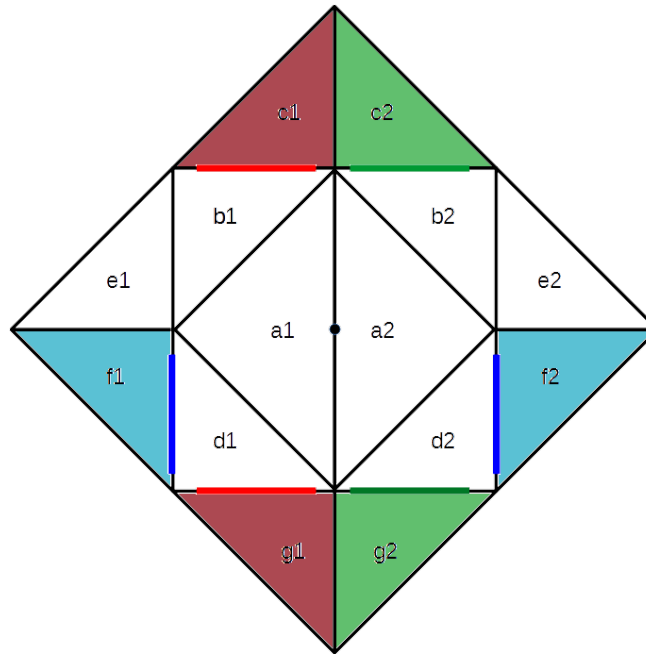
The first central axis as dependencies all primitives on the right. Similarly the other central axis as dependencies all primitives on the left :



Here are the dependencies for four oblique axis :



And finally here are the latest dependencies :



3.3 Setup

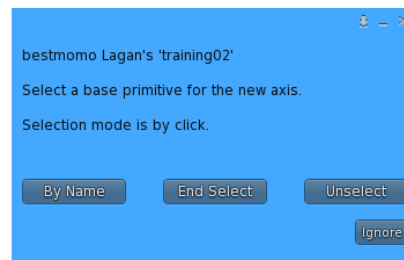
We will perform the setup like we did in the previous chapter, we have just a little more axis !

The first thing to do is to put in the object the elements necessary for the setup (scripts and axis):



3.3.1 Axis

We get the menu to add axis :



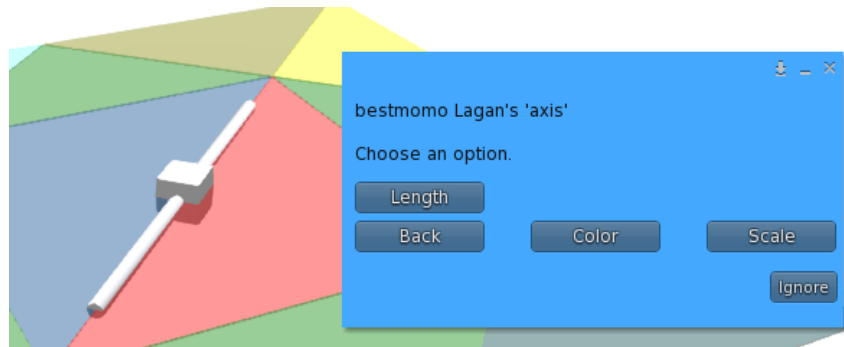
As I said before there is no order to respect for axis creation, The system at the end manage everything for it to work.

However it seems more logical to me to go from the center to the periphery, this is also the order that is set for the operation.

I will not detail the creation of the axis as I did in the previous chapter because it's always the same process.

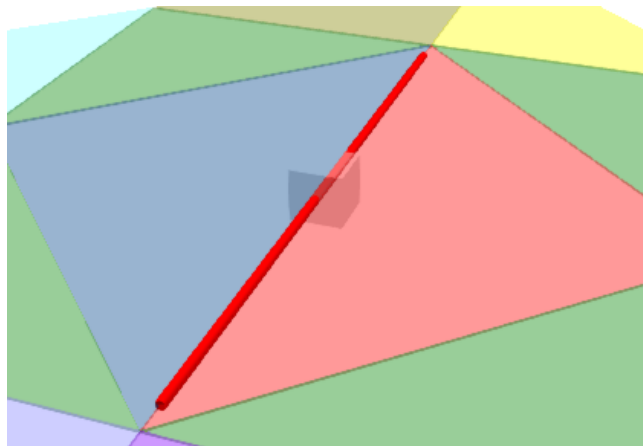
We have seen that we will have two axis in the center. We'll have to avoid their bunk or you'll have trouble selecting one or the other !

Create the first by selecting the primitive root in the center and then **Save**. Then choose **Options** from the menu:



Here you can change length, size and color of the axis. It is not necessary to go through the menus and you can use the viewer editor to do so.

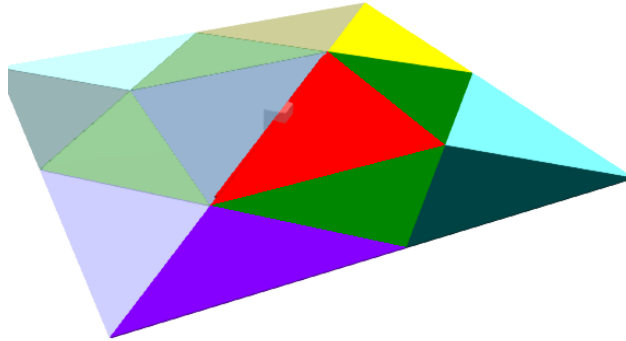
What I advise you in this case is to make the axis thinner, longer and of a different color. For example, with this result:



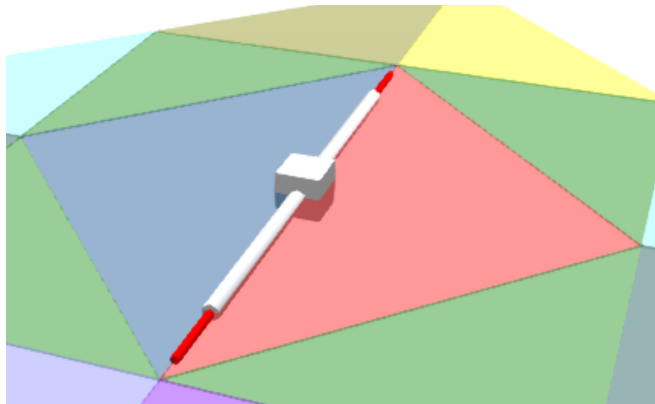


It is also possible to make a local displacement of the axis on its Z axis for the shift, so it will not be superimposed on the next axis.

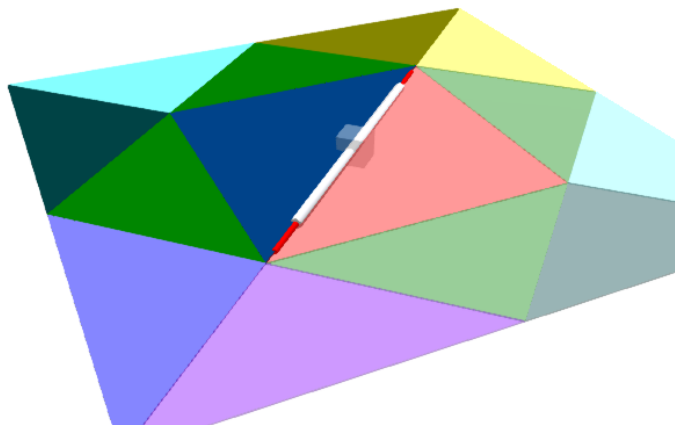
Save the axis (**Save**) and then select the dependent primitive by clicking it :



Click **End Select** and create the second axis (**Add axis**) by selecting the primitive root again. As you have taken the precaution of changing the first axis you should not confuse :

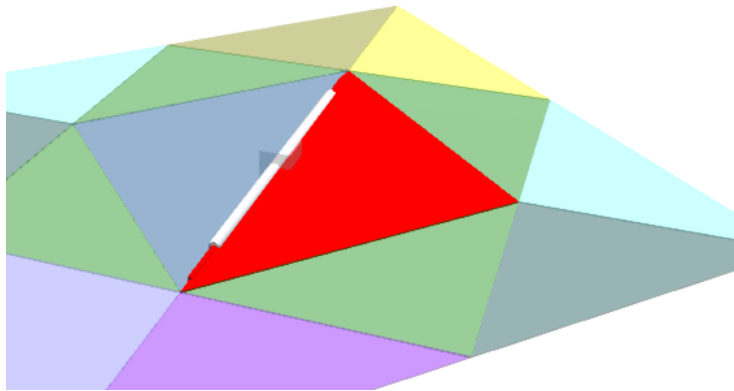


Establish dependency with all primitives on the left :

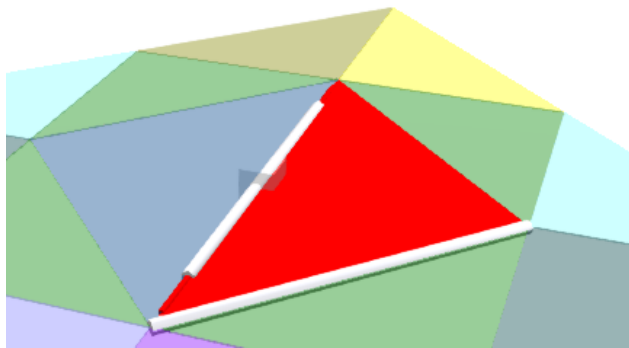


Click **End Select** to confirm.

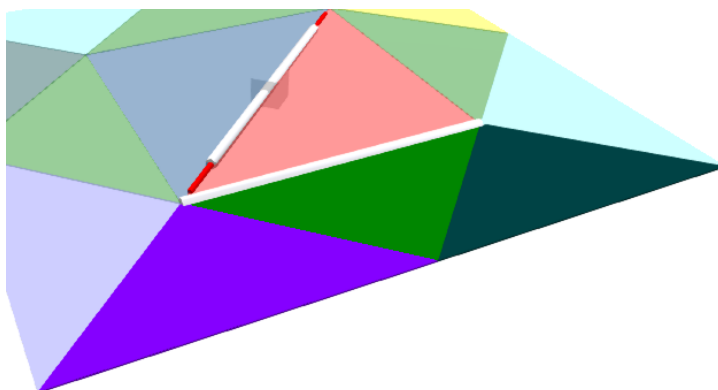
Click **Add axis** again and select this primitive :



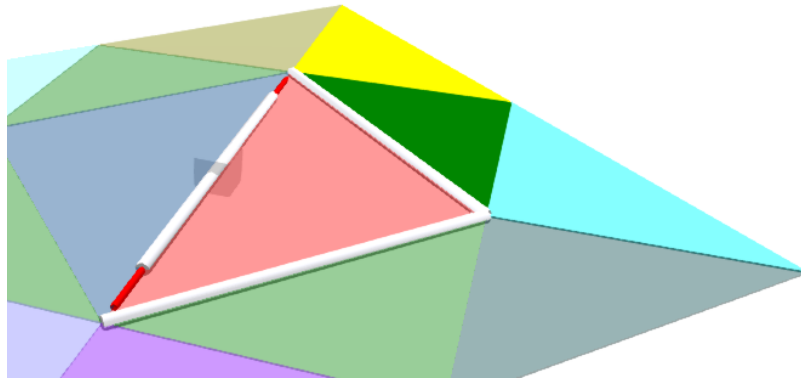
Select **End Select** and **Side** to send the axis to the edge :



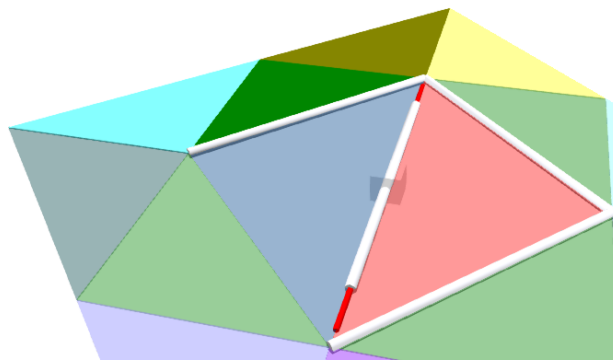
Click **Save** and select the dependent primitives :



Create the same way other axis on this primitive with dependencies (this time you will have to click on **Next** after **Side**) :



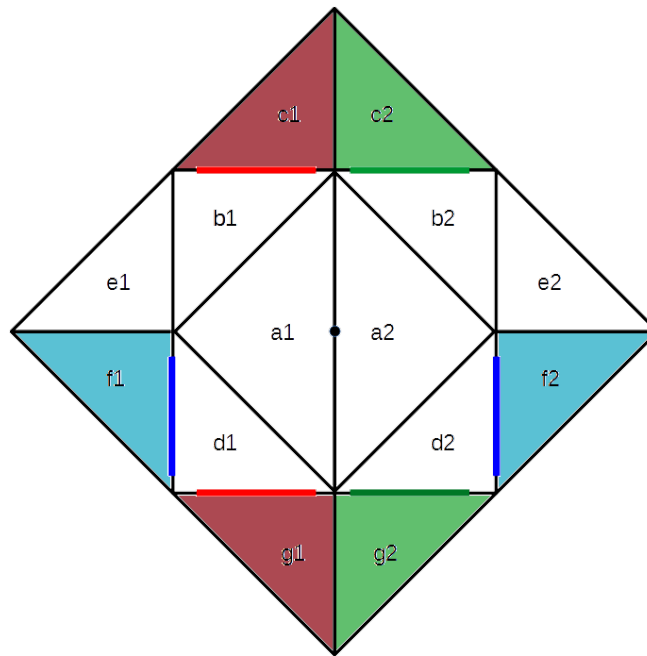
In the same manner create the axis of the other side with its dependencies :



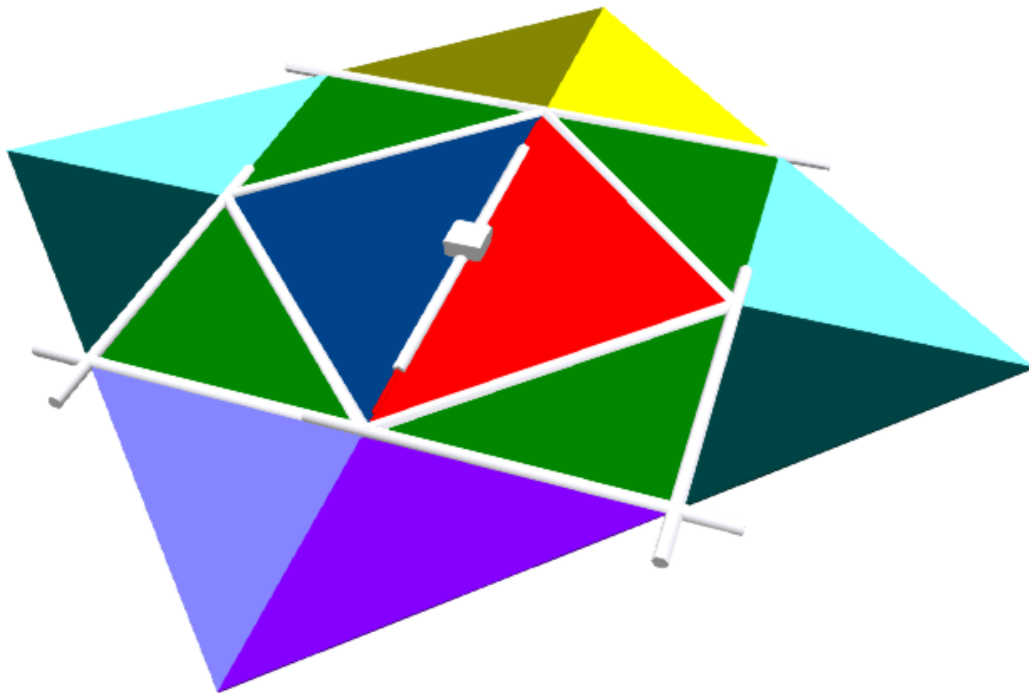
And the other :



Continue with the other axis and their dependencies as described in this figure :



You must ultimately have something like this :

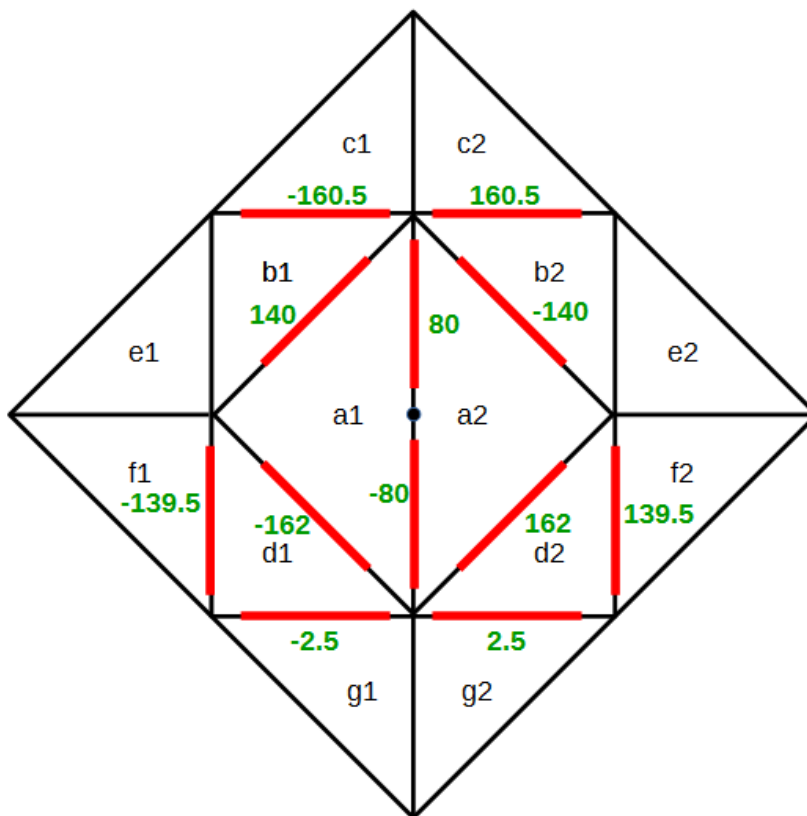


Then click on **End** to complete this part of the setup.

3.3.2 Rotations

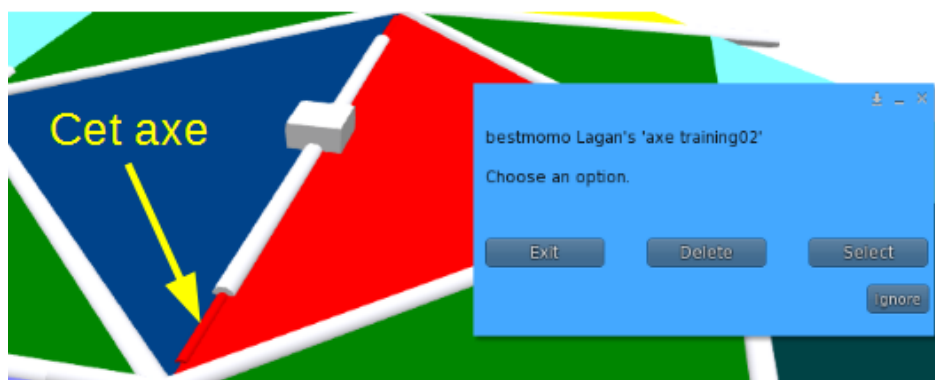
Now the rotations.,,,

Here is a diagram with the rotations mentioned in green around each axis :



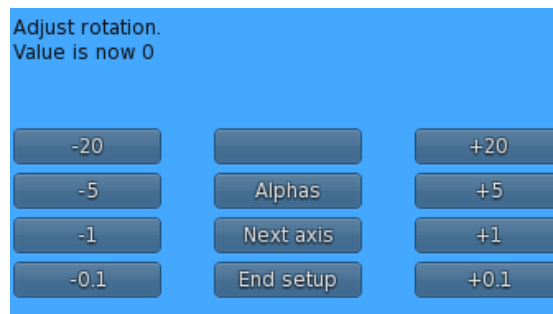
We saw in the previous chapter how to adjust the rotations : just select an axis and use the buttons to change the value of the rotation.

I suggest you start with the two central axis, for example the first one set up and which has been changed appearance :



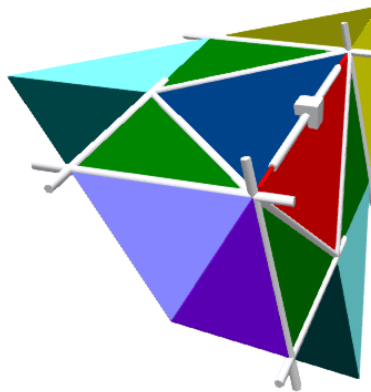
Click on **Select** to select it.

So you get the menu for the rotations :



As we want 80 degrees we must click 4 times on **+20** button.

All dependencies of this axis (primitives and their axes) follow the movement :



Click on **Next axis** and select the other central axis. This time set rotation -80 degrees :

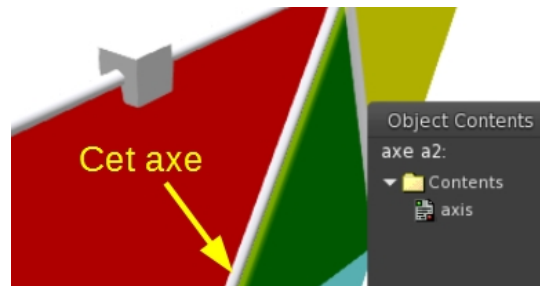


It is our first folding !



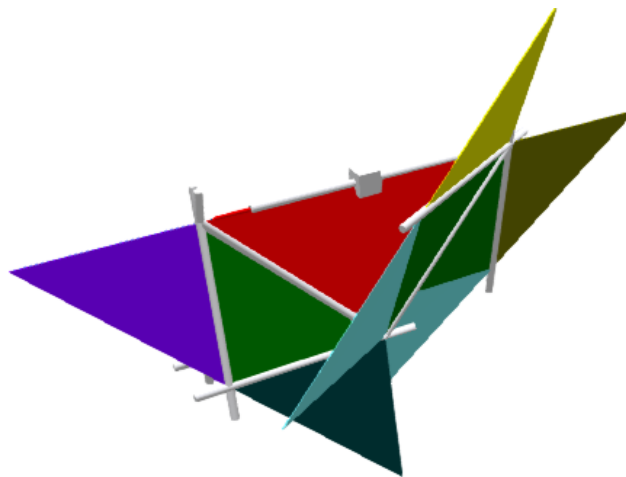
*There is a faster way to get to the desired rotation. Just use the public chat and enter the value preceded by rot. For example for 80 degrees we enter **rot80**!*

Continue adjusting rotations in the order you want. Note that axis have made their name with that of their basic primitive. For example if we look at the name on an axis on **a2** primitive :



This axis is called **a2**. It may be useful to distinguish when axes made some rotations and primitives are nested.

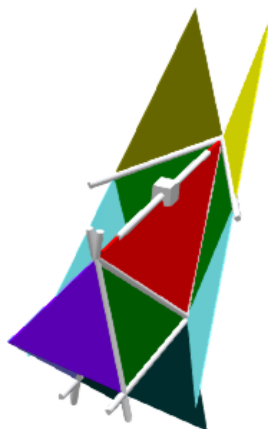
You can also continue setting rotations with this axis, we know it takes a rotation of -140 degrees. After the adjustment you have to get something like this :



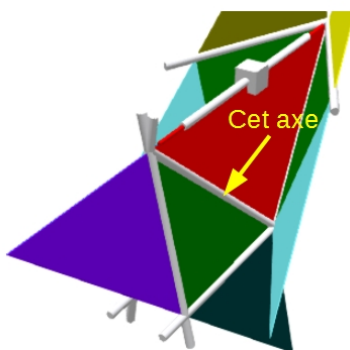
As you can see the situation become a bit confusing and naming primitive and axis can help you !

Set in the same manner the rotation of the symmetrical axis with 140 degrees.

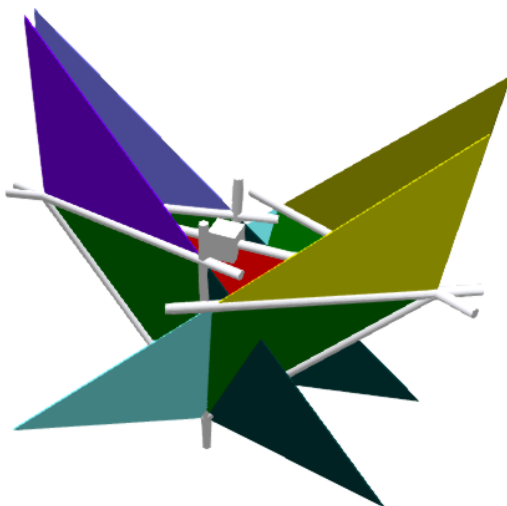
You should be there :



Continue with this axis and a rotation of 162 degrees :

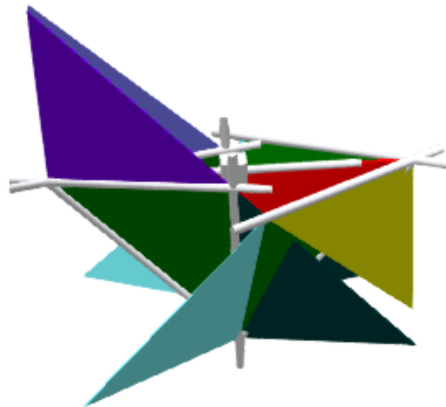


And symmetrical with -162 degrees. You should be there :



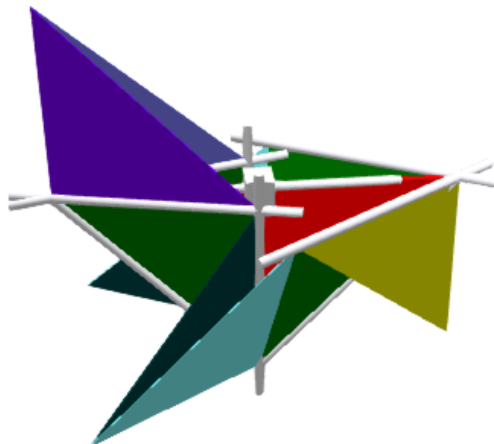
We solved half rotations !

We will move forward with the two primitives (c1 and c2) by adjusting the axis of rotation (b1 axis and b2 axis) the values 160.5 and -160.5 :

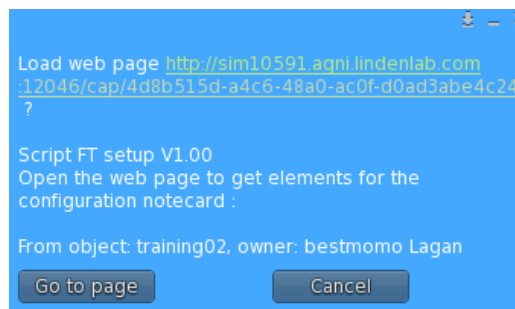


Continue with the last 4 axis by taking the values in the diagram I made at the beginning of this chapter for the rotations.

In the end the cocotte must be well formed even if it leans dangerously :



Then click on **End setup**. You get this menu :



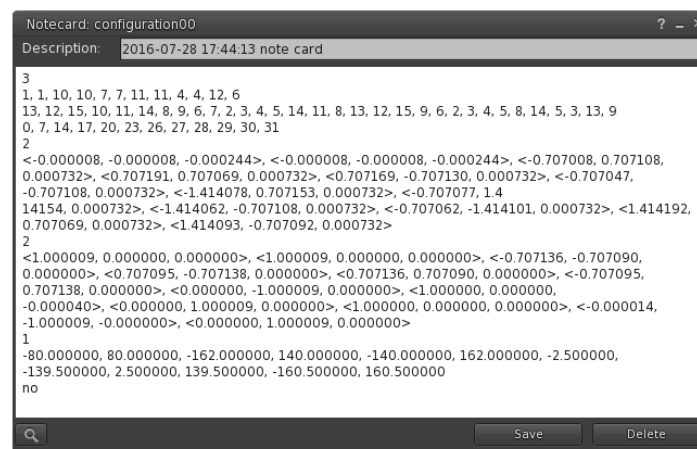
Click on **Go to page** to open the browser and get the values for the notecard,

3.3.3 Notecards

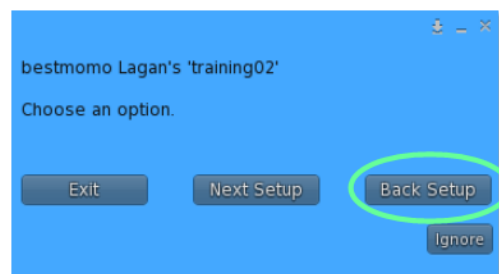
Inevitably you will recover many values :

```
3
1, 1, 10, 10, 7, 7, 11, 11, 4, 4, 12, 6
13, 12, 15, 10, 11, 14, 8, 9, 6, 7, 2, 3, 4, 5, 14, 11, 8, 13, 12, 15, 9, 6, 2, 3, 4, 5, 8,
14, 5, 3, 13, 9
0, 7, 14, 17, 20, 23, 26, 27, 28, 29, 30, 31
2
<-0.000008, -0.000008, -0.000244>, <-0.000008, -0.000008, -0.000244>, <-0.707008, 0.707108,
0.000732>, <0.707191, 0.707069, 0.000732>, <0.707169, -0.707130, 0.000732>, <-0.707047,
-0.707108, 0.000732>, <-1.414078, 0.707153, 0.000732>, <-0.707077, 1.4
14154, 0.000732>, <-1.414062, -0.707108, 0.000732>, <-0.707062, -1.414101, 0.000732>,
<1.414192, 0.707069, 0.000732>, <1.414093, -0.707092, 0.000732>
2
<1.000009, 0.000000, 0.000000>, <1.000009, 0.000000, 0.000000>, <-0.707136, -0.707090,
0.000000>, <0.707095, -0.707138, 0.000000>, <0.707136, 0.707090, 0.000000>, <-0.707095,
0.707138, 0.000000>, <0.000000, -1.000009, 0.000000>, <1.000000, 0.000000,
-0.000040>, <0.000000, 1.000009, 0.000000>, <1.000000, 0.000000, 0.000000>, <-0.000014,
-1.000009, -0.000000>, <0.000000, 1.000009, 0.000000>
1
-80.000000, 80.000000, -162.000000, 140.000000, -140.000000, 162.000000, -2.500000,
-139.500000, 2.500000, 139.500000, -160.500000, 160.500000
no
```

Create the **configuration00** note with these values :

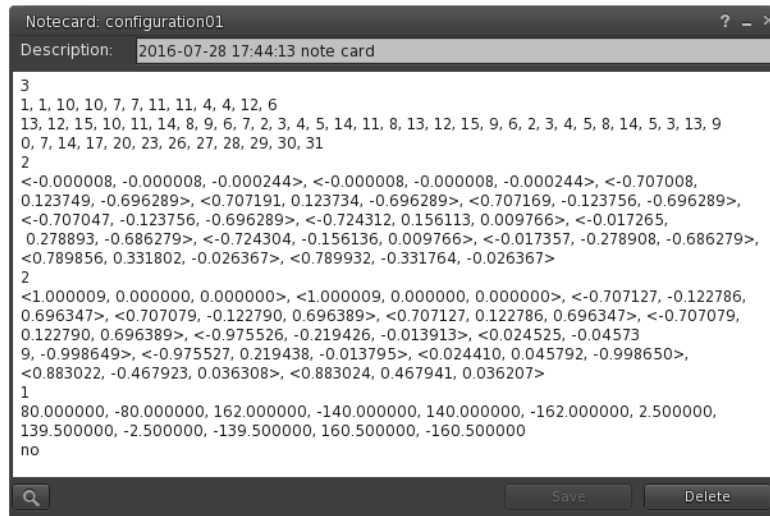


In menu click **Back Setup** to create inverse setup :



The cocotte back to its original state and you have an identical menu to pick values.

Create the **configuration01** notecard with these new values :



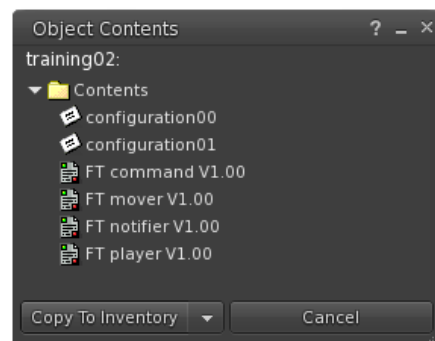
Click on **Exit**. The setup is now complete !



As I have already mentioned the value on the first line of the note, which is 3 by default, represents the duration of the movement in seconds and can be freely modified. But do not expect high accuracy from this side because the movement is very dependent on available resources.

3.4 Action

Simply slide into the cocotte both notes and the following scripts :

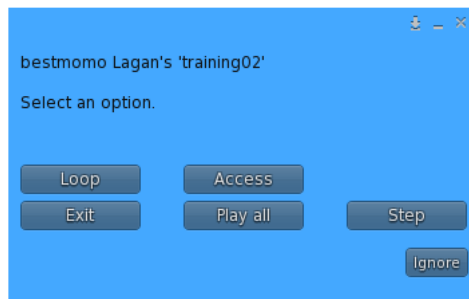


Note that the command script (command) and readable and writable. You can create a custom command. In the header you have parameters you can change safely :

```
// -----
//   Parameters you can change
// -----
```

```
string ACCESS_MENU   = "All"; // Must be "All", "Group", "Owner"
string ACCESS_OWNER  = "Sorry but access is reserved to owner !";
string ACCESS_GROUP  = "Sorry but access is reserved to group. You must activate it !";
integer MOVE_LOOP    = TRUE; // Must be TRUE or FALSE
```

An important parameter is **MOVE_LOOP**. The default is **TRUE**. This means that the setup was performed as a round trip with two notes. Therefore the menu suits :



There is the **Loop** button to start an infinite movement, which would not be the case if the parameter **MOVE_LOOP** is set to **FALSE**.

If you set the wrong value you may get unpredictable results with deformations.

The command script has delivered basic possibilities with:

- **Play all** : to start all successive movements (so long as there are notes to read)
- **Step** : reading a note and movement,
- **Loop** : infinite playback (a **Stop** button is then provided in the menu to stop the movement).

Furthermore a summary of the programming menu access is provided to limit to the group or the owner.

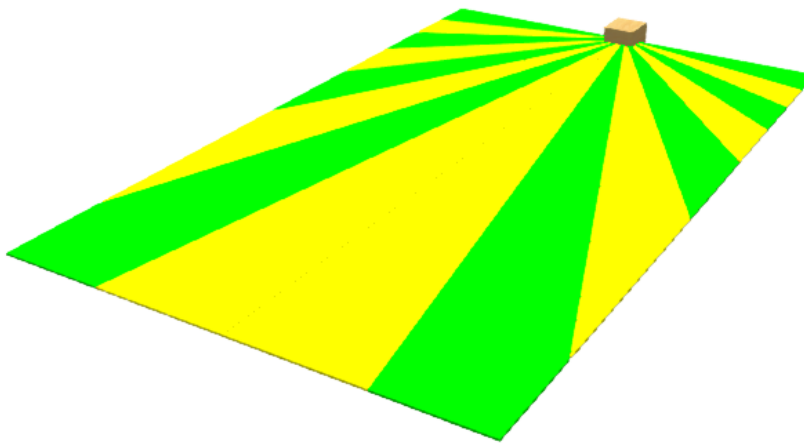
You obviously have the freedom to change all that at your convenience! I will detail elsewhere in a later chapter the information to send to the **mover** script

4. The plane

Let us now another example of three successive setups with certain primitives set alpha. A video is available here : <https://www.youtube.com/watch?v=fg9yiGWv308>.

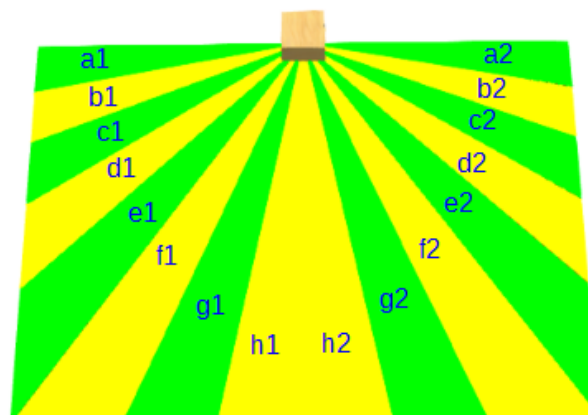
4.1 Object

You can find in the distribution folder the object **training03** :



This is the base object to realize the plane. It consists of 17 classic primitives. It has a root center that will serve as fixed reference. Primitives related were cut for the succession of forms.

For setup requirements primitives have all been named so :



4.2 Setup 1

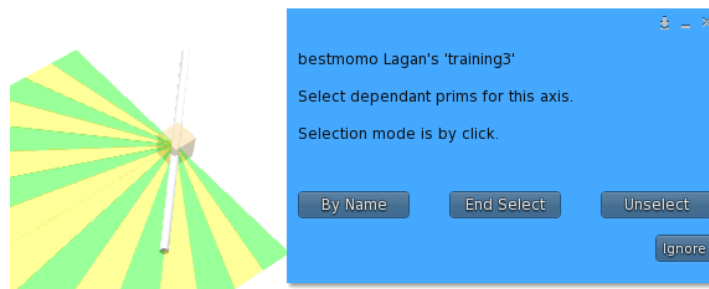
For the first setup we first put the scripts in the object :



Click on this primitive then on **End Select** :



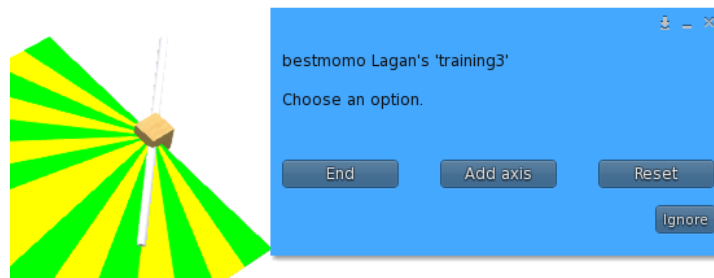
Click on **Side** , **Next**, **Next** and **Save** :



Select these 4 primitives with clicks :



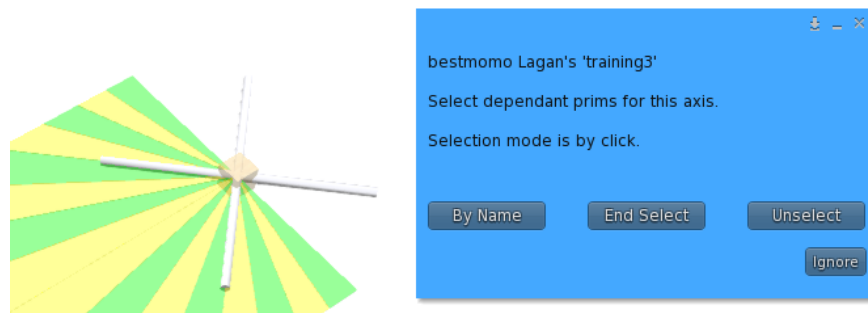
Click on **End Select**, your first axis is OK :



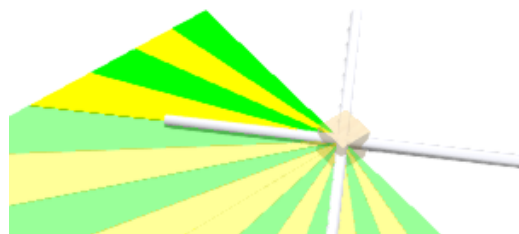
Click on **Add axis** and choose this primitive :



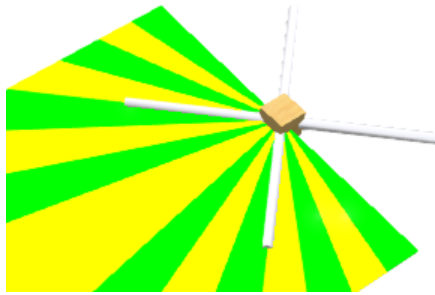
Click on **End Select** and proceed as for the previous axis to set it (**Side** , **Next**, **Next** and **Save**) :



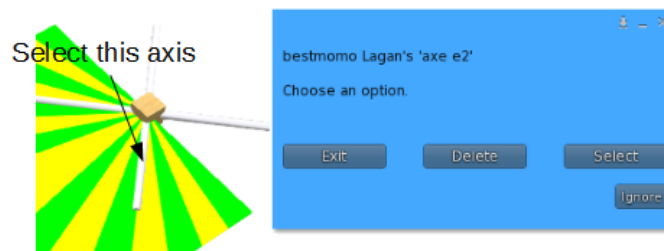
Choose these 4 primitives :



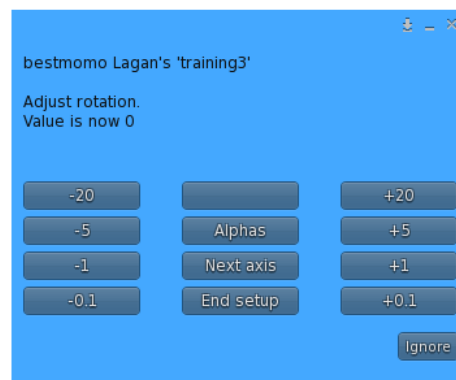
Click on **End Select** and **End** :



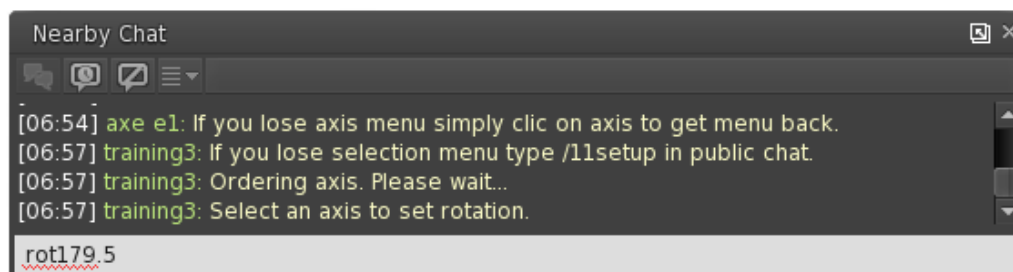
The axes are in place we can go to rotations. Click on this axis :



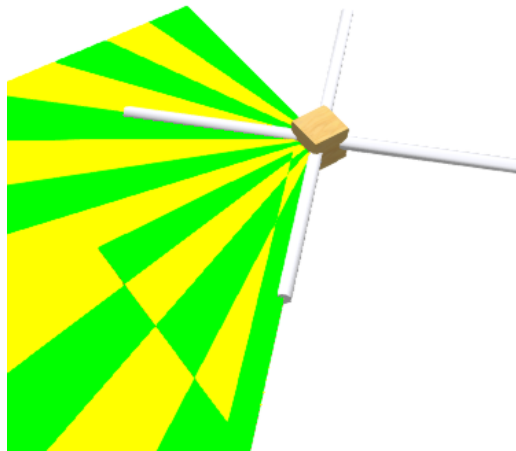
Click on **Select**, you get the rotations setting menu :



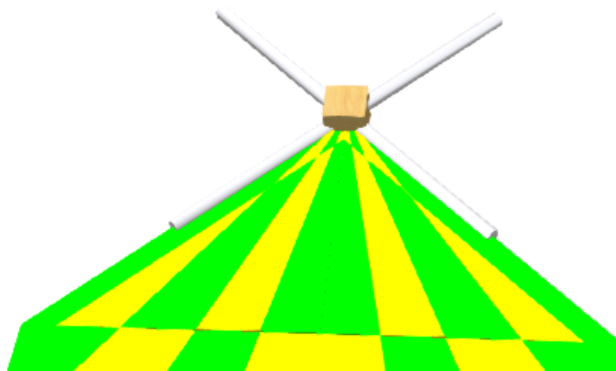
We will not use this menu but enter the value in the public chat :



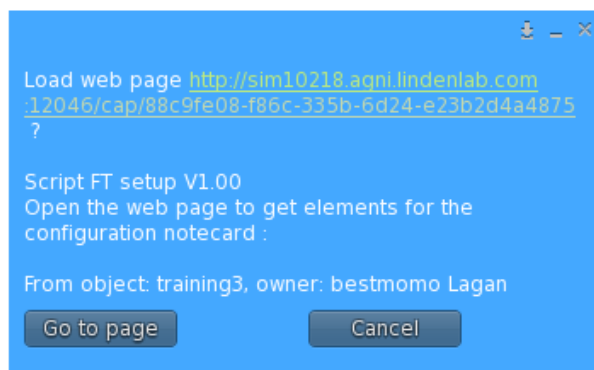
The first fold is done :



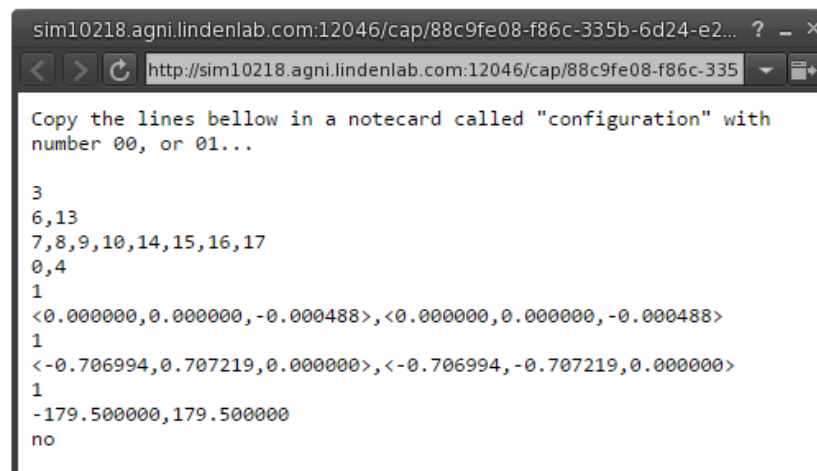
Do the same with the other axis (**Next axis** and selection) with a rotation of -179,5 degrees :



Then click on **End setup** :



Click on Go to page to open the browser



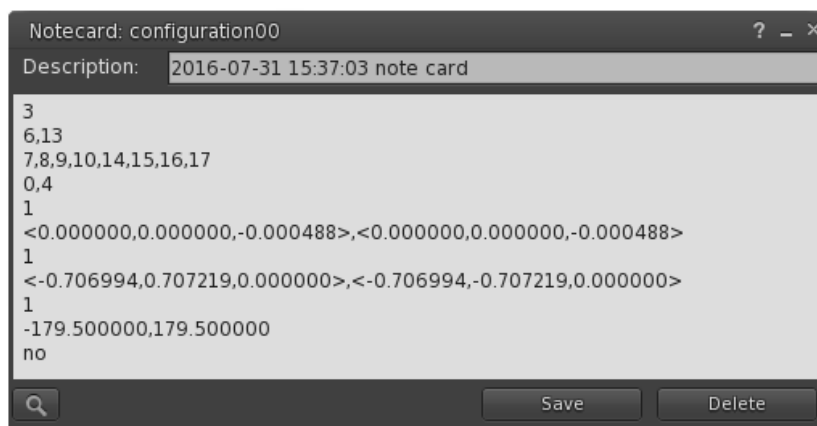
sim10218.agni.lindenlab.com:12046/cap/88c9fe08-f86c-335b-6d24-e2... ? _ x

http://sim10218.agni.lindenlab.com:12046/cap/88c9fe08-f86c-335

Copy the lines bellow in a notecard called "configuration" with number 00, or 01...

```
3
6,13
7,8,9,10,14,15,16,17
0,4
1
<0.000000,0.000000,-0.000488>,<0.000000,0.000000,-0.000488>
1
<-0.706994,0.707219,0.000000>,<-0.706994,-0.707219,0.000000>
1
-179.500000,179.500000
no
```

Create a **configuration00** notecard with these values :



Notecard: configuration00

Description: 2016-07-31 15:37:03 note card

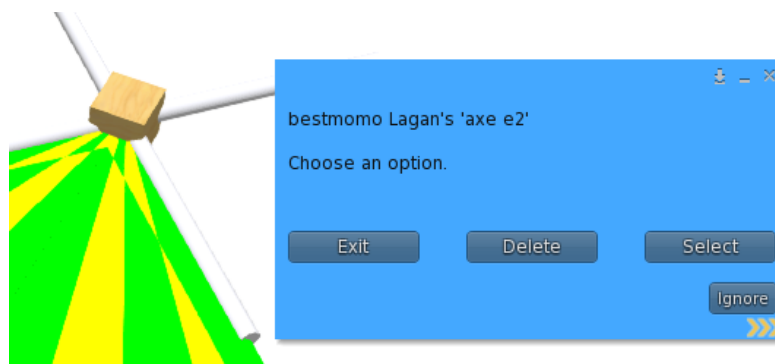
```
3
6,13
7,8,9,10,14,15,16,17
0,4
1
<0.000000,0.000000,-0.000488>,<0.000000,0.000000,-0.000488>
1
<-0.706994,0.707219,0.000000>,<-0.706994,-0.707219,0.000000>
1
-179.500000,179.500000
no
```

Save Delete

Then click on **Next setup** to go to the next setup...

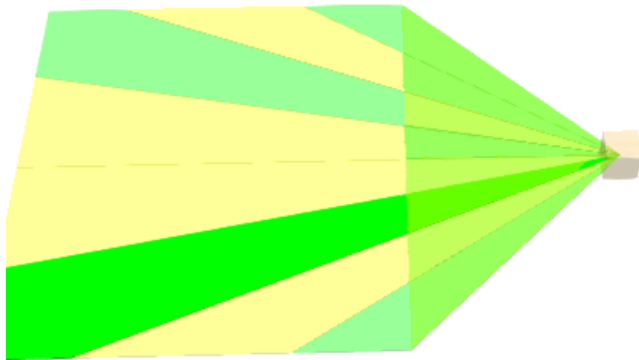
4.3 Setup 2

We will first remove the two axes. Select first one :

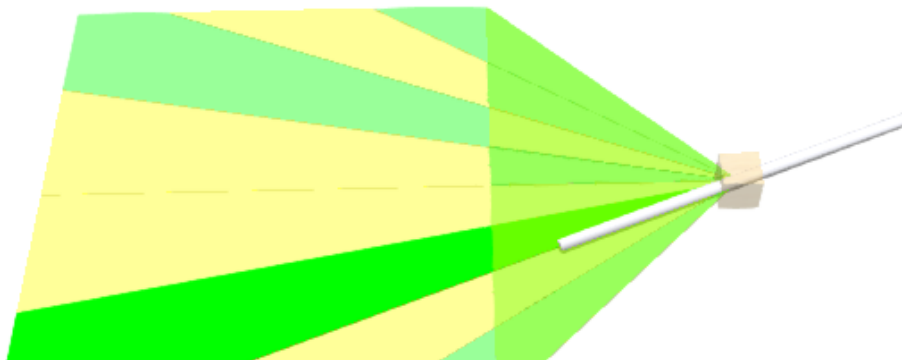


From the menu choose **Delete** to remove it. Do the same with the other axis.

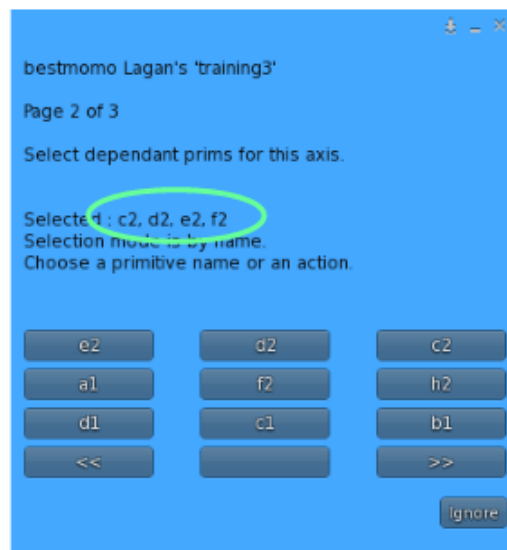
Click on Add axis and click on this primitive :



Click on **End Select**, **Side**, **Next**, **Next**, **Next** :

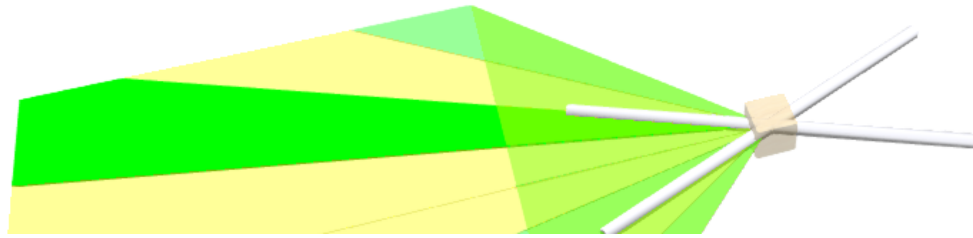


Click **Save** to save the axis. For dependant primitives switch mode "by name" and with the menu select primitives c2, d2, e2 and f2 :

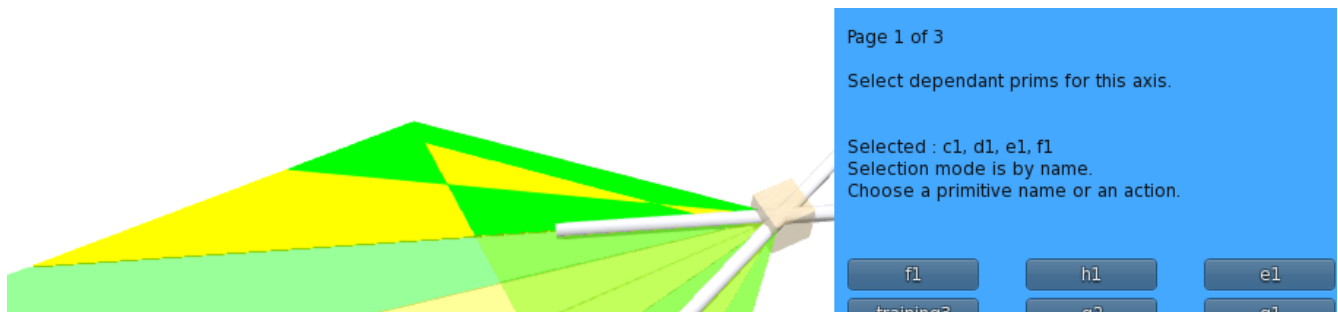


Click on **End Select** to memorize.

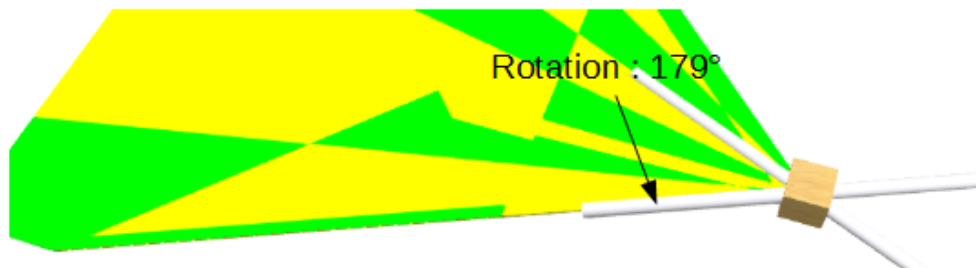
Follow exactly the same way with the symmetrical axis :



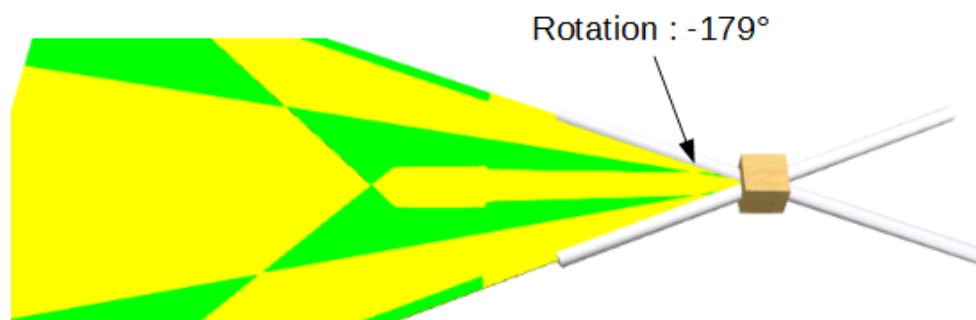
The basis primitive is g1. Dependent primitives are c1, d1, e1 and f1 :



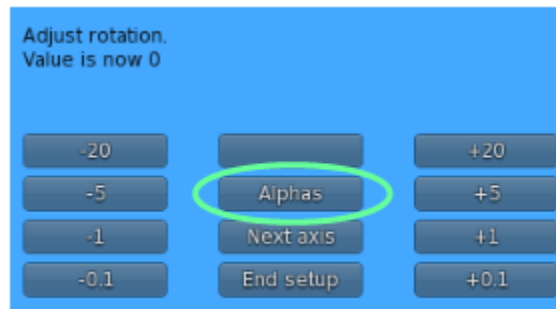
Click **End Select** and **End** to go to the rotations. For the first axis set a rotation of 179° :



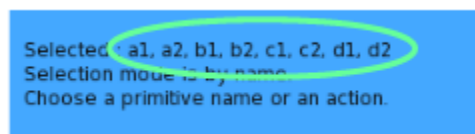
For the second a rotation of -179° :



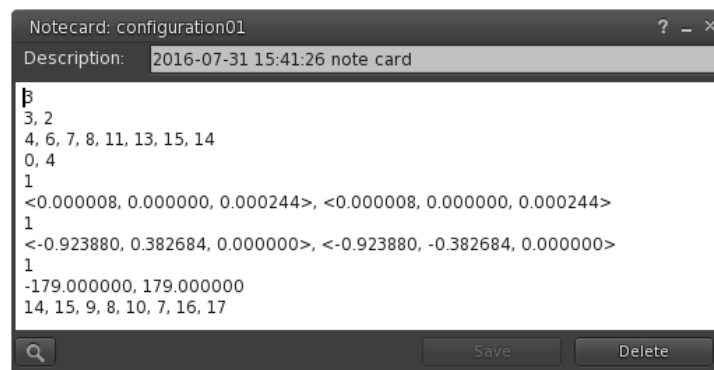
Then click on **Alphas** :



Switch to mode selection "by names" and select these 8 primitives :



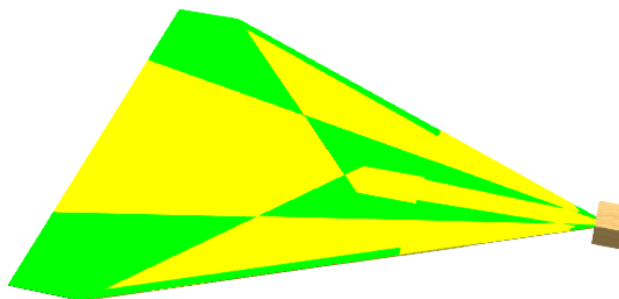
Click on **End Select** and **Go to page**. Create a **configuration01** notecard with the values :



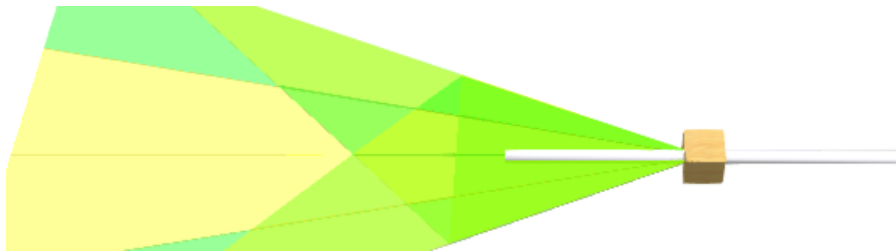
Then click on **Next setup** to go to the next setup...

4.4 Setup 3

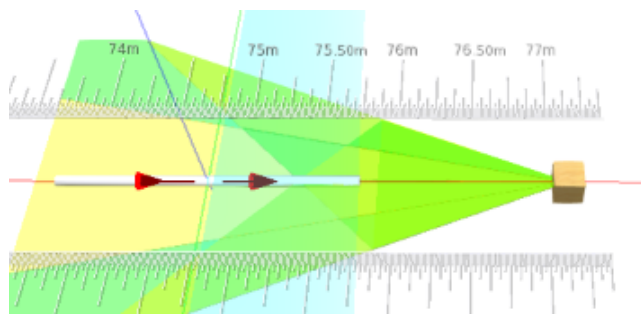
We will first remove the two axes. Click on axis and choose **Delete** :



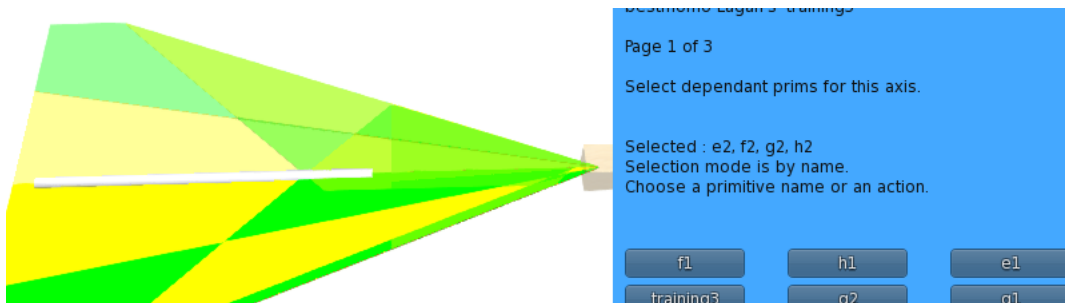
Click on **Add axis** and select the root primitive and End Select :



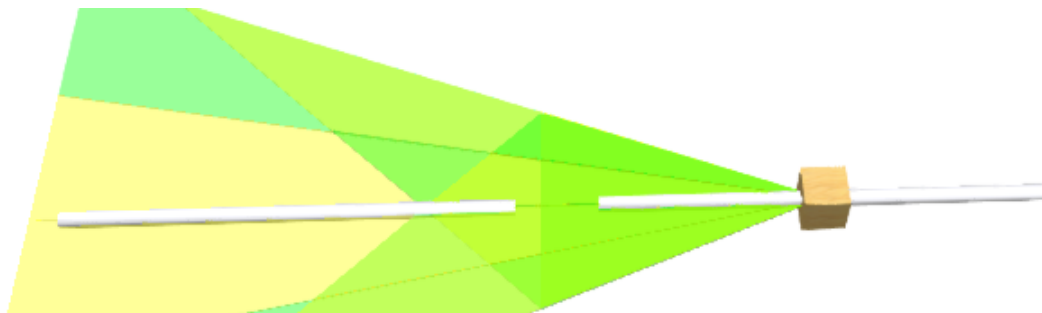
With the viewer editor slide axis :



Then click on **Save**. As dependent primitives choose e2, f2, g2 and h2 :

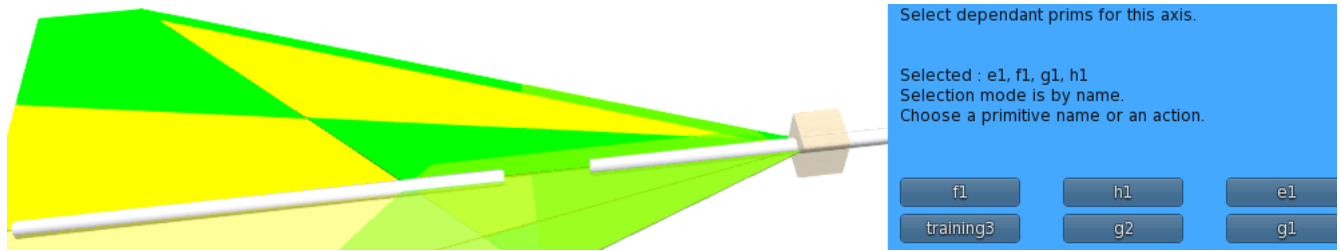


Click on **End Select**, **Add axis** and choose again the root primitive and **End Select** :

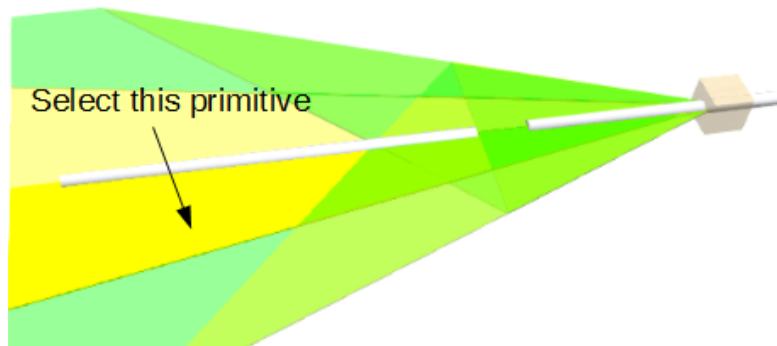


Having moved the first axis makes it possible not have overlay. We could also change the dimensions of the first as was done previously, but the movement is much faster.

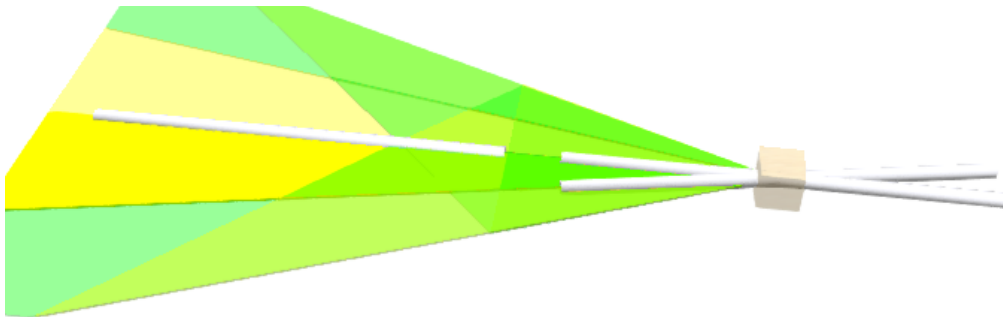
Click on **Save**. As dependent primitives choose e1, f1, g1 and h1 :



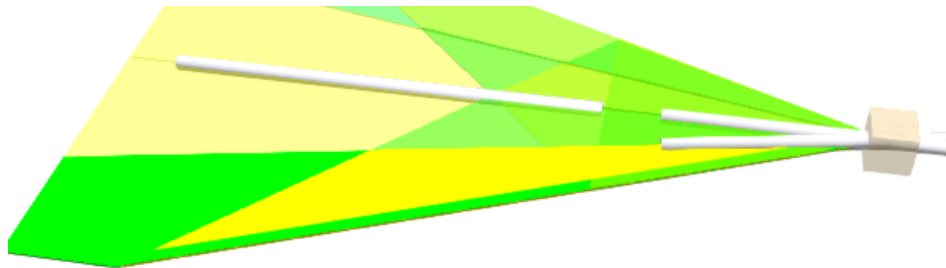
Click on **Add axis** and select this primitive :



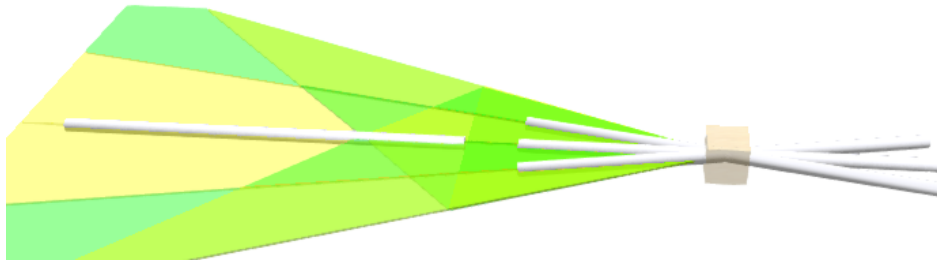
Click **Side**, **Next** and **Next** :



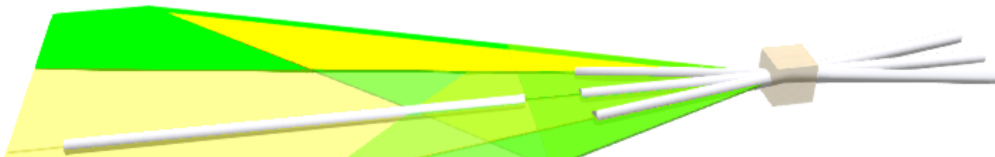
Click on **Save** and for dependent primitives choose f2 and g2 :



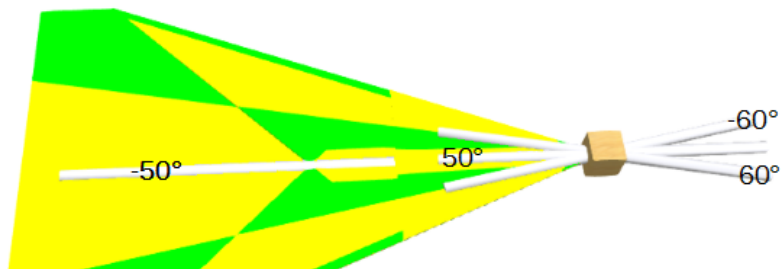
Click on **End Select** and **Add axis** then add the symmetrical axis :



As dependencies choose f1 and g1 :



Click on **End Select** and **End** to go to the rotations. Set these rotations :



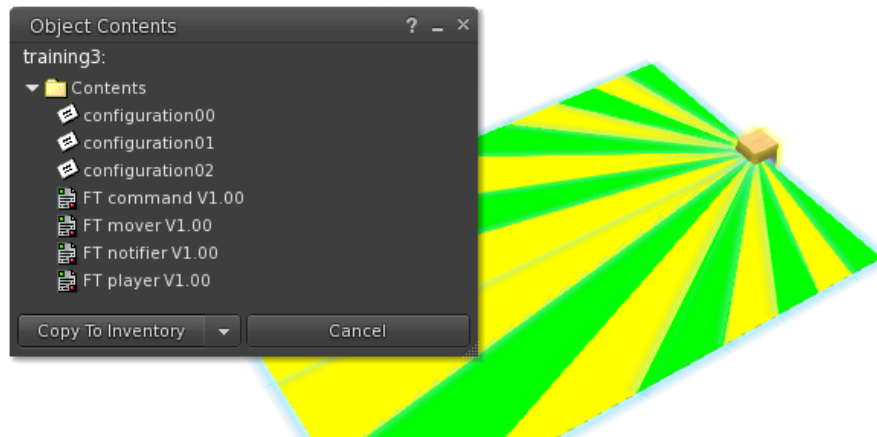
Which ultimately should look like this :



Click on **End setup**, **Go to page**, and create the **configuration03** notecard.

4.5 Action

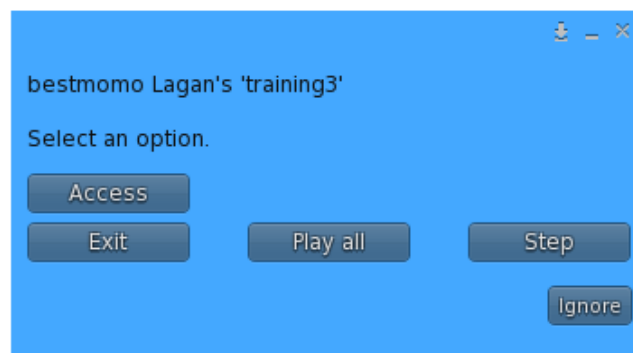
Take an object from starting state and add notes and scripts :



In **command** script change **LOOP** value :

```
integer MOVE_LOOP = FALSE;    // Must be TRUE or FALSE
```

On click you get the menu :



With **Play** all the three movements follow one another, and with **Step** you order them individually. As it has not setup to return once the third movement is completed we can not do anything...

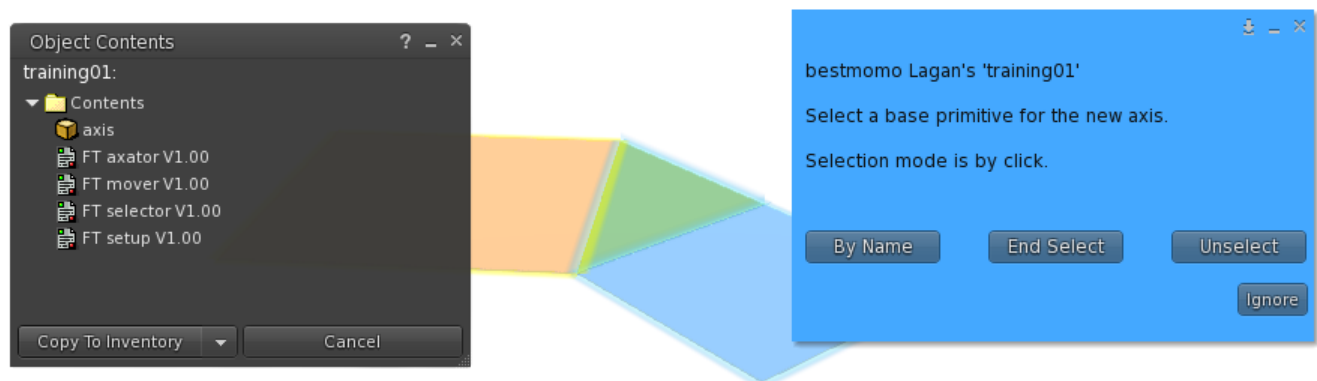
5. FAQ

After practicing with the help of the previous chapters you would normally be now operational. But it certainly still a number of issues and questions. This chapter is intended to respond.

5.1 If we are wrong ?

You can be wrong by clicking the wrong primitive or the wrong axis, or the wrong button in a menu. In some cases it is not serious and can be corrected, in other cases there is no appeal. Let's do some point...

You have created your object and you put axis and scripts, you have the menu :

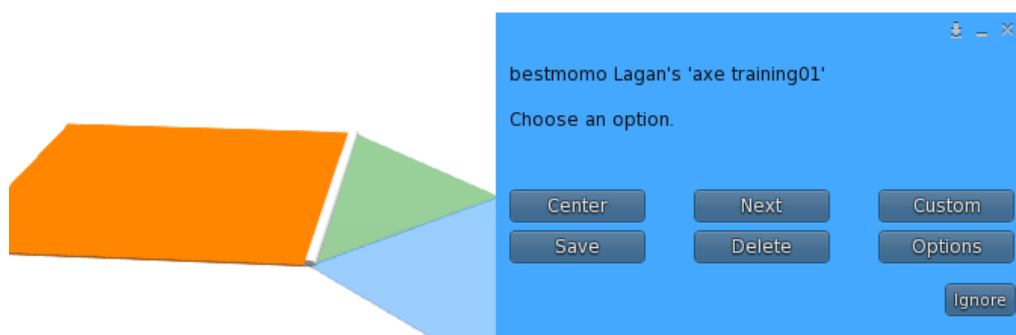


If you look at the public chat you have a message :

```
[06:53] training01: FT setup V1.00 ready.  
[06:53] training01: If you lose selection menu type /11select in public chat.
```

If you accidentally lose the menu, for example by clicking **Ignore**, you can retrieve it by entering **/11select** in public chat.

You go ahead and create your first axis :

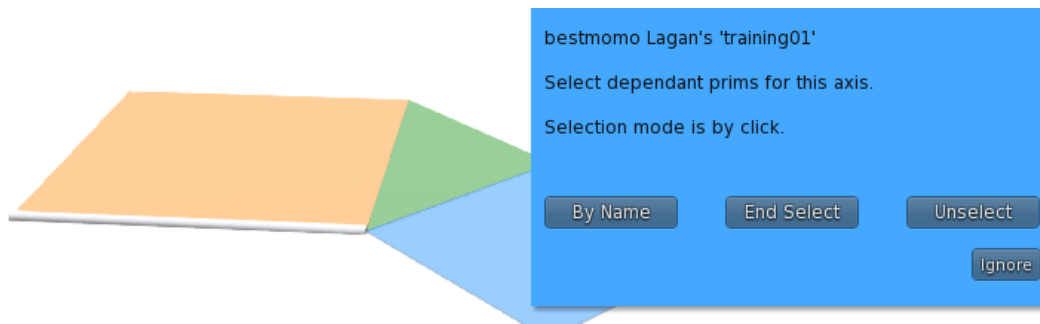


Look again in public chat :

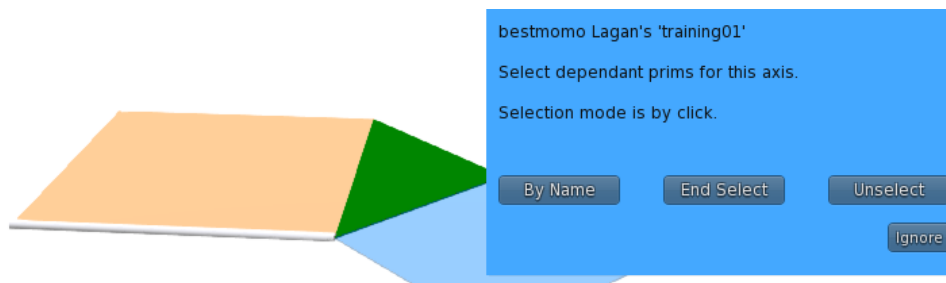
```
[06:56] axe training01: It's a standard box  
[06:56] axe training01: If you lose axis menu simply clic on axis to get menu back.
```

So if you lose the menu you get it again by clicking on the axis.

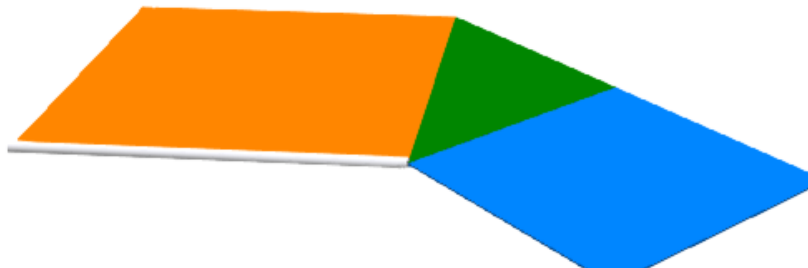
You go on and save your axis :



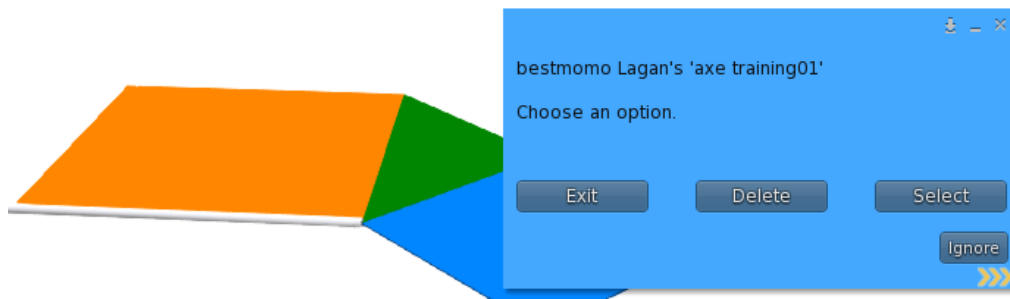
But you realize that it is poorly positioned and you no longer have access to the axis menu since you are in choosing its dependencies. In this case go to the end by selecting a dependent primitive, any :



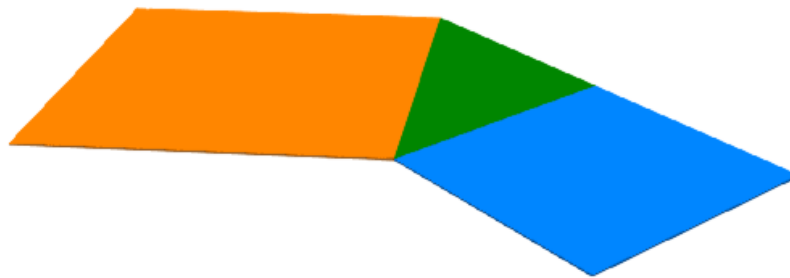
Click on **End Select** to fix axis :



Click on axis to get the menu :

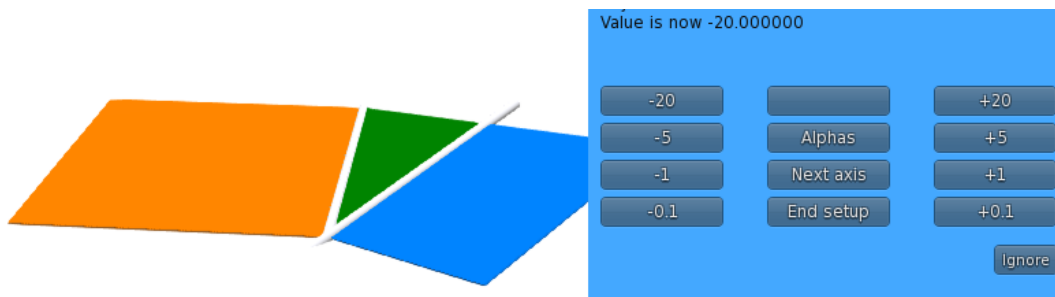


Choose **Delete** to delete axis :



You get into the starting position. Create axis and this time do it right otherwise it will again remove !

When you're setting rotations :



Be careful not to click **End setup** instead of **Next axis**. The setup is complete without possibility of return !

5.2 How to make a round trip with 3 states ?

The system allows for the setup return the previous setup. If we make two successive setups can be the return of the second but not the first since lost references.

Suppose you have 3 states : A, B and C. Here is how to proceed :

1. make the setup $A \rightarrow B$ and create the configuration00 note,
2. do the setup back $B \rightarrow A$ and create the configuration03 note,
3. create production with configuration00 note and play the motion to state B,
4. from this state make the setup $B \rightarrow C$ and create the configuration01 note,

5. make the setup back C → B and create the configuration02 note,
6. create the production with the 4 notes.

The only downside is we have to put again the axis in step 4.

5.3 How to modify the command script ?

The **command** script is editable and you can adapt it to your liking. However you must have sufficient knowledge of LSL to do apart from the information parameters :

```
// -----
//   Parameters you can change
// -----

string  ACCESS_MENU   = "All"; // Must be "All", "Group", "Owner"
string  ACCESS_OWNER  = "Sorry but access is reserved to owner !";
string  ACCESS_GROUP  = "Sorry but access is reserved to group. You must activate it !";
integer MOVE_LOOP     = TRUE;  // Must be TRUE or FALSE
```

I recall that the **MOVE_LOOP** parameter should be **TRUE** if you have planned a return trip and **FALSE** otherwise.

For the rest of the code you can make changes at your own risk!

What you should know is that the **command** script must send the correct information to the script **player** in charge of movement. Here are the commands sent to **player** :

Name	Code	Comments
P_PLAY	1401	Launches the movement by reading all notes
P_STEP	1402	Launches the movement by reading one notes
P_LOOP	1403	Launches the infinite movement
P_LOOP_STOP	1404	Stops the infinite movement
P_RESET	1405	Resets all primitives in their initial state

And responses from **player** :

Name	Code	Comments
P_END	1406	End of all movements
P_END_STEP	1407	End of one movement (STEP mode)

So you can build a custom command script.