Tutorial fracraft

June 2, 2013

by belouga

1 Introduction

fracraft is a javascript file (fracraft.js) to make creations in minecraft, that works with WorldEdit. You can copy something, move it, scale it, rotate it in any direction. You can also make some fractal structures using iterations. To make your own creations you edit the javacsript file, once you understand the instructions you will need to write around 5 to 20 new lines to make something worth it.

2 Needed

To use the script you need the followings:

- minecraft
- World Edit: http://wiki.sk89q.com/wiki/WorldEdit . It is included for example in the mod Single Player Commands.
- The Rhino javascript engine of World Edit: http://wiki.sk89q.com/wiki/WorldEdit/Scripting
- Its best to modify the file worldedit.properties, with scripting-timeout=30000, such that you can use more memory/
- the script fracraft.js (put in the crafscript directory)
- Its best to also having Rei Minimap (to know the East North directions)

3 Try It

First of all, **BEWARE THAT CRASHES ARE POSSIBLES !** The script can use a lot of memory in some special cases. This could crash your Minecraft ! (for which I am not responsible). Having crashed minecraft a few times, I just had to restart it and everything was still there, but this may not be your case. If you follow the instructions everything should be fine.

Here are a few examples of what you can do using the script fracraft.js. All thoses are "patterns" written in the script fracraft.js. Sometimes they have parameters you can modify, and you can use them to write your own patterns. For most of those constructions you copy an ensemble of blocks that you have selected, so you will always obtain different results (easily more aesthetic than in some of the snapshots !).

First do the following:

- open minecraft, a new world (better superflat for now).
- Build something.
- Select it with WorldEdit: for example have a sword in hand, type command /farwand, then left click the lower left corner and right click the upper right corner of the cube.
- If your selection is 3 blocks to the east, 4 blocks up and 5 blocks to the north then it is of size 3x4x5 . You can check this with //size. You can also use //expand and //contract to adjust the size.

Try the following in minecraft (you can undo with //undo at any time):

- Make your house twice as big. Select your house. Type in the minecraft terminal: */cs fracraft biggerhouse*

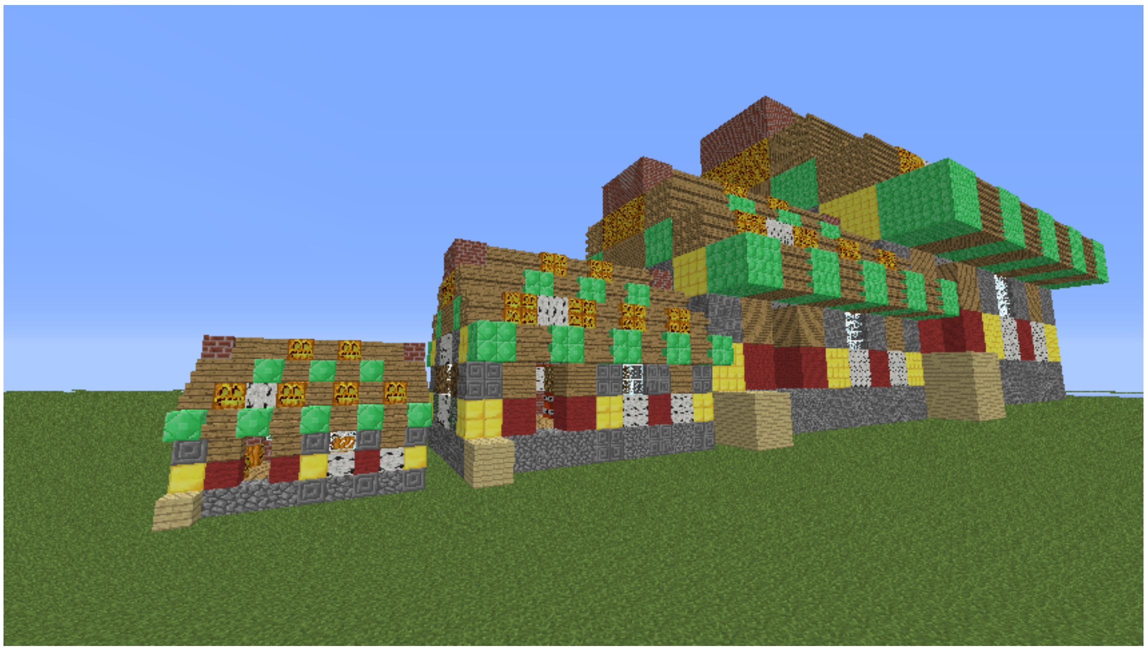

Figure 1: biggerhouse

- Make a Big flower. Make a 2x2x2 selection: */cs fracraft bigflower 30* . If you repeat the command result will be different each time (there is some random).

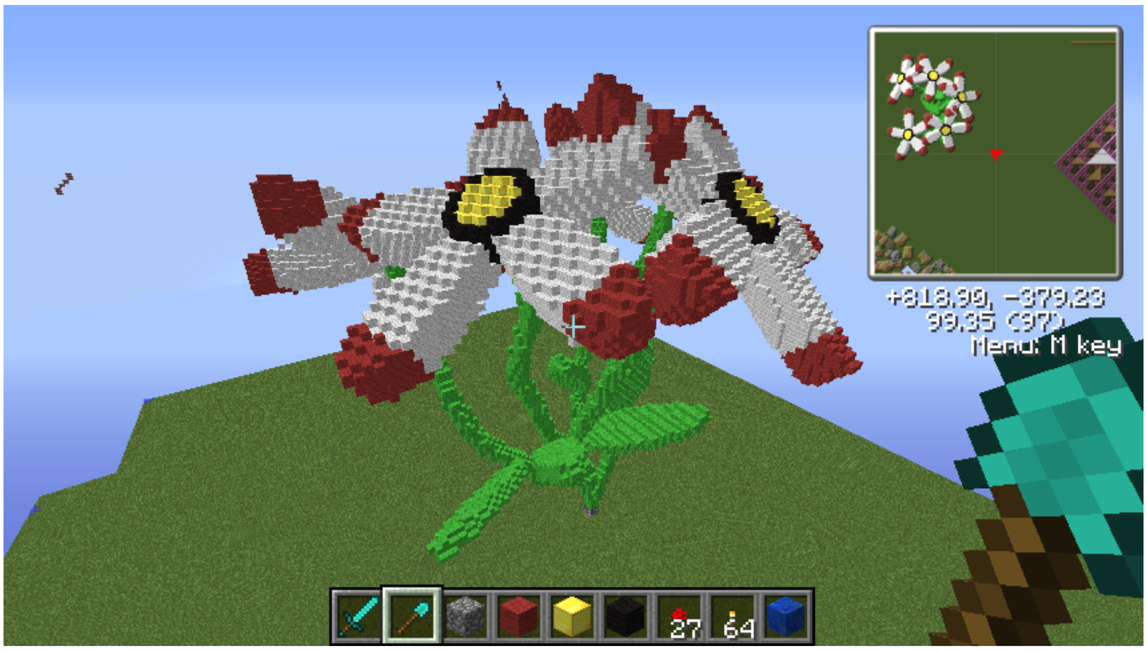

Figure 2: bigflower

- Rotate an entire village. Select an entire village area. /cs fracraft inception -1, then /cs fracraft inception -2, then /cs fracraft inception -3, etc... Be patient and slow as this can use a lot of memory.

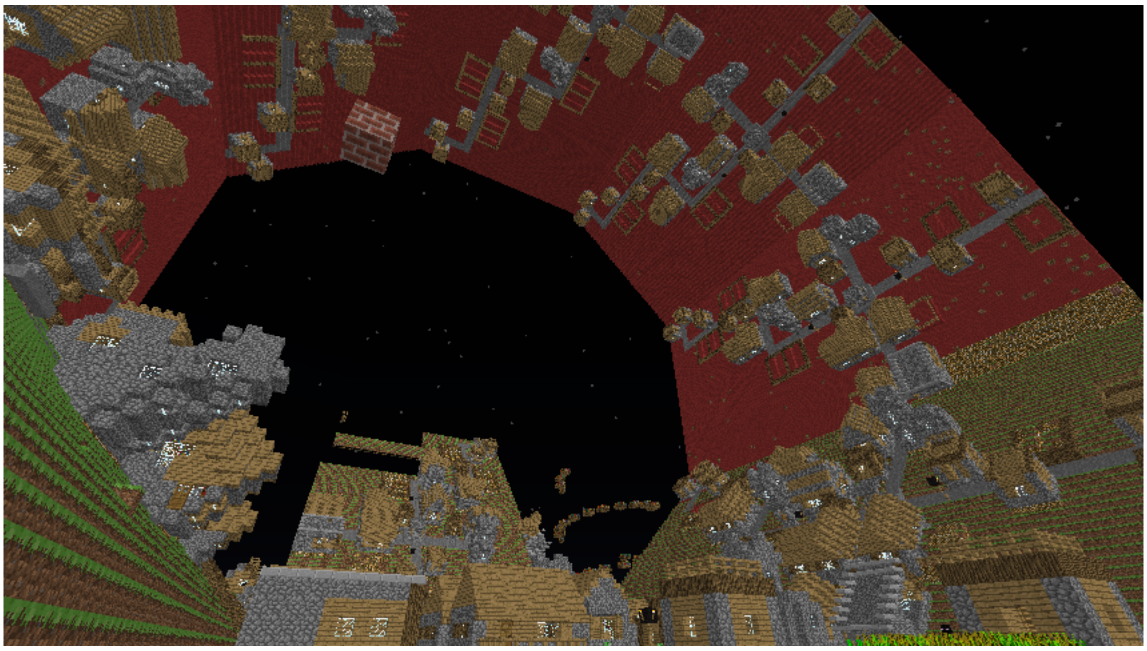

Figure 3: inception

- Make the spongecube (a kind of fractal). Make a 27x27x27 selection and fill it with //set diamond. Type: /cs fracraft spongecube 0. Then type: /cs fracraft spongecube -1, then /cs fracraft spongecube -2, then /cs fracraft spongecube -3.

- Make the snowflake cube: Same as the spongecube but type /cs fracraft snowflakecube 0. etc...

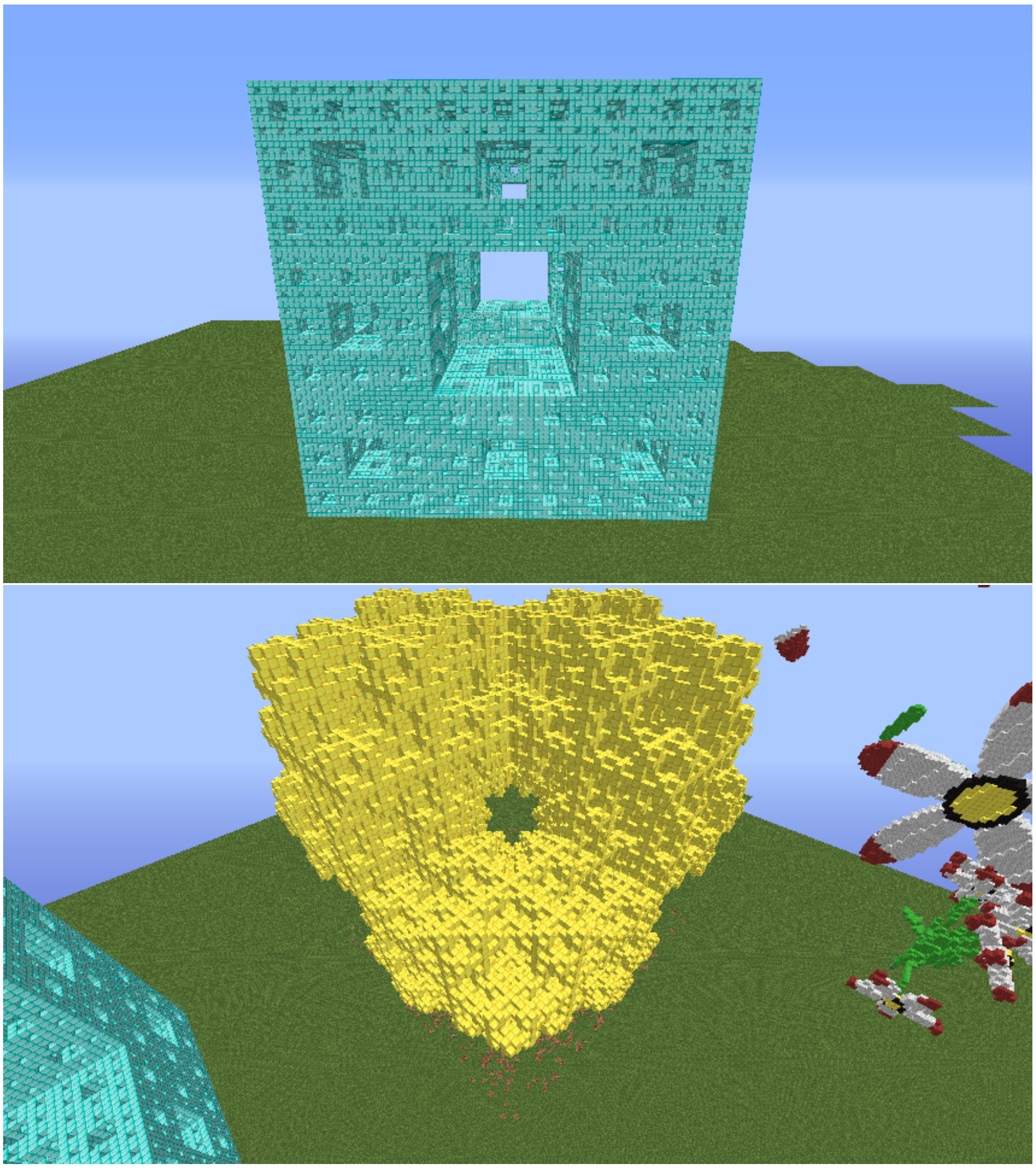

Figure 4: spongecube and snowflakecube

- Make the fcube (another kind of fractal). Make a 32x32x32 selection. To copy what is inside your selection, use : /cs fracraft fcube -1 , /cs fracraft fcube -2, /cs fracraft fcube -3, etc.. (here up to -5). To make it different, you can rather specify the color at each iteration. Type: /cs fracraft fcube 0 stone, then /cs fracraft fcube -1 obsidian, /cs fracraft fcube -2 sandstone, /cs fracraft fcube -3 wool:red, etc... up to -5. Then, you can empty the fcube: type /cs fracraft fcubeinside 0, cs fracraft fcubeinside -1, etc... up to -3. You can then enter the fcube.

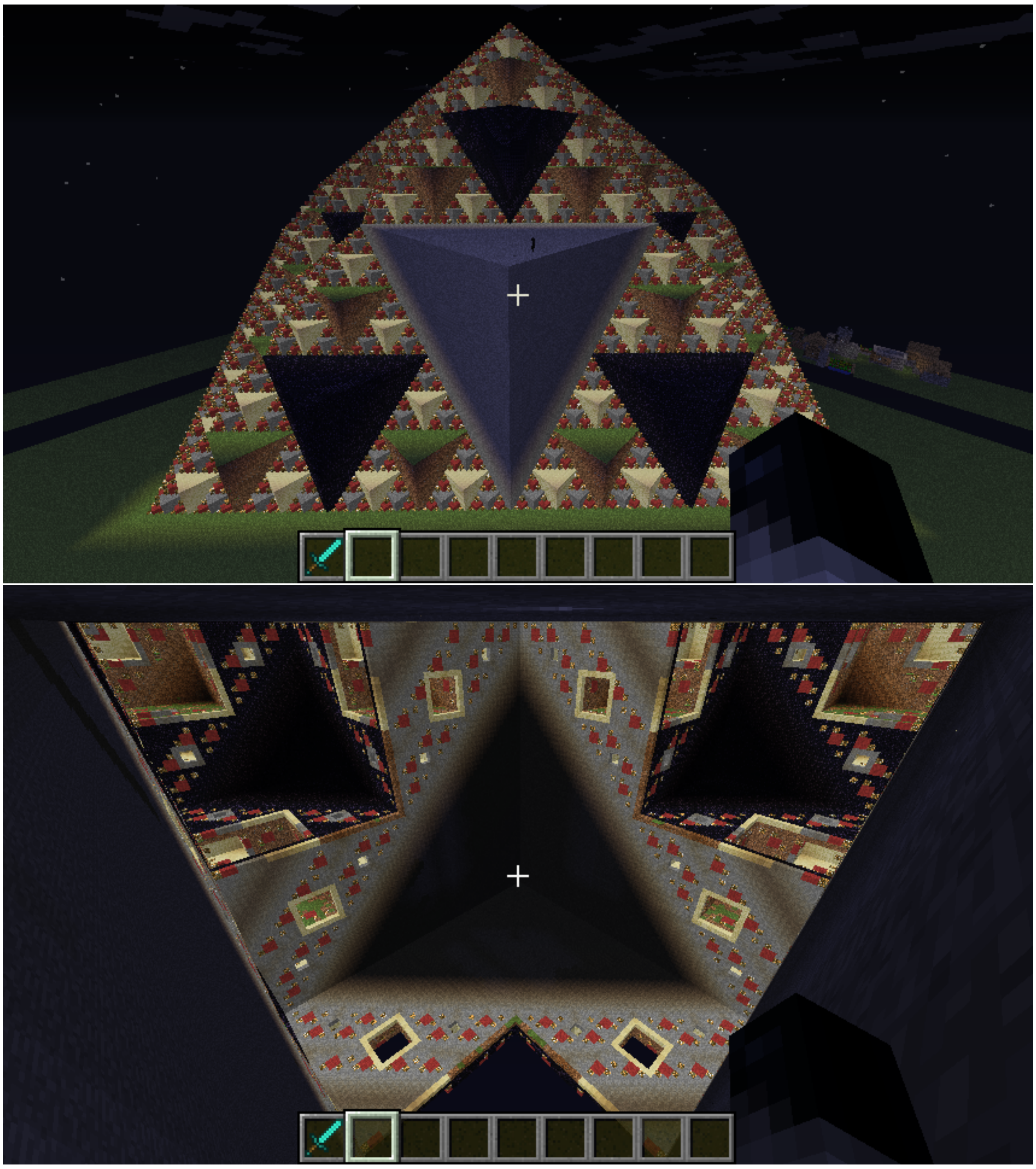

Figure 5: fcube and fcubeinside

- Make the pythagore tree in 2D (another kind of fractal). Make a 32x32x1 selection (i.e. a square in the east-west direction). It works like fcube: /cs fracraft pytree2d 0, /cs fracraft pytree2d -1, etc...

- Make a modified version of the pythagore tree in 3D. Make a 32x32x32 selection. It works like fcube: /cs fracraft pytree3d 0, etc...

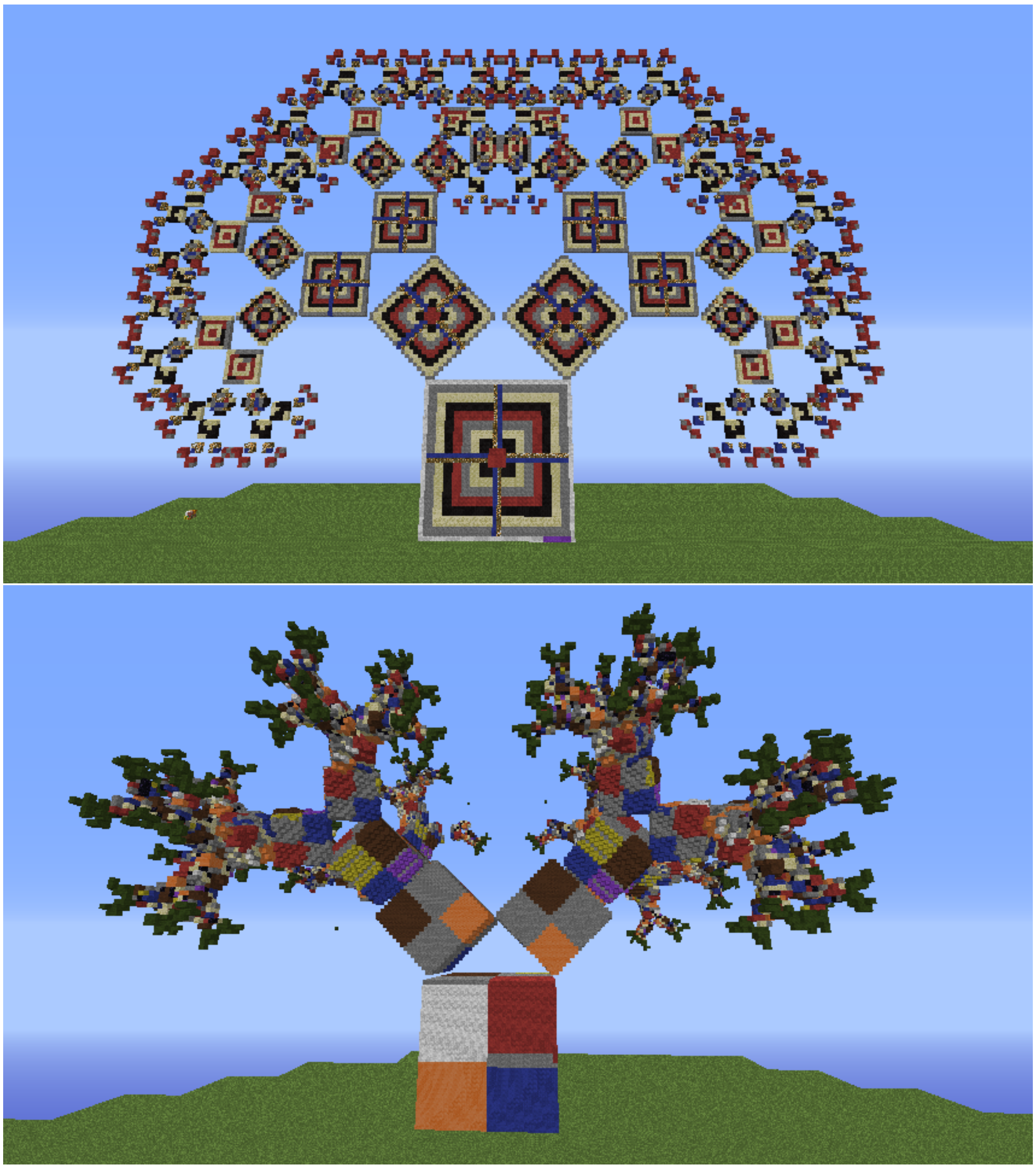
Figure 6: pytree2d and pytree3d

- Make some torus fractals: make a 32x32x32 selection, then works like fcube: /cs fracraft manytorus 0 stone, /cs fracraft manytorus -1 obsidian, etc...

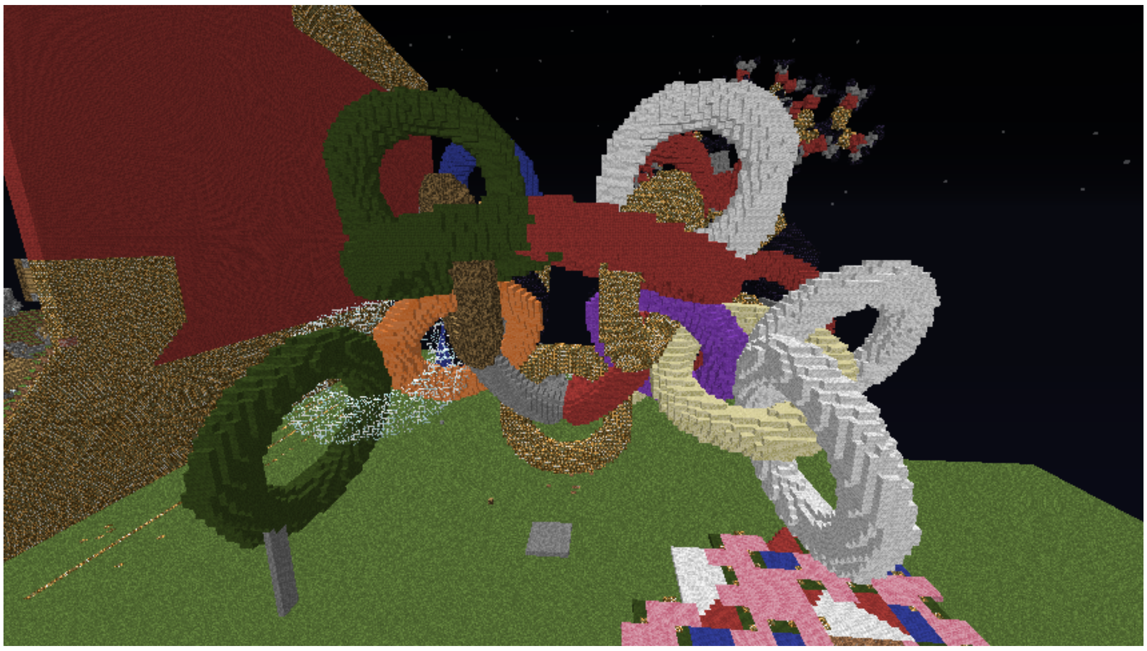

Figure 7: manytorus

- Selecting your house: /cs risingspiral 40 (one or more rising spirals that copy the selection).

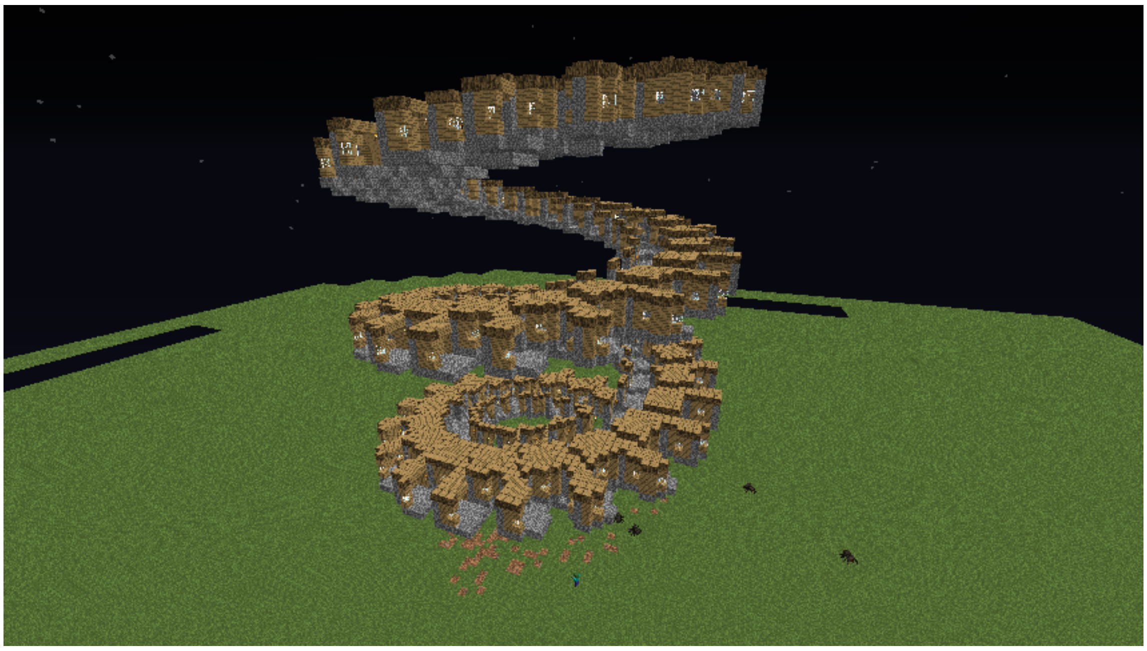

Figure 8: risingspiral

- Selecting your house: /cs fracraft randomcube 40 (it rises up randomly).

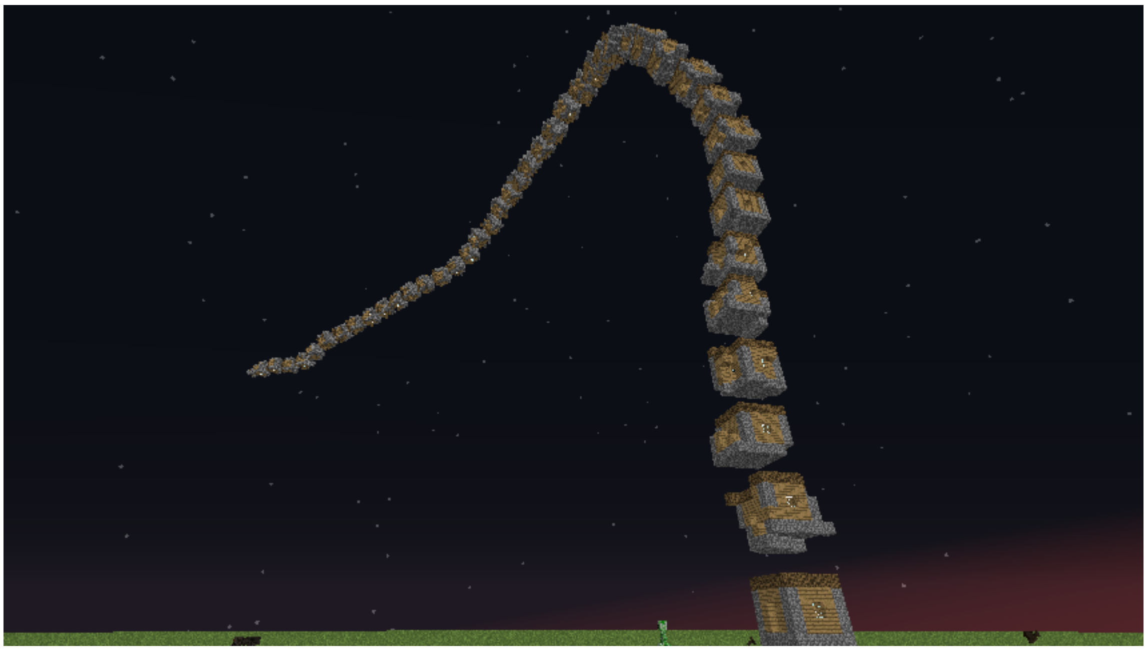

Figure 9: randomecube

They are some others you can try:

- Selecting your house, /cs fracraft revolution 20 (makes an horizontal circle copying your house).
- Selecting your house: /cs fracraft saturnrings 40.
- From a 1x1x1 or 2x1x1 selection: /cs fracraft downstairs 70 (stairs straight to the bedrock).
- From a 1x1x1 selection: /cs fracraft rainbow 120.
- From a 64x1x64 selection (on the ground): /cs fracraft dragoncurve -11 wool:red (another fractal).

4 Make a Build

Here we do a short tutorial on how to edit the script to make your own creations. You also have more instructions at the beginning and at the end of the fracraft.js.

Get started

- First open minecraft, a new world (possibly flat).
- Build a cube that is 4x4x4 blocks (just for this tutorial).
- Select it with WorldEdit: for example have a sword in hand, type command /farwand, then left click the lower left corner and right click the upper right corner of the cube.
- For this tutorial make sure there is something in your selection: for example type //set stone.
- Open the file fracraft.js that is in the craftscript directory, with any text editor. You can have minecraft running and edit the file at the same time.
- Add the following lines at the end of the script (before return mat;}):

```
if (pattern=="mypattern") {
addref();
t=[10,0,0]; s=[1,1,1]; r=[0,0,0];
addbuild();
}
```

This is your pattern. Now test it in minecraft . In the console, type: */cs fracraft mypattern*. This will create a copy of your initial selection 10 blocks to the east.

Transformations

The script uses some simple transformations in a 3D space (there is some maths involved). This is shown in the figure. The minefract space is with axis x-y-z that point to the east-up-south. The cube in red is your selection. Now you create a new cube (in black) using the transformations t,s, and r. t gives the translations along x,y, and z (in that order). s gives the scaling factors along x-y-z. r gives the rotations along x-y-z (in degrees). addref() initializes t=[0,0,0], s=[1,1,1], r=[0,0,0] (such that the new black cube would match the red cube). addbuild() adds a build: with this the new cube that you have created will be build in minecraft world. It will be a copy of the initial selection (by default).

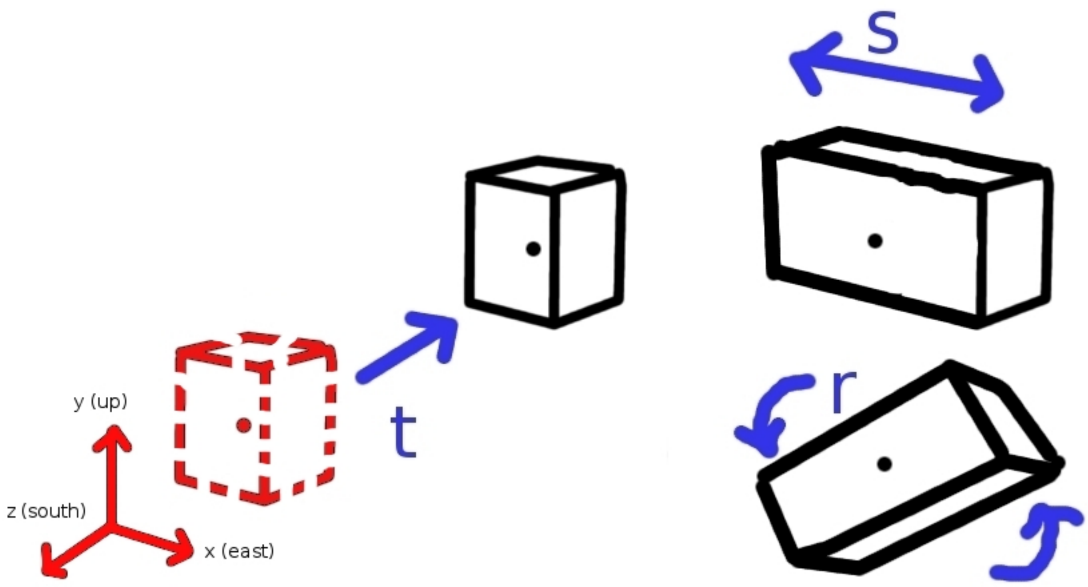

Figure 10: t,s,r to make a build

Tests

You can test changing t,s,r in the script:

- t=[0,10,0]; will move the new cube 10 blocks up.
- t=[0,10,-10]; will move the new cube 10 up and 10 north.
- s=[1,2,1]; will largen the new cube by a factor 2 in the up/down
- s=[0.5,2,1]; will largen the new cube by a factor 2 in the up/down and reduce it by a factor 2 in the east/west.
- r=[0,45,0]; will rotate the cube counterclockwise 45 degrees around the y axis.
- t=[0.1, 11.5, -2.7]; s=[0.5, 2.2,3.1]; r=[45,25,30]; this will make many transformations. You can use any values for t,s,r (except s=0 or s<0), but it will not always build nicely in minecraft (and sometimes it will not build at all !). This is because the new cube has to be interpolated to the minecraft grid, and this is not always straightforward.
- You can do various builds. Just repeat the addref, choose some t,s,r and then addbuild in you pattern.

5 Make more complex Builds

Use variables

Inside your pattern you can put some javascript code (new variables, for loops, if, Math...). Some variables are already provided that you can use to make your build. The most usefuls are sx,sy,sz: the sizes of your current selection, px,py,pz the player position with respect to the initial selection, and n the current iteration (see section on fractals). For example, if you use:

- t=[sx,2*sy,3*sz]; this will move the new cube to one time the selection size to the east, twice up, and three times south

- s=[1/sx,2/sy,1]; this will make the new cube to be of size one block to the east, 2 blocks up, and will not affect the north size.

-t=[2*px,2*py,2*pz]; this will make you (the player) at the middle between the initial selection and the build.

You can provide additional arguments when calling fracraft: */cs fracraft pattern iterations arg1 arg2 arg3*. pattern is the name of the pattern you use, iterations is the number of iterations (see next section), and arg1, arg2, arg3 are any argument you want to provide.

- Inside your pattern, add the following: *var a=getarg(1)*. Then type: */cs fracraft mypattern 0 sandstone*. it will read arg1 that is "sandstone".

Change the blocktype

- *addfill(blockid, blockdata)*. This is one of the most usefuls. It will affect the last recorded build, so addfill() must be put just after the corresponding addbuild().Instead of copying the initial selection, this will fill the new cube with the blocktype. A blocktype is defined by its ID and data (http://www.minecraftwiki.net/wiki/Data_values#Saplin
For example addfill(1,0); will fill the build with stone.

- *addfillarg(1)*;. This is a useful shortcut. if you provide arg1 with a blocktype string (sandstone, wool:red,etc...), it will directly use addfill() to set the last recorded build to be filled with that blocktype. If you type: */cs fracraft mypattern 0 sandstone*, the build will be filled with sandstone.

- They are also some filters that read/write only some blocktypes when you copy (addexcin, addexcout...)

Use math expression

You can add restrictions to your build using a math expression (similar to //generate). After addbuild(), put for example *addmath("x*x+z*z+y*y<=1");*. This will make your build limited to a sphere instead of a cube. It works as follows: suppose your new cube is by default of relative coordinates $-1 \leq x \leq 1$, $-1 \leq y \leq 1$, and $-1 \leq z \leq -1$. With addmath, you will write blocks inside your new cube only where the math expression is true. This does not work exactly like //generate, because you have to use the javascript maths notation (Math.abs(), Math.pi(), Math.cos()..., &&, |). Use only x,y,z and numeric values in your expression.

6 Make Fractals

fracraft uses a system of iterations similar to some fractals. This can be used to make more complex creations. Like in previous tutorial, build and select a cube that is 4x4x4. Add the following lines at the end of the script:

```
if (pattern=="testchild") {
//make a build
addref(); addbuild();
// make a child
addref();
t=[10,10,0]; s=[1,1,1]; r=[0,0,0];
addchild();
}
```

For this simple example, the build is exactly in the region of the initial selection (if you type: */cs fracraft testchild*, you will just copy your selection at the same place). With addchild(); you have added a new cube that is a "child". You have used t,s,r again, but because it is followed by addchild() it serves to define a child (instead of a build). Now run the script with five iterations. Type: */cs fracraft testchild 5* (this make the iterations 0, 1, 2,3,4,5). You will create a serie of five new cubes.

Parents to Childs

The childs change the frame (or system of coordinates) used to make the builds. When you type: */cs fracraft testchild N*, you make N iterations. At each iteration, you change the frame from the one of the parent to the one the child(s). This is shown in the figure.

- At the iteration 0, the parent is the initial selection (red cube). If it has some builds they are constructed in the parent frame. You define the child1 using t,s.r. and addchild(), which is the black cube. This black cube is not constructed in minecraft, it only sets a new frame.

- Passing from iteration 0 to iteration1, the childs become parents.

- At the iteration 1, the parent is now child1 (black cube). If builds are constructed, it is in the frame of child1. The t,s,r used previously to define child1 in the frame of the initial selection are now used to define child2 (green cube) in the frame of child1.

- Passing from iteration 1 to iteration 2, child2 becomes the parent.etc..

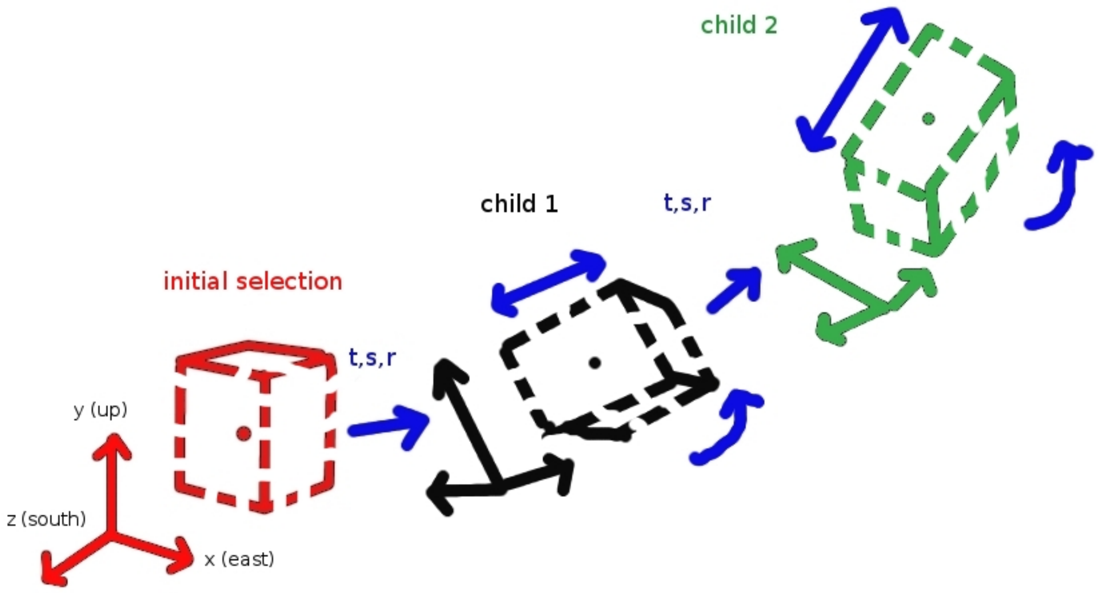

Figure 11: parent to child

Transformations

For each iteration, the transformations t,s,r work the same but they are relative to the frame current parent. In terms of maths, the origin position of the 3D space has been moved (with t), and the basis vectors x-y-z of have been rescaled (with s) and rotated (with r). Now consider those examples, reminding that at the iteration 0 the parent is the initial selection, and that at iteration 1 the parent is child1:

- Say that at iteration 1, child1 has moved one block east with respect to the initial selection. Now all the new builds and childs will be constructed one block further to the east.

- Say that at iteration 1, child1 is twice bigger than the initial selection. Now all the new builds and childs will be twice bigger. Also the t that define those will correspond in reality to translations of twice the blocks.

-Say that at iteration 1, child1 is rotated with respect to the initial selection. Now all the new builds and childs are rotated. For example the t=[1,0,0] is not towards the east, but towards the new “east axis” that is rotated.

There is an exemple in the figure. This is the same code as before but starting from a bigger selection (16x16x16) with t=[sx/2,1.5*sy,0]; s=[1,1,1]; r=[0,45,45], and around 20 iterations. You see that the same structure repeats in a frame that is changing. If you use some s>1 the structure becomes bigger as you iterate, and if you use s<1 it becomes smaller (until it is too small to be build).

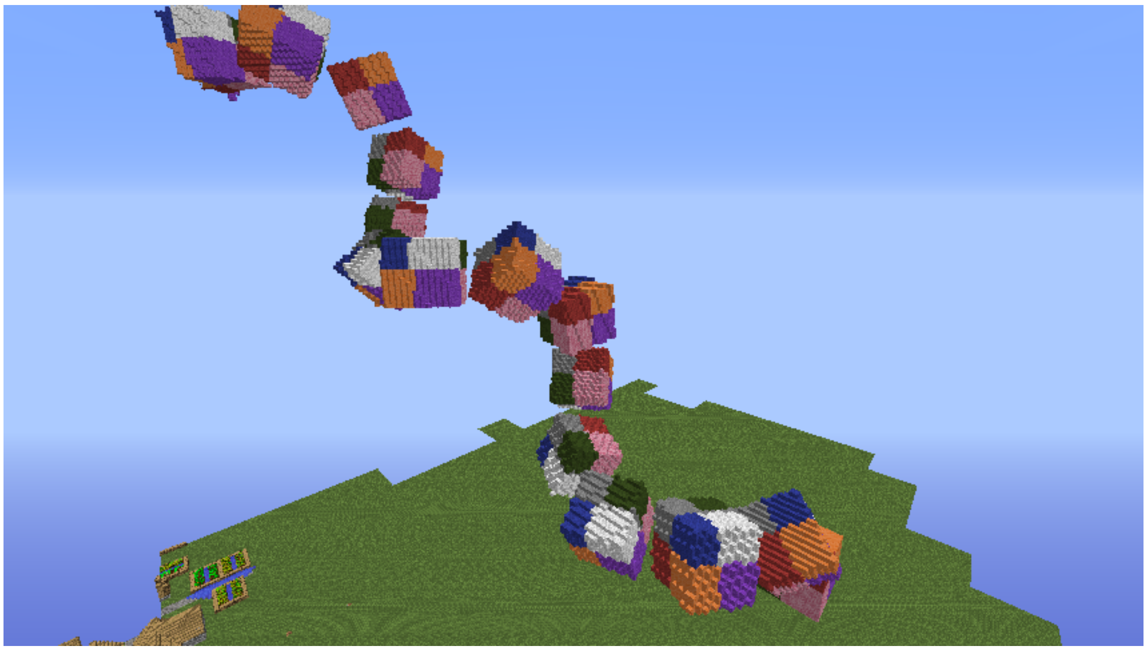

Figure 12: exemple of childs with translations and rotations

Various childs

You can make various childs. Modify the script for example as:

```
if (pattern=="testchild") {
//make a build
addref(); addbuild();
// make a first child
addref();
t=[10,10,0]; s=[1,1,1]; r=[0,0,0];
addchild();
// make a second child
addref();
t=[-10,10,0]; s=[1,1,1]; r=[0,0,0];
addchild();
}
```

Here you have defined two childs. At iteration 0 there is one parent making two childs. At the iteration 1 there will therefore be two parents, each making two new childs. So you will have 1 parent at iteration 0, 2 at iteration 1, 4 at iteration 2, 8 at iteration 3, etc...

7 Tips for large creations

For large creations you could crash the minecraft. Here are a few tips to avoid this:

- Better not select too large areas.

- Better not put “special” blocks in your initial selection (chests, doors, vines, ...). Those do not copy well anyway.

- Do not make two much childs or two much iterations, because the total number you will have to deal with will become (childs)^iterations. This can become huge very quickly.

- A very useful trick to avoid this is to build iteration by iteration. For example, type: */cs fracraft testchild -3*. This will only build elements at the iteration 3 (instead of building at all iterations 0,1,2,3), which limits memory use. For your large creations, you should use: */cs fracraft testchild 0*, */cs fracraft testchild -1*, */cs fracraft testchild -2*, */cs fracraft testchild -3*, etc... to create it step by step.

- For repetitive taks, you can bind a calling sequence to a key with WorldEdit. For example, type: */bind k /cs fracraft copyunder.* Pressing k will do the calling sequence. You can unbind with: */unbind k.* copyunder copies the selection at same height but at the x-z where the player is standing. Binding //undo is also very useful.

8 Last Notes

- You may have noticed that when using rotations, the new builds do not rotate each block accordingly. This may be fixed in a new version.

- Don't hesitate to send your creations, you just have to give your pattern to copy/paste in the script.