TreeCracking Documentation (currently only birch and oak)

(if you need help consider joining the minecraft@home discord server or dming me in discord(epic10l2))

This Documentation is going to teach you everything about the steps of TreeCracking:

Getting coordinates of a forest

Recreation

How to get and input data

How to run the TreeCracker(using google colab)

How to get structure and potentially world seeds from tree seeds(These are seeds that are output from the TreeCracker, they are not world seeds yet)

Too many structure seeds

I recommend you to test this on a test world first until you try to crack a seed from a video using this.

Getting coordinates of a forest

Probably the easiest way to get coordinates of a forest is to look for a moment where your “victim” opens F3, though this is not always the case, that's why we use Texture Rotations.

Texture Rotations

Basically some blocks have a texture that varies(rotates). This is entirely dependent on the coordinates so it is identical across all seeds.

Luckily there is a tool to automate the process of finding coordinates.

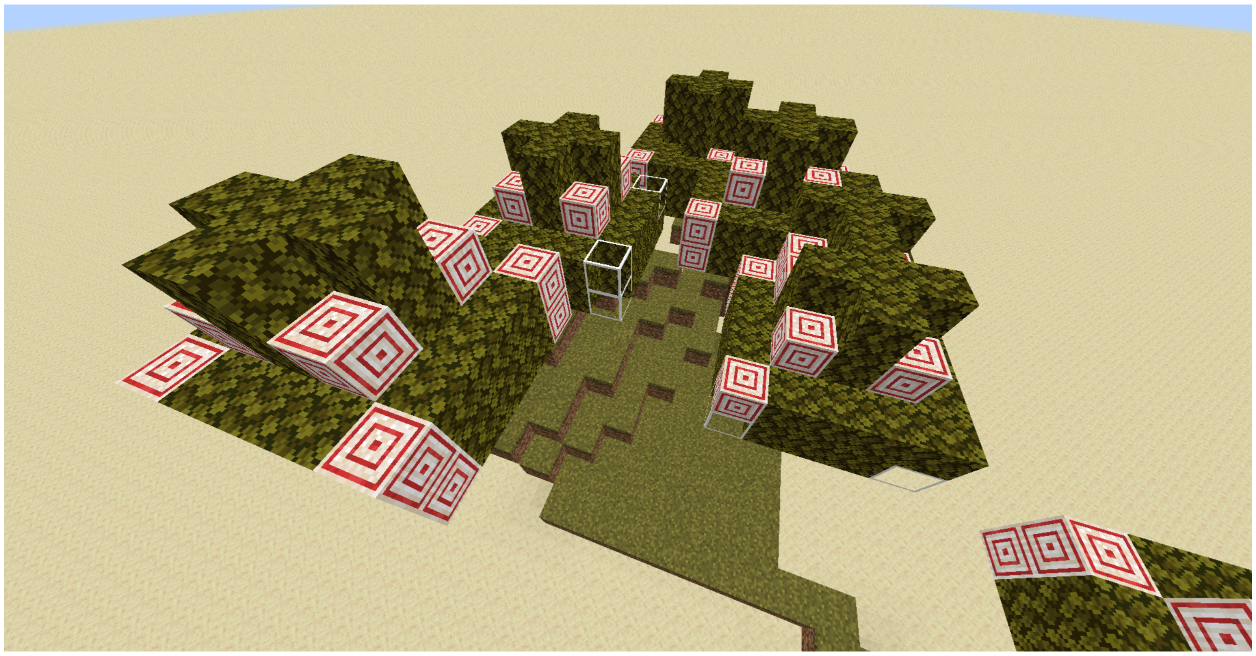
https://github.com/19MisterX98/TextureRotations

If you need any help with the tool I recommend asking in 19MisterX98s discord but you can always just join the minecraft@home discord server(if you’re not already there) and ping me(username is epic10l2).

If your “victim” uses optifine or a texture pack everything is a lot more complicated, so you’re gonna have to ask in the minecraft@home discord if any1 can provide you any help. Other more uncommon ways are to use clouds or/and grass offsets.

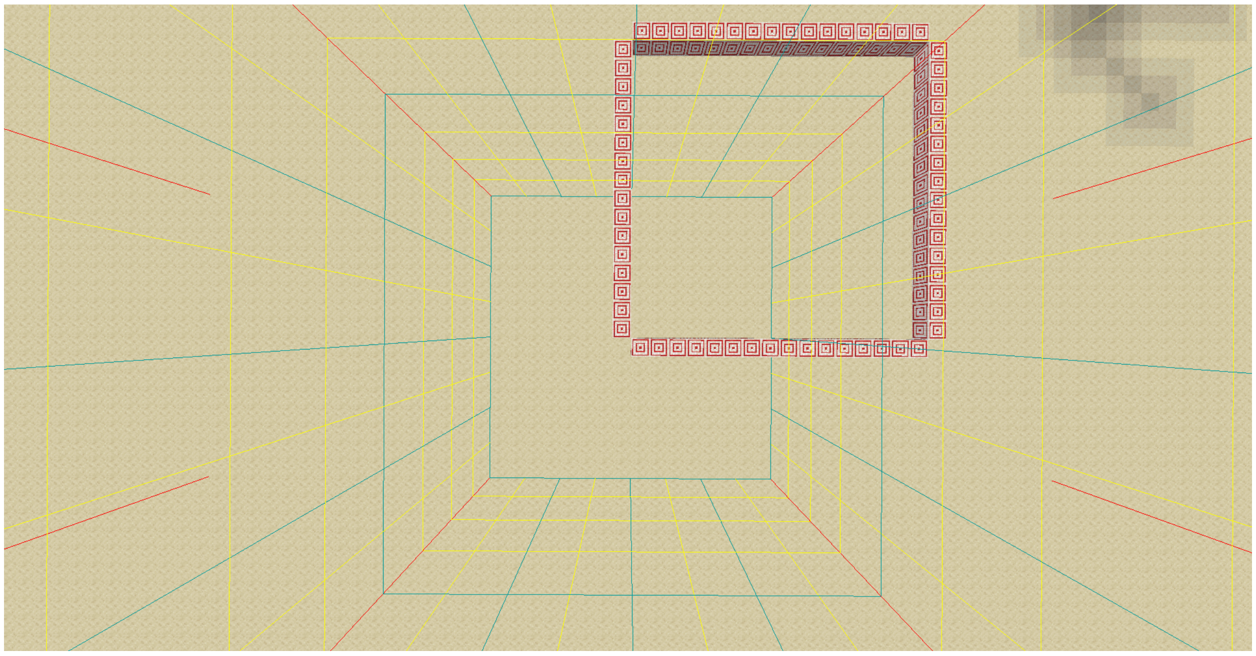
Recreation

To get the right coordinates i recommend recreating the area Block by block make sure to get the leaves of the Tree correct, more specifically the leaves at the corner. For you to understand the location of the important leaves i'm going to provide a picture:



Here you can see which leaves specifically you need to know of. I just placed leaves where there are leaves , target blocks where I am not sure and glass indicates the absence of leaves. You can use any other block to indicate the existence of leaves for example, you can just let them out or use green wool to indicate leaves. You can also use smth like a redstone block or red wool (or concrete) to indicate unknown leaves.

Also you should make sure the Trees are in one population chunk. For 1.13 + this is just a normal chunk but for 1.12- the population chunks are offset by 8:



The Target blocks are the outline of a population chunk(this is just an example, you can offset by 8 into the opposite direction and find a different population chunk).

How to get and input data

This is probably one of the most confusing parts and one of the main reasons I made this documentation.

First you'll have to make sure you have at least 48 bits of Information.
Every Tree position you know is about 8 bits.
Tree heights are about 1.6 bits and leaves are 1 bit of information.
If you don't have that information in a single population chunk you can first use less information and filter the structure seeds as described in the “Too many structure seeds” part.

So first you'll need to download the treecracker from here:
https://github.com/Gaider10/TreeCracker
And extract it.
After you downloaded the file you need to open the input_data.cuh and create a new input using this:

```
__device__ constexpr TreeChunk get_input_data() {
    return TreeChunkBuilder(Version::v1_14_4, Biome::Forest)
```

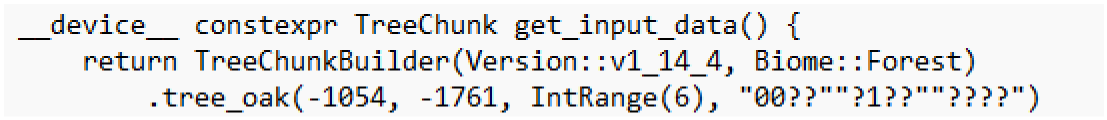
At the TreeChunkBuilder you first specify the Minecraft Version with v(MinecraftVersion) for example for 1.16.4
return TreeChunkBuilder(Version::v1_16_4, Biome::Forest)
Currently these are all the versions you can input:

V1_6_4,
v1_8_9
v1_12_2
v1_14_4
v1_16_1
v1_16_4
If your version is not in there don't worry, I recommend you use the next highest version.
(Note: 1.15 was never tested)
Then you specify the biome after the version in the same line
Currently supported biomes are

Forest
BirchForest
Taiga

We're gonna stick to the forest for the sake of simplicity when learning.

After that we input our first tree data like this inside the function:

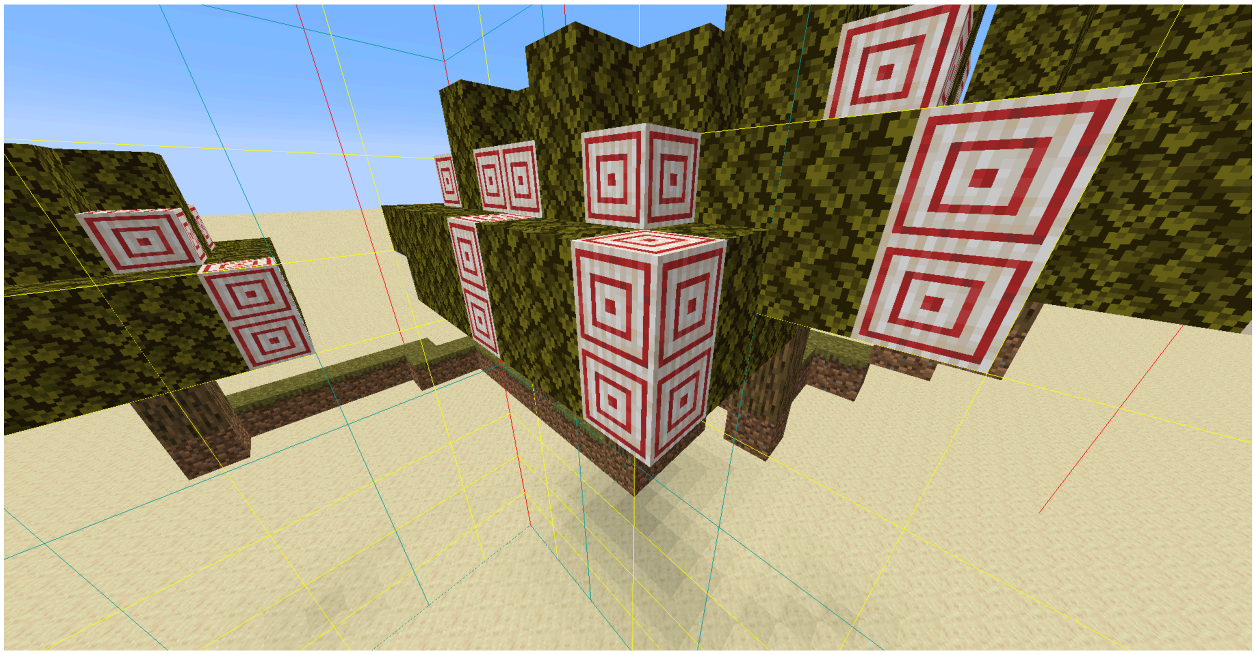


```
__device__ constexpr TreeChunk get_input_data() {
    return TreeChunkBuilder(Version::v1_14_4, Biome::Forest)
        .tree_oak(-1054, -1761, IntRange(6), "00??""?1??""????")
```

(I recommend sticking to that formatting)

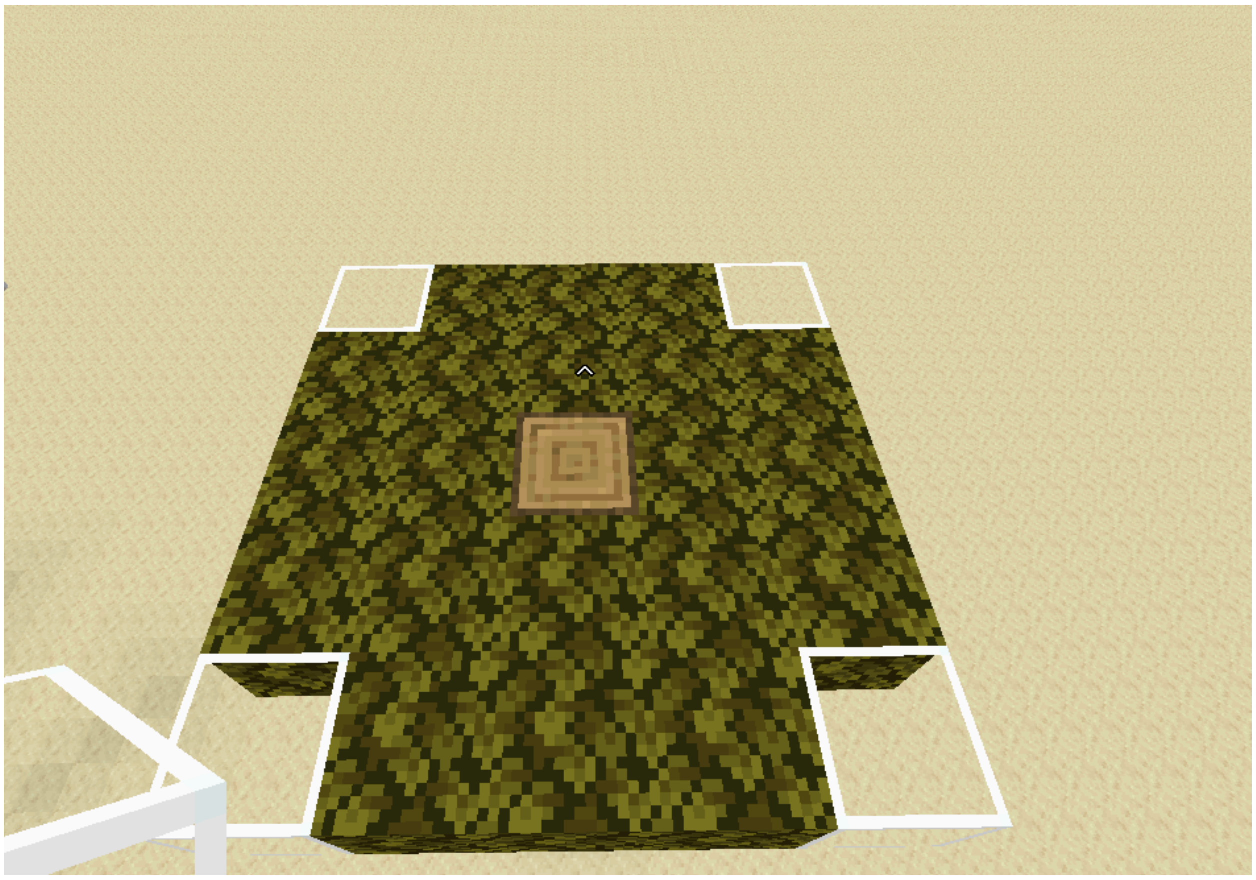
First we add a new oak tree using .tree_oak
After that we have the parameters first we input the coordinates like this(input x and z, not y)
.tree_oak(x,z)
Replace x and z with the actual x and z coordinates
WARNING: You need to use block coordinates , these are under XYZ in F3
After every parameter we set a ",".
Then we add the height of the Tree
Here like this(oak goes from 4 to 6 and birch from 5 to 7):
.tree_oak(x,z,IntRange(height)
Replace height with the height of the tree.
Now comes the more confusing part. We add the leaves.
First you need to understand that there are 4 layers in a birch or oak tree the top one never changing:

I also recommend adding trees where you have the most data first. Birch Trees are rarer in normal forest so i would definitely put them at the start(if you got a lot of information about them).



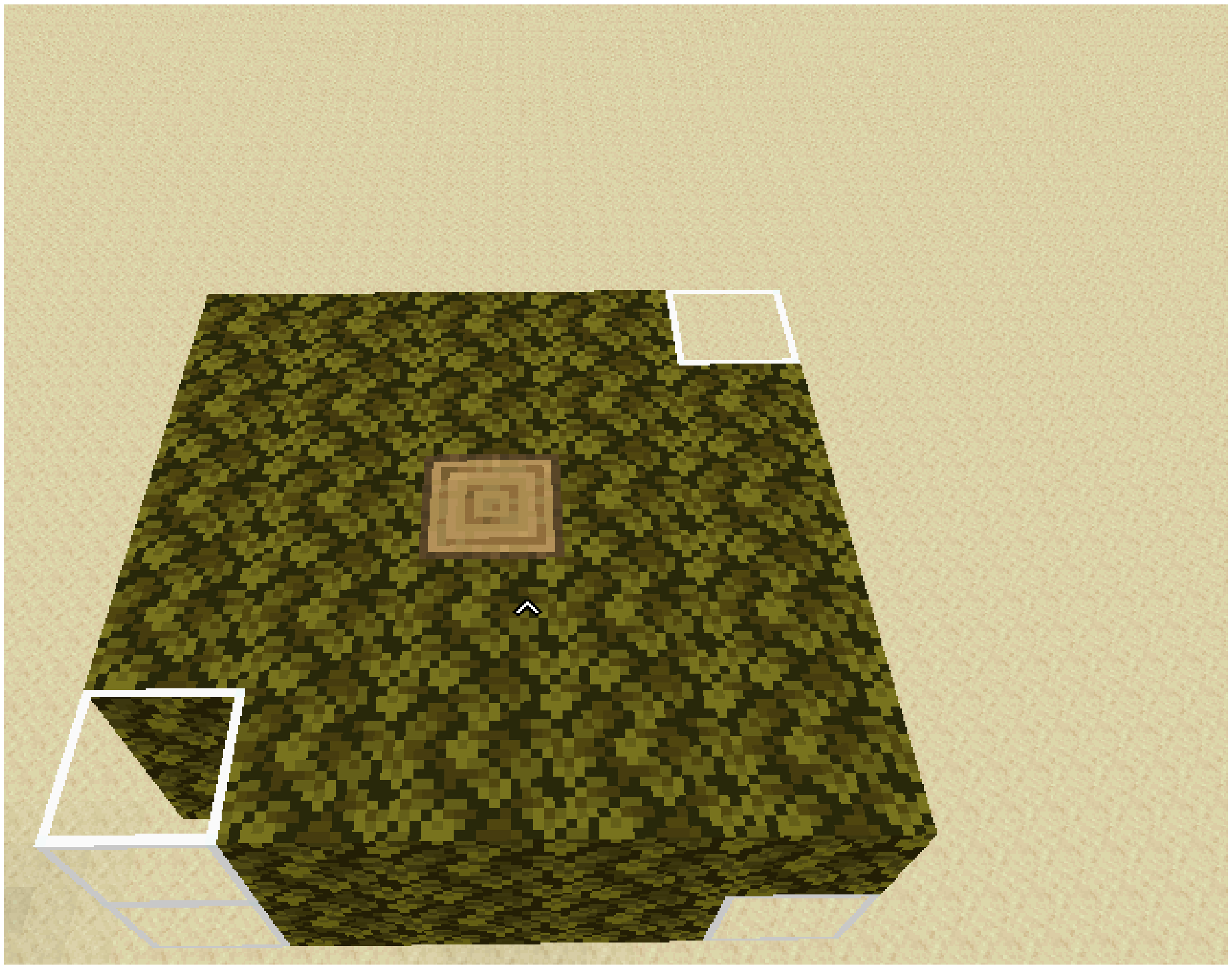
The target blocks are blocks that can exist or not exist(that is randomly decided by minecraft)

First we add the first layer of leaves:



This is an example of the first tree layer here we have all leaves missing means we use this(im facing north btw):
.tree_oak(x,z, IntRange(height), "0000""????""????")
There first 0 is the absence of the north-west leaf, the second is the absence of the southwest leaf, the third is from the northeast leaf and the last from the southeast one.
We go from northwest to southwest to northeast and then to southeast when inputting leaves on any of the three layers.
If any of the leaves are there you replace the corresponding 0 with an “1” and if you don't know if the leaf is there or not you replace it with a “?”.

Now onto the next layer:

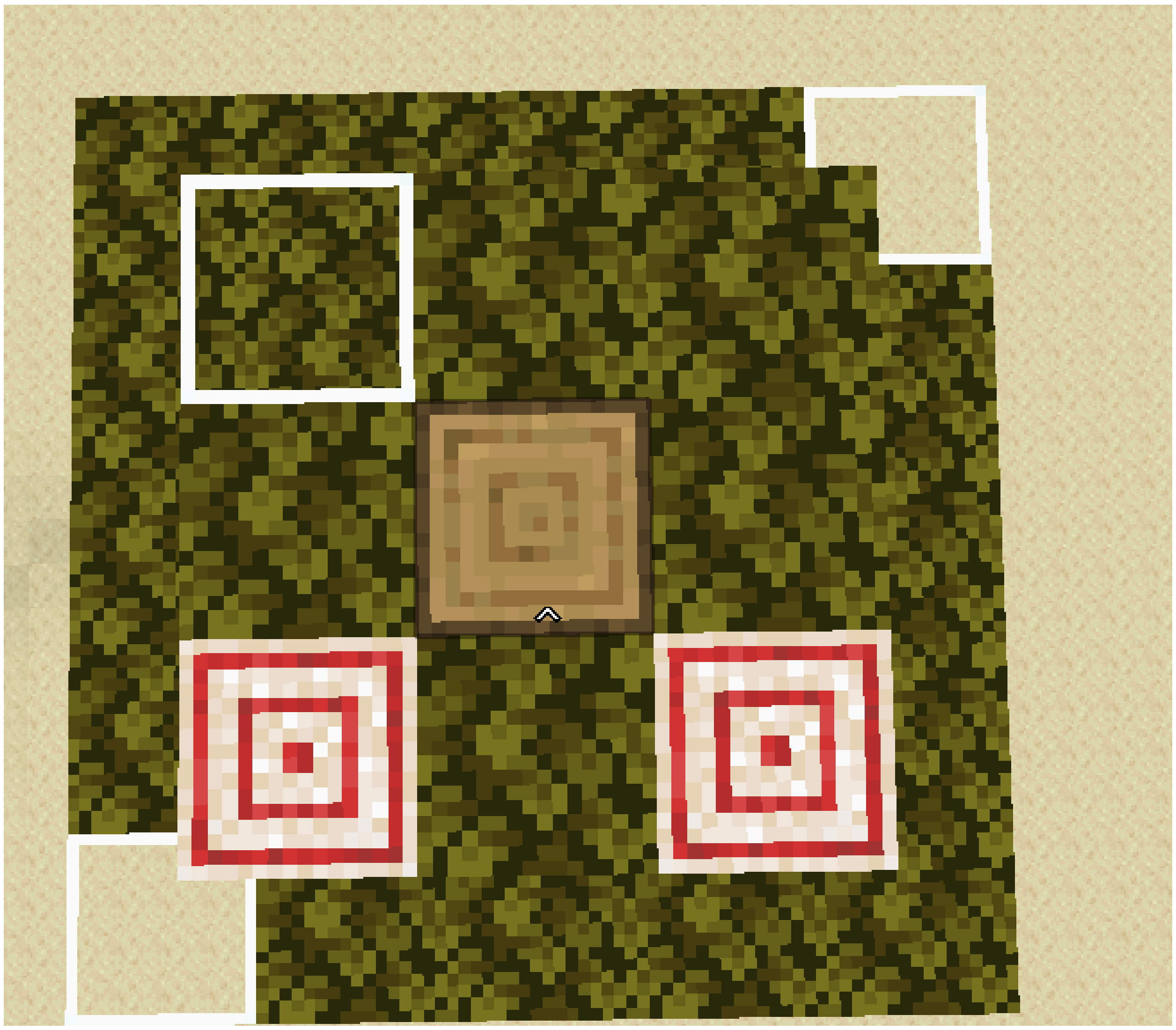


In this example im still facing north and we have the northwest leaf there so a 1 then we don't have the southwest leaf so a 0 Then we don't have the northeast leaf so a 0 again and then we have the southeast leaf meaning a one this results in the sequence 1001:

.tree_oak(x,z, IntRange(height), "0000""1001""????")

(It's in the second leaf input cuz we are in the second layer)

Now onto the third layer:



The target blocks indicate that we don't know if the leaf exists or doesn't exist.In the code we indicate that as a ?
So this sequence is from nw to sw to ne to se 0?1?
We input that into the third layer so we have this:
.tree_oak(x,z, IntRange(height), "0000""1001""0?1?")
And that's our completed tree!
You can repeat that with as many trees in a single chunk as you want(of course if they naturally exist only).

About 5 or 6 trees should be enough to get 48 bits.
4 trees can also be enough if you have a lot of leaf information.
To input a second tree you just put the second .tree_oak for example right under the last one like this.

```
.tree_oak(x,z, IntRange(height), "0000""1001""0?1?")
.tree_oak(x,z, IntRange(height), "0000""1001""0?1?")
```

If you are down you finish the chunk using

```
.build();
```

It should look smth like this

```
.tree_oak(x,z, IntRange(height), "0000""1001""0?1?")
.tree_oak(x,z, IntRange(height), "0000""1001""0?1?")
.build();
```

Don't create a second chunk specification because all of the trees are in the same chunk so they are in the same function right under each other.

How to run the TreeCracker(using google colab)

First you ofcourse need a google account.
Go to colab.research.google.com
Create a new notebook.
When you have created your new notebook you'll seed some options at the top of the screen. Go to runtime and there to change runtime. There you can select the T4 GPU(this should connect you to a new runtime if not you can just press connect at the top right of the screen, you should also see T4 GPU there).
This will provide you with a powerful enough Nvidia GPU to run the program(WARNING: Do not connect to the runtime too long else you will run out of GPU time, every user has a limit per day when they can use their GPU, in case you run out of time you can just use another google account).
At the left of the screen go to the folder option and create a new folder called src.
Then you open the extracted TreeCracker and copy everything from the src folder into the src folder in your notebook.
Then you need to compile the code for that you will click the +code button at the top left of your screen. That will create a new codecell in there you type this command.

```
!nvcc -o first ./src/main.cu
```

Then you run the codecell.
You should end up with a file called first(you might need to first exit the folder window and reopen it for it to update).
You run that file with the following command:

```
!./first
```

You should have to wait around 3 minutes up to abt one hour, (it depends on the amount of data) for it to finish.
When it's done you should get an output.txt when the folder tab is updated. In the output.txt you will find tree seeds, these are not world seeds yet.

Getting Structure and World Seeds

First step to getting a World Seed is to get a structure seed. To get a structure seed from our tree seeds we use a chunk random reversal tool.

https://github.com/Gaider10/PopulationCrr

To use this tool you're first going to have to download output.txt with all the population seeds.
Then you change your runtime to CPU only and connect to a new runtime.
Make a new folder again called src.
Download the CRR from the link above and extract it(you cannot import stuff from zip files in google colab).
Copy the contents of the src folder in the CRR to your google colab src folder.
Then run this command in a code cell:

```
!g++ src/main.cpp src/crr.cpp -o main -O3 -lpthread
```

This will compile the code.
After that upload the output.txt you downloaded with all the population seeds and then rename it to input.txt.
Now you have to run it.
That is the more complicated part. For 1.16.4 and 1.16.1 you will use this command.

```
!./main input.txt output.txt 1 1.13 80001 71 71 32 32 0 0
```

The last 2 values are set to 0 and the 71 in that example there are the x chunk coordinates so 71 being the minimum and 71 being the maximum(replace that with the x coordinates of your chunk) and the 32 being the minimum z coordinate and the other 32 the maximum z coordinate of the chunk(once again replace that with your chunk coordinates).
For 1.14.4 you do the same but replace 80001 with 60001

1.12 Is ofcourse a little more complicated.
The first command is the same as 1.16.4 and 1.14.4 but the second command is this one:

```
!./main input.txt output.txt 1 1.12 71 71 32 32 0 10000
```

The first 2 are once again the minimum and maximum x chunk coordinates and the after that are the maximum z chunk coordinates but now. You must use the population chunk coordinates instead.
You get them using this formula.
(x-8)>>4
How do you use the formula?
Well x is just a block coordinate inside the population chunk then you subtract 8 from it and then you open up the windows calculator. Go to the top left where you see the 3 - (the settings/configuration) you click on that and change to programmer. Now you are in the windows programmer calculator(i believe it's only windows 10 and 11 not sure tho).

Make sure you have selected DEC at the left of the screen.
Now input the block coordinate -8 and click the >> button, then press 4 and click =.
Do this with both x and z coordinates to obtain the population chunk coordinates.
Make sure you use block coordinates not the XYZ coordinates as shown earlier.
Now you can run the code with the right configuration of chunks.

After running the code you will get another output.txt.
There are possible structure seeds. Those are the lower 48 bits of a full 64 bit World Seed.

Download the output.txt just in case you need it.

In case you got too many structure seeds like more than 20 you can skip the World Seeds section of the document and first go to “Too many structure seeds”.

World Seeds.

To obtain world seeds from structure seeds we use a program called SeedCandy.
https://github.com/WearBlackAllDay/SeedCandy/releases
Download the latest .jar and run it.
In SeedCandy make sure you select the right version.
Now head to the structure seeds tab.
If the seed was randomly generated you can input all your structure seeds 1 seed per line and then click on reverse nextLong() this will give you all possible seeds that could have been randomly generated.
If your seed was not randomly generated you can use Biomes to filter the rest of the seeds. This should be self explanatory(this feature is also in the SeedCandy structure seeds tab).

Once again if you got too many structure seeds read the “Too many structure seeds” part of the documentation.

Too many structure seeds

If you got too many structure seeds you can filter them again using the second.cu file
For this you're going to need at least one more tree from a second chunk. Tho the more Trees the better.
Delete the previous input data and redo it with the tree/trees from the second chunk.
Now you need to compile second.cu you do that with this command:

```
!nvcc -o second src/second.cu
```

When refreshing the folders tab you should see a new file called second. This is the compiled code with the new chunk input. Now you get the output.txt that has all the structure seeds in it and upload it. Rename the output.txt to input.txt.
Now run this command:

```
!./second
```

This will filter the structure seeds into less structure seeds(not tree seeds).
You can get the world seeds of those using the methods i explained in the “World Seeds” part of the documentation.
If you still have too many you can redo this process with multiple trees and chunks over and over again.

If you result in 0 seeds it's most likely due to wrong data.

Huge Thanks to Andrew for creating the Treecracker and The Population chunk random reverser

Huge Thanks to 19MisterX98 for creating a program to find coordinates based on Texture Rotations.

Big Thanks to edd for showing me how to use the Treecracker(p.s i might consider adding a tutorial on how to use your macro in the documentation)