Jarosław Maciejewski

Minecraft in FrameVR.io

FrameVR.io[1] is an internet project that allows you to explore the virtual world, mainly using VR glasses, although you can also explore using a computer or smartphone. It offers, among others the ability to share your screen, voice chat, password protect your scenery, load your own scenery, add various resources to the scenery: pictures, 360° spheres, 360° movies, PDFs, videos, sounds or models. There are a lot of these options.

I decided that one of my scenery in FrameVR will be based on the famous world from Minecraft, which is available to the public at: https://framevr.io/nitro. It turned out that a lot of people were delighted with my idea of the scenery in FrameVR (which is very nice to me) and started asking me how I did it.

So, in this article, I want to provide a description from creating a world in Minecraft to exporting to FrameVR.

Here is a short list of my comments at the beginning:

- you should have Minecraft installed on your computer,
- you should have basic Blender skills,
- you should have node.js installed[2], which should be added to the system path,
- it does everything on Windows, but should be done on other systems without any problems.

Step one: Minecraft

We will start our adventure with Minecraft, but before we launch the game, it is worth drawing a preview of our world on a piece of paper or in your head. Then we will know what to do in Minecraft and we will save ourselves time for unnecessary thinking about what else to add to the scenery. Our world should not be large, all elements placed reasonably so as to fit within the limit imposed by FrameVR for the environment, which is currently 15 MB.

Before you run Minecraft, I recommend installing the following plugins:

- Fabric API[3] – API base for most other plugins,
- Fabric Mod[4] – a plug that "connects" the above API to other plugins (with Fabric API it is required),
- Warp Mod Fabric[5] – a plug-in that allows you to manage and navigate through a saved list of places,
- World Edit[6] – a known plug that allows you to perform some operations more easily.

1 FrameVR.io - web page: https://framevr.io/
2 NodeJS - web page: https://nodejs.org/en/download/
3 Fabric API - web page: https://fabricmc.net/use/
4 Fabric Mod – web page: https://www.curseforge.com/minecraft/mc-mods/fabric-api
5 Warp Mod Fabric – web page: https://www.curseforge.com/minecraft/mc-mods/the-warp-mod-fabric
6 World Edit – web page: https://www.curseforge.com/minecraft/mc-mods/worldedit

The above plugins are recommended but not required, however, it is worth making your work with Minecraft easier.

If we are ready, we have a plan, we know what we want to do, we run Minecraft (from the original instance in the latest version or Fabric instance), we go to the single player mode, we create a new world.

When creating a new world, we choose the following options (they are my recommendations):

- game mode: creative - access to all options,
- difficulty level: peaceful - all mobs are nice to us,
- allow cheats: enabled - we enable the possibility of entering commands in the chat,
- in Game Rules:
  - World Updates:
    - Advance time of day: off - turn off the automatic change of time of day
    - Update weather: wyłączony - we turn off the automatic change of weather,
    - Random tick speed rate: 1000 - we speed up the game time a bit,
  - Drops - here we give everything off,
  - Spawning - here we give everything off,
- More world options...
  - Generate structures: on/off - it's up to us whether we want to generate additional structures, e.g. villages,
  - Bonus chest: disabled - we don't need it,
  - World type: the type of map depends on us, I work on a super flat one.

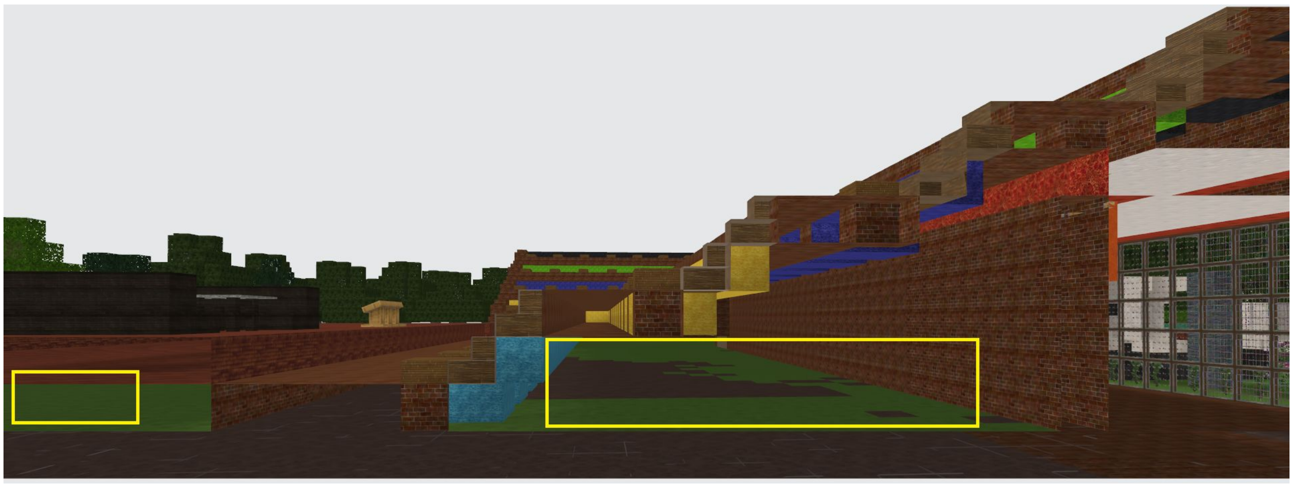
If we set everything up in the options we wanted, we create the world and enter it.
We immediately enter two commands in the chat, i.e. the sky is always cloudless and we always have noon:

```
/weather clear
```

```
/time set noon
```

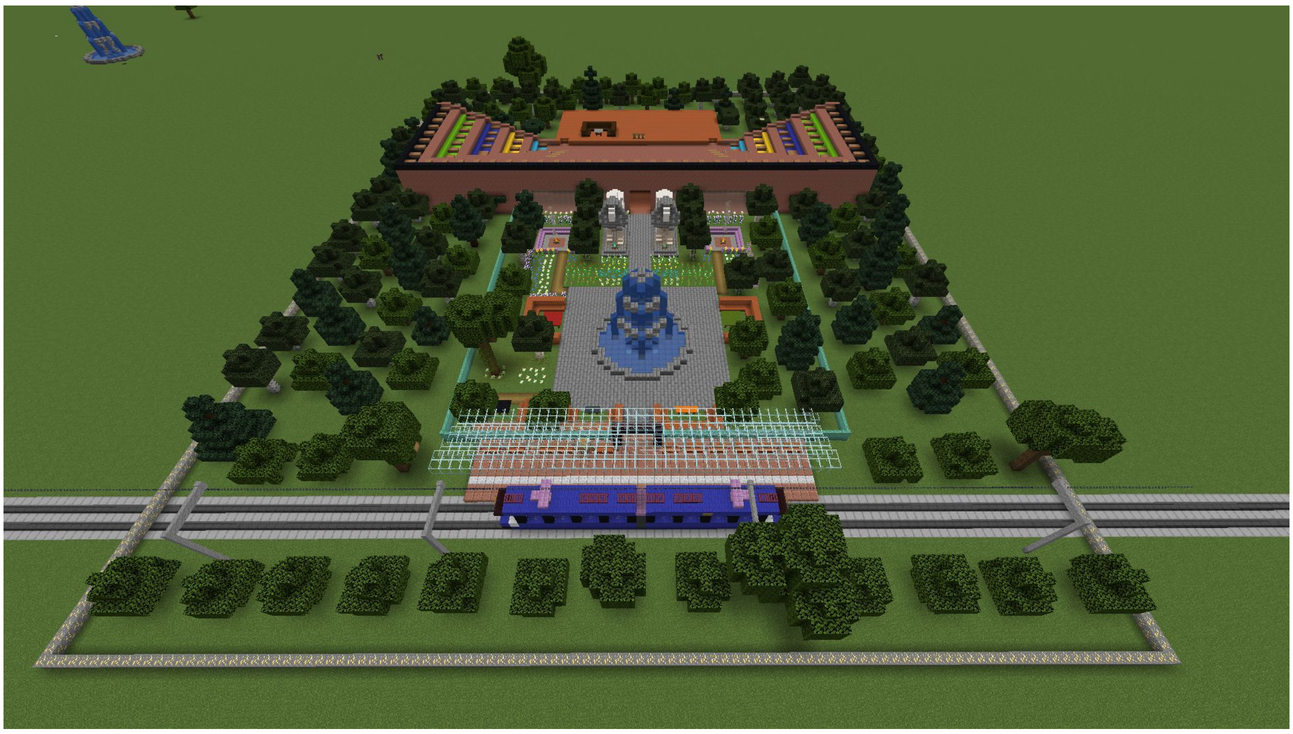
When creating our world, we keep the following tips in mind:

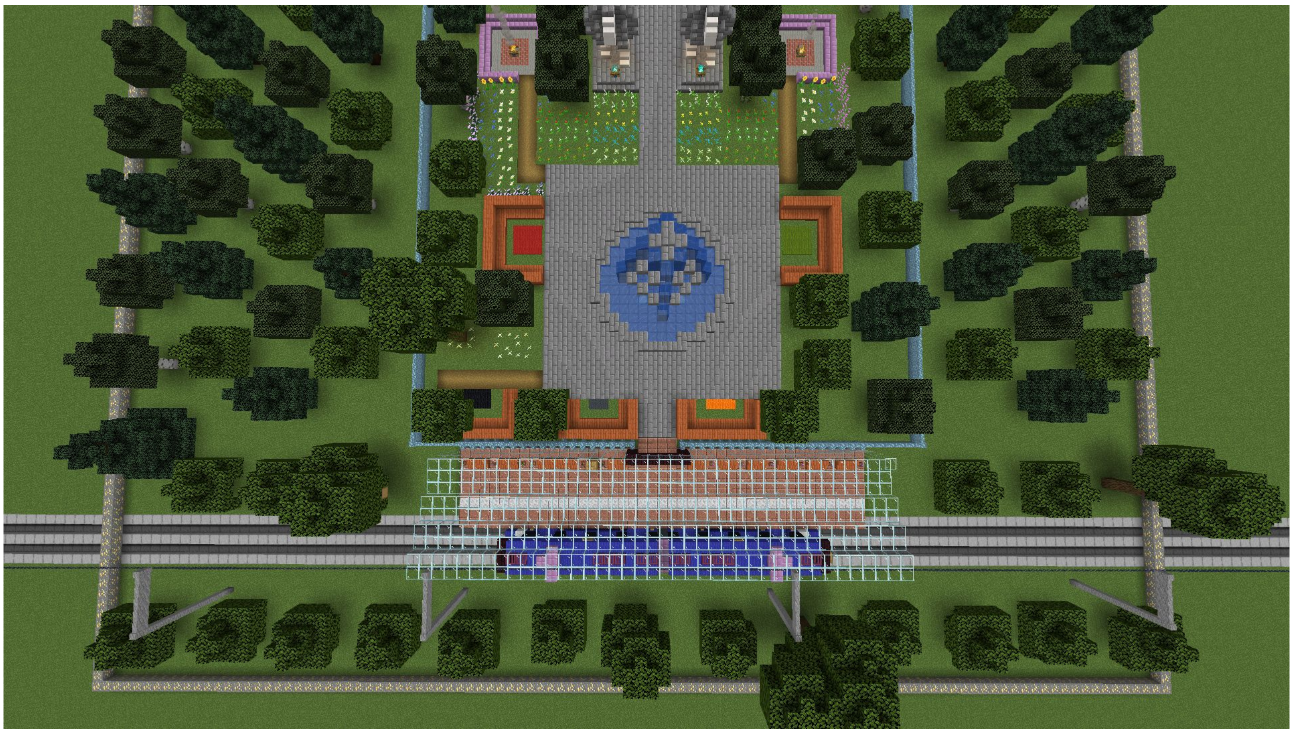
- there is no need to put text on the plates, because they will not be exported,
- we do not create full building structures, we delete those blocks that will not be visible to avatars because they will unnecessarily increase the size of the output file when exporting (Picture 1).

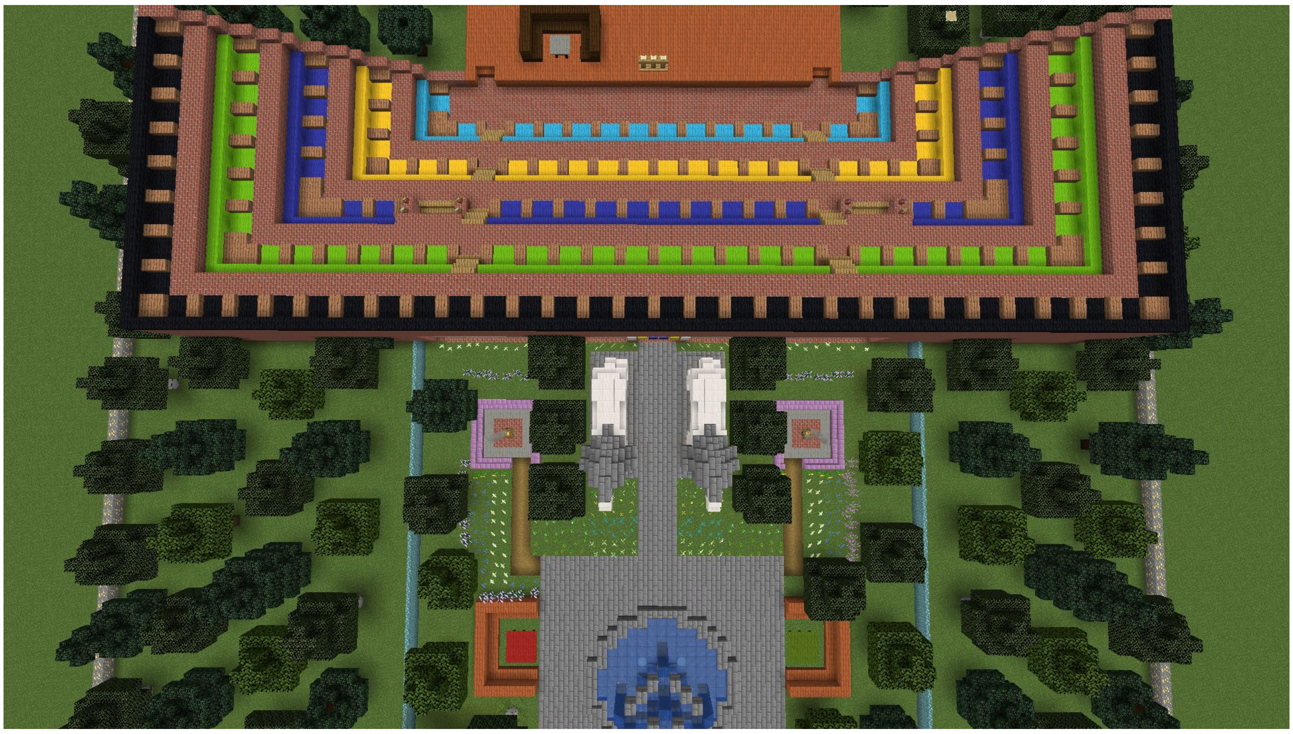

Picture 1: Empty spaces in the part invisible to avatars

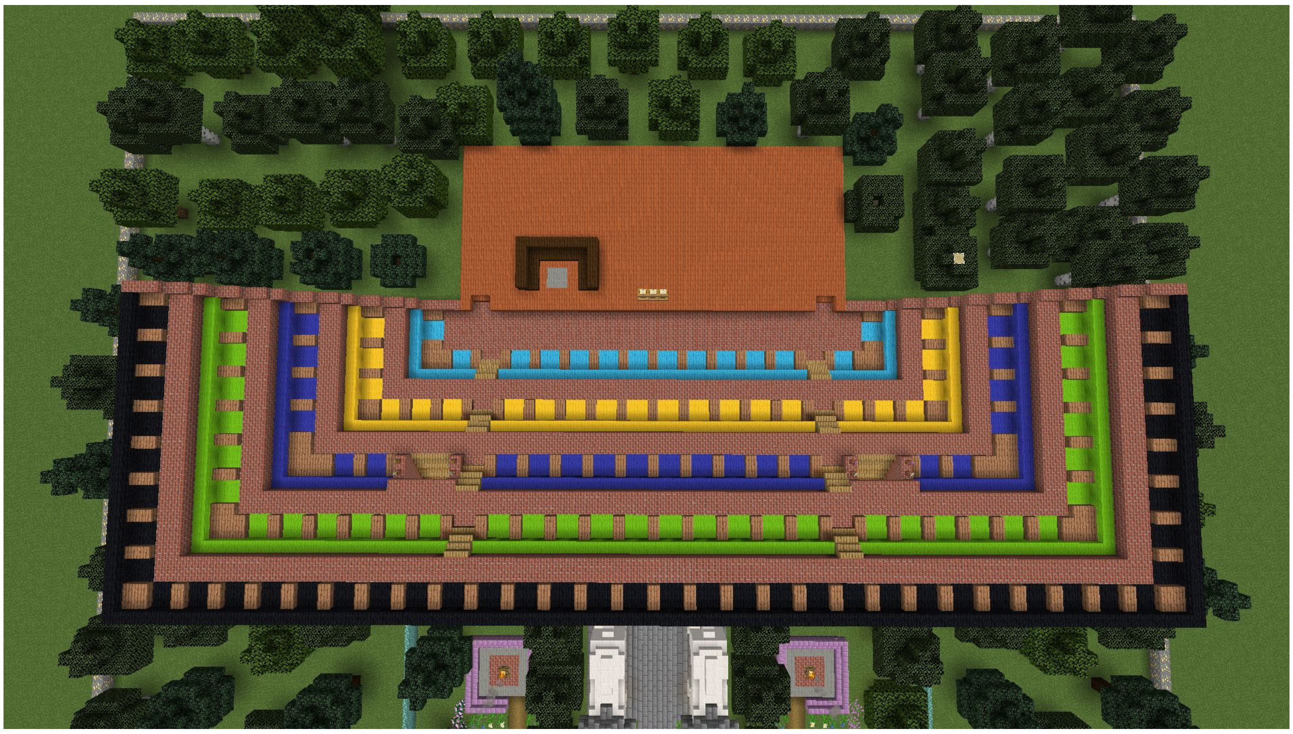
For example, in the picture above, you can see a cross-section of the stand, where there are no blocks underneath - the space is empty there.

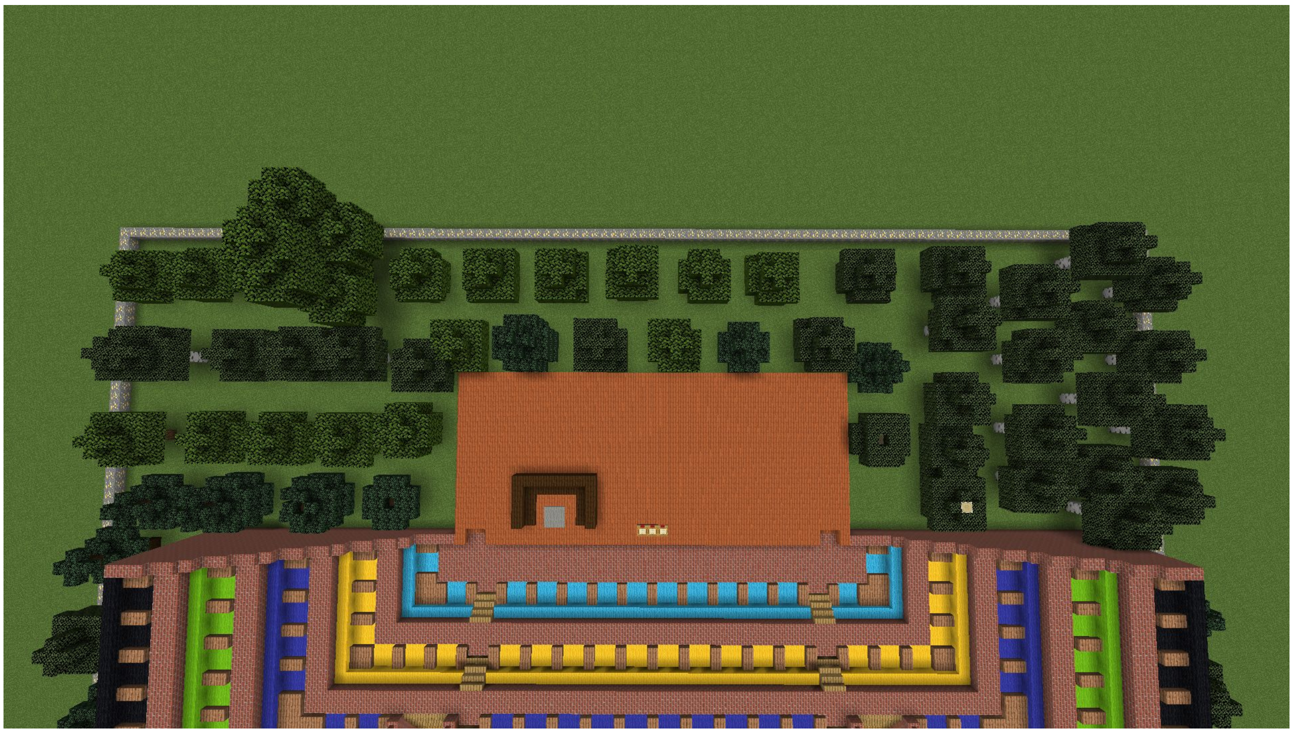
In general, my world in Minecraft looks like this:

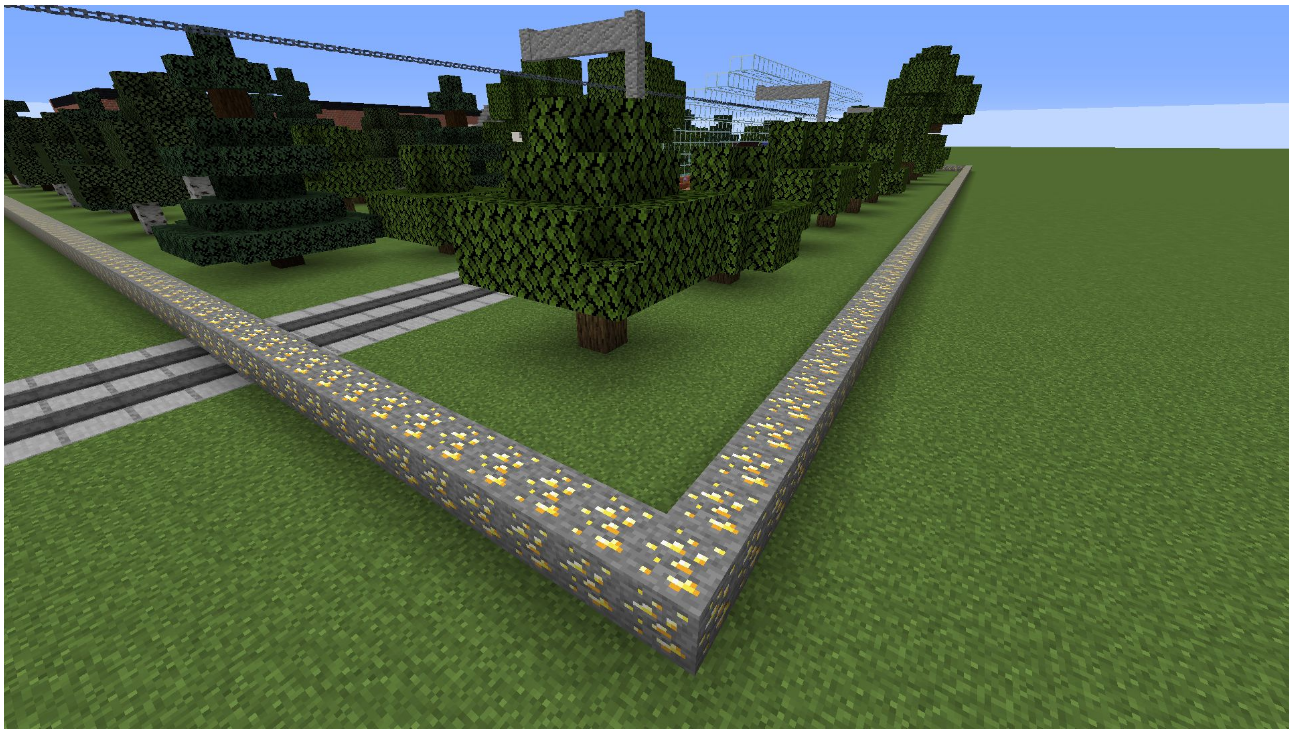

Picture 2: My world in Minecraft #1


Picture 3: My world in Minecraft #2


Picture 4: My world in Minecraft #3


Picture 5: My world in Minecraft #4


Picture 6: My world in Minecraft #5


Picture 7: The border of a sector of my world made of gold ore blocks

The characteristic rectangle made of gold ore blocks attracts attention (Picture 7). In my world, it marks the limits to which the planned idea is to be built.

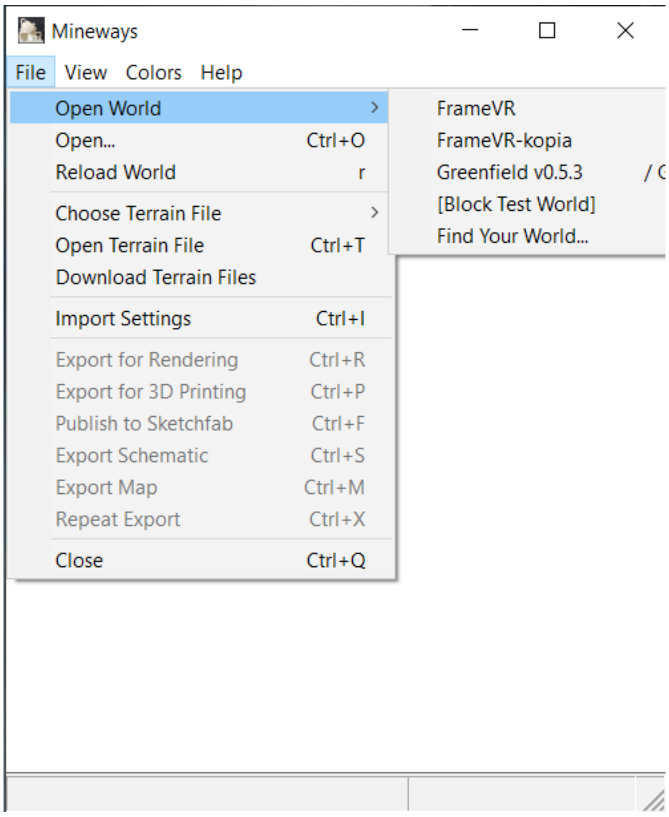
If our work in Minecraft is ready, we are leaving the world and the game.

Step two: Mineways

The next step is export our Minecraft world to OBJ file that can be easily read by Blender. To do this, we need the Mineways program to reach this end point[7], which we download from the website, unpack the ZIP archive and run it. Although the program description in this thread is quite extensive, the program itself is easy to use. I recommend that you close other memory eaters programs before exporting, because while exporting, we may get a message that the program was unable to reserve enough memory.

We start our steps in this program by loading our world into the program. We do this by clicking on 'File' in the top bar of the program and selecting 'Open World' (Picture 8). We will see here a list with the names of the worlds from Minecraft, so we choose our target from which we have made our world.

7 Mineways – web page: http://www.realtimerendering.com/erich/minecraft/public/mineways/downloads.html#downloadImgs



Picture 8: Select the world to be loaded into the program

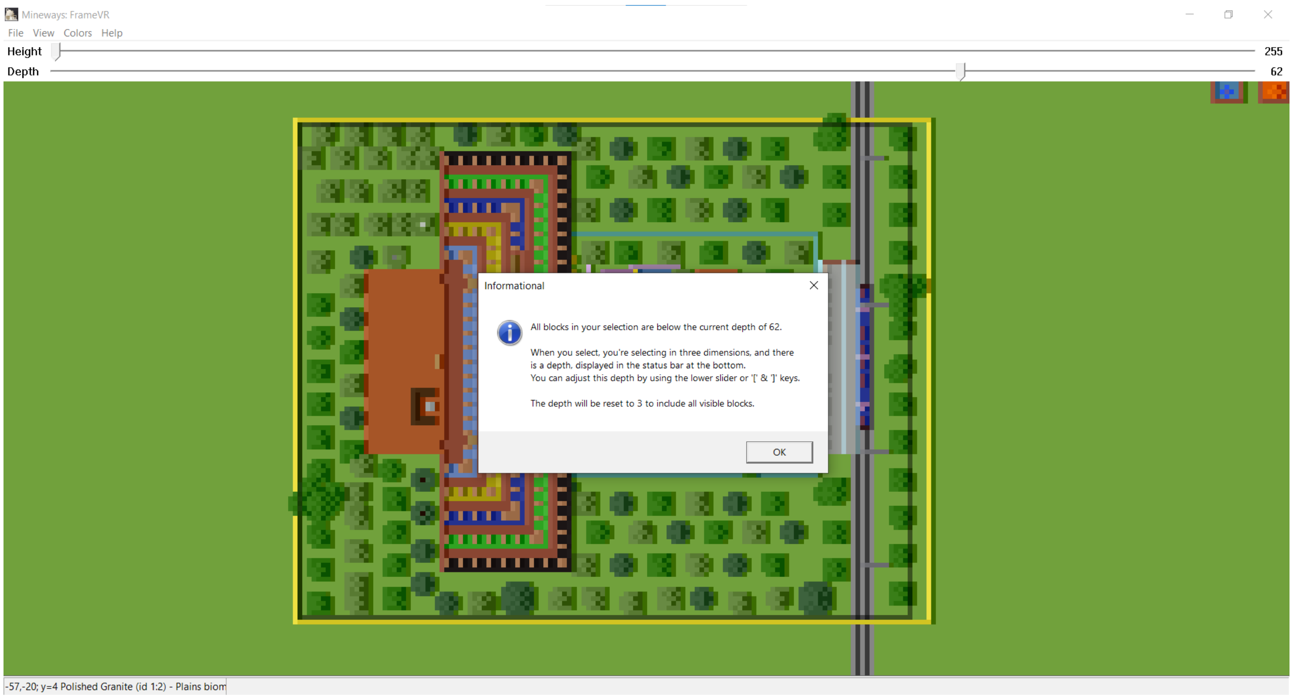
Our world will be loaded into the program. We see everything from above.
Operating the program with the keyboard and mouse is very simple:

- -/+ or moving the mouse wheel - zooming the map,
- holding the left mouse button - moving the map,
- holding the right mouse button - selecting the area to be exported.

A message will appear (Picture 9), which will inform you that the depth level has been set by the program and will be changed. Click on OK.

On the map, we see a section of our world that is surrounded by the characteristic yellow lines that are actually gold ore blocks.

We zoom in on the map and by holding down the right mouse button we make the appropriate rectangle that will cover the area we want to export (Picture 10). We can improve this area, just zoom in on the map, move it to the selected border and use the mouse cursor (it should be changed to the icon for changing the vertical dimension) to change the borders of the purple area.
Be sure not to mark the yellow lines when selecting the area.



Height 255
Depth 62

Informational

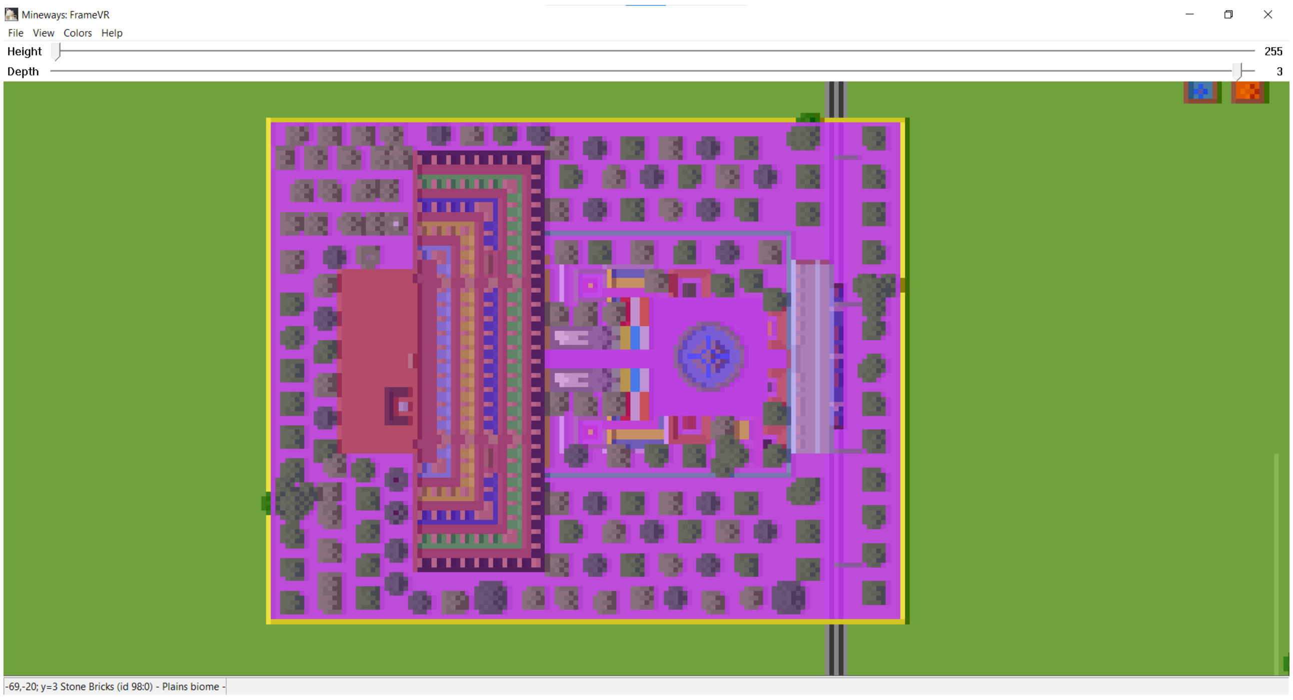
All blocks in your selection are below the current depth of 62.

When you select, you're selecting in three dimensions, and there is a depth, displayed in the status bar at the bottom.
You can adjust this depth by using the lower slider or '[' & ']' keys.

The depth will be reset to 3 to include all visible blocks.

-57,-20; y=4 Polished Granite (id 1:2) - Plains biom

Picture 9: Successful world import into the program with a message



Height 255
Depth 3

-69,-20; y=3 Stone Bricks (id 98:0) - Plains biome -

Picture 10: Selected area to be exported

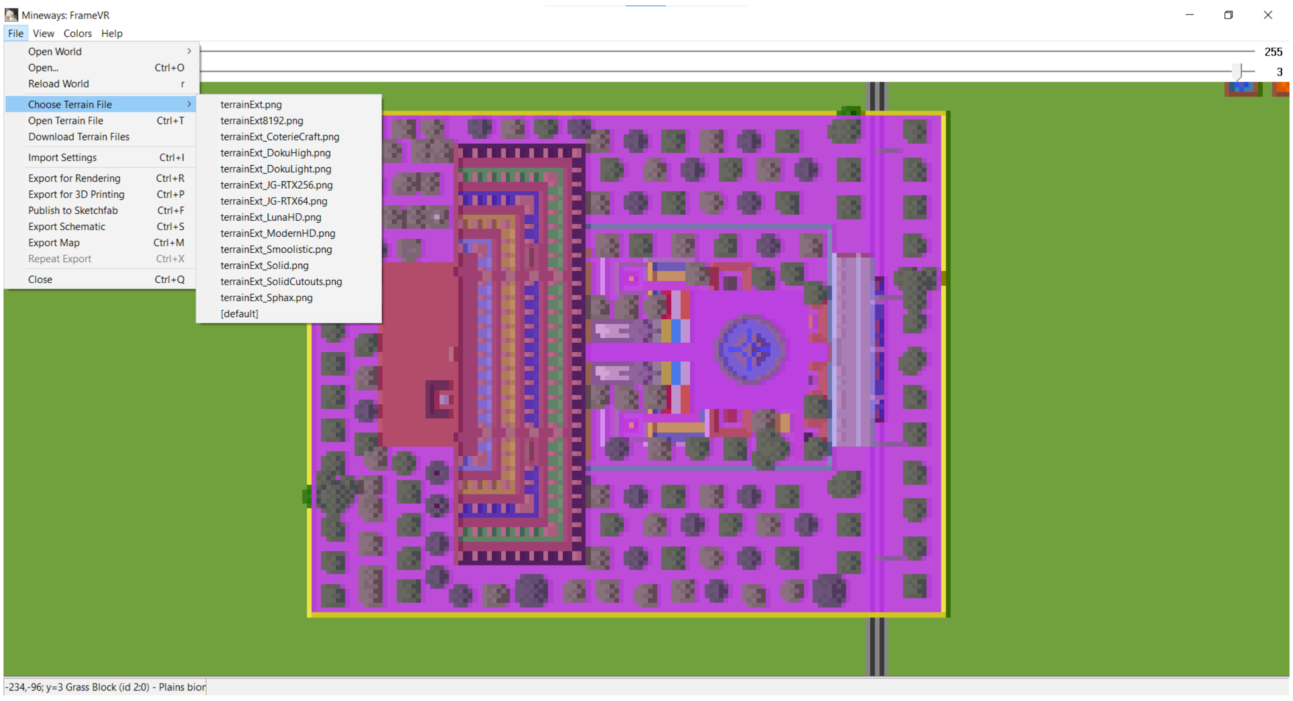
In the upper part of the program window you will find two bars:

- Height - the maximum height is 255 and that is what the bar is set to. Up to this amount, the program will take into account the blocks set.
- Depth - this is the minimum height from which blocks will be considered.

Now we decide what textures are to be applied to the individual blocks from Minecraft. To change it, click on 'File' => 'Choose Terrain File'. We have a list here with the names of the texture packs that are in the Mineways program folder (Picture 11). By default, the package used is 'terrainExt.png', so if we want to use only Minecraft block textures, we don't need to change anything here. Otherwise we can choose a different package from the list. The appearance of the individual texture packs can be found on the Mineways website (Documentation => Textures)[8], where there are project links to the Sketchfab website and other preview images.

I use the terrainExt_JG-RTX256 package, which contains other higher resolution textures and which are pleasing to the eye, especially since FrameVR is dedicated to people who will navigate the scenery wearing VR glasses.

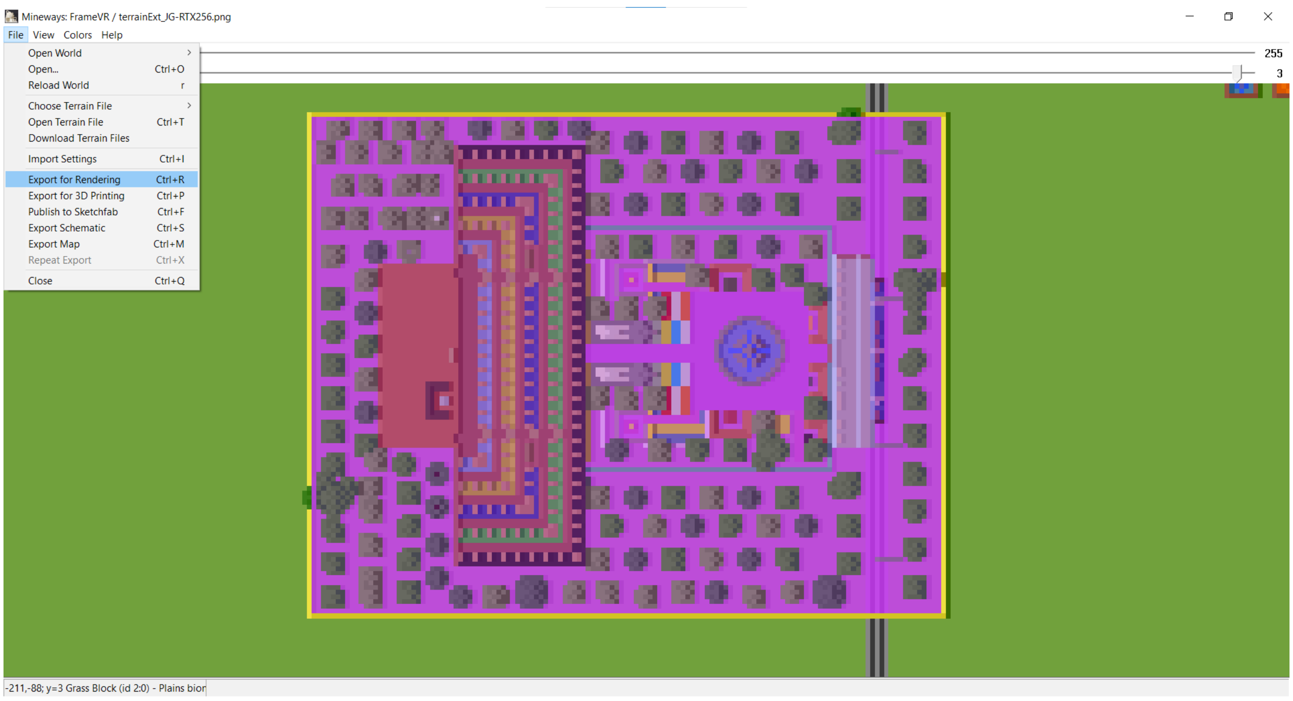
You can add your own texture assets to the Mineways folder. On the website of this program you will find instructions on how to convert to a single file, where it basically comes down to issuing two commands on the command line.



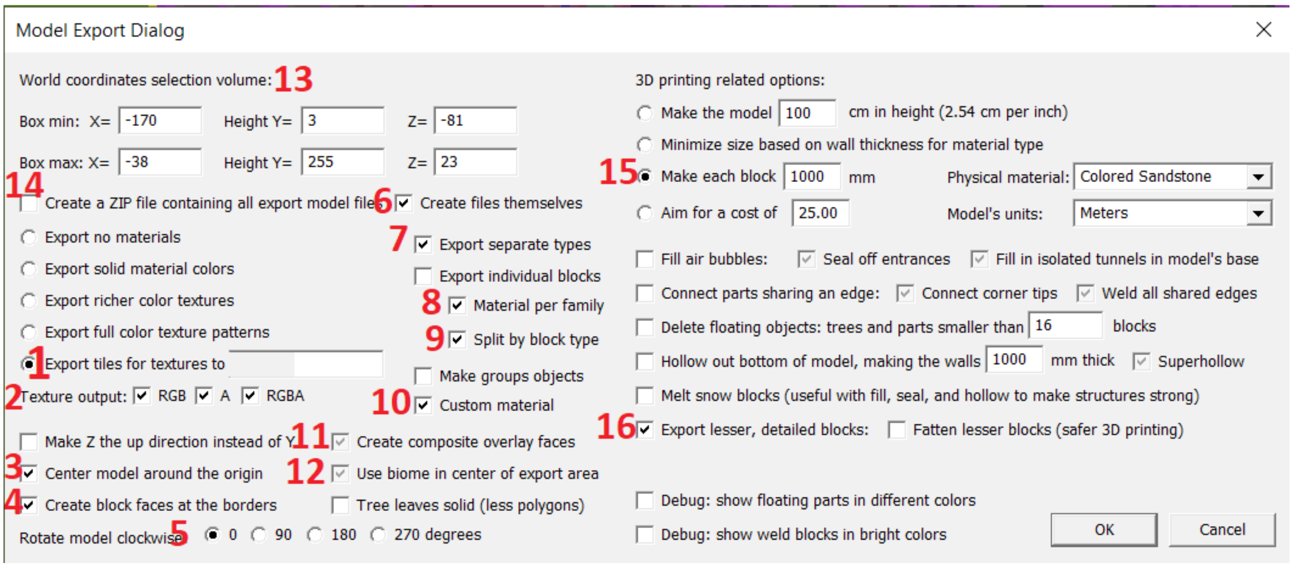
Picture 11: Texture List

From the 'File' menu, select the 'Export for Rendering' option (Picture 12) and in the new dialog we choose the location and name of the file, where we want to save our OBJ file with the selected fragment of our work. For the file type, make sure the option: 'Wavefront OBJ, absolute (* .obj)' is selected.
After clicking on 'Save' another dialog box will appear with more options to choose from.

8 Mineways – website, texture - http://www.realtimerendering.com/erich/minecraft/public/mineways/textures.html



Picture 12: Selecting the Export for rendering



Model Export Dialog

World coordinates selection volume: 13

Box min: X= -170 Height Y= 3 Z= -81

Box max: X= -38 Height Y= 255 Z= 23

14 ☐ Create a ZIP file containing all export model files 6 ☑ Create files themselves

○ Export no materials

○ Export solid material colors

○ Export richer color textures

○ Export full color texture patterns

1 ● Export tiles for textures to

2 Texture output: ☑ RGB ☑ A ☑ RGBA

7 ☑ Export separate types

☐ Export individual blocks

8 ☑ Material per family

9 ☑ Split by block type

☐ Make groups objects

10 ☑ Custom material

☐ Make Z the up direction instead of Y 11 ☑ Create composite overlay faces

3 ☑ Center model around the origin 12 ☑ Use biome in center of export area

4 ☑ Create block faces at the borders ☐ Tree leaves solid (less polygons)

Rotate model clockwise 5 ● 0 ○ 90 ○ 180 ○ 270 degrees

3D printing related options:

○ Make the model 100 cm in height (2.54 cm per inch)

○ Minimize size based on wall thickness for material type

15 ● Make each block 1000 mm Physical material: Colored Sandstone

○ Aim for a cost of 25.00 Model's units: Meters

☐ Fill air bubbles: ☑ Seal off entrances ☑ Fill in isolated tunnels in model's base

☐ Connect parts sharing an edge: ☑ Connect corner tips ☑ Weld all shared edges

☐ Delete floating objects: trees and parts smaller than 16 blocks

☐ Hollow out bottom of model, making the walls 1000 mm thick ☑ Superhollow

☐ Melt snow blocks (useful with fill, seal, and hollow to make structures strong)

16 ☑ Export lesser, detailed blocks: ☐ Fatten lesser blocks (safer 3D printing)

☐ Debug: show floating parts in different colors

☐ Debug: show weld blocks in bright colors

OK Cancel

Picture 13: Pre-export options selection dialog

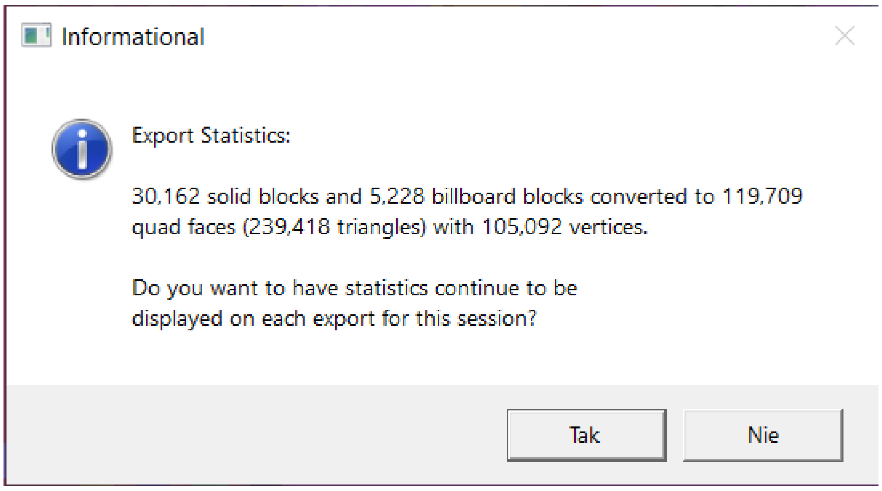
In this window (Picture 13):

- [1] click on the option: 'Export tiles for textures to', which will cause that all block textures will be saved separately to the 'textures' folder. If for some reason the field next to it is blurred, don't worry about it. This option also selects other options by default,
- [2] below the option described above there is an option: 'Texture output', where the options: RGB, A, RGBA should be selected,
- [3] 'Center model around the origin' should be marked,
- [4] 'Create block faces at the borders' should be marked,
- [5] 'Rotate model clockwise' should be set to '0' 'degrees',
- [6] 'Create files themselves' should be marked,

- [7] 'Export separate types' should be marked,
- [8] 'Material per family' should be marked,
- [9] 'Split by block type' should be marked,
- [10] 'Custom material' should be marked,
- [11] 'Create composite overlay faces' should be marked,
- [12] 'Use biome in center of export area' should be marked,
- [13] 'World coordinates selection volume' shows the coordinates of the selected area in the main program window,
- [14] We can decide ourselves whether we want the program to additionally create a ZIP file with all model files when exporting to OBJ,
- [15] 'Make each block...mm' - there should be a number1000,
- [16] 'Export lesser, detailed blocks' should be marked.

In fact, we don't have to bother with selecting the above options. Just compare your dialog with the picture and check [1], [9], make sure there is the right number in [15].

If everything is correct, click on 'OK'. The program will export our world to an OBJ file.


Informational

Export Statistics:

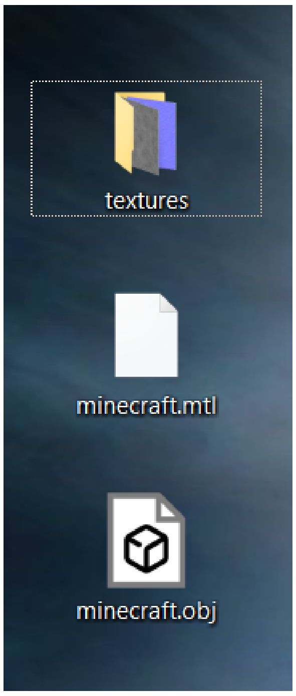
30,162 solid blocks and 5,228 billboard blocks converted to 119,709 quad faces (239,418 triangles) with 105,092 vertices.

Do you want to have statistics continue to be displayed on each export for this session?

Picture 14: Message about successful export to file

When the whole process is finished, a message will be displayed (Picture 14), which shows statistics and a query whether it should appear every time when exporting to a file until we close the program completely. Click on 'No' and close the program.

An additional 3 items should appear in the OBJ file save location: an .obj file, a .mtl file, and a textures folder (Picture 15).



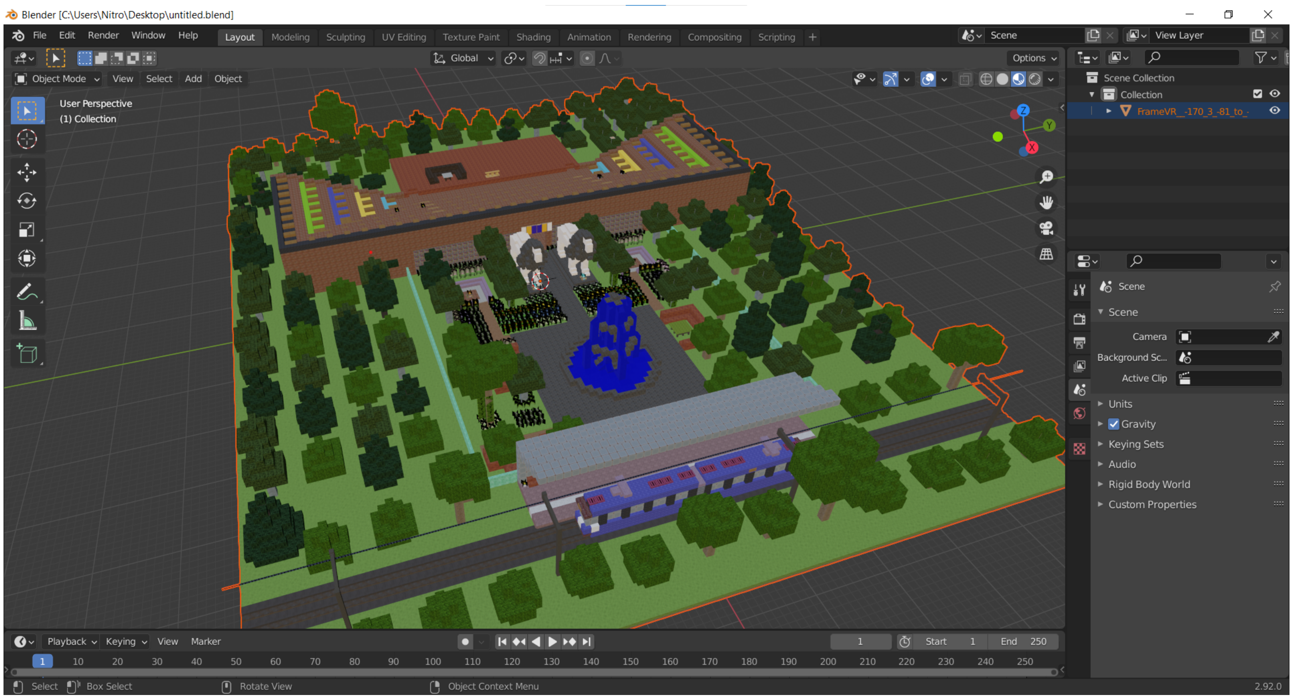
Picture 15: Exported OBJ file with textures

Step three: Blender

In Blender, we will process our model in more detail, add other objects and export it to a GLB file that is supported by FrameVR.

So we run Blender, delete the current content of the workspace, and import our OBJ model to Blender. To do this, select 'File' => 'Import' => 'Wavefront (.obj)' from the menu, select the location where our OBJ model is located, click the 'Import OBJ' button (there is no need to change the settings on the right windows). Our model will be imported to Blender.

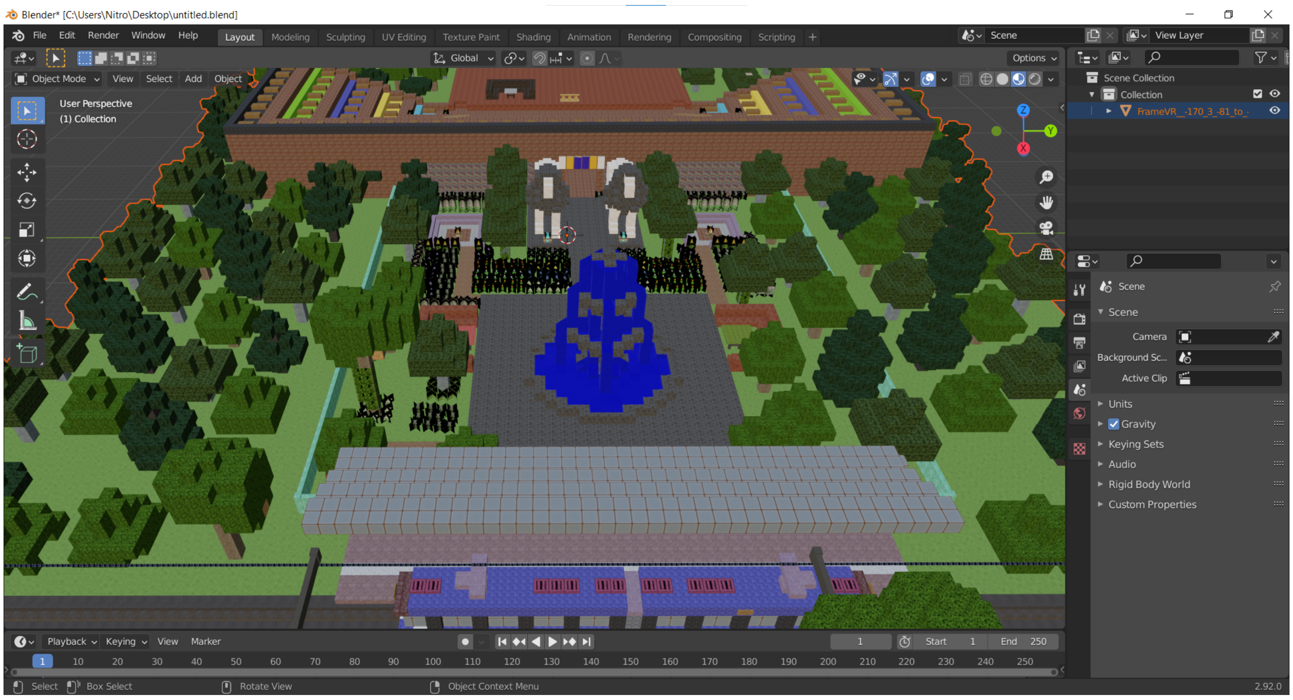
We change the way of displaying textures (key [Z] and then [2]) - 'Viewport shading' to 'Material Preview'. Thanks to this, we can see the model with all textures regardless of the lighting (Picture 16).


Blender [C:\Users\Nitro\Desktop\untitled.blend]
File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Scripting
Scene
View Layer
Object Mode View Select Add Object
Global
Options
User Perspective
(1) Collection
Scene Collection
Collection
FrameVR_-170_3_-81_to_
Scene
Camera
Background Sc...
Active Clip
Units
Gravity
Keying Sets
Audio
Rigid Body World
Custom Properties
Playback Keying View Marker
Start 1 End 250
Select Box Select Rotate View Object Context Menu
2.92.0

Picture 16: Imported Minecraft snip to Blender

We move our object to set the starting point for the avatars on the surface of the block. In Blender, this is easy as the starting point will be the 'World Origin' point indicated by two clearly intersecting lines: red and green. Initially, this point is also indicated by the locate, move, and transform cursor. Let's also set the appropriate height - with the 'super flat' world we lower the object.

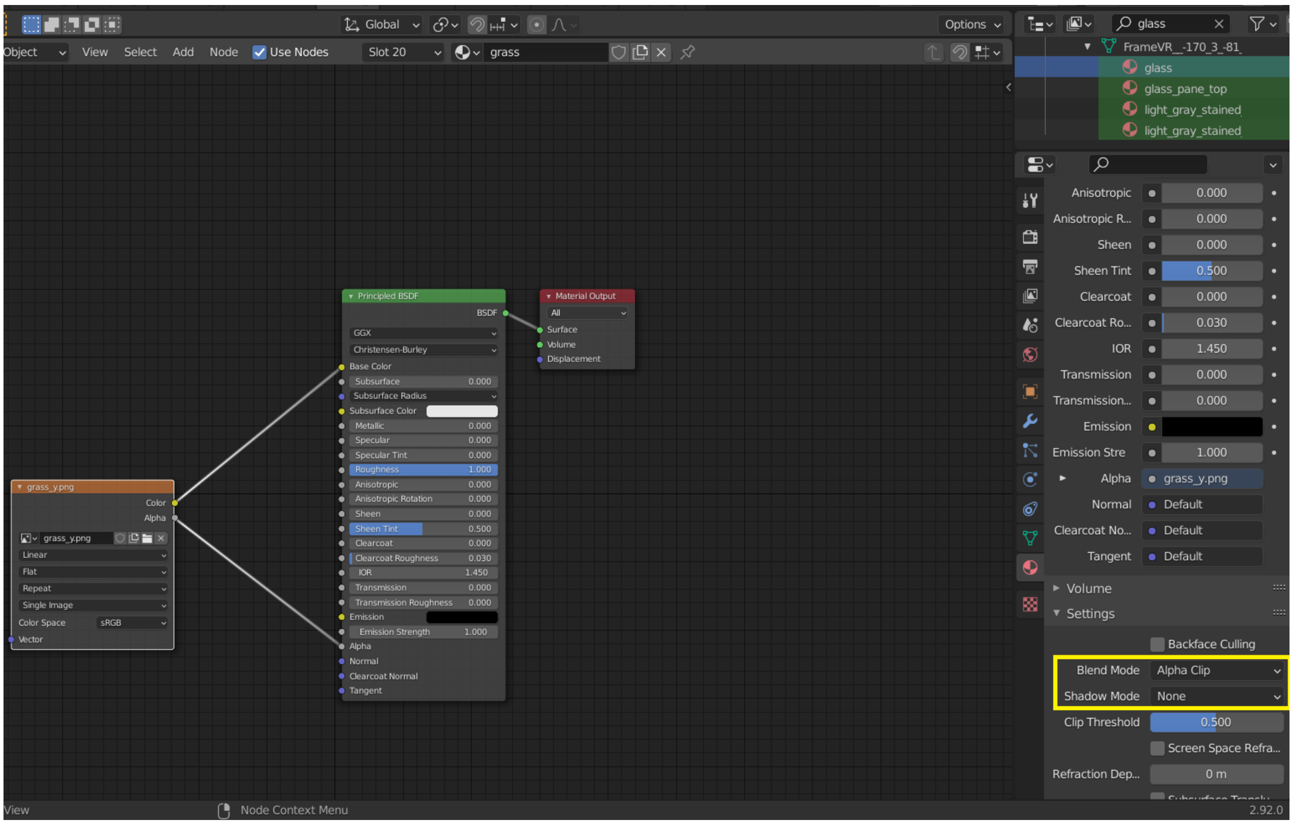
On the 'Object Properties' tab in the 'Transform' section, where we have given the location of the object, let's make sure that X, Y and Z contain only integers.


Blender* [C:\Users\Nitro\Desktop\untitled.blend]
File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Scripting
Scene
View Layer
Object Mode View Select Add Object
Global
Options
User Perspective
(1) Collection
Scene Collection
Collection
FrameVR_-170_3_-81_to_
Scene
Camera
Background Sc...
Active Clip
Units
Gravity
Keying Sets
Audio
Rigid Body World
Custom Properties
Playback Keying View Marker
Start 1 End 250
Select Box Select Rotate View Object Context Menu
2.92.0

Picture 17: A snippet of the Minecraft world from a different perspective

If we look at our world, we will notice some imperfections, e.g. a tram shelter made of glass is not transparent, the water is too blue, there are black borders around the flowers. It doesn't look very interesting.

To fix this problem, for these textures you should set 'Alpha' as the image referring to 'Base color' and 'Blend Mode' set to: 'Alpha Clip' or 'Alpha Blend' (Picture 18).



Anisotropic	0.000
Anisotropic R...	0.000
Sheen	0.000
Sheen Tint	0.500
Clearcoat	0.000
Clearcoat Ro...	0.030
IOR	1.450
Transmission	0.000
Transmission...	0.000
Emission	
Emission Stre	1.000
Alpha	grass_y.png
Normal	Default
Clearcoat No...	Default
Tangent	Default

Volume
Settings
Backface Culling

Blend Mode	Alpha Clip
Shadow Mode	None
Clip Threshold	0.500

Screen Space Refra...
Refraction Dep... 0 m

Picture 18: *Set an image as transparency + blending mode and shadows*

If we have a lot of textures, it is quite a time-consuming task.

That's why I'm here to help - we'll use my Python script to fix this problem. To do this, go to the 'Scripting' tab and create a new window. Paste my script in this window and run it (Script 1).

```python
import bpy
import re
from mathutils import *
from math import *

for material in bpy.data.materials:
    material.shadow_method = 'NONE'

    #When method=BLEND
    arrayMaterials = [".*water_still.*", ".*water_flow.*", ".*glass.*",
".*sunflower_back.*", ".*sunflower_front.*", ".*trapdoor.*"]
    for am in arrayMaterials:
        if re.match(am, material.name):
            links = material.node_tree.links
            nodes = material.node_tree.nodes
            material.blend_method = 'BLEND'
            links.new(nodes['Principled BSDF'].inputs['Alpha'], nodes['Image
Texture'].outputs['Alpha'])

    #When method=CLIP
    arrayMaterials = [".*leaves.*", ".*fern.*", ".*dandelion.*", ".*blue_orchid.*",
".*allium.*", ".*azure_bluet.*", ".*tulip.*", ".*oxeye_daisy.*", ".*cornflower.*",
".*lily_.*", ".*torch.*", ".*vine.*", ".*sunflower_top.*", ".*lilac.*", ".*peony.*",
".*campfire_fire.*", ".*campfire_log.*", ".*soul_campfire.*", ".*grass.*",
".*sunflower_bottom.*", ".*MW_bed.*", ".*bamboo_small_leaves.*",
".*bamboo_large_leaves.*", ".*ladder.*", ".*stonecutter.*", ".*bell.*", ".*cobweb.*",
".*_door_.*", ".*rail.*", ".*rose.*", ".*poppy.*", ".*scaffolding.*", "sugar_cane",
"sea_pickle", ".*fungus.*", ".*coral.*", ".*kelp.*", ".*mushroom.*", "crimson_roots",
"warped_roots", "nether_sprouts", "iron_bars", "chain", ".*lantern.*"]
    for am in arrayMaterials:
        if re.match(am, material.name):
            links = material.node_tree.links
            nodes = material.node_tree.nodes
            material.blend_method = 'CLIP'
            links.new(nodes['Principled BSDF'].inputs['Alpha'], nodes['Image
Texture'].outputs['Alpha'])

    #Show backface
    arrayMaterials = [".*water_still.*", ".*water_flow.*", ".*sunflower_back.*",
".*sunflower_front.*"]
    for am in arrayMaterials:
        material.show_transparent_back = False
        if re.match(am, material.name):
            material.show_transparent_back = True
```

Text 1: Transparency bug fix script [file: blender_script_set_alpha.txt]

This code fixes the mentioned errors with the transparency of objects, turns off shadows.
If you go back to the 'Layout' view you will see that the problems will disappear.

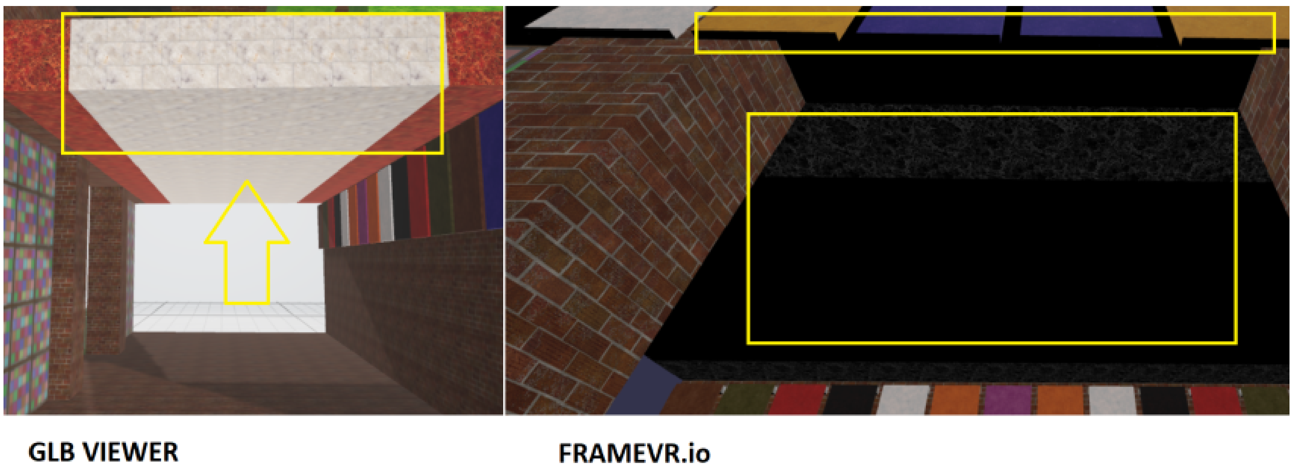
However, I encourage you to use 'Fly Navigation' and fly around the entire facility, looking for such strange errors related to the incorrect display of textures and fixing them manually. If you see something like this, add it to the script above, but I tried to include all known block textures in this script.

Only execute this script when you delete the old object and add a new object exported from Mineways.

If we have fixed bugs related to an object, we can add other objects to our Blender project, e.g. for me they are avatar Alex, avatar Steve, Golem, cat.

If in FrameVR our avatar cannot enter, for example up the stairs, add a transparent 'plane' as a board to climb at a certain angle (0-60°).

If we added objects, placed them appropriately on the scenery, etc., then we have to do one more operation. When we add such scenery to FrameVR.io, black spots can be noticed in some places (Picture 19), that appear on the underside of objects.



Picture 19: Black spots

To solve this problem, add the original material image as 'Emission' to each material, and set 'Emission Strength' to 0.3.
With many materials and many objects it is quite a time consuming task, so I come again with my own script (Script 2), which will quickly achieve this goal.
So we switch back to the tab: 'Scripting', delete the previous content, paste the following content and run:

```
import bpy
from mathutils import *
from math import *

for objects in bpy.data.objects:
for materialSlots in objects.material_slots:
if not materialSlots.name:
continue
material = bpy.data.materials[materialSlots.name]
material.shadow_method = 'NONE'
material.show_transparent_back = False
if hasattr(material.node_tree, 'nodes'):
nodes = material.node_tree.nodes
if hasattr(nodes, 'Image Texture'):
links = bpy.data.materials[materialSlots.name].node_tree.links
links.new(nodes['Principled BSDF'].inputs['Emission'], nodes['Image
Texture'].outputs['Color'])
nodes['Principled BSDF'].inputs['Emission Strength'].default_value = 0.3
```

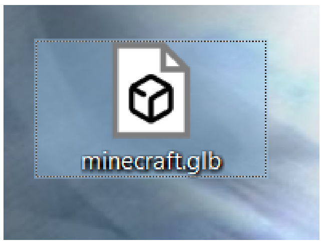
Text 2: Script that sets emission in objects [file blender_script_set_emission.txt]

We can execute this code every time we add a new object to our project.

If our Blender project is ready, it is worth saving it in Blender format to have the project ready for editing. Before saving, don't forget to enable the option to pack everything into one file ('File' => 'External Data' => 'Automatically Pack Into .blend').

The export to a GLB file looks like this:

- in the 'File' menu select: 'Export', and then: gITF 2.0 (.glb/.gltf),
- In the new window, indicate the location where the file is to be saved,
- Make sure that on the right side in 'Format' we have the selected option 'gITF Binary (.glb),
- Click on the button 'Export gITF 2.0',
- Our project will be saved to a single GLB file.



Picture 20: GLB file

Step Four: Compressing the GLB File

The maximum size of a GLB file that can be loaded as an environment on FrameVR is 15 MB. If our file has a larger size, we either reduce the number of elements on the Minecraft map and in Blender project, or we can try to reduce the size of the original GLB using a compression algorithm created by Google. The Draco algorithm add-on comes to the rescue, which is executed as a Node.js script called gltf-pipeline.

Open a command prompt in the folder where the GLB file is located and enter the command to install the gltf-pipeline project available throughout the system:

```
npm install -g gltf-pipeline
```

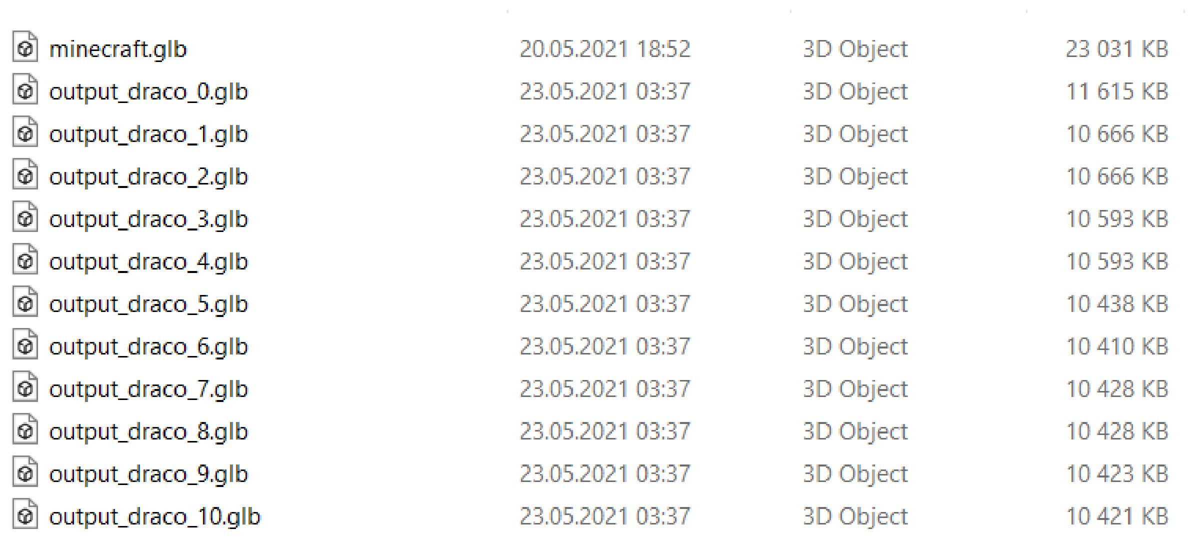
Then enter and confirm the following command:

```
for /L %i in (0,1,10) do gltf-pipeline -i minecraft.glb -o output_draco_%i.glb -d --
draco.compressionLevel %i
```

where:

- minecraft.glb - the name of the input file, which must be the same as the one exported using Blender.
- output_draco_%i.glb – output file name, %i it serves as information on the level of compression used.

We are waiting for the command to be carried out.



minecraft.glb	20.05.2021 18:52	3D Object	23 031 KB
output_draco_0.glb	23.05.2021 03:37	3D Object	11 615 KB
output_draco_1.glb	23.05.2021 03:37	3D Object	10 666 KB
output_draco_2.glb	23.05.2021 03:37	3D Object	10 666 KB
output_draco_3.glb	23.05.2021 03:37	3D Object	10 593 KB
output_draco_4.glb	23.05.2021 03:37	3D Object	10 593 KB
output_draco_5.glb	23.05.2021 03:37	3D Object	10 438 KB
output_draco_6.glb	23.05.2021 03:37	3D Object	10 410 KB
output_draco_7.glb	23.05.2021 03:37	3D Object	10 428 KB
output_draco_8.glb	23.05.2021 03:37	3D Object	10 428 KB
output_draco_9.glb	23.05.2021 03:37	3D Object	10 423 KB
output_draco_10.glb	23.05.2021 03:37	3D Object	10 421 KB

Picture 21: Original GLB file + GLB files after Draco compression

In the example image above, we can see the original GLB file exported from Blender and 11 other additional files that are output files using Draco compression. A compression ratio of 0-10 is given at the end of each name.

You can already see that with the compression at the 0 level, Draco reduces the original GLB file by almost 50%, then to level 2 we get even smaller files slightly over 50%. However, from levels 3 to 10, compression doesn't really matter that much.
Therefore, I even recommend modifying the above command to generate GLB with compression Draco at 0-2 [change from (0,1,10) to (0,1,2)].

If, even after compression using the Draco algorithm, our GLB file is too large, consider changes in the Blender project, which will reduce the size of the GLB file.

Attachments (embedded in this PDF):

- blender_script_set_alpha.txt
- blender_script_set_emission.txt
- commands.txt