**Boundaries**

There are no hard guideline to boundaries, but you can definitely run into issues if your stage’s blast zones are too big or small. Generally, Dreamland is a good baseline…

Dreamland’s DECIMAL x boundaries are -9000 to 9000 blast and its y boundaries are -3500 to 8300 blast.

However, it’s more than just the raw numbers, you need to consider the distance from platforms and edges.

Dreamland’s distance from edges to blast zone is 6682. The distance from top platform to ceiling is 6758. Your stage will become a bit more questionable if it varies more than 1000 from these distances.

**Vertices and Tris/ Console Lag**

If you plan on playing your stage on console, you need to consider the burden on the console. Two of the main sources of burden on the console are the number of vertices and triangles your model has. Your modeling program can give you a rough estimate, but the GE Editor has the more accurate count. It can go up tremendously on import to the game. To check this, go to Room Position Mode, right click your model, hover over Move/Scale/Rotate Room, then click calculate Tri Count. There are no hard numbers here, but look to the current models to get a good estimate. More than 500 triangles and 600 vertices will begin to cause noticeable lag. If your stage has more than one room be sure to add them all together to get an accurate count.

A good way to cut down on vertices is to remove doubles. This can be done in Blender by pressing w to open the specials menu or just clicking mesh > clean up > remove doubles. Another way to is to decimate your model. This will reduce the quality and change it considerable if used to a great degree, but can be used within reason. Also make sure to merge any vertices that are not connected, with the merge by distance option also in the clean up menu. If the vertices do not exactly touch they will not merge. Grab faces on your model and wiggle them to locate unconnected vertices then snap them together and merge them. Unconnected and duplicate vertices are common models dumped from emulator plugins, often is it best to use these as a reference to build your own models from instead.

The vertex count within the GE Editor is updated upon saving your edited rom. The Editor vertex count will in most cases be higher than it is in your modeling software. Due to the way the n64 renders, a vertex will be counted more than once if the triangles sharing it have different textures, texture flags or different UV mapping coordinates. Be efficient with uv mapping and edit advanced texture options sparingly. If possible merge multiple textures into one.

There are a few options you can use to get your vertex count lower in the editor. Any changes in room edit or lighting modes will likely cause changes to the vertex count so this should come after finishing any edits in room mode. In edit room position mode, Select all triangles with control+a right click and in triangle tools you will find group together primary triangles an reorder triangles for transparency/topflag/alpha. Use one or both and save your rom then check the vertex count. The results can be somewhat random and may take a few tries of choosing one or both of those options and saving and then rechecking the vertex count. Try to get the lowest possible count that results.

**Non Z-Buffered stages**

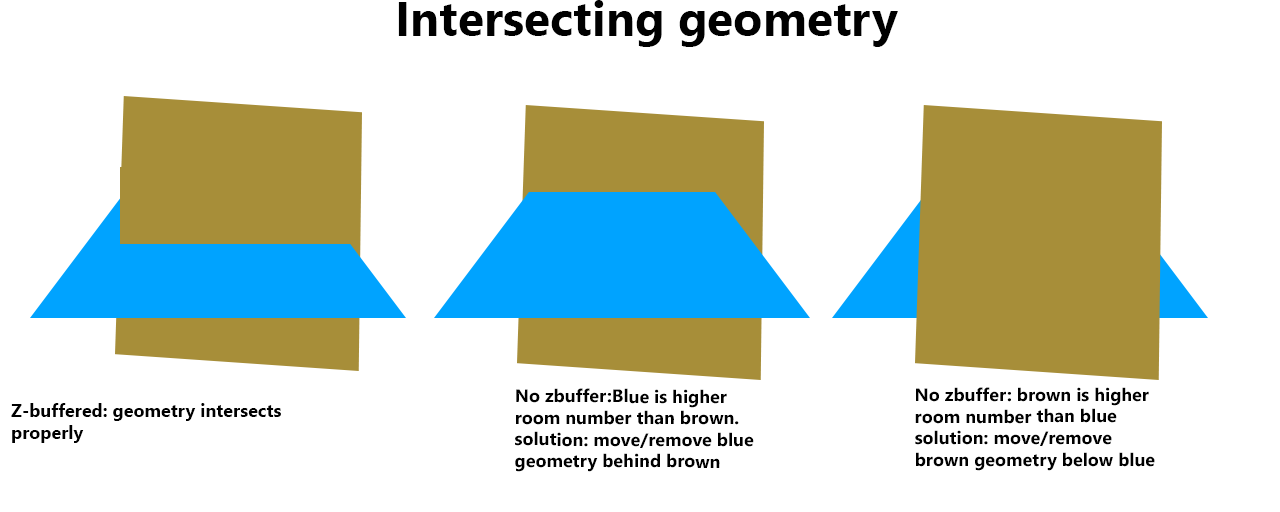
Stages that are just platforms with no foreground or background geometry will see no benefit from this. If you have a lot of foreground or background geometry This will give a decent boost to framerate since Stage models default to zbuffered for all rooms. The vanilla game stages are made in this way. First the stage model **must not have any intersecting triangles**. You will need to separate you model into different Room types Assigning them in the object or group name Ex. Room00\_Type00, Room03\_Type01

The available types are

|  |  |
| --- | --- |
| Type00 | Background Geometry. Anything behind the play area. Draws behind all other Types |
| Type01 | Play Area. The ground and platforms that characters walk on must be this type, as well as moving platforms. Type01 rooms are Zbuffered. This is the default type if no type is specified |
| Type02 | Foreground geometry. Draws In front of Type00, and type01 layers used for foreground geometry. Using this type on hyrule based stages causes crashes, |
| Type03 | Foreground. Draws in front of all other Types |

Check the types used in the level you are replacing and follow the same structure. You will need to split the level model into different rooms based on the draw order. Higher numbered Rooms of the same type draw in front of lower ones. Examine the how rooms are split and their types in the vanilla levels to get an idea of how to separate your rooms. You can use triangle tools> to front or to end in room position mode to force triangles in the same room to draw in front of or behind one another, this will have no effect on triangles in different rooms however. Make sure to test your changes on console regularly as there can be crashes on real hardware. There is a bit of trial and error involved here, It might be easier to only apply this to large setpieces For example Tree in the Deku Tree stage which was the main source of lag in that stage.

The two illustrative images below provide some explanation on how to order the rooms and tris of your stage.





**Untextured Tris**

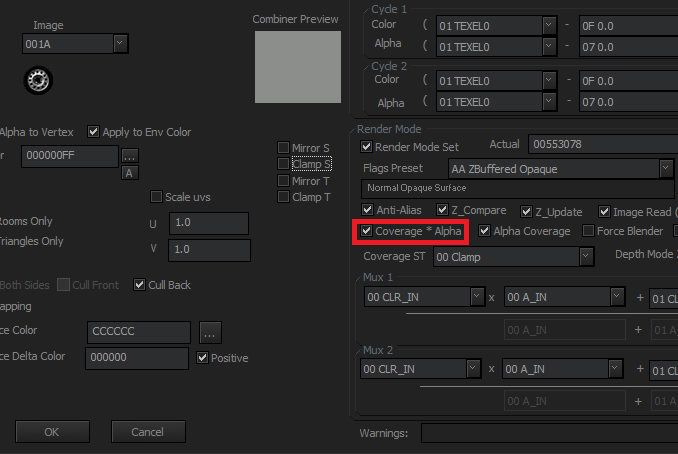
Untextured tris must all be at the beginning (Move to Begin), and the Texture LUT must be set to 0000 None. Also change untextured texture to FFFF in editor as you move to begin and turn tlut to none. That way the editor is sure to know it's untextured for output as combiner was used here but it still has texture commands and tlut on to rgba16.

**Stages Besides Hyrule**

To be able to modify Peach’s Castle and Kong Jungle, you need to make a simple edit to your ROM. You need to open your ROM in HxD and go to offset 921B8. At the end of the word there is an FFFF, change those to 0000. Make sure to do a checksum fix afterwards.

It is possible to edit a few other stages, but difficult due to inconsistency. When you open your ROM in the editor, you will see certain stages marked uniquely (Imgs A, Imgs B, Imgs C – those are the three shared sets A/A, B/B, C/C). You can modify these stages, but you need to modify both and they both must have the same textures (they don’t both need to use all the same textures, but they both must have the exact same texture file when you upload their model into the editor). Ensure you use the same texture.txt. You also likely need to delete all action block data. Even with this, it’s likely the stages will fail. You may need to save each stage multiple times. It still may not work in the end.

**Transparency**



You may want to put in textures that are transparent in part. In order to do this, you will have to use Room Positions mode in the GE Editor. The easiest way to do this is to make sure your transparent color is true black (in hex 000000). This is the default alpha color for the editor, but you can switch it as you see fit. Right click the triangle you want to make partially or completely transparent and click Replace Triangle. Click Advanced Options and make sure the Coverage \* Alpha box is checked.

For fully transparent textures (ex. Water) select the desired triangles and in their advanced options untick fog, just below that and to the left change cycle 1 preset to ‘NormalTextureVertexAlpha’ (2 options up from default). Next, under render mode clear the actual box and enter ‘5059F8’. on the left end of the

 Window check the transparent box, then click the “A” button on the left and enter the desired transparency value. 255 is fully opaque 128 is half transparency and so on. Finally click ok right click your triangles and find triangle tools > to end.

Importing a new model will reset advanced texture changes so if you are still iterating the stage model you will have to reapply any transparency options for your stage.

Alpha/transparent textures generally should be last in order of tris.

**Warping**

Your stage may begin to warp if it has faces that are very large. You will start to see strange effects, especially on console. To fix this issue, you will have to split the faces into smaller ones.

**Room Effects**

As you should know, each object in your imported model should be named Room01, Room02, ect. However, different rooms have different effects. Different Rooms can be in front of or beyond characters or effects. This has not been fully explored. Certain Rooms will move, specifically those in Peach’s Castle.

**Removing Hazards**

Removing the tornadoes from Hyrule is simple. You cannot delete the tornadoes, but you can move them beneath all your clippings. If you do this, they will never spawn.

For other stages, you will need to alter the ROM by using HxD to change the following values at their offsets to 00 00 00 00. Make sure to do a checksum fix afterwards.

0x81734: 0C 04 16 FA [Dream Land wind]

0x8364C: 0C 04 1A B0 [Sector Z arwing]

0x83C10: 0C 04 20 AD [Planet Zebes acid (VISUAL)]

0x851AC: 0C 04 26 22 [Mushroom Kingdom POW blocks]

0x8541C: 0C 04 25 28 [Mushroom Kingdom scales]

0x85424: 0C 04 25 DD [Mushroom Kingdom Piranha Plants]

0x857BC: 0C 04 27 A1 [Congo Jungle barrel]

0x86160: 0C 04 28 ED [Hyrule Castle tornadoes]

0x86974: 0C 04 2B D2 [Saffron City Pokémon] (does not eliminate door graphics)

0x86CB4: 0C 04 2C DE [Peach's Castle Bumper]

0xAA04C: 80 10 84 48 [Change to 80 10 56 F8 to disable acid COLLISION in Planet Zebes]

**Moving Platforms/Hazards**

Moving Platforms must be of type 01(see non z-buffer section for more details about types) and must be the second, third, or fourth room of type01. A stage with moving platforms must be imported over Peach’s Castle or Congo Jungle. You may use the clipping surface types to create moving hazards but walls don’t seem to work currently (you can use slanted floors instead). Clipping for moving platforms Must be mapped to the room for it. You can do this by right clicking the clip>change clipping room mapping. Match the clipping to the room you use for the geometry. For example, if my moving platform was the second type01 room, choose 02 for the room mapping.

A screenshot of a cell phone

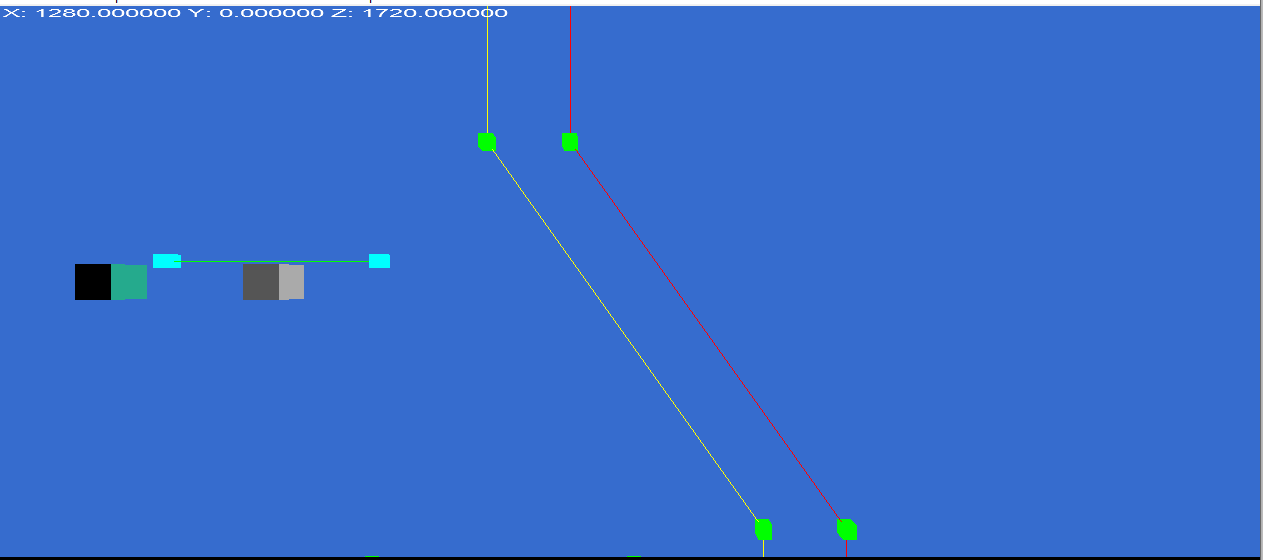
Description automatically generatedMoving platform clipping must be centered around the map origin, which should be located at coordinates 0,0,0 Now we use the action block editor to create the movement script. In the menus at the top select Edit Setup>Edit action blocks. Click reset all actions on a newly imported stage to remove leftover movement data from the original stage. Select the command track that corresponds to your clipping room mapping ex. Track 02.

In the add dropdown box, you’ll mainly want to use the Set Linear interpolation value [Blocking] command. Click add toad it to the list. On the right hand side, change bit to the type of movement you want, X coord or Y coord. X moves horizontally, y moves vertically. Rotation is an option too but only for decoration, clipping won’t move with it. Tick enable and enter the spot you want your platform to move to in variable1 field. Fill in the count field with how long in milliseconds you want it to take to move to that spot, as a hexadecimal value. You can use windows calculator in programmer mode to convert decimal to hex. Add additional points to move to in the same way, when you are finished add the loop command to make your platform loop its path indefinitely.

**Clipping**

Clippings can be tricky to deal with and can cause issues if not careful (or seemingly just at random). It’s important to test any clippings you have. Any clippings that are meant to be attached, should be snapped together via right clicking after selecting both points and choosing “Snap Together”.

Ceilings can be particularly odd to work with. Certain clipping angles should be walls and not ceilings and vice versa. For instance, the image below has parallel left and right walls:



If you were to make a similar layout, you would most likely have issues with the left wall. The angled portion of the left wall should instead be a ceiling. If not set as one, it could cause issues with your wall and allow a player to slide through. This angle is fine for the right wall since the game does not expect an angle to be considered a ceiling. This should make intuitive sense once you think about it.

Another common issue is grabbable ledges. You when making a ledge, both clipping points that form the corner should be marked as grabbable. If not, this will lead to an issue where players can get stuck in between the two clippings. Sometimes, even after doing this, you will be unable to grab the ledge or the ledge will act oddly. Generally, this can be fixed by making the adjacent clipping points also be grab ledges (this shouldn’t cause an issue if the adjacent points are on a flat plane and they would not normally be grabbable). Otherwise, you may want to simply delete the clipping and try again with new a new clipping or maybe even an unmodified ROM.

**Animated Textures**

To make a particular tris or set of tris swap between different images you must also use a “specialtexturemapping.txt” during import. The images generally should be 16 colors. More complicated textures and transparency can cause issues. Here’s an example of a special texture mapping for animated textures:

Special Image RGBA16: a.bmp

AltRGBA16: b.bmp

AltRGBA16: c.bmp

AltRGBA16: d.bmp

AltRGBA16: e.bmp

AltRGBA16: f.bmp

AltRGBA16: g.bmp

AltRGBA16: h.bmp

AltRGBA16: i.bmp

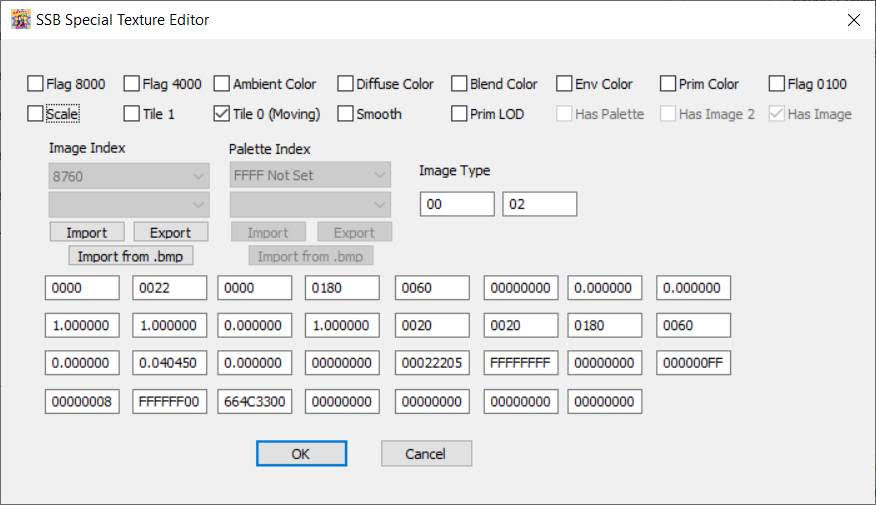
AltRGBA16: j.bmp

AltRGBA16: k.bmp

AltRGBA16: l.bmp

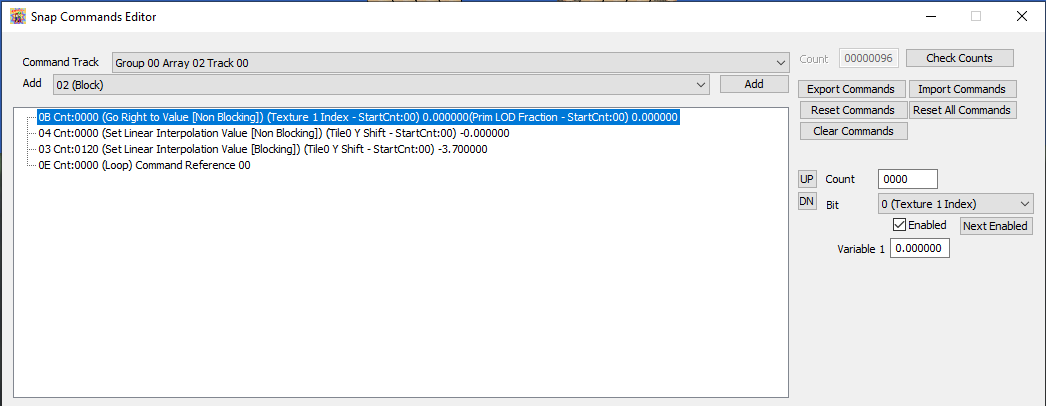
AltRGBA16: m.bmp

Image type should be 00 02 (RGBA 16). This is set in the SSB Special Texture Editor.



**UV Scrolling**

UV scrolling is a texture-based effect that causes the texture to loop across the faces it is assigned to. You can see this effect in Dreamland’s pond. In order to do this with a texture in a custom stage you must use a “specialtexturemapping.txt”, which a separate specialized file for animated textures. The texture can have no more than 16 colors. After you convert your model using the editor, go to actions and update the associated group (eg. Group01) array, (eg. Array 01) and Track (eg. 00) (make sure to enable the texture and prim lod fraction to 0.0). Here’s an example of a successful array:



**“Lights” (like Race to the Finish)**

If you want to add lights akin to the ones in Race to the Finish, it’s a little complicated. First, you’ll need to set the room to secondary. The texture/image you use will need to have its alpha channel properly defined, the Image must have alpha channel matching greyscale image (not all white). In your textures files, import the light texture as Transparent so it has transparency.

Change render mode to 005041C8 so it has the fade effect.

To do a color, instead of just white, you will need to set prim color. Set the vertex colors to match primitive colors and set combiner to FC309661442E7F3F to use vertex color instead of environment color. Turn lighting off so vertex colors are used.

You can change combiner to get different effects, such FC309661FF2FFF3F.

You can change the alpha in Prim Color to make it less pronounced.

You could also try, which looks more spot on as to the lights in Race to the Finish

Combiner FC309661442E7F3F

Prim Color FFFFFFFF

Render Mode to 005041C8

Lighting off (editor imports this way)

TLUT left off

Vertex color as real color (not env color, as they used, since editor doesn’t support).

If you don’t want to use alpha channel, and just auto-alpha and not need alpha channel, you want I image type (not IA). That’s light.bmp:80 (type 88 will get you IA).

Now if you want lights to move or “flicker” like Race to the Finish, you need to do a bit more (see ssblightingtest3.xdelta). This requires Special Image RGBA16. Add a SpecialTextureMapping.txt that is empty with just that single image (it needs a special entry/track or it can’t move). Import the level with that this time. You can do moving images if want too. Change things as stated before to get the correct alpha combiner/rendermode.

Now go to action blocks. Delete everything except the setting the Texture 1 index on each. Then import the snapcommands from the original race to the finish at end. You can change this as you see fit. It lowers alpha significantly in primitive color so just be aware. You can of course totally change this to suit your needs and alpha.

Now click Edit Associated Special Image button, and check the Prim Color checkbox up top.

Save + rooms, and now the light will change like Race to Finish.

You can use environment color (ssblightingtest4strobe.xdelta), which requires adjusting combiner back to FC309661552EFF7F (to use env color intead of vertex color/shade), set that too in the actions (that’s bit 1, while bit 0 is prim color), and toggle Env in the special texture mapping to get lights that change color.

**Textures (Misc.)**

The UVs for any texture should be 0.0-1.0, not something odd like 0.11 – 0.97. This causes texture glitches when done.

You may want to try to recreate specific texture effects used by the original stages, like the transparent clouds in the background of Yoshi’s Island. These are specialized textures, that are not the typical image format that the editor imports textures as (for instance the Yoshi’s Island clouds are 4-bit greyscale). To imitate them you can do this two ways. You can simply reuse the original texture in your stage and not add it to your textures.txt or you can change the image to its specific image type. Yoshi’s clouds are Type 80 (4-bit greyscale), so for the texture you want to imitate that effect, you need to

“00000760.bmp:80”

in textures.txt. The “:” followed by the image type number specifies what type of image it should be.

In either case, you’ll then have to set the correct image settings. You should mimic the settings shown in the original textures “Replace Texture” information in your new texture (ie. Combiner, Render Mode, ect.)

**Image Types**

|  |  |  |
| --- | --- | --- |
| Color formats | | |
| 0 | RGBA | Color and alpha |
| 1 | YUV | Luminance and Chrominance |
| 2 | CI | Index and look-up palette |
| 3 | IA | Grayscale and alpha |
| 4 | I | Grayscale |
| Bit sizes | | |
| 0 | 4-bit | I, IA, and CI |
| 1 | 8-bit | I, IA, and CI |
| 2 | 16-bit | RGBA, IA, and YUV |
| 3 | 32-bit | RGBA |