The Free and Open Independent Minecraft Map Maker's Magazine

# MAPMAG

## ISSUE 12 - FLIGHTS OF FANCY

## IMMERSIVE CUTSCENES
By Cavinator1

## FIRST TIME MAP MAKING
With Eris

## AMULET UPDATE
Tooling for the future

## VEHICLES
Ride along with Popular Youtube

## MAKING FLY SPY
The new science fiction flying game for Bedrock

# THE FLIGHTS OF FANCY ISSUE

Map Making is the ultimate craft in Minecraft. Building entire worlds, mechanisms, and narrative devices tests the skills and patience of even the most experience game designer. In this issue we tackle the big challenges in modern Minecraft map making.

To get you started, Eris walks you through jumping into the craft in “Confessions of a first time map maker”. Share the enthusiasm and delight as an entire world is created from nothing.

One important narrative device in the Map Maker’s inventory is the cut-scene. Cavinator1 explains how to hold a player’s attention and impart information using cinematic devices.

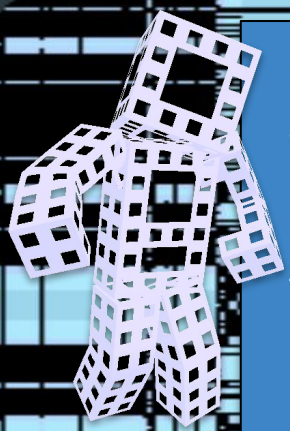
Blocky Vehicles help flesh out any map and are particularly effective in city streetscapes (which are a personal favourite of mine). In the article “Vehicles”, Popular YouTube explains methods to bring your vehicles alive using a range of design and build tricks that anyone can apply to their own creations immediately.

The tail end of this issue is an exploration of the map making process behind “Fly Spy”, a new science fiction open world flying map for the Bedrock Marketplace. In this article I walk you through the many different systems available to map makers working on commercial products for the Minecraft Marketplace.

As always, this issue has been crafted by members of the community in their own time to bring you the best and most current information available at the time of publishing. Because the Minecraft map making scene is constantly changing we rely on each other to share discoveries and provide support so as  to stay afloat when progressing our personal projects, so feel free to reach out to the authors of each article directly, through the @MapMakingMag twitter account, or by writing your own article for the next edition of the most excellent community Minecraft Map Making Mag on the planet: MapMag!

Until next time, happy editing!

@abrightmoore (Editor)



## Submission Guidelines

We are interested in what YOU have to say. Content you make for **Map**$^{Mag}$ can be sent to:

**mapmakingmag@gmail.com**.

The best letters, articles, art, and other work may be selected for inclusion in **Map**$^{Mag}$ editions or on affiliate websites and other communication channels. Because **Map**$^{Mag}$ is made by the community for the community, **Map**$^{Mag}$ is free for readers and we don't pay you for anything. You give us permission to include your work in the magazine.

Any content you submit must be your own work, or work that you have the right to submit. By sending us your work you agree that we may edit it for readability or make changes we think are necessary for the magazine. If we decide to include your work you acknowledge that you have granted us the right to publish your work in **Map**$^{Mag}$ and you understand that your work may be quoted or discussed on the internet by anyone in the world without limitation.

All other rights to your work remain with you. You own your work. We are allowed to use it for **Map**$^{Mag}$. It is that simple.

We will credit you by real name, game name, social media account, or another method that you prefer and that we mutually agree. We will not share your email address without your express permission. If you do not tell us how to credit you for your work then you may not be published in **Map**$^{Mag}$.

If we refer to you or your work in **Map**$^{Mag}$ you acknowledge that we do so in good will and our intention is not to damage or harm.

## DISPUTES

Writing about what you enjoy and hearing from other people with similar interests can be great fun. When people are excited about what they are doing sometimes things can get a little heated in a large community. If you have any concerns over what **Map**$^{Mag}$ is doing or how we are doing it then please contact us describing your concern. This will allow us to understand how we can do better. We can be reached at **mapmakingmag@gmail.com**.

By reading this magazine you agree that the Contributors, Production Team, and anyone associated with this activity are not liable for any damages to the fullest extent permitted under law. You agree that any dispute arising from this publication is governed by the laws of New South Wales, Australia.

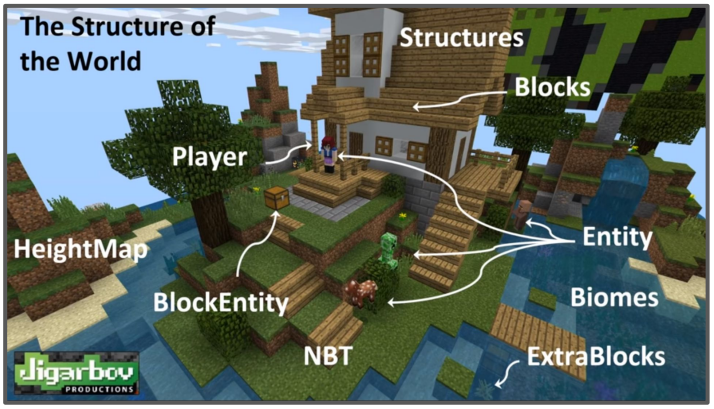
# AMULET UPDATE

Amulet is a tool that intends to provide offline design and edit capabilities to Minecraft worlds. It currently provides a layer of abstraction on the ever-changing Minecraft world format. It has been designed to provide easier ways for map makers to access their projects over time as Mojang and Microsoft continue to change and enhance the game.

The Amulet Editor project was introduced to the world in late 2018 at Minecon Earth's community panel on creating worlds with MCEdit. (You can watch the recorded stream by clicking the link to the right).

At the time of writing, many important decisions have been implemented in Amulet. Currently you can convert world formats forward and back over the chasm introduced by Minecraft 1.13's transition to blockstates.

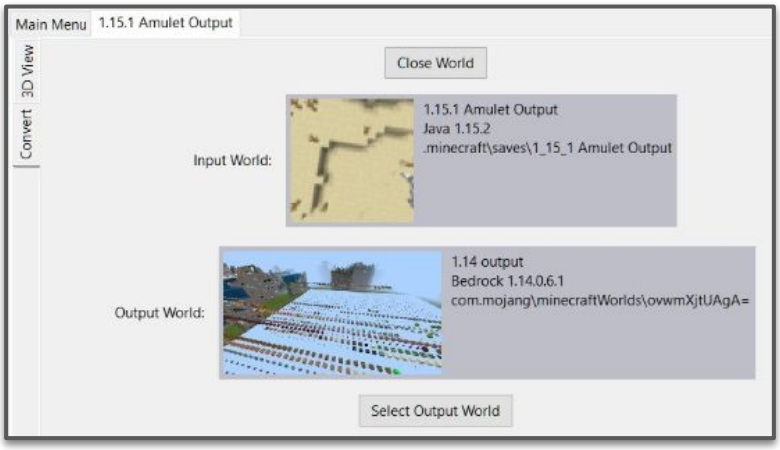
The navigation, selection, and editing of the world is being worked on. Early releases of the project provide glimpses of what this may end up looking like.




*Watch the recording of Amulet announcement here https://www.youtube.com/watch?v=f466vaGBx-0*

You can help the project team by trying out the releases and adding your feedback to the github issues lists. There will always be opportunities for the data transformation information to be expanded too.

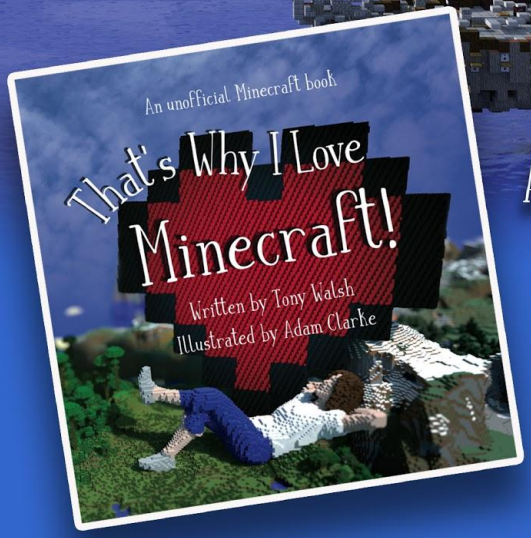
To learn more and stay connected you can join the Discord and connect with the team on Twitter at https://twitter.com/amulet_editor



# "THAT'S WHY I LOVE MINECRAFT!"

It sums up beautifully what every Minecraft player feels and what anyone who is yet to play can never fully understand.

Stampy Cat




A stunning celebration of the video-game that is helping to build a better world

Written by Tony Walsh
Illustrated by Adam Clarke

Published by The Wizard and The Wyld

Paperback £10.00

## www.why-i-love-minecraft.com

The Wizard and the Wyld

# STEPHEN REID

## @IMMERSIVEMIND

# Immersive Minds

Working with people to develop skills for:

- Work
- Learning
- Life

Communication
Citizenship
Critical Thinking
Numeracy
Analysis
Evaluating
Teamwork
Problem Solving
Creativity
Literacy
Negotiation
Justification
Empathy
Decision Making
Enterprise
Self Confidence
Judgment

### ICT in Education...

Using technology creatively to enhance learning across the entire curriculum and...

- Outdoor Education
- Employability
- Environmental Science
- Study Skills
- Motivation/Aspirations
- Alcohol Awareness
- Anti-Bullying/Cyber-Bullying
- Entrepreneurship
- Internet Safety
- Social Media Engagement

### Global Network...

Active projects with teaching staff in these countries...


### Pioneering Games-Based Learning...

Using games and play to enhance and support curriculum learning and life skills development, in children and adults...

- Pupil Training
- Professional Development
- Resource Development
- Teaching
- Public Speaking
- Project Management
- Consultancy
- Coaching
- Curriculum Design

### Minecraft in Education...

Using Minecraft to support learning across the curriculum...

A global Minecraft server dedicated to training and supporting teachers and parents.

In-Game CPD sessions per month


IT'S TIME TO GO PRO.
NEW MAP-MAKING OPPORTUNITIES AWAIT YOU AT WWW.PATHWAY.STUDIO.
@PATHWAYMC

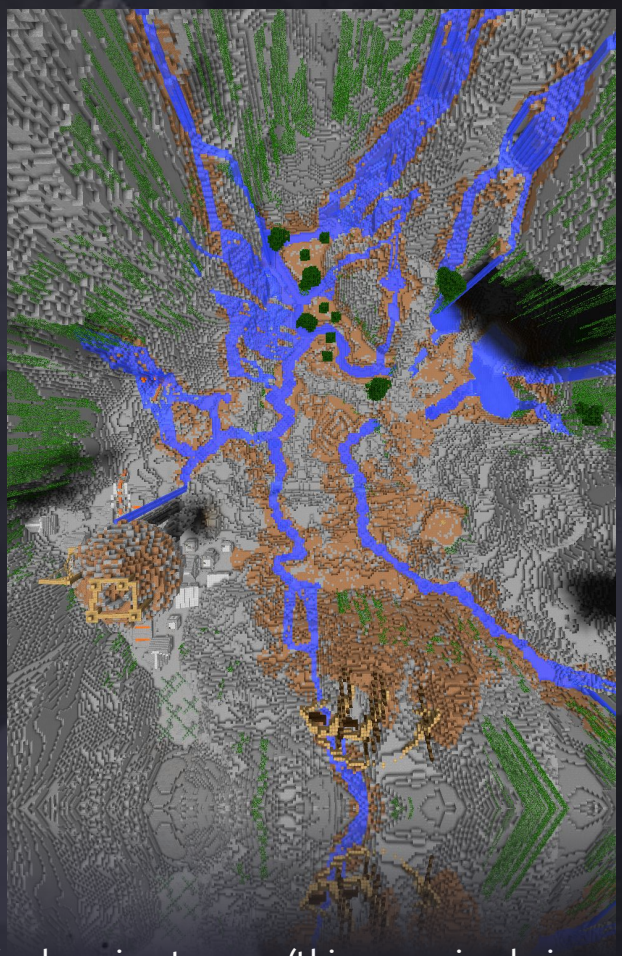
# Confessions of a first-time map maker

Written by Eris

Like any good CTM, Minecraft map making has many pitfalls; I fell into most of them. As I come to the end of the journey of making my first map, I wanted to look back at the process.

The Malevolent Chronicles: Chapter 1 – Harbinger is in closed Beta and will be released in November 2019. I've tried to keep this article fairly spoiler free whilst showing off a little bit of what the map has to offer.

In the interest of brevity, I won't go into detail on every single mistake I made here, but I did want to talk about some of the things which held me back.



## Minecraft is infinite, your time is not:

There are many amazing resources out there for learning to map (this magazine being one of them!), but it's important to keep things in perspective. I was very hung up on knowing *how* to make a map. My logic went that to make the best map, I needed the best tools. I watched hundreds of YouTube videos, sought out every MCEdit filter and WorldEdit tutorial I could find. I tried to learn everything I could about mapping.

Mappers have more tools than ever before and there's still more on the way. There's so much you can learn about how and when to use these tools but ultimately, learning these things takes time. I found "Tool Hunting" wasn't always a good use of that limited resource. Almost always the best thing I could do was to use the tools I knew and just start making stuff.

The best way I found was to repeatedly try and fail to turn my ideas into something tangible in game, even if the results were often dreadful. Spending all my time trying to read about or watching videos on how to make stuff wasn't going to make me good at making stuff.

My first couple of "areas" were just big cubes with some lava in. They looked bad and played even worse. It didn't matter however, because I thought they were incredible. It was incredible that I'd been able to make something myself, from scratch. Going through this process was great for me firstly because it taught me how to make better stuff, but more importantly it was fun! Ultimately that's the only reason I kept coming back to it.

Had I not been through that process, I wouldn't be able to make the stuff that I am genuinely proud of and this map would never have been finished.

## Comparison is the thief of joy:

I started making CTM areas in 2012 but I have yet to release a map of any length. I started work on Harbinger in 2013. A big part of why it has taken me so long to be releasing a map is that I got far too hung up on how my map would compare to others.

To paraphrase Obi Wan: there will always be a bigger map. Wanting to make the grandest map out there isn't feasible for 99% of maps, especially on a first attempt. Focusing more on what I could do well and being more selective about what lessons to learn from looking at other maps has made mapping a lot more enjoyable for me.

Mapping is a hobby, hobbies are meant to be fun. If anything takes away from that, then you should stop it. It's sounds simple, but it will improve the quality of what you produce as well.

The only way to improve is by doing a lot of it and by releasing it into the wild to be criticised. I don't expect Harbinger to be perfect, but I have finally reached the point where I'm fine with that.

Don't worry about whether your map is the best there has ever been. Just make something. Even Heliceo's first map wasn't RageCraft 3! *

## Getting closer:

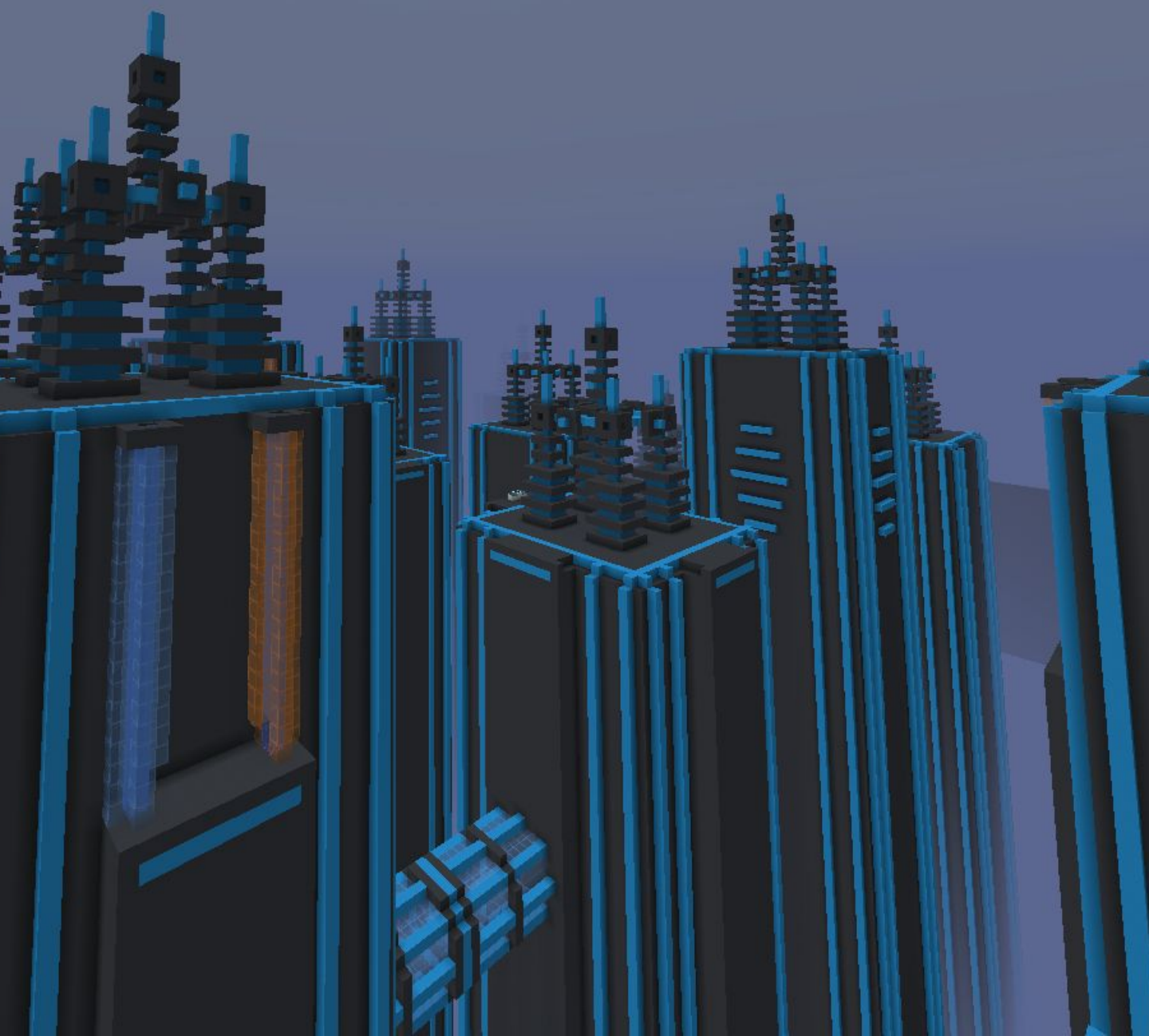
My first CTM map is very close to being done! Just being able to fly around in my map and look at it all is an amazing feeling. I would highly recommend getting into map making as it is one of the most rewarding gaming experiences I have ever had. I can't wait to see other people playing it!

Find me on Twitter, **@ErisMakesMaps** if you want to keep up with my progress!

**(Heliceo was the mapper behind the RageCraft series. RageCraft 3 is one of the most complete CTM experiences available even years after it initially released. This is a* **very** *bad joke.)*

# Designing Immersive Cutscenes in Minecraft Maps

By Cavinator1 (@_Cavinator1_)



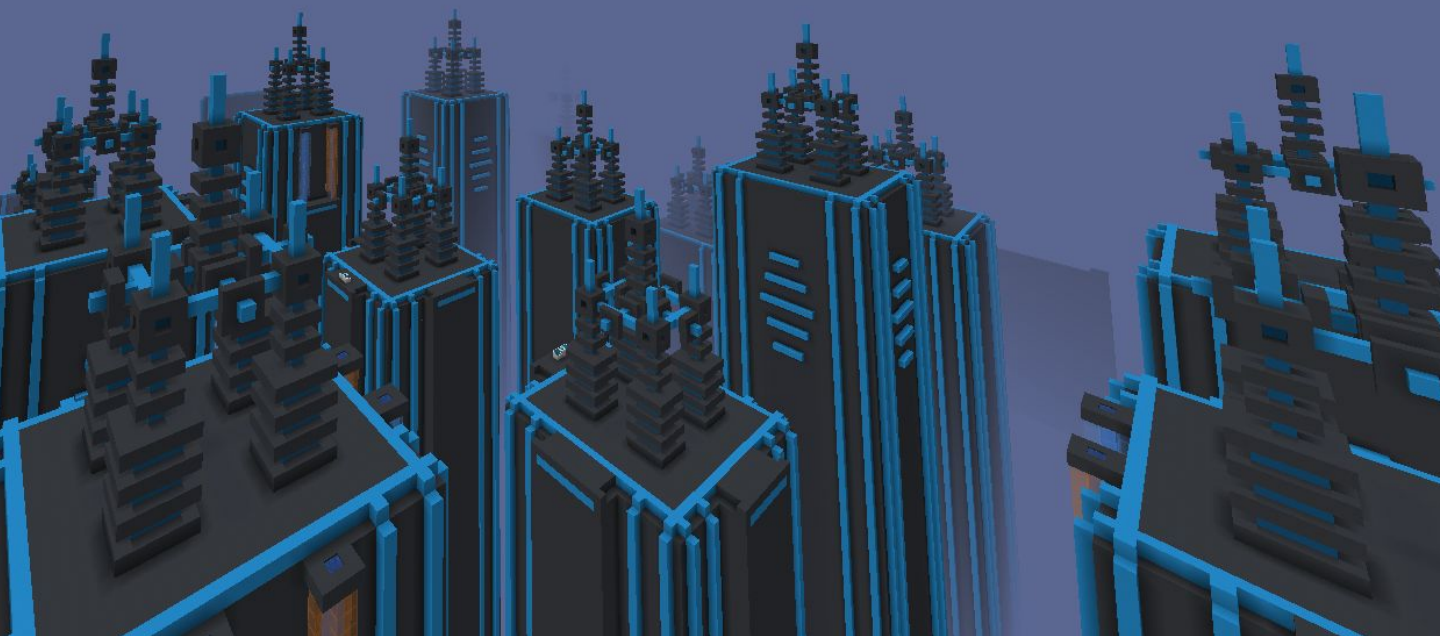
# Designing Immersive Cutscenes in Minecraft Maps

By Cavinator1 (@_Cavinator1_)

Minecraft maps are great. Many more advanced Minecraft maps, especially maps that feature a story, feature cutscenes - the player is often briefly put into spectator mode and their movement is controlled via a series of well-tuned command blocks or functions to make them move around, casting great, cinematic pans and sweeps across the map.

Many Minecraft maps have used cutscenes, some in a more advanced manner than others. Guardians of the Nether, Siegebreaker, Metroid Bounty Hunter, A Hole New World, Exodus Season 3, The Unseen Forces III, Assassin of Steve 3 (a lot of “threequels” here) and additionally my upcoming sci-fi adventure map, Celestial Protocol, has a very hefty use of cutscenes.

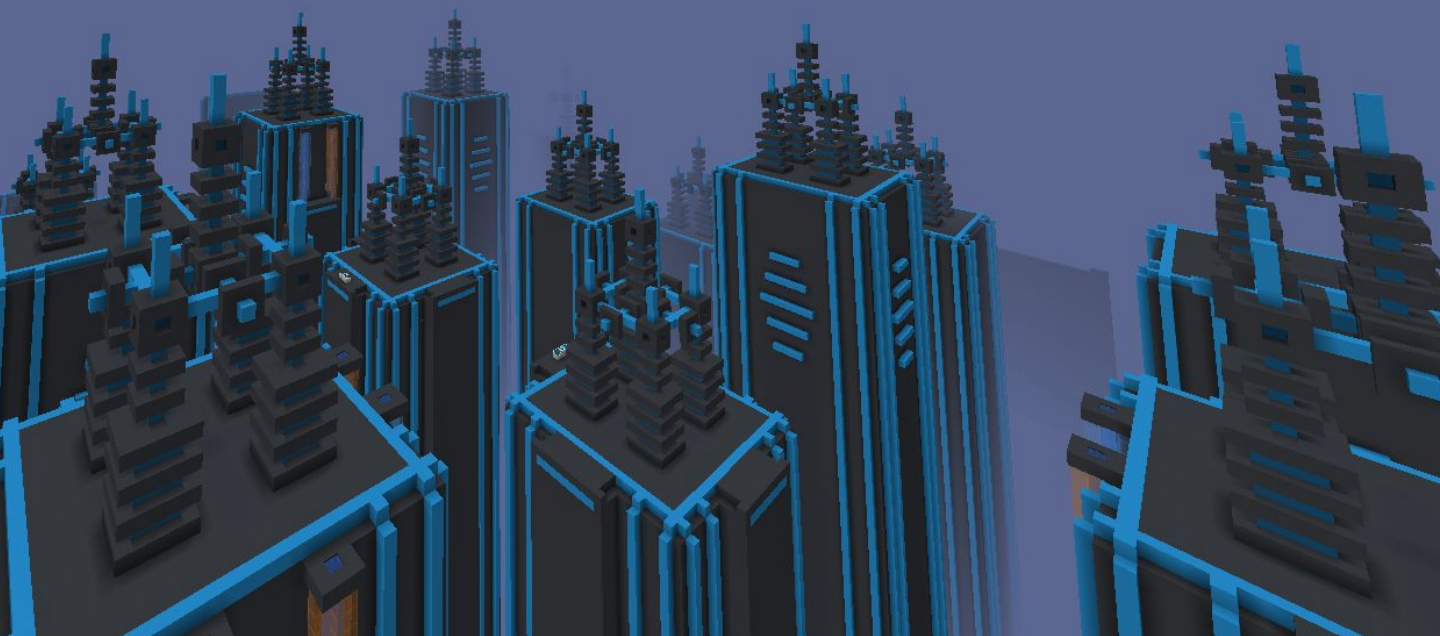
This article will spend a little bit of time on how to implement cutscenes using commands, in modern versions of Minecraft Java Edition, but the ability to implement cutscenes is only the first step. Designing immersive cinematics is an art, with numerous components in order to make them interesting and really stick out and keep your players’ attentions, especially when pairing them with conversations. A cutscene done badly might as well not have been done at all.



# How to Implement Cutscenes

Implementing cutscenes using commands can be hard to learn (and harder to master) but once you get the hang of them, they are extremely worthwhile to include in many maps. Before 1.13's intensive command overhaul, in wide use was TheRedstoneScientist's Cutscene Generator and it's more advanced version CutscenePRO. Sadly, these tools have not been updated for current versions of Minecraft, but it is still possible and surprisingly simple to create your own cutscenes using commands.

The method I am using to implement them in Celestial Protocol is inspired by how CutscenePRO worked, but updated and optimized for use with functions and 1.13's new command system.
(Note: This guide will assume you already have a comprehensive understanding of functions, datapacks, and commands. If you don't know exactly what something does, go look it up :P )
It is easiest to implement cutscenes using functions, and normally you will only need three function files. We'll give them the general names "begin", "cutscene" and "end".



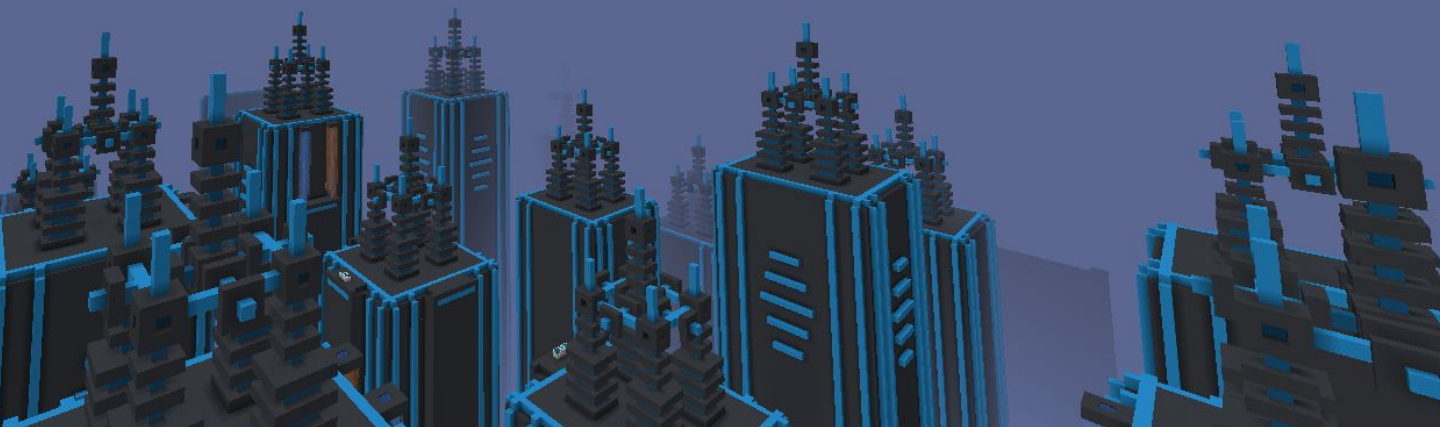
The “begin” function is run when you want to begin the cutscene, and it should contain these commands:

```
# Summon cutscene AEC
/summon minecraft:area_effect_cloud -1628 84 -1097
{Rotation:[7.7f,8.8f],Age:0,WaitTime:30000,NoGravity:1,Invulnerable
:1,Particle:"dust 0 0 0 0",Tags:["c1"]}

# Initialise scoreboard and gamemode
/scoreboard players set @a pan 0
/scoreboard players set @e[type=area_effect_cloud,tag=c1] pan 0
/gamemode spectator @a

# Start cutscene
/data merge block 0 64 0 {auto:1}
```

The first command summons an area effect cloud, which will be teleported around by the “cutscene” function in the intended movement.
The important thing to note is that the area effect cloud must be summoned at the coordinates where you want the cutscene to begin and with the starting rotation as well.

The final command will turn on a repeating command block that contains the following command, making the “cutscene” function run once every tick. This command block should be placed either in the spawn chunks, a forceloaded chunk, or in a location which will be loaded for the entire cutscene.
/execute as @e[type=area_effect_cloud,tag=c1] at @s run function <”cutscene” function>

You will need one repeating command block for every cutscene in your map, so if you have more than one cutscene in your map be sure to always change the coordinates in the /data merge command. Additionally, the area_effect_cloud should also have use a different tag on each cutscene (e.g. tag “c2” for cutscene #2, and so on).



The “cutscene” function is run every tick by the repeating command block.

Each tick it increases the score “pan” by 1, which is essentially a timer score. Depending on what value the score is up to, every tick the area effect cloud is teleported in a certain direction along a camera path a tiny bit (and the player along with it) in order to create the effect of a moving camera.

*You might ask, why not just teleport the player? Because the player can still move and look around in spectator mode, and you don’t want that to happen because it can cause the player to ruin the cutscene. Having the player be teleported to the area effect cloud will lock the player’s movement and the direction they are viewing.*

A “cutscene” function with two camera paths will look like this:

```
# Begin: -1628 84 -1097 7.7 8.8
tp @s[scores={pan=0}] -1628 84 -1097 7.7 8.8

# Point 1: -1589 84 -1086 29.6 10.6 after 5 seconds
tp @s[scores={pan=1..99}] ~0.3939393939 ~ ~0.1111111111
~0.2212121212 ~0.0181818182
tp @s[scores={pan=100}] -1589 84 -1086 29.6 10.6

# Point 2: -1543 84 -1046 37.8 5.8 after 5 seconds
tp @s[scores={pan=101..199}] ~0.4646464646 ~ ~0.4040404040
~0.0828282828 ~-0.0484848485
tp @s[scores={pan=200}] -1543 84 -1046 37.8 5.8

# Update player position to AEC
tp @a @s

# Add 1 to scoreboard for next iteration
scoreboard players add @s pan 1
scoreboard players add @a pan 1

# End cutscene
execute if entity @s[scores={pan=201}] run function <”end”
function>
```

Let's look at one camera path up close:

```
# Point 1: -1589 84 -1086 29.6 10.6 after 5 seconds
tp @s[scores={pan=1..99}] ~0.3939393939 ~ ~0.1111111111
~0.2212121212 ~0.0181818182
tp @s[scores={pan=100}] -1589 84 -1086 29.6 10.6
```

Notice the camera path runs for 5 seconds (100 ticks), teleporting the area effect cloud and player in tiny increments while the score pan is between 1 and 100.

You can easily extend and adapt the function for your own needs by changing the coordinates and the length of each camera path, as well as copying the commands for one camera path to create more camera paths for a cutscene.

Obtaining the exact values that you want the player to be teleported each tick can be tricky business. There are 20 ticks per second and ideally for a smooth cutscene you'd want to plan out the exact coordinates you want the player to move between in a cutscene, which will lead to you having to do a LOT of math to calculate how many blocks per tick the player should be teleported. CutscenePRO did all this automatically, but we don't have CutscenePRO anymore. But fear not, I have a solution!

During the entirety of Celestial Protocol's development (so far) I have been using my own spreadsheet which allows you to enter the 'starting' coordinates and the 'ending' coordinates of a camera path in order to calculate these values. You can also enter the number of seconds that the camera path should last.
*Do note that it calculates the values for 1 tick less than the number of seconds you enter (e.g. if you type in 5 seconds, it calculates for 99 ticks, not 100). This is so on the 100th tick of the cutscene, have it run a command for teleporting the player to the 'ending' coordinates of the path for readability and to keep consistency.*

I have created a blank copy which you can access **here**. Note that it is set to read-only, so you will need to make your own copy of the spreadsheet in order to use it.

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | | x | y | z | dx | dy |
| 2 | Start point | -1628 | 84 | -1097 | 7.7 | 8.8 |
| 3 | End point | -1589 | 84 | -1086 | 29.6 | 10.6 |
| 4 | | | | | | |
| 5 | Difference | 0.3939393939 | 0.0000000000 | 0.1111111111 | 0.2212121212 | 0.0181818182 |
| 6 | (Relative coordinates per tick) | | | | | |
| | | | | | | time in secs |
| | | | | | | 5 |

You can see these values are used for Point 1 above

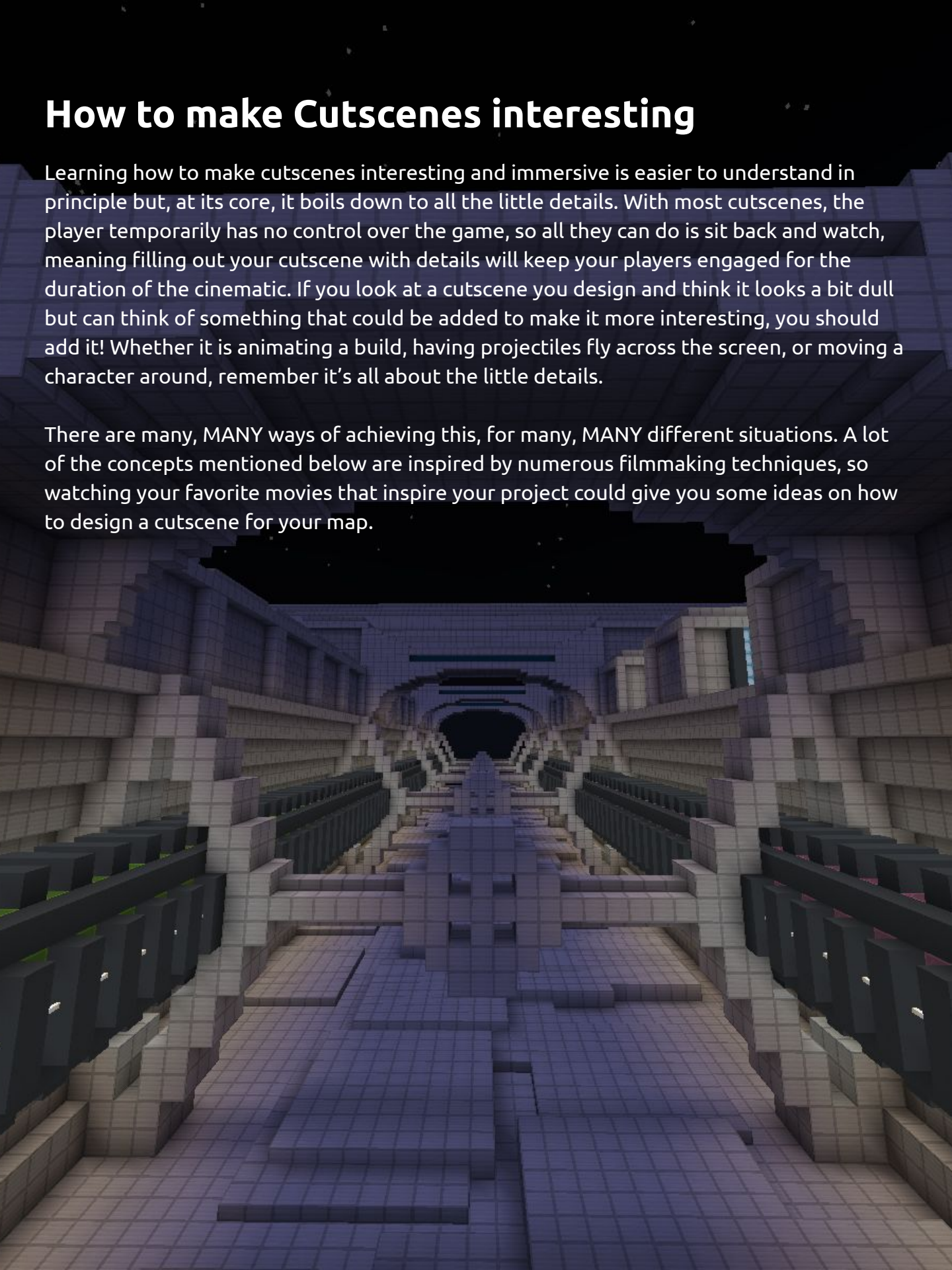
Finally, the “end” function should simply contain the following commands, which will stop the “cutscene” function from running, kill the area effect cloud, put the player back into adventure mode, and teleport them back into play so they don’t just drop from the sky (a lot of cutscenes will take you into the sky).

```
# Kill AEC and stop cutscene running
kill @e[type=area_effect_cloud,tag=c1]
data merge block 0 64 0 {auto:0}

# Put player into next position
scoreboard players set @a pan -1
tp @a -1628 64 -1107 0 0
gamemode adventure @a
```

# How to make Cutscenes interesting

Learning how to make cutscenes interesting and immersive is easier to understand in principle but, at its core, it boils down to all the little details. With most cutscenes, the player temporarily has no control over the game, so all they can do is sit back and watch, meaning filling out your cutscene with details will keep your players engaged for the duration of the cinematic. If you look at a cutscene you design and think it looks a bit dull but can think of something that could be added to make it more interesting, you should add it! Whether it is animating a build, having projectiles fly across the screen, or moving a character around, remember it's all about the little details.

There are many, MANY ways of achieving this, for many, MANY different situations. A lot of the concepts mentioned below are inspired by numerous filmmaking techniques, so watching your favorite movies that inspire your project could give you some ideas on how to design a cutscene for your map.

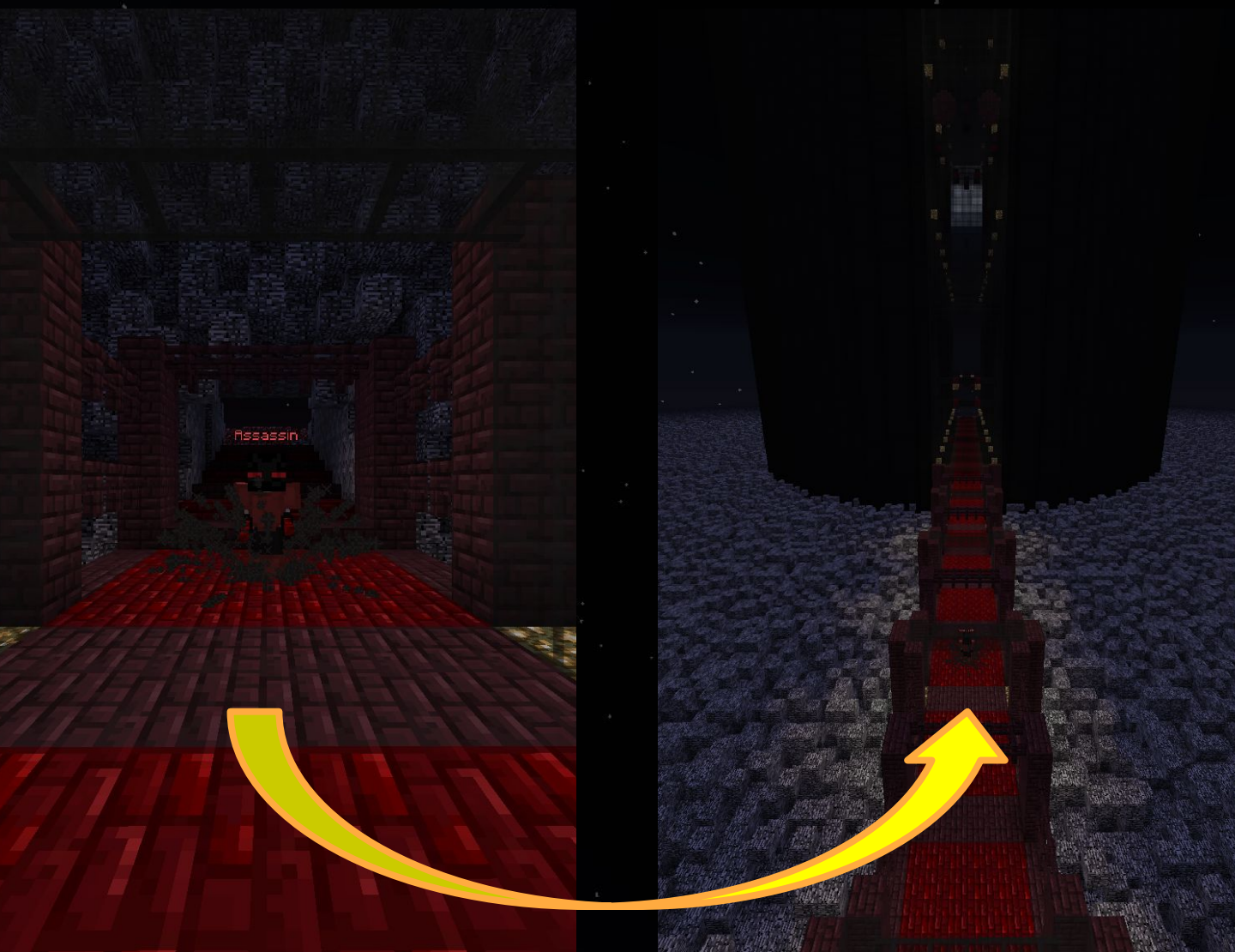


# Establishing Shots

Establishing Shots are simple. Are you travelling to or entering a new area of the map? A simple establishing shot where a cutscene pans over the area can help the player get a sense of the location they are about to enter.

An example of an establishing shot is from the very first cutscene in my map Assassin of Steve 3, which features the villain, the Assassin, teleport into the player's view. He then teleports forward, behind the camera, so the camera turns around to reveal a long bridge that he is teleporting down in numerous steps, then the camera pans up, away from the bridge, to reveal the bridge leads to an enormous dark tower.

There are hundreds of webpages out there that describe basic cinematography techniques (which most people boil down to camera angles)



# Details

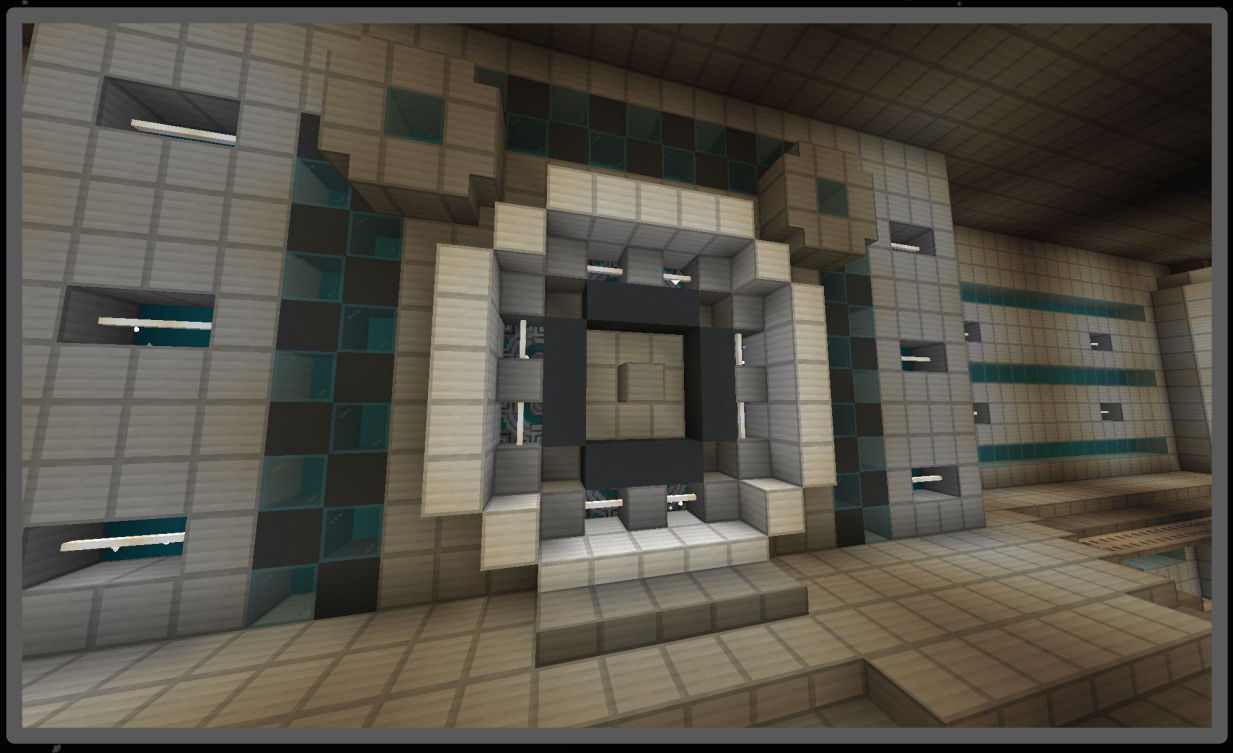
Cutscenes are not just dramatic views of your builds. Well, sometimes they can be, but to make cutscenes really stand out, it's important for something to happen while viewing them.

It is relatively easy to get any command to run at a specific point in time during a cutscene. In the main "cutscene" function, simply insert your command, preceded by the following:

```
execute if entity @s[scores={pan=<time in ticks>}] run <a command>
```

Using this you can make anything happen while the cutscene is taking place, whether it be:

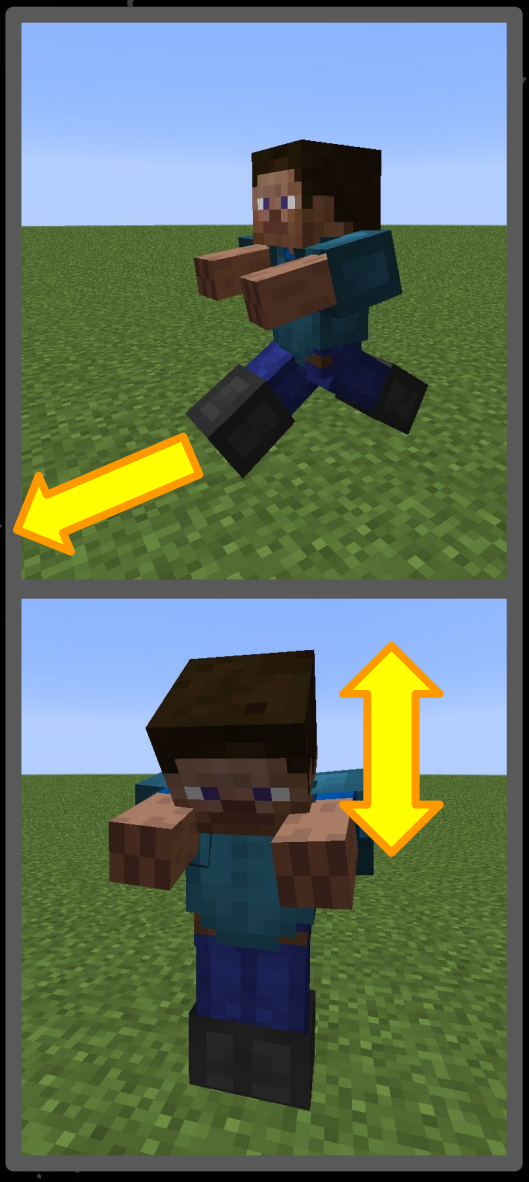
- Convey dialogue, exposition or other text in the chat, titles or actionbar. Text can be combined with what the cutscene is looking at to guide the player.
- Play sounds using the /playsound command.
- Place or remove blocks to create animated builds. Can be utilised in many ways, for example an animated sliding door that opens by having one layer of blocks disappear at a time.
- Display particle effects.
- Animate mobs and entities (more detail on the next two pages).



# Characters:

Many Minecraft maps feature different characters that you will talk to, and during a long cutscene it is important to keep characters dynamic and interesting or else the player will get bored just staring into their face. Here are a few ways of doing this:

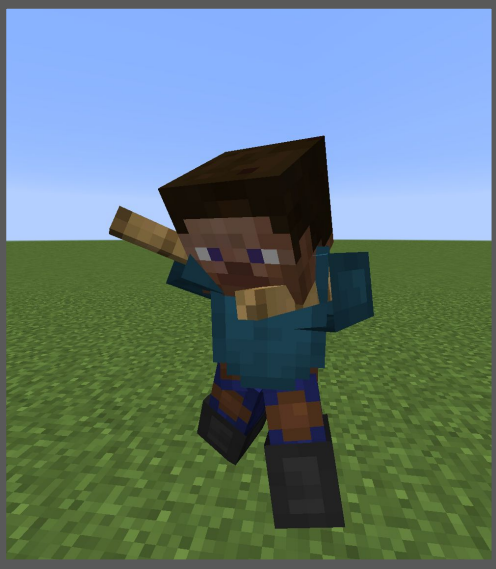
**Movement and actions:**
Having the character move around and doing something can make characters feel more dynamic. The method for creating cutscenes described above can be adapted for any other mob or entity that you use for a character, and you can use this to great effect by having the character move around.
You can also use the /replaceitem command to enable characters to pick up or drop items or change clothes (e.g. putting on armor before a fight)



**Head movements:**
Having a character move their head around during the cutscene (by using a /tp command that changes their rotation) can greatly make a character feel more alive, and some applications of this technique are:

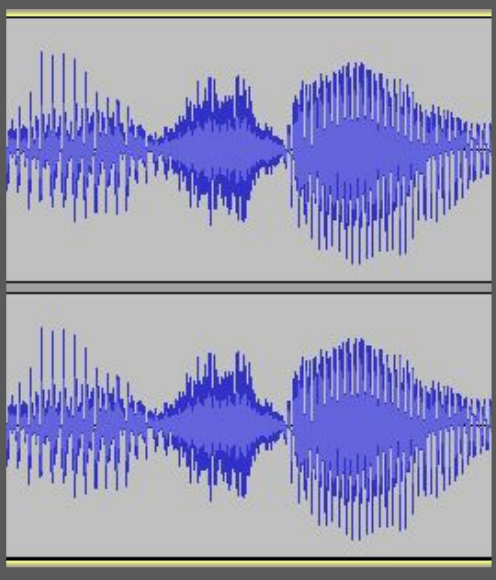
- Nodding or shaking their head.
- Looking around an area.
- If they are talking specifically to or about a different character or object they might look at them.

**Body movements:**
Body movements are more for use when the character is a custom armor stand, since armor stands allow endless customisation of the pose of arms and legs. You could have them wave their hand or kick out a leg or something. It can be difficult to make movements look fluid and realistic, but the end result is well worth it.



**Facial expressions:**
If you're using a custom resource pack to give characters custom heads, different facial expressions can additionally make the character more dynamic.

**Voice acting:**
This is a topic that deserves an article of its own but doing voice acting for characters can make them far more dynamic and interesting than they ever can be when just talking to you using /tellraw.



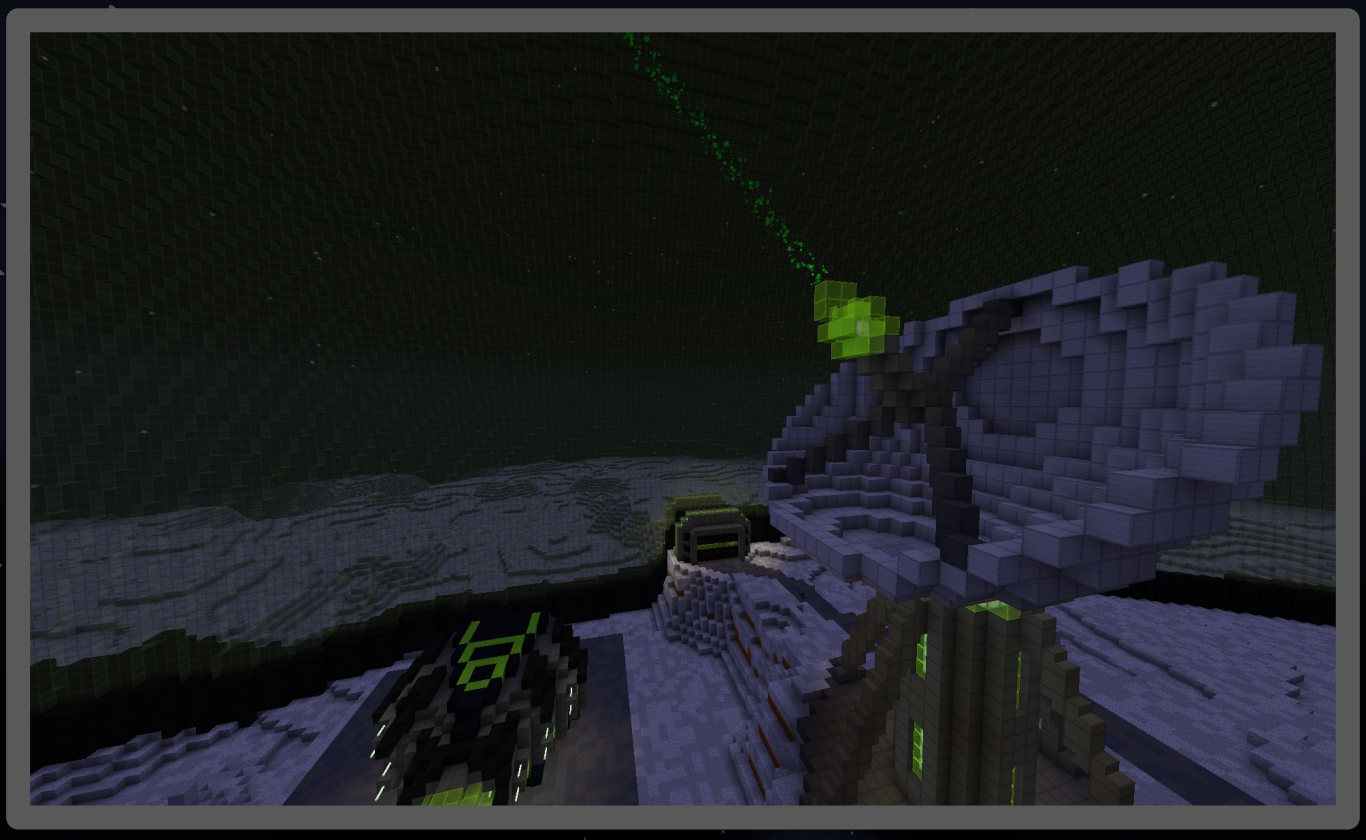
# Show, don’t tell:

Do you have an evil villain giving a monologue about his (or her) evil plan? Don’t just force the player to sit there and wait for the evil villain to finish said monologue. Make the villain’s evil plan more interesting by having a cutscene show how their evil plan works!

The concept of show, don’t tell is a common term in filmmaking and in Minecraft maps use of this concept works exceedingly well given the enormous creative potential Minecraft has for creating visuals.

For the evil villain example, suppose they have an evil death ray in outer space that will zap a ray straight down Mount Everest and cause the entire planet to explode upon hitting the Earth’s core. Have the cutscene show the evil death ray fire down to Earth, through Mount Everest, and then seeing the planet explode not only makes the cutscene more interesting but it also makes the stakes feel more real.

This concept is in no way limited to evil villains with death rays. There is an endless amount of applications this could be used for, like flashbacks or stories about events in the past.



# Conclusion:

My article barely scratches the surface on what is possible. Minecraft is a creative game with endless possibilities, after all.

Possibly the best way to design cutscenes is not to come up with a story around the cutscenes, but to come up with cutscenes around the story. To achieve this, it's best to write the story for your map and plan out roughly how everything will go and what cutscenes you want before you start building it. It doesn't have to be 100% complete and bursting with details, as long as you get a general idea for what you want throughout the map.

Then, when it's time to implement certain sections of your story into the map, then you can plan out exactly how each cutscene will go. It's okay to add extra details such as character head movements after a cutscene is implemented - some concepts such as these are easy to add but go a long way.

*Cavinator1 is a self-proclaimed professional command block raider and is best known for the Assassin of Steve trilogy and his advancements datapack. Many of the screenshots used in this article are of his upcoming project Celestial Protocol, an ambitious sci-fi adventure map that has been in development since the beginning of 2018. If you like what you see feel free to follow on Twitter, where sometimes he'll post more developmental screenshots and videos.*

# VEHICLES

**By: Popular YouTube ( @TubePopular )**

This article will discuss how to build your very own vehicles in your world! I'm going to break it down to 3 simple steps:

1. The Base
2. The Build
3. Fancying it up

# STEP 1: Base

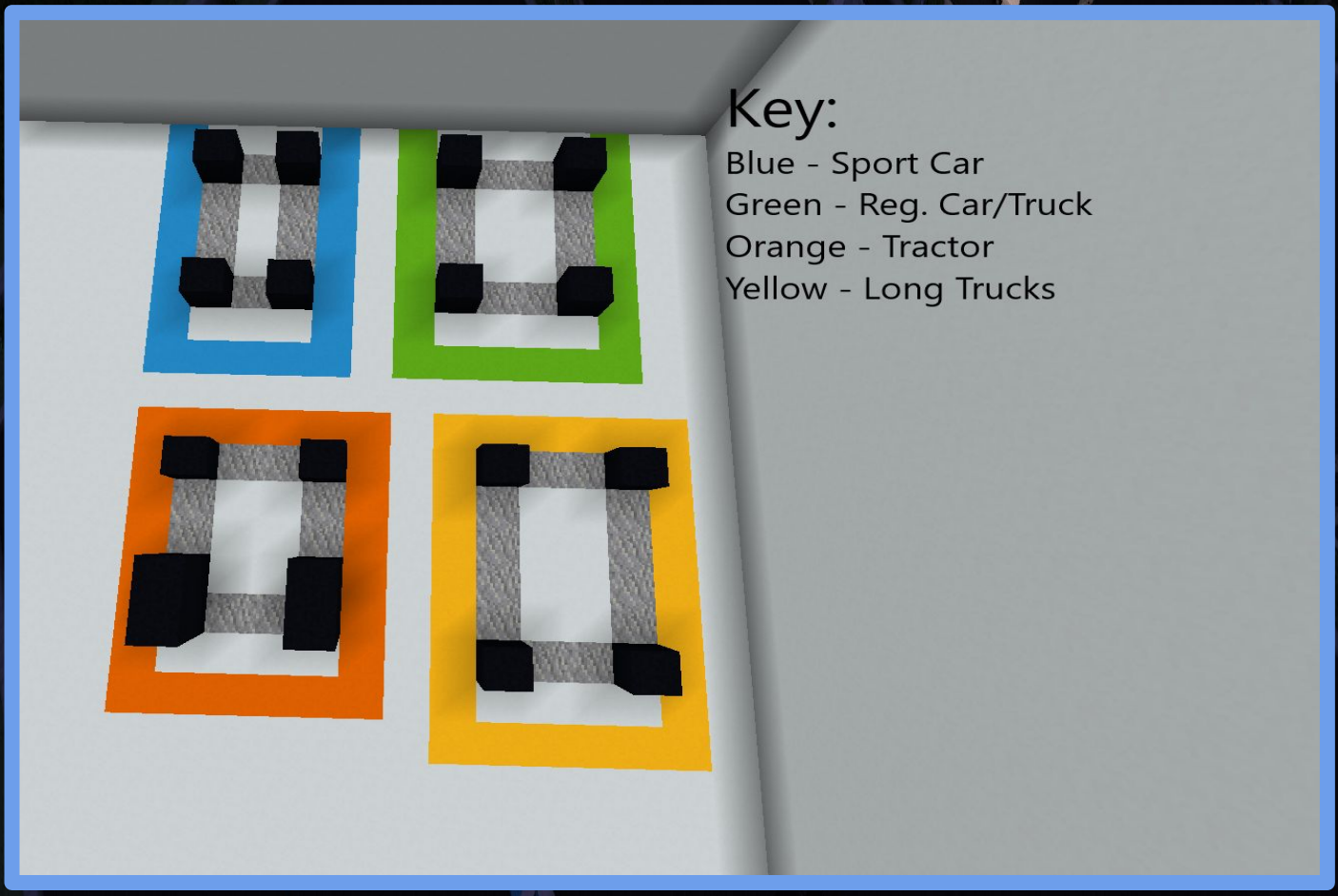
Wheels comes in all different shapes and sizes depending on what type of vehicle it is. Here is  a basic diagram, showing you possible wheel dimensions for different vehicles.

Sport Car- 5 by 3

Reg. Car/Truck - 5 by 4

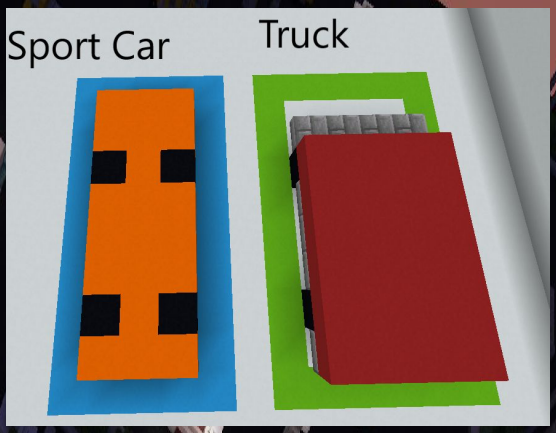
Tractor - 6 by 4 (Last two wheels are 2 high & 2 wide)

Long Trucks - 7 by 4



# STEP 2: The Build

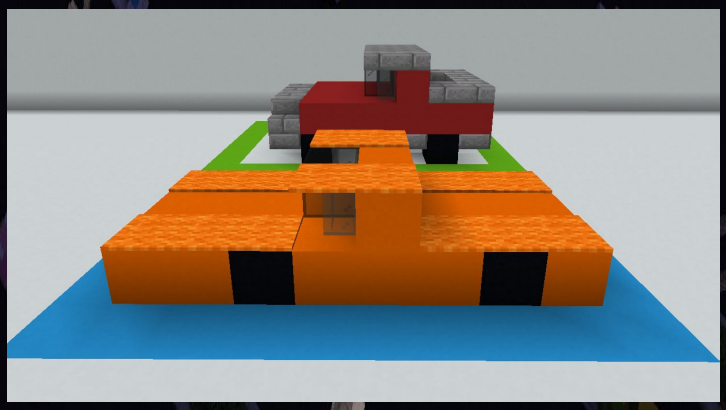
Vehicles are unique so the builds are going to be unique, so to keep it simple, I'm going to show you the basic build of a sport car and truck.




**Sport car** - Fill in the inside of the car with a choice of your favorite color concrete. (Concrete gives the best choice for building a sports car due to its vibrant color) Fill in a two block area before the front wheels and 1 block long after the last wheels.

**Truck** - Fill in the outer layer with slabs around the car and fill in the top part of the truck excluding the front of the truck with concrete of your choice.

Now let's build the top of the car.



**Sport car** - On the 2nd block after the first wheel build a row to the other side of the car on the top, then place black stain glass in the front center and two black panes on both side of the glass.Add orange carpet going down the sides of the car.

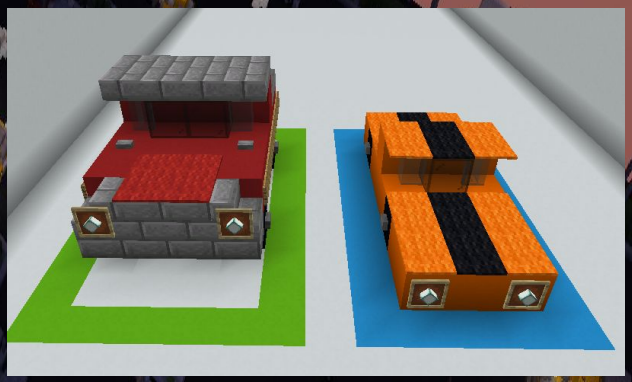
**Truck** - Place two stone brick stairs on the each corner of the front of the car where you have the space and then place two stone bricks in the center. On the 3rd block after the first wheel, build a row to the other side of the car on the top, then place black stain glass in the front center and two black panes. Cover the roof with stone brick slabs and the around the back of the the truck to make the trunk.

# Step 3: Fancying It Up!

This step is where the fun happens!

**Sport Car** - Place stone buttons on all 4 wheels. Now time for some sport strips! Place any color wool carpet down the center of the car. Now onto the headlights and tail lights. Place 2 item frames on the front/back of the car, for the headlights place sea lanterns inside of the item frame and for the tail lights place redstone. In Between the two tail lights place a sign and write some random numbers for the licenses plate.

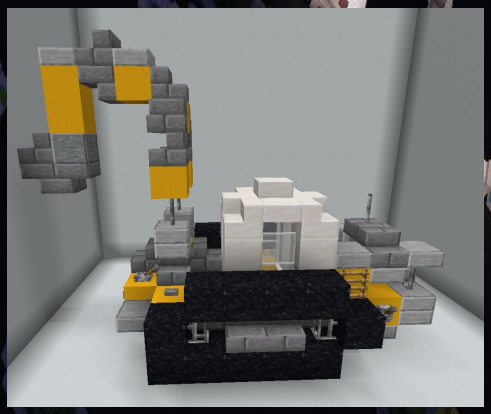
**Truck** - Place stone buttons on all 4 wheels, like the car. Then on the exterior of the truck place signs following all the way down to where the  tail lights will go. Place 2 items frames on the front/back of the truck. for the headlights again you can use sea lanterns and for the tail lights you can use redstone. Now to the front of the truck, place four red carpets on the front part of the truck going in 2x2 blocks. Then you can add some additional buttons to the front of the truck for details.



# CONCLUSION

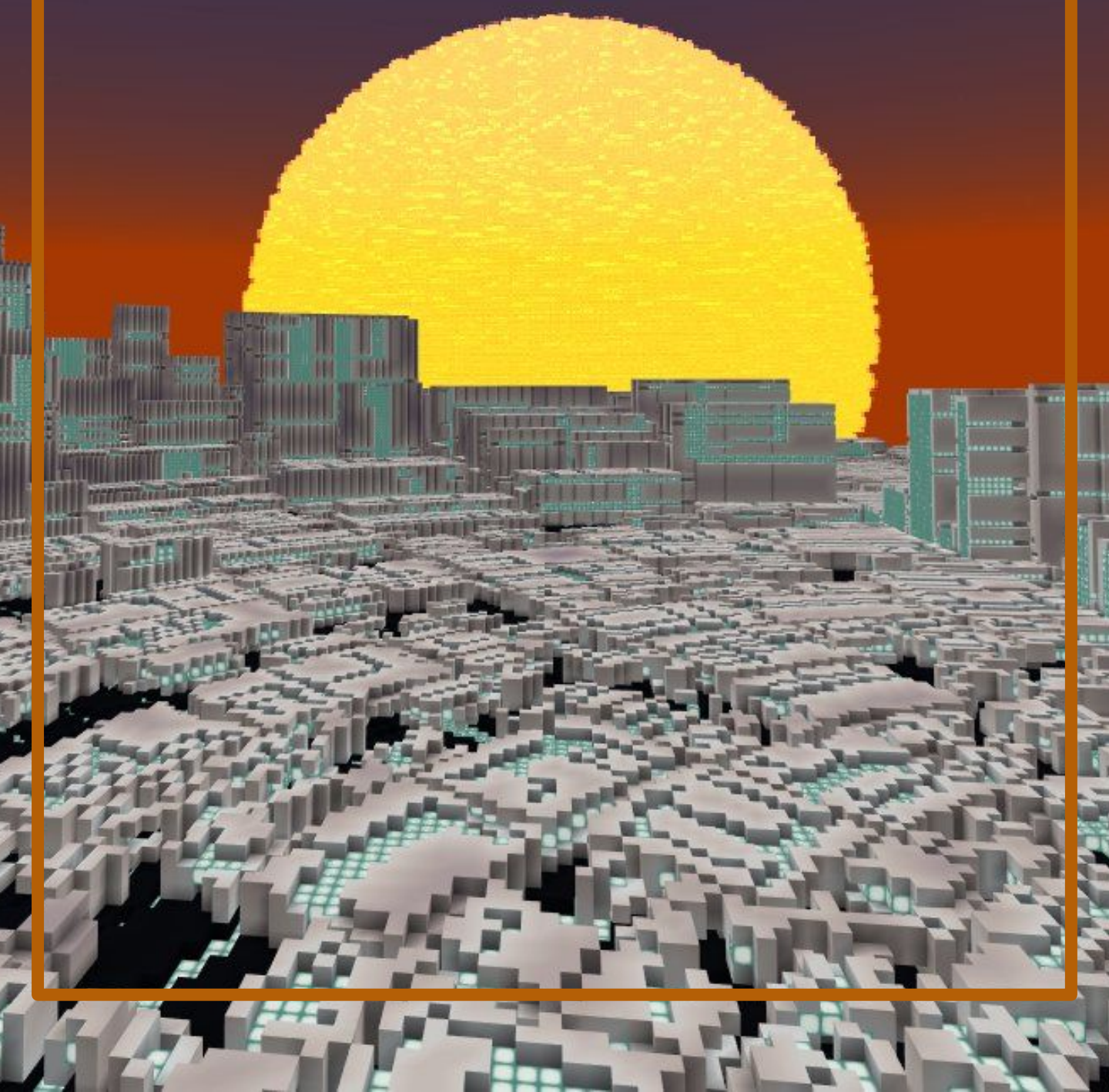
Of course this is just only a snippet of what is possible when creating vehicles. These are very simple versions and you can go into way deeper detailing the vehicles to really make them realistic. For example: Using fireplaces or cobwebs as smoke coming out of tractors, using more stairs and adding angles in your car, or using colored heads as extra details. In conclusion the possibilities are endless when creating vehicles!

Here are some examples of detailed vehicles:



# MAKING FLY SPY

The exciting new science fiction flying game for Bedrock.



# MAKING FLY SPY

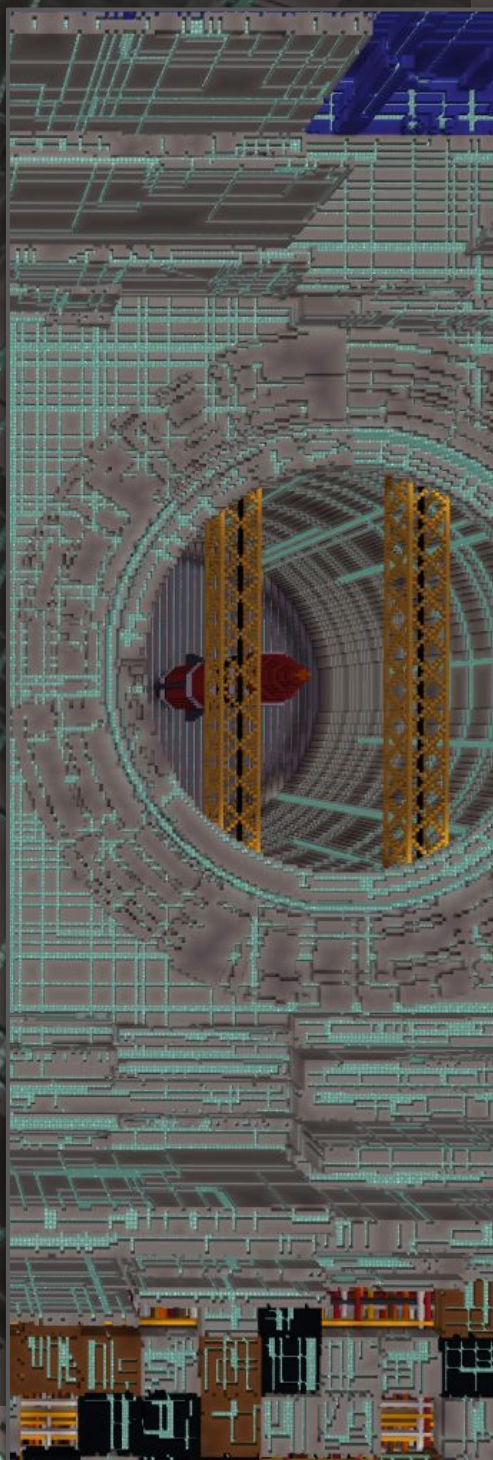
An epic scale sci-fi gliding map development log, from start to finish
By @TheWorldFoundry

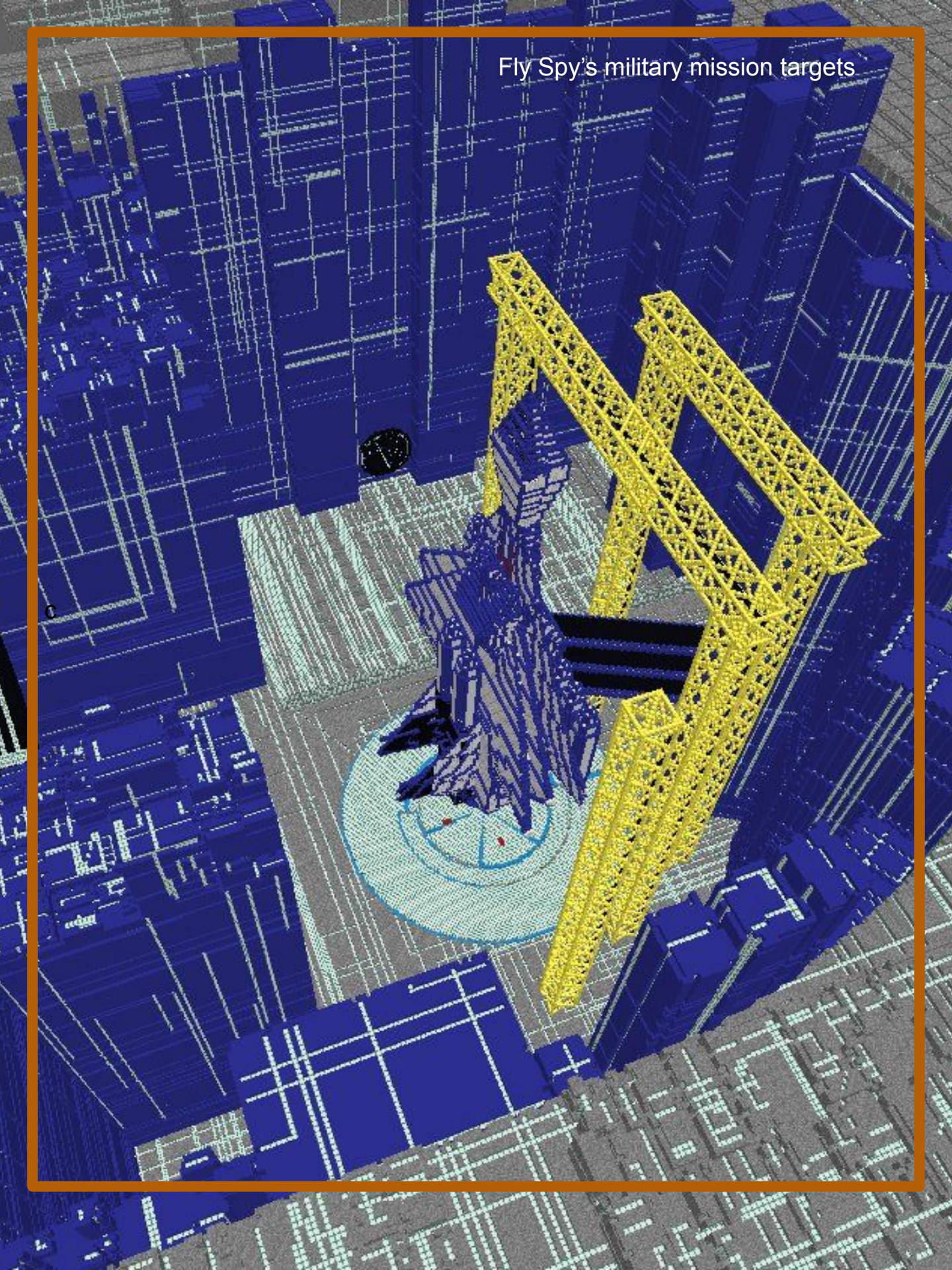
Fly Spy puts the player in a futuristic universe where people are at war with machines. The player is on a mission to explore the enemy world and catalog military sites ahead of a planned assault. The player has the power of flight in low gravity to traverse an expansive alien 3D landscape.

This article describes the process of creating this world, from environmental build through to realising the game mechanics and mission objectives. If you are looking for ways in which you can approach (and finish) your own projects then this article is for you.

The Fly Spy map was in active development from October 2019 through to January 2020. It is a Marketplace product, and so is a commercial project. You do not need to buy the map to benefit from this article.

In Fly Spy, the player has been inserted into an enemy world as an infiltrator. The player starts in a secret makeshift spawn base where the mission briefing and objectives are communicated. From here, the player is led through a safe tutorial section that introduces the Minecraft elytra gliding mechanic. From here, the player can explore a large open-world complex of tunnels and surface features where they acquire mission tags by flying close to areas of interest. Once all the tags are acquired, the player must return to base to win the game. After the game is won, the human assault begins and the player can guide explosive charges to destroy the base.




Fly Spy's military mission targets

# Building a Massive Play Zone

The world of Fly Spy is one that has not been designed for humans. It is an alien landscape of machine outcroppings whose purpose can only be guessed. The surface of this world is intended to disorient and spark the player's curiosity: who made this place, and what was their purpose in doing so?

The backstory is that the world has been made by machines. It is an uncomfortable disordered place. It is the sort of place that might have been built from a machine learning algorithm that has decided that function is more important than aesthetics.

It looks truly alien.

The art style of Fly Spy is inspired by the illuminated world of the Tron movies, as well as the detailed greebling of the Star Wars universe (both franchises are property of Disney). Minecraft has a distinctive blocky style that lines up features on a grid, allowing for clean lines for grid-aligned voxel structures. The decision was taken to fill the world with rectangular prisms to take advantage of the rendering engine supplied by Minecraft.

The glowing effect is achieved using Sea Lantern blocks. To generate structures, the MCEdit Unified tool was used along with a custom script that creates the objects.

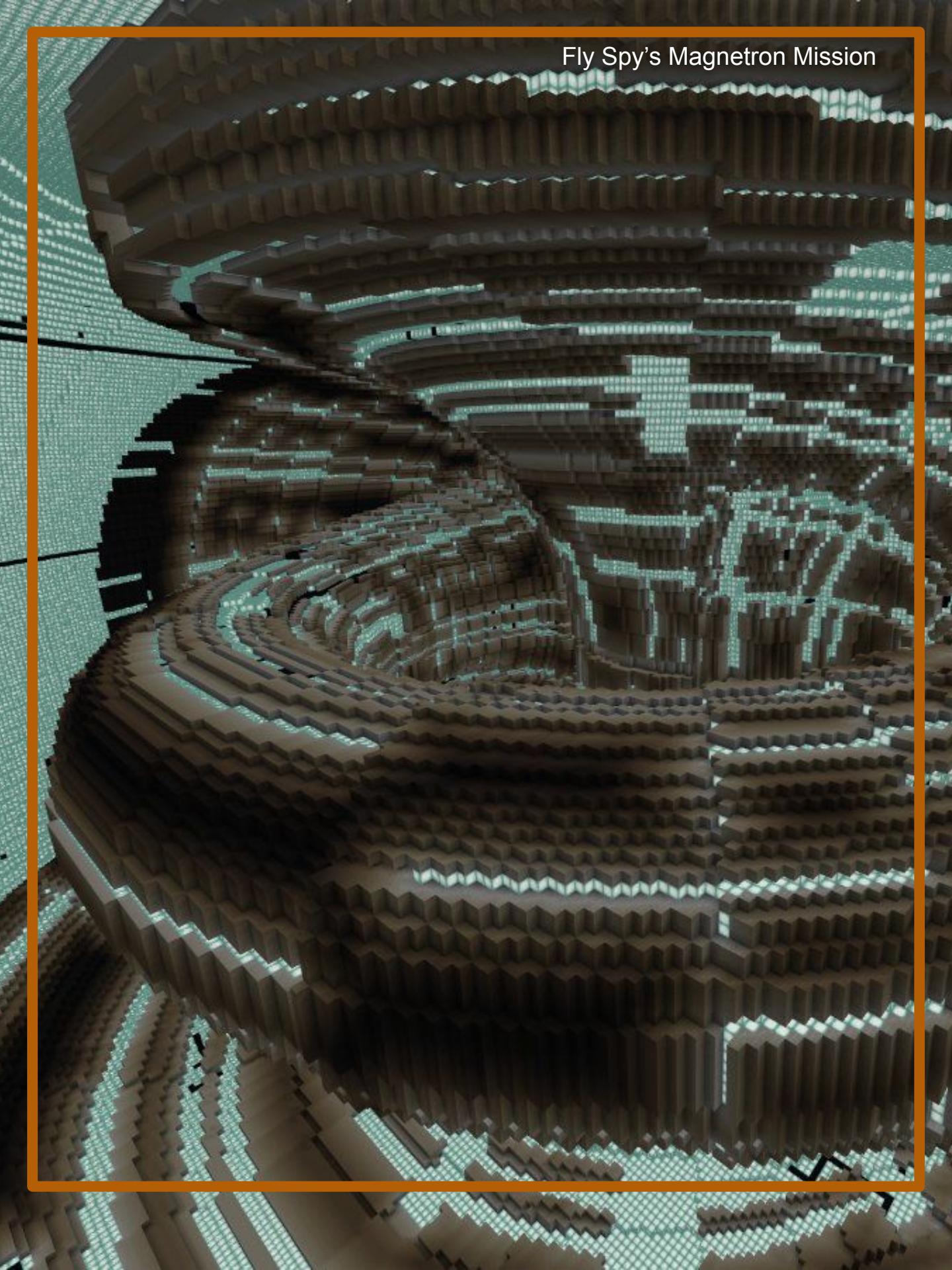
# Environment Hacks

Clouds in regular Minecraft appear as vaporous blocks in the sky at level 128. Where present, they intersect with tall structures and cause visual confusion. The default clouds have been removed from the world by replacing the clouds texture image file with a transparent image. This means that players do not need to fiddle with settings before playing and enjoying the game as the author intended.

To ensure consistent lighting on the surface of the world, the time has been locked at dusk. This also has the effect of bringing the Minecraft sun close to the horizon. Time has been stopped to ensure the player experiences the world the same way all the time.

The sun is always in the extreme distance. Having the sun close to the horizon is a good idea for gameplay reasons because it provides a reference point for player navigation in what can be a disorienting and confusing landscape.

The default Minecraft sun can be swapped out by a larger and more interesting sprite by replacing the texture file using a custom resource pack. Minecraft changes the colour slightly because of the way it handles sunrise and sunset, so when you use this method yourself you may find unexpected (but welcome) effects when you check your work in-game. In Fly Spy, an in-game sun was created using lava blocks then has been screen-shot and used as the basis of the enormous star texture you see at the horizon in-game.


Fly Spy’s Magnetron Mission

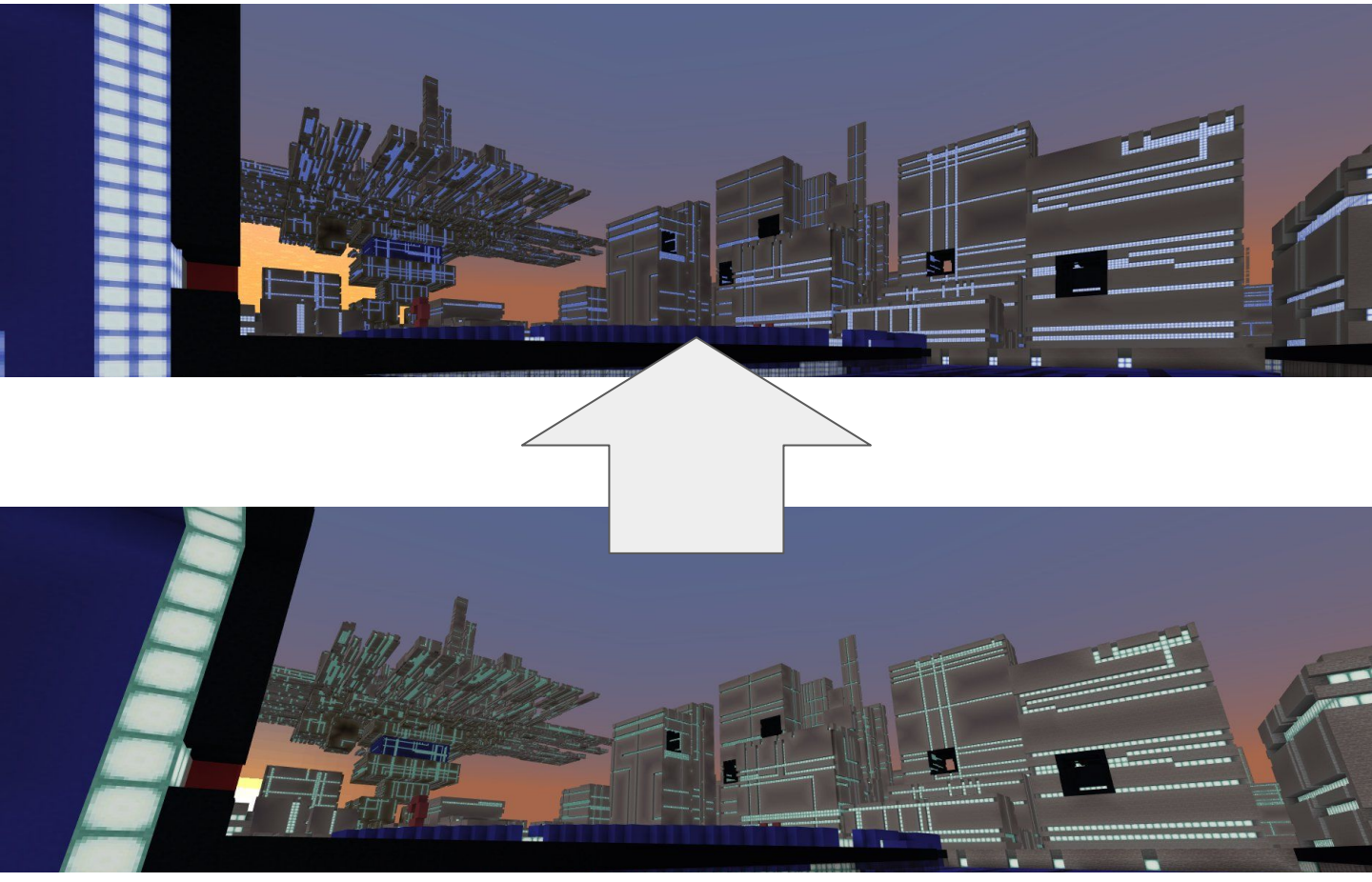
# Textures and Skinning

The appearance of player wings in Minecraft can be changed by modifying the pixels in the Elytra texture. The goal here was to make the player look like there is a machine strapped to their back.

The sea lantern sheen is normally green, and by shifting the hue slightly blue the luminous glow of lights looks less organic.

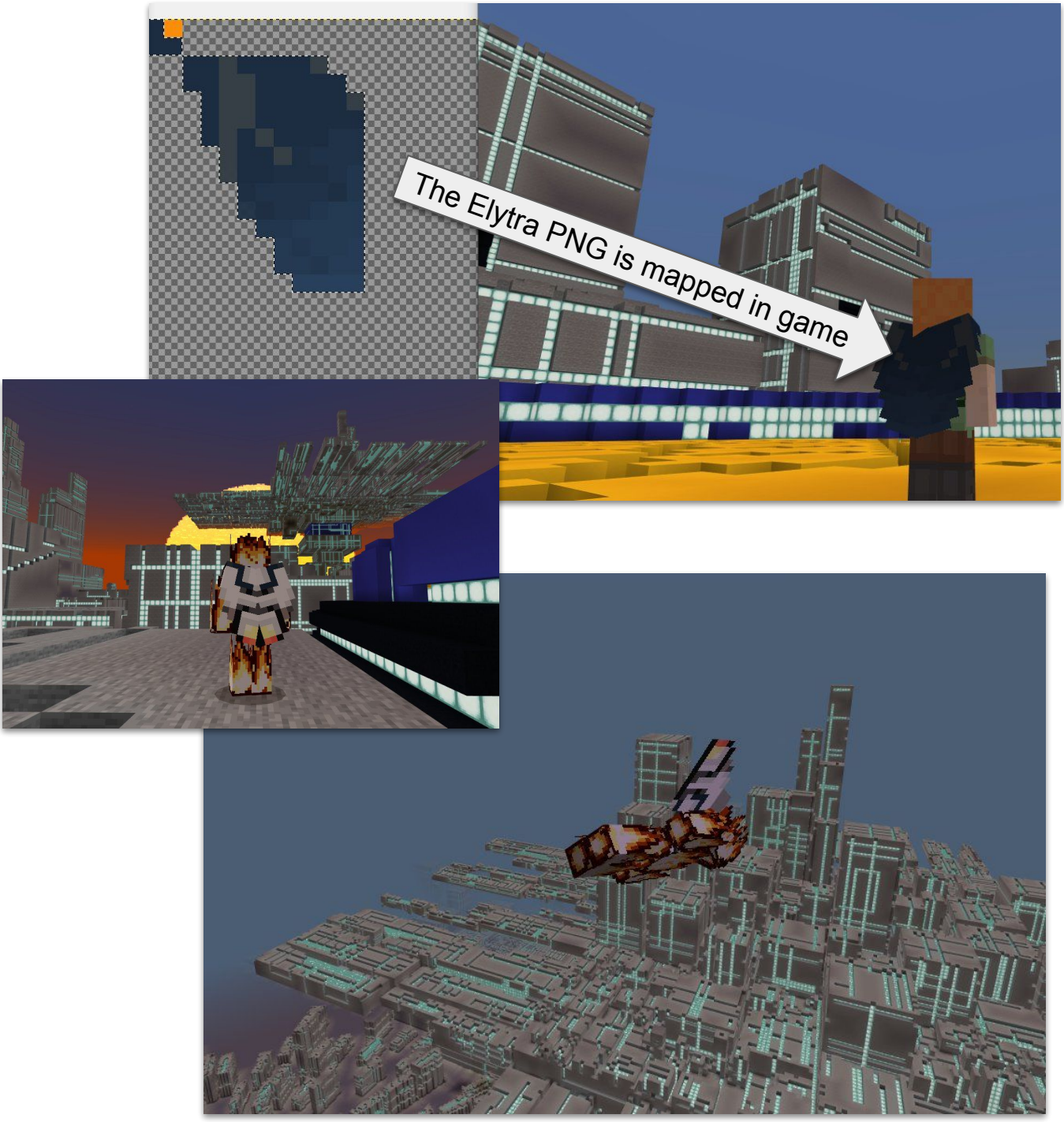
A player skin is provided with the map to encourage playing the game in third person view. The skin is in 128x128 pixel template format and modelled on a combined flight-suit / space-suit / spy theme.

*Flight suit and wings*



*Changing sea lantern hue*

# Elytra Texture

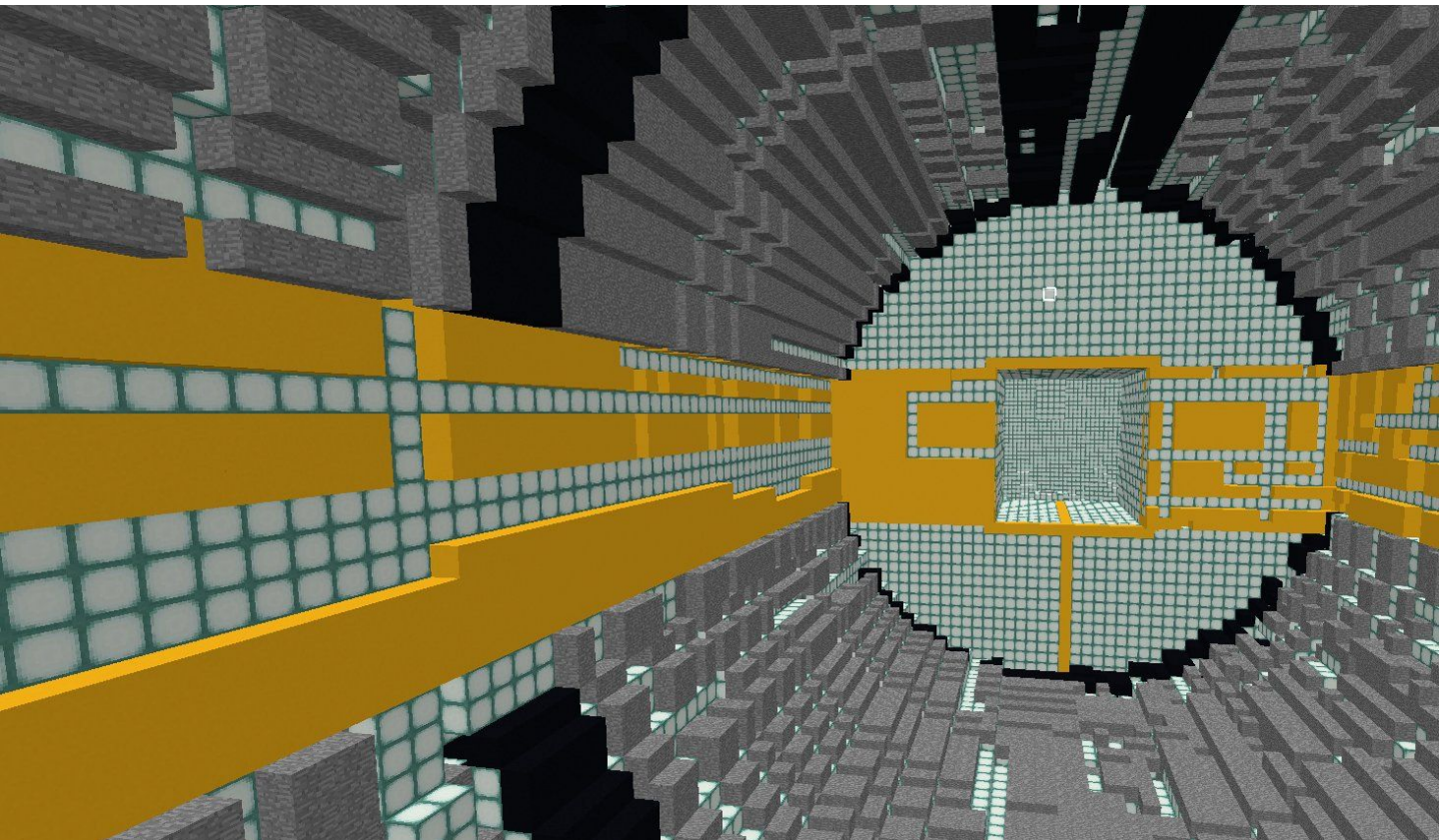


# Mission Mechanics

As the player explores the world and discovers mission objectives their progress is tracked on a wall at the base. The wall is regenerated as the player moves into the area and cleared as they move out. Small in-game pictures are shown to provide clues as to what the player should look for.

Completed missions are shown by a transparent block being placed over the relevant icon. This is done by checking the tags that are on the player. Tags are set when the player is in close proximity to a target and then checked at the base.

When the player is at the base, the game frequently checks to see if all the tags have been collected and moves onto the end-game phase if so. This consists of a game-win message, and then initiates custom explosives targeted at the player location, intended to demolish the machine world (and eventually the player!)



# Sounds and Speech

Fly Spy starts with a short musical flare and from there the player is assisted in their exploration of the world by messages from a human handler, talking over the flight suit communication system.

Custom sounds in Minecraft are configured using .json file entries. The audio data file is packaged into a resource pack, and the relative path in the file system is used to identify it. This is given a namespace identifier in the resource pack sound files, and appropriate metadata hints are added to tell Minecraft how to play the file ingame. The sounds can then be played by command blocks on events, like when the player finds a new mission objective.

Sounds that Minecraft bundles can be replaced using the same method. In Fly Spy the floatin machine assault devices issue ominous clicks and deep bass sounding resonating growls by swapping the default sounds.

Fly Spy uses voice acting to convey information and reward the player with backstory and guidance.

By using the narrative of a communication link, the audio format can be lower in file size than would otherwise be required which allows more audio to be included on lower-end devices. Audio has to fit in memory to work reliably, otherwise it needs to be managed as a stream by the game engine.

Custom in-game music is also included in Fly Spy to provide a backdrop for the action.

# Device and Platform Differences

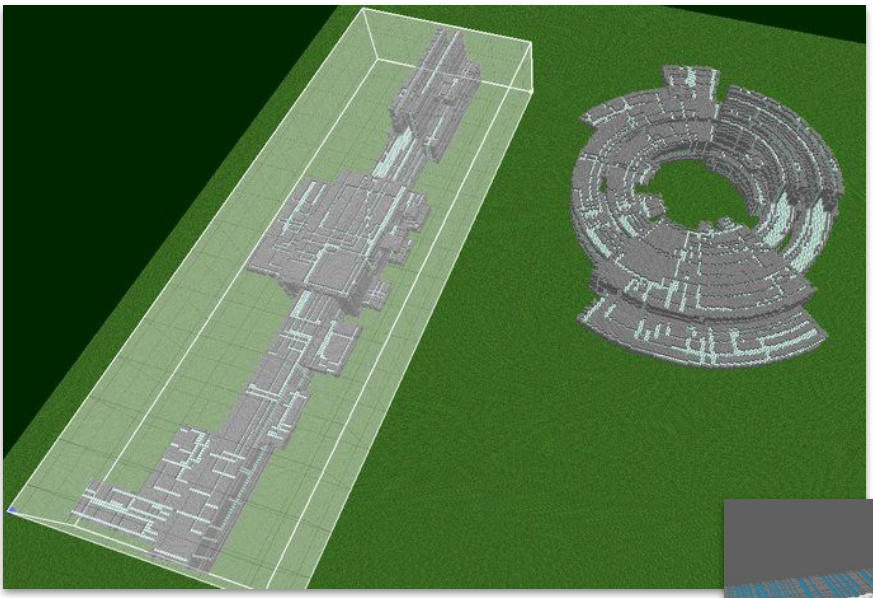
The bedrock engine restricts the chunk draw distance on mobile and lower power GPU devices. This means that the visible area around the player can be greatly limited by the choices players make about which device they will play on today. Realms and server hosted worlds also have a feature that reduces the number of chunks that are sent to a client, regardless of how powerful the client really is. Fly Spy is playable on mobile, but asks the player to stay close to the surface, walls, and ceiling features to navigate more easily.



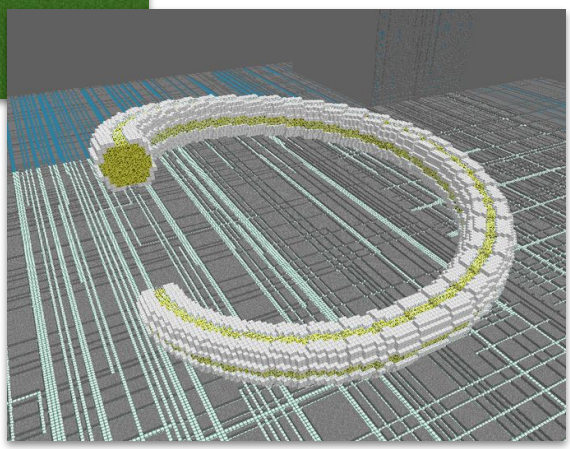
For this reason, gameplay may be improved by placing directional clues for the player to interpret and use in navigating your world on many devices. Helping players find their way through your map is a level design discipline which can be developed through experience.

Strategies include navigating your level from the end back to the start to identify and remove frustrating areas, as well as placing ‘bread crumbs’ for the player to find and follow. Lighting choices can also help focus attention and draw a player in an intended direction. In Fly Spy, red pipes provide a guide for the player to follow from the start area.

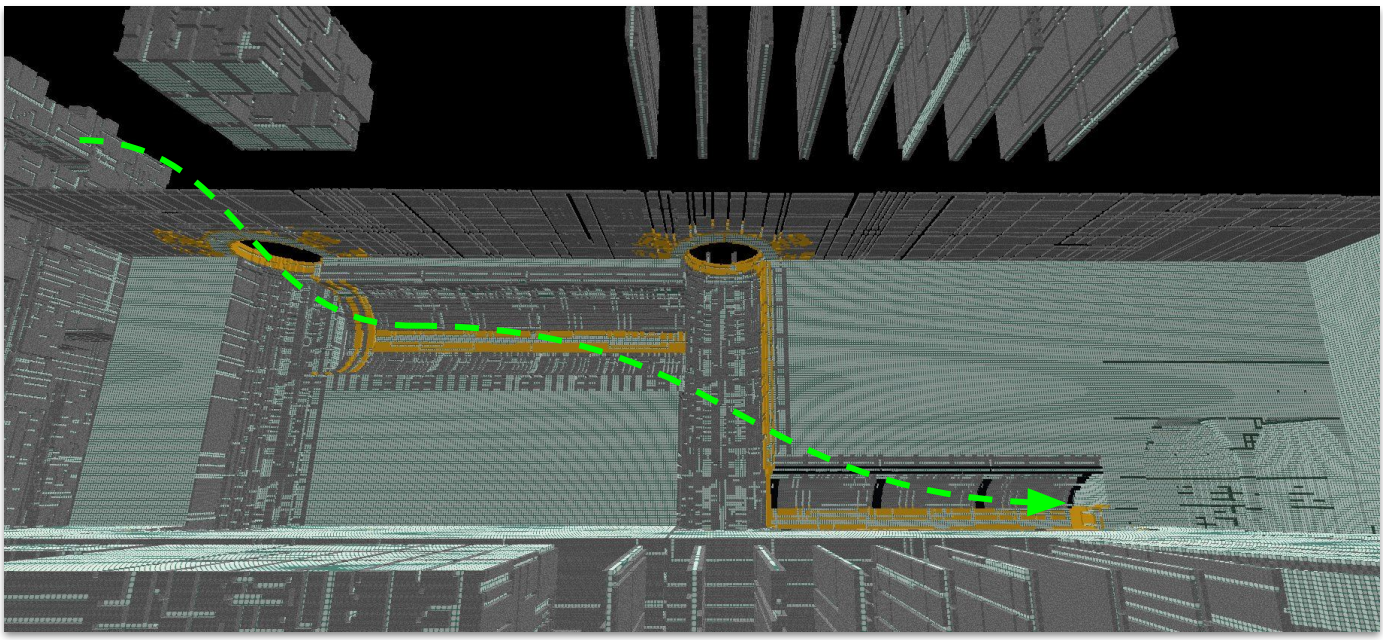
# Level Design



The mission objectives are logically the same as a classic inventory. As the player explores the world they ‘collect’ locations. With this approach, more sophisticated missions can be created that involve combining locations visited. The audio system uses this to provide hints in certain areas if the player hasn’t managed to find the mission entrance!



Levels designed for Elytra gliding must be created so that the highest glide ratio is no greater than 9.5 blocks horizontal to each 1 block down. Because this is part of the basic game, we cannot design a course that has a requirement on the player to travel, say 15 blocks across and only one downward. They cannot do it without rocket assistance. As a result, all of the mission courses in Fly Spy are designed with this glide ratio in mind. You can see the way this works in this level cutaway below:



# Fly Away!

Gliding over the surface of a mysterious planet-sized machine and then diving into twisted pipe-filled tunnels is heaps of fun. It draws on tropes from popular science fiction block-buster movies, games, and childhood dreams.

I had a lot of fun creating Fly Spy. It is the sort of project that I had been planning for many years before the technology and game engine were ready. Like most projects I would have packed many more ideas into this game if I had unlimited time to spend making games. Living in the real world though, I think I stopped at the right time and polished this to a high standard of creativity and fun.

Fly Spy has many secrets. One of them is that I have avoided many of the more dynamic systems in the Minecraft game engine. These include the custom entity and block modelling capabilities. Many maps these days are making use of the ability to create new creatures and vehicles for their stories. For me though, I steer clear of technology that is still evolving. The clear lesson for me is that it is very easy for all your hard work to be broken with the next release of Minecraft. If you have time to fix things that Mojang breaks that's not such a big problem. Otherwise, I stick to block textures, entity textures, sound and in-game mechanics for most of my work.

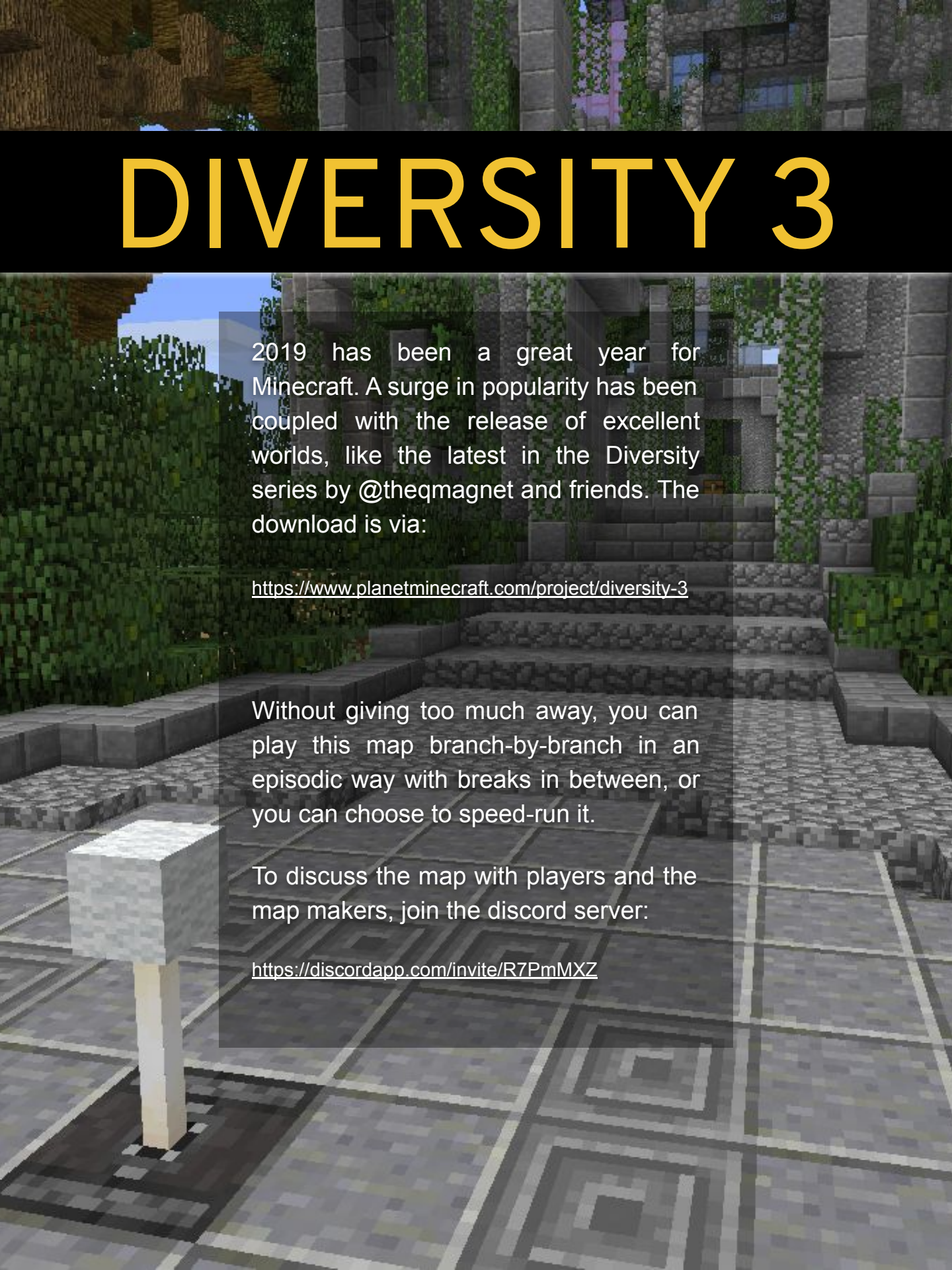
I hope this article has offered up something of interest to you as a player and map maker. If you would like to continue the conversation you can reach me on Twitter as @TheWorldFoundry or @abrightmoore.

The World Foundry

# TOTALITY

A Minecraft map by Cold Fusion

https://www.planetminecraft.com/project/totality/



# DIVERSITY 3

2019 has been a great year for Minecraft. A surge in popularity has been coupled with the release of excellent worlds, like the latest in the Diversity series by @theqmagnet and friends. The download is via:

https://www.planetminecraft.com/project/diversity-3

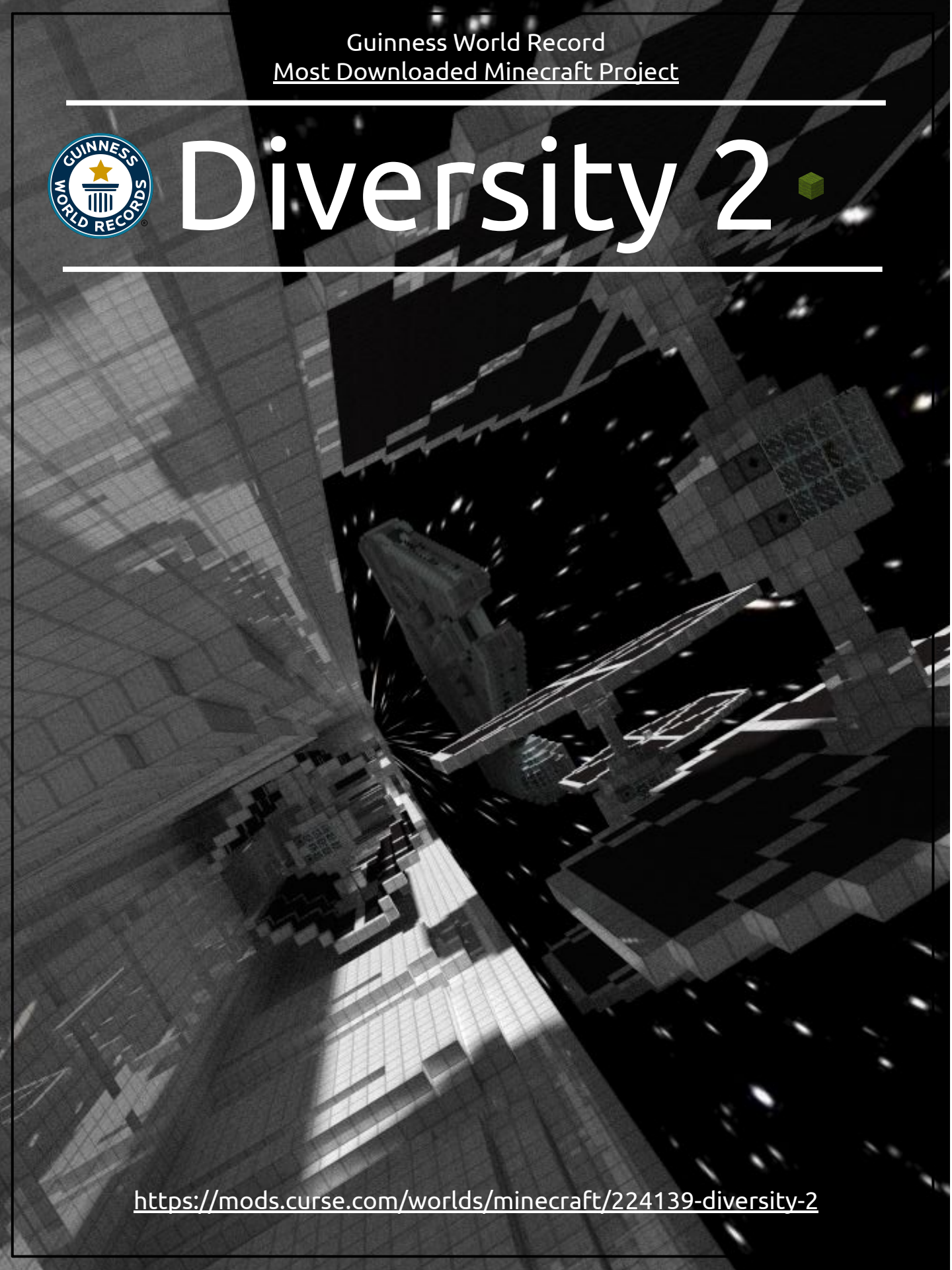
Without giving too much away, you can play this map branch-by-branch in an episodic way with breaks in between, or you can choose to speed-run it.

To discuss the map with players and the map makers, join the discord server:

https://discordapp.com/invite/R7PmMXZ

Guinness World Record
Most Downloaded Minecraft Project

# Diversity 2



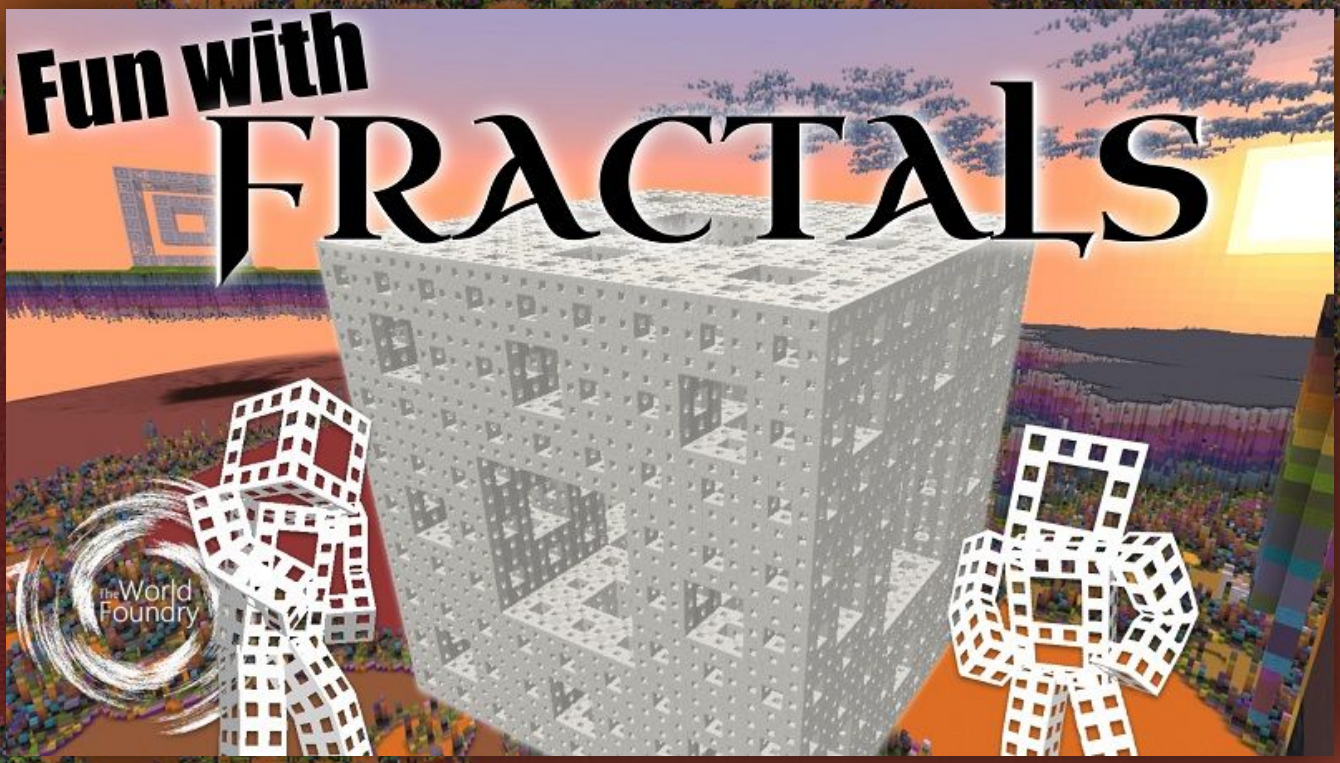
https://mods.curse.com/worlds/minecraft/224139-diversity-2

# MAPCADEMY

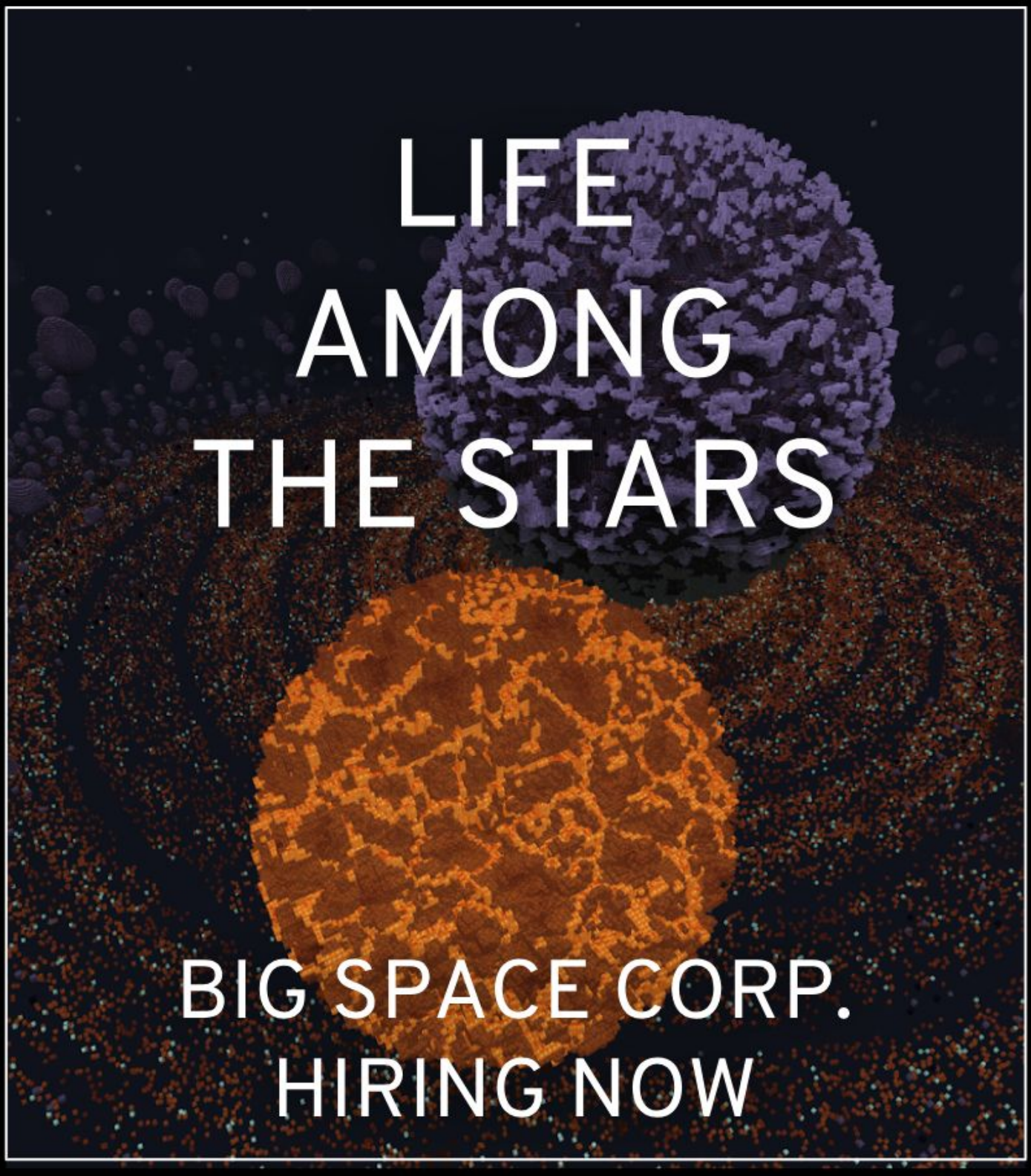
Get it on the Marketplace!
(Click here)

Congratulations! You have been successful in your application to the Minecraft map maker academy. Your first year studies will include:

- Dungeons
- Dropper
- Labyrinth
- Mob Spawning
- *Death!*




A Marketplace map




Space Station Z is available now on the Marketplace

The Free Independent Minecraft Map Maker's Magazine

# MAPMAG

## ISSUES 1-12


http://www.testfordev.com/MapMag

@MapMakingMag | MapMakingMag@gmail.com