The Free and Open Independent Minecraft Map Maker's Magazine

MAPMAG

ISSUE 8 - COMPLETE THE MONUMENT

@MapMakingMag

CONTENTS

THE LOBBY

Welcome back to the Minecraft Map Making Scene!

Welcome to the eighth issue of Map Making Mag. In this issue we spawn here in the lobby with the task of completing the monument!

Complete the Monument (CTM) maps are a gamestyle in Minecraft that challenges the player to collect blocks from around the world and return them to their resting place on a pedestal. The task can be made more difficult (and interesting!) through a variety of challenges the fiendish Map Maker has laid out to block the way.

We have littered WOOL BLOCKS throughout this issue for you to locate and use to complete your own Map Making Mag monument. You will find them in the articles and art as you navigate the world of Issue 8.

To research this issue's theme, your intrepid Editor was immersed in the CTM Community to gather ideas and understand the ways of CTM map makers and players. The community is vibrant and active, hosting many projects as well as themed events for you to join in.

Making a CTM Map is something you can start today. CTMs can be combat driven quests through ever more challenging levels, equipped with equipment with mysterious ancient lore. CTM can test the skills of a Map Maker, offering avenues for you to work with environmentals, level design, custom NBT, and modified mobs. Tweet your creation to @MapMakingMag.

The genre is open to innovation with many exciting maps released over the years that break the unspoken rules to create exciting hybrids for players of all styles. You may have already played a CTM without knowing it!

We haven't forgotten the inventors this issue, with a bumper crop of content from NCS:Computing that bridges the physical world into the virtual using the Micro:bit. Until next time...
Happy Map Making!

- Adrian Brightmoore, Editor
  Twitter: **@abrightmoore**

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.

SUPER HOSTILE

The Story of Vechs

Vechs is a legend among Minecraft players and map makers alike. Vechs defined the CTM genre by making games that play like normal Minecraft worlds, except “with a goal to complete”. His work since 2010 has inspired countless artists to recreate their own worlds filled with challenging monsters and prize wools.

Notorious for their combat difficulty and relentless spawning of powerful mobs, Vechs’ CTM maps have challenged players to come up with unique solutions to recover the wool prizes. Sometimes the best approach is to inch carefully forward to avoid becoming overwhelmed by mobs. Other times the timid are rewarded by unexpected swarms of monsters spawning from unexpectedly long ranges.

Vechs provides an extensive archive of map making tutorial streams on his YouTube channel, and can be found streaming live builds from time to time.

Recently Vechs has taken the Super Hostile experience online, with an MMO server hosting custom content created by the master himself.

Vechs’ maps can be downloaded from http://superhostile.mindcracklp.com/

Did you know…

Because Vechs started CTM before game commands were available, the player had to make sure they followed some rules to play the maps. These standard rules evolved over time, and are listed here in Vechs’ words:

1. Find and complete the Victory Monument.
2. Find wool in chests (Fleecy Boxes) or on special boss enemies (Fleecy Mobs) scattered around the map, do NOT use dyes or spider string for wools!
3. You may craft the Iron, Gold, and Diamond blocks.
4. Use at least Easy difficulty, and never Peaceful. Vechs’ maps are balanced around normal.
5. Do not leave the boundaries of the map and go into normally generated Minecraft terrain.
6. Survive in any way you can think of. (Yes, you can mine, craft, and place blocks.)You may use spider string wool and sheep wool to craft beds and other items, but not for the Victory Monument.
7. You may not pick up, craft, or move Ender Chests.

CRAFTSMAN

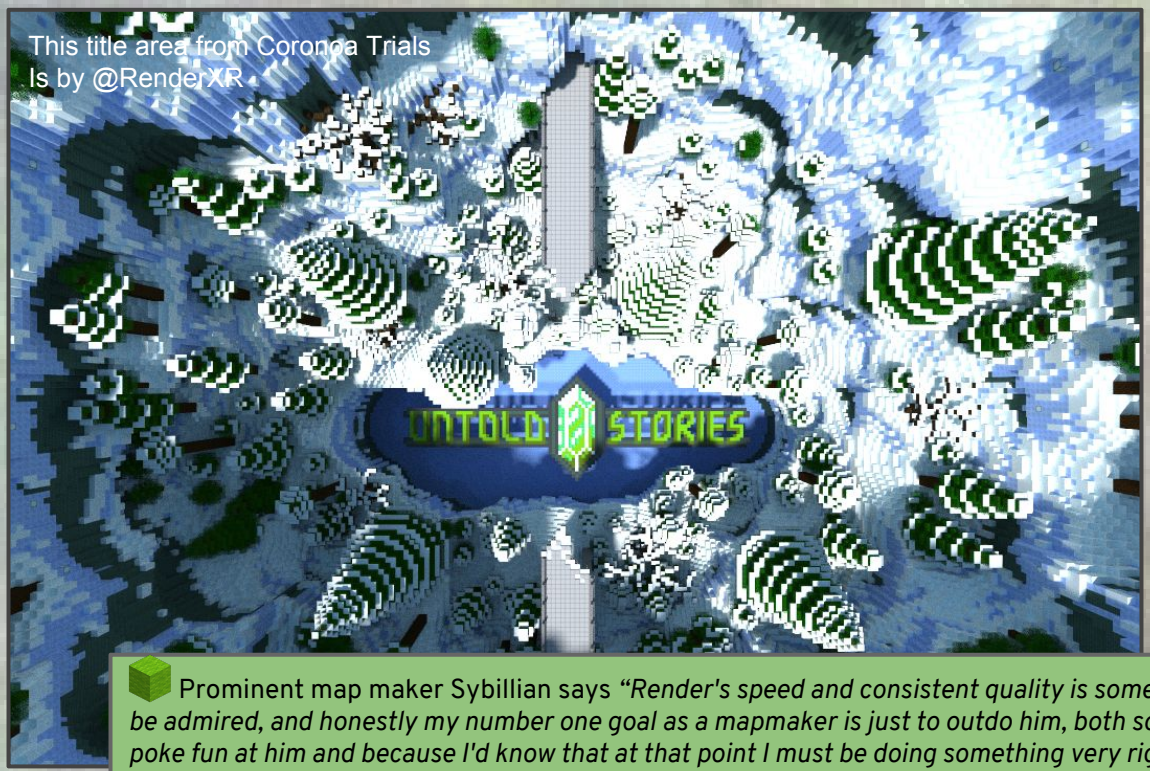
The Untold Stories of RenderXR

RenderXR has been playing Minecraft since Beta 1.2, and in his own words, he “really loves CTM maps due to their exploration and difficulty”. The first map he played “was Vechs’ Infernal Sky which made me in love with Vechs' maps. After waking up I started toying with making a map”.

The Untold Stories series charts his progress in the craft, starting with a massive complex world with many varied environments, Goliath offers traps aplenty. Bigleaf Forest focusses the action on surviving on a woodland island while hunting the hidden parts of the monument.

Instalment 3, Myriad Caves, adds a custom food system within a vast network of interconnected caves. Untold Stories 04 - Corona Trials - is uniquely positioned as an easier challenge with consistent and beautiful builds. Corona Trials is currently hosted on Realms, making it easy to explore the CTM style of gameplay with a few clicks.

With more maps on the way, catch up on the available installments by downloading from http://www.minecraftforum.net/forums/mapping-and-modding-java-edition/maps/2295375-ctm-collection-renderxrs-untold-stories-series



This title area from Coronoa Trials Is by @RenderXR

Prominent map maker Sybillian says *“Render's speed and consistent quality is something to be admired, and honestly my number one goal as a mapmaker is just to outdo him, both so I can poke fun at him and because I'd know that at that point I must be doing something very right.”*

DREAMWEAVER

Orange1095 blends strategic battles with adventure

Some map makers burst on the scene to change it forever and then disappear as their interests are drawn toward new and different challenges. Orange1095 was very prolific through 2013 through 2015, then used the skills developed through making and sharing impressive Minecraft CTM adventure worlds to move into game design.

The signature series “Emerald Dream” is three maps offering inventive storylines. The first map is available on request to the author. The second, titled “Dominion” has the player exploring an open world with dungeons and overworld locations to explore and defeat. The third, titled “Keystone”, leads the players through an episodic saga through different time periods.

The series has been the subject of countless Let’s Plays, though to avoid spoilers it is best to run up your own copy of a map and dive right in. Prolific map reviewer Ron Smalec hosted a map maker guest-laden play through in 66 episodes, available here: https://www.youtube.com/watch?v=pCVfXGe_Dd8&list=PLW71Hy4zymroFd1uhyxMc60tAn2M6_Hin

While Orange1095 has moved on, the legacy of his CTM maps and map making advice and turorials remain for you to discover and enjoy with a quick link click: http://www.minecraftforum.net/forums/mapping-and-modding-java-edition/maps/1527238-ctm-emerald-dream-series-by-orange1095-1-6-2-5000

The Monument

The Monument in Diversity 2, by @theqmagnet, drops falling blocks of wool from the ceiling

The Monument in CTM maps is a place where you place your prizes from completing the map challenges. It is often close by your spawn and has slots for wool blocks to be positioned.

The Monument can be a simple pedestal which fills up until there is no more room, indicating that the player has beaten the map and won the game. Otherwise the Monument can have logic within in the form of Redstone contraptions like a Block Update Detector (BUD) or Command Block wizardry so that the game detects your success, or transforms the Environment as goals are met.

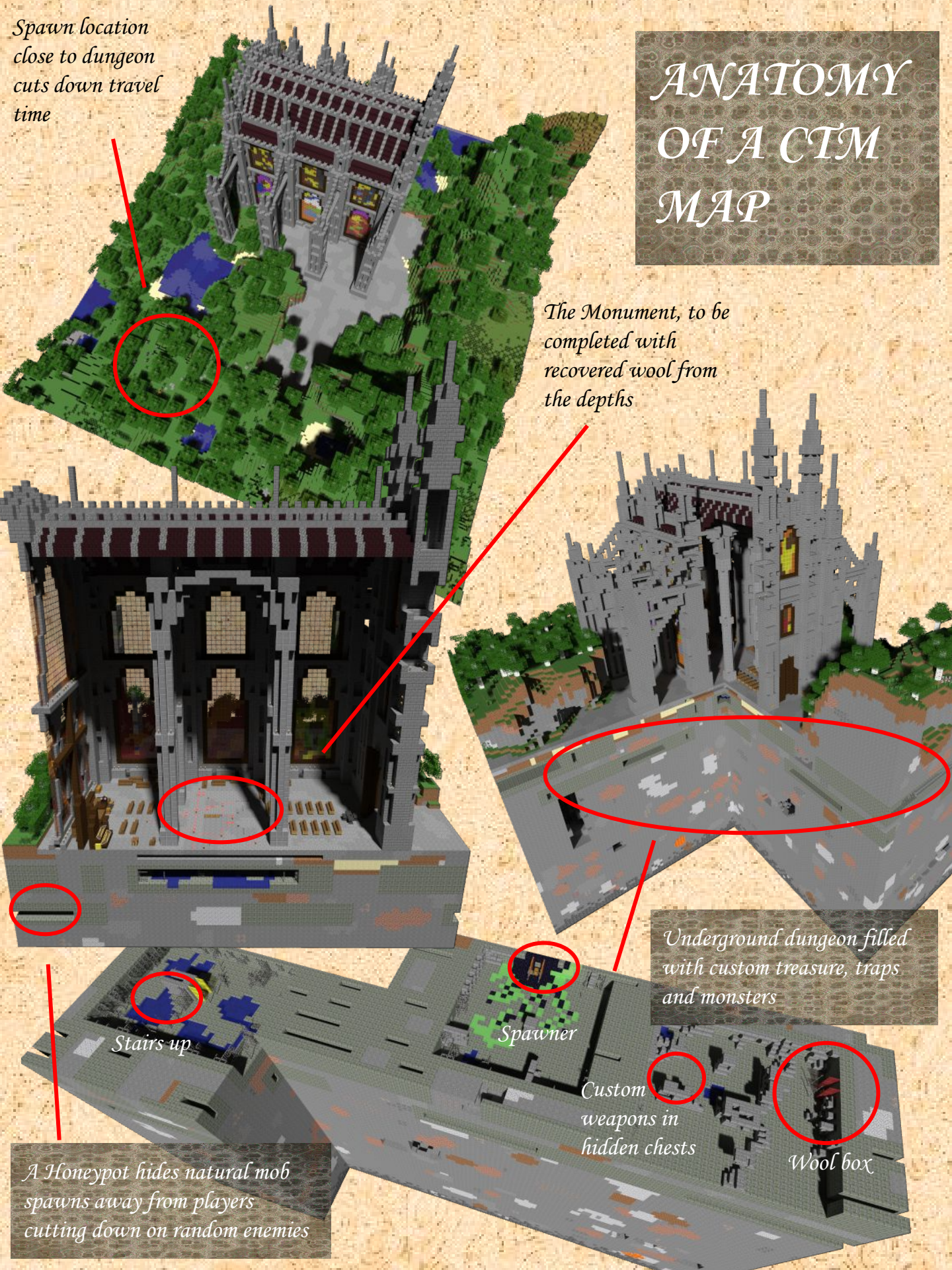
Whatever the level of complexity, the M in CTM is all about the Monument and so extra care should be given to how you build it. Some questions to consider:

How does the player know what to do? Will there be a Tutorial challenge?

What will the Monument look like empty and full?

What effects, if any, will I add when a block is placed, or the game is won?

How easily can the player return to the monument?


Spawn location close to dungeon cuts down travel time
ANATOMY OF A CTM MAP
The Monument, to be completed with recovered wool from the depths
Underground dungeon filled with custom treasure, traps and monsters
Stairs up
Spawner
Custom weapons in hidden chests
Wool box
A Honeypot hides natural mob spawns away from players cutting down on random enemies

Graphic by @MCAdecia

Across three weekends in July and August 2017 the CTM Community hosted a map development challenge where map makers were tasked to build a "Castles, Cathedrals, and Citadels" themed CTM world in 3 very short days.

A bumper crop of games were created. Thirteen maps are available today for download to play. They include custom builds, new mechanics, and in some cases unique custom resource packs and block models. You can get the maps through the post at https://redd.it/6tt78x or via the CTM Community discord at the following link: http://discord.gg/VZMkzf8

The challenge series is called "Strawberry Jam" and has an honour system based around a few rules:

- There must be at least three CTM blocks to recover
- Building cannot start before the competition has started

The Strawberry Jam challenge is a frequent event on the CTMC calendar and is scheduled at infrequent intervals. At the time of writing, challenge 16 is running in October 2017 and is themed "Nightmare".

Participation is open to everyone with a computer, access to Minecraft, and a love of building Minecraft maps. The CTMC discord dedicates a channel to offer help and assistance throughout the weekends.

Clear your calendars, and get cracking!

The Making of Instability, a Mini CTM Fit into a Confined Space

By PearUhDox

A view of the whole map "Instability". The playable area fits into a 25 by 25 by 256 space.

Quite a few mappers have made CTM maps based off of area constraints and limits before, notably the first map done this way, Minimalist, by Kaladun. The original inspiration for Instability was in Vertigo, an old Race for the Wool map by last_username. In Vertigo, players had to fight up a 16 by 16 by 256 lane to collect various colors of wool before an opposing team. In honor of that map, a few of the areas, such as the Gauntlet, were built similarly to areas on Vertigo. Instability, the Mini CTM for 1.12, was designed to fit in a 25 by 25 by 256 space, where the player can not leave the area, or build outside of it. The map features 5 wool to collect

However, due to the confined space of the map, dungeons had to be designed very carefully to overlap on each other, but not to cause a ridiculous amount of mobs and unbalance the map. Thus, many of the dungeons are designed to literally take up as much space as they can in the area, to avoid too much clutter in the map. Almost every spawner in the map was also reduced in detection radius so the player could explore each area without constantly being afraid of spawning mobs at the end of the dungeon, when the player was standing at the start of the dungeon.

As a result, natural mob spawns and fire tick were also disabled in the map when it was released, to better balance it to be fair.

The map featured several different types of areas and tried to brush on many of the “typical CTM areas” but with unique twists. The Scalding Cavern, the second objective area of the map, is a watery sandstone based area, which is fairly typical, but also features creepers as the primary enemy, as well as the “scalding water” (water which withers the player on contact, unless they are wearing a special chestplate found elsewhere in the map). The hectic design is intended to mix up the water area, which are normally slower paced areas.

The Scalded Caverns

Another area with a twist is the last area of the map, Illusioner’s Escape. It was designed to play very much like a void area, but without the danger of the actual void itself. The player in this area is attempting to levitate and bridge up to the top of the map, where an Illusioner, a new mob added in 1.12, awaits their challenge and guards the Black Wool. By sitting over the top of another area, the player can play a “void area”, in which one bridges to several platforms being attacked by a variety of mobs (in this case, vexes, shulkers, and ghasts) with less of a fear of the void itself. The Illusioner was really just brought in because I liked the new mob, and wanted an excuse to use it. Although in the end it fit well into the last area.

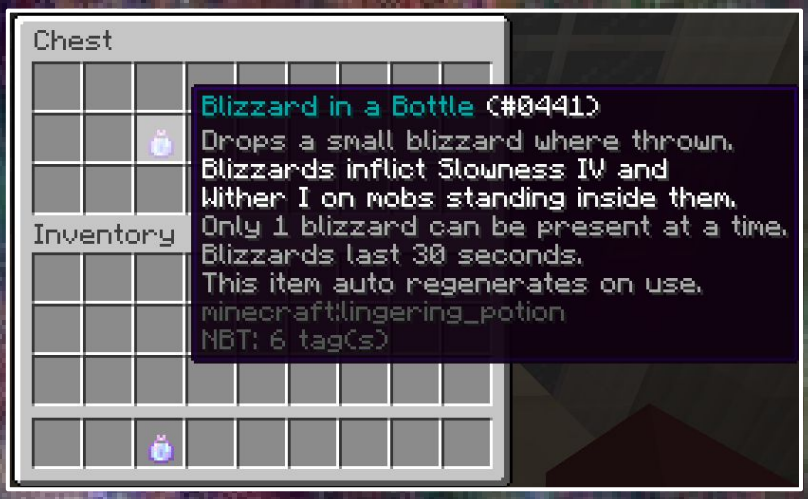
Above: The Illusioner’s Escape, an area in which a player must platform to multiple air balloons above the rest of the map.

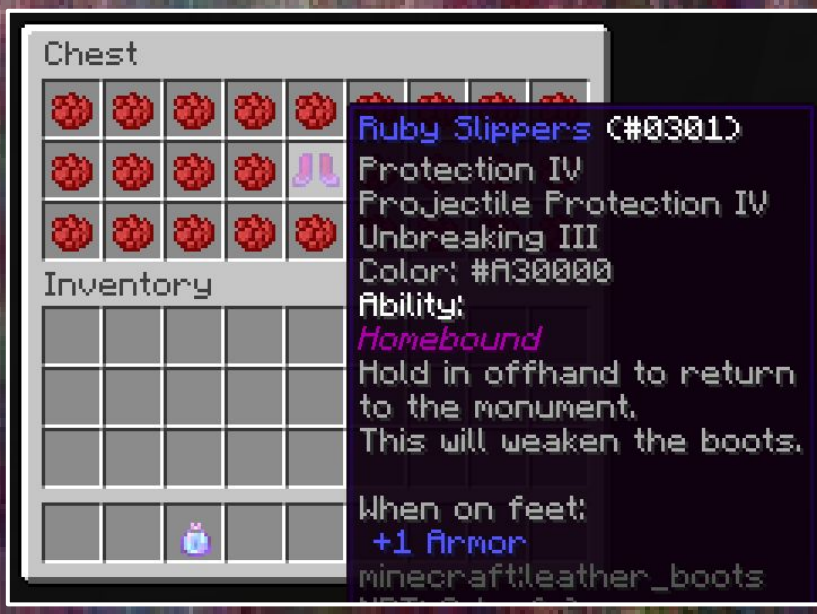
Right: The Illusioner who guards the Black Wool Box, seen above and behind the Illusioner. Picture was taken in development of the map.

Being a Mini CTM, since I wanted the player to experience a full loot curve in the map, I had to up the loot each area, getting better and better almost immediately into the next area. I also liked to provide the player with resources, such as Iron Blocks in the first area, but make the player wait to get them until they got a stone pickaxe in the second area. In my maps.

I also wanted to encourage player exploration and experimentation, which is why there are a few interesting custom items in the map, such as the Bark Belt, which buffs the player for standing on dirt, (with the intention the player will build little dirt buff pads around the map) and also why there are many secret items hidden just barely around corners in the map, to encourage the player to really immerse themself in the map and find every last secret, especially if they get a bit of reward out of it in the end.

Several custom items found in the map.





Overall, the map was designed to be a fast paced but have enough content to enjoy for several hours of time, and that is something I feel as if I did well. Being more condensed, the map lacks a bit in detail on aesthetic, and the map did go through several balance changes before being in a fairly suitable version that it is in now, the main one being that the first area was way too difficult on initial launch, and was rebalanced just to feel much better. The map also suffered from "first launched syndrome" where I made quite a few rookie mistakes, mainly assuming that the player would find some ideas and things obvious in the map that ended up being very cryptic. But in the end, Instability stands as a success story for me because of its unique theme and being the first map I have ever published for public use. It's success and positive feedback has inspired me to make more maps, and its flaws have better taught me several important things to remember when mapping.

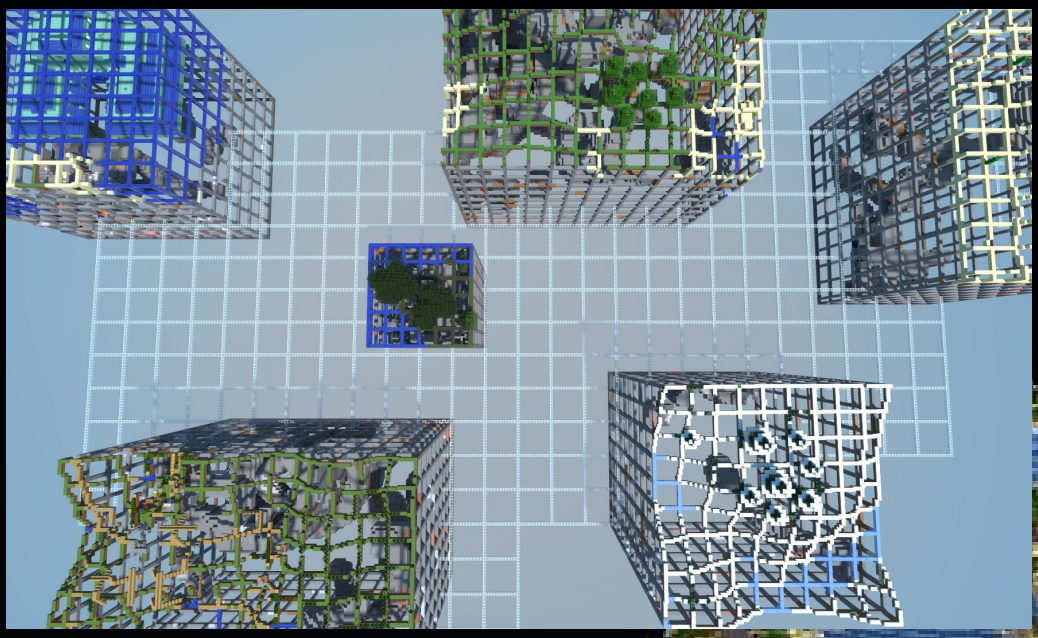
Another area in the map, called the "Abyssal Outlook"

PearUhDox is currently working on 2 full CTM Maps, titled "Illumina" and "Ascension" both of which are well on their way. Instability, as well as his other maps can be downloaded on his forum page here: http://www.minecraftforum.net/forums/mapping-and-modding/maps/2848255-ctm-collection-pearuhdoxs-ctm-corner

Written: September 8th, 2017

Abstraction, CTM Freedom

By Jigarbov

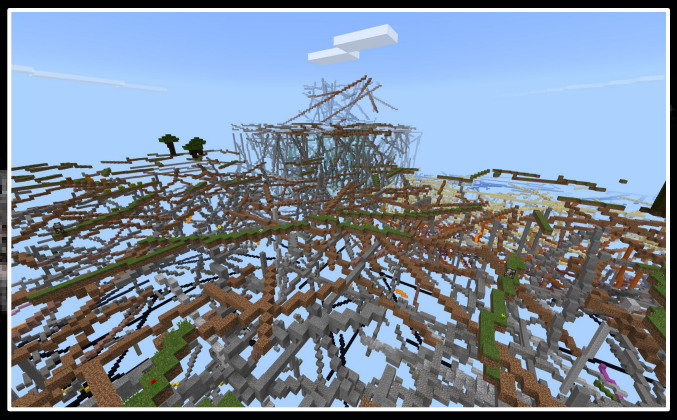


Abstraction: GRID *is a structured survival world with grids of different biomes to explore.*

I am no stranger to the CTM scene. ***Eronev 2: The Soul Cauldron*** pushed the limits of what a CTM was back in the day with a story focused CTM with puzzles to get each block instead of giant rooms filled with monsters to gain the wool. The same could be said with the Diversity series which to this day many claim it doesn't fit the "CTM" genre because of the way the blocks for the monument are gathered. In my new series "**Abstraction**" the rules are once again being bent in some pretty significant ways.

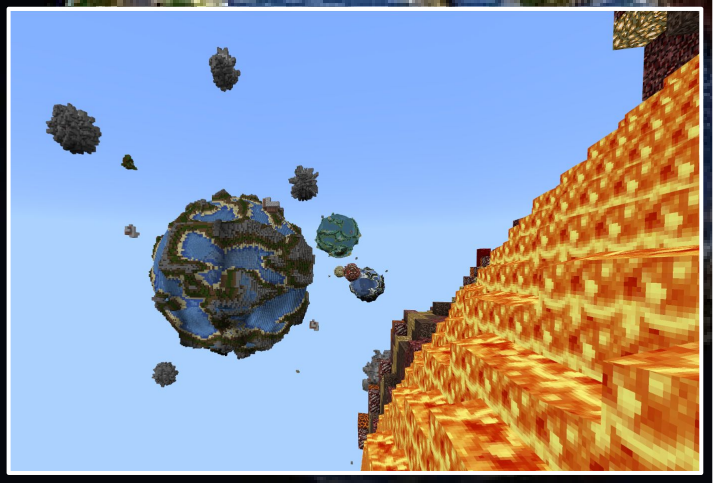
What makes Abstraction different is the freedom in which you collect the colored blocks to complete the monument. For the first time you are allowed to use any method you want to get the monument colors. Craft a dye from a flower? Done. Find a random sheep colored grey? Take it. This freedom allows players to take completely different paths through completing the quest to complete what I call the "Color Monument" in the Abstraction maps. Of course you can find chests with each of the colors in there but they often aren't wool but filled with dyes and colored terracotta. Each map has been painstakingly created in a way to facilitate this way to play so it isn't as easy as you might think, even with this freedom.

Each Abstraction map follows a different theme. **GRID** is a world made of grids, it looks familiar but due to the GRID structure, the landmasses are hollow, supplies are limited and it gives a very skyblock feeling to the survival aspect. **VECTOR** is made up of… you guessed it… VECTORs. It has a similar feel to GRID except the world almost becomes a parkour challenge at the same time. Spawners are placed at strategic locations to increase tension while you're out balancing on beams of gravel. This sky high challenge on blocks that can fall at the slightest update makes simply traversing the world a challenge rather than being inundated with hordes of monsters.

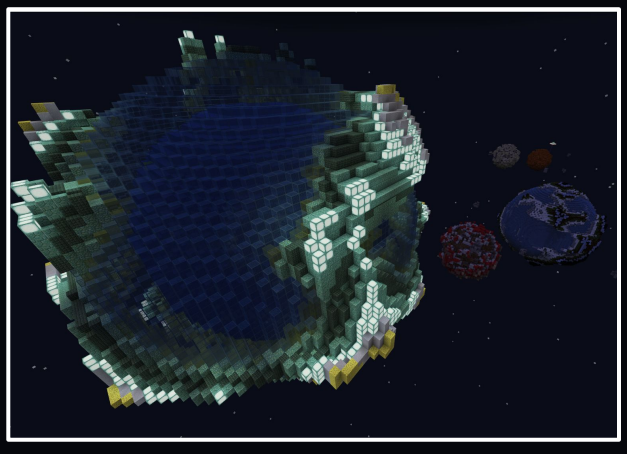


Abstraction: VECTOR *puts you on sky high tightropes*

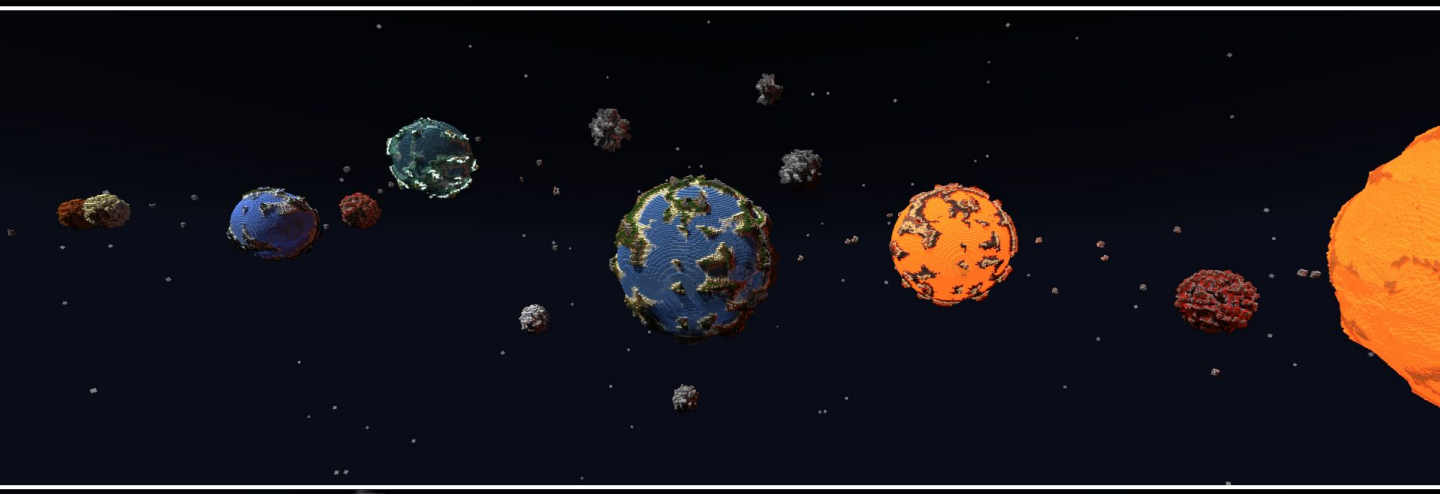
The third in the series is called **SOLAR**. While it bears resemblance to our own solar system, people will notice that it is not the same. It's made up of a number of planets and asteroids floating above the void. Obviously gravity still exists so you are limited to exploring the tops of the worlds unless you try to dig into the core. In the core of each planet are completely different kinds of environments, be they filled to the brim with standard CTM-style monster hordes or filled with ocean. So hold your breath and see if you can find the chests with the colors, or scour the surface for any sign of flowers to craft your own color to place on the color monument.



[Above] ***Abstraction: SOLAR*** *with a view from the lava planet. If you are able to stretch out your render distance you will get quite the view*



[Right]At night, ***Abstraction: SOLAR*** *really shines with planets designed in a way to stand out against the darkness of space*



Abstraction: SOLAR *in it's entirety is a hugely expansive world to explore*

While many of these maps and styles may look familiar to those with a long history in Minecraft JAVA, they represent the first time that people are able to play experiences like this on mobile and console. Crafting worlds like this take a lot of energy and time and without the support of the Marketplace they would not have been made, which is why they are available there for purchase.

Full disclosure, I also wanted to take this opportunity to give a shout out to @abrightmoore for all his help with the creation of these maps. Solar in particular may look familiar to anyone who has used his "world foundry" filter that creates wonderful planetoids in mcedit. He was also the inspiration for the "Abstract" nature of these maps. So thanks!

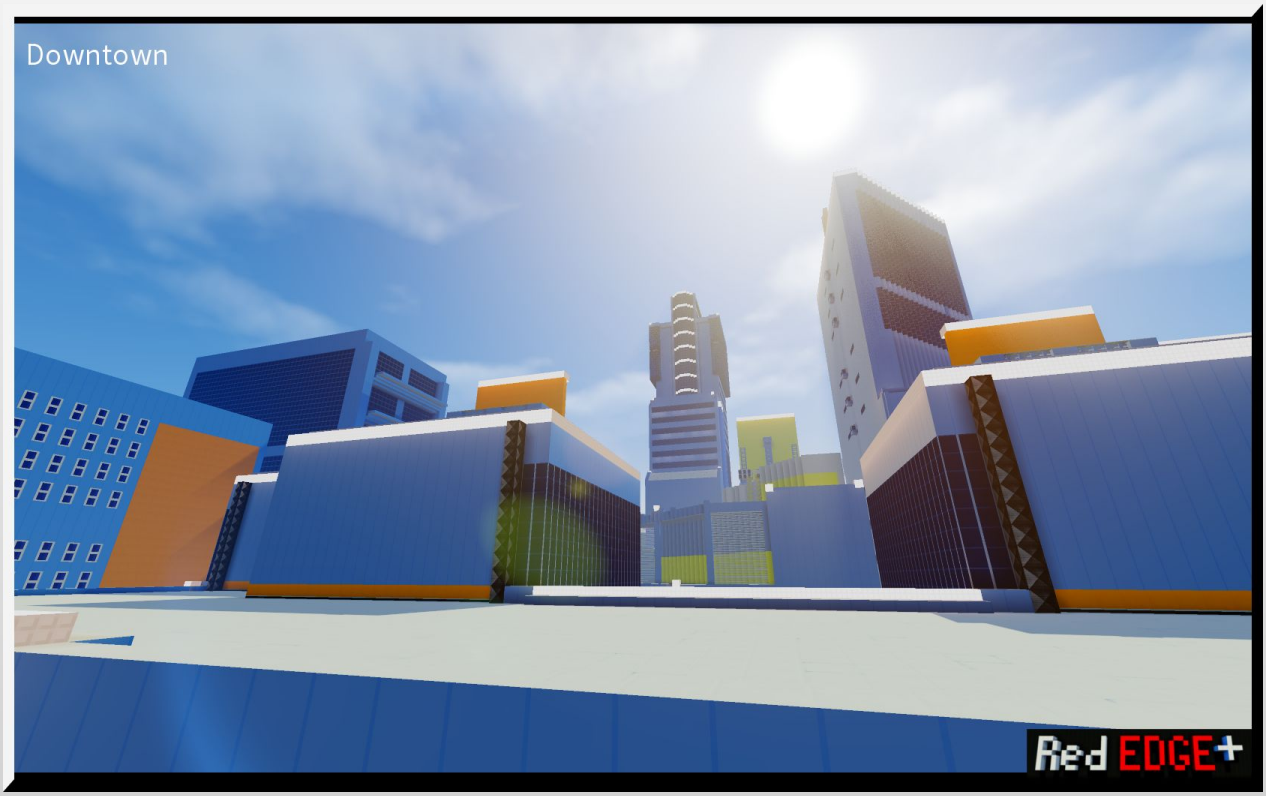
The "Abstraction" series of maps are all made by

Jigarboy PRODUCTIONS

Quick City-Building for Parkour

(aka “Parkour is more than random floating blocks”)

By DjEAR



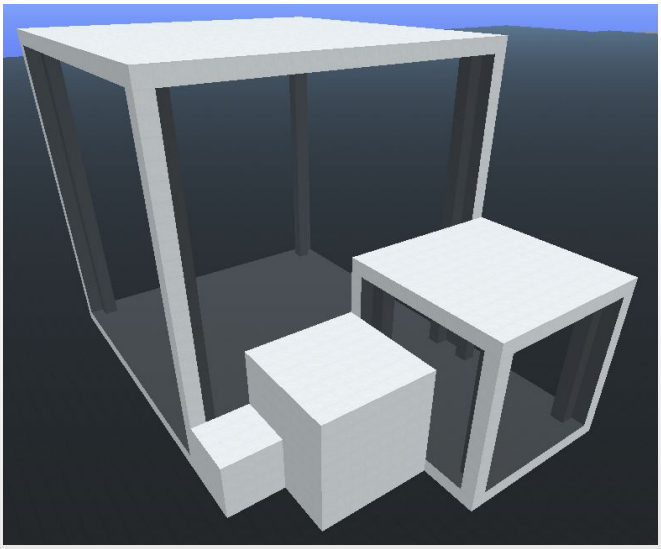
RedEDGE+ (Downtown District)

My name is DjEAR, and for the past few years I've been working on bringing the experience of DICE’s “Mirror's Edge” to Minecraft with my project “RedEDGE+”. So many maps execute on gameplay or story, but lack a visual wow-factor that imprint those experiences and map identity onto the player, and don’t stand out from the crowd.

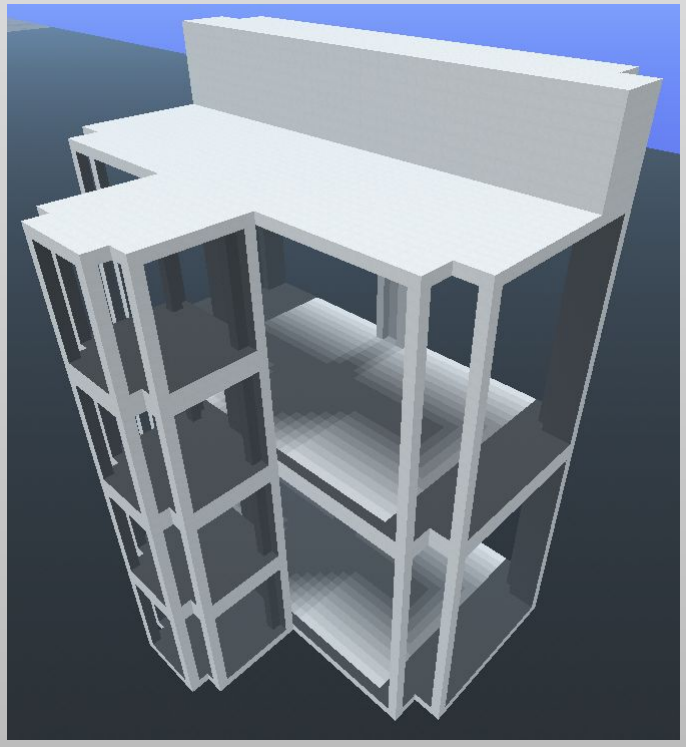
My inspiration for the project came after playing Lord_Pancake’s vanilla-built Mirror's Edge map “Edgecraft”. The level of detail, and visual closeness to the original game took me by surprise, and I've since expanded upon that to a larger scale and visual style closer to that of the reboot/sequel 'Mirror's Edge: Catalyst'.

Today, I want to explain one of the harder parts of this process that I continue to ask myself – how do you make a bunch of rectangles look like ACTUAL buildings? and how can you populate a map with a detailed city as quickly as possible?

The answer starts in an incredibly helpful tool for many map-makers, MCEdit – where much of the prep work and heavy lifting is done. My basic building blocks start with *6x6*, *8x8*, 1*2x12*, and *24x24* wireframe cubes. Once I had these created, saving them to a "*.schematic*" file makes loading, copy/pasting them a snap. I settled on 24 as the largest because a minimum height of 24 blocks is enough to kill the player upon impact from just 'walking' off the side of the building. Most of my buildings begin at a standard 48 blocks high, double that minimum – to sell the idea of being on top of a large structure, well out of reach of the traffic below. These rooftops are the main game floor, and while we can always drop down to street level if we want, it's in our best interest to provide the player plenty of upward, and downward direction to move thru the city. This also helps to easily control where players can, and can't drop down to without dying from a higher position.

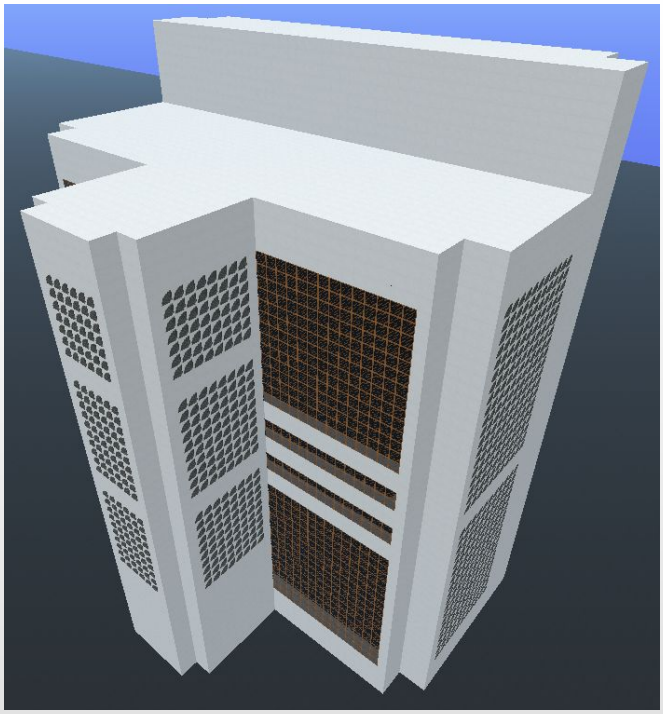


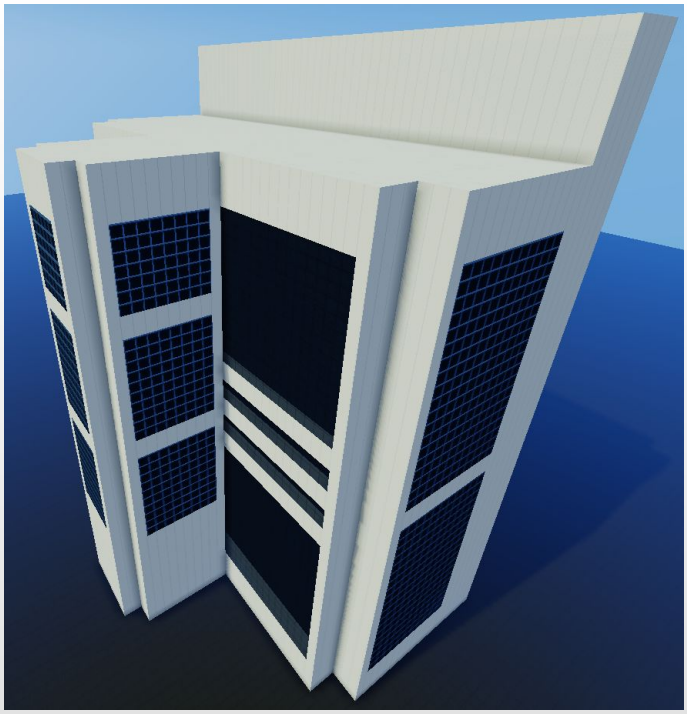
Alright, enough talk – let's design a building! If you take a look at the example – I started with a 48x48x24 tower (4 of our 24x24 cubes), stacked some 12x12 cubes up the center of one side, and brought the back end up a bit with my 8x8 cubes. Then, to add a bit of detail into this structure, I "rounded" out the corners a tad, to turn our stacks of cubes, into something a bit less uniformly geometric. I like to leave the insides of my buildings hollow just like you see here. Later on – I can always go back and create various ins, outs, and routes for the player to take within this building.



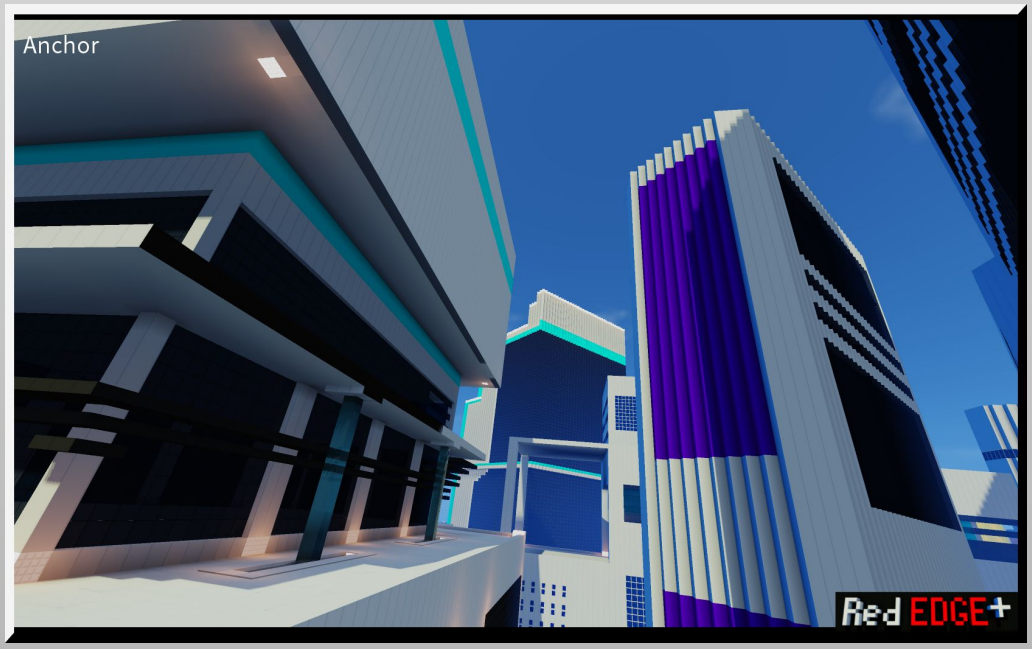
RedEDGE+ (Building Interior)

Watch the RedEDGE+ | Alpha Trailer





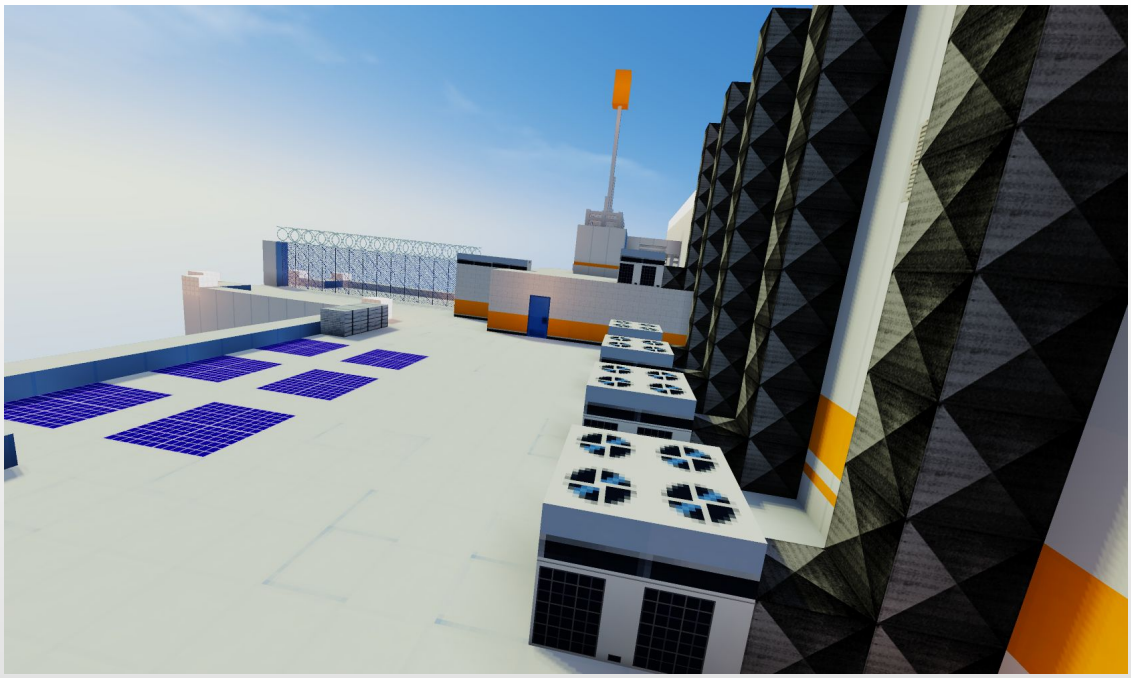
In this next shot, I've added some windows to the build in various – but fairly simple patterns. I won't go into them here, but just like our pre-saved wireframe blocks, having other pre-saved .schematics of various common building elements helps save a LOT of time. I have several different window patterns at different cube sizes, that can then be loaded and pasted into place over the outer surfaces of our building to quickly dress it up with very minimal time spent afterwards cleaning up any empty areas with the standard 'white concrete' that makes up the vast majority of buildings from Mirror's Edge. Just taking a look at the structure inside Minecraft we can see it's already starting to look pretty good. **Now's a good time to save the ENTIRE building to a .schematic for later use!**



RedEDGE+ (Anchor District)

All that's missing at this point, are some top-level details. Just like the shape of your building, or the window dressings you come up with, get creative! A few solar panels and more than enough AirCon units, along with some vents to the roof. A gated fence up in front to force the player across the small roof-access structure sitting in the center, and some simple orange accents to help guide the player visually and add more contrast.

Lastly, I always add some careful lighting to brighten up the whole thing at night. The main playfield has GlowStone in most corners for a base-ambient light level, with slabs connecting them, which helps to visually tell where the edge of the building is when standing on it. This is all brought home with transparent torches to guide the player with light, in the direction they need to go. I also use torches around the corners of the windows to help the building pop a bit better in the dark, and that's it!



Where you go from here is up to you, every building will likely require some attention to the top-level of detail, but once you've started building a collection of .schematics for various pre-built structures, you can quickly start placing down, rotating, moving, and combining buildings into different positions – before making additional tweaks and edits to help them stand out on their own.

The best designs show you where you need to go, and let the player choose how they want to get there. Some routes might be more difficult or rewarding than others based on your skill level, and that sense of forward momentum is amplified with a bigger playground to enjoy than just attempting to make precise hops from 1 floating block to another. Go out and run!

For More Information:

DjEAR

RedEDGE+

MCEdit

Written Saturday, September 16th 2017

INTERLACED MINDS

BALANCING IN MAP DESIGN

BY THE RAHN -INTERLACED MINDS, HEAD OF BALANCING/DESIGN

Whenever we play a game, we expect a certain challenge. No game can be conceived without a challenge. It can be of very different natures: intellectual (puzzle and mystery games), emotional (horror games, dating sims), precision and skill (shooters, platformers) among many others. Usually, those kinds are combined to deliver more interesting and complex experiences. When a player chooses to play a game, they expect to face a certain kind of challenge with just the right amount of difficulty. It is the game designer's task to choose what kinds of challenges they want in their games and how difficult those challenges must be; and to make sure that the difficulty they want, the one the player expects and the one actually present in the game are the same. It is this equilibrium what we call balance. But how is it achieved? Much has been written about the mathematical formulas involved in balancing,

but there is one fundamental principle upon which all balancing relies: all the positive things that help and support the player throughout the game, minus all the negative things that constitute the challenge and obstruct the player's way towards victory must add up to 0.

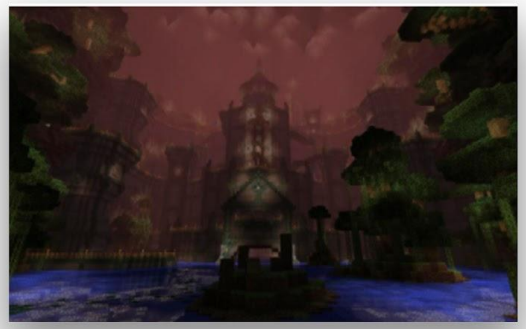
Indeed, this means we must assign numerical values to every element in the game whose existence makes it harder or easier in any way.

BALANCING IN MAP DESIGN

This task of value assignation is the hardest part of balancing. First, one must identify those elements. Some of them are very easy to find: a sword has attack points expressed as numbers that directly benefit the player; a Zombie can hit the player with damage points that directly harm the player. Those two numbers are relatively easy to balance against each other, but Minecraft maps are rarely that simple. Some numbers are a bit harder to calculate: how often should Zombies spawn? How easy it is for the player to obtain the sword? How many times can the player hit the Zombie before the sword breaks? Has the player been given the choice between a sword and a bow? What is the probability for a mob to drop rare and valuable loot? Furthermore, some elements don't even have an obvious number associated to it: how hard should a puzzle be? How much distraction should the player be subjected to when performing a precision move? How obvious or hidden should the path to a chest containing rare loot be? Unfortunately, there is no trivial solution to these questions. Let's look at some examples.

The following paragraph contains spoilers for Heliceo's "Ragecraft III: The prophecy" so if you plan on playing it you may want to skip it. Don't worry, more examples will follow.

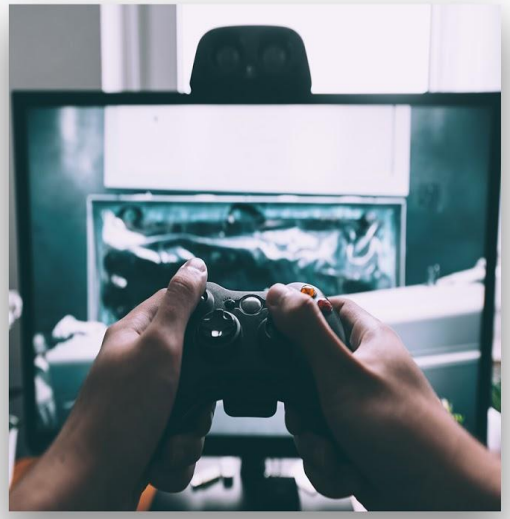
In Ragecraft III the player is given the optional task of collecting three unique items, very carefully hidden all over the map. With all three combined, the player is granted one extra heart container for the rest of the map. Here is where it gets interesting: while the first two are quite useless items by themselves, the third one is Nether Wart, the one and only piece of Nether Wart to be found in the entire map.



BALANCING IN MAP DESIGN

The lore box of the item itself container, or the possibility of brewing potions. All other potion ingredients are abundant in the map, but the otherwise absence of Nether Wart makes it impossible to brew potions from scratch, limiting you to only modify the ones you find as loot. It is an obviously difficult choice: both feel powerful and they have been so meticulously balanced against each other that taking the choice is a hard challenge by itself.

End of spoilers here, let's look at other examples. Think of the game "tag". Yeah, the one kids play where one kid is called "it" and must chase around after the other kids, tagging the kids as "it" when touched. Ever felt the game was too easy or too hard? That's because it was unbalanced, of course! The kid's speed and endurance are the key factors when trying to even out a game of tag, and countermeasures are sometimes put into place: limiting faster kids to walking instead of running; making it impossible for slower, younger kids to be "it"… Indeed, making advantages and disadvantages add up to zero, or at least close to zero. Both of these examples

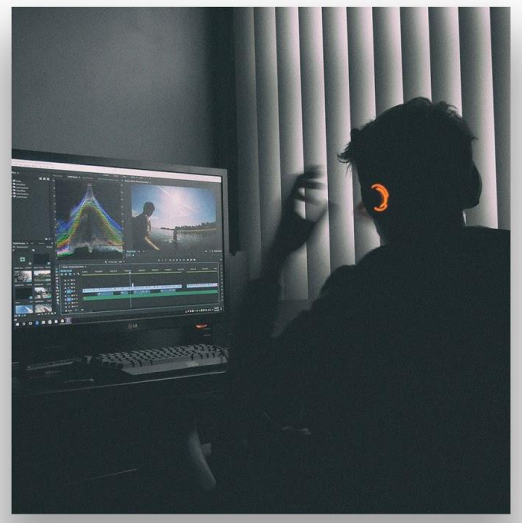


involve things being balanced and feeling balanced. Where is the difference? We say something is balanced when numbers have been calculated so that it is neither too good nor too bad; when benefits and costs add up to something close to zero. In turn, players will say something feels balanced when they get the impression the challenge is fair, regardless what your numbers said.

Earlier on we spoke about how hard it is to assign numerical values to abstract challenges and having them add up to zero. Well, all that hard process will only yield numerical balance. Unfortunately, players rarely care for numerical balance, except when playing technical games. As we discussed at the beginning of the article, players

BALANCING IN MAP DESIGN

will come to our games seeking fair and interesting challenges. That fairness and interest may be based upon numbers, and mathematics will definitely help get close to that objective or even reach it completely sometimes, but more usually than not there will be a gap between what numbers say is balanced and what players will feel balanced. A very useful tool to bridge that gap is playtesting. Playtesting, as all other testing in any game development (Mapmaking included) should be applied as early as possible in the development, and be repeated every time a significant change is introduced. For those of you not familiar with the term, it consists on taking your current version of the map, finished or not, and playing it through as if you were the player. Besides being useful



or finding bugs, errors and exploits, it also helps developers get a feeling of how their game is progressing. Much like cooks taste their dishes while cooking and before delivering the food to the guests, we as Mapmakers should “taste” our maps and see if they’re missing a “pinch” of Mob Spawners, or if they maybe have too much Parkour in a certain section. But wait, you may argue. Why even bother with complex mathematics and number conversions if experiences are all that matter? While playtesting is a powerful tool, it is somewhat repetitive to use, and may lead to misassumptions if not properly executed. One of the golden rules of CTM says: “do anything you can think of to survive”. That rule is there to promote creativity in problem solving and to make the player thinks of alternative routes and solutions. Interesting CTM maps should be so flexible that players can find solutions not even the map creator could think of when making it. How can you playtest through those solutions? That is where numerical balancing proves its worth. You may not know in advance what players will do with the Items, Traps and

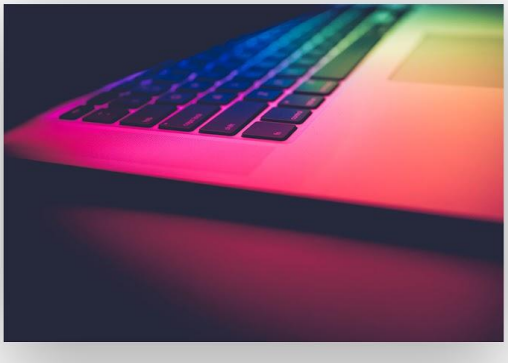
BALANCING IN MAP DESIGN

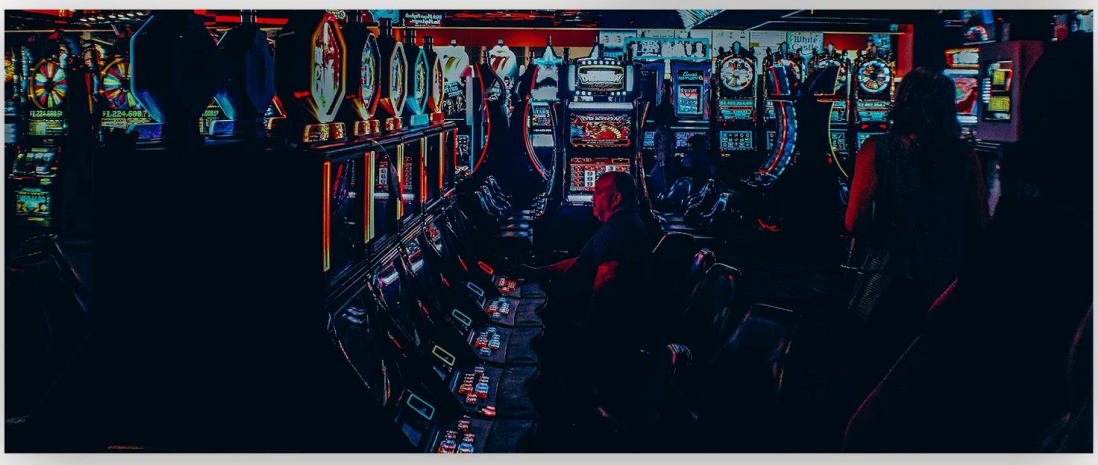
Mobs you put in his/her way to a full monument. But a complete and structured knowledge of how all relevant numbers in your map interact with each other will help you be sure that what is meant to be hard will be hard and what is meant to be easy will be easy.

A lot of questions were asked throughout this article, and their answers remain unclear: How do you assign numerical values to abstract challenges? How would you integrate mathematics and playtesting? How difficult should my map be? There is not a single, universal answer for those. It is now up to you to look at your "work-in-progress" map and wonder "Will the player experience the same way that I want them to?", "Does the difficulty naturally progress throughout the map?", "What will make the player say: "Now, this is a map I want to play""?

FURTHER READING

For more infmation see;
http://bit.ly/2yKWf8r





INTERLACED MINDS

WHO ARE INTERLACED MINDS?

Interlaced Minds is a new mapmaking group with over 30 members!

We are currently working on a large-scale project. Our aim is to revolutionise mapmaking with our unique story-based, immersion focused, sci-fi/ dystopian adventure “Broken Worlds” We are currently working in partnership with the build team Astrium, who are producing a beautiful 8k x 8k world, where our first trilogy takes place. Additionally we are working with BlockBench creator Jannis, on a revolutionary NPC/ armorstand animator that will allow us to convery our story in visual ways never seen before. Our core team, have been working tirelessly, designing every aspect of the map: script, cinematography, gameplay, mechanics, balancing, music, sound effects, models, textures, voice acting, animation, combat, puzzles, enemies,

items. Such an undertaking is no small decision, but we really want to make an imprint on the community

And we’re looking for more talented members!

Open jobs:

- Command/ Functions
- Builder
- Voice Actor
- Composer
- Sound Effects coordinater
- Modeller
- Builder
- Writer
- Game Design
- Graphics
- Video producer

Apply here: http://bit.ly/IMapply
Social: http://bit.ly/2ytIPSe
Credis: http://bit.ly/2xZbydx

The Future of Technical Advancements in Mapmaking - CreeperMagnet_

In the recent 1.12 update, Mojang introduced two new groundbreaking technical mechanics. Some of you might know the first, functions. That's a whole other topic. However, the second is a bit more obscure… advancements.

Advancements were introduced as the customizable replacement of achievements, and have lots of cool features. The main one we'll be focusing on here is the fact that they can *reward functions when complete*.

What does this mean? This means that commands can be super efficient, can have different triggers, and even not require any clocks at all, because advancements don't require commands checking for criteria every tick!

How is this done?

Well, advancements, as some of you might know, are fully customizable. They can have *criteria*, *rewards,* a *display,* and a *parent*. Let's take for example, the vanilla advancement for entering a nether fortress for the first time, find_fortress.

find_fortress.json

```
{
    "display": {
        "icon": {
            "item": "minecraft:nether_brick"
        },
        "title": {
            "translate": "advancements.nether.find_fortress.title"
        },
        "description": {
            "translate": "advancements.nether.find_fortress.description"
        }
    },
    "parent": "minecraft:nether/root",
    "criteria": {
        "fortress": {
            "trigger": "minecraft:location",
            "conditions": {
                "feature": "Fortress"
            }
        }
    }
}
```

I'm going to copy this advancement from my minecraft folder's assets, and transfer the copy from there, to the world/data/advancements/… folder of my world folder.

However, the advancement needs to be inside a namespace folder to work, so I'm going to use the namespace "demo". This means my file would be in the folder world/data/advancements/demo/find_fortress.

Now, the advancement editing phase!

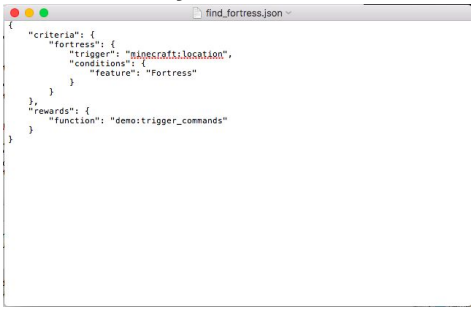
We won't be worrying about the *display* or *parent* options, as the advancements we're going to be using don't need to be displayed, so we'll delete these options.

However, what we do want are the *criteria* options. In this example, you can see that inside the main *criteria* brackets, there is a *trigger.* In the find_fortress example, you can see that the *trigger* is minecraft:location, and it has a *condition*, a *feature* tag. Right now, this *feature* tag is set to Fortress. This means, whenever a player enters a *location,* in this case a Nether Fortress, then the advancement will complete.

So what is the point of this? Well, once the advancement completes, the player who completes it gets granted anything within the *rewards* tag. In this case, we have no rewards tag (yet), so we need to add a rewards tag in here!

So what should be in the rewards tag? Your function, of course! In this example, I have created the function "trigger_commands" inside the folder "demo". The trigger_commands folder goes inside the world/data/functions/… folder of your world file. We'll worry about what's inside the file later.

For now, we need to add the *rewards* line of the advancement. We want to have it reward the function we just made, so we add the following text:



Don't know how to create a function? Just simply make a simple text document and change the file ending from .txt or .rtf to .mcfunction, place it in the proper folders, and you're good to go!

And now we're complete! Mostly, at least.

Now, we need to work on our function, trigger_commands. Inside this file, we want to revoke the advancement, so the player can earn it again, and trigger the whole process over.

To revoke the advancement, we simply put this line of code in our function file:

```
advancement revoke @s only demo:find_fortress
```

Then, we want something to happen when the advancement is completed, otherwise known as when the player enters a nether fortress.

We'll just do a simple /say command.

```
advancement revoke @s only demo:find_fortress
say I found a nether fortress! Yay!
```

And there you have it! A simple advancement trigger that can run a command when you're in a nether fortress.

Of course, this isn't the only thing you can do with advancement triggers. These advancements can detect if you're in a certain biome, if you're hit by a certain entity, if your inventory is changed, and a whole load of other stuff you can't do with command blocks. There's so many options for advancements! The full list of triggers, and how to use each, can be found at: https://minecraft.gamepedia.com/Advancements

-CreeperMagnet_

Advancement Hunting – By Cavinator1

I like achievements. Achievements can either be guiding for the player throughout a game, or give them some extra interesting challenges to try to complete. If a player has already played their favourite game backwards and forwards and wants to try something different, they can become an 'achievement hunter,' having an end goal to get every single achievement available in the game.

The recent release of Minecraft 1.12 replaced the simple and rarely updated achievements system with advancements. It was suddenly possible for players to create and add their own custom achievements to Minecraft, both for survival worlds or for custom maps, a task that previously would have required a complex command-block contraption to be added to a Minecraft world.

Over the years of playing Minecraft I had come up with a long list of my own achievements that I would write down in a notebook and then cross out when I 'got' that achievement in any survival world that I played. And so, during 1.12's snapshots, I set out to create my own set of advancements. I added many of my ideas to the advancements pack. Then I playtested the pack with my good friend @MichaelMKenny in our survival world.

Without any ideas for a better name, we decided to call it BlazeandCave's Advancements Pack, and put it up on Planet Minecraft for download, including easy installation instructions, for anyone to play and use in their survival worlds or even on servers, from WhoCraft to De Club.

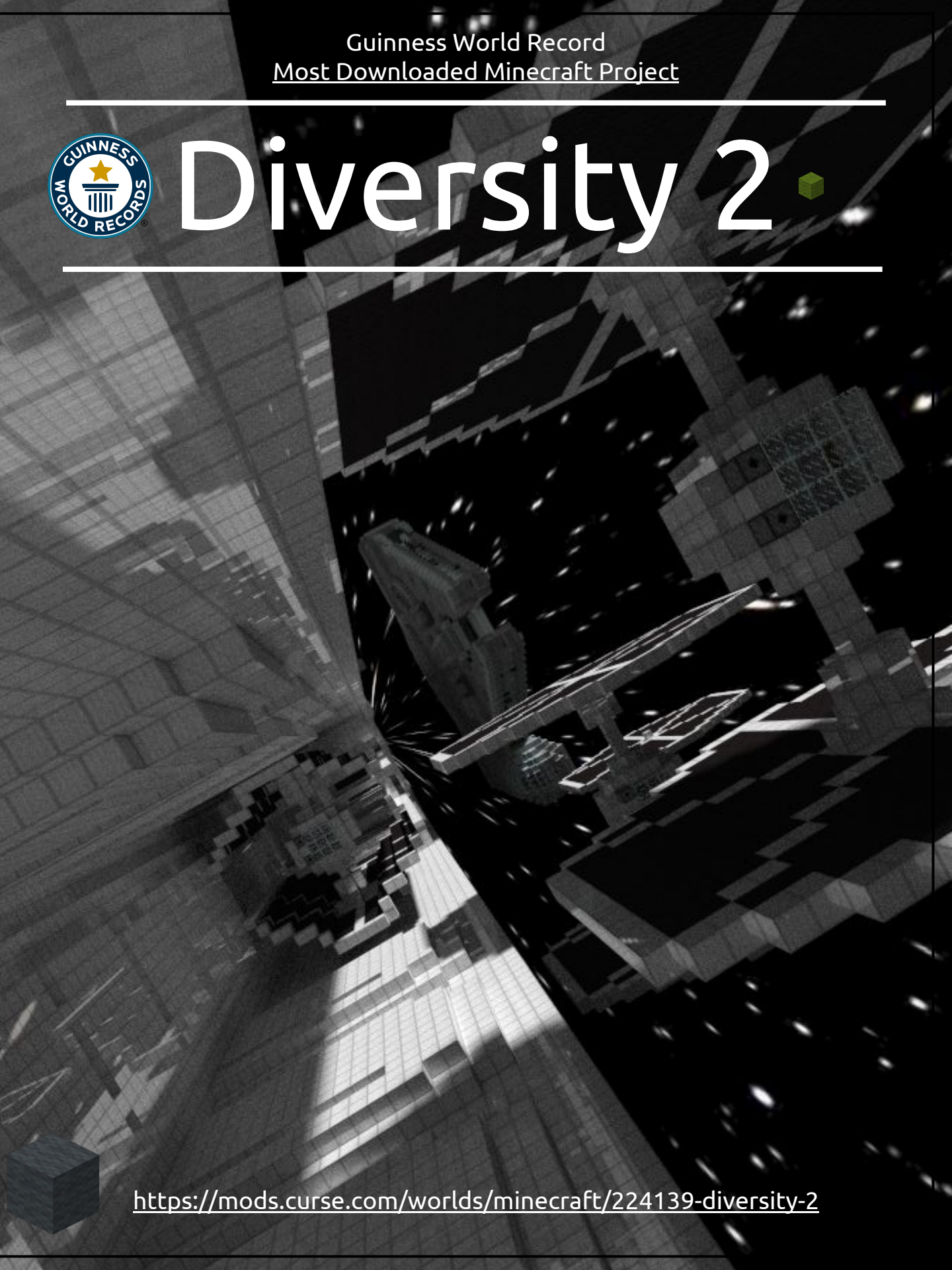
The Pack has so far been downloaded over 1000 times, and been diamonded, favourited and commented on positively numerous times.

The Pack currently adds over 250 advancements to your survival world, bringing the total up to 329, spread across ten tabs, with more being added whenever we update it. It includes reward functions that can also be optionally installed alongside the Pack, which makes many advancements give item and experience rewards upon obtaining them.

We are accepting ideas for new advancements to be added to the Pack, so if you have any, feel free to comment on the Planet Minecraft, Minecraft Forum, or Reddit page, or tweet me on Twitter at @_Cavinator1_.

Guinness World Record
Most Downloaded Minecraft Project

Diversity 2



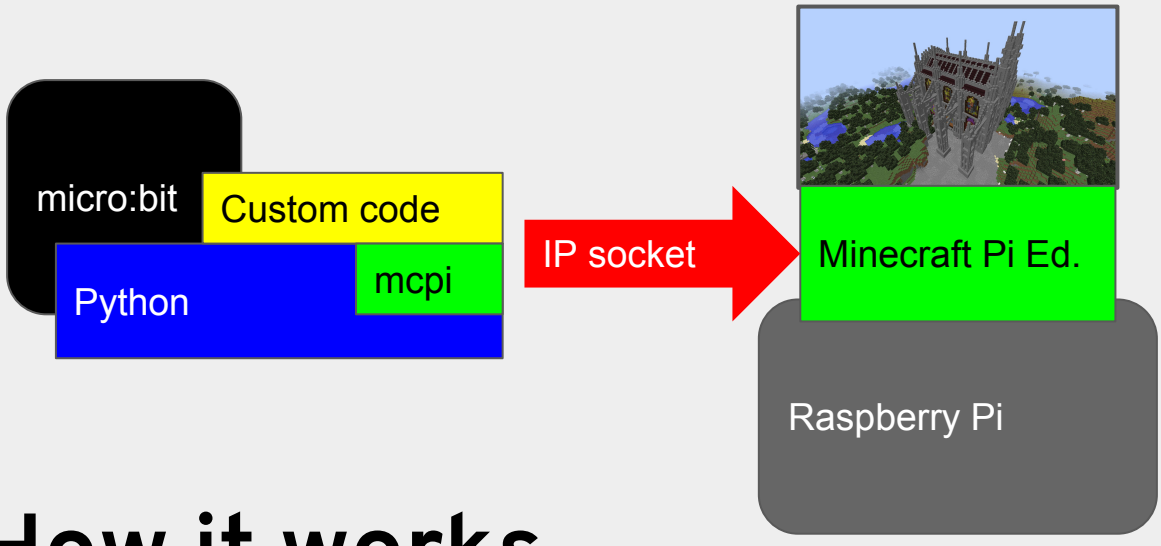
https://mods.curse.com/worlds/minecraft/224139-diversity-2


PHYSICAL
COMPUTING
WITH
MINECRAFT
1
2
3V

Setup and Equipment

In this issue we have been gifted the fun of hooking up sensors in the real world to the virtual world of Minecraft. With a touch of Python code bridging the gap, NCS:Computing takes us through a bewildering range of practical and fun applications using the micro:bit card.

The micro:bit is a low-power ARM chipset based credit-card computer intended for use in education environments. It comes with an accelerometer onboard, which allows the tiny device to detect which way it is tilted and moved in physical space. In this way the movement of the device can signal changes to a program which then takes action within a Minecraft world running on a networked Raspberry Pi using the mcpi Python interface.



How it works

How to get started

David Whale has developed the BitIO library. It essentially lets you code your Microbit in normal Python 3. Last summer I created these resources for Micro:bit:

Each program involved a Micro python program which sat on the Micro:bit and then a separate Python program that then interpreted the data sent by the Micro:bit. This was technically possible but elongated.

David has essentially stripped out much of the complexity. Now there is a generic hex file which sits on the Micro:bit. The only coding that you do is in Python 3.

Here are the steps to get a simple demo working:

1. Go to Davids github: https://github.com/whaleygeek/bitio

2. Look on the far right of the webpage for a green box which says 'clone or download'. Then click 'Download Zip'. Then save the zip file to your Pi.

3. Using the file manager find the downloaded Zip file and extract it.

4. Now plug in your Micro:bit to the Pi. Go into the Bitio master folder and you will find a hex file called 'bitio.hex'. Copy this and paste it onto the Micro:bit. This should now be flashed.

5. Go to the Bitio master folder and find the src folder. You will see several examples. The one that we will test uses the accelerometer to control the position of the player in Minecraft.

6. The file is called 'tilt_mc.py' open this in Python 3.

7. Open Minecraft and create a new world. Press f5 to run it the Python script.

8. The following dialogue will display in the Python shell window:
No micro:bit has previously been detected
Scanning for serial ports
remove device, then press ENTER

9. So unplug the Micro:bit and press enter.

10. Next the following text will display:

Scanning…
found 67 device(s)
plug in device, then press ENTER

11. Now press enter and the following text will display:

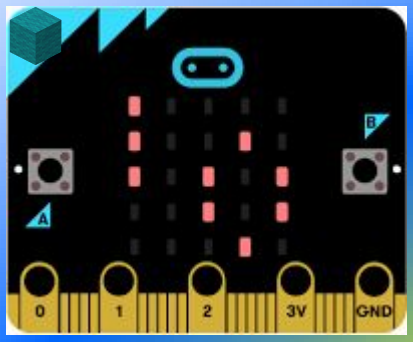
Scanning…
found 68 device(s)
found 1 new device
selected:/dev/ttyACM0
Do you want this device to be remembered? (Y/N)

12. Press enter to confirm and the following text will display:

Your micro:bit has been detected
Now running your program

13. If you pick up your Micro:bit then and open Minecraft you should see the player move according to the direction you are tilting the Micro:bit.

Physical Computing with Minecraft 2: Extending the Accelerometer demo: Walking Rainbow Blocks

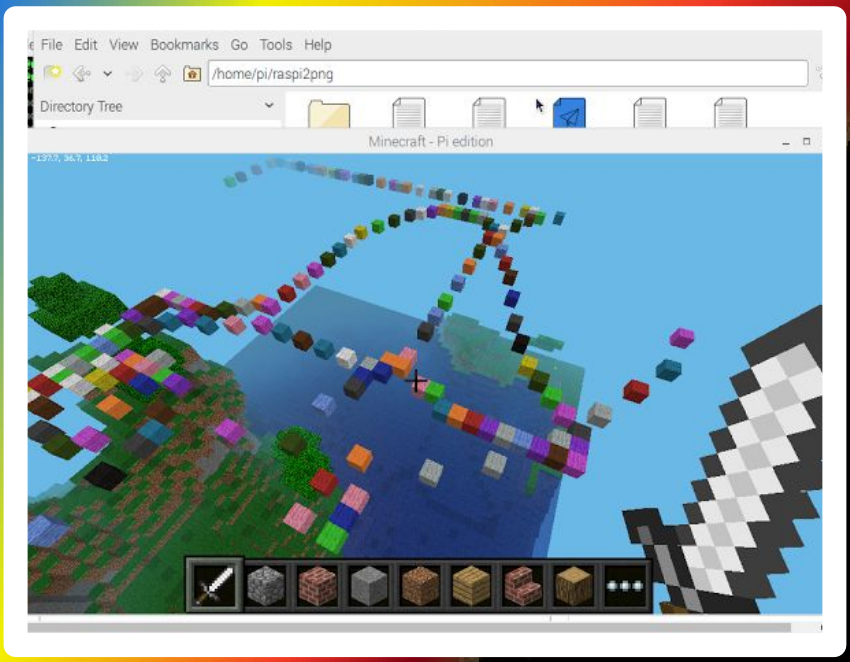


This takes the basic 'tilt_mc.py' demo from David Whales BitIO master available here: https://github.com/whaleygeek/bitio

In the first blog the demo moves the character around the screen according to the direction you tilt the Micro:bit. The code below extends this to:

1. Use a list to randomly select a wool block.
2. Drop this block underneath the player the end result is a colourful mess. :) (See below)

The code will need to be saved in the 'src' folder to work.



Below is the code, you can download it here:
https://github.com/ncscomputing/HpAnthologyV2/raw/master/Walking%20tilt.py

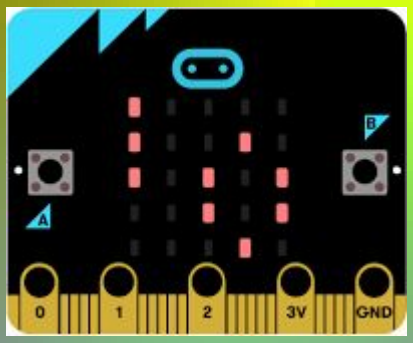
Code:

```
#Original written by David Whale located here as part of his BITIO library
#https://github.com/whaleygeek/bitio/blob/master/src/tilt_mc.py
#adapted by @ncscomputing 19/07/17


import mcpi.minecraft as minecraft
import mcpi.block as block
import microbit
import time
import random

mc = minecraft.Minecraft.create()

blocksList = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]

while True:
    pos = mc.player.getTilePos()
    x = microbit.accelerometer.get_x()/300 # -ve=left/+ve=right
    y = microbit.accelerometer.get_y()/300 # -ve=forward/+ve=backward

    pos.x += x # east/west
    pos.z += y # north/south

    mc.player.setTilePos(pos.x, pos.y, pos.z)
    mc.setBlock(pos.x, pos.y-1, pos.z,35,random.choice(blocksList))

   # time.sleep(0.5)
```

Enjoy :)

You can now download a pdf version of these tutorials as part of the updated #hackpack resource booklet here:
https://github.com/ncscomputing/HpAnthologyV2/raw/master/Hackpack%20Anthology%20V2%200.3.pdf

Physical Computing with Minecraft 3: Extending the accelerometer 'Tilting Rainbow Road'

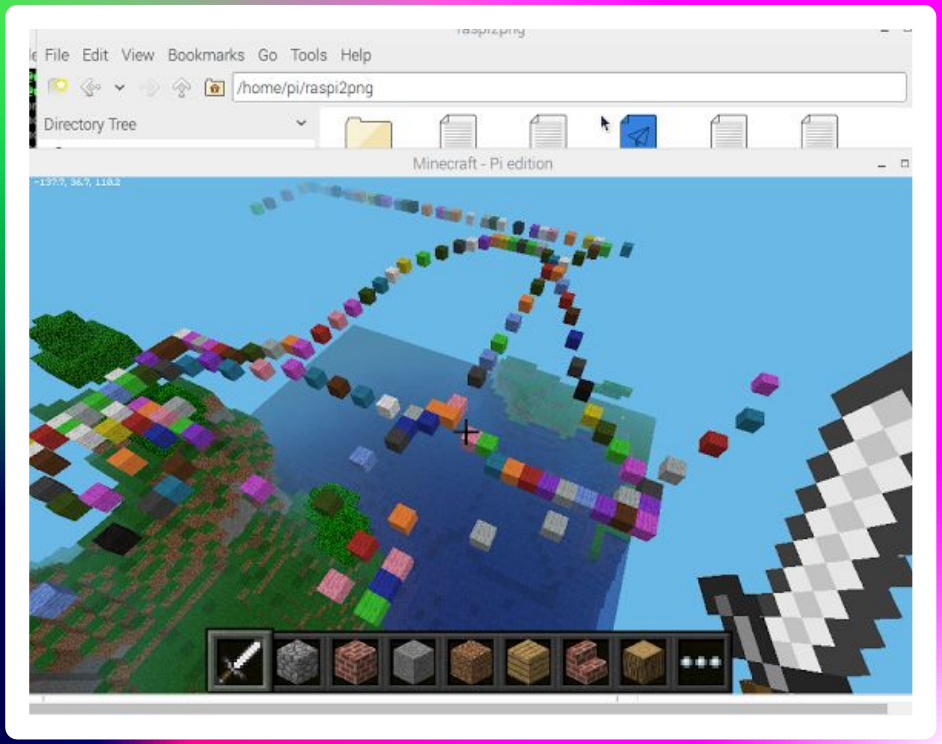


This takes the basic 'tilt_mc.py' demo from David Whales BitIO master available here: https://github.com/whaleygeek/bitio

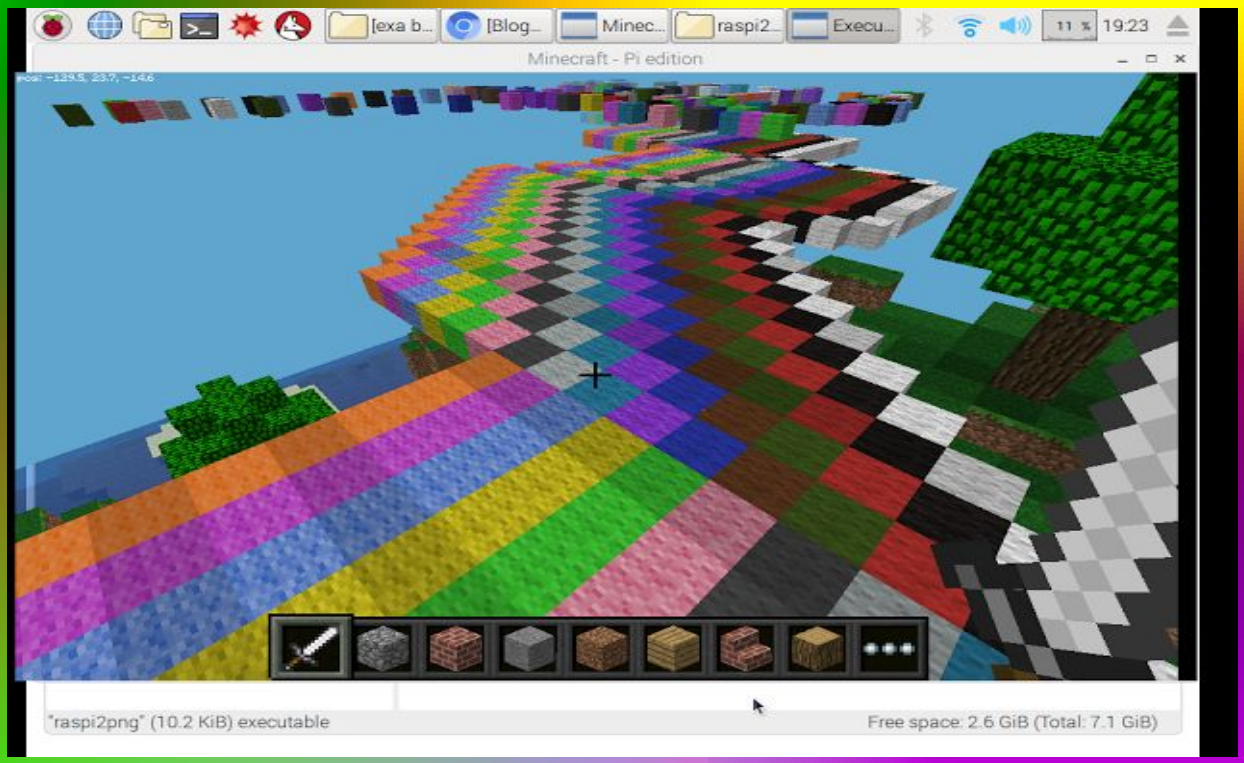
In the second blog the 'tilt_mc.py' demo was extended to:

1. Use a list to randomly select a wool block.
2. Drop this block underneath the player the end result is a colourful mess. :) (See below)

The code was saved in the 'src' folder to work and did this >>>



This third blog will extend this further to drop a roads worth of blocks each time the character is moved from a tilt of the Micro:bit. The end result will look like this:



Again you will use the accelerometer to move player and it will drop the road to the left of the character. The code will need to be saved in the src folder. (This is all explained in blog 1) Here is the code (a marginally more complex version of the previous blog):

Code:

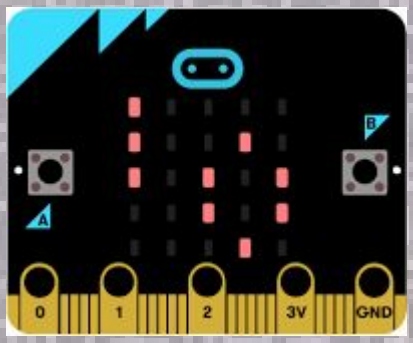
```
#Original written by David Whale located here as part of his BITIO library
#https://github.com/whaleygeek/bitio/blob/master/src/tilt_mc.py
#adapted by @ncscomputing 20/07/17


import mcpi.minecraft as minecraft
import mcpi.block as block
import microbit
import time
import random

mc = minecraft.Minecraft.create()
```

```
blocksList = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]
time.sleep(7)
mc.postToChat("Rainbow Road 3.0 Micro:bit controlled")
while True:
    pos = mc.player.getTilePos()
    x = microbit.accelerometer.get_x()/300 # -ve=left/+ve=right
    y = microbit.accelerometer.get_y()/300 # -ve=forward/+ve=backward

    pos.x += x # east/west
    pos.z += y # north/south

    Count = 1
    while Count <=16 :
        mc.player.setTilePos(pos.x, pos.y, pos.z)
        mc.setBlock(pos.x-Count, pos.y-1, pos.z,35,Count)
        Count = Count+1
    time.sleep(0.25)
```

Download the code from here:

https://raw.githubusercontent.com/ncscomputing/HpAnthologyV2/master/Rainbow%20road%20tilt.py

Physical Computing with Minecraft 4: ‘Tilt me Around the World’

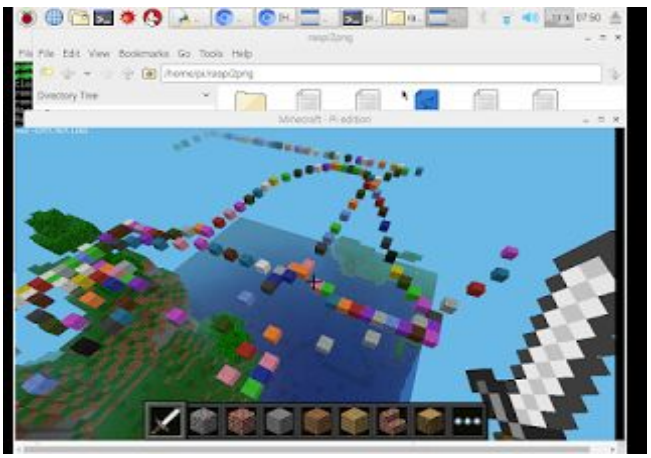


This takes the basic 'tilt_mc.py' demo from David Whales BitIO master available here: https://github.com/whaleygeek/bitio

Credit to Damien Mooney for the world rendering script: https://damianmooney.wordpress.com/2016/02/16/raspberry-pi-minecraft-iss-tracker/

In the second blog and third blog I extended the basic demo to drop random wool blocks where ever you tilted the Micro:bit and in tutorial three I extended this to lay a road of sorts.

Blog 2 (http://warksjammy.blogspot.co.uk/2017/07/blog-2-extending-accelerometer-demo.html)



Blog 3 http://warksjammy.blogspot.co.uk/2017/07/bitio-blog-3-extending-accelerometer.html

In this blog post we will hike up the challenge to combine both push buttons in addition to the accelerometer.

Section 1 of the code:

If you press the a button on the Micro:bit then a data file is read and world map is then rendered into the Minecraft world. You can then use the accelerometer to navigate the character around the map by tilting the Micro:bit.

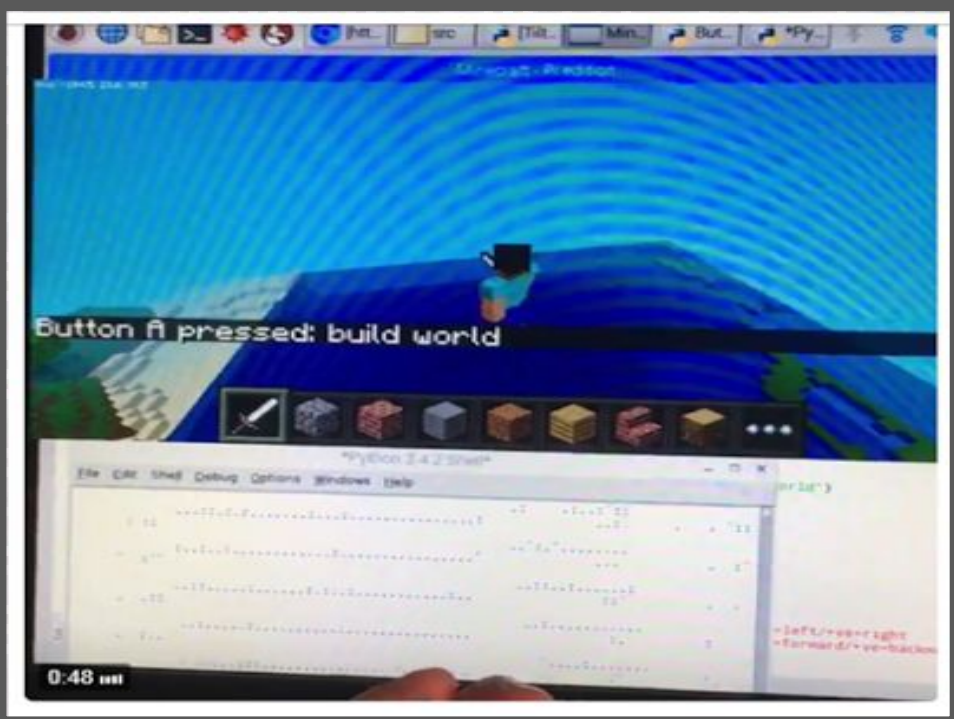
Here is a video to show this in action:

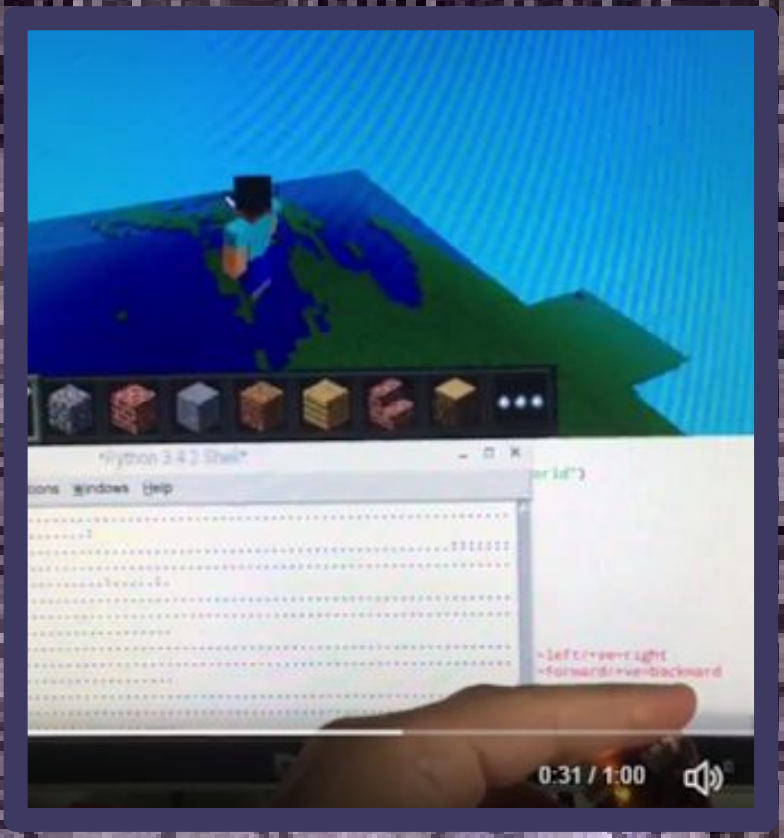
https://twitter.com/ncscomputing/status/885213278007328772

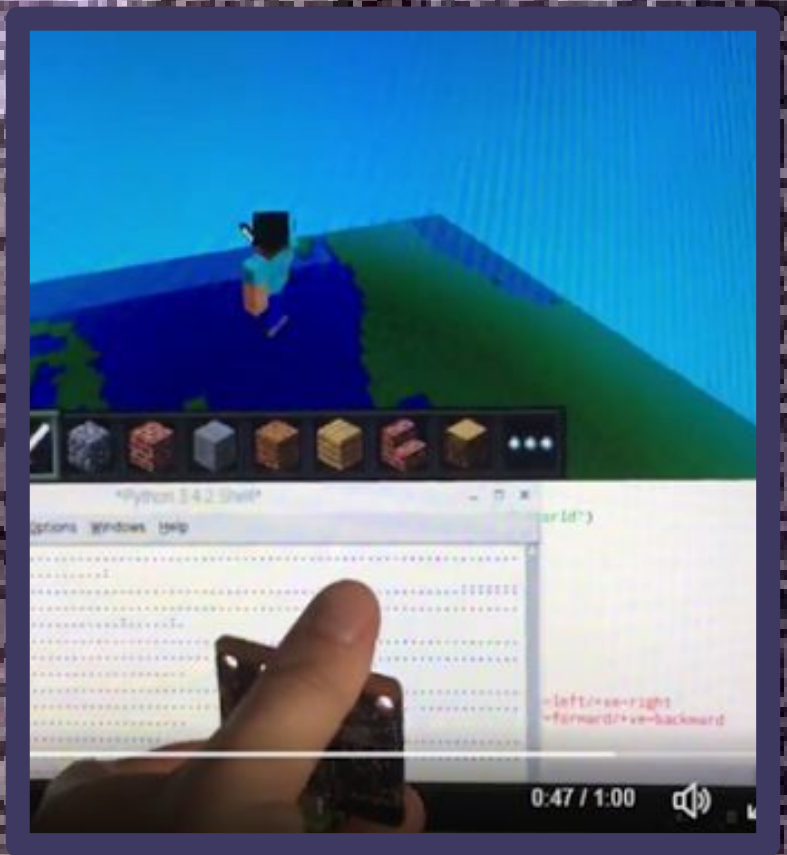
Section 2 of the code:
If you press the B button then the character teleport's to predetermined locations on the map. Here is a video of this in action:

https://twitter.com/Warksraspijam/status/894492363330113540

Here are some static images:




0:31 / 1:00


0:47 / 1:00

The files you will need are here should you not want to code the BitIO python file:

Files:

World map data file:

https://github.com/ncscomputing/HpAnthologyV2/blob/master/world3.txt

Build world library:
https://github.com/ncscomputing/HpAnthologyV2/blob/master/BuildWorldDM.py

BitIO file:
https://github.com/ncscomputing/HpAnthologyV2/blob/master/blog4AWACC.py

If you do want to have a go at making the BitIO file then here are your instructions:

This is what the code will do:

Instructions

1. Ensure that you have read blog 1:
http://warksjammy.blogspot.co.uk/2017/07/blog-1-getting-started-with-bitio.html
and downloaded the BitIO master files from David Whales github

2. Go into the 'Bitio master' folder, find the 'src' folder and save 'World map data file' and 'Build world library' from the links above into that folder.

3. Create a new Python 3 script and type out the following:

Code

```
#“Written by : @ncscomputing /@warksraspijam”

from mcpi import minecraft as minecraft
from mcpi import block as block
from datetime import datetime
import time
import random
import BuildWorldDM as bw

###
"""
Written by @ncscomputing on top of the bitio produced by David Whale
https://github.com/whaleygeek/bitio
```

```
world building code imported library from Damien Mooney's blog:
https://damianmooney.wordpress.com/2016/02/16/raspberry-pi-minecraft-iss-trac
ker/

"""
import serial
from mcpi.minecraft import Minecraft
import time
from mcpi import block as block
import random
import microbit
mc = Minecraft.create()

# button.py - demonstrates using a button



print("micro:bit connected - press button A to test")
WoolList = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]

EngX = 1.0
EngZ = 50.0

UsaX =  53.8
UsaZ = 39.7

CanadaX = 59.0
CanadaZ = 61.3

IceLandX = 10.0
IceLandZ = 61.1


def Teleport(x,z,Country):
    mc.player.setPos(x,20,z)
    mc.camera.setFollow()
    mc.setBlock(x,1, z,35,random.choice(WoolList))
    mc.postToChat(Country)

while True:
    time.sleep(0.25)
    #=======buildworld call when a button pressed
```

```
if microbit.button_a.was_pressed():
        mc.postToChat("Button A pressed: build world")
        print("Button A pressed: build world")

        bw.Build()
        time.sleep(2)

        microbit.display.show("build world")


    # manual teleport using acceleromter readings
    pos = mc.player.getTilePos()
    x = microbit.accelerometer.get_x()/300 # -ve=left/+ve=right
    y = microbit.accelerometer.get_y()/300 # -ve=forward/+ve=backward

    pos.x += x # east/west
    pos.z += y # north/south

    mc.player.setTilePos(pos.x, pos.y, pos.z) # set player position

    #time.sleep(0.5)
    #=========
    if microbit.button_b.was_pressed():
        print ("Button B pressed: Manual teleport")
        mc.postToChat("Button B pressed: Teleport")
        time.sleep(0.5)
        Teleport(EngX,EngZ,"England")
        time.sleep(8)
        Teleport(UsaX,UsaZ,"USA")
        time.sleep(8)
        Teleport(CanadaX,CanadaZ,"Canada")
        time.sleep(8)
        Teleport(IceLandX,IceLandZ,"Iceland")
        time.sleep(8)
```

4. Save it as something.py in the same 'src' folder as the previous files.
5. Open Minecraft, create a new world.
6. Make sure your Micro:bit is plugged in to your PC/Pi.
7. Run the python file that you created above.
8. Try pressing the 'a' button let it build the world map.
9. Try using tilting the Micro:bit to see if you can navigate around the world map.
10. Now try and press the 'b' button and see if the teleportation works.

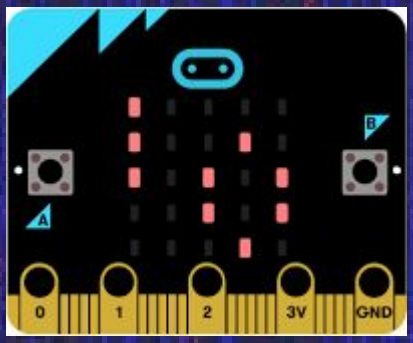
Extension:

Try extending the teleportation code to go to extra countries on the word tour.

Enjoy :)

You can now download a pdf version of these tutorials as part of the updated #hackpack resource booklet here:
https://github.com/ncscomputing/HpAnthologyV2/raw/master/Hackpack%20Anthology%20V2%200.3.pdf

Physical Computing with Minecraft 5: ‘Touch the pins……’

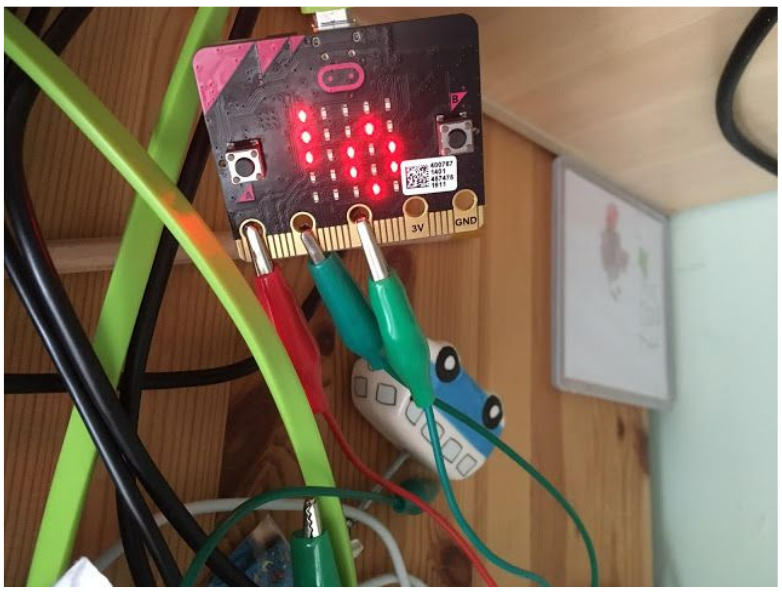


This takes the basic 'touched.py' demo from David Whales BitIO master available here: https://github.com/whaleygeek/bitio/blob/master/src/touched.py

As ever if you have not quite noticed may playground is Minecraft. So in this simple version you will drop different wool blocks according to the pin that you touch. To "touch a pin" you will need to touch the metal end of the cable attached to the pin and then touch the ground pin to complete the circuit and for the code to work. It took me ten minutes to figure this out.

Here is a link to a video of what touching pins basic code does: https://twitter.com/Warksraspijam/status/894975273124458496

Here is a photo of how the pins were set up during the video:



Here is the link to the simple version code:
https://raw.githubusercontent.com/ncscomputing/HpAnthologyV2/master/Touched%20Simple.py

Instructions:

1. Set up the 3 cables as above on the Micro:bit
2. Plug your Micro:bit into the Pi/PC.
3. Ensure that you have read blog 1:
http://warksjammy.blogspot.co.uk/2017/07/blog-1-getting-started-with-bitio.html
and downloaded the BitIO master files from David Whales github

4. Go into the 'Bitio master' folder, find the 'src' folder
5. Create a new Python 3 script.6. Call it "touched_simple.py"
7. Type in the following code:

Code

```
# touched.py - demonstrates using pin touch

import microbit
from mcpi import minecraft as minecraft
from mcpi import block as block
import time
import random

WoolList = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]


mc = minecraft.Minecraft.create()


while True:

    time.sleep(1)
    if microbit.pin0.is_touched():
        msg = "Pin 0 touched, Wool block colour 0"
        print(msg)
        mc.postToChat(msg)
        pos = mc.player.getTilePos()
        mc.setBlock(pos.x, pos.y-1, pos.z,35,0)
```

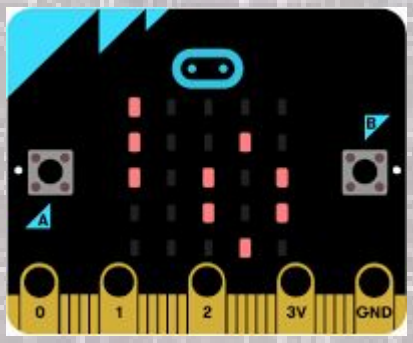
```
if microbit.pin1.is_touched():
     msg = "Pin 1 touched, Wool block colour 1"
     print(msg)
     mc.postToChat(msg)
     pos = mc.player.getTilePos()
     mc.setBlock(pos.x, pos.y-1, pos.z,35,1)
 if microbit.pin2.is_touched():
     msg = "Pin 2 touched, Wool block colour 2"
     print(msg)
     mc.postToChat(msg)
     pos = mc.player.getTilePos()
    mc.setBlock(pos.x, pos.y-1, pos.z,35,2)
```

8. Open Minecraft, create a new world.
9. Run the code by pressing f5.
10. Try touching pin 0, what happens.

Extension:
Try changing the block ID's

You can now download a pdf version of these tutorials as part of the updated #hackpack resource booklet here:
https://github.com/ncscomputing/HpAnthologyV2/raw/master/Hackpack%20Anthology%20V2%200.3.pdf

Physical Computing with Minecraft 6: ‘Touching pins....takes you places’



This takes the basic 'touched.py' demo from David Whales BitIO master available here: https://github.com/whaleygeek/bitio/blob/master/src/touched.py

As ever if you have not quite noticed may playground is Minecraft. This version builds on from the previous blog about 'touching pins' (http://warksjammy.blogspot.co.uk/2017/08/bitio-blog-5-touch-pins.html)

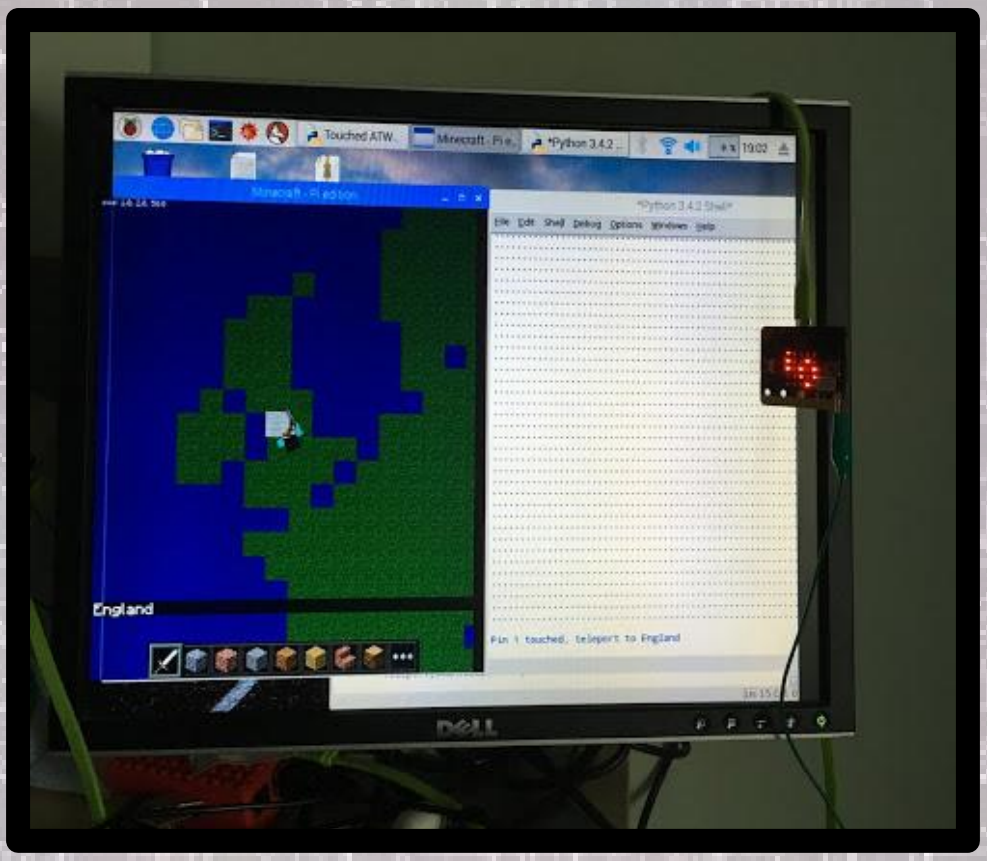
In this version you will touch a Micro:bit pin 0-2 and either:

- pin 0- build the world map in Minecraft.
- pin 1- teleport to England.
- pin 2- teleport to the USA.

To "touch a pin" you will need to touch pin 0 for example with one finger and complete the circuit by touching the 'gnd' pin and then the code will work.

Here is a link to a video of what touching pins basic code does: https://twitter.com/Warksraspijam/status/895355608404242436

Here is a photo of how it will look:



Here is the link to the simple version code:
Teleportation and pin reading code
https://github.com/ncscomputing/HpAnthologyV2/blob/master/Touching%20pins%20takes%20you%20places.py

Build world code:
https://github.com/ncscomputing/HpAnthologyV2/blob/master/BuildWorldDM.py

Data file:
https://raw.githubusercontent.com/ncscomputing/HpAnthologyV2/master/world3.txt

Credit Damien Mooney's world building code:
https://damianmooney.wordpress.com/2016/02/16/raspberry-pi-minecraft-iss-tracker/
Instructions:
1. Plug your Micro:bit into the Pi/PC.
2. Ensure that you have read blog 1:
http://warksjammy.blogspot.co.uk/2017/07/blog-1-getting-started-with-bitio.html
and downloaded the BitIO master files from David Whales github

3. Go into the 'Bitio master' folder, find the 'src' folder
4. Create a new Python 3 script.
5. Call it "touching pins takes you places.py"
6. Type in the following code:

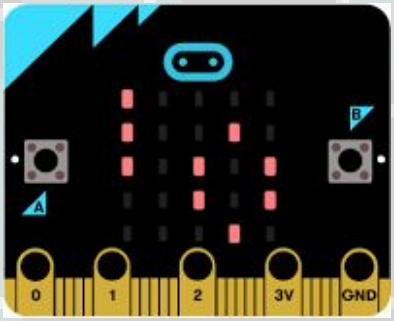
Code

```
# touched.py - demonstrates using pin touch
import microbit
from mcpi import minecraft as minecraft
from mcpi import block as block
from datetime import datetime
import time
import serial
import random
import BuildWorldDM as bw

WoolList =[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
EngX = 1.0
EngZ = 50.0

UsaX = 53.8
UsaZ = 39.7

mc = minecraft.Minecraft.create()

def Teleport(x,z,Country):
    mc.player.setPos(x,20,z)
    mc.camera.setFollow()
    mc.setBlock(x,1, z,35,random.choice(WoolList))
    mc.postToChat(Country)

while True:
    time.sleep(1)
    if microbit.pin0.is_touched():
        print("Pin 0 touched, build world")
        bw.Build()
    elif microbit.pin1.is_touched():
        print("Pin 1 touched, teleport to England")
        Teleport(EngX,EngZ,"England")
    elif microbit.pin2.is_touched():
        print("Pin 2 touched, teleport to the USA")
        Teleport(UsaX,UsaZ,"USA")
```

8. Open Minecraft, create a new world.
9. Run the code by pressing f5.
10. Try touching pin 0, what happens. Now try pin 1 and 2.

Extension:
Try adding in extra countries.
You can now download a pdf version of these tutorials as part of the updated #hackpack resource booklet here:
https://github.com/ncscomputing/HpAnthologyV2/raw/master/Hackpack%20Anthology%20V2%200.3.pdf

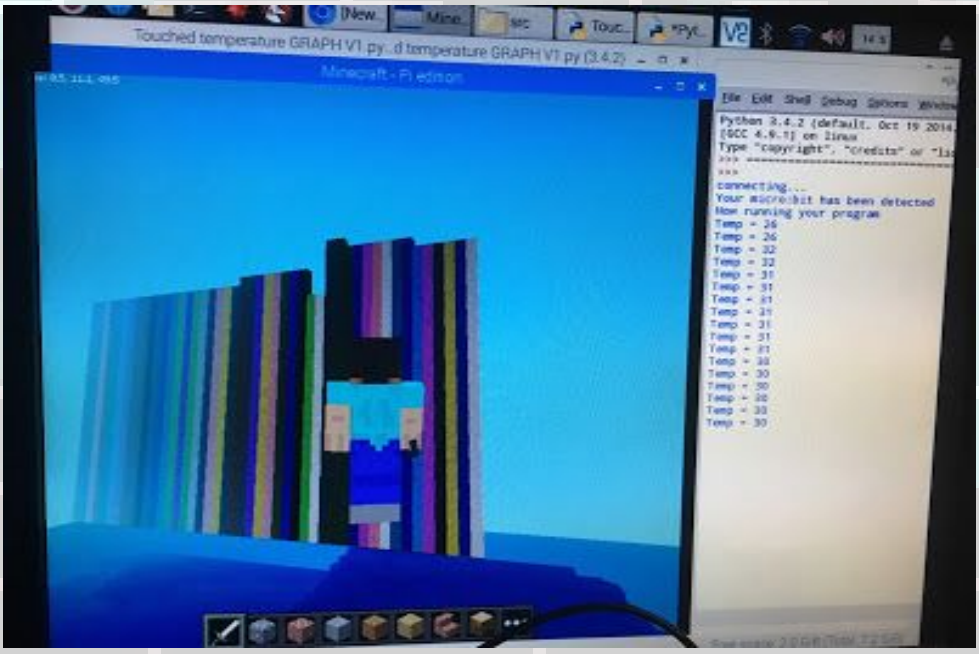
Physical Computing with Minecraft 7: Graphing live data in Minecraft



David Whale has been busy crafting more add on's to his BitIO library for the MIcro:bit. A few days ago he added the Micropython function to check the CPU temperature. His library can be accessed from here:

https://github.com/whaleygeek/bitio/
Here is the link to the code:
https://raw.githubusercontent.com/ncscomputing/HpAnthologyV2/master/Minecraft%20Microbit%20CPU%20temp%20graph.py

I have built / recycled Minecraft graphing code to create a CPU temperature monitor. Every time you press the a button it takes a temperature reading and then out puts this to Minecraft as a random coloured wool bar. Here is what you end up with:



Instructions:

1. Open python 3 on the Raspberry Pi.
2. Plug your Micro:bit into the Pi/PC.
3. Ensure that you have read blog 1:
http://warksjammy.blogspot.co.uk/2017/07/blog-1-getting-started-with-bitio.html
and downloaded the BitIO master files from David Whales github

4. Go into the 'Bitio master' folder, find the 'src' folder
5. Create a new Python 3 script.6. Call it "Minecraft CPU temperature monitor.py"
7. Type in the following code:

Code

```
import microbit
from mcpi import minecraft as minecraft
from mcpi import block as block
import time
import random

WoolList = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]


mc = minecraft.Minecraft.create()

orx,ory,orz = mc.player.getPos()

mc.postToChat("Start Graph")

TempBlock = 35,14

HumidityBlock = 35,3
Temperature_List = []#stores temp data

DataStreamCount= 0

def BuildDataBlockTemp(ImportedBlock):# take data for temp
    temp = int(microbit.temperature())
    Temperature_List.append(temp)
    orx,ory,orz = mc.player.getPos()

    for i in range (0,temp):
        x,y,z = mc.player.getPos()
        mc.setBlock(x+30,i,z,35,ImportedBlock)
    mc.player.setPos(orx,ory,orz+1)
    msg = "Temp = {0}".format(temp)
```

```
    print(msg)
while True:

    time.sleep(0.25)
    if microbit.button_a.was_pressed():
        TempBlock = random.choice(WoolList)
        BuildDataBlockTemp(TempBlock)
```

8. Open Minecraft, create a new world.
9. Run the code by pressing f5, checking for any errors that arise.
10. Press the a button on the Micro:bit.

Extension:
Try automating the graph so that it carries on every 10 seconds

You can now download a pdf version of these tutorials as part of the updated #hackpack resource booklet here:
https://github.com/ncscomputing/HpAnthologyV2/raw/master/Hackpack%20Anthology%20V2%200.3.pdf

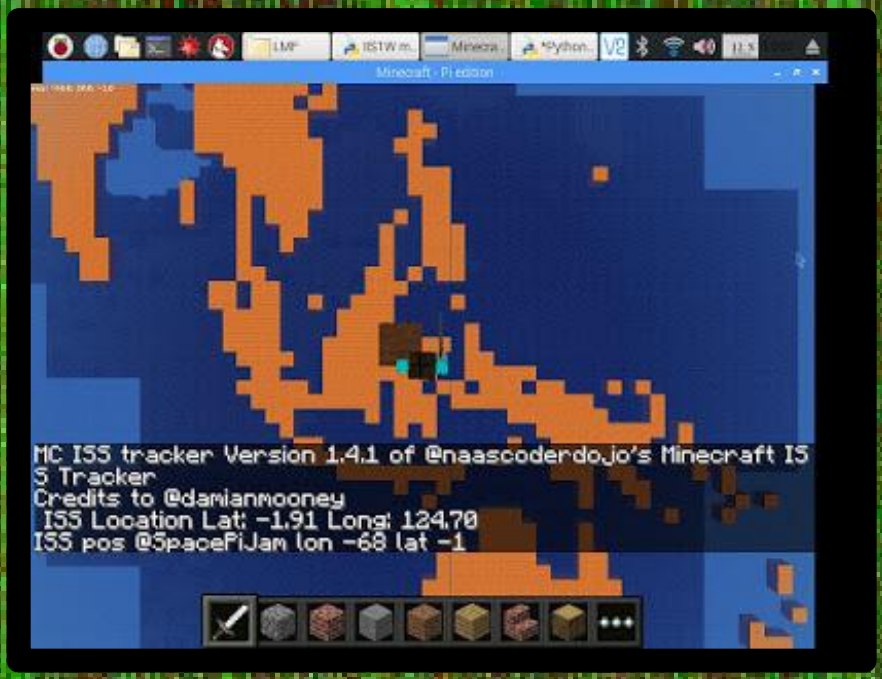
24 hours tracking the ISS in Minecraft and Twitter

During the last week or so I have been gradually tweaking my spin on Damien Mooney's ISS tracker. link https://damianmooney.wordpress.com/2016/02/16/raspberry-pi-minecraft-iss-tracker/

My bolt on is to take a screenshot record of each ISS movement and then tweet that picture from a twitter bot that I set up (@warksmessabout1).

This has worked for a while in trials, but has always fallen flat after a few hours. By trial and error I have hit upon a version which has run for 27 hours continuously updating 4 times an hour.

Here is an example screenshot of one of the recent updates from earlier today:



Here is the main twitter bot code:

```
"""
Tweeting ISS current position at set intervals with Minecraft Pi.
Get current ISS position from http://wheretheiss.at/ and map it on
a raspberry pi with minecraft
Damian Mooney wrote this minecraft Tracker in 2016. I have added a few
features on top, namely:
walking random wool blocks to mark current and positions
ability to screenshot the position using Martin O'Hanlons raspi2png tutorial:
(http://www.stuffaboutcode.com/2016/03/raspberry-pi-take-screenshot-of.html)
ability to tweet that using twython python library
"""
__author__ = '@damianmooney' #additions by @ncscomputing
from mcpi import minecraft as minecraft
from mcpi import block as block
from datetime import datetime
import time
import urllib2
import json
import random
import subprocess # to run shell script to autocall raspi2png
import sys
from twython import Twython
import BuildWorldDM as BuildWorld
```

```
consumer_key = ''
consumer_secret = ''
access_token = ''
access_token_secret = ''

api = Twython(consumer_key,consumer_secret,access_token,access_token_secret)

WoolList = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]

""" call where the iss at api thanks to Bill Shupp"""
def getiss():
    response =
urllib2.urlopen('https://api.wheretheiss.at/v1/satellites/25544')
    mydata = response.read()
    return mydata

""" longitude: convert our longitude to a minecraft co-ordinate"""
def do_coord(longitude):
    mine_long = longitude * -.55
    return mine_long


#if __name__ == "__main__":


mc = minecraft.Minecraft.create()
mc.postToChat(" Minecraft ISS Tracker for @naascoderdojo")
mc.player.setting("autojump", False)
mc.player.setPos(6, 20, 50)
time.sleep(10)
while True:


   try:
        BuildWorld.Build()
        time.sleep(5)
        iss = getiss()
        pos = json.loads(iss)
        lat = pos['latitude']
        lon = pos['longitude']
        mc.postToChat("MC ISS tracker Version 1.4.1 of @naascoderdojo's
Minecraft ISS Tracker")
```

```
        mc.camera.setFollow()
        mc.postToChat("Credits to @damianmooney")
        mc.postToChat(' ISS Location Lat: %.2f Long: %.2f' % (lat,lon))
        new_long = do_coord(lon)
        mc.player.setPos(int(new_long), 20, int(lat))
        mc.setBlock(int(new_long), 20-1, int(lat),35,random.choice(WoolList))
        msg = 'ISS pos '+'@SpacePiJam'+' lon %d lat %d' % (new_long, lat)
                    #print msg
        mc.postToChat(msg)
                    #ory to location that you havesaved raspi2png in

        a=subprocess.check_output('./raspi2png -d 3 -p
"myscreenshot.png"',shell=True)
        photo = open('myscreenshot.png', 'rb')
        time.sleep(8)
        api.update_status_with_media(status=msg, media=photo)
    except:
        mc.postToChat("update issue")
        a=subprocess.check_output('./raspi2png -d 3 -p
"myscreenshot.png"',shell=True)
        photo = open('myscreenshot.png', 'rb')
        api.update_status_with_media(status="update issue", media=photo)

    time.sleep(450) # --only update once every 15 minutes
```

It reads in and renders the world text file from here:
https://raw.githubusercontent.com/ncscomputing/HpAnthologyV2/master/BuildWorldDM.py

The data file is here:
https://raw.githubusercontent.com/ncscomputing/HpAnthologyV2/master/world3.txt

Enjoy :)

The End...

The Minecraft Map Making community has once again come together to assemble a unique collection of articles for your reading pleasure in this 8th edition of Map Making Magazine. Finding and following the authors of your favourite articles can be a unique way to complete your own Twitter monument, so give it a go before you close this issue out.

Complete the Monument maps are a mainstay of the Minecraft scene. We hope that in this issue you were able to pick up some tips and tricks on how to create your own wooly scavenger hunt. Remember to track down a CTM map you have read about in this issue and play it through with a critical eye on what is fun and what works best. In the best tradition of Map Making, you can take all these ideas an combine them into your own CTM map for all of us to enjoy!

This month's MapMag included a hunt for 16 wool blocks. How did you do? If you have found them all then let us know on Twitter @MapMakingMag for a retweet and boost.

Of course we have also shared stories about master map makers, as well as map making topics like how to create parkour city scapes and how to work with the new Achievement system. Past issues of Map Making Mag are all available for you to explore as well, so make sure you take a look at http://www.testfordev.com/MapMag

MapMag is an Open project and attempts to involve the community as a whole through compiling and publishing articles from anyone who wants to tell a story or share their skills and expertise. This includes you: get involved!

COMPLETE THIS MAP MAKING MONUMENT

White	Orange	Magenta	Light Blue	Yellow	Lime	Pink	Gray
Light Gray	Cyan	Purple	Blue	Brown	Green		Black

Have you found them all?

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

About the Magazine

This project is a community driven and contributed magazine. By publishing we seek to develop the wonderful craft of Minecraft Map Making. All content remains the property of the respective author and is used with permission. All trademarks referenced in this publication remain the property of the respective trademark holder.

Last Issue Errata

Issue 7 - Some links were not clickable due to articles imported as graphics. We will do better!

The Map Mag Team

MapMag includes Articles and Art from:

@abrightmoore
@lemoesh
@MCAdecia
PearUhDox
DjEar
@jigarbov
@Anistuffs
CreeperMagnet_
@_Cavinator1_
@InterlacedMinds
    The Rahn
    Oliver Hynds
    BhunaBoy
@ncscomputing

Assistance in this production was provided the CTM Community

@... your name could be here - write an article or provide art for future editions! See submission guidelines in The Lobby.

This publication is a community effort and this issue has been compiled with input from the Minecraft Map Making community.

MapMag is supported by donations from: @immersiveminds and @cocoamix86

The Free Independent Minecraft Map Maker's Magazine

MAPMAG

ISSUES 1-7

http://www.testfordev.com/MapMag

@MapMakingMag | MapMakingMag@gmail.com