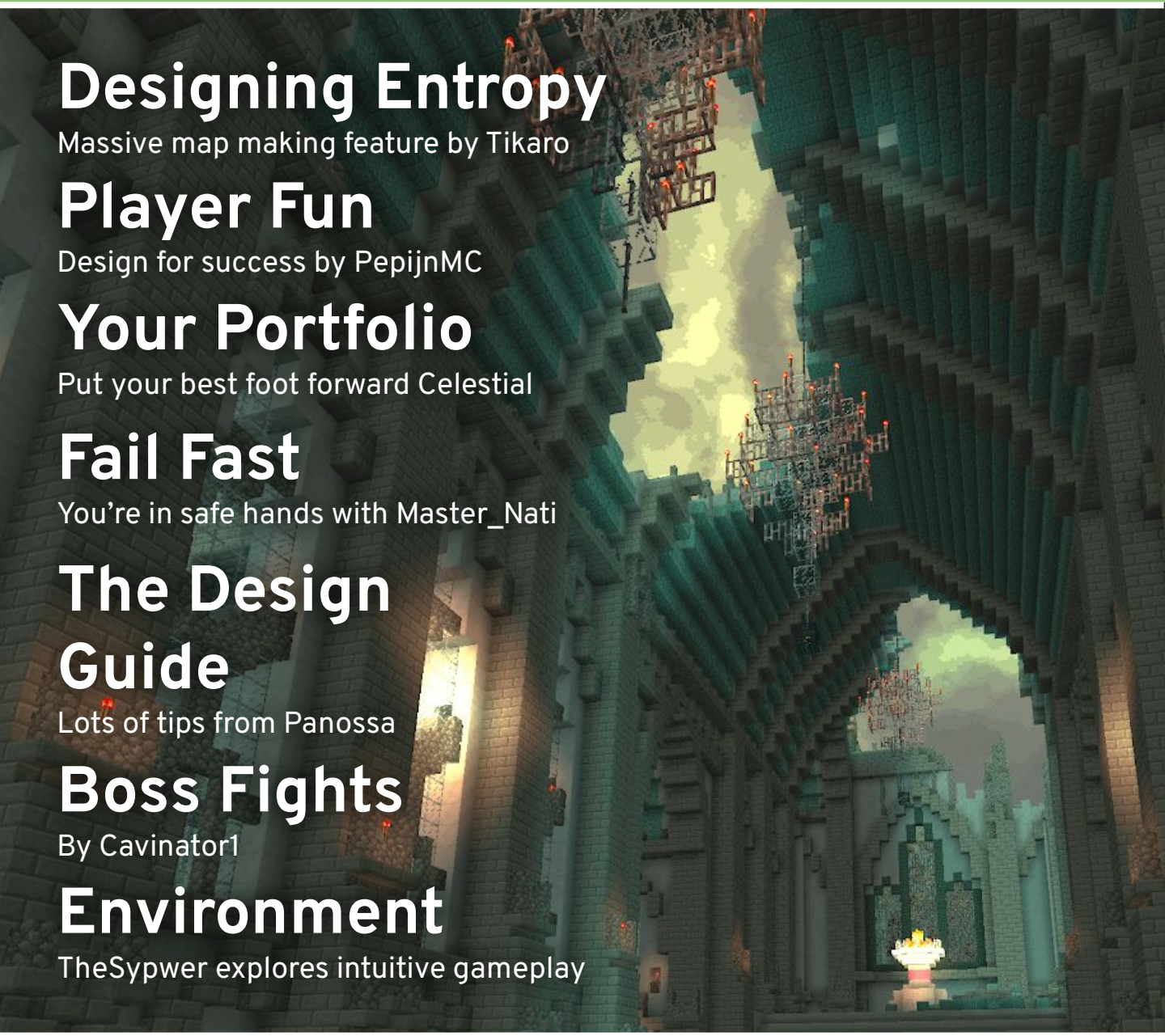
The Free and Open Independent Minecraft Map Maker's Magazine

MAPMAG

ISSUE 10 - DESIGN



Designing Entropy
Massive map making feature by Tikaro

Player Fun
Design for success by PepijnMC

Your Portfolio
Put your best foot forward Celestial

Fail Fast
You're in safe hands with Master_Nati

The Design Guide
Lots of tips from Panossa

Boss Fights
By Cavinator1

Environment
TheSypwer explores intuitive gameplay

Cover Art by Tikaro

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

Contents

Image by @theCommonPeople

The Lobby

Design in Minecraft is the art of thinking and planning what your map will look like and how it will behave. Given the large number of maps we are all 'working on' without release dates, it seems like we are all doing a LOT of design!

This issue hopes to move you past whatever is holding up your next big content release by offering tips and tricks that you can use in your map design process. Tikaro's recent Complete the Monument mega-map Entropy features a thorough post-mortem with ideas on what worked well, as well as considerations of what didn't work so well. Download the map to explore the concepts in detail once you read the article and then let us know how your own map designs are going.

We also have a bunch of helpful material from the community packed into this issue. This extends the great work by Map Makers around the globe in issues One through Nine, so if you are new to MapMag make sure you leave some room to catch up on what's gone before.

Remember, this is YOUR magazine. Feel free to get involved in each issue. Simply look for the pinned tweet on Twitter's @MapMakingMag and start typing!

- @abrightmoore (Editor and publisher)

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.


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

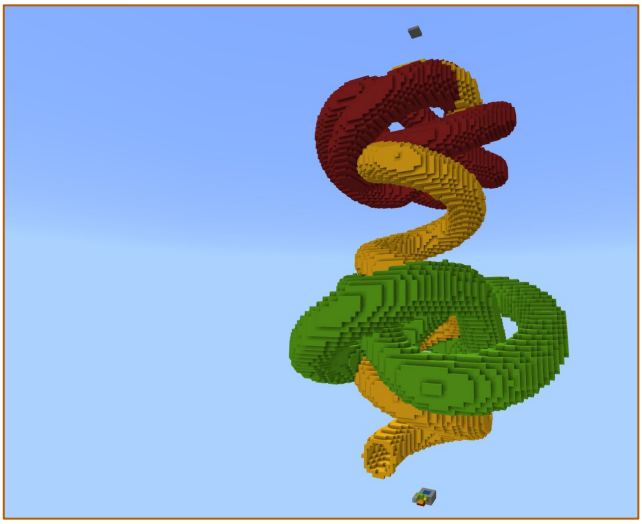
Minecon Earth 2018

MCEdit Dropper … 25/09/2018
Adventure 0.55MB

One of the perks of making maps is that you can get to talk about it with some of the most fascinating people in the world. Map Mag offers a great opportunity to start a conversation, and between issues there are a huge number of Minefaire events around the world to attend as well. Once a year, however, things get truly exciting with Mojang's premiere Minecraft event: Minecon Earth!

Introduced in 2017, Minecon Earth replaces the (almost) yearly convention called Minecon. Minecon Earth is a global stream of onstage live interviews and activities, with transitions of scenes covered by prepared video segments. In 2018 Mojang increased community participation through the use of a large number of well known streamers as hosts, and by adding community interest panels to the official Team Mojang YouTube channel after the stream concluded.

Your intrepid editor tagged along to the Minecon Earth event in Boston to help out with the community panel on "Making a Map with MCEdit". Along with Podshot, Naor, Trazlander, and GentleGiantJGC we introduced MCEdit to new map makers, gave away a free Dropper map on Bedrock and Java, and talked about the development roadmap. You can watch the panel recording and get the map here: https://www.youtube.com/watch?v=f466vaGBx-0



Community Map Challenge

The mapping panel team would like you to create your own Dropper level to include in a community contributed map to be released later in 2018. Simply create your own Dropper level in Java or Bedrock and send it through to the MapMag team. We will then compile the submissions into a single map for everyone to play.

Try not to make your map too complex as far as mechanics go since we'll need to port it between Bedrock and Java.

For inspiration, check out "The Library" by @rsmalec. In this map 50 levels were brought together by legendary Map Maker Ron Smalec in 2013.

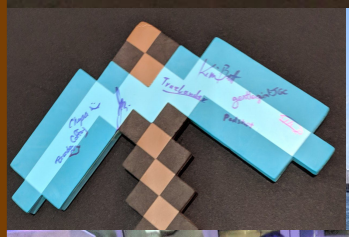
Also try the biggest Dropper maps in history by Bigre: "The Dropper" and "Dropper 2".

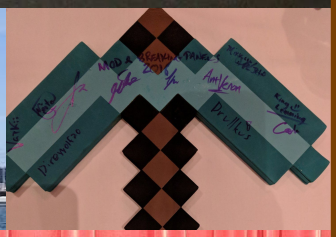
Level 10

- Create a level of your own by Nov 30 2018
- Use MCEdit or any other tools.
- Java Edition or Bedrock Edition worlds.
- We will make a community dropper map
- Publicly release for free later in 2018.
- Email it to MapMakingMag@gmail.com tweet at @MapMakingMag
- (You can enter more than one level if you like)

Minecon Earth 2018 Happy Snaps

The MCEdit development team of @Trazlander, @Naor2013, @GentlegiantJGC, @Podshot_ at the recording



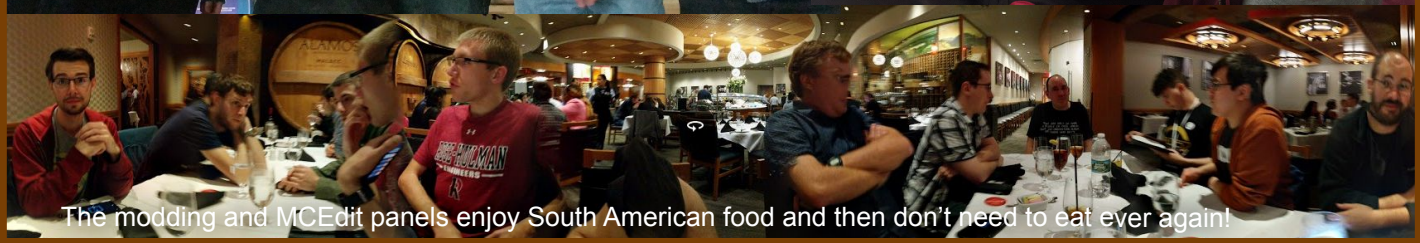


After rehearsal catch-up

@Chupacaubrey, and the gang, pose casually

Pathway proprietors @neonerz and Nickflame joined the fun

@kingbdogz, Jaryt, and @gentlegiantJGC explored Boston from the sky!

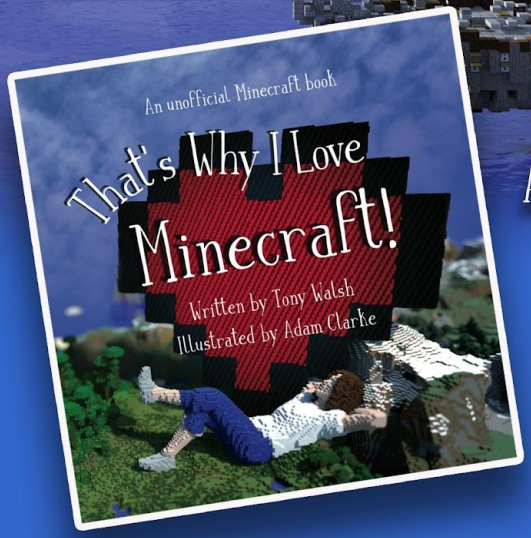


The modding and MCEdit panels enjoy South American food and then don't need to eat ever again!

"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

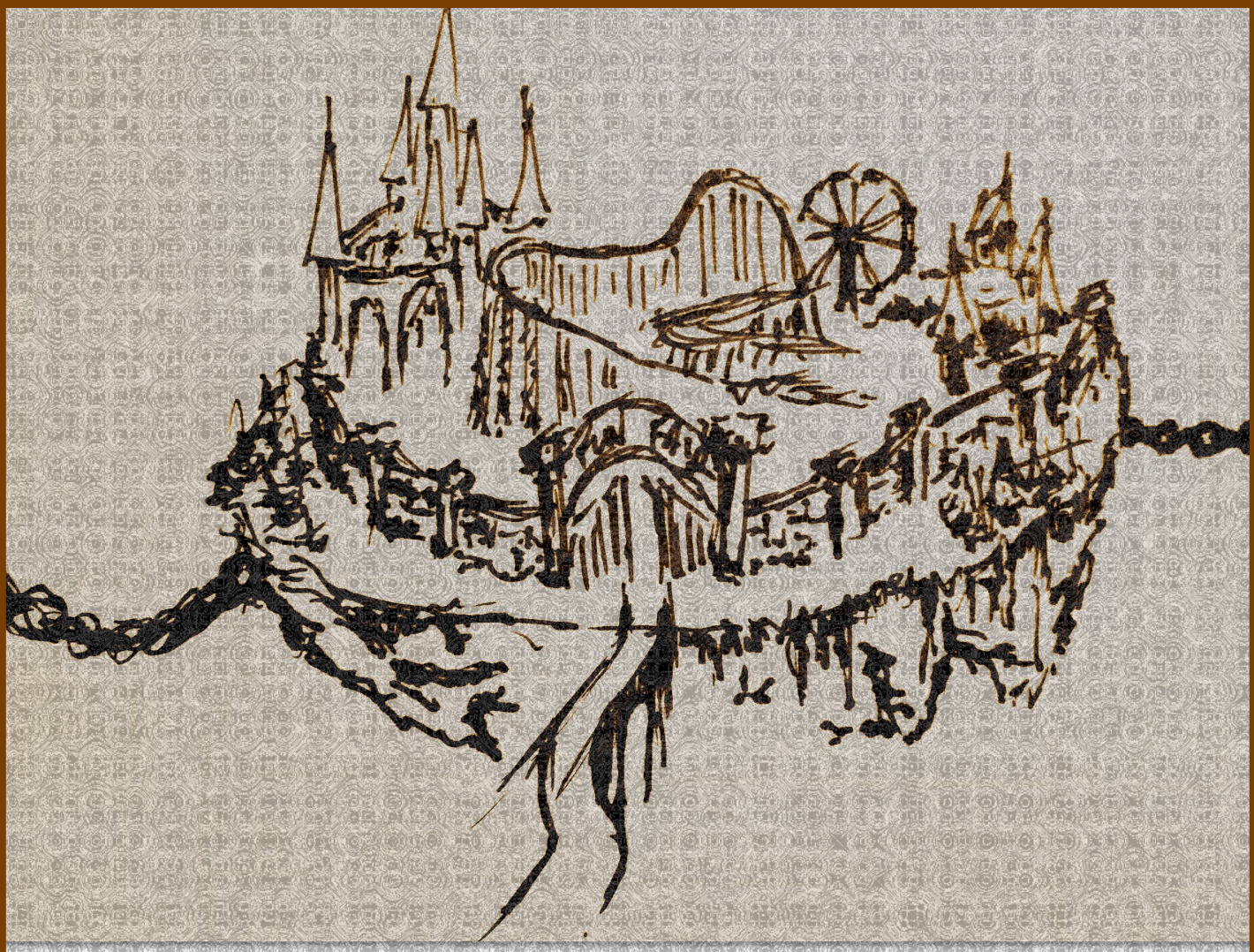
Written by Tony Walsh
Illustrated by Adam Clarke

Published by The Wizard and The Wyld

Paperback £10.00

www.why-i-love-minecraft.com

The Wizard and the Wyld



THE DESIGN ISSUE


FLAMING SWORD
Shaking
The Old Tunnels
TOWER OF THE FLAMING SWORD
Ruined Hardreach Castle
Puzzle to open
The Witch's House
Hidden Path
Hostage
Maze
Village
During Crisis
- Few Villagers
1. "Hostages"
2. "The witch"
3. "Old mine"
4. "The Attack"
After Resolution
- Villagers with quest
Garrisons
Overland
Loot
Alarm!
Arise!


Designing
Entropy
By Tikaro

A Playable Morbid Candy

Post-mortem on Entropy (by Tikaro)

Quite recently, I have released my last and biggest Minecraft project: a CTM map called "Entropy" (which, to those interested, can be found here). Writing a post-mortem was something I wanted to do after the project was finished, and I'm grateful that Abrightmoore has offered me a good place to do it.

A little preface: I will talk about the design decisions made during the development, their consequences—both positive and negative—and give advice on what can be done better if you find yourself in similar situations. I will not give how-to guides on building in certain styles or making specific things or anything of that sort.

With that out of the way, let's talk about CTM maps. Most of the ones I'm familiar with have the same core structure (or close to it): take X self-sustained areas, glue them together, sprinkle with a bit of solid gameplay and a good idea or two—voila, there's a CTM. I don't mean to belittle the effort that goes into creating even a simple map: it can take months of arduous work! But it's an important note that I wanted Entropy to remain unique and recognizable long after I have moved on from Minecraft. With that core goal in mind, the first good decision was made: Focusing on aesthetics.

Good decision: Focusing on aesthetics

I am very proud of Entropy's visuals. They were intended to be stellar, and I'm not gonna blush while saying that they are.

Lots of things go into a map's visual design—but building style is one of the most important ones. Entropy's terrain is nothing to write home about—I've never had much interest in making natural landscapes. So the terrain could have been given a bit more polish at a couple of points, but it does its job well enough.

The "job" in question is being the backdrop for all of the structures (which I'm a lot more fond of). Castles, citadels, and cathedrals are the lifeblood of Entropy. Most of them are big, some even unreasonably so. The aggressive use of lots of pseudo-gothic architecture in the second half of the map was a particularly good choice: seeing as this is not a common style in CTMs, it added to the singularity of the visuals and strengthened the atmosphere.

I have learned a lot about making structures during the development cycle. While it's not the main goal of the article, I want to share two guidelines I think to be paramount while creating any large man-made structure:

Start with interesting shapes. It's awfully difficult to make a gigantic box look interesting, no matter how much building experience you have.
Simple details often work best. There is a point when something becomes so detailed, it degrades into a bunch of visual noise. An enormous castle's facade must not look like a bowl of oatmeal from afar.

Numerous structures aren't the only thing Entropy's visuals have going for it. Aggressively using a resource pack was what brought it all together. Lots of textures were tweaked to be less saturated or to have a different hue, making some blocks work well together when with Minecraft's default textures they would look questionable at best.

From the very start the map was designed with an open sky in mind: most areas are either completely on the surface or have lots of skylight. This yielded two benefits. First, the map feels bigger and more open than it actually is. Second, the sky itself could be used to augment the visual design. Optifine, which is required to play the map, allows resource packs to add their own skyboxes (and with quite the sophisticated settings). Seeing that the player sees the sky almost constantly, this was an easy way of adding a lot more unique taste and atmosphere.

So, to wrap this point up: focusing on aesthetics during the development ensured that the map turned out visually stunning and memorable. Extensive texture pack use also made sure that it would stand out from the rest of the maps in how it felt from the screenshots alone.

Good decision: Building a strong atmosphere around a single theme

Entropy doesn't flirt with many different themes simultaneously, it sticks with one and runs with it. That theme is a "decaying world, derelict of a greater past". It's not a unique theme—in fact it's quite a popular one nowadays—but it doesn't have to be unique to work.

To establish the kind of depressing atmosphere befitting such a theme, visual design was paramount. Huge ruined castles, vistas of dead seas and barren mountains—all were intended to give the player a feeling that he is somewhere where structured, ordered human life is no longer welcome. Add a bleak, greenish tint to everything (even the light!) and grim, cloudy skies—and the world loses the lighthearted high fantasy feel Minecraft innately possesses.

The lore of the map plays an important role in establishing atmosphere as well. Presenting it as vague scraps of information in item descriptions made sure that the player lacks any well-structured knowledge. This leaves them hanging in the constant air of uncertainty, which fits the already oppressive visuals, (that is of course if we assume that the player cares about the lore, which is not always the case).

Lastly, the loot design. This is less prominent in the second half of the map, but I made a conscious effort to limit player's resources and make most upgrades feel, at least at the start, scarce and precious. Having no infinite safety bag of resources to fall back to means that the player will think twice before doing something completely reckless. I don't think I have struck the perfect balance with the proportions of "fodder" items and precious items, but overall the loot does its job in adding just one more slight touch to the atmosphere.

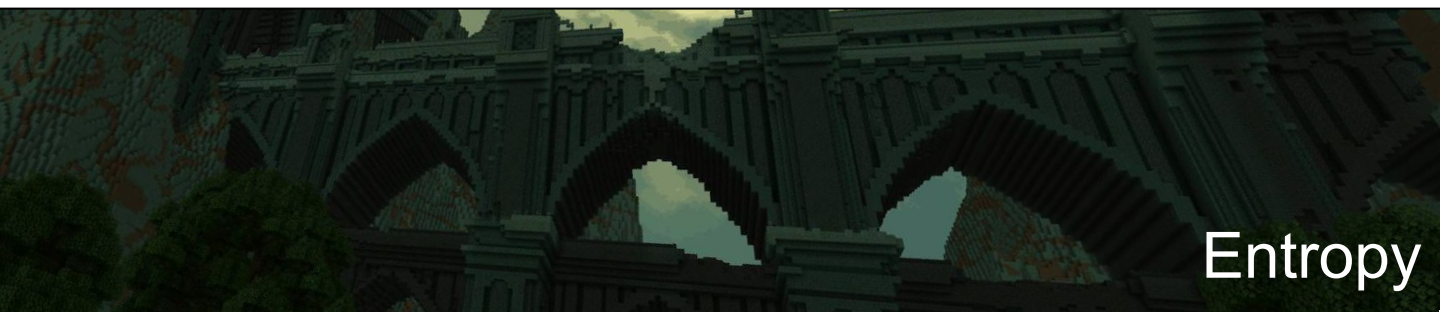
Good decision: Emphasizing continuity of the world

A lush rainforest five minutes away from a molten volcano and seven from frozen mountain peaks is, you will have to agree, a bit ridiculous. Yet this is normal for CTMs, seeing as most areas are self-sustained and isolated from each-other, glued together only by the map's overarching theme and lore. I think I saw this trend going away recently, but it was definitely present during the time I was active in the map-making community.

And yes, Minecraft is a game, and geography of a map doesn't have to make perfect physical sense. Having illogical jumps between areas takes away some of the immersion, and it's something that's very easy to avoid while the development is still in the planning stage.

Because of this, I have put conscious effort to design Entropy's world to be consistent and logical in its geography. Out of 20-ish areas in the map, only two are "ripped away" from the main world, accessible only via a teleport. The rest are integrated into the overarching terrain and flow neatly one from the other. I would even go so far as to say that Entropy is not many areas making one whole, but one whole which can be logically segmented into many areas.

But, to be fair, there is one glaring point at which this geographical continuity is broken by necessity: Ghedagot's entrance has you teleported by "raising a ladder" to access the city's walls. This had to be done because Minecraft's height limit is only 256 blocks, and both Ghedagot and the preceding area make full use of it. (There's another advice that can be given from this example: know what to sacrifice when a sacrifice must be made. World continuity is nice and all, but here it was a heck of a lot easier to add a teleporter than to redesign two areas to fit neatly together)



Good decision: Giving out lore only through visuals and item descriptions

I've seen some maps use long sign boards or books to let the player know what the upcoming area is about. No matter how well-crafted your lore is, this kind of exposition does not take any effort from the player, only their time. The time they are not exploring the world you've built for them—and isn't that the reason lots of people play CTMs?

Entropy approaches storytelling from a different direction. A little bit is told through visuals alone, by showing you for example an abandoned carriage with some blood and a skull next to it. But this kind of storytelling is always subtle, in the background of the action. It's more of a spice on top of the main way the lore of the map is told—items.

Some items you find inside the map have a description, which consists of a sentence or two telling you some bit of information about the world. These kinds of "exposition items" are, in my opinion, a fantastic way to share your map's lore. They don't break the player's immersion as a long stretch of signs would, because they don't feel like some "outsider" narrator talking to the player. They also require the player to find them, which means they can be used as a reward for exploration. This both alleviates the problem of passivity the books and sign boards have, and gives the developer a bit more freedom in loot design.

There is a downside to "exposition items". If your player is eager to rush through and don't take pleasure in meticulous exploration—they will most certainly miss a lot of the lore that they would've seen if it were presented in books or sign boards. Make decisions accordingly.

Well, I have definitely sung enough praises to Entropy. Postmortems aren't just about the decisions were good. The second half of this article is about the decisions that went poorly during the development.

Bad decision: Sticking with an overly-ambitious plan

The map is huge. It's not breaking any records in gameplay duration or objective count. But I'm sure it's close to breaking some in sheer physical size.

The decision to make the map so big was fueled by several ambitions. First—not shortly after I began the development process, I realized that I would not want to make another big CTM map. Therefore, there was a desire to one-up everything I already made, to make a crowning "definitive" map that outperformed any of my previous works in everything. Unsurprisingly, I was overly zealous in judging how large of a scale I could manage.

To really nail this problem home, consider this: the map isn't even as big as originally intended. The Black wool, traditionally the final objective of a CTM map, is entirely missing. This final area was supposed to be a lot bigger than anything seen in the map previously: an interminable, arduous journey through ruins so ancient they stop looking like ruins altogether. Sounds like a great atmospheric finale, right? Well, not so much when you begin considering that all of that has to be built by someone. So, after a bit of fiddling around with this idea, I decided to gut it all, since it would never be done in the way befitting everything already in the map.

The problem of scale didn't manifest itself just in the absence of the true final area. Ghedagot is awfully big, and it took a long time to make. So did the Citadel, and by the time I was done with both I was no longer happy or sad—just extremely glad it was over.

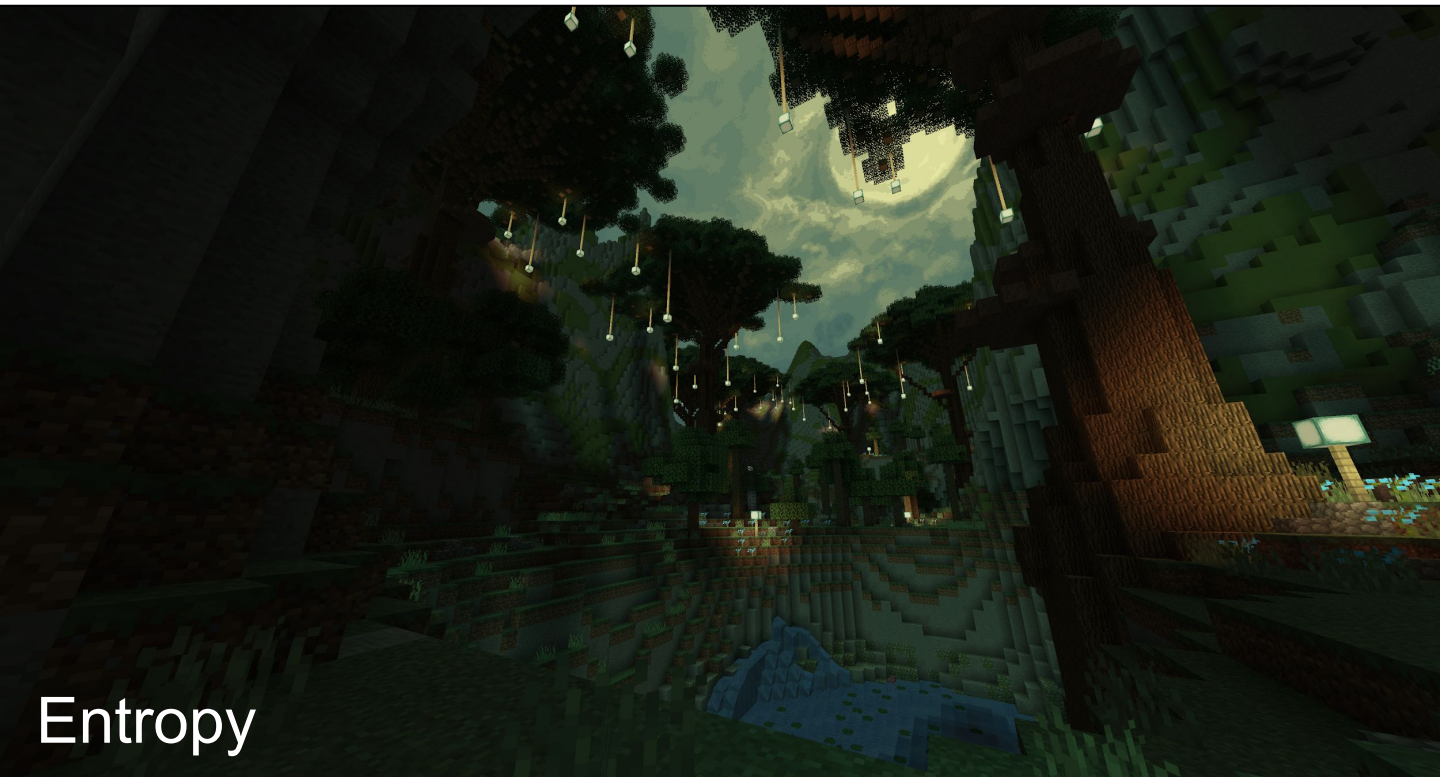
If I could remake the map with all the knowledge and experience I gained, I would definitely make it much more tight and compact. Cut the needlessly long transition areas, make the Citadel and Ghedagot smaller. Make everything fit together closer. Maybe add big vistas of inaccessible stuff to preserve the sense of grandeur. (To be completely honest: if I were to remake the map now, I wouldn't do it in Minecraft. It's an unparallel game-dev playground, but its limitations were a pain to work with)

Bad decision: OVER-focusing on aesthetics

It's natural that because I focused on aesthetics and atmosphere, other things had to give in. In hindsight, it was sadly the gameplay and mechanics I didn't give much attention to. The overall layout of the areas is good—they are varied enough and interesting to explore. Their placement in the world is also fine. But it's the very low-level things like clever traps and interesting new mechanics which are seldom present.

I attempted to fix these pitfalls much later, by implementing some band-aid fixes and polishing the map after it went through several test phases. But those fixes, of course, can't compare with what could have been made if gameplay was given more attention during production.

Speaking of which, the fact that the map is so big meant that it took a lot of effort to test. Because of that, it was not tested as much as it should have been—resulting in shakier balance. This, again, was partially fixed in the last stages of development.



My main advice for avoiding this kind of problem is: make sure your priorities are set straight. No matter your focus while making a map, be intimate with gameplay. The way making Entropy went was: lay out the areas and build all the aesthetics first. Add traps, enemies, and loot after most of the areas were visually done. Straight up—that's a horrible way to go about it! It separated designing visuals from designing gameplay—and that's a grave mistake, as gameplay is the most important part of a map.

The better way to go about it would be to build a rough prototype of an area, populate it with enemies and loot, then play-test and polish until gameplay was good. Only then should the rest of the aesthetics be put in place. After all, what player would care that the 120th column of this one giant castle isn't as broken as all of the others?

Bad decision: Leaving some gameplay mechanics underdeveloped
This was not just one big conscious decision, but the cumulative result of many small ones made over a wide time interval. Despite gameplay not taking top priority, the map does have several interesting mechanics—in idea when not in execution. Three come to mind: the travel system, the trinkets, and the advancements.

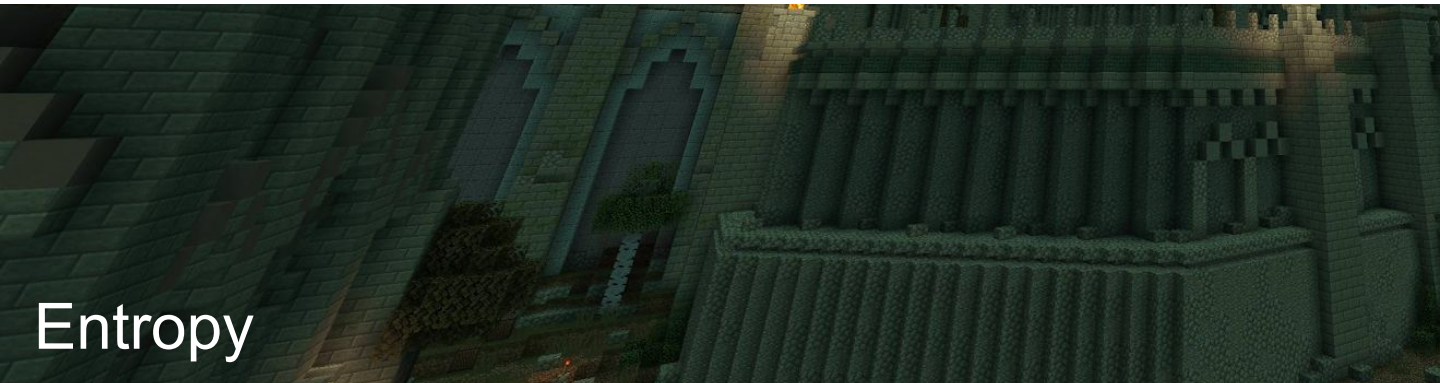
The travel system I think is the best of the three. The idea is simple: the player finds special waypoints as they advance through the map. They can fast-travel from one waypoint to any of those they have already found. Straightforward, right? Perhaps that's why I like it the most out of the three. The execution is robust and the system does only what it needs to do. One thing that could be improved is the placement of these waypoints. They already alleviate a lot of unnecessary backtracking, but there are still a couple of areas which are annoyingly far away for no particular reason.

If fast-travel is the best system in execution, trinkets are the most unique. A "trinket" is an item which you can place in two special slots in your inventory. Upon doing so, you would gain a passive benefit: a weak regenerating shield, or a constant debuff to all nearby enemies etc.

In theory, they sound great. In reality, the trinkets didn't turn out to be particularly useful, and are a bit buggy. Two of the three truly useful ones are found within the first fourth of the map, and the third one requires carefully exploring last areas, and at this point it's no longer useful since the map is almost over. These issues would be caught early on if the map was diligently tested throughout the development, which echoes back to the previous section.

Lastly, the advancements. I hesitate to call it a real "mechanic", as they are primitive and fairly pointless. The only advancements in the map are the story ones. Sure, they have pretty icons, but they betray the main idea of an achievement: to acknowledge some feat. Progressing through the map is not always a feat, it's expected behavior.

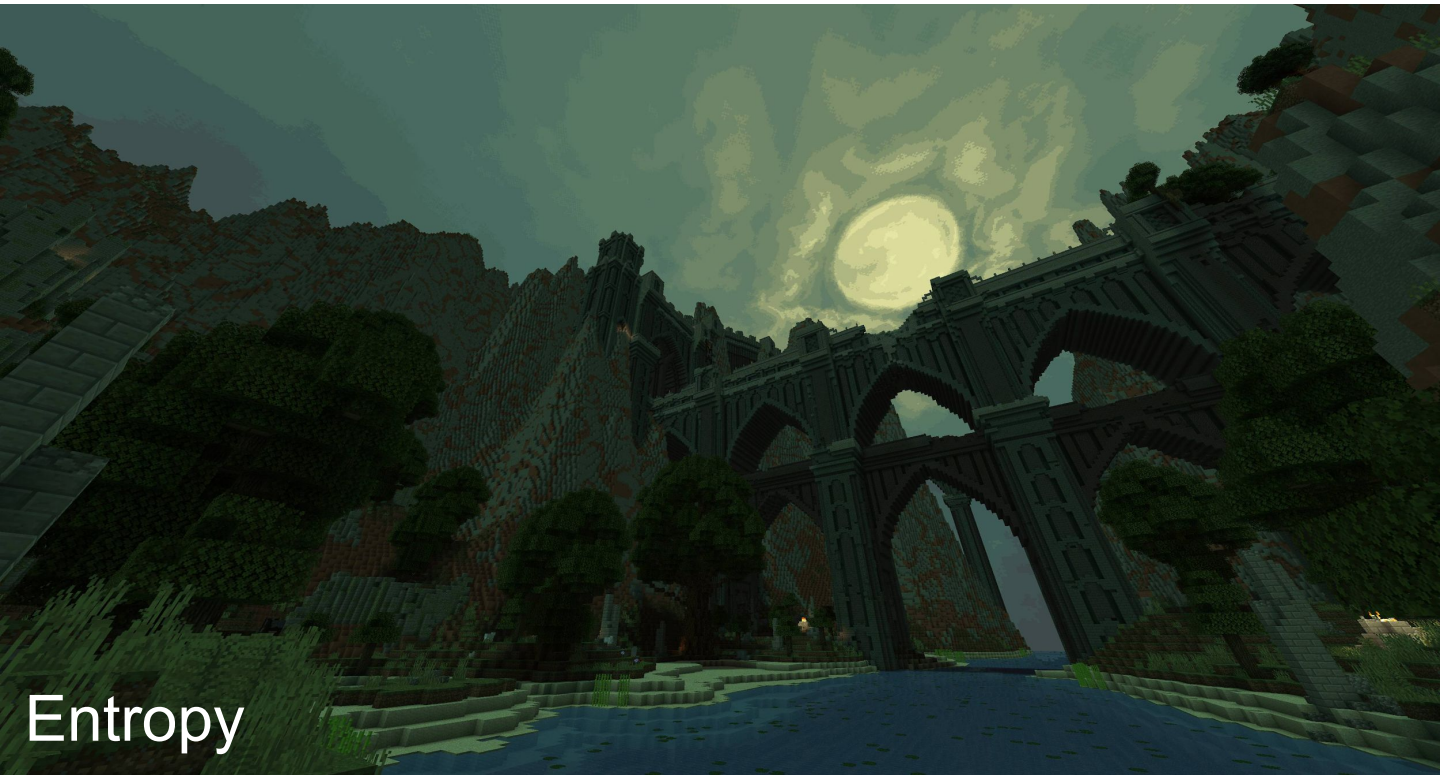
All of the mechanics-related issues arose from a lack of attention, due to deciding my time is better spent elsewhere. The way how this could have been avoided would be to play-test the map more frequently, which would bring to light that some mechanics weren't up to speed and thus allow them to be refined



Bad decision: Dropping the quality near the end of development

The development time was unreasonably long—I began in autumn of 2015, and the map was released on the verge of autumn 2018! Minecraft has unsurprisingly became stale during this time, and I found lots of new and more interesting projects to work on. But Entropy was never abandoned completely, due to me frequently thinking "well I'm too deep in now, I gotta finish this one way or another".

And "another" way it went—by the end of development my objective was to just get the map released by any means necessary. I've recruited all map-makers that I knew that were willing to help me to get the map playable and test it. I cut all the content I knew would take too much work to get up to speed. Basically the end of development was a rush to release, which of course meant a drop in quality.



You don't need to be told that this is not an ideal way to go about finishing a project. It was a "bad" decision for the final product's quality, but I stand by it—if I didn't do it, the map likely would have never been released, rotting away on my hard drive.

There's a valuable lesson to be found in this. Two, actually. First one is: be diligent in your planning and stick to it as closely as you can. Should I have been more realistic in what I planned this map to have, it would have been made a lot quicker and more polished as well. Second: if things are going south, don't be afraid to cut and destroy anything that you know won't make it. Aside from getting enjoyment and experience from the development process, your end goal is to publish a product. Don't get caught in a trap of endless polish because you think what you have created isn't "quite there yet".

Conclusion

In the end, Entropy was an incredible learning experience for me. It taught me how to approach truly big projects alone—and shown many pitfalls to avoid.

I'm content with the quality of the map. It has achieved everything it set out to achieve in terms of visuals and atmosphere. Its world is consistent, its lore is developed and interesting. And its rough corners will be a valuable lesson.

Thank you for reading, I hope you found a thing or two to take away from my experience.

DESIGNING YOUR **PORTFOLIO**

By Celestial from Saphire Studios

I've taken some time to come up with a few things that might help you make a good portfolio.

1. Don’t grab everything you’ve ever created.
Set aside time to go through all of your pieces, exclude anything you’re not proud of or don’t think is your best work.
I've found that what you put in your portfolio for people to view, you get in return; so be sure to put the things you want to do in your portfolio, and minimize the stuff you don't want to do. So if you want to be an in game terraformer, don't put mostly structural photos in there, but mostly terrain.

2. Select only your strongest pieces.
You don't want to clutter your portfolio with a lot of "meh" projects. Be critical about your own work, and only put the projects in the that you're proud of. And dare to say that it's one of your best.

3) Showcase your most unique and creative work.
There are a lot of people that do the things you're doing, so to have a better chance at getting into a team, you'd have to be different, make sure your portfolio shows this. It can be anything from a unique style, to an interesting mashup, be creative!

4) Showcase the design process.
(Concept art, block-outs, lay outs, etc)
This will generally help with getting into a team, the outcome is always nice to see, but seeing the process that you go through is even better! A lot of teams have a map that goes through a lot of different iterations, so to be able to adapt your work, is crucial if you want to succeed.

Teams can usually see if you're able to adapt well, if you show your entire process.

Making a Fun Game for Different Player Types

By @PepijnMC from Pixel²

Image courtesy of Lookout's Point render by Pixel²

How to make a “fun” game?

By @PepijnMC from Pixel²

Everyone likes different things, that’s no surprise. What might be fun for one person, could be boring for someone else. So it’s never a good idea to aim towards pleasing everyone when making a game, because it’s simply not possible. What you can do however, is try to follow some very general tips that should make your game more “fun” to play for most people.

First and foremost, the player should feel like they DID something, that their actions or decisions had some sort of effect. Games are all about interaction. You don’t have to go all out on this, but just make sure that throughout the game the player can perform some sort of meaningful action (a skilled jump or shot) or make some sort of meaningful decision (what gear should I equip?) that affects how they experience the game. It’s very hard to NOT have this in your game naturally, but it’s good to think about.

Related to the first point: if you have randomness, keep it fun. Pure randomness means the player has no control, it can be funny for a short while but it can get really frustrating really fast. It’s generally not a good idea to have pure randomness in your game, instead try to limit the randomness by introducing choices the player can make to increase their odds of a good outcome. A good mix of randomness and decisions can lead to a lot of replayability.

Last but not least, keeps things interesting while also familiar. Players need to feel like there’s more ahead, new areas/mechanics/secrets/etc to explore. But don’t overwhelm them with a lot of new stuff all at once. Keep it linked to what the player already knows, from before or maybe even from other well established games. This will make players feel engaged but not completely lost.

Again, these are just some general tips. This is not a step by step tutorial, but rather some topics to consider when making a game. There is no secret recipe for a fun game and I left a lot of details out, but luckily there are a lot of other articles to explore.

Art courtesy of Lookout's Point render by Pixel²

Types of players

By @PepijnMC from Pixel²

Every player is different. It's very obvious but it's a very important thing to realize and think about when designing a game. One person likes to have all the freedom in the world while others might get lost and prefer a more linear story. This is not something you can just outright "solve", you just need to accept and understand it. Know your (future) audience.

When making a stand-alone game, you have a lot of freedom when it comes to "picking" your audience and usually it's the group of gamers that you belong to yourself. You made the game, which hopefully means you like it, so like-minded people (when it comes to games) are more likely to enjoy it as well. This is probably the most simple approach, just make a game you like and it will attract people with the same general taste in games as you. The downside is that you might not really know how many people this includes. The upside however, is that it's generally very easy to understand what those people might want from the game, or from your other games, in the future. In the end it comes down what your goal is when making a game, which is a whole topic on its own.

But most of us aren't making a stand-alone game (yet). We work within Minecraft, which has its own audience. So whatever audience our game will have is a subset of Minecraft's audience. Luckily for us, Minecraft has proven to be a very versatile game with a very large audience worldwide and a lot of different playstyles. This goes for both the Java and Bedrock editions of the game. Age and nationality do play a role, Minecraft primarily has a young American audience, but that doesn't really define one playstyle. Kids don't just all like one thing and one thing only.

A tl;dr: If you enjoy making maps, don't worry too much about how many people will play your creations. If you like it, there's bound to be someone else who likes it. But how to make that person aware your map exists… is often the main barrier.

Art courtesy of Lookout's Point render by Pixel²

Intuitive Gameplay

Through Environment

By @TheSypwer

Intuitive Gameplay Through Environment

By @TheSypwer

How do you tell the player about how a specific mechanic works? There are many ways to do that; You could use tellraw, signs, floating texts… but for all of this the player would have to read and learn, this might be when a game gets boring. I think one of the most efficient and fun way of teaching is not teaching at all, but instead making the map so the player can learn by themselves.

For example: Let's say I'm doing a parkour map and i want the player to try not to fall on the floor or else they'll die, and the design is just like in this picture:

The problem is, the player has got no instructions teaching them falling on the floor is deadly.

So by making the floor red (a bright color that gives precise warning) you're telling the player not to fall on it.

Intuitive Gameplay Through Environment (Cont.)

Warning players about the deadly floor is not the only thing you can do with this design technique.

Now let's try using this technique again, for something else. Let's say i want to tell the player they'll get jump boost if they would step on the green block. This is a more specific thing we're trying to do now, we'll need to invite the players over the block first and than we should make sure the'll jump.

Here, I used green (a peaceful inviting color) for the jump boost blocks. The only way the player can go, is up there to the button. The player will most likely try jumping there but there's the chance that it's not self-explanatory enough, so we gotta tell the player green blocks give jump boost with another way.

Now, I have added a little icon using a retextured painting covered with green using the status effect icon of jump boost so that the player will understand green is attached to the jump boost effect.

There are many other ways you can use this technique in many other concepts. You'll just have to use your creativity.

Designing

Challenges

and

Boss Fights

By Cavinator 1

Designing and Balancing Challenges and Boss Fights in Adventure Maps.

By Cavinator1 (@_Cavinator1_)

Introduction:
Some may think that to make a good map you only need two things: Building and Command Blocks. Perhaps a story to give the map context. Maybe even a resource pack to give the map a unique look or to introduce in some custom models or textures.
But it's the way you use all these that really counts when creating a map - the ability to design challenges and boss fights for maps that are unique, fun, interesting, intuitive and most importantly, at the right level of difficulty.

As a mapmaker who has designed over 40 different boss fights across my three Assassin of Steve maps, over the years I have picked up a number of different tricks and tips on creating a good boss fight. When I first started out I made lots of rookie mistakes that ultimately led to certain boss fights being unbalanced or too hard, and after receiving feedback and watching numerous YouTube let's plays of people playing my maps, I learnt some valuable lessons in boss fight design.

In this article I'll mostly talk about designing a good boss fight for an adventure map, but a lot of what I say here can be adapted for other challenges within maps such as within the dungeon before the boss, or even if the 'boss' isn't necessarily a boss, but an objective, a goal to reach, such as perhaps the goal might be to infiltrate a facility without being seen by guards, and steal something, then escape.

The Boss:

When designing a boss fight, or indeed any challenge in a Minecraft adventure map, the first thing you should do is decide what it is. What are you fighting against? What is your goal? Your boss could be a custom mob with command-block assisted special powers. Or maybe it's something that you build, like a giant ghast, with attacks and weak points designed using command blocks and functions. There are very few limits when it comes to designing a challenge in a map, so your boss could be anything.

When designing a boss in Minecraft they generally fall into one of two categories - the first is that the boss is a custom mob, often with a large amount of health - a "mob boss".

The second main category is that the boss is something you build, and then the structure is given special powers using command blocks - a "built boss".

When coming up with a good boss, you might discover some interesting mechanic within the realm of commands and functions and decide to create the boss around it… such as that you can make a floating attack skeleton by summoning a skeleton with a bow and setting its NoGravity tag to 1.

There are two major components of any boss battle - the boss's attacks, as well as how you can attack and damage the boss.

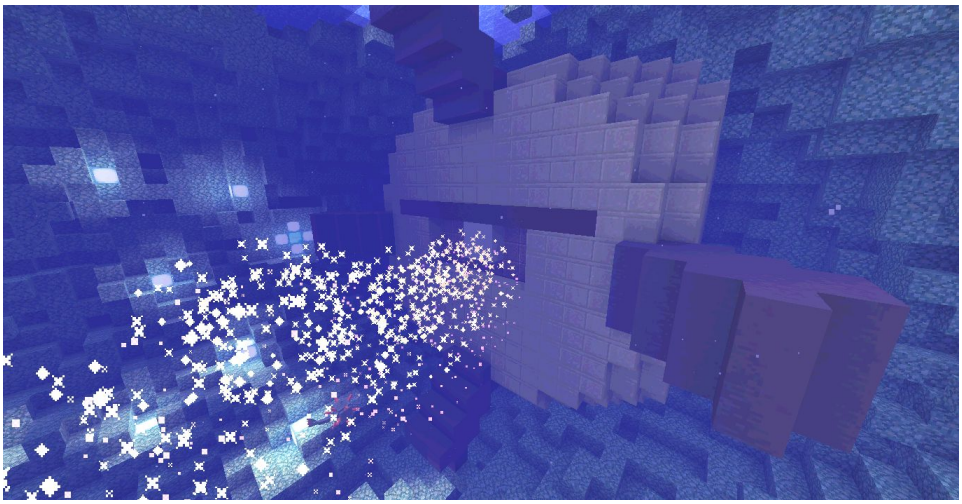
The Boss's Attacks:

What powers does your boss have? How does it pose a threat to the player? Can it shoot fireballs at you? Does it spawn minions?

Considering the first category of bosses, where the boss is a custom mob, typically the boss attacks using a combination of its vanilla AI and command block-assisted special powers, but you can make mob-bosses whose attacks are purely command-block based. This is simply done by making the mob's NoAI tag be set to 1, and then it can either stand still and "cast" all its attacks at you or teleport around, or even have custom movement using a tool like CutscenePRO - the Green Ender Captain's Ender Ghast does this in Assassin of Steve 3.

Your boss should have a number of different attacks created using command blocks and functions rather than just relying on the normal AI and attack patterns that Minecraft gives a certain mob. One example of a custom ability is spawning fireballs above the player that then fly down and explode upon hitting the ground in rapid succession, forcing the player to move fast to avoid being engulfed by the explosions. Another example, for a skeleton, wither skeleton, or stray, is having the boss switch between a sword and a bow randomly during the fight or depending on how far away the closest player is.

With the second category, for built bosses, the boss attacks are purely command-block based, and can be anything from shooting fireballs at the player to spawning minions. Lord Prismbeam in Assassin of Steve 2 is a giant elder guardian built out of sandstone blocks, and to replicate the signature laser-attack of guardians on a much larger scale, fireworks of various colours flash in lines through the water, damaging the player using the instant harming effect should they be caught in the way.



If the boss spawns minions, the minions should either only spawn once and once they're dead they're gone for good, or if you have minions repeatedly spawning during a boss fight, they should not be too difficult to defeat, or at the very least are not too much of a threat, and they should not get in the way too much of damaging the boss.

For instance, if your boss is an evil statue in the wall and you have to do some parkour to damage the boss, you shouldn't make the minions be skeletons with bows because it is very difficult to do even simple parkour with skeletons constantly shooting at you.

An important thing to always remember is that you don't want your boss to be too overwhelming or too deadly with respect to the rest of the map, or too easy, either. You want your players to feel a sense of accomplishment upon defeating the boss. If the boss is too hard, players will quickly get frustrated. If the boss is too easy, after easily defeating it, players will go "meh."

The number of possible boss attacks that you can design using command blocks is literally limitless and you can be as endlessly creative as you like.

How to Damage the Boss:

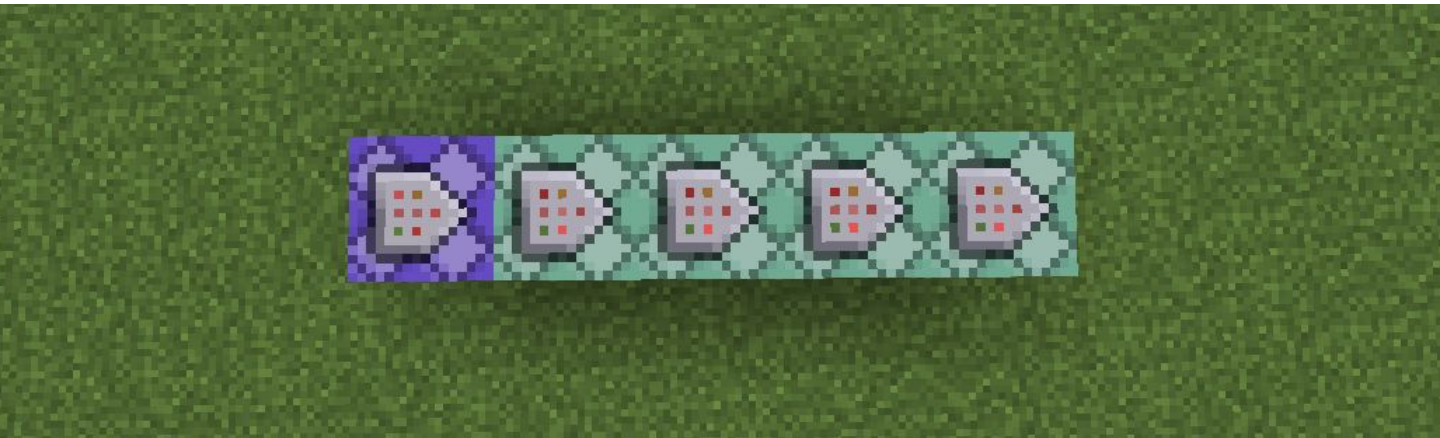
The final thing you should consider is how do you damage and defeat the boss, how do you solve the challenge? Do you damage it by shooting some buttons in its eyes? Or by doing some parkour to climb inside its mouth?

It is **extremely important** to make it clear on how to damage a boss, on how to solve a challenge. This could be done using signs, or text in the chat.

If your boss is a mob that you fight, you could just give the mob lots of health to make it a “traditional” boss fight, in a sense, such as the very first boss fight in all three Assassin of Steve maps - Slaw, the Silver Queen, and the Sand Monster, are all simple mobs that have much more health than other, more normal mobs. The Silver Queen can burrow around and spawn silverfish while the Sand Monster can create a sand-shield that makes it temporarily invulnerable and also slams the ground halfway through the boss fight, dropping you and the Sand Monster into a cave. The important thing to note is that you don’t want a boss to be too hard, or take too long to kill, either. Don’t give your boss so much health that after about five minutes’ of fighting the player is left wondering if there is something else special you’re supposed to do to defeat the boss.
1.13 adds the new /bossbar command, which can be used in conjunction with other commands to track how much health a certain entity has, and is an ideal way of showing players how much progress they have made on battling a boss.

The following commands create the bossbar and boss mob:
/summon minecraft:zombie ~ ~1 ~ {CustomName:"{\"text\":\"Boss\"}"}
/bossbar add minecraft:bill Bill
/bossbar set minecraft:bill players @a

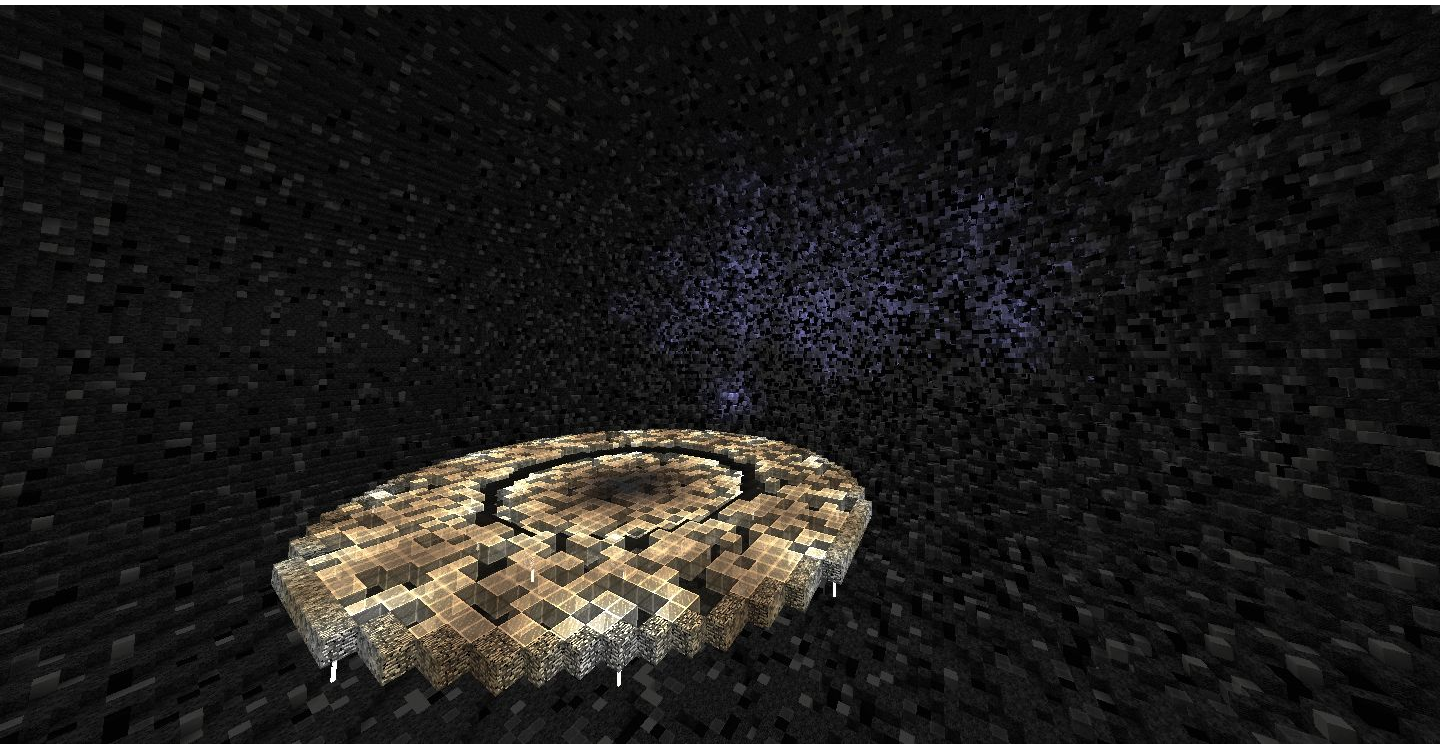
While these following commands go in repeating command blocks to update the bossbar.
The first command makes the “max” of the boss health bar be the same as the max health of the boss mob
/execute store result bossbar minecraft:bill max as @e[type=zombie,name=Bill,limit=1] run data get entity @s Attributes[0].Base
While the second command is for the boss mob’s current health.
/execute store result bossbar minecraft:bill value as @e[type=zombie,name=Bill,limit=1] run data get entity @s Health



There have been boss fights where the mob-boss has the Resistance V effect, making it unable to take any damage, but such a boss can be “damaged” by doing other challenges. A simple example of this is in my first Assassin of Steve map, where the Captain Yarma boss is an iron golem that is unable to take damage, and in order to defeat him you have to knock him into a pit in the centre of the arena. It is made clear that this is how you defeat him by some messages in the chat from your ally Stuart, who states that Captain Yarma is too strong for you, then takes control of Yarma’s cannon to blast open a hole in the floor, creating the pit you have to knock him into. Finally, Stuart tells you to defeat Captain Yarma you must knock him into the pit, where he’ll be trapped.

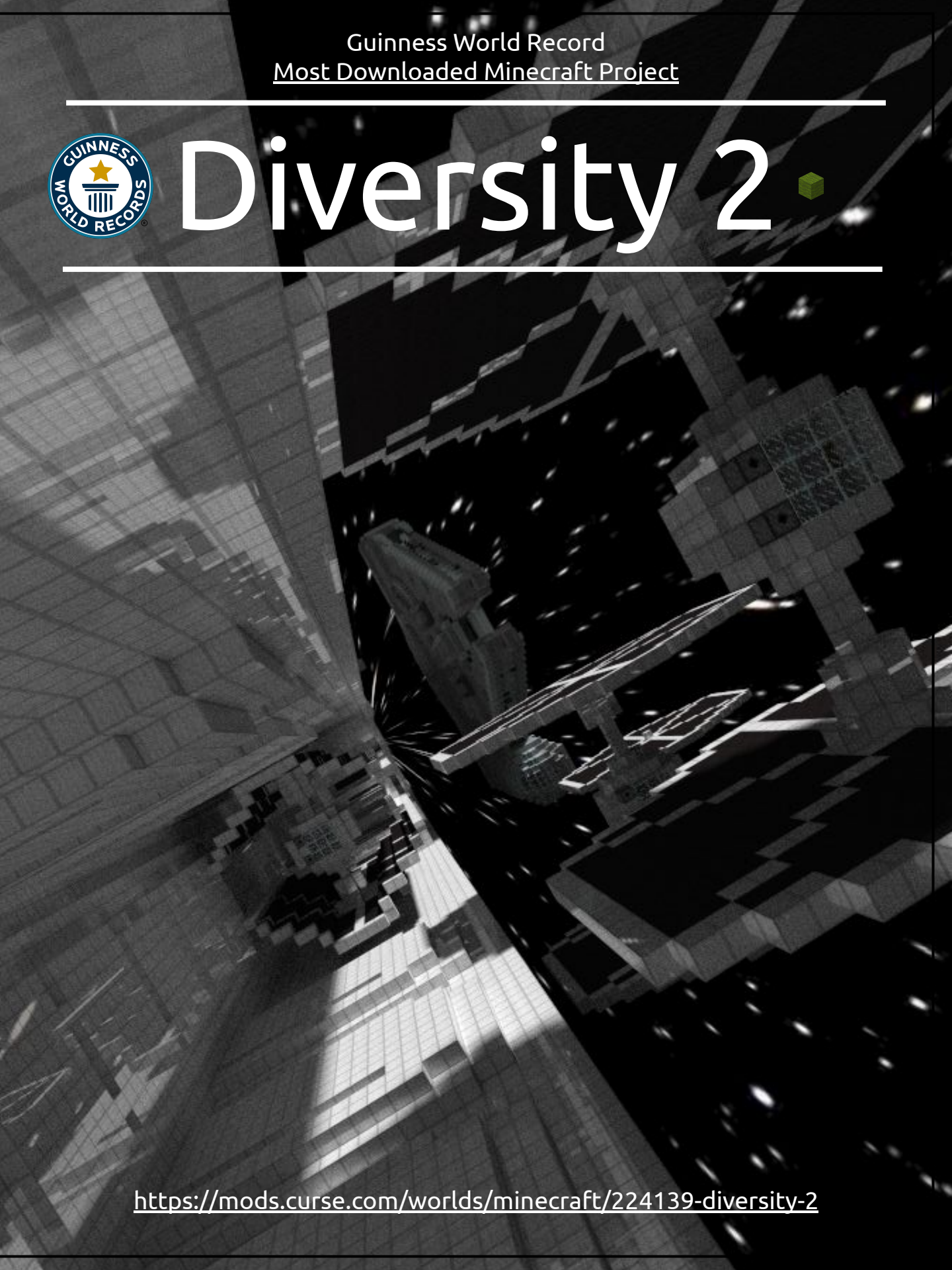
If your boss is a built boss, one of the most common ways of defeating built bosses entail shooting buttons on the boss using a bow. In Herobrine’s Return by Hypixel, both of the built bosses - Skeletor and the giant demon-boss (can’t remember his name), are both damaged and defeated by shooting buttons.
Other creative ways of defeating a built boss include doing some parkour to get up to the boss’s weak spot and attack it.
Another way of damaging the boss could be to farm certain drops from minions that spawn and then place in a hopper or give to a villager, which then activates something that damages the boss. If this method is employed, the minions should be easy to defeat, always have a high chance of dropping the items, and you shouldn’t need to farm the items for too long before you get enough, lest the player gets bored or annoyed.
The Mega Ghast boss fight in Assassin of Steve 1 requires you to farm certain drops from the minions that spawn and then give to villagers, who then give you a special fire charge that can be placed in a hopper to fire out of a cannon at the Mega Ghast.

There are literally endless ways of defeating built bosses, from disabling power cores to placing sponges to drain all the water away. If it fits in your head, it can probably be created in Minecraft using command blocks and functions.



Guinness World Record
Most Downloaded Minecraft Project

Diversity 2



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

Fail to Succeed

By Master_Nati: @UnicornNati

So some of you may have read this title and thought: “How does that even make any sense?” But what you have to keep in mind is that failure and success aren’t mutually exclusive. Yes, they are antonyms, but that doesn’t mean one can’t originate from another. Just like you need light to be able to speak of a shadow. This is what I’m going to try and explain in this article; how you can improve by learning to accept failure. But I’m going to take this idea one step further and give you some tips on how to fail more often, which remember, isn’t necessarily a bad thing.

“Why should I fail, isn’t the easiest path just succeeding all the way through?”

Technically yes, in a perfect world. But you aren’t perfect, neither am I and neither is someone else. We all make mistakes, we’ve made them before and we will continue to make them constantly. We can’t all be right, because there is no definite ‘right’ in something as complex and subjective as mapmaking. Some people will enjoy certain aspects of your map more than others. That’s why we all playtest, to have a bigger sample size of what our desired audience wants. But playtesting too has its flaws. People will almost never suggest big changes because that would be too demanding, even if they really feel these changes are necessary. If this occurs (not like you’ll ever know) it means it’s too late, because changing the core idea of a playable map requires a lot of attention and it would be arguably better to just start anew. Playtesters also rarely know exactly what they want, just like you when making the map. They might feel like a weapon is too overpowered, but the real problem might be that a different weapon controls too bad meaning other weapons look worse. Or the problem might be that it just *feels* better. So in conclusion, let people playtest sooner and have them point out all the obvious flaws because one of them might not have been obvious to you.

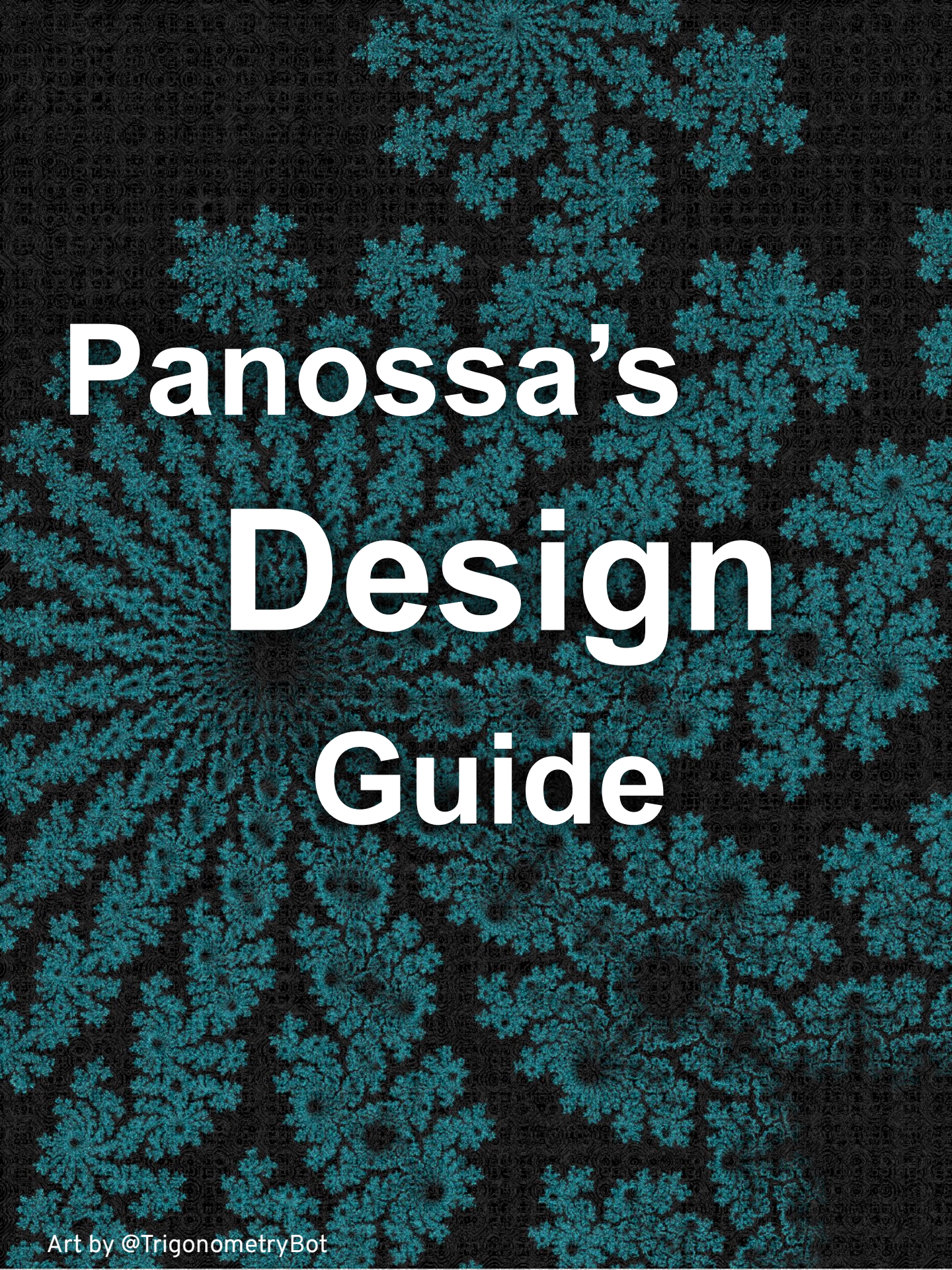
Fail to Succeed

“Ok, ok, I should let people playtest sooner, but that isn’t really failing is it, just identifying the problems early on.”

Right again, partially. Because I haven’t yet told the full story. What if I told you that ‘failing’ while playtesting early is also too late? Now you’re probably scraping your head trying to figure out how that makes any sense, but allow me to explain. You need to be able to distance yourself from your products. Drop all the emotional connection you have with it and replace it with rational thoughts. Then you will be able to identify things beyond problems. Things that aren’t exactly problems, but shouldn’t be welcome in your game either.

“That sounds very abstract. And I don’t think cutting off all emotion is a good thing, shouldn’t everything be made with love?”

Well, I believe that there is a subtle difference between a product ‘made with love’ and a product ‘made with emotion’. What I mean by that is that it is important that developers put their soul and everything in the product, it shouldn’t be treated as a child. You can be very proud of your work, but don’t become overly attached.
I am aware of the very thin line I’m walking here, there are no real instructions I can give on what to do exactly in this area. All I can leave you here are the words of *Extra Credits:* “Fail faster, no idea is made fully formed, your ideas can’t be precious, your ego can’t need protecting, every failure is a chance to get it right.”


Panossa's
Design
Guide
Art by @TrigonometryBot

The wonders of gameplay design

By Panossa

One of the many attributes a map/game should have to be considered "good" is mindful gameplay design.

This is one key aspect to keep the player engaged for long and is also, in some countries, the most important part of a game. Why only in some countries? Most game studios in the US and Europe place story or visual design at the top of their priority list (Life is strange, Call of Duty, in that order). They first think about how they can visually stun the player or what they want to tell him, then they build a mostly simple method to convey e.g. the story, hence the core gameplay of "Life is Strange" is not special in any way. The same goes with most AAA titles by EA, Ubisoft etc.

Nintendo, on the other hand, has an entirely different approach on games. In the development process of many Zelda and Mario games the new gameplay and puzzle elements are ready before a map, a story or even new visuals are added to the project. This is why most Zelda and Mario games have the same story but are very different nonetheless.

Now, what is good gameplay?

First of all, **mechanics** are the core part of the gameplay experience. Something the player didn't see before, seamless implemented and ready for every player to master. Mechanics can be relaxing or competitive. Button mashing is often replaced by muscle memory training aka hotkey combinations. A good example here is one of the crazy triple+ jumps in Mario Odyssey. Not that easy to learn but mastering the skill pays out.

However, even a simple mechanic can be used in so many ways it begins to seem variable. (Celeste's C-sites, Crypt of the Necrodancer). Here is another great train of thought on the topic of mechanics by Jak544.

The next big package of the gameplay design is the **flow**. It starts with seamless transitions in the menus and ends with a perfect and responsive implementation of a mechanic. The ability to cancel a combo, chain different movesets or events is insanely valuable once inserted into the game. Of course bugs are big problems here if the core element of the game is a mechanic. That's why, unlike Life is Strange (where bug testing can even be automated), a game like Mario Odyssey needs many playtesters. And so do you, if your map relies on a complicated mechanic (or many different of them). So far only one relevant bug has been found in Mario Odyssey, though all normal mechanics are working perfectly.

Game Feel

By Panossa

The next part of the gameplay design is called "game feel" here and represents a very abstract part of the gameplay experience of any player. It describes how the gameplay feels to the player, obviously. This part of the article is heavily inspired by Mark Brown's video on the same topic.

The key parts of the game feel design can be viewed like a relationship between the player and the game. While the player wants to make an **impact**, the game provides the **immersion**. Both push their respective tasks onto another. The player wants to make as much **impact** as possible (which is why games like Life is Strange are so hyped) and the game tries to **immerse** the player into the world as much as possible.

Immersion can be achieved through background music, exciting stories, level design and more. And it's very important to remember the **immersion** breaks through randomness unless the randomness is a key part of the game. The game needs to **feel alive**.

Giving the player a way to make an **impact** can be done through various ways, depending on what the core gameplay is. In fighting games it's mostly sound and visuals. If the player shoots/kicks/stabs/... he really needs to feel it. The easiest way to achieve this sound-wise is by making the hit sounds loud, the reactions of the environment realistic and rewarding the player for doing something right.

Visuals can help here too. Many games use some tricks to create the illusion of an extremely hard hit on an enemy. They can make them flash up, stumble, pause for a split second. (Examples: Nuclear Throne, God of War)

If a platformer is being made, the movement of the player can be modified in many ways. It can have friction, momentum and its own weight to it. A simple example of that would be partially using ice for jump'n'runs.

Last but not least, it's possible to implement the **impact** via the story. NPCs can react to decisions or to phrases the player said. Even if they are just dialogue changes. The game needs to **be responsive**.

Tip: Start your game development with the gameplay. If you finished the rest, go back to it again and spice it up. Textures, particles, sound effects, everything goes.

Keeping Players engaged

By Panossa

The last part of the gameplay package is the **engagement**. This is the part of the gameplay experience which decides whether a player will stick to the game for long or just quit the game after a while.
Many different aspects are part of it, the type of player (covered in another article) is very important, but here is one of them explained in detail: feedback loops.

Feedback loops are rewarding/punishing mechanisms of games and of course need to be balanced.

There are two kinds of feedback loops: **positive** and **negative**.

These terms come from e.g. biology and thus describe not a positive/negative effect on the player but on themselves. To explain this further the following two games should be analyzed: Call of Duty and Mario Kart.

Keeping Players engaged

By Panossa

A positive feedback loop is a **reinforcement** loop. Meaning an effect gets stronger over iterations. A player gets a boost if he scores well, then it's easier to score even more to get even better boosts. In Call of Duty a player gets an ability as a reward for multi-kills which helps him get even more consecutive kills to get the next ability. This makes the game unbalanced. Good players are rewarded while bad players get nothing.

Additionally, positive feedback loops work other way round, too. Not only does a player get boosts if he is good. But if he is bad he doesn't get boosts which in turn can make the game way too hard later on. This is also a positive feedback loop, because the worse the player is the worse his situation gets; the **reinforcement** of the effect is present.

Negative feedback loops are the opposite in every way. They tend to keep good players from running away and bad players from being left behind. In Mario Kart a player who is on the first place gets bad items which doesn't help him get an even bigger advantage. The further a player is from the first place the better his items get. He can catch up.

It's apparent **negative** feedback loops are **balancing** loops. They try to get the players towards a middle/center. In Mario Kart singleplayer the AI enemies are even slowed down if a player falls behind.

Now, what is the lesson from this to learn?

Keeping Players engaged

By Panossa

Many games have negative and positive feedback loops and are popular nonetheless. Maybe they are just games of a well known franchise or have other strengths.
The lesson is: if a map creator wants to keep the player engaged he could (besides from improving any of the other parts of the game) weaken the loops if needed or just don't add too strong feedback loops.

Negative feedback loops are easily fixable. A player on the last place shouldn't get something that can easily catapult him to the top. Just like the best player shouldn't get punished too hard for being good. In Mario Kart the first place can often hold himself on the first place by just being good at driving and gets items for his personal defense. Just like the last place isn't on the front after one item use.

Positive feedback loops can be great to show players how much impact they can have but they get boring quite fast. To fix positive loops the developer has at least two possibilities. The first possibility is to just stretch the loop. If a player gets rewards for different achievements but he needs many hours to get to the limit of those rewards/boosts, he will play the game long enough. While long loops account for diversified, long gameplay, this is often hard (and takes long) to implement without making the gameplay too monotonous hence the game gets boring because of repetition. It's recommended for bigger teams or teams with much time.

The other way to fix a positive feedback loop is to reset the momentum. In the game Minit the player has one minute until he dies, though the map is the same every run. He learns more about the map but is reset periodically. This helps to stretch the game while keeping it exciting. And the game goes on...

In general there is the possibility to add a counteracting feedback loop. The negative loop of Mario Kart could be weakened by gradually speeding all players up as long as the first place is getting away. This way the first player won't feel cheated by the game and all other players feel a chance of winning. Another possibility is giving the first place better items if he is leading for a long time. Not OP items but just more of the items which can help him against e.g. the blue shells.

Approaches for game design

by Panossa

There are two main ways one can approach the design of a game/map...

1. Planned design

This is often the approach of big companies or groups which don't have much time to "play around" or just don't want to do that. Think about the main aspects of the game and think about what you need to convey that. Which textures are needed, how many levels do you need, which mechanics should the player learn throughout the map? Plan everything, make todo lists and if you can, divide the tasks you know created.

2. Debug Design

This is another approach designing. I will focus on the developer behind "The Witness" here. Jonathan Blow created first the well received game, Braid. It's a 2D game with the main mechanic of changing time. When he made the game he first thought of the idea of binding the time of the game to a trigger. The player could move through the level while rewinding time. So he tested that and came to the conclusion, he can now change every part of what he just created.

The finished game has many different time mechanics. In some levels you are "rewinded" as well as everything around you. Sometimes you have props and enemies who can just ignore the time passing by. The triggers can change, too. Some levels even rewind time only if you walk left. If you walk to the right, the time continues and if you don't move, the level doesn't, either.

Approaches for game design

by Panossa

In "The Witness" Jonathan Blow used this approach again. His main idea was to create puzzles where something at a starting point needs to get the end on a grid. The finished game now has a lot of different grids with their own rules you can learn on the grid itself or through looking at the environment. There is a really big amount of different grid puzzles. The core is always the same. But it's different enough to keep many players engaged for hours.

Basically, debug design consists of two different possibilities. Either you experiment with an idea you have. Or you take a look at e.g. one feature your game engine has (e.g. slimes being changeable in size) and experiment with that. (You can make slimes eat each other so they get bigger and do more damage to the player)

Dungeon Design

by Panossa

In search of material which describes how to make memorable, good quests and dungeons I found one interesting principle in a video by Adam Millard. He says dungeon/quest design can generally be described with the four X'es. I don't agree with all of it so here's my own interpretation of those.

1. eXplore
2. eXpand
3. eXploit
4. eXterminate

eXplore means there needs to be room to explore without something hindering you. Add lore, cool gimmicks/mechanics etc. The world needs to **look alive** to make the player want to explore it. Lore hidden in buildings or structures in general is a good way to start. Variety is key.

eXpand stands for the expansion of the territory knowledge, which in randomly generated dungeons is automatically given cause the player progresses through exploration. In custom open world maps/games you especially need to take care of that as the player can run wherever he wants. Here, the world needs to **feel alive**. Change everything from time to time. Weather, biomes, different kinds of NPCs/monsters. E.g. every enemy should be bound to a room style. His visual design, attack pattern etc. If you put enemies where they don't belong, even a player in his first playthrough should notice it's the wrong place.

eXploit simply means the player should be free to exploit all of his knowledge of the world, the mechanics and the lore to progress through the dungeon. If he knows how to fight something more easily, let him do that. If he knows a trick to speedrun though everything, let him do that. And so on.

eXterminate is something that's a core part in most dungeon systems. It means there should be fights and diversity in those. Bosses are part of that diversity.

Open world design

by Panossa

There are not many big open world maps for Minecraft out. Some mapmakers just don't want to make a huge map, some just don't know how to approach this task. This article displays two different principles to create an open world map/game.

The first one is the story based design. It goes like this:

1. Accumulation:
First, you collect or create all the data you have planned for the story. The number of bosses, biomes, villages, story paths, substories etc.

2. Array:
The next step is to sort all those planned features. Which are really needed, which are probably something that belongs in another map or can be left out? How linear should the open world experience feel? Here's a good moment to talk about something like the "dependency trees" by Mark Brown, though this would be far too much for this article so here's a link.

3. Abstraction:
Making an abstraction of the map is probably the best idea at this point. Ironically though, this step of the world creation process is also about adding details. This makes sense soon. First, it's needed to make a simple representation of what the map would look like. The outline.

In fact, here is an example for that. This is the simplest deconstruction of the map of "Zelda: Breath of the Wild":

Open world design (cont.)

by Panossa

Of course this abstraction doesn't help much if it stays like it. In this 3rd step the task is to recursively add abstractions. Take, for example, the upper right corner of the BotW map. After an imaginary zoom into this part of the map it's possible to start abstracting it again. How is this corner built? What are the features needed here? Here's the example of that.

Of course it's possible to take as many detailing iterations as needed to begin with step (4). It really depends on the size of the map and on the time on hand. Generally, if it feels like there are enough details to start building one part of the map, it's enough.

Optionally it's possible to slide in another substep in this part of the creation process. Normally, this abstraction process is something that happens outside of the actual game. On a piece of paper. Or as an image. But there is another way which was used e.g. in the map creation of the map of BotW. The third dimension inside the actual game provides another perspective. This is where the team behind BotW used the "triangle rule" and a few other neat tricks. It's way too much to talk about that in this article but here is an article describing this idea. (Not in english, use of translators advised.)

4. Assets:

Now it's time to take those feature ideas from (2) with the abstraction map from (3) and start at any point with the design of the actual map. Now the building process starts. Assets are built on the places they already belong to. The pieces come together.

Open world design (cont.)

by Panossa

The second famous way of making an open world map is the modular design.

This is basically the first one the other way round, only this time it's shorter.

1. Assets:

The first task is to build just everything that comes to mind. Small houses, dungeons, interesting trees and so on. Assets can also be quest concepts or full quests as well as textures and just random ideas for everything. Gameplay, Appearance, Story...

2. Array:

While "the pieces come together" was the end of the first approach, it's almost the start of this one. One asset or set piece is defined as the start of the map. From that, all other pieces are gathered together as long as they make some sense together. A desert hut won't fit near an iglu. But it's possible that there are many assets or pieces in general which fit well together. One really big and good example of this modular design (especially this "array" step) can be seen at the Minecraft MMORPG server Wynncraft. Here is the live map of that server.

3. Ad-lib:

Now that the map is ready, the story can be built around what has been build. Maybe even new ideas came up revolving around the pieces that came together during the build process.

Open world design (cont.)

by Panossa

In conclusion it needs to be said that these two design principles are of course not the only two possible.

There are teams which make assets parallel to planning the map. Some assets, quests or mechanics won't be used in that case but can be used maybe in the next map. Other teams may combine other parts of those principles together.

For example: making a story based around assets and then putting the assets together without having a full plan for the whole map.

In a direct comparison of those two principles the following should be clear: while the story based design uses up more time and planning (especially if you want to add something in a later update), it is most of the time the better idea because the player notices that extra work and planning and it's the better experience overall.

The modular design can look pretty random and usually doesn't impress the player as much as good story based design. But it's time efficient and the map can be rearranged and transformed in any way pretty fast.

Story based design:

+ In general the better experience
- Harder to expand upon updates

Modular design:

+ Low cost
+ Can be rearranged
- Often looks lazy/bland

Puzzle design

by Panossa

This article describes Mark Brown's theory on how (good) puzzles work. For the full explanation with great examples, see this video. So here are the pieces which make up a puzzle:

1. Rules/limitations:
The player has to learn the basic rules of the world he is in. Rules include movement controls, gravity or other influences, consequences of specific actions.

(Example from Snakebird: eating fruits makes you longer which is good because you can reach more places but it's a curse because the player gets stuck everywhere.)

Tip: Temporary tools can be introduced here and augment the experience. Sometimes seemingly useless tools can be used in puzzles later on.

2. The catch:
Every puzzle has a logical contradiction or a conflict. See the first puzzle of Portal. There is a door and a switch. If you press the switch, the door opens. But you aren't fast enough to reach it. You need to put something on the switch.

Every puzzle teaches you another way of thinking about the game through the catch. Sometimes there are red herrings which help the player being stuck in old thinking schemes. Important: Don't be unfair. The player should have the chance to get to the solution by just looking around, having his past knowledge in mind. Breaking the basic rules does not make a good puzzle. The best puzzles use small aspects of what the player already knows but didn't pay attention to.

Puzzle design (cont.)

by Panossa

3. The revelation:
The important part of a puzzle is the journey, the discovery, the "eureka" moment. It should be generally effortless to execute the solution once you know it. Except if making hard gameplay is not the main part of the game.

4. The assumption:
Good puzzles let the players make one or more assumptions about how the puzzle works but then being different if you think about it again. This is not the place to be evil! Giving players wrong assumptions is used to make the revelation more rewarding and gives the player a starting point to solving the puzzle.

5. The presentation:
Show the player everything he needs to know. The goal of the puzzle is usually in plain sight. Making the puzzle a part of a huge labyrinth usually doesn't help. The presentation can be used to lure the player into a wrong assumption, too.

6. The (learning) curve:
Good games usually build up new puzzles on top of the old ones. More precisely they want the player to use everything he needs to know to reach this puzzle in the first place. The difficulty needs to ramp up, too. More possible moves, more possible endings/lost chances.

Narrative design

by Panossa

The design of a story or the lore of a game/map is a huge task if you really try to be good at all of its aspects. This article describes a really important principle of good narrative design: "Show, don't tell".

For a story to be remembered the best trick is to show the player as much info as possible without telling what it really means. The best story works the best if the player actually has to think for himself while experiencing it.

Here's an example: The player talks to an NPC who says his wife ran away because he is an alcoholic and now he wants her back. He describes how lonely he feels and that his wife is probably at the party down the road. Now the player needs to find the woman and deliver the bland words of her husband.
Now picture the mission like this: The player talks to an NPC who says he didn't see her wife for a day now and maybe the player wants to help him. Now the task is to find some clues around his house. We find many empty beer bottles laying around, then we see that his wife has her own room (it seems like they had problems for quite some time together). On her desk we see a flyer for the big party near the husband's house. Now the player runs up to the woman and gets the confirmation about the theory his head already developed. He knows about the same amount about these two people but it has way more impact on him. Plus, he gets the bonus of feeling smart because he figured it all out.

In "real" games you can use a wide variety of tools to get the hints across. Facial/body expressions, voice, props on the ground/wall/ceiling etc. On the other hand, in real games you have the image in front of you. Compare that to books where the reader needs to picture everything anyways. It's harder to use "Show, don't tell" in games/maps but it pays out.

Conclusion: The less you tell your players directly, the better the story. Of course it's important to not make the revelation being too hard. Letting players search for hours through Wikipedia is probably not a good idea. Finding the balance here needs a tad of time but it's extremely valuable.

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!*

The Free Independent Minecraft Map Maker's Magazine

MAPMAG

ISSUES 1-10

http://www.testfordev.com/MapMag