Darren Wershler
Bart Simon
August 2022
Digital Pedagogy Institute
v8

Rattletrap Platforms: Modded Minecraft and Digital Pedagogy

[INTRO SLIDE]

Notoriously, many of the platforms and tools we use for digital pedagogy at the university level are *not very good*: a farrago of poorly documented, inflexible off-the shelf packages, conflicting plugins, and glitchy scripts that utilizes computer resources poorly and spends nearly as much time down as up and running. But are there pedagogical advantages to teaching with rattletrap platforms?

[INFRASTRUCTURAL REFLEXIVITY SLIDE]

Drawing on our current research project "Material Allegories and Minecraft" and our use of heavily modified ("modded") versions of the game in classrooms since 2014, We argue that sometimes rattletrap platforms and tools are *better* for teaching because they foreground the material conditions of learning and action. One of the explicit teaching goals of our Minecraft classes is to create an ongoing awareness of these conditions in our students. In our paper "The Allegorical Build," in A 2021 special issue of _Gamevironments_ titled "Revisiting Teaching and Games," we call this awareness "*infrastructural reflexivity* (213).

Infrastructure is a broad concept because what it describes is messy, complex, and difficult to discern. In her foundational essay “The Ethnography of Infrastructure,” Susan Leigh Star writes that the common conception of infrastructure is of “a system of substrates” that is “by definition invisible, part of the background for other kinds of work [and] ready-to-hand”. As the embodiment of standards and protocols, infrastructure is hard to observe because it extends into and depends on other structures, technologies, and social arrangements to which it connects in a standardized way. Because it does not have to be reassembled or reinvented every time a task is performed, it really only becomes visible at moments of breakdown, lag, and other types of failure.

The reflexivity part is both material and cognitive. Material reflexivity is the actual feedback loop between a player and some material aspect of the game, whether it's a deliberate procedure, or an exploit or glitch. Cognitive reflexivity is allegorical in that it describes the awareness of that loop in relation to infrastructure as a condition of possibility for learning (or collective action of some sort).

[GLITCHED GOLEM SLIDE]

To achieve infrastructural reflexivity, we don't desire to make digital learning platforms flawless and invisible. We **want** bugs and glitches to occur, because they foreground the material conditions of learning and action. Rattletrap platforms defeat consumerist expectations; they allow for failure and rethinking; they create scenarios where genuine emergent phenomena can unfold; and, by virtue of the fact that students need to figure out where to turn when something goes wrong, they point to the larger systems that position, structure and regulate activities in the digital and physical classroom, the university and beyond. We contend that rattletrap platforms allow for the development of infrastructural reflexivity in students. But first, some context.

Context

[MINECRAFT SPLASH SLIDE]

Despite its unpopularity with many self-identified gamers -- perhaps even because of that -- Mojang's (2009) and Microsoft's (2014) Minecraft remains the bestselling video game in history. With a daily active player count in 2022 between 2.8 and 3.6 million, the game is still increasing in popularity (https://techacake.com/how-many-people-play-minecraft/), with a racially, neurally and gender-diverse player base, with an average age of 24 (https://www.pcgamesn.com/minecraft/player-age). Because of its ubiquity, video game critic Ian Bogost contends that the impact of Minecraft as a cultural artifact is on par with the personal computer itself (https://www.nytimes.com/2016/04/17/magazine/the-minecraft-generation.html). The portion of the Minecraft milieu on which our research focuses is the part that makes use of independent, player-made modifications, or *mods*.

The oldest Minecraft mods appeared in mid-2010, when the game was still in alpha development. There is substantial debate about what those mods were, exactly, but they have always been part of the game's history. Minecraft mods fall into two rough categories "client-side" (affecting only the play on your particular computer) and "server-side" (affecting all players connected to a particular server). Generally the two types of mods are bundled together into *modpacks* in order to create a particular kind of game experience (we call this bundling "mod choreography" because it requires the management of all sorts of technical and aesthetic conflicts in order to produce a coherent experience).

[CURSE MINECRAFT SLIDE]

None of this is easy. Minecraft was originally written in Java, and provides no official support for Java modding, so modders have had to collectively interpret the game's code over time, write their own mods in Java, and construct programs called *launchers* that coordinate and manage the mods for use in a given game. As of August 2022, there are over 111,000 active Java mods on CurseForge (the primary mod repository) alone (https://en.wikipedia.org/wiki/Minecraft_modding). Remember, these are all coded by amateurs, and they work together, and with a given Minecraft's server's capabilities, with varying degrees of success ... and frequent failure.

There has been a fair bit of research on Minecraft, particularly in education. However, the bulk of this research is precisely about the smoothness and immersiveness of the game: the idea that students ave so much fun they forget they are *learning*. This is the lure of gamification: the conviction that somehow people learn best when they don’t think they are learning. We are interested in the opposite scenario. Does software that's barely "good enough" for teaching (i.e. affordable, accessible, and familiar to our students), allow for emergent phenomena unanticipated by game designers and classroom instructors alike to occur *because of its weaknesses* ... and thus for unique "teachable moments"? Take, for example, the case of Moloch and the Bees.

Moloch
[LANG MOLOCH TEXT SLIDE]

The current version of our Minecraft course links the game with a series of readings on the history and culture of modernity. In our blocky little Minecraft world, Moloch is, literally, the machine in the garden: an allegorical manifestation of the most rapacious, remorseless aspects of industrial capitalism, drawn directly from Fritz Lang's 1927 film _Metropolis_, as the sign we placed on its back indicated.

[LANG'S MOLOCH SLIDE]

Since Biblical antiquity, the name “Moloch” has been synonymous with enormous sacrifice and servility … but the moderns used it explicitly as a metaphor for industrial capitalism. In the _*Grundrisse*_, for example, Karl Marx writes, “money is the hangman of all things, the Moloch to whom everything must be sacrificed, the despot of commodities.”

[BIBLICAL MOLOCH SLIDE]

One of the reasons we included Moloch within the game was precisely in order to make Minecraft’s heterotopic nature more visible. In this case, the tendency of Minecraft’s visual aesthetic to reduce everything to kitsch actually helps to drive home the ambivalent qualities of modernity we were trying to capture.

[OUR MOLOCH SLIDE]

As a pedagogical tool, our Moloch was designed to accomplish two tasks, both of which stem directly from the name’s historical connotations. The first was to serve as an introductory tutorial to the game; students are divided into groups and each group receives their own Moloch monument. The second was to try and provide a solution to the overproduction that is an inevitable part of Minecraft play. I'll talk about each of these functions briefly.

[MOLOCH TASKBOOK SLIDE]

Moloch is as much a blank parody of educational technology as it is an allegory for it. We use the term “blank parody” because

Moloch actually does perform a useful task in that it teaches our students some of the basic techniques for modded Minecraft play (which can differ substantially from vanilla procedures). And reciprocally, it taught the people on the research side of this class something about mod-making and educational design; we have, in fact, since released it as a legitimate mod. But we designed it to look and act like an evil industrial demon in part because of our deep and abiding concerns about many of the things that ed tech may bring about, including an increasingly depersonalized classroom, and the further de-skilling and commercialization of higher education personnel.

Some of our students bothered to ask why they had to listen to Moloch at all, and the truth is, they could completely ignore it. In the first class where we used Moloch, not meeting Moloch's demands for excess commodities produced by the group triggered a mod called "Bloodmoon" (https://www.curseforge.com/minecraft/mc-mods/bloodmoon), in which nighttime monsters spawn faster, closer to the players and in greater numbers than usual.

[BLOODMOON SLIDE]

Because many players relish combat as a challenge, this wasn't much of a punishment. With some students, the newfound discovery that it was possible to simply ignore Moloch's demands (and experience the thrill of a bloodmoon) became an opportunity to discuss the workings of ideology, which could easily become part of the general course curriculum. But we had something else in mind.

[MOLOCH HEART INTERFACE SLIDE]

Moloch’s other, primary purpose was to help us manage overproduction. One of the problems with Minecraft in terms of game design is that by late game, players have typically automated their builds to such an extent that what they end up doing is sitting back and relaxing as their inventories mount. But what we wondered was if we could build something that would embody what modern artist and philosopher Georges Bataille called "the general economy": the principle of waste, expenditure, sacrifice, circulation and loss that coexists with the regular, "restricted" economy based on savings and profit (and, by some accounts, may even make the restricted economy possible). Could we build something rapacious enough to act as a kind of entropic balance to the game's tendency to overproduction and achieve a kind of late game-balance? The short answer is, No. The longer answer constitutes the bulk of this talk ... and has a lot to do with *bees*.

The TAG Minecraft Bloc has been playing and researching modded Minecraft since 2014, and I've been teaching a version of this class since that same year. But the winter 2021 session of our course, during lockdown, was the first time we had used anything other than the vanilla version in the classroom. The modpack we built for this class was deliberately small, and we thought we understood the capabilities of the mods in it fairly well. There was of course, one fatal exception, which totally exceeded the modest scale we had anticipated for in-game production in this class: the bees.

The Bees
[INNOCENT BEE SLIDE]

Bees of various sorts have been in Minecraft for years now, as part of mods like Forestry, Extra Bees, Buzzy Bees and Magic Bees. Bees were added into the vanilla version of Minecraft Java addition on December 10, 2019, as part of Minecraft 1.15. So when my collaborators on this project (Bart Simon and Nic Watson) said they were adding bees to the pack (Productive Bees, it turned out), I thought, "Oh, nice, We've had bees before.

This is a great way to allegorize Mendelian genetics and Punnett squares and all of that haute-bourgeois science stuff from early modernity." I had no idea what was coming.

[SMALL BEE FARM FROM AIR]

During my regular check-in flights around the map, I would often pass over the bases of various teams and look down at the little rows of glass houses over the beehives. This was also a lesson for me in the drastic difference between the strategic view from overhead and the tactical view on the ground (a difference that I regularly teach via Michel De Certeau's "Walking in the City"). If I had ever bothered to land and see what was going on underneath those little glass houses, it would have saved me a real shock later.

But looking down at the tidy apiaries, once again, I thought, "Nice." I knew the domesticated Productive Bees were making honey that could be refined into blocks of relatively rare resources by the players. But in most Minecraft mods, the bees eventually die off and have to be replaced. So while there is a benefit to becoming a beekeeper, it's a lot of work to keep your hive stocked. But for the student groups, who had by this point outlined several hugely ambitious term projects that would require them to produce and process vast amounts of raw materials, any possible advantage was worth investigating.

One of the sets of techniques the students in this particular class learned early and learned had to do with online collaboration. By January 23rd, 2021, one student had posted a Google Doc containing what they had learned about beekeeping in a place where the whole class could access it. On February 2, Nic Watson posted an apparently innocent question in the course team's chat: "I wonder if anyone will abandon traditional mining and just start using productive bees for everything."

I still wasn't paying close enough attention.

On February 20th, a beekeeper in another group wrote in their channel, "Want to set up an auto sorting for the beehives now so that you aren't taking from beehives"? Another member of the same group wrote, "All right, if you believe in your bees, I'm open to the killing machine." When the first sketches for the aforementioned machine appeared in their channel, I thought, "Interesting." (Still not paying attention). On February 21st, the first student in this group replied, "There is a very particular artistic touch to building massive bee farms that are fully automated." I decided to investigate; that last part -- "fully automated" -- set off my spider-sense.

[BEE TOWER SLIDE]

The first picture of the completed auto sorting hive system -- which I eventually started thinking of as the Bee Machine -- arrived in the Discord on the 22nd. It appeared to be an enormous hexagonal building with a single thin, very tall glass tower on one side. I went out to see it the next day. It was a long trip because it was well removed from the area around spawn.

[BEE TOWER MAP LOCATION SLIDE]

The first time I walked up to the Bee Machine, activating the devices and entities inside, the entire class server crashed. At the time, I thought it was a coincidence. (Not paying attention.)

[BEE TOWER INSIDE 1,2,3]

There is nothing modern about the design of the Bee Machine.

It's not postmodern either. It's a "machine landscape" that vaults directly over the postmodern into what cultural theorist Liam Young calls "architectures of the post-Anthropocene" In the January, February 2019 issue of _*Architectural Design*_ magazine.

[MACHINE LANDSCAPES SLIDE]

Machine landscapes are places that demonstrate bluntly that capitalism really doesn't need humans except as the most frail and dispensable of components. Amazon warehouses, roboticized container shipyards, data centres, Bitcoin mines, and wind farms: All such places function with only a handful of human operators. All the Bee Machine needed to function was one player present somewhere in the same chunk, away from the keyboard.

Our Moloch, modelled closely on Lang's, with its bricks and its glowing eyes and wires and sooty chimneys and infernal mouth, was designed to shock modern audiences into action almost a hundred years ago. The Bee Machine (which a classmate dubbed "the Bee Gulag" on April 3) makes Moloch look puny and insignificant by comparison.

[BEE FARM LOOT SLIDE]

When I got inside and saw the endless streams of valuable resources falling down chutes and passing along conveyor belts and then found the storage drawers into which they were sorted, I began to wonder if it had something to do with the server crash. At that time, around March, they were probably 15,000 diamonds alone in the Bee Machine storage system. I began to get the terrible feeling that I had missed something, and flew straight to what I thought was a **smaller** bee farm closer to the centre of the game.

[DREADED DRAWERS SLIDE]

One of the first things I saw was as sign that could have been written specifically for me. "Don't be daunted by the dreaded drawers," it said. When I looked in the diamond drawer, there were twice as many diamonds as in the Bee Machine, produced by a vast factory subterranean that automatically sorted and processed the output of the bees.

[SMALL BEE FARM LOOT SLIDE]

I had missed it because the entire, unbelievably efficient contraption was underground, powered by an enormous array of uranium-fueled, ice-cooled Thermoelectric Generators fueled with uranium (all of this comes from a mod called Immersive Engineering).

[SMALL BEE FARM POWER SLIDE, CENTRIFUGE SLIDE]

Only the quaint little glass bee enclosures above it were visible from the air. It was a completely different design than the Bee Machine, but equally posthuman.

I sat there for a long time and watched the game play itself.

While I was trying to figure out what it all meant, I went back to Young's machine landscapes issue. Again, there was something nagging me, and then, there it was: Bitcoin mining.

[BITCOIN FARM SLIDE]

The bee farms are utterly a product of our current culture because they are formally identical to the way a Bitcoin farm operates. You take a basic unit -- a cell phone, a cheap rack-mount PC with a big graphics card, or a pod with a bee and a full baby bee upgrade, which gives a 20 percent chance of spawning a new bee. Such a hive fills fast. When there are 10 to 12 bees, you put in your desired combination of production and time upgrades and you arrange them spatially in an array. What such machines do is blackbox the labour and the infrastructure they require to function in order to chase the dream of effortless profit.

Bitcoin farms are both fascinating and pernicious because they provide the appearance of generating wealth at no cost. There only appears to be no cost if you don't pay attention to infrastructure and public goods like roads, the electrical grid, the phone system, and so on. But by February 10th 2019, the BBC was reporting that researchers at Cambridge had found that Bitcoin mining consumes around a 121.36 terawatt hours a year, more than the entire country of Argentina. Because, as David Gerard notes, Bitcoin mining is grotesquely inefficient, its energy use and CO2 production will continue to spiral outward.

There's a lot to consider here. The bee farms produced a level of wealth that is commonly seen only from contraptions created by YouTubers working in creative mode. This isn't so much capitalist excess as it is wealth generation as entertainment. Bart and I began to wonder (after thinkers like Mackenzie Wark and Mark Fisher): are these even the procedures of capitalism anymore? There was no deficit. There was massive inequality across the server. But it's not like anyone really *wanted* anything in-game, because all of the beekeepers were extremely generous with their wealth, and everyone had too much anyway.

But the real inequities weren't on the Symbolic level; they were ethical. That is, they had nothing to do with the treatment of the virtual bees, or with who had more imaginary diamonds and emeralds in their imaginary house. The problems were ethical because they concerned material, interpersonal relations between students as citizens of the server, and had everything to do with the use of real infrastructural resources.

By April 2nd, Nic Watson had discovered that the server's block tag registry was taking up a whopping up 2.6 gigabytes of space. (This is at least an order of magnitude larger than it should be; a vanilla tag registry, which tracks every object currently in the game, is a few megabytes in size; a modded one maybe double the size of a vanilla registry.) How was that even possible?

Two days later, after some forensic work, Nic added, "So the bees are just polluting the tag registry as they fly around, deciding where to go next," and then, on the same day, "So this is even better than I could have imagined. This is literally the bees' decision-making processes, their brainwaves disrupting the servers, computational processes and the players activities and machines and so on." What was happening that as the bees flew around inside the Bee Machine tagging each separate flower, the registry ballooned in size because each sensed flower produced another registry entry. (That is, it's a result of poor amateur coding -- of rattletrap platforms) One of the students summed it all up nicely: "So we're pioneers in using bees to blow everything up."

The point of all this is that server crashes and slowdowns have real material consequences for students, who already have an asymmetrical experience because of the different hardware configurations they use to access the game. Who was able to log on and play, and who couldn't complete their work because the server was chugging under the increasing load created by the other groups' bees became a literal class issue. The in-game infrastructure created by the most ambitious student groups was adversely affecting the infrastructure for the entire game world, making it difficult for student groups with more modest projects to complete their term work. But building close to other players so you can take advantage of their being on the server when you are not in order to keep your factory loaded into active memory -- and functioning -- is yet another form of infrastructural reflexivity. In yet a third scenario, playing late at night (when fewer other players are on) in order to inconvenience fewer classmates with the lag that your machines cause is a not just allegorical reflexivity; it also becomes an allegory of shift work.

Once the students began to realize that they, themselves, were the cause of the problem -- that is, they reached the moment of infrastructural reflexivity -- their grumbling about "quality of service" began to drop off drastically. When I went back through the chat logs, it became obvious in retrospect that some of them had suspected the bees affected game infrastructure all along. On February 11th, one of our more active beekeeper students posted the following in one of the public channels: "Also the bees sometimes strain the server a little". On the same day, Nic replied, in jest (he thought), "Yep. Your bee farm is ruining everything." Understatement and humour disguised the nature and extent of the problem for another month and a half.

After our early April revelations about our slow but ongoing infrastructural disaster, the teaching and research team began discussing the bees heavily in our private channels. The students had pushed the server to the point that it was crashing regularly and we had to reset it daily. But this is exactly the kind of thing that we hope will happen when we set up a modded server: emergent behaviour that takes us somewhere completely unexpected. We had no intention of fixing the situation as long as we could ensure that the bee-heavy server would survive until the end of the term and everyone could complete their work ... while becoming aware of exactly why it was now more difficult to do that work, even if the bees provided the necessary resources to do so.

Reflections/Conclusion

[ENDERMAN 13 SLIDE]

We want you to notice a few things about our story here -- the mix between the fantastic fictional world of our Minecraft classroom; readings on the technology and culture of modernity; the technical operation of the server (along with the skill of the sysop, Nic Watson) that makes the world possible; and our students' thoughts and actions. Most educational software borrows from the the conceit that technical operations need to be black-boxed for the student. Good user design is, in Donald Norman's sense, uncomplicated. A program for learning about geography or math or modernity should be about geography or math or modernity, not programming or systems architecture. Yet, our breakable and broken server becomes more than just a teachable moment. It's more subtle than that... slowly, methodologically, (and occasionally abruptly and alarmingly) alerting students (who have been primed by the course material) to the material and infrastructural conditions that makes their ideas, actions and learning possible. This is not about Minecraft; this is about the digital culture in which we all live. It is the pedagogical thread that links Minecraft to Bitcoin to Zoom to the Metaverse.

[THEO'S CLASS SLIDE]

Digital learning must be about learning what the digital *is*; otherwise, it's just about consumption (and "education as service"). Software is never invisible and it never should be, but that presents ed tech designers with a new and more interesting problem: how to make usable broken things. Our digital classroom is broken, but it works. Glitches, lag and crashes are instructive when they are temporary and addressable by students. The system must work well enough to trouble the notion of what "well enough" means. Anything less and the students will not have a chance to discover anything; anything more and they will have no need to discover anything because all their needs are being met ... which is consuming, not learning. (All of this requires substantial infrastructural resources to run the class, but that's a detailed topic for another time.)

The story of the bees eventually gives way to new in-game practices as students begin to understand the relation between the Productive Bee mod's impact on the server and other factors, including their ability to complete their homework. While it was up to Nic to engineer the server to keep the class going (so the students could complete their projects and get their grades etc ...) the ideal classroom would have the students probing, testing and modifying the conditions of their own experience ... and collectively at that. When infrastructural reflexivity takes hold, the game becomes, in part, a metagame (in Stephanie Boluk and Patrick Lemieux's sense) of testing the conditions under which the game is even possible, as students attempt to discover "the diverse practices and material discontinuities that emerge between the human experience of playing video games and their nonhuman operations" (Introduction).

In other words, infrastructural reflexivity is almost the antithesis of gamification. It doesn't wrap educational subject matter in features plucked from games to make it more attractive. Instead, infrastructural reflexivity draws attention away from the symbolic elements of game and classroom alike, focusing on the conditions of their possibility. What could otherwise be an annoying or frustrating impediment to learning suddenly becomes the focus around which learning occurs; infrastructural reflexivity puts the glitch to work.

As one result of this realization, creating some degree of infrastructural reflexivity in students is now an explicit learning outcome of the course. In order to get to that outcome, though, we have to count on our platforms to continue to break down in new and unanticipated ways.