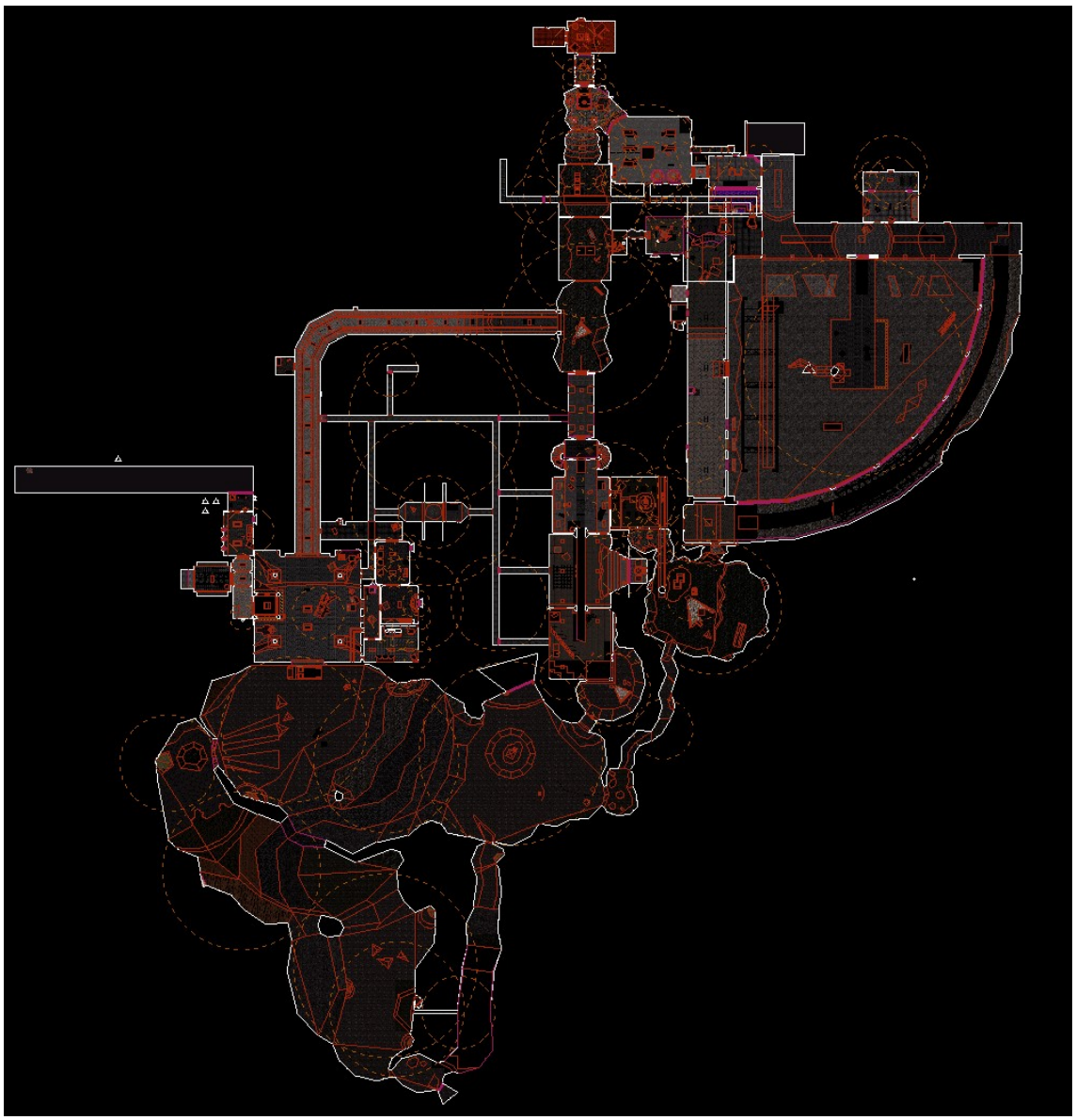
Congratulations!

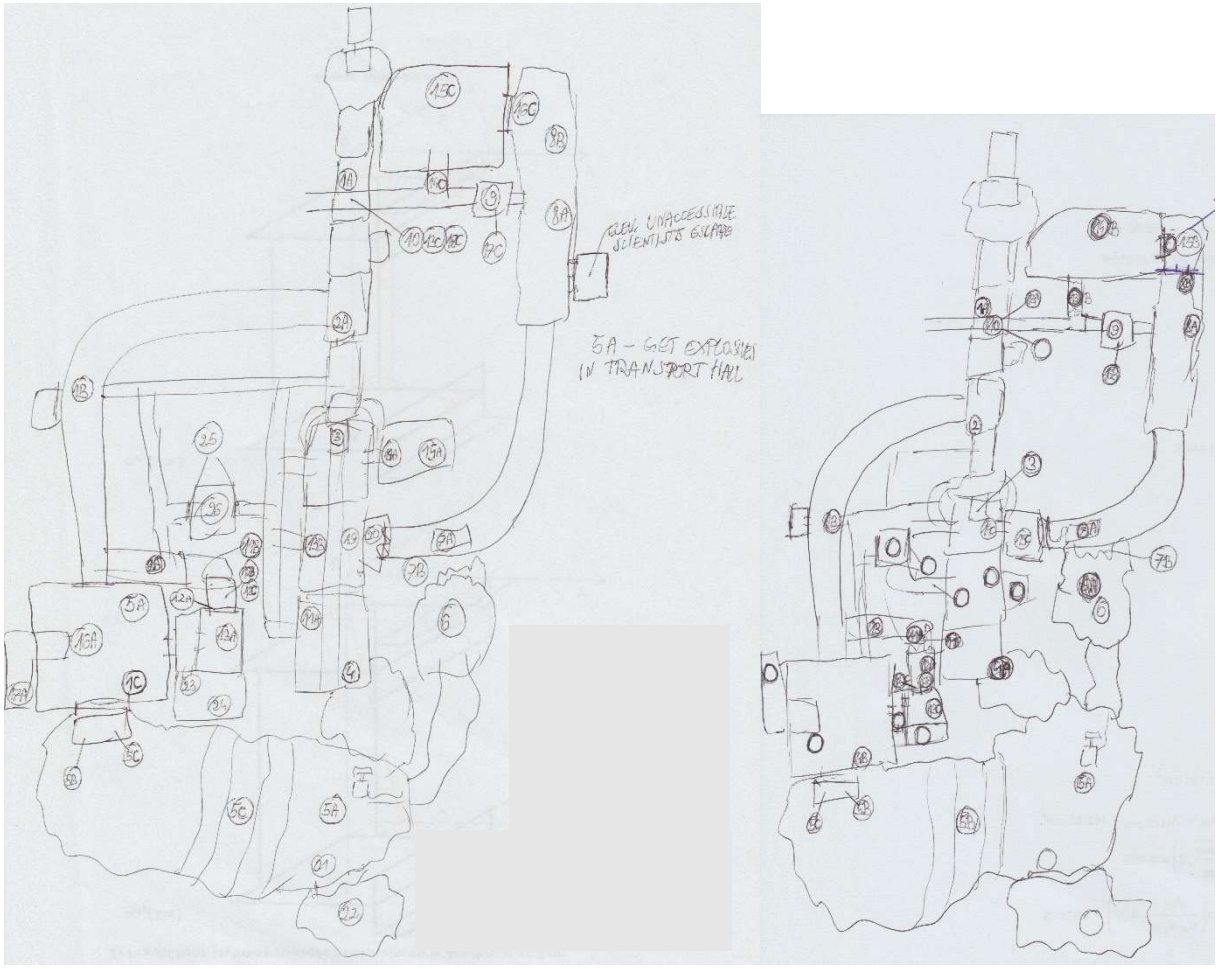
You have found the secret password that unlocks this file with extra bonus content. It contains three parts: concept art and inspirational photos used for designing the level, insight into some of the effects used and a short checklist of Easter eggs.

Part I: Concept art and real life inspiration for in-game design

I tend to draw a lot on random pieces of paper (usually one-sided pages from old work stuff) while mapping – it helps for general layout, design of particular elements and proper channelling of effects. It's just fast sketches, so don't expect any kind of professional art in here – but since I've preserved most of the stuff I drew during development of this map, why not share it? Most of it has been done during the early years of development. Some notes are in English and some are in Polish, there's no pattern there. Let's start with map's general layout – here's how it looks in the final version of the map (without the final area):

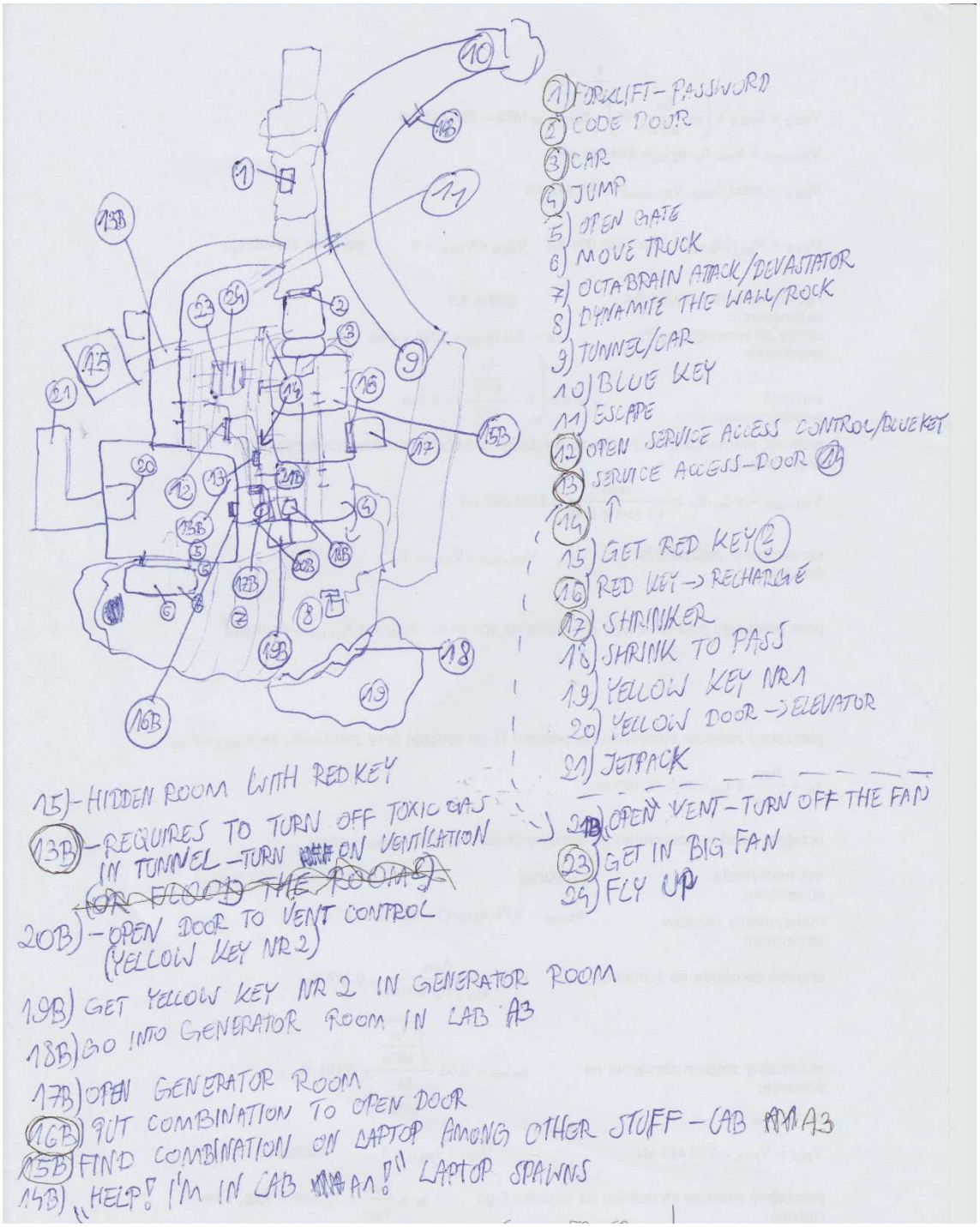


Here are some earlier iterations (largely not to scale) of the layout I've made mostly to grasp the idea of progression around the map. These sketches are from around 2009-2011.

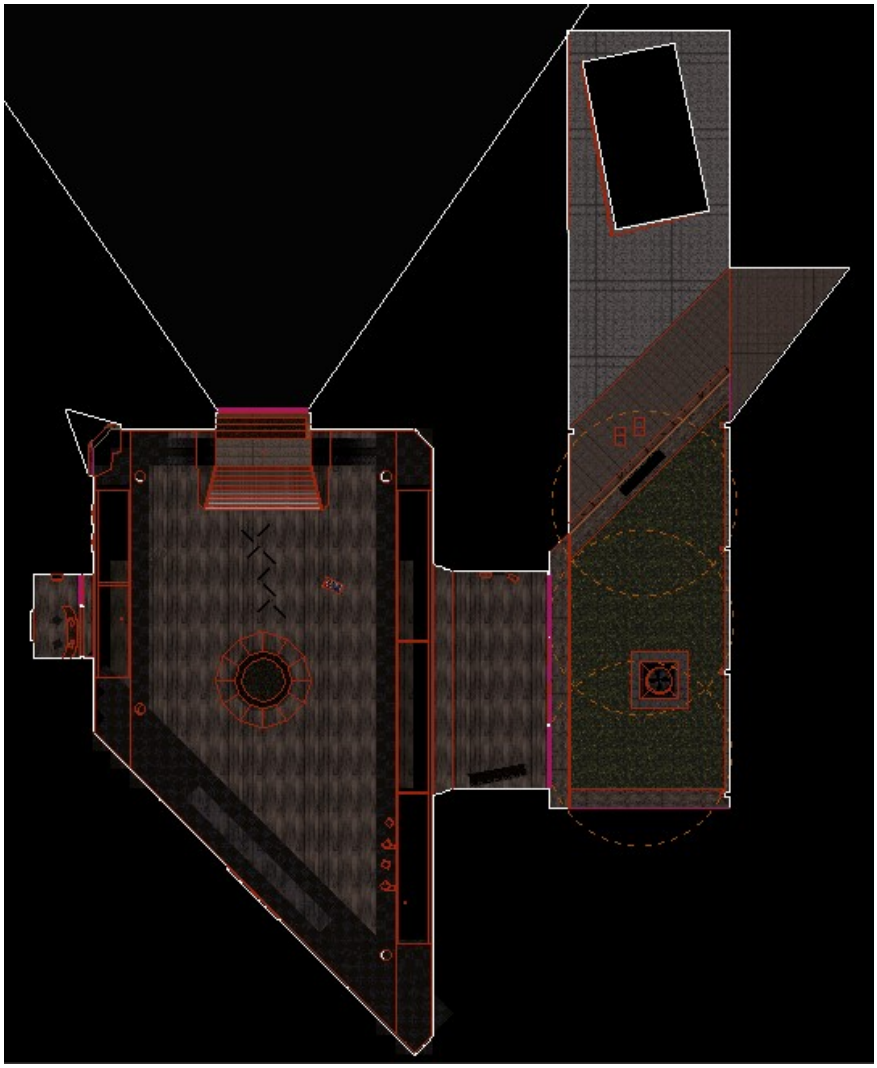


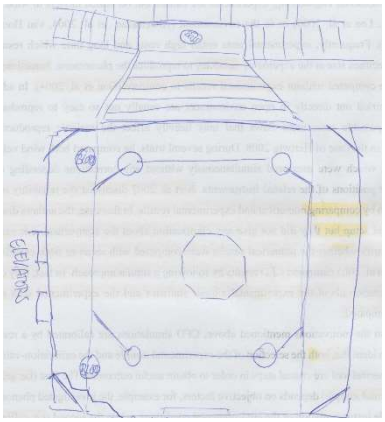
As you can see, these iterations lack some significant parts of the map – mostly the large cave in part B. I wanted to make the trolley ride there from the beginning and had a clear idea on how it should be done, just wasn't sure what should the environment around it be like. Also the whole part B is reduced to a single corridor where you would pick up the keycard and use it. Red key puzzle was originally planned to be in the same room where you turn off the toxic gas. Finally, the cave sections after you shrink were much less complex – in fact I only had the idea that they would require shrinking to access. These sketches also indicate that these parts – caves and whole section B – were the last ones added to the map.

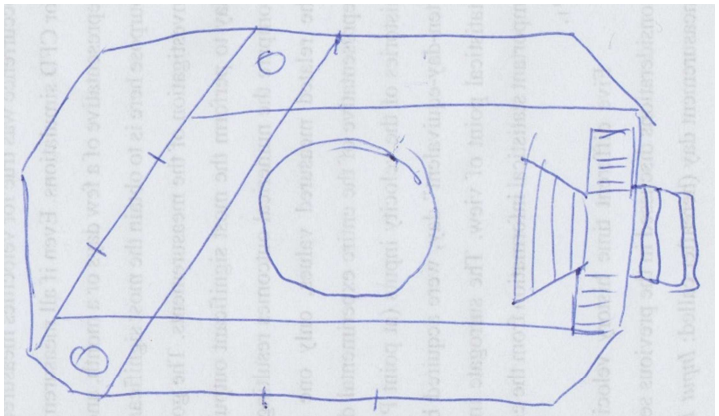
This is an even earlier iteration of the map's layout and progression, probably from 2009. Now even the lab with toxic gas isn't present, and that section is used as part of the semi-circular path for the trolley ride (yes, I've always really wanted the trolley to follow a path that wouldn't be straight!). Also, the rooms of part C were far from being designed back then and I had quite a different idea on how to make them. What's more interesting is the preserved "script" of the map – the final version required me to plan the script in a few parallel paths really due to all the junctions where you have multiple choices on where to go.



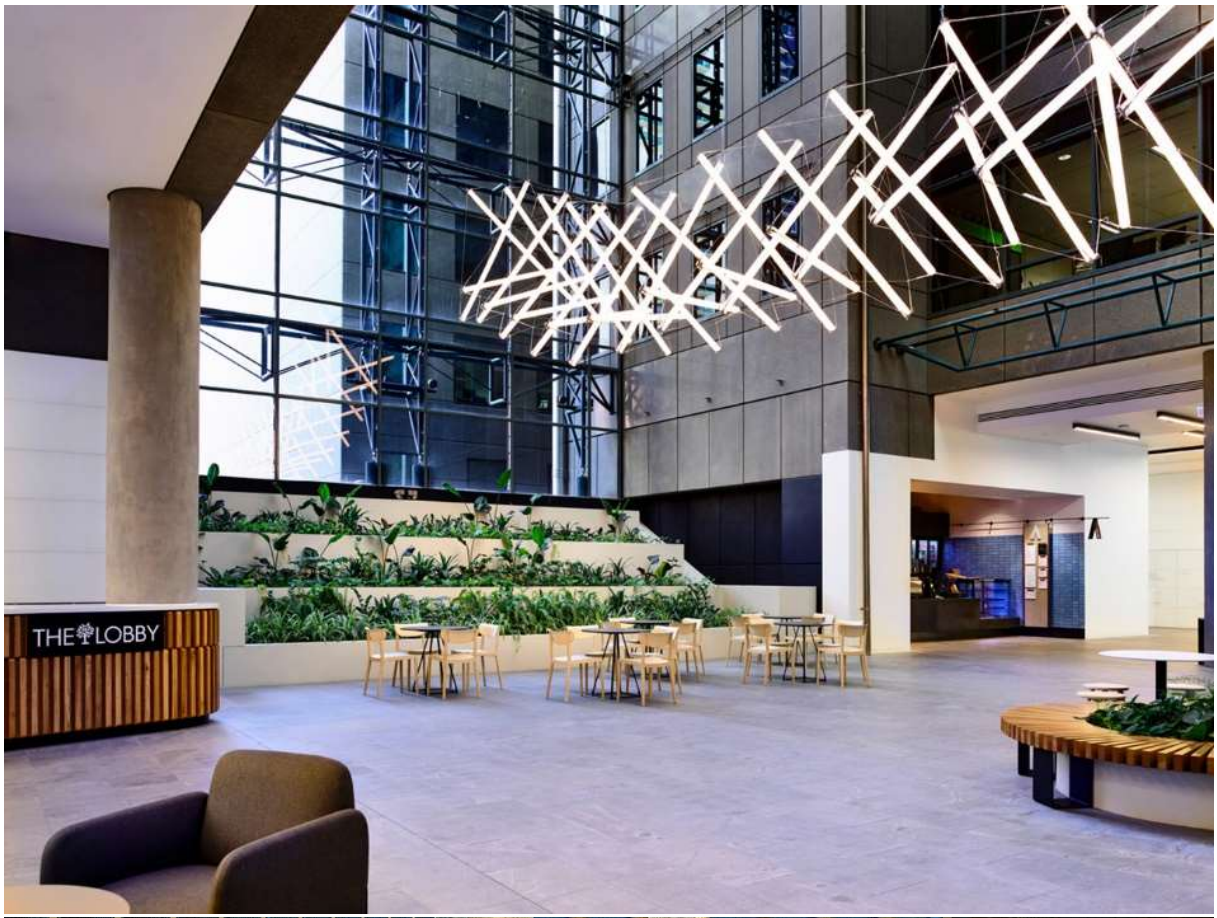
Designing the final area of the map also required some planning – I wanted to make an area that would be surprising and memorable for the player, but also serve well as the final battleground. There were some changes made after first beta-testing phase on this area to make the combat more dynamic (especially with more agile Cycloid Emperor). However, first idea was to make some demolished buildings where you'd be taking on all these bad guys – again, this is one of the last areas (or even *the* last area) I designed for this map, but I went for a modern corporate lobby – the 2D plan of final layout can be seen below:

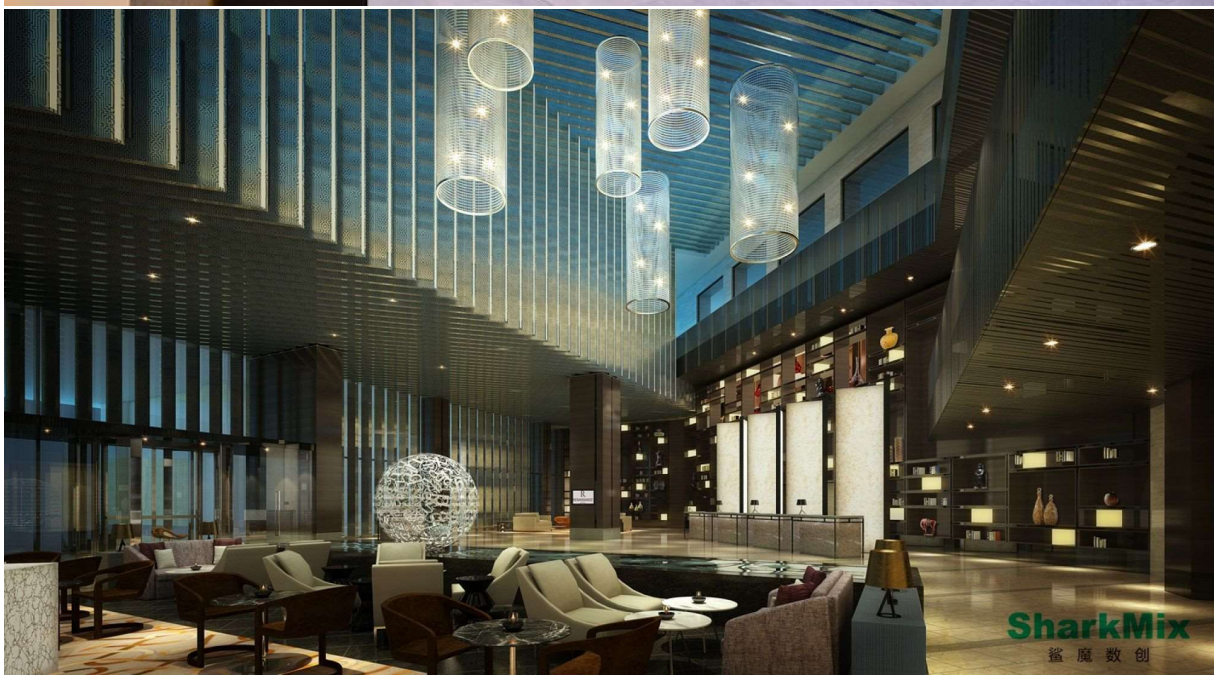


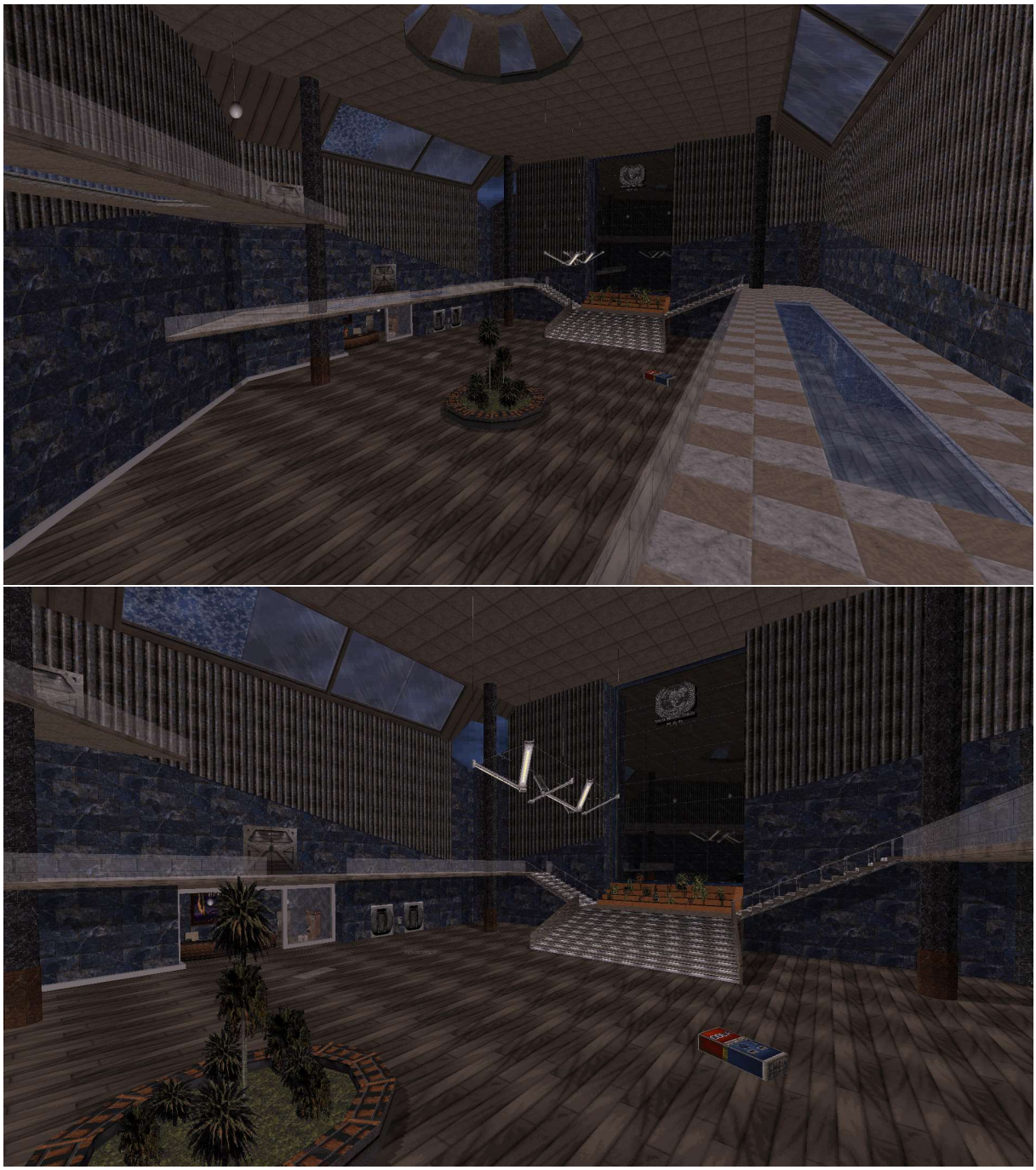




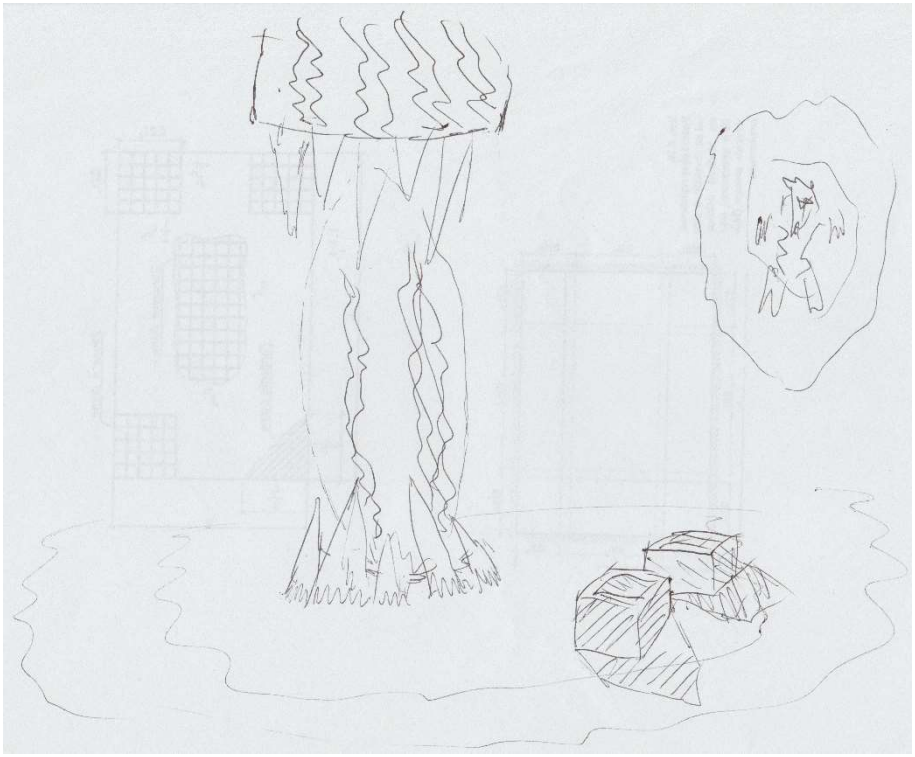
It evolved rather quickly from more rectangular schemes – lots of diagonal lines, both horizontal and vertical, in the end became pretty much the main architectural topic of this place. For the details, colours and textures I used some real life photos for inspiration – the impact of the two pictures below was the largest, although I intended to give them my own, more unique vibe.

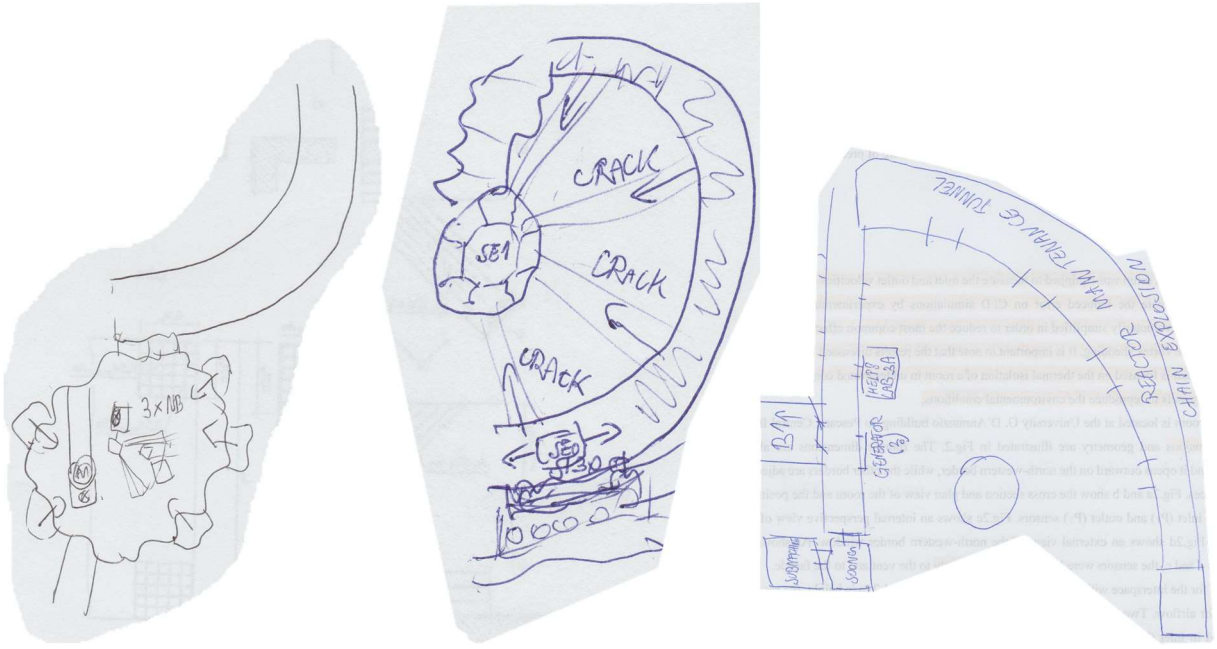






Another part of the map that required heavy sketching and inspirational photos was the trolley ride, but basing on the opinions of beta-testers, it worked well and I'm particularly satisfied with it myself. Trolley ride area is connected to the dark cave with dormant new beasts slowly awakening, which also required some sketches that are shown below. The first one is a very early iteration, where I was tinkering with an idea of large glowing alien structure in the middle of the dark cave, serving as the only light source. It wasn't linked in any way to the giant tentacle… Also yes, that is how I draw new beasts. Judging from the drawing on the backside of the page, I'd date the earliest of these sketches to be from 2011 or 2012.

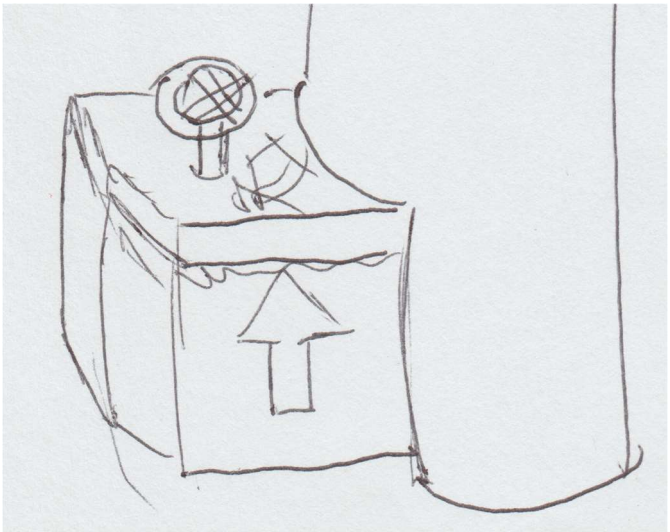


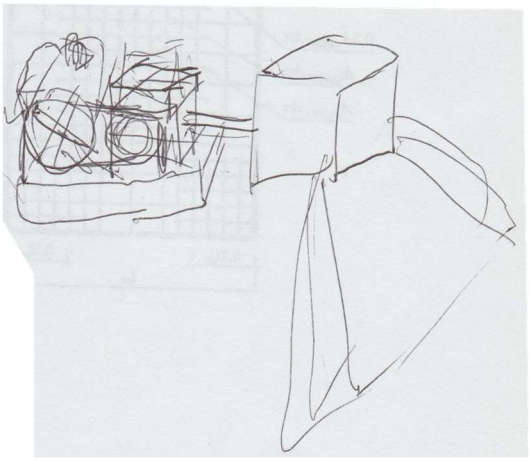


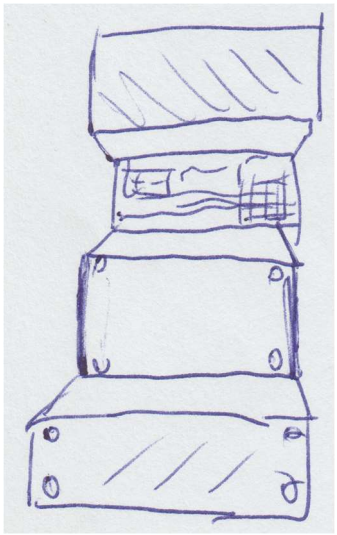
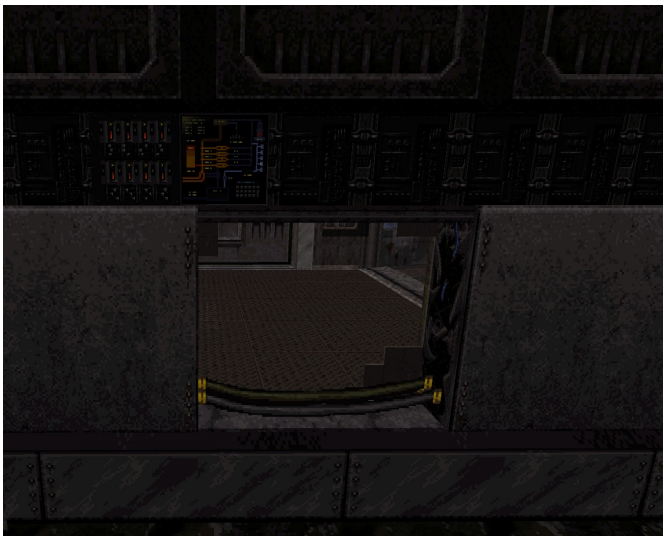
The first sketch above shows more advanced version of 2D plan of the dark cave and connection to the trolley tunnel. The sketch in the middle shows first iteration of how I intended the tunnel and ride to be like – I was planning to put some kind of reactor in the centre, which also got further expanded in the last sketch – the geometry is right on this one (besides the section B drawn here being a lot less complex than in the final version of the map), but the idea of making the large room in the middle as some kind of reactor was still present.

Below is an inspirational photo I used for the trolley itself compared with the final in-game design. Sadly, the yellow palettes in Duke shade in an awkward way most of the time…

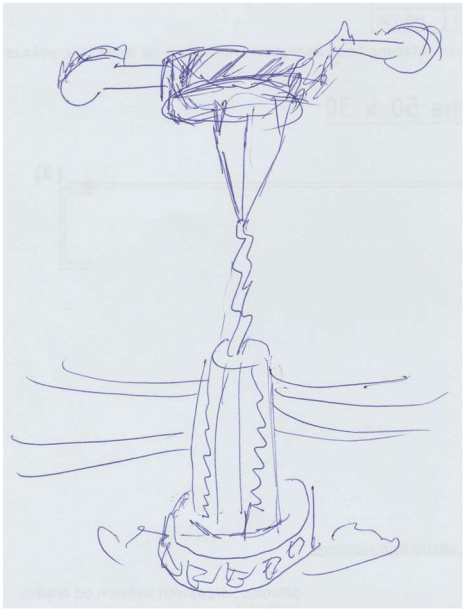
And now some smaller details, starting from the dark cave referenced above with their in-game counterparts:

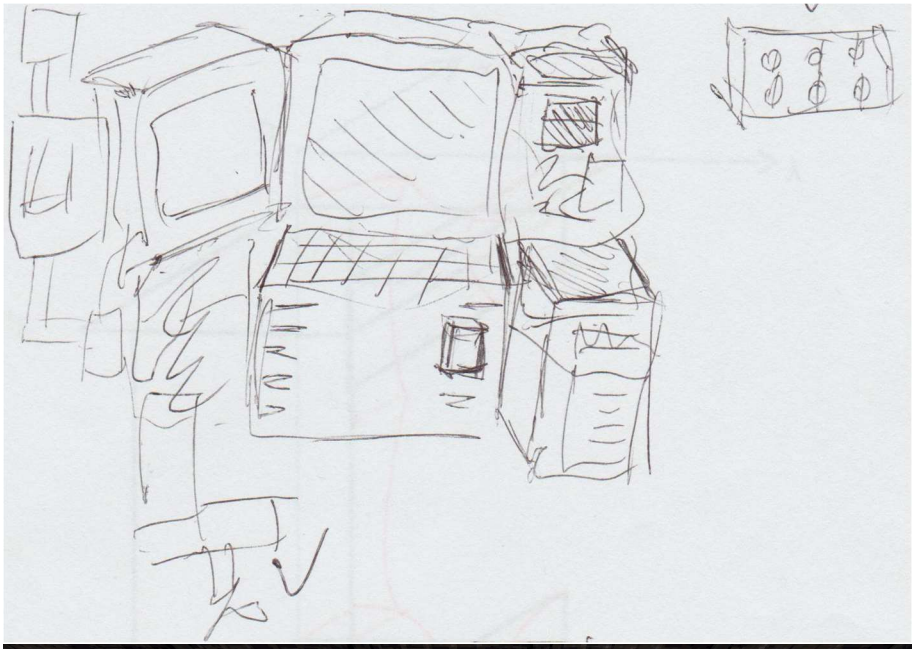


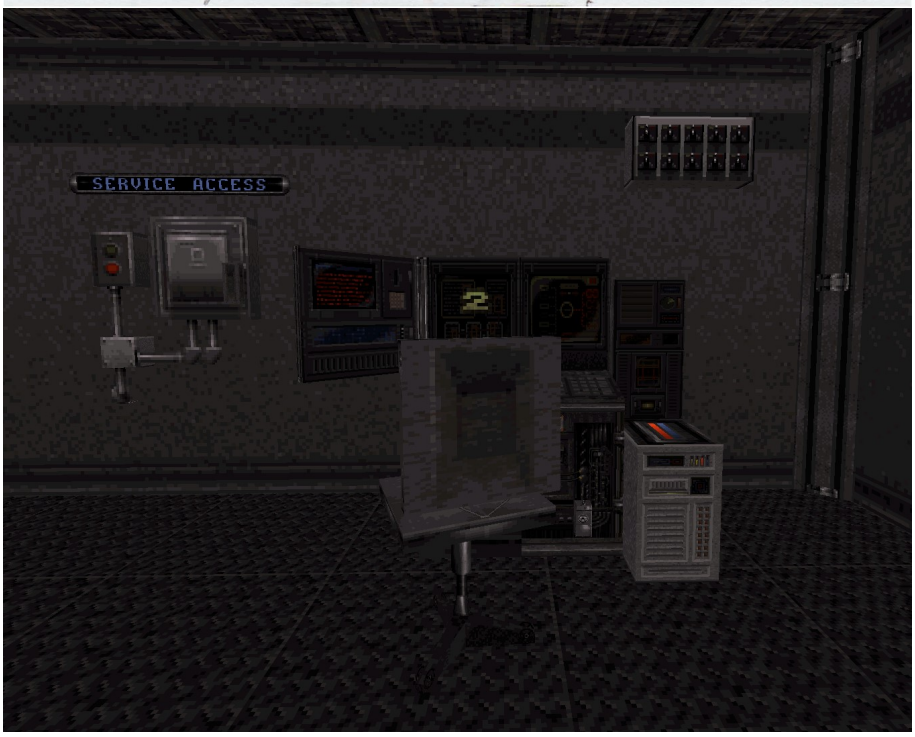


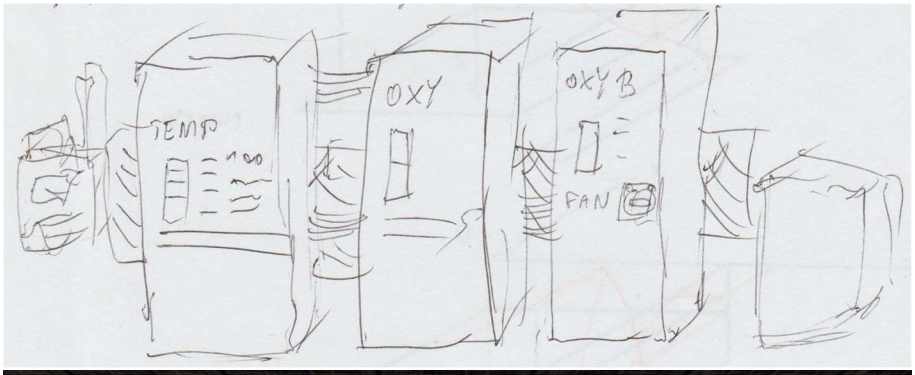
Details like machines from other parts of the map – good way to not make them look too generic is sketching them on paper first to know what effect you want to achieve in the end:

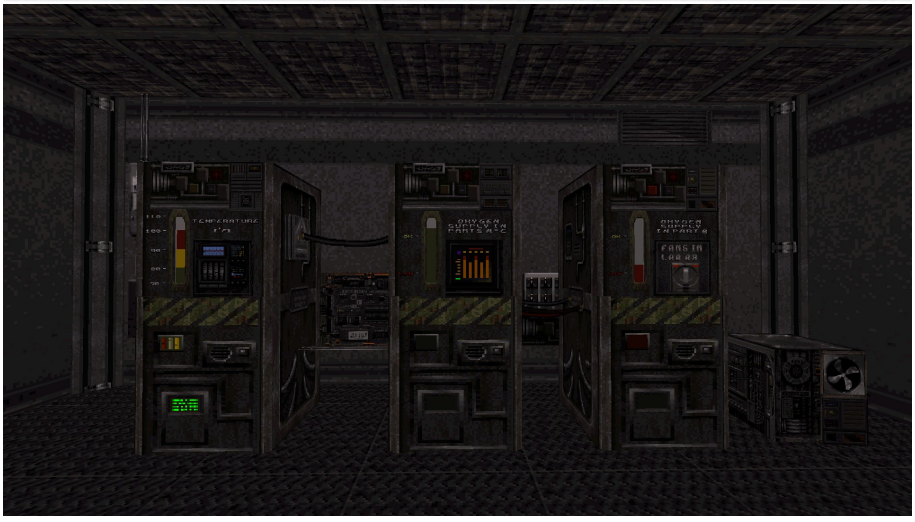


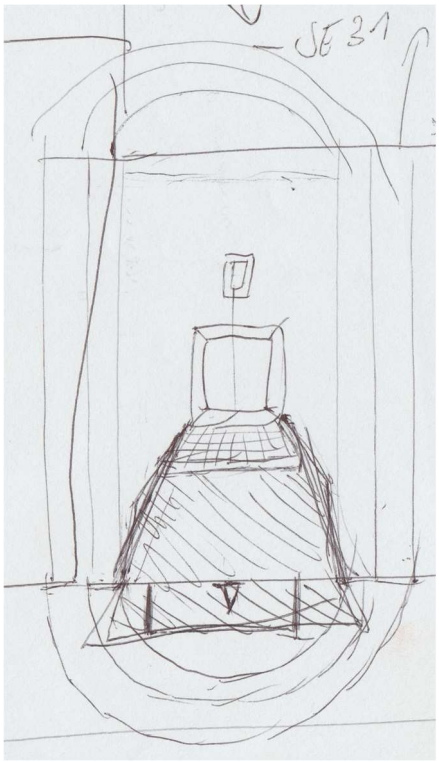



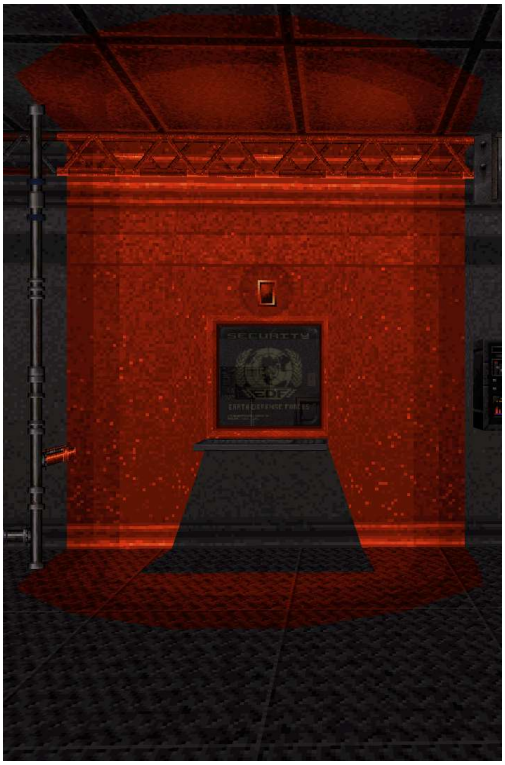


This room (C3) is one of my personal favourite rooms in the map, as it pretty much opens the largest, three-way progression junction (in addition to spawning a lot of monsters across the map). I also like it design wise – especially the control panels. Probably not many people noticed it, but after you turn off the main large fan in room C1, the temperature slowly rises, as shown at the console in the next screen.

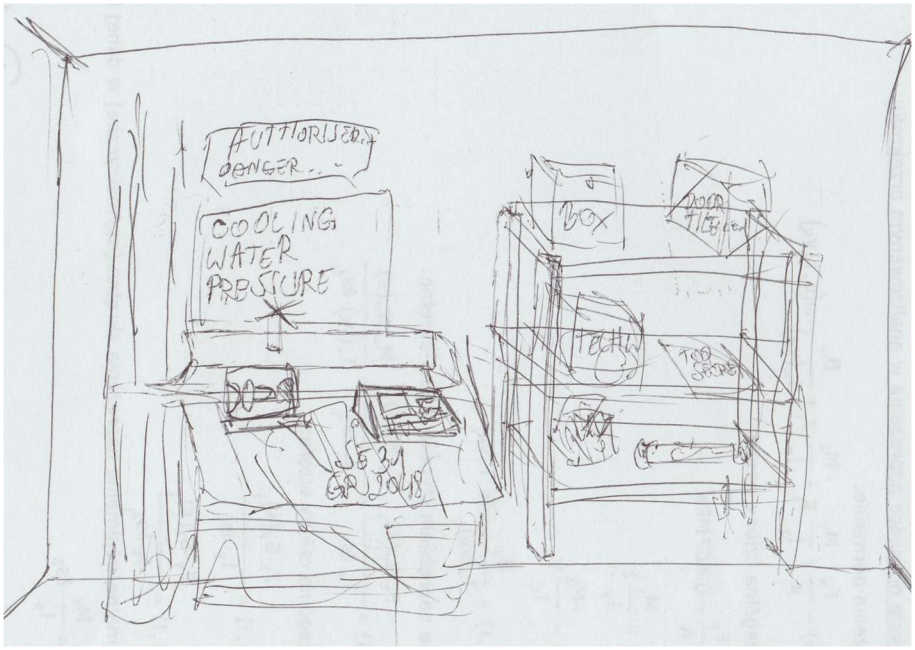

TEMP
OXY
OXY B
FAN

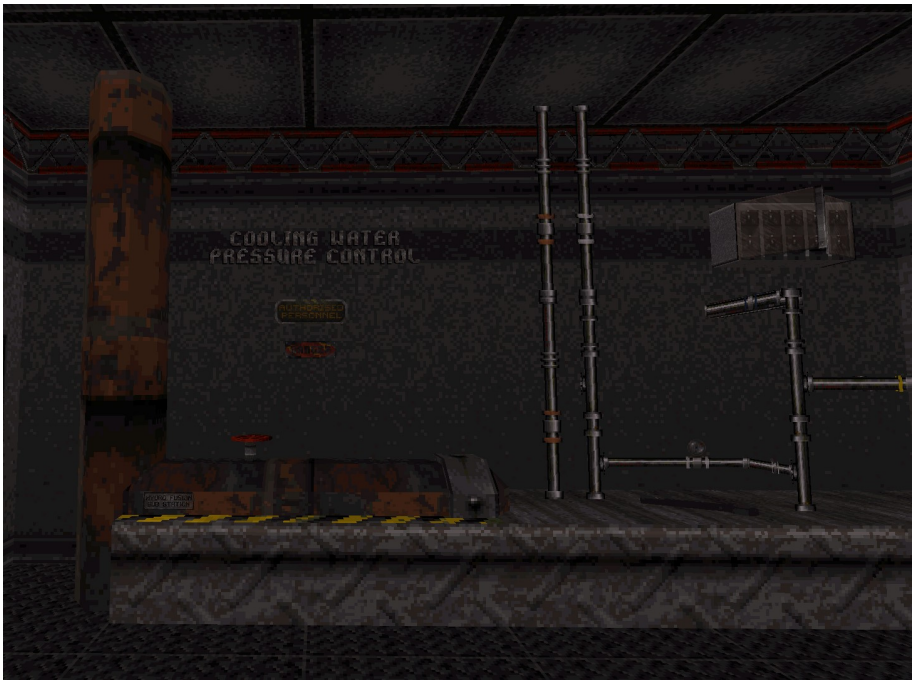


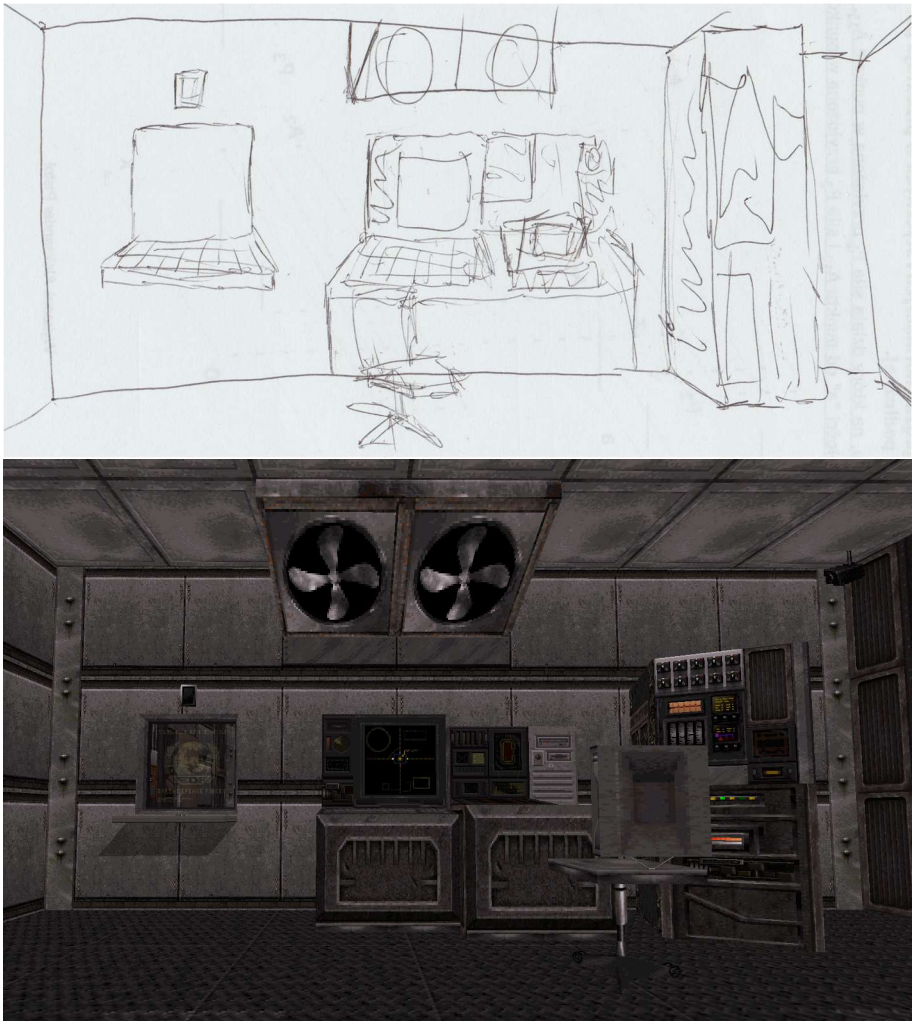




Room C2 underwent quite a bit of changes from the earlier version, especially the location of the valve you have to operate.



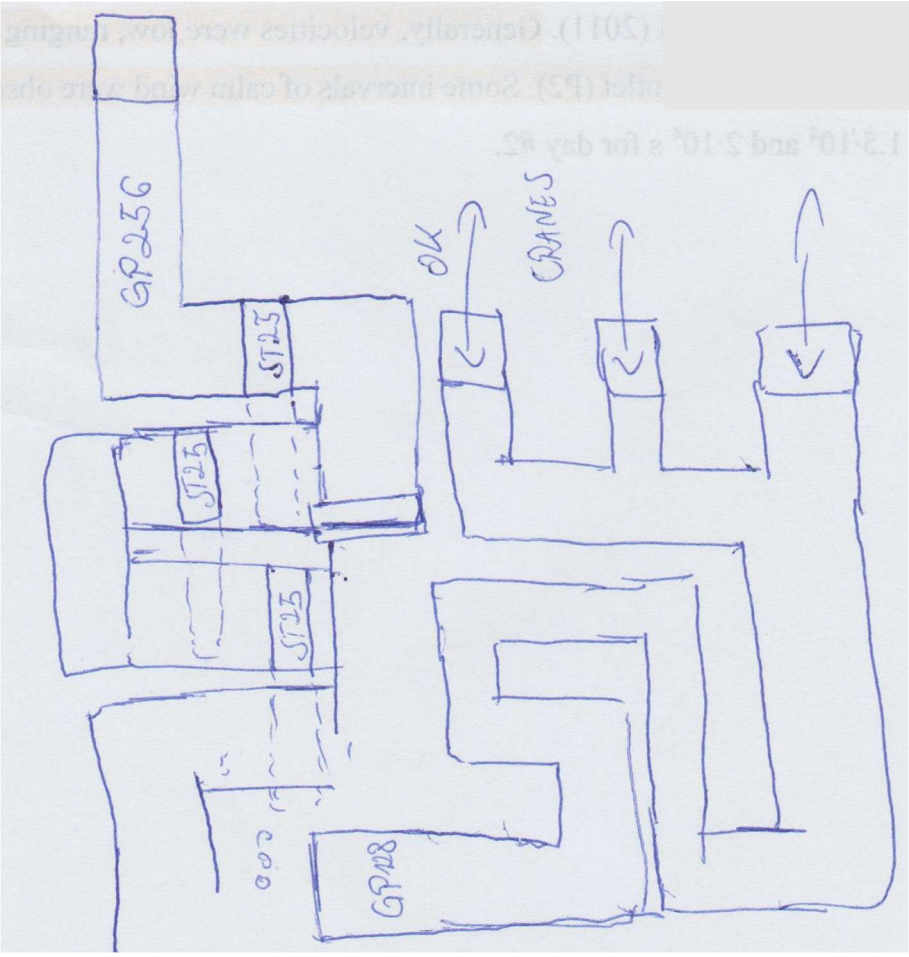




The cute little water distributor (did you notice you can drink water from it to refill health like from all the other fountains?):

The door between section B corridor and the large cave with tentacle were directly based on doors from Klingon ships in *Star Trek*: *The Next Generation*. The screen below is from S02E08: *A Matter of Honor*.

And finally, the red key puzzle/minigame, everyone's favourite part of the map. I wanted this to be challenging and difficult like old arcade games, up to the point of being annoying – but at the end giving lots of satisfaction and relief when you can finally grab the key. Also there's the bonus "challenge" to retrieve the keycard without intermediate saves – try it! Here's the general sketch of the path – the trickiest part was to think up some obstacles that would not be janky or unfair.



At the beginning I made the path with more turns, but then it got simplified for the final version. Also the path was widened 50% after the first phase of beta testing. I experimented with wall-aligned and view-aligned keycard sprite, but decided for the second option, as it is easier to track it's exact position this way. Also the conveyor speed was adjusted to not be too sluggish, but also not too fast. The trickiest part was probably making the card to respawn each time it falls in the acid – despite having a good way of arranging it, it would still sometimes bug for no apparent reason and not teleport back to the starting position, but after the changes and extensive further testing it seems to work in all cases now.

Part II: Insight into the effects

This part will probably be most useful for mappers or people with some different interest in the Build engine mechanics. Despite Duke 3D celebrating his 25th anniversary this year, there's still a lot of underexplored or plainly obscure effects that were barely used before or in completely different contexts to what can be seen here, most of them I've discovered by complete accident by experimenting with something else, then much later found some use for them in combination with other effects. I'd still encourage anyone more interested in what's going on here or how some of these effects work to just open the level in Mapster, but in case it turns out to be an intangible mess of randomly put sprites, overcrossing channelling web and seemingly random numbers that would collapse upon moving a single sprite, do read on below. There's some hints on both how to actually look at the map in Mapster as well as technical details about the more interesting effects.

First we need to establish two general techniques which are extensively used within this map. The first one of these is making the sprites to be "independent" of their physical location. One way to do it is using the offset parameter that can be easily edited through F8 menu in 2D mode. However, it has certain limitations: there is a certain limit of the offset we can set, it usually requires some experimentation to get the correct offset value, the sprites still have to be placed within the boundaries of the sectors and offset takes the same part of the sprite structure as slope, so both can't be used together. It is sufficient for some simpler stuff and also has some advantages over the 2nd method I'm going to present (offset sprites can be easily moved around with no need to edit them again). However, what I'd really like to concentrate here is manually editing *sectnum* values of the sprites.

This value essentially indicates which sector the sprite belongs to. In order to be able to tinker with it, you have to edit Mapster settings in file *mapster32.cfg*, precisely the lines:

```
; Fix sprite sectnums when saving a map or entering 3D mode
fixmaponsave_sprites = 0
```

The last value has to be set to 0. Now you can edit sprites' sectnums through F8 menu in Mapster by manually typing the number of the sector you want your sprite to belong to. This gets reset only when moving the sprite around – but if you move a selection of bunch of sprites together, their sectnums *usually* do not change, so you have to be careful with it. Also remember the engine will not render the sprites unless the sector they belong in can be seen!

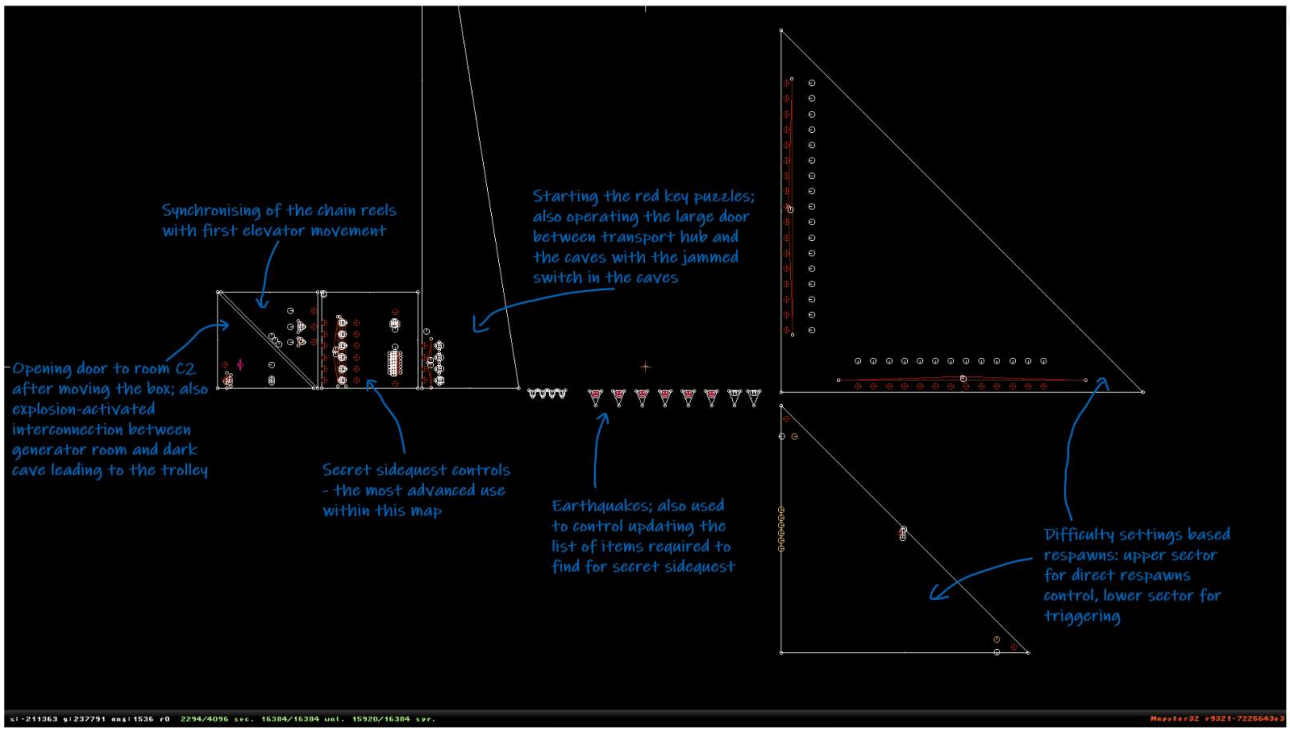
Now here are the potential uses for this technique:

- Rendering the sprites which are physically placed in sectors smaller than the size of the sprite, so that they would disappear from certain perspectives – now they can belong to larger parent sectors around to mitigate this problem;
- Having sprites that are located outside the boundaries of the sectors, for example flat sprites used as parts of spriteworks that would only require parts of them being visible;

- Mitigating the palette/shade/effectors effects from the sectors, so they do not apply to the sprites – or on the contrary, getting the sprites to be affected by an effect from another sector!
- Saving resources with more complex effects – we'll get to that later.

The second technique is the shooter/target logic, which can be used for more advanced activation of effects in Build engine. This technique has been introduced to me by **HighTreason** and the best way to know what we're talking about here is to just check his tutorial video on the technique itself: https://www.youtube.com/watch?v=yfspllLqw50.

This is the logic gates setup from *Submachine*, with small descriptions suggesting which part is controlling which effect within the map:



The only problem with this method is how resource-hungry it can become for more complex effects, especially when it comes to walls. You also want to have these controls outside of the player area, so nothing gets activated accidentally, and pretty far from the player area, so the player can't hear the annoying shrinker thuds. The difficulty settings based respawns controls have 17 triggers for *Let's Rock* and 13 for *Come Get Some*, so it would effectively require at least 180 walls if I used a new, triangular sector for each one of them.

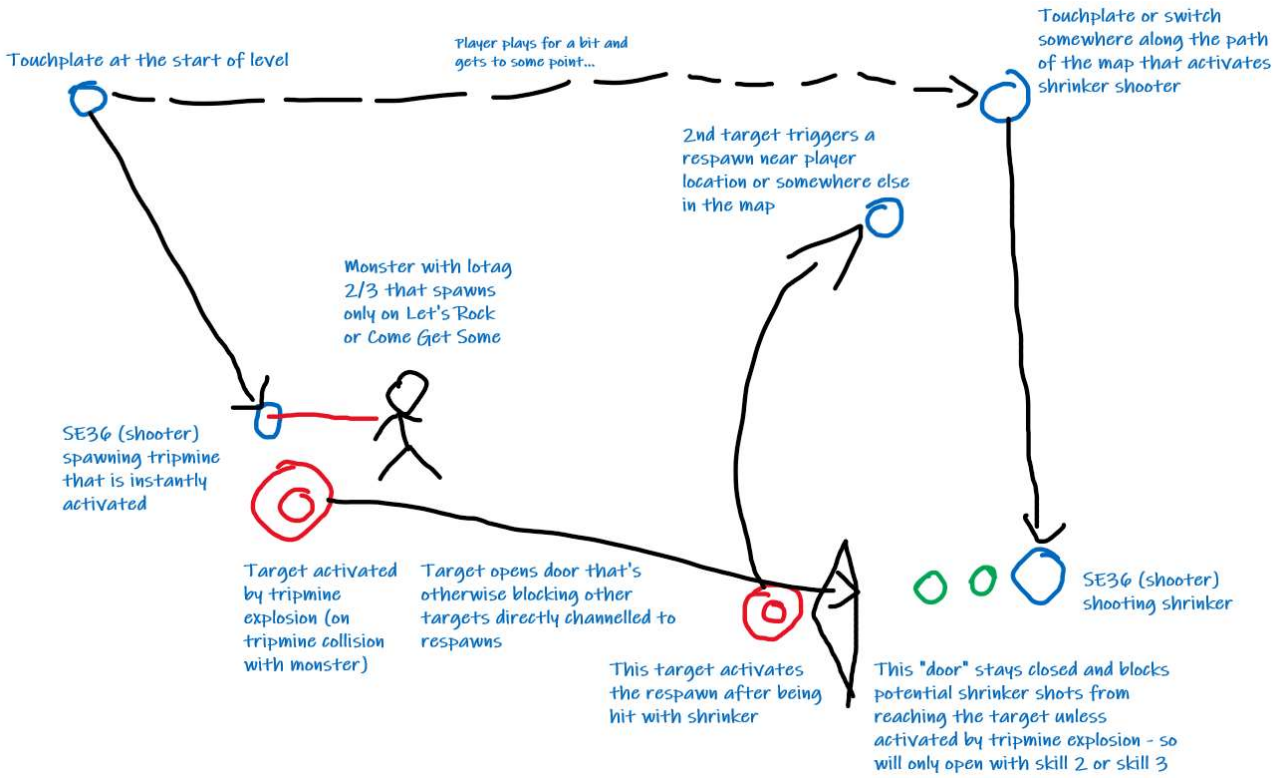
Instead, the reasonable thing to do here was to implement it together with previously mentioned technique of changing sectnums – so activators/masterswitches that control the triggering of the respawns and speed sprites that control the type of projectile used are placed in already existing sectors, while only the actual shooter sector effector is placed near the target, but technically belongs to the sector with activators. The projectiles in this case will still appear properly and hit the targets as intended, but with large savings of the

resources! The only thing that has to be kept in mind here is that the physical positions of floors and ceilings of both sectors have to correspond with each other, otherwise the projectile shot from the sector effector shooter would appear within the floor/ceiling and not be able to hit the target.

Now let's move to the specific effects used in the map. As I've already started elaborating on this topic, we'll start with the **difficulty settings based respawns**.

As we know, it's easy to set difficulty settings on monsters placed within the map with proper lotags – but normally it can't affect the respawning monsters, which would appear always no matter the difficulty setting. Considering that more than half of the monsters in this map are respawning, it would be pointless to introduce difficulty settings like that. Also, one of my principles was to make the easiest difficulty setting completely monster-free, for people who would just want to concentrate on solving puzzles and calmly enjoy the map at their own pace without having to worry about combat.

The trick I've implemented here is pretty easy – having dummy monsters that would activate the respawns respectively for *Let's Rock* and *Come Get Some* by triggering a tripmine explosion. The sketch below should better explain it:



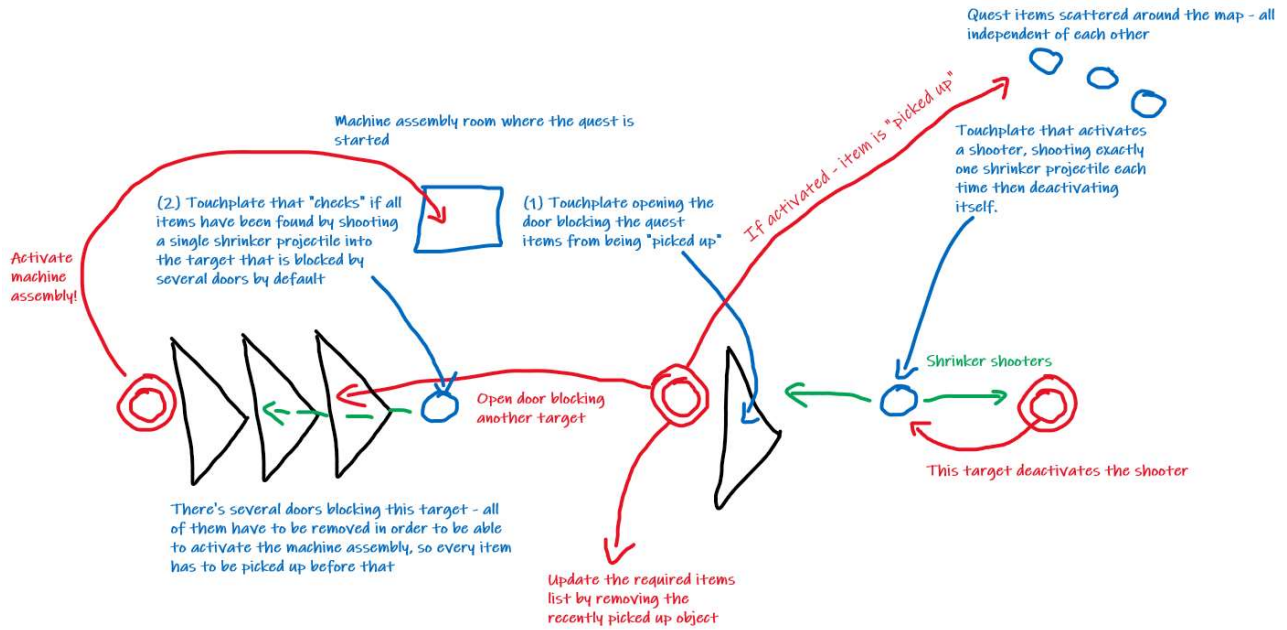
As mentioned above – all the shooters belong and are controlled by activators in a completely different part of the map. You might want to check these mechanics within the map if you will be willing to implement it somewhere yourself.

Moving on to another effect, also one that makes heavy use of the target/shooter logic and perhaps my personal favourite thing in the map – the **secret side quest**. If you have found it, you know the drill – you have to find a number of items scattered around the map in order to assemble a machine which will serve as a powerful weapon. These are the basic assumptions for the quest:

- The quest items are present in the map all the time, but can only be "picked up" when the quest has been found and started;
- The quest items can be collected in any order or at any moment during the progression of the map;
- The machine assembly is only activated once all the items have been found;
- The list of required items will update to keep track of player's progression with the quest and indicate what else has to be found.

I don't want to spoil the locations of the quest start and the items or the reward for the quest. I will instead concentrate on the mechanics. If you are willing to complete the quest on your own, you can safely read further in this part, however I'd advise against checking the map in this case, as you risk spoiling yourself the fun.

Ability to pick up random "items" which are actually regular sprites has been done numerous times in user maps before – most notably by Bob Averill, Alejandro Glavic or the Oostrum brothers. The easiest way to do it is probably using SE31 with very fast speed that would look smooth. I will not elaborate more on this thing alone – the more interesting part is to have the pick-ups trigger only after starting the quest. Again, all the mechanics are better explained on the sketch:



Updating the list of required items could have been done again with SE31 – but that would require a lot more resources (read: walls) and would not look that smooth. Instead, I used here a shooter that would blow up a flat sprite with a hitag channelling to the hitags of sprites

it should remove in the list. Again, this all has to be done in remote locations, in this case I re-used the sectors initially made for earthquakes. Shooter will shoot a single mortar that would destroy a dummy sprite and trigger a target deactivating the shooter. This dummy sprite has the same hitag as the sprites that produce the name and image of the quest item on the list, so destroying the dummy sprite will also destroy the sprites on the list, despite the fact that they're not affected by the explosion. Technically, they could be also simply blown if the player would be able to reach the sector with messages – but he can't!

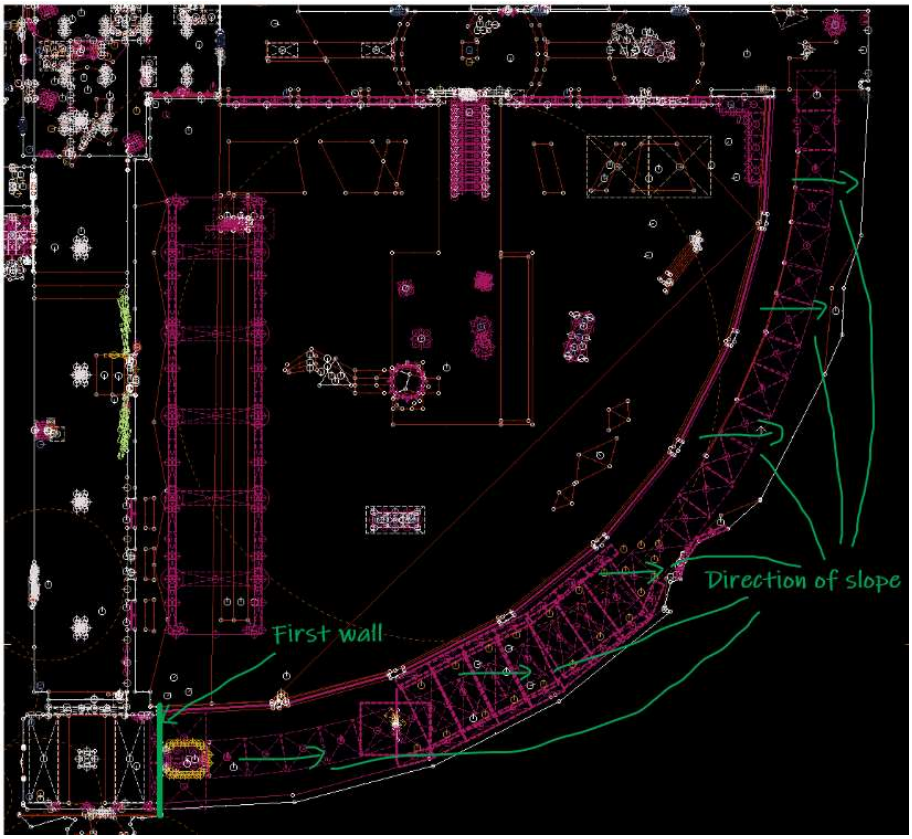
Another effect that I'd like to dismantle here is the **trolley ride**. This is essentially based on the *rotate rise bridge* effect, which was used for example in the main game in E2L8 behind the blue key transport, in the area with Octabrains and Sentry Drones before the reactor. It requires SE0 inside the sector linked with SE1 as pivot point, and ST30 at the sector. In general, this effect is super useful as it can also act basically the same as swing door, but without the annoying auto-closing when bumping into an obstacle. It has certain advantages over swing doors: the rotation degree can be adjusted and most important one: it can move sprites! The only potential drawback compared to swing doors is that it requires activator/masterswitch for activation.

Upon activation, the sector will rotate around the pivot point, and its floor will rise to the height of SE0 sprite (in case of using it instead of swing door, you generally don't want that to happen, so you can simply put SE sprite on the floor). SE1 sprite angle determines whether the rotation should be clockwise or counter clockwise, while SE0 angle determines if the effect should rotate the *sector* or *points* around pivot point. Speed sprite determines the degree of rotation (256 being equal to 90 degree and default value). It also determines the speed of rotation – no matter what value is put, the rotation will always take the same amount of time to complete, so larger values mean faster rotation. However, the rising of the sector floor isn't exactly synced with the rotation, at least in most cases. By the time the rotation has ended, the sector floor will rise/lower by exactly 32 "page ups" or more precisely, 32 768 Build vertical units. If the gap between floor and SE sprite height is smaller than this value, the rising will end before the rotation, and if it's larger – the rising will continue after the rotation has ended.

In this case, both movements had to be in full sync. Luckily, the height difference was pretty much spot on (despite me building parts on either side of the trolley path before), so I had only very little adjusting to do. The main problem was the slope on a curved sector (the whole path had to be a single sector, without red lines that would be crossed by the trolley). The picture below might be better as an explanation.

To put it simple – at the first part of the path, the sector will slope properly, but at the end, it will be sloped in the same direction, so to its side walls and not really matching the way of trolley. What helped in this case was masking the path along the way on the floor and ceiling (which had to be done anyway due to sector below and above the trolley being literally black empty holes) with sloped sprites that could be precisely positioned and adjusted for proper direction. Then all that was left to do was making sure the speed is fast enough so any desynchronization along the way is unnoticeable to the player and when the trolley ends its journey, it is properly stabilized on the ground. Stuff like explosions, collapsing tunnel wall

or destructible scaffoldings on the way were just cosmetics and didn't take anywhere near as much time as the trolley ride itself to prepare – but seems it's one part of the map that every of the testers memorized quite well, so was worth the effort!



Also some credit should be given here to **Tim Conneen**, author of an old "Stupid SE0 tricks" for inspiring the idea of a vehicle driving along a curve.

We'll soon be taking ST30 tricks even further, so stay with me. The next section will be about the **sound effects** within the map, particularly ambient sounds that can be turned on and off.

The first way to do it is by combining together an ambient music & SFX sprite with activator locked. Music sprite should be set just the normal way you'd do for ambient sounds (lotag for sound number, hitag for range), activator locked should be linked to the switch/touchplate and voila! The ambient sound will only be hearable upon turning it on. Of course, the sector these 2 sprites are in has to be void of any other effects, as they would take priority over these sprites' functions. This is an easy method that is seemingly rarely utilised by mappers, but it has one significant drawback: the sound can be turned on easily, but it will only be possible to turn it off if the switch (and most importantly, player) is beyond the sound's range. In this map, it was used for main fan sound, alarm in the generator room (can be switched off with emergency switch in the corner of the room) or fans in the jetpack corridor.

The real deal, however, is the **working 4-channel radio** in one of the secrets. Since you operate it directly in the same place where the ambient sounds can be heard, the method

described above was out of the question. Now, the other methods described below are only valid for certain sounds – the ones that have repeat flag on. This can be checked in F2 sound menu and is indicated by the letter "R" at the beginning of flags listing next to the sound's information. These sounds would loop infinitely.

The first idea was to use 2-way trains, which can trigger these sounds with a nice fading effect. The problem here was, the activation of the sounds had to be *before* even reaching the radio, so had to be linked to a few triggers before. It worked well – as long as the game was saved and reloaded between the triggers. Otherwise, we would just get a noisy cluster of all sounds together. Also, this method required a lot of space and adjusting the volume was quite impractical. Nevertheless, one of these trains stayed just to transport the speaker which plays muzak from E4L3.

For other sounds, I had to rely once again on ST30 tricks. Or particularly, on a completely obscure one. Now we know that we can activate rotation of ST30 with an activator or masterswitch… But what if we put an activator locked in that sector? First thing to notice is, the sector will just rotate infinitely like usual SE0/SE1 combination, then stop upon *activation* of activator locked. Activating the activator here would move the sector by certain number of degrees (set by speed sprite) then stop. Both activator and activator locked may also link to different channels. Now another anomaly here is, despite having a properly installed SE1 as pivot point, in this case the sector will sometimes rotate around this pivot point, and sometimes rotate around… *something*. To make it work properly, the floor of the sector (and SE0 sprite) have to be set at Build's 0 – no more, no less. Why? Because it's Build logic. This effect is generally interesting and I'm sure there's a lot more potential applications for it.

This is an effect I've discovered by accident while experimenting with something else a while back. Now what does it have to do with ambient sounds activation? Simple as that – putting a music sprite with one of "R" flagged sounds tagged, dummy activator locked that isn't linked to anything and normal activator linked to the switch to operate the effect in a sector prepared as described above will make it constantly play the sound upon activation, then stop upon deactivation – simple as that and works as smooth as possible! Also adjusting the volume is much easier in this case.

While we're at obscure Build effects, let's move to the **gantry crane** in the tentacle cave. Personally, I dig steel structures and gantries, so making one in Duke was pretty cool! Now how does it work?

In this case, it's a combination of SE30 – 2-way train and SE31 – floor rise. That and a lot of cosmetics, including messing with sectnums. The principles are as follows:

- SE30/ST31 will carry sprites, providing they are in the main train sector, so the same one as SE30,
- SE31 can be combined with ST31, providing it is *not* in the same sector as SE30…

Or really, is it like that? Try it – it will not work, unless you use masterswitch instead of activator for some reason. This mechanic is also used in the tram that travels through Lab A1-A3 (for the fence that closes behind the player trapping them inside).

If that's not enough weird build stuff for a single effect, we need some more stuff. The gantry travels horizontally first, and *only then* it lowers down its cargo. Here's where SE31, the most reliable pal of every mapper, comes to the rescue. Have you ever set a hitag on a SE31, then wondered why it's not working…? Well, it is working, but the hitag of SE sprite gives the effect a delay! Yes, a delay that is independent from a masterswitch or any other effects combined with SE31. Simple? Well, it comes at a price – the delay will only work upon the 2nd activation for some reason. In this case, the obvious solution was to put the gantry at the lower, final position in Mapster, then make a dummy activation of it at the beginning of map, so pressing the switch to operate gantry actually is the 2nd activation.

The cosmetics involved – the gantry consists of a total of 3 2-way trains really, two of which only house chains that lower to different heights to imitate the pull effect. These chain sprites are physically placed all within the main gantry sector, but their sectnums are manipulated to belong to other train sectors. Also, incorporating sounds for the effect into this whole cluster is not the best idea – I just used a dummy swing door to provide the operation sounds here.

There's a lot more pretty rare and unique effects within this map, mostly a smaller scale ones though. For example, I always liked to experiment with **different door types** – like the Klingon door mentioned above, the diagonal split/slide door leading into the lab with red key puzzles, transparent sprite-made split doors to the jetpack room and armory or the complex door effect that led you into the secret hiding the password to this archive. I also like experimenting with more unorthodox types of **dynamic door lights** (prong teeth door to the corridor with blue key, double slide door into mirror floor room in section A, split door into red key puzzle room or pretty much any swing door in the map). These effects don't really use any advanced quirks and can be studied and disassembled easily just within Mapster if anyone is curious about them. And final hint – slide doors are just perfect for moving vertices around in the map! For example, this is how the tentacle movement was made.

Part III: Easter eggs and references

This part simply lists the references and Easter eggs as a checklist – the whole fun is to find them in the game yourself!

- *Companion Barrel* achievement in *Duke Nukem Forever*;
- Legendary Duke 3D user map by Bob Averill, *BOBSP2: Paranoia*;
- One of the more influential Duke 3D mapper of the early 2000s, Kevin Cools/Kef Nukem/Kuffi;
- A *Star Trek: The Next Generation* character;
- *Shining* – alright, this one is as obvious as it gets…