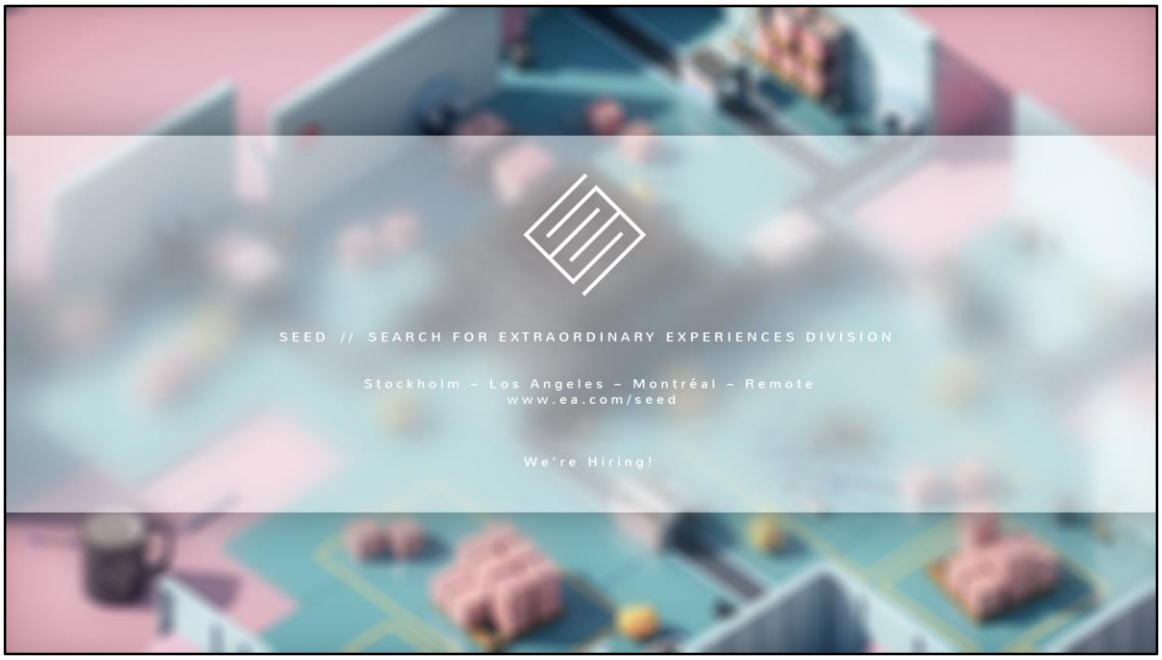
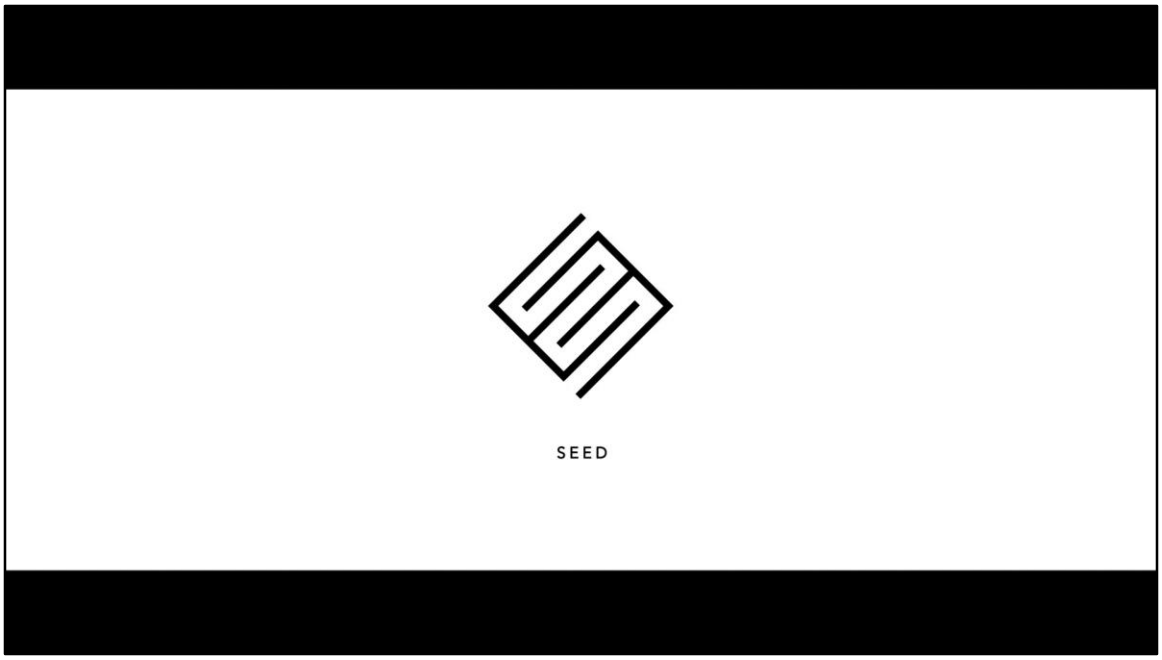
Hi, everyone. First of all, I'd like to thank SideFX and IndyZone for inviting me to be a part of CEDEC; It is absolutely exciting and still hard to believe that I'm half across the globe giving a presentation in Japan. But without further adieu, I'd like to welcome you to the topic of today, which is "Functional Symbiosis of Art Direction & Proceduralism".



For those who do not know me – my name is Anastasia Opara, very happy to meet you. I'm a part of SEED, which is a research division of Electronic Arts. We are a cross-disciplinary team with a mission is to explore, build and help define the future of interactive entertainment via creative & technical exploration.

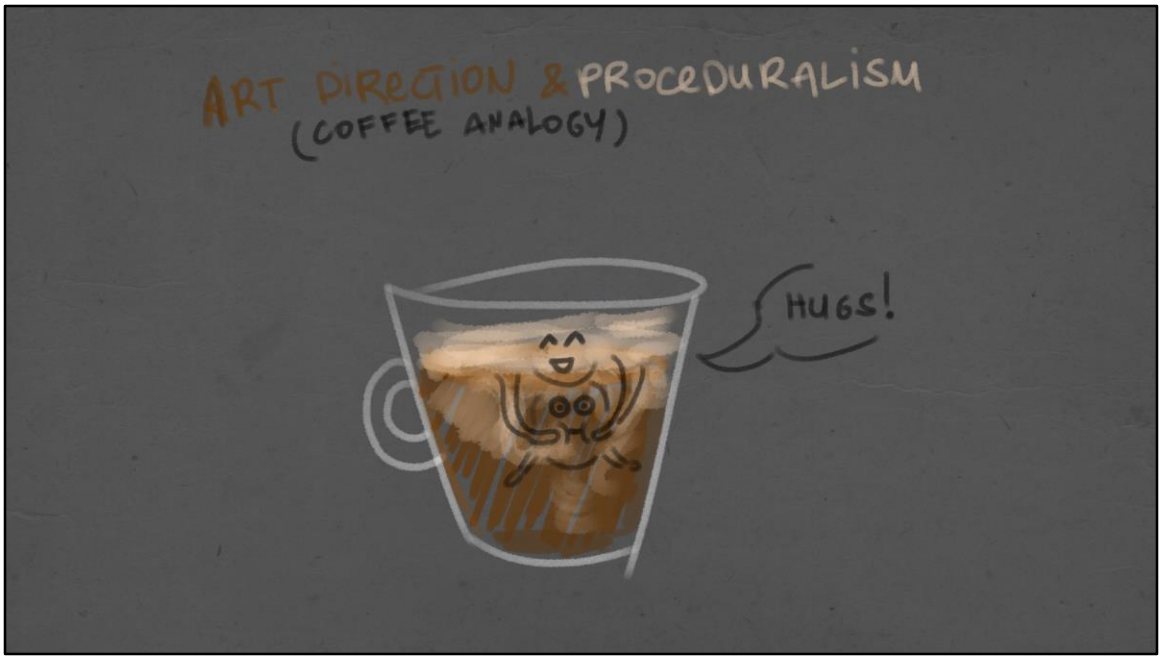
[30sec]



And the project that we are going to be using as a basis for today's discussion is Pica Pica.

It is an exploration of hybrid real-time raytracing rendering, deep learning agents and procedural level generation, the result of which is a mini-game for AI agents built in SEEDs *Halcyon* R&D engine, where AI learns how to navigate and interact with the environment via Reinforcement Learning.

Our ART team was quite small: we started as 2, and then 3 people, with about 2,5 month to go from initial brainstorm to finish. We all worked on the art direction while distributing other tasks among each other. I was responsible for the procedural aspects.



And pivoting art direction around a set of instructions or rules that produce final art can seem alien due to a concern of loosing certainty over the visual outcome.

We might be tempted to add proceduralism to the creative pot at the very last stage, when we have a clear established vision or close to that phase. And that IS a valid solution, however, that it is not necessarily the only option…

…and I'd like to show you how we did NOT explicitly separate, but co-created them simultaneously, allowing mutual influence and blend into a continuous mixture.

We tend to associate Art with something carefully crafted that in the process of the crafting solidifies into a particular singular configuration that does not encompass uncertainty or change. In fact, it discourages it as we define everything about it and nail that singular location in the creative space.

But what if you aim for a range of possible configurations. And you would like to include all these positions…

Well, in that case, you are not thinking about a particular point, you are thinking about a whole subset at the same time – you are basically defining an artistic ‘possibility space’.

And Art Direction becomes more like sculpting that possibility

space. You might want to exclude some part, maybe you are not a fan of a particular color combination, or on the contrary you might want to add something, like allowing balconies in your procedural house setup. And it is, as any creative process, very iterative.

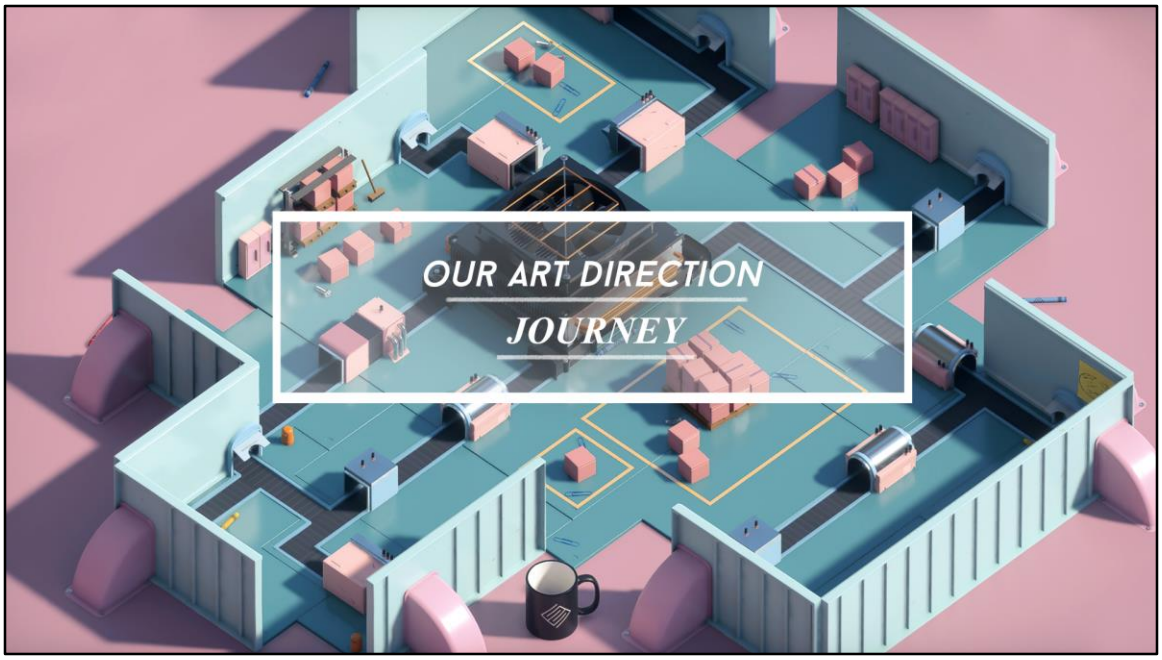
And is partially about finding parameters or rules that describe that possibility space.

And if we think about these points as a part of all the creative space, a space of anything you could possibly create.

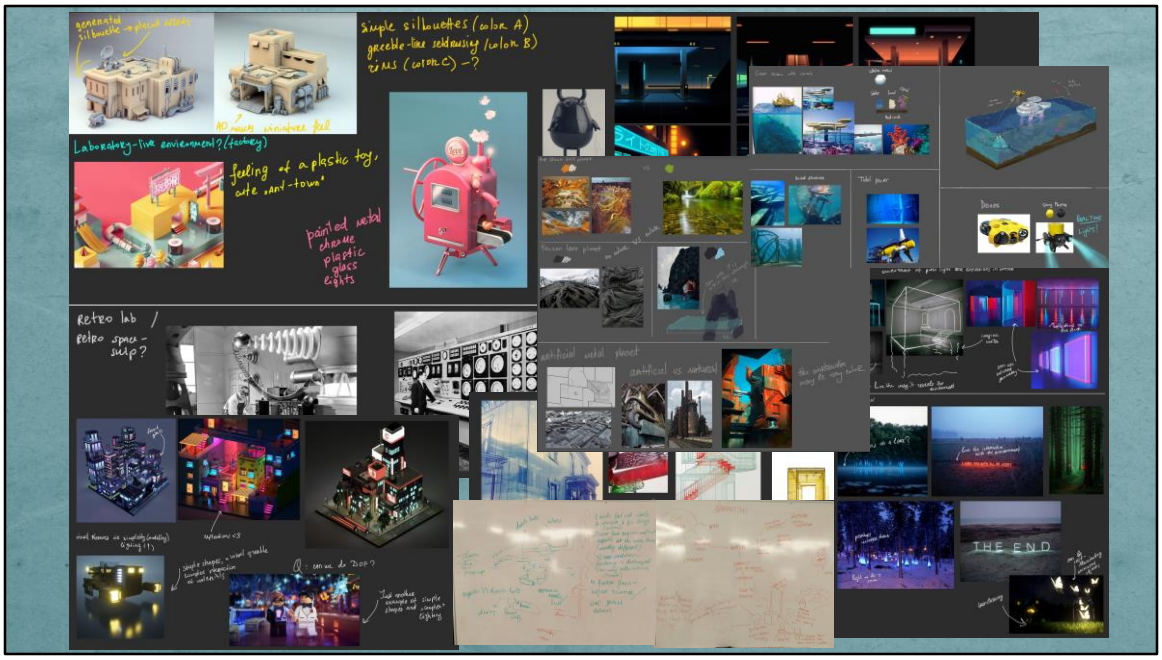
Then proceduralism is nothing more than taking a slice of that multi dimensional space, flattening axis that we do not care about, and trying to mimic and approach the resulting distribution as close as possible it via an algorithmic representation.

And it often sound simpler, when you talk about it, hey just take a slice, trivial, right. Well, in reality, exploring that creative space and shaping your slice, or even finding that slice…

…can often feel like traversing through a jungle with a butter knife.



To give an example from our art journey of the environment development, and warning – a chaos of images incoming

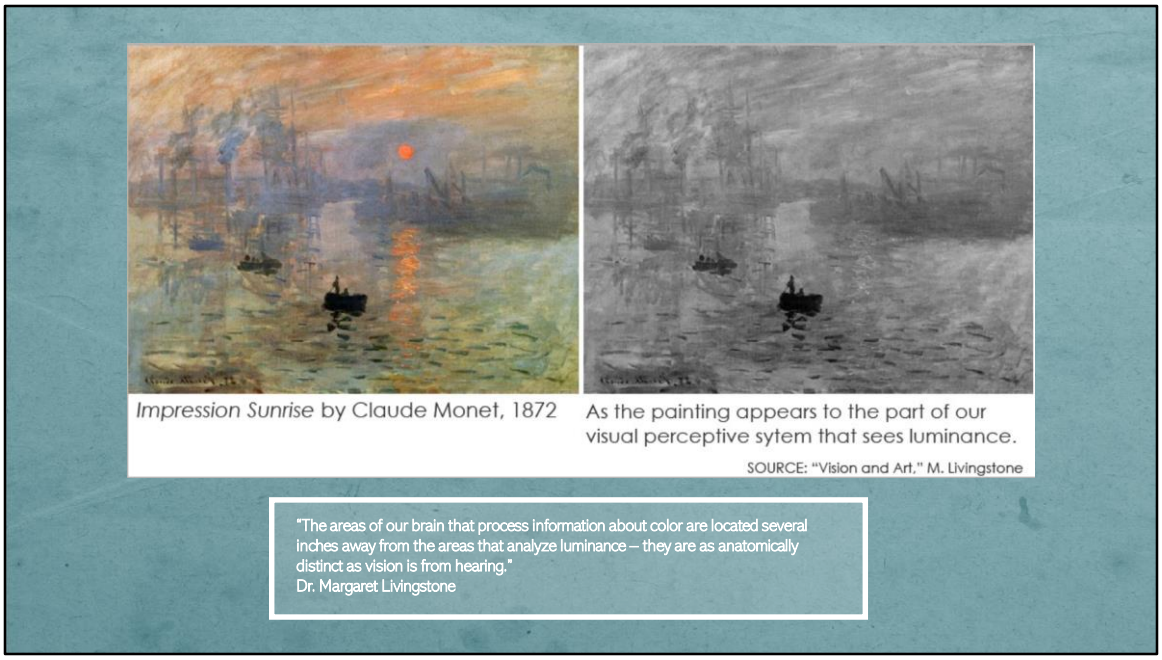


We started with a bunch of high level art style ideas. At that point, all we knew was that we wanted to showcase real-time ray tracing technology together with self-learning agents and procedural levels. And it was a fascinating dialogue of evolving tech along with art – asking ourselves: what kind of art style we can afford to showcase our technology, what size of level we would like that would benefit to our agents and so on. Then we proceeded to making first tangible concepts, at that point we settled on it being a factory.

And I personally am greatly inspired by a visual greeble of heavy machinery, an embroidery of tubes, cogs and metal beams. As a procedural artist, I was already thinking about all the systems I could create and was like “oh yes please, give it to me”. But as artist, we seek contrast.

So, what would happen if we combine it with something clean, sterile, almost candy-like? I’m a big fan of these pristine 3D graphic design renderers, and it seemed like a perfect fit for showcasing our real-time raytracing.

And we really appreciated the use of very clean color palette in these works, which combined nicely with an idea from a book called “The Biology of Seeing” by Margaret Livingstone.

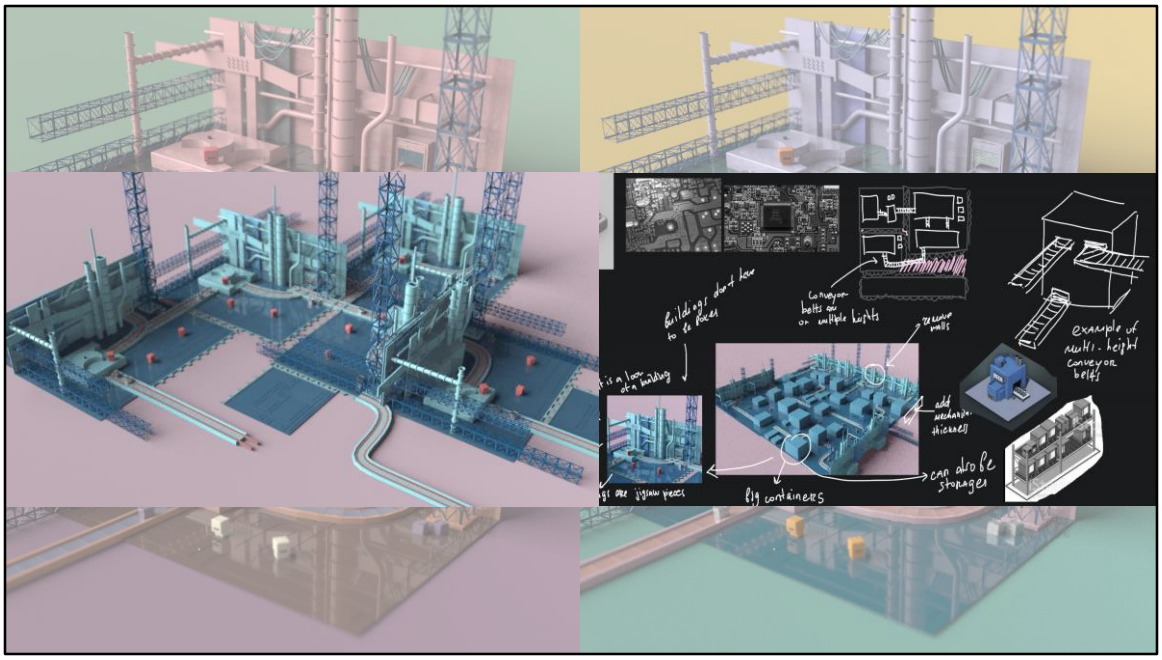


So, a bit of a biology lesson: according to Dr. Margaret Livingstone, “

, which means we process light and color information seperately. Our ’colorblind’ visual system was evolved earlier and is reposnsible for motion and depth detection. That is how most mammals see actually. And then we have another part that processes mostly only color.

Doctor Livingstone also included a very nice example of Monet’s painting, where if to remove color, no sun is longer visible. Yet, in the colored version, it draws out attention so much establishing an additional point of attentionn on top of the high constrast black boat, and I think, it very elegantly proves we can create 2 different layer of contrast and attention in a single image via lumuninance and color.

[5min]



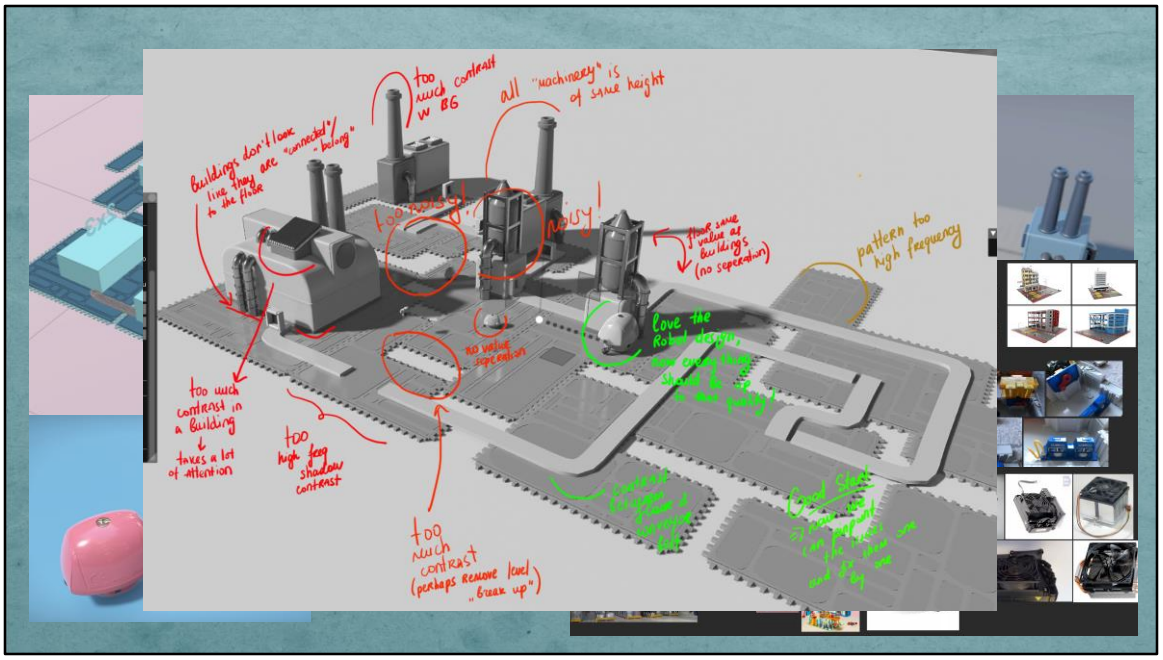
Here is the initial environment sketch I did in Houdini. So, Let me show you how we applied the luminance/color ideas.

In luminance pass, we separates low contrast/frequency areas of floors with higher contrast/ frequency details of the walls. At the same time, the walls had rough materials to focus on the complex interplay of light and shadow, and since floors were a much simpler geometry, it had a smooth material to allow dim reflections of the wall's complexity as well as moving agents.

And we thought to bring out important gameplay elements separation with purely color contrast, just like Monet's paintings.

We tried multiple color variation, even tried to reverse the luminance-color contrast in some cases.

However, the main problem became - how do you scale this small sketch scene up to a level? It has interesting ideas, but it doesn't include any information whatsoever about the level layout, machinery or how gameplay elements connect.



So, we were just throwing ideas and seeing whether they work. I made the first iteration of our layout system, took a couple of algorithms from the concept actually, like the pattern on the floors. We also thought it might be cool if not all the tiles are assembled, so you actually have some loose pieces. And we would have some pre-made factory buildings populating these islands and they would be connected via conveyor belts. Okay, that doesn't seem so bad.

But then when we put it together, something was seriously not working.

Around the same time my teammate Paul finished one of the robots, that really established the quality bar, in my opinion. Once I saw these cute little guys, I was like "Oh my god, environment should look just as cute and tangible as these robots".

So, back to the drawing board!!

First of all, we had big contrast issues all over the place. You eyes were wondering around, not knowing where to look, not having a visual anchor.

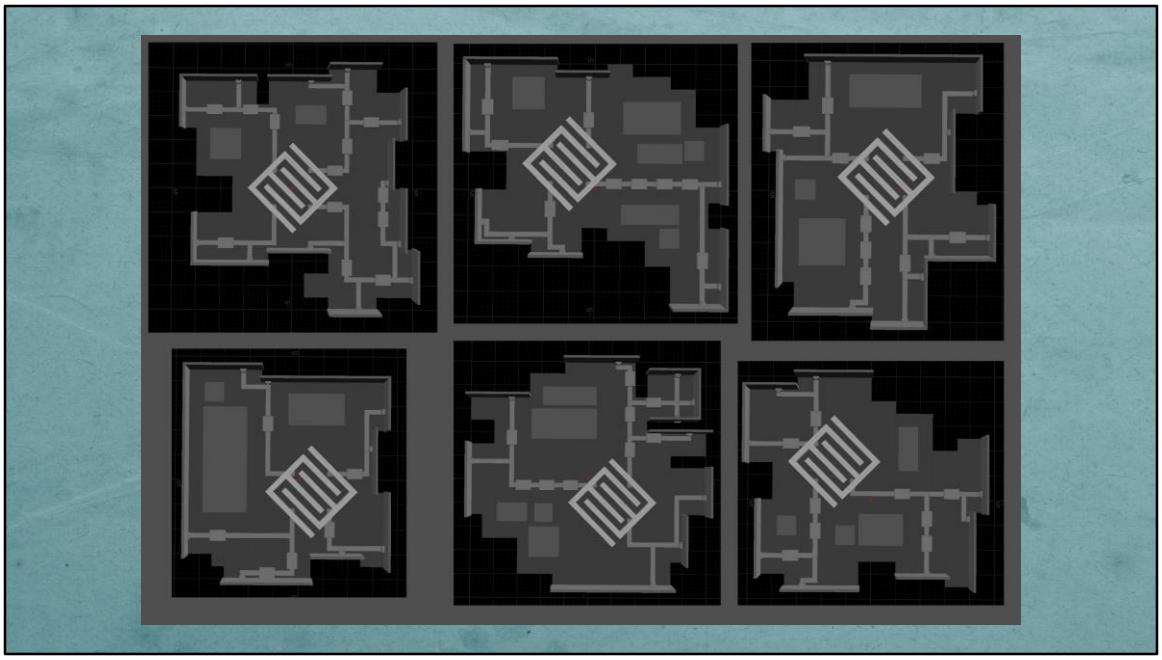
The cool idea of having our floor tiles to be in a process of assembly turned out to be

a contrast killer, especially when we started applying more color, as you got patches of floor in the middle of the level that created unintentional points of attention.

We also had a disharmony of high and low frequency details due to mismatch of scale and in fact not having an established scale at that particular moment.

Ken had a spot-on critique regarding that, and a solution to inspire more from actual physical toys and miniatures. He even brought a couple of toys in the office and a huge library of reference photos. And I think, it really helped to align our vision.

this an example of the analysis during the development. And those are just some of the issues, that looking back, I'm very happy we caught early on and stayed mindful of what is going on.



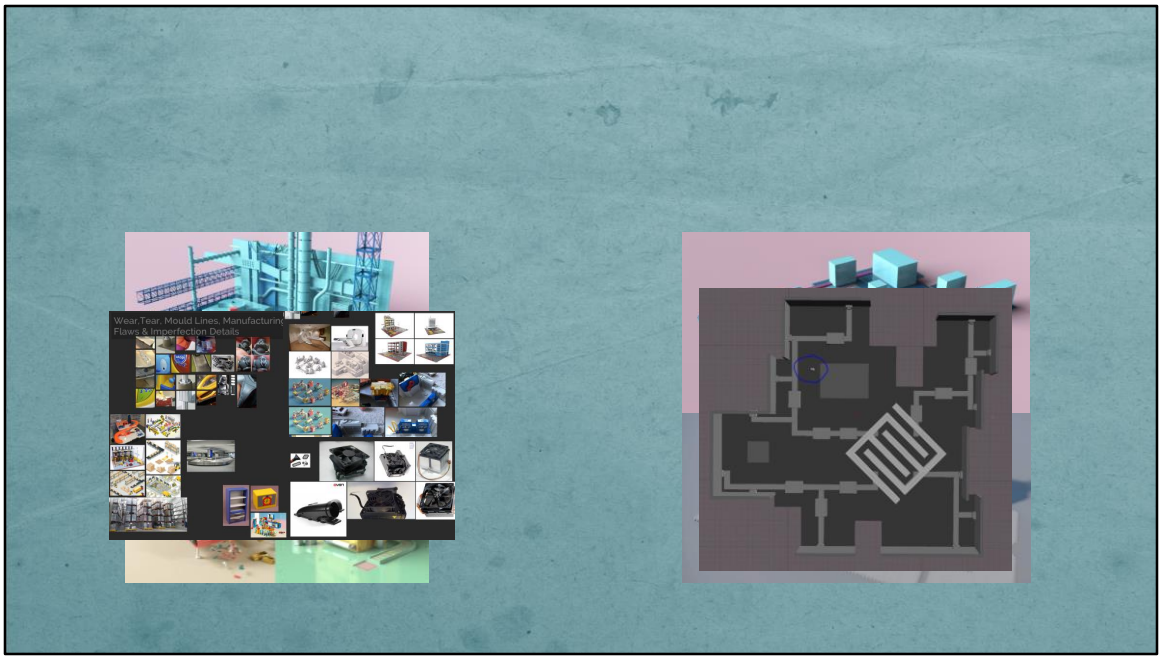
So, by trying to answer all those questions, the procedural system went through a complete restart. And the proposed changes were the following:

Let’s have a big hub in the middle, something to draw your attention towards, like a center of composition.

Instead of factory buildings, let’s have small process machinery, that wouldn’t fight for attention with the main hub.

And to make the level feel half-assembled, it will have missing walls and tiles.

In the places of no assets, we will put some logical set-dressing. At that point, I had no idea what that might even be. I was just treating it as a first sketch of a painting by starting with general high level composition. And once we had that foundation, things started to come together.

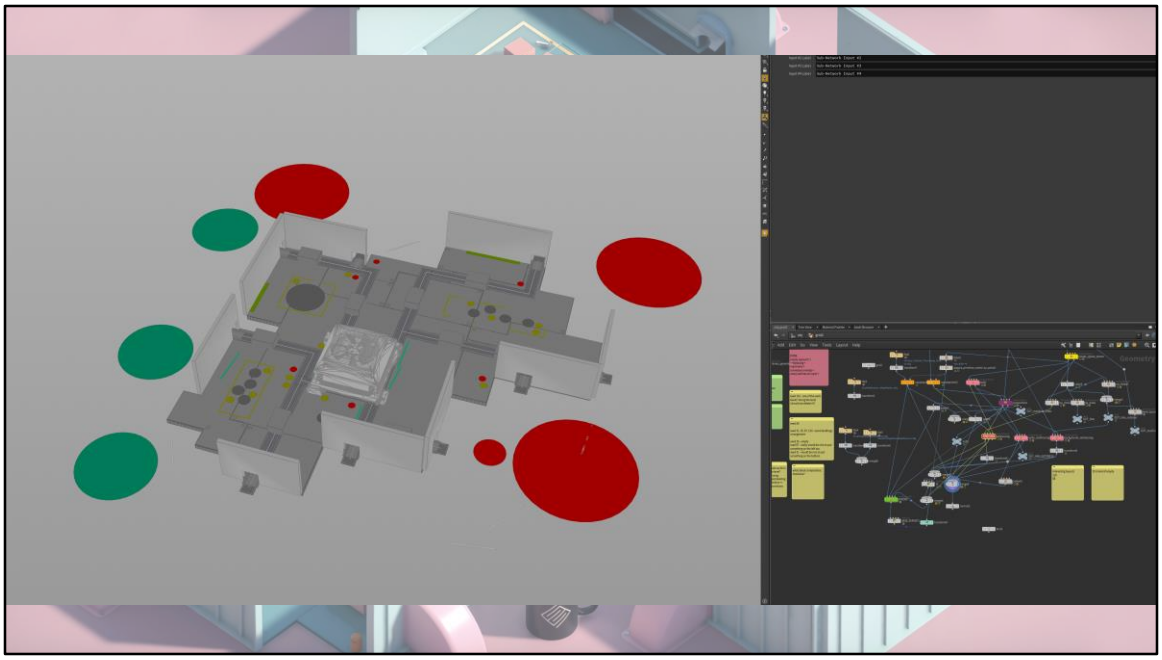


So, the way I see the process that was happening is a feedback loop between Art Direction and Procedural System.

First we had an idea, we tried it in Houdini, we did some feedback, then we tried something else, and that spawned new ideas from Art Direction side and so on. So, it is actually not just a loop…

…but more of a spiral that eventually converges somewhere in the middle.

[+1min]

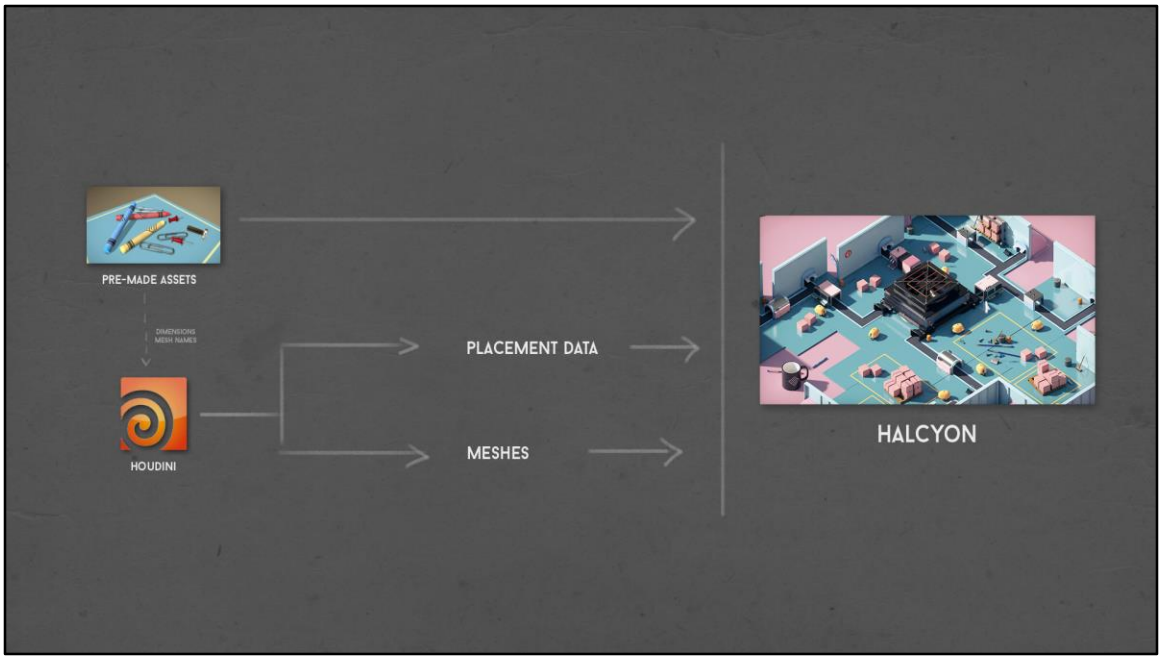


And through this iteration another important point was being resolved - what do you proceduralize and what stays static or manually constructed? And it might sound tempting to just proceduralized everything – but you ought to analyze what makes most sense, especially under time constraints. And for us, the layout was an obvious most important candidate.

So, we decided that all the smaller assets are to be created manually, and procedural system would be there to place them in a logical manner. We even went as far as having pre-made arrangements of those assets, which we called hero piles, and we placed them together with procedurally assembled ones.

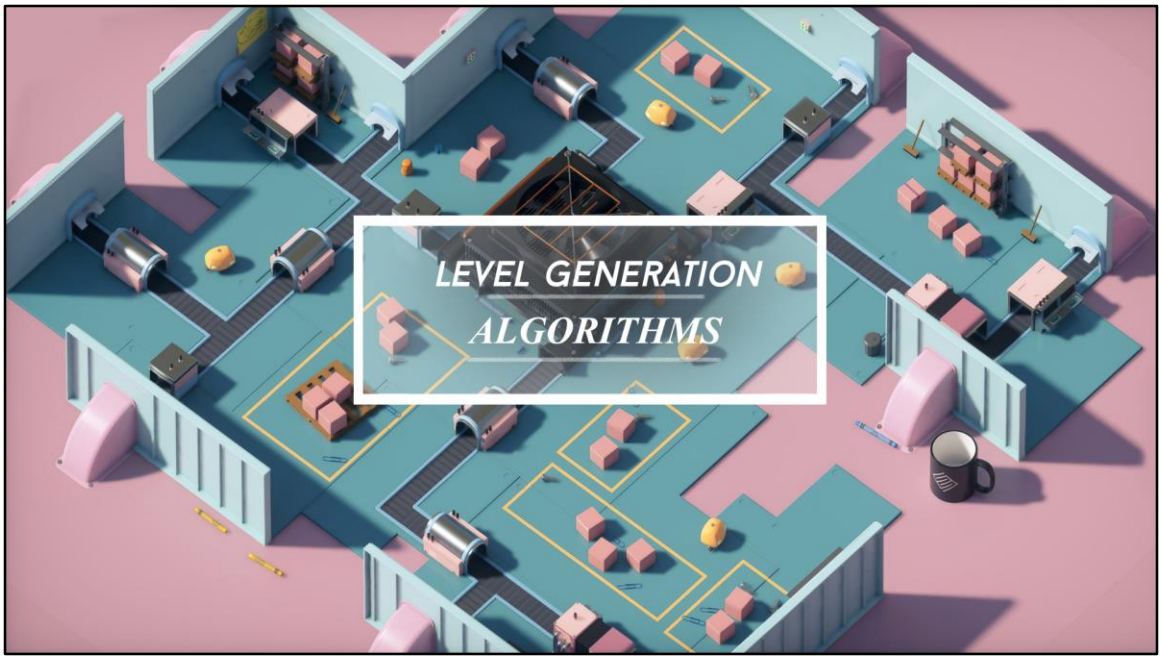
This is how it looked inside of Houdini, where circles and those

colored lines are abstract representations of the set dressing . And things like floors, conveyor belts, walls were generated and exported as unique meshes. And you might ask “hey why not make conveyors modules and assemble them as well”, well, again, it comes down to – what is faster? Is it faster to design a whole modular system and test it to make sure it doesn’t break or just generate a mesh in Houdini? So we went with the latter.



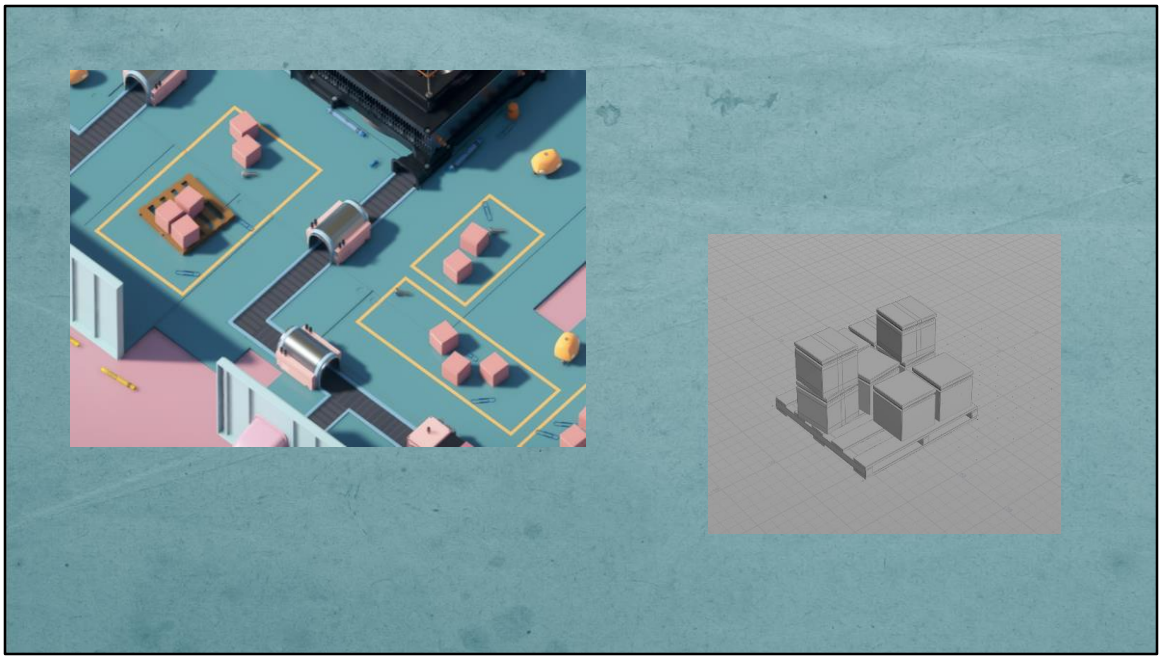
So that is how our procedural pipeline looked like in the end.

We had a library of pre-made assets, we also were generating meshes from Houdini. Then we also export placement data as json, which upon loading in Halcyon fetched all the specified assets and meshes and placed them.



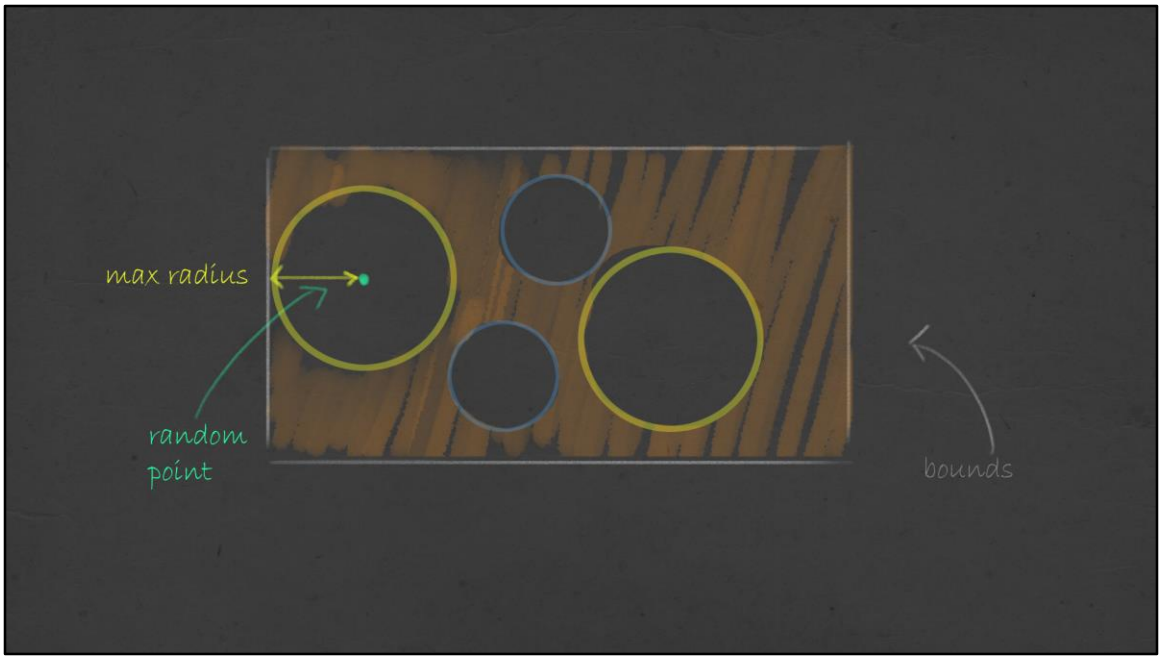
So, now let’s talk about what was happening inside of Houdini – or more precisely, the algorithms themselves.

[8min 30sec]



So I'll start a bit backwards to motivate why in the end we did what we did.
Once we had the level generation algorithms, we though about object set dressing and how to populate level without them intersecting.

So, what we did is we separated all the objects we had into categories based on size, and since we knew we wanted random rotations, what shape does describe all possible rotations of an arbitrary shape – circle. So if we solve circle packing problem, we solve the object packing. For those of you who are not familiar with circle packing, here is a quick overview.



There are many ways to do it, but here is one we initially used. We start with the bounds.

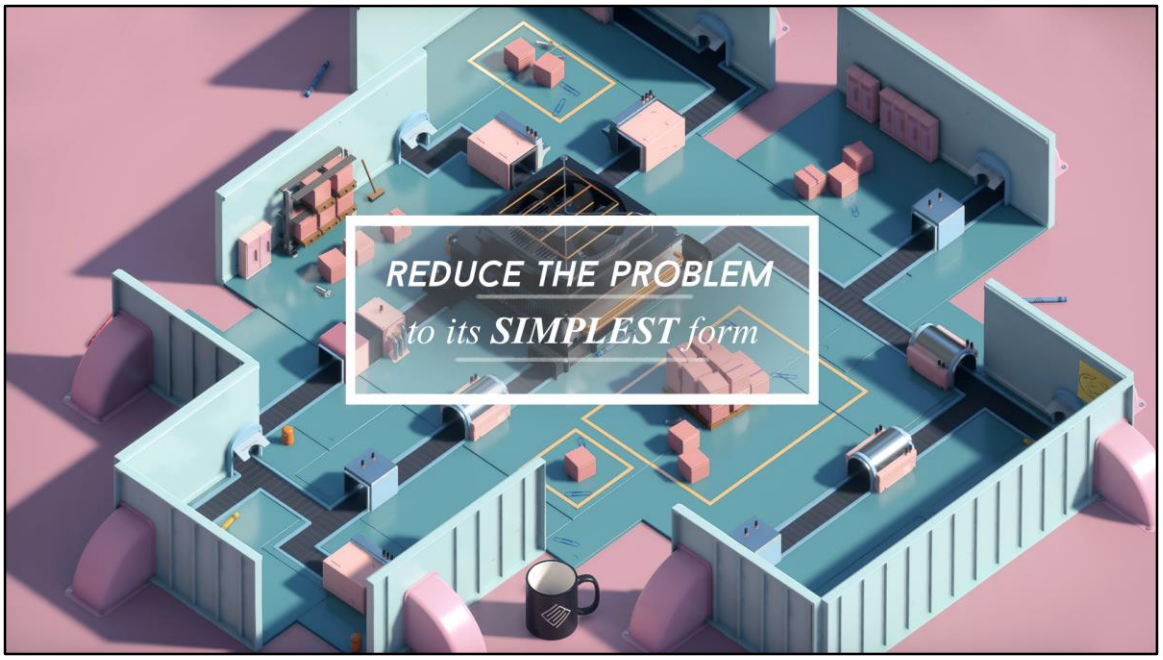
We pick a random point.

And we place our circle and check whether it intersect with the bounds.

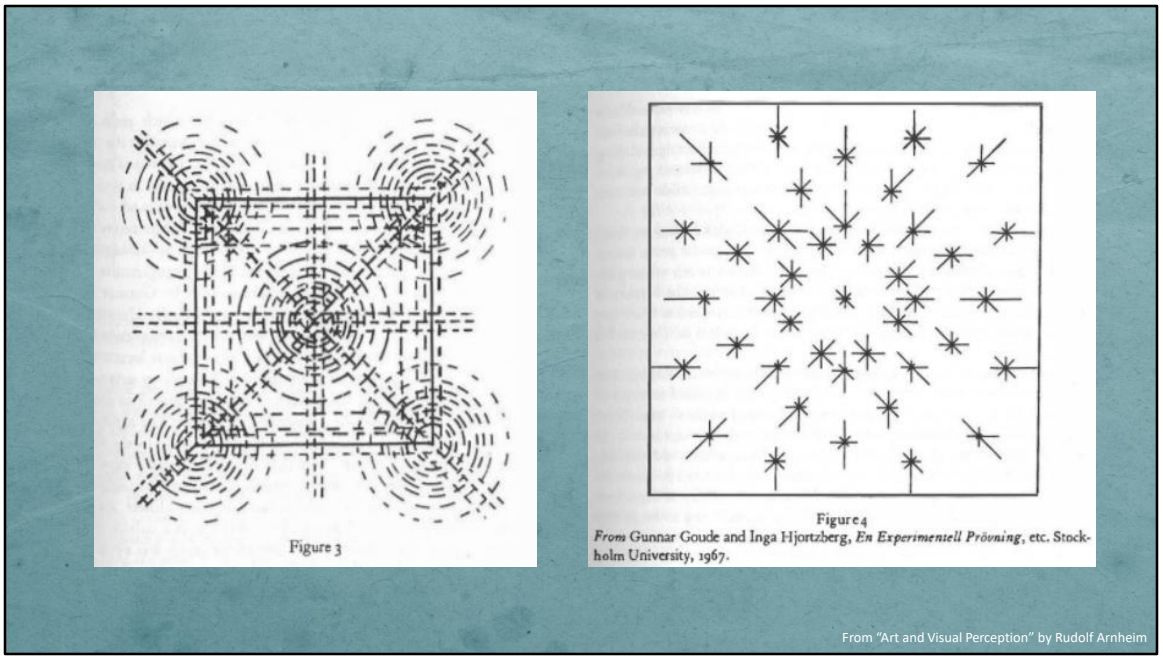
Then we subtract the circle area from future possible spawn area.

And if we so happen to create a new circle that overlaps with the existing ones, we can simply check by distance from the centers of those circles.

Since we have multiple categories of size of our assets, we start with placing the bigger sized circles, and the we proceed to smaller, just to ensure that bigger circles have a chance to be placed.



So, instead of solving a harder problem, we find the simplest representation. Get down to the core of what we are actually trying to solve and reduce the it to its simplest form. And once you know the simplest form, you can start solving much more complex problems, such as, instead of random placement – how do we create a visually pleasing layout and assets arrangement? And I want to add special thanks to Kim Goossens for the inspiration regarding the topic.

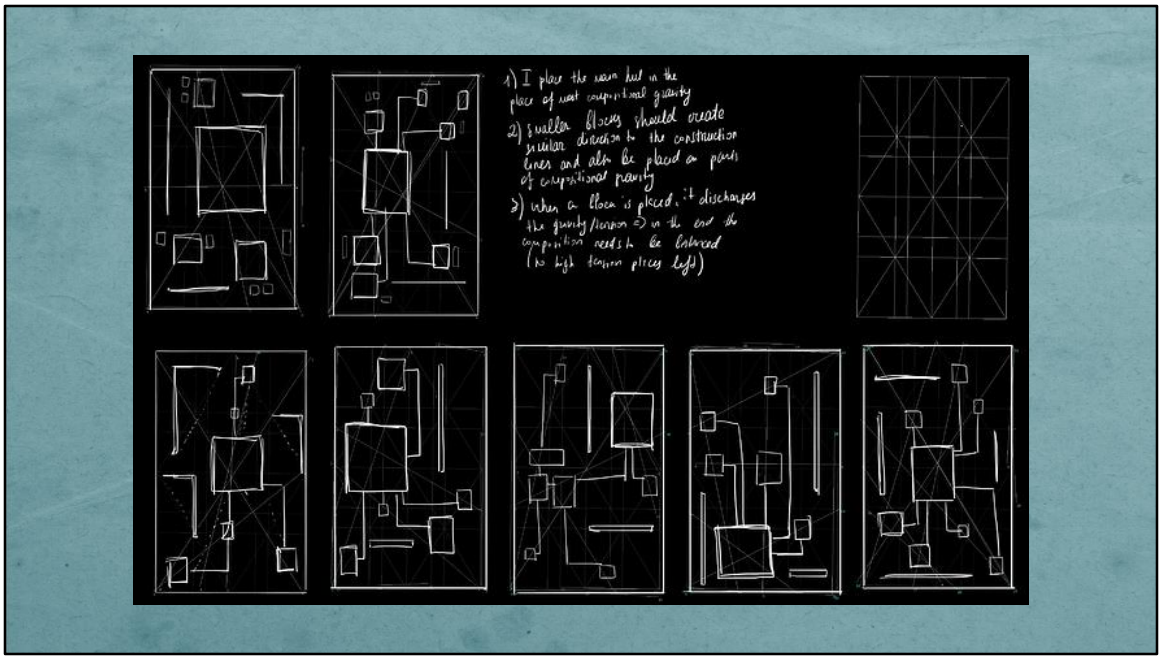


The key idea came from the book called "Art and Visual Perception" by Rudolf Arnheim, where he showed the following image on the left

Which was associated with a placement of a single black disk on the canvas and tracking how balanced or non-balanced its location felt. Via informal exploration he concluded that there were invisible points of influence, which he called structural skeleton.

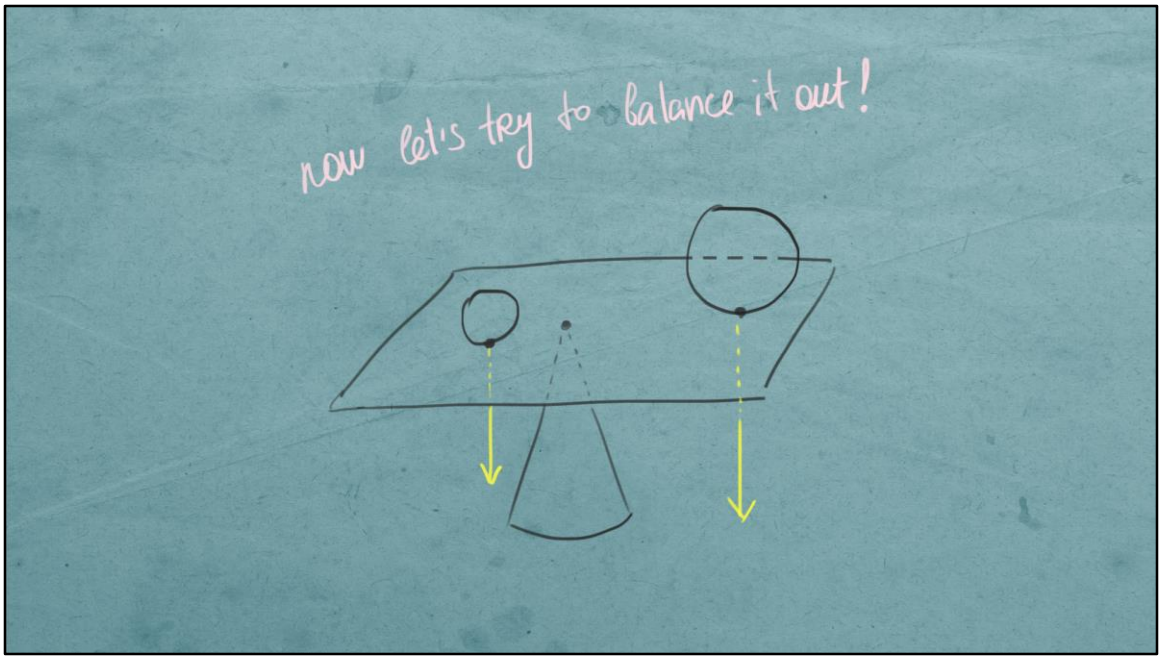
And on the right, the disk was moved to various locations, and subjects were asked to indicate whether it exhibited a tendency to strive in any direction, and if so, how strong this tendency was. And we can clearly see that just mere location of the disk

was creating a perceived sensation of movement and the further away it was, the stronger the sensation presented itself.



Okay, that sounded pretty good, so I took the compositional lines, and started sketching level layouts and how I would manually arrange them on a canvas and recording what exactly is happening in my brain, what are the internal guidelines I'm following.

And I noticed a very curious regularity – I would create misbalance at first based on some intersecting compositional lines, and then I would try to balance it out. And I would balance it out relative to the center of the canvas.

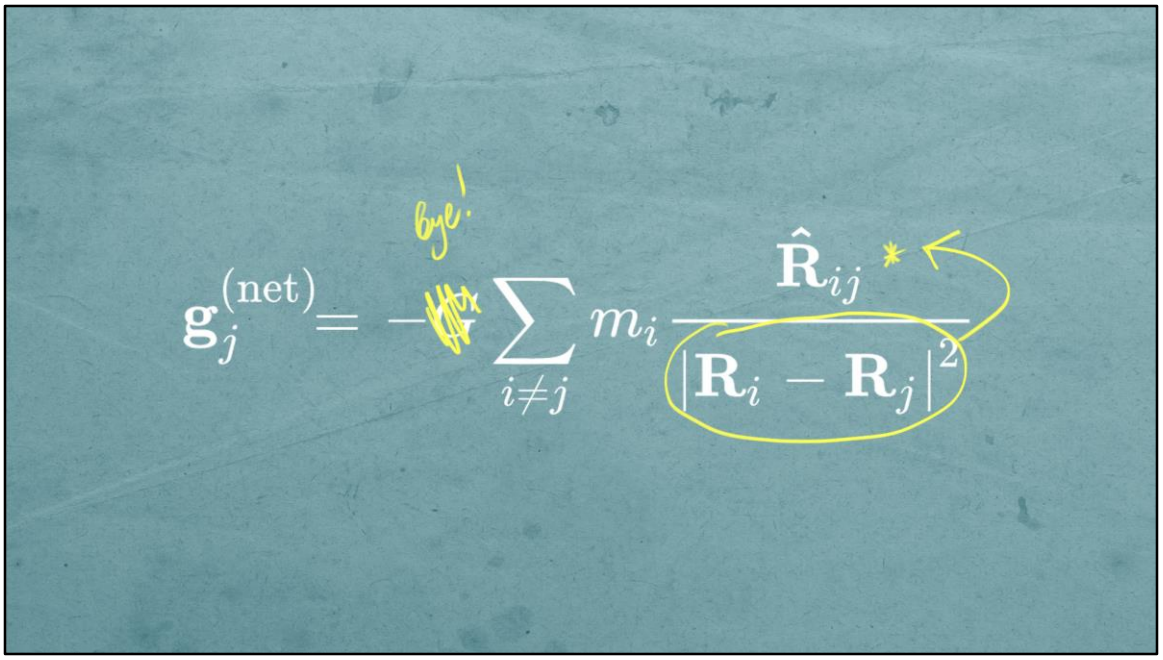


So, the following visualization came to my mind. Imagine you have a plane surface of a size your canvas that is balancing on a tip of a triangle. And it is perfectly stable until you put something on it.

And in order to counter that new mass you just placed on this surface and prevent it from falling, you need to put something else in a very specific location with a certain mass to try to bring this imaginary scales back into the equilibrium.

And a disclaimer right away - composition is a big topic, where things like shape, value, direction, size, rhythm – all of them contribute to the perceived mass of the object. So instead of trying to encapsulate all of them at once, I'm going to be dealing

purely with size as a visual parameter of the mass function of an object.
So, how do we actually calculate the influence of an object on the equilibrium of our composition? Well…

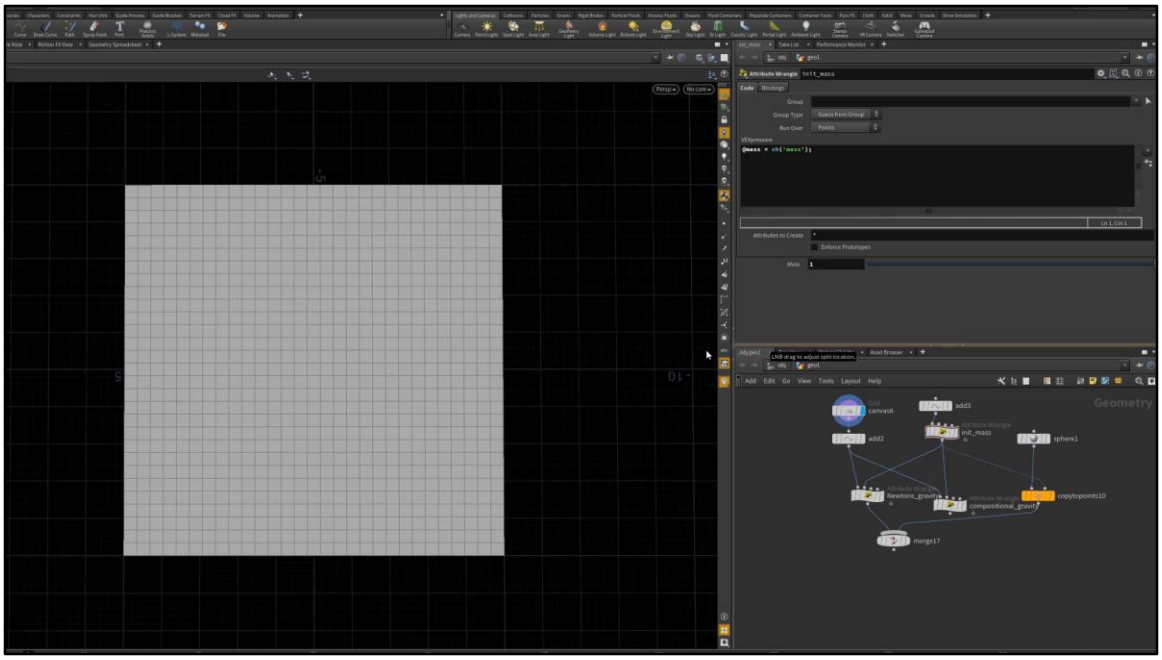


..why not start with Newton's law of universal gravitation? Since we are talking about compositional gravity. And this is a gravitational field version of Newton's law that simply read…

The gravitational force at a point j is a sum of all the directions to other points excluding self multiplied by their mass and divided by the distance between them. Trivial ☺ But as you probably remember from school Physics classes, the closer the object is, the stronger its gravitational influence, whereas in our case we actually do not care as much for objects that are closer to our canvas center, because they destabilize our composition much less than those that are at the farthest corners.
So, we have to kind of modify this formula a bit.

First of all, who cares for constants. So bye, we do not need you, you just clutter our viewport.

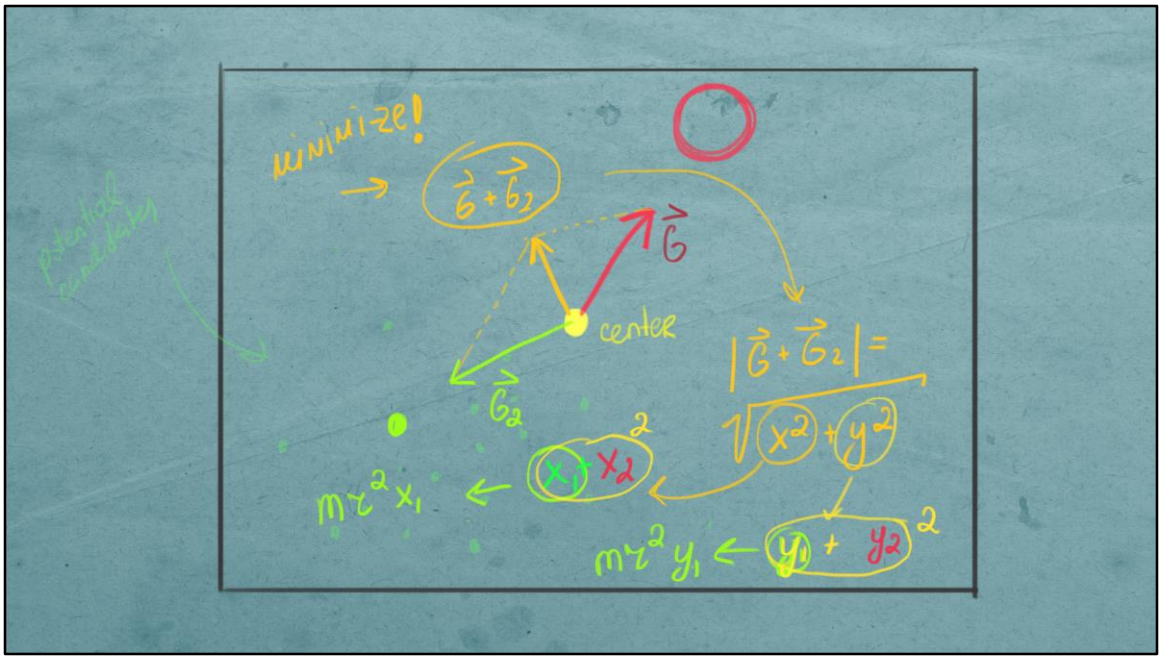
With regards to accounting for compositional influence, it is actually inverse of the gravitational influence in Newton's formula. So we can just take that bottom part and move it up, because as I mentioned the further the object is from the center the more critical it is to balance it out. And we are always going to be calculating only one point j, which is our canvas center. (though potentially you can calculate also the border corners as shown in Arnheim's illustration or some compositional lines intersections, that actually would be pretty sweet)



So, this is a demonstration of both Newton's formula and our modified Compositional Gravitation formula, just to give you some visual reference of how those forces affect our canvas. And with Newton's formula, you can clearly see that the closer points around the object get attracted to it, which you can observe by their vector magnitude and direction that is pointing towards the sphere. Whereas, our Compositional Gravitation formula does exactly the opposite – the closer the point is, less affected it is by our sphere. And of course, we can modify the strength, influence distance, and the spread of this force.

And it doesn't have to be only one object affecting our canvas. We can have as many as we want, and as you can see with addition of a new object, they start to kind of fight for the

influence on the canvas points. And the bigger the mass of the second object is, the more it pulls everything towards itself.



So, let's say, we have our initial dis-balancing object, which is indicated as red circle here. And our center point is being pulled towards that object by a force, which I named G1. So, how do we actually figure out where to place a counter balance in this case?

Well, first of all, we need candidates. And I sample them randomly based on normal distribution and the direction that is determined by dot product relative to G1, and that just gives me an opposite side of the canvas relative to vector G1.

So, let's say we are currently checking just this particular green point out of all the candidates.

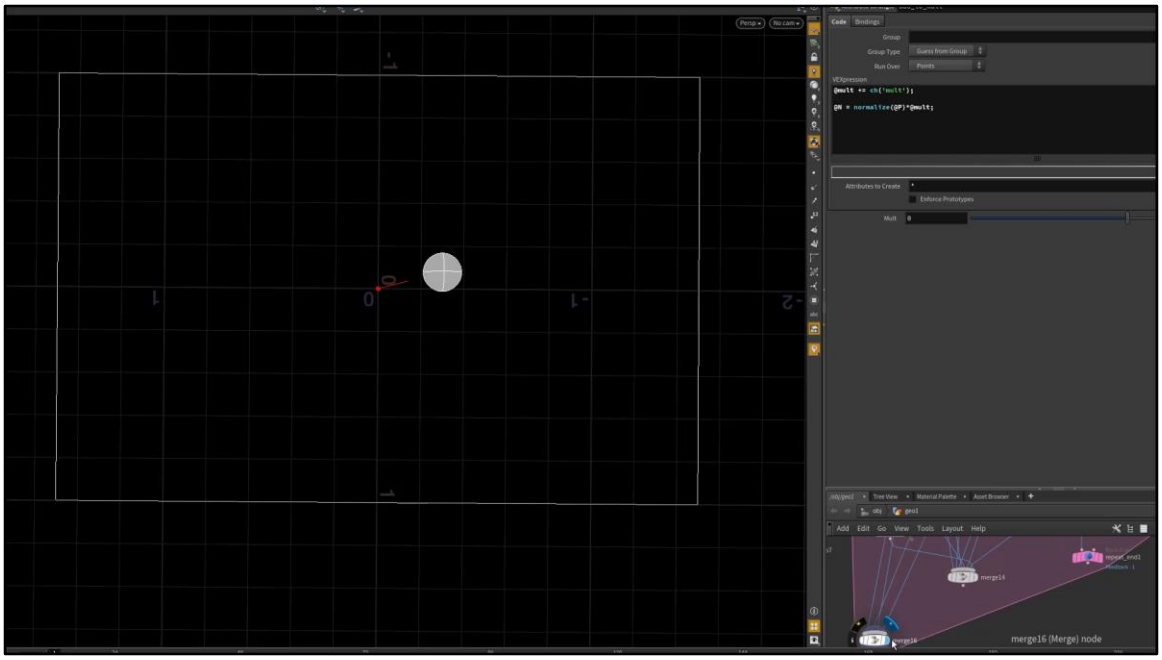
And the influence of green point on the center of our canvas we will call G2.

So, the challenge becomes figuring our what best G2 can be, in order to minimize the sum of G1 and G2. Because we want this green point to balance out the canvas so much, that the center goes back to equilibrium, which mean 0 vector, as all forces cancel each other out.

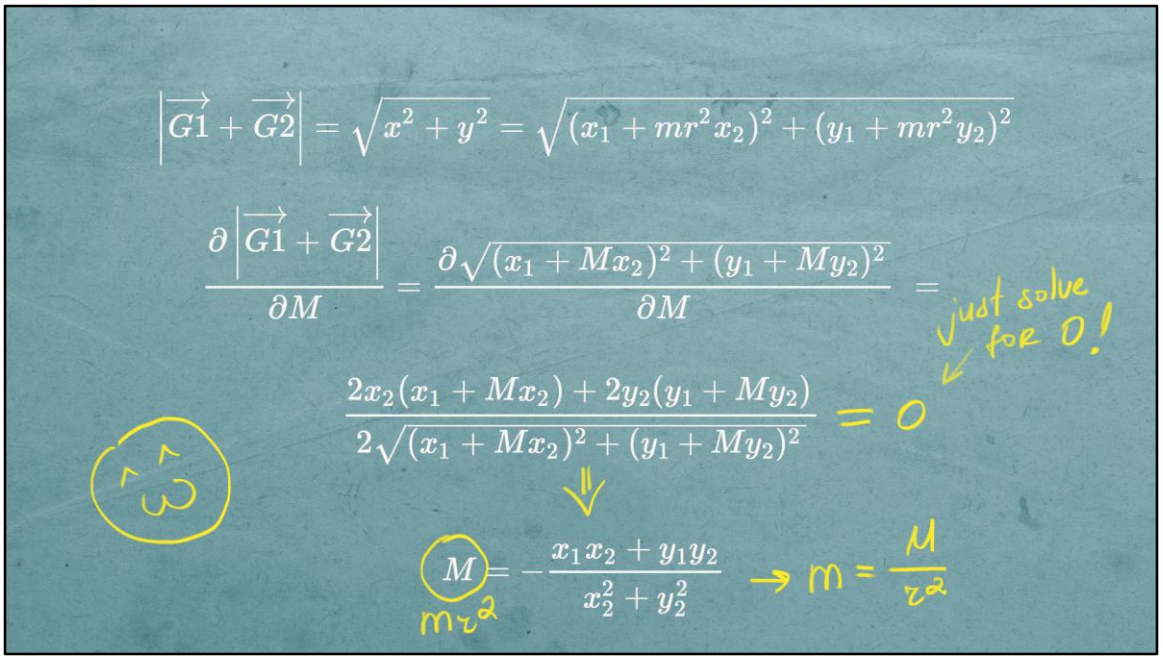
And we do not actually care about the direction of the sum vector, we just care that its magnitude should be zero. And if you remember we can calculate magnitude with Euclidean distance, which is a square root of sum of x and y squared in 2 dimensions.

And those X and Y can be easily calculated, because our sum vector's X and Y are nothing more than simply adding Xs of G1 and G2, and Ys of G1 and G2, which is a basic rule of vector addition.

And in order to calculate X and Y of G2, since we do not know yet, what G2 is. We can just use our Compositional Gravity formula, which states that vector G2 is mass of the object multiplied by distance to the center of the canvas squared and multiplied by its direction.
So, I hope you are still with me, and here is a more visual representation of what we trying to do.



So, we have our center of composition, and the red line represent the pull vector towards the sphere. The green point also has an influence on the center, which the green line. And the sum of green and red lines is indicated with blue line. And all we are trying to achieve, it to make the blue line as short as possible. And as you can see we kind of have a huge possibility space. We can make green line shorter or longer, but how do we programmatically find the best green line that minimizes the blue line length?

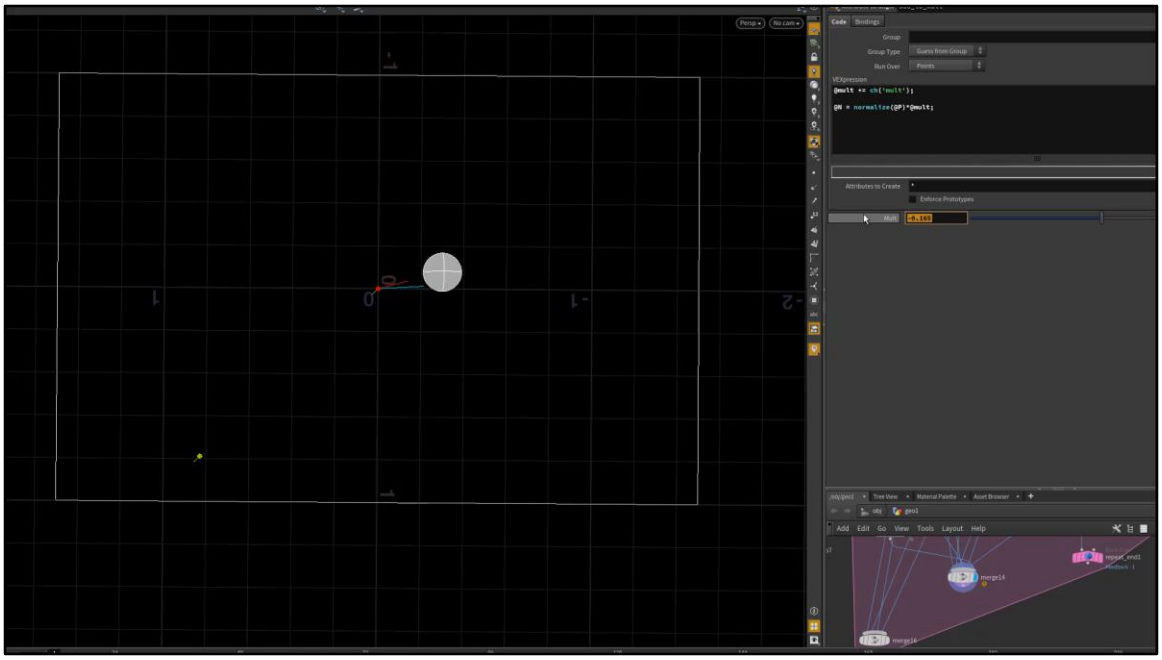


So, here is just a cleaned up version of the formula for the blue line's magnitutde. And as I mentioned before, we want it as small as possible.

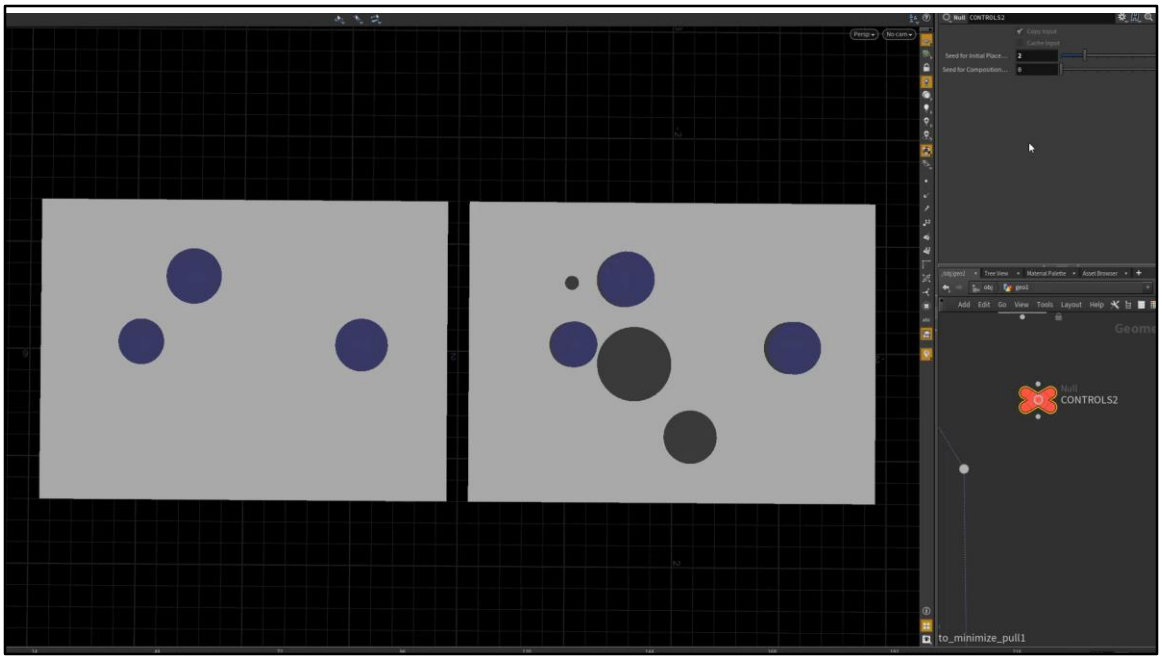
This formula might seem like a lot of variables, but remember, all of those are actually known to us. All except the mass of the green point, which is lower m here. And to make calculations less cluttered, lets just assume that mr2 is capital M, and that is what we are trying to find.

And since we trying to minimize the formula, we just take the derivate, and we get this beautiful boy on the bottom. It might seem somewhat complex, but once you actually solve for 0…

…it becomes nice and cuddly. And by the way, we are solving for 0, because we want to find the minimum of the function, which means the slope is going to be 0, and this function always has only minimum, so we are very lucky.
And once we calculated our capital M, it becomes a piece of cake to get our mass value out of it. Or you can easily reverse the last step and get distance based on mass, if you want to figure out how far you should place given mass for most equilibrium. However, on practice it can cause a lot of self-intersection with other objects, so I prefer to randomly sample position and calculate the mass, rather than the other way around.
And potentially you do not even need to go through this math – you can just randomly sample masses and hope you find a good one – as we do not really need our algorithm to be too perfect, otherwise it will balance out our composition too quickly without adding enough elements. And that is exactly why I'm not finding both mass and distance and maybe even direction this way, as I want there to be imperfection and chance of randomness.

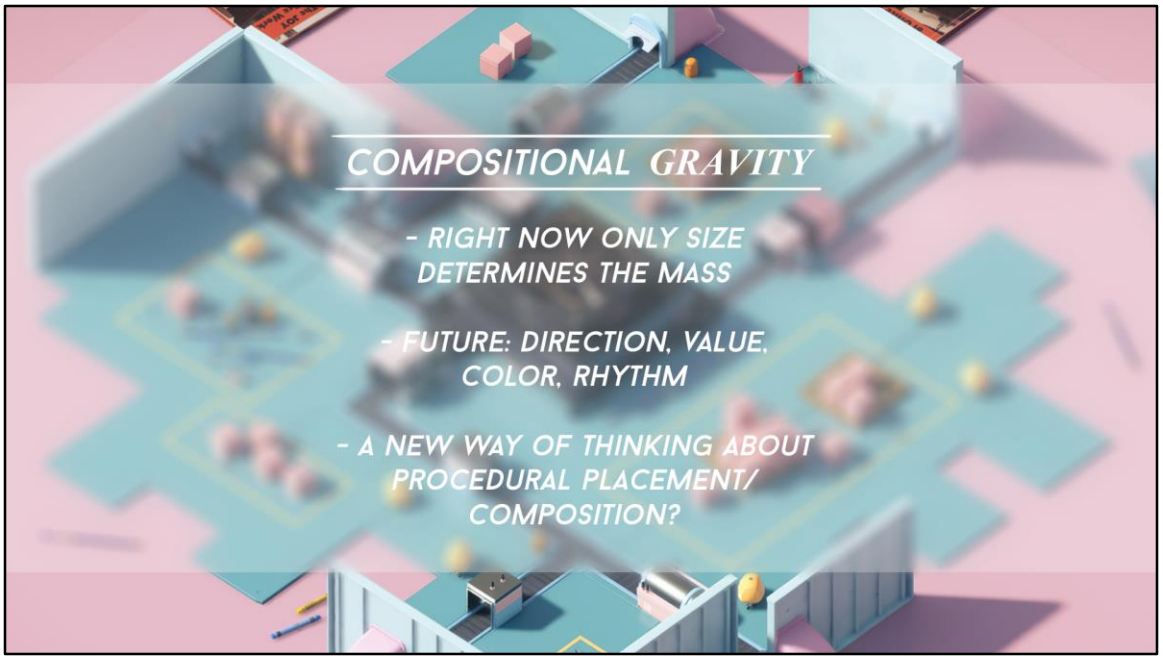


So, here is just a visualization of the algorithm finding the best G2 vector for the green point. And as you can see we are sampling multiple green points, and they have various success, some of them have longer blue lines, some of them actually almost perfect. So, we calculate, let's say 30 of them, and then we just measure which one did best. And that is the one we keep.



Once we associate mass with size, we have something like this, where blue circles are initially placed based on compositional lines intersection to create some disbalance, and the grey ones are created by the algorithm that tries to achieve compositional gravity equilibrium. And since there is a randomness factor involved, you can also just change the seed, which will alter the way the algorithm arranges new masses for you.

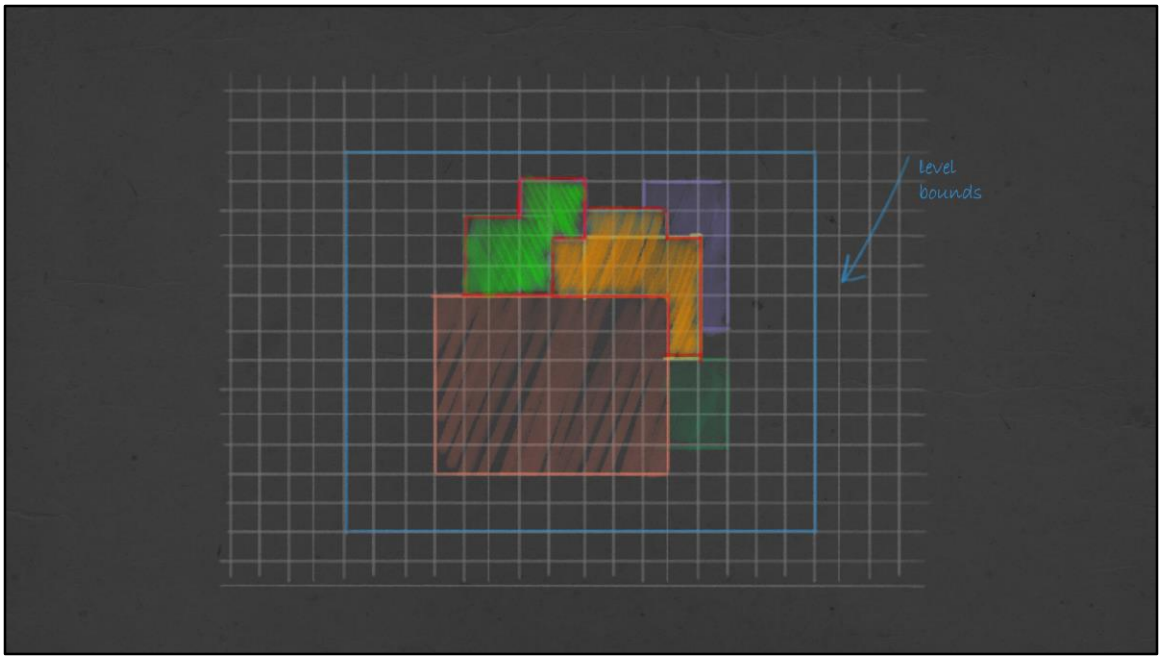
And circles of course, are just for debugging, but that is actually what we use for placement of assets in Pica Pica as well, where circle just represents the bounding area for a given asset or even cluster of assets, as you can create both micro and macro compositions using this technique.



And this was just a very quick prototype of the idea, but in the future it might be super interesting to include things like direction, value and color into mass calculation. Because right now, as I mentioned, the mass is calculated purely from the size of the object.

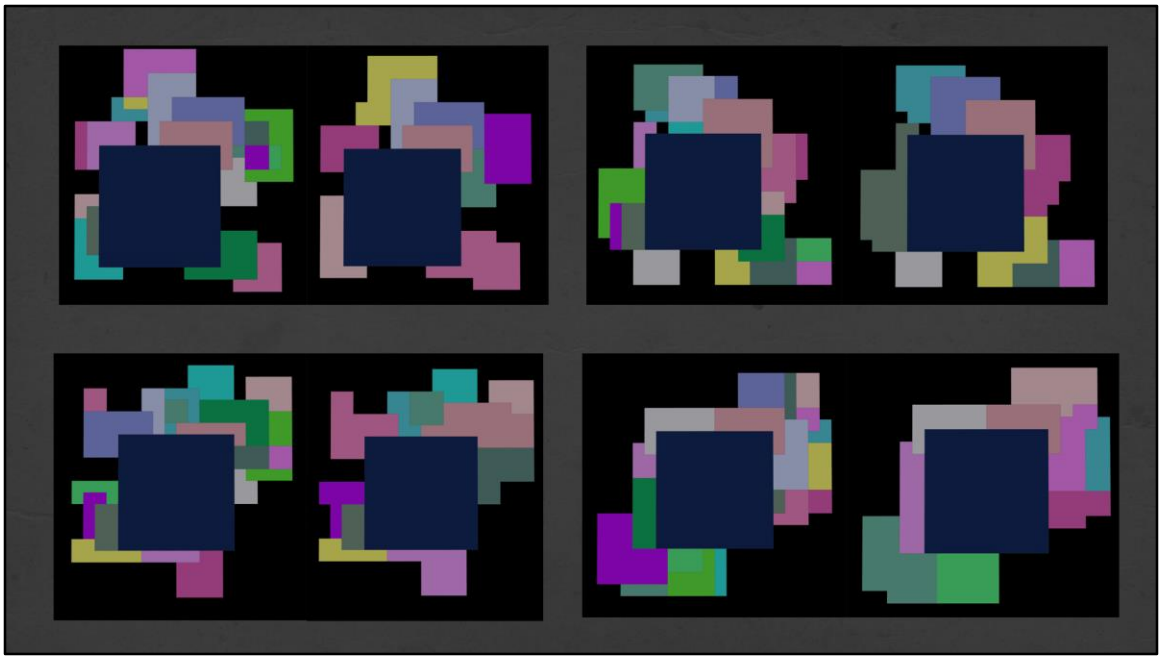
But I think, it might be a **new and exciting way we think about procedural placement** and scattering.

Moreover, we can then use the same principle of compositional gravity in other algorithms, such as layout generation.



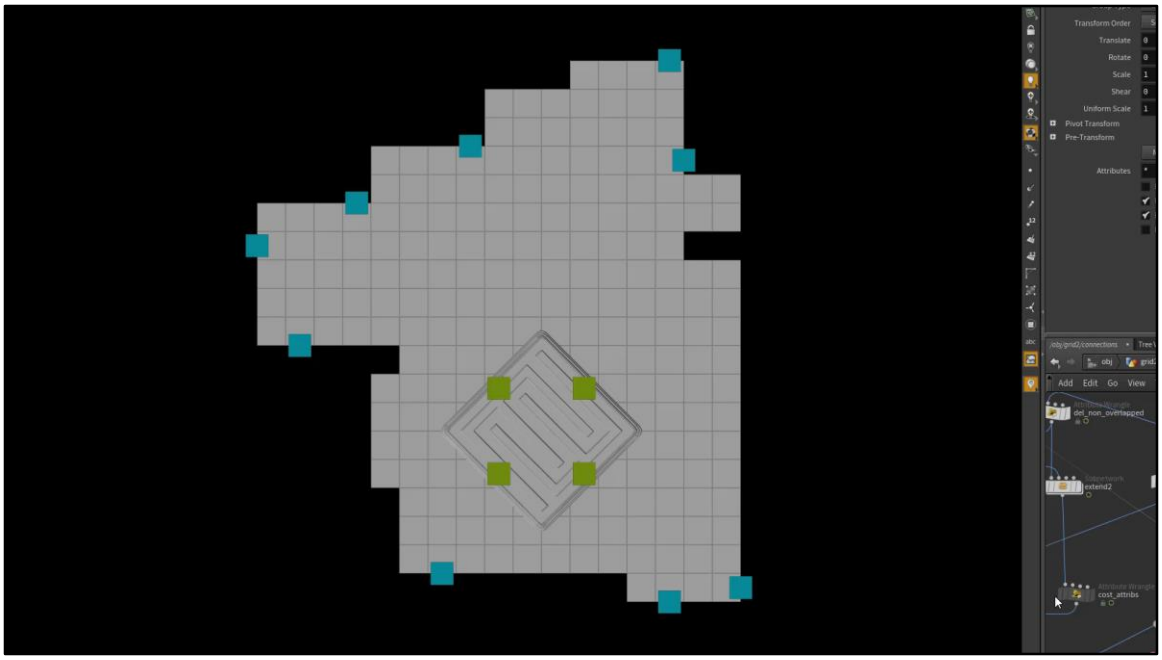
So, the way we generate the level is we start with a grid system. We have user-defined level bounds, and the area inside we call canvas.
We have an initial spawn area, which is 1/3 of our total canvas. This decision is hard coded with accordance to composition rules, as we do not want our main hub to be out of the primary attention area. And then we just select a random start point within that area.
And we place an arbitrary sized tile, which will hold the main hub building, the CPU.
And the exterior of that tile becomes the new allowed spawn points, from where we now can apply the compositional gravity algorithm. For every point we find best sized tile to approach equilibrium, and then we just pick the point that did best.

Once we place the new tile, we can get overlaps.
But those are easy to check for, since we can query our grid system and ask whether that particular grid position is already occupied.
Then we again take the exterior of existing tiles as new possible spawn points.
And we repeat this process X amount of times. However, we have a lot of small and partial tiles. And we want to give an impression of a more sophisticated tile packing.
So we check every tile that is too small and we detect how many points does it share with neighbor tiles.
And then we merge them with the neighbors that have most points in common.

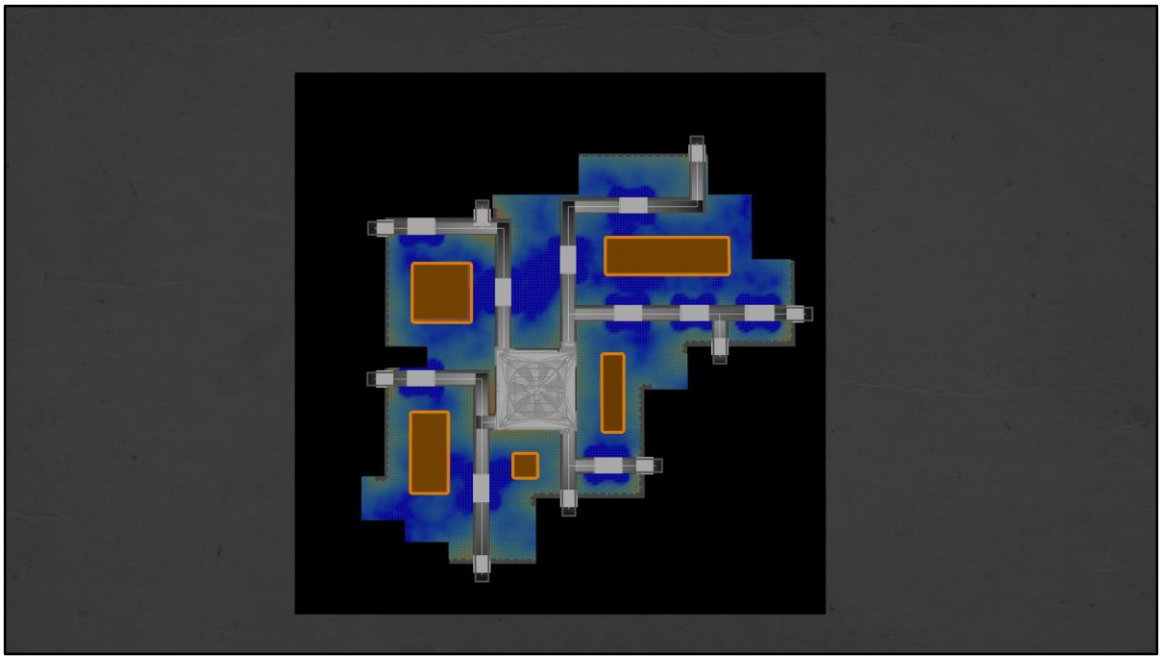


So, here is an example, where on the left you see original tile placement, and on the right that is how it looks after the merging process. And it’s quite funny, because I was asked how did I manage to pack such arbitrary shapes together. And the truth is…

I didn’t. I broke that visual complexity down into simpler steps. And together they give an impression of something much more sophisticated going on. Often, you do not need to solve complexity if you can layer simple rules and their interaction will create a much more complex appearance. With a similar approach we create the conveyor belts layout.



We simplify it down to a basic path finding problem. Initially, it was tricky to solve all paths at once. But then we broke it down into 2 steps: create the main lines first and then connect the rest of the input points to them. So, divide and conquer.



So, once we have that data, we can start thinking about how we populate it with assets. And we have 2 different systems for that: bigger piles and also smaller more chaotic scattered set-dressing.

For the first one, we detect all the available area by again querying the grid system.

We shrink the resulting enclosed shapes with poly-expand node to get rid of areas that are too small to support our bigger piles.

However, as a result of that operation we have non-rectangle areas, which wouldn't work with our pile creation.

So, what we do is that we take each of those shapes…

We check for points that are inside the bounding box.

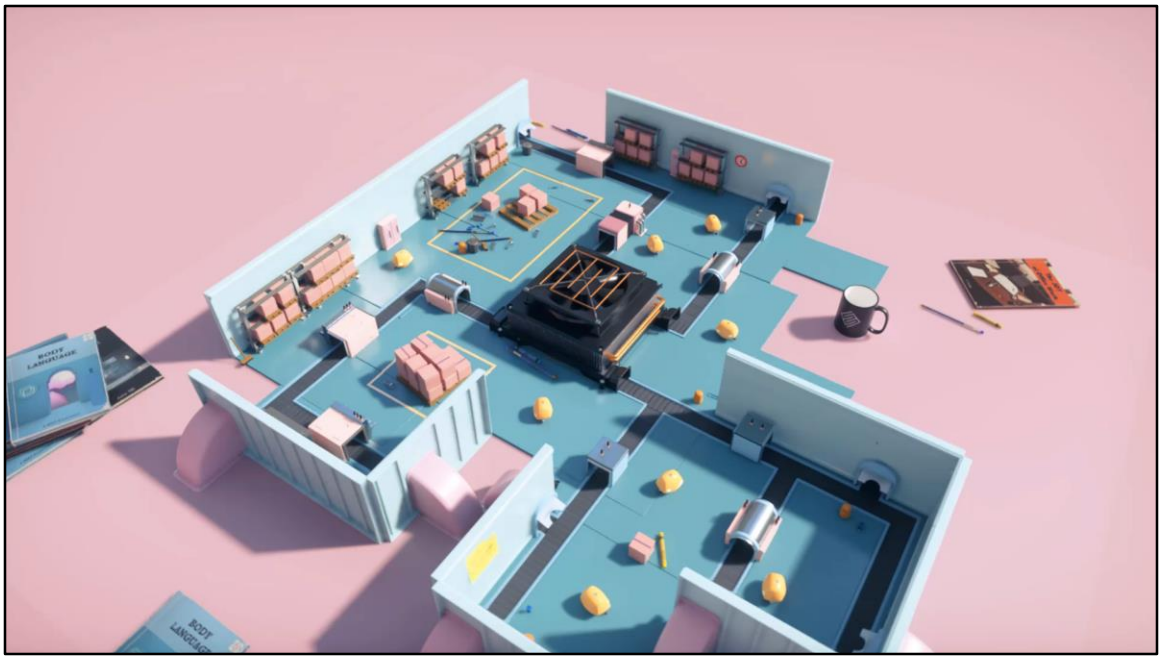
And for those points we shoot rays for the edges that it holds….

And then we pick the direction with the minimum distance, and that is our cut. And according to my colleague, it is apparently similar to BSP or Binary Space Partitioning algorithm, which I didn't know about at a time.

But once we have the cuts, we can perform the shrinking again. And now, again, we can use our compositional gravity principle to approximate which of those possible locations we actually want. There are many ways you can do it, but for time sake, we simply check X amount of random combinations of those and then choose the one that performs best according to compositional gravity evaluation.

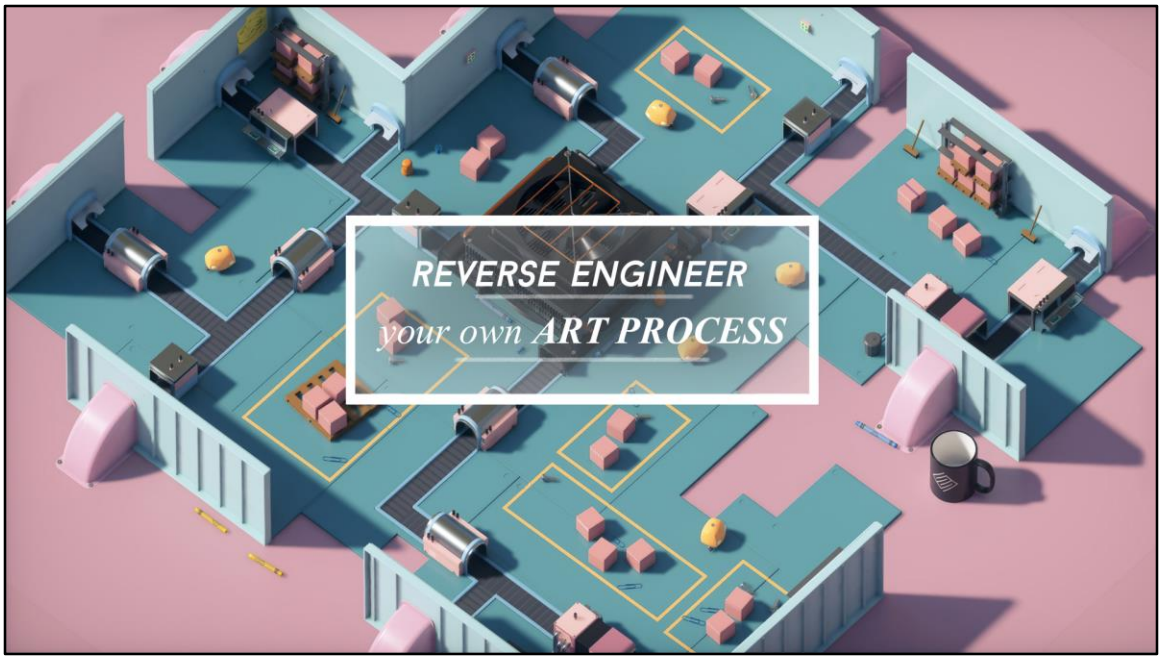
And voila! We have our final shapes that we can populate with procedural and manually created piles.

For the smaller set-dressing however, we have a compositional mass for every object, from which we construct this heatmap, of where the asset is more likely to get spawned. For example, machinery has this vivid blue around them to ensure that no asset prevents agents from navigating around those machines. And yet again, we can layer our compositional gravity principle on top of that just like we did with circle placement.



And once we have all the algorithms combined, this becomes the result of our possibility space, the result of us approaching proceduralism as an art direction problem.

And to summarize, I'd like to give you, what I think, is a key takeaway when it comes to inter-discipline of art and proceduralism.



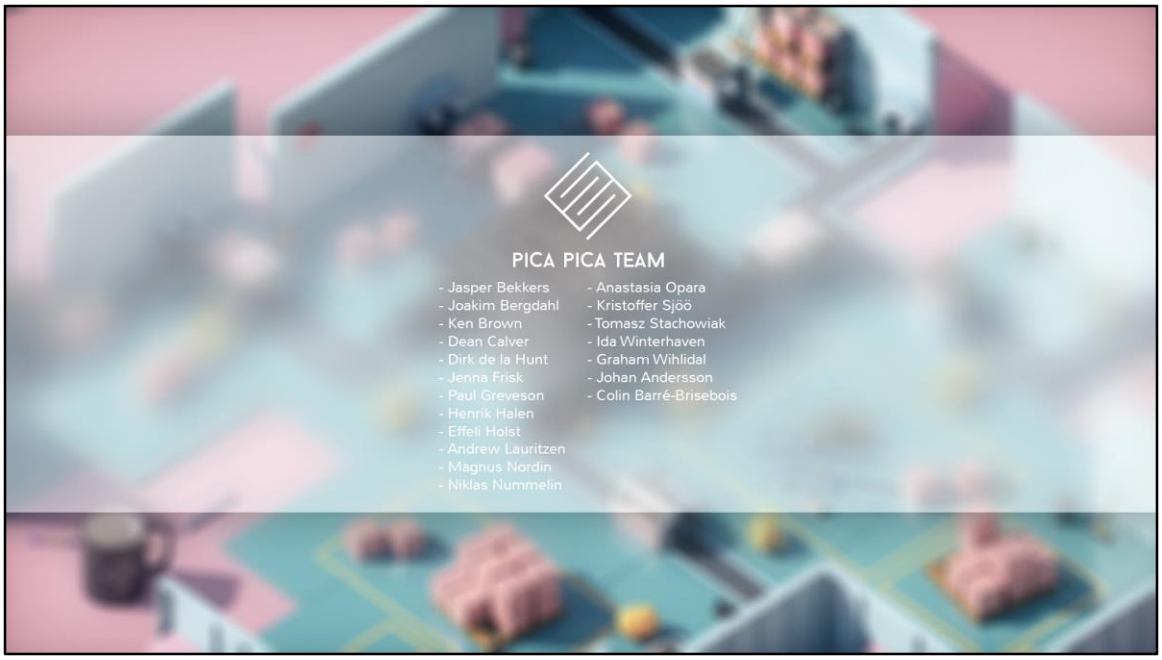
As artists we exploit familiar patterns for which we are attuned and specialized for through evolution, culture and everyday experience. Our visual systems are optimized for the patterns of our environment since being able to encode and process sensory information efficiently is vital due to how costly the brain maintenance is. And art was always about the investigation of brain and perception. The tricks and cut corners our mind employs. We see with our brain, not our eyes. So you can say that beauty is in the brain of the beholder. And to study that beauty, we need to reverse engineer our brain, our perception, our art process.

Brain actively anticipates upcoming sensory input rather than passively registering it opening a door for us, artists, to

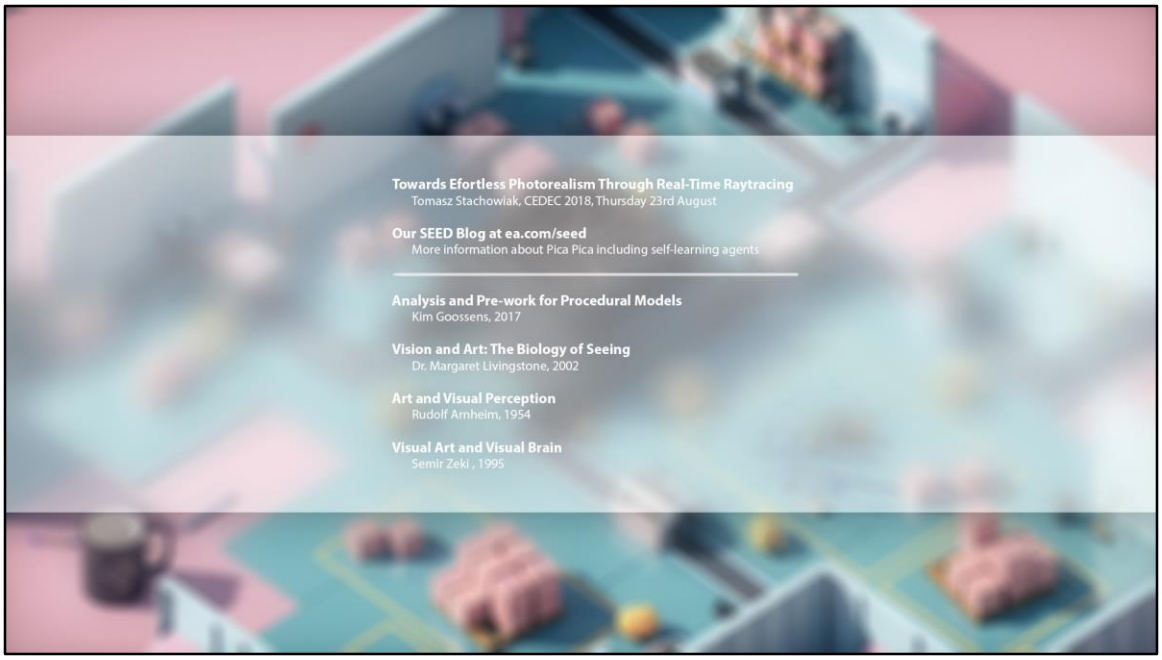
manipulate these expectations thus eliciting emotional response. And if traditional art was about higher level perception of color, light, shapes, I believe, Proceduralism is about investigation of our thinking in categories, rules and patterns.

And before we wrap up…

I’d like to give a shout out to our Art Team, the results of whose amazing effort on the visuals you witnessed today, as well as….

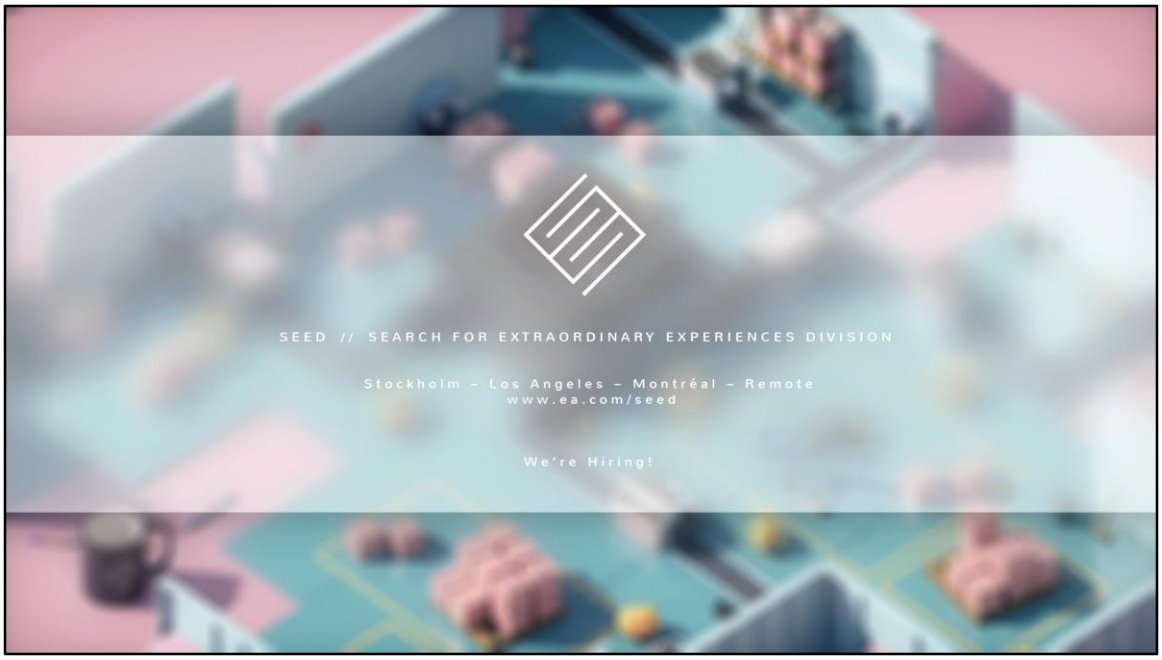


the rest of Pica Pica team. You guys rock!



And if you are interested in the rendering aspects aspect of the project you should check out the brilliant talk by Tomasz Stachowiak here at CEDEC called “Towards Effortless Photorealism Through Real-Time Raytracing” as well as our SEED blog, where you can find about self-learning agents in the Pica Pica project. And if you are interested in learning more about procedural approach, I very much recommend checking out Kim Goossens ’Analysis and pre-work for procedural models’, and yeah that would be it…

Thank you very much for listening. And if we have time for questions, I’d gladly hear them out.


SEED // SEARCH FOR EXTRAORDINARY EXPERIENCES DIVISION
Stockholm – Los Angeles – Montréal – Remote
www.ea.com/seed
We're Hiring!