Postmortem: Stardock and Oxide Games' Ashes of the Singularity

gamasutra.com/view/news/270965/Postmortem_Stardock_and_Oxide_Games_Ashes_of_the_Singularity.php

Brad Wardell is the founder of Stardock Entertainment and the cofounder of Oxide Games

Ashes of the Singularity is a real-time strategy game that takes place in the distant future where humanity has begun to expand into the galaxy. Each match takes place on a particular planet where the player builds and controls vast armies to conquer it.

Released in April 2016, *Ashes of the Singularity* was a fairly well known quantity in hardware circles due to its distinction as the first native DirectX 12 game (it runs on Windows 7/DirectX11 as well). The question on many people's minds was whether the game itself could outshine its various technological achievements.

This postmortem will walk you through a few of the things that went right, but mostly we want to focus on the things that didn't go right. That way, other developers, as well as gamers, can gain a better understanding of the process for creating a brand-new large-effort game, from scratch, in 2016.

Reception

Within the first week of the game's release, it made the top sales charts and has received some very positive reviews due to its strong replayability, lots of maps, good AI, and multiplayer features as well as some mixed reviews due to the expectation that RTS games should focus more on a story with characters. We will discuss the lessons we learned from this.

In our minds, good strategy games create their own stories from their game play. However, that doesn't somehow make the criticism of the game's relatively light campaign invalid. As game developers, our job is to deliver games based on the demands of the market.

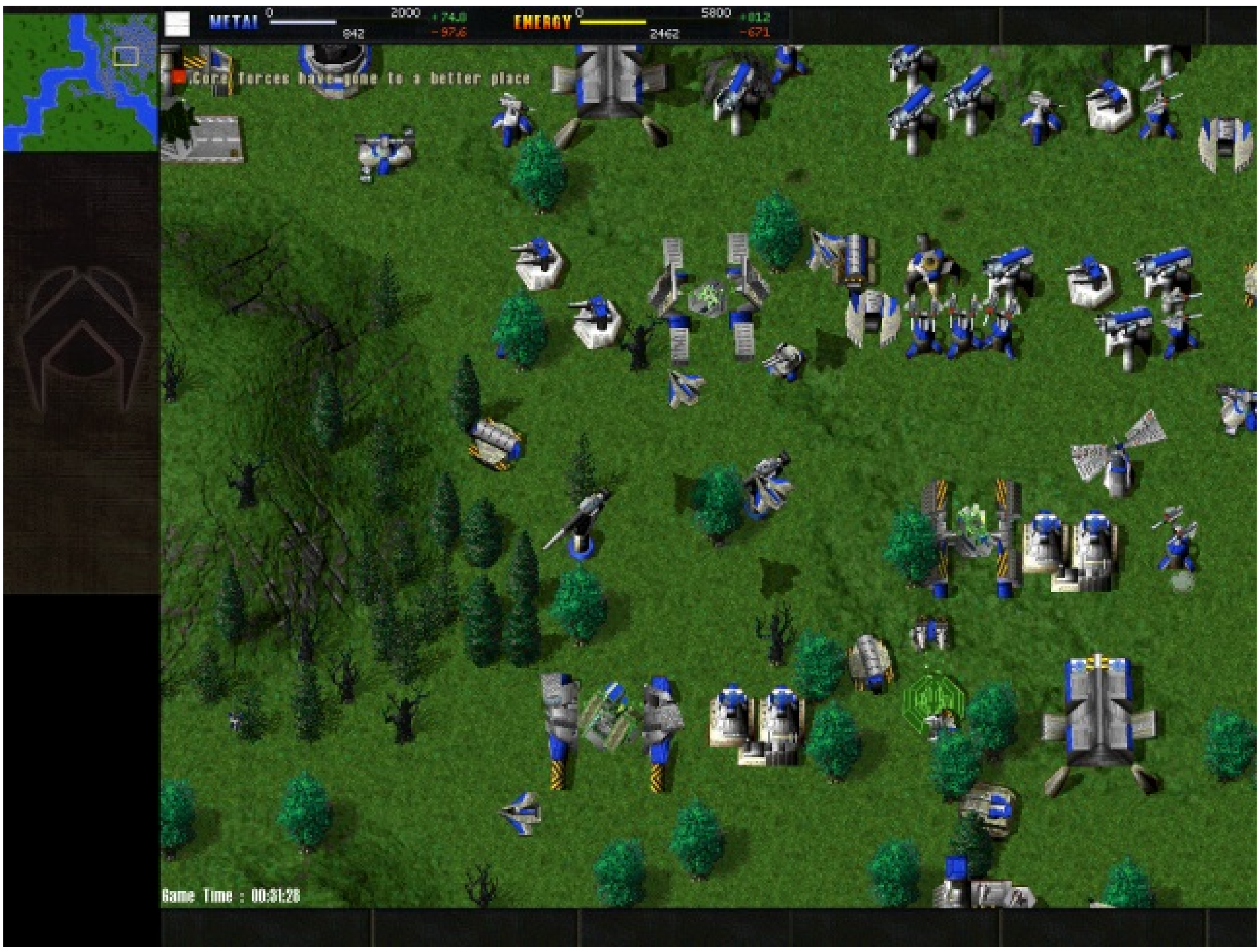
In short, *Ashes* has had a very successful launch. And, lest one is tempted to say "Of course you'd say that," I will point you to my last postmortem.

How we got here

No game is made in a vacuum. Well, not yet. I mean, they could probably make a game in space in space suits, I guess. Maybe next time.

Anyway, the origin of *Ashes of the Singularity* goes back to 2000, shortly after Cavedog (the makers of *Total Annihilation*) closed. We had recently worked with Blizzard and GT Interactive on the *StarCraft* expansion, *StarCraft: Retribution*. Our proposal was to license the *Total Annihilation* engine and create either a new game or a *Total Annihilation II*.

The



original Total Annihilation

Our proposal included two primary gameplay additions to *Total Annihilation*. First, we wanted to let players zoom out further if they wanted. Secondly, we wanted to allow players to build a new series of structures called "Orbitals" that would give them global abilities.

The idea was that *StarCraft* units had abilities and we wanted to give players the ability to act tactically, but without turning the game into a click-fest. Thus, building an orbital would unlock a new button on the screen that would have a cool-down. An example would be, for instance, an ability that would let the player insert a construction unit on a far off big map if they had vision or an ability that would give units a temporary shield while engaged in combat.

Unfortunately, GT Interactive went defunct and was acquired by Infogrames. So no *Total Annihilation 2*, and no new game. We were starting to feel cursed.

Build a game engine? Or license a game engine?

After it became clear that a new RTS built on the *Total Annihilation* engine was no longer viable, Stardock, like many game developers of that time, found itself struggling to build a third-generation game engine.

This talk of generations and game engines probably requires a little background. Briefly:

- **1st Generation:** DOS, 16-bit, 640K max, single-threaded 320x200 max with 256 colors, sprites
- **2nd Generation:** Windows, 32-bit, single core, 640x480 typical with 256 colors, sprites
- **3rd Generation:** Windows, 32-bit, single core, 2GB max memory, DirectX 9c

By 2006, Stardock had just barely managed to cobble together a 3rd generation engine and released *Galactic*

Civilizations II.

From 2006 to 2010, Stardock worked to develop a true 3rd generation engine called Kumquat which was going to be used to power *Elemental* and *Society*. However, in doing so, it learned just how technically challenging it is to develop.

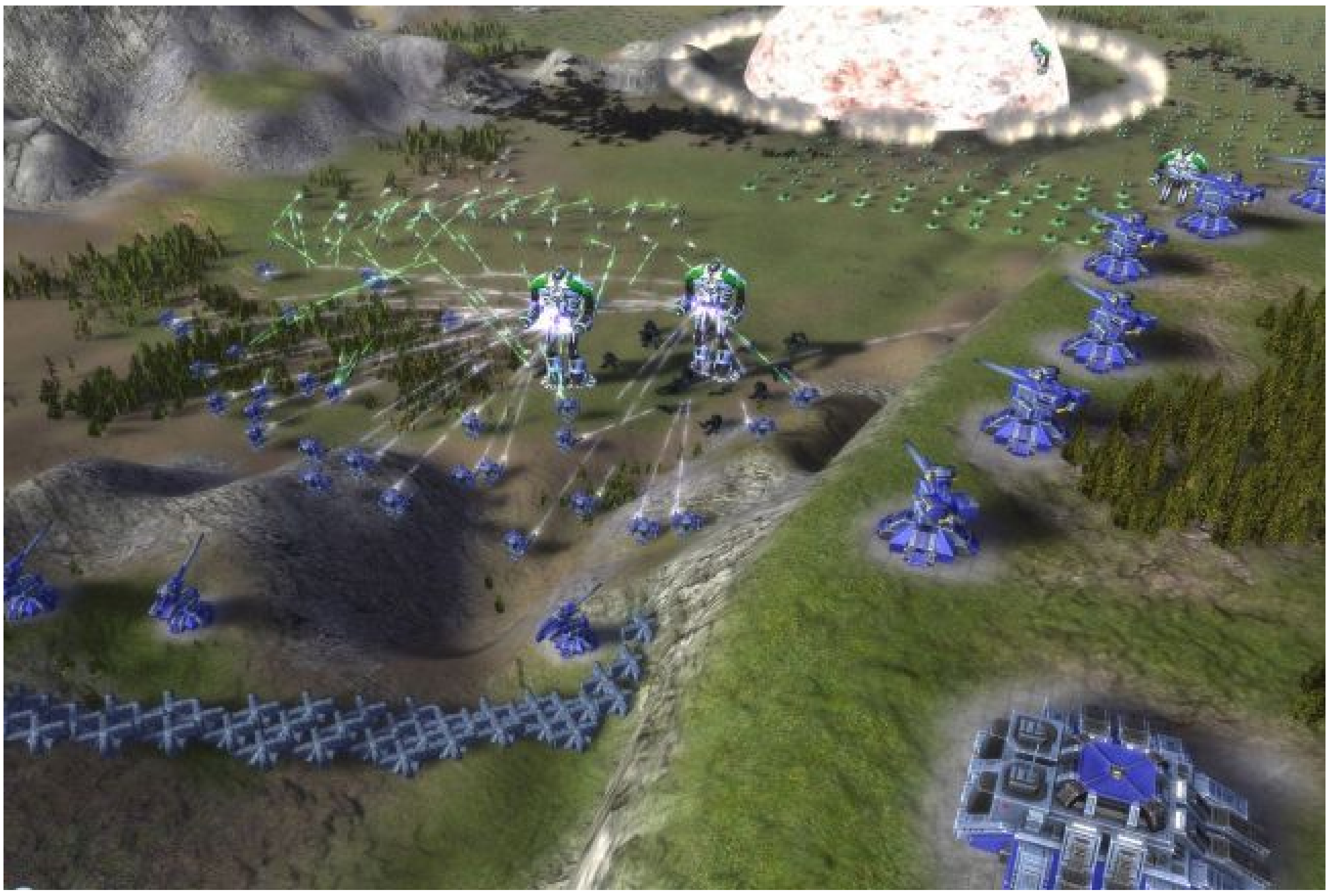
If you, the reader, are wondering why everyone just uses Unity these days, now you know why. Unity is a fantastic, mature, fully featured 3rd generation engine. What they've accomplished is non-trivial.

Stardock learned a great deal of hard lessons while developing its own 3rd generation engine:

1. 2GB of memory disappears very fast if you want to have a large map with hundreds of unique units.
2. You don't really have 2GB. You have whatever the largest chunk of free memory available when you allocate (memory fragmentation). If your clever engine does a lot of dynamic memory allocations, you can run out of memory at 1.3GB with relatively few tools to explain this (2009).
3. The ramifications of some of the DirectX 9c limits had on creating really cool, deformable, randomly generated terrain in terms of texture limits, vertices, etc.
4. DirectX 9 only lets the main thread (the one that launches the game) actually talk to the video card, making it very difficult to get any sort of high fidelity visuals when combined with lots of units.

Stardock's partner, Ironclad, successfully built their own 3rd generation engine (the Iron engine) for *Sins of a Solar Empire*. Similarly, Gas Powered Games had pioneered third generation engines with the *Dungeon Siege* engine, which later became the basis for *Supreme Commander*.

Gas



Powered Games' Supreme Commander

By 2011, it became clear that Stardock didn't really have a viable third generation game engine. This was an existential issue because historically, the company's strength had been based on the scope of its games. If we had to license our engine from a third party, we would lose some of our distinctive abilities to make truly custom games.

For example, Paradox's development studio has a third generation engine called Clausewitz that they've used to build *Europa Unversalis, Hearts of Iron, Victoria, Crusader Kings*, and *Stellaris*. As a result, Paradox has been able to leverage their powerful engine to make a host of outstanding and unique game titles that are also very robust.

In short, Stardock needed a robust game engine if it wanted to continue to develop distinctly unique large-scale game designs.

Oxidation

For a long time, we've dreamed of making a game where the player could zoom in and see individuals living out their lives while zooming out to see an entire world with various wars being fought out by dozens or even hundreds of players (ranging from nation states down to liberation fronts).

Unfortunately, no such engine existed and in fact, such an engine would have to be a theoretical 4th generation engine. What would be a 4th generation engine?

- 64-bit
- Scale seamlessly with multiple cores
- Every core could talk to the graphics API simultaneously

Only an engine that performs as listed above could seamlessly deliver high fidelity visuals and massive scope at the same time. Given that Stardock had struggled to create a 3rd generation engine, even thinking of building a 4th generation engine was daunting.

Then…a miracle happened.

Jon Shafer (*Civilization V*) and Soren Johnson (*Civilization IV*) were working with us on other game projects and introduced me to several of the former leads of *Civilization V*: Dan Baker, engine lead, Tim Kipp, systems lead, Brian Wade, lead developer, and Marc Meyer, UI engine lead.

With *Civilization V* done, they were looking to do something incredibly ambitious. They wanted to create a new type of game engine that would allow for strategy games of nearly unlimited design sophistication and with the literal fidelity of CGI in movies.

Their dream was my dream.

But could they actually deliver such an engine? What they wanted to create wasn't just a 4th generation engine, but one that promised to deliver visual fidelity beyond anything we had previously considered.

To be clear: their engine was going to use object space lighting like you'd see in a CGI movie, except in real-time. While I was familiar with deferred rendering, forward rendering, and forward+ rendering, what they proposed was something beyond my development experience.

Whenever someone suggests they do something like do movie like CGI rendering in real-time the obvious question is: If this were possible, why hasn't someone done it yet? That was why I was so skeptical of their proposal. However, their credentials gave me confidence that they could actually pull it off.

First, they had just finished developing the most successful strategy game of all time, *Civilization V* (even as I write this, it has over 50,000 people playing it simultaneously). Secondly, each of them had a long track record of technical success. Dan was one of the developers of DirectX, Brian had been the lead developer and/or AI programmer on countless well known and beloved PC games, Marc had developed the UI system that helped make *Civilization V* such a success, and Tim was the guy who built a threaded animation system for the Xbox 360 and helped build the engine for *Battle of Middle Earth II*.

Their proposal: fund their start-up and they'll build the engine that would solve all or problems. I did. And so did they.

Ashes of the Singularity is born

Now that *Ashes* is out and is a success, there is always the temptation to look back and think "if only we had spent more, we could have launched with so much stuff. But in 2012, the project seemed very risky.

Stardock's Ashes of the Singularity

Let's recap on their proposal (understanding that this is 2012) for their new engine:

1. It would be a core-neutral engine. That means, there is no "main thread". There is no "graphics thread". The more cores you add, the better the engine will perform. As a reminder: In 2012, most people only had single core CPUs. Core-Duos were relatively new and only crazy people had more than 2 CPU cores.
2. The engine would be 64-bit only. You wouldn't be able to even use it unless you had a 64-bit machine.
3. Every CPU core would talk to the graphics API (DirectX) - something that, at the time, no one to my knowledge was doing. Moreover, DirectX (at the time) serialized its output to the GPU so this very expensive engineering endeavor would have limited benefit.
4. The graphics engine would render using object space rendering (OSR) as opposed to deferred rendering, forward rendering, or forward+ rendering. OSR is basically how CGI in movies is done (think Pixar's "Bug's Life" level rendering but in real-time) and use OSL (Object Space Lighting).

The backdrop of this proposal can't be ignored: Stardock had just been burned spectacularly in the most public o ways by trying to develop an excessively ambitious 3D engine for a fantasy strategy game.

Fortunately, we had recently set up a fund specifically to invest in new ideas like this. So this endeavor was given a very modest budget.

Designing an RTS on a very limited budget

To put *Ashes'* budget in perspective: *Supreme Commander* 1.0's reported budget was 9 times bigger and GPG started with the *Dungeon Siege* engine base.

The reason the initial budget for *Ashes* was so small was because it was such a high-risk project. The industry is littered with “a bridge too far” games, ranging from *Strike Commander* to *Jurassic Park: Trespasser*.

When it came time to design the game, we had to keep our budget in mind and Oxide invented a number of ingenious technologies that dramatically reduced the cost of making the game (which we’ll get into shortly).

From a design point of view, we had to be very careful with our resources. For example, myself along with the other Oxide founders, are engineers. That meant that the art for the game would need to be contracted out either to Stardock proper or to third-parties and it had to be done in such a way that wouldn’t kill our budget.

Some examples of where we reduced our budget included:

- Focus on the sand box game (skirmish)
- Limited animation (units are inorganic, don’t walk)
- Limited map assets (e.g. no fighting in cities)

What we lacked in art assets, however, we made up for with gameplay features. The goal was to have a game with “good bones” to build from.

If the game was successful, we would have a new IP built on an engine that has an indefinite lifespan ahead of it. A 64-bit, core-neutral engine with “real” lighting won’t get dated any time soon. We could build on that – as long as the base game succeeded in having “good bones”.

Mantle & DirectX 12 change the playing field

With all the coverage *Ashes of the Singularity* has received due to it being the first DirectX 12 game, it’s easy to forget that neither it nor Mantle existed when we began development.

When AMD announced Mantle, a new graphics API set to compete with DirectX 11, we were well placed to demonstrates Mantle’s benefits. Since every CPU core in our engine wanted to talk to the GPU simultaneously, *Ashes* got a massive performance benefit.

The bump in performance from DirectX 11 to Mantle was so substantial that it caused some marketing headaches for AMD. For example, during the prototype tests on an AMD 390 we saw performance gains of over 60 percent. However, marketing materials would go out that indicated only a 20 percent boost, because there was concern that no one would believe such a massive jump.

For reference, on the shipping version of *Ashes of the Singularity*, the average frame-rate of an AMD 390 at 2560x1440 with highest settings is 30FPS on DirectX 11 and 53FPS on DirectX 12. That’s an 80 percent boost in performance based on the real-world results of thousands of players submitting their benchmarks.

Meanwhile, Microsoft also saw the real numbers we were getting and began working with us the “new” DirectX they were working on which would become DirectX 12.

The difference between DirectX 11 and DirectX 12 is very easy to explain: On DirectX 11, all your threads in your game can talk DirectX at once but DirectX 11 only talks to the GPU one thread at a time. By contrast, on DirectX 12, all the threads in the game can talk to the GPU at the same time.

In theory, a 10-Core Intel Broadwell-E, for instance, could do 10X the performance in DirectX 12 than in DirectX 11 -- provided the video card was fast enough to keep up.

What went wrong?

gamasutra.com/view/news/270965/Postmortem_Stardock_and_Oxide_Games_Ashes_of_the_Singularity.php

What went right?

Hindsight is 20/20 and with the game released we can now look at the decisions we made that we are particularly proud of….as well as the decisions that make us wish for viable time travel.

Let's start with what went right.

1. The engine worked!

If you've ever worked on a game engine, please comment below to share your own experiences with others so that they can understand the significance of this.

I cannot overstate how important this is. Stardock/Oxide now, in 2016, have a very powerful 4th generation engine that can be used on countless future games. That means future games like *Star Control* and lots of unannounced games can make use of it. You should see the aliens in the new *Star Control* game using OSL; they look something out of a CGI movie, except they can talk to you in real-time.

One can load up *Ashes of the Singularity* and instantly imagine lots of games that could be easily made from this. So not only can we establish Ashes of the Singularity as a go-to forward looking RTS but we can look at lots of other genres too.

2. The benchmark

Because we were going to be the first to use Mantle (and DirectX 12), we needed to make sure that we didn't end up with arrows in the back due to bugs in the API or driver issues. Thus, the benchmark was born. The idea behind the benchmark was to help identify and track progress of the APIs, the drivers, and of course our own engine progress. As part of this venture, Microsoft, AMD, and NVIDIA were all given direct access via Perforce to our code repository so they could make their own custom builds to try out different optimizations.

This also allowed us to get a lot of coverage that we otherwise wouldn't have gotten which makes launching a new IP a lot easier.

3. Procedurally generated maps

How many RTS games have you bought over the years that you liked but only had a handful of maps? Sure, the maps are nice, but a half dozen maps just doesn't cut it and waiting a year for an expansion just kills it.

The maps in *Ashes of the Singularity* are procedurally generated. That is, we design them but they are somewhat less labor intensive to create than the typical RTS that uses a tile based map system.

As a result, *Ashes* shipped with almost 50 maps on its first day, all of which are different from one another. It also allows us to be fairly generous with new maps with us only needing to charge on DLC that includes new art assets as opposed to the labor of creating the map design.

4. The multi-core AI system worked

AI is usually the weakest part of an RTS. In fact, it's so assumed that real-time strategy games will have bad AI that it has caused us some grief in our reviews because some reviewers (you know who you are) barely bothered with the sandbox game, focusing instead on the scripted (and relatively bare bones) campaign. It didn't occur to them that the single player sandbox was the meat of the game.

In *Ashes*, the AI operates on multiple CPU cores at once (a minimum of 4) and does so asynchronously from the rest of the game simulation. This is important because you can watch, in real-time, the AI adapt and counter your strategies. It also matters because as we get better at the game ourselves (or ahem, watch others) we can keep making the AI better and better.

5. Working with AMD & Microsoft

When you partner with large companies, you can often suffer if they are unresponsive. However, we were constantly impressed with just how nimble AMD and Microsoft were in helping us solve problems, providing marketing support, listening to feedback and getting us hardware. If you work at a hardware vendor and you want your product to be supported, get people who will do something with it. It makes a huge difference.

6. The scripting system

The scripting system was built mostly for the benchmark, which in turn made it very easy for us to later build a series of custom missions for the campaign.

The scripting system in Nitrous gives someone without access to the source the ability to change the camera, spawn units, bring up text, play audio set up players, and countless other things that will be of huge benefit in the future.

7. Building a multiplayer ladder

Stardock was able to recruit Adrian Luff, a long-time veteran of Blizzard who helped in the creation of Battle.net, to architect our own multiplayer meta game service to use on *Ashes of the Singularity* and *Offworld Trading Company*, as well as other games in development such as the new *Star Control*.

Having this in the game on day 1 helped ensure that we not only had a good multiplayer community to build on, but that it would not simply disappear after the initial rush.

8. Design & AI were developed in tandem

For strategy game designers, it is important to keep in mind how your design will affect the AI. Brian Wade, who was the lead developer at Firaxis and worked closely with Sid Meier was in charge of the AI, and I, the AI developer of *Galactic Civilizations I/II*, was the designer.

Together, we could plan out game design changes and brainstorm how the AI might make the most of such changes.

In all, we were very happy with how the project went. However, hindsight being what it is, we also made a number of mistakes that we can grow from in order to do better in the future. Since *Ashes of the Singularity* 1.0 is a brand-new IP in the age of digital distribution, we are in the fortunate position to take these lessons and improve on the game as we go forward.

1. Too small a budget

What I wouldn't have given to have had more artists. Or, you know, artists. Oxide employs a total of 2 of them. We were able to work with some of the best artists in the industry who gave us absolutely amazing concept art, but we often lacked the manpower to make full use of it for 1.0 due to budget.

When Windows 10 shipped with DirectX 12, we knew the game was going to be a winner. Unfortunately, we were so far into the project that a much bigger budget wouldn't help much. The game was designed around our initial budget.

2. Non-dedicated lead designer for most of the project

This is not just me falling on the sword. For most of the game's development cycle, I was not available to focus on the day to day design decisions that were needed.I took on the responsibility of lead designer, but because of all of my other obligations and projects, I was unable to focus on it exclusively until Summer 2015 (only 8 months before it shipped). And of course, "exclusively" in my case means exclusive - except for my day job running Stardock, designing products like Start8, Object Desktop, Multiplicity, *Sorcerer King*, and keeping an eye on all our other endeavors.

One big reason this was a problem for us was that we wasted some of our very finite budget with re-designs. By the time I was able to look at whether a particular game design that seemed good on paper worked or not, it was fully implemented, which means changing it was much more expensive.

Some examples of how my unavailability hurt us include:

1. Originally, buildings were going to consume power that came from the power generators. Thus, you wouldn't be able to build more buildings if you didn't control enough regions. This was one of the many decision directions we had early on that made the game overly reward rapid expansion. We are still having to improve balance on it even now.
2. The Substrate, as their name might imply, originally covered the ground with a material called "substrate" that only they could build on. Besides being too similar to the Zerg in *StarCraft*, it basically crippled the Substrate because rapid expansion + only being able to build on substrate meant they weren't competitive.
3. I wasn't available to choose which unit concepts would ultimately become units and thus, those decisions were made in abstention. When the game was first designed, the mantra was "The units ARE the icons". That is, that the shapes of the units would be so distinct that they would act as icons thus allowing players to zoom out and recognize the units very far away. While the units we ultimately went with are visually very cool, they are not as distinctive as we would like.
4. There was a lot churn on how the buildings would work. For example, at one point we developed an entire system where players could build onto their factories and other buildings to give them upgrades. So for example, your factory could get a lab added to it that would unlock the unit's secondary weapons. The system just didn't work very well for the game, but by the time I had time to fully evaluate it, the feature was largely done and the cost spent.
5. The units were originally designed to all come out in squads. In fact, we envisioned squads potentially having 100+ sub-units in it. For example, you wouldn't build an Archer or a Brute, but instead you would construct an Assault Force or a Picket which would be produced and come out in large numbers. If this idea sounds great as you read it, you now understand how hard game design is. What sounds great on paper stops being so great when you have a new player click a button and have a giant swarm of tiny blobs coming out of the factory.

And that's just a handful of the examples. The point being, the lack of a dedicated lead game designer and my absence on the project meant that even with our tiny budget, we churned through a considerable amount of it unnecessarily.

3. We didn't recognize the growing desire for story in RTS games

I was vocally against having any sort of campaign in *Ashes of the Singularity*. We didn't have one in *Sins of a Solar Empire* and I saw no reason to have one in this game either. Despite being a Diamond *StarCraft* player and having worked on *StarCraft: Retribution*, I have never played a *StarCraft* campaign.

My concept of the game's Ascendancy Wars was essentially a large map of the galaxy with dozens of planets with the player and several AI factions trying to conquer it planet by planet.

However, during development our friends at Uber had introduced a similar concept into their game, *Planetary Annihilation*. Combined with Oxide's strong recommendation that we needed a story-driven campaign and the early access feedback, I reluctantly agreed to have a campaign in the game.

Having a campaign was the right decision, but I wish we had planned on having one from the outset so that we would have had more time to flesh it out and polish it. You can see this in the reviews where some reviewers only played the campaign which was designed only as an introduction to the heart of the game: the infinitely replayable sandbox part.

On a given project like this, you make thousands of decisions. Not recognizing that PC gamers have come to expect highly polished, story driven campaigns was a mistake.

Fortunately, we are in a position to not just update the campaign with voice-overs and more content, we can ensure that future campaigns we add to the game are given the kind of depth that players seem to enjoy.

4. The hardware requirements had a significant impact on our sales reach

Our hardware requirements include a 2GB GDDR 5 video card and a CPU with at least 4 cores. Those requirements cut off about half the user base. We knew this going in and it was a price we were willing to pay to make sure we could create a future-proof game.

What we didn't realize is just how rough that would be for us both in terms of Steam reviews (people buying the game with marginal hardware) and game reviews. It was striking how many game sites struggled to find a reviewer who met the system requirements to review the game. This meant that our game didn't necessarily go out to strategy game experts but whoever could run the game.

5. Steam Early Access hurts initial impressions

I won't say we will never do Steam Early Access again. But I will say that I'm not inclined to use it on a game that is using a brand-new engine.

Throughout the game's development, we had to slowly make our way back from a very low Steam Early Access review score. Many users either don't know or don't care that "early access" means the game isn't done.

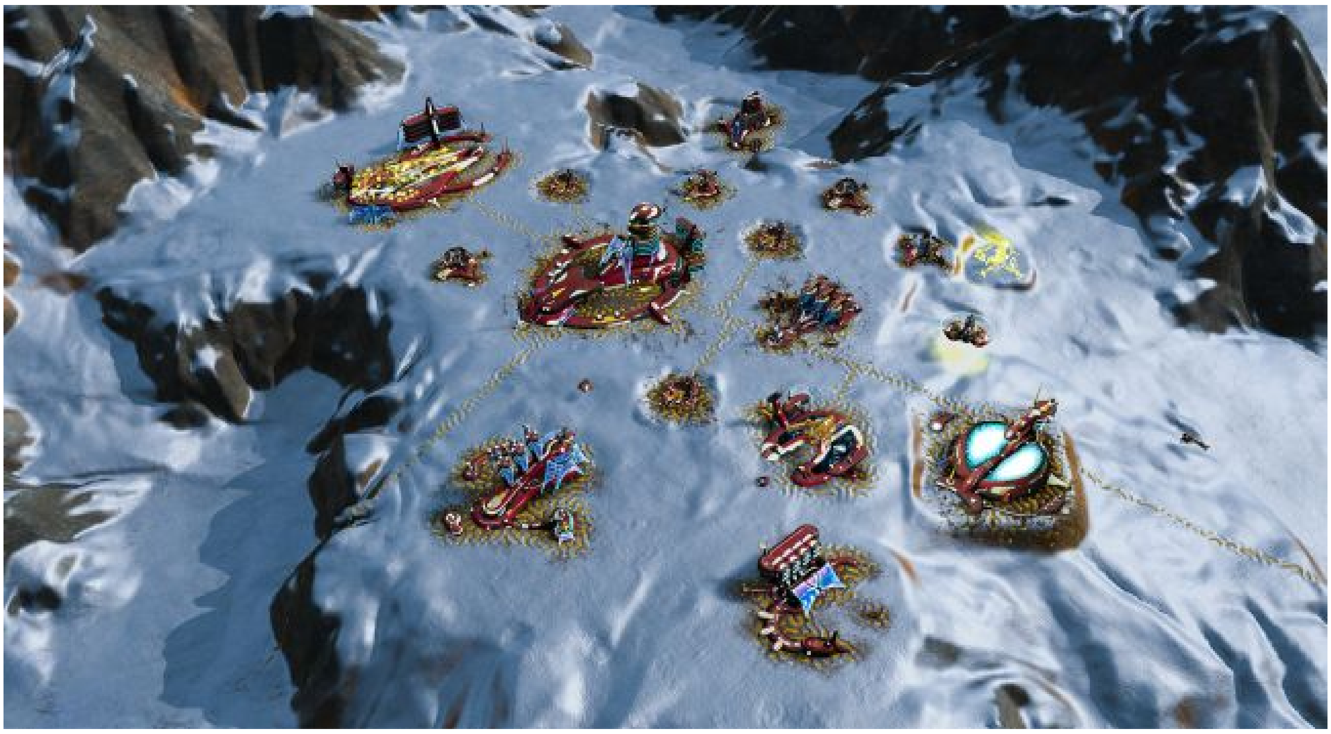
As I write this, post-release, the positive Steam reviews are slowly moving our review score up. But it is not a great situation to start day 1 with "Mostly Positive" and have to try to inch your way back to "Overwhelmingly Positive."

No matter how clear you make the game remind people in Early Access that not all the features are in and that the game will probably have bugs, there is a sufficient percentage of players who will download the game, play it for 10 minutes, find it "unfinished", give it a negative review, and return it.

Even more seriously, we felt locked into our Early Access price. When we started making the game, the price we had chosen was the industry norm. However, since then, the flood of iOS ports and just sheer volume of games showing up on Steam now has pushed down the price people now expect to pay for PC games in general. Thus, our initial $49.99 list price became a significant barrier to entry.

6. Not budgeting terrain eye candy

This is related to the limited budget, but visual eye candy on the maps like ancient ruins or other props were not considered to be important enough to budget art on. Unlike in other areas where we had endless concept drawings for things like units, buildings, etc, we didn't even entertain trying to add "fluff" to the maps since they would not have any game-play effect.



However, post-release, both in reviews and by players we received numerous requests for more "stuff" to be on the maps, just for flavor.

7. Allowing the media to define the game as a "spiritual successor to *Supreme Commander*"

This issue has been tough for us because *Ashes of the Singularity* is most definitely not intended to be a *Supreme Commander* successor. It's much more similar to *Total Annihilation* than *Supreme Commander*. And while *TA* and *SupCom* may seem similar, fans of *Supreme Commander*: *Forged Alliance* will tell you that they are very different.

Moreover, the media defining *Ashes* as a *SupCom* successor offended a lot of *Supreme Commander* online communities who found *Ashes* lacking in the areas they cared about. As a result, the game received a lot of… shall we say…motivated negative reviews online (looking at you, Metacritic user reviews) from these communities.

Ashes of the Singularity is *similar* to *Supreme Commander* in that it has a streaming economy, large scale, and the concept of a constructor and constructor assist. Both those games share that with *Total Annihilation*.

However, they are very different too. *Supreme Commander* is much more focused on base building with adjacency bonuses on buildings and a philosophy of making a player's economy self-sufficient from map resources. By contrast, all resources in *Ashes* ultimately derive from what is on the map. That's a huge difference for some people.

Some final thoughts on postmortems

gamasutra.com/view/news/270965/Postmortem_Stardock_and_Oxide_Games_Ashes_of_the_Singularity.php

Conclusions

As we see the *Ashes of the Singularity* user base grow, we are feeling good about its prospects long-term. Obviously, with a time machine, knowing that the Nitrous engine was going to be able to deliver far beyond our most optimistic hopes, I'd have made sure that the game had come out with a lot more content by giving it a bigger budget.

One of the benefits of the modern digital distribution process is that games are living things. They continue to get updated and evolve long after release. With the game's success we can continue funding it to build up its richness in terms of content and features knowing that the underlying engine will happily use it.

As I write this, I realize I've been writing these postmortems for Gamasutra/Game Developer for 22 years. It is possible I wrote the very first one back in April 1994 (Issue #2 I believe) when I was still in college. It's amazing how much the game industry has changed during that time.

Looking back on all those postmortems, it's interesting how they have changed. Most people don't realize this but when the postmortem series began here, it was done because that was *the end* of that game's life. It was that game's memoirs.

Sure, there might be a sequel or an expansion pack someday, but the purpose of the postmortem was not just to share with other developers what we learned from a given game project. It was also a recognition that the mistakes we made were final and unchangeable.

For example, it's almost amusing now to look back at some of the errors that were considered mistakes. 13 years ago, in my Gamasutra postmortem on *Galactic Civilizations*, one of the big mistakes was the user manual.

Seriously. The user manual. It was one of the big 5 mistakes. Today, we'd update the user manual PDF and call it a day.

The point being, one of the greatest advances in gaming for players and developers is that games are no longer set in stone at release. Lest you think that this just means games today are "buggy" at release, no - they were just as buggy back then. You simply didn't get updates as conveniently as you do now. Who can forget downloading an *Ultima VI* critical bug off a BBS or trying to get the patch for *Black & White* so that you could actually finish the game?

Now, at the same time that games are increasingly treated as disposable, they are, ironically, becoming much better deals. When you buy a game, you can interact with the people who made it. Those people now have the ability to incorporate updates into the game not 6 months later, but 6 days later.

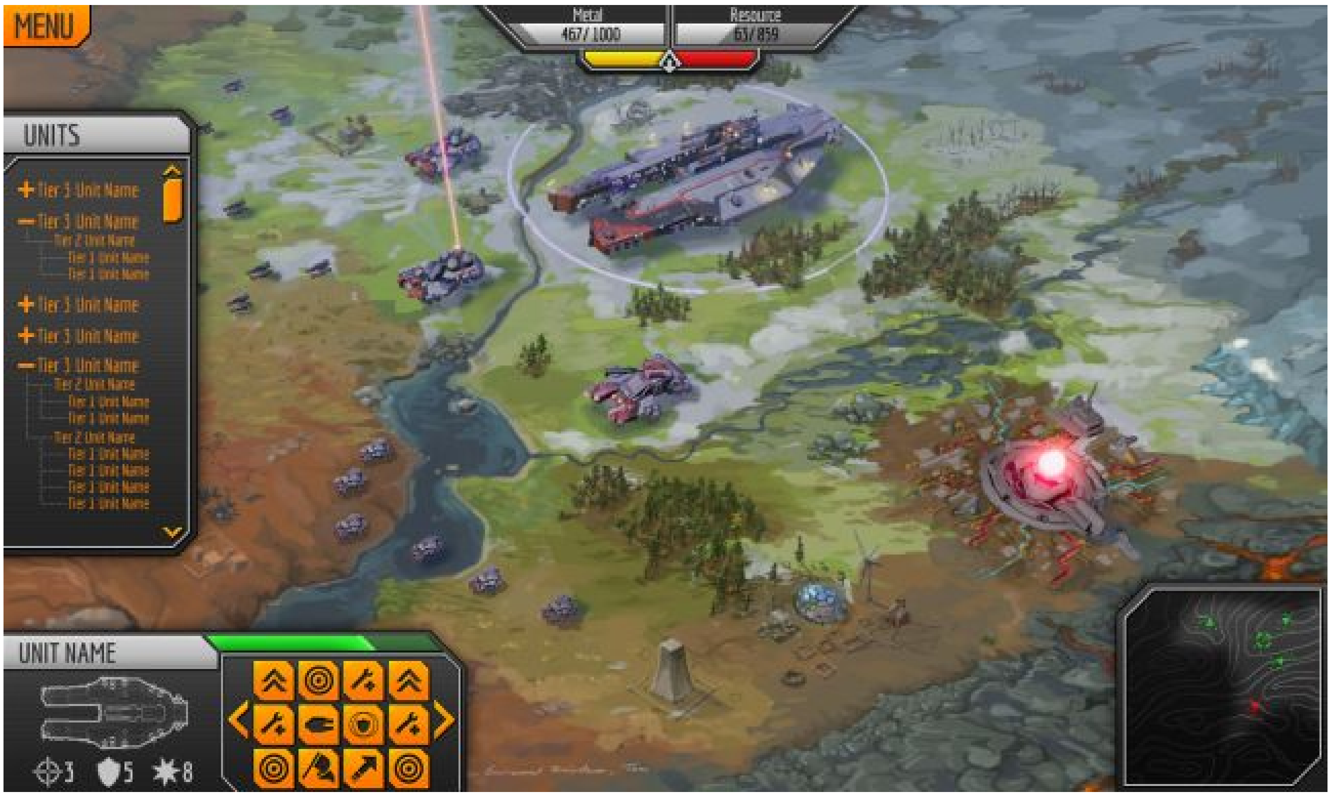
With last year's release of *Galactic Civilizations III* and now *Ashes of the Singularity*, developers like Stardock can take the initial success of these games and build onto them for years at a time.

Thus, a postmortem now is no longer a memoir - it's a roadmap of things that can be done to make your games better.

Development History Walkthrough

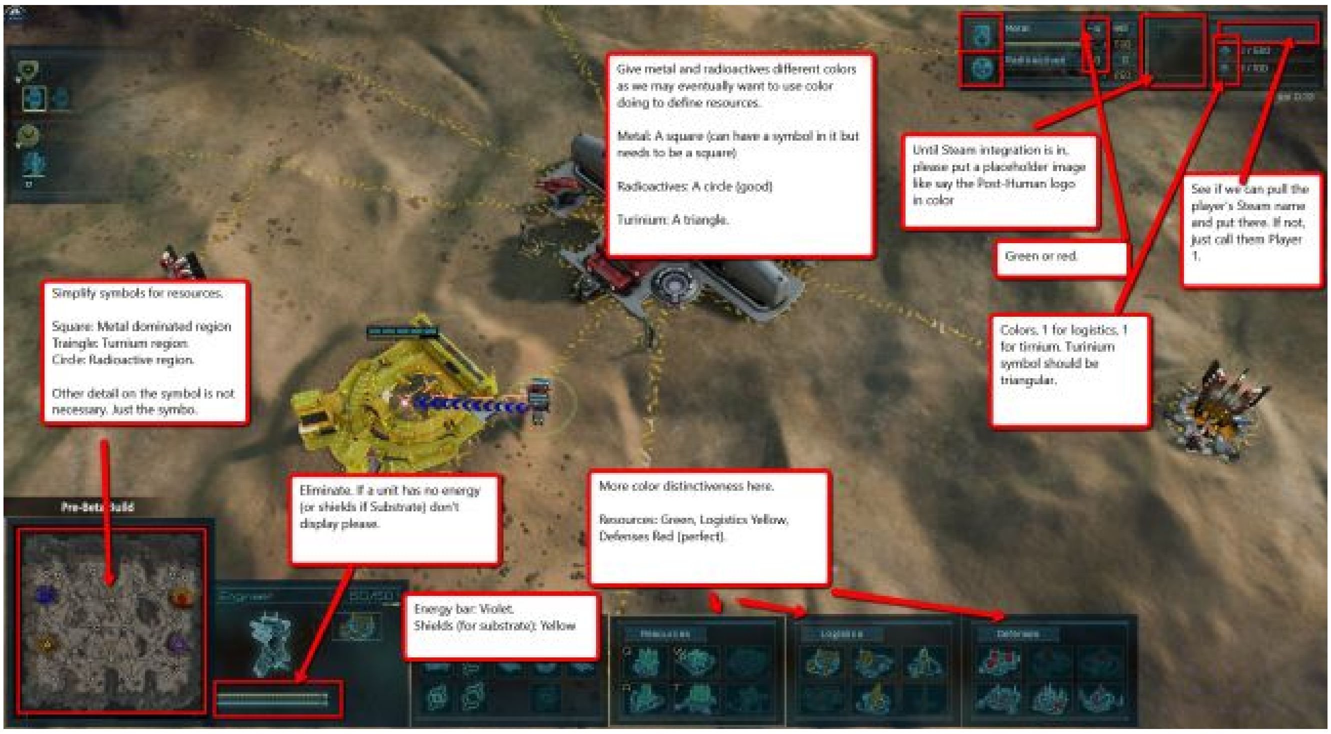
Bonus: Here's a walkthrough of where we were long, long ago.

A very



early UI design concept.

UI


Give metal and radioactives different colors as we may eventually want to use color doing to define resources.

Metal: A square (can have a symbol in it but needs to be a square)

Radioactives: A circle (good)

Turinium: A triangle.

Until Steam integration is in, please put a placeholder image like say the Post-Human logo in color

Green or red.

See if we can pull the player's Steam name and put there. If not, just call them Player 1.

Colors. 1 for logistics, 1 for timium. Turinium symbol should be triangular.

Simplify symbols for resources.

Square: Metal dominated region
Traingle: Turnium region.
Circle: Radioactive region.

Other detail on the symbol is not necessary. Just the symbo.

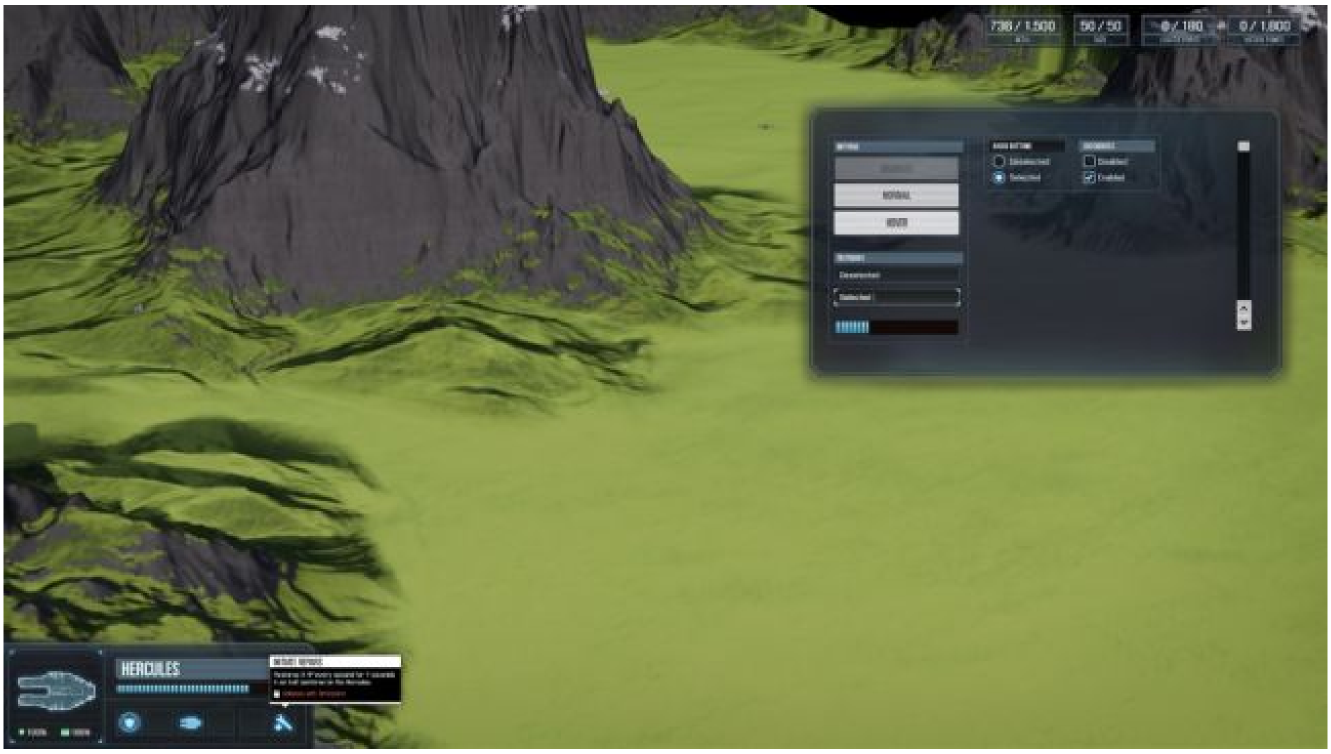
Eliminate. If a unit has no energy (or shields if Substrate) don't display please.

More color distinctiveness here.

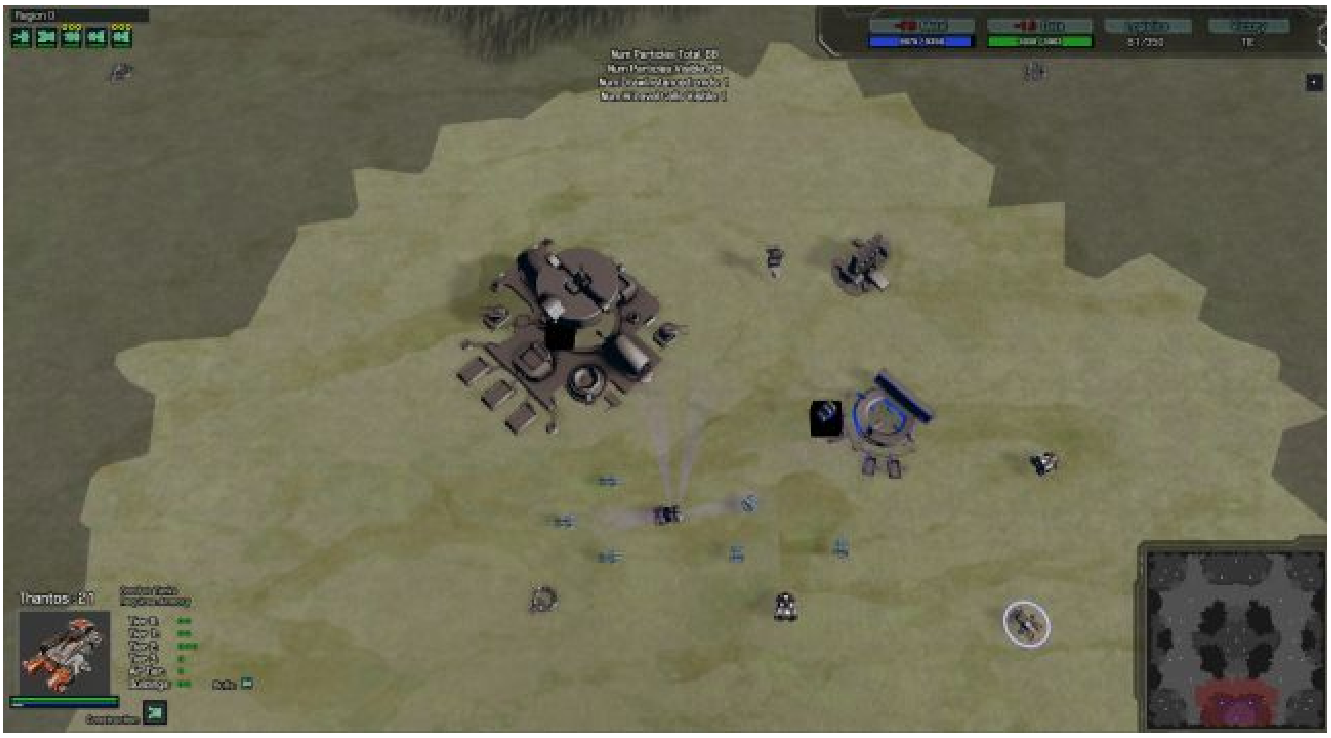
Resources: Green, Logistics Yellow, Defenses Red (perfect).

Energy bar: Violet.
Shields (for substrate): Yellow

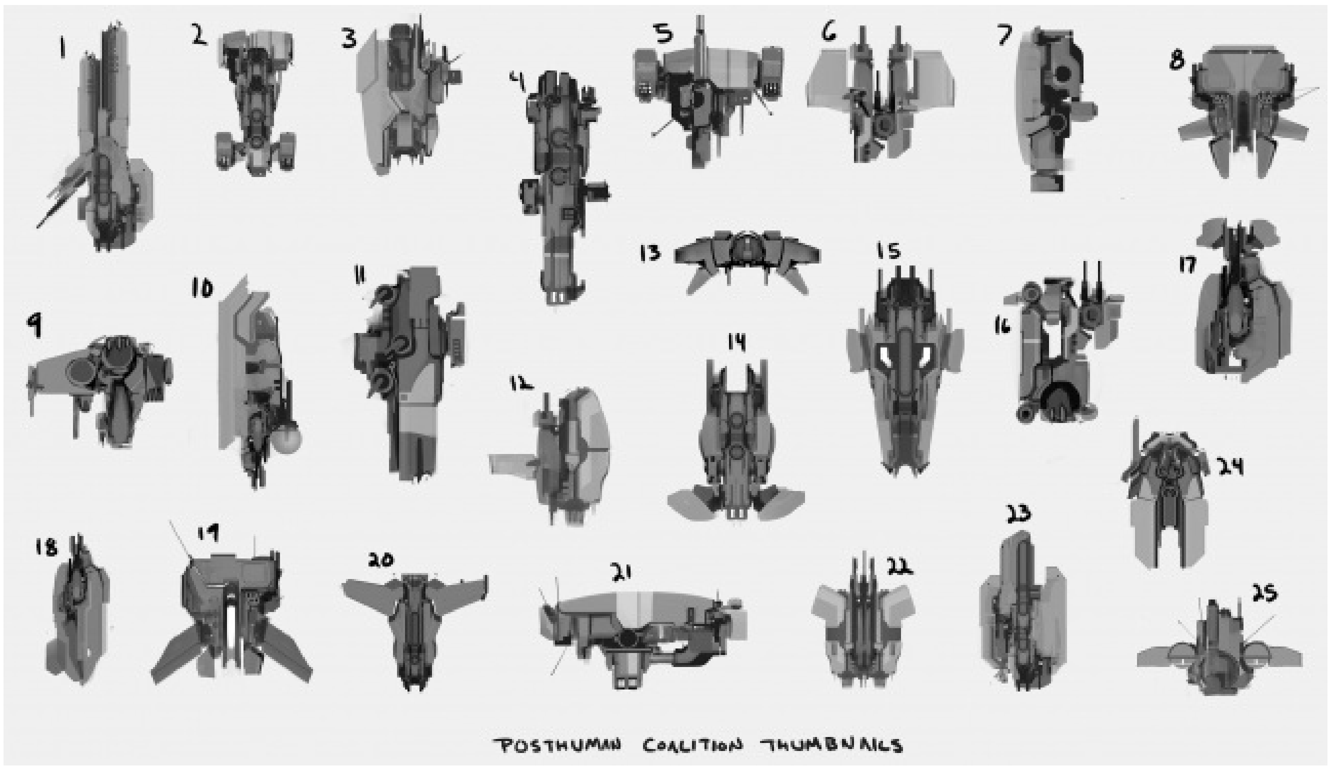
feedback.

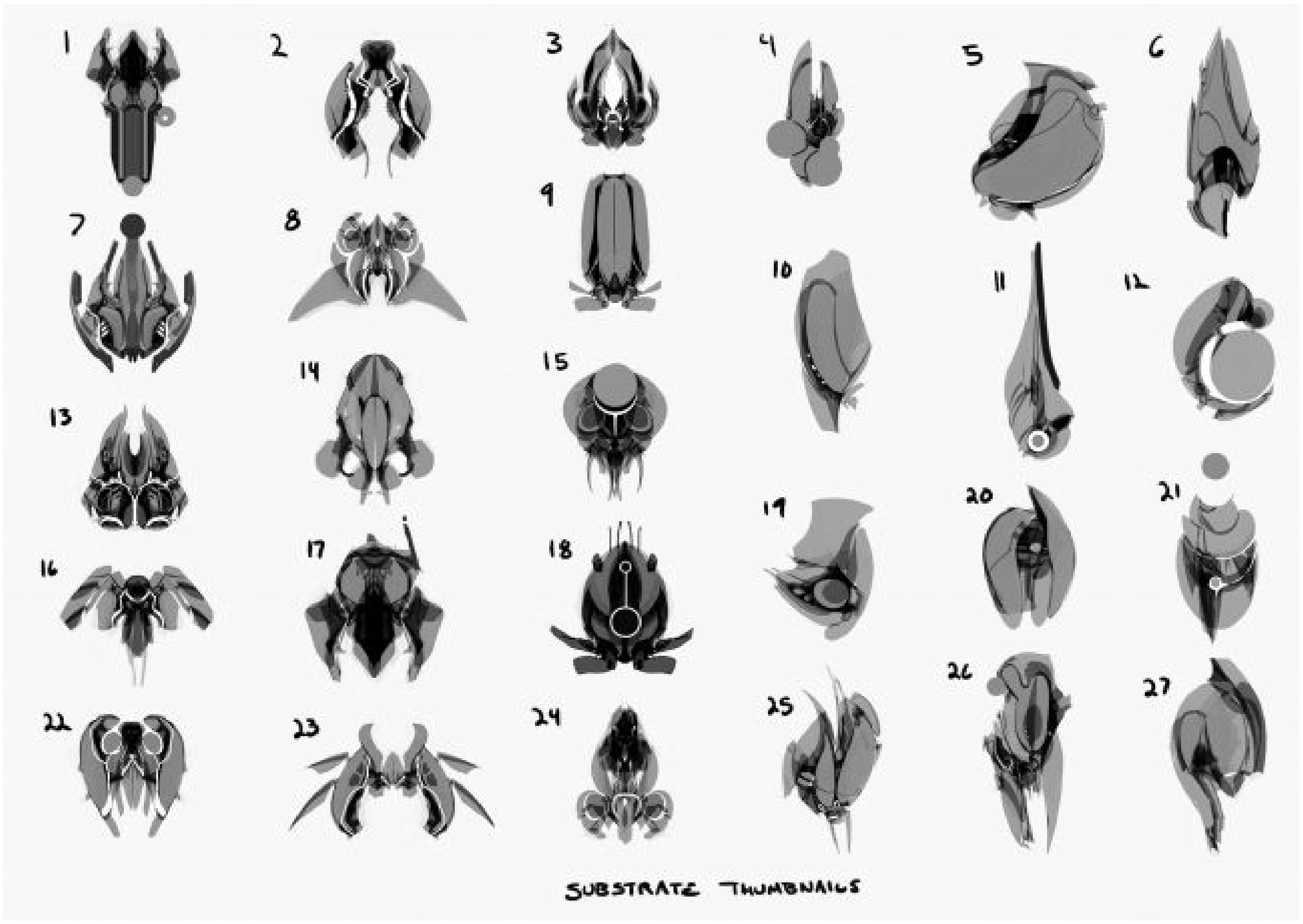


Super early on as we got the terrain system started. Here's an early shot of the terrain system as we got it started.



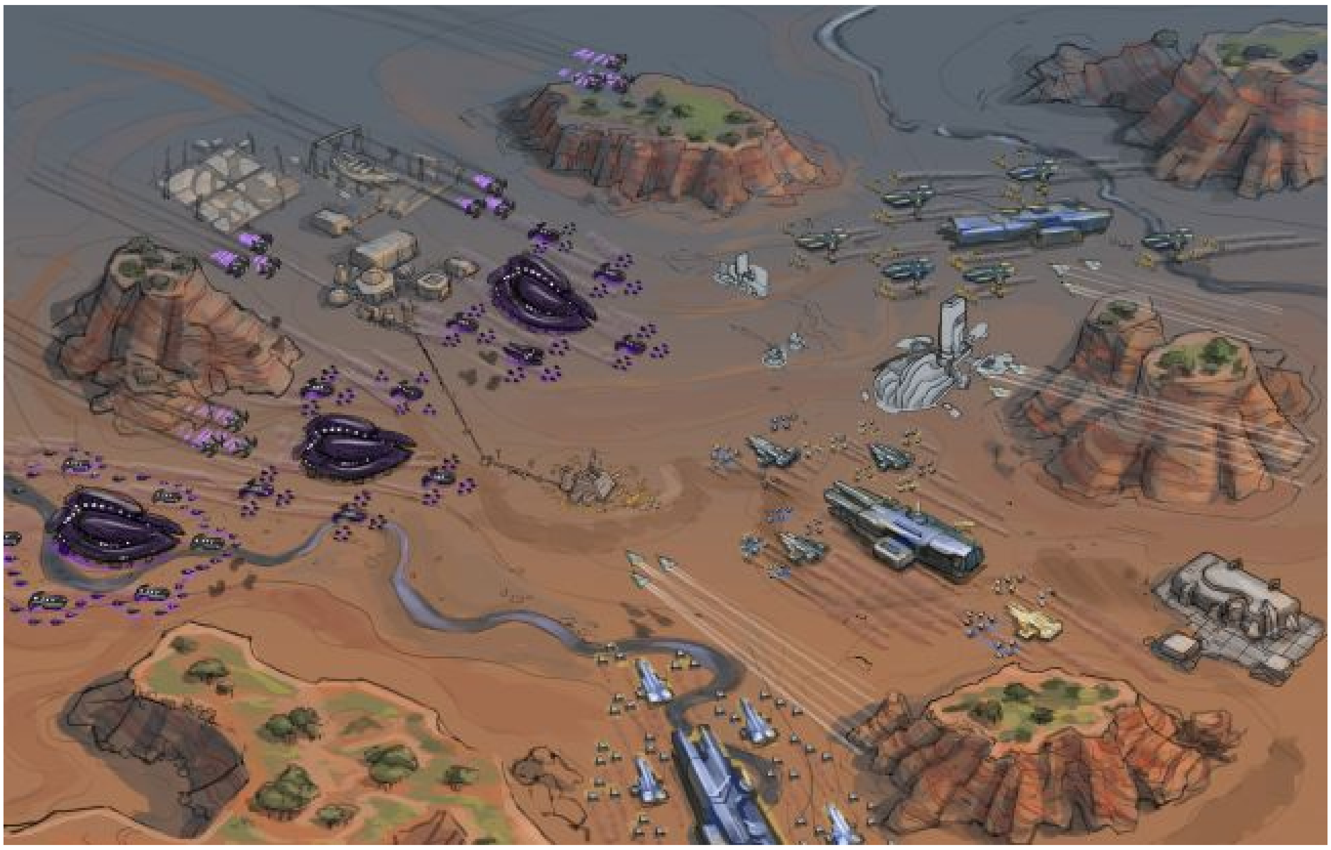
Placing structures onto the map.



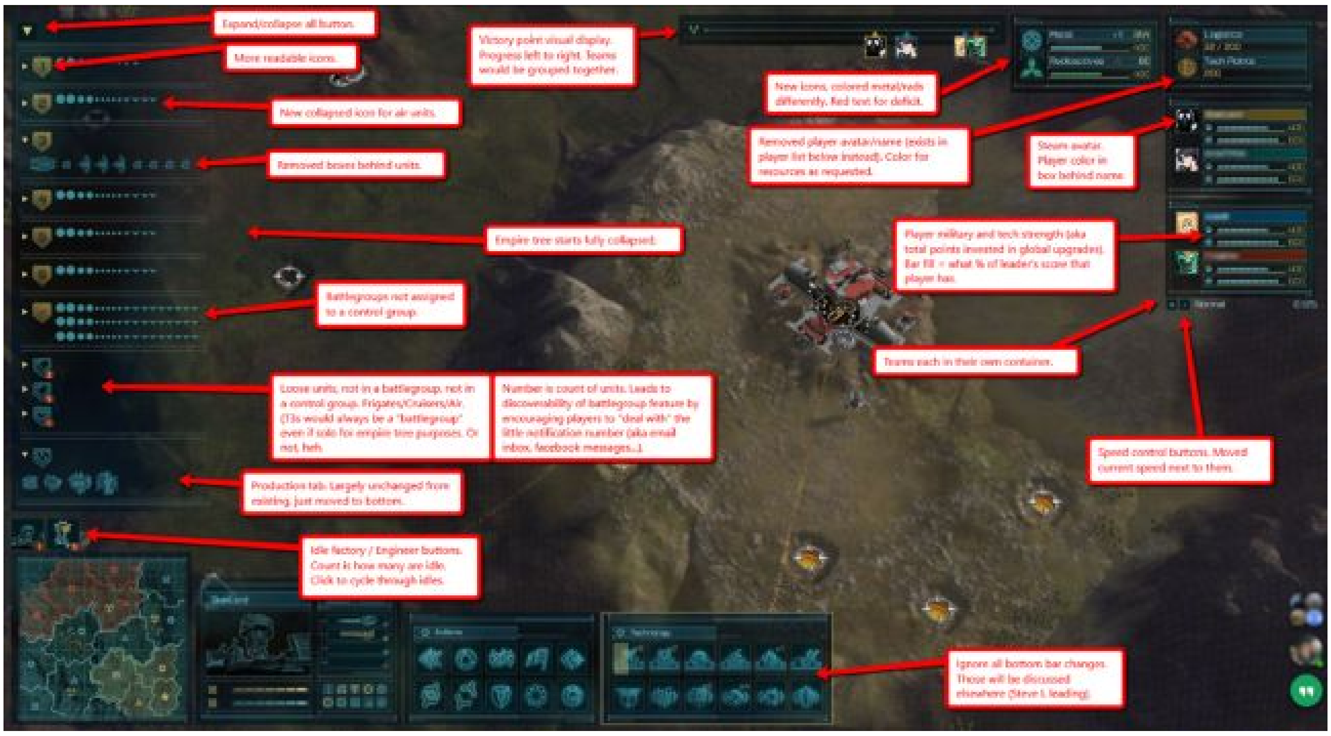


Early on, our mantra was, “the unit is the icon.” For budget and performance reasons, we had to switch to more symmetrical unit designs.In the long-run, we hope to revisit this.

Art



style concept. We didn't want to go for true realism. We wanted something in between cartoony and realistic.



Expand/collapse all button.

More readable icons.

Victory point visual display. Progress left to right. Teams would be grouped together.

New collapsed icon for air units.

Removed boxes behind units.

New icons, colored metal/rads differently. Red text for deficit.

Removed player avatar/name (exists in player list below instead). Color for resources as requested.

Steam avatar. Player color in box behind name.

Empire tree starts fully collapsed.

Player military and tech strength (aka total points invested in global upgrades). Bar fill = what % of leader's score that player has.

Battlegroups not assigned to a control group.

Teams each in their own container.

Loose units, not in a battlegroup, not in a control group. Frigates/Cruisers/Air. (T3s would always be a "battlegroup" even if solo for empire tree purposes. Or not, heh.

Number is count of units. Leads to discoverability of battlegroup feature by encouraging players to "deal with" the little notification number (aka email inbox, facebook messages...).

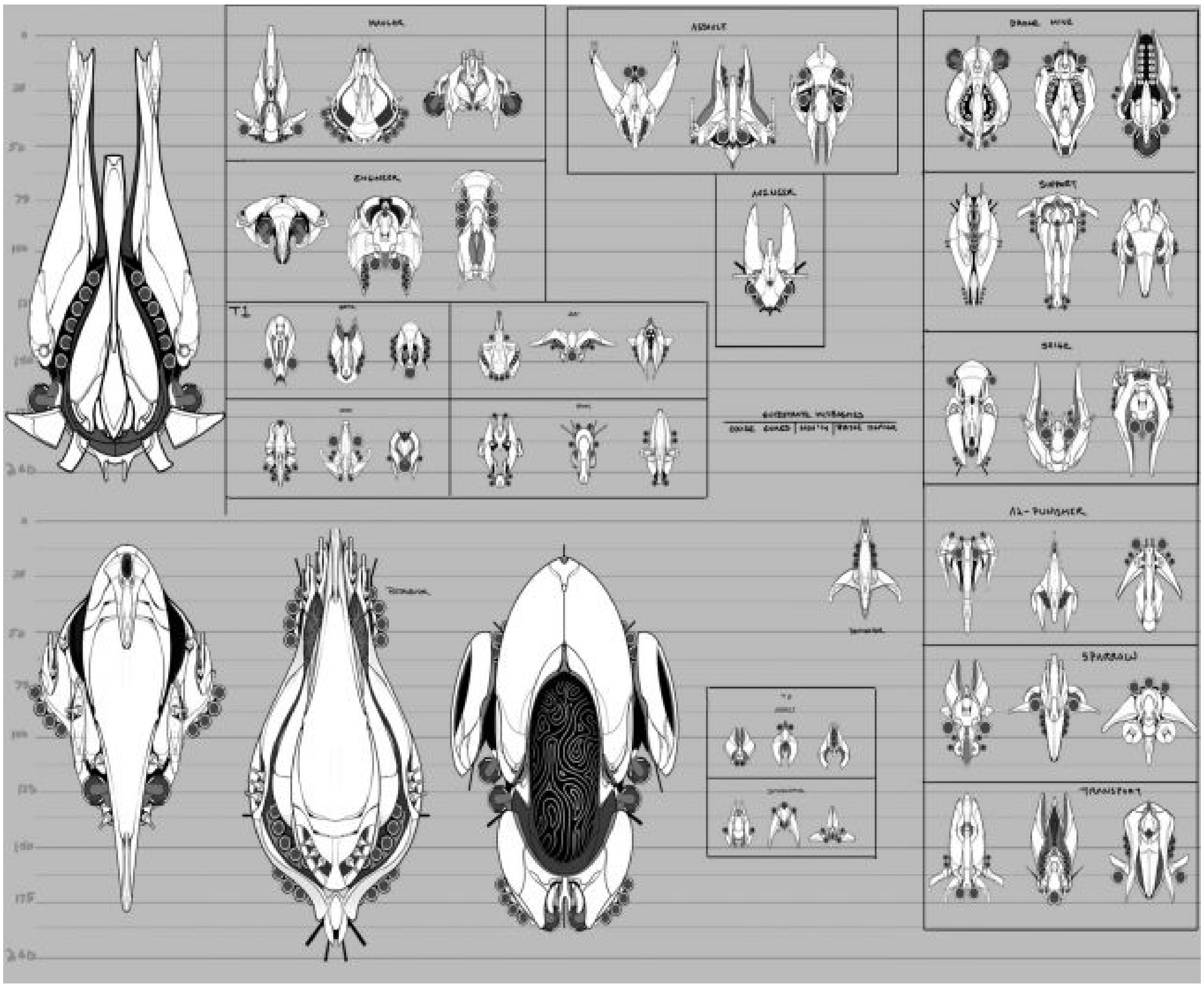
Speed control buttons. Moved current speed next to them.

Production tab. Largely unchanged from existing, just moved to bottom.

Idle factory / Engineer buttons. Count is how many are idle. Click to cycle through idles.

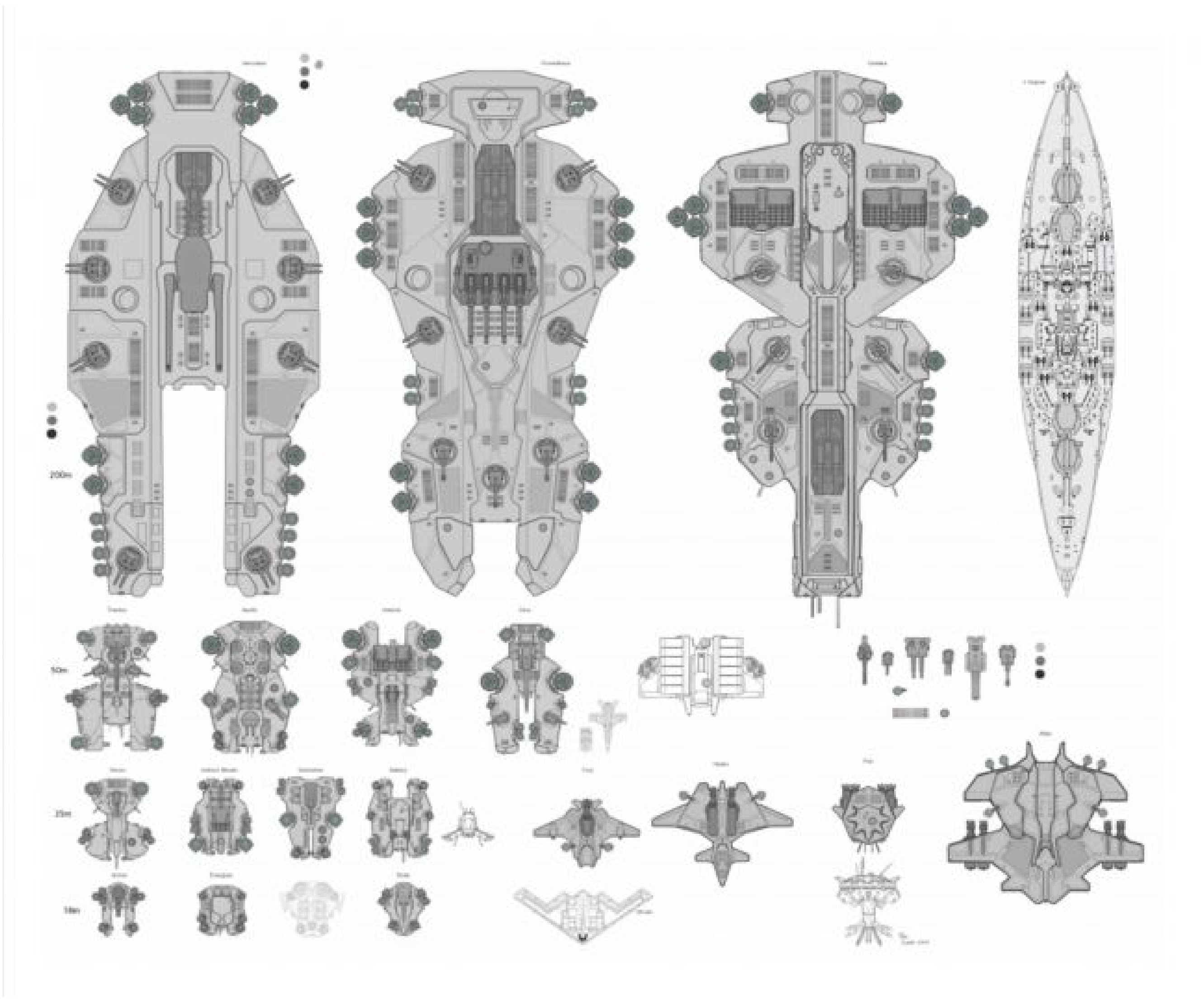
Ignore all bottom bar changes. Those will be discussed elsewhere (Steve L leading).

More UI feedback.

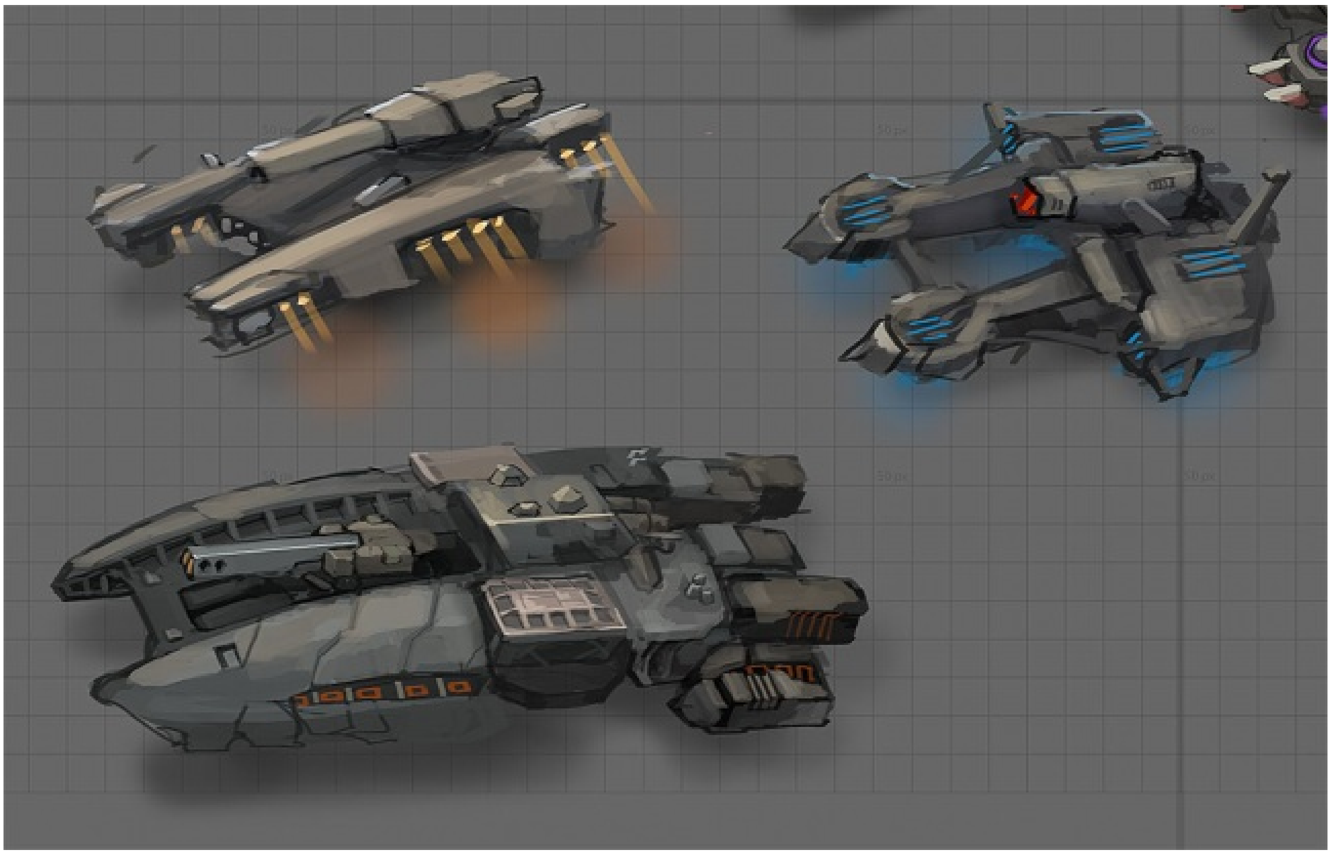


Early Substrate unit matrix. Budget constraints and time for balancing forced us to reduce this count.

We



debated over scale. How big should the units be relative to other things?



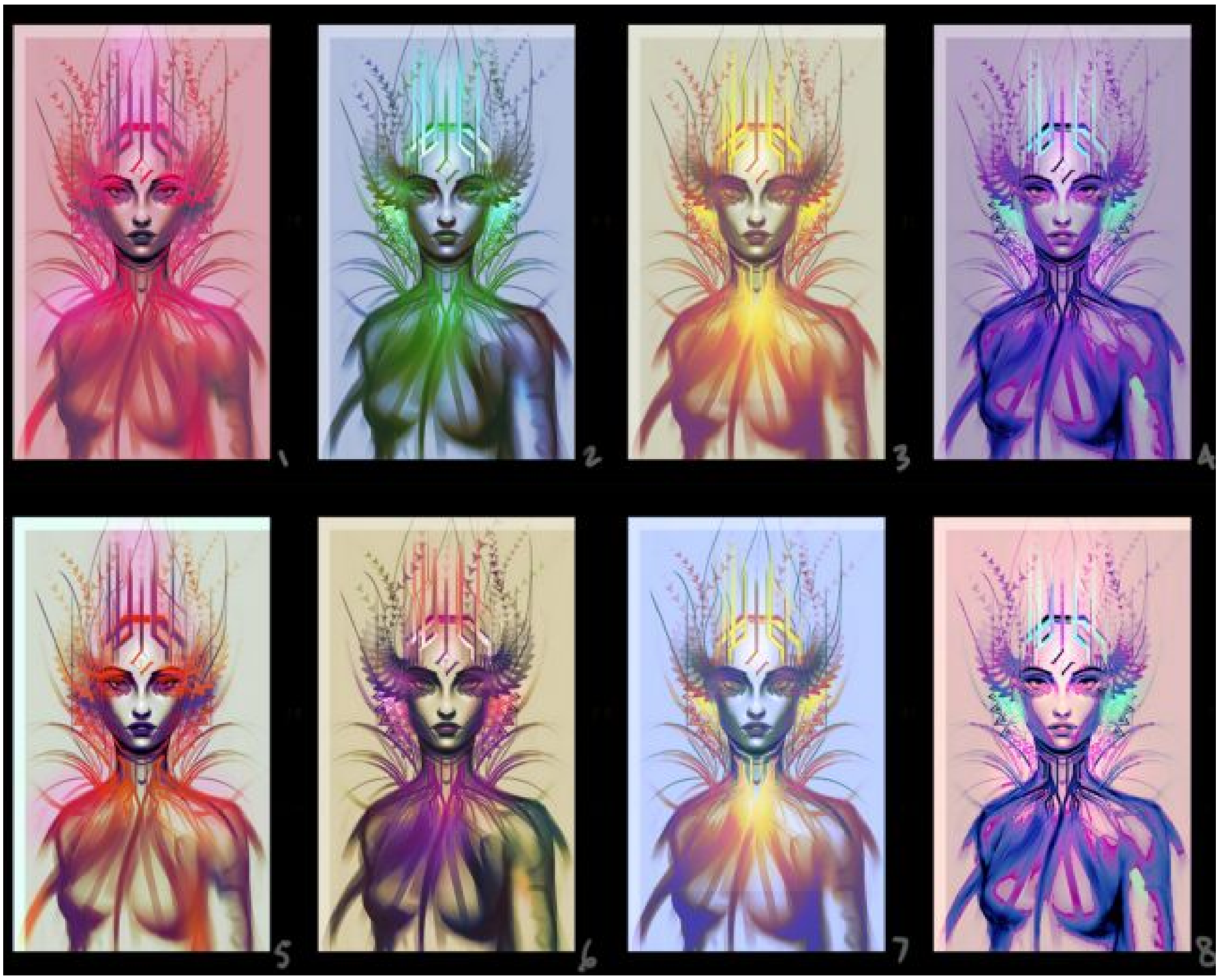
Another set of style concepts.

We

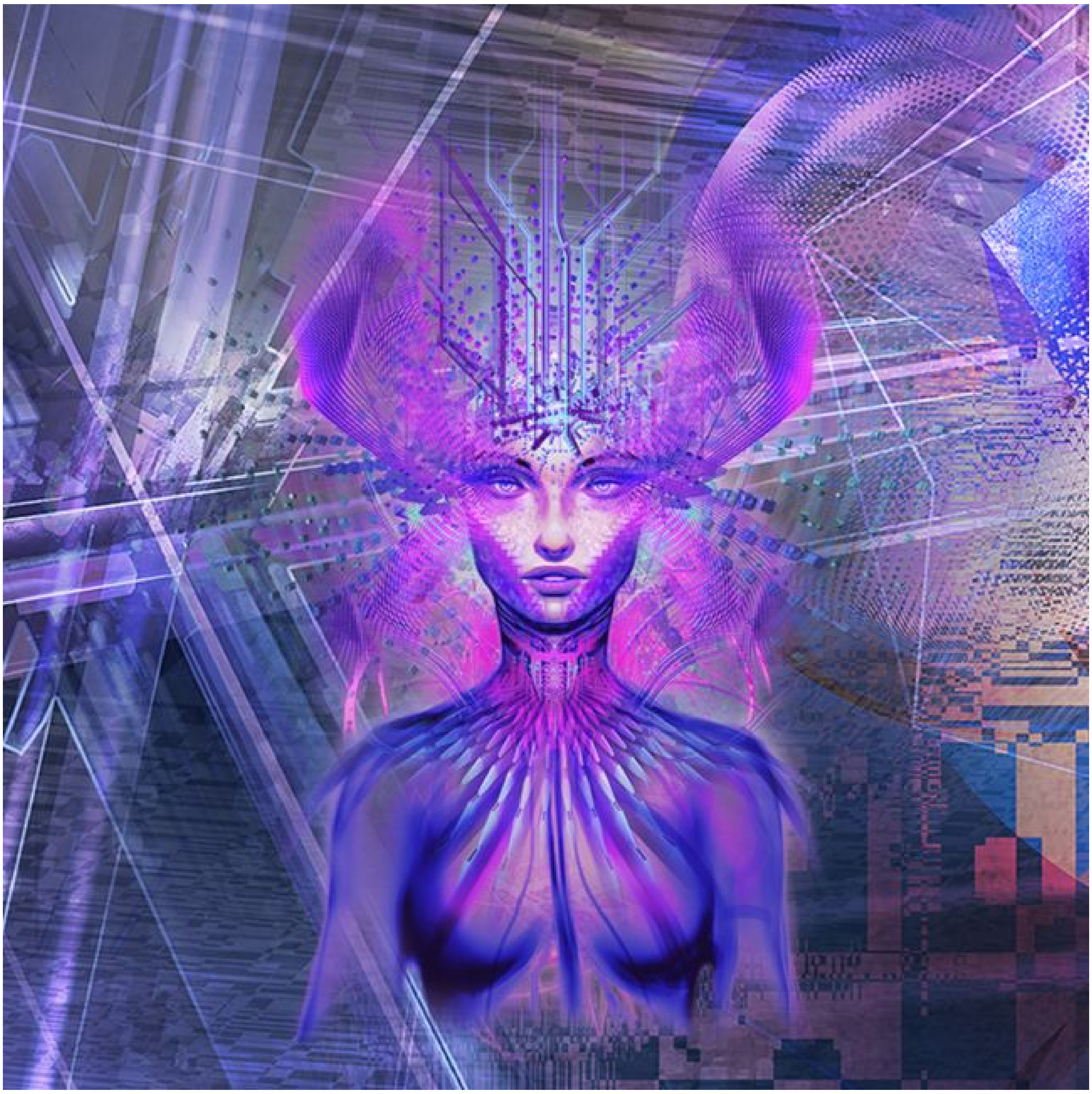
struggled for ways to show players the way the Metaverse that Post-Humans lived in looked like. Ultimately, we had to abandon showing it.

The

campaign character of Mac was originally intended to be visually represented in the game. Unfortunately, we couldn't quite pull off the quality bar we felt necessary to include him.



Similar concepts for Haalee that didn’t make it in.



Another Haalee concept as she was to appear in the game. Very late into development, the campaign cut-scene that had the player confront their opponent looked like this.