JUMP POINT

ISSUE: 06 05

IN THIS ISSUE →→→

Editor: David Ladyman Roving Correspondent: Ben Lesnick
Layout: Michael Alder

FROM THE COCKPIT

GREETINGS, CITIZENS!

Five and a half years. 66 issues. It's about time for a break. The first five years I pretty much pulled the pieces of **Jump Point** together on my own, with most of it written by Ben Lesnick and Dave Haddock, adding in Will Weissbaum and then Adam Wieser along the way. Since the beginning of this year, design and layout have been ably handled by the expanded UK Marketing team. And now, as one of the newest members of the writing team, Ben is taking over the editorial reins, and I can't think of a better person for the job. Dave, Will and Adam will also continue writing, with the quality that we have all come to expect.

I have been a freelance contractor on **Jump Point** and on *Star Citizen*, even during the time I was part of the Design team. This has been the longest continuous gig I've undertaken in my 20+ years of freelance work, and it's been one of the most enjoyable. I've met a lot of dedicated, talented people, all focused on creating the game that everyone wants it to be. And even though I'm stepping down from **JP**, I may be involved in other CIG projects in the future.

Meanwhile, there are far too many people to thank and to recognize, but I do want to mention a few more. First, Chris Roberts, for calling on me in the middle of the night in December 2012, two days before the first deadline. Add everyone at CIG ATX for making me welcome, even though I wasn't officially staff — Eric Peterson, John Erskine, Mike Jones, Chris Graves, Nannetta Wade and Michele Reshetnik — plus all the guys and gals I have had the privilege to work with: Rob Irving, Pete Mackay, Chris Smith, Jason Cobb, David Swofford, Paul Jones, Justin Binford, Andrew Hesse, Melissa Estrada, and all the QA, Design, Art, Dev and everyone else who is, or has been, part of this grand enterprise.

And while it is stated often enough to be trite, it is also true: thanks to all of you who back and subscribe and believe. It wouldn't be possible . . . you know.

Speaking of dreams, it has been a dream of mine for thirty years to republish my own board game, *Star Traders*. We successfully kickstarted it, but only because Chris, Ben, Clifford "Miku" Zernicek, and a whole lot of other staff and CIG backers threw your weight behind it. Thank you!

And finally, a note about names. From the beginning, each **Jump Point** issue has had a name along with a serial number. The very first one, I had nothing to do with: the Mayan End of the World Edition. However, that established a precedent, and a very wide range of possibilities. In general, I've named each issue on a bit of wordplay based on the articles in the issue. This month, no wordplay, simply aspiration.

Excelsior means "Onward and Upward." It has been a personal motto for most of my life — I have aimed to continually do better, both professionally and personally. And now I add my own aspirations to those of everyone else for *Star Citizen*: may it continue to grow onward and upward and fulfill the ambition of *Excelsior*!

David

David.Ladyman@cloudimperiumgames.com
Ben@cloudimperiumgames.com

OPTIMIZATION

"Optimizing" sounds like a good thing, doesn't it? But what is it? This month we chat with the four of the CIG team, scattered throughout the company, who take the lead on optimizing the game.

BEGIN TRANSMISSION →

JP: *Let's start with introductions. Could each of you please give your title, and a short description of what you do?*

CLIVE JOHNSON, LEAD NETWORK PROGRAMMER: I manage our network programming team; assigning tasks, designing solutions to network problems, working with other leads. Occasionally I even do some programming.

ROB JOHNSON, LEAD GAMEPLAY PROGRAMMER: I manage a team of ten other gameplay programmers who work on various game systems. Also sometimes do some coding too and try to help look after the game code in general.

CHRISTOPHER BOLTE, PRINCIPLE ENGINE PROGRAMMER: Kinda involved in everything, with the focus on optimization, from tools to collecting performance data over platform abstractions, to writing engine parts — like the zone system, p4k, tag system, the threading systems — which are used by other programmers for efficient code.

MARK ABENT, LEAD GAMEPLAY PROGRAMMER: I work on systems which directly impact the mechanics of ships. This ranges from the items operating within the ship (power planet, power system, heat system, UI links) to utilizing engine tech to build complex systems which describe a ship: zones / object containers — pretty much whatever Bolte / Clive / Steven / Chris R can provide :D. I also host Bug Smashers. Bugs are fun.

JP: *In a minute, I'm going to ask what exactly optimization is, but let's talk about this first: How did you get to where you are today? Did your schooling help you get started? What did you do before you began with CIG?*

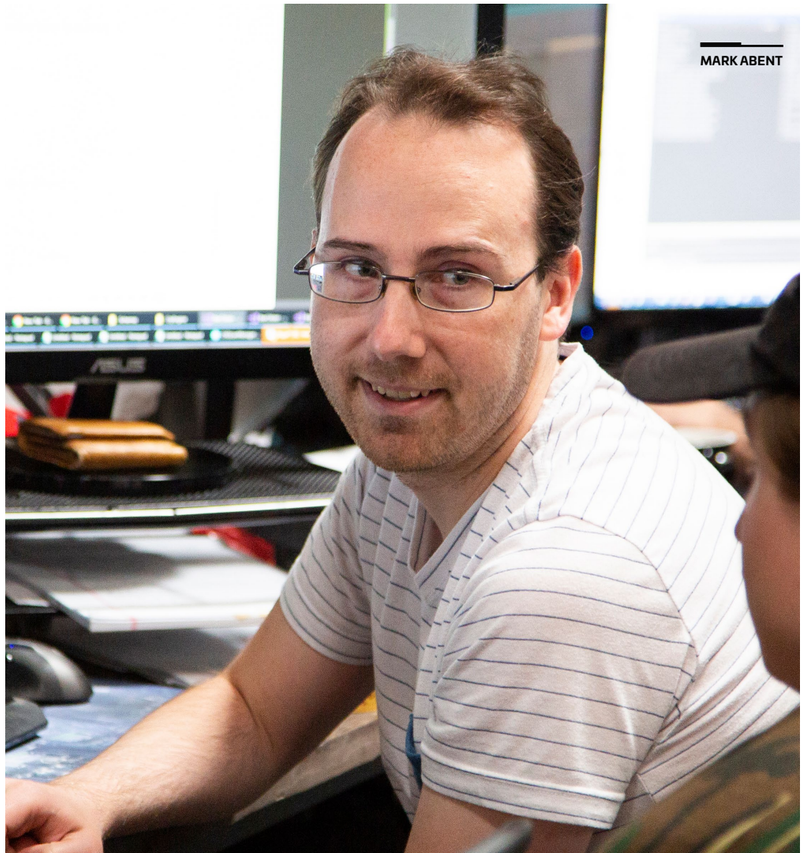
MARK A: I started programming in my teen days, modifying a game called *Half-Life*. It pretty much got me into the programming scene. I joined a project called 'Black Mesa,' which took on remaking the original *Half-Life* game. This opened big opportunities such as getting hired for my first professional gig on a title called *Insurgency*. I stayed on the project through release, until I saw the kickstarter for *Star Citizen*. I wanted to join immediately after learning it was a spiritual successor to games like *Freelancer* and *Wing Commander*. I was hired in 2013 to start work on *Arena Commander*, and have loved it ever since.

ROB J: Before CIG I worked as a lead programmer on some of the LEGO games made at TT-Fusion, which is how I met Erin and a few of the


CLIVE JOHNSON


CHRISTOPER BOLTE


ROB JOHNSON


MARK ABENT

other guys here. Previous to that I did mobile phone games back in the good old days of the Nokia 3210, that was mainly porting old SEGA games like *Sonic*, which was a cool way to get started in games. I got that job after finishing a degree in Engineering Maths with Computer Science down in Bristol.

CHRISTOPHER B: I started programming when I was roughly ten years old on a C64, trying to do my own games. All in Basic, but I tried to learn Assembly and failed. :D

Later I studied computer science, in which I got a masters degree. During my studies I began an internship at Crytek. My primary focus was making particles and animation updates run in parallel and on SPU for the PS3 version of the GCG09 Demo (the first time showing CryEngine on PS3 in public).

Afterwards I worked full time at Crytek, on *Crysis 2*, *Crysis 3*, *Crysis* on Console and *Ryse*, always focusing on optimization and parallelization of code for consoles.

After seven years with Crytek, I started looking for new adventures and ended up with Cloud Imperium Games, where I've worked on the mentioned low-level systems for the last three years towards a fluent gameplay at our planned scale.

CLIVE J: I have a degree in Electronics Engineering and masters in Software Engineering, and started programming on my Commodore 64 something like 35 years ago. 22 years ago I got my first job programming games and in that time I've moved around different studios, both in the UK and Canada, and worked in several programming disciplines: gameplay, systems, animation, physics and networking. I even ran my own company at one point. Of all the roles I've had, networking is definitely my favorite, as it brings to bear a lot of the knowledge and experience I've learned along the way — even stuff I never thought would be useful.

JP: *This is a discussion of optimization, but only one of you has mentioned it in what you do. What exactly is optimization, and how do you do it? Let's start with a very basic, short description, and then we can expand on that.*

CLIVE J: Making stuff more optimal. ;)

CHRISTOPHER B: As Clive said, optimization is the process of making operations more optimal. :) In other words, it is always about figuring out how to get the same (or a reasonably similar) effect with less instruction.

Sometimes it is math optimizations, sometimes algorithmic, and often it is tinkering to find the trade-off between how much we need to update, and up to which quality.

ROB J: (^^ i.e., make things go faster.)

MARK A: Optimization from a gameplay programmer standpoint is trying to achieve some gameplay mechanic requested by Design without degrading the frame-rate. We tend to request tools/systems from the engine team to fit our needs, so that we may be able to satisfy the gameplay requirements within reason.

CLIVE J: Any computer has limitations. They only have so much memory and their CPUs and GPUs only run so fast. Optimization is the process of getting the most you can out of your game while staying within those limits.

ORDERING TIMERS TO TRIGGER IN SEQUENCE REDUCES PROCESSING TIME.

KEEPING APPROPRIATE DATA ON THE CLIENT SIDE FREED UP PROCESSING TIME ON THE SERVER.

JP: *When you arrive at work in the morning, do you tell yourself, "today I'm going to optimize XXX" ?*

ROB J: I usually get those thoughts haunting me at 3:00 in the morning.

Unfortunately, when I wake up I've forgotten how I was going to do it and have to start again.

MARK A: We try not to keep Bolte and Clive up at night as much as possible. :D

CHRISTOPHER B: I actually sometimes think about optimization in the morning (or mostly all day long in the background), as those are very interesting problems on the engine level, where you have to think about various CPU and memory tradeoffs, also how others will use the system, etc. Which is work I really enjoy, as it is kind of a mathematical puzzle. The downside is, I don't see much of our game, as I look at numbers all day. :)

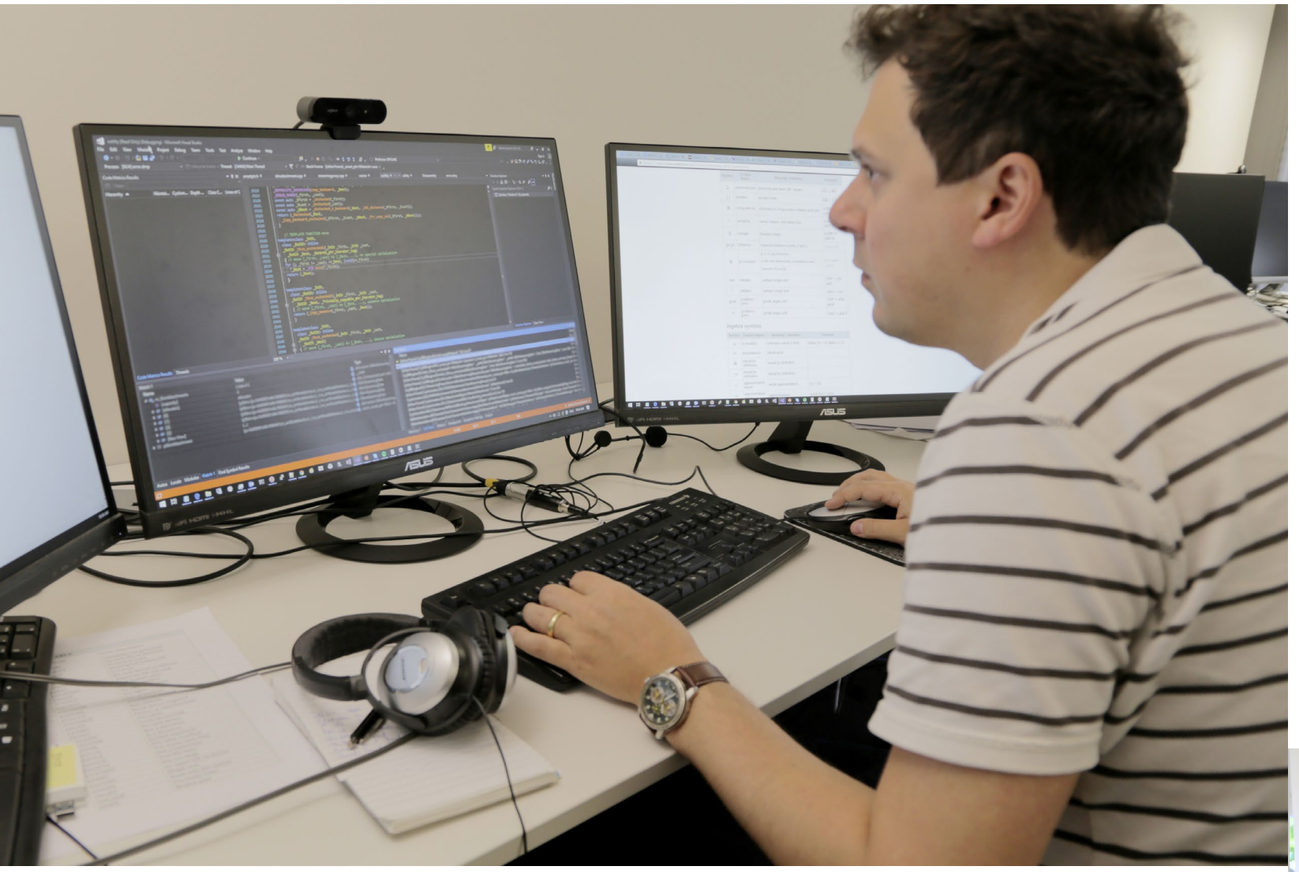
The other problem is, some/most optimizations are not small changes, and can depend on other changes. Or I need to talk to others to get the intention, to provide optimizations which make sense for them. Hence it mostly takes longer than a morning to do meaningful optimizations.

CLIVE J: When you optimize something, you are trying to make it use less of the computer's finite, and therefore valuable, resources. To do this you must first find out which resources it is using most, then try to find ways of making it use less. It's not a mechanical process, like following a recipe; it's much more creative than that. You have to invent new ways of doing something.

MARK A: Here's a simple example. Pretend you have a few dozen timers which send off an event or notification after the timer expires. Let's also assume each timer takes 1 second to process. The simple thing would be to run through every timer every frame to check if the timer is finished. But this can be quite expensive when you have a lot of timers. Now you could order those timers in descending order so the first timer will trigger before the second and so on. You then only need to check the first few timers rather than the whole list, as you know the timers past a certain point won't need to be triggered at this exact time. Of course this won't be the solution for every problem, but it's an example of what you can do to optimize a particular problem.

JP: *How do you choose what needs to be optimized? And how do you decide how to prioritize what you need to do?*

ROB J: We capture some profiling data using a tool such as Visual Studio or RAD telemetry. Data will show how much time is being spent in various areas of the code. We order that from the most expensive,



and work down the list trying to figure out where we can make potential savings.

Some savings are far easier to make than others, i.e., maybe in some cases we're running something on the server that can be client only. Whereas in some cases an optimization could take months/years, say if we need to refactor 200 old Lua entity classes to C++ for example.

CHRISTOPHER B: Programmers are lazy, so we want to optimize the minimum necessary. :)

For this we tend to measure with tools how long certain operations (or a whole frame) takes. We have tools to collect such performance data from clients and servers. Then a lead (sometimes me, sometimes Rob, or Carsten, our Technical Director) looks over them, and we assign out tasks, prioritized based on the impact to the game. For example, if one function takes 10 ms each frame, we prioritize it over a 5 ms function that only happens at some frames.

Other times the code owners sit together and discuss how to structure the code from a high level to make it more efficient. Those are the optimizations that give a lot of performance, but take a long time. Some examples are to change the game code to components, for more efficient multithreading, or to multithread the rendering, or network bind culling. On the other hand, such optimizations can yield massive improvements, which isn't possible with small fast local changes.

In the end, it is all data-based. We measure, improve, measure again till we are happy (or throw changes away if we are not happy, as unfortunately not all optimizations play out how theorized before).

Then, at least for my case, I need to also prioritize based on how many other programmers can do optimizations if I build a certain feature.

For example, if I need to do feature XYZ now for the entity component scheduler (a system to manage the updates of parts of the game logic), adding new features there won't make the game faster, but they allow other programmers to make the game faster.

ROB J: So the time it takes to actually implement the optimization can have a bearing too. We basically would look for good value optimizations for the time we spend on them.

MARK A: Usually when we get the list, we will look over the top worst performers. If we can make a simple resolution to adjust the algorithm with a much faster approach, we will. Sometimes you are lucky and you can spend a few minutes to a few days on some adjustments.

Other times, you have to refactor an entire system due to the nature of the design. A legacy system designed to work on a single thread doesn't scale too well on a multi-threaded infrastructure.

We also take what we have learned from adjusting these old systems, so newer systems work with the newer engine infrastructure.

JP: *I realize that sometimes solutions don't work, but has an optimization ever backfired on you, so that it saved time in the way intended, but overall it added time to the process?*

CHRISTOPHER B: More than often, which is why we measure it and throw it away when it doesn't fulfill the hopes. Which is of course easier said for small optimizations, and not large optimizations. Too often it is a tradeoff, especially if the systems get more complex. And optimizing one part can make another part slower.

But even then, the work should have resulted in a better understanding of the problem, which should make the next optimization approach more likely to succeed. (Another way to say experience and failing helps in the long term. :))

Another possibility is to have too many restricted assumptions during optimizations, which works fine for a time, but then Design have more requirements, and adding such features to the optimized system makes it slower than before. Which is kinda why optimizations are done late at the project, when all requirements are unlikely to change.

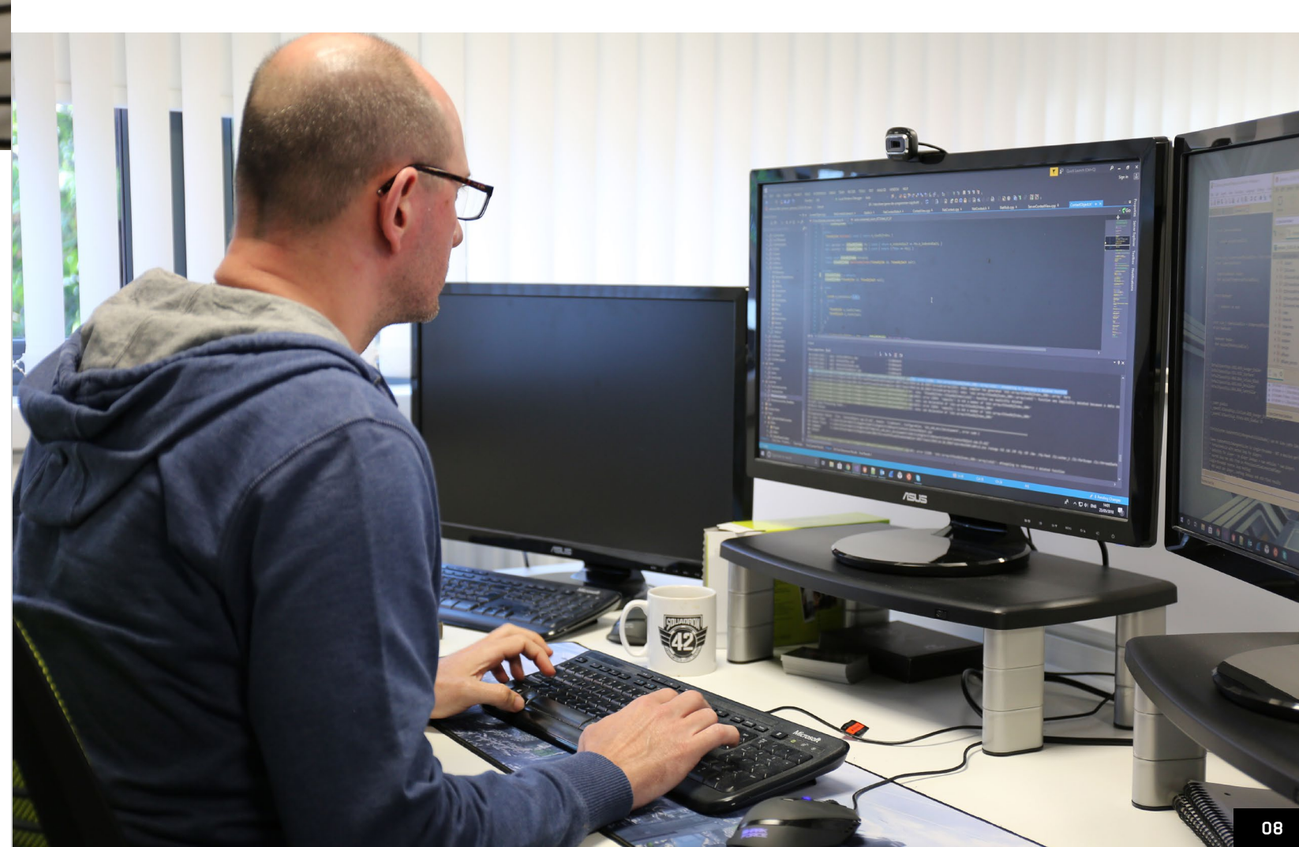
MARK A: Yes. One particular situation that comes to mind is IFCS. Originally, we had it running on the physics thread, but it caused some performance problems due to the way it interacted with the main thread and physics. It ran one IFCS update in sequence. It's a very intense system to calculate thruster locations on a ship to apply impulse on a rigid physics entity based on player input and item states (lower power, destroyed items, etc.). The optimization route we took was to move the calculations from physics/main thread link to a component job process which can run in parallel with other IFCS operations. We gained quite a bit of performance, but IFCS suffered as a result due to de-coupling it from physics, as it used the physics step to apply impulse at the appropriate time of the physics step. We had to take another pass to perform the calculations in the job, then apply the impulse in the physics step. It was a bumpy road, but it brought us closer to what we originally had with the improved performance.

I should note, this work was done by John Pritchett.

ROB J: Also more commonly, you find optimizations can lead to bugs. It could be we removed code that we thought wasn't needed, but it was, or in re-writing code we make the game faster but introduce bugs.

MARK A: It's even more fun with order changes and new bugs are discovered.

CLIVE J: Optimizations can sometimes backfire in the sense that they restrict your options later on. You'll optimize something by exploiting the fact that certain situations can never happen. They could happen but the design is such that they won't. Later the design will change a bit and now the exact things that were never supposed to happen will happen all the time. Not only does your optimization no longer work, it's now actively blocking the game being developed. This is one of



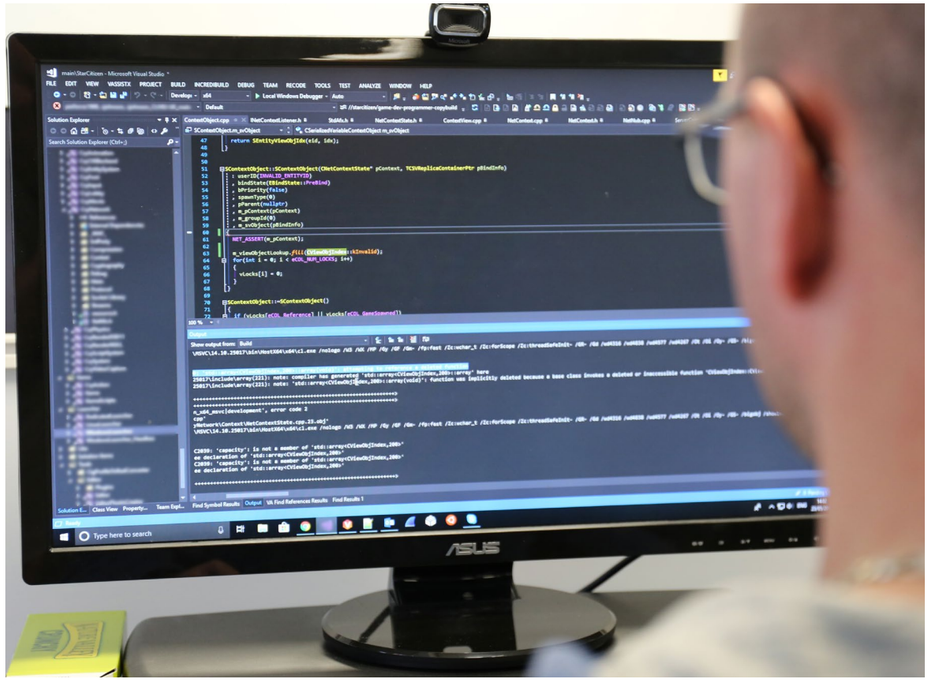
the reasons optimization of a system is usually done only after the design has been locked down and carved in stone.

JP: *I know bugs and optimizations are different things, but do you approach them differently? Is the thought process different between killing a bug and creating an optimization?*

CHRISTOPHER B: In the end, not much different. Both are about understanding the problem, thinking it through and adjusting the logic into the desired end result. (The correct algorithm, or faster execution, preferably both at the same time.)

ROB J: I think optimizations often can require more creative coding than bug fixes. There's usually one or two obvious ways to fix a bug, however with optimizations you may need to look at both high and low levels, i.e. look at the bigger picture as well as just the code in question. For example, we may need a totally new system or a big change to an existing one to get a good win. Or maybe we need to refactor something to parallelize it over multiple cores so we can process more code at once. Or maybe we find a few lines of code that we can fix for a good win. You often can't predict this until you dig into the various areas that show up as expensive in the profile data.

CLIVE J: I think they're quite different. When killing a bug you start by going into full detective mode and trying to figure out where in the code the problem is and why it is happening. This is where the creativity happens. Sometimes you have to get quite inventive to get to the root of a problem and fully understand it. Once you've found the problem, fixing it is usually pretty straightforward. With optimization, the creativity is at the end of the process; you're devising a solution, not the means to track the problem down.



JP: *You say optimizations are done late in the project, but aren't you optimizing pretty much all along the way? What is the difference between figuring out how to do something as fast as possible (initial design) and figuring out how to do something even faster (after the design is locked down)?*

ROB J: We try hard to write optimal code as we go. We test it locally before submitting and review each other's code before it gets submitted. However, with the scope of *Star Citizen* it's hard to test all use cases before code goes in, and also predict how others will use your code in future. In future we aim to have more automated tests to catch issues before they get submitted into the game. However whilst we build that framework up, we still need to post-optimize code that's already active in the game.

CHRISTOPHER B: I would say it's based on necessity. Don't forget that everyone internally also needs to run and play the game (QA tests it, designers edit the live levels, etc.), so we need a reasonable performance. Then we use this data to drive the architectural decisions (and requests from Design), as those optimizations take a long time.

As we cannot look into the future, we can only operate on a best guess to ensure we don't code ourself into a corner (by too strict optimization), while providing the necessary performance. Also, sometimes optimization can restrict design, as not everything designers wish for can be computed in real time.

Mostly we try to estimate the cost based on scale, which is called BIG O-Notation, to get an idea how many operations we need for something. (For example, doing some operation each frame for each existing object most likely won't work, which is at the design stage.) Later we look into more lower level details, like threading support, optimize the core utilization, memory/cache efficiency and so on.

CLIVE J: What I meant to say is that optimizations on a system are done late in the development of that system, not the game as a whole. When we get a task to work on,

we go through several stages of development: design, implement, debug, optimize. So we're optimizing each piece we work on as it is developed. Quite often though, you won't get realistic loads placed on your system until it gets out into the wild, and that may mean a follow-up optimization pass. And of course sometimes as other systems get added, older systems creak under the strain and need to be replaced, so more optimization happens. It's true that as the game gets close to being finished we'll probably go on a massive optimization sweep, but we try not to save all that work to the end.

MARK A: We may get an initial design, which we build out to spec with optimizations in mind to the day. However, since we work in agile/sprint fashion, the iterations we build on the initial design may require changes to the design and/or system itself. If you aggressively optimize too early, then it may make iteration difficult to the point of requiring an entire refactor. Lost time. On the other hand, if you optimize to a degree with iteration in mind, then when the future iterations come it's not as big of an impact to adjust the system. In theory, if you keep this flow going, then by the time the system is fully fleshed out you can start doing more aggressive optimizations with the understanding later iterations will take more time.

JP: *Let's talk specific examples for a few minutes. What's the most frustrating situation you've encountered while optimizing, either finally resolved or still unresolved?*

CHRISTOPHER B: Regressions. The game has way too many moving parts to oversee them all. And sometimes it feels like not looking at something will result in it getting slower/using more memory. But this is just the reality of the complex distributed development. Everyone has their tasks, and no one oversees all parts and metrics which can be affected. Which is why we are working towards a more automated system like Rob mentioned, so that we have a constant more detailed look at all parts.

CLIVE J: The Zero Sum Game: you spend a lot of time working on an optimization, only to find the designers have added a ton more entities to the map and you end with performance the same or worse than when you started. That's happened a few times to us on the networking side.

JP: *So you've improved the game (it's much better than it would be otherwise), but no one can tell the difference?*

CLIVE J: Yeah. :(

ROB J: +1 for Clive's answer. For me, the most frustrating thing can be making a saving in frame time, only for that to be negated by other people's changes. This is why we in the longer term need to keep working on ways to catch de-optimal code or setups before they make it into the game.

JP: *I gotta say, that would be real frustrating for me, too. "What have you done all this month?" "Made the game just as fast as it originally was."*

CHRISTOPHER B: 100% agree with Clive. Sometimes it takes a designer a few clicks to take up all the free performance budget a developer has spent weeks on. (Not blaming designers — this just goes into the many metrics and no one being able to oversee each metric.)

CLIVE J: Oh, I blame the designers.
For everything. ;)

CHRISTOPHER B: This goes into optimizing at the end of the cycle, where everyone focuses on optimizations, and not 50% are doing optimizations, and 50% adding new features/content which eat up the optimizations.
What a world: a game without designers, only programmers, perfectly optimized, but the most ugly graphics and zero fun. :D

JP: *Sounds like cancelled-out optimizations is Frustration #1 with just about everyone. Are there any others?*

ROB J: Other frustrations include optimizing code that has no longterm future, however sometimes needs must. The longer we work on this codebase though, the less we need to do this, so hopefully the future is brighter in that sense.

CHRISTOPHER B: Time pressure can also be a frustration. As Clive said, optimization is like a clever math puzzle. Now imagine taking a math test eight hours a day for weeks, to hit the performance goals for a release. (And it's a math test where you don't even know for sure that an answer to the questions exists.)
And to make it more fun, if you mess up (highly likely in complex optimizations), you are rewarded with a bug you need to fix in addition to the optimization (and the bug fix could even erase our previous optimization).

CLIVE J: I guess my second biggest frustration would be that the amount of time spent on an optimization is no guarantee of how much it is going to save. You can have a great idea that on paper seems like it should save you tons of CPU, or memory, or bandwidth. It's going to take a while to do but it'll be worth it. Or so you think. Once you've done all the work, it turns out it doesn't actually save you that much. That's just part of the "fun," as Chris puts it.

ROB J: Another frustration for us has been finding all the cases where performance nose-dives. There have been times when we haven't been able to fully catch all these cases until we opened the game up to a wide PTU, which is usually only a week or so before we want to release live, which leads to some fun last-minute fixes!
And then there was the time when performance dropped because we gave everyone a gigantic Reclaimer to test on PTU at the same time ... :O

JP: *I know how to optimize that situation! Let's look at the other side.*

BIND CULLING RESTRICTS WHICH ENTITIES EACH CLIENT NEEDS TO KNOW ABOUT AND SHOULD IMPROVE CLIENTS' PERFORMANCE.

What optimization(s) are you most proud of?

CLIVE J: All of them. Every win is a win, and as I said on RTV it's a bit like being asked to pick a favourite child. They all required solving different problems, so each have their own character. Though thinking like this is probably just a sign that I need to get out more.

CHRISTOPHER B: Nothing in partial, mostly the whole subsystems. The p4k system performs nicely (in regards to build update times and bandwith, but also IO speed). I am mostly happy with the low-level threading (we "just" need to change all high-level code to use it more). The zone system in itself is also pretty well done I think. It works on 64 bit with 1E13 on each dimension, and does real time culling of ~500,000 objects multiple times each frame.

And of course it supports multiple zones. And very efficient object position updates. And multithreading. And portals, and different data types, etc.

But to put it into perspective, at the core, the zone system is a combination of Octress and an AABB tree; you could write both in perhaps 4000 lines of code. But the zone system uses over 40,000 lines of C++ code alone for all our cases and to handle optimizations.

ROB J: I was proud that we got the 3.1.0 patches back to a more reasonable framerate. There's still more to do there, though.

The biggest win I ever got was telling Design to remove some demo locations from the PU map that shouldn't have been there. I enjoyed that, as it was +10 FPS for zero code changes. :)

JP: *To the extent that you can talk about it, what optimization are you working on today? What's at the top of your stack?*

ROB J: Right now the main optimization-based work I'm looking at is converting a lot of old entity types to new component-based entities. This is needed for object container streaming, which will get us faster load times and allow us to have more entities in the map on the client. Plus it also modernizes the code to better use our newer component updating systems, which should also lead to some faster code.

CHRISTOPHER B: As always, multiple things:

- Fixes for ObjectContainer Streaming, which is required for network bind culling
- Working on performance capture tools to more continuously collect performance data
- More features for the Entity Component Scheduler for more efficient game code updates
- Some features for the zone system to cope with rotating better. (When we place stations on a planet, we put them into the planet zone rotated, and rotating enlarges their AABB, resulting in worse culling than in the non-rotated authored source.)

CLIVE J: Bind culling. It's a way of restricting which entities each client needs to know about and then making sure only those entities exist on that client. It should save clients quite a lot of memory, reduce loading times and improve their performance. The server still needs to know about everything, but won't need to send as much stuff across the network, so we should get some nice bandwidth savings too.

JP: *As always, we've taken more time than we planned, but we're just about done. Any last words?*

CHRISTOPHER B: Often said, but never enough: A large thanks to the backers who make this game possible.

CLIVE J: Yes. I'd like to add that optimization isn't just being worked on by a few of us; it's a consideration rarely far from the minds of any developer (even designers ;)). It's definitely a team effort and will take all of us working together to reach the end goal. And of course, as Chris said, a massive thanks to our backers!

MARK A: Shout out to our backers! They are the heart of *Star Citizen*! Time to go smash some bugs!

ROB J: Yes, a huge thanks to our backers. It's great fun working on this game, and we really want to pay you all back with some good performance improvements over the next few patches.

And also thanks to my wife Rachel for putting up with me thinking about optimizations at 3:00 in the morning.

CHRISTOPHER B: Yes! Thanks to my wife Zora, for putting up with all the time thinking and talking about the game. :)

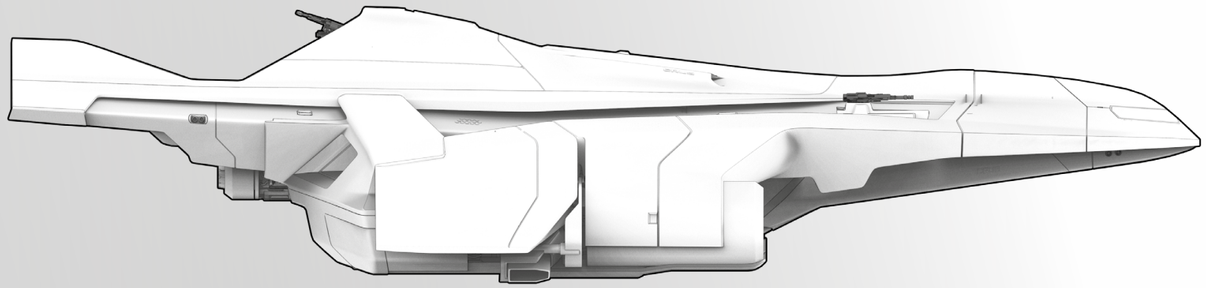
CLIVE J: Thanks also to my wife Leonie for not complaining too much when I'm thinking about work instead of listening to her.

JP: *Thanks, guys — I've enjoyed it all!*

END TRANSMISSION ←

WORK IN PROGRESS... CRUSADER HERCULES STARLIFTER

CRUSADER INDUSTRIES



AIMS

- Ship made to transport ground vehicles to a planet and between planets
- Able to lift at least 2 Tumbril Nova tanks at the same time. No passengers.
- Has 2 variants, military with 1 more turret, and a gunship with half the cargo space but more guns.

AESTHETIC

- Crusader Industries design rework, rotating VTOL thrusters.
- “Cargo plane”-like design with front and rear entry (cf C5 Galaxy).
- Cockpit with a lot of visibility, Cutlass style pilot seat. Solid exterior structure.

	Cargo (C2)	**Military (M2)**	**Gunship (A2)**
Length	94m	94m	94m
Width	70m	70m	70m
Height	23m	23m	23m
Powerplants	1x Large	1x Large	2x Large
Shield	2x Large	2x Large	2x Large
Armour	Medium	Heavy	Heavy
Stations (including Seats)	2	3	8 (inc 5 stations in the rear)
Cargo Capacity	624 SCU	468 SCU	234 SCU
Weapons	2x S4 front fixed ballistic cannons	2x S4 front fixed ballistic cannons	2x S4 front fixed ballistic cannons 2x S5 remote side ballistic cannons
Turrets	2x remote turrets (S5) with 2x S3 guns each	3x remote turrets (S5) with 2x S3 guns each	3x remote turrets (S5) with 2x S3 guns each 2x remote side turrets(S6) with 2x S4 ballistic Gatlings
Missiles			Bomb system (instead of rear ramp) Either 60x S3 bombs or 4x S10 bombs

The vehicle depicted herein is undergoing concept and design as of the release of this publication. Specifications and appearance are subject to revision during development.

KEY CONTRIBUTORS :
LEAD DESIGNER: JOHN CREWE
SHIP DESIGNER: CORENTIN BILLEMONT
ART DIRECTOR: PAUL JONES
ARTISTS: MICHAEL OBERSCHNEIDER & SARAH MCCULLOCH

GENESIS OF THE STARLIFTER

Typically, *Star Citizen*'s game designers focus on two major questions when deciding which ships to add into the game.

Firstly, what will be fun for the player? This thought gave rise to many role-specific ships beyond the traditional set of fighters and bombers typically found in space combat games, such as racers, explorers, and passenger transports. Secondly, what ships are missing from our existing systems and how do players move from one place to another? While this might not be as exciting as wild new design ideas, it's just as important. Players will need, for example, a smaller mining ship to start their career that will ultimately lead to more robust designs like the Orion.

The Hercules Starlifter comes from a rarer third process: it's a ship that needs to exist. Successful early work on basic ground transport like the Ursa Rover and Dragonfly bike created a pipeline and demand for terrestrial vehicles. Meanwhile, the engine team was hard at work on Alpha 3.0, which would premiere massive expanses of ground terrain that could be explored, refined, colonized, and ultimately fought over. These two ideas would quickly feed off each other and inspire the development of a much greater variety of industrial vehicles like the Tumbril Cyclone buggy series and the Nova tank (with further types in development). *Star Citizen* would now have entire planets to explore and a host of cool vehicles that could be used on them... but no easy way to get them there.

Chris Roberts identified this in an early meeting about the Nova tank, realizing that players would absolutely need to move their tanks, buggies, cargo, and future modular base parts from place to place on planetary surfaces and from system to system. After all, what fun is having a fleet of tanks and the ability to visit enormous new worlds if the two can't be brought together? Therefore, the Hercules concept was born. The design team's next job was to figure out exactly what the Hercules was: a small orbital dropship, a giant cargo plane, or something else entirely? Lead Designer John Crewe tasked veteran ship-builder Corentin Billemont with the job. Several conceptual directions were explored and the team settled on a large interstellar ship that could carry a vast amount of cargo or several vehicles.

In addition to these broad strokes, Billemont specified a series of possible cargo configurations that the ship should be built to support: 12 Cyclone buggies, 6 Ursa rovers, 2 Nova tanks, or an unspecified hoard of space bikes. He also noted that the hull should be able to transport a variety of smaller not-jump-capable spacecraft from place to place, including the Argo, P52 Merlin, Razor, M50, and 85X. While this may change in-game with the final production model for balancing reasons, these overall metrics were important for the initial sizing and design.

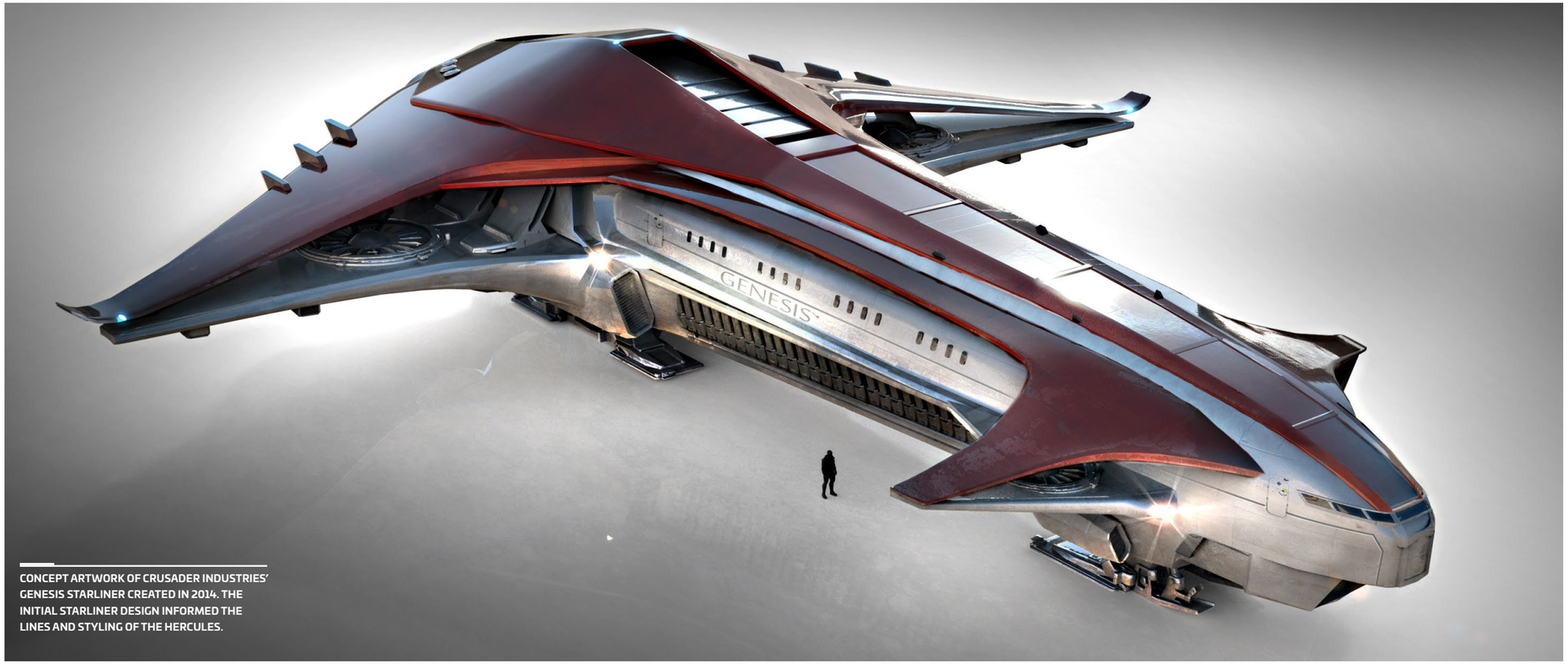
WHAT'S A STARLIFTER, ANYWAY?

With the role and size decided, the next question became 'who in *Star Citizen*'s universe would build a large military transporter?' From the outside, it would seem the answer was always fated to be Crusader Industries. Their Genesis Starliner launched with accompanying lore that mentioned the existence of a "Starlifter" variant:

"The United Empire of Earth military also modifies the standard Genesis spaceframe for a number of different roles, including a troop transport, patrol hunter-killer, SWACS/C&C platform (referred to informally as the Starlifter, Starhunter, and Starseeker) and even as an advanced aerospace prototype testbed! Imperator One, a highly customized Genesis Starliner with added security and luxury features, is used to ferry the sitting Imperator when traveling off-world."

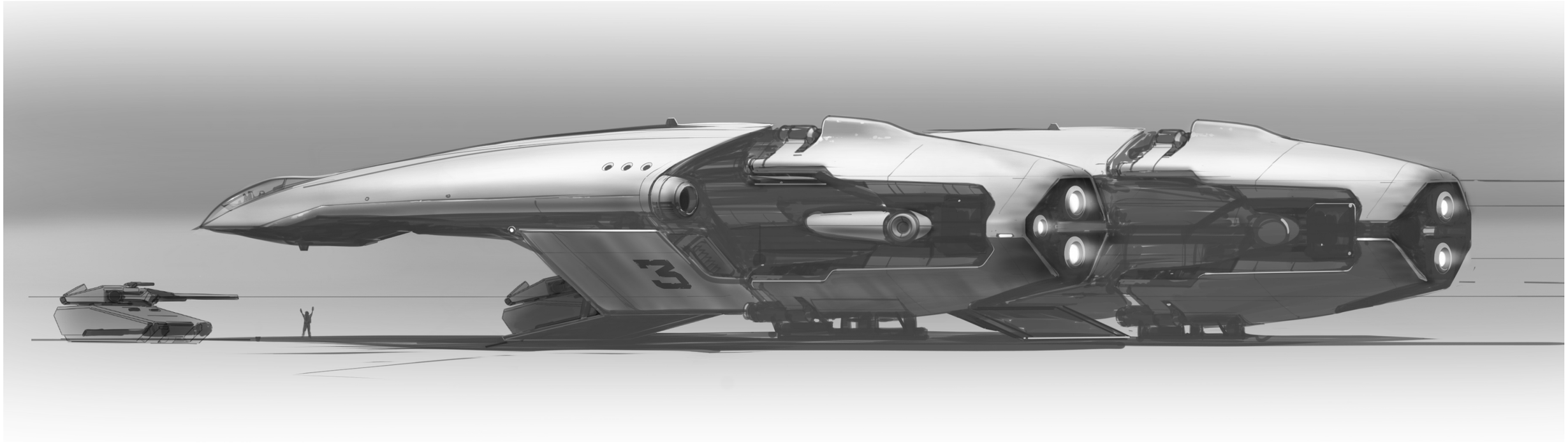
For much of its development, the new cargo ship was referred to only as the Starlifter, before ultimately being named Hercules. Aviation buffs will recognize the Starlifter name immediately: it's a reference to the 'Lockheed C-141 Starlifter', a large military transport that saw service through the latter half of the 20th century. The final name, Hercules, references the famed C-130 Hercules transport plane which continues to serve today, and just like its *Star Citizen* descendant, ranges from a rugged cargo plane to a powerful gunship.

Designers putting together the initial Genesis Starliner design had considered the possibilities of different versions, but instead chose to focus on the passenger transport mechanic.



CONCEPT ARTWORK OF CRUSADER INDUSTRIES' GENESIS STARLINER CREATED IN 2014. THE INITIAL STARLINER DESIGN INFORMED THE LINES AND STYLING OF THE HERCULES.

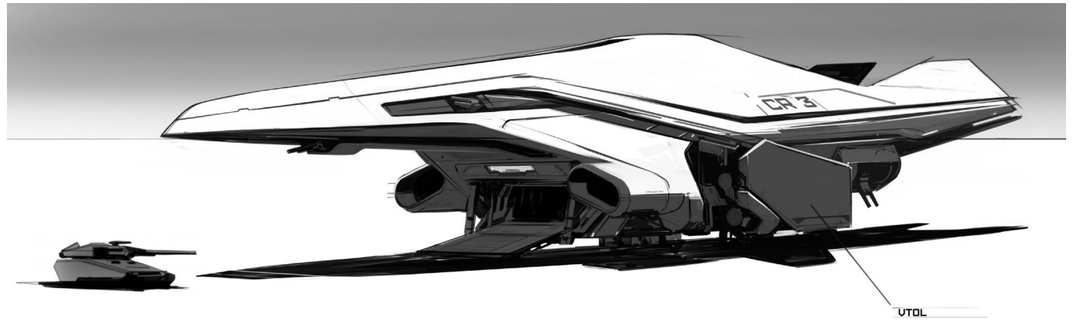
As it was already decided that the Hercules would not be a Genesis variant, but a whole new model, the floor was open to any of the 'verse's military contractors to take ownership. Several different options were considered, and one was even presented to the community as part of the 'Next Drake Ship' poll. The options were hotly debated, but the ultimate decision was to do the hardest thing: create an all-new military spacecraft and properly update and refine the Crusader style guide. This would mean more work in the short term, but provide a better outcome in the end, with the ship team able to put the earlier Genesis Starliner into production as well as lay the groundwork to expand the Crusader fleet much more rapidly. One of the game's more obscure and unloved companies could now become one of the 'verse's major ship developers!

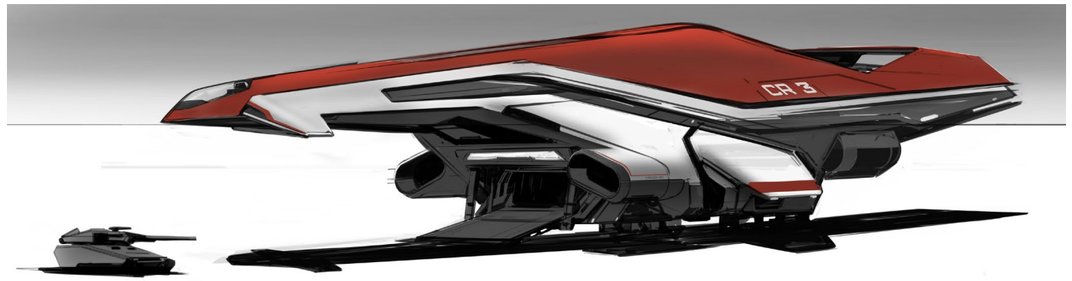


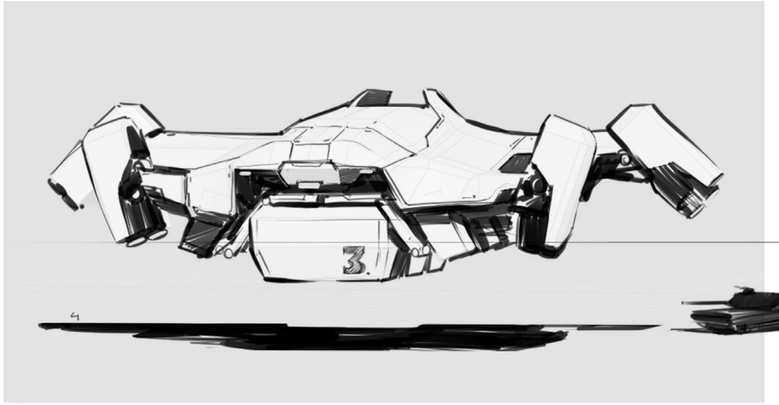
ART DOES SOME HEAVY LIFTING

Art director, Paul Jones, selected Michael Oberschneider, the artist behind the Anvil Hawk as the 'prime contractor' for the Hercules project. The art team had a tough job ahead of them. Unlike previous concepts, there wasn't a great body of work representing Crusader Industries to draw from. The first step was to expand on the work done to the Genesis Starliner, but they would need to go well beyond and finalize a look and feel for the Crusader brand in the process. To deal with this challenge, the Hercules art development process began with a larger-than-usual collection of reference material:

The first of these were obvious choices, massive military planes that included the American C-5 Galaxy and the Russian An-225 Mirya. These designs were particularly useful, as they meld traditional aircraft lines with the ability to store and deploy large ground vehicles - the same intended role for the Hercules. Both the C-5 and An-225 feature noses that swing open to allow vehicles directly into their cargo bays, a concept that would strongly inform the Hercules. The team also looked at a variety of specific features from other planes, including the rotating engines on the V-22 Osprey, and the ball and nose turret housings of Boeing's iconic B-17 Flying Fortresses. However, the team felt strongly that large military aircraft were just one piece of the puzzle; the Hercules would need to mix in bolder touches from unlikely sources to feel truly unique and to best define Crusader Industries.

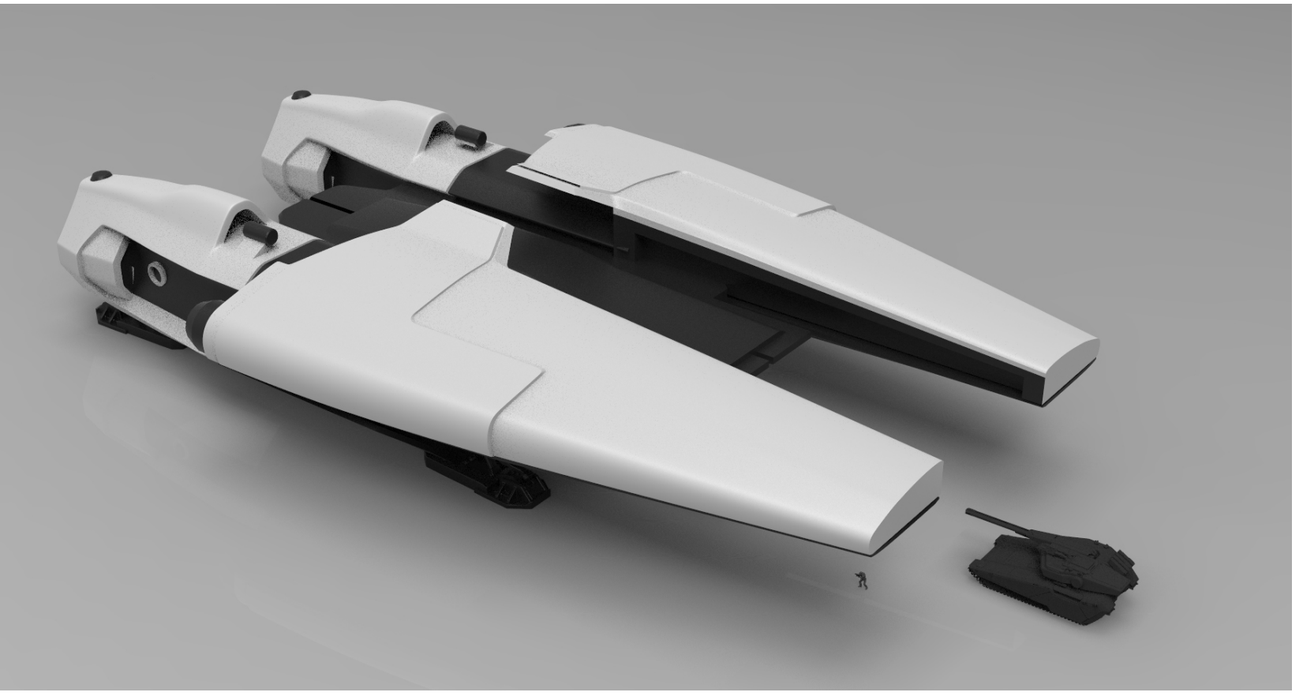


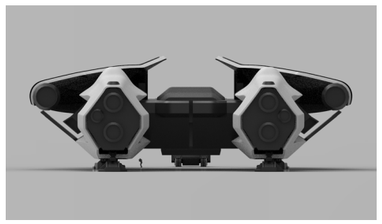


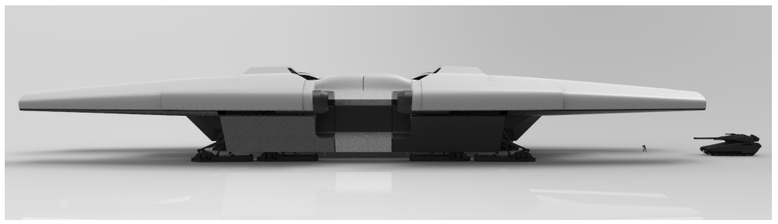


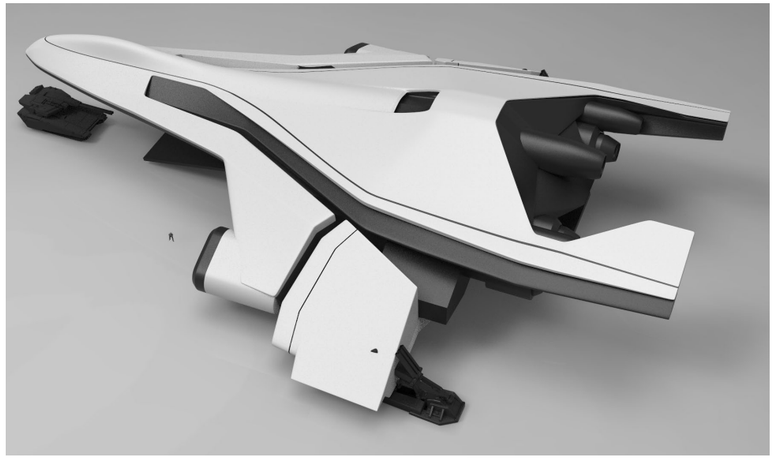
To inspire them, the artists studied rocket launchers, sports cars, and even a horseshoe crab, noting the animal's smooth outer shell that protects its delicate legs and antennae. Art director, Paul Jones, also provided Oberschneider with all the legacy Genesis Starliner art to help decide what could be referenced from Crusader's first ship.

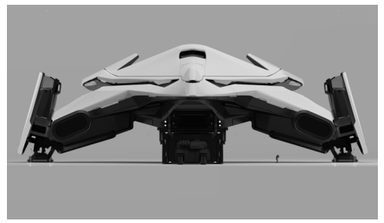
The concept art came next. For the initial hull, Oberschneider began by creating a series of two-dimensional sketches that took the concept in two basic directions. The first leaned more towards a traditional unibody military aircraft inspired by cargo planes, while the second was a high-concept science fiction design with rotating engines and broad flying wings. These were presented to Chris Roberts, who made several selections leaning to the more traditional military look.

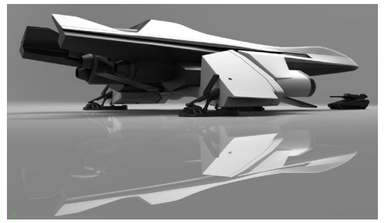


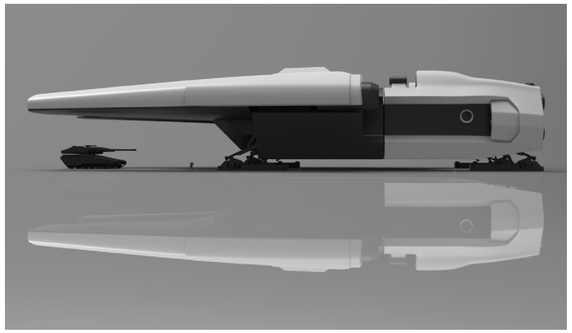


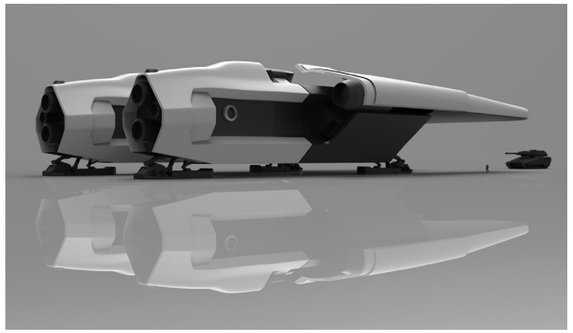






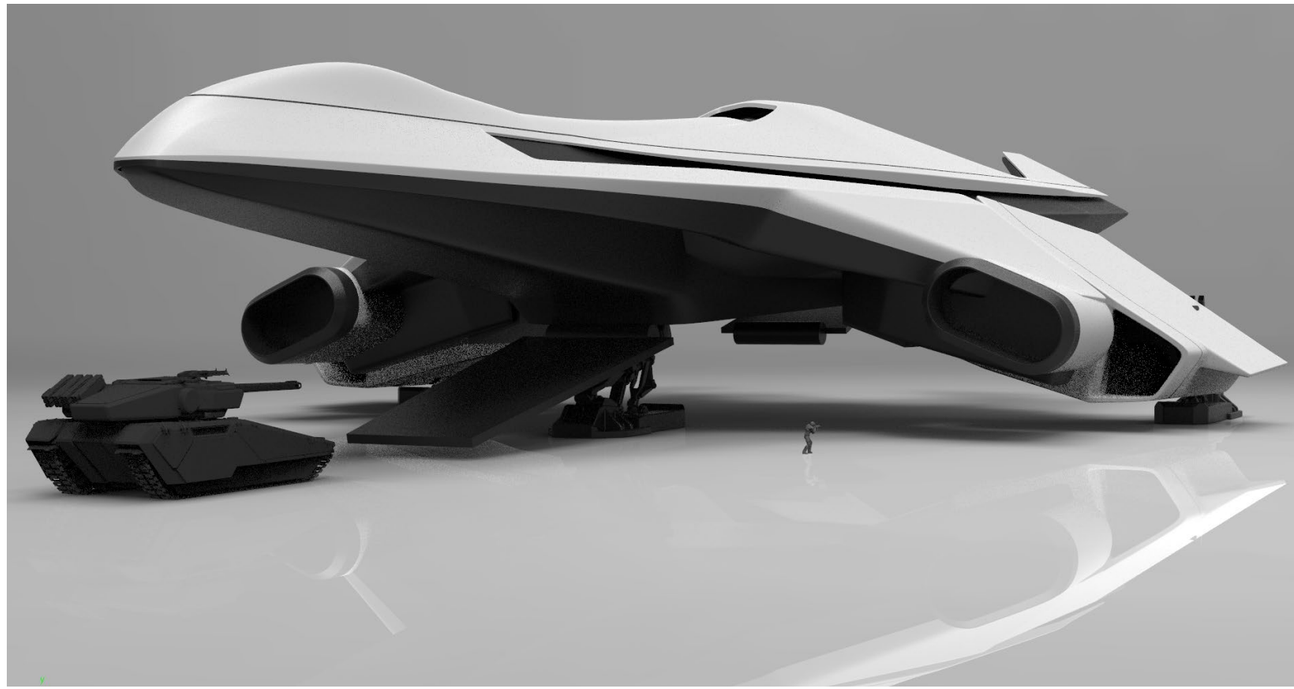






Next, Oberschneider constructed two different 2D models for review, which again led down different paths; one had a very traditional military aircraft feel, while the other was a more futuristic take with a split fuselage. Chris preferred the aircraft-styled ship, and the concept went into the refinement process. At this stage, the artists made it their focus to subtly incorporate some of the design elements of the Genesis Starliner, including nods to the parallel 'double wing' look and enormous vertical takeoff & landing (VTOL) fans.

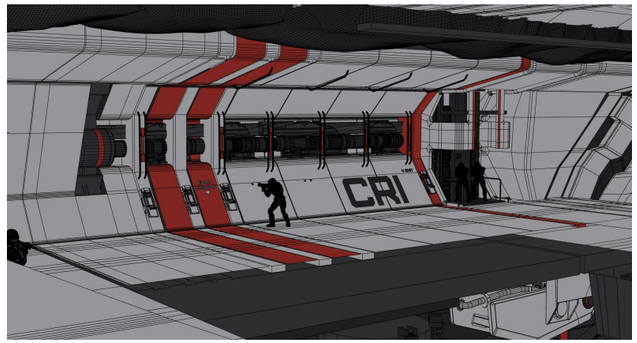
Discussing the future of the Crusader look, Paul noted that the earlier Starliner would likely undergo small modifications when brought into the game to reference the new work done on the Hercules (including reducing the size of the VTOL system, as it turns out he's not a fan!). From here, the exterior was refined in a series of art passes and reviews which Jones described as "drama free." Just as the game itself had clearly called out needing the Hercules, the ship itself seemed to emerge from the marble fairly seamlessly.

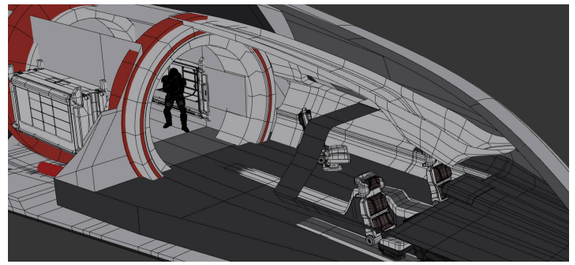


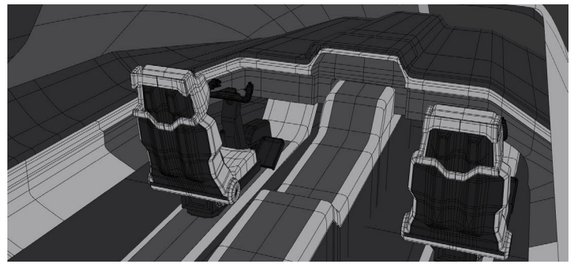
BRINGING OUT THE INTERIOR

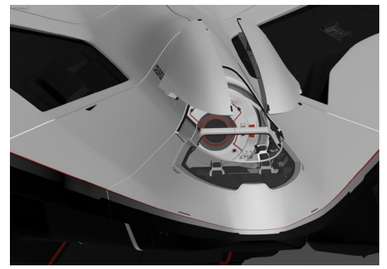
To develop the Hercules' interior, Oberschneider was joined by veteran ship artist Sarah McCulloch, and the pair worked to incorporate all the necessary equipment requested by the design team. The Hercules called for a variety of mid-sized generators and other systems, including life support, shields, coolers, radars, scanners, power plant, jump drive, gravity generator, and fuel tanks. With the broad hull surrounding the cargo area, there was plenty of space for these components while still leaving them accessible to the crew while in flight. These systems were spaced equally around the fuselage rather than focused in a single place, reducing the risk of damage from single, well-placed enemy shot. At the request of the design team, a second shield generator was added into the cargo bay to protect the Hercules' presumably precious cargo.

For the overall shape of the interior, Oberschneider and McCulloch quickly settled on a system inspired by the large military aircraft they had studied: an upper 'living deck' would include the cockpit and necessary facilities, while the lower (but still accessible) cargo bay could be configured to suit the application. This created something that Art Director Paul Jones would call 'deceptively simple': a spacecraft with a large interior volume that didn't need a huge amount of special work done. The design specified no additional animations, so ramps, doors, and seats were modified from existing assets.

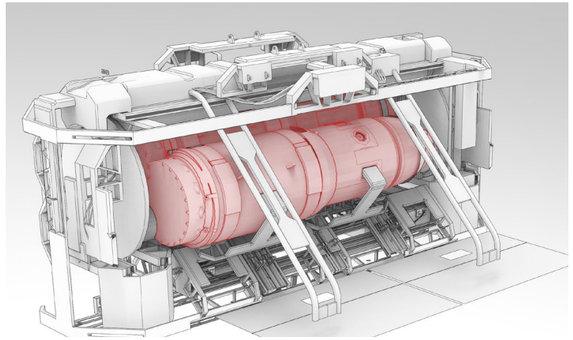


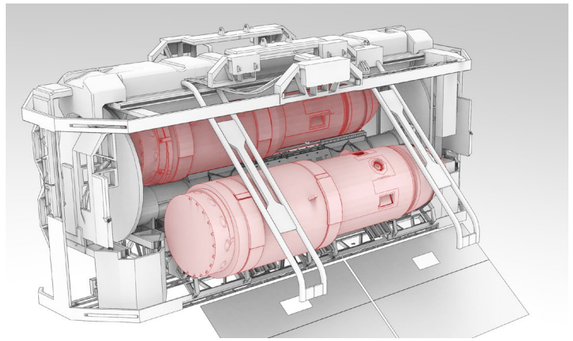


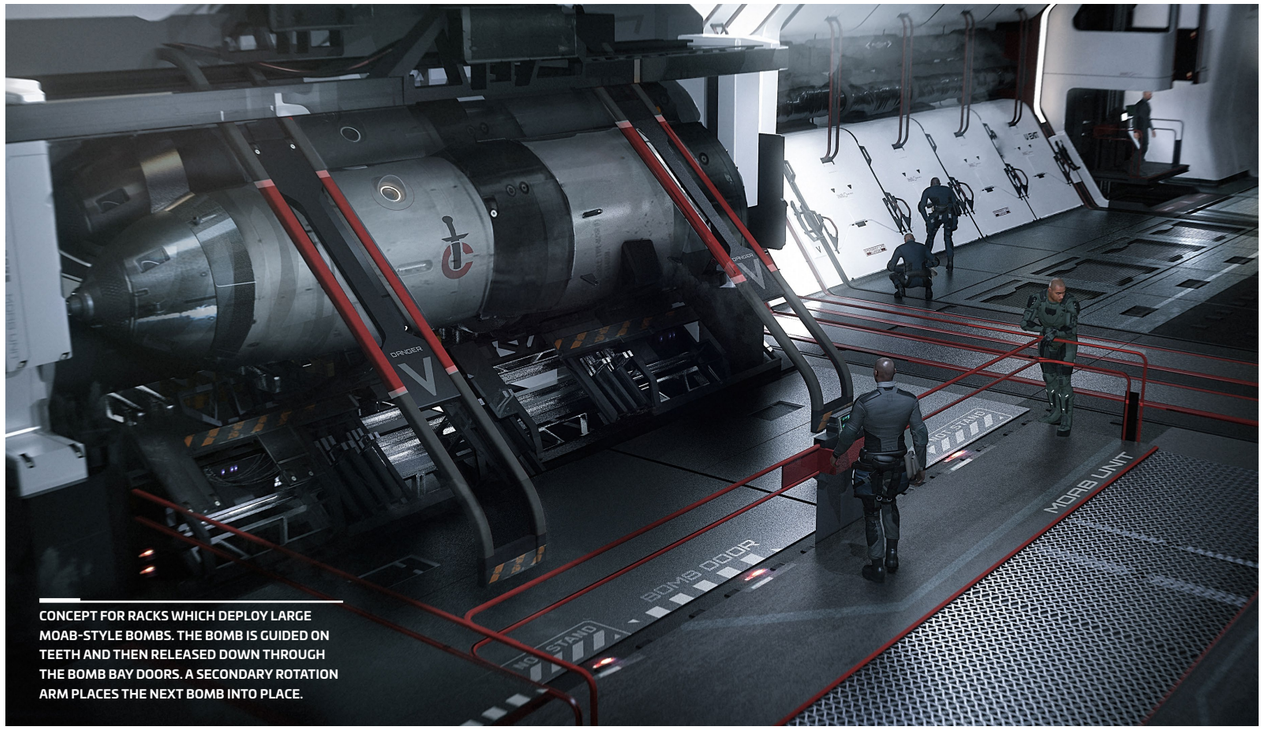




One system that received special attention was the pilot ejection process. Large *Star Citizen* ships often have stand-alone escape pods, while smaller ones typically have fighter-style ejection seats that shoot pilots and gunners away from their damaged craft. While the Hercules was to be a large ship, the former system didn't seem quite right. Instead, the team experimented with an ejection system derived from modern military aircraft (and also proposed for the space shuttle): a break-away section of the upper fuselage which allows fighter-style ejection seats to rocket players away from harm. The result uses existing systems and animations to do something a little bit different.







CONCEPT FOR RACKS WHICH DEPLOY LARGE MOAB-STYLE BOMBS. THE BOMB IS GUIDED ON TEETH AND THEN RELEASED DOWN THROUGH THE BOMB BAY DOORS. A SECONDARY ROTATION ARM PLACES THE NEXT BOMB INTO PLACE.

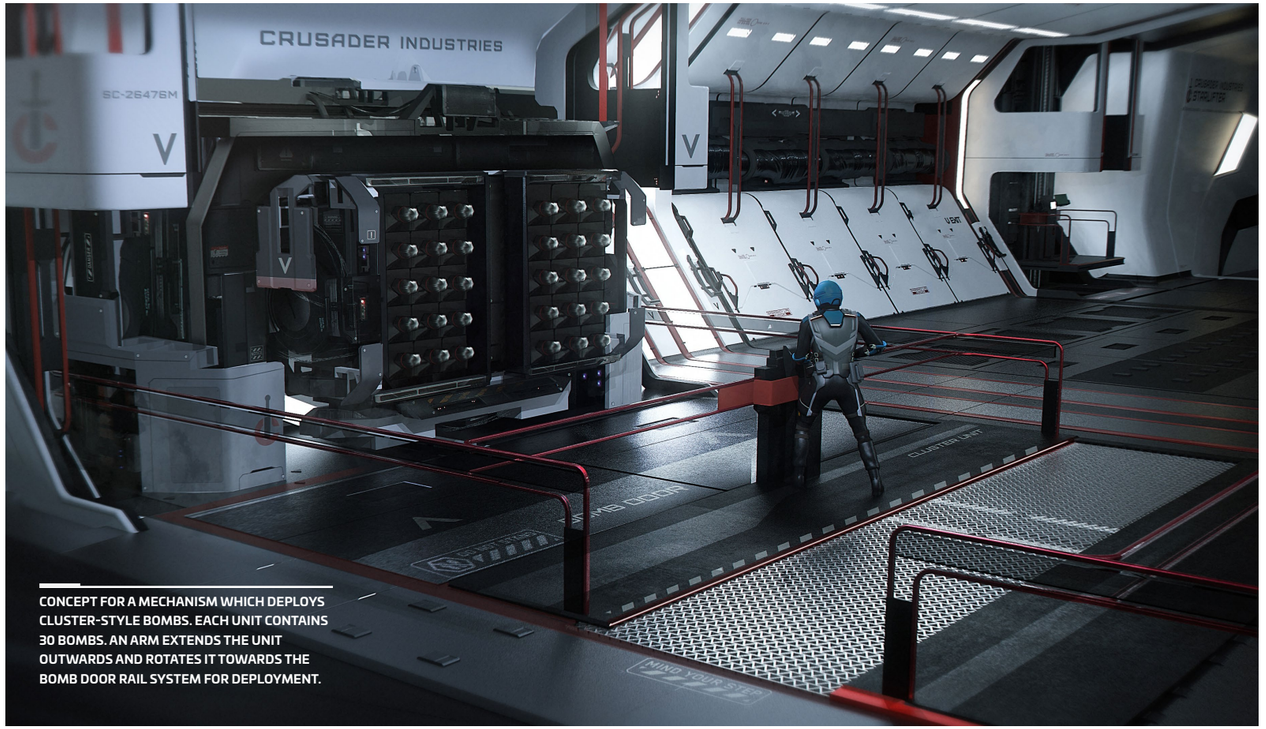
They began incorporating styling and touches from these aircraft into the standard Hercules in the same way Boeing might turn a 737 into a submarine hunting P-8 Poseidon. The large cargo bay proved to be a godsend as it could be readily turned into a bomb bay without a great deal of additional exterior work. The biggest debate occurred when designers lobbied to add the bombardier station below decks in the cargo area/bomb bay. While the artists preferred to add a station to the crew deck, both options were explored and it was ultimately decided that it made more sense to keep the crew together and separate from live munitions.

In addition to adding exterior weapons and several additional turrets, Oberschneider and McCulloch went on to develop artwork for several bomb systems that could be dropped into the cargo bay area. They created detailed graphics showing how the A2 Hercules could deliver large MOAB-style bombs, more traditional bomb pods, or numerous cluster bombers.

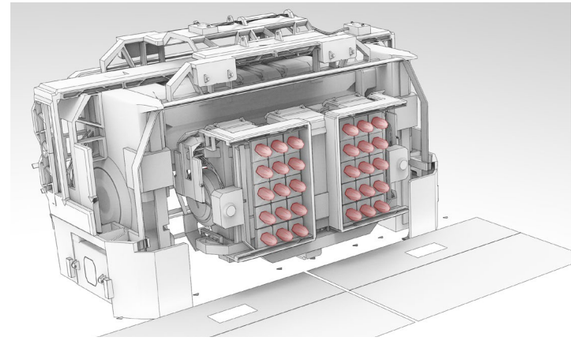
BALANCING VARIANTS

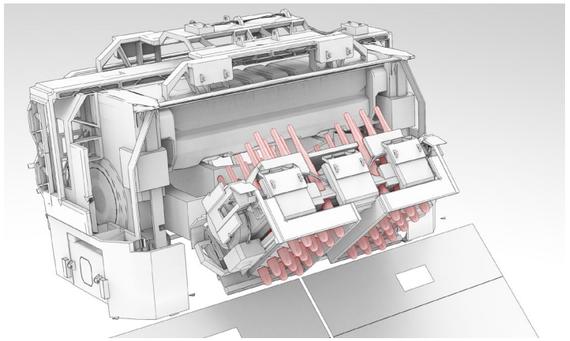
The decision to produce different variants of the Hercules came late in the creation process. While the initial design offered the option of a gunship similar to the spacecraft's real-world namesake, it was originally planned as something to revisit further down the road. However, because the Hercules cargo ship was shaping up so nicely, it was decided that the team would go ahead and develop artwork for a further two types: the C2 Hercules, a civilian version, and the A2 Hercules, a military bomber and gunship.

Civilianizing the ship for the C2 Hercules was a relatively easy task, but adapting it into a working bomber was a special challenge. The Art team again gathered a stack of traditional reference imagery that included attack helicopters, heavy bombers, military-converted passenger aircraft, and the otherworldly B-1B Lancer heavy bomber.



CONCEPT FOR A MECHANISM WHICH DEPLOYS CLUSTER-STYLE BOMBS. EACH UNIT CONTAINS 30 BOMBS. AN ARM EXTENDS THE UNIT OUTWARDS AND ROTATES IT TOWARDS THE BOMB DOOR RAIL SYSTEM FOR DEPLOYMENT.





HERCULES' LABORS

The Crusader Industries Genesis Starliner had been a tough ship to pitch: while everyone recognized the importance of having civilian passenger and cargo spacecraft in the game, it was tougher to design it as a mechanic that players could control. For the final Hercules presentation, the marketing team wanted to show how the ship would be useful and interesting in the 'verse. Paul Jones' artists first set about creating a series of images showing the Hercules at work. Seven images were rendered and composited to show all aspects of the design: a flare shot, showing a single base model flying over a planetary landscape, a straight down shot showing the gunship dropping bombs on a target, a group of military Hercules landing together on a dusty airfield, a shot of the bomber on the ground with munitions spread out in front, a civilian model unloading at a base, a formation of gunships ready for action, and an interior cargo shot showing how much internal space is available.

With these in hand, the lore team worked double duty to create a Crusader Industries brochure touting the ship's features, as well as a series of diary entries telling the story of Ciera Brun, a Citizen assisting with a humanitarian operation that makes heavy use of the Hercules spacecraft. The web team at Turbulent adapted the new styling for Crusader Industries into a presentation post and the Hercules lineup launched to an overwhelmingly positive response. The main message was exactly what had led to the Hercules' development in the first place: this ship is a 'missing piece' that the game needs to proceed. As it comes online in the 'verse, it will greatly support a number of existing vehicles and arenas and will allow the team to focus on building more interesting toys for planetary adventures.

REFERENCES:

HERCULES STARLIFTER PRESENTATION

https://robertsspaceindustries.com/comm-link/transmission/16550-Introducing-The-Hercules-Crusaders-Premier-Tactical-Starlifter

HERCULES STARLIFTER SHIP PAGE

https://robertsspaceindustries.com/pledge/ships/crusader-starlifter/M2-Hercules

CRUSADER HERCULES STARLIFTER BROCHURE

https://robertsspaceindustries.com/media/rsmrf96ceweoir/source/Crusader-Hercules-Starlifter.pdf

GENESIS STARLINER PRESENTATION

https://robertsspaceindustries.com/comm-link/transmission/14801-Introducing-The-Genesis-Starliner

GENESIS STARLINER SHIP PAGE

https://robertsspaceindustries.com/pledge/ships/starliner/Genesis-Starliner

GALACTAPEDIA

RADEGAST DISTILLERY

Radegast Distillery is a historic producer of whiskey located in Port Renatus on Mars, Sol System. Initially founded as a bar in 2520 by the Wilkes family, Radegast became a common stopover point once word spread about the house whiskey they produced in the back. Its popularity grew to the point where a full-size distillery was added in 2574.

Both bar and distillery were purchased from the Wilkes family by the Pan-Galactic Beverage Corporation (PGBC) in 2602. Radegast enjoyed 300 years of galactic success before falling out of the public eye in the early 29th century. In 2920, the brand underwent a resurgence when a private investor bought and rebuilt the original bar in the style of the 26th century. Today, Radegast is consistently ranked as one of the biggest sellers of Martian Whiskey in the UEE.

HISTORY

Dimitri and Danica Wilkes, eldest children of a second-generation Mars-born family, opened Radegast Bar in 2520. The bar saw early popularity among the Port Renatus locals, winning Most Welcoming Bar/Restaurant in a 2532 edition of Mars Today. Dimitri used the storage room (and his regulars) to experiment with home-brewing. He tried several unsuccessful beer recipes and made a notorious attempt at distilling gin, before finally trying his hand at whiskey. The recipe was an instant hit, catapulting the bar to even greater heights of popularity. Eventually, the Wilkes siblings retired on their profits and passed the bar down to their children.

In 2574, Danica's granddaughter added a permanent distillery to the family operation. Zara Wilkes' house whiskey, Radegast Gold, gained a wide-reaching reputation for its smoothness and clarity of flavor. Radegast Distilleries opened a second facility on Angeli, Croshaw in 2585. The first untapped casks of Angeli whiskey were shipped to Port Renatus in 2598. The journey unexpectedly, but pleasantly, changed the Whiskey's flavour, prompting the launch of Radegast Homeward. This limited release was stored in Angeli oak barrels, but matured in zero-G, before returning to Mars for its final aging.

The company's success drove the Pan-Galactic Beverage Corporation (PGBC) to make an offer to buy the Radegast name, whiskey, and bar in the late 26th century. The Wilkes family wanted to make sure that the label stayed under their control, so a formal sale wasn't finalized until 2602. With the added benefit of PGBC's production and distribution network, Radegast whiskey began to appear in all corners of the UEE.

DECLINE

The last member of the Wilkes family left the Radegast brand in 2754, citing personal differences with the leadership of PGBC. This ended 180 years of Wilkes-led distillation of Radegast whiskey, as well as PGBC's access to the Wilkes-owned water source on Mars, Perepllut's Well. However, PGBC retained the rights to the Radegast name, and began selling a low-cost version of its Gold whiskey in 2766. Their first attempt at a new recipe, Radegast Neo, was appallingly received.

In the early 2800s, PGBC came under fire after they were accused of diluting Radegast Gold and passing it off as full strength. The company denied the accusations. However, the scandal repeated throughout the 2810s, finally reaching its climax in 2821 when evidence was published on the spectrum by independent investigator Willis Kemper. The Radegast brand was unable to maintain its reputation, and while the high-end limited editions were still considered top quality whiskeys, sales plummeted.

REBIRTH

PGBC dissolved in 2918, and with it, their right to the Radegast name. A private investor purchased the brand and the building where the original bar once operated. Two years later, the newly rebuilt Radegast bar reopened, rebuilt to look as it did in the 26th century.

Descendants of the Wilkes family were hired to distill classic versions of their whiskey from the original recipes crafted by Dimitri and Zara. The new Radegast whiskeys soon grew in reputation, and the brand was reborn. Master Distiller Hanshi Wilkes currently oversees all production.

LIST OF PRODUCTS

- Radegast
- Radegast Gold
- Radegast Neo (Discontinued)
- Radegast Mars Oak Reserve
- Radegast 18 SEY Old
- Radegast Archive 21 SEY Old
- Radegast Wilkes' Reserve

LIMITED RELEASE

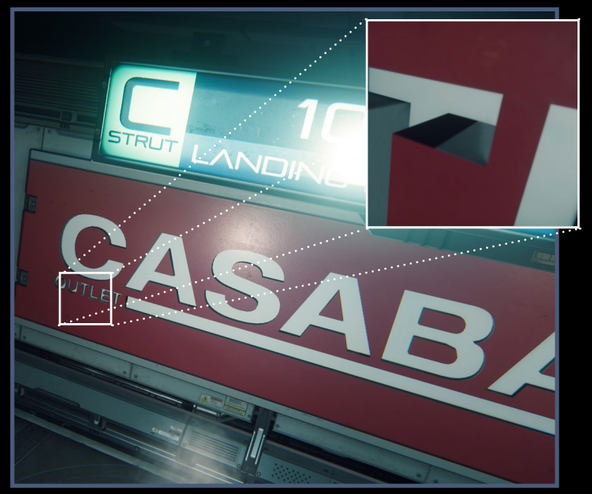
- Radegast Homeward 15 SEY Old
- Radegast Homeward 40 SEY Old
- Radegast Master Distiller's Edition

WHERE IN THE 'VERSE?

Every month, we post a close-up image of something in the universe.
All you need to do is tell us where you think it was taken.

Ben@cloudimperiumgames.com

We'll reveal the answer next month, and share some of the best responses we received.
This month's image is courtesy of Ray Warner, our Assistant QA Manager in the UK.
Where in the 'Verse did he find it?



Ray also gave us last month's image. But Where in the 'Verse did he find it?

Several Citizens knew, but the first to tell us was Elliot 'Selbie' Sellars. He told us:

BEGIN TRANSMISSION →

The moment I saw the red 'T' I knew it was from the Casaba Outlet sign in Port Olisar. I run past it so often it's been burned into my mind.

END TRANSMISSION ←

Congrats, Selbie!

PELCKI, Maxtorr78, Sarge701, Lurian, LordOfThePit, Bot-93 and Storm Tiger were all within a day or so of Selbie's response.

Sarge701 had the most interesting response: *It was on my way to buy some new clothing, because someone stole my fancy shirt and cap out of my cabin here on Port Olisar while I was out for some negotiations.*

Please remember to send us a screenshot of what you find, so that I can give partial credit if what you've found is close to the actual image.

ONE QUESTION

We asked the CIG staff to answer one question for us this month. Here's what they had to say.

WHAT IS THE LAST BOOK YOU HAVE READ (OR THE MOST RECENT BOOK OF ANY GENRE)?

CHRISTOPHER ECKERSLEY, LEAD TECHNICAL QA TESTER, UK

The Horus Heresy: Tales of Heresy — various authors. A great collection of short stories from the history of the *Warhammer 40,000* universe. "The Last Church" especially stands out. The bleak backdrop of the Imperium of Man makes even the Messer-era UEE seem like a bright and promising future.

KAIA LINDSEY, QA TESTER, ATX

The last sci-fi book I read was *Starfish*, by Peter Watts. It's so good I typically read it 2-3 times a year.

TYLER SIMMONS, HR COORDINATOR, LA

The Cold Commands, by Richard K. Morgan (#2 in *A Land Fit for Heroes*).

BEN CURTIS, PROPS ART DIRECTOR, UK

I don't read much, but I do listen to audio books on my daily commute and have just finished up the *Takeshi Kovacs* trilogy (beginning with *Altered Carbon*), by Richard Morgan.

STEVEN KAM, JUNIOR COUNSEL, LA

Skin Game, by Jim Butcher (*Dresden Files* #15). Oh, and just remember — if the building is on fire, it isn't my fault.

LENA BRENK, PRODUCER, DE

SF: *We Are Legion (We Are Bob)* and the sequel *For We Are Many*, by Dennis E. Taylor.
Fantasy: Currently reading the *Skulduggery Pleasant* series, by Derek Landy.

TRAVIS CONKLIN, ASSOCIATE ANIMATOR, ATX

Last SF novel read (listened to really) was *Expeditionary Force: Zero Hour*, by Craig Alanson.

PHIL WEBSTER, QA MANAGER, UK

I recently read through Stephen King's *The Dark Tower* series, since I've read most of his other work. It was definitely an oversight on my part — the series is fantastic! I think it's easily his magnum opus. (Don't bother seeing the movie, that is not fantastic!)

JEFFREY PEASE, DEVOPS ENGINEER, ATX

Hell's Gate, by David Weber and Linda Evans. My dad is who I blame for my love of stories. Any time I visit him, I ask if there's a book he'd recommend because I've always enjoyed the books that he recommends.

ANDREAS JOHANSSON, PU LEAD LEVEL DESIGNER, DE

The Mote in God's Eye, by Larry Niven and Jerry Pournelle. It has an interesting premise where humanity makes first contact with an alien race. Quite unique and well worth the read.

JOHN SCHIMMEL, SENIOR PRODUCER, CONTENT, LA

Not SF, but *Bad Stories* by Steve Almond, author of *My Life in Heavy Metal*, because he has a wonderfully wise view of the power of narrative.

CHRISTIAN SCHMITT, GAME SUPPORT AGENT, DE

The Legion, by Simon Scarrow.

MAC MCMONNIES, PRODUCTION ASSISTANT, ATX

Leviathan Wakes, by James Corey – because I need to Remember the Cant.

STEVE BENDER, ANIMATION DIRECTOR, LA

The Raw Shark Texts, by Stephen Hall.

MILES LEE, MANAGER, BUILD OPERATIONS, ATX

Homecoming #3: Hero, by R.A. Salvatore. I have been reading the *Legend of Drizzt* books for nearly 15 years as soon as each is released. These hold a very special place in my life and the entire series has been a constant inspiration to me.

WILLIAM WEISSBAUM, LEAD WRITER, LA

A Closed and Common Orbit, by Becky Chambers. It is the second book in the *Wayfarers* series, the first being *The Long Way to a Small, Angry Planet*, and it features a rich, vibrant world with characters that I find myself wanting to spend more time with, plus all the fun, wonderful weirdness you want in a sci-fi story. Highly recommend.

DENNIS CROW, SENIOR PRODUCER, LA

I've been reading *Ship of Destiny* (#3 in the *Liveship Traders* series, by Robin Hobb). I'm enjoying the series because it combines two of my favorite things: dragons and pirates.

MICHAEL DALSTON, EMBEDDED LIVE DESIGN QA TESTER, UK

'Salem's Lot, by Stephen King. I've been on a King binge since the release of *It* last year. I absolutely adore his scarily realistic inner monologues, alongside the meshing of classical and Lovecraftian style horror.

JAMES MCCUE, ANIMATOR, UK

The last book I read which would fit the description is *La Belle Sauvage*, by Philip Pullman. I read it since it acts as an earlier story to *His Dark Materials* trilogy which I read a few years ago. I'd say the thing I love about the book most is the richness of its characters. You feel genuine affection for them as they are thrown into situations of great adversity.

GEOFFREY COFFIN, SYSTEMS DESIGNER, UK

Pyramids, by Terry Pratchett. I've been trying to read a little more in my spare time and have been alternating between Pratchett books and Lovecraft short stories. The art for *Pyramids* always appealed, even if it's not considered one of the better Discworld books. Having just finished *Pyramids*, I'll be trying *Guards! Guards!* next.

MATTHEW LIGHTFOOT, PRODUCER, UK

The White Donkey: Terminal Lance, by author and illustrator Maximilian Uriarte. Uriarte is a USMC veteran and has an online web comic strip, *Terminal Lance*. *The White Donkey* is a graphic novel to illustrate the experience of a Marine during the Iraq war and also their return home. It has elements of comedy but it also tackles some of the tough truths of the transitions between war and peace.

MICHEL KOOPER, LEAD ENVIRONMENT ARTIST, DE

I'm currently reading *Babylon's Ashes*, #6 in the *Expanse* series, by James Corey. I started reading the series after watching the first few episodes of the tv show and was immediately hooked! In my opinion it's a modern SF classic with awesome characters, a great story and an injection of cool ideas grounded in science. It's also an ongoing source of inspiration while working on *Star Citizen*. It's a great way to fill my daily commute and get in the mood for work. I would recommend any *Star Citizen* fan to check it out.

Do you have one question you want to ask the staff?

Send it to Ben@cloudimperiumgames.com and we might choose your question for next issue.