Cryptic Game Engine Futures

sdangelo – 12/07/2020

Overview

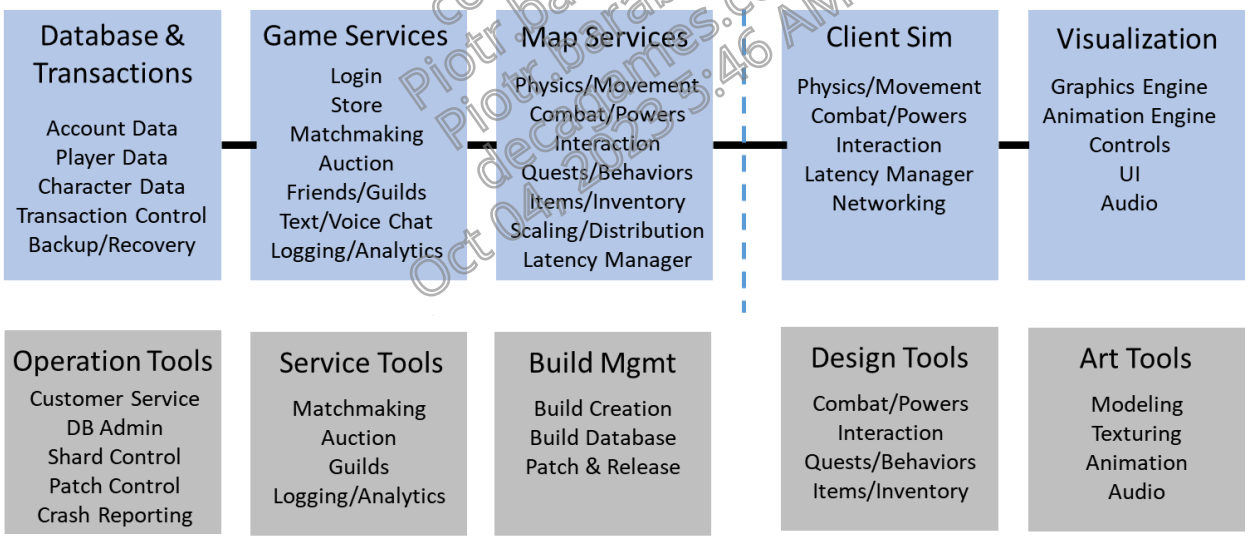
Cryptic's Game Engine has been a huge boon to the studio over the past 12 years since it was first used. It far surpassed anything out there for a long time and still surpasses other game engines in many ways, but it is starting to fall behind in some key areas. It behooves us to evaluate our future plans for the Game Engine.

This document outlines the major topical areas related to Cryptic's Game Engine and to other available engines in order to help educate and then to open the discussion on the future. A proposal is included for a possible path to pursue and challenges are highlighted as well.

This is a huge topic and a short document like this cannot fully explore the implications, but it is hoped that this document does provide a solid basis for common discussion.

Overview of Online Game Engines

In order to evaluate game engines, it is necessary to understand the major components that comprise an engine. When considering games with online play, the list of engine components is quite significant. Here is a very simplified chart of key features:



In order to get an online game with similar features to the current Cryptic games live, there are a lot of features that need to not only be implemented, but tested and operational.

Qualities of the Cryptic Game Engine

The Cryptic Game Engine's primary positive quality is that it is fully operational and tested across a huge range of technologies, including the ones in the diagram but also many more features that did not fall neatly into the diagram.

That said, the engine has some unique qualities that sit at its heart and are not easily replaced. Some of these qualities include:

- **Designed to provide seamless gameplay with a 500ms ping time**, allowing players worldwide to play against a single data center with no appearance of lag. As far as we know, this is unique to the Cryptic Engine and it's a huge selling point. (It is shown on the diagram as "Latency Manager".)

- **Designed to be "server authoritative".** This means that all logic (including physics) runs on the server and not on the client. This makes it very hard to cheat the game. Engines such as Unreal do not offer this and without it, you need to consider other "anti-cheating" solutions or simply accept cheating.

- **Designed to dynamically provide data that alters the game.** The server authoritative nature is not just for game play, it includes game data. Other game engines assume the game is static and the client can know everything. Ours assumes that almost everything in the game can and does change.

- **Designed to be spot-patched**. We can patch individual small files as needed, making it very easy to update the game with patches regularly with optimized sizes for every player. We have found no other solution that comes remotely close. Unreal generates some of the worst patching experiences due to its assumption that games are unchanged after being built. You sometimes have to repatch the whole game if you change even one line of text.

- **Designed to have very low cost of operation**. The database handles massive transaction loads 10x what a SQL database can handle and the Map Services can handle 20x the players that Unreal can handle per machine. This is a huge savings by controlling the software.

The above qualities are all things that would need to be investigated and evaluated in any engine that we seriously consider for the business.

Qualities of Market Game Engines

The major licensable game engines, Unreal, Crytek, and Unity, all were designed primarily for single-player experiences where any servers involved are used only to store and log small chunks of data generated through game play. For example, recording that someone reached a given level or purchased a given piece of DLC.

When they do have server components for multi-player, they generally prefer peer-to-peer unauthoritative play, or rely on bulky servers that respond poorly to high latency networks. Thus, only players on good networks do well and the publisher is encouraged to have servers in many geographies.

Finally, these game engines focus heavily on the client components such as the Visualization box. They generally have zero support for the wide range of features in the rest of the diagram. Developing those other features is left as an exercise for the developer.

If you go an ask any online game developer what their experience was like trying to get Unreal, Crytek, or Unity to work with their game, they will tell a horror story and how they felt it would have been better in most cases to just build their own engine. Most developers who operate more than one title end up with a proprietary engine over time.

This does not devalue some of the amazing technology in these engines, but it does highlight that there is a mismatch between the design goals in the market engines when compared to online MMO games.

Build vs. Buy

The decision to build versus buy technology is always a challenging one for a company. The general rule of thumb, however, is that if something is critical to how your business differentiates itself you should build. If the technology is simply supporting, you should look to buy if you can find something that fits your needs.

A Little History

In 2006/2007 when the Cryptic Engine entered development, there were no viable market solutions for any of the above technologies other than graphics. We surpassed Unreal 3 easily for graphics and Havok's engine for graphics and physics was not very strong. By the time STO launched in 2010, the only licensed technologies we used were for voice chat (Vivox), physics (PhysX), audio (FMOD), and billing (Vindicia).

Over the years, we have introduced a few other technologies, including Coherent UI engine, Backtrace for crash reporting, and shifting to WWISE for audio, but overall most of the engine is still very proprietary.

Should We Have Licensed an Engine?

The question comes up regularly on whether we made the right choice to build rather than buy.

One of the key components that affects our choices is that our games last 10 to 15 years in operation and most licensed technologies for games do not last that long. If we had bought into Unreal 3 for our engine, it would have stopped working for us long ago and we would have been forced to migrate. Even Unreal 4, which only game into production ready state in 2015 is going to be outlasted by our games. If we had invested heavily in a licensed engine, we would be facing a complete reworking of each of our games every 5 years because the engines are not built for online game lifecycles. They are designed for single-player "make it and ship it" games that do not get regular updates.

Why License Engine Technology Now?

The challenge we have at this time is primarily with the Visualization technology and Art Tools boxes. There have been massive advancements in these areas and we have been unable to acquire and keep the right software development staff to keep up. Looking to buy in this area rather than keep building, even with the drawbacks from licensing, may be necessary due to our inability to staff. We are also finding that it is easier to hire and keep artists when they can work with commercial tools.

Proposed Future Path

The game software market has continued to develop and there are more options available now than ever. We should do a deep evaluation on where we can effectively license technology instead of building and maintaining proprietary solutions.

Two Game Engines for a While

Because we have four games live on the current game engine and because it will take time to develop and test new technology, we are likely to have a period where both the old game engine and the new game engine need to be in development. This is discussed further in the next major section of this document, but keep that in mind while reading this section.

Licensing a Visualization Engine

The area where we are struggling to maintain parity in the market it with the Visualization and Art Tools boxes. While we can and do match specific features, it is impossible to match the hundreds of man years of effort going into the modern engines. We are also finding that hiring and retaining artists is harder on a proprietary art pipeline than it would be on a commercial engine.

To this end, we should evaluate commercial engines and select one to be our new front-end.

The proposed goal is to adopt a new engine for rendering, physics, animation, movement, controls, UI, audio, and art asset pipeline. This set of features form the heart of most commercial engines and we should aim to lean into what they do well.

It is also proposed that we look very carefully at and consider replacing any client-server networking, latency management, and game play logic components. Based on what we know now, no available engine does a good job with high-latency, server authoritative game play and we should plan to either develop this on our own or decide on an alternate set of architectural targets.

It is further proposed that we plan to evaluate the build, release tracking, and patching features of the engine, potentially building our own solution if needed.

Owning the Server Side and Design Components

Adopting a new Visualization and Art Tools pipeline will dramatically change the server side architecture from what we are using now, so while we may be able to adapt large portions of the existing Cryptic Game Engine, we should plan on major changes needing to be developed and tested and for the resulting server side logic to not be shared between the old and new game engines.

The same story goes for the designer tools and the back end that supports them. This includes subsystems such as items, inventory, rewards, missions/quests, interaction, volumetric logic, and behavior logic such as FSMs and behavior trees. All of these are systems specific to MMO style play that are not supported in any licensed game engine. Cryptic's implementation relies heavily on Cryptic's client-side engine and so will need reimplementation when we move to a new Visualization solution.

Revisiting Database and Transaction

The Cryptic database solution is proprietary and this has served us well, providing a massive increase in transaction throughput over any SQL solution. There are, however, some NOSQL products that have

grown in maturity in the past decade that should be evaluated to see if we can move to a distributed data solution for game data and no longer need to support our proprietary solution for the new engine.

Revisiting Revision Control, Build, Release, and Patching

The Cryptic system uses a custom revision control system. We should consider moving off of "Gimme" and onto a commercial system such as Perforce.

Cryptic uses a custom build management solution. We recently evaluated commercial solutions and did not find anything that really met our needs, but we could look again. We also have a newer system, "builder 2", that is partially developed that could be brought to completion.

Cryptic uses a custom release management and patching solution named "PatchMaster". This build tracking and patch system is fairly simple and effective with no equivalent in the market. We should consider keeping this system (but separate it completely from the "Gimme" code). The client side should also be simplified to rely entirely in HTTP without using the proprietary PCL protocol.

Everything in this section could be done for the old engine and shared with the new engine.

Revisiting Account Management

Cryptic uses its game database for its account database mostly because it was available. We should instead replace this with a straightforward SQL solution. Account data should both easily move to SQL and benefit from substantially easier reporting and validation as well as reducing the cost to develop and maintain customer support tools.

This shift could be done on the old engine and shared with the new engine.

Some Additional Features to Pursue

While working on the new engine, there are some additional features and problems we should consider addressing. All of the below have proposed solutions on the current engine, but we have lacked resources to dedicate to them. When we pursue a new engine, these should be considered.

- How to push updated revisions live without downtime.
- How to better utilize multiple monitors for editing.
- How to reduce patch size by streaming audio (particularly VO) instead of patching it.

Getting From Here to There

We have four live games on the old game engine and that means that we must continue supporting that old engine, potentially for another decade. This means keeping a reasonable level of software engineering support on the old engine. The proposal is to keep this software team in Los Gatos with the games that utilize it.

The new engine will be a huge software development effort and will likely take at least 2 years of work with a large software team before it reaches a level of completion where game play would work correctly enough to build the first game on that engine. Some prototypes could be done while the engine is being developed, but likely no substantial progress on game play could get underway until the engine is stable. The proposal is to build the new engine and its software team in the Site B office that we propose to open in 2021.

Once the new engine is ready, we can discuss plans on whether it makes sense to migrate any of the older games from the old engine to the new one. This will depend a lot on how compatible the two are in basic concept. If they are radically different, then migration will be impractical. This begs a larger question on whether we should constrain the new engine to be compatible enough for migration. For a concrete example, there are hundreds of thousands of items in our existing games that have icons, powers, and abilities. If the new engine does not handle items in a substantially compatible way, it would be impractical to manually recreate all of these items in the new engine.

We also believe that having two engines will cause social problems in the studio. We expect to find it harder over time to have people working on the "old engine" be satisfied with their job. They are likely to feel like they are shepherding a dead technology or product rather than doing valuable work. As such, we are likely to see slow but steady decline in staff that is willing to work on the old engine and the four products that are built on that engine. If we don't provide a migration path to the new engine, these products will have a substantially reduced market lifetime simply due to inability to staff them with quality employees.