Turbulenz Games Platform: Technology Introduction

This documents explains the technology decisions made by Turbulenz and provides our partners with insight into the technology architecture powering the Turbulenz Games Platform, launching in fall 2011

Turbulenz: Our vision for the future of Games

Video games and their development have continuously evolved. Consoles have dominated the games industry by creating the highest quality products and generating a large percentage of the revenue in the industry. However, this is now changing; social gaming, mobile, and the web in general have emerged as significant competition to both revenue and game quality that consoles have enjoyed for so long. Developers of classic console games are now evaluating how to use the web for existing IP and new properties as well.

Turbulenz was conceived and implemented to deliver a truly new and focused platform for creating and playing games on the Internet: **a console for the web.**

What exactly is Turbulenz?

Turbulenz is a platform for high quality internet games, just as PlayStation is a platform for console games. As a platform, Turbulenz contains a wide stack of components, all required to create, play, and monetize games online.

1. Developer SDK

The Turbulenz SDK contains all the tools and libraries that developers can use to create their games. The SDK includes documentation, samples, asset processing tools, and a local game server that can emulate all the live APIs. This is equivalent to the UDK that comes with the Unreal engine.

2. Game Engine

The Turbulenz Engine is a JavaScript game engine. The engine contains all the classic services expected and required by a game developer, including: graphics, audio, physics, animation, and networking. As a comparison, the Turbulenz Engine is equivalent to the Unreal engine.

3. Game Services

The engine also includes a completely new generation of APIs that span the engine and the cloud, including: save games, leader boards, badges, chat, friend lists, challenges and payments. These features are powered by modern HTTP request APIs, and are just like the APIs provided by Twitter, Facebook and Foursquare. To the game player they contain all the features offered by Xbox Live and Steam. But to enthusiasts and developers they allow huge scope for integrating their games and the Turbulenz services with everything the web has to offer.

4. Developer Hub

Developers can create their products locally using the SDK and Engine. But the developer hub provides a staging service to test the performance, scalability and features of their game. The hub provides an integration point where developers can deploy development versions of their games and then share the result with testers, guests and other team members. Whilst the game is on the hub, everything is behind closed doors, but with a single click the game is published live for everyone to play. Once this happens the hub evolves into an analytics and feedback site, where a developer can learn about how their game should be evolved. The hub is equivalent to a mixture of web services including: analytics, feedback, and payment details. The hub also provides many of the features provided by Sony's SCE DevNet and Microsoft's Game Developer Network.

5. Gaming Site

Most critically Turbulenz delivers the most modern and engaging gaming destination. Once a developer hits "publish" on the hub, the game is available to play directly on the Turbulenz gaming site. The gaming site acts as a destination for players but also acts as a central point for hosting demos, incremental games, or full games on any external website, exactly as YouTube movies can be embedded remotely. The gaming site integrates the games seamlessly within a truly modern web application, that blends together all of a game's fun and social features with the wider gaming community. By comparison, the closest thing to the Turbulenz gaming site is a mixture of a PlayStation 3 and Facebook. Turbulenz could be described as a console for the web, and is filled with social features to elevate the gaming experience into something new.

Why is Turbulenz an Internet generation engine?

All of the most successful game engines have been explicitly designed or at a minimum heavily re-factored to embrace the new platforms' characteristics. For the current generation of consoles, this required engines to start embracing an architecture suitable for multiple cores. It is from this perspective that we describe Turbulenz as an Internet generation engine. Turbulenz wasn't designed for a console or a phone, but rather the Internet. This means that many technology decisions were taken to embrace everything unique to the Internet as a platform, most specifically the language, resource management, streaming and style of the service APIs.

Whilst Turbulenz was designed for the Internet, we have also made sure that the engine is high performance on all the supported devices. This means taking everything there was to learn from the latest generation of console engines and applying this to the core of the Turbulenz Engine as well. We believe that the result is highly suitable for devices today and is perfectly positioned for the devices and distribution mechanisms of tomorrow.

How is the Turbulenz Engine a JavaScript game engine?

At the heart of the Turbulenz technology is a sophisticated game engine that is completely implemented in JavaScript[1]. JavaScript was first introduced by Netscape in 1995 as a simple scripting language to manipulate web pages. Since then it has evolved into arguably the most relevant and important language for product development today. Any business invested in the Internet and more specifically the web are creating products that span a user's browser and their backend servers. Whilst the implementation language for the backend varies the language used for the web front-end will always be JavaScript[2]. Modern Internet focused businesses such as Facebook and Google are heavily invested in creating applications in JavaScript, and the browser owners - mainly Mozilla, Apple and Google - have invested heavily in the performance of their JavaScript runtimes so that sophisticated applications can now be created and executed directly in the browser.

Turbulenz has embraced this movement and applied exactly the same direction to our game platform. Just as Google created Google Mail and Google Maps as applications that run directly in the browser, Turbulenz has also created the Turbulenz Engine allowing games to be created that run directly in the browser.

As Turbulenz is a JavaScript engine it means that the application exists and executes in the same space as the web page. This means that the developer is able to blend together what we would originally think of as the gaming content and the page around the game. The game and the page are ultimately the same thing. The Turbulenz gaming site leverages this to deliver our 'console in the browser' experience. The game interacts with the platform features of the site and the remote API features of the service to deliver a new and unique experience.

Does JavaScript have sufficient performance for modern games?

JavaScript performance has changed dramatically since the language was introduced. It is no longer

1 Many games already adopt an engine and script separation, most commonly scripting with Lua or UnrealScript.

2 Development tools such as GWT and Objective-J still ultimately execute as JavaScript.

a scripting language that is interpreted as a byte code, but rather just-in-time (JIT) compiled. This means that developers can take advantage of the language's flexibility and rapidly develop without compromising performance. Once the application has been just-in-time compiled, it then runs as native machine code.

Originally we created a technology demo using the Quake 4 assets. The demo created an FPS on top of the standard Turbulenz Engine but implemented a modern deferred renderer in place of Quake's original forward renderer. Once we had this multiplayer demo running at over 60 fps with particle effects, full screen effects, character animations, a full physical environment, 3D audio, networking and the deferred lighting solutions, we decided that JavaScript was absolutely capable and suitable for creating modern games.

How is the Turbulenz Engine an HTML5 game engine?

HTML5 is a set of standards and APIs that have been grouped together to denote a new generation of web technology. Whilst the HTML5 standard continues to evolve, the modern browsers are attempting to implement and comply with the evolving standard.

We describe the Turbulenz Engine as an HTML5 game engine because it is also a modern technology that leverages many of the features available in HTML5. Additionally, as we previously described, the Turbulenz Engine is implemented in JavaScript and is able to use many of the new features provided in HTML5, most specifically WebGL.

How does the Turbulenz Engine compare to WebGL?

WebGL is an API standard included in the HTML5 definition. When WebGL is implemented in a browser, it extends the features of its native JavaScript engine to allow a web page to dynamically access the devices' underlying graphics hardware. As a graphics API, WebGL is based on OpenGL ES 2.0.

As a pure technology comparison; WebGL is equivalent to DirectX and Turbulenz is equivalent to Unreal.

Comparing WebGL and Turbulenz is just like comparing DirectX and Unreal. Unreal uses DirectX, in the same way Turbulenz can use WebGL. Unreal and Turbulenz are game engines. WebGL and DirectX are rendering APIs.

This means that when a game is created using the Turbulenz Engine, the engine will automatically detect browsers supporting WebGL and use it as a graphics device.

How does the Turbulenz Engine work on browsers that do not support WebGL?

We have created the Turbulenz Engine to work on all popular browsers used today. We have been careful to make sure that developers are able to target every browser and hence not limit their potential market size. We believe that the Turbulenz Engine architecture allows an application to work on yesterday's, today's and tomorrow's browsers. This is essential to make the engine and a platform as a whole viable for developers looking to adopt stable and future proof solutions.

Today, WebGL is only available on a small fragment of browsers[3]. However, with the Turbulenz Engine, developers can target all browsers. Turbulenz is able to achieve this by providing a binary fallback that can be used when the browser doesn't provide all the services the engine requires. The binary extension can be installed by gamers who are unable to upgrade their browser. The extension then injects all the missing APIs into the browsers native JavaScript engine. This means that from the game perspective all features will always be available.

Whilst we do provide this binary extension, it is purely for backwards compatibility with older browsers. We absolutely believe in web standards and the open web. But because of the current distribution of web browsers used around the Internet we're required to provide the extension as a transition solution until all browsers provide all the services required.

So yesterday and today Turbulenz gives you global compatibility with the binary extension, then today and in the future Turbulenz gives you global compatibility with WebGL.

In summary, the Turbulenz Engine is an HTML5 game engine implemented in JavaScript, the only language included in all browsers. The Turbulenz Engine is not a plugin. The current binary extension is implemented to allow the JavaScript engine to work on all browsers today. In the future the game and engine will continue to work natively with WebGL. Turbulenz gives developers an engine and

[3] Currently WebGL is included with and enabled by default in Mozilla Firefox 4 and Chrome 9. WebGL is only included in developer builds of Safari and Opera.

most importantly a migration strategy.

How does Turbulenz compare as a platform to alternatives?

Once we start thinking about Turbulenz as a platform rather than just an engine we can compare it to alternative platforms. Currently we would break the technology and media industry in general into the following categories:

1. Per Device Platforms

Sony's PlayStation 3 and Microsoft's Xbox 360 are examples of single device platforms. Games created for these platforms can only be distributed on these devices. This doesn't preclude the option to port the game to other platforms, but ultimately the game is tied into a single hardware device. Not only are the games linked to a single device, they are also locked into a single network for distribution and limited social connections. By their nature and ownership, these platforms are controlled and manipulated for their owners' ultimate benefit.

2. Per Vendor Platforms

iOS and Mac OSX by Apple, Android by Google, and Microsoft Windows / XNA are examples of platforms where a single game can be developed and distributed across a single vendor's platform[4]. In this group there are multiple hardware devices all linked by a single software provider. The software or often the operating system defines the platform across a number of hardware devices. More recently this has been extended to also mean that applications are distributed by, and access is controlled by, a single vendor administered store.

3. One Internet Platform

Above the two other groups is the Internet platform. At this level, a game is accessible across all hardware platforms. Examples of platforms in different media industries are: Amazon Kindle for books, Netflix for films, and Facebook for social relations. We believe that Turbulenz is the first gaming platform that fits into this generation of platform that is defined by the Internet rather than a single hardware device or a single vendor platform. Amazon Kindle is an excellent example. The Kindle was originally a device just like the PlayStation 3, however today Kindle content can be acquired on the Amazon web store then consumed on Microsoft Windows, iOS, BlackBerry, Mac OS X, Android, Windows Phone 7 and directly in the browser. The content becomes free to be consumed across different devices, and is ultimately hardware agnostic.

By design, Turbulenz is an Internet generation platform and can take advantage of the emergent connections on the Internet. Web products and services such as Facebook and Twitter, and more recently services such as FourSquare, have evolved and thrived because they have embraced the Internet. These services have taken off for many and varied reasons. Facebook has grown in comparison to alternatives because it reached a critical mass of people using the service. Facebook can be accessed directly through their web-app but also through their mobile clients. Twitter has grown as a tool that redefined the speed and closeness with which users can communicate, and can be accessed via a wide range of applications. Whilst their core products were successful and sufficient to encourage adoption, they have thrived because they all exposed their platforms as services for external parties to leverage and integrate with.

In exactly the same way, Turbulenz provides a wide range of Internet services that mirrors these comparisons. Just like Amazon Kindle, Turbulenz powered games can be played on the web and our target of any Internet connected device. Just like Twitter, the social features in Turbulenz can all be accessed on mobile devices for people who want to stay involved in the fun when not playing. Just like FourSquare, every single Turbulenz feature can be accessed and manipulated through our web APIs. And finally just like Facebook, Turbulenz has been filled with social features designed to connect players and encourage their increased engagement with their games.

[4] Again, there is nothing to stop developers porting games between different vendor platforms. But the application by definition of the vendor's platform and architecture is unable to move without development effort. But even then, the different versions of the product are isolated from each other as different standalone versions.