

OTOY Technology White Paper

GDC 2006 Lecture by Jules Urbach

JulesWorld LLC

13351-D Riverside Drive # 620

Sherman Oaks CA 91423

Email: info@otoy.net

Tel: (818) 762-2307

Fax : (818) 301-1957

Table of Contents

What is OTOY?.....	2
JPEG analogy.....	2
Scalability of content, like a JPEG, only better.....	2
What can OTOY do?	3
Overview of the OTOY Business Model.....	4
Selling content	4
Advertising and sponsorships.....	4
What are the immediate goals of the company and the technology?.....	5
What kind of distribution and sales models will be in the OTOY system?.....	5
Overview of the OTOY Technology:	7
What Proprietary Technology is in OTOY?	7
How is the network kept secure?	8
How does OTOY web content get to end users?.....	8
How does the OTOY technology get installed on the end user's machines?..	9
How is it possible for OTOY content to work on cell phones and devices without the runtime?	9
What does the OTOY runtime do after it is installed?	10
Why is technology like OTOY not everywhere today?	10
Overview of the Development Tools for OTOY	12
How is content created in OTOY?	12
Templates for Developers and Content Creators.....	12
Summary of Key Technology Features:.....	15
Long Term Goals of OTOY	16

What is OTOY?

OTOY is an OS independent platform for creating, sharing, deploying, and monetizing a virtually unlimited scope of content and applications.

The range of content that runs on the OTOY platform is no less diverse than the web pages on the Internet or the applications that can be installed on a PC. In fact, OTOY is a superset of both of these technologies. It is a miniature operating system with the ability to deliver - to any device - applications the size of web pages, which are no less powerful, and visually indistinguishable from, content that ships on DVDs or takes hours to download.

JPEG analogy

If Microsoft Office in its entirety were as small and as easily manipulated as a JPEG, every picture on a web page would be a rich application that could be swapped out instantly, updated daily, or mixed with a gallery of other application-streams to create completely new content and services. Instead of releasing a new version of MS Office every year, Microsoft could update it daily as easily as CNN.com changes its headline picture. Rather than selling an application, Microsoft could support the service through a much broader range of revenue streams: ads, sponsorships, subscriptions or an 'iTunes' like store where each feature of Office could be sold, like a single on an album. The separation between an application and the content it renders would also disappear, as content would contain or reference all the code that would be needed to render itself, which would be no larger than downloading another 'JPEG' sized file.

This is exactly what OTOY does. OTOY content can be code, or it can be data, or it can be a combination of both. Either way, the core OTOY technology takes care of transmitting, securing and rendering the content. More importantly, this process is completely agnostic of where or how the content was authored or the device on which it is being run.

Scalability of content, like a JPEG, only better...

A JPEG image can be viewed by any kind of device that has a JPEG viewer built for it. The JPEG image file, regardless of the device it is viewed on, or how it is transmitted, is usually created by the author with no particular concern regarding the device it will be seen on.

A mobile phone may have to rescale and halftone the image to display it properly on a small screen. A digital camera, with more memory and a better display, could pan and zoom a portion of the full image on its screen. A modern PC could render the image completely, without cropping or scaling it. OTOY content operates on this same principle. The content will run on any device and scale itself accordingly. This is straightforward for images or static media, or even simple applets that use vector graphics.

What makes OTOY unique is that all applications are 'virtualized'. That is, the media and code for content is stored in such a manner that it can be rebuilt on a range of devices with drastically different features (i.e. mobile devices vs. a PC) without having to worry about repackaging the content for different mediums.

Both Java and Flash have 'mini' versions of their runtimes which are designed for phones. But their feature set is reduced considerably, and content needs to be reworked, particularly in the case of Java, or anything that uses palettes or windows. With OTOY, the same runtime and content will scale from a cell phone to a PlayStation 3 without the need for developers to make different versions of the application.

What can OTOY do?

The existing set of components that have already been built into OTOY include a 3D Game engine, P2P services, web browsers, chat/social networking portals, blogging tools, IM clients (including AIM, MSN and gateways), e-mail clients, file/folder sharing and browsing systems, HTTP and FTP web servers, video, music, photo, and rich document viewers and editors. The total size of all of these OTOY applets, combined, is 77k on top of the runtime engine.

An entire OTOY city scene that would rival a level on a DVD PS2 game is about 800k. This gives an example of the media compression built into the technology. Every type of data (lossless binary, lossless image, lossy image, audio, and video) has a proprietary counterpart in the OTOY runtime that is 30% better than zip, 5x better than JPG, 2x better than PNG, and 2x better than MP3 and MPEG2, respectively.

But by far the most compelling aspect of the technology is that OTOY applications and media can be layered and mixed together by developers and users alike, in ways that defy the current limitations set by either web content or traditional executables.

For example, the web component in OTOY can render a fully browseable web page on the surface of a 3D object in a game, using Photoshop-style ink shaders to blend the web page with the original texture of the 3D model – even

on the oldest of video cards. The object can further be pulled out of the game by the player at any time, with the web page still on it, passing across the edge of the host game's window and traversing the desktop space. It can then be dropped into a completely different environment or game, and seamlessly interact with whatever properties or rules are defined by its new container.

There are no limits as to how deeply content, objects and services can be mixed together and repurposed, other than what is defined procedurally by the authors of both the assets being referenced (such as the 3D model in the above example and the web site being rendered on it) and the context in which it is being used (which can vary from a public, massively multiplayer game to a private chat among friends from a buddy list or address book).

Overview of the OTOY Business Model

The monetizable audience for OTOY is enormous and exceptionally scalable. It includes consumers, content providers and developers of virtually every device or system that connects to the Internet. This encompasses cell phones, embedded systems (potentially next generation music players and DVD players), set top boxes, game consoles, portable gaming systems, and of course PCs and Macs.

Selling content

Digital signatures are required for any service, media or data to pass through the system between users. Thus, content can be securely locked and attached to a sale and to a specific client (through any backend, such as PayPal). This immediately creates a marketplace for 1st and 3rd party premium content (such as 3D game objects, subscription services, new application components, audio-visual media) and generates a large secondary marketplace, since OTOY allows end-users to sell and resell content within limits imposed by the original authors of the content.

Advertising and sponsorships

The volume of end-user participation is directly monetizable through advertising and sponsorships, which become even more compelling as sources of revenue when embedded in 'sticky' applications such as chat, linear content and games, all of which are core applications of OTOY's feature set .

OTOY will also be able to upsell digital certificates, enhanced authoring tool components and use of the P2P system (for both content and processing) to developers.

Licensing models for larger companies are also very viable, as many sites will want to maintain and manage an OTOY 'world' parallel to their web content.

What are the immediate goals of the company and the technology?

The short term goal of the web technology is to quickly create and foster the largest possible audience of consumers, developers, sponsors, licensees and end-users within the OTOY system. The current state of the technology, development tools and content library already establishes OTOY firmly in this space.

The initial testing of the technology has been underway since January 2005, and an expansion of the beta program is pending the finalization of the development toolkit for OTOY 2.0.

What kind of distribution and sales models will be in the OTOY system?

Digital Objects for sale:

A primary market of digital assets and components in OTOY can include chat and game avatars, self contained mini games, web based massively multiplayer systems, scripted behaviors and rendering filters.

Secondary Marketplace

The secondary market for game objects is potentially as large, if not larger, than the primary market. The ease of use in reselling content within the system by end-users creates a secondary marketplace without relying on 3rd parties like eBay.

Subscription Services

Subscription models both for 1st and 3rd party content will be developed for specialized game worlds, commodity objects sold in groups (like baseball card packs), closed social communities, and audio and video content.

Audio and Video Media

The sale and retransmission of audio-video broadcasts, both of television, music, radio and film feeds can be explored. Of course, this needs to be backed up with a license by the copyright holder of the content.

Video and audio can be limited to a proprietary compressed format that is encoded with the serial number of the computer compressing the stream. This is in addition to a digital key trail that would exist for media swapped outside of the user's buddy lists in the OTOY P2P network.

OTOY 2.0 is capable of photorealistic real time broadcast streams of 3D linear content. For assets that can be rendered this way, this approach is more efficient than using traditional methods relying on compressed video streams, which limit fidelity, scalability and interactivity.

This model will be explored for delivering linear content to set top boxes, media PCs and embedded platforms where the 3D engine can operate. The linear content created in OTOY 2.0 can also be repurposed into compressed video at any resolution.

All linear content in OTOY 2.0 will be ready for next generation displays and formats (i.e. HDR displays, NHK's proposed 32-megapixel broadcast standard, etc.)

Overview of the OTOY Technology:

What Proprietary Technology is in OTOY?

The entire OTOY framework is contained in a small runtime library, built in C/C++ and assembly, containing about 5-6 million lines of code.

The primary differentiating factors of the technology are exemplified in the uniqueness of the runtime:

- Its tiny size (less than 800k)
- Its portability to almost any type of device (including cell phones, embedded devices, game consoles and set top boxes)
- The vast scope of its features, which rival that of a full fledged operating system
- Mechanism for installing and distributing the runtime instantly and transparently to end user's systems
- Embedded and fully cross platform graphics system for high end 3D, 2D, vector and procedural effects. There is no dependency on any 3rd party hardware or software components. The graphics library can faithfully render a complex 3D game with all logic and rasterizing performed completely within the runtime.
- Viral distribution of both the runtime and content through existing buddy lists, e-mail contacts, and web pages. Plug-in size is so small that distribution of the entire client in a few seconds through a link is possible.

The OTOY runtime contains a virtual machine that interprets its own scripts, uses its own virtual desktop and file system (which can wrap around local file streams or raw socket connections) and performs all rendering (2D, 3D, vector, imaging, text, formatting, document, layout, culling), encryption (RSA 128-4096 bit, EC) and compression (lossless image, data, text, x86 binary code, and lossy audio and image/video).

All of the functionality used by OTOY applications and content are contained in the core library and depend on no specific functions external to itself.

Each runtime operating on a client's machine composes a node in the OTOY P2P network which can be used for almost limitless purposes, including data and virtual storage, as well as parallel processing of scripts and virtual hosting of applications for thin clients.

At 800k, the runtime is smaller than a 20-second preview track on iTunes. It installs instantly and transparently on a user's machine from an IM, an e-mail or a web page. The runtime can be re-sent virally to other users (through existing buddy lists, emails etc) whenever the user shares any OTOY content with a *new* user outside the network.

A unique feature of OTOY is that content passing through the network can also run on devices without the runtime being installed. This exposes thin clients (i.e. older cell phones) to the system immediately, and will push external users to get the runtime and become full consumers of the platform.

How is the network kept secure?

All content in OTOY (and the limits imposed by its authors) is authenticated at runtime through digital signatures. Content does not run outside of the limits set by the content's author, whose signing key is itself defined by a parent certificate with the power to grant these rights. There is a chain of certificates leading to the root OTOY key, which allows all portions of the API to be used on any machine. This master key is never given out, but is used internally to change core pieces of the technology within the same framework that allows developers and content creators to change or deploy content above the core layer. Therefore the entire system is flexible enough to rewrite itself from the ground up over time. With a framework similar (but not as statically defined) as COM's, or as large as .NET's, the components built in OTOY will be agnostic to past and future versions of other components with which they interact.

How does OTOY web content get to end users?

OTOY online content can be placed in a web page. Or it can be sent in an e-mail, or as a link in an IM or SMS text message. Regardless of how it is disseminated, when a URL pointing to OTOY content is clicked on or viewed (from any application), the OTOY network will send the content to the end-user's machine along with any runtime components needed to adapt the content to best run on that system.

How does the OTOY technology get installed on the end user's machines?

If a user views OTOY content on a system without the OTOY runtime (i.e. it is a virgin machine), OTOY will then install itself almost instantly through a stub mechanism (usually between 30-100k) that is downloaded just once (with no more than one click required by the user to install).

Users on thin machines (i.e. old cell phones), that cannot support the native client, would instead be routed to a virtual session of the application, hosted by an available node on the P2P network that scales the content for the thin device in real time.

On a system where the native OTOY client *is* installed, the runtime renders the content seamlessly within the host application (as an overlay, palette, embedded frame, etc.). The user never has to quit an application, restart the machine, leave a web page, or do anything other than click 'OK'. All further updates to the engine are handled transparently.

The OTOY stub manages all platform specific calls made by the runtime. All of the runtime above the stub layer is self-contained and portable to almost any hardware 'as-is'.

The OTOY stub is instantly (and transparently) installed from the URL (in web page, e-mail, document or Instant Message) that references OTOY content. The stub can be distributed in various forms (web plug-in, exe, media codec, QuickTime plug-in on the Mac OSX, Netscape plug-in for Safari or FireFox, ActiveX control for IE/Outlook) and multiple stubs can reference the same runtime. There is also a versioning system built into the stub layer, as well as digital signature checks for all components that it loads.

How is it possible for OTOY content to work on cell phones and devices without the runtime?

Unlike Java, Flash or managed .NET, which, as of the time of this writing, depend on their runtime libraries being installed locally on end users' machines for content to run, OTOY can actually use its network of existing runtime clients to remotely host a complex application 'on the grid' and retransmit the application window to extremely thin clients where the runtime does not exist at all.

The OTOY system scales and reformats the content on the remote nodes and rebuilds it in a format that the thin device can work with (usually an HTTP stream). The power to do this comes from the unused cycles in the P2P network of OTOY client machines where the runtime is installed. The layer of abstraction

making this possible is not just a part of the playback mechanism, but also a core feature of the authoring tools and scripting language which packages all content and code in progressive layers suitable for this kind of scalability.

This means that the key portions of the content (i.e. the scripts and object boundaries) download first and render quickly on even the thinnest devices like a cell phone, while more powerful systems, like a new PC, can keep downloading larger and richer portions of the content (such as extra texture layers like normal maps, HDR data, higher resolution meshes etc.)

What does the OTOY runtime do after it is installed?

OTOY operates on several levels, not just as a content viewer, but also as a transparent part of the desktop outside of the browser's boundaries.

OTOY automatically can work within existing services, such as the user's default browser, desktop, file system, IM and e-mail client, so that content can seamlessly be injected into each type of application from any link. The runtime engine is thus able to pull content from any URL and transparently integrate it with existing web browsers, e-mail clients, and IM programs.

OTOY objects can render transparently over the desktop and provide alternative shell mechanisms (i.e. 3D folders), even on legacy operating systems.

Every piece of content that operates outside of the boundaries of the browser has two limits: a special certificate needs to be granted to the author of such content, and the user must allow the content to run through a dialog box interface.

Why is technology like OTOY not everywhere today?

The limiting factor facing many web technologies has been the difficulty of incorporating all the mechanisms needed to bridge the power of compiled applications with the 'thinness' of web media – and not being tied to any hardware or software base. The small portable OTOY runtime and authoring toolset (an OS in and of itself), the ability to run content as meta-apps on devices without the runtime, and the compression and scalability of content make OTOY uniquely positioned to dominate a space that currently has no comparable solution.

Outside of OTOY, certain pieces of this technology exist, but not all in one place:

Adobe Flash/Dynamic HTML+JavaScript

Flash, like OTOY, is 800k, and portable to cell phones. It has a great portable visual graphics library, which when combined with JavaScript and DHTML, results in very powerful and portable applications. But these are severely limited in sophistication, speed and flexibility and they do not have the compression that keeps OTOY and its content small. The current crop of OTOY applications: 3D games, alternative web browsing interfaces, media sharing, the ability to replace the OS and mix existing executables and media as imaging layers....all of this goes far beyond anything currently possible in DHTML/JavaScript and Flash. As of this writing, Adobe's 'Apollo' application (combining flash and desktop acces) has not yet been released. It remains to be seen how much of this model will be adopted in its feature set.

It is also worth noting that the X3D specification may help unify an open meta-format for high end 3D media (including shaders). The implementation, portability and consistency of runtime clients capable supporting this specification, as well as their ubiquity, will determine if X3D can enjoy a viability comparable to that of Flash or dynamic HTML .

Microsoft .NET + XNA/WGF + Windows Vista

.NET/WGF in Windows Vista is perhaps the most interesting approach being taken to address the 'webification' of traditional applications. The problem with the .NET framework, even today, is that it is huge – and not yet ubiquitously installed. It is about 80 MB (100 times larger than OTOY). The usefulness of .NET (and WGF down the line) will depend and how quickly it is adopted and how well Microsoft can abstract some of the more complex aspects of the runtime for thin devices (i.e. 3D rendering).

SUN Microsystems : Java

Java suffers from the variations and inconsistencies of its virtual machine across platforms (e.g. differences between mobile J2ME and PC Java VMs).

Overview of the Development Tools for OTOY

How is content created in OTOY?

The entire authoring system is built into the runtime player, giving every consumer of OTOY the same tools to build and edit content as the most prolific or advanced developer. The difference between the two is the digital signature key that accompanies the content they transmit to others. This signature determines the scope of what the transmitted content can do remotely and further authenticates its sources when content becomes modified sequentially by multiple users and redistributed.

Templates for Developers and Content Creators

Many of the OTOY features and API provided to developers as application templates (including chat, file sharing, web sharing, and P2P) are themselves built entirely in script and deployed using the very same system provided to developers and consumers for their own content. Besides demonstrating the flexibility and scalability of the scripting components in building almost any type of application or service, this process also results in these features being implemented in the fraction of the time it would take to do in C/C++.

Using only the tools provided in OTOY's basic scripting language, the existing toolset has the capacity to build almost any type of application. Developers of all kinds (i.e. C/C++, .Net, Flash, and HTML) can build services and applications in a simple framework that encompasses the syntax of those languages, but which also allows developers with almost no programming knowledge to build extremely complex games.

This model has been proven time and again with the simplicity of 3D Groove games done by first time programmers in a matter of weeks. Hiding the complexities of 3D programming (i.e. matrix transforms, vector / 3D math) and replacing them with an intuitive English-like syntax that allows developers to push, pull, and move objects without even thinking about what is going on behind the scenes is the key component to creating simple development tools, particularly in script. Even the verbosity of the function names is an important aspect of a script system for novice developers. While it is almost surreal for serious programmers to think this way, first-time game developers have done amazing things in script without having a deep programming knowledge of math or 3D, revealing the untapped potential achievable when script is made simple to developers who may not have such skills.

A corollary to the above: the script system in OTOY, while allowing English-like syntax to be used for building games and applications by novices, compromises *none* of the low level functionality of a language like C or C++. The parser in OTOY can read C/C++/JavaScript (and even some Pascal and VB declarations) alongside the simple script syntax. Every C operator is allowed in script, as are typed functions, structures and variables, along with more flexible untyped script counterparts. Adding functionality to the script itself, overloading operators (multiple inheritance of objects is possible, and definable at runtime too) and performing low level functions on sockets, memory buffers (for images, meshes, or anything else) is all possible in script. Protected memory spaces (which are slower than unchecked memory spaces) are allowed up to a 64 MB, and developers who have received a certificate for running compiled code may link in any DLL or compiled function in script and also use the full power of script as they would within a local secure application.

OTOY 2.0 extends the scripting language by including advanced rendering functions that are compiled on the GPU, yet operate transparently within general purpose scripts. While these features will certainly become the norm with Direct3D10 hardware, OTOY provides these features on Shader Model 2 cards (i.e. 2002 and newer). The shader syntax in script is presented as a unified model for programming the geometry, vertex and pixel pipelines of the rendering engine with the same flexibility and simplicity as general purpose scripts.

Although the OTOY system is fully independent of the OS, the API in script can still access almost any part of the OS, including any executable or library, and a wrapper is provided for transforming the interface to such external objects into native script objects, usually sprites or surfaces, that operate no differently than non-external media. This is an even more powerful feature with the added P2P server components built into the runtime library. A cell phone can connect to an OTOY client with PowerPoint installed, have the runtime on that machine actually run a PowerPoint document (it can be invisible to the user on that machine) and create an interface to the document, with full event synchronization, that works through the cell phone's web browser. The shifting of server and client functionality occurs transparently. The functionality of client and server, and even the space within the network where the application 'resides' is completely blurred. This is the likely future direction of web applications; developers building applications in OTOY, even if they are not fully aware of it, are automatically creating applications for a grid computing environment.

The entire file system in OTOY is designed to eventually be virtualized through the P2P network, and to amorphaously span many blocks of computers. As with BitTorrent, a file may be retrieved through its hash, rather than any static address.

Files and folders may be placed in virtual storage, not necessarily on any one server, but across a range of machines. The reliability of machines is flagged and tagged, and perhaps even charged for (by users themselves, or enterprising companies building on top of the OTOY system). Files are mirrored, and local copies stored when possible. File sharing networks like BitTorrent, and services like Photobucket.com have already touched on the possibilities of a massive 'internet hard drive'.

Companies are even seeding their products into P2P networks as a means of distributing software more easily than through a single server. The logical step is for end-users to stop thinking of files being stored in any one place, but rather have the files 'owned' by them and easily retrievable by any device they are authenticated on.

The retrieval and dissemination of data through the network is the first step towards the dissemination of code fragments themselves. Given the dynamic layer of the language and objects, it is very easy to pass along script fragments, as threads, parallel processes, templates of classes and functions and execute them 'virtually' through a part of the system. Thus, on top of a virtual 'Internet Hard Drive' we get a virtual 'Internet CPU'.

The limiting factor of grid computing models for mainstream applications seems to be the tools developers have to build them in the first place. For a system that is essentially defined by the connection of nodes, individual PCs (and web applications themselves) on the internet operate largely in isolation. The existing exceptions - chat, messaging, file sharing and massively multiplayer games - are all a core focus of what OTOY was designed to handle.

The above features are already implemented as OTOY components. But the entire palette of applications that are possible within such a context have yet to be completely formulated or explored. They likely will not be until a system, such as OTOY, is more widely used to merge disparate communal services and combine them in novel ways that are simply not available through existing web browsers and current OS APIs. The script and rendering system in OTOY is designed to address this as best as possible.

Summary of Key Technology Features:

- OTOY provides a simple, tiered, authenticated and uniquely powerful script and content packaging system that blends the line between content creators and end-users through a peer-to-peer mechanism.

- The small size of the engine and the simplicity of injecting it on an end user's machine remove almost all barriers that hinder viral distribution on a mass scale. A stub library needs to be downloaded just once for all other portions of the technology to work and update transparently.

- All OTOY consumers can build, view, combine, edit, and publicly or privately send, buy or sell data and services through the system. However, users are restricted to whatever limits are set by the author of the particular content or application being transmitted. This mechanism is enforced through a chain of digital signatures, whose root key is controlled by OTOY.

- By providing the technical infrastructure of this network and letting developers build securely on top of this layer, OTOY essentially becomes the de-facto operating system for its own content as well as the gateway through which all data in the system is authenticated, bought, traded or sold – regardless of any underlying hardware or software on the client.

- All OTOY content can automatically be scaled and reserved by any node within the P2P network so that the content can be viewed and used even on the most limited of embedded devices (such as older mobile phones) without the runtime client being installed on the target device.

- The basic building blocks of OTOY content - beyond 3D web games - include core application services for chat, messaging, document sharing, file sharing, audio-video media, linear 3D broadcast content, virtual 3D worlds and shared community spaces (2D and 3D).

Long Term Goals of OTOY

The emergence of web services and mobile devices poses a genuine challenge to the traditional deployment of applications. What is missing between these two poles has been a cohesive framework that allows developers to deploy applications with the complexity and richness of compiled executables (including photorealistic 3D games), while retaining the platform independence and thinness of a web application that is completely portable. OTOY bridges this gap.

The technology, viewed in its entirety, extends far beyond merely serving as a runtime engine for rich web content. Innumerable unexplored business models are possible through the novel mixing of thin applications, in a P2P system, with a rich API that makes no differentiation between client and server nodes - and has no dependency on the underlying OS.

The logical conclusion of a completely self contained platform such as OTOY is to provide a framework that entirely replaces the need for an underlying operating system.

The ancillary impact of such a system, one that is capable of streaming and rendering 3D assets with photorealistic fidelity on a majority of devices, is to efficiently export linear content in a way that challenges static formats currently employed in television, home video and downloadable video content channels.