Making Money With HTML5

Written By Matthew Bowden
Edited By Dexter Friedman

About The Author

Matthew Bowden is a self-employed writer and software developer. Matthew spends his days creating media content, aspiring to travel more, and answering far too many emails. He currently resides in Australia.

Foreword

I wrote this book to show game developers how to make money from their HTML5 games. *Making Money With HTML5* combines over a year of research and networking efforts, providing everything worth knowing about this new and lucrative market. I'll show you how to get started, where the money is, and how to avoid common pitfalls. Soon enough you'll be able to cut your own slice of the delicious HTML5 pie!

Completing this book has required a gargantuan effort from both myself and my excellent editor, Dexter Friedman. Together we've managed to create an invaluable resource packed with uncommon advice, experience, and knowledge. While this is a short book, it is also void of the usual filler content we're all used to scouring through. *Making Money With HTML5* will not waste your time.

On that note, let's get started.

Chapter 1 - Introduction

What Is HTML5?

HTML5 is an odd acronym that is difficult to explain plainly. For the sake of simplicity, let's just call it the next version of HTML with an extra large helping of CSS3 and JavaScript. HTML5 has a major drawcard that's all too uncommon within the wonderful world of web development: cross-compatibility.

HTML5 games, which are the focus of this book, can be played in desktop and mobile browsers without the need for an additional plugin, a specific operating system, or even a separate code-base! HTML5 attempts to introduce consistency to web-based media.

What does this mean for game developers? With HTML5 technology, users can play your game on their desktop computer at work, on their phone during transit, and on their tablet at home – seamlessly.

Why Bet On HTML5?

Commercial HTML5 game development is appealing for three main reasons: the technology is new and exciting, the functionality is unique, and there is demand without supply.

HTML5 is a fledgling technology compared to its main rival Adobe Flash, which currently dominates the market for desktop web media. Adobe recently withdrew official support for Flash on mobile devices, and has called HTML5 a *"game changer that is creating new markets."*

Despite this, very few Flash developers seem to be willing to make the transition. This leaves us with a brand new market that developers just aren't taking seriously, while progressive companies across the globe are paying out thousands of dollars for cross-platform HTML5 content.

How Much Money Can Be Made?

Since early 2012 I've been licensing HTML5 web games to companies worldwide, and have established a sizeable monthly income through my work. I started publishing a monthly online income report on my blog:
http://www.truevalhalla.com/blog

Listed below is a selection of my HTML5 earnings during 2012. All monetary values mentioned within this book are in US dollars (USD).

Month	HTML5 Earnings	Link
July	$6,000	http://www.truevalhalla.com/blog/?p=238
August	$6,346	http://www.truevalhalla.com/blog/?p=329
October	$4,989	http://www.truevalhalla.com/blog/?p=395

As you can see, there is a definite opportunity to make a few dollars here, or even a proper living. I know developers who have quit their jobs just to make HTML5 games, and I currently make the majority of my income with them!

With my experience and guidance you too can start making money with HTML5.

Chapter 2 - Preparation

Getting Started

Before you make your first dollar, or even your first game, you need to know what to expect from the path ahead. Certain limitations can be problematic if you're new to commercial game development.

For example, to sign contracts and join ad networks you typically have to be 18 years old. If you are not this age or older you might find it difficult to start Making Money With HTML5. You may need assistance from a parent or guardian to help conduct online transactions, if they are willing.

To accept payments from third parties you will need to join an e-commerce platform such as PayPal. However, even as the world's most popular payment platform PayPal is still not supported globally. Investigate whether PayPal is supported in your country, and sign-up if you plan to accept payments online.

Consider your legal obligations. It's useful to register a business to help validate your dealings, and while taxation law varies from country to country you should be fully aware of its implications. It's also useful to have access to a lawyer and accountant sooner rather than later.

What else might hold you back, or become a problem? Search for solutions before moving forward too quickly.

Editors & Engines

Although this book is not tied to any specific method of HTML5 game development, you should be aware of your options.

You can actually create HTML5 games entirely within a text editor. This is a popular choice for many developers because of the minimal costs involved. You can also find plenty of HTML5 frameworks which give you a foundation to work off. Some frameworks are free, and some are not.

Another option is to use a specialized game development engine. Engines provide you with built-in functionality that quickens and simplifies the process of making games. However, engines ultimately allow you less control, and can limit your monetization options. The good ones are normally not free, either.

Deciding between using a text editor or an engine is important. If you're not sure which option is best for you, then experiment until you find your preference. Popular text editors include Sublime Text and Notepad++, and popular HTML5 game development engines include GameMaker: Studio and Construct 2.

Your optimal development environment will depend entirely on your personal preferences.

Web Presence

Once your HTML5 games are finished they need to be placed on the internet so others can access them. They are web applications, after all.

If you don't already have a website of your own it is worth investing in one. A website allows you to maintain control of your work, centralize your content, and portray yourself professionally. You can buy shared web space or take advantage of free hosting services to host your website.

If you're looking for greater reliability or control, consider investing in a virtual private server (VPS) or a dedicated server. HTML5 games can receive thousands of plays very quickly if they are picked up by large websites. If you think you'll need it, invest in performance.

Listed below are reliable web hosts, and useful resources for learning about web development.

Paid Hosting	Link
HostGator	http://www.truevalhalla.com/recommends/hostgator
DreamHost	http://www.truevalhalla.com/recommends/dreamhost
TelVPS	http://www.truevalhalla.com/recommends/g/telvps

Free Hosting	Link
Dropbox	http://www.truevalhalla.com/recommends/dropbox
Weebly	http://www.truevalhalla.com/recommends/g/weebly
AwardSpace	http://www.truevalhalla.com/recommends/awardspace

Learning Resource	Link
W3 Schools	http://www.truevalhalla.com/recommends/g/w3schools
Tizag	http://www.truevalhalla.com/recommends/g/tizag
Mozilla	http://www.truevalhalla.com/recommends/g/mozilladev

For some hosting providers you'll need to download an file transfer protocol (FTP) client such as Filezilla or SmartFTP. These programs allow you to upload files to your web host, and will become vital to your workflow because you'll be uploading files constantly.

Going Mobile

When developing commercial HTML5 games, you should have a mobile-orientated focus. The mobile web is where HTML5 really shines, and it's where you'll make the most money. If you're only interested in desktop web media then you should be using Flash.

The major mobile platforms are Apple's iOS and Google's Android. On each platform there are different types of devices that your HTML5 games need to work on, primarily phones and tablets. These devices have varying resolutions and performance capabilities, and can run several different internet browsers.

To save time and effort, I recommend that you purchase a small variety of mobile devices.

Remember that HTML5 is a fairly new technology. You shouldn't expect your games to be playable on every single mobile device in existence. Build your HTML5 games for a modern audience, and don't waste your time aiming for perfection. You'll want your games to work properly on devices that are popular and have a large percentage of the marketshare – these are where most of your players are.

On iOS, your games should work on iPod Touch 4th generation and higher, iPhone 4th generation and higher, and iPad. A quick reference list of iOS devices can be found here:

http://en.wikipedia.org/wiki/List_of_iOS_devices

On Android, developers face the significant problem of market fragmentation. There are a vast number of different Android phones, each with their own performance capabilities, resolutions, and browsers. Take a look at this list to see just how many Android variants there are:

http://en.wikipedia.org/wiki/Comparison_of_Android_devices

As you can imagine, this fragmentation makes developing for Android quite an unappealing prospect.

I tend to focus on getting my HTML5 games to perform well on iOS – then if they run on Android it's just a bonus. While you can rely on iOS to be consistent, you can bet on Android to give you a headache.

Make An Investment

To establish an optimal debugging environment I recommend you purchase three devices: an iPhone (or iPod Touch), an iPad, and a recently released mid-high range Android phone.

This gives you access to a device on each major mobile platform with varying specifications. While these purchases may be costly, they will save you an incredible amount of time in the long-run. At the very least, buy an iPod Touch.

This is a field where investments really pay for themselves. You shouldn't be afraid to spend money in order to get a foot in the door. If you purchase a framework or engine license, web space, and mobile devices your investment can become quite significant.

My own setup costs totalled around $1,600 and I've spent more since then, but it all returned to me quickly. I spent $200 on my engine (GameMaker: Studio), $1,250 on devices (iPod, iPad, HTC Evo 3D), and around $150 on hosting (virtual private server).

While you can get away with being frugal, you should be prepared to pay for any miscellaneous costs that may arise.

Chapter 3 - Development

Choosing Platforms

HTML5 games work on a variety of platforms, but they don't always work well. Performance and functional incompatibilities are common issues. You need to know which platforms you should be focusing on so that you don't waste time catering to the flaws of the less important ones. If your HTML5 games don't work properly on the major platforms, you will significantly limit your audience, and your potential revenue.

When developing HTML5 games, the browser is king.

For personal computers, your games should be playable in modern versions of the most popular desktop browsers: Internet Explorer (55% market share, Jan 2013), Firefox (20%), and Chrome (18%).

For mobile devices, the browsers worth supporting depend on the aforementioned dominant mobile operating systems: iOS and Android. Safari is the most popular iOS browser (and commands 60% of the entire mobile browser landscape), while a variety of browsers fight for the top spot on Android (Firefox, Chrome, Dolphin, Opera.) I wouldn't recommend going out of your way to support any specific Android browser over the other.

If you want to add support for additional platforms, like Blackberry, Kindle, Windows Phone 8, or even consoles like the Xbox 360 or Wii U, these should be an afterthought.

Try to maximize compatibility: avoid using advanced hardware not found on all platforms, such as an accelerometer. Minimize your reliance on functionality that is not supported globally, such as local storage and multi-touch. The more platforms and functions that you choose to fully support, the more time-consuming testing and quality assurance will become. Before I ship any of my games I test them on four different devices and multiple browsers!

Design your games to work well on the platforms that are the most important to you and your audience. Newer platforms may sound exciting, but be wary: they are often minefields.

Resolution Management

Effectively displaying HTML5 games across multiple platforms can be a frustrating challenge for developers. The resolution difference between mobile phones, tablets, and desktop displays is often quite substantial.

On mobile, it's desirable to try and mimic the way native apps display in fullscreen mode. While fullscreen is not always the best solution to resolution management for HTML5 games, there are several ways to approach this display challenge. It should be noted that, currently, none of them are perfect.

The first method is to develop your game at a low resolution, and then scale up as necessary. My HTML5 games are designed more for a mobile audience than a desktop one, so I use a resolution of 320 x 480 pixels and scale up based on the resolution of the device the game is running on. This significantly reduces visual quality, and mildly reduces performance.

The second method is to create your game at a high resolution, and then scale down as necessary. This can be an issue on mobile devices that don't have the memory or processing power required to display high resolution assets. However, this method is a good option if you're only focusing on desktop computers and tablet devices. In exchange for significant performance loss you can maintain visual quality.

The third method is to include both standard and high definition graphics in the one game, and then display and scale the appropriate variant based on the user's platform specifications. While scaling will still result in performance loss, you can ensure your game looks good on any device without isolating low-end mobile devices. The main consequence of this method is increased loading and development times.

My recommendation is to create your HTML5 mobile games with a resolution of 320 x 480 pixels, and then scale up based on the resolution of the user's device. With a low resolution like this your games should run well on most mobile devices while remaining playable on desktop. You'll lose some visual integrity but on mobile devices this is not particularly noticeable.

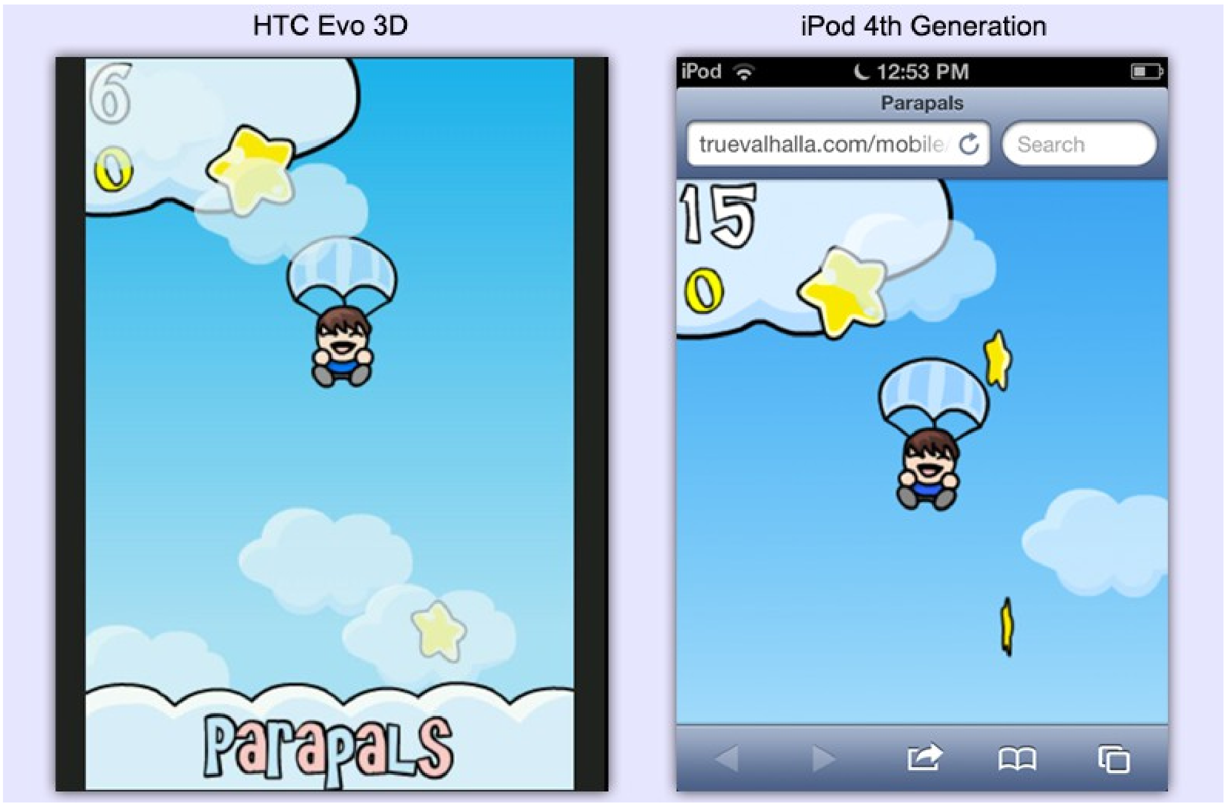
A resolution of 760 x 600 pixels is a good choice for HTML5 games that are designed for desktop, and this resolution also displays well on most tablets, websites, and blogs. Facebook now accepts HTML5 games on their distribution platform, and they require that games are not wider than 760 pixels.

Displaying your HTML5 games properly is very important, though striving for perfection is a lost cause. Some mobile browsers don't even return their correct dimensions all of the time! Plus, resolutions and scaling are just the beginning of your display concerns...

Browser Obstructions

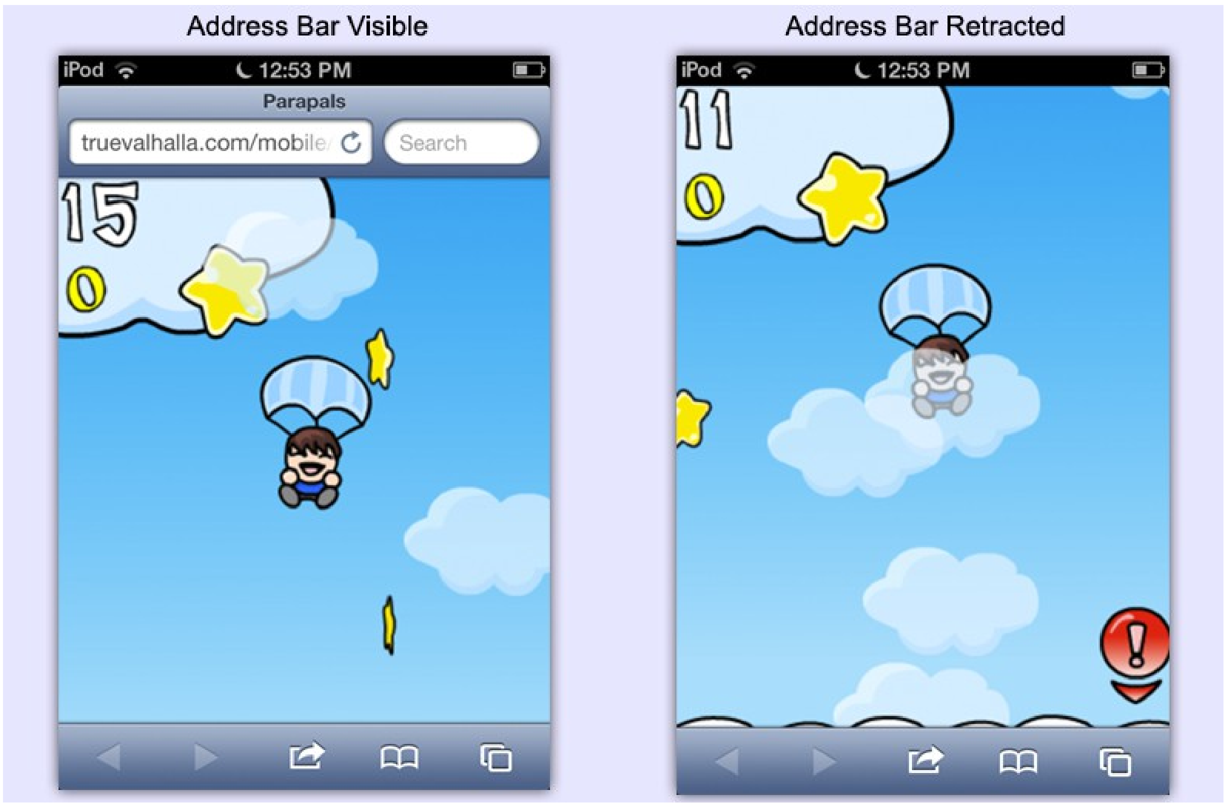
Since HTML5 games are played on the web, a browser's user interface (UI) can become problematic if it obstructs the view of your game. This is particularly common in mobile browsers.

The following is a screenshot of an HTML5 game running on an iPod, and an Android device:



As you can see, the default iOS browser, Safari, greatly obstructs the player's view of the game.

You can use JavaScript to remove the address bar, but it doesn’t fully solve the problem:



You'll notice that the address bar at the top of the screen has been retracted. However, the task bar at the bottom cannot be hidden. This problem occurs primarily on iPod and iPhone.

Luckily, there are ways to reclaim the screen real estate lost on these iOS devices.

One solution to regain lost screen real estate on iOS is to force players to add the game to the home screen of their device. This action must be performed manually by the player, from within Safari itself. Doing so allows the game to be launched from the home screen in fullscreen mode, as if it was any other app. However, some players may find it irritating to manually install an app to their home screen in this manner.

Another solution for this problem is to simply design around the Safari task bar. In my HTML5 games, I define a region at the bottom of the canvas which is the same height as the obtrusive task bar. When the game is played on an iPod or iPhone, this region will be obstructed from view by Safari. The remainder of the canvas can then be used for the actual game.

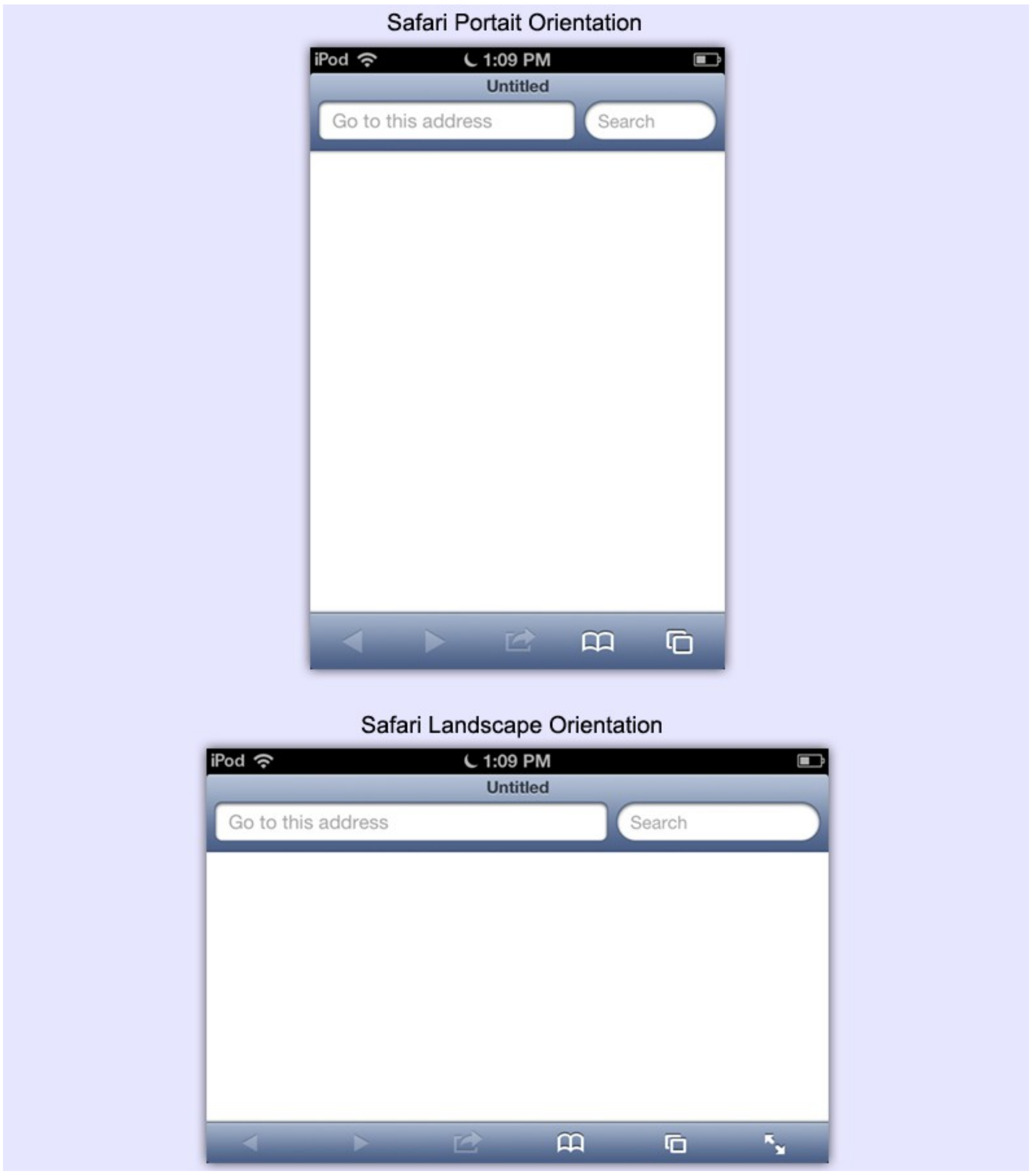
Since the region is still visible on other platforms, it's worth filling it with more than a solid color. Trying adding the name of the game, a copyright notice, or a nice graphic to populate the space.

Device Orientation

Native mobile apps (from the App Store or Google Play) can lock the screen orientation to either portrait or landscape, but this cannot currently be achieved with HTML5. This becomes an issue when a user turns their device to the side and the canvas doesn't turn with it. This issue is only pertinent to mobile devices, because desktops and laptops typically don’t have orientation changes.

Your game should constantly check whether a device has been turned. If it has, the user should be told to return the device to the correct orientation. These instructions can be relayed by use of text or graphics, or both. If you're feeling particularly creative, you could utilize both orientations within your game. In portrait mode the player might see the normal game world, and in landscape mode they might see a map of their current level.

Within Safari, the browser's UI also looks different based on the orientation of the device:



I recommend that you address the display issues discussed in this chapter, although doing so is entirely optional. Games that handle orientation changes properly and maximize screen space tend to look more complete – the more “professional polish” your game has, the more likely it is to make money.

Non-Standarization

HTML5 is far from perfect. Sometimes your games will run slowly on specific mobile devices, no matter what you do. Sometimes your graphics won't display properly, no matter what you do. Sometimes audio won't play, no matter what you do.

These issues are due to a lack of standardization. A recent update of iOS introduced the Web Audio API, which greatly improves support for sound within Safari. However, until HTML5 audio is supported by more browsers it's a safer bet to avoid including audio in your games.

By differentiating between problems that you can and can't fix, you'll find HTML5 game development much less frustrating. As with any new technology, there will be a few initial hiccups that you’ll have to deal with.

Outsourcing

As a GameMaker: Studio user, it was difficult for me to address some of the issues discussed in this chapter. I had little control over how my HTML5 games would be displayed, especially on mobile devices. So, I contacted a capable JavaScript programmer and had the “Mobility Engine” created. This engine substantially extends GameMaker's functionality by introducing flexible scaling, orientation management, browser interface control, and more. Instead of waiting for the program's developers to add the functionality I needed, I simply outsourced it.

If you want to add a professional touch to your games, you can outsource artwork, audio, or anything else you need. For a small, high quality HTML5 mobile game, my budget for art work is typically between $200 and $500. You can also source assets from vector stock art sites for very low costs.

Whether it is code, art, or whatever else, always acquire full rights to the final product when outsourcing. This allows you to monetize the outsourced work if others have an interest in it. I license the Mobility Engine to GameMaker: Studio users who are interested in displaying their games nicely across mobile phones, tablets, and computers. The market for this functionality represents just one potential source of HTML5 income.

Chapter 4 - Monetization

Commercial Content

Monetization is one of the most interesting aspects of working with HTML5. The market is so new, so open to experimentation, and so full of opportunity. There is a lot of money to be made.

HTML5 games can be commercialized in a variety of ways, but the common methods of making money with HTML5 rely on the concept of a publisher. Unless you're already able to drive hundreds of thousands of users to your games, publishers are the key to your success.

In this context, a publisher is a representative for a company or website that wants to distribute your content. Publishers can typically drive an incredible amount of traffic to games that they acquire. My own portfolio of HTML5 games received more than a million plays on portals during 2012, generating around $34,000 in base revenue.

Some publishers will pay you for the privilege of distributing your games, and some will offer you a share of the revenue generated from distributing your games. Either way, your games get played, and you get paid.

What's the catch? Well, publishers stand to make money from your games too. There are many different ways a publisher can monetize quality HTML5 content, but often it depends on what type of commercial arrangement you've agreed to.

There are two main options: sponsorships and revenue shares.

Sponsorships

Sponsorships are a popular option between publishers and developers alike. The publisher will pay you a set fee for the right to use your game. What that specifically entails depends on the terms you set, or the contract involved. Normally, the publisher will simply host your game on their web portal.

There are several types of sponsorships: *exclusive*, *non-exclusive*, and *rental*.

If you sell a publisher (also called a “sponsor” in this context) an *exclusive* license to one of your games, you are not allowed to offer that game to any other publishers, or distribute it beyond your own website. This arrangement is appealing to publishers because it weakens their competition, and allows them to monopolize a game's success. Exclusive sponsorships are often reserved for only the highest quality games.

More commonly, you will want to offer publishers a *non-exclusive* license. This type of sponsorship allows you to license your game to as many publishers as you wish.

While exclusive licensing may result in one big pay day, non-exclusive licensing allows you to earn over a long period. This is an important distinction. Currently the HTML5 market is so young, and it may not be effective to offer exclusive licensing to your clients.

Since it is so difficult to judge future demand for HTML5 content, I would strongly recommend that you avoid exclusive licensing, and instead opt for non-exclusive or *rental* sponsorships.

Rental sponsorships are not common, but they are worth mentioning. Sometimes a publisher will not want to pay a hefty sum of money for a permanent license, and will instead prefer to pay you on a monthly basis for the right to use your game. If the game is not popular, they can cancel the rental agreement and stop paying you. However, if the game performs well they may maintain the rental agreement for a long period of time.

Several of my games have been in rental agreements with one publisher for many months now, and they've earned far more than I would have charged at the time for non-exclusive licensing. Sometimes you will be given a choice between a rental sponsorship and a non-exclusive sponsorship. Determining which option is best for you is, well, up to you.

Each of these types of sponsorships has the potential to make a lot of money. If you sell a game with an exclusive license, you can afford to charge a higher price. You will not be able to make money from that game once you've sold it, so publishers understand that exclusivity entails a costly investment on their part. If you sell non-exclusive licenses, you have to charge much less, but you can approach publishers now and in the future. Or if you establish a good rental agreement, you might still be making money with HTML5 for decades to come!

While the vast majority of my own HTML5 earnings have come from non-exclusive and rental sponsorships, some publishers do not offer payment directly. Instead, they will try to entice you with revenue shares.

Revenue Shares

For every company that is willing to pay you cash in hand, there are many more that will offer you a revenue share. Typically ad revenue is the monetization component in such an agreement. A publisher will either have ads on their website or they may ask you to integrate ad code into your game. You will then receive a share of any revenue generated.

Now, this doesn't sound too bad at first, but in reality, revenue shares are rarely worth your time. Not only does the publisher get your game for free, but you have no guarantee that you'll even make a dollar. Most of the time you won't.

The vast majority of publishers that offer revenue shares are utterly incapable of delivering quality traffic to your content.

Think about it for a moment. If a publisher is willing to pay you upfront for your game, then they are confident they can drive so much traffic to it that they'll make their money back, plus a nice profit. When a publisher isn't willing to pay you an upfront fee, it's unlikely that they actually have the means to properly monetize your content.

That isn't to say that *all* revenue shares are a waste of time. Occasionally, perhaps to mitigate risk, well-established publishers will offer revenue shares to developers. A revenue share with a good publisher has a lot of commercial potential.

I know developers who are making hundreds of dollars per game, per month, just from ad-supported HTML5 games that have been distributed by a competent publisher. The difficulty is separating the few good publishers from the many bad ones.

Once you do find a good revenue share to take part in, your monetization options are not limited to just ads either. You could also integrate in-app purchases, a subscription model, paid offers, and more. This version of *Making Money With HTML5* will not discuss these revenue options in-depth, however future revisions of the book will provide further insight as more data becomes available.

Finding suitable providers for any of these commercial services can be challenging given the budding state of the HTML5 market. However, there are a few companies that are making an effort to push the industry forward.

Advertising Solutions

HTML5 ad servers will only monetize your mobile traffic; they don't typically allow the use of ads within desktop applications. Remember, mobile is where you'll make the most money.

Listed below are several well-established HTML5 advertising servers.

Provider	Link
LeadBolt	http://www.truevalhalla.com/recommends/leadbolt
Smart AdServer	http://www.truevalhalla.com/recommends/g/smartadserver
Google AdSense/AdMob	http://www.truevalhalla.com/recommends/g/admob

My personal preference is the excellent service called LeadBolt. They provide a range of advertising options for both native and web applications, all of which are easy to customize and use. I've been impressed by all aspects of LeadBolt's service.

I've listed Google AdSense only due to its popularity. Google recently banned the account of a high-profile HTML5 game developer, costing him a very significant amount of money in the process. The developer's implementation of Google's HTML5 banner ads was legitimate, and yet his account was closed without warning. It's unlikely an appeal will be granted.

My own Google AdSense account was closed many months ago under similar circumstances, however it only cost me my account balance of $160. I got lucky, believe it or not. Since transitioning to LeadBolt I have had no similar issues.

I strongly recommend LeadBolt over Google AdSense as your HTML5 ad server of choice. LeadBolt deliver excellent earnings, serve quality ads, and maintain an open line of communication with their clients.

Once you've decided on a provider, you can begin the process of integrating ads into your HTML5 games.

Formatting Ads

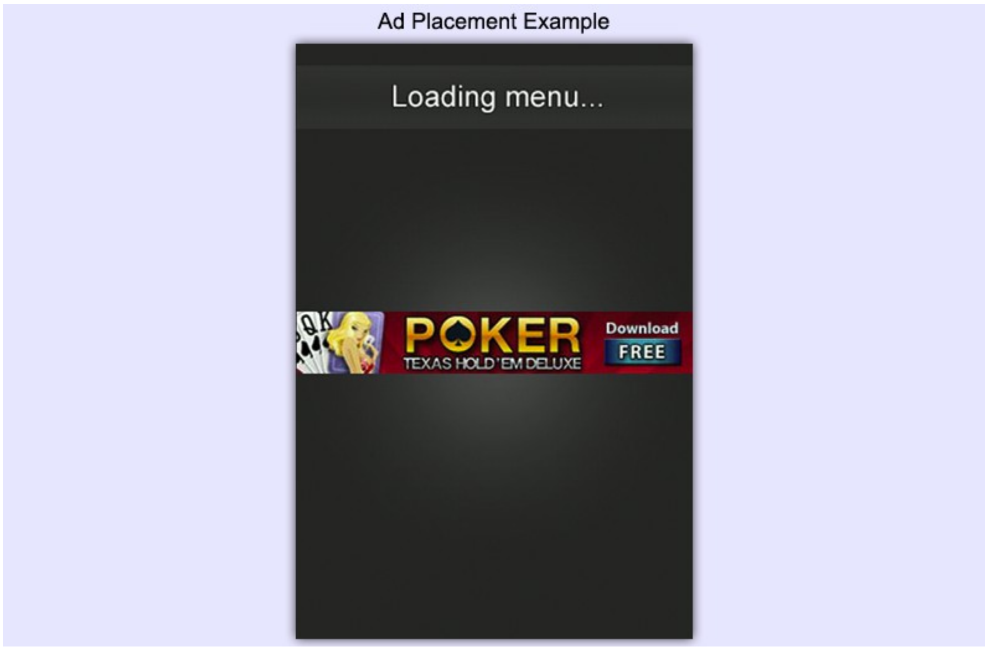
Optimal placement and formatting of ads is an integral part of successful advertising, so much so that there are entire books dedicated to the subject. However, the basic concept is that you need to get users to click your ads without actually asking them to.

I find that image banner ads are far more effective in games than text banners are, so I'd recommend setting your ads to only show images. You should experiment and see what works for you, though. In my HTML5 games, I'm seeing an average click-through-rate (CTR) of 12%, and I'm making around $4 per 1000 plays (CPM) with my ad provider, LeadBolt. Both of these values are quite high.

My ads are displayed in a way that is meant to mildly irritate the user, without actually frustrating them. Developers often try to be subtle with their ad placement in an effort to not upset their audience. Perhaps that is important for a native app where you have to worry about ratings, but with web games you can take off the safety gloves and make a real effort to monetize your traffic.

Ensure that your audience actually sees your ads, and has a reason to click through.

Within my HTML5 games, I prominently display a mobile banner ad on a fake loading screen for 5-10 seconds before showing the menu. Whenever a player returns to the menu they have to go through this page again. Additionally, the player is redirected to the ad page at the end of each game round. It's not subtle, but my ads are getting the attention they need. They are centered, and there are few other distractions on the page.



While you don't want to frustrate players to the point where they quit, you do have to convince them to leave your game. That's not the easiest mindset for most developers to get into. If you want to make serious money with ads, you need to be aggressive.

As a result of my own aggressive ad placement, I've been seeing impressive earnings from my ad-supported HTML5 games. The difficult part was finding publishers who could send players my way.

Finding Publishers

The most common question I'm asked about commercial HTML5 game development is “How do I find publishers?” Not only is it a difficult question to answer well, but it's one I tend to avoid answering at all.

Consider what you know about the HTML5 market. Which companies have the most influence? Which HTML5 titles are the most popular right now? Can you name five developers that are actively monetizing HTML5 games?

If your mind drew blank on any of those questions, that's entirely understandable. Currently there are few resources available about HTML5, yet alone the commercial aspects of it. This book is probably the first to address HTML5 monetization with any form of depth!

The general lack of information is due to the size of the market. This is a very small, niche field to work in at the moment – but it won't be like this forever. Every new developer that joins the scene becomes a new competitor, and as such, information about publishers is typically not shared with the masses.

While I won't be sharing my personal contact list in this book, I will show you how to build your own. Developers that put in the effort to build their own contact list will ultimately find more value in their business dealings than they would if every developer knew every publisher. Publishers are not as difficult to find as they used to be, it just takes some effort.

Start by doing what you do whenever you want to find something online: visit a search engine. With a few specific queries, I found my first publisher back in early 2012. Search for phrases like “buy html5 games” and “html5 game sponsor.” While you're at it, set Google Alerts for these types of phrases; you'll automatically receive an email whenever Google indexes content that mentions them.

Next, search for existing HTML5 game developers – preferably independent ones with a commercial focus. Find out what their games are called. Use this information to refine your search, and discover where these games are hosted. A site hosting one of these games may be owned or supplied by a publisher, so find a contact form and ask for details. A website I contacted once gave me the emails and names of several publishers even though they were not one themselves.

You should have a small list of prospective publishers by this point. But you can do more. Take the fight to the developers themselves! These are the people who know the names you're looking for, after all.

Searching the social media profiles of independent HTML5 game developers is an excellent way to find publishers. Many people follow their business associates (and competitors) on Twitter, and since this information is publicly available you can manually search through a developer's profile to find new leads. If they have a public Facebook profile, you can search through the pages they follow. Similarly, business networking sites like LinkedIn provide you with an opportunity to see who is connected with who.

You don't have to limit yourself to stalking social media, though. Send a friendly email to any HTML5 developers you find and strike up a conversation. If they grace you with a reply, tell them about what you're trying to achieve, and ask them to help you out. If you're lucky they might throw you the names of a couple of publishers. Don't be hinderance; be polite, and patient.

Using these methods will lead you to publishers, however you should also give publishers a way to find you.

Your social media platform should promote your interest in HTML5 games. Add a HTML5 hashtag to your Twitter profile, use the HTML5 logo where you can, and mention HTML5 in as many places as possible. Get involved in relevant forums and discuss HTML5 with other developers. Comment on popular blogs about HTML5. Start your own blog or website based around HTML5, and use it to promote your own portfolio. You need to establish yourself as an authority figure within the HTML5 game development community, if possible. There's no better time to do so!

To further increase the chances of a publisher finding you, promote your content at every opportunity. Submit your HTML5 games to sites like Kongregate and Newgrounds, and any other website that can increase your exposure. Ensure that any publisher who plays your game has a way to contact you.

While there are currently no *well-established* HTML5 market sites in existence, these will appear over time. When they do, the process of finding publishers should become much easier. Recently http://www.fgl.com (Flash Game Licenses) began accepting HTML5 games, a promising step towards establishing a proper market for HTML5 sponsorships. However, until there is a properly centralized marketplace for HTML5 content you'll have to search for publishers manually, and interact with them personally.

Communicating With Publishers

The idea of talking with a representative for a large corporation can be quite daunting. I recently had a conference call with representatives from internet giant Amazon. I was a bit nervous, to say the least. For the most part, you will communicate with publishers through email or instant messages.

Communicating with publishers professionally quickly becomes second nature, but not everyone has had practice writing emails to prospective business associates. It requires a certain finesse.

Your goal should be to sound like a professional without overdoing it. To achieve this, your writing needs to be clear, confident, and to the point. Many of the people you'll talk with receive dozens of emails per day, each vying for their attention. Keep your emails short and on topic, and always be polite. You don't want to sound rushed or arrogant.

Start with a basic formal greeting. Briefly introduce yourself and describe what you can offer. The publisher is looking for HTML5 content, and you can supply it – keep it that simple.

Provide links to your games, and optionally a brief single-sentence description for each. If your games are well-designed you shouldn't need to provide any instructions on how to play. Attach screenshots of your games if they have appealing visuals.

Finally, ask the publisher to reply if they are interested in setting up a partnership. By hinting that you're expecting a reply, you're more likely to receive one. Conclude with a pleasantry and generic sign-off.

I've included an example of an email you might send to a prospective publisher:

```
Hello John,

I heard you're looking to license web content. I'm a developer seeking
non-exclusive sponsorships for my newly released HTML5 mobile games. You
can play them at the links below.

[list of links to games]

I hope you'll be interested in my work. I will eagerly await your reply!

Have a nice day.

Regards,
Matthew
```

Be sure to sign-off with your real name – you're not *xDestructor128x* when you're conducting business. I try to address my contacts by their first name in every email I send them, too. You should also avoid putting a price on your games at this point. If you're ignored, at least you'll know it wasn't due to overly aggressive pricing. Seek an expression of interest before naming your fee.

If you don't receive a reply within a day or two, do not assume that you've been rejected; many publishers are very slow to respond. If you haven't received a reply within a week you should send a follow-up email. Continue to send a new email each week until you get a reply. Persistence can work, but you need to subtle about it. You should not show your frustration or anger. Try something like this:

```
Hello again John,

I sent you an email last week about my HTML5 games, but I didn't hear back
from you. Have you had a chance to try them?

Regards,
Matthew
```

Follow-up emails should be extremely brief. By concluding with a direct question you increase the chances of getting a reply. If nothing else, it encourages the publisher to dismiss you with a short response.

It can actually be quite difficult to get replies from publishers, even when you've been working with them for many months. I've found that grouping multiple games together increases the chances of getting a timely response. Once you're established with a publisher, try to avoid sending them an email until you have two or three new games ready to ship. This saves the publisher time as they can send fewer emails, and fewer individual payments.

Getting Paid

When a publisher expresses an interest in any of your games you'll need to determine a suitable price to charge for licensing. The minimum fee I currently accept for a non-exclusive license is $400, the most I ever charged was $950, and the average is approximately $550. While I don't offer exclusive licenses to publishers, I would currently charge between $4000-$5000 for one. Rental fees are normally non-negotiable, but $50-$75 per month is an average rate.

You can (and should) set different prices for different publishers. Charge large companies more, especially if they're based in Europe. A quick Google search can give you with an idea of how well financed a company is, and where they are situated. Use this information to determine a suitable value for your work.

Negotiating a fee requires good judgement. With some experimentation, you'll be able to determine a publisher's range, and charge accordingly. Always overcharge slightly in case the publisher is willing to pay more than you expected. You should be willing to lower your price if asked to, however you must have a limit. If a publisher knows that you consistently falter on pricing they may take advantage of you.

HTML5 games are currently worth a lot of money - keep it that way by maintaining high prices for quality content. Sometimes it is worth throwing away a sale in order to protect future business. I could sell my games for $150-$300 per non-exclusive license to some publishers, but for the sake of maintaining the value of HTML5 games I do not. Only accept a fair price, and do not settle for less than what your content is worth.

If you manage to establish a fair rate, you're on the home stretch. Unfortunately, the process of actually getting paid can be the most tedious part of all.

You may be paid a percentage upfront, entirely upfront, or entirely on delivery. Sometimes you won't have a choice, but you should always request upfront payment. If you're paid upfront, you can maintain control throughout the rest of the transaction. You don't have to bend to the publisher's will if you've already received your payment.

Normally you'll accept money in one of three ways: PayPal, wire transfer, or check. I prefer to be paid with PayPal because payment is instant, and PayPal stores individual currencies separately. Beware of exchange rates! If you receive a wire transfer payment in Euros, and you live in America, that payment may be automatically converted to US dollars. It is better to be able to choose when to convert the payment into your local currency so you can get the best exchange rate possible. Use a universal currency converter to monitor exchange rates (http://www.xe.com/ucc).

If you're receiving payment on delivery, it can take up to 90 days to be paid. This is just a standard practice that some companies adopt, and will be highlighted in any contract you sign.

Not every company will ask you to sign a contract, and whether you require one to do business is your decision. If you don't have a contract, and the publisher uses your games in a way you didn't intend, there's not much you can do.

When a contract is required, you should obviously read it very carefully. Don't be afraid to ask the publisher for clarification of any clauses that you're uncomfortable with. Consult a lawyer if you feel the need to.

Once you understand what you're agreeing to, print the contract, sign it, scan it, and return it in PDF form. It's worth buying a good printer-scanner around the time you begin approaching publishers.

In addition to a contract, sometimes you'll be asked to submit certain documents for taxation purposes. Taxation law varies between countries, though usually publishers will be clear on which documents they need, and how you can acquire them.

Certain documents are only available to people who own, or represent, a business. It cost me nothing to register my business, but there may be fees involved depending on your country's bureaucratic processes. A quick Google search like “how to register a business in [your country]” can provide you with a starting point if you find this to be a necessity.

The last step of this process is to deliver your game to the publisher.

My method for this is simple. I compress my game and its assets into a folder, upload the folder to Dropbox (http://www.truevalhalla.com/recommends/dropbox), and then provide a link to this folder to the publisher. Dropbox allows me to modify the contents of the folder before the publisher views it, if necessary. You could just send an email attachment, but this offers you less control than a link to a dynamic folder does.

Sometimes you'll be asked to provide more than just the game files. I've been required to supply icons, screenshots, and even original promotional materials on occasion. You can charge for this, if you wish. I tend to work it into my fees without drawing attention to the extra hassle. All images should be formatted as PNG, unless otherwise stated.

Another common request from publishers is to integrate an application programming interface (API) into your game. Typically API's give you access to functions that tie into the publisher's platform. For example, an API may be used to send highscores to the publisher's database, or to validate user sessions. The implementation of this type of functionality can be quite complicated, so be sure to charge extra for API integration. I've recently adopted a policy of charging a $400 one-time fee for API integration.

The overall process for defining your fee, getting paid, and providing deliverables can be quite daunting. At first, you'll find it tedious, but over time the process actually becomes enjoyable.

What Type of Games To Make

With an understanding of how to monetize HTML5 games, you might now be wondering what types of games to actually make. Developing commercial titles can take weeks or months, so you need to ensure that your games will actually be of interest to publishers.

Each publisher has a defined audience that they're targeting, and you need to provide content that will appeal to this audience. You're not going to sell a brutal top-down shooter to a publisher that specializes in games for toddlers, for example.

To play it safe, you should aim to create family-friendly casual titles. They don't necessarily need to appeal to any specific demographic.

You can make games in any genre, though puzzle and arcade games are popular options. Take inspiration from the casual Flash games market if you're finding it difficult to come up with ideas. Focus on simple concepts that are highly accessible, and avoid use of blood, language, and excessive violence.

If you know a publisher has an interest in a certain genre you can create content specifically for them, and ask for a high price in return. However, speciality games reduce your overall reach. Gambling games, sports games, social games, and so-called girl games (a legitimate niche) all have limited general appeal, but they may fetch higher prices from certain publishers.

In terms of quality, there are so few HTML5 games in existence that you don't need to push the envelope. If your game has appealing visuals, decent gameplay, and displays nicely on mobile devices, you shouldn't find it too difficult to score a few sales. Even the most basic games have a place within the current HTML5 games market. At the moment, a game made in 3 months has only slightly more earning potential than a game made in 3 days!

When trying to decide what kind of game to make next, keep one thing in mind above all else: design for publishers, not players.

Chapter 5 - Conclusion

Closing Thoughts

HTML5 has a very bright future. Support for the technology is growing, and massive companies are taking a close interest in it. Game developers at the forefront of the market stand to benefit from the HTML5 movement the most. Build a strong portfolio now, and prepare for the inevitable demand for cross-platform HTML5 games.

In this book I've tried to highlight the concepts and processes that led me to find success with HTML5 game development. I truly hope that *Making Money With HTML5* has provided you with some insight into this line of work.

Despite the mobile-orientated, publisher-driven focus that I've encouraged, there is more to *Making Money With HTML5* that is yet to be discovered, let alone discussed. Future revisions of this book will include additional information on more aspects of commercial HTML5 game development.

If you found this book useful, please consider sharing it through social media or recommending it to anyone who might have an interest in *Making Money With HTML5*. The following short link will always lead to the book:
http://www.bit.ly/HTML5eBook

This version of *Making Money With HTML5* is one I created specifically for torrent sites. If you've downloaded this book without paying for it, that's OK. I don't mind, really. If you found the contents of this book useful, and are in a position to support me, please consider purchasing the book using the URL above. As a self-employed content creator, products like this put food on my table.

But whether you have paid for the book or not, I'd like to thank you for reading. I hope it was a good read.

Contact

I'd love your feedback! Let me know how I can improve, and what you found useful in this version of *Making Money With HTML5*.

Please contact me using the information below if you have any comments, questions, or concerns. If you find a publisher, and would like to know my opinion on them, just drop me a line. I've had experience with the vast majority of them, and I'm glad to offer my advice to help you get a fair deal.

Email: matthewbowden@outlook.com
Twitter: http://www.twitter.com/TrueValhalla
Facebook: http://www.facebook.com/TrueValhalla

Special Thanks

Dexter Friedman	Graciela Kosinski	Jacob Harris	Charlie Saunders
Luke Williamson	Moritz Voss	Pauli Jokela	Joseph Dailey
Israel Nava	John Jones	Eamon Price	Loren Costet-Lahmar
Cameron Hutchison	Marcin Pierzchała	Kleber Neves	Gerred Dillon
Katrina Guillermo	Juan Rivera	Juan Rodríguez	Richard Youngblood
Jack Hood	Dave Homan	Wouter Alberts	Lee Horobin
Thomas Haaks	Júlio Rodrigues	Zachary Helm	Kyle Sloka-Frey
Christopher McDermott	Trevor Hanes	Nikhil Hemanth	Wisnewski Software
David Nolan	Jakub Cech	Martin Winkler	Santiago Garcia Mazariegos
Colby Terry	JNM Statham	Willem Pretorius	Gary Murphy
Rocky Mountain Biosystems	Xavier Laumonier	Aditya Tedjamulja	Raphael Corbin
Aditya Dharma	Hanno Braun	Morne Booysen	Batuhan Çetin
Tristan Hall	Chris Glover	Stephan Weiss	Daniel Newman
Darijo Bakulić	Seymour Dijkman	Roberto Alvarez	Alex McCullie
James Suggs	Jesse Venbrux	Shaun Spalding	Morten Schouenborg
Euco Games	Ben Cochrane	Marco Carrillo	Gary MacDonald
Dan Phillips	Ian Wong	Gamma Web Limited	Alexander Arthur
Christopher Sanyk	Samuel Batista	Dustin Hubbard	David Bell III
Itay Avigdor	Laurane Parris	Emmanuel Barroga	Corey Nieves
Daniel Rihoy	Daniel Schemann	Khyle Dean	Cory Rammer
Jason Streby	Daniel Neil	Barry Grub	Will Murphy
Mark Lambden	Mike Lawrence	Constantin Graf	Jeremiah Goerdt
Matthew Humphries	Arik Chadima	Justin Center	John Leffingwell
Jakub Joras	Hugh Compton	Nathan Deane	Thongrop Rodsavas
Diane Mueller	David Batty	John Bussell	Christian Wisniewski
RAM Technologies	Michał Wróblewski	RapidFire Studio	BRS Group Limited
Alec Armstrong	Didier Bieuvelet	Andrew Cheek	E Gomez
Philip Watson	Luzinete Sales F De Lima	Alex Fernandez	Nathan Hayes
Igo Liebig	Kenneth Burkhart	Gabriel Beckett	Matthew Greer
Timon Knigge	Jerome Paul Esteban	Travis Robbins	Elle Chen
Rob Buckley	Aden Humbert	Javier Chavez	~