tysdag 11. mai 2010

73 tips for bringing your game to the web!

GDC Canada

tysdag 11. mai 2010

* Me - Games industry since 95, Bullfrog, Core Design, DICE.
- Used to boxed product until I joined DICE as a Development Director early on Battlefield Heroes. Took the game and its web platform to commercial.
- Now believe this is the one true way of developing and publishing games.
- Quicker turnarounds to get your game out
- It is fun! :-)

* Tore - could not make it today, so it will be just me presenting.

Together with Mike Man, we Have founded...
* Vostopia.com - Building an avatar system for free to play 3D games. Integrated marketplace and portal.
The following are tips we have learned and are learning along the way. We hope they will serve as a good guideline and as a checklist for your progress. I aim to cover quite a lot of ground, and won't

Service

- Think service, not product
- Diversify, read, learn, listen and play

vostopia.com™

tysdag 11. mai 2010

* Service, not product – web games are always online, get you direct feedback from users, and most importantly – the work really starts when the game launches. The entire team need to get this under their skin. The rest of the talk will go into a bit more detail on several of the implications of this.
* Diversify, read, learn, listen and play – There is a lot of ground to cover, and you can not hire specialists for every field. Make sure your team read up and learn about the myriad of options, and most importantly – play!

Staffing

- Get good web people
- Embrace web dev mentality
- Keep core of team on for live dev

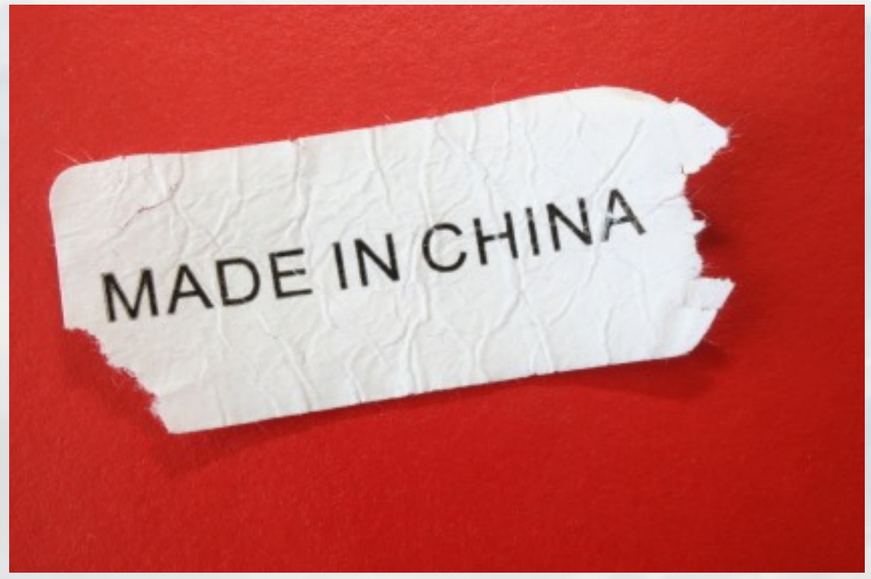
vostopia.com™

tysdag 11. mai 2010

* Get good web people - At the very least get one. Web dev and game dev have until now been two different worlds, so get someone who knows how the web world works.
* Embrace web dev mentality - Shorter development times, early launches, frequent updates. These all apply equally to web/online games, but can be hard for established game development teams to get into.
* Keep core team on for live dev - In the old model of standalone retail games, development often is fire and forget. Once the game is in the shop (and has had its 0-day patch), responsibilities for sporadic updates get handed over to a separate live team. With web games, launch is only the beginning, so it's more important than ever to keep continuity on the team for all the core areas.

Outsource

- Have the capabilities in-house
- Keep the same team
- Choose a partner who know their stuff & dare to say no



vostopia.com™

tysdag 11. mai 2010

* Have capabilities in-house - Outsource for extra capacity, not for work you haven't got the ability to do or at least oversee yourself. You need to follow up closely and have to be prepared to do some finishing on the work returned. Even if you are working with a very good outsourcing team, it's hard to get to 100% when not sitting in the same place. You can get very close though.
* Keep the same team - Unwanted changes on your team can be disruptive. This goes for the in-house and out-of-house parts of your team. Try to achieve a steady flow of work for a team that's ideally smaller or the same size as your in-house team.
* Choose a partner who know their stuff & dare to say no - It goes without saying that the outsourcing team should be qualified. It is important that they also dare to challenge your instructions. They are specialists after all, and might have insights you don't possess.

Game Technology Choice

- Accessibility is king
- Keep initial download small
- Avoid new browser plugins
- Do as much as possible in the browser

vostopia.com™

tysdag 11. mai 2010

*Accessibility – web users impatient, easily started from web page is important, gradual registration after play, count your clicks and waiting carefully. Make very sure it adds value.

*Keep initial download small – goes with previous point; stream content; download only the crucial parts for each step

*Avoid new browser plugins – If you can do with 2D or very simple 3D, use flash. 95% install base of flash 10. For 3D, Unity seems to be winning. Do not roll your own – it will have security issues. Imagine the news stories. Users may not be able to install the new plugins. Internet users have been trained to distrust new plugins.

*Do as much as possible in the browser – Users are familiar with the interface. They multitask efficiently and share links etc. Browser side is cheaper to develop for and has better tools. Ideally only game itself uses game tech. Leaderboards, lobbies etc can be delegated to the browser.

Browsers

- HTML is a cheap and powerful rendering engine
- Javascript is your friend
- Use partial updates

tysdag 11. mai 2010

* HTML cheap and powerful rendering engine - Html+css is a very powerful layout language. Use it for as much of the UI as you can. It is also supremely flexible and doesn't require patching to update. It will also make it possible to link into your UI so players can post links to high scores, profiles etc to social networks. It is also cheaper (but not free) to develop for than traditional game UI.

* Javascript is your friend- Javascript is a cool language. Unfortunately it has been dragged into the mud by different implementations in different browsers. Use one of the modern js libraries - jQuery, modtools etc. jQuery has a host of plugins to simplify js develpment significantly.
- Use js to render html on the client. Build a more responsive site and remove load from your servers by sending data directly to the browser and letting the browser filter and convert the raw data to html. The players all have powerful computers. Use theirs instead of your own.

* Use partial updates- Use jquery.address to minimise (but still allow) page refreshes. Web games are more web applications than websites. You don't want to lose game client state by refreshing the whole page. Use Ajax techniques to refresh parts of the page, and by using anchor-tricks as provided by the jquery.address plugin you can allow users to bookmark, copy/paste links etc. deep within your content.

Browsers

- Real time updates with comet
- Use a good development framework
- Hide the game client for overlays

tysdag 11. mai 2010

* Real time updates with Comet- By using comet-type requests you can effectively notify clients about changes without the client initiating the request. This can help overcome some of the limitations inherit in the web architecture. Use comet to notify users about logged on friends, implement game lobbies, chat, etc

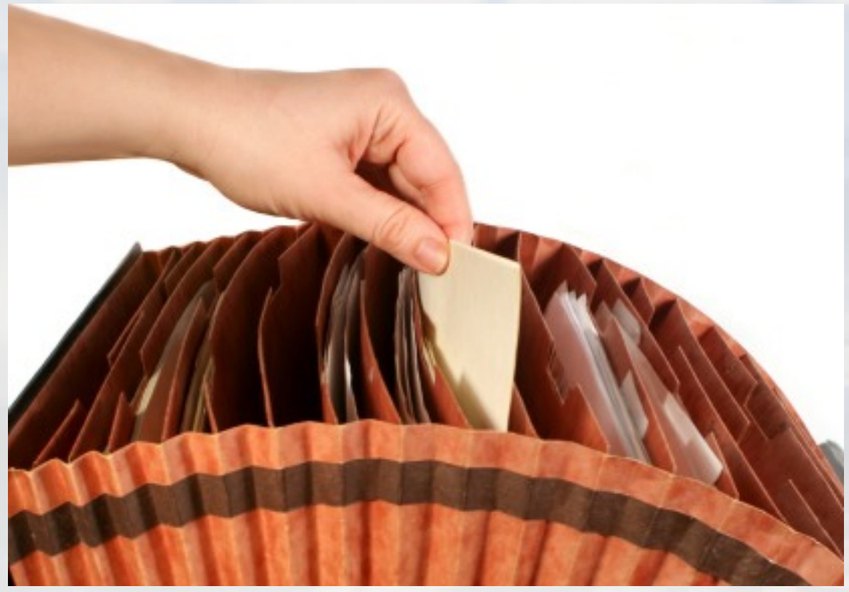
* Use a good development framework - Did I say “don’t roll your own”? A good web development platform these days should significantly help you not only creating a scalable backend, but also a rich frontend by simplifying javascript development. ESN Planet (http://www.esn.me) is a good choice.

* Hide the game client for overlays - A tiny browser tip at the end. Some html elements such as the game client do not play nice with lightboxes and overlays. A useful trick is to set the width and height of the game client element to zero while showing the lightbox.

- Before I leave the browser section - don’t support IE6. Too much hassle.

Content Management Systems

- Make sure you need one
- Make sure it's customisable
- Make sure it doesn't lock you into expensive hardware or software



vostopia.com™

tysdag 11. mai 2010

* Make sure you need one - I have spent a lot of time trying to crowbar a website into an ill-fitting CMS. Chances are most content is player generated (not developer or publisher). Stats, friends, achievements etc. CMS not a good fit for this. Will require a lot of customisation.
* Customisable - If you go for it, make sure it's easy to add new types and widgets. For player and publisher generated. CMS can be useful as a framework for front end (if it's good). Vendor customize - beware, don't lock in, game references.
* HW/SW lock in - You want your game to go big. Don't tie your profitability down by having high per server (or worse - per CPU) costs for OS, DB and/or CMS. There are good free or nearly free alternatives. Drupal and Joomla for CMS, MySql of PostreSQL for DB.

Geolocation

- Latency and matchmaking by region
- Useful for demographics
- The closer, the more understandable

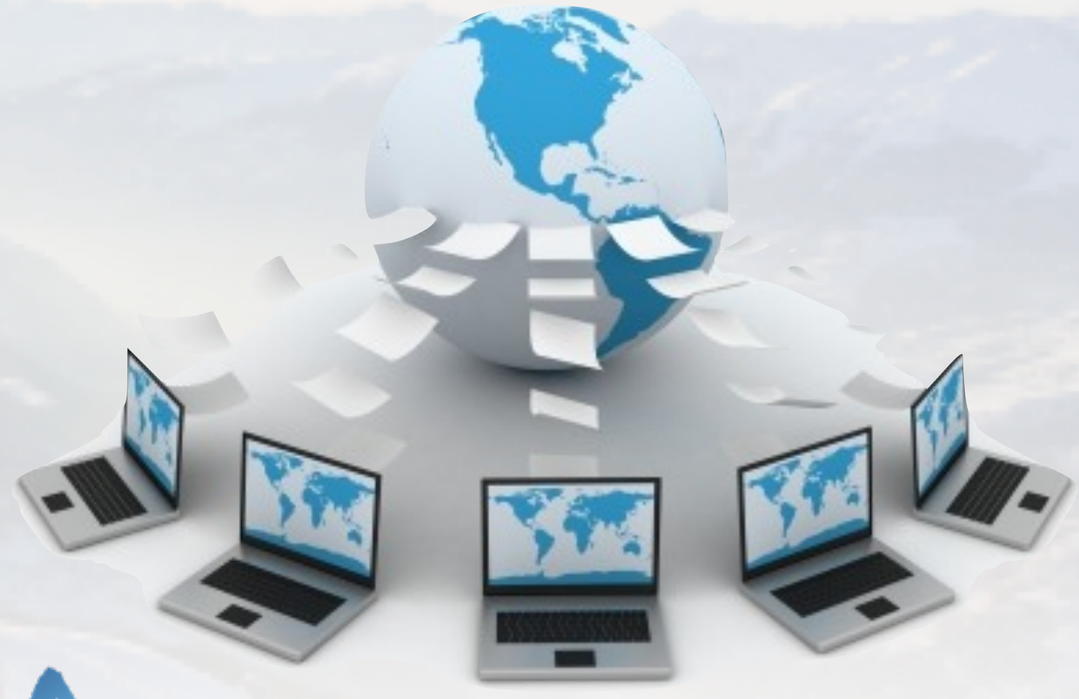
vostopia.com™

tysdag 11. mai 2010

* Latency and matchmaking by region - Latency is the enemy of online multiplayer gaming. Mix in country/region info from an ip geolocation service in your matchmaking scoring to increase the chances of lower latency.
* Demographics - Understand your players without boring questionnaires. General principle - automated information. Marketing, sales, operations, gameplay needs. Web games are global by nature. Markets to push, events to cater to - Christmas, cherry blossoms, Halloween, Oktoberfest etc
* Closer, Understandable - Everything from subtle actions to hard walls. Auto-suggested language, matchmake weight to own country, currency select, country lockout.

Content Delivery Networks

- Use CDNs for everything static
- Use browser cache effectively
- Let someone else host your js libs



tysdag 11. mai 2010

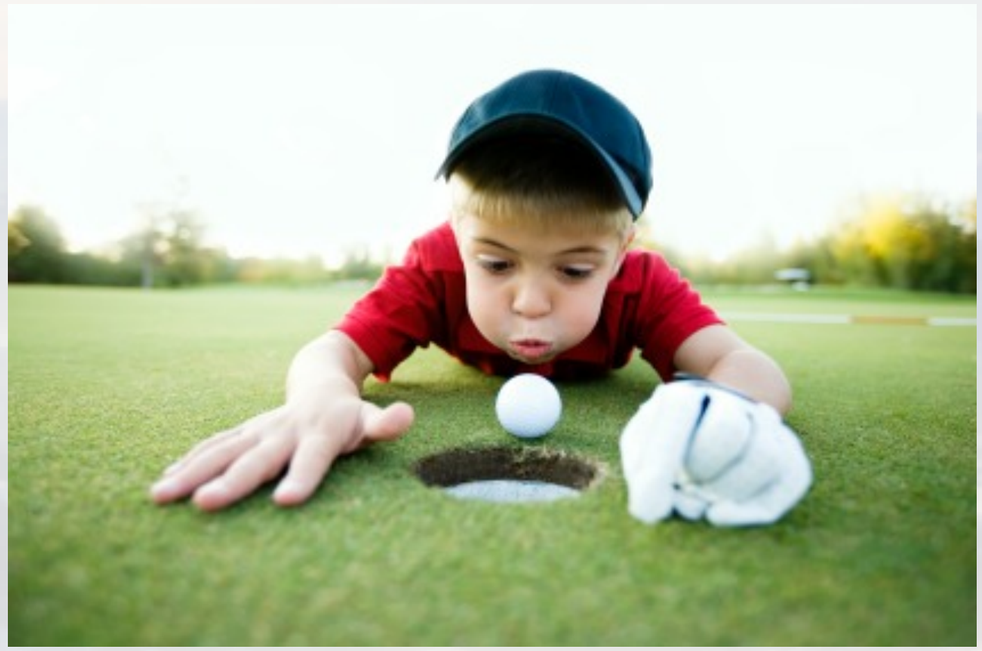
* Use CDNs for everything static - Using a Content Delivery Network is a great way of saving bandwidth from your server and latency to users. By distributing the data to servers near the end users, CDNs are able to serve static content such as game client, images, stylesheets and javscript libraries with higher bandwidth and lower latency to end users.

* Use browser cahce effectively (by incorporating resource version in filename)
In order to minimize bandwidth and latency you want to set the cache expiration date as high as possible. However this needs to be balanced by the need to update the content. By having the version of the resource in the filename of near static files, you can create new versions of files which will be used as soon as they are referenced from the dynamic content.

* Let someone else host your js libs - Standard javascript libraries such as jquery are often already hosted by third parties (such as google). By using hosted js libraries, you save bandwidth and increase the chance of the user already having the library in the browser cache

Cheating

- Protect the fair player
- Combat cheating early
- Matchmake cheaters together



tysdag 11. mai 2010

* Protect fair player - Hard to impossible to beat cheaters. Don't trust user input. Push cheat prone processing to server. Most important - don't let it affect the fair players. Build this into your game mechanic. Tools can help: DMW Anticheat, Hackshield, GameGuard, Punkbuster, Valve anti-cheat
* Combat cheating early - Start integrating your anticheating tool of choice early on, but even more importantly - include cheat prevention in your design processes from the earliest stages. It is cheaper to make a change on paper than in code. You don't necessarily implement all the cheat protection, but you build a knowledge for how to recognise particular cheats and how to quickly prevent them if they become a problem.
* Matchmake cheaters together - This is tricky, and might require some design sacrifices, but could be doable for your game. If our main goal is to protect the fair players, it matters less when people cheat as long as the fair players are not affected.

Security

- Never trust user input
- Avoid XSS and XSRF attacks
- Read and learn

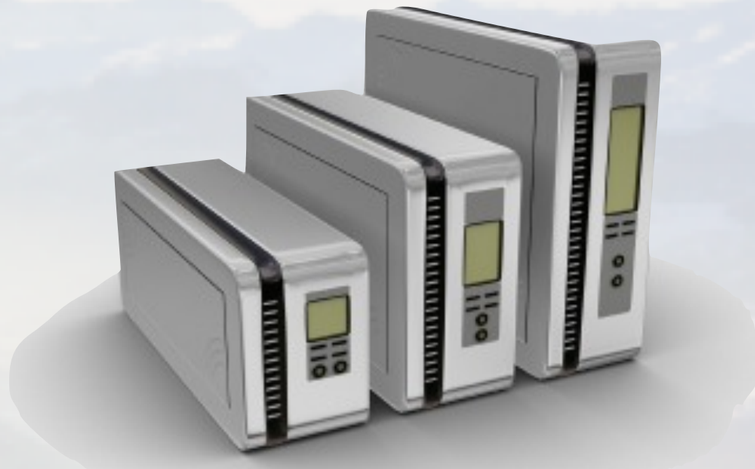
vostopia.com™

tysdag 11. mai 2010

* Never trust user input - From forum posts to game controls, anything client side can be compromised. Number one cause of security concerns is trusting some user side system that can't be trusted.
* Avoid XSS, XSRF- These deserve a particular mention. Cross Site Scripting and Cross Site Request Forgery attacks. Escape input, be very careful about allowing html on your site through forums or image links, and in general, allow as little text based user input as possible.
* Read and Learn - There are many good resources out there. Owasp development guide & Google browser security handbook are some good examples. The Owasp top ten is particularly useful for a quick overview. As a mirror of the cheating tip - you have to think security from the start.

Hosting

- Start small and early
- Use the cloud
- Let the experts help you



vostopia.com™

tysdag 11. mai 2010

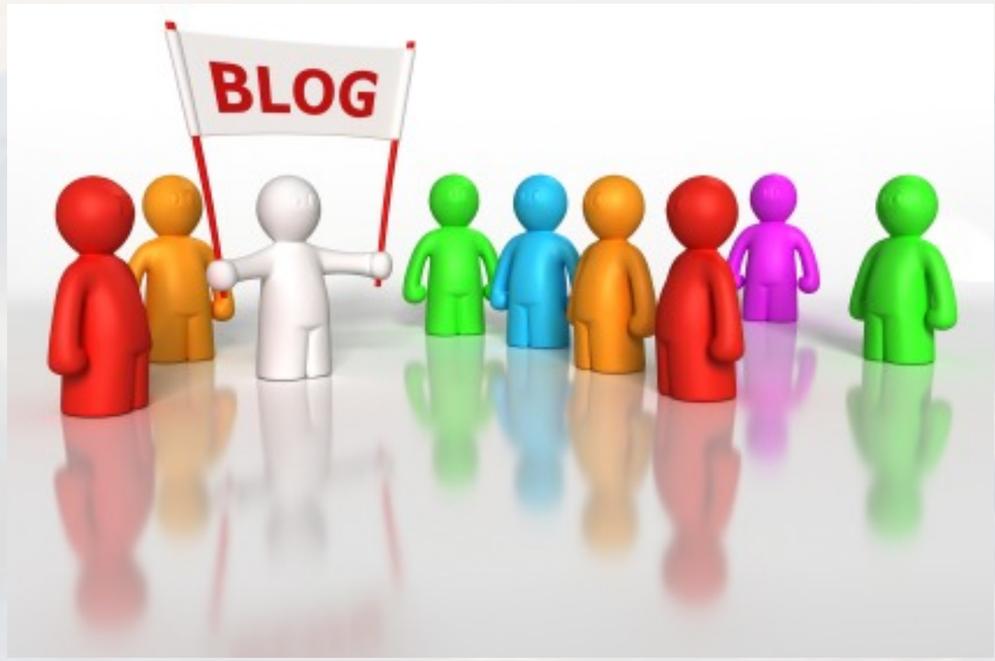
* Start small and early - If you are using dedicated servers for hosting, it can make sense to start with a small server in your data centre of choice early on in development. As you get ready for launch and get your production servers ready, you can repurpose the original server as a staging server.

* Use the cloud - Cloud hosting is a bit more expensive than dedicated servers, but gives you more flexibility and better capability to scale. If you have more than 2 machines, you are doing something wrong if you need to ssh into each of them.

* Let the experts help you - Don't try to do it all on your own. For scaling server architecture you can buy off the shelf solutions. The 300 pound gorilla in this space is RightScale. Zynga, Crowdstar and Playfish all use them. You will still have to write your game server layer, but the load balancing and database layers are all handled by off the shelf components.

Social features

- Create a space for your players (and keep it alive)
- Understand your audience
- Feed the interest
- Build social mechanics into your game



vostopia.com™

tysdag 11. mai 2010

* Create a space – Build features on your game's site or integrate with existing social network (or a combination) to allow your players to form and maintain an active community. Of course setting up a forum is a good candidate for creating a community, but it can also involve allowing comments on actions / updates, user pages etc. Work with and augment the players' existing social networks instead of trying to recreate them.
* Understand audience – Having an active community with ways to express themselves (discussions, polls, voting etc) give you a good opportunity to get specific feedback and to guide the future development of the game.
* Feed the interest – Taking an active interest in your community is crucial. Give information, post news, **answer questions**, and create challenges all increase the interest in the game. Players like being listened to, and being active and helpful in the community can really give your game a boost and the community will reward you for it.
* Build social into game – Social doesn't just have to be on the website – you can build it right into the game. Allow players to share what they are doing, their achievements right from the game, then chat about it on the web afterwards.

Facebook

- Be on facebook or connect with facebook
- Your grandmother can play!
- Short sessions with mechanics to come back

vostopia.com™

tysdag 11. mai 2010

* Connect - It's hard to ignore 400million users. Integrate your game with facebook directly, or use facebook connect to reduce login barriers and bring friend information to your game. Your game will instantly know of the players' friends who are also playing the game and will be set up for the players to more easily recruit their friends who aren't yet playing.
* Grandma - FB games redefined hardcore player: minority from traditional gaming -> majority in FB. Average player in FB is a female aged between 40-48. Keep mechanics simple. Sowing seeds in Farmville is a mechanic. A wall jump in Mario is too complicated.
* Short sessions & come back - 50% of sessions 15-60 minutes. (1-5h/week). 95% of players come back several times per week. Encourage this! Expectation of a short session is also a low barrier. Gentle learning curve, semi-automatic gameplay.

Customer Support

- Anticipate customer questions
- Online communities are our sentinels
- Make it a game
- Disproportionate unscalable CS

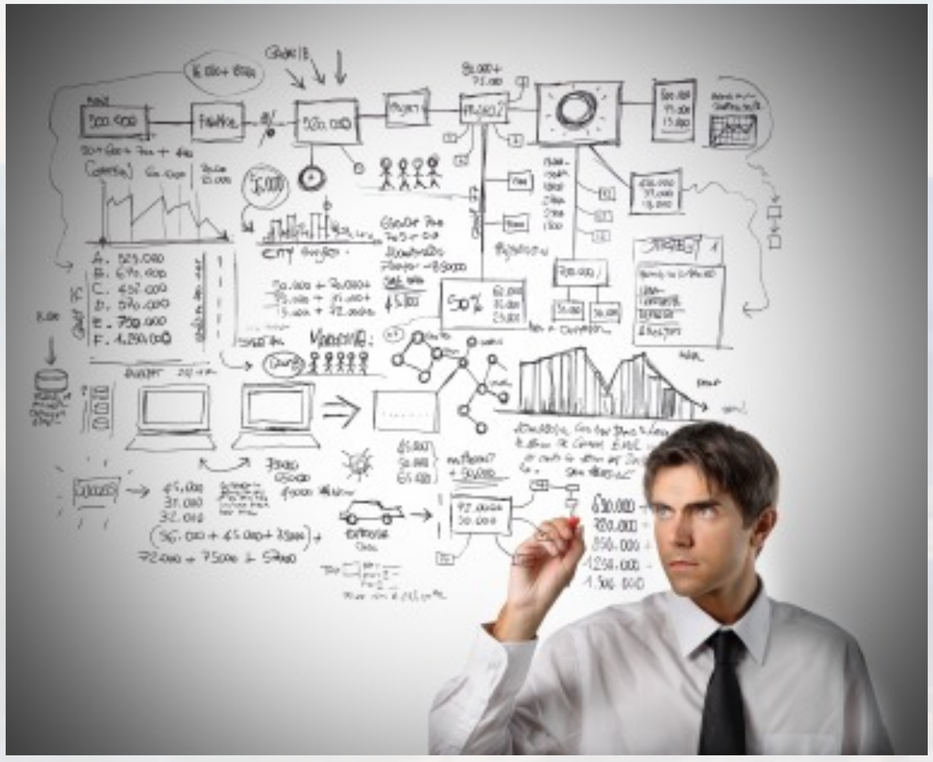
vostopia.com™

tysdag 11. mai 2010

* Anticipate customer questions - Build a solid FAQ. It is a lot cheaper than repeatedly answering all the questions. Any time you answer something new, make sure to get it into the faq. There are good FAQ tools out there. GetSatisfaction.com is one example.
* Online communities are sentinels - You've populated the FAQ with the basic questions you expect and have answered a round of questions from the community. Because web games are typically high-volume and low margin, you have to be clever about spending on support. Some things (such as payments) still need paid for CS.
* Make it a game - You can apply game mechanics to community CS. Check out stack overflow, uservoice as examples for this. If you have an achievement system for your game, you might want to integrate achievements based on your points scoring from these tools.
* Disproportionate unscalable CS - In the beginning, take a personal interest in CS on a level that doesn't scale. This helps you understand your customers and your product and it creates a stronger core of happy customers.

Business Models

- Integrate with portals
- Consider advergames
- Use in-game advertising (if it fits)



vostopia.com™

* Integrate with portals - Depending on your goals, this could remove a lot of the hard work as a lot of the back end work should be done for you. It will of course also put you in front of hundreds of thousands or even millions of players. Of course you might take a revenue share hit, but if you have an unproven game it might be the way to go. Most portals offer non-exclusive deals.
* Consider advergames - If you can get a deal with a partner for creating an advergame, it can be a good way for your team to test the waters in web games. Advergames are in themselves a growing market. This could replace the traditional development funding a publisher would do, although budgets will be fairly limited.
* In-game advertising - If it fits the style of game you are doing, in-game advertising could be worth considering. If you don't have a big franchise or a mega hit, you may have success selling exposure directly if the game is a good fit.

Free to play, microtransactions

- Don’t reinvent the wheel
- Build a friendly grind
- Sell items that make players’ lives easier / better

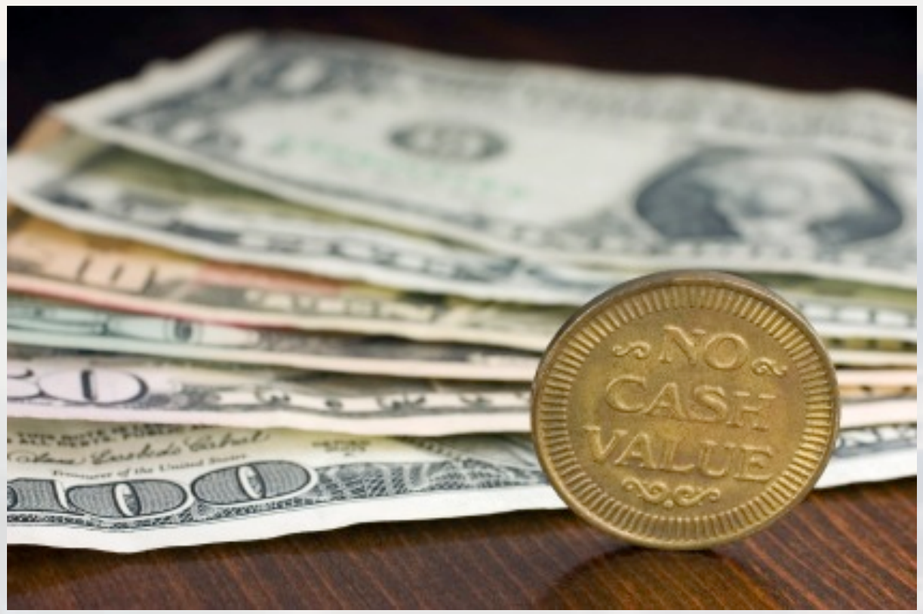
vostopia.com™

tysdag 11. mai 2010

* Don’t reinvent wheel – Handling micrtotransactions can be a fairly daunting prospect. Do not roll this from scratch. There are systems that do a good job of handling player accounts, payments, wallets, offer management, inventory etc. Three examples are LiveGamer, PlaySpan and FatFoogoo.
* Build a friendly grind – To build a successful virtual item sale game, you need to build a grind into your game so players have to come back over time and work to unlock new items. This can be simply through leveling up, or completing missions, getting achievements etc. It doesn’t have to be world of warcraft to work, but we can all learn from what they are doing.
* Sell items that make players’ lives easier – XP Boost for leveling up quicker, tractor in farmville, to its conclusion of items that effectively let the games play themselves such as fish feeder and tank cleaner in Happy Acquarium. Appearance / cosmetic items do sell, but as individual item sales go, items that make players’ lives easier win. The best ones are those that don’t directly affect other players’ experience of the game. Experience boosts are good, game mechanic altering things less so.

Free to play, microtransactions

- 2 currencies - earned and bought
- Teach players to spend
- Build in moneysinks



vostopia.com™

tysdag 11. mai 2010

* Earned & bought – This is the current best practice. Most flexible for designing your economy. Positive reinforcement of giving virtual currency / allowing player to buy free items. Beware to not have the exact same offer for bought and earned as that establishes an exchange rate between the two. This is still new ground, but could lead to taxation of earned currency.
* Teach players to spend – The free currency is a good starting point for this. Matched with convenience items it's even better. Upsell to paid currency items – make sure they are compelling enough. Could be item slots for paid items only, or typically, game mechanic altering items. Zynga are the masters of this – play their games.
* Build in moneysinks – You need a way for players to consistently spend their money. Rentable items. Consumable items: Fuel, ammunition, health packs.

Metrics

- Measure it, then control it
- Don’t roll your own
- Test alternate versions

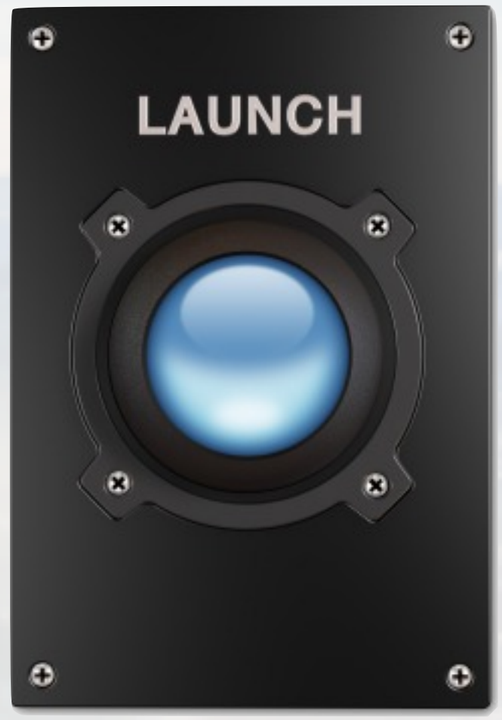
vostopia.com™

tysdag 11. mai 2010

* Measure then control – Tom De Marco said “You can’t control what you can’t measure”. **We can measure!** Boxed product, you only know roughly how many players you have. Online / web games give unique opportunity to learn a lot from the player. Feed back to game design, web design, marketing – **targeted!**
* Don’t roll your own – Running theme... Capturing metrics doesn’t have to be terribly time consuming, and the web industry have solved the tech part of the problem. You can tie in to Google analytics, Omniture or Clicky among others by sending http requests at key points in your game. Remember that learning where to measure is also an iterative process.
* Test alternate versions – When bringing out new features, make sure to test at least two different versions (A/B test). Could be wording, placement, graphic style etc. Tech for web pages can be used in your game as well (depending on game tech). Allows you to constantly optimise your experience -> lead straight to increased retention, conversion.

Launch Early

- Minimum viable product
- Launch softly, carry a big oscilloscope
- Know if you have a hit quickly
- Marathon, not a sprint



vostopia.com™

tysdag 11. mai 2010

* Minimum viable product – Build the minimum compelling experience you can and put it out there. Think weeks or months rather than years. The tools and shortcuts are there for both game and web development – **use them**.
* Launch softly – The web is **not** a box product **big bang** launch. Get the game out and seed it to a few users (in the hundreds or low thousands). Concentrate on **learning** from those and start **measuring** how your service performs in the wild (gameplay, load, to a lesser extent security). If – like us in Norway – you have access to a small representative market where a marginal language is spoken ;–) you could launch **only** in **that language** to minimize **damage** potential.
* Know if you have a hit quickly – Your first **few weeks** in the face of real users will give you plenty of indication for how well your game is doing. Use this info as you will. You will get early indications on whether you have **failed quickly** or if you have a **potential hit** on your hands.
* Marathon not a sprint – The dev cycle of a web game might be a relative sprint to the start line – launch – but you need to plan for the long haul for a chance to build a compelling service players want and to **innovate** in the face of very **quick moving competition**.

Marketing

- Know how Acquisition, Retention, Conversion, Upselling applies to you
- Focus on Retention first
- Build for viral
- Revenue share for acquisition is an option

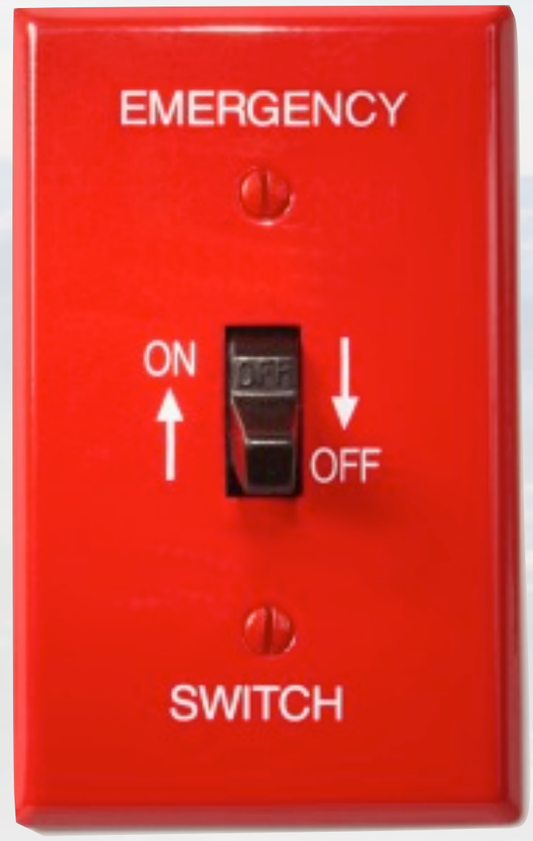
vostopia.com™

tysdag 11. mai 2010

* Know how A, R, C, U applies - Think of all marketing activities in these terms. You have a whole new toolbox at your disposal. Because you are now running a service, you can use development resources for marketing activities. Refer a friend schemes, include rewards or achievements for recruiting. Publish progress or requests for help on players' social networks. Ideally, build it into the game mechanics in a meaningful way. Give the players the tools to do it for you.
* Focus on Retention first - Don't sail with a leaking ship. If you start with relatively low numbers, make sure you hit your early retention (and conversion) targets before more aggressively acquiring players. Ideally when you acquire, you will also have some viral mechanics in place so paying for acquiring one user will give you a fraction of a user extra.
* Build for viral - AFacilitate the spreading of your game through built-in viral mechanics. There is relatively low hanging fruit such as allowing the player to post interesting updates to social networks. High scores, achievements are examples of this. Also allow the player to invite friends/contacts by importing lists from the popular web-mail systems or social networks. With a little more work, you can do refer a friend schemes which lead to rewards. If you have a game that has strong social elements as well, these viral mechanics will work even better.
* Rev share is an option - And in most cases a very good one. If you have a compelling service, have plugged the holes, made people stay happily, and refer your service to friends, it might be time to shop around sites with a lot of players for referring players to you. These could be game portals that are a fit for your game, or sites frequented by your target audience. You will have to give away a percentage of lifetime revenue for a referred user. If there is still margin in it for you after the rev share, this is worth it. Also remember, that user's friends are yours.

Emergencies

- Always build in an on/off switch
- Keep on working
- If all else fails...



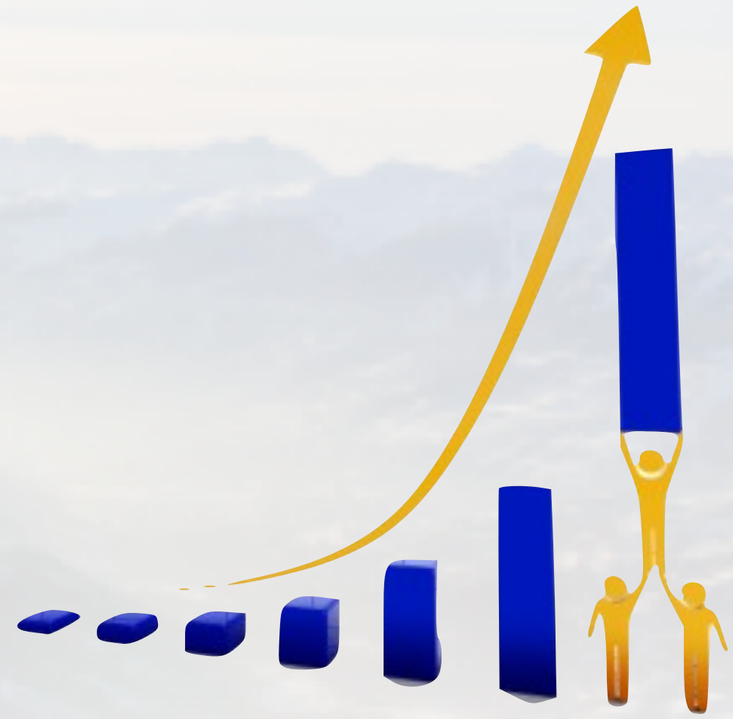
vostopia.com™

tysdag 11. mai 2010

* Build in on/off switch - For each feature you build, make sure there's a simple way of switching it off. Functional beats fancy any day for this.
* Keep on working - Related to the previous point. Build in Defcon levels. Whether you had to switch a feature off or outside events put parts of the system offline, you should be set up so you can keep working. You can probably keep running without your friends system, forum, leaderboards. Even serving your social game without social. The typical fail points are loading and storing data. Comparementalize data load/store by creating independent components. Assume that a call to a data component will fail, and don't let that affect the rest of the site. And then, and then, and then... **the radio broke!**
* If all else fails... - Communicate, communicate, communicate! Fail whale - or for those who remember - Guru meditation. Update your twitter feed, facebook page etc and let your players know what is going on and roughly when it looks likely that you can be back up. Don't be too precise though. If you've done your job right with your service, players will rather quickly find out that you're back online.

Live development

- Iterate! - Keep improving
- Deploy on Tuesdays
- Still in beta



vostopia.com™

* Iterate! Keep improving - Steady rate of progress more important than starting with a perfect product / service. Can be hard for a boxed product team to get under skin. If you manage to bring out relevant updates frequently, players will feel listened to and the value of your service will increase. No-one remembers what the game was like when it first launched into beta.
* Deploy on Tuesdays - Or monday afternoon or wednesday, just not Fridays. You want a bit of ramp up time for the launch and you'd rather be in the office fixing any issues on wednesday and thursday than on a saturday and sunday.
* Still in beta - If gmail could do it for 5 years, you can do it too. People have quite a high tolerance for mistakes if they feel you are trying to add value. For a while you can get away with your whole service being in beta. After this grace period (probably less than 5 years ;-) it makes sense to set up a public beta server where you can test new features on the early adopters before rolling them out to everyone.

Questions?