

EPSI Bordeaux
73 rue de Marseille
33000 Bordeaux

Ultimate Washer Inc.
711 Commerce Way E, Unit 1
33458 Jupiter FL, USA

Les technologies web sont elles le futur du jeu vidéo?

Par Maxime LAMAISON

Directeur de recherche:
Sylvain Labasse

Promotion 2013

REMERCIEMENTS

Je tiens à remercier dans un premier temps Sylvain Labasse qui m'a encadré tout au long de la rédaction de ce mémoire, mais aussi toute l'équipe pédagogique de l'EPSI (École Privée des Sciences Informatiques) de Bordeaux pour m'avoir accompagné durant mes 5 années d'études dans cette école.

Je tiens également à remercier tous les maitres de stage et collaborateurs qui m'ont accueilli lors de tous mes stages en entreprise et qui ont contribué à entretenir ma passion de l'informatique et a confirmé que la voie que j'ai choisi était la bonne.

Enfin, je remercie mes parents et mes amis pour leur soutien durant ces années d'études, ainsi que mes camarades de classe qui les ont rendu particulièrement agréables.

TABLE DES MATIÈRES

INTRODUCTION	1
I - Les graphismes via les langages web	7
<i>A - les performances graphiques des navigateurs web</i>	8
<i>B - la 2D</i>	14
<i>C - La 3D</i>	23
II - Le jeu vidéo, ce ne sont pas que des graphismes.	39
<i>A - Etat des lieux du développement standard</i>	40
<i>B - Les capacités du web pour répondre au développement historique de jeux</i>	49
<i>C - Des possibilités inédites de jeu</i>	59
III - Les implications des jeux HTML5 pour les acteurs du jeu vidéo.	64
<i>A - Ceux qui jouent</i>	64
<i>B - Ceux qui les fabriquent</i>	70
<i>C - Les points de vue des géants du web et du jeu video</i>	77
CONCLUSION	85
Glossaire	88
Sources	95
English Version	97

INTRODUCTION

En 2012, Mark Zuckerberg annonçait que sa plus grosse erreur était de trop avoir parié sur le HTML5. D'ailleurs, après la hype de 2011, 2012 a été une année plutôt compliquée pour le HTML5 qui a subi des attaques de toutes part. Mais pour savoir ce qu'il en est réellement, il faut revenir un peu arrière.

Internet est probablement l'invention la plus révolutionnaire de la fin du 20e siècle. Même si les origines du plus grand réseau d'ordinateurs mondial, on sait pourtant que c'est le 1er janvier 1983 que le nom "Internet" est apparu pour devenir aujourd'hui un outil incontournable et indispensable. Que cela soit au niveau de la communication, de la culture, de l'information, du divertissement, et bien d'autres domaines, internet est aujourd'hui partout et on note d'ailleurs un nombre d'utilisateurs grimpant à une vitesse folle: si en 2013 on comptabilise environ le nombre d'internautes (défini sur internetworkstat.com comme "une personne de plus de 2 ans qui s'est connectées dans les 30 derniers jours") à 2.2 milliards (1/3 de la population mondiale) ce qui double le nombre d'utilisateurs par rapport à 2008.

Internet est donc un système d'interconnexions de machines qui constituent un réseau informatique sans centre névralgique. Ce vaste réseau permet à ces machines de communiquer et d'échanger ainsi des informations qui permettent entre autres d'offrir à un utilisateur des services variés comme l'email, la télévision, le partage de fichiers et surtout l'accès au World Wide Web.

Pour accéder à internet, il faut réunir plusieurs outils: un équipement adapté (ordinateur, téléphone, console de jeu vidéo, etc.), un fournisseur d'accès internet (FAI) qui permet à l'utilisateur de se connecter au dit

r  seau, un lien "physique" entre l'  quipement et le FAI (c  ble adapt  , WiFi, 3G, etc.) et enfin un syst  me client adapt   a ce lien.

Il faut   galement noter qu'un logiciel sp  cifique est n  cessaire suivant l'application internet que l'utilisateur veut ex  cuter: un client SMTP et POP ou IMAP pour le courrier lectronique, FTP pour le transfert de fichiers, un logiciel de P2P pour le partage de fichiers de pair a pair (mod  le particulier ou le client est aussi un serveur), un navigateur web pour l'acc  s au World Wide Web.

Cette derni  re application d'internet, le World Wide Web, aussi surnomm   la toile, est souvent confondu avec la totalit   de l'internet bien qu'elle n'en soit qu'une partie. Invent  e plusieurs ann  e  s apr  s ce dernier, en 1990, c'est pourtant la toile qui a contribu    l'immense succ  s d'Internet. Si le d  but du web (un des nombreux surnoms du World Wide Web) fut plut  t f  brile, car cette technologie   tant poss  d  e par le CERN qui en a les droits d'auteur, c'est    partir de 1993 que celui-ci devient public et qu'il commence r  ellement    se d  velopper: on pourra noter lors de cette ann  e 93 que si au cours du mois de juin, seuls 130 sites taient consid  r  s comme fiables, ils taient au nombre de 623 en d  cembre. Le nombre de sites web et leur usage ont augment   au rythme annuel de 341634 % depuis cette ann  e (10 022 sites en d  cembre 1994). 1993 marque   g  alement l'apparition des premiers navigateurs Web comme Mosaic ou Lynx.

En 1995, Microsoft lance la "guerre des navigateurs", soit le combat conomique men   par les g  ants de l'informatique voulant chacun que leurs technologies soient utilis  es.

Une page web est construite gr  ce    diff  rents langages informatiques, le HTML (1990) dans un premier temps permet l'  criture (puis l'ajout d'images ou de vid  os, mais plus tard) dans les pages et la cr  ation de liens entre celles-ci et le CSS permet quand a lui la mise en forme de

cette page. Les langages permettant la “discussion” avec un serveur et une base de données (par exemple PHP) ont également permis l’évolution du web, car l’ajout de données “de masse” sur son site personnel devient plus aisé. Enfin, l’apparition de langages comme JavaScript a permis une interaction utilisateur-page web plus efficace en rendant cette page “plus vivante” (avec l’apparition de popup posant une question par exemple).

Le langage web le plus prometteur actuellement est sans aucun doute le HTML5, la prochaine révision du HTML. Seulement au-delà du langage de balise qu’est le HTML, le HTML5 est plus considéré comme un concept à lui tout seul. Ce concept de HTML5 réunirait à lui tout seul les langages de développement HTML5, CSS3 qui est l’évolution du CSS, le JavaScript et une multitude d’API pour faciliter le développement web tout en lui donnant accès à de nouvelles possibilités.

Les évolutions du web ont ensuite permis des interactions plus directes avec les utilisateurs, ces derniers pouvant générer directement du contenu sur une page sans connaissances informatiques: c’est ce qui est appelé le Web 2.0 (apparu en 2003 pour la première fois). En effet, l’apparition de blogs, forums, wikis (le plus connu étant wikipédia lancé en 2001) et de réseaux sociaux (MySpace en 2003, Facebook en 2004) popularisent le web en lui offrant une puissance communicative, informative et publicitaire encore plus importante.

Cependant, le terme de Web 3.0 n’est pas encore tout à fait clair sur son contenu. Si l’hypothèse la plus avancée était celle du Web 3.0 = Web sémantique (Web des objets), certains pensent encore à d’autres possibilités (comme le Web 3D par exemple) ou avancent qu’en pensant de cette manière, le Web 4.0 pourrait voir le jour avant le numéro 3, certains chefs d’entreprises (comme Eric Schmidt, PDG de Google) préfèrent d’ailleurs parler de “futur du web” au lieu d’utiliser l’expression Web 3.0.

Les évolutions successives des technologies internet (logicielles et matérielles), du web, du traitement des données et de la sécurité informatique ont également fait naître un des piliers de l'informatique connectée: le Cloud computing. IaaS, SaaS, DaaS ou PaaS sont aujourd'hui des outils bien connus des entreprises qui utilisent la puissance des serveurs distants pour utiliser des applications, sauvegarder et échanger des données, communiquer tout en diminuant les couts générés par toutes ces applications. Les évolutions des navigateurs web et des langages informatiques permettent également de lancer directement d'autres applications d'internet (comme l'envoi de courriel avec le service offert par Gmail), mais aussi d'utiliser des applications informatiques "de base" (par exemple avec Google Drive, qui permet l'utilisation d'un traitement de texte gratuit tout en sauvegardant ses données en ligne). L'énorme avantage du Cloud computing est sans conteste sa facilité d'accès: en effet, se connecter a une application publique ou privée via un navigateur internet est d'une facilité déconcertante et permet ainsi de créer un environnement informatique complètement dématérialisé.

Le Cloud computing est également abordé de toutes parts par l'industrie du divertissement: on peut noter la présence en ligne de services en ligne d'écoute ou de téléchargement de musique, mais aussi de visionnage de films ou lecture de livres.

Il n'a donc fallu que peu attendre pour voir celui que certains surnomment le "8e Art" débarquer sur les plateformes de Cloud computing: le jeu vidéo.

Le jeu vidéo fut popularisé en 1972 par Pong, dont le gameplay (ressenti de joueur par rapport à l'expérience de jeu) fut suffisamment bon pour que les utilisateurs s'y intéressent. Ce fut donc le début de l'industrie vidéoludique et de la course au développement des jeux vidéos. 1985 est une autre date cle du jeu vidéo avec l'apparition de la NES de Nintendo

qui marque l'explosion des consoles de salon (la NES compte à ce jour à pratiquement 62 millions d'unités vendues) et surtout l'explosion auprès du grand public. Depuis, le secteur du jeu vidéo n'a cessé de croître pour finalement dépasser, en 2003, le chiffre d'affaires mondial généré par le cinéma (en 2003 les chiffres de ventes dans le monde atteignaient 25,2 milliards d'euros). On remarquera qu'à l'instant du cinéma, le jeu vidéo a également ses icônes (telles Mario, Lara Croft ou Pikachu) qui permettent à ce dernier d'accroître son expansion grâce aux produits dérivés.

Cette reconnaissance par le public est aujourd'hui manifestée par des rassemblements dédiés (E3 à Los Angeles, Paris Games Week à Paris...), des événements compétitifs incluant un ou plusieurs jeux vidéo (coupes du monde, tournois...) et même des joueurs et équipes professionnels.

Il est aussi important de reconnaître au jeu vidéo une création d'emploi importante, ne cessant de s'accroître avec le temps et l'apparition incessante de nouveaux supports: parmi les principaux métiers, nous pouvons retrouver: les producteurs qui gèrent le développement du jeu, les développeurs qui vont programmer informatiquement le jeu, les concepteurs qui ont en résumé une fonction de scénariste, les infographistes, les musiciens qui assistent les concepteurs, les testeurs et les traducteurs.

Le jeu vidéo possède également le record du plus important lancement de l'histoire des médias (comprendre musique, films et jeux vidéos) avec plus de 9,3 millions d'unités vendues le jour de sa sortie avec le jeu Call Of Duty MW3. Toutes ces données montrent ainsi que le jeu vidéo représente un enjeu économique colossal.

De plus, le jeu vidéo est maintenant entré dans la vie de plusieurs millions de personnes (parfois d'ailleurs de façon inquiétante) puisqu'une étude américaine menée par la Kaiser Family Foundation annonce qu'en 2010, un enfant ou adolescent passait en moyenne 4 heures par jour à jouer aux jeux vidéos.

Cependant, si le jeu vidéo a souvent été perçu comme difficile d'accès, c'est peut-être à cause de l'abondance de supports pour jouer: consoles de salon, consoles portables, téléphone, tablettes, ordinateur de bureau, bornes d'arcade, etc. Offrant chacun une expérience de jeu différente et trouvant parfois son type de joueur.

Internet et le monde du jeu vidéo sont très liés aujourd'hui. En effet, que cela soit au niveau de simples jeux "passe-temps" en Flash disponibles sur le web ou bien au MMORPG rassemblant des joueurs du monde entier pour une partie unique à plusieurs millions: le plus grand réseau du monde a parfois été un moteur pour le jeu vidéo et parfois améliore grâce à ce dernier. Il est maintenant possible de jouer, mettre à jour, acheter, revendre, louer, pirater des jeux vidéos sur le net. L'avènement des Market Place, ces espaces en lignes qui agissent en quelques sortes comme un grand magasin virtuel y ont été pour beaucoup: si bien qu'aujourd'hui tous les supports et toutes les marques ont leur MarketPlace: Apple a l'App Store pour son iPhone et son système OSX, Google a le Google Play pour Android et le navigateur Google Chrome, Sony a le PlayStation Store pour la PS3 et les différentes consoles portables de la marque, Microsoft le XBox Live, Steam est un immense magasin en ligne disponible sur Windows, Mac OSX et certains systèmes Linux, etc. Tous ces exemples montrent que, de la même manière que les supports sont extrêmement nombreux pour les utilisateurs, il en est de même pour les marketplace en ligne ce qui s'avère encore plus troublant.

De quelle manière améliorer ces déconvenues pour les utilisateurs. La solution peut venir du Web et des navigateurs: en effet, nous avons vu par l'intermédiaire du Cloud computing que ces derniers offraient de plus en plus de possibilités aux utilisateurs privés et professionnels tout en restant performant. Or, un navigateur web étant un logiciel installable et souvent présent sur tout support accédant au Web, il est donc possible de se demander s'il ne serait pas plus agréable d'offrir la possibilité de jouer

à son jeu favori depuis son navigateur web, sans autre installation nécessitée. Cela permettrait de se libérer de logiciels externes prenant de la mémoire sur le disque dur et du temps à installer. Encore mieux, serait-il possible de jouer au même jeu même si le support est différent? Par exemple, prenons un jeu lambda, vous jouez tranquillement chez vous depuis votre support favori, en quittant votre domicile et en ayant un compte utilisateur, vous pouvez reprendre cette dernière telle que vous l'avez laissée depuis un support plus portable. Et le clou du spectacle, serait-il possible grâce aux navigateurs web de mélanger les supports entre eux? Gardons notre jeu lambda, vous jouez sur votre ordinateur personnel, mais n'aimez pas le gameplay offert par le clavier: connectez votre tablette au même jeu grâce à votre compte utilisateur pour en faire une manette et profiter ainsi d'une expérience de jeu pleine.

Ces dernières questions éveillant ma curiosité, je vais donc répondre à la problématique suivante: les technologies web représentent-elles un avenir viable pour le jeu vidéo?

Dans une première partie, je me concentrerai sur les possibilités graphiques offertes par les technologies web actuelles. Puis dans un second temps, le jeu vidéo n'étant pas seulement une expérience graphique, j'évoquerai les autres moyens dont disposent les technologies web pour offrir la meilleure expérience de jeu possible. Enfin, je parlerai de la manière dont l'industrie du jeu vidéo peut exploiter cette nouvelle forme de jeu.

I - Les graphismes via les langages web

Dans un jeu vidéo, le premier élément de jugement de la part des joueurs est sans aucun doute le graphisme. S'il est vrai que certains exemples montrent que le scénario ou le gameplay ont quelquefois fait

oublier l'aspect visuel du jeu par leurs richesses, les exemples sont rares et ne sont souvent appréciés que par une communauté restreinte de joueurs passionnés. En effet, les graphismes sont la "première impression" du jeu, ils sont souvent un facteur déterminant en terme de choix de ce dernier. Que cela soit par leur originalité ou réalisme, ils représentent un critère important dans le monde du jeu vidéo. On notera d'ailleurs que jeuxvideo.com en fait un critère déterminant dans leur système de notation d'un jeu. Nous verrons donc dans un premier temps quelles sont les possibilités offertes par les navigateurs pour interpréter les langages web, puis comment ces technologies permettent la création d'un jeu vidéo en 2D, en 3D et enfin quelles alternatives et nouveautés sont choisies par les développeurs pour améliorer le développement des jeux vidéo en ligne.

A - les performances graphiques des navigateurs web

1 - Historique

Je vais aborder dans ce mémoire de nombreux thèmes concernant les navigateurs web. Aussi il est à mon sens indispensable de commencer par expliquer le fonctionnement d'un navigateur web, ne serait-ce que pour la compréhension générale à venir.

Un navigateur web est un logiciel mis à disposition de l'utilisateur afin que celui-ci puisse naviguer et profiter de l'ensemble du World Wide Web. Concrètement, il s'agit au minimum d'un client HTTP. HTTP est un protocole de transfert de données d'un client vers un serveur. Donc le navigateur web va permettre à l'utilisateur de transferer les données d'un serveur web vers son ordinateur.

Un navigateur web est constitué le plus souvent d'une interface utilisateur pour faciliter son utilisation, d'un moteur de rendu graphique pour afficher

les données à l'écran sous une forme plus agréable et éventuellement d'un moteur gérant les scripts et d'un gestionnaire de plugins.

Les navigateurs web sont aujourd'hui présent sur une multitude de plateformes: ordinateur personnels, téléphones mobiles, tablettes tactiles, appareils photos, consoles de jeux vidéos, etc. Les constructeurs s'acharnent d'ailleurs à essayer de permettre un accès au Web depuis n'importe quel appareil électronique.

Le premier navigateur Web s'appelait WorldWideWeb et date de 1990. Développé en Objective-C il fut placé dans le domaine public en 1993. Cependant le "premier" navigateur web connu du grand public fut NCSA Mosaic développé par le NCSA (National Center for Supercomputing Applications, soit le centre de recherche américain pour les applications des supers ordinateurs) situé sur le campus de l'Université de l'Illinois. Sa popularité et sa marque indélébile dans la popularisation du Web pour le grand public lui d'ailleurs ont valu une plaque commémorative en son honneur.

C'est en 1995 qu'est apparu le premier des 5 leaders actuels de la "bataille des navigateurs": Internet Explorer de Microsoft qui a dominé outrageusement le marché (avec le plus haut pic d'utilisation entre 2001 et 2002 où il était estimé que 9 ordinateurs sur 10 l'utilisaient) jusqu'à l'apparition d'un autre géant actuel, Mozilla Firefox, navigateur open source apparu en 2004.

Cette bataille des navigateurs menée par les géants de l'informatique actuels fait toujours rage aujourd'hui, chacun voulant asseoir sa domination par toujours plus d'inventivité. Il se dégage donc 5 leaders actuels sur le marché:

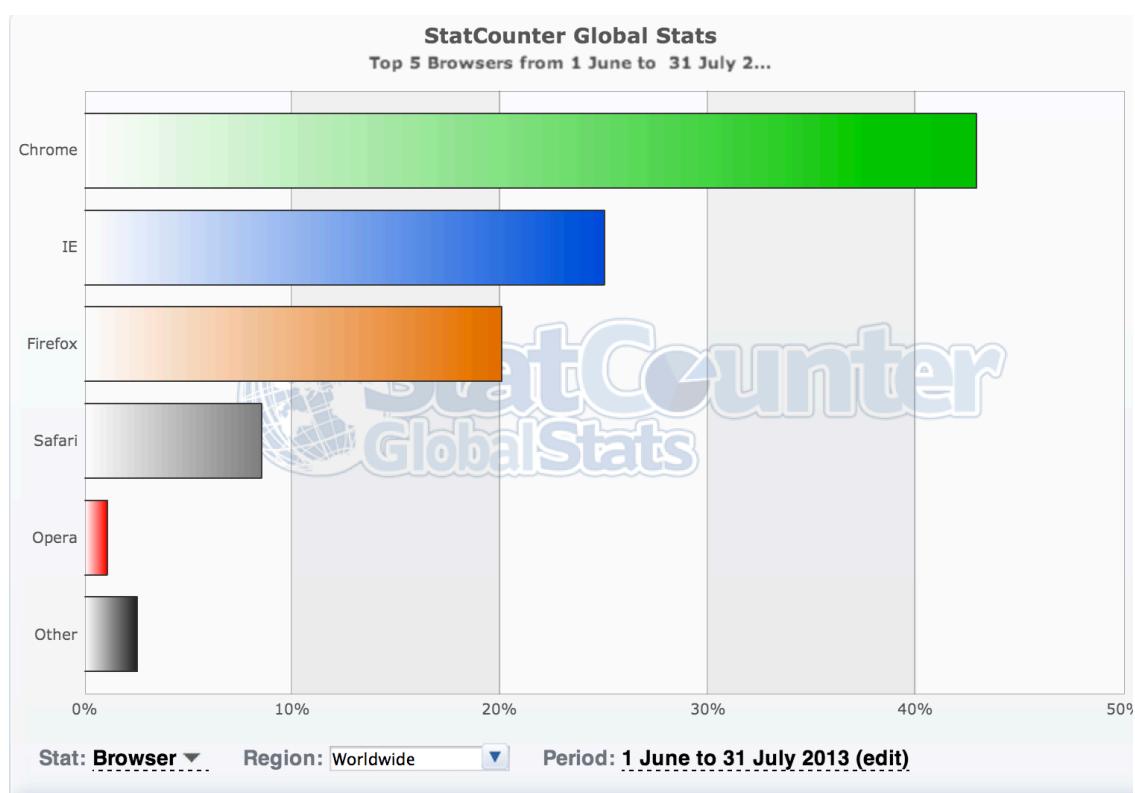
- Internet Explorer de Microsoft, arrivé en 1995 qui est le leader actuel au niveau des accès depuis un ordinateur personnel (56,15 % en juin 2013). Internet Explorer est aujourd’hui à sa 10e version.
- Mozilla Firefox de la Mozilla Foundation, arrivé en 2004. Il se distingue de ses concurrents en étant un logiciel libre, développé grâce au bénévolat. Il a connu un succès énorme dès son début, il était en 2010 le navigateur le plus utilisé en Europe. Firefox est aujourd’hui à sa 25e version.
- Safari d’Apple disponible depuis 2003. Il a été développé lors du retour en force d’Apple durant les années 2000 et surtout suite à l’abandon progressif du développement d’Internet Explorer sur les plateformes Mac. Pour l’anecdote, Steve Jobs tenait à ce qu’il s’appelle Freedom pour montrer la libération vis à vis d’Internet Explorer et de Microsoft. Safari est aujourd’hui à sa 6e version.
- Google Chrome de Google, disponible depuis fin 2008. Chrome porte la marque de fabrique de Google dans le monde des navigateurs: la simplicité. Il est également l’un des navigateurs les plus performants et innovants du marché. Toutes ces qualités font que Chrome est aujourd’hui le navigateur le plus utilisé dans le monde (43% en juin 2013), profitant de l’essor des mobiles et tablettes utilisant le système Android. Chrome est aujourd’hui à sa 28e version.
- Opera, apparu en 1996, est le moins connu des 5. En effet, il ne représente qu’ 1,07% du marché mondial (juin 2013) mais peut se targuer d’être le 3e navigateur du monde mobile (16% en juin 2013). C’est ce dernier fait qui me pousse à l’introduire dans la liste, car le jeu vidéo trouvant un énorme marché dans le monde mobile, il est évident qu’un des leaders devait être pris en cause.

Cette liste est bien sûr non exhaustive, et il existe une multitude d’autres navigateurs. Néanmoins, ces 5 sont partageant plus de 97% du marché mondial, il m’a semblé que je pouvais éclipser les autres. De plus, je me concentrerai principalement sur les versions actuelles de ces navigateurs, pour diverses raisons: à l’instar d’une vieille voiture, les vieux navigateurs

ne respectent pas forcément toutes les conditions de sécurité utilisées dans les navigateurs récents et aussi car il manque dans ces anciennes version des spécifications caractérisant l'évolution des standards du web qui peuvent impacter la navigation.

De plus, j'ai la conviction que l'utilisation de vieux navigateurs impacte directement sur l'évolution du web, or les développeurs ne devraient pas perdre du temps à développer toutes leurs applications pour toutes les anciennes versions (c'est déjà assez compliqué sur les versions récentes).

Les navigateurs web sont continuellement en évolution, que ce soit pour correspondre aux standards du web de leurs temps, ou bien pour parer des virus ou aux autres logiciels malveillants.



Voici un graphique représentant le marché actuel des navigateurs internet (source: statcounter.com)

2 - Fonctionnement

Nous allons maintenant parler des possibilités graphiques ainsi que des moteurs JavaScript de ces navigateurs, pour la bonne et simple raison que c'est le nerf du développement de jeux vidéo en ligne sans nécessiter de plug-in externe (ce qui est le cas avec Flash d'Adobe). Je me concentrerai sur les possibilités des navigateurs dans leurs versions actuelles, sauf si certaines sont pertinentes comme Internet Explorer 9 (6,8% du marché mondial en juin 2013).

Le navigateur affiche une page web sous sa forme graphique grasse à son moteur de rendu. Techniquement, une page web est le plus souvent constituée grâce au langage HTML. De nombreux moteurs de rendu différents ont vu le jour, mais avec la complexification des standards du web (objets formant le socle d'une page web), seuls quelques un sont prédominant aujourd'hui.

Afin de tester la conformité des moteurs de rendu des navigateurs web, le Web Standard Project a mis au point des tests appelé Acid. Acid3 est le test présent depuis 2008, pour réussir ce test, le navigateur doit obtenir un score de 100/100 avec ses réglages par défaut. Acid4 devrait voir le jour et se concentrer sur l'implémentation des technologies SVG et CSS3.

Le W3C possède une série de tests originellement nommés Test Suites afin de vérifier de leurs côtés le respect des standards du web par les navigateurs afin d'aider au développement sain du web.

Voici les moteurs de rendu présent chez les 5 leaders du web :

- Trident par Microsoft pour Internet Explorer (Mobile et Ordinateur personnel), il a obtenu 100/100 au test Acid3 depuis la version 9 du logiciel

- Gecko par Mozilla pour Firefox, il vise le respect des standards du web et est open source. Il est à la 12e version et lui aussi a obtenu 100/100 au test Acid3 depuis la version 4 de Firefox (car SVG a été enlevé du test n°3).
- Presto pour Opera et Opera Mobile. Il en est à sa 12e version et a passé le test Acid3 en 2008 avec une version bêta de Preso, ce qui montre la qualité de ce moteur de rendu.
- Webkit d'Apple pour Safari, Safari mobile et Chrome. Tout comme presto, il a passé avec succès le test Acid3 alors qu'il était toujours en version bêta.

Si le moteur rendu est responsable de la majorité de l'affichage des éléments au sein des pages web, l'évolution des langages web a rendu nécessaire l'inclusion de moteurs JavaScript responsables de l'interprétation du langage du même nom. Le JavaScript étant le langage prédominant dans le développement de jeux vidéo en ligne accessible sans plug-in, ces moteurs ont une incidence particulièrement importante puisque ce sont eux qui feront fonctionner le cœur du jeu.

Chaque Navigateur possède son propre moteur JavaScript, mais Google Chrome possède sans doute le plus performant du marché et est talonné de près par celui de Firefox.

Cette première sous partie était surtout vouée à exposer les différents acteurs prenant part à la suite. Les 5 grands acteurs sont certainement voués à prendre part au développement des standards du web et c'est grâce à cela que le développement web et l'évolution des langages pourra sereinement continuer son court.

Nous pouvons donc maintenant rentrer dans le vif du sujet en étudiant les possibilités graphiques offertes par les langages web.

B - la 2D

Un jeu vidéo en 2D voit son espace de jeu représenté sur 2 axes. Le plus célèbre exemple de jeu en 2 dimensions est sans aucun doute Mario. Avant l'avènement de la 3D, nous pouvions trouver des simulations de jeu en 3 dimensions: le premier exemple est la 2,5D dans laquelle un personnage en 3D évolue dans un monde en 2 dimensions (par exemple dans *Alone in the Dark*) ou c'est l'environnement qui va évoluer de façon à donner un aspect de profondeur (comme dans le jeu *Mario Kart* par exemple). Le deuxième exemple est surnommé la perspective ¾, dans lequel le monde dans lequel le joueur évolue est incliné à 45 degrés pour donner une impression de progression sur 2 axes au lieu des 2 réels (utilisée par exemple dans le jeu *Zaxxon*).

Il existe 2 grandes technologies pour représenter des environnements en 2 dimensions dans un navigateur web sans utiliser de plug-in: SVG et canvas. Nous allons voir de quelles manières peuvent être utilisées chaque technologie, leurs avantages et inconvénients et surtout quels sont leurs débouchés pour la création d'un jeu vidéo en ligne.

1- SVG (Scalable Vector Graphics)

La technologie SVG est née en 1999 grâce à l'association de plusieurs sociétés au sein du W3C pour concurrencer les solutions propriétaires de Microsoft et Adobe.

Comme son nom l'indique, SVG va permettre l'intégration dans une page Web d'objets vectoriels. Cette technologie est très utilisée dans le domaine de la cartographie en ligne. SVG est supporté par les 5 grands navigateurs actuels (Chrome, Safari, Firefox, Internet Explorer et Opera), s'il existe des exceptions, je ne manquerais pas de les noter.

À travers le langage XML, il est donc possible de décrire des objets pour les intégrer au DOM (Document Object Model), définit la structure logique d'un document (HTML ou XML) et la manière d'y accéder et de le

manipuler. Cela veut dire qu'à l'instar de toutes (ou presque) les autres balises HTML du code source de la page, les éléments SVG vont pouvoir être édités de la même manière qu'une balise normale (avec du CSS, des événements JavaScript ou même à un rôle Aria). On assimile cette technologie à de la retained mode, mode où l'application agit sur des objets déjà créés dans une page, à cause de la présence en mémoire des éléments graphiques.

SVG possède un ensemble de fonctions de base permettant de dessiner (carrés, rectangles...) et d'agir sur ces objets.

Par exemple le bout de code suivant:

```
<svg height="50px" width="50px">// environnement de 50x50
<rect id="mon_rectangle" height="30px" width="40px"
fill="blue"/>
// rectangle de 30x40 rempli de bleu
</svg>
```

Donnera dans la fenêtre du navigateur:



SVG est intégrable dans le code de plusieurs manières: il peut être intégré directement en utilisant la balise HTML5 `<svg>`, en tant qu'élément extérieur (nécessite l'extension de fichier `.svg`) ou en tant qu'élément CSS.

En ce qui concerne les animations via SVG, si la technologie permet d'utiliser des animations de base (comme les translations par exemple), la présence des éléments SVG dans le DOM permettra d'interagir avec ces derniers grâce à JavaScript. Par exemple, en rajoutant à notre rectangle le bout de code suivant:

```
onclick="alert('vous venez de cliquer sur le rectangle');"
```

Nous obtiendrons une fenêtre surgissante nous affichant le message ci-dessus après avoir cliqué sur notre rectangle bleu.

Il est également possible d'interagir avec notre élément rectangle grâce à du CSS. Cette possibilité est d'ailleurs soucieuse, car les groupes de travail du W3C n'ont pas encore tranché sur la priorité des animations CSS3 sur les animations SVG, si bien que chaque navigateur décide pour l'utilisateur: Opera et Firefox supportent très bien le SVG alors que les navigateurs Webkit (Safari et Chrome) préfèrent plutôt les animations CSS3.

Voici une capture d'écran du tableau SVG provenant site web caniuse.com. Comme nous pouvons le voir, SVG basique est parfaitement supporté par l'ensemble des navigateurs récents. Il est néanmoins possible de contourner ce problème pour IE 8 en utilisant une librairie alternative, par exemple Raphael.js.

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser
								2.1	
						3.2	2.2		
						4.0-4.1	3.0		
	8.0	21.0	27.0			4.2-4.3	4.0		
	9.0	22.0	28.0	5.1		5.0-5.1	4.1	7.0	
Current	10.0	23.0	29.0	6.0	15.0	6.0-6.1	5.0-7.0	4.2	10.0
Near future	11.0	24.0	30.0	7.0	16.0	7.0			
Farther future		25.0	31.0						

Ensuite, concernant les performances de SVG: les éléments entrant tous dans la composition du DOM dès le chargement de la page, les performances seront réduites avec un grand nombre d'éléments. En effet, plus les éléments seront nombreux et complexes, plus le navigateur mettra du temps à afficher la page.

Le gros avantage de SVG réside donc dans sa capacité à décrire des formes très complexe sans demander un gros effort de calcul de la part de la machine: en effet, le navigateur dessinera tout ce que le SVG lui demandera de dessiner. De plus, SVG étant un format d'image vectoriel, la taille de la surface de travail ne sera pas prise en compte dans l'effort pris par le navigateur pour dessiner l'objet.

Un autre avantage de ce format et la possibilité de dessiner via des logiciels externes pour ensuite exporter nos images au format .svg puis les introduire dans la page web.

Enfin SVG étant un format d'image indépendant, la présence de javascript ne sera pas nécessaire pour l'affichage des images. Neanmoins, la plupart des animations et des evennements d'un jeu video etant gérés par Javascript, cet avantage n'en est pas réellement un.

2- Canvas

L'élément <canvas> est une spécialisation du HTML5. Il fut introduit à l'origine par Apple pour être utilisé dans Webkit pour les logiciels Dashboard et Safari. Les navigateurs utilisant Gecko l'ont ensuite adopté, puis Opera pour enfin être standardisé par le groupe de travail WHATWG.

Il faut également savoir qu'un bon nombre de spécialistes considèrent canvas comme l'outil numéro 1 dans la course a la mise a mort de Flash (dans une certaine mesure).

Canvas est particulièrement apprécié des développeurs de jeux vidéo non-adeptes des langages web, car les librairies graphiques de jeux vidéo étant historiquement très bas niveau (très proche de la machine), les développeurs seront plus tentés par celui-ci pour démontrer leur savoir-faire en termes de programmation.

Il faut considérer l'élément canvas comme une surface de dessin au pixel près appelée bitmap (littéralement "carte de bits"). Il est possible d'accéder à cette surface de dessin grasse à des fonctions JavaScript primitives. Ces fonctions sont à quelque chose près les mêmes que pour SVG (carrés, triangles, courbes...).

Par contre, la principale différence entre canvas et SVG est l'absence de mise en mémoire pour le canvas et il faut donc que le développeur se souvienne de tout ce qu'il a dessiné et où il l'a fait. Ceci peut représenter à première vue un défaut d'accessibilité de la part de Canvas, même si en théorie, tous les pixels du bitmap sont accessibles.

Donc à l'inverse de SVG où le DOM affiche tout ce qu'il va se passer, le canvas peut être considéré comme une boîte noire dont seuls le ou les développeurs auraient les clefs.

Par contre, ce défaut d'accessibilité et cette non-présence dans le DOM rend les interactions JavaScript plus difficiles à effectuer: pour SVG, l'objet étant défini dans le DOM, un simple événement onclick saura connaître de lui-même ses limites d'actions. Au contraire, en l'absence de précision, le même événement dans le canvas déclenchera soit un événement onclick dans n'importe quel endroit du canvas ou bien aucun événement du tout. Pour ce genre d'événement, il faut au préalable présicer les coordonnées des pixels du canvas concernés par l'événement: on en revient à un défaut d'accessibilité puisqu'il faut connaître les positions exactes des pixels, ce qui peut être relativement compliqué dans le cadre d'un objet en mouvement (qui sont plutôt très présent dans les jeux vidéo!).

Reprenons l'exemple de notre rectangle, mais cette fois-ci en rouge:

```
<canvas id="monCanvas" width="50px" height="50px" />
```

Le canvas, soit l'espace de dessin de 50 par 50 est défini de cette manière, mais contrairement à SVG, il n'y aura pas plus de détail, le reste se passant en JavaScript:

```

var canvas = document.getElementById("monCanvas");
var ctx = canvas.getContext("2d");
ctx.fillStyle = "rgb(255,0,0)";
ctx.fillRect(0, 0, 30, 40);

```

Ce qui donne une nouvelle fois:



De la même manière que pour SVG, caniuse.com montre que l'élément canvas sous sa forme basique est supporté par la majorité des navigateurs du parc informatique mondial.

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser
								2.1	
								2.2	
							3.2	2.3	
							4.0-4.1	3.0	
							4.2-4.3	4.0	
	8.0	21.0	27.0				5.0-5.1	4.1	7.0
	9.0	22.0	28.0	5.1					
Current	10.0	23.0	29.0	6.0	15.0	6.0-6.1	5.0-7.0	4.2	10.0
Near future	11.0	24.0	30.0	7.0	16.0	7.0			
Farther future		25.0	31.0						

Canvas est très intéressant dans une recherche de performance du rendu, en effet, l'approche programmatic permet d'activer les fonctions d'optimisation que le développeur juge nécessaire (contrairement à SVG ou c'est le navigateur qui décide de la manière dont il dessine les objets). Canvas semble donc approprié dans un contexte où le rendu sera plus chargé en élément. Au contraire, canvas sera en théorie moins performant

dans des contextes où la surface de travail est très grande, car le nombre de pixels est amené à augmenter.

Canvas possède un avantage offert par la communauté de développeurs soutenant HTML5: certains Framework ont déjà vu le jour afin de faciliter la création de jeux vidéo en 2D comme Jaws.js ou RPG.JS qui vont sans doute permettre d'intéresser certains développeurs soucieux de se lancer dans le développement JavaScript sans Framework.

Il semble donc déjà que Canvas se désigne comme plus intéressant dans la perspective d'un jeu vidéo complexe et multiplateforme.

Toutes ces informations amènent donc une question évidente pour la suite: laquelle de ces 2 technologies est la plus pertinente pour le développement de jeux vidéo dans un navigateur web?

3- SVG ou Canvas?

Commençons d'abord par récapituler les différences entre ces 2 technologies:

	<canvas>	SVG
Niveau d'abstraction	Basé sur les pixels (PNG dynamique)	Basé sur des formes & vecteurs
Eléments	Un unique élément HTML similaire à dans son comportement	De nombreux éléments graphiques qui font alors partie du Document Object Model (DOM)
Interaction	Modifiable uniquement par Script	Modifiable par Script et CSS
Modèle événementiel	Interaction utilisateur très granulaire (x,y de la souris)	Interaction utilisateur abstraite et gérée par le DOM
Performance	Les performances sont meilleures avec des petites surfaces et/ou un nombre important d'objets à l'écran (>10k)	Les performances sont meilleures avec un nombre inférieur d'objets (<10k) et/ou sur de plus larges surfaces de dessin

Nous avons vu précédemment que canvas est plus adapté dans le contexte de grandes surfaces de jeu. De plus, la taille des éléments ne rentrant pas en compte, il est tout à fait possible d'imaginer SVG comme

technologie de développement d'un jeu vidéo de type point and click, de réflexion ou de stratégie "basique".

En effet, le SVG étant une technologie de prédilection dans les domaines de la cartographie, des schémas et plans de toute sorte, notamment grâce à la possibilité de zoomer ou dézoomer sans perte de qualité graphique, le développement des styles de jeu noté plus haut sont tout à fait envisageables grâce à SVG: même si le nombre d'éléments peut s'avérer élevé et augmenter le temps de chargement, les interactions entre les objets étant moins fréquentes que dans d'autres styles de jeu vidéo (comme Mario qui gère des mouvements bien plus importants par exemple).

Venons maintenant vers canvas, comme je l'ai expliqué, tous les pixels de la surface du canvas sont manipulables. Il est donc bien plus aisé dans un premier temps de ne manipuler que des points précis. On peut donc par exemple penser à ne modifier que certains points d'une image: par exemple dessiner des pixels bleus sur une surface précise pour en faire un sabre laser dans le cadre d'un jeu reprenant la saga Star Wars.

De plus, étant donné que canvas ne met aucun objet en mémoire, le mouvement sera géré en effaçant puis redessinant à la volée une nouvelle image le temps de chargement est moins lourd que dans le cadre de SVG (qui lui devrait dans tous les cas charger toutes les images de mouvement possible même si ces dernières ne sont pas utilisées). Ces traitements ont bien sûr un cout: les calculs devant être effectués par le moteur JavaScript sont plus nombreux, mais étant donné que ces derniers sont de plus en plus performants, ils arrivent très bien à gérer toutes ces transformations.

Nous pouvons donc voir que canvas apparaît plus adapté aux petites surfaces comportant un grand nombre d'éléments affichés et modifiés en temps réel ce qui est un atout non négligeable dans le cadre des jeux vidéos. Au contraire, SVG paraît plus adapté aux jeux plus "calmes"

comportant un faible nombre d'éléments utilisables, mais dont la surface de jeu importe peu.

Un autre élément non négligeable en faveur de canvas est l'arrivée de technologie permettant l'accélération matérielle (comme par exemple CanvasMark) via la carte graphique pour aider le moteur JavaScript à rafraîchir les pixels du canvas.

Cependant Microsoft émet à l'heure actuelle quelques doutes à ce sujet: le géant informatique pense que donner l'accès à de tels éléments de la machine physique via le navigateur peut être risqué, car certains pirates pourraient cacher des malwares très embêtants dans les jeux vidéos utilisant ces technologies.

Concrètement, dans le cadre d'un jeu vidéo en 2D, si canvas a légèrement l'avantage par rapport à SVG, il sera nécessaire aux concepteurs de très bien étudier la structure du jeu en développement afin de choisir la technologie la plus adaptée à celui-ci.

Je n'ai pourtant pas encore évoqué un point intéressant: c'est qu'il est possible de mixer les 2 technologies afin de profiter de leurs avantages: l'API de canvas possède une méthode `drawImage()` pour dessiner n'importe quelle image à l'intérieur du canvas: nous pouvons donc imaginer la possibilité d'introduire dans ce dernier un élément svg très complexe à dessiner via JavaScript. Au contraire, nous pouvons imaginer un détourage d'une photo au pixel près via un élément canvas pour ensuite former des éléments svg destinés à former un puzzle.

Les possibilités offertes par les 2 technologies s'avèrent donc être découpées et le mélange de celles semble être semble être un choix très envisageable concernant les graphismes d'un jeu vidéo en 2 dimensions. Dans tous les cas, ce type de développement demandera toujours moins de ressources que l'utilisation d'une autre technologie (Flash ou Java par

exemple) puisque seuls un navigateur et un éditeur de texte sont nécessaires.

Cependant, canvas possède un atout qui fait définitivement pencher en sa faveur pour le développement de jeux vidéo: la gestion de la performance due à la programmation de l'ensemble de l'intérieur du canvas. De plus, canvas possède un atout indéniable pour l'avenir du jeu vidéo en ligne: WebGL qui permet l'utilisation de la 3D dans le navigateur web sans aucun plug-in à installer.

C - La 3D

La 3 dimension représente le monde dans lequel nous vivons tous les jours, avec 3 paramètres: hauteur, largeur et profondeur.

En 1992, le premier jeu vidéo en 3D en temps réel a vu le jour: il s'agissait du FPS (First Person Shooter) Wolfenstein 3D considéré par beaucoup comme une référence. La 3D subjective en temps réel donnait alors au joueur une impression de liberté sans limites: ce dernier voyait en direct le monde dans lequel il évoluait bouger en même temps que lui. Depuis, les jeux vidéo en 3 dimensions n'ont cessé de proliférer, offrant toujours plus de réalisme pour offrir au joueur une immersion totale. Historiquement, les premières images 3D sur un ordinateur sont apparues grâce à la bibliothèque OpenGL (Open Graphics Library), créée en 1992 par Silicon Graphics. OpenGL est aujourd'hui utilisable par tous les systèmes d'exploitation si la machine physique possède une carte graphique adaptée. Le Khronos Group a sorti bien plus tard OpenGL ES (OpenGL for Embedded System), une API bien plus légère destinée aux terminaux mobiles et consoles de jeux portables. Microsoft a lancé avec Windows95 la bibliothèque Direct3D offrant les mêmes fonctionnalités qu'OpenGL mais uniquement disponible sur

Windows dans un premier temps, puis sur XBox (la console de jeu de Microsoft) à sa sortie.

Le projet Farenheit (initiative de Microsoft et Silicon Graphics) a été une tentative d'uniformisation d'OpenGL et de Direct3D a été tentée, mais les difficultés financières de Silicon Graphics ont mis fin à celui-ci.

Pour accéder aux fonctions des bibliothèques 3D, il faut passer par l'utilisation d'un moteur 3D, qui va se charger de faire le lien entre l'utilisateur et la bibliothèque. Cependant, un moteur 3D ne gère que l'affichage des objets, il est d'ailleurs souvent combiné à un moteur physique et un moteur de jeu pour gérer les collisions (moteur physique) et les entrées utilisateurs (moteur de jeu).

La 3D sur le web n'a pas attendu 2009 et WebGL pour exister: en effet, dès 1997 il était possible grâce à Java3D de faire de la 3D accélérée matériellement. Plusieurs autres solutions ont vu le jour depuis, comme Flash ou JOGL, et de nombreux jeux sont apparus. Seulement, l'inconvénient de ces technologies est qu'elles nécessitent toutes un plugin à ajouter au navigateur et à mettre à jour pour pouvoir être utilisées. De plus, il y a 10 ans, envoyer l'énorme quantité de données que nécessite une application 3D était souvent trop important pour les connexions des utilisateurs. De plus, les ordinateurs des particuliers n'étaient pas suffisamment performants pour exécuter le code reçu. Ce n'est bien heureusement plus le cas aujourd'hui, car les connexions internet et les puissances des ordinateurs ont considérablement évolué. Le Khronos group a alors lancé sa spécification WebGL, permettant l'utilisation du standard OpenGL via du code JavaScript, ce langage étant lisible nativement par les 5 grands navigateurs.

Nous verrons donc dans un premier temps les conséquences de l'apparition de WebGL, puis ensuite qu'elles sont les alternatives à JavaScript dans le cadre du développement du jeu vidéo en ligne.

1 - WebGL

WebGL (pour Web -open- Graphics Library) est une spécification 3D créée par le Khronos Group. Elle permet via du code JavaScript d'afficher des animations 3D directement dans le navigateur via le standard OpenGL, sans plug-in extérieur (si le navigateur est suffisamment récent). Ainsi, le navigateur web ne va plus seulement utiliser son moteur JavaScript, mais va profiter des pilotes OpenGL de la carte graphique pour que cette dernière s'occupe de l'affichage.

Le rendu en WebGL se fait au sein de la balise HTML5 Canvas. Comme nous l'avons vu précédemment, celle-ci possède ses avantages et ses inconvénients. Mais avec WebGL ceux-ci sont différents puisque c'est maintenant la puissance de la carte graphique de l'utilisateur qui va permettre ou non l'affichage, plus seulement le navigateur lui-même.

Outre les avantages liés à Canvas, WebGL possède d'autres points forts: si le navigateur Web est récent, il sera capable de supporter l'API WebGL. Ainsi, contrairement à son principal concurrent, Flash, l'utilisateur n'aura qu'une seule installation à effectuer pour pouvoir lancer une application WebGL: celle d'un navigateur web performant.

Les problèmes liés à l'exécution de cette application graphique seront alors dus aux performances de la machine elle même, mais ce problème n'en est pas réellement un puisque c'est déjà le cas dans le cadre de l'installation d'un jeu vidéo ou une configuration minimale est requise.

Quel est donc l'avantage? Et bien aucune installation, mis à part celle d'un navigateur web n'est requise: il est donc possible de tester un jeu même si la machine n'est pas suffisamment puissante pour l'exécuter, même en sachant que ce test est voué à l'échec.

Au contraire, si un jeu vidéo est acheté en tant qu'application externe, ou même utilisée dans un navigateur via l'exécution d'un plug-in externe, les contraintes sont plus nombreuses:

- l'installation de l'application peut simplement ne pas marcher
- elle peut nécessiter le téléchargement d'un ou plusieurs plug-ins
- l'installation de ces plug-ins peut échouer

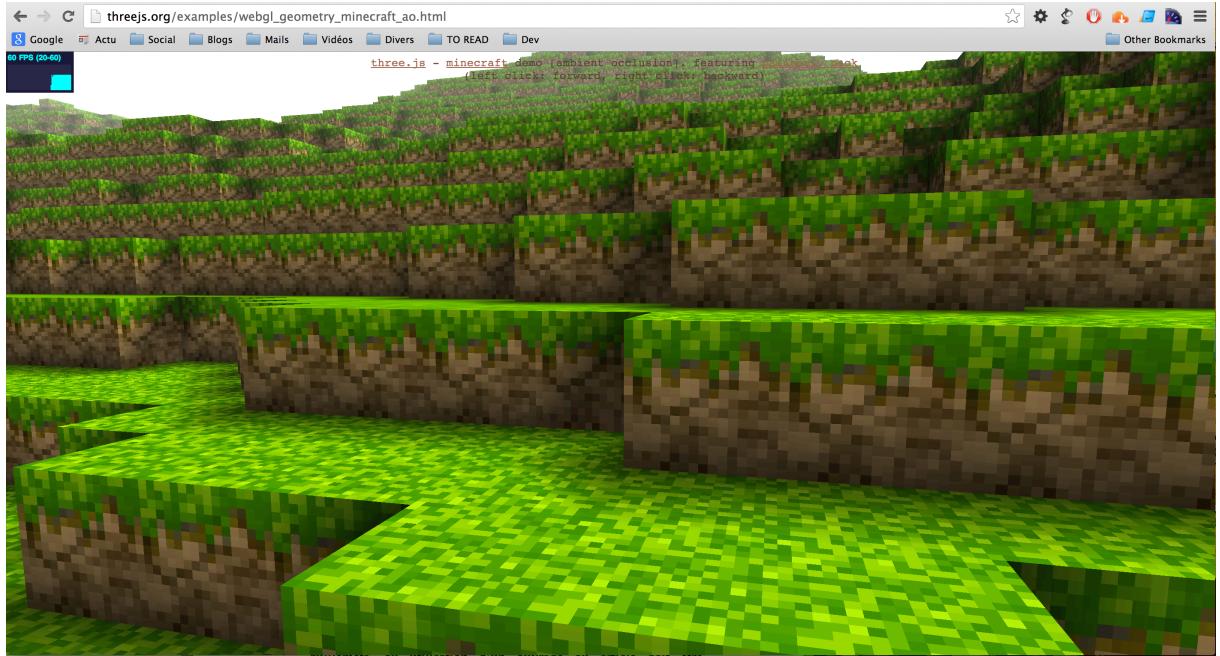
Le résultat est donc le même qu'en utilisant WebGL, a ceci près qu'il est plus long et que toutes ces installations prennent inutilement de la mémoire sur le disque dur.

Outre cet avantage pratique, il faut savoir que WebGL est basé sur OpenGL qui, si cela entraîne quelques limites, va permettre aux animations de pouvoir s'exécuter dans les navigateurs mobiles, ce qui n'est par exemple pas le cas avec Flash.

Un des principaux avantages de WebGL est sa forte documentation en ligne. Étant une API open source, les forums, wikis, tutoriels et réseaux sociaux (reddit a un fil consacré à WebGL) pullulent sur le net afin d'offrir aux développeurs le maximum de retours, infos et aide que possible.

WebGL est également "simple" à prendre en main: en effet, tout comme pour canvas, seule une connaissance du JavaScript et des balises HTML5 est requise pour écrire une application en WebGL.

On pourra également noter que certaines API ont vu le jour afin d'aider les développeurs dans leur utilisation de WebGL: par exemple le très réussi Three.js qui pour certains est amené à devenir pour WebGL ce "jQuery est à JavaScript aujourd'hui". Outre son grand nombre de fonctions aidant à l'utilisation de l'API native de WebGL, Three.js permet l'utilisation de scripts dans Blender (logiciel libre de modélisation et rendu 3D) pour convertir les objets en json et les utiliser dans notre environnement 3D.



Voici un exemple du jeu Minecraft développé grâce à WebGL est Three.js lancé depuis mon ordinateur personnel avec Chrome 28.

Un autre avantage est qu'utiliser WebGL dans une application ne requiert que l'utilisation de JavaScript, de ce fait, de la même manière que pour les jeux vidéo en 2 dimensions, seuls un navigateur et un éditeur de texte seront nécessaires.

Enfin, WebGL est supporté nativement par 4 des 5 grands navigateurs (Chrome, Opera, Safari et Firefox), et même si Internet Explorer ne supporte pas encore cette technologie dans sa version actuelle la version 11 devrait le faire. De plus, il est possible d'utiliser un plug-in pour qu'Internet Explorer supporte WebGL dans ses versions 9 et 10.

Cependant, il existe certaines limites à l'utilisation de WebGL: la première vient d'être citée au-dessus: Internet Explorer ne reconnaît pas encore cette technologie et elle nécessite l'utilisation d'un plug-in pour être reconnue dans ce navigateur. Ceci n'en fait pour le moment pas un atout suffisamment important pour forcer les développeurs de jeux vidéos à délaisser Flash pour WebGL dans le cadre d'un développement d'un jeu en

3D (Internet Explorer représentant 25% de parts de marché). Microsoft s'est défendue en annonçant que l'API WebGL n'était pas fiable au niveau de la sécurité. Kenneth Bradley Russell (co-développeur de Java OpenGL et aujourd'hui employé chez Google dans la branche WebGL de l'entreprise) a répondu qu'aucune API Web permettant l'accélération matérielle via la carte graphique n'était 100% fiable au niveau de la sécurité, Silverlight (la technologie propriétaire de Microsoft pour le Web3D) inclue. Par ailleurs, certains membres de la communauté HTML5 ont accusé Microsoft de ne pas reconnaître WebGL pour favoriser ses propres API (comme Silverlight) et donc retenir les développeurs sur Windows. Microsoft a quand même annoncé que WebGL serait supporté à partir d'Internet Explorer 11, de ce fait cette limite n'est encore que temporaire.

Ensuite, Apple et Google ne voulaient au début pas de WebGL dans leurs navigateurs mobiles respectifs. Google a tout de même cédé face à certains constructeurs mobiles comme Sony et Samsung qui ont intégré WebGL à leurs firmwares Android, le rendant supportable sur leurs téléphones. Tout comme pour Microsoft, certains professionnels accusent les géants informatiques de ne privilégier que les intérêts économiques: en effet, la mise à disposition d'applications (notamment les jeux vidéo) directement dans le navigateur mobile n'obligerait plus les utilisateurs et développeurs à passer par les systèmes payant de magasins d'applications.

Voici le tableau issu de caniuse.com de la technologie WebGL:

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser
						2.1			
						2.2			
					3.2	2.3			
					4.0-4.1	3.0			
	8.0	21.0	27.0		4.2-4.3	4.0			
	9.0	22.0	28.0	5.1	5.0-5.1	4.1	7.0		
Current	10.0	23.0	29.0	6.0	15.0	6.0-6.1	5.0-7.0	4.2	10.0
Near future	11.0	24.0	30.0	7.0	16.0	7.0			
Farther future		25.0	31.0						
Parent feature: Canvas (basic support)									

Un autre défaut de WebGL est donc ses failles de sécurité: en effet, l'accès direct a la carte graphique via un navigateur est une aubaine pour les pirates qui pourraient introduire dans les applications utilisant WebGL des malwares bénéficiant d'un droit d'entrée direct a certains programmes normalement cachés de l'ordinateur de l'utilisateur: pour prendre un exemple de la vie réelle, cela revient a laisser un coffre de banque sans aucune surveillance. Encore une fois, le projet WebGL étant supporté par de très grandes entreprises (Apple, Google, Nvidia, Mozilla,...), cette limite ne sera que temporaire.

De ce fait, certains navigateurs bloquent par défaut les accès à la carte graphique, à cause des dits problèmes de sécurité liés à WebGL. Aussi, cela force l'utilisateur à effectuer des manipulations plus ou moins complexes suivant le navigateur.

Ensuite, il faut avouer que si les ordinateurs actuels sont suffisamment performants pour afficher un jeu vidéo développé en 3D via WebGL, ce n'est pas encore le cas de toutes les tablettes et mobiles. Même si l'iPad2 se targuait d'avoir des performances graphiques 10 fois supérieures a celles de son prédecesseur, le rendu s'en trouvait encore un peu lent. De plus, toutes les tablettes et mobiles ne sont pas aussi performants que l'iPad d'Apple. Cependant, encore une fois, ces limites ne sont que temporelles puisque Flash n'étant pas supporté sur la plupart des systèmes d'exploitation mobiles et les fabricants dotant leurs tablettes et téléphones d'équipement toujours plus performant, ce n'est qu'une question de temps avant que WebGL ne fonctionne parfaitement.

WebGL est donc une technologie prometteuse qui même si elle est encore très jeune (environ 4 ans) a de beaux jours devant elle. Si elle devient au même titre que HTML5 un standard du web, on peut aisément imaginer que le développement de jeux en utilisant cette API va très vite se développer. En tout cas, les premiers exemples d'exploitation graphique de WebGL font rêver l'ensemble du web (exemples sur <http://>

www.chromeexperiments.com/ ou <http://threejs.org/>). Il est aussi important de signaler que WebGL a fortement contribué à l'évolution des navigateurs et des moteurs JavaScript.

Je n'ai par contre pas encore évoqué une contrainte de toutes les technologies citées plus haut: le langage de développement: JavaScript. Le fait est que celui-ci ne fait pas l'unanimité chez l'ensemble des développeurs web et de jeux vidéo. Ces derniers lui préfèrent les langages historiques de développement de jeux comme le C ou le C++. C'est pourquoi j'ai décidé d'évoquer les possibilités qu'ont les développeurs pour s'affranchir du JavaScript tout en offrant un environnement graphique performant dans les navigateurs web actuels.

Le fait que JavaScript soit comme son nom l'indique un langage de script, il est aujourd'hui possible d'utiliser d'autres technologies pour ensuite les compiler en JavaScript.

L'autre solution pour s'affranchir de JavaScript étant l'utilisation d'un plugin tiers permettant l'exécution de l'application.

En optant pour l'une ou l'autre des solutions, on se retrouve dans tous les cas avec une liste de technologies prometteuses.

2- Alternatives à Javascript

Comme je l'ai expliqué, JavaScript n'est pas apprécié de tous les développeurs, même si à l'heure actuelle, les performances du JavaScript sont plus qu'honorables. Ainsi certains développeurs ont trouvé plusieurs parades: la première est d'utiliser un autre langage informatique pour ensuite compiler le résultat en JavaScript. L'autre solution est d'utiliser des logiciels tiers pour ensuite exporter le jeu vidéo sur la plateforme voulue, tout en gardant qu'une seule source.

Les Langages Complilés

Historiquement, les développeurs de jeu vidéos sont plutôt des utilisateurs de C ou de C++. Et il faut aussi se rendre à l'évidence: jamais un moteur JavaScript ne pourra seul reproduire les performances d'un code natif. S'il existait un moyen de lancer du code natif via ActiveX, cela restait cantonné aux systèmes Windows. Cependant, Mozilla et Google ont débarqué avec des technologies permettant l'utilisation de code natif à l'intérieur du navigateur en garantissant la sécurité pour l'utilisateur.

Ainsi, Alon Zakai, chercheur pour la Mozilla Foundation a développé le logiciel Emscripten destiné à compiler à la volée (ie juste avant le lancement de l'application) du C++ utilisant directement les fonctions d'OpenGL en JavaScript utilisant l'API WebGL. Ce logiciel a été développé dans le cadre du projet OdinMonkey (moteur JavaScript actuel de Firefox) de Mozilla introduit dans Firefox 22 destiné au jeu vidéo en ligne. Ce projet donne d'ailleurs des résultats époustouflants puisque le projet BananaBread (reprenant le moteur de jeu Unreal Engine), exploitant asm.js qui va rendre le JavaScript plus proche du langage natif, donne un résultat pratiquement identique à un jeu vidéo natif.

Emscripten fonctionne sur ce principe: le code C ou C++ est transformé via une machine virtuelle de bas niveau en JavaScript et le rend utilisable par un navigateur web.

L'avantage de ce genre de logiciel étant de pouvoir porter sur le web des programmes existant déjà en C++ sans avoir à fournir l'effort conséquent de tout réécrire en JavaScript.

Le projet Emscripten a un concurrent direct chez Google, puisque la firme de Mountain View a lancé en décembre 2011 le Google Native Client permettant lui aussi de compiler à la volée du C++ en JavaScript. Google Native Client trouve un franc succès chez l'ensemble des professionnels des jeux vidéos puisque Square Enix a annoncé en juin 2012 l'utilisation de cette technologie pour porter sur le web le jeu vidéo Tomb Raider

Series et dont les graphismes n'ont rien à envier a ceux de la PS3 ou de la XBox 360. De plus, le jeu vidéo Quake a également été porté sur le web grâce à cette technologie. Le problème avec le Native Client est qu'il est basé sur l'API Pepper, qui n'est pas supportée par les 4 autres navigateurs. Il est néanmoins possible de lancer des applications compilées grâce à Native Client dans Firefox grâce a un plug-in (ce qui va a l'encontre des efforts du HTML5).

Une autre technologie proposée par Google est son nouveau langage de développement: le Dart destiné à remplacer JavaScript, car ce dernier possède "des problèmes qui ne peuvent être résolus avec une évolution du langage". En effet, Google pense que les limites du développement JavaScript ont été atteintes et que même si des progrès concernant l'optimisation du JavaScript ont été faits (compilation à la volée, accélération matérielle, exploitation des processeurs), ce langage n'atteindra jamais les performances d'un langage compilé.

Dart est destiné a lui aussi être compilé en JavaScript pour être compris par tous les navigateurs actuels, mais il peut également être utilisée depuis un serveur PHP ou dans une machine virtuelle intégrée au navigateur (c'est le cas pour Chrome) si ce langage est adopté par tous les géants du milieu du web. Il est donc tout à fait envisageable de voir un jour des jeux vidéos développés grâce à cette technologie.

Cependant Dart a reçu un accueil mitigé de la part de la communauté des développeurs web qui lui reprochent:

- de ne rien apporter de réellement nouveau, citant en exemple coffeescript qui lui aussi est compilé en JavaScript. De plus, certains se demandent pourquoi Google a tant investi dans les technologies traditionnelles du web (HTML, CSS et JavaScript) pour ensuite présenter Dart.
- la plupart des lacunes de JavaScript sont destinées à être comblées dans la prochaine version. La communauté reproche donc à Google d'aller

contre les efforts de standardisation du web, chose que Google reproche souvent aux autres acteurs du milieu (notamment à Microsoft).

- Google n'a pas considéré les avis de la communauté JavaScript pour le développement de ce langage.

Dans le domaine du jeu vidéo, l'accueil a été moins brutal et certains ont même annoncé Dart comme le futur langage de prédilection du jeu vidéo HTML5.

Le problème de toutes les technologies citées ci-dessus vient dans le domaine du mobile: si les machines utilisant Android vont certainement bénéficier à termes de ces technologies, rien ne dit pour l'instant qu'Apple et Microsoft le permettront dans leurs propres navigateurs mobiles, que cela soit pour des raisons économiques (plus d'application store nécessaires) ou bien concurrentielles: que cela soit pour l'iPhone, iPad ou les machines utilisant Windows Mobile, il est nécessaire d'avoir en sa possession un ordinateur Apple pour développer sur iPhone/iPad et Windows pour Windows phone. Or le principe même du développement web est le peu de ressources qu'il nécessite (un éditeur de texte et un navigateur).

Si Google et Mozilla se posent pour l'instant comme les entreprises aidant le plus au développement des jeux vidéos en ligne grâce à toujours plus d'inventivité, d'autres acteurs proposent des solutions totalement différentes en matière de développement de jeux vidéo.

Les logiciels spécialisés

La volonté et la nécessité de produire des jeux de plus en plus disponibles sur la multitude de plateformes actuelles ont incité certains développeurs à se tourner vers des logiciels pouvant exporter les applications sur toutes les plateformes souhaitées.

Ainsi, nous avons à disposition des logiciels toujours plus innovants facilitant la propagation du jeu vidéo. Même si la plupart d'entre eux nécessitent l'utilisation d'un plug-in pour être exécutés dans un navigateur web, ils sont d'une telle importance dans la communauté qu'ils se devaient d'être présenté en tant qu'options.

Je vais donc évoquer dans cette partie 3 technologies qui permettent ce genre de développement: Flash Stage 3D, Silverlight 3D et Unity 3D.

Flash Stage3D a été lancé en 2011 pour permettre l'amélioration des applications Flash en 3D. L'avantage de Flash 3D est que le langage est déjà maîtrisé par un bon nombre de développeurs qui utilisaient cette technologie pour produire des jeux vidéo en 2D. Seulement, le nombre d'inconvénients est aussi conséquent: tout d'abord, Flash n'est pas supporté sur les navigateurs mobiles. Il est cependant possible de distribuer ce genre d'applications sur les plateformes de vente (application store), mais cela ne correspond pas à la vision générale de ce mémoire, qui voudrait surtout que toutes les applications soient disponibles directement depuis un navigateur web.

Enfin, du point de vue des développeurs web Flash a le défaut de n'intégrer aucune des technologies de développement (HTML, CSS et JavaScript).

Silverlight 3D est le plug-in de développement d'application web de Microsoft. Celui-ci fonctionne de manière plus ou moins similaire à Flash. Cependant, Silverlight 3D n'est accepté sur aucun navigateur mobile, et il nécessite de posséder Visual Studio, donc un ordinateur Windows, pour pouvoir être utilisé en programmation. De ce fait, il s'éloigne de l'accessibilité habituelle offerte par le développement web qui encore une fois peut être réalisée grâce à un simple éditeur de texte.

De plus, Silverlight ne possède pas la documentation de Flash et de WebGL ce qui peut le rendre difficile à utiliser si l'on n'est pas habitué à .Net.

Passons donc à celui qui me paraît le plus prometteur: Unity 3D: Unity est un middleware (logiciel qui sert de pont à différentes applications informatiques) spécialisé dans la 3D en temps réel. La possibilité d'exporter en quelques clics son application sur différentes plateformes a séduit immédiatement les développeurs de jeu vidéo souhaitant diffuser au maximum leurs applications. Par contre, de la même manière que Flash, une application Unity ne pourra être exportée vers mobile que par l'intermédiaire d'une réelle application, et non par le navigateur, ce dernier nécessitant d'ailleurs un plug-in pour pouvoir exécuter une application Unity (sauf pour Chrome). Unity a la particularité d'offrir 3 langages de développement: le JavaScript, le C# et le Boo, le résultat de l'application étant compilée en MONO. Ce choix dans le langage de développement a permis à Unity de susciter l'intérêt de développeurs venant de plusieurs mondes différents. Il est important de noter que Unity est améliorables par l'utilisateur lui-même puisqu'il est possible de coder soit même ses propres améliorations.

L'avantage d'Unity est que tout le logiciel permette de simplifier le développement du jeu vidéo et permet à des équipes plus réduites de produire de très bons jeux vidéos. À noter que pour l'aspect 3D, il est possible pour un graphiste 3D d'importer ses travaux depuis la plupart des logiciels. Google permet l'utilisation de son Native Client pour compiler des projets Unity dans son navigateur web.

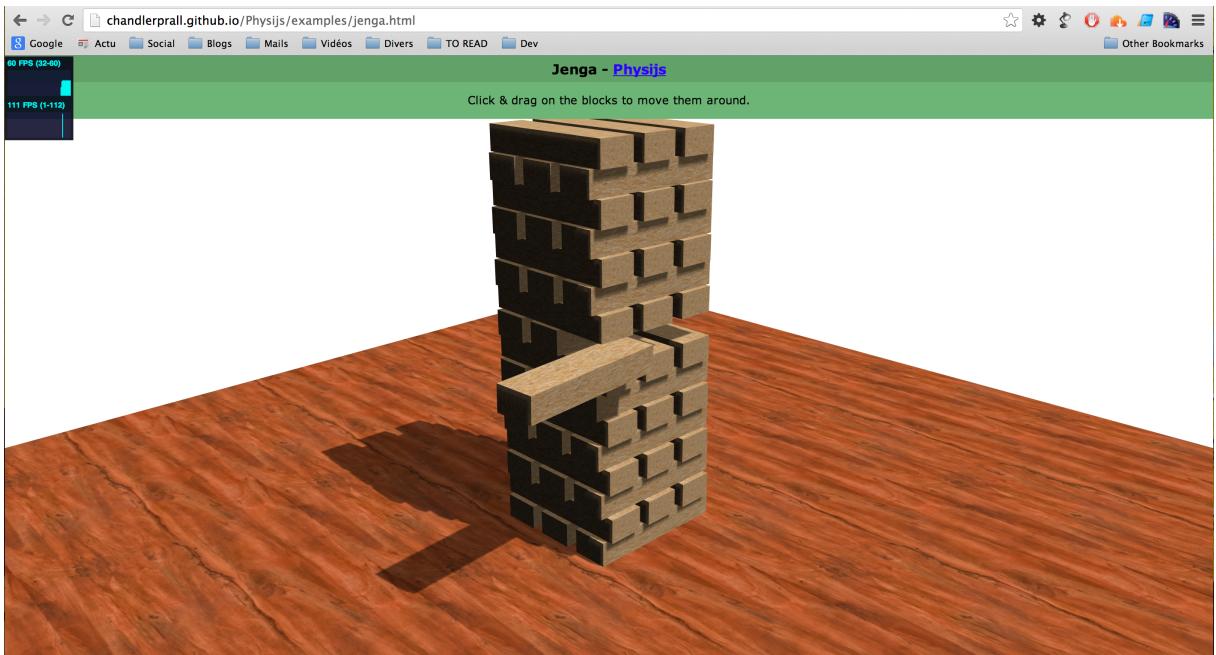
Par contre, il est impossible d'intégrer dans le projet Unity des composants HTML ou CSS. De plus, à l'instar de ses prédecesseurs, l'export d'un projet sur mobile devra être distribué par l'intermédiaire d'un magasin, ce qui encore une fois est différent de l'accès via un simple navigateur.

En conclusion, les possibilités offertes pour le développement des graphismes de jeux vidéo sont encore très nombreuses et pour certaines très prometteuses. Il convient donc pour l'instant de suivre de très près

les évolutions des technologies et de leur appartenance aux standards du web pour décider laquelle utiliser. Dans tous les cas, pour pouvoir passer par un navigateur sans utiliser de plug-in, il faudra certainement prévoir dans tous les cas une utilisation (ou une compilation) de JavaScript. Du côté des navigateurs, même si Chrome et Firefox se présentent comme les navigateurs de prédilection des futurs joueurs et développeurs, on voit que des progrès considérables ont été effectués afin de faire évoluer le jeu vidéo en ligne. Les géants de l'informatique travaillent pour l'instant main dans la main avec le W3C pour décider quels seront les standards de demain et prendre de vraies décisions concernant les applications web (dont font partie les jeux vidéos).

Il convient donc de noter qu'en tenant compte de la rapidité de l'évolution des technologies web et des actuels résultats de développement graphiques offerts par les développeurs, la 3D sans plug-in dans un navigateur web est à deux doigts de devenir un standard et d'être pleinement exploitée.

Je terminerai cette partie par un exemple, qui à mon sens est plus démonstratif que les mots, de ce qui est réalisable en utilisant WebGL et Three.js, deux technologies évoquées plus haut:



Cet exemple provient de threejs.com et a été exécuté sur mon ordinateur personnel depuis le navigateur Chrome 28.

Nous venons donc de voir à quel point l'évolution des navigateurs permet aujourd'hui de penser pouvoir égaler les performances graphiques des plateformes pour le développement de jeux vidéo directement dans un navigateur web. Seulement, si les graphismes sont une part importante du jeu vidéo, ils ne constituent pas à eux seule l'entité qu'est un jeu vidéo: en effet, un jeu vidéo, c'est aussi du son, un système de jeu agréable (aussi appelé gameplay), des interactions entre objets, une intelligence artificielle digne de ce nom pour les adversaires et enfin un aspect multijoueurs, que cela soit en réseau ou directement dans le même environnement (même si l'utilisation d'un navigateur en ligne suppose bien évidemment le multijoueur en réseau).

Tous ces autres aspects notés vont faire qu'un jeu va passionner les utilisateurs et les fidéliser en communauté. Ces aspects sont également susceptibles de rendre un jeu culte: par exemple, n'importe quel joueur, du plus passionné au simple joueur "du dimanche" connaît la musique de Mario, sait qu'il va pouvoir affronter ses amis virtuels ou physiques dans

Call of Duty ou va apprécier la possibilité d'utiliser un volant pour jouer à Gran Turismo.

Nous allons donc voir dans cette partie de quelle manière les technologies web peuvent permettre à un jeu vidéo web de bénéficier de tous les points cités. Dans un premier temps, j'effectuerai un état des lieux sur les technologies utilisées dans un jeu vidéo standard, pour ensuite en expliquer la manière dont les langages web arrivent à les imiter. Enfin, je parlerai des possibles évolutions que le jeu vidéo en ligne et multiplateforme peut offrir un gameplay inédit et particulièrement original en mélangeant les plateformes utilisant des navigateurs web.

II - Le jeu vidéo, ce ne sont pas que des graphismes.

Nous avons vu dans la première partie que le moteur graphique du jeu vidéo gérer l'affichage. Aussi, pour gérer le reste des aspects du jeu, un nombre important de moteurs différents est utilisé:

- le moteur physique du jeu: il va permettre de gérer les mouvements des entités du jeu, leurs collisions, l'application des forces (accélération, déformation, etc.) sur ces entités.
- le moteur son, qui comme son nom l'indique, va s'occuper de l'ensemble des interactions sonores du jeu vidéo.
- le moteur d'entrées/sorties: cette partie s'occupe de la gestion des périphériques externes du jeu vidéo. Il n'est pas encore possible de contrôler nos jeux vidéo uniquement par la pensée (et même dans ce cas, il faudrait gérer les entrées et sorties), de ce fait ce moteur va s'occuper d'une partie du gameplay du jeu vidéo: à savoir celle offerte par la lecture des périphériques de jeu choisis: puis-je jouer avec mon clavier? Mon clavier et ma souris? Une manette? Un volant? De plus, ce moteur va également s'occuper de l'interface utilisateur et de la lecture et l'écriture de données, par exemple pour charger ou sauvegarder une partie en cours.
- l'intelligence artificielle (IA) va quant à elle permettre l'interaction avec les personnages non jouables du jeu. Cela peut être simple, par exemple dans le cas d'un jeu d'échec où une seule IA est gérée, mais peut s'avérer beaucoup plus compliquée dans le cadre de jeux offrant plusieurs adversaires (par exemple dans le cadre d'un jeu type Beat Them All), et encore plus compliquée si elle mélange plusieurs adversaires, alliés et personnages neutres (comme dans le cadre d'un jeu de foot par exemple).
- le moteur réseau qui sera utile dans le cadre de jeux vidéo en ligne.

Tous ces moteurs réunis (avec le moteur graphique vu en partie 1) forment ce qui est appelé le moteur de jeu, qui va , a l'aide de code informatique, coordonner les fonctionnalités offertes avec les règles du jeu (seconde partie du Gameplay) pour offrir a l'utilisateur le jeu vidéo final tel qu'il a été vu par l'équipe de développement.

Dans un premier temps, nous allons donc voir de quelle manière les moteurs de jeu des jeux vidéo standards (comprendre, hors du navigateur) utilisent ces moteurs et de quelle manière ils sont mis en place. Cette première partie va surtout permettre de faire une comparaison plus claire avec les jeux vidéo web.

A - Etat des lieux du développement standard

Aujourd'hui, l'importance des jeux vidéo est telle qu'elle représente un marché plus important que le cinéma! Ceci est avant tout dû à l'important nombre de plateformes de jeux (téléphones, consoles, ordinateurs, tablettes, etc.). Cependant il faut bien comprendre que chaque plateforme est différente et nécessite certaines spécificités de développement. C'est pourquoi il faut bien identifier la cible de son jeu avant d'en envisager le développement sous peine de se retrouver confronté à certains problèmes.

J'aborderai dans un premier temps une liste de plateformes pouvant accueillir des jeux vidéo et leur spécificités. Puis nous verrons ensuite quelles sont les différentes technologies et outils utilisés pour développer ces jeux vidéos. Enfin, je me lancerai dans l'explication d'une technologie de plus en plus présente dans l'univers du développement des jeux vidéo: l'option de l'application "hybride" pour franchir la barrière des plateformes.

1 - les différentes plateformes de jeu

Il existe un nombre incroyable de plateformes de jeu. De nos jours, tout ce qui est électronique peut potentiellement être une plateforme de jeu: c'est pourquoi, avant d'envisager le jeu lui-même, il faut d'ores et déjà faire un choix pour l'inventeur du jeu en fonction des moyens qu'il a. Le choix d'une plateforme va avoir des incidences sur le futur du jeu vidéo en question: les moyens à déployer pour développer un jeu sur PS3 ne sont pas les mêmes que sur iPhone par exemple. Dans le second cas, développer un jeu tout seul est envisageable, mais seulement dans quelques cas exceptionnels dans le premier tant les ressources demandées sont grandes.

La première plateforme de développement, et sans doute la plus connue, est son ordinateur personnel. La raison est toute simple: aujourd'hui la plupart des foyers des pays développés possèdent un ordinateur (71% des Français possédaient en 2010 au moins 1 ordinateur selon médiamétrie). Développer un jeu sur cette plateforme est donc avantageux si l'on veut pouvoir le distribuer à grande envergure. De plus, l'avantage d'un ordinateur est qu'il est possible d'utiliser pratiquement n'importe quelle technologie de développement et les bibliothèques sont très nombreuses et très documentées.

Le problème du développement de jeu vidéo sur ordinateur peut se trouver au niveau du matériel des machines: en effet, tous les ordinateurs ne sont pas équipés des mêmes systèmes d'exploitation et des mêmes composants. Il est donc possible de voir son jeu marcher parfaitement sur son ordinateur personnel, mais ne même pas démarrer sur celui de son voisin.

Ensuite vient le développement sur tablettes tactiles (dont les statistiques montrent qu'elles seront en plus grand nombre dans les familles en 2014) et téléphones portables. Pour l'instant, les systèmes d'exploitation les plus présents sur tablettes sont Android de Google et iOS d'Apple. Le premier problème est que ces systèmes nécessitent l'utilisation de 2 langages

différents pour des applications natives: Java pour Android et Objective-C pour iOS. Sauf s'il a la possibilité (et le temps) de développer sur plusieurs plateformes, le développeur est d'ores et déjà confronté à un choix. De plus, dans le cas d'Android, le développeur va devoir penser à rendre compatible son jeu avec les différentes machines utilisant le système d'exploitation (comme pour les PC). iOS a donc un avantage puisque seuls les iPad sont équipés d'iOS et ne présentent presque pas de problèmes de compatibilité (sauf si la machine est trop vieille).

L'avantage des tablettes et des téléphones est de pouvoir travailler sur un gameplay différent grâce aux possibilités tactiles offertes par ceux-ci. Par contre, dans tous les cas, il faudra passer par l'achat d'une tablette ou d'un téléphone et d'une licence (environ 100\$ pour chaque système d'exploitation) pour pouvoir tester son jeu sur et le mettre en vente (ce qui peut représenter de grosses sommes pour un particulier).

L'autre possibilité pour les développeurs est d'opter pour l'option des consoles de jeu. Ces machines sont en général faites pour le jeu vidéo, ce qui présente un avantage par rapport aux tablettes, téléphones et ordinateurs personnels. Mais encore une fois, le nombre de consoles est grand, et le développeur va devoir travailler en amont pour bien identifier la meilleure console pour le jeu qu'il veut produire (par exemple, il est impensable de voir un jeu comme Final Fantasy XIII sur une Game Boy). Par contre, tout comme l'iPad, le développement sur une console en particulier permet d'outrepasser les problèmes de compatibilité puisque toutes les consoles sont les mêmes (une PS3 reste une PS3 même chez son voisin). Aussi, le développement sur console demande souvent certaines ressources, dont la console elle-même, ce qui peut représenter un investissement dans le cadre d'un développement amateur.

Enfin, il est possible d'envisager la possibilité du multiplateforme pour un jeu vidéo. Cependant, il faut savoir que le développement d'un jeu en tant qu'amateur est souvent compliqué, et que même avec une équipe

complète (en général un graphiste, un développeur, un compositeur et un scénariste au minimum) arriver à terminer son jeu sur une plateforme est long et compliqué. Dans le cadre d'équipes professionnelles, les travaux sont en général répartis en équipes spécialisées: une équipe par plateforme est souvent sollicitée.

Donc, comme nous venons de le voir, avant même d'envisager l'utilisation d'une technologie de développement plutôt qu'une autre pour son moteur de jeu, il faudra qu'un travail important en amont soit réalisé afin de savoir quelle plateforme est la plus adaptée au jeu.

Passons maintenant à un inventaire des différentes technologies utilisées par plateformes.

2 - Les technologies standards de développement

Il existe pour le développement d'un jeu vidéo une multitude de techniques plus ou moins compliquées. Nous en verrons ici 3 différents: l'utilisation de Game Maker, dont le principal avantage est qu'ils ne nécessitent pas de compétences en programmation, l'utilisation des langages BASIC, qui ne sont pas très compliqués à apprendre, et enfin l'utilisation de la programmation pure, qui permet de faire presque tout ce que l'on veut.

Les Game Maker

Un Game Maker est un logiciel permettant de se passer de programmation (si l'utilisateur le souhaite) pour "développer" un jeu vidéo. Cette solution est la plus rapide pour quelqu'un n'ayant aucune compétence en développement. Ainsi, l'utilisateur va pouvoir se concentrer sur les autres aspects de son jeu.

En général, un Game Maker est un moteur de jeu sous la forme d'un puzzle: les différents bouts de code sont préfaits par le logiciel et

l'utilisateur les assemble pour obtenir son jeu. Parmi les Game Maker les plus connus, on retrouve par exemple RPG Maker, permettant de développer des jeux type RPG (jeu de rôle) sur PlayStation et Windows sans taper la moindre ligne de code.

Comme nous l'avons vu, l'avantage de ce type de solution est l'accessibilité. Cependant, ce type de développement rend pratiquement impossibles les améliorations puisque les bouts de code du futur moteur de jeu seront toujours les mêmes, sauf si une nouvelle version du Game Maker sort.

L'utilisation de langages BASIC

Un langage BASIC (Beginner's All-purpose Symbolic Instruction Code) est fait pour les débutants. Il représente un bon compromis entre programmation pure et utilisation d'un Game Maker. De plus, il existe certains logiciels utilisant ces langages dans un but spécifique au jeu vidéo comme blitz ou PureBasic. Cependant le problème de ces langages se trouve tout d'abord au niveau de la performance: il ne sera jamais possible pour eux d'atteindre le niveau d'un langage de programmation. De plus, les logiciels permettant leur utilisation sont pour la plupart payants

La programmation pure

La programmation pure est la voir la plus difficile, mais c'est également la plus efficace puisqu'elle permet de réaliser tout ce que l'on veut et qui produit les meilleurs résultats. Les professionnels du jeu vidéo utilisent la plupart du temps cette solution.

En théorie, il est possible de développer un jeu dans n'importe quel langage, mais dans la pratique certains sont plus pratiques que d'autres voire même imposés par les constructeurs pour certaines plateformes.

Je vais maintenant faire une présentation des langages de programmation les plus utilisés par plateformes:

Les consoles

Les consoles de jeu viennent en tête de liste, car ce sont des outils spécialement conçus pour le jeu vidéo. Sont elles pour autant les plus accessibles en matière de développement? Et bien en vérité, cela dépend bien souvent du constructeur. Je n'évoquerai ici que les 3 constructeurs de consoles les plus importants de ces dernières années: Microsoft avec sa XBox 360, Nintendo avec sa WiiU et sa DS, et enfin Sony avec sa PS3 et sa PSVita.

Microsoft est celle qui a rendu le développement de jeu le plus aisé pour la XBox. En effet, le géant informatique met gratuitement XNA à la disposition des développeurs. XNA est une serine d'outil permettant de programmer ses jeux pour XBox en C#. Il est disponible aujourd'hui avec la plupart des versions (même les gratuites) de l'IDE de Microsoft Visual Studio.

Concernant Nintendo et ses consoles, que ce soit la DS ou la WiiU, c'est le C++ accompagné des bibliothèques et compilateurs adaptés à chaque console pour développer.

Enfin, Sony a rendu le développement sur ces consoles plus compliqués (officiellement, pour ne "permettre qu'aux vrais professionnels de faire de très bons jeux avec une très longue durée de vie" selon le président de Sony CE Kazuo Hirai). Mais dans la pratique, développer sur PS3 est une plaie pour tous les développeurs, Marvin Donald, directeur du développement du jeu Darksiders sur PS3 a d'ailleurs fait une comparaison tout à fait hors du commun à ce propos. De plus, le kit de développement permettant le déploiement d'application sur PS3 n'est disponible qu'en version payante pour la somme de 7500 euros.

Concernant le développement sur PSP, il existe des kits de développement gratuits pour pouvoir développer des jeux vidéos, et contrairement à la PS3, où le langage C++ est utilisé nativement, c'est le C# qui permet de développer des jeux pour la console portable de Sony.

Comme nous pouvons le voir, le C++ est prédominant dans le monde des consoles, seulement le développement d'un jeu vidéo n'est pas qu'une affaire de langage, et ce n'est pas parce que notre programme est écrit en C++ qu'il marchera sur toutes les consoles. En effet, les architectures des consoles étant toutes différentes, il faut adapter son jeu en fonction de celles-ci.

Les mobiles

Tout comme les consoles, il existe différentes plateformes mobiles, les 3 plus utilisées dans le monde sont iOS d'Apple, Android de Google et Windows Phone de Microsoft.

Le premier souci du développement mobile est que 3 langages différents sont utilisés pour les 3 plateformes: Apple demande l'utilisation d'Objective-C, Google de Java et Microsoft de XNA (donc C#) pour Windows Phone.

Les ordinateurs personnels

Ici, n'importe quel langage de programmation peut être utilisé.

Seulement, développer soit même un moteur de jeu complet sans l'utilisation d'une bibliothèque est long. De plus, des moteurs préconstruits (par exemple BASS pour le moteur son ou GNE pour un moteur réseau) et multiplateformes sont disponibles:: pourquoi donc s'en priver, sachant que c'est le moteur de jeu coordonnant tous les autres qui fera un jeu réussi.

Comme nous venons de le voir, la programmation de jeux vidéo, en plus d'être complexe, est très diversifiée. Ceci peut donc poser des problèmes de moyens ou de temps aux plus petites entreprises, voire même au moyennes si elles souhaitent se lancer dans le multiplateforme. C'est pour

cette raison que certaines compagnies ont commencé à prendre la tangente en utilisant des logiciels hybrides permettant le développement de jeux vidéos multiplateformes à partir d'une seule source.

3 - L'option du langage unique

L'option du développement de jeux vidéo à langage unique est à mi-chemin entre le jeu disponible sur le Web sans installation de plug-in et l'application native. Cette option est généralement offerte par des logiciels qui vont compiler le langage de base en la version adaptée à la plateforme. Ils sont généralement utilisés, car ils permettent de créer plus rapidement un jeu vidéo tout en profitant d'une version finale disponible dans le langage natif lui permettant ainsi de bénéficier des market places des distributeurs.

Le logiciel le plus prometteur dans ce type de programmation est certainement Unity. Déjà évoqué dans la première partie pour le développement d'applications 3D, Unity permet également de construire soit même tout un moteur de jeu, à la fois pour ordinateurs, mobiles, tablettes et même consoles de jeu vidéo.

Comme expliqué dans la première partie, il est possible de développer avec Unity une application dans un seul langage (JavaScript, C# ou Boo) pour ensuite le compiler en une application finale. Unity existe en plusieurs versions: la version gratuite qui permet de développer des jeux pour ordinateurs personnels et pour navigateurs web, puis une version payante (1500€) qui rajoute de nouvelles possibilités. Il est possible de mettre à niveau la version payante pour iOS (nécessite Mac OSX), Android, BlackBerry, consoles Sony, WiiU et XBox 360. Il faut compter environ 750€ par mise à niveau de sa version pro ou alors 1500€ par version pro spécialisée.

L'acquisition de Unity peut paraître conséquente en termes de prix, mais si une petite ou moyenne entreprise veut pouvoir rivaliser avec les grands

studios de développement de jeux vidéo et développer des jeux multiplateformes, c'est à mon sens un investissement valable, car le prix dépensé pour Unity sera économisé par la suite sur les différentes compétences demandées aux développeurs qui formeront un groupe homogène en termes de capacités.

Concernant les performances des applications Unity, si elles sont certes un peu en dessous des applications directement développées en langages natifs, car la compilation de l'application ne peut pas entre autant contrôlée qu'avec ces derniers, elles restent tout à fait honorables. De plus, les constructeurs ne cessant de dévoiler des machines toujours plus puissantes et à intervalles réguliers, les développeurs n'arrivent pratiquement pas à en exploiter la pleine puissance (par exemple, la PS4 va sortir en novembre 2013 alors que la PS3, pourtant âgée de 7 ans, n'est pas encore exploitée à son plein potentiel).

Une autre solution, qui est de plus très utilisée par les professionnels du jeu vidéo, est l'Unreal Developpement Kit (UDK) d'Epic Games qui va permettre d'utiliser le moteur de jeu Unreal Engine pour construire son jeu vidéo et pouvoir l'exporter sur Windows, Mac OSX, iPhone, PS3 et XBox. L'avantage d'UDK est qu'il est distribué gratuitement par Epic Games, de distribuer des jeux dans un but commercial moyennant une licence de 99 dollars. Epic Games demande quand même 25% des revenus commerciaux si ceux-ci dépassent la barre des 50000\$ par jeu. Autant dire que l'utilisation de ce moteur de jeu est une aubaine pour les entreprises n'ayant pas la possibilité de développer des moteurs spécifiques à leurs jeux. Concernant l'utilisation, UDK est comme Unity3D composé en partie d'une interface graphique pour la partie 3D (généralement mise en place par les graphistes), mais aussi d'une interface de développement pour construire le jeu vidéo. Cette interface demande l'utilisation d'Unreal Script, un langage propriétaire d'Epic Games qui permet d'oublier la gestion du matériel physique.

Aussi, même s'il ne correspond pas trop à la problématique de ce mémoire puisqu'il ne permet pas le développement de jeux web, UDK est un élément ayant une grande importance dans le développement de jeux vidéo, et il est normal de ce demander si Epic Games permettra un jour la compilation vers des langages purement Web.

En conclusion, nous pouvons voir qu'à l'heure actuelle, le développement de jeux vidéo en langage natif est d'une grande diversité, qui peut s'avérer déconcertante pour les plus petites entreprises. Seulement, les jeux vidéos en ligne dans un navigateur prenant de plus en plus d'importance au fur et à mesure de l'évolution d'internet, il convient de se poser la question suivante: les capacités des langages web peuvent-elles permettre ce genre de développement dans les navigateurs sans l'utilisation de plug-in?

B - Les capacités du web pour répondre au développement historique de jeux

Les technologies web évoluent toujours plus vite, si bien qu'aujourd'hui, un navigateur web est vrai couteau suisse permettant à l'utilisateur de pratiquement tout faire sur le net: regarder des films, écouter de la musique, travailler et communiquer en équipe, etc.

Les jeux vidéo n'échappent pas à cette règle, et si nous avons vu dans la première partie que les navigateurs étaient capables d'afficher des graphismes sans avoir à rougir des applications natives quand est-il du reste des composants d'un jeu vidéo?

Et bien pour faire court: les technologies web permettent aujourd'hui à un navigateur, sans l'utilisation d'un plug-in, de réaliser l'ensemble des composants d'un jeu vidéo.

Nous allons donc voir dans cette partie quelles sont ces technologies, et dans quelles mesures elles peuvent être mises en place en confrontant les différents composants du moteur de jeu à celles-ci.

1 - Le moteur de son

Les navigateurs d'aujourd'hui sont parfaitement capables d'émettre des sons via les enceintes de l'ordinateur, en témoignent les nombreuses vidéos, radios ou même applications musicales en ligne (comme Deezer ou Spotify Web Player par exemple). La technologie privilégiée auparavant était le Flash.

Seulement, le HTML5 a permis l'introduction de la balise `<audio>` et du Web Audio API qui ont changé la donne: il devient possible de lire du son dans un navigateur web sans plug-in.

La balise audio a l'avantage d'être supportée par tous les navigateurs web du marché, mais également d'appartenir au DOM. Ceci permet donc, grâce à l'utilisation de JavaScript, de la déclencher quand bon nous semble.

Seulement, dans un jeu vidéo très complexe, on retrouve généralement une multitude de sons: le chargement de tous ceux-ci en même temps que la page peut s'avérer très long. De plus, il est impossible pour la balise audio d'accéder au signal analogique de la musique (pour faire varier l'intensité du son par exemple) ou de la déclencher le selon le positionnement ou la direction d'un tiers.

Heureusement, l'API Web Audio est beaucoup plus performante, cela venant du fait qu'elle va tout exécuter en JavaScript (la différence entre la balise audio et cette API est plus ou moins la même qu'entre SVG et canvas). Elle permet de varier le rythme et les appels de ces sons. Ceci est très utile, il est ainsi possible de générer des sons de façon toute à fait

aléatoire, ou bien alors de répéter ce même son de manière infinie (par exemple dans le cas d'un tir de mitrailleuse).

Ensuite, JavaScript et l'API Web Audio vont permettre de faire varier le son en fonction de la position d'un élément du canvas: par exemple si le personnage principal passe à coté d'une horloge dans un monde en 3D, le son sera plus ou moins fort selon la distance et la position de ce personnage par rapport à cette horloge.

Enfin, il est possible grâce à l'API Web Audio d'accéder au signal analogique d'un son, et de le modifier: ceci est extrêmement utile si la vitesse doit être gérée. Par exemple, si une voiture vous passe à coté à grande vitesse, le son n'est pas le même à 1 mètre de vous quand 100 mètres: il est possible de gérer cela grâce à cette API.

A mon sens, la balise `<audio>` et l'API Web Audio sont complémentaires et l'utilisateur va devoir choisir entre les 2 au moment de faire son jeu (Web Audio étant plus compliquée à maîtriser). Mais il est clair qu'elles permettent de couvrir l'ensemble de ce que demande un jeu vidéo en terme de son.

Cependant, l'API Web Audio ne fait pas encore l'unanimité et selon caniuse.com, elle n'est disponible pour le moment que sur Chrome, Safari, Safari mobile, Opera et Firefox. De plus, aucune indication n'est pour le moment donnée sur un éventuel support sur les autres navigateurs.

2 - Le moteur réseau

Le moteur réseau va trouver sa principale utilité dans un jeu multijoueur. Ceux-ci représentent désormais une importante quantité de jeux vidéo au point que certains joueurs n'achètent plus que les modes multijoueurs (par exemple dans le cas de Call Of Duty où les histoires sont très courtes). On peut également noter l'avènement de jeux comme World Of Warcraft qui est uniquement jouable en ligne avec d'autres utilisateurs.

Les technologies web ont développé des techniques pour permettre à l'utilisateur d'un navigateur web de mettre à jour ses données sans avoir à recharger sa page. Cette technologie s'est démocratisée sous le nom d'AJAX. AJAX permet d'exécuter de façon asynchrone en JavaScript des requêtes HTTP sur le serveur. C'est grâce à ce genre de technique qu'il est possible de discuter en ligne avec ses amis sur Facebook par exemple.

Seulement, le W3C ne voit pas d'un très bon oeil ce détournement d'HTTP de son usage initial (déconnecté), et travaille donc sur un nouveau standard du web pour recevoir des informations d'un serveur sans rafraîchissement: Websocket.

Websocket est un protocole de communication réseau à part entière destiné à devenir un standard du web. Concrètement, cela va permettre d'ouvrir une connexion permanente entre le navigateur client et le serveur pour échanger des données (contrairement à AJAX où une connexion était ouverte à chaque fois).

Web socket est encore une jeune technologie en cours de spécification, et chaque nouvelle technologie vient avec son lot de problèmes: dans le cas de Websocket, certains proxys n'accepteront pas une connection client serveur permanente pour des raisons de sécurité et couperont simplement cette connexion. Ceci peut être très problématique, surtout en plein milieu d'une partie.

Du côté des navigateurs, Websocket est supporté par tous, ce qui est rare et a le mérite d'être très utile.

De la même manière, un bon nombre de langages serveur supportent WebSocket et permettent un grand nombre de connexions en même temps. Cette gestion de la performance peut être très problématique au niveau du serveur puisque le nombre de messages sortant du serveur est

égal aux carres du nombre de messages entrants, car dans le cadre d'un jeu multijoueur, chaque joueur a besoin de savoir, ne serait-ce que la position des autres.

3 - Le moteur d'entrees/sorties

Comme nous l'avons vu, le moteur d'entrée et sortie permet la gestion de la sauvegarde de données mais aussi la gestion des périphériques de jeu.

La gestion des données

La gestion des données est très importante dans le jeu vidéo: en effet, pouvoir enregistrer et charger sa partie est primordial quand on sait que le temps passe, pour finir certains jeux se comptent, en dizaines d'heures. Il existe même des jeux qui n'ont pas de fin, par exemple World Of Warcraft. Il est donc impensable de penser perdre toute sa partie en quittant son jeu.

C'est pour cette raison que la sauvegarde de données est cruciale. Seulement, de quelle manière enregistrer sa partie dans un jeu web?

La première solution pourrait être l'utilisation de bases de données, ou les états du joueur au moment de la sauvegarde seraient enregistrés.

Seulement, si le jeu a du succès, le nombre de données et le traitement de celles-ci pourraient vite devenir insoutenables.

C'est pourquoi l'utilisation d'une autre spécification HTML5, l'API Web Storage permet, à l'instar des cookies, de stocker des données dans le navigateur web. Ainsi, c'est le joueur qui est responsable de sa sauvegarde et non le serveur distribuant le jeu vidéo.

Ainsi, à l'instant de certaines applications comme Outlook ou Thunderbird qui utilisent les données de la dernière connexion pour travailler hors ligne, il est possible de garder sa dernière sauvegarde dans les fichiers du navigateur pour pouvoir jouer même si on ne dispose pas de connexion.

Les 2 techniques ont leurs avantages et leurs inconvénients: dans le cas de Web Storage, il ne sera pas possible de pouvoir reprendre sa partie depuis un autre navigateur. Dans le cas de l'utilisation d'une base de données, c'est la gestion de celles-ci qui s'avérera compliquée, mais il sera possible de la reprendre sur n'importe quel navigateur compatible avec le jeu.

La gestion des périphériques

Le moteur d'entrée et sortie permet également de gérer les périphériques de jeu (comme une manette par exemple).

Si JavaScript est tout à fait capable de gérer les entrées générées par les claviers et souris d'une machine, l'évolution des supports de jeu lui permet bien d'autres choses aujourd'hui.

Dans un premier temps, il est possible grâce à JavaScript de reconnaître des événements impliquant les écrans tactiles, ce qui peut être bien évidemment un plus dans le cadre d'un jeu vidéo destiné aux tablettes ou mobiles. Mais encore mieux: la Gamepad API (uniquement sur Chrome et Firefox pour l'instant) permet l'utilisation de manettes directement dans un navigateur: cela représente donc une innovation majeure si le jeu est destiné à être lancé depuis le navigateur d'une télévision ou d'une console de jeu.

4 - Les autres moteurs

Les autres composants du moteur de jeu sont réunis, car ils ont tous la possibilité d'être écrits en pur JavaScript. Aussi, cette sous partie a pour but de montrer les évolutions technologiques permettant de pallier une des faiblesses de JavaScript.

La faiblesse des technologies web se trouve dans JavaScript: qui a la particularité de s'exécuter dans un seul thread. Ceci est assez réducteur quand on sait que les processeurs actuels possèdent jusqu'à 8 coeurs logiques.

Cette faiblesse peut poser problème, car dans l'hypothèse d'un traitement important (comme la gestion de l'IA), cela va bloquer la page web, et ainsi massacer l'expérience utilisateur. De plus, les navigateurs possèdent un système de protection contre les scripts mettant trop de temps à se terminer.

La raison de cette faiblesse est qu'à l'origine, aucune application web n'était envisageable lors de la création de JavaScript. De plus, même si les développeurs donnaient l'impression que plusieurs choses se passaient en même temps, ce n'était en fait que le décalage de certains appels de fonction, ou la variation de certains éléments du DOM qui donnaient cet effet.

L'API Web Worker répond à ce problème, en permettant à des scripts de s'exécuter en tache de fond, sans incidence sur la page principale. Le JavaScript a donc la possibilité de s'exécuter sur plusieurs threads!

Le premier avantage de l'API Web Worker est qu'elle est supportée par tous les navigateurs web actuels. Par contre, les web workers ne permettent aucun accès au DOM, ce qui rend son utilisation avec SVG compromise.

Cependant, les web workers sont particulièrement efficaces dans des scénarios impliquant le traitement de l'image (comme le fait le moteur physique par exemple): en utilisant les données d'un canvas, on peut cibler les traitements sur des points précis. Bien entendu, plus nous aurons de coeurs, plus le traitement sera rapide. On peut également envisager d'utiliser les web worker pour coder l'Intelligence artificielle et la rendre plus performante, puisque dans un laps de temps moins court, elle pourra envisager plus de possibilités (très pratique pour un jeu d'échec).

5 - Faciliter ce développement via le web

Nous venons de voir quelles sont les technologies utilisées pour réaliser les différentes parties d'un moteur de jeu. Seulement, toutes les développer tout seul peut prendre énormément de temps, qui plus est pour un développeur ne connaissant pas très bien le JavaScript.

Nous allons donc voir dans ce paragraphe s'il existe des solutions permettant de faciliter le développement de jeu vidéo.

Les moteurs de jeu web

De la même manière qu'il existe des moteurs de jeu préfabriqués pour les langages classiques de développement, il existe des moteurs de jeu, spécialistes ou non, pour les technologies web.

Il est donc possible de trouver des outils gratuits et payants afin de pouvoir accélérer le développement de son jeu: par exemple, si on veut développer un RPG en 2D sur son navigateur, RPG.js est la solution recommandée. RPG.js est d'ailleurs basé sur Ease.js qui est une librairie issue de la suite CreativeJS. Ces 2 solutions pour faciliter le développement sont gratuites.

Si l'objectif sont les jeux vidéo de plateforme en 2D, le moteur de jeu open source Quintus est la solution adaptée.

En ce qui concerne la 3D, nous avions évoqué en première partie Three.js, et bien cette API peut également être utilisée en tant que moteur de jeu physique. PhiloGL est quand à lui un moteur de jeu 3D utilisant WebGL pratiquement complet.

Il existe également des solutions payantes plus ou moins performantes, GameMaker HTML5 (99\$) ou Construct 2 (99€ pour un amateur, 329€ dans un but commercial) offrent par exemple de vraies solutions graphiques, au lieu de tout développer en codant.

Cependant, il faudra toujours devoir consacrer une partie de la logique au développement en JavaScript. De plus, nombreux sont les jeux d'ores et déjà développés et les travailleurs du jeu vidéos sont pour la plupart spécialistes d'une autre technologie. Aussi, il est important d'envisager le portage ou la compilation comme possibilité externe.

Portage et Compilation

Beaucoup de développeurs voudraient viser le développement web pour leurs jeux, mais ne veulent pas s'embêter à réécrire l'existant: il existe donc certaines méthodes pour convertir le code déjà écrit en JavaScript.

Nous l'avons vu en partie 1, Emscripten et Native Client permettent de convertir du C++, maîtrisé par la plupart des développeurs de jeux professionnels, en JavaScript.

Mais il en existe d'autres, par exemple Mandreel, un SDK permettant de convertir ses jeux iOS, Mac, PC et Android vers JavaScript. C'est donc une aubaine pour une entreprise possédant déjà des jeux dans une ou toutes ces plateformes et voulant en faire un jeu Web.

Cependant, il faut bien comprendre que le portage est aussi à double tranchant: certains jeux développés pour une plateforme spécifique n'auront des performances optimales que pour cette plateforme. Et certains développeurs perdront plus de temps à vouloir porter leur jeu et comparer les performances qu'à développer à nouveau leur jeu dans la technologie appropriée.

Il est également important de noter que le chemin inverse est possible: en effet des plateformes de développement, par exemple Ludei, permettent d'exporter son code HTML5 vers toutes les autres dans leurs langages natif. Même s'il ne s'agit pas exactement du même principe que de lancer son jeu directement dans son navigateur, l'utilisation d'un seul langage de programmation pour plusieurs plateformes est une autre possibilité à prendre en considération si on envisage le développement HTML5. De plus, les performances et supports des technologies des navigateurs n'étant pas homogènes pour l'instant, cette solution est probablement la plus performante à l'heure actuelle.

Ensuite, certains développeurs optent pour une tout autre solution: nous avons parlé en partie 1 de l'utilisation d'un langage unique, et bien la solution de l'hybridation est toute aussi utile. En utilisant des solutions par exemple Phonegap, il est possible de développer le cœur du jeu vidéo en utilisant HTML5 et JavaScript puis de la compiler afin que ce même jeu puisse être utilisé comme une application native. Cette solution est un petit peu à mi chemin entre le tout natif et le tout web. De plus, ce genre d'application est généralement acceptée par la plupart des magasins en ligne et ne pose généralement pas de soucis de compatibilité puisque chaque plateforme voit sa compilation changer. Cependant, on pourra quand même noter quelques soucis de performances par rapport à une application native.

En conclusion, nous venons de voir que les technologies Web avancent toujours plus vite pour répondre au développement de jeux vidéo traditionnel. Et au jour d'aujourd'hui, elles commencent même à présenter une alternative de plus en plus intéressante.

Seulement, la possibilité de l'utilisation de langages web dans un but multiplateforme a un avantage indéniable sur la programmation classique: elle offre des possibilités de jeux très intéressantes. De plus, nous voyons qu'aujourd'hui les constructeurs de consoles rivalisent d'ingéniosité pour

proposer des systèmes de jeux innovants (comme la Wii de Nintendo ou le Kinect de Microsoft). C'est pourquoi nous allons voir dans cette dernière sous partie de quelle manière le Web peut lui aussi apporter de la nouveauté chez les joueurs.

C - Des possibilités inédites de jeu

Pour certains développeurs, le web doit être considéré comme une plateforme à part entier avec ses propres possibilités. Nous allons voir dans quelle mesure le web peut changer certains aspects du gameplay du jeu vidéo, tout d'abord en évoquant les avantages de la plateforme web: nous allons voir quels peuvent être ces avantages dans le cas de l'utilisation d'une seule machine. Puis ensuite nous verrons quelles sont les possibilités si l'on veut mélanger plusieurs plateformes en même temps. Enfin, nous terminerons par une petite vue à court terme de ce que pourra offrir le jeu vidéo web dans les 2 prochaines années.

1 - Les possibilités de la plateforme Web

La première possibilité de la plateforme est offerte par les URL. En effet, celles-ci peuvent transmettre des données aux autres utilisateurs. De plus cette fonctionnalité est totalement multiplateforme puisque tout support pouvant accéder à internet peut lire les URL.

Aussi, ce transfert de données peut être très utile, notamment dans l'optique d'un jeu en ligne et a vocation sociale: il suffirait par exemple de générer un lien avec ses coordonnées géographiques puis le diffuser sur un réseau social ou un chat pour inviter ses amis à nous rejoindre pour une partie.

Ensuite, en termes d'accessibilité, la plateforme web est probablement l'une des plus personnalisables. Ainsi, on peut même penser aller jusqu'à

introduire du code externe au jeu lui-même. Prenons un exemple, vous construisez vous-même votre IA pour ensuite la faire combattre contre vos amis (ou leur propre IA), ainsi, non seulement le jeu est divertissant, mais en plus il a des vertus éducatives. Sans forcément aller jusque-là, les possibilités de personnalisation sont décuplées étant donné que la popularité du web.

Cette accessibilité peut également faire participer la communauté à des avancées via la génération de plug-in fait pour le jeu. Cela existe déjà dans les cas de certains jeux dont les versions piratées (ou non) offraient des modifications par rapport aux versions originales: par exemple dans GTA où il était possible de remplacer les voitures du jeu par de vrais modélés.

Enfin, il faut comprendre que même si un jeu est développé pour un ordinateur de bureau, le fait qu'il soit sur le Web le rend accessible par n'importe quel ordinateur. Ce qui veut dire que même un jeu nécessitant une sauvegarde n'est plus uniquement utilisable via son propre ordinateur, mais potentiellement via tous les ordinateurs (si la puissance de ceux-ci permet de lancer le jeu).

Tout ceci ne représente pas une vraie innovation de jeu en elle-même puisque pratiquement tous les jeux vidéo en sont capables. Néanmoins, il était important de rappeler une nouvelle fois que le jeu vidéo web n'a rien à envier aux autres.

Nous pouvons donc maintenant passer à une autre approche du jeu vidéo: un jeu vidéo dont les plateformes ne seraient plus la limite et pourraient être mélangées.

2 - Vers le mélange des plateformes

Le premier exemple de mélange des plateformes qui vient à l'esprit serait une reprise du jeu: par exemple, imaginez vous en pleine partie de votre jeu préféré, mais également dans l'obligation de partir de devant votre ordinateur. Et bien, si votre jeu vidéo est également jouable sur mobile, vous n'avez qu'à sauvegarder la partie sur votre ordinateur pour mieux la reprendre après le départ sur votre mobile, le tout en gardant la même configuration qu'au début (ce qui peut s'avérer très utile, car comme je l'ai dit, certaines durées de jeux peuvent atteindre des dizaines d'heures).

Ensuite, nous pouvons imaginer un jeu combinant en direct les possibilités des différentes plateformes. Par exemple en rendant possible l'accès à des améliorations du gameplay pour les appareils pouvant être connectés, mais ne supportant pas le jeu dans son ensemble. Prenons encore une fois un exemple, issu du jeu en HTML5 DayZ, un FPS où il faut collecter des objets pour se défendre contre des assaillants. Et bien en se connectant à son compte avec son téléphone pendant la partie, il est possible d'accéder à une carte qui va indiquer les ennemis et les meilleurs objets en temps réels. En conséquence, l'utilisation d'un 2e écran permet l'amélioration de l'interface utilisateur, et donc par extension du gameplay.

Enfin, il est également envisageable de combiner les plateformes sous un autre aspect: utiliser les terminaux mobiles comme manette de jeu. Le jeu sur clavier n'est pas adapté à tous les styles de jeu, et tout le monde ne veut pas acheter une manette pour son ordinateur. C'est pourquoi la possibilité de convertir son smartphone ou sa tablette en manette est encore une fois une amélioration du gameplay. À l'heure actuelle, certains jeux web par exemple Brass Monkey utilisent ce système, mais la partie smartphone nécessite un peu de code natif.

Cependant, le W3C est a commencé en 2011 a mettre au point l'API WebRTC (permettant la communication en temps réel) va entre autres permettre la communication entre 2 navigateurs différents en temps réels. Ainsi il est tout à fait envisageable de voir se connecter un navigateur

mobile et un navigateur desktop en même temps. Le problème c'est que Web RTC n'est pour l'instant disponible par défaut que dans Firefox, Firefox Mobile, Chrome et Opera. Ericsson a annoncé en 2012 le premier navigateur iOS et Android compatible WebRTC pour téléphone, appelé Bowser. Il n'y a par contre aucune nouvelle concernant Safari et Microsoft a annoncé l'inclusion d'une API similaire dans Internet Explorer.

Il faut cependant noter une chose: même si elle tend à se réduire au fil des années, la différence de puissance entre un téléphone et un ordinateur personnel est encore importante. De plus, un jeu vidéo sur mobile n'a absolument pas le même gameplay. Il faut donc à l'heure actuelle penser avant tout adapter son jeu en fonction de la plateforme, en proposant une autre approche. Par exemple, ne vous attendez pas à pouvoir jouer à Battlefield 3 sur votre iPhone: il n'en a pas la puissance. Et même s'il l'avait, le nombre de touches d'un PC est trop important pour un jeu de ce calibre. Afin de pouvoir garantir à l'utilisateur la possibilité de continuer à jouer à son jeu favori sur la même partie, il faut prévoir une version mobile du jeu: plus pauvre graphiquement et dont le gameplay correspond plus à un téléphone. Dans le cas de Battlefield, nous pourrions imaginer passer d'un FPS à un jeu à la 3e personne avec vue à 45°.

3 - A quoi pouvons nous nous attendre?

Nous venons de le voir, le jeu vidéo en HTML5 a évolué incroyablement vite ces dernières années. Voici donc, en regroupant les informations venant de professionnels du web, ce que nous pouvons attendre du développement de jeu en HTML5 dans les 2 prochaines années.

Tout d'abord, l'utilisation de l'API WebRT permettra sûrement de faire passer de plus en plus de jeu en HTML5 pour des jeux natifs, ce qui selon les professionnels rendraient meilleure l'expérience utilisateur.

Les professionnels voient également arriver dans un futur proche la possibilité de jouer dans un navigateur directement intégré à sa télévision. Des professionnels comme ceux de l'entreprise Ouya s'y préparent déjà et proposent une manette pour télévision.

Enfin, le premier jeu originalement conçu en HTML5 devrait connaître son premier succès et ainsi attirer plus efficacement les grands studios de développement, mais aussi favoriser l'évolution des navigateurs vers le haut et non dans la guerre commerciale.

En conclusion, il faut bien se rendre à l'évidence: le web offre aujourd'hui au jeu vidéo de nouvelles armes pour rivaliser avec les outils et technologies historiques de développement et dans certains cas, il offre même plus de possibilités. Que les développeurs choisissent de construire des jeux vidéos pour être utilisés directement dans le navigateur ou de les compiler via un framework pour pouvoir bénéficier des magasins en ligne, les solutions sont variées et méritent toutes une réflexion précise sur le choix de celles-ci.

Ce choix peut avoir une importance cruciale pour le futur du jeu vidéo (et même plus largement d'une application en général).

Néanmoins, les équipes doivent prendre en considération d'autres points que le développement du jeu lui-même, c'est pourquoi nous verrons dans la prochaine partie quelles sont les incidences des jeux vidéos en HTML pour les acteurs de ce monde vidéoludique. Quels sont les apports pour les développeurs? Y'a-t-il des inconvénients à distribuer un jeu en HTML5 dans un but commercial? Quels points peuvent inciter les joueurs à délaisser leurs consoles pour jouer dans leur navigateur?

III - Les implications des jeux HTML5 pour les acteurs du jeu vidéo.

Nous l'avons déjà vu en introduction: le jeu vidéo est aujourd'hui une industrie dont les bénéfices se comptent en dizaines de milliards de dollars par an. Aussi, l'arrivée d'une nouvelle plateforme sur le marché représente bien évidemment un intérêt potentiel pour les investisseurs. Nous avons également vu que l'informatique a considérablement évolué grâce aux jeux vidéo, donc pour quoi ne pas se servir des jeux vidéo comme facteur d'évolution du web? De plus, l'arrivée d'une nouvelle plateforme de jeu est souvent une source de curiosité pour certains qui veulent se rendre compte par eux même quelles sont les possibilités offertes par celle-ci.

Nous verrons donc dans un premier temps ce qu'impliquent les jeux vidéos en HTML5 en ce qui concerne les joueurs. Puis nous nous concentrerons sur les nouveaux aspects que cela peut changer dans les entreprises de développement de jeux. Puis enfin, nous répondrons aux éventuelles questions sur les manières de vendre et/ou de tirer des bénéfices de ces jeux en HTML5.

A - Ceux qui jouent

C'est une évidence, mais ce seront en partie les joueurs qui mettront ou pas les jeux vidéo en HTML sur le devant de la scène. Pourtant, dans les faits, un joueur ne se lèvera jamais un matin en se disant qu'il veut jouer uniquement à un jeu en HTML5, il se dira seulement qu'il veut jouer. Ce sont donc d'autres facteurs qui vont influencer ses choix: cela peut être la performance, la beauté des graphismes, l'accès sur mobile, la facilité de jeu ou bien d'autres aspects.

C'est pour cette raison qu'il faut se pencher sur les implications d'un jeu vidéo en HTML5: quels sont ses atouts pour séduire les joueurs? Quelles sont au contraire ses faiblesses?

Le principal atout "séduction" du HTML5 est sans contexte son accessibilité: un jeu est potentiellement accessible de toute plateforme possédant un navigateur internet. Aussi, le joueur n'aura pas à se demander quand il pourra reprendre son jeu. Cela peut d'une importance cruciale (tout est bien évidemment relatif), puisque certains jeux sont addictifs: prenons par exemple un jeu assez simple comme Tetris, on peut imaginer une personne y jouant chez elle sur sa tablette tactile.

Seulement si cette même personne doit quitter son domicile, comment continuer à jouer à Tetris? Le mobile peut être une solution, mais le mobile n'est peut-être pas de la même marque que la tablette et n'a donc pas Tetris installé. Le HTML5 répond aisément à ce problème, puisque tout est accessible via un simple lien, et s'il possède un téléphone suffisamment récent et une connexion Internet, le joueur pourra continuer ses parties de Tetris en prenant par exemple le bus. Le côté multiplateforme du HTML5 peut être une arme très importante dans son développement, surtout depuis qu'Apple a refusé de supporter Flash sur ses appareils mobiles.

Ce côté multiplateforme permet également de profiter de celles-ci pour tester diverses expériences utilisateurs: prenons par exemple un jeu de conduite, il sera agréable de jouer sur sa console grâce à un volant pour profiter d'une ambiance optimale, mais nous pouvons aussi utiliser la gestion de la direction de sa tablette, et donc l'utiliser comme volant pour une tout autre jouabilité. Ainsi, le joueur peut choisir la plateforme qui lui convient le mieux sur toutes celles proposées.

Ce souci d'installation est le 2e atout: avec HTML5, aucune installation n'est requise pour pouvoir jouer. Un simple clic sur un lien est requis, cela rend donc son accessibilité bien plus simple. Par exemple, pour jouer à un

jeu natif sur iPhone, vous devez d'abord passer par l'App Store, puis télécharger le jeu pour ensuite pouvoir y jouer. Si vous ajoutez à cela le fait que le jeu peut potentiellement ne pas vous plaire (et de ce fait prendre de l'espace disque inutilement), cela peut rendre certains joueurs réticents au téléchargement. Ce problème n'en est plus un avec le HTML5, puisqu'avec un simple clic l'utilisateur a un accès direct au jeu et peut le quitter aussi vite qu'il est venu.

Viens ensuite le 3e atout du HTML5: les mises à jours ne sont pas nécessaires, en tout cas pas par l'utilisateur. En effet, le jeu étant accessible via un navigateur comme n'importe quel site web, ce sont les développeurs qui vont se charger des mises à jour, et non l'utilisateur. Cela est avant tout un signe de fiabilité, puisque sans s'en rendre forcément compte, le joueur joue avec la version la plus fiable et optimisée de l'application.

Enfin, le 4e atout découle du web 2.0 lui même, qui a vocation à relier les personnes entre elles. Aussi, le HTML5 peut se vanter de pouvoir bénéficier de la puissance de ce web social nativement, sans recours à une application tierce. On peut ainsi s'imaginer jouant à son jeu préféré contre ses amis Facebook tout en mettant en lumière les résultats de ces affrontements sur Twitter sans quitter le jeu. Cela offre donc une puissance virale immense, qui est très importante dans une société hyperconnectée et plus ou moins dépendante des réseaux sociaux.

Tous ces atouts réunis amènent une conséquence cruciale: nous avons vu dans les précédentes parties que le HTML5 pouvait aujourd'hui reproduire la plupart des jeux vidéos phares de ces dernières années: s'il est possible de jouer au même jeu depuis n'importe quelle plateforme sans avoir à passer par un magasin, cela peut représenter une économie importante. Par exemple, prenons un jeu comme Super Mario dont le prix est variable: si le joueur doit acheter une version pour chacune de ses plateformes,

cela peut très vite se compter en centaines d'euros. Au contraire, s'il achète une version HTML5 jouable à la fois depuis son ordinateur personnel, sa tablette, son téléphone et potentiellement sa console de jeu (si elle dispose d'un navigateur web): cela représente une économie importante. Encore mieux, nous avons évoqué la possibilité de mélanger ses plateformes: s'il est possible de jouer sur sa télévision en utilisant son téléphone comme manette, il n'est même plus nécessaire d'acheter de console de jeu.

En ces temps de crise économique, l'avantage prix réduit que peut représenter le HTML5 est incroyable. Mais il n'est pas pour autant décisif.

En effet, le secteur du jeu vidéo ne semble pas connaître cette crise puisque les ventes de consoles et de jeux sont constantes, et les jeux en HTML5 présentent néanmoins quelques défauts.

Tout d'abord au niveau des performances, sur une machine possédant la même configuration, le HTML5 est moins performant que le langage natif de la machine. Si cela n'est pas préjudiciable avec des jeux simplistes (en 2D par exemple), cela peut s'avérer plus problématique dans le cas de jeux bien plus compliqués. Par exemple, un portage d'un jeu aussi complet que Call Of Duty (qui présente à la fois des graphismes 3D en HD, du multijoueur, des configurations de personnages variées et des mouvements permanents) il faudra une machine plus performante pour lancer le jeu en HTML5 avec la même qualité que sur un PC, une PS3 ou une XBox.

Ainsi, si l'utilisateur veut jouer à des jeux de très haute qualité dans leur version HTML5, il devra penser à se munir d'une machine plus performante, et donc souvent plus chère à l'achat.

Si les joueurs sont pour la plupart habitués à devoir se munir de machines plus performantes pour jouer aux jeux derniers cris, cela n'est pas forcément le cas des utilisateurs de tablettes et de smartphones, dont la plupart présentent d'ailleurs des performances équivalentes (si bien sûr ils

sont de la même génération). De la même manière, certains utilisateurs conservent leur téléphone sur une durée d'environ 2 ans, aussi ces derniers vont sûrement préférer les jeux en développement natifs, car ils offrent de manière assez flagrante une différence de performance d'exécution du jeu en défaveur du HTML5.

Ensuite, si le jeu en HTML5 est automatiquement mis à jour, ce n'est pas le cas des navigateurs web: aussi l'utilisateur doit s'assurer d'avoir son navigateur tout le temps à jour afin de bénéficier du meilleur environnement possible pour son jeu. De plus, le support des standards du HTML5 n'est pas le même sur tous les navigateurs, il est donc fort possible d'avoir à utiliser un certain navigateur par rapport à un autre. Encore une fois, ce n'est qu'un léger détail sur un ordinateur personnel, mais dans le cas des mobiles, c'est plus problématique, car il n'est pas tout le temps possible d'utiliser autre chose que le navigateur de base: par exemple sur iOS, il n'est pas possible d'utiliser un autre navigateur que Safari Mobile, car même si Chrome existe sur iPhone, ce n'est en vérité que le navigateur Safari dont l'habillage a été modifié pour lui donner un look plus conforme à Google.

Ensuite, le jeu vidéo web étant la plupart du temps un jeu en ligne, il faut obligatoirement une connexion internet pour jouer. Ce n'est pas le cas pour une application native puisqu'après le téléchargement de l'application, il est possible d'y jouer partout. Nous avons vu précédemment qu'il est possible de stocker localement des données dans le navigateur grâce à l'API Web Storage, mais cette solution couture néanmoins la perte de l'aspect multiplateforme d'un jeu vidéo web.

Enfin, si le côté multiplateforme est sans aucun doute séduisant, il faut également savoir que certains jeux ne sont tout simplement pas adaptés à certaines plateformes. J'ai souvent pris l'exemple de Call Of Duty, qui est très agréable sur un ordinateur personnel et sur consoles, mais sur une

tablette ou un téléphone, l'expérience utilisateur ne serait sans doute pas optimale et il ne serait peut être pas utile de développer de telles versions.

Tous ces détails montrent que pour jouer efficacement à un jeu en HTML5, il faut donc à la fois une machine performante, une connexion internet et un navigateur à jour. Cependant, cela ne constitue pas des points suffisamment importants pour complètement mettre le HTML5 hors course, puisque la plupart des joueurs avisés sont dotés de machines ultras performantes pour exécuter les jeux vidéos derniers cris, les tablettes et téléphones sont de plus en plus puissants (au point d'égaler les ordinateurs personnels sur certains points) et les consoles ne sont pour la plupart pas exploitée à 100% de leurs capacités: alors que la PS4 doit sortir au mois de novembre alors que la pleine exploitation de la PS3, pourtant âgée de 7 ans, n'est pas encore atteinte. Nous voyons donc que les performances plus élevées demandées par un jeu en HTML5 ne sont pas un problème en soi. Qui plus est le dernier E3 (le plus grand salon mondial du jeu vidéo) a démontré que les constructeurs de consoles commencent à envisager une différente utilisation de ces dernières en se dirigeant vers la construction de véritables machines axées sur le divertissement.

Concernant les mises à jours de navigateurs, la plupart des géants du web permettent les mises à jours de leurs outils sur des systèmes d'exploitation assez vieux, de ce fait l'utilisateur n'aura qu'à prendre la peine de faire cette mise à jour, ce qui d'un certain côté est moins embêtant que de devoir mettre à jour plusieurs jeux.

Enfin, la connexion obligatoire est plus ou moins un faux problème, puisque la plupart des foyers sont aujourd'hui équipés d'une connexion, que des points d'accès WiFi sont disponibles dans de plus en plus de lieux publics et que la plupart des opérateurs fournissent des abonnements avec internet illimité.

Ainsi, nous pouvons voir que si les jeux en HTML5 ont bien évidemment des défauts, ils sont tous surpassables pour quelqu'un ayant envie d'utiliser cette technologie.

Cependant, comme je l'ai dit, peu de joueurs se lèvent le matin en se disant qu'ils veulent utiliser obligatoirement du HTML5, et c'est donc grâce à tous ses atouts, l'évolution des technologies, des plateformes et de l'environnement que le HTML5 séduira ou non les joueurs.

C'est pour cette raison que les prochains acteurs du jeu vidéo que dont je vais parler vont avoir un rôle important dans le développement et la popularisation du HTML5: je veux bien sûr parler de ceux qui développent et vendent ces jeux.

B - Ceux qui les fabriquent

Si des studios de développement de jeux sont mondialement connus, comme Ubisoft ou EA Games, il existe en vérité un nombre colossal d'entreprises développe des jeux vidéo. C'est pourquoi se pencher les différentes implications du HTML5 dans le cadre du développement et de la monétisation de ces jeux est très important. En effet, si les joueurs sont un acteur majeur du jeu vidéo, les entreprises le sont également. Elles ont même une influence plus importante sur la technologie qu'est le HTML5 puisque ce sont elles qui choisiront ou non de l'utiliser. 2012 a par ailleurs été une année assez difficile pour le HTML5, car bon nombre d'entreprises choisissaient de ne plus opter pour cette technologie (par exemple Zynga). Cependant, 2013 marque un renouveau et il faut donc regarder quels peuvent être les apports et les inconvénients de cette technologie dans un premier temps pour les développeurs qui vont construire les jeux, puis dans un deuxième temps aux aspects commerciaux que le standard peut permettre d'envisager. Le tout permettra d'avoir une version plus claire sur le business model que peut représenter l'utilisation du HTML5 pour la production de jeux.

1 - Du côté des développeurs

Cette partie a pour but de traiter les implications du développement en HTML5 du point de vue des développeurs, et plus largement des créateurs de jeux.

Ils représentent ceux qui vont créer l'ensemble du jeu, ils sont la plupart du temps réunis en équipes. Ils auront une influence assez grande sur le futur du développement de jeux en HTML5 puisqu'ils sont les principaux concernés par la technologie: ce sont eux qui vont devoir apprivoiser le HTML5 pour créer des jeux à la fois performants et conformes aux demandes du marché.

Le développement en HTML5 a son lot d'avantages et d'inconvénients: tout d'abord, ce développement est multiplateforme dans la mesure où il est possible de produire un jeu en HTML5 depuis n'importe quel éditeur de texte standard. Aussi, les développeurs sont donc libres de choisir le système d'exploitation et l'interface de développement qui leur convient le mieux. De plus, cela va entraîner une réduction des coûts liés aux éventuels achats de logiciels: dans le cas du développement d'un jeu classique, si les plateformes ciblées sont Android, Windows Phone et iOS, et bien cela entraînera l'utilisation d'au moins 2 systèmes d'exploitation différents (Windows et MacOSX) et de 3 interfaces de développements différentes (en Java pour Android, Visual Studio pour Windows Phone et XCode pour iOS et l'Objective-C). Pour le HTML5, il est juste nécessaire de posséder un éditeur de texte, un moyen d'héberger les applications en ligne et les 3 plateformes de test: soit potentiellement moins de dépenses, la dernière étant obligatoire même dans le cas du développement natif.

De plus, le code HTML5 étant unique pour toutes les plateformes ciblées, les coûts et temps de développement sont inférieurs aux développements de plusieurs applications natives.

Ensuite, le HTML5 ne passe par aucune vérification et est donc utilisable directement partout ce qui n'est pas tout le temps le cas de toutes les plateformes: par exemple l'acceptation de son application iPhone sur l'App Store d'Apple est longue et fastidieuse.

Nous avons surtout évoqué les conséquences du HTML5 sur les couts de développement qui sont réduits, mais le véritable point fort du HTML5 c'est qu'il permet de créer des équipes de développeurs homogènes puisqu'ils seront tous experts dans la technologie HTML5. Ainsi, la gestion des équipes de développement est bien plus aisée puisqu'il est facile de jongler avec les développeurs pour la création d'équipes en fonction des projets sans avoir à tenir compte des compétences de chacun.

Et c'est surtout là que se trouve le vrai point fort, puisque l'utilisation d'une seule technologie va permettre, en réunissant des experts de celle-ci, de l'utiliser à son plein potentiel.

Cependant, il ne faut pas faire l'erreur de croire que le HTML5 est une technologie magique qui va permettre de tout faire très facilement. Ce n'est tout simplement pas vrai, et c'est pour cette raison qu'après la folie de 2011, et toutes ces théories sur les futures possibilités du HTML5, il y a eu une forte désillusion en 2012. Non, le HTML5 ne permet pas de réaliser tous les types de jeux possibles: pour la bonne et simple raison que si un jeu n'est pas adapté à une plateforme dans un premier temps, ce n'est pas avec le HTML5 que vous l'adapterez. Prenons un exemple, même si dans l'absolu il est possible avec beaucoup de travail de développer une version mobile d'un jeu de PS3, l'expérience utilisateur a de grandes chances d'être terrible et non adaptée, car la PS3 est une plateforme plus compliquée qu'un téléphone. Ceci rendra donc inutile tout le travail fourni pour rendre son jeu PS3 jouable sur iPad.

Ensuite, même si le HTML5 rend plus facile le portage sur différentes plateformes, il ne rend pas plus facile le développement du jeu en lui

même: développer un jeu vidéo reste une tache compliquée que le langage utilisé soit JavaScript ou n'importe quelle technologie native. De plus, l'utilisation de JavaScript pour développer un jeu n'est pas encore rentrée dans les moeurs des plus irréductibles développeurs et même si les API pour améliorer le langage sont de plus en plus nombreuses, certains développeurs auront du mal à passer d'un langage comme Java à JavaScript.

Enfin, si les compatibilités entre plateformes ne sont pas à gérer dans la plupart des cas de développement, il faudra néanmoins passer par la compatibilité entre navigateurs qui est tout aussi compliquée. Il faudra donc encore une fois faire un choix sur les possibilités de son jeu et sur l'utilisation de composants nouveaux, mais non supportés par tous les navigateurs: par exemple, l'utilisation de WebGL n'est pas encore compatible avec le jeu sur iPhone.

Aussi, je pense en conclusion que les développeurs vont devoir commencer par envisager le web comme une plateforme particulière et un peu ingrate pour l'instant, mais à la manière des jeux vidéos qui ont fait évoluer les machines pour toujours plus de performances, on peut imaginer que la plus la communauté des développeurs de jeu en HTML5 grossira, plus les avancées sur le web seront nombreuses. Néanmoins, il faut se rendre compte que si le HTML5 offre quelques avantages aux développeurs de jeu, il change complètement la donne quant à la manière de les distribuer.

2 - Du côté commercial

Si le HTML5 permet aujourd'hui de développer des jeux vidéo attractifs, il n'en résulte pas moins une question pertinente: comment distribuer ces jeux dans un but commercial?

Car il est évident que pour populariser ce système de développement dans les entreprises, il faut qu'il y ait un business model efficace. Le bon côté du HTML5 est qu'il possible d'adapter plusieurs business model déjà existants pour monétiser son jeu.

Le premier business modèle est l'utilisation de la publicité, ce qui permet à l'utilisateur de jouer gratuitement tout en finançant votre entreprise grâce aux espaces publicitaires achetés sur votre page web. Comme je l'ai dit, ce système a l'avantage de ne rien faire payer à l'utilisateur, par contre, la présence de pub sur une page web peut être dérangeante si celles-ci sont trop voyantes ou trop intrusives, ce qui peut gâcher l'expérience utilisateur.

Il est également possible d'envisager le business model par licence: le joueur achète le droit de jouer via un compte utilisateur, puis par exemple grâce à une combinaison de mot de passe et d'email pour accéder au jeu. Une variante de la licence est l'abonnement: il est alors possible à l'utilisateur de s'abonner à un jeu pour une durée limitée (1 jour, 1 semaine, 1 mois, etc.) et choisit alors de renouveler ou non cet abonnement: ce business modèle a l'avantage d'être apprécié des utilisateurs qui peuvent arrêter le paiement à n'importe quel moment si le jeu ne leur convient pas.

Enfin, il est possible de proposer le même business modèle que pour les freemium: soit un accès gratuit au jeu, mais avec des fonctionnalités limitées, un paiement étant nécessaire pour profiter de la totale expérience de jeu. Ce système offre l'avantage d'offrir une version de démonstration au joueur qui va pouvoir décider de continuer ou non de jouer.

Ainsi, nous pouvons voir qu'il existe une multitude de moyens pour réaliser un profit grâce au jeu vidéo en HTML5. Seulement, il est possible d'augmenter ces profits en exploitant la multitude de moyens de diffusions de ces jeux, qui peuvent être distribués de plusieurs façons.

Le meilleur moyen de distribuer son application et d'utiliser le web lui même, et donner un nom de domaine à son jeu. Ce système donne au moins une accessibilité très importante puisqu'un nom de domaine est disponible depuis n'importe quelle machine possédant un navigateur. Il serait selon moi judicieux d'utiliser le Web pour des jeux de grandes envergures, par exemple World Of Warcraft, qui à mon avis ne collent pas tout à fait aux systèmes proposés par les magasins en ligne. De plus, tous les navigateurs, notamment sur mobile, n'offrent pas les mêmes possibilités que les navigateurs d'un ordinateur personnel.

Il existe cependant un moyen de contourner ces limites, car nous avons vu qu'il était possible de créer des applications hybrides, notamment en utilisant des frameworks comme Phonegap ou CocooJS, afin de pouvoir distribuer ses applications via les magasins en lignes classiques. Et même si le but caché d'un jeu en HTML5 est justement d'éviter ces magasins en ligne et leurs politiques de vente d'application (elles prennent un pourcentage sur les ventes), ou dans le cas d'Apple la politique de vérification des applications, il est pertinent d'utiliser également ses moyens de distributions, car ils sont plutôt bien optimisés pour la vente. Seulement, de plus en plus de magasins en ligne spécialisés dans le HTML5 commencent à voir le jour, pour n'en citer que deux en exemples, le Chrome Web Store et le Mozilla Market Place acceptent aujourd'hui les jeux en HTML5 pour leurs navigateurs respectifs.

Ensuite, nous pouvons imaginer, à l'instar de Steam, une plateforme d'achat de jeux vidéo en ligne, un espace de vente de jeux spécialisés en HTML5. Nous pouvons également imaginer des magasins en lignes développés spécialement par les entreprises produisant des jeux vidéo en HTML5.

Mais il existe d'autres moyens de distribution: notamment grâce à l'utilisation des réseaux sociaux. Ces derniers ont souffert à cause des magasins en ligne des constructeurs, et seraient potentiellement

enthousiastes à l'idée de se voir proposer un jeu en HTML5 pour être intégré dans ces derniers et ainsi ne voir aucune redirection vers une tierce partie qui va bénéficier des apports publicitaires, du trafic et de la vente de jeux: l'exemple actuel le plus frappant est la folie autour de certains jeux Facebook, comme FramVille ou Candy Crush qui sont très utilisés (on se demande encore comment Mark Zuckerberg a pu annoncer que son erreur majeure était d'avoir trop parié sur le HTML5) . Quels sont les avantages de ce système: et bien ils sont réciproques, d'une part l'entreprise peut bénéficier de la viralité des réseaux sociaux pour se faire connaître, et on peut même imaginer ses entreprises ne vendant plus que le droit d'utiliser ses jeux (une sorte de licence, mais payée par le réseau social). Outre cette solution, nous pouvions imaginer un partenariat avec le réseau social sur les retombées publicitaires ou aux investissements grâce à ces jeux. Du côté du réseau social, celui-ci va gagner en trafic et en temps de connexion: puisque l'utilisateur va rester connecté durant sa session de jeu. Ainsi, le réseau social va gagner du trafic, du temps de connexion et probablement de potentiels nouveaux utilisateurs. Ceci va entraîner une augmentation du prix de l'espace publicitaire et des investissements dans cette branche. Nous pouvons donc voir que ce style de est très efficace pour les deux parties le distribuant: à la fois pour l'entreprise et pour le réseau social, c'est donc probablement une bonne idée pour gagner de l'argent grâce au jeu vidéo en HTML5.

Enfin, passons à la dernière catégorie, celle de l'utilisation des opérateurs mobile. Ces derniers sponsorisaient des plateformes de jeu propriétaires qui rendaient accessible les jeux grâce a un certain type d'opérateur: par exemple un jeu pouvait être accessible grâce a Orange mais pas SFR. Or tout comme les réseaux sociaux, ces moyens de distribution ont énormément souffert de l'apparition des magasins en ligne distribuant des applications natives. Aussi, étant donné le nombre conséquent d'opérateurs mobiles et que les avantages sont les mêmes pour les deux

parties que dans le cas des réseaux sociaux, il est très probable de voir certaines entreprises se diriger vers ce moyen de distribution.

Par contre, le jeu dans un navigateur web n'est pas à proprement parler du jeu vidéo pour certains utilisateurs, qui préfèrent une expérience plus tournée vers l'application, car elle est plus immersive. Si les applications hybrides permettent de donner ce côté application native au jeu vidéo, il existe pour les ordinateurs personnels un projet développé par Firefox, nommé WebRT, qui va permettre de créer dans le dossier application de l'utilisateur une application fictive. Cette dernière est fictive, car ce n'est en réalité qu'une fenêtre de navigateur web dépouillée de tous les ornements lambdas: soit aucune barre de navigation: ainsi, l'utilisateur n'ayant accès qu'au jeu en lui même, il peut se croire dans une application native alors qu'il aurait le même résultat dans son navigateur favori.

En conclusion, nous pouvons voir que la fabrication et la distribution de jeux vidéos en HTML5 possèdent d'innombrables options permettant de rivaliser avec le développement d'applications natives. Du côté des développeurs nous pouvons voir que les couts de développement sont diminués, et du coté des distributeurs, il existe un nombre incalculable de plateformes et de business modèle utilisable pour augmenter la visibilité et ainsi montrer son jeu au plus grand nombre.

Seulement, nous venons de voir que si les joueurs, développeurs et distributeurs de jeux vidéo pouvaient tous trouver leur bonheur grâce à ce style de développement, qu'en pensent réellement les derniers grands acteurs de la vie des jeux vidéos, à savoir les géants des milieux du web et les géants du secteur du jeu vidéo.

C - Les points de vue des géants du web et du jeu vidéo

Ces derniers acteurs auront une importance au moins aussi grande que les fabricants et les joueurs eux-mêmes, puisque finalement, ce sont ces derniers qui auront le dernier mot sur les possibilités offertes à la fois par les navigateurs, mais aussi par les plateformes de jeu. Aussi, ils peuvent complètement ruiner une décision de développement si au dernier moment ils annulent la compatibilité d'une technologie. Nous allons donc voir dans un premier temps de quelles manières ces géants aident au développement de jeu en HTML5, puis dans un second temps de quels sont les points sur lesquels cette nouvelle plateforme peut être mal perçue par ces mêmes géants, puis enfin quelles sont les possibilités à envisager pour le futur du jeu en HTML5.

1 - Les pas vers la popularisation du jeu HTML5

Les géants du web donnent aujourd'hui des signes encourageants vers le support des technologies HTML5. En effet, ils écoutent généralement les standards validés par le W3C et l'incluent à leur navigateur de façon automatique, permettant ainsi une évolution du web plus homogène. On pourra même noter que dans certains cas, ces géants du web prennent des initiatives et incluent des technologies prometteuses sans attendre l'aval du W3C.

On pourra également noter que les géants du web, Mozilla en tête, mettent en place des groupes de travail afin d'exploiter certaines technologies du web et ainsi la développer de façon commune: par exemple le WHATWG (Web Hypertext Application Technology Working Group) qui est une collaboration entre plusieurs développeurs de navigateurs web dont font partie Mozilla, Apple et Opera.

Mais il existe également des groupes de recherche sur différentes technologies pouvant représenter les évolutions prochaines du Web. Mozilla y participe activement, et produit souvent des technologies qui s'avèrent très prometteuses pour le web (dont la plupart ont été citées

dans les précédentes parties). On pourra également noter les effets de concurrence saine entre ces géants qui font continuellement évoluer notre perception et notre navigation sur internet. Un très bon exemple est le moteur JavaScript V8 de Google qui a d'une certaine manière remis au goût du jour ce langage alors que la plupart des professionnels commençaient à envisager des solutions alternatives.

Ensuite, nous pouvons saluer le travail de ces groupes de recherches qui font continuellement évoluer le développement web, que cela soit par l'invention de nouveaux langages ou l'amélioration de ces derniers, qui contribue à favoriser le développement web et par extension celui des jeux vidéos sur le net.

Enfin, les géants du web possédant des magasins en ligne font un premier pas vers le développement web en y acceptant les applications hybrides. Les plus optimistes d'entre nous pourront penser que c'est une avancée encourageante qui pourra être exploitée plus profondément dans un futur proche, notamment car l'omniprésence de ces magasins en ligne aux yeux des utilisateurs peut également être utilisée comme levier de promotion de son jeu, même si ce dernier est accessible plus aisément en cherchant directement sur le web.

Du côté des géants jeu vidéo, nous commençons à noter un changement de cap. En effet, certains professionnels du jeu vidéo estiment que les consoles historiques sont aujourd'hui dans leur phase de maturité, voire même dans un début de déclin tant l'avènement des tablettes et smartphones pour le jeu vidéo a été important sur ces plateformes, et de ce fait rivalisent dans l'invention de nouvelles technologies, mais aussi dans l'inclusion de technologies déjà excitantes. Par exemple, Nintendo fait un premier pas en mettant à disposition un framework pour développer des applications en HTML5 et Unity3D.

De plus, le dernier E3 a montré une volonté d'ouverture des possibilités des consoles, en les transformant plus en plateformes multimédias qu'en console de jeu au sens propre: possibilité de visionner des vidéos, de discuter en visioconférence, d'accéder au web et bien sûr de jouer pour ne citer que quelques exemples. Nous pouvons penser dans un futur proche la possibilité de développer sur XBoxOne et PS4 des jeux directement conçus en HTML5, et même si ce ne sont pas les titres les plus importants de la console elle même, cela montrerait une véritable volonté d'acceptation de ce standard en tant que réelle alternative au développement.

Nous pouvons donc voir des signes encourageants quand à l'acceptation du standard HTML5 par tous les géants. Sauf que ces signes ne sont-ils pas finalement que l'arbre qui cache la forêt?

2 - Des signes trompeurs ?

Les géants dominant les secteurs du web et du jeu vidéo sont avant tout des entreprises qui se destinent à faire du profit. Et mis part la Mozilla Foundation qui n'a pas de modèle économique à proprement parler, les autres entreprises ne voient pas encore dans le HTML5 un business model suffisamment valable pour parier dessus à 100%. Cela peut représenter un vrai frein dans l'évolution du HTML5 car si les distributeurs veulent parier sur le côté multiplateforme, ils doivent faire attention aux compatibilités de ces applications sur chaque navigateurs.

Tout d'abord, le jeu vidéo en HTML5 est accessible directement dans un navigateur web, ce qui rendrait obsolètes les magasins en ligne de ces entreprises. Si cela paraît anodin, c'est quand même une grande source de profit qui volerait en fumée, car ces magasins prennent un pourcentage sur la vente de chaque application. De plus, les entreprises ne pourraient plus contrôler les applications entrant dans leurs magasins: dans le cas

d'Apple, ils contrôlent toutes les applications entrant sur leur App Store afin de vérifier si celles-ci correspondent à une certaine charte.

Au delà de cette perte d'argent et de contrôle, cet abandon de la nécessité du magasin en ligne peut également signifier la perte de joueur, puisque la plateforme devenant obsolète et sans exclusivités a proprement parlé: il n'y aurait donc que la présence d'un certain navigateur qui importerait sur la machine, plus le système d'exploitation. Au-delà du fait de perdre des joueurs, cela voudrait dire que les entreprises devraient investir dans le perfectionnement de leur navigateur web ou dans l'acceptation de concurrents sur leurs systèmes d'exploitation.

J'ai parlé des exclusivités, par exemple Halo sur XBox, si une version du jeu sortait en HTML5, qu'est-ce qui empêcherait un joueur d'Halo d'acheter une PS3 ou un iPad (soit deux concurrents de Microsoft) pour jouer à son jeu favori. De la même manière, les utilisateurs pourraient se tourner vers des téléphones moins chers au lieu d'opter pour les très chers iPhone ou Galaxy S3 (même si ces derniers sont plus puissants, donc plus fiables sur la durée).

Ensuite, j'ai parlé de l'accessibilité du développement en HTML5, qui est utilisable depuis n'importe quel système d'exploitation et éditeur de texte. Or certaines entreprises ont profité de cet engouement pour le développement mobile pour ne le rendre accessible que sur leurs propres systèmes d'exploitation: c'est par exemple le cas pour iOS ou Windows Phone: ces deux systèmes d'exploitation nécessitent, même dans le cas d'application hybride, des environnements de développement adaptés à ces plateformes. Nous pouvons donc imaginer, au delta de la perte des joueurs, la perte des développeurs qui pourraient choisir de plus acheter les machines obligatoires et se tourner ainsi vers la concurrence.

De plus, cette perte de développeurs entraîne les géants du web à se tourner vers un nouveau style de guerre des navigateurs: à savoir la guerre des technologies.

Nous pouvons y voir plusieurs participants: certains comme Mozilla ou Google tendent plutôt vers l'acceptation des nouvelles technologies et standards dans leurs navigateurs respectifs. C'est une façon plutôt proactive de voir les choses puisque ces compagnies préfèrent intégrer par avance les technologies web même si celles-ci sont susceptibles de mourir assez vite, juste au cas où elles deviennent des standards. De plus, nous pouvons noter chez Google un moyen de vouloir s'imposer comme le navigateur de choix des utilisateurs et des développeurs en intégrant des plug-ins spéciaux développés pour ce navigateur, mais également en inventant de nouvelles technologies facilitant à la fois le portage d'application (avec le Native Client par exemple) et même le développement lui-même (avec par exemple son langage de développement: le Dart). Cette approche est d'une certaine manière la meilleure, puisqu'elle génère un climat de concurrence qui tend à faire évoluer vers le haut le web.

D'un autre côté, nous avons les entreprises, comme Microsoft ou Apple, qui sont un petit peu plus frileuses, et qui ne vont dans la plupart des cas introduire un standard que si celui-ci est validé par le W3C. Cette approche est plus prudente dans la mesure où elles ne veulent faire évoluer le web qu'avec des technologies viables et performantes.

Dans les 2 cas, ces approches cohabitantes font aujourd'hui que le développement de jeu vidéo web est assez compliqué si les entreprises veulent réellement cibler toutes les plateformes et tous les navigateurs web. Dans le cas contraire, Firefox et Chrome se posent pour l'instant en tant que précurseurs dans l'acceptation des jeux vidéo en HTML5.

Néanmoins ces quelques points noirs ne permettent pas à eux seuls de ne pas pouvoir envisager le développement de jeux en HTML5 de façon performante.

3 - De multiples possibilités

De quelle manière les acteurs du jeu vidéo peuvent-ils tous trouver un schéma performant afin d'exploiter le jeu vidéo en HTML5? Et bien en vérité il y en a une multitude et certaines sont potentiellement développables a court voire moyen terme.

La première solution serait de développer pour chaque plateforme un navigateur (ou une amélioration du navigateur déjà présent) spécialisé dans le jeu vidéo: ce navigateur intégrerait tous les composants nécessaires à une exploitation optimale du jeu vidéo en HTML5. Nous pourrions ainsi imaginer les géants du web s'entendre avec les constructeurs des différentes plateformes et systèmes d'exploitation (dont certaines entreprises se trouvent dans les 2 camps) pour produire un logiciel payant pour le jeu. Ce navigateur pourrait alors être vendu aux personnes voulant jouer, ou bien si les acteurs du jeu ne veulent pas faire payer les utilisateurs, elles peuvent faire payer les entreprises par des contrats de licence du navigateur en question. Même si ce schéma fait perdre quelques avantages de couts de développement d'un jeu en HTML5, cette technologie possède d'autres atouts pouvant attirer l'utilisateur, le côté multiplateforme en tête.

D'ailleurs, ce schéma pourrait d'ailleurs aussi correspondre aux consoles de jeu vidéo, qui ne sont dans le fond que des ordinateurs spécialisés.

Ensuite, nous pouvons également penser qu'une ouverture complète de tous les magasins en ligne aux jeux vidéo HTML5 pourrait être avantageux, autant du coté des entreprises que des studios de développement: qui pourrait ainsi bénéficier de l'apport des très grands

magasins en ligne pour lancer leur application (même si la vente de cette dernière est soumise à une perception du montant de la vente), mais garderait aussi la liberté d'offrir une expérience de jeu dans un navigateur web, et donc de vendre leur solution sans ces magasins en ligne. Nous nous dirigerions alors dans un système comme celui de Steam par exemple, ce magasin de jeux en ligne regroupe une multitude de jeux, mais il est également possible pour le joueur d'obtenir son jeu par d'autres moyens que Steam.

Enfin, à l'instar des opérateurs ou des réseaux sociaux qui peuvent parier sur certains jeux en HTML5, nous pouvons également imaginer que certains jeux puissent inclure de la publicité pour le navigateur d'une certaine entreprise, voire même ne l'inclure que sur un seul magasin en ligne, offrant ainsi une certaine forme d'exclusivité. Par exemple, si Apple décide d'investir dans un jeu en HTML5 peu connu à la condition que celui-ci ne soit distribué que sur l'App Store et non sur les autres magasins en ligne, la marque à la pomme bénéficiera d'une certaine exclusivité puisque le jeu en question ne semblera au premier abord accessible que depuis iOS.

Nous pouvons donc voir que la bataille pour le jeu en HTML5 est loin d'être finie même si la technologie bénéficie en 2013 d'un retour de "hype" assez encourageant, au contraire elle semble juste commencer. Dans le meilleur des mondes, tous les constructeurs, développeurs et distributeurs s'entendraient pour donner une chance à cette technologie, mais il faut bien avouer que c'est loin d'être le cas, chacun ayant de nombreux intérêts économiques en jeu.

Cependant, HTML5 semble offrir suffisamment d'avantages pour le joueur et le développeur pour ne pas être négligée à l'avenir, et il faudra attendre un gros succès de jeu en HTML5 pour confirmer cela et ainsi pousser les géants du web à exploiter pleinement le potentiel du standard.

CONCLUSION

En conclusion, il faut bien se rendre à l'évidence, le développement de jeu en HTML5 n'en est qu'à ses débuts, mais s'annonce plus que prometteur, puisqu'aujourd'hui il est possible de réaliser un jeu en 3D, multijoueur et multiplateforme. J'en veux pour preuve le développement des jeux comme HexGL ou BananaBread qui sont aujourd'hui des exemples à suivre et sur lesquels s'appuyer pour construire le futur du jeu vidéo en ligne, et pourquoi pas le futur du jeu vidéo.

Cependant, comme je l'ai dit dans ce mémoire, le HTML5 n'est pas magique, est le développement de jeux restera au moins aussi complexe qu'il ne l'est déjà. Par ailleurs, il faudra que les développeurs considèrent le web comme une plateforme à part entière, avec ses qualités et ses défauts. De plus, je pense que l'exploitation du côté multiplateforme passe avant tout par une exploitation optimale de ces plateformes pour le même jeu. Je vais encore une fois citer un de mes exemples favoris, mais une réplique de Call of Duty version PS3 sera injouable sur un iPhone, par contre, l'utilisation de l'iPhone pour un autre rôle que sur PS3 tout en restant connecté à la même partie en ligne avec ses amis peut être une révolution pour le jeu vidéo.

Comme nous l'avons vu, le HTML5 offre aujourd'hui énormément d'API permettant de faire évoluer au mieux la programmation de jeux vidéo, et le futur de ce type de développement s'annonce plus que brillant. De plus, si nous rajoutons au simple HTML5 la possibilité d'utiliser d'autres technologies web comme Unity3D, de porter des applications existantes avec par exemple Emscripten, mais également la possibilité d'exploiter l'hybridation des langages de développement, nous pouvons constater que le développement de jeux vidéos dans les navigateurs web ne peut que bien se porter.

Nous pouvons donc aisément nous rendre compte que le 8e art et le développement web sont loin d'avoir mis fin à leur collaboration et que les possibilités actuelles de développement sont colossales.

Cependant, le Cloud Computing commençant aujourd'hui a arriver à maturité, nous voyons aujourd'hui débarquer un nouveau concept pouvant lui aussi révolutionner le jeu vidéo en ligne: le Cloud Gaming (en français le jeu vidéo à la demande).

Ce concept permettrait tout comme pour la vidéo à la demande de jouer à un jeu vidéo lancé sur un serveur distant: ainsi, il n'y aurait même pas besoin que l'utilisateur possède la puissance requise sur sa machine pour jouer: juste d'une bonne connexion internet. En clair, cela permettrait au joueur de jouer en streaming.

NVidia a dernièrement lancé sur le marché sa NVidia GeForce Grid, qui est une multitude de serveurs mis à la disposition des développeurs de jeu qui vont pouvoir mettre au point leur jeu vidéo et l'héberger. Ainsi, les utilisateurs vont pouvoir se connecter à ce serveur et profiter de leur jeu vidéo depuis n'importe quelle plateforme. Quand on voit aujourd'hui à quel point fonctionnent les services de musique et vidéo à la demande, le jeu vidéo à la demande est clairement une solution d'avenir, qui va certainement révolutionner le web et le jeu vidéo, que ce soit au niveau du marché que des expériences de jeu. Nous parlions précédemment de la menace que pouvait représenter le jeu vidéo web pour les magasins de jeux vidéos en ligne, et bien le jeu vidéo à la demande est une menace au moins aussi importante, et les constructeurs vont devoir se mettre au diapason assez rapidement pour ne pas se faire dépasser.

Cependant, l'exploitation du jeu vidéo à la demande peut être un virage très intéressant pour les joueurs, les développeurs, les distributeurs et les constructeurs, qui pourraient tous bénéficier de services de jeu à la demande pour tous se faciliter la vie. Et de la même manière qu'il existe

aujourd’hui des séries produites par des services de vidéo à la demande (comme par exemple House of Cards produit par Netflix) qui concurrencent celles des chaînes productrices séries, nous pourrions imaginer voir émerger de nouveaux grands studios de développement de jeux vidéo arriver sur le marché pour concurrencer un secteur déjà très bien fourni.

Dans tous les cas, toutes ces innovations ne peuvent qu’être de bon augure, à la fois pour le jeu vidéo et pour le web. Il semble d’ailleurs que cela soit un très bon choix pour les acteurs d’Internet de parier sur les jeux vidéo, car on peut aujourd’hui constater à quel point l’informatique a bénéficié de l’apport des jeux vidéo pour évoluer, il n’y a qu’à regarder à quoi ressemblaient les graphismes 10 ans auparavant pour voir une différence flagrante. Ainsi, l’Internet va inévitablement bénéficier des apports technologiques des jeux vidéos pour évoluer et devenir encore plus incontournable qu’il ne l’est déjà.

C’est pour cela que je terminerai en vous disant ceci: continuez de jouer en ligne, vous aidez ainsi à l’évolution du net!

Glossaire

Acceleration matérielle:

C'est un procédé qui consiste à confier une fonction spécifique normalement effectuée par le processeur à un matériel plus adapté. Par exemple, la carte graphique d'un ordinateur va gérer tous les processus d'affichages à l'écran

AJAX (acronyme d'Asynchronous JavaScript and XML):

AJAX est une architecture permettant de construire des applications web dites riches. Il va combiner les requêtes HTTP pour dialoguer avec un serveur Web, le XML ou le JSON pour le transfert de données de la réponse et le Javascript, CSS et le DOM pour transformer la page web en fonction de cette réponse.

API (Application Programming Interface):

C'est un anglicisme pour Interface de Programmation. Une API est un ensemble de classes, méthodes et fonctions qui vont servir à faire interagir des composants logiciels.

Boo:

C'est un langage de programmation objet dont le développement a commencé en 2003.

C#:

C'est un langage de programmation objet créé par Microsoft afin que la plateforme .NET puisse utiliser la totalité de ses capacités.

CERN:

Le CERN est l'Organisation Européenne pour la Recherche Nucléaire. Elle a été créée en 1952.

Cloud Computing:

Le Cloud Computing est un concept informatique qui représente une nouvelle manière de fournir des services. L'utilisateur n'installe plus d'application sur sa machine mais en profite directement en se connectant à un serveur.

Coffeescript:

Le CoffeeScript est un langage de programmation qui se compile en JavaScript. Il a été créé pour faciliter la lecture du JavaScript mais y ajoute aussi des fonctions. Brendan Eich, le créateur de JavaScript, a annoncé qu'il voyait CoffeScript comme une influence pour le futur du JavaScript.

CSS:

Cascading Style Sheets est un langage de programmation qui est utilisé pour mettre en forme des pages web.

DaaS:

Data as a Service est un service particulier de Cloud Computing qui permet l'accès à distance à un dépôt de données.

Document Object Model:

C'est un standard du W3C qui permet, indépendamment de tout langage de programmation, d'accéder, de modifier ou de mettre à jour le contenu d'une page HTML.

FPS:

Un First Person Shooter est un jeu de tir en vue subjective dans lequel on voit l'action telle que la voit le protagoniste.

Framework:

Un Framework est un ensemble de composants logiciel qui sert à créer les fondations d'un autre logiciel. Il peut être spécialisé dans un langage ou une plateforme particulière.

Gameplay:

C'est un anglicisme pour caractériser les éléments d'une expérience vidéoludique. Il est considéré par beaucoup de joueurs comme un facteur déterminant de la réussite d'un jeu vidéo. Certains s'amusent d'ailleurs sur le Web à répertorier et tester les jeux vidéo au plus mauvais Gameplay.

HTML:

L'Hypertext Markup Language est un format de données conçu pour représenter les pages web.

HTML5:

Il peut s'agir de 2 choses:

- le HTML5 est dans un premier temps la 5e version du format de données HTML
- pour certains développeurs, le HTML5 est un concept réunissant le HTML5 en tant que langage, le CSS3 et l'utilisation de Javascript avec ses nouvelles API. C'est principalement de ce dernier point que je parlerai le plus dans ce mémoire, principalement par envie de gain de temps pour ne pas avoir à réécrire tous les langages à chaque fois.

IaaS:

L'Infratsructure as a Service est un modèle de Cloud Computing grâce auquel il est possible de louer à un fournisseur une infrastructure logicielle et matérielle virtuelle et y accéder directement en ligne. Par exemple, il peut s'agir d'un hébergement web.

JavaScript:

Le JavaScript (créé en 1995) est un langage de programmation orienté objet permettant de créer des scripts agissant du côté d'un serveur ou du côté d'une page web.

JQuery:

JQuery est une bibliothèque JavaScritp libre permettant de simplifier les interactions avec une page HTML.

JSON:

JavaScript Object Notation est un format de donnée dont la notation est un dérivé des langages ECMAScript (dont JavaScript est une spécialisation).

Langage bas niveau:

Un langage est dit de "bas niveau" lorsque sa programmation se rapproche de celle de la machine.

Langage compilé/Compilation:

La compilation est un procédé utilisant un logiciel appelé compilateur pour transformer un code source dans un langage informatique cible (le langage compilé).

Librairie/Bibliotheque:

Une bibliothèque, ou librairie, est une collection de fonctions prêtes à être utilisées par des programmes.

Logiciel libre:

Un logiciel libre est un logiciel dont l'utilisation, l'étude, la modification et la duplication en vue de sa diffusion sont permises légalement et techniquement.

Malware:

Un malware est un programme conçu dans le but de nuire à un système informatique.

Market Place:

Un market place est un magasin en ligne de programmes informatiques.

Moteur de rendu:

Un moteur de rendu est un logiciel permettant de créer des images à partir de données.

MMORPG (massively multiplayer online role-playing game):

Cela signifie: un jeu de rôle en ligne massivement multijoueur. C'est un type de jeu mélangeant le jeu de rôle et le jeu en ligne.

MONO:

Le MONO est une mise en œuvre open source de la plateforme .NET de Microsoft.

Objective-C:

C'est un langage de programmation orienté objet qui est principalement utilisé dans les systèmes Apple.

Onclik:

Onclik est un événement du DOM qui va déclencher une action au lors d'un clic sur un objet.

Open Source:

La désignation open source s'applique aux logiciels dont la licence respecte les directives de l'Open Source Initiative. C'est à dire que le code est accessible pour créer des travaux dérivés et que la redistribution est libre.

PaaS:

Platform as a Service est un type de Cloud Computing permettant d'accéder en ligne à une application spécifique. On peut prendre en exemple les applications de Google comme GMail ou Google Drive.

Plug-in:

Un plug-in est un logiciel qui vient se greffer à un autre logiciel pour en augmenter les capacités. Un plug-in ne peut pas vivre sans le logiciel cible.

Point and click:

Le Point and Click est un style de jeu qui se joue principalement grâce à la souris.

Proxy:

Un proxy est un logiciel qui va se placer entre 2 applications pour en surveiller les échanges.

PHP:

Hypertext Processor est un langage de programmation principalement utilisé pour la génération dynamique de pages web.

Rôle Aria:

Les rôles aria sont des attributs ajoutés aux balises HTML pour en préciser la fonction.

RPG:

Un Role Playing Game est un jeu vidéo représentant un jeu de rôles.

SaaS:

Le Software as a Service est un type de Cloud Computing qui va permettre d'accéder à un logiciel présent sur un serveur en vue de son utilisation.

Standard du web:

C'est une expression désignant les technologies recommandées par le W3C.

Thread:

Un thread représente l'exécution d'un ensemble d'instructions du langage d'un processeur.

W3C:

Le World Wide Web Consortium est un organisme de normalisation des technologies web. Cette organisation est à but non lucratif.

Web 2.0:

Le Web 2.0 représente le Web Social où les utilisateurs participent eux aussi à la génération du contenu d'un site web.

XML:

Le XML est un format de données représenté tout comme le HTML sur le principe du balisage.

Sources

=> www.siteduzero.com

- Visité notamment dans le but de comprendre certains concepts

=> jeux.developpez.com/tutoriels/

- Visité pour apprendre les technologies à la base du jeu vidéo.

=> developer.mozilla.org

- Site web utilisé pour la recherche des technologies HTML5 les plus importantes pour le monde du jeu vidéo.

=> wiki.mozilla.org/Platform/AreWeFunYet

- ce site web recense l'ensemble des technologies qui lient le web aux jeux vidéo et en explique les bases.

=> www.html5rocks.com

- site web créé dans le cadre d'un Google project. Il évoque la plupart des technologies HTML5 de demain.

=> www.html5gamedevelopment.com

- il s'agit tout simplement d'un énorme source d'information et d'actualités à propos du développement de jeux en HTML5.

=> rawkes.com

- site web personnel d'un développeur de jeux en HTML5 très influent et qui a participé à de nombreuses conférences.

=> caniuse.com

- Utilisé pour connaître les compatibilités des technologies présentées.

=> Et pour finir, j'ai également utilisé les moyens offerts par le Web 2.0 en utilisant wikipédia pour certaines définitions, mais également Twitter en suivant de nombreux comptes de développeurs HTML5.

English Version

Video games were always part of the computing business. In fact, the first popular video game was developed in 1972 and named Pong. For many gamers, this game is always a reference. But the 8th art became famous more than ten years later, in 1985 with the Nintendo NES its first gaming icon: Mario.

Today, video games are a complete industry, creating jobs and generating more revenue than the movie industry since 2003. There are famous game icons known everywhere in the world like Mario, Pikachu or Lara Croft. Also, the video games have the record of the best opening of all entertainment medias (Music, Movies and Games) with the famous Call Of Duty MW3. Video games are everywhere, and it's not strange to find today gaming championships (like for example the PES League), gaming conventions (like the E3) or even professional gamers (like Ken Bogard). But the most interesting part with video games is that they were a considerable factor for the computing evolution, for hardware and softwares.

So as Web Gaming is today growing every day, with small games that you're playing in your browser or with big multi players games like Call Of Duty, we can begin to think that Web will also benefit from games to evolve.

The internet is probably the best invention of the 20th century, and now it's every where: on computers, phones, tablets, gaming consoles, TVs and it's possible to imagine that one day, every electronic devices will be connected to the Internet.

But how the Internet can bring new stuff in the game business? Is it really possible to make new games using web technologies? Is the web the future of gaming?

To answer that question, I'll first explain how a video game is built today, then I'll check all the ways web technologies can bring to make powerful

games, and to finish what are the consequences of web gaming for all the main actors of this business.

A video game is long and difficult to make because you have to connect different technologies together to make one. A video game is first of all made with a graphic engine, who will be used to display graphics on the user's screen. But it's also a physic engine, who will deal with all the interactions between the displayed objects. A game also has a sound engine, because you might have the best graphics in the world, without any sound, a game is like a mute movie. You'll also find in a game an AI engine who will make possible to interact with other parts of the game. In case AI is not enough, you might have to use a network engine to enjoy the possibilities of online player versus player. And then, a game is also made by its logic and all the interactions between the inputs (joystick, keyboard, mouse,...) that will make the best gameplay possible.

All these engines combined are making the game engine, who will make possible to create the best game ever.

So what is the problem with games development today? Well, you have so many different platforms, technologies and computing languages that you can only see game development as a big mess. In fact, you might use C++ to develop games on a PS3 or a PC, but you will also have to use Java to make an Android game, Objective-C if you also want it on iPhones and iPads, and I'm only talking about the seen part of the iceberg, because you also have several ways to use technologies who will work on some platforms, but not on others. So, as game development is already complicated because you need time, efficient teams and sometimes money to test what you're making, if you add the fact that you have so many choices in this messy world of game development, you'll never know where to begin.

So, I wondered myself if web technologies are not the answer to all of this, especially HTML5.

If we go back a two years ago, in 2011, everyone was very hyped about this standard and all the web developers said to themselves: "this is our time!". Actually, all the hype was destroyed in 2012 when weaknesses of HTML5 were exposed, and we even began to see some HTML5 bashing. But, as every technology, HTML5 followed its hype cycle and 2013 is the beginning of HTML5 maturity, we know can see what we might be able to do in the future.

So what about using HTML5 for gaming? Actually, you can pretty much make all sort of existing games in HTML5, some experts are thinking that 70% of all the actual games can be done in HTML5 with almost the same performances than the native technology.

But how can we do this? We can produce a cool graphic and physic engine using the canvas HTML5 tag. This will allow developers to draw 2D graphics inside with JavaScript. And if you want 3D, you can use the WebGL API that will help you design powerful and reliable 3D environments.

The only real problem of canvas is that you'll need a powerful platform and a recent browser to be sure to run the application (which is not really a problem knowing the hardware constructors are using nowadays).

Then if you want to make a cool multiplayer game to play with all your virtual friends around the world, you can use either AJAX or Websockets to connect your game to a server that will make possible to play with everyone you want.

Then, if you don't want to buy the 499\$ PS4 but still be able to play FIFA with a joystick: then developers can still use the GamePad API to be able to use XBox or PS3 joysticks with Javascript.

For the rest of the game, every part of it is done in Javascript, who can even be multithreads today using Web Workers.

But, even if there doesn't seem to be any problems with JavaScript, some developers don't like it, and they prefer C or C++ which are historic game development language. Well, that's not a problem anymore, because today, you can use compilers, like for example Emscripten, who will allow developers to convert existing application in their JavaScript equivalent.

So, the HTML5 hype seems to be back, but what is the point of using this technology for gamers, developers and games sellers?

For gamers, there are many good points: first, web games in HTML5 are accessible from any platform supporting HTML5. That means if you have to leave your house and your personal computer, you can still continue your game using your smartphone.

For developers, you're now aiming for one competence to make games for every platform, so you might be a part of an expert HTML5 team and make very good games for all platforms quickly.

On the sellers point of view, you can bypass the Market Places which are mandatory with native development and enjoy the power of the Internet infinity: you can now sell your application to whoever wants it, like careers, social networks or even keep it on your own market place. The other advantage of HTML5 games can come from social networking, because you're now playing in a browser, so you can still tweet your results while playing. For sellers, this is fantastic, because the game can be known in less time with that kind of sharing.

But there are also problems with HTML5: first, every browsers don't support every new technologies, that means you might have to be careful

when you make your game or to use hybrid games to be every platform support the game.

Then, even if in theory everything can be multi-platform, do you really imagine yourself playing an exact copy of Call of Duty on your iPhone? It would be impossible to play! But you can use the platform detection to adapt the game in function of where you're playing: if you're playing COD from your computer, no problem: we can go 3D, but from a phone we can imagine a total different user experience, like a commander or a spy role to disturb your team's enemies. This kind of gaming would certainly be very interesting!

So in conclusion, even if everything is showing that HTML5 may have one day the power to be a complete powerful part of gaming development. But people have to know that this technology is (still?) not magic, and making a game is still very difficult even in HTML5. But web gaming has today a new challenger on the field, with their expertise in the cloud business, NVidia arrived with his GeForce Grid will might allow GaaS (Gaming as a Service), a new type of cloud computing where you only need a very good connection to stream the game. This technology would make possible to play very powerful game even from a phone because this is the server who will make all the work, you will only have to rely on your internet connection and play.

Anyway, whether it's HTML5 games or cloud gaming, these technologies will only evolve if they are used, so I'll finish this by advising you something if you believe in web games: go play!