DEVENIR PROGRAMMEUR DE JEUX VIDEO

GUIDE DE SURVIE

Auteur : David MEKERSA (david@gamecodeur.fr)
Version 1.1 rev 2 - Mai 2017

Retrouvez moi sur :

GAMECODEUR
École de jeux vidéo en ligne

www.gamecodeur.fr

Table des matières

Une formation gratuite
en complément de ce guide

A vous lecteurs. Une fois que vous aurez lu ce guide (ou dès les 1ères pages...), vous aurez peut être envie de vous former à la programmation.

Parce c'est bien de donner envie... mais comment passer du côté obscur et se mettre à coder ?

Je vous ai donc préparé une formation gratuite de 3h pour vous apprendre toutes les bases de la programmation. Etape par étape... et accessible aux débutants.

Pour suivre cette formation, et accéder à d'autres contenus exclusifs, rendez-vous sur :

http://www.gamecodeur.fr

Il vous suffit de vous inscrire (en gratuit) et la formation vous sera proposée et vous pourrez commencer immédiatement à apprendre.

A tout à l'heure !

Un guide pour réaliser votre rêve : créer des jeux vidéo

Imaginez créer vos propres jeux vidéo… Le plaisir ressenti est intense. Vous donnez vie à un univers infini, dont les limites ne sont que celles de votre imagination. Et si vous avez téléchargé ce guide, c'est que votre imagination est fertile et que vous ressentez le besoin de passer un cap, de changer littéralement votre vie en passant de l'autre côté du miroir. Créer au lieu de consommer...

Que vous soyez juste un(e) passionné(e) ou que vous souhaitiez faire carrière, sachez qu'il n'a jamais été aussi facile de créer des jeux vidéo. Les outils et le matériel nécessaire à la création d'un jeu, autrefois réservés aux professionnels et aux experts, sont désormais à la portée de tous. La connaissance l'est aussi, mais de manière plus confuse. On croule bien souvent sous la masse d'information, la plupart du temps en anglais, et on ne sait pas par où commencer.

Pour nombre d'entre vous, cette discipline vous semble encore complexe, réservée à des élites.

Ce n'est pas le cas.

J'ai moi même appris seul et j'ai été amené à transmettre mon savoir à des jeunes, ou à des moins jeunes. Ils ont trouvé ça bien plus simple qu'ils ne le pensaient et nombre d'entre eux créent aujourd'hui des jeux vidéo.

Pourquoi ?

Car je n'ai pas été déformé par le processus d'éducation traditionnel. Je ne suis pas passé par une école spécialisée. Une de mes qualité c'est d'être très très pédagogue. Je suis ainsi capable, lorsque j'explique quelque chose, de me mettre à la place de mon "élève" et d'adapter mon discours. Mais **j'utilise également une méthode qui m'est propre, qui permet d'apprendre à programmer facilement, sans douleur, et accessible à tous.** Je souhaite vous la transmettre.

Mon objectif est de rendre accessible à tous la programmation de jeux vidéo !

Ce guide est un condensé de mes acquis, de mes opinions, de ma vision du métier de programmeur de jeu vidéo. Une vision qui m'est propre, qui est différente. Ce guide va peut être vous aider à passer de l'autre côté du miroir…

MAIS ATTENTION : Si vous faites partie de la majorité, vous allez lire ce guide (ou juste le début) et ne rien faire. NE FAITES PAS PARTIE DE CETTE MAJORITÉ ! Je vous offre une opportunité unique de réaliser votre rêve. Vous avez un objectif noble : CRÉER ! Alors allez au bout et passez à l'acte. Tout ça compte pour vous (ne pensez pas aux autres) alors n'abandonnez pas !

Astuce : Pour garder la motivation, suivez mes vidéos sur Youtube ou venez me parler sur www.gamecodeur.fr.

Bonne lecture.

Remerciements

- ***Jennifer d'Alboy**, mon épouse, pour son coaching quotidien*
- ***Lucas Baclé, Simon Ginestet, et Antoine Calas**, mes 3 premiers "apprenants" (des adolescents de 14 ans) qui m'ont aidé à affiner ma méthode et ont démontré son efficacité*
- ***Dino Dini** (auteur de Kick off) pour ses encouragements*
- ***Benoit Hozjan** (Kheops Studio) pour ses nombreux conseils*
- ***Laurent Michaud** (IDATE) pour m'avoir donné confiance en moi*
- ***Anne Spirau** pour sa complète relecture attentive et active !*
- *Et ceux et celles que j'oublie, qui m'aident à "connecter les points"*

Qui suis-je ?

Je m'appelle **David MEKERSA**. Je suis professionnel du jeu vidéo depuis 2009 et je vis de ma passion. Mon épouse, Jennifer, m'aide au quotidien dans mon organisation, mes prises de décisions, et la gestion de mon studio.

Je n'ai aucun diplôme de programmeur et j'ai tout appris seul.

Je crée des jeux vidéo pour le plaisir depuis l'âge de 13 ans. J'ai toujours rêvé de vivre de cette passion. Depuis le jour où je suis parvenu à afficher un caractère sur l'écran de mon premier ordinateur (un Amstrad CPC) j'ai compris que ma vie allait changer… Un univers infini de création qui s'ouvrait à moi.

J'ai appris seul à programmer. A l'époque pas d'Internet et quasiment aucune documentation. Quelques livres, des amis… J'obtiens mon premier job de programmeur en 1993. J'ai depuis occupé de nombreux postes à responsabilité. Mon CV est impressionnant et ma voie était toute tracée : m'emmerder dans un bureau, sur des outils prise de tête, à gérer des données de gestion…

Je maîtrisais, à un moment de ma carrière, plus d'une dizaines de langages de programmations ! C'est là que j'ai élaboré, sans le savoir, ma propre méthode pour apprendre un langage de programmation.

Cette voie toute tracée, je n'en voulais pas. Je n'avais jamais renoncé à mon objectif : **vivre de ma passion, créer des jeux vidéo**… et en 2009 j'ai sauté le pas, quitté mon emploi très bien payé et créé mon premier jeu commercial tout seul !

J'ai ditigé de 2008 à 2016 mon propre studio de création de jeux vidéo : Casual Box. Avec mon équipe, nous avons créé plusieurs jeux commerciaux, et de nombreux jeux et applications pour le compte de clients externes. Plus de 25 productions au total (voir mon CV).

Mon premier jeu, **Geisha: The Secret Garden**, créé entièrement en Basic **BlitzMax** en moins de 6 mois, a été téléchargé à plus de 800 000 exemplaires et a rapporté près de 200 000 $.

J'ai également créé le jeu d'aventure **Age of Enigma**, traduit en 13 langues et vendu plusieurs dizaines de milliers de copies dans le monde entier. Age of Enigma a représenté 1 an et demi de travail avec une belle équipe de passionnés. J'ai codé le moteur de Age of Enigma entièrement en **C++**. C'est le projet le plus complexe que j'ai été amené à réaliser.

J'ai ensuite créé et commercialisé un jeu mobile en 2 mois seulement, en langage **Lua**, qui s'appelle **Chicken Deep**.

J'ai aussi écrit le livre "Create 2D mobile games with Corona SDK" pour les éditions Focal Press.

Aujourd'hui, **j'ai décidé de combattre les idées reçues** et de rendre la programmation de jeux vidéo accessible à tous en partageant ma vision, mes méthodes et en créant un guide gratuit où je livre toute mon expérience. J'espère qu'il vous sera utile et qu'il vous donnera l'impulsion nécessaire pour REALISER VOTRE RÊVE !

Je propose aussi une école en ligne, composée d'ateliers de formation en vidéo, via mon site gamecodeur.fr. Créée en avril 2016, elle est la meilleure formation disponible sur Internet pour apprendre à programmer des jeux vidéo.

Elle propose des ateliers vidéo, en Français, pour apprendre à installer tous les outils que je conseille et suivre pas à pas ma méthode des 5 fondamentaux.

Chacun d'eux s'accompagne de supports écrits, d'exercices pratiques et d'un *coaching* individuel. Mais également une communauté, des méthodes, du développement personnel...

Tout au long de ces années, j'ai élaboré une méthode qui permet d'apprendre à programmer facilement, et qui va à l'encontre des approches traditionnelles.

C’est une méthode, mais aussi une philosophie ! Je crois en effet beaucoup à l’importance du mental dans l’apprentissage d’une nouvelle discipline, aussi complexe qu’elle puisse paraître.

Après tout, la complexité est-elle réelle ou bien seulement dans la tête de ceux qui enseignent ?

Tout est simple, il n’y a que les gens qui soient compliqués.

J’ai retenu cette citation, qu’un de mes patrons me répétait très souvent. C’est une vérité simple, prenez en conscience et votre vision de tout ce qui peut s’apprendre va changer.

Intéressé(e) par ma formation ? Voir à la fin de ce guide si vous souhaitez aller plus loin. En attendant, entrons dans le vif du sujet !

A qui s’adresse ce guide ? Et les conseils qui vont avec...

A ceux qui veulent devenir programmeurs de jeux vidéo

Néanmoins, beaucoup de sujets abordés concernent tous les métiers du Jeu Vidéo.

Il s’agit d’une série de conseils, de méthodes, pour aborder ce métier (ou loisir) passionnant. Que vous ayez l’intention d’apprendre seul(e) ou de faire une école, ce guide vous aidera à être mieux préparé à la phase d’apprentissage qui vous attend.

Aux *Game Designer*

Vous n’en avez pas marre de ces codeurs qui ne font jamais exactement ce que vous leur demandez ? Vous n’êtes pas frustré de ne pas pouvoir tester vos concepts, vos idées ?

Apprendre les bases de la programmation pourraient vous transformer, à terme, en ce qui se fait de mieux : un programmeur / *Game Designer* !

Et si vous n'y arrivez vraiment pas avec la programmation, abordez alors des outils tels que Multimedia Fusion 2 ou GameMaker, avec une préférence pour ce dernier qui vous ouvrira les portes de la programmation quand le coeur vous en dira...

Aux infographistes

Mêmes conseils que pour les *Game Designer*... Débarrassez vous de ces codeurs récalcitrants et transformez-vous en homme/femme orchestre !

Donnez vie à vos oeuvres, vous pouvez le faire !

Aux adolescents

Jouer aux jeux vidéo est un loisir sans équivalent. L'immersion, le plaisir est tel que rien ne peut rivaliser !

Rien sauf... créer ses propres jeux !

Au lieu de ne faire que consommer (parfois sans modération au grand désespoir de vos parents), passez du côté obscur : créez !

Lorsque vous aurez affiché votre 1er *sprite** à l'écran, vous réaliserez que tout devient possible ! Et vous pourrez peut-être en faire votre métier !

* Voir le dictionnaire dans ce guide

Aux parents

Chers parents. Regardez ce que je conseille, juste au-dessus, aux adolescents.

Vous devez encourager vos enfants à :

- **Diversifier leurs expériences vidéo-ludiques**

Je veux dire par là : au lieu de les empêcher de jouer, achetez-leur des nouveaux jeux ! Ils doivent découvrir que le jeu vidéo est un univers vaste, varié, et créatif. Arrêtez de les laisser jouer uniquement à un seul jeu (parfois pas de leur âge !) et à les laisser faire sans comprendre leur univers. Achetez des jeux avec eux, jouez avec eux, c'est le meilleur conseil que je puisse vous donner, il va changer votre vie de parent d'enfant addict aux jeux...

Au passage : un jeu vidéo ça ne se pirate pas, ça se paye. Apprenez-leur la valeur des choses, et à respecter l'univers de la culture et de la création numérique. Sur STEAM, on trouve des jeux à partir de 4 euros... combien coûte un paquet de cigarettes ou un magazine de potins ?

- **Créer au lieu de consommer**

Vos enfants doivent apprendre que dans la vie, il y a ceux qui consomment et ceux qui produisent. Le fossé entre les deux n'est pas énorme. Si on ne fait que consommer dans la vie... on est une sorte de mouton... Pour ne pas être toute sa vie un mouton, et se demander un matin à quoi on sert, on peut alors choisir d'être du côté de ceux qui produisent. Produire un peu ou beaucoup, car il est inutile de devenir une star pour se sentir comblé(e) ! Une fois que votre enfant aura appris les rudiments de la création d'un jeu vidéo, il ne jouera plus jamais comme avant aux jeux vidéo. Il analysera le jeu, se demandera comment il a été fabriqué. Et ça lui donnera envie de faire pareil, de progresser... La graine aura été semée.

Appel à idées

Si vous n'avez pas trouvé dans ce guide un sujet que vous souhaitez me voir aborder, contactez-moi par mail : david@gamecodeur.fr, je me ferais un plaisir d'améliorer ce guide si je trouve le sujet pertinent.

Avertissements

1. Ce guide est gratuit.

2. Ce qu'il contient est basé sur mon point de vue, mes méthodes, mon expérience.

3. Ce guide n'apprend pas à programmer. Il vous guide dans vos choix, vous donne des pistes, des méthodes et vous motive. Un tel guide ne peut pas traiter de la technique pure. Restez en contact sur mon site www.gamecodeur.fr pour suivre mes futurs ateliers pratiques de programmation en vidéo.

4. Ma méthode, basée sur 5 fondamentaux et un certain état d'esprit, fonctionne. Elle n'a pas vocation à fonctionner pour tout le monde, ou du moins, pas à la même vitesse pour tous. De plus, elle ne s'applique pas à

tous les objectifs, mais elle est une base solide pour affronter toutes les situations...

5. Si vous avez des opinions différentes, vous n'êtes pas obligé d'adhérer à ma vision, ni à suivre mes conseils. Mais je vous assure qu'ils sont bons :)

6. Il n'y a pas d'approche universelle de ce marché et de ses métiers. Ma vision n'engage que moi.

7. Pourquoi est-ce que je fais ce guide gratuitement mais en demandent votre email ? Vais-je ensuite vous envahir de spam et pirater votre ordinateur par la force de ma pensée ? Non je ne vais rien faire de tout cela. Je fais ce guide tout deux raisons simple : dans une démarche de développement personnel (je partage mon savoir) et dans le but de créer une école de jeux vidéo en ligne (www.gamecodeur). Distribuer ce guide me permet de créer des vocations, de me faire connaître, et de créer une communauté autour de mon projet. Libre à vous de vous désinscrire, mais faites moi confiance, nous allons partager de grande choses ensemble si vous continuez à me suivre.

Pourquoi devenir programmeur de jeux vidéo ?

Pour changer de métier

Vous souhaitez vous reconvertir, acquérir de nouvelles compétences pour postuler dans le domaine du jeu vidéo, du multimédia, de l'interactivité ? Vous êtes au bon endroit.

Pour vivre de sa passion

On peut devenir développeur de jeu vidéo indépendant et en vivre. Nombreux y sont parvenus, pourquoi pas vous ?

Pour créer au lieu de consommer

Vous pouvez passer de l'autre côté du miroir et créer des jeux vidéo. Croyez-moi, vous ne jouerez plus jamais de la même façon.

Pour pratiquer un loisir différent

Pourquoi obligatoirement en faire un métier ? Créer des jeux vidéo est le loisir numérique le plus enrichissant qui soit.

Quels sont les (vrais) métiers du jeu vidéo ?

Les métiers fondamentaux

- Programmeur

La base de la base. Les grands manitous… Sans eux, pas de jeu vidéo. Ils donnent vie au jeu vidéo en “programmant” son comportement et tout ce qui le compose. C’est la valeur sûre sur le marché du travail. Très recherchés, et indispensables, leurs salaires sont aussi parmi les plus intéressants.

Autre avantage de ce métier : sa polyvalence. Si un jour il ne trouve pas de travail dans le domaine du jeu vidéo, un programmeur pourra s’adapter et travailler dans d’autres domaines : applications mobiles, informatique traditionnelle, web, etc.

- Infographiste

Un fondamental, et aussi une discipline très vaste. Un infographiste peut être 2D, 3D, et dans le cas de la 3D il peut encore être spécialisé. Je ne peux que conseiller la polyvalence dans le cas de l’infographie 3D.

Dans tous les cas, un infographiste ne devra pas oublier qu’il est avant tout un artiste. Il doit maîtriser la base : le dessin et le travail des couleurs. Je reçois si souvent des Portfolios contenant des travaux approximatifs, aux couleurs hideuses et aux proportions ratées…

Dans ce métier, seuls les meilleurs trouveront du travail facilement. Mon conseil : soyez au top, visez l’excellence !

- *Sound Designer* / Compositeur

Le Sound Designer crée les bruitages d’un jeu et son environnement sonore. Il peut aussi être compositeur et créer la ou les musiques du jeu. C’est le métier le plus saturé qui soit. Je reçois une candidature par jour quasiment, toutes d’artistes de talent. Seuls les meilleurs, et ceux qui sauront se créer un nom,

pourront percer dans l'industrie du jeu. D'autant plus que c'est un besoin ponctuel, on embauche rarement (jamais ?) un Sound Designer ou un compositeur. Tous travaillent à leur compte.

Les autres métiers

D'autres métiers, plus spécifiques, peuvent se rencontrer dans des studios de plus grande taille :

- *Game Designer* (*GD*)

Il imagine et formalise les concepts de jouabilité, l'univers et le contenu d'un jeu (et les émotions véhiculées par le jeu par ailleurs même si cet aspect est souvent oublié des formations...). Mais ne savoir faire que ça, c'est pour moi comme inventer des recettes de cuisine sans savoir cuisiner... Je conseille aux GD de savoir prototyper, et mieux encore : de savoir programmer. De nombreuses écoles forment aujourd'hui au *Game Design*... Pensez-vous vraiment que l'industrie pourra absorber tous ces jeunes si peu polyvalents ? D'autant plus que ce métier, exercé seul, est réservé aux gros studios, assez rares sur la planète...

- *Level Designer* (*LD*)

Même combat que le *Game Designer*, et encore plus spécialisé !
Il organise le contenu du jeu : niveaux, difficulté... Mieux vaut savoir faire autre chose et ne pas se contenter de cette compétence pour espérer trouver du travail.

- Testeur

Il teste les jeux et rapporte à l'équipe de développement les anomalies (bugs) et les parties du jeu qui nécessitent d'être retravaillées. Pour moi ce n'est pas un métier, c'est une compétence (mais ça n'engage que moi...). Même si on trouvera des testeurs attitrés chez les gros studios, combien sont testeurs professionnels de jeux vidéo sur Terre ? Et combien veulent faire ce métier ? Ne faîtes pas le calcul, vous auriez peur. Donc si vous vous engagez dans des études de testeurs (ça existe) : ayez un plan B, apprenez aussi à programmer par exemple.

Et il y a bien sûr les métiers du management : chef de produit, chef de projet, etc. A savoir qu'un simple programmeur peut évoluer vers un poste de chef de projet ou de chef de produit dans sa carrière.

Les combinaisons

Le top, ce sont les combinaisons. Voici mes favorites :

- Le programmeur / *Game Designer*

Il est capable d'imaginer un jeu et de le créer ! Qui mieux que lui va combiner l'imagination avec les contraintes (ou possibilités) techniques de la machine ?

- Le programmeur / *Game Designe*r / Infographiste

Il est souverain ! Il peut créer un jeu de A à Z (hormis le son et la musique). C'est souvent eux qui donnent vie à des jeux d'exception qui marquent les esprits...

Les idées reçues

Créer un jeu vidéo, c'est compliqué...

FAUX !

Si on part de ce principe, tout est compliqué ! Cuisiner c'est compliqué, conduire c'est compliqué, apprendre une nouvelle langue c'est compliqué, et même... apprendre à lire c'est compliqué !

Beaucoup de personnes se trompent parce-qu'elles regardent le haut de la montagne et se disent : "c'est compliqué, c'est impossible d'aller là-haut !". **Avant de devenir sportif de haut niveau, il faut d'abord apprendre à marcher !** Vous ne croyez pas ? Pourtant beaucoup tenteront de vous apprendre à courir le 100m en 10 secondes dès le début. Pourquoi ? Je ne sais pas, je sais juste qu'ils se trompent.

L'erreur classique, c'est de commencer à apprendre la programmation de jeux vidéo dans sa globalité, ou de commencer par un outil trop complexe (bien que les programmeurs aguerris vous affirmeront "avec ça c'est facile !").

Je vous l'affirme : apprendre à programmer, c'est plus facile que d'apprendre à lire !

Avec ma méthode, **mon fils de 14 ans a appris à programmer en quelques jours**. J'ai passé moins de 5 heures à le former... Aujourd'hui, il crée ses propres jeux vidéo et souhaite faire une école d'ingénieur en informatique !

J'ai moi-même appris seul à programmer, alors que mes professeurs prétendaient que je n'étais pas fait pour l'informatique à cause de mes notes en mathématiques ! **Aujourd'hui je crée des jeux vidéo professionnellement et j'en vis confortablement !** Heureusement que je ne les ai pas écoutés !

Je l'affirme : On peut apprendre à programmer rapidement, et on peut créer un premier jeu vidéo FACILEMENT ! **A condition de se donner un objectif raisonnable et de commencer par les fondamentaux.**

Si vous pensez que votre 1ère création sera un jeu en 3D, jouable en réseau, c'est que vous espérez gravir l'Everest lors de votre 1ère expédition… C'est perdu d'avance.

Ma méthode d'apprentissage est basée sur ce principe : **chaque objectif est réalisable**, qu'on ait 14 ans ou 77 ans, et la progression est raisonnée. A partir de là, tout devient possible !

Il faut être bon en mathématiques...

FAUX !

Bien entendu, être adepte des mathématiques est un avantage. Certains concepts seront plus faciles à appréhender. **Pourtant on peut tout à fait réussir à créer des jeux vidéo sans véritable talent pour les mathématiques.**

Je n'ai personnellement aucune notion réelle de mathématiques ou de trigonométrie, je ne sais même plus ce qu'est le théorème de Thalès, ou encore, je ne sais pas calculer l'aire d'un cercle, ou calculer un angle...

Pourtant, certaines de ces notions ont pu m'être nécessaires. Et je les ai utilisées. Mon secret ? GOOGLE !

Tout est disponible sur Internet. Il est bien plus facile de devenir un expert en recherche d'information sur les forums et autres sites Internet que de maîtriser les mathématiques et la trigonométrie. Bonus : au détour d'une recherche, on apprend des choses qu'on n'avait pas eu envie d'apprendre à l'école !

Et si on ne trouve pas la réponse, il suffit de poser la question. Une armée de passionnés se fera un plaisir de vous expliquer, exemple à l'appui.

Autre réalité : beaucoup de langages de programmation intègrent des notions de mathématiques complexes sous forme de fonctions prédéfinies.

Par ailleurs, certains types de jeux vidéo ont peu recours aux mathématiques. Il est peu probable d'avoir besoin de calculs savants pour un jeu de type Pac Man, ou Frogger (pourtant ces 2 concepts ont généré des millions de dollars en 2015 avec Crossy Road et Pac Man 256 !).

Alors, si vous n'avez pas la bosse des maths… pas de problème, bienvenue !

Il faut faire une école spécialisée…

FAUX ! Mais ça peut aider.

Le saviez-vous ? De nombreux hits du jeu vidéo on été créés par des amateurs, qui n'avaient aucun diplôme. Je n'ai moi-même qu'un BAC de Secrétaire Comptable (et oui, je sais bien taper à la machine du coup…).

Le diplôme ne fait pas tout (j'ai par ailleurs oublié toutes notions de comptabilité...).

L'école, c'est un cadre pour plus d'immersion

Voici les avantages de suivre un cursus de formation ou un cursus scolaire :

- **Vous vous assurez des heures à apprendre et pratiquer**. Peut-être que vous n'auriez pas cette capacité seul à la maison.

- **Vous aurez à vos côtés des professeurs, encadrants, camarades, plus qualifiés que vous** et vous pourrez bénéficier de leur connaissance.

En résumé : l'école permet un accès plus rapide à la connaissance et à la pratique… Mais la connaissance et la pratique peuvent s'obtenir sans école.

Un exemple que j'affectionne et que je cite en référence : Chris Davis, le créateur du jeu **The Escapist**, était couvreur, et n'avait aucun diplôme. Les *Succes Stories* de ce type sont nombreuses !

Au passage : je n'ai aucun diplôme et je vis du métier de programmeur depuis 20 ans, dont 8 ans dans le domaine du jeu vidéo.

Un diplôme aide votre plan de carrière

Si vous souhaitez faire carrière, un diplôme vous ouvrira des portes (certains gros studios recrutent sur diplômes), ou facilitera l'obtention d'un Visa pour les USA (sans diplôme d'état, un VISA pour les US c'est mission impossible, vous le saviez ?).

Mais encore une fois, **vous pouvez apprendre à créer des jeux vidéo sans aller à l'école, en apprenant seul ou avec l'aide de formations adaptées**.

Savoir programmer avant d'aller à l'école est un avantage

Savoir créer des jeux vidéo avant même de commencer votre cursus de formation vous permettra d'approfondir vos connaissances, au lieu de ramer sur les bases : ces dernières seront déjà acquises !

Cela va prend beaucoup de temps pour apprendre…

FAUX !

Apprendre les bases de la programmation, c'est très rapide si vous suivez la bonne méthode.

On peut apprendre les bases en quelques heures

Je prétends qu'en fonction de votre capacité à apprendre, cela peut prendre seulement quelques heures. Ensuite, avec l'envie et la pratique, quelques heures par semaine suffiront à faire de vous un créateur de jeux vidéo.

Il y a bien entendu des personnes plus douées que d'autres, mais c'est valable dans tous les domaines.

Par exemple, savoir lire ne signifie pas qu'on va devenir un gros lecteur. Certaines personnes n'ont pas le goût de la lecture, et cela leur demandera plus d'efforts.

Programmer, créer, c'est la même chose.

Il faut apprendre les fondamentaux… pour savoir si on est fait pour ça

Apprendre les fondamentaux pour savoir coder un petit jeu vidéo, c'est un bon moyen de savoir si vous êtes fait(e) pour ce métier.

Ma méthode d'apprentissage est basée sur l'apprentissage de fondamentaux :

- **5 fondamentaux pour maîtriser un langage de programmation**
  - Les variables et expressions
  - Les fonctions
  - Les structures de contrôle
  - Les tableaux et listes
  - La programmation Objet

- **5 fondamentaux pour créer un jeu vidéo à partir de ce langage**
  - Afficher
  - Déplacer
  - Animer
  - Jouer un son
  - Recevoir des entrées

Avec cette méthode, il aura fallu 5 heures (maximum !) pour que des adolescents apprennent à programmer un jeu vidéo à mes côtés. Seulement 5 heures… Bien entendu, la pratique permettra d'appréhender des concepts plus complexes. Comme je l'ai déjà expliqué : **si on apprend peu chaque fois, on progresse petit à petit, pour finalement réussir à faire des choses très complexes !**

Il faut savoir parler anglais…

PRESQUE FAUX !

J’ai appris sans savoir parler anglais, j’avais 13 ans. Je me suis débrouillé comme je pouvais, il m’a suffit de comprendre quelques termes.

C’est plus difficile aujourd’hui. Sur Internet, beaucoup d’informations utiles sont disponibles en anglais.

Il va falloir apprendre l’anglais tôt ou tard

Des nouvelles méthodes existent, basées sur un concept totalement différent de celui que propose l’enseignement scolaire…

Avez-vous appris le français en commençant d’abord par la grammaire, la conjugaison ? Non… vous l’avez appris lorsque vous étiez enfant, en l’écoutant et le parlant,. Nombreuses sont les nouvelles méthodes qui se basent sur ce postulat : pratique et écoute, sans aborder la théorie, qui ne sera abordée qu’une fois la langue parlée instinctivement.

Pour vous débrouiller sur Internet, inutile d’être très bon. Votre objectif, si vous avez des lacunes en anglais, est donc raisonnable. Lancez-vous.

Comment apprendre à programmer et se renforcer

Comment apprendre les bases de la programmation : ma méthode

Utilisez un langage simple et appliquez **ma méthode des 5 fondamentaux** :

- 5 fondamentaux pour **maîtriser un langage** de programmation
  - Les variables et expressions
  - Les fonctions
  - Les structures de contrôle
  - Les tableaux et listes
  - La programmation Objet

Une fois ces 5 fondamentaux maîtrisés, vous pourrez apprendre les 5 autres fondamentaux :

- 5 fondamentaux pour **créer un jeu vidéo** à partir de ce langage
  - Afficher
  - Déplacer
  - Animer

- Jouer un son
- Recevoir des entrées

Quel matériel ?

Un PC de base fera l'affaire, même une antiquité, pour peu qu'il soit connecté à Internet. C'est tout ce qu'il vous faut pour commencer !

Le passage à la 3D pourra nécessiter un ordinateur plus puissant, mais ceci viendra en son temps et vous serez déjà plus aguerri !

Dernier point : PC ou Mac, tout est disponible sur les 2 systèmes, inutile donc de faire un choix. Personnellement, je code mes projets sur les 2 systèmes en même temps car j'ai un PC au bureau, et un Macbook à la maison...

Par quel langage commencer ?

Je conseille de **commencer par Lua avec Love2D** (https://love2d.org), puis d'évoluer vers un langage plus élaboré comme Haxe ou Java, et enfin passer au C# (pour Unity).

Le passage par le C et C++, pour les plus courageux, sera une expérience d'une exceptionnelle richesse.

Combinez ces langages avec des Frameworks bien sûr. Ce sont les Frameworks qui vous apportent ce qu'il faut pour afficher, animer, etc. Sinon un langage seul ne sert à rien...

Quel framework pour ces langages ? Voici ceux que je conseille et pourquoi !

Pour Lua : Love2D

Lien : https://love2d.org/

Pourquoi Lua ?
Apprendre à programmer avec Lua va vous permettre de comprendre la programmation sans vous encombrer de toutes les contraintes qui découragent la plupart des apprentis programmeurs. Le langage est hyper simple, facile à mettre en oeuvre, on peut commencer en tapant ses premières lignes de code sans enrobage de déclarations et autres directives de programmation d'aucune sorte !

Pourquoi Love2D ?

Avant de commencer avec des outils plus haut niveau, je considère indispensable de savoir comment créer un jeu vidéo sans l'aide de librairies qui vont faire le travail à votre place. Avec Love2D : trois fonctions load/update/draw, il ne manque que votre imagination, et l'assurance de comprendre ce qui fait le coeur d'un jeu vidéo.

Pour Haxe : HaxeFlixel

Lien : http://haxeflixel.com/

Pourquoi Haxe ?
Haxe est un langage et un *toolkit* créé en France, par Nicolas Cannasse. Il est un condensé de bonnes pratiques en matière de programmation, facile à apprendre (une fois qu'on est passé par le Lua) et il introduit tous les concepts haut niveau que vous rencontrerez en C++, Java ou C#. Il est gratuit et Open Source, et la communauté qui le supporte est très active. Et, cerise sur le gateau : il peut cibler, avec le même code, le PC, le Mac, les mobiles, le Web et même les consoles Next Gen !
C'est pour moi un outil d'avenir, que les programmeurs Flash (AS3) adoptent en masse depuis la mort annoncée de Flash, attirés par OpenFL le Framework de référence de Haxe et qui s'appuie sur l'API Flash !

Pourquoi Haxeflixel ?
C'est un Framework éprouvé (portage de Flixel, un Framework célèbre de Flash) et très bien pensé. Il vous donnera tout ce dont vous avez besoin pour créer un jeu vidéo et il est parfaitement documenté. Basé sur OpenFL, vous avez l'assurance de sa robustesse et de son évolutivité.

Pour Java : LibGDX

Lien : https://libgdx.badlogicgames.com/

Pourquoi Java ?
C'est le langage de référence pour beaucoup, et il vous assurera la reconnaissance de vos compétences sur le marché du travail. Il vous apportera une bonne maîtrise des concepts de programmation Objet et nombreux jeux sont développés en Java. Il vous permettra aussi de programmer autre chose que des jeux vidéo, et si vous visez Android, vous aurez tôt ou tard besoin de coder quelques lignes de Java...

Pourquoi LibGDX ?

Je n'utilise pas Java, je n'en suis pas un expert, mais j'ai pu noter que dans mon réseau, LibGDX était une référence. Ma référence : Halfway, jeu que j'affectionne particulièrement (http://halfwaygame.com).

Pour C# : Unity

Lien : https://unity3d.com/

Pourquoi C# ?
Déjà, parce que Unity3D… Mais aussi parce que c'est l'évolution naturelle du C++. En maîtriser la syntaxe, à l'instar de Java, vous ouvrira les portes d'autres outils (présents ou futurs) et vous assurera une belle mention sur votre CV !

Pourquoi Unity3D ?
La référence… Et la clé pour un CV réellement attirant. Par contre, l'erreur pour beaucoup est de commencer à apprendre avec Unity. C'est tellement loin des concepts de programmation traditionnel, tellement complexe en matière d'interface éditeur, que 90% de ceux qui commencent par Unity abandonnent et se disent "créer un jeu vidéo c'est pas pour moi". La faute à tous ceux qui colportent l'idée reçu qu'avec Unity on n'a quasiment pas besoin de programmer, que c'est facile à apprendre et accessible à tous. Un beau mensonge, et bravo à l'équipe marketing de Unity !
Attention encore : ne misez pas tout sur Unity ! Arriver sur le marché du travail avec une seule compétence vous pénalisera. De plus, n'apprendre qu'un seul outil parce qu'il est une référence est une erreur et sera à terme une grande faiblesse (voir mon chapitre consacré à ce sujet délicat…).

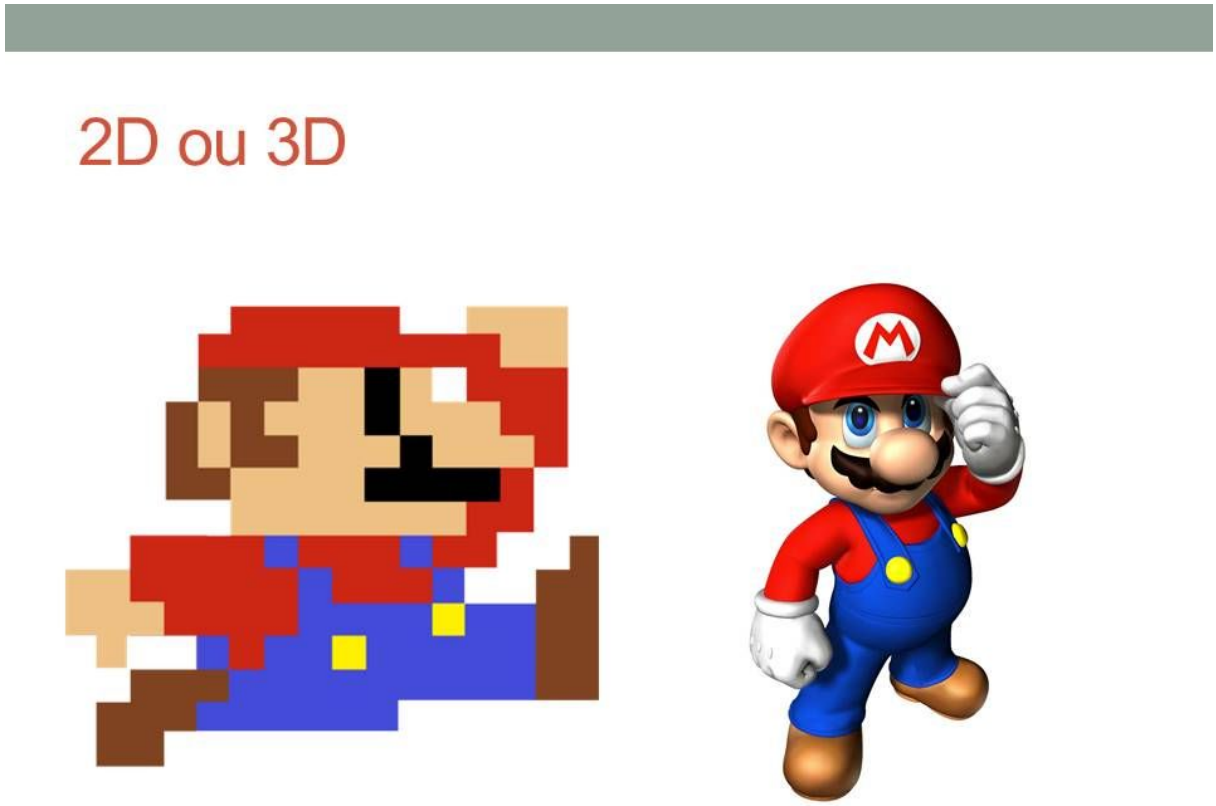
Pour aller plus loin

Alors bien sûr, chacun de ces outils est en anglais… Il va falloir déchiffrer un peu ou vous améliorer pour pouvoir lire les documentations, voire même simplement installer ces outils. Ne me blâmez pas, c'est ainsi. Par contre, sachez que **je prépare des ateliers en vidéo, en Français, qui pourront vous apprendre à installer ces outils et à suivre pas à pas ma méthode des 5 fondamentaux**.

Pour plus d'informations sur mes ateliers, rendez-vous sur www.gamecodeur.fr

Maintenant que vous savez quel langage et quel Framework je vous conseille de choisir, commencez à apprendre ! Voici quelques étapes indispensables...

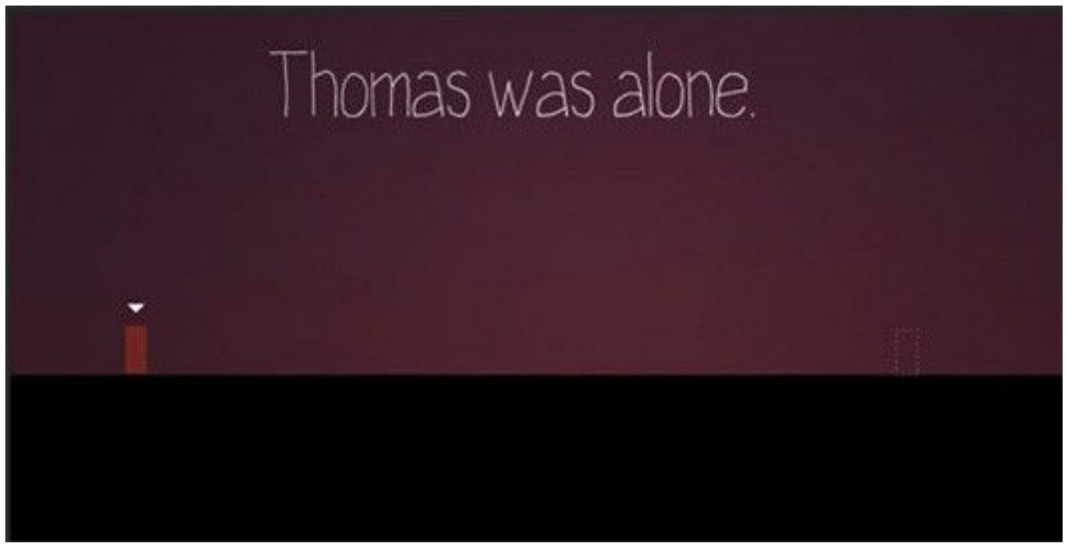
Etape 1 : Commencez par choisir entre 2D et 3D



Il faut commencer par la 2D !

Tous les concepts de programmation seront là, et vous pourrez évoluer vers la 3D qui va vous obliger à ingurgiter un masse de concepts complémentaires.

De plus, la 2D permet d'avoir un résultat rapide avec une exigence moindre. Faire un jeu en 2D avec un résultat visuel acceptable c'est à la portée de tous car on peut faire preuve d'imagination pour utiliser un minimum de graphismes :



En 2D, tout est possible avec de l'imagination

Et puis il y a le retour en force du pixel sur le marché :

Mais dès qu'on met le doigt dans la 3D, on n'a plus droit à l'approximation. Tout a vite un aspect ringard...

Anyone can learn? Oui mais *en 3D, on a vite l'air ringard...*

Et apprendre la 3D n'est pas obligatoire. On peut être un grand créateur de jeu vidéo et n'avoir jamais abordé la 3D. La 2D peut être un choix technique, mais aussi artistique.

Si vous débutez et que vous voulez échouer tout de suite, faïtes comme beaucoup conseillent : installer Unity3D et suivez un tuto sur la création d'un jeu en 3D... Au bout de plusieurs heures vous aurez un semblant de jeu à l'écran, mais vous n'aurez rien compris à ce que vous avez fait, et vous abandonnerez comme 95% des apprentis programmeurs...

Croyez-moi, commencez en 2D avec un vrai langage de programmation...

Etape 2 : Apprenez les bases de la programmation

Commencez pas installer un des outils que je vous ai conseillé, par exemple Love2D si vous débutez.

Programmer est plus simple que vous ne le pensez, c'est comme faire une liste de courses :

```
Début du programme

Va au supermarché !
Cherche du beurre Bio
Si tu ne trouve pas de beurre Bio
```

```
    Prend de la margarine Bio
Sinon
    Prend du St Hubert Omega 3
Fin
Reviens du supermarché

Fin du programme
```

Ensuite, à votre rythme, apprenez en suivant ma méthode des 5 fondamentaux :

- Apprenez ce qu'est une variable et une expression
- Apprenez à créer des fonctions, les appeler, et leur ajouter des paramètres
- Apprenez les structures de contrôle (boucles, conditions, etc.)
- Apprenez à créer des tableaux et des listes (indispensable pour les niveaux, les objets à l'écran, les ennemis, les tirs, etc.)
- Apprenez les rudiments de la programmation Objet (POO), ce n'est pas si compliqué que ça en a l'air !

Beaucoup de tutoriels ou de documentations ne suivent pas cet ordre… hélas ! Il va donc falloir avancer "à la carte", en cherchant la bonne information dans l'ordre que je vous conseille, sans brûler les étapes.

Par la suite vous pourrez complexifier : gérer des collisions, des *tilemaps*, de la physique… Abordez chaque concept individuellement !

Un de mes patrons (du temps où j'en avais un…) m'a dit un jour : *si tu dois avaler un éléphant, découpe-le en petits morceaux*.

Rappel : **Je prépare des ateliers en vidéo, en Français, qui pourront vous apprendre à installer ces outils et à suivre pas à pas ma méthode des 5 fondamentaux**.

Voici comment j'illustre la Programmation Objet :

Même la programmation objet c'est facile

Objet : **Voiture**
Méthodes :
- Démarre
- Roule(destination)
- Stoppe

- MaVoiture = Nouvelle Voiture
- MaVoiture.Démarre()
- MaVoiture.Roule(supermarché)
  - Cherche du beurre Bio
  - Si tu ne trouve pas de beurre Bio
    - Prend de la margarine Bio
  - Sinon
    - Prend du St Hubert 41
  - Fin
- MaVoiture.Roule(maison)
- MaVoiture.Stoppe()

Etape 3 : Continuez en créant un projet très modeste

Il est très formateur d'apprendre en ayant un projet. Une fois que vous avez les bases de la programmation, appliquez ma méthode des 5 autres fondamentaux en ayant choisi un projet.

Je conseille des jeux rétro comme :
- Casse Brique
- Pac Man
- Frogger
- Space Invader
- etc.

Voici mes 5 autres fondamentaux :

Les 5 fondamentaux

1. Afficher
2. Déplacer
3. Animer
4. Jouer des sons
5. Recevoir des entrées (clavier, souris)

En maîtrisant ces 5 fondamentaux du jeu vidéo et les bases de la programmation, vous pourrez déjà créer des jeux !

Alors sans aide, il faudra de la persévérance ! N'hésitez pas à passer du temps sur chacun des fondamentaux, recommencez, ragez, mais n'abandonnez pas !

Sachez que dans les années 80, il n'y avait pas Internet. Seuls quelques livres nous aidaient. On devait donc se débrouiller parfois avec des bribes d'informations. Nous apprenions par l'échec et la frustration. Peut-être la meilleure école...

Voici à quoi ressemblait mes premiers programmes en 1985 sur Amstrad CPC :

Le langage BASIC

Et pour apprendre j'avais juste le livret vendu avec l'ordinateur… Finalement tout était simple à l'époque. A vous de vivre cette simplicité aujourd'hui. Limitez vos sources d'informations. Internet est une jungle et peut donner le tournis… c'est donc à vous de résister à la panique !

N'hésitez pas à suivre des tutos en vidéo, voire à investir un peu dans des cours particuliers : changer de vie vaut bien quelques dizaines d'euros dépensées !

De plus, investir vous assurera du contenu de qualité, et la possibilité de poser des questions.

Maintenant que vous savez programmer et créer des petits jeux…, je vous conseille d'apprendre un autre langage de programmation ! Voici pourquoi...

Pourquoi apprendre plusieurs langages et outils fera de vous un meilleur programmeur ?

Ne succombez pas aux sirènes de l'outil universel ! On vous dira aujourd'hui que pour créer un jeu il vous suffit de connaître C# et Unity… Il y a quelques années c'était Java, avant ça le C++, etc. Et demain ?

Ceux qui survivront sont ceux qui auront “appris à apprendre”

Ceux qui survivront et dont les studios s’arracheront les services seront ceux pour qui l’outil ou le langage n’auront aucune importance. Il s’agira de ceux qui ont compris si profondément ce qu’est la programmation que changer de langage ne leur prendra que quelques jours.

Oui : ceux-là auront la capacité à apprendre un nouveau langage en quelques jours !

Lorsque j’ai décidé d’apprendre Lua, j’ai commencé mon 1er projet dans la même journée… Idem quand je suis passé à Haxe : j’ai commencé directement à prototyper Lost Colonies en Haxeflixel, sans avoir eu besoin de me former en profondeur, j’ai appris sur le tas en quelques heures ! Pourquoi ? Car je sais apprendre, mon cerveau est calibré pour ça.

Lost Colonies : Space Dungeons, prototypé en quelques semaines à deux, avec un langage que nous utilisons pour la première fois, vient de recevoir une aide du CNC et verra le jour en 2016 !

Un conseil primordial : vous devez favoriser votre capacité d’apprentissage en diversifiant vos apprentissages.

Je vous conseille d’apprendre au moins 5 outils et langages différents

Une anecdote qui va vous amuser :

J’ai déjà été amené à tester 5 candidats programmeurs ayant été formés 3 ans sur Java dans une grande école nationale réputée (je n’en donnerai pas le nom...).

Ils ont été incapables, même en une journée, d’apprendre un langage type BASIC et de dessiner des carrés à l’écran ! D’autres, comme des IUT, ont réussi en 2h...

Ceux qui ont échoué n’avaient pas appris à apprendre… Trouver seuls la bonne information (en anglais mais très accessible), installer un outil, en comprendre les bases en quelques minutes : ils ne savaient pas le faire. Sans compter leur incapacité à appréhender un problème simple, trop habitués à

traiter des sujets complexes (ou à se les complexifier parce-que plus c'est compliqué, plus on flatte son orgueil…).

Je répète mon conseil : diversifiez vos connaissances en expérimentant de nombreux outils et langages de programmation.

Voici encore quelques pistes :

Langages :

- Basic (Monkey X ou autre)
- C / C++
- Java
- C#
- Lua
- Javascript

Outils / *Frameworks* :

- Gamemaker (http://www.yoyogames.com/studio)
- Haxe (avec un *framework* comme Haxeflixel)
- LibGDX
- Et même la Pico-8 (www.lexaloffle.com/pico-8.php), expérience très enrichissante et qui vous fera découvrir tout l'univers de la contrainte et du minimalisme !

Et pour la 3D

- Unity3D
- Unreal Engine
- Et pourquoi pas Haxor (http://www.haxor.xyz/)

Bien entendu cette liste n'est pas exhaustive, mais ce sont des pistes sérieuses.

Mon premier jeu vraiment commercial, **Geisha: The Secret Garden, a été créé en Blitmax, un framework utilisant un langage type BASIC**. Cela m'a permis d'aller très vite, de me focaliser sur le Gameplay. **Je l'ai terminé en 6 mois** et distribué dans le monde entier.J'en ai vendu plus de 35 000 copies et il s'est téléchargé à plus de 500 000 exemplaires. Il est même sorti en boite dans les supermarchés en France et aux USA.

D'autres auraient crié au scandale :

- “Du BASIC ? Faut que tu maîtrises ta chaîne de production mec, faut que tu te codes ton propre moteur en C++ !
- Ouais ouais… Au fait il sort quand ton jeu toi ? Tu sais, celui sur lequel tu bosses depuis 2 ans ?”

On peut aussi commencer sans programmer, comment ?

Comment faire sans infographiste et sans *Sound Designer* ?

Avancer sans graphismes

On peut commencer un jeu en utilisant soit des formes géométriques, soit des images (moches ou pas) téléchargées sur Google Image ! Attention ce sera temporaire, juste pour avancer, et vous devrez changer pour des images dont vous avez les droits pour commercialiser un jeu.

En attendant, avec cette méthode, vous ne perdez pas de temps et validez au plus vite vos idées en allant droit au but.

Créer soit même ses graphismes

Pourquoi ne pas apprendre à faire ses propres graphismes ? Si des infographistes rêvent de devenir programmeur (et y parviennent…), pourquoi pas l’inverse ?

Quelques outils utilisés par la communauté 2D :

http://pyxeledit.com/ (excellent, je vous conseille d’investir dans cet outil)

http://www.aseprite.org/ (gratuit)

http://www.getpaint.net/ (gratuit)

L’outil de référence 3D gratuit :

http://www.blender.org/

Acheter / télécharger des graphismes légalement

On peut aussi investir quelques dollars (je vous le conseille, payer pour avancer est assez gratifiant, on se sent “pro”). J’utilise ce site en particulier :

https://www.gamedevmarket.net/

Gratuits mais de moindre qualité (il faut fouiller) :

http://opengameart.org/

Et on peut aussi télécharger des graphismes de jeux 2D célèbres (à condition de ne s'en servir qu'à des fins de loisir sinon gare à la violation de droits...) :

http://www.spriters-resource.com/

Créer ses propres sons gratuitement

Pour vos bruitages, l'excellent AFXR, dans sa version améliorée et "online" renommée BFXR pour l'occasion :

http://www.bfxr.net/

Il vous permet de créer toutes sortes de petits sons "8 bits", idéal pour une Game Jam !

Acheter des sons pros pour quelques dollars

J'achète mes sons chez Sound Rangers. Chaque son vaut quelques dollars et l'énorme choix permet de se passer d'un Sound Designer mais faut pas le dire...

http://www.soundrangers.com/

Créer des jeux vidéo sans programmer

Il est tout à fait possible de créer des jeux vidéo sans programmer. Certains jeux que vous connaissez peut-être, et qui ont fait la fortune de leur créateur, ont été créés sans une seule ligne de programmation !

Alors bien sûr, il y a des inconvénients; à vous de voir s'ils sont gênants pour vous :

- Vous n'apprendrez pas à programmer réellement, même avec des outils qui permettent de mixer avec du code
- Vous ne pourrez pas toujours viser les consoles (mais certains de ces outils le permettent)
- Vous aurez des limites, et même si elles sont surmontables, on passe parfois tellement de temps à bidouiller, alors qu'il nous aurait fallut quelques minutes en sachant programmer…

Je conseille néanmoins ces outils pour ceux qui coincent sur la programmation, afin qu'ils puissent créer des jeux vidéo sans complexe. Et un outil tel que Gamemaker ou encore Stencyl permet de programmer, donc cela peut être un tremplin pour la suite !

Gamemaker

http://yoyogames.com/

L'outil qui monte ! Avec ses nombreux succès récents (dont Risk of Rain, Hotline Miami, Hyper Light Drifter...), il est aujourd'hui reconnu par le "milieu" indé.

Multimedia Fusion

http://www.clickteam.com/fr

La référence en France, très souvent utilisé dans les écoles de *Game Design*. Pour l'anecdote, parmi ses créateurs on retrouve François Lionet, créateur de l'AMOS sur Amiga. L'outil qui se cache derrière le jeu millionnaire Five Nights at Freddy's...

Stencyl

http://www.stencyl.com/

Produit intéressant car permettant de programmer, si besoin, en Haxe.

Construct 2

https://www.scirra.com/

L'outil qui a été utilisé par Aurelien Regard pour créer Next Penelope. Belle communauté et bel outil. A tester.

Apprendre en s'imposant des contraintes

Jouer les faussaires

La plupart des artistes apprennent en recopiant les œuvres des artistes qu'ils admirent. Je vous conseille d'adopter le même comportement. Bien entendu il ne s'agit pas de recréer un jeu dans sa totalité, mais plutôt d'en étudier une technique, un élément qui vous plaît et que vous aimeriez être capable de reproduire dans vos jeux.

Pour que l'objectif soit réalisable, encore une fois, il faut qu'il soit modeste.

Voyez l'exercice comme un jeu.

Exemple : reproduire l'effet de nuages défilants dans l'arrière plan de **TowerFall Ascension**. Il n'est même pas nécessaire de posséder le jeu, une simple vidéo Youtube suffit.

Le Retro-Gaming comme source d'inspiration

Le Retro-Gaming est aussi une source d'inspiration et d'apprentissage inépuisable. Les meilleurs *gameplay* sont là, et à l'époque, ils étaient très variés ! Si vous ne connaissez pas encore l'émulateur MAME, c'est indispensable de vous y mettre. Il vous permettra de jouer à tous les jeux d'arcade des années 70 à 90.

http://www.mame.net/ (pour les ROMS… je vous laisser chercher…).

Saviez-vous qu'un jeu multi-millionnaire comme **Crossy Road** est inspiré du jeu **Frogger** ? Ses auteurs ont eu la chance de signer ensuite avec Bandai Namco et de réaliser **Pac Man 256**...

The Escapists, vendu à plus de 500.000 exemplaires sur Steam, est inspiré de **Skool Daze**, un jeu de ZX Spectrum…

Down The Mountain, jeu mobile à succès, est basé sur **Q-Bert**, un jeu d'arcade des années 80…

Cloner des "standards" est un exercice que je recommande comme une priorité. Vous découvrirez la richesse des mécaniques de l'époque (où tout était basé sur les 1ères secondes de jeu pour capter le joueur) et l'imagination débordante des créateurs pour compenser la faiblesse du *hardware*...

S'imposer des limites pour multiplier les exercices

Donnez-vous un temps limité pour réaliser l'exercice (voir la méthode des pomodoros). Il faut que chacun de ces exercices soit réalisé dans un temps très court.

Faire des *Game Jam* peut changer votre vie

Dès que vous connaîtrez les rudiments de la programmation de jeux vidéo, intéressez-vous aux *Game Jam*.

Ce sont des concours de programmation de jeux vidéo dans un temps limité, avec un thème imposé. Le standard est de créer un jeu en 48h.

C’est extrêmement formateur car c’est comme monter dans un autobus avec des amis : on finira par arriver quelque part et on est obligé d’avancer.

Je conseille de faire un maximum de *Game Jam*, sans vous mettre la pression sur les 48h ou le résultat : déjà commencer une *Game Jam* vous fera progresser, même si vous abandonnez en cours de route.

Je conseille ces *Jam* :

- La Ludum Dare (prononcez Loudoum Daré)
  http://ludumdare.com/compo/
- La Global Game Jam
  http://globalgamejam.org/
- La One Game a Month (un jeu par mois)
  http://www.onegameamonth.com/

J’organise par ailleurs chaque année la Retro Game Jam sur Montpellier : http://retrogamejam.com !

Suivez ces *jam* de près ou de loin, postez ou pas vos jeux à la fin, mais ne négligez pas cet exercice !

Bonus : **nombreux succès sont nés de projet de *Jam*** : Nuclear Throne, Badass Inc, Titan Souls, Evoland, Metrocide, etc.

Vivre de sa passion

Trouver un job dans le jeu vidéo, quelques conseils

Il m’est très difficile de donner des conseils dans ce domaine. Je peux tout de même vous donner une approche qui m’est propre et qui s’est montrée efficace tout au long de ma carrière. Cette méthode fonctionne par ailleurs pour tous les métiers...

Demandez-vous ce qu’attend votre employeur, mettez-vous à sa place. Embaucheriez vous le candidat ou la candidate que vous êtes ?

Voici le genre de questions, en temps qu’employeur, que vous vous poserez :

- Est-ce qu’il ou elle m’inspire confiance ?

- Est-ce que je ressens un bon feeling, et partager mon quotidien avec cette personne me sera t'il agréable ?
- Est-ce qu'il va me faire perdre du temps, de l'argent, ou au contraire en gagner ?
- Est-il ou est-elle polyvalent(e) ou au contraire spécialisé(e) ?
- Est-ce que je connais cette personne ? Est-ce que quelqu'un de mon entourage la connaît ?
- Est-ce que je peux vérifier rapidement ses qualités, ses références ?
- Que dis Google sur lui ou elle ?

Vous voyez, tout ceci est humain. Un employeur va plus facilement embaucher quelqu'un qui va lui faire gagner du temps (et donc de l'argent), qu'il connaît déjà (ou que quelqu'un connaît pour lui), qui a de l'humour, est souriant(e) et inspire confiance.

Construisez-vous une image dans le milieu : game jams, événements, salons, site web, bonne réputation numérique, réseau et amis dans le milieu, activité associative, connaissance du marché et des contraintes financières…

Pour trouver des annonces, je vous conseille le site de l'AFJV, référence en la matière :
http://emploi.afjv.com/index.php

Et soyez mobile ! Si vous cherchez du travail prêt de chez vous, vous perdez 90% d'opportunités… Votre futur boulot est quelque part sur terre, il faut vous préparer à bouger. Et apprenez l'anglais...

Quelques méthodes de travail

La méthode TODO / DOING / DONE

Une méthode simple, et qui marche

Pour commencer un projet et s'organiser pour le réaliser, c'est simple :
- Découpez votre projet en "**cas d'utilisation**"
- Créez un tableau à 3 colonnes TODO / DOING / DONE
- Listez ces actions dans la colonne TODO
- Bossez !

C'est trop simple ? Pourtant beaucoup de grands l'utilisent… Et c'est quasiment la même méthode que dans les usines Toyota au japon (via la méthode Kanban).

Qu'est-ce qu'un cas d'utilisation ? (En anglais on dit plutôt *Use Case*...)

C'est quelque chose que peut faire votre joueur, ou le jeu lui même. Par exemple :

- Le joueur choisit "Jouer" dans un menu
- Le joueur choisit son niveau dans une liste
- Le jeu sauvegarde la position du joueur à chaque changement de niveau
- Le personnage peut sauter
- Les ennemis poursuivent le personnage quand il s'approche d'eux
- etc.

Chaque action devient une tâche pour vous, et vous savez enfin quoi faire !

Donnez-vous un délai de réalisation global, et des objectifs à la journée. Utilisez la méthode des *pomodoros* pour avancer vite (voir plus loin).

Trello, l'outil gratuit pour appliquer cette méthode

Vous pouvez organiser ces tâches dans un outil gratuit tel que https://trello.com/.

Keep it simple!

Je vous conseille d'avoir une organisation la plus simple possible, des *post-it*, une simple liste sur un papier... et non pas de tout informatiser ! Sinon vous abandonnerez vite la mise à jour de vos listes de tâches, croyez-en un "vieux de la vieille" qui gère des projets depuis 20 ans !

Les *pomodoros*

La technique des *pomodoros* consiste à vous concentrer 25 minutes sans interruption, faire 5 minutes de pause, et recommencer. C'est une technique simple et efficace si vous avez des problèmes de concentration. Un *pomodoro* ne doit pas s'interrompre !

Voir :
https://fr.wikipedia.org/wiki/Technique_Pomodoro

Bonus : Les 10 commandements du programmeur de jeux vidéo professionnel

Voici les 10 commandements qui sont affichés dans mon entreprise. Pour la peine, je vais devoir vous tutoyer...

1) Je coderai pour les autres

Le code est parfaitement structuré (indentation et découpage) et en anglais si c'est dans un cadre professionnel. Les méthodes et fonctions sont courtes, aérées. Le nommage est clair (notation hongroise). Le code est pensé pour résister aux bugs et effets de bords inattendus (programmation défensive). Les points clés sont commentés, et les commentaires sont écrits au fur et à mesure (et non pas après coup sinon ils seront stupides). Demande-toi régulièrement : si je donne ce code à mon collègue, aura-t-il encore envie d'être mon ami ?

2) Je mettrai mon projet en sécurité

Le projet est en sécurité sur un ordinateur bien entretenu. Un *repository* (utilisez Git ou Bitbucket) est créé dès le 1er jour et chaque jour un *commit* est réalisé. Le projet doit pouvoir se dupliquer sur un autre poste en quelques minutes. Les fichiers de travail (PSD, TXT, etc.) doivent être stockés dans un répertoire distinct et sauvegardés eux aussi sur le *repository*. Ne laisse rien au hasard et permets-toi un peu de paranoïa.

3) J'atteindrai l'objectif final à mi-chemin

Si le projet doit être terminé en 2 mois, il faut avoir toutes les principales fonctionnalités en place au bout d'un mois. Idéalement tout est mis en place de manière globale (*rough*) dès le début du projet. De manière générale, pense toujours en mode *Jam*. Puisque tu peux faire un jeu en 48h, comment peux-tu faire aussi peu en plusieurs semaines ? Pose-toi les bonnes questions et lance-toi des défis.

4) Je traiterai mon oeuvre comme un tableau

Un tableau est crayonné totalement, puis les couleurs générales sont posées, puis des détails sont ajoutés sur l'ensemble du tableau, par raffinements successifs. Le projet doit subir le même traitement : tout en place (*workflow*) dès le début, puis ajout des fonctionnalités générales, puis amélioration de chaque partie de manière lissée. Il est inutile de trop aller dans le détail d'une *feature* au détriment des autres.

5) Je ferai un retro-planning

En début de projet, puis à intervalle régulier, tu devras avoir une vue générale de ce qu'il reste à faire. Et tu devras visualiser ces tâches en perspective du temps qu'il te reste. Découpe le planning en semaines, et répartis clairement les fonctionnalités (*use-cases*) du projet sur ces semaines. Ensuite, assure-toi

régulièrement que tu respectes les délais. Dans le cas contraire, recommence le travail de planification sur la durée restante.

6) Je ferai une version par semaine

Fixe-toi un jour de la semaine (idéalement le mercredi ou le vendredi) pour compiler, *packager* et livrer une version. Cela peut être en interne au départ, puis vers l'extérieur dès que le produit est utilisable. Tu appelleras cela une VI (Version Interne) et tu lui donneras un numéro. Ainsi, tu penseras toujours ton projet comme quelque chose de livrable, tu ne négligeras rien et tu gagneras du temps (beaucoup de temps !) sur la fin du projet.

7) Je donnerai vie chaque jour à ma création

Le polish ne se fait pas d'un coup à la fin du projet. Car le polish participe à l'expérience utilisateur, et l'expérience utilisateur s'évalue tout au long du projet. Il faut penser le polish au quotidien, et non comme un point final. Un bouton à afficher : puis-je le faire apparaître avec un *tween* (effet d'affichage animé) ? Un écran à afficher : puis-je faire un fondu ? Une information à afficher : que puis-je faire pour la rendre agréable à découvrir ? Etc.

8) Pour moi, le mieux sera l'ami du bien

Autre comportement classique du développeur : faire le minimum pour s'épargner travail, stress et déception. Pourtant, la plupart du temps, faire "mieux" ne prend pas plus de temps que de faire "bien". Quand tu commenceras une fonctionnalité, tu devras de te demander si tu n'es pas en train de faire seulement "bien" alors que tu pourrais faire "mieux". Inspire-toi des produits que tu admires pour prendre ta décision. Ta petite voix te dis "ça prendra trop de temps !" : elle ment. Par contre, ne complique pas inutilement. Inutile de prévoir à l'avance ce dont tu n'es pas sûr d'avoir besoin un jour. Fait simple, mais pas simpliste !

9) Je montrerai rapidement à mon prochain

Tous les développeurs ont inconsciemment peur de montrer leur travail. De plus, le voyage les intéresse plus que la destination. Tu dois avoir conscience de ce comportement qui peut nuire au projet. Tu devras montrer rapidement et souvent. Si tu respectes les autres commandements cela devrait être aisé. Ce n'est pas encore prêt à être montré ? Tu as sûrement organisé cette impasse. Remets-toi en question. Ton objectif principal : la confrontation avec la réalité !

10) Je ferai bien du 1er coup

Tu ne produiras pas de code “sale” en te disant : je le nettoierai quand ça marchera. Tu ne laisseras pas des dizaines de lignes en commentaire “pour plus tard”. Tu ne coderas pas négligemment pour gagner du temps. Tu ne laisseras pas consciemment des bugs pour les corriger plus tard.

Tu feras bien du 1er coup !

Le dictionnaire du programmeur de jeux vidéo

2D (2 dimensions)

Style et technique graphique de prédilection des jeux indépendants. Très souvent associé au Pixel Art, la 2D est devenue une forme d'art. Dans ce style, les graphismes sont dessinés et animés à plat.
Voir : 3D

3D (3 dimensions)

Style graphique réaliste, utilisant des technologies complexes pour créer et simuler du volume. Bien plus complexe à mettre en œuvre et bien plus exigeant en terme d'effort et de qualité que la 2D.
Voir : 2D

AAA

Genre de jeu couteux, réalisés par de grosses équipes et délivrant un contenu conséquent. C'est l'opposé d'un jeu indépendant. On peut le comparer au cinéma indépendant par opposition aux « Block busters ». On dit « Triple A ».
Voir : Jeu indépendant

Bug / Bogue

Le cauchemar du programmeur. C'est une erreur dans son code (ou dans son éditeur, ou dans son Framework...) générant un comportement gênant ou bloquant du programme. La recherche de bugs s'appelle « déboguage » (*debugging* en anglais).

Codeur

C'est un mot dérivé du mot « code » qui désigne les lignes d'instructions d'un programme. C'est la manière décontractée de qualifier un programmeur. Un programmeur est ennuyeux, un codeur est cool !

Editeur / IDE

Un éditeur est un outil permettant de faciliter la vie du programmeur de jeu vidéo. Il permet de réaliser des opérations qui auraient normalement nécessité de saisir des lignes de code. Un IDE est un outil similaire, mais

destiné à faciliter la saisie du « code », son exécution et sa vérification (déboguage).

FPS

Frame Per Second. C'est le nombre d'images par seconde qu'un jeu vidéo est capable d'afficher. Un jeu est jugé « fluide » à partir de 30 FPS mais l'idéal recherché par tout bon codeur est d'atteindre 60 FPS.

Game Jam

C'est un concours de création de jeux vidéo dans un temps limité, habituellement entre 24 et 48h. C'est le loisir régulier de tout bon programmeur de jeux vidéo, lui permettant de justifier la consommation de pizza et l'oubli de son hygiène corporelle.
Voir Ludum Dare.

Graphismes

C'est le terme désignant les images affichées par un jeu vidéo. On dit d'un jeu qu'il a de « bons graphismes », ou « des graphismes tout pourris ». Le métier de celles et ceux qui créent des graphismes (bons ou pourris…) s'appelle : Infographiste.
Voir 2D et 3D.

IDE

Voir Editeur.

Intelligence Artificielle (IA)

Il s'agit de simuler, par un programme, une certaine forme d'intelligence. Le comportement autonome d'un ennemi dans un jeu par exemple. On dit parfois « Une IA de crustacé ».

Jeu indépendant

Non, ne me demandez pas la définition d'un jeu indépendant…
Voir AAA.

Langage (de programmation)

Lorsqu'on programme un jeu vidéo, on donne des instructions à l'ordinateur. On les formule en respectant une syntaxe et une notation précise, c'est le langage de programmation. Il existe de nombreux langages de programmation.

Ludum Dare

C'est la *Game Jam* la plus célèbre. On prononce « Loudoum daré », sinon on a l'air ringard…
Voir : Game Jam.

Programmation orientée objet (POO)

C'est une forme avancée de programmation. Cela permet de programmer en organisant mieux son code, et de frimer auprès de ceux qui programment de manière traditionnelle.

Retrogaming

C'est une pratique récente, consistant à jouer à des jeux vidéo plus anciens, souvent sur un matériel lui aussi ancien : consoles de salon des années 80 ou 90, etc. C'est une pratique indispensable pour tout bon créateur de jeux vidéo.

Sprite

C'est un élément graphique, la plupart du temps animé et qui se déplace à l'écran. Par exemple, le personnage principal d'un jeu. Autrefois le nombre de *sprite* affichés à l'écran était limité par le matériel.

Pour me contacter :
David MEKERSA
Email : david@gamecodeur.fr
Téléphone : 06 33 74 54 40

GAMECODEUR
École de jeux vidéo en ligne

Suivez mes ateliers sur Gamecodeur.fr !

Des ateliers en vidéo, en Français, pour apprendre à installer tous les outils que je conseille et suivre pas à pas ma méthode des 5 fondamentaux.

Chacun d'eux s'accompagne de supports écrits, d'exercices pratiques et d'un *coaching* individuel !

DE PLUS JE VOUS OFFRE 4H DE FORMATION GRATUITE

Pour plus d'informations sur mes ateliers et pour suivre ma formation gratuite "les bases de la programmation", rendez-vous sur :

www.gamecodeur.fr

Pour vous inscrire à ma mailing list et recevoir mes prochains guides, conseils, et ateliers, inscrivez-vous sur :

http://eepurl.com/bLb5sH

Et merci de m'avoir lu jusqu'à la dernière page !