Ludification pour la motivation en apprentissage de la programmation

S. Hoarau

LIM (Laboratoire d'Informatique et de Mathématiques), Université de La Réunion
seb.hoarau@univ-reunion.fr

Abstract. Dans le but de motiver des étudiants de L1 en Sciences à programmer, nous avons testé des plateformes de programmation informatique ludique. Nous livrons ici notre retour d'expérience et nos questionnements sur deux d'entres elles, codingame.com et checkio.org particulièrement bien conçues et très actives. Quels sont les atouts de ces plateformes et comment les utiliser ? Certaines se tournent déjà vers l'enseignement et pourraient demain offrir de nombreuses possibilités pédagogiques.

1 Introduction

Comme le rappellent S. Ali Houssein et Y. Peter dans leur état de l'art des outils de soutien à l'enseignement de la programmation [1], la tâche d'apprentissage de la programmation est complexe. Pourtant, dès le lycée il semble que nous ne mettons pas tout en œuvre pour stimuler nos futurs étudiants : "apprentissage de l'algorithmique au papier-crayon avec peu de programmation et le plus souvent avec une calculatrice" nous rapporte C. Nijimbere dans sa thèse sur l'enseignement des savoirs informatiques [6].

A l'Université de La Réunion, des étudiants de première année scientifique suivent une unité d'enseignement intitulée Initiation à la Programmation. Chaque année, le taux d'échec est élevé (environ 80%). Ce taux d'échec est directement lié au taux d'abandon (plus de 50%): les étudiants, découragés, ne viennent plus en cours, ni même en séances de travaux dirigés et ce dès la deuxième ou troisième séance. Dès lors, il devient difficile pour eux de réussir. Une enquête qualitative auprès des apprenants nous renseigne sur quelques uns des facteurs de cet abandon. Parmi ces facteurs le manque de motivation revient systématiquement. C'est sur cette motivation que nous avons décidé d'agir en jouant sur plusieurs critères : des séances d'appentissage variées (alternance de petites séquences de cours traditionnels, mini évaluations par QCM, résolution *en live* d'exercices plus complets), utilisation de plateformes d'apprentissage de la programmation par le jeu. C'est ce que nous nommons la ludification, une réponse possible à la question que se posent les auteurs de Alice, un environnement interactif 3D pour l'aide à l'apprentissage de la programmation orientée objet [8] : "Comment enseignons-nous ?". Question qui selon eux, et nous sommes entièrement d'accord, va de pair avec "A qui enseignons-nous ?".

L'analyse de S. Ali Houssein et Y. Peter a l'intérêt de mettre en avant le fait que la plupart des outils d'aide à l'apprentissage de la programmation se focalisent sur l'évaluation automatique du code et/ou la présentation graphique de l'exécution de ce code. Cela semble indiquer l'importance de ces deux critères. Les plateformes que nous avons testées les intègrent tous deux à un niveau plus ou moins avancé. Pourquoi le choix de plateformes développées par des entreprises privées, à l'origine non prévues pour de l'enseignement ? Lorsqu'on regarde le tableau des outils de [1], on constate que la plupart datent d'avant 2010 et qu'ils ont un côté vieillot [3, 4] peu à même de rallumer la motivation d'un étudiant de L1. D'autre part, comme le souligne [5] dans leur PIDR sur la PLM de M. Quinson et G. Oster [2], le temps de développement d'un outil sérieux d'aide à la programmation est réellement conséquent et nécessite de la ressource humaine qualifiée, ce que ne possède pas forcément une petite université comme la notre.

C'est pourquoi nous nous sommes intéressés à ces plateformes très professionnelles sur leurs designs et leurs fonctionnalités. Et même si au départ elles ne sont pas prévues pour des activités d'enseignement, elles sont en plein essor comme le montre les dernières évolutions, mineures (plugin navigateur pour pouvoir utiliser son éditeur de texte favori à la place de l'IDE) ou majeures (possibilité de créer des classes, intégration de cours en ligne).

Dans la suite de cet article, nous proposons d'étudier deux plateformes de ce genre pour mettre en avant leurs avantages et leurs inconvénients et voir comment les utiliser conjointement pour tirer le meilleur des deux.

2 Quelles sont les principales difficultés des apprenants ?

Suite à une enquête qualitative sur un petit échantillon d'étudiants de L1 Sciences, nous avons répertorié quelques points de difficultés soulevées par les apprenants :

- exercices de difficulté pas assez progressive
- exercices trop souvent à contenu mathématiques et **peu motivants**
- manque d'interactions avec les enseignants (chargés de cours ou de TD)

Le point concernant le côté peu motivant des activités proposées est particulièrement frappant : les étudiants n'ont pas envie de persévérer dans la recherche de la solution. Ils abandonnent et attendent le résultat. Or, il est certain que la difficulté de la résolution d'un problème informatique nécessite une grande persévérance associée à une envie de *tenter des choses*. C'est essentiellement dans l'optique de remettre de la motivation que nous avons testé deux plateformes d'aide à la programmation par le jeu.

D'autre part, comme nous le disions en introduction deux critères semblent indispensables pour un outil d'aide à la programmation :

- L'évaluation automatique du code : l'apprenant, s'il aime que son enseignant s'intéresse à lui, apprécie également un peu d'autonomie devant la machine, surtout lorsqu'il répète plusieurs fois les mêmes erreurs et qu'alors la correction par une machine le rassure quant au sentiment de ne pas être jugé.

– La qualité graphique de la plateforme ; cet aspect joue un grand rôle dans l'exécution du programme de l'apprenant mais aussi plus basiquement dans l'interface utilisateur : les étudiants ont plus envie de *jouer* avec un outil qui est beau.

3 Ludification par des plateformes existantes

3.1 Codingame.com

Probablement une des meilleurs plateformes du genre. À l'origine créée pour servir de plateforme de recrutement [9], Codingame est fréquentée par des milliers de programmeurs du monde entier. Le concept de jeu repose sur des missions ou challenges plus ou moins difficiles à résoudre via un programme informatique (plus d'une trentaine de langages possibles). Les activités peuvent se classer en deux catégories :

– les puzzles d'entrainements où on trouve des puzzles solo de difficulté croissante (facile, moyen, difficile et très difficile)
– les challenges : permettant d'affronter d'autres programmeurs. On y trouve les petits challenges à 8 programmeurs max et limité à 5 min (les *clash of code*) et les combats de bots d'une très grande qualité à la fois graphique et algorithmique.

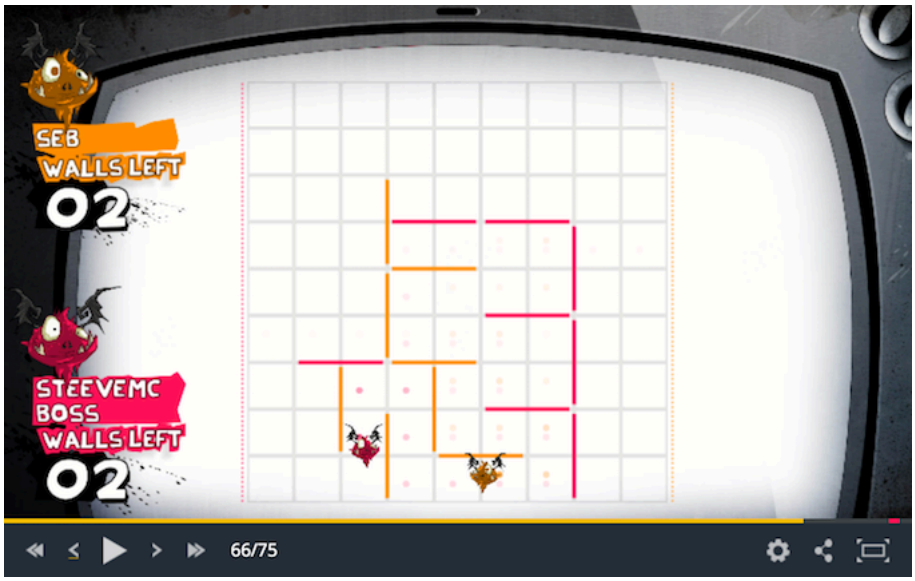

Fig. 1. The Great Escape, un challenge de bots de la plateforme codingame.com

C'est l'un de ces challenges multi-joueurs de combat de bots que nous avons testé : The Great Escape (fig.1) est la version combat de bots du jeu Quoridor (voir par exemple https://fr.wikipedia.org/wiki/Quoridor). Le principe est simple à 2 ou 3 joueurs, chacun contrôle un dragonnet qui démarre sur le côté d'une grille 9x9 et doit se rendre sur le côté opposé. A chaque tour, le dragonnet peut avancer d'une case dans l'une des 4 directions ou bien poser un mur pour tenter de ralentir ses adversaires.

La programmation d'une technique efficace dépasse largement le niveau programmeur débutant (algorithmes de plus court-chemin, de min-max, retour sur trace etc.). Toutefois, si l'enseignant code des bots particuliers (ne plaçant que des murs verticaux avec un trou de passage et ne cherchant pas à atteindre le bord opposé par exemple, fig.2) et de difficulté croissante, on peut inciter les apprenants à progresser. Cette année plusieurs apprenants sont arrivés à coder l'algorithme de Dijkstra vu en cours et à avoir un dragonnet efficace. Un mini tournoi a même pu être réalisé lors d'une séance en amphi.

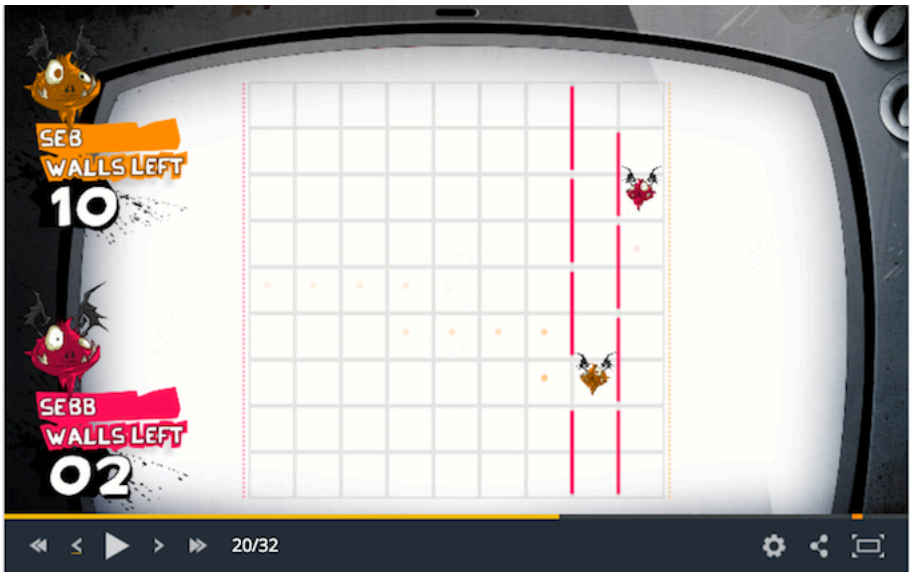

Fig. 2. Le dragonnet rouge ne joue pas vraiment et place simplement des murs pour tester la capacité du jaune. Ici nul besoin d'un algorithme sofistiqué

La prise en main à la fois des notions élémentaires de programmation et de l'interface de la plateforme nécessite néanmoins plusieurs heures sur les puzzles solo, et surtout ceux de difficulté *facile*. Or c'est là un des points faibles de cette plateforme : les puzzles, même dits simples ne le sont pas forcément pour des débutants et surtout ne sont pas assez nombreux. Cependant, comme le rappelle S. Ivanova [7], codingame n'est pas prévu au départ pour des débutants mais bien

des gens qui ont déjà des compétences en programmation et qui veulent soit progresser, soit se faire repérer par un éventuel recruteur. D'autre part Codingame n'offre pas d'environnement *enseignant* pour suivre efficacement l'évolution des apprenants d'une même classe.

Codigame sera donc a privilégier pour ses challenges de bots ou ses *clash of code* de 5 minutes. Il faut lui associer une autre plateforme pour le début de l'apprentissage, ce qu'on appelle l'initiation. Une plateforme comme codecademy peut éventuellement être utilisée mais elle est bien moins ludique et ressemble d'avantage à un minimooc. Nous avons testé une autre plateforme plus ludique, elle aussi basée sur un ensemble de missions : checkio.org.

À suivre le futur projet de Codingame : incorporer des cours **gratuits** à la plateforme (https://www.codingame.com/contribute/my-courses) ; projet inspiré du Khan Academy Project : "You can learn anything. For Free. For Everyone. Forever" (https://www.khanacademy.org).

3.2 Checkio.org

Checkio.org se présente comme un ensemble de *planètes-cailloux* (fig.3) à conquérir en réalisant les missions associées. Ces missions sont des problèmes de programmation. Elles sont nombreuses, réparties toujours sur quatre niveaux de difficulté (elementary, simple, moderate, challenging) et documentées quant aux notions abordées (boucles, chaînes de caractères etc. ce qui est un plus lors du choix du problème par l'apprenant).

Fig. 3. La page d'accueil de checkio.org avec ses planètes à conquérir

Néanmoins la résolution des missions n'est pas visuelle comme dans codingame. De plus, parmi les exercices élémentaires beaucoup sont de nature mathématiques ; nous retrouvons donc une des critiques courantes des étudiants s'initiant à la programmation.

Les deux principaux avantages de checkio.org, sont sa non linéarité (même au niveau élémentaire on a le choix entre plusieurs îles pour commencer son exploration) et la possibilité (récente) pour un enseignant de créer une classe et gérer l'avancée des apprenants. Signalons enfin un troisième point intéressant : chaque exercice résolu apporte des points, la publication de sa solution à un forum dédié à l'exercice en apporte également de même que la relecture et le commentaire du code d'un autre participant. Cette incitation à partager son code et à en discuter représente une activité intéressante et bénéfique dans l'apprentissage de la programmation.

4 Première expérimentation

Après une première année d'expérimentation pendant laquelle nous avons surtout testé plusieurs plateformes (codecombat, codecademy, codingame, checkio) pour pouvoir choisir laquelle utiliser avec les étudiants, nous avons opté pour codingame la plateforme la plus aboutie en termes de simulation graphique du résultat du code de l'apprenant.

Ci-dessous nous livrons quelques inconvénients qui nous ont fait abandonner les autres plateformes :

- codecombat : utilisation d'un pseudo code, une sorte de surcouche au-dessus du véritable langage (python ou javascript), le modèle économique (la version gratuite est vite limitante)
- codecademy : pas basé sur du jeu même si la ludification par des points et des badges est présente, c'est plus une plateforme d'apprentissage de type MOOC

Les cours magistraux ont permis d'introduire les concepts de base de la programmation python ainsi que l'interface de codingame. Il n'y a pas de séances de travaux pratiques en première année. Une séance de travaux dirigés entièrement consacrée à codingame a permis aux étudiants de prendre en main l'interface. Puis, sur deux séances de TD, les étudiants divisés en deux groupes traitaient deux énoncés différents : l'un à réaliser classiquement l'autre via la plateforme codingame. A la suite des TD nous avons demandé aux étudiants de répondre à un mini questionnaire portant sur le temps passé sur l'exercice en dehors de la séance en présentielle, sur le degré de réussite du TD et sur sa qualité. Malheureusement, arrivées tardivement dans le semestre ces séances de TD ont souffert de la déjà très grande désaffection et très peu ont répondu aux questionnaires (moins de 20) rendant toute interprétation non significative.

Les trois ou quatre dernières semaines de l'UE ont été consacrées à l'algorithme de plus court chemin Dijkstra et à la réalisation par les étudiants en groupes de deux ou trois d'un code pour les différentes phases du challenge The Great

Escape. La phase finale de TGE a été organisée lors d'une séance de cours avec matches en direct.

Les résultats enregistrés à cette UE ne sont pas meilleurs que ceux des années précédentes. Néanmoins, nous gardons espoir car plus de 170 étudiants ont adhéré au tournoi The Great Escape alors qu'à la même période ils n'étaient plus qu'une centaine à assister aux cours magistraux.

Cette première expérimentation nous a permis de détecter plusieurs points négatifs à corriger :

- Déjà un grand nombre d'abandons au moment de l'utilisation de la plateforme : il conviendra d'améliorer la communication pour ne pas perdre d'étudiants avant le début de l'utilisation de la plateforme.
- Missions pas assez progressives sur codingame ce qui explique que les étudiants ont été découragés prématurément. Lors de l'interrogation de certains étudiants, nous avons appris que plusieurs n'avaient pas osé répondre au questionnaire : ils ne voulaient pas annoncer que malgré un temps conséquent passé sur les missions ils n'avaient pas réussi à les résoudre.
- Feedback trop faible des chargés de travaux dirigés qui, à mon avis, n'ont pas suffisamment investi la plateforme.

5 Conclusion et perspectives

Nous avons présenté deux plateformes d'initiation à la programmation par le jeu. Bien que non conçues spécifiquement pour faire de l'enseignement, ces outils offrent une façon ludique de présenter des exercices de programmation et la possibilité aux apprenants de se confronter les uns aux autres ou de partager leurs solutions. D'autre part, certaines de ces plateformes s'orientent peu à peu vers l'enseignement et offrent la possibilité de créer une *classe* pour le suivi des apprenants.

Nous avons particulièrement apprécié les challenges de bots de la plateforme codingame même si le niveau de difficulté élevé nécessite de ruser un peu pour y amener des débutants de façon progressive.

En jonglant entre les plateformes d'enseignement classiques type moodle ou claroline et ces plateformes de jeu, l'enseignant en programmation a réellement la possibilité de créer un environnement riche et attractif ou l'apprenant aura le choix et la qualité.

Nous comptons poursuivre nos efforts l'année prochaine en nous servant des deux plateformes conjointement :

1. checkio.org en début de formation pour la facilité des exercices et la fonctionnalité *classe* permettant un meilleur suivi. Nous nous servirons également plus de la ludification de type badges, progression par niveau pour inciter les apprenants à atteindre des paliers.
2. codingame.com pour le tournoi de bots qui sera le point d'orgue de la formation, à réaliser dès que possible et sur la durée de l'UE, type mini projet mais aussi exploiter les mini-challenges *clash of code* en séances récréatives pendant les cours magistraux.

Concernant la massification, techniquement ces plateformes sont déjà prévues pour des dizaines de milliers d'utilisateurs et supportent donc parfaitement la charge. Quant aux problèmes classiques rencontrés lors de l'évaluation d'une filière (comment empêcher les apprenants de se donner des solutions par exemple) ils ne sont pas nouveaux. Au contraire, on pourrait imaginer le choix de quelques missions clés qui doivent rester verrouillées jusqu'à la fin de l'apprentissage pour servir de missions d'évaluation finale.

References

1. S. Ali Houssein, Y. Peter: État de l'art des outils de soutien à l'enseignement / apprentissage de la programmation. Acte de l'atelier Apprentissage Instrumenté de l'Informatique (2016)
2. M. Quinson, G. Oster: A Teaching System to Learn Programming: the Programmer's Learning Machine, Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, 260–265 (2015)
3. N. Guibert, L. Guittet, and P. Girard: Initiation à la Programmation "par l'exemple" : concepts, environnement, et étude d'utilité. Acte Colloq. EIAH'05, Montpellier, pp. 461–466 (2005)
4. A. Moreno, N. Myller, E. Sutinen, and R. Bednarik: Visualizing programs with Jeliot 3. Proc. Work. Conf. Adv. Vis. interfaces. ACM, pp. 373–376 (2004)
5. M. Morainville, C. Huguenin: Serveur d'applications pour la Programmer's Learning Machine. Rapport Technique, Projet Interdiciplinaire ou Découverte de la Recherche, TELECOM Nancy (2014)
6. C. Nijimbere: L'enseignement de savoirs informatiques pour débutants, du second cycle de la scolarité secondaire scientifique à l'université en France. Une étude comparative Thèse de Doctorat, Volume 1, Université Paris Descartes (2015)
7. S. Ivanova: Learning computer programming through games development. 12th International Scientific Conference eLearning and Software for Education (2016)
8. A. Florea, A. Gellert, D. Florea and A.-C. Florea: Teaching programming by developing games in Alice. 12th International Scientific Conference eLearning and Software for Education (2016)
9. S. Caullier: Des challenges en ligne pour mieux recruter. www.lemonde.fr (2014)