Actes de l'atelier "Apprendre la Pensée Informatique de la Maternelle à l'Université"
Brest, France, June 13, 2023

SPY : Un jeu sérieux partagé pour étudier l'apprentissage de la pensée informatique

Mathieu Muratet[1, 2]
(1) Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
(2) INSHEA, 92150 Suresnes, France
mathieu.muratet@lip6.fr

RÉSUMÉ

Dans cet article nous présentons le jeu sérieux SPY comme un EIAH partagé à la fois pour son usage, son code source et ses traces d'interaction. SPY est un jeu sérieux *open source* sur le thème de la pensée informatique. Nous présentons les scénarios actuellement inclus dans le jeu comme une illustration des champs des possibles. Ces scénarios sont utilisables en l'état par des enseignants qui souhaiteraient faire travailler leurs élèves sur les compétences de la pensée informatique. Nous montrons comment le jeu peut être étendu à travers la création de nouveaux scénarios et niveaux. Enfin nous partageons les données produites par le jeu au format xAPI afin de créer une base de données ouverte permettant aux chercheurs de la communauté de les analyser.

ABSTRACT

SPY: a shared serious game to study computational thinking.

In this paper we present the serious game SPY as a shared ITS for its use, its source code and its traces. SPY is an open source serious game on computational thinking. We present the scenarios currently included into the game as an illustration of the scope of possibilities. These scenarios can be used by teachers who would like to make their students work on computational thinking. We show how the game can be extended through the creation of new scenarios and levels. Finally, we share the data produced by the game in xAPI format in order to create an open database available to researchers.

MOTS-CLÉS : jeu sérieux, pensée informatique, données ouvertes, partage.
KEYWORDS: serious game, computational thinking, open data, sharing.

1 Introduction

Depuis maintenant quelques années, l'informatique disciplinaire réapparaît dans les programmes scolaires (Baron et Drot-Delange, 2016), on parle de pensée informatique. Selon Wing (Wing, 2006), la pensée informatique met en jeu un répertoire de cinq capacités cognitives : (1) la pensée algorithmique, (2) l'abstraction, (3) l'évaluation, (4) la décomposition, et (5) la généralisation. L'appropriation de ces compétences par les enseignants de l'école primaire est complexe à résoudre. En effet, enseigner cette nouvelle discipline demande un investissement particulièrement important de leur part (Kradolfer *et al.*, 2014) notamment en raison d'un manque de formation (initiale et continue). Ainsi, par manque de maîtrise de ces compétences les enseignants se trouvent en

difficulté pour construire leurs propres scénarios pédagogiques ou juger de la pertinence d'outils pour faire travailler ces compétences à leurs élèves (Roche *et al.*, 2018).

De nombreuses ressources permettant de travailler les compétences de la pensée informatique existent. Certaines mettent en avant l'informatique débranchée qui est un point d'entrée de la science informatique riche et qui présente l'avantage de ne pas nécessiter de matériels coûteux (Alayrangues *et al.*, 2017). Ces mêmes auteurs notent que mener de telles activités requiert des connaissances en informatique, et des compétences en pédagogie. Ce « super prof » maîtrisant ces deux facettes reste une exception, le passage à l'échelle avec ce type de scénario pédagogique reste donc difficile.

D'autres ressources exploitent des robots programmables. Ici l'objet tangible fournit un artefact fertile sur le plan cognitif pour développer des compétences mathématiques et des formes de pensées algorithmiques (Komis et Misirli, 2011). Ces robots programmables sont alors exploités dans le cadre de scénarios pédagogiques. L'enseignant doit alors maîtriser les concepts sous-jacent pour soutenir les travaux des élèves. Michel Spach (Spach, 2019) souligne là encore, dans le contexte de la robotique pédagogique, les freins au développement de ces ressources : « Le défaut de maîtrise conceptuelle des enseignants est sans doute à l'origine du manque de référencement et du déficit d'institutionnalisation des notions et des concepts abordés dans les situations pédagogiques ».

Enfin, de nombreuses ressources numériques et ludiques complètent ce panorama (Miljanovic et Bradbury, 2018 ; Vahldick *et al.*, 2014). Ces EIAH (Environnement Informatique pour l'Apprentissage Humain) couvrent un spectre large allant de l'enseignement primaire à l'université ; de l'initiation à la programmation avec des langages à base de blocs à la programmation d'intelligences artificielles avec des langages avancés. L'offre est donc riche et consistante même pour les élèves de l'école élémentaire, pour autant les ressources existantes restent difficilement adaptables (Saddoug *et al.*, 2022), d'une part car la plus part de ces EIAH ne fournissent tout simplement pas d'outils informatiques permettant de modifier les scénarisations existantes ou d'en créer de nouvelles et, d'autre part, pour les rares qui le permettent comme Kodu, Scratch ou Code.org, ils ne proposent pas d'aide à l'enseignant pour comprendre les situations créées comme, par exemple, expliciter les compétences en jeu.

Les ressources numériques présentent un certain nombre d'avantage par rapport aux approches débranchées ou robotiques. D'une part, d'un point de vue de l'usage elles ne demandent pas l'achat de ressources spécifiques comme c'est le cas pour les robots de sol et peuvent embarquer des aides à la fois à destination des apprenants et des enseignants. D'autre part, d'un point de vue recherche en informatique elles présentent l'avantage de pouvoir générer des traces d'interactions qui peuvent être étudiées pour analyser les productions et les comportements des apprenants mais aussi des enseignants.

Cet article a pour objectif de présenter et partager un EIAH particulier sur le thème de la pensée informatique, il s'agit du jeu sérieux SPY que nous présenterons dans la section 2. Ce jeu sérieux est présenté dans une perspective d'ouverture vis à vis de la communauté (section 3) afin de proposer une ressource libre d'utilisation, modifiable et partagée (à la fois du point de vue de son code source et des données produites). Enfin nous conclurons cet article en proposant quelques pistes de recherches.

2 SPY : le jeu

SPY est un jeu sérieux d'apprentissage sur le thème de la pensée informatique. Il a été conçu pour un public d'élèves de cycle 3 (CM1, CM2, 6ème). Le principe du jeu est de programmer un ou plusieurs agents à l'aide d'actions simples afin de les déplacer sur une grille vers des positions particulières. Ces actions sont représentées sous forme de blocs que le joueur doit agencer en séquences.

Pour programmer le robot, le joueur bénéficie de blocs d'action (« Avancer », « Pivoter à gauche », « Pivoter à droite », « Attendre », « Activer un terminal » et « Faire demi-tour »), de blocs de contrôle (« Si Alors », « Si Alors Sinon », « Répéter n fois », « Tant que » et « Répéter indéfiniment »), de capteurs (« Mur en face », « Mur à gauche », « Mur à droite », « Passage en face », « Passage à gauche », « Passage à droite », « Ennemi en face », « Zone surveillée en face », « Porte détectée en face », « Terminal détecté sur ma position » et « Sortie détectée sur ma position ») et d'opérateurs (« Et », « Ou » et « Non »).

La figure 1 présente le tutoriel du jeu, dans cette scène le joueur doit faire avancer le robot d'une seule case et dispose pour cela d'une seule action « Avancer » qu'il doit glisser dans la zone d'édition de « Karl » (le nom du robot). La figure 2 présente un niveau avancé du scénario « Sélectionneur » (avec la solution) demandant au joueur de programmer deux robots avec une seule ligne de programme, le joueur devra alors trouver une solution générale permettant de résoudre deux problèmes proches mais différents. Le joueur devra alors manipuler des structures de contrôle, des capteurs et des opérateurs. La solution incluse dans la figure 2 peut être traduite ainsi « tant que je ne suis pas sur la sortie, avancer d'une case et si un mur est détecté sur la prochaine case en face alors pivoter à gauche (fin tant que) ».

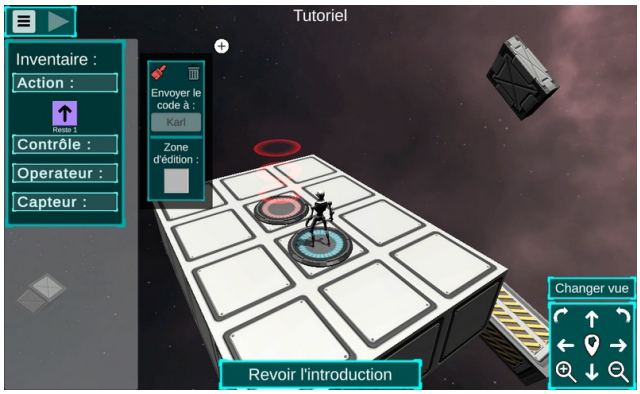

Figure 1: Tutoriel du jeu SPY

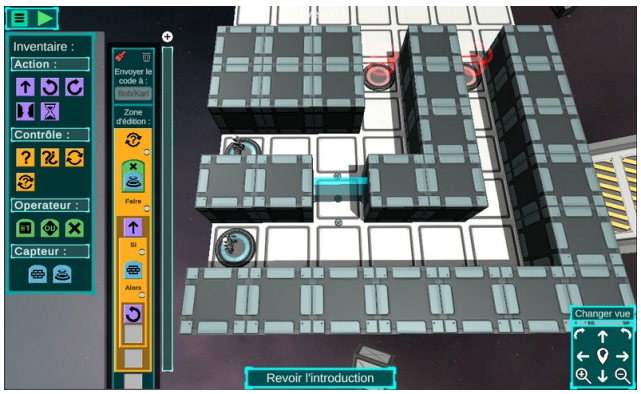

Figure 2: Un niveau du scénario « Sélectionneur »

Au moment de l'écriture de cet article, SPY contient 6 scénarios. Le premier intitulé « Explorateur » est composé de 9 niveaux et ne demande au joueur de ne manipuler que des blocs d'action. Les 4 premiers niveaux donnent seulement accès au blocs requis, le joueur doit donc les agencer dans le bon ordre sans avoir à se demander si tel ou tel bloc est utile ou non. Les 5 derniers niveaux fournissent les blocs d'action en quantité illimitée, c'est alors au joueur de déterminer quels blocs doivent être utilisés et dans quel ordre. Pour chacun de ces niveaux le joueur peut décomposer son programme en sous parties, il peut déposer quelques blocs, exécuter son programme, observer la nouvelle situation et poursuivre sa résolution sans avoir à recommencer du début.

Le deuxième scénario intitulé « Collaborateur » est composé de 12 niveaux. Dans ce scénario le joueur devra faire collaborer deux robots. Il devra notamment apprendre à nommer correctement les zones de programmation afin de contrôler les bons robots. Dans ce scénario 3 niveaux ne demandent pas au joueur de programmer mais proposent des solutions pré-construites, le joueur doit alors lire ces programmes, les comprendre et choisir celui qu'il souhaite envoyer au robot. Deux niveaux contiennent des solutions erronées que le joueur doit corriger.

Le troisième scénario intitulé « Répétiteur » est composé de 10 niveaux. Ce scénario est centré sur la notion de répétition. Ici les blocs actions sont limités non pas pour réduire la charge cognitive du joueur mais pour forcer l'utilisation du bloc « Répéter n fois ». Ce scénario exploite aussi les agents ennemis qui contiennent des actions que le joueur peut consulter mais ne peut pas modifier. Le joueur doit alors anticiper les actions de l'ennemi et programmer son robot en conséquence.

Le quatrième scénario intitulé « Sélectionneur » est composé de 8 niveaux. Ce scénario permet de découvrir les structures de contrôle « Si Alors » et « Tant que » ainsi que quelques capteurs et opérateurs. Deux niveaux de ce scénario cachent la position à atteindre, le joueur ne peut donc pas savoir où déplacer son robot, l'algorithme de résolution est donc donné dans la consigne en langage naturel ou sous la forme d'un algorigramme. Le joueur doit alors traduire cette consigne à l'aide du langage formel du jeu.

Le cinquième scénario intitulé « Infiltration » est composé de 22 niveaux. Il s'agit d'un scénario de synthèse qui reprend l'ensemble des compétences travaillées dans les 4 premiers scénarios. Dans ce scénario, les niveaux sont liés les uns aux autres au travers d'une histoire originale.

Le sixième scénario intitulé « BlocklyMaze » est une reproduction du jeu originale du même nom[1]. L'objectif ici était de vérifier qu'il était possible de décrire les niveaux d'un autre jeu avec le modèle de niveau de SPY. En effet, l'ensemble des niveaux de SPY sont décrit dans des fichiers XML extérieurs au jeu, il est ainsi possible de créer ses propres niveaux de jeu et de les agencer sous la forme de scénarios.

Cette présentation succincte du jeu et de ces scénarios a pour objectif de donner un aperçu du champ des possibles qui ne demande qu'à être étendu par la création de nouveaux niveaux, de nouveaux scénarios, de nouvelles fonctionnalités, de nouvelles analyses de données. Dans la section suivante nous présentons les différentes composantes du jeu partagé.

3 SPY : l'objet partagé

SPY est tout d'abord partagé en tant que ressource libre d'utilisation. Le jeu est accessible en ligne[2] sous la forme d'une application WebGL fonctionnant sur navigateur. L'application fonctionne sur ordinateur et sur tablette. Avec ces 6 scénarios par défaut, cette ressource peut être utilisée telle quelle par les enseignants. Pour aider les enseignants à appréhender les compétences travaillées dans chacun des scénarios, une analyse automatique des niveaux composant le scénario sélectionné est présentée (voir figure 3). Actuellement trois référentiels sont intégrés au jeu (le PIAF (Parmentier *et al.*, 2020), le CRCN[3] et celui spécifique à SPY), l'enseignant peut ainsi avoir un aperçu des compétences travaillées dans chacun des scénarios (Muratet, 2023).

1 https://blockly.games/maze, accédé le 24/05/2023
2 https://spy.lip6.fr/, accédé le 24/05/2023
3 https://eduscol.education.fr/document/20389/download, accédé le 24/05/2023

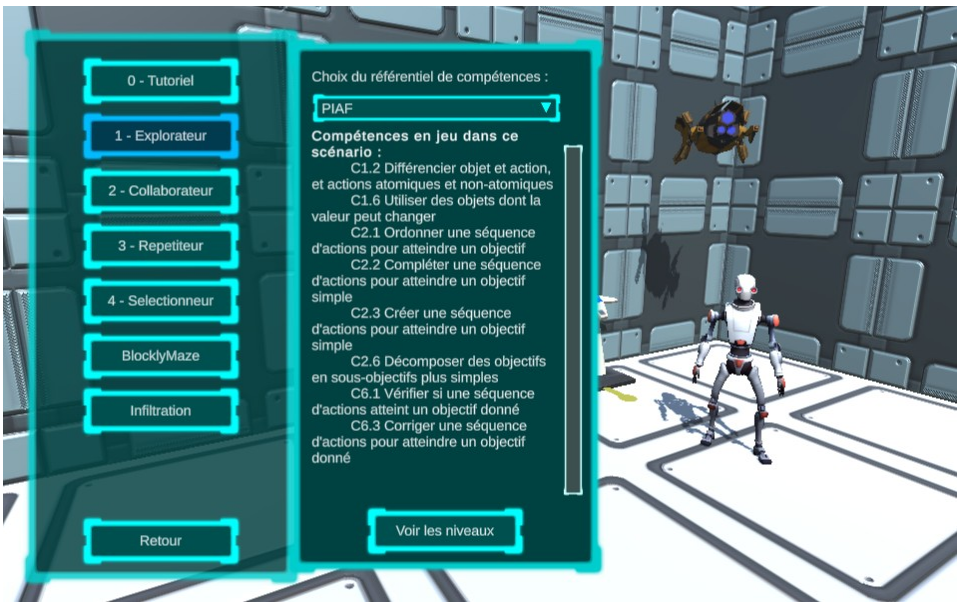

Figure 3: Analyse des compétences issues du référentiel PIAF dans le scénario « Explorateur »

L'éditeur de scénario (voir figure 4) inclus dans le jeu permet un premier niveau d'adaptation. Cet éditeur permet de filtrer la base de données des niveaux de SPY par compétences, de consulter chaque niveau en visualisant notamment les compétences en jeu et d'agencer les niveaux retenus en séquence. Un enseignant peut ainsi créer son propre scénario ou modifier un scénario existant en le (re)composant à partir des niveaux existants. La consigne de chaque niveau du scénario peut être redéfinie en vue d'adapter son contenu à un nouveau profil d'élève ou pour lier les niveaux les uns aux autres sous la forme d'une histoire.

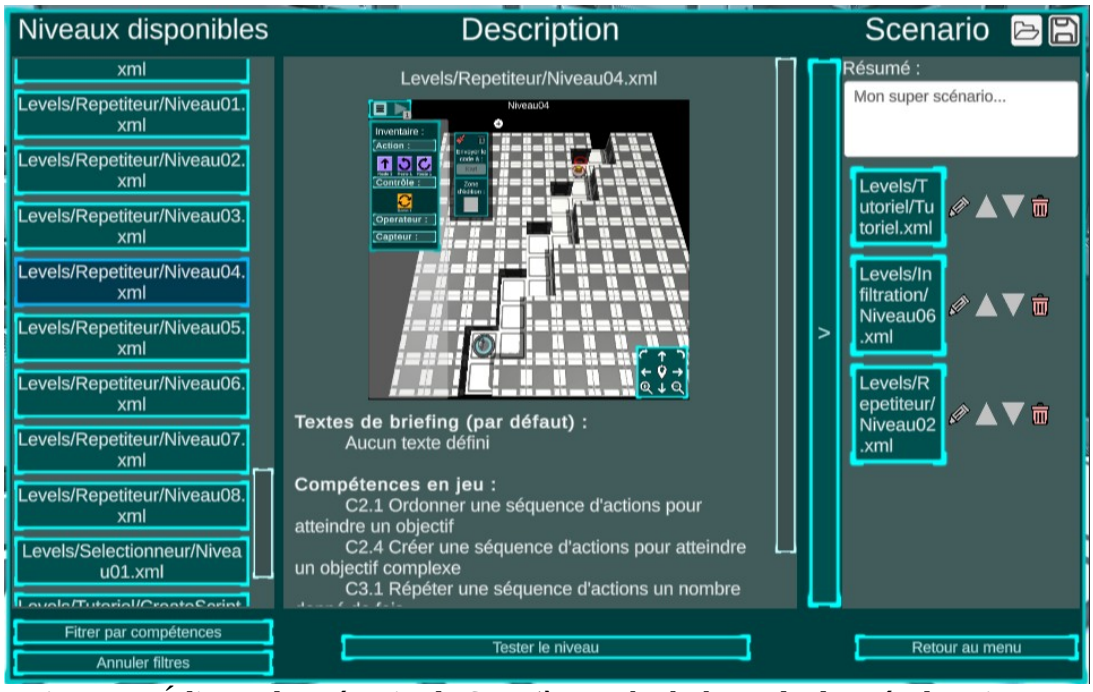

Figure 4: Éditeur de scénario de SPY (à gauche la base de donnée des niveaux du jeu, au centre le niveau actuellement sélectionné, à droite les niveaux constituant le scénario en cours d'édition)

SPY est aussi partagé en tant que projet open source[4]. Le jeu est développé par Sorbonne Université avec l'environnement Unity[5]. Nous ne rentrons pas ici sur les détails d'implémentation du jeu et nous présenterons simplement un résumé du modèle de donnée décrivant un niveau[6]. L'objet « map » décrit la topologie du niveau sous la forme d'une matrice. Les valeurs de la matrice codent la nature du sol (mur invisible, sol, mur, point de départ, point d'arrivée). L'objet « dialogs » décrit les informations affichées au chargement du niveau, il permet par exemple de présenter une consigne. Le contenu d'un dialogue peut être un texte, une image, un son, une vidéo ou une combinaison de tous ces éléments. L'objet « blockLimits » décrit les blocs disponibles dans l'inventaire du joueur, chaque type de bloc peut être présent en quantité illimitée, limitée ou non disponible. Ceci permet de réduire la complexité d'un niveau en ne proposant que les blocs utiles à sa résolution ou au contraire forcer l'utilisation de certains blocs. Les objets « player » et « enemy » permettent de positionner respectivement sur la carte des robots contrôlés par le joueur et des ennemis observant une partie de la carte. Chaque agent (robot et ennemi) écoute sur une ligne de communication. L'objet « script » permet de définir une zone de programme qui enverra son contenu sur une ligne de communication. Si un robot ou un ennemi écoute sur cette ligne alors il sera commandé par cette zone de programme. Une zone de programme peut être vide ou pré-remplie. D'autres objets peuvent être définis pour par exemple positionner des portes et des terminaux de contrôle, bloquer le *drag&drop*, limiter le nombre d'exécution... Bien que le jeu ne contienne pas pour l'instant d'éditeur de niveaux, il reste néanmoins possible de créer de nouveaux niveaux en éditant manuellement les fichiers XML.

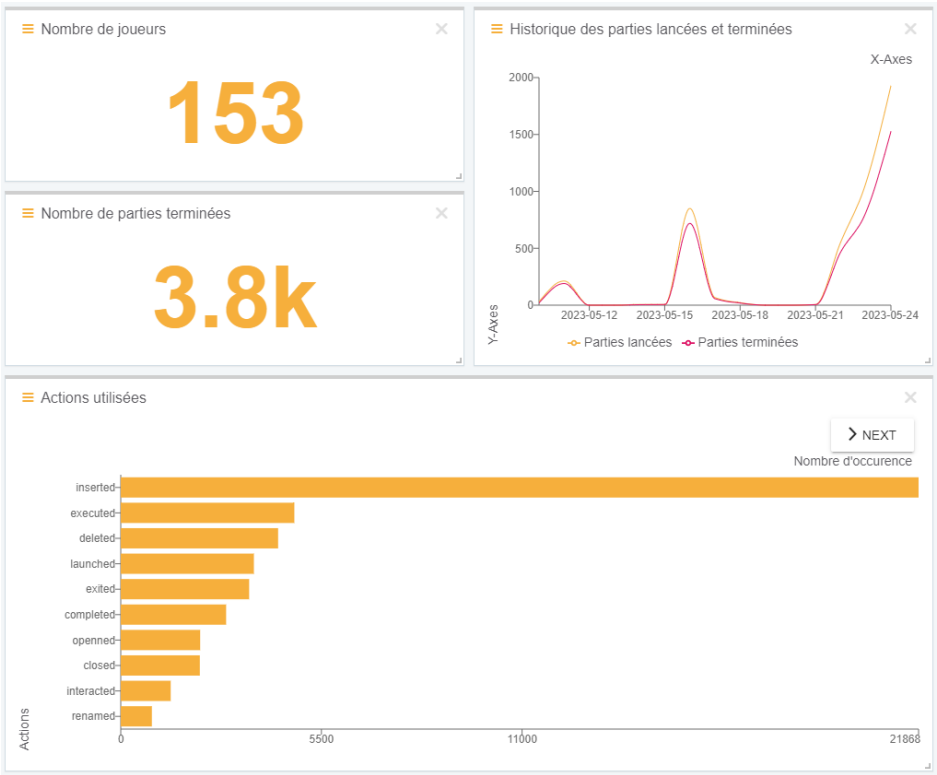

Figure 5: Exemple de visualisation générée par LearningLocker.

4 https://github.com/Mocahteam/SPY, accédé le 24/05/2023
5 https://unity.com/, accédé le 24/05/2023
6 https://github.com/Mocahteam/SPY/blob/master/Doc/LevelModel.xml, accédé le 24/05/2023

SPY est enfin partagé du point de vue des données d'interactions générées par le jeu. L'ensemble des interactions sont tracées au format xAPI[7] et stockées sur un LRS (Learning Record Store). Les données sont accessibles via un LearningLocker[8]. L'outil LearningLocker permet de générer des visualisations comme illustrées dans la figure 5. Chaque trace au format xAPI est appelé *statement* et se compose d'un triplet Acteur/Verbe/Objet (Qui ? Fait quoi ? Sur quoi ?). Chaque nouvelle connexion au jeu génère un identifiant de session anonyme unique que le joueur peut noter en vue de recharger sa progression ultérieurement. Cet identifiant de session définit l'acteur du *statement*. Les verbes et les objets permettent de caractériser le contenu des interactions. Les interactions actuellement tracées sont les suivantes : lancement du niveau, ouverture des dialogues, passage au dialogue suivant, passage au dialogue précédent, fermeture des dialogues, création d'une zone de programme, nettoyage d'une zone de programme, suppression d'une zone de programme, renommage d'une zone de programme, insertion d'un bloc, suppression d'un bloc, modification d'un bloc, exécution d'un programme, mise en pause de l'exécution, exécution pas à pas, arrêt de l'exécution d'un programme, fin du niveau, fermeture du niveau. Certains *statements* contiennent des extensions qui permettent de stoker des informations complémentaires comme par exemple les conditions de fin du niveau (succès/échec, score) ou le contexte dans lequel un bloc a été ajouté/supprimé. Actuellement seule la partie de SPY manipulée par l'élève est tracée mais nous envisageons de tracer également l'éditeur de scénario et s'il existe un jour, l'éditeur de niveau.

4 Conclusion

Dans cet article nous avons présenté le jeu SPY comme un EIAH partagé à la fois pour son usage, son code source et ses traces d'interaction. Nous avons décrit les scénarios actuellement présents dans le jeu comme une illustration des champs des possibles. Ces scénarios sont utilisables en l'état par des enseignants qui souhaiteraient faire travailler leurs élèves sur les compétences de la pensée informatique. Nous avons vu qu'il était possible d'augmenter le jeu en créant de nouveaux scénarios à partir de la base de données des niveaux existants et de créer/modifier des niveaux par l'édition de fichiers XML. Cette fonctionnalité est peu accessible mais, à défaut d'un éditeur, donne tout de même la possibilité de créer et de tester de nouvelles situations. Enfin nous proposons de partager aussi les données produites par le jeu au format xAPI afin de créer une base de données ouverte permettant aux chercheurs de la communauté de les analyser.

Nous terminerons cet article par un dernier partage, celui des questions de recherche. Nous esquissons ainsi quelques problématiques et perspectives de travaux qui pourraient être menés avec cette ressource. Comme nous l'avons présenté en introduction, un des enjeux dans ce champ de l'enseignement de la pensée informatique est d'outiller les enseignants (particulièrement de l'école primaire) avec des ressources qu'ils comprennent et qu'ils peuvent s'approprier. Quelles informations (compétences, conseils, recommandations) peuvent être extraites de la description des niveaux pour aider l'enseignant à s'approprier le jeu ? Comment les enseignants transforment-ils une telle ressource pour l'adapter à leurs élèves ? Que peut-on comprendre du processus de raisonnement des élèves à travers leurs traces d'interaction ? Comment produire des *feedbacks* à destination de l'élève pour l'aider lui et indirectement l'enseignant à dépasser les difficultés rencontrées ? ... Voici donc quelques questionnements qui, sans être exhaustifs, peuvent être abordés avec cette ressource partagée.

---

7 https://adlnet.gov/projects/xapi/, accédé le 24/05/2023

8 Toutes les informations permettant d'accéder aux traces du jeu sont disponibles à l'adresse suivante : https://spy.lip6.fr/openTraces.html, accédé le 24/05/2023

Références

Alayrangues, S., Peltier, S. and Signac, L. (2017). Informatique débranchée : construire sa pensée informatique sans ordinateur. *Colloque Mathématiques en Cycle 3 IREM de Poitiers*, IREM de Poitiers, Poitiers, France. pages 216-226.

Baron, G-L. and Drot-Delange, B. (2016). L'informatique comme objet d'enseignement à l'école primaire française ? Mise en perspective historique. *Revue française de pédagogie Recherches en éducation*, pages 51–62.

Komis, V. and Misirli, A. (2011). Robotique pédagogique et concepts préliminaires de la programmation à l'école maternelle : une étude de cas basée sur le jouet programmable BeeBot. *Didapro 4 – DidaSTIC*, Patras (Grèce).

Kradolfer, S., Dubois, S., Riedo, F., Mondada, F. and Fassa, F. (2014). A sociological contribution to understanding the use of robots in schools: the thymio robot. In *International Conference on Social Robotics*, Springer, Cham, pages 217–228.

Miljanovic, M.A. and Bradbury, J.S. (2018). A Review of Serious Games for Programming. In: *Joint International Conference on Serious Games*. Lecture Notes in Computer Science, volume 11243, pages 204-216. Springer, Cham.

Muratet, M. (2023). Comment caractériser et analyser les compétences de la pensée informatique d'un jeu sérieux ? *11ème Conférence sur les Environnements Informatiques pour l'Apprentissage Humain (EIAH 2023)*, 12 - 16 juin 2023, Brest, France.

Parmentier, Y., Reuter, R., Higuet, S., Kataja, L., Kreis, Y., Duflot-Kremer, M., Laduron, C., Meyers, C., Busana, G., Weinberger, A. and Denis, B. (2020). PIAF: developing computational and algorithmic thinking in fundamental education. In: *EdMedia+ Innovate Learning, Association for the Advancement of Computing in Education (AACE)*, pages 315–322.

Roche, M., de La Higuera, C. and Michaut, C. (2018). Enseigner la programmation informatique : comment réagissent les professeurs des écoles ? *Notes du CREN*, n°27, pages 1-8.

Saddoug, H., Rahimian, A., Marne, B., Muratet, M. and Sehaba, K. (2022). Review of the Adaptability of a Set of Learning Games Meant for Teaching Computational Thinking or Programming in France. *Special Session on Gamification on Computer Programming Learning*, Prague, Czech Republic. Pages 562-569.

Spach, M. (2019). Activités robotiques à l'école : approches de pratiques d'enseignement et effets sur les apprentissages. *Recherches en didactiques*, volume 28, pages 68-87.

Vahldick, A., Mendes, A.J. and Marcelino, M.J. (2014). A review of games designed to improve introductory computer programming competencies. *IEEE Frontiers in Education Conference (FIE) Proceedings*, Madrid, Spain, pages 1-7.

Wing, J.M. (2006). Computational thinking. *Communications of the ACM*, volume 49(3), pages 33–35.