Rapport projet ISG SPY

Jeux sérieux

Sorbonne Université

Master ANDROIDE

Auteurs :

- HADBI Imane 21201851

- HOUÉE Ronan 21308445

- LIN Zhengqing 21317948

Janvier 2025

Introduction :

Dans le cadre de l'unité ISG, le projet SPY se concentre sur l'amélioration d'un jeu sérieux éducatif en introduisant de nouveaux concepts et en enrichissant l'expérience d'apprentissage des élèves. Le jeu SPY a été conçu pour enseigner les principes fondamentaux de la pensée algorithmique.

Dans cette version améliorée, nous avons ajouté de nouveaux concepts pédagogiques essentiels, tels que l'introduction des variables et des fonctions. Ces ajouts visent à renforcer la compréhension des élèves en matière de programmation et d'algorithmique en les engageant dans des tâches pratiques qui développent leur logique et leur capacité de raisonnement.

D'autre part, nous avons également introduit un tableau de bord (dashboard) qui permet de suivre les performances des joueurs. Ce tableau de bord fournit des informations détaillées sur les progrès des élèves, leur expertise et leur efficacité dans la résolution des différents niveaux, tout en leur offrant une interface visuelle et interactive pour analyser leurs performances et les améliorer.

Objectifs :

Les objectifs du projet sont d'introduire des concepts avancés dans le jeu existant pour enrichir l'apprentissage des élèves et améliorer leur expérience avec la programmation. Parmi les concepts ajoutés, on trouve :

- **Les variables :** Permettre aux élèves de stocker et manipuler des informations dynamiques pour résoudre des problèmes de manière plus flexible et personnalisée.
- **Les fonctions :** Offrir la possibilité de regrouper des actions répétitives dans des blocs réutilisables, simplifiant ainsi la création de programmes et évitant la redondance dans les algorithmes.
- **Le tableau de bord :** Permettre le suivi des performances des joueurs à travers des graphiques interactifs. Ce tableau de bord aide à mesurer leur progression, à analyser leur expertise dans des domaines spécifiques (comme les boucles et les conditions), et à identifier les niveaux où ils peuvent encore progresser.

Partie 1 – Les variables

Responsable : HOUÉE Ronan

Fichiers concernés :

- XML : Variables, niveaux 1 à 4

- Components : Camouflages, BasicAction, BlockAdded, Colored, ColorShifted, ColorShiftedToFirst, ColorShiftedToValue, ColorShifter,

- Systems : BlockLimitationManager, ColorManager, CurrentActionExecutor, DragDropSystem, KeyManager, LevelGenerator

Résumé gameplay :

Pour cette partie, la campagne « Variables » a été ajoutée. Elle consiste en quatre niveaux, qui ont pour objectif d'introduire le concept de variable avec une difficulté graduelle et de manière ludique.

On définit une variable comme étant un objet stocké en mémoire, associé à une valeur qui peut être modifiée lors de l'exécution et lue pour être utilisée.

Pour cela, on va considérer la couleur du robot comme étant une variable. Cela permet d'avoir un changement visuel intéressant pour la démographie ciblée par SPY (enfants), tout en étant simple à appréhender et à voir évoluer.

L'introduction au concept de variable va donc se faire avec une difficulté graduelle, tout en essayant de démontrer d'utiliser des variables dans son programme :

Avant le quatrième niveau, le terme « variable » n'est pas utilisé ; on propose simplement à l'élève d'aider le robot à franchir des portes intelligentes grâce à un camouflage.

Dans un premier temps, l'élève doit récupérer des « Keys » (objets similaires aux Coins) pour changer de couleur.

Dans le **premier niveau**, deux blocs sont introduits :

Un bloc « Se scanner » qui permet de comparer la couleur du robot à celle des portes sur les cases adjacentes, et ouvre les portes de même couleur.

Un bloc « Camouflage » qui permet de prendre la couleur de la dernière Key ramassée.

L'idée est de présenter de la manière la plus simple ce concept, puis de l'enrichir dans un second temps.

Dans le **deuxième niveau**, l'élève doit récupérer deux keys, puis il atteint deux portes qui sont placées l'une après l'autre. On arrive dans une situation ou la méthode précédente ne fonctionne plus, on introduit donc une nouvelle mécanique :

Le robot peut maintenant stocker les camouflages récupérés (= une variable est stockée en mémoire) : quand il ramasse une Key d'une certaine couleur, un bloc de Camouflage vers cette couleur est créé, et il pourra l'utiliser plus tard. Cette méthode permet de surmonter la difficulté présentée par le placement des portes, mais comporte plusieurs inconvénients :

- Il faut avoir un type de bloc différent par couleur que prend le robot
- Mais surtout, cela signifie qu'il est impossible d'utiliser le bloc avant d'avoir ramassé la Key.

Pour cela, on va dans le **troisième niveau** démontrer l'utilité du second aspect des variables : la capacité de les modifier à souhait.

Au lieu d'avoir des Keys à ramasser qui créent des nouveaux blocs de camouflages, on ne dispose plus que de deux blocs : « Se scanner » pour ouvrir une porte, et « Changer de couleur ».

Quand ce bloc de changement de couleur est placé dans l'input line du Robot, il se dote d'un champ de texte où l'utilisateur pourra directement taper la couleur à prendre. Cela permet non seulement de ne plus avoir qu'un seul bloc pour tout type de couleur, mais surtout de ne pas avoir à ramasser la key pour voir son bloc camouflage apparaitre.

La couleur du robot devient ainsi une variable paramétrable simplement par le joueur, et qui peut être utilisée pour former un algorithme menant à la résolution du niveau.

Description technique :

Ajout de blocs au cours de l'exécution grâce au component blockAdded et modification du blockLimitationManager.

Objets avec une **couleur modifiable** dotés d'un component ColorShifter.

Emission de components **ColorShifted** pour indiquer au ColorManager qu'un ColorShifter vient de changer de couleur, et qu'il doit changer son apparence visuelle.

Les components **ColorShiftedToFirst** et **Camouflages** servent pour les niveaux 1 & 2.

Le système **KeyManager** gère les collisions avec les clefs, de la même manière que CoinManager.

Modification de **LevelGenerator, CurrentActionExecutor** et **DragDropSystem** pour prendre en compte les nouveaux éléments de gameplay / types de BasicAction définis

Pistes d'amélioration envisageables :

Ajout du mode daltonien

Ajouter de capteurs liés aux couleurs des objets

Ajout de types de variables différentes : portes à ouvrir avec un mot de passe (chiffres / lettres)

Ajout d'autres types d'interaction avec la variable de couleur (la porte s'ouvre si la couleur du robot est l'une de celle qui compose celle de la porte par exemple)

Partie 2 - les fonctions

Responsable : HADBI Imane

Fichiers concernés :

- XML : Fonctions, niveaux de 1 à 3
- Components : Function, FunctionInit
- Systems : BlockLimitationManager, CurrentActionManager, ScriptGenerator

Résumé

Cette section présente l'ajout d'une nouvelle fonctionnalité dans le système de programmation visuelle : l'introduction des fonctions. L'objectif principal est de permettre aux utilisateurs (élèves) de regrouper des actions répétitives dans des blocs de fonctions réutilisables. Cette approche offre une initiation aux concepts fondamentaux de la programmation structurée.

Les fonctions introduisent deux principaux éléments :

1. **FunctionInit** : Permet de définir une fonction avec un corps d'actions.
2. **FunctionCall** : Appelle une fonction préalablement définie pour exécuter son corps.

Description Technique

1. Définition et Exécution de Fonction

La définition d'une fonction repose sur l'élément FunctionInit, qui contient :

- **functionBody** : une référence au corps d'actions qui compose la fonction.
- **inExecution** : un drapeau booléen permettant de suivre l'état d'exécution de la fonction.

Le système vérifie si le corps de la fonction est vide :

- Si oui, il initialise le functionBody en ajoutant les actions à exécuter.
- Si non, il bascule entre deux états :
    - **Début d'exécution** : Lorsque la fonction est appelée pour la première fois.
    - **Fin d'exécution** : Une fois que toutes les actions de la fonction ont été exécutées, il retourne à l'endroit où la fonction a été appelée en utilisant une pile d'appels (functionCallStack).

2. Appel d'une Fonction

Les appels de fonctions utilisent l'élément FunctionCall. Lorsqu'une fonction est appelée, son emplacement est stocké dans la pile pour garantir que le programme reprend correctement là où il s'était arrêté après l'exécution de la fonction.

Concepts Pédagogiques

Niveau 1 : Introduction aux Fonctions

Dans ce niveau, l'élève apprend à regrouper des actions simples dans un bloc de fonction. Le terme « fonction » est introduit, et l'utilisateur est guidé pour créer un **FunctionInit** avec un ensemble d'actions comme **"tourner à gauche"**. Ensuite, un **FunctionCall** est utilisé pour appeler cette séquence répétitive.

Objectif : Montrer comment les fonctions réduisent la redondance et simplifient le script en remplaçant plusieurs actions par un seul appel.

Niveau 2 : Appels Multiples d'une Même Fonction

Au niveau 2, l'élève apprend à utiliser la même fonction plusieurs fois dans un script. L'objectif est de simplifier le code en utilisant plusieurs appels à une fonction préalablement définie. Cela permet à l'élève de comprendre l'intérêt de réutiliser une fonction dans différents contextes.

Objectif : Mettre en pratique l'idée de réutilisation des fonctions pour simplifier les programmes et éviter la duplication des actions.

Niveau 3 : Fonctions Complexes et Logiques Conditionnelles

Dans ce niveau, l'élève crée des fonctions plus complexes, souvent en combinant plusieurs actions et en ajoutant des conditions logiques. Par exemple, il pourrait être nécessaire de créer une fonction pour représenter un mouvement en forme de "U" horizontal en fonction des obstacles rencontrés.

Objectif : Permettre à l'élève de comprendre l'importance des fonctions pour organiser des actions complexes et utiliser des conditions dans un cadre de programmation structuré.

Pistes d'Amélioration

Noms de Fonctions

Actuellement, le système limite les utilisateurs à une seule fonction par niveau. Une amélioration possible serait de permettre la définition de plusieurs fonctions, chacune avec un nom unique. Les appels aux fonctions se feraient en utilisant leurs noms respectifs, facilitant la distinction entre différentes actions complexes.

Introduction des Paramètres

Un ajout futur pourrait permettre aux fonctions de recevoir des paramètres, augmentant ainsi leur flexibilité. Par exemple, un paramètre pourrait définir le nombre de pas à effectuer dans une séquence d'actions.

Partie 3 - Tableau de bord

Responsable : HADBI Imane

Le projet **Spy Dashboard** vise à fournir une plateforme interactive et visuelle pour suivre les performances des joueurs dans divers scénarios de jeu. Développé en utilisant **Dash** (Python), cet outil s'inscrit dans une initiative pédagogique ou ludique, destinée à analyser les comportements et progrès des joueurs tout en identifiant leurs points forts et faibles. Le tableau de bord offre une interface utilisateur intuitive, combinée à des graphiques détaillés et des fonctionnalités de personnalisation.

Cette partie met en lumière les fonctionnalités clés, les composantes techniques, ainsi que les points forts et les perspectives d'amélioration de ce projet.

1 : Fonctionnalités principales

1.1 Affichage des joueurs

- **Classement des joueurs :** Les joueurs sont affichés dans une liste classée en fonction de leur score total. Les trois premiers sont représentés par des icônes distinctives.
- **Interactivité :** Lorsqu'un joueur est sélectionné, ses scénarios et ses données spécifiques apparaissent dans la partie droite du tableau de bord.

1.2 Gestion des scénarios

Chaque joueur dispose d'une liste de scénarios joués ou non. -Les scénarios joués sont mis en évidence pour une meilleure lisibilité. Lorsqu'un scénario est sélectionné, plusieurs informations sont affichées :

- **Niveaux terminés :** Les niveaux déjà accomplis par le joueur dans le scénario sélectionné.
- **Temps passé :** Calculé comme la différence entre le lancement et la complétion d'un niveau (si plusieurs tentatives existent, seule celle avec le meilleur score est retenue).
- **Score obtenu :** Accompagné d'un système d'évaluation en étoiles pour chaque niveau (1 à 3 étoiles).

1.3 Visualisations interactives

- **Graphique d'expertise du joueur :** Un graphique montre le pourcentage d'expertise dans deux domaines clés :
    - **Boucles (Loops) :** Proportion des niveaux impliquant des boucles achevés par le joueur.
    - **Conditions (Conditions) :** Proportion des niveaux utilisant des conditions terminés.
- **Diagramme circulaire :** Représente le taux moyen d'accomplissement pour le scénario sélectionné. Ce taux est calculé comme la moyenne des niveaux terminés par les joueurs, divisée par le nombre total de niveaux du scénario.
- **Graphique des temps moyens :** Affiche une barre horizontale pour chaque niveau d'un scénario, représentant le temps moyen passé par tous les joueurs sur ce niveau.

2 : Structure technique

2.1 Technologies utilisées

Le projet s'appuie sur un ensemble de technologies modernes et robustes :

- **Dash :** Framework principal pour le développement de l'application web.
- **Pandas :** Pour l'analyse, la manipulation et l'agrégation des données issues des joueurs et des scénarios.
- **Plotly :** Génère des graphiques interactifs et attractifs.
- **HTML/CSS :** Personnalise l'apparence de l'interface et améliore l'expérience utilisateur.

2.2 Organisation du projet

1. **Chargement des données :** Les données des joueurs sont stockées dans des fichiers CSV telles que le temps de jeu, les scores, et les scénarios joués, qui sont traités et agrégés via des scripts Python. Un fichier modèle (model.csv) définit les caractéristiques des niveaux :
    - Présence de boucles ou de conditions (indiquées par des valeurs binaires : 1 ou 0).
    - Scores maximum et étoiles attribuées.
2. **Traitement des données :**
    - **Temps passé :** Une fonction calcule le temps passé sur chaque niveau en identifiant les événements "lancement" et "fin". La fonction 'calculate_time_spent' extrait le temps passé sur chaque niveau en croisant les événements "launched" (lancement du niveau) et "completed" (fin du niveau). Si un niveau est joué plusieurs fois, seule la tentative avec le meilleur score est considérée.
    - **Expertise :** Une fonction calcule le pourcentage d'expertise basé sur les niveaux impliquant des boucles et des conditions. La fonction 'calculate_expertise' mesure les compétences d'un joueur dans deux domaines : les boucles et les conditions. Elle compare les scores obtenus aux scores maximaux définis pour chaque niveau dans le fichier modèle.
    - **Classement :** Une fonction génère le classement des joueurs avec des icônes pour les trois premiers. La fonction 'ranking' génère un classement des joueurs basé sur leur score total. Les trois premiers joueurs reçoivent des icônes spécifiques pour les distinguer visuellement.
3. **Affichage dynamique :**
    - Les graphiques interactifs sont mis à jour dynamiquement grâce à des callbacks Dash, en fonction des sélections de l'utilisateur.
    - L'état sélectionné (joueur ou scénario) est géré par des composants Dash comme dcc.Store et les callbacks Python.

3 : Points clés et défis techniques

3.1 Points clés

- **Facilité d'utilisation :** Une interface fluide et réactive, adaptée aux utilisateurs novices comme experts.

- **Visualisations intuitives :** Les graphiques offrent une compréhension rapide et claire des données.
- **Personnalisation dynamique :** L'utilisateur peut explorer les données spécifiques à chaque joueur et scénario.

3.2 Défis techniques

- La manipulation de fichiers volumineux nécessite des optimisations pour réduire les temps de calcul et le temps de chargement.

Lien Git : https://github.com/Exteal/SPY

Conclusion :

Le projet SPY a démontré l'efficacité des jeux sérieux éducatifs pour l'apprentissage de l'algorithmique. Les concepts introduits, tels que les variables, les fonctions réutilisables et le tableau de bord interactif, ont renforcé l'engagement des utilisateurs tout en facilitant une compréhension progressive des notions fondamentales.

Ces avancées offrent un bon équilibre entre apprentissage et ludification, tout en mettant en lumière des axes d'amélioration, comme l'intégration de nouvelles fonctionnalités et l'optimisation des performances. Ce projet constitue une base solide pour développer des outils éducatifs innovants et adaptés aux besoins des élèves.