Pierre-Henri Ginoux
Mehdi Nait-Sidous

Projet RICM4
Serious Game

Table des matières

Le contexte et le domaine d'application ciblée.

Le contexte

Dans le cadre des projets innovants RICM4, nous avons travaillé sur un projet intitulé **Serious Game: Handicap, parole et geste**. Ce projet consiste à développer un logiciel utilisant la notion de jeu vidéo pour aider des enfants ayant des difficultés dans la maîtrise de la parole et où le geste est utilisé pour compenser ce déficit, et ainsi apprendre de nouveaux mots plus facilement. Un second objectif est de permettre le recueil des différentes actions effectuées par l'enfant au cours du jeu pour déterminer les progressions et les difficultés rencontrées par celui-ci afin de mieux comprendre les processus cognitifs mise en route et affiner les stratégies d'aide.

L'application

Notre tâche était de développer une application regroupant des petits jeux interactifs en 2D utilisant des éléments prédéfinis et fournis par les chercheuses comme des personnages, des habitations et des véhicules. On accède à cette application par un système de log in. L'ajout d'un nouveau jeu doit se faire de manière très simple. Les détails des différentes interactions doivent pouvoir être conservées par joueur dans une base de données pour les analyser ensuite ultérieurement.

Les outils

Pour ce projet, nous développons en Python avec avec l'aide du framework Kivy. C'est une librairie Python open source, pratique pour le développement en cycle court, qui permet de créer des interfaces graphiques. Le fait qu'il soit cross platform, sous licence MIT, et qu'il gère le multi-touch font de lui est un des principaux framework Python. Kivy a été pensé pour permettre du prototypage d'interface graphique aisément tout en produisant du code élégant. Avec Kivy, *« the code goes in main.py, the layout goes in controller.kv ».* Nous avons également utilisé Sqlite comme base de données portable. Les tables sont stockées dans un fichier en local pour pouvoir ensuite être uploadées.

Le cahier de charges

Nous avons rencontré les chercheuses à deux occasions. Durant ces entretiens, elles nous ont fait part des contraintes importantes qu'elles souhaitent voir intégrer au jeu.

Contraintes de l'application

- Sauvegarde locale des statistiques

Les chercheuses nous ont explicitement demandé de pouvoir enregistrer certaines interactions et données au cours du jeu (comme le nombre d'échec, le nombre de points...)

- Possibilité d'ajout de jeu facilement

Si un développeur doit se baser sur notre travail pour ajouter un nouveau jeu, il doit pouvoir le faire sans difficultés

- Facile d'utilisation, intuitif

L'enfant doit être capable de comprendre le fonctionnement de l'application et des jeux très rapidement

- Identification via login

Les données stockées dans la base de données doivent être propre à chaque enfant

- Base de données dynamique

Possibilité de rajouter ou d'enlever des nouveaux attributs à la base de données.

Contraintes de jeu

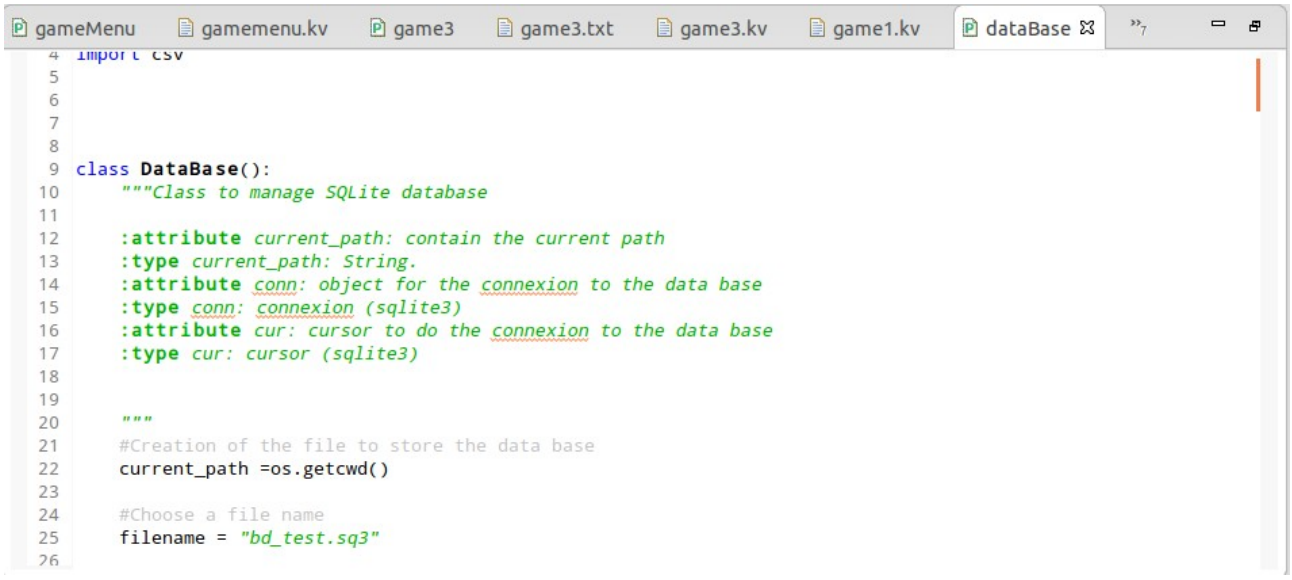
- Son

Possibilité de jouer des sons pour interagir et divertir l'enfant

- Gameplay intéressant

L'enfant doit vouloir jouer au jeu, cela doit être vécu comme un moment de détente et non une punition

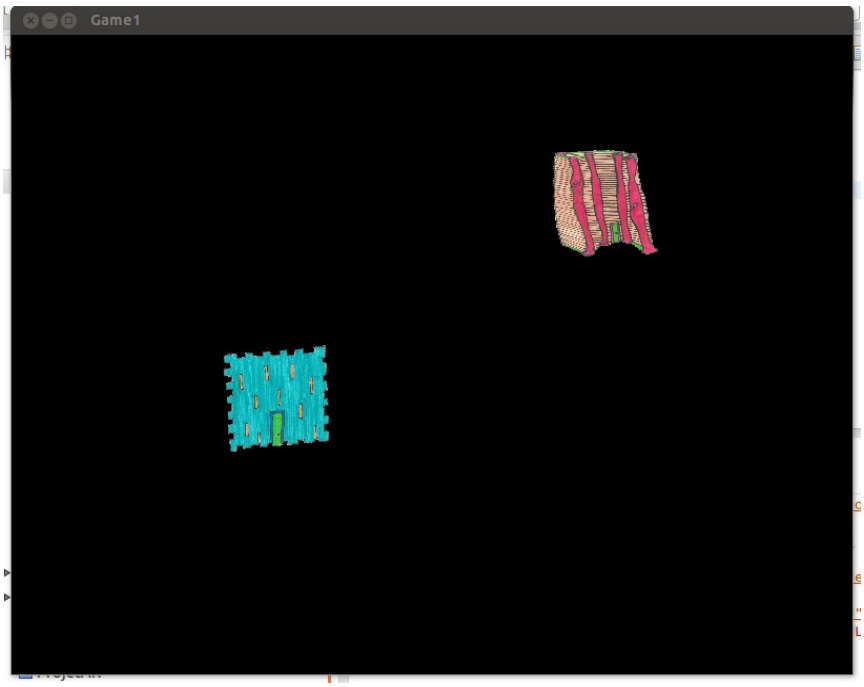
Nous avons également tenus à ce que notre code soit facilement réutilisable par d'autres développeurs, et facile à prendre en main. Nous avons par conséquent décidé d'appliquer une méthode rigoureuse pour commenter et documenter toute nos fonctions, comme vous pouvez voir par exemple sur l'image suivante :



Architectures successivement implémentées

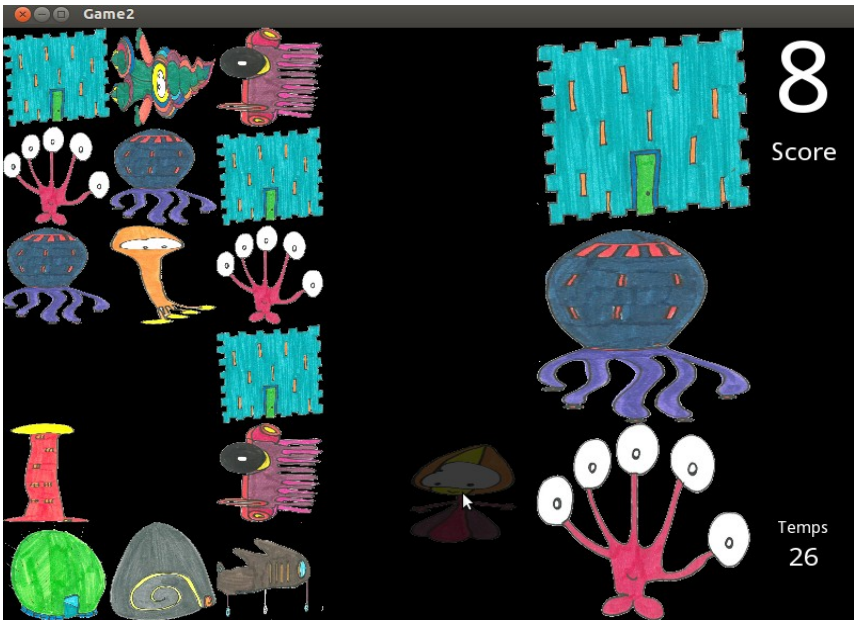
La première de ces deux rencontres nous a permis de développer une preuve de concept. Nous avons mis en place un grand nombre de mécanismes réutilisable pour un grand nombre de jeux.
La deuxième rencontre, quant à elle, nous a permis de les situer au niveau des possibilités qui s'offraient à elles pour le développement de leur jeu. C'est à ce moment-là que le cycle du développement du jeu « final » a été lancé.

Premier prototype



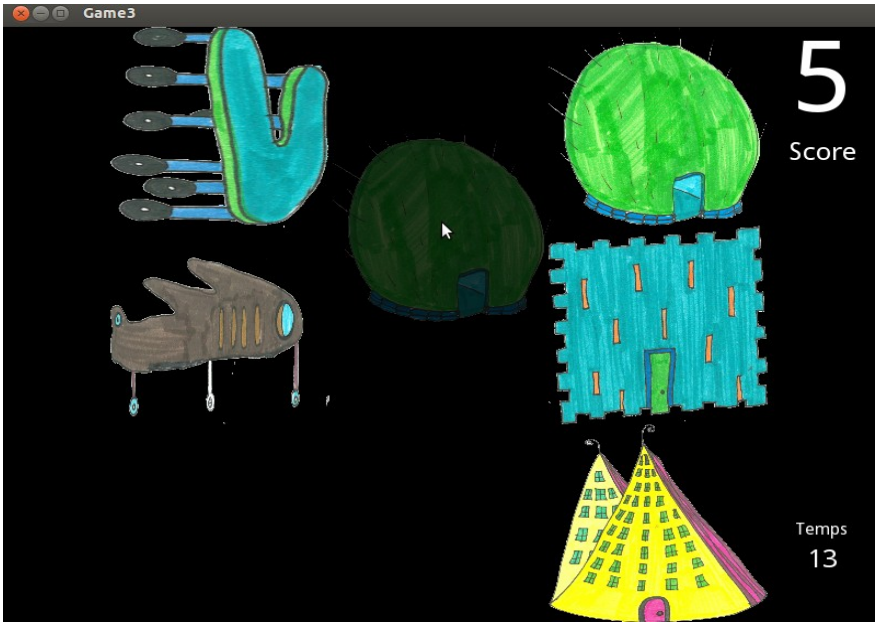
Grâce à Mr. Richard, nous avons eu l'idée très tôt dans ce projet d'implémenter un premier prototype le plus rapidement possible. Dû à notre manque d'expérience en Python, ce prototype a mis 3 semaines au lieu des 2 prévues pour voir le jour. En moins de 300 lignes, nous avons été capables de mettre en place les principaux mécanismes de jeu (login, drag & drop, collision simple), la gestion des sons, et la base de données. Cela peut paraître faible pour le temps passé dessus, mais ces 300 lignes ont constitué une base solide pour la suite de notre projet. Pour cette étape, nous nous sommes efforcés d'avoir le code le plus épuré et souple possible.

Premier jeu



Ensuite, nous avons débuté le développement du 1er jeu. Ici, l'objectif était de montrer aux chercheuses ce dont nous étions capables de faire en termes de gameplay. Le jeu développé était très simple : l'enfant devait reconnaître le type d'un objet (véhicule, habitation, ou personnage) et ensuite le faire glisser sur la catégorie correspondante, le score augmentait de 5 points pour une bonne réponse et diminuait de 1 pour une mauvaise. A cette étape, nous avons également ajouté la notion de score et de chronomètre qui vont nous être utile par la suite.

Application finale



Cette étape correspond à ce que les chercheuses attendent concrètement de l'issue de

notre projet. Nous avons mis en commun nos idées pour mettre en place notre mini jeu « final ». Le jeu produit, contrairement à celui de l'étape précédente doit pouvoir être utile à des enfants handicapés. Nous avons donc conclu que notre jeu se déroulerait en 2 étapes : une phase d'apprentissage, et une phase de vérification de l'apprentissage. La première phase consiste tout d'abord à faire associer à l'enfant un objet avec un autre élément contenant uniquement la forme de l'objet en question. Une vidéo est joué à chaque fois pour permettre à l'enfant d'apprendre le mot associé à cet élément. A la fin de cette première phase (quelques minutes), la phase de restitution des connaissances débute. Ici, l'enfant est « testé » sur les mots qu'il a appris dans la phase précédente.

Métriques

Nous avons utilisé pyMetrics pour mesurer les différentes métriques associés à notre code. Ici, on regarde en particulier le nombre de lignes de code et le nombre de lignes de commentaire.

	Nb ligne de code	Nb ligne de commentaire
dataBase.py	186	31
game1.py	75	0
game1.kv	29	1
game2.py	218	43
game2.kv	344	33
game3.py	426	70
game3.kv	33	1
gameMenu.py	36	1
gameMenu.kv	21	2
main.py	71	0
main.kv	71	3
TOTAL	1510	185

Problèmes rencontrés et les solutions élaborées

Ce projet a été l'occasion pour nous d'apprendre un nouveau langage ainsi que de découvrir un framework prometteur. Durant les premières semaines du projet, nous avons essayé de nous approprier ce langage et cette nouvelle philosophie dans la conception.

Nous avons souhaité coder de la manière la plus propre possible par rapport au standard Kivy. Seulement, il est très difficile de trouver du « beau » code Kivy sur Internet afin de prendre exemple. Sur GitHub par exemple, la très grande majorité des projets sont codés en s'écartant des bonnes pratiques préconisées par les développeurs de Kivy. (Séparation du code et du layout, un seul fichier .kv par screen...).

Une autre mauvaise surprise a été le fait que notre IDE (Eclipse) ne reconnaît pas les fichiers .kv, donc pas la coloration syntaxique, ni d'auto-complétion pour ces fichiers. Si le projet était plus long et plus conséquent, il aurait été plus judicieux de créer un plugin Eclipse pour faciliter le développement.

Kivy est un framework haut-niveau. Même s'il est possible de travailler sur des éléments plus bas niveau, les développeurs de Kivy conseillent de se contenter d'utiliser les classes fournies et documentés. Nous nous sommes retrouvés dans un cas où la méthode de collision de Widget fournies par Kivy n'était pas assez précise. Nous avons donc décidé de coder notre propre fonction de collision et cela n’a pas été facile.

Nous avons également beaucoup de problèmes à gérer les vidéos sur Kivy, le rendu varie beaucoup selon le matériel. Par exemple un bout de code qui permet la lecture fluide et correcte sur la machine de l’un d’entre nous provoque une erreur sur la machine de l’autre du type « don’t know how to handle video/x-h264 ».

La génération aléatoire pour le jeu 3 a aussi été assez compliquée à mettre en place. En apparence, cela peut paraître simple car il suffit de tirer 3 objects et 3 formes de manière aléatoires. En pratique, il faut que les 3 objets tirés soient différents les uns des autes, que les 3 formes tirées soient différentes les unes des autres, et qu'il y ai une et une seule combinaison possible entre les deux ensembles. Cela a nécessité beaucoup de temps pour arriver à un rendu irréporchable.

Conclusion

Il y a très peu d'applications Kivy dont il est possible de s'inspirer et nous espérons que notre application puisse à d’autres développeurs ayant rencontré des problèmes similaires aux nôtres.

Toutefois, ce projet a été très enrichissant pour nous, et nous espérons que l’application que nous avons développée sera utile au travail des chercheuses pour la suite.

Perspectives envisageables

Avec un peu plus de temps, nous aurions pu finir le second jeu correctement. Cela sera fait dans les plus brefs délais, mais pas avant la soutenance.

Du côté des points à améliorer, il aurait été intéressant d'utiliser les fonctionnalités proposées par les dernières versions de Kivy, notamment le Screen Manager pour optimiser notre application.

Nous avions aussi pensé à implémenter un autre mécanisme de collision visant à pouvoir faire glisser les objets sur les côtés lorsque ceux-ci rencontrent un obstacle. Ainsi, la manipulation des objets serait beaucoup plus agréable pour l'enfant.

Enfin, nous avons pensé ce projet comme une première étape, une découverte de ce qu'il était possible de faire avec Kivy. Ce projet pourra servir de base pour un stage, ou pour un autre projet s’inscrivant dans sa continuité.