Rapport de gestion du projet S9

Boximon Fighter - Jeu avec commandes par signes

Antoine BRALET / Guillaume DURET / Paul METEYER

29 janvier 2021

1 Introduction

Dans le cadre de notre dernière année d'études à CPE Lyon dans la majeure Imagerie, il nous a été demandé de réaliser un projet choisi par nous-mêmes. Le but de ce projet était de mettre en pratique toutes les connaissances acquises durant notre cursus à CPE Lyon (bien qu'il soit bien plus axé dans le domaine de notre majeure). Le projet a duré

Parlons maintenant du projet en lui-même. Ce projet est un jeu vidéo de type *Survival* qui consiste pour le joueur à résister aux nombreuses vagues d'ennemis (chaque vague a un nombre limité d'ennemis). Pour jouer au jeu, il faut utiliser sa main gauche qui contrôlera les déplacements du personnage (via le clavier) et sa main droite qui sera placée devant une caméra pour lancer les différents sorts et ainsi tuer le maximum d'ennemis.

Le lien du Github avec tout le code final est présent sur le lien : https://github.com/Guillaume0477/PROJET_Image_JV

2 Deroulé du projet

2.1 Organisation des tâches

Le projet était suffisamment dense pour séparer équitablement les tâches entre les différentes personne du groupe :

- Antoine Bralet a principalement travaillé sur la partie Python. En effet, il a réalisé la détection des signes avec l'extraction des paramètres d'entrée de la SVM, le calibrage des paramètres nécessaires à la segmentation de la main ainsi que son suivi au cours du temps (*Tracking*);
- Paul Meteyer a quant à lui travaillé sur la partie visuelle, c'est-à-dire sur la partie Unity. Il a mis en oeuvre le principe ainsi que le fonctionnement du jeu vidéo du personnage jusqu'à l'ennemi en passant par les différents menus de jeu;
- Guillaume Duret a lui principalement travaillé sur la partie de liaison entre Python et Unity. Ceci a permis d'assurer l'envoi des données de Python au jeu vidéo pour exécuter les actions dans le jeu vidéo. Il a également travaillé sur la segmentation de la main ainsi que sur la SVM et sa base de données (pour assurer le bon fonctionnement de la SVM). De plus il a réalisé les textures des différents sorts avec un shader graph.

La liste précédente n'est pas exhaustive : cette répartition n'a pas empêché chacun de discuter et travailler sur les parties des autres selon l'avancement et les priorités du projet.

2.2 Organisation logistique

Afin de mener à bien ce projet, nous avons exploité au maximum l'ensemble des créneaux nous étant alloués. Notons que, pour des raisons de conflits d'emploi du temps (Master ID3D), certaines séances ont dû être repoussées.

L'ensemble de ces séances se sont effectuées à distance à l'aide des outils de conférence proposés par *Microsoft Teams*. Ceci a ainsi permis à chacun de travailler de son côté tout en sollicitant l'aide des autres s'il avait besoin d'aide. De plus, *Microsoft Teams* nous servait aussi à gérer les états d'avancement, c'est-à-dire que nous avions un fichier *Word* nous permettant de notifier les avancements en cours de chacun pour ne pas se perdre.

Enfin, le projet nécessitait des sauvegardes régulières et un historique facilement retrouvable des différents codes Python et Unity mis en place, c'est pour cela qu'un répertoire *git* a été créé. Ce répertoire *git* a été subdivisé en plusieurs branches dont une pour chacun des membres pour pouvoir avancer sans risquer de mettre en péril le travail des autres.

2.3 Jalons suivis

Les jalons suivis lors du projet ont été revus au cours de celui-ci afin de pouvoir respecter la date limite de rendu et pouvoir obtenir un résultat final jouable. Ce nouveau jalonnage s'est alors organisé comme suit.

Jalon 1

Jeu vidéo : Gestion du joueur, création d'un ennemi simple (une sphère) et création d'un projectile envoyé à l'ennemi;

Calibrage : Calibration des paramètres de la main (couleur et taille de la boîte englobante);

Segmentation : Segmentation sur les canaux RBG. Suivi de la main sur la vidéo (*Tracking*);

Mise en place du lien entre Python et Unity.

Jalon 2

Jeu vidéo : Ajout d'un ennemi plus réaliste (Boximon) et gestion avancée des ennemis avec animations, prefabs et générateurs d'ennemis au cours du temps;

Segmentation de la main : Utilisation des canaux HSV et création d'une norme basée sur les 6 canaux. Nettoyage de l'image;

Extraction de paramètres : Paramètres classiques (Boîte englobante, centre de gravité, etc...) et exploration des données apportées par l'enveloppe convexe.

Jalon 3

Jeu vidéo : Implémentation des autres sorts (onde de choc, mine et bouclier), mise en place de menus, mise en place d'un didacticiel, améliorations graphiques (textures à partir d'un shader graph paramètrable, terrain, ...);

Extraction de paramètres : Projections sur les directions principales (PCA);

Classification : Mise en place de la SVM et des bases de données.

3 Gestion des problèmes vécus

Cette section vise à développer les différentes difficultés rencontrées ainsi que les méthodes mises en place afin de les surmonter. En effet, si internet a été d'une grande aide de part ses multitudes d'informations, nous avons également beaucoup puisé dans d'autres ressources.

3.1 Articles scientifiques

Une fois les différents paramètres "classiques" extraits et fournis à la SVM, il semblait clair que les paramètres utilisés n'étaient pas suffisants. Pour cette raison, nous nous sommes lancés dans une recherche bibliographique afin de voir comment il était possible de détecter les doigts de la main dans la littérature. Deux articles ont particulièrement attiré notre attention. En effet, dans [El Fiorenza et al., 2019; Xu et al., 2017], l'idée est de passer par le calcul de l'enveloppe convexe puis des défauts de convexité afin de remonter jusqu'au nombre de doigts présents dans la segmentation. Ceci nous a alors orienté afin de déterminer de nouveaux paramètres. Si les défauts de convexité ne se sont pas montrés pertinents car trop long à s'exécuter lors de nos expériences, l'enveloppe convexe s'est quant à elle montrée particulièrement intéressante et nous a fourni quatre paramètres supplémentaires.

De nouveau dans une logique de récupérer des paramètres pertinents, nous est venue l'idée de projeter les pixels de l'image sur les directions principales de la segmentation. Ces directions peuvent être extraites à partir d'une Analyse en Composante Principale (PCA) vu en quatrième année à CPE. Nous nous sommes alors penchés sur son implémentation dans la littérature que nous avons pu trouver dans [Rehman and Lee, 2018]. Ceci nous a alors bien conforté dans l'idée d'utiliser la PCA car cet article s'en sert pour recaler des images, donc bien d'étudier et comparer les directions principales de l'image (fait qui nous importait particulièrement ici).

3.2 Cours suivis

Suite à la lecture des articles scientifiques [El Fiorenza et al., 2019; Xu et al., 2017], il semblait important de trouver un moyen de calculer l'enveloppe convexe rapidement et efficacement. C'est alors que nous vint l'idée de mettre en place une marche de Jarvis évoquée en cours d'Animations à CPE Lyon. La rapidité de cette méthode nous a permis de conserver une implémentation temps réel de la détection de paramètres.

Nous avons également été amenés à apprendre des notions de *Machine Learning* et plus particulièrement de *Support Vector Machine* (SVM). Nous avons alors pu nous replonger dans ces cours afin de mettre en place correctement et proprement la SVM permettant de déterminer quel était le signe effectué par l'utilisateur à partir de paramètres extraits. Notons de plus que cette SVM était particulière car multi classes : plus de deux signes devaient être distingués, ce qui a donc nécessité l'utilisation de méthodes de type Un contre Tous (OVR) et Un contre Un (OVO), méthodes que nous avions étudiées et implémentées en TP de *Machine Learning*.

3.3 Réflexion collective

Connaissant les différents signes que nous souhaitions détecter lors de ce projet, nous nous sommes alors réunis afin de trouver quels sont les principaux paramètres qui permettraient de les discriminer le plus efficacement possible. Ainsi des idées telles que la *bounding box* ou encore le centre de gravité ont donc pu germer, mais plus intéressant encore : la distance signée de valeur absolue maximale au centre de gravité. Cette dernière nous paraissait en effet particulièrement pertinente afin de distinguer le pouce en bas du pouce en haut. Par la suite les recherches d'articles nous ont permis d'étoffer ces paramètres comme précisé en Section 3.1.

Nous avons compté le nombre d'ennemis tués dans un compteur affiché dans le Canvas mais ceci avait posé un certain nombre de problèmes. En effet, il faut tout d'abord se rappeler que notre jeu vidéo contient huit générateurs de monstres placés de manière stratégique sur le terrain afin que les ennemis puissent venir de tous les côtés. De ce fait, toutes les X secondes, les générateurs faisaient apparaître un ennemi et chaque ennemi avait son propre script et sa propre variable qui compte sa mort. Ainsi, à chaque nouvel ennemi, la variable qui comptait le nombre d'ennemis morts se réinitialisait à zéro et donc ne dépassait jamais la valeur 1. Pour pallier à ce problème, il suffisait juste de mettre la variable globale utilisée pour incrémenter le Canvas en paramètre *static*.

De plus, nous voulions réaliser le compteur de vagues et du nombre d'ennemis tués dans un seul et même script (celui qui gère l'IA de l'ennemi) mais ceci nous posait un problème. En effet, le fait que ce soit l'ennemi qui gère l'incrémentation des vagues ne nous permettait pas de mettre un "temps de pause" entre la fin d'une vague et le début de la vague suivante, ce qui rendait le jeu trop difficile et moins réaliste. Ainsi, en ayant séparé les codes dans différents scripts : le compteur de vagues a son propre script et même chose pour le compteur d'ennemis, il est possible d'avoir ce "temps de pause" entre les différentes vagues.

La gestion du bouclier possédait quelques difficultés. Il est possible de citer par exemple le fait que le joueur soit placé à l'intérieur de la sphère et il était donc nécessaire d'ajouter un script pour inverser les normales de la sphère et ainsi visualiser la texture du bouclier de l'intérieur. De plus, il était nécessaire de gérer la collision de celui-ci uniquement avec les monstres car si la collision est activée avec le joueur, celui-ci se retrouve sur la sphère et non à l'intérieur. Si la collision avec le terrain est activée, la sphère se retrouvait surélevée tout comme le joueur. En n'ayant que peu d'expérience avec Unity, nous avons effectué une discussion entre nous afin de réfléchir sur différentes méthodes pour pallier aux différents problèmes. La solution a donc été d'ajouter une hit box carrée autour du personnage au lieu de la hit box sphérique d'origine. Cette solution permet simplement d'empêcher les monstres de toucher le joueur sans le surélever et celui-ci suit naturellement le joueur sans l'ajout de quelconque script.

3.4 Réflexion sur des résultats imprévus

Une fois les projections sur la seconde direction principale de l'image extraite par la PCA effectuées, nous avons pu noter une baisse de la généralisation des résultats de la SVM. Ceci nous a alors poussé à rechercher

la cause de ce problème car les paramètres extraits nous semblaient particulièrement pertinents de prime abord. Il est alors apparu, après plusieurs tests et une recherche approfondie, que la méthode était bel et bien fonctionnelle. Néanmoins, en ne tenant compte que de la seconde composante principale, nous faisions l'hypothèse que la direction générée traversait la main horizontalement. Cette hypothèse n'était, en réalité, pas respectée sur toutes les images. En effet, il suffisait d'avoir la main ouverte et un espacement entre le pouce et l'auriculaire plus grand que celui entre le majeur et le poignet pour que la première direction principale soit celle qui nous intéresse. De cette façon, nous avons alors décidé de prendre en compte non pas uniquement la seconde mais également la première composante principale ce qui a fait bondir l'efficacité de notre SVM de 68% à 80%.

Un problème à été d'améliorer la base de données pour que la classification fonctionne aussi sur le mouvement intermédiaire entre les signes. En effet, il se trouve que la classification en temps réel différait de la classification des images, dans le sens où la main a un temps de passage entre deux signes qui peut complètement fausser la classification. La solution a donc été d'ajouter, au sein de la base de données, des images intermédiaires pour que la SVM sache reconnaître les signes intermédiaires. Cette solution a aussi l'avantage d'accélérer la reconnaissance des signes car la SVM commence à classifier un geste dès le début du mouvement.

De plus, l'envoi des données classifiées par la SVM au jeu à chaque frame serait trop rapide dans le sens où l'animation d'un sort ne serait pas finie que le jeu Unity recevrait déjà une nouvelle demande de sort. En effet, en effectuant un mouvement de main allant de la position initiale à la position finale de la boule de feu, cela envoyait une dizaine de signaux de boule de feu. La solution a donc été de réaliser une pile pour chaque signe et d'envoyer un signe dès qu'une pile est pleine puis de les réinitialiser puis recommencer. De plus, pour améliorer encore la stabilité sans impacter la manière de jouer, une solution a été de toujours devoir revenir à la position initiale avant de lancer un nouveau sort.

4 Améliorations envisagées

La segmentation de la main est réalisée manuellement (avec l'hypothèse du gant qui est utilisé pour l'extraction) et pourrait être améliorée en utilisant l'information des frames précédentes pour prédire la segmentation de la frame suivante comme décrit dans l'article [Coogan et al., 2006].

Il existe également des méthodes de segmentation de la main à l'aide de réseaux de neurones CNN qui peuvent segmenter la main même en utilisant un fond d'une couleur similaire à celle de la main comme décrit dans l'article [Urooj and Borji, 2018].

Il est encore possible de complètement remplacer la segmentation et la SVM par un autre réseau de neurones. Il est possible de citer comme exemple le problème de reconnaissance des lettres en langage des signes qui se trouve être un problème similaire au notre. La description de cette méthode qui utilise une *kinect* est dans l'article [Rioux-Maldague and Giguere, 2014].

Du point de vue de la gestion des signes, il était initialement prévu de gérer des signes dynamiques. En effet, en complément du lancement de sorts avec des signes statiques, l'idée est d'utiliser des signes dynamiques pour plus d'immersion dans le jeu. Par exemple, il est possible d'imaginer avancer la main rapidement pour simuler le lancement d'une boule de feu.

Du point de vue développement du jeu vidéo, les améliorations et l'ajout de fonctionnalités sont sans limites. Cependant, plusieurs ajouts rapides pourraient être intéressants :

- Pour le moment le joueur est uniquement une caméra à la première personne. Il faudrait avoir un personnage physique avec des animations qui correspondent aux différents sorts et laisser le choix à l'utilisateur de choisir entre la vue à la première personne et la vue à la troisième personne. Ceci permettrait de mieux pouvoir contrôler le personnage en fonction des ennemis autour de lui. À la première personne, il n'y a aucune information sur les monstres qui viennent derrière le personnage.

- Avec l'ajout des signes numérotés, il serait possible d'attribuer des caractéristiques aux sorts lancés telles que la glace, le feu et l'électricité. Il serait ainsi possible de créer des faiblesses et des résistances aux ennemis.

5 Recul sur le projet

5.1 Recul professionnel

Cette Section vise essentiellement à avoir un retour critique sur notre gestion de projet, tant sur les points positifs qui nous ont permis d'avancer rapidement et efficacement que sur les points négatifs qui nous ont mis un coup de frein.

Ces derniers étaient essentiellement d'ordre logistique. En effet, compte tenu des conditions sanitaires que nous connaissons, nous avons dû essentiellement travailler sur nos propres ordinateurs. Par malchance, nous avions chacun des opérateurs systèmes (OS) différents, et malgré des *dual boots* mis en place, ceux-ci ne permettaient pas de faire fonctionner le projet (problème de caméra, problème de librairies python,...). Cette différence d'OS a également induit un autre problème concernant les compatibilités lors de la récupération des codes Unity effectués par chacun depuis Github. En effet, l'ensemble du projet est sauvegardé sous Github, y compris les spécificités liées à l'OS, et non pas seulement les scripts modifiés. Afin de résoudre ce type de problèmes nous aurions dû mettre en place un fichier *gitignore* permettant de ne mettre que les fichiers communs dans Github.

Pour terminer sur les "difficultés", il semble important de noter que nous avons dû revoir nos attentes à la baisse car nous pensions avoir une semaine supplémentaire pour faire ce projet. En effet, l'emploi du temps de l'année sur e-campus était celui de l'année 2019-20. Nous aurions ainsi dû comparer les différents emplois du temps à notre disposition afin de ne pas perdre trop de temps sur certaines étapes. Effectivement, la segmentation nous a occupé très longtemps car nous cherchions à avoir des résultats les plus robustes possibles, temps que nous aurions certainement réduit afin de pouvoir mettre en place d'autres fonctionnalités.

Néanmoins, notre organisation et séparation des tâches ont été relativement efficace. Chacun pouvait travailler sur les tâches qui l'intéressait mais aussi sur lesquelles il se sentait à l'aide. De cette façon, nous avons pu rapidement mettre en place des algorithmes efficaces et avancer dans notre projet.

Finalement, le fait de se fixer des horaires précis - guidés par les créneaux réservés au projet - nous a permis de mettre en place une méthode de travail sérieuse et rigoureuse mais aussi d'être disponible les uns pour les autres lorsque le besoin se faisait sentir. De cette façon, le travail d'équipe et l'entraide ont été de mise lors de ce projet ce qui nous a permis de ne pas rester bloquer trop longtemps et de trouver rapidement des solutions comme évoqué en Section 3.

5.2 Recul personnel

Tout d'abord, le projet fut une très bonne idée pour mettre en pratique ce qui a été vu et appris durant tout notre cursus à l'école. En effet, les travaux pratiques vus à l'école étaient des briques que nous devions assembler pour construire la maison (la réalisation de ce projet).

Finalement, le projet nous a apporté beaucoup de choses positives et a permis d'approfondir toutes les connaissances vues succinctement dans les différents travaux pratiques. Par exemple, le moteur de jeu Unity a été vu très rapidement dans le module de Jeux Vidéos et ce travail a permis de bien comprendre comment le moteur de jeu fonctionne et comment faire un jeu vidéo très proprement. C'est exactement la même remarque concernant les méthodes de segmentation de la main (filtrage, dilatation, etc...) et les méthodes de Deep/Machine Learning pour entraîner la SVM à l'aide d'une base de données faite par nous-même.

6 Conclusion

En conclusion à ce projet, chacun a pu trouver des tâches à accomplir en lien avec ses attentes. Plusieurs notions ont pu être mises en place et nous ont permis de mettre en oeuvre nos compétences pluridisciplinaires acquises à CPE : segmentation, traitement d'image, *Machine Learning* et jeux vidéos notamment. Nous pensons en effet avoir réussi à mener à bien ce projet malgré les quelques difficultés mentionnées. En effet, le résultat final est jouable et chacun d'entre nous a pu apprendre, développer et apprécier les compétences nécessaires à l'accomplissement de ce projet.

References

T. Coogan, G. Awad, J. Han, and A. Sutherland. Real time hand gesture recognition including hand segmentation and tracking. In *International Symposium on Visual Computing*, pages 495–504. Springer, 2006.

C. El Fiorenza, S. Kumar Barik, A. Prajapati, and S. Mahesh. Hand gesture recognition using convexity defect. *International Journal of Innovative Technology and Exploring Engineering*, 9(1):1161–1165, Nov. 2019. doi: 10.35940/ijitee.a4489.119119. URL `https://doi.org/10.35940/ijitee.a4489.119119`.

H. Z. U. Rehman and S. Lee. Automatic image alignment using principal component analysis. *IEEE Access*, 6:72063–72072, 2018.

L. Rioux-Maldague and P. Giguere. Sign language fingerspelling classification from depth and color images using a deep belief network. In *2014 Canadian Conference on Computer and Robot Vision*, pages 92–97. IEEE, 2014.

A. Urooj and A. Borji. Analysis of hand segmentation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4710–4719, 2018.

Y. Xu, D.-W. Park, and G. Pok. Hand gesture recognition based on convex defect detection. *International Journal of Applied Engineering Research*, 12(18):7075–7079, 2017.