

Projet traitement d'image

Diamond Harvester

8 mai 2022

Étudiants : Izzo Valentino et Frossard Loïc

Professeur : Tièche François

Développement logiciel et multimédia

Niveau 3 - HE-arc

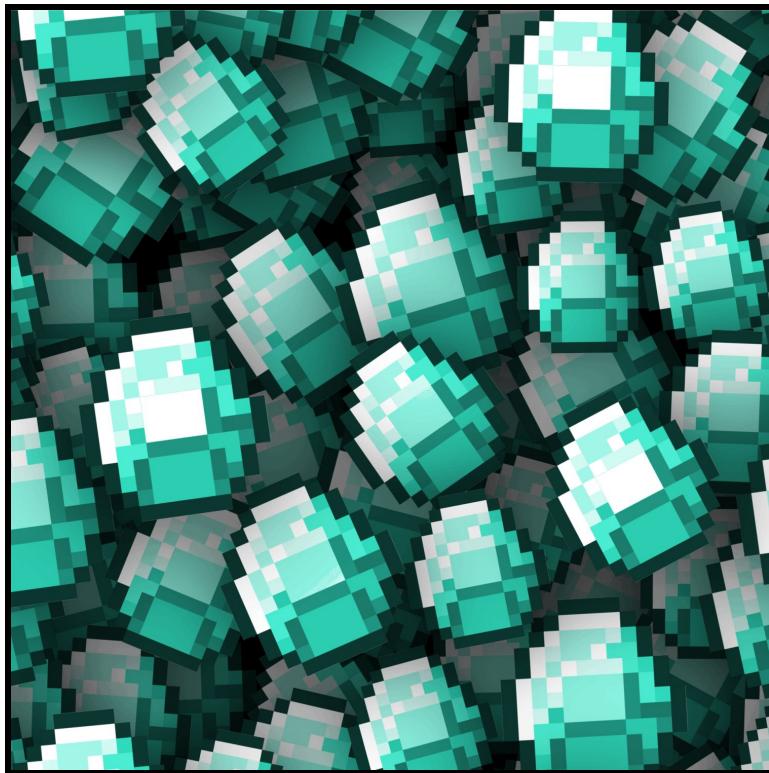


Table des matières

Introduction	3
Utilité du projet	3
GitHub	4
Analyse	4
Diamants et lave	4
BOT	5
Réalisation	5
Détection des blocs	5
BOT	6
Player	6
Game loop	7
Résultats obtenus	7
Détection des blocs	7
BOT	8
Vidéo de démonstration	9
Améliorations possibles	9
Conclusion	9
Bibliographie	9

1. Introduction

Dans le cadre du cours de traitement d'image, un projet doit être réalisé regroupant les connaissances apprises durant les deux semestres de 2022. Le nom du projet est Diamond Harvester. C'est un projet qui a pour but de reconnaître le minerai de diamant et la lave dans le célèbre jeu vidéo Minecraft. Une partie du projet est de rendre le déplacement du personnage dans le jeu de manière autonome, cela veut dire réaliser un programme qui déplace le personnage à l'aide du clavier et de la souris.

Le principe du projet est de démarrer l'application dans une mine et laisser le programme récolter du diamant et éviter de mourir en évitant la lave.

2. Utilité du projet

Dans le jeu vidéo Minecraft, il est nécessaire de trouver des minerais pour améliorer ses outils. Le diamant est l'un des meilleurs minerais, mais aussi le plus rare. Minecraft fonctionne avec un système de couche de blocs, par exemple la couche où le joueur apparaît pour la première fois dans le monde est entre 60 et 85. Dans les dernières versions du jeu, une statistique démontre que la couche -58 est la plus propice pour trouver du diamant. Ensuite pour rendre le minage optimisé, une astuce est de miné en ligne droite à la couche -58 pendant un certain nombre de blocs et revenir sur ses pas et minant avec un espace de deux blocs.

Sur l'image ci-dessous, voici un exemple d'un minage optimisé. En bleu, des minerais de diamants et en gris des blocs cassés.

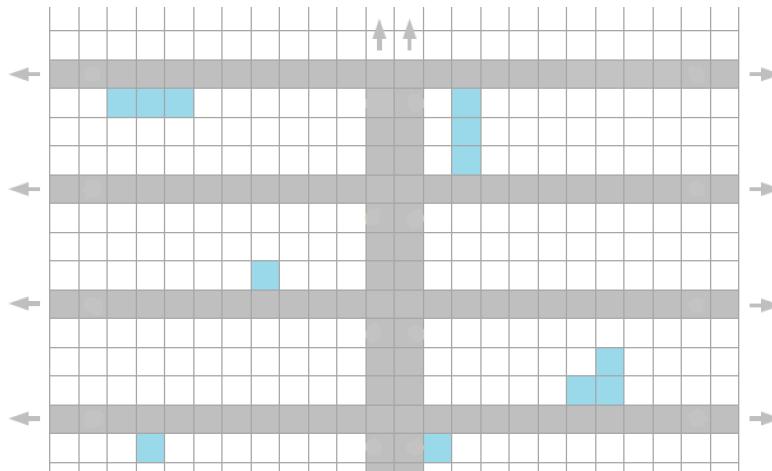


Fig. 1 schéma minage optimisé

Étant une tâche fastidieuse, le but du projet est d'automatiser le procédé.

3. GitHub

Pour ce projet l'utilisation de Github s'est limitée à la sauvegarde des fichiers. Pour la collaboration, Visual Studio Code propose de partager une session pour développer en même temps. Donc une seule branche "main" est utilisée sur GitHub.

4. Analyse

Dans ce chapitre se trouve, une explication des objets à reconnaître dans le jeu Minecraft et comment le bot doit se comporter en fonction de certaines situations.

4.1. Diamants et lame

Le bloc principal à reconnaître est le minerai de diamant. Voici ci-dessous une image qui montre ce bloc.

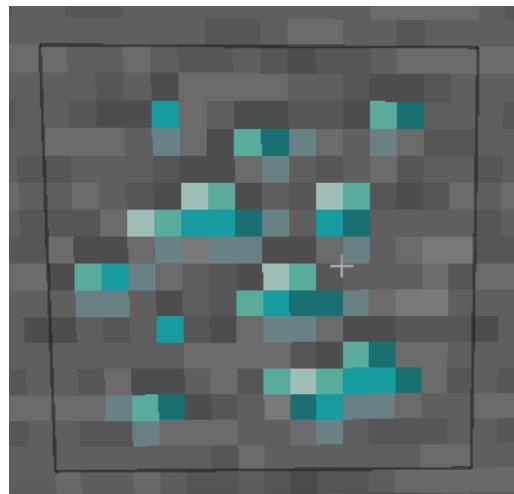


Fig.2 Un minerai de diamant

Le deuxième bloc à reconnaître est la lame, car à cette couche il est possible de trouver des lacs de laves et il est nécessaire d'éviter de tomber dedans sinon c'est la mort assurée.



Fig. 3 De la lave

4.2. BOT

Le bot doit pouvoir creuser en ligne droite sans s'arrêter et poser des torches sur un mur pour éclairer, sinon il fera trop sombre. Ensuite, au moment où un minerai de diamant est reconnu, il doit s'arrêter pointer son viseur sur le bloc de diamant et creuser. Après avoir récupéré le diamant, il continue son chemin et si de la lave est détectée, il recule dans son chemin et s'arrête pour éviter de tomber dedans et le programme s'arrête, car cela devient trop compliqué pour automatiser le passage de la lave.

5. Réalisation

Ce chapitre contient les explications de la réalisation de la détection des minerais et de la création du bot.

5.1. Détection des blocs

La détection des blocs est réalisée via une capture d'écran du jeu vidéo, ensuite l'image est transformée en couleur HSV. Il est nécessaire d'appliquer un masque sur l'image, car il est possible que dans le jeu on ait un diamant dans notre inventaire qui est en bas de l'écran. Cela peut poser problème à la détection du diamant, donc la barre d'inventaire est retirée de l'image avec le masque ci-dessous.



Fig 4. Le masque de l'inventaire

Ensuite une érosion suivie d'une dilatation sur l'image permet de faire ressortir les détails du minéral. Après la recherche des contours du minéral, cela permet de les afficher sur la vision ordinateur pour visualiser le bon fonctionnement de la détection.

Pour la lave c'est la même chose, cependant aucun masque n'est fait et une simple érosion est effectuée.

5.2. BOT

Le principe du bot est de réaliser des entrées clavier et des déplacements de souris pour se déplacer dans le jeu. Pour ce faire, la bibliothèque Python "pydirectinput" permet de réaliser certaines fonctions, dont voici celles utilisées :

- moveRel(x, y) : Déplacement de la souris depuis son état initial.
- keyDown(name of the key) : Reste appuyé sur la touche donnée en paramètre.
- keyUp(name of the key) : Relâche la touche donnée en paramètre.

5.2.1. Player

Une classe "Player" a été créée, elle contient tout le comportement du bot. Des variables permettent de donner le statut du bot qui est "isMoving" un booléen pour l'état du déplacement et "isMining" un booléen pour l'état du minage. Une dernière variable qui permet d'ajuster la fréquence où le bot pose une torche. Ensuite voici les fonctions que contient la classe "Player" qui sont appelées par la "game loop" expliquer au prochain chapitre.

La fonction "place_torch" va réduire de 1 la variable qui stocke la fréquence de posage de torche. Quand celle-ci arrive à 0 voici les étapes effectuées par le bot :

1. Arrêt du déplacement et du minage du bot.
2. Temps d'arrêt de une seconde pour attendre la fin de l'inertie du personnage.
3. Déplacement de la souris sur la droite.
4. Posage d'une torche avec le clic droit de la souris.
5. Déplacement de la souris pour revenir au centre de la tranchée.
6. Activation des variables d'états pour que le bot continue de creuser la tranchée.

La fonction "mining" appuie sur la touche pour creuser à l'aide de la fonction "keyDown"

La fonction "move" appuie sur la touche pour avancer à l'aide de la fonction "keyDown".

La fonction "focusOnDiamond" réalise le déplacement de la souris sur la position donnée du diamant à l'aide de "moveRel".

La fonction "step_back" est appelée quand de la lave est détectée, alors le bot doit s'arrêter et reculer de quelques secondes pour être assez loin de la lave.

La fonction "stop" arrête le bot de se déplacer et de miner.

5.2.2. Game loop

Le "main" de l'application est la liaison entre le bot et la détection des blocs. Le programme est une boucle qui en continu prend une capture d'écran du jeu, dans le cas où un diamant est détecté, une calibration est effectuée où à la suite 5 captures d'écran sont effectuées et à chaque fois la position du diamant est donné au bot qui se déplace le plus proche du diamant. Cette calibration permet d'être sûr que le bot vise correctement le minerai. Après le bot mine le diamant et la boucle continue. La boucle contrôle la présence de lave, si c'est le cas il signale au bot de reculer. Si rien n'est détecté alors la boucle dit au bot d'avancer et creuser tout droit.

6. Résultats obtenus

6.1. Détection des blocs

La détection du minerai de diamant fonctionne, via la vision par ordinateur, il y a bien autour du rassemblement de blocs ci-dessous un contour bleu et à son centre un point

blanc. La position du point blanc est donnée au bot qui lui déplace sa souris et ensuite creuse le bloc.

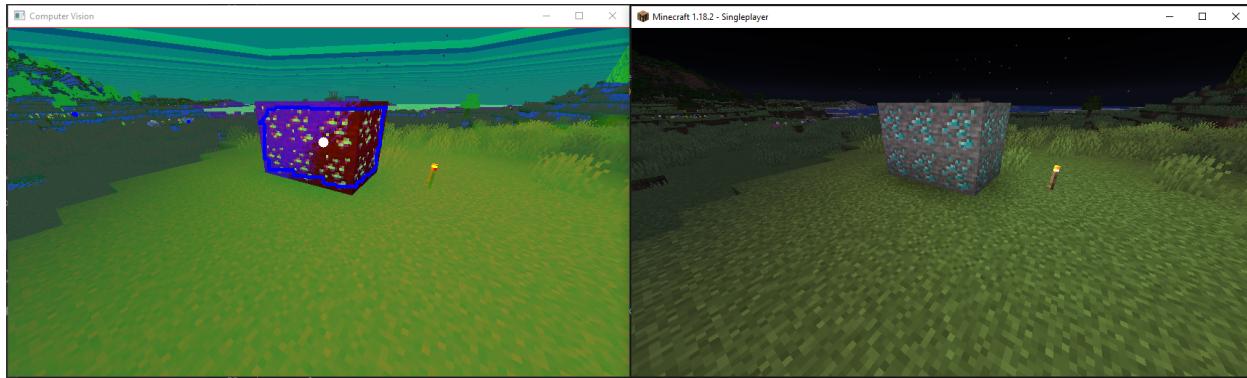


Fig 5. Vision ordinateur VS vision du jeu (diamants)

Pour la lave, voici une image montrant le résultat de la détection de la lave. Il y a bien le contour bleu qui montre la grandeur du lac de lave et le point bleu à son centre.

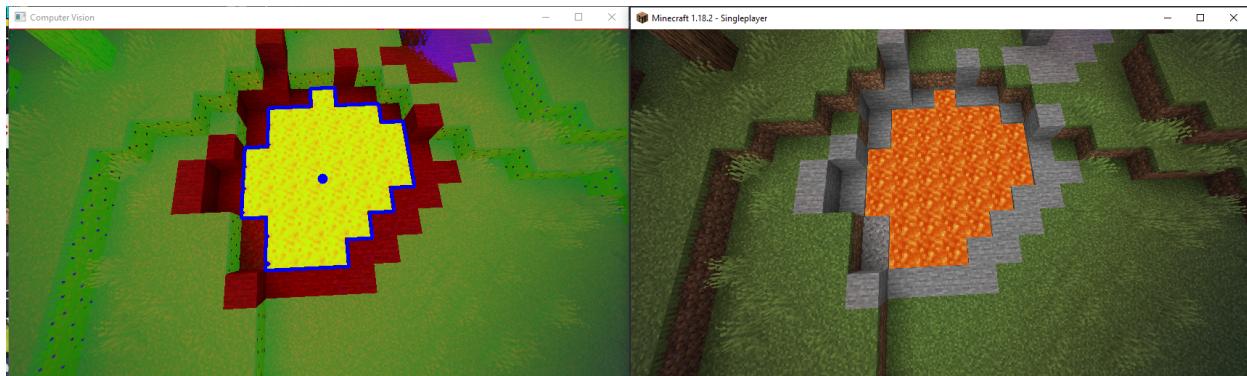


Fig 6. Vision ordinateur VS vision du jeu (lave)

6.2. BOT

Le bot est capable de creuser sans s'arrêter une tranchée de la mine optimisée. Il ajoute des torches pour pouvoir augmenter la luminosité et faciliter le traitement de l'image. Il peut aussi creuser un diamant.

Il y a un problème c'est qu'après le minage d'un minerai de diamant, le bot n'est pas capable de remettre la souris devant la tranchée pour continuer d'avancer. Cependant une solution possible est de sauvegarder tous les déplacements de la souris effectués par le bot au début de la détection d'un diamant et ensuite refaire les mêmes mouvements, mais depuis la fin du minage du diamant et théoriquement le bot devrait se trouver devant sa tranchée pour continuer de creuser et trouver des diamants.

7. Vidéo de démonstration

Dans les documents du projet, une vidéo montrant les résultats obtenus est disponible. Il est nécessaire d'avoir le jeu pour tester en direct le projet et le jeu étant payant, c'est pour cela que le choix d'une vidéo à été retenu.

Dans la vidéo les seuls moments où nous réalisons une action, c'est quand le bot a miné un diamant, nous remettons le curseur de la souris au centre de notre tranchée, car le bot n'est pas capable de le faire.

8. Améliorations possibles

Voici quelques améliorations et ajouts possibles pour le bot et la détection d'images

- Détection du vide : Il est possible que à la couche -58, qu'il y ait des grottes naturelles et si le bot creuse au-dessus de celle-ci, il faut éviter qu'il tombe dedans, car c'est la mort assurée.
- Détection d'autres minéraux qui se trouve à la même couche.
- Le bot remet le curseur tout seul au centre de la tranchée pour continuer automatiquement la recherche de diamants.

9. Conclusion

En conclusion de ce projet du cours de traitement d'image, nous sommes arrivés à un bon résultat pour la partie traitement d'image. Cependant, du côté du bot, cela s'est avéré plus compliqué que prévu et le résultat final n'est pas à la hauteur de nos attentes.

10. Bibliographie

[1] Schéma du minage optimisé, Computer craft,

<https://www.computercraft.info/forums2/index.php?/topic/19556-advanced-branch-mining-turtle-with-self-setup/>

[2] Code de la capture de fenêtre, Ben Johnson, GitHub

https://github.com/learncodebygaming/opencv_tutorials/blob/master/004_window_capture/windowcapture.py