LES TROPHÉES NSI

ÉDITION 2024

DOSSIER DE CANDIDATURE

PRÉSENTATION DU PROJET

NOM DE VOTRE PROJET :	PYCRAFT
MEMBRES DE L'ÉQUIPE :	ARTHUR SIRVENT
MEMBRES DE L'ÉQUIPE :	OSCAR VALAYÉ
MEMBRES DE L'ÉQUIPE :	STANLEY RAMANANTSOA
NIVEAU D'ÉTUDE :	PREMIÈRE
ÉTABLISSEMENT SCOLAIRE :	LYCÉE FREDERIC BAZILLE AGROPOLIS MONTPELLIER
ENSEIGNANTE/ENSEIGNANT DE NSI :	M. BRINGUIER

RECRÉER MINECRAFT EN PYTHON ? NOUS, ARTHUR, OSCAR ET STANLEY L'AVONS FAIT !
TOUT PART DE L'AFFICHAGE À L'ÉCRAN, AVEC DES MILLIONS DE BLOCS AFFICHÉS CHAQUE SECONDE, AVEC TEXTURES ET OMBRES.
ON PEUT SE DÉPLACER ET MÊME VOLER DANS CE MONDE !
ET ENFIN LE TERRAIN : UNE GÉNÉRATION PROCÉDURALE INFINIE, AVEC DES MONTAGNES, DES PLAINES, DES DÉSERTS, DES ARBRES, MAIS AUSSI DES ÎLES FLOTTANTES, ET MÊME DES VOLCANS !
ET BIEN SÛR, ON PEUT PLACER ET CASSER DES BLOCS, POUR CRÉER DES STRUCTURES UNIQUES !

> PRÉSENTATION GÉNÉRALE :

Pouvez-vous présenter en quelques mots votre projet ?
Comment est né ce projet ? Quelle était la problématique de départ ?
Quels sont les objectifs ? À quels besoins répondez-vous ?

Tout a commencé début septembre, quand notre professeur de NSI nous a parlé des trophées NSI. Nous avons vu en ce concours une occasion de nous challenger à travers ce défi (en effet, nous codions depuis plusieurs années et maitrisions Python déjà très bien, nous avons donc pu commencer dès septembre).

L'idée est alors venue de faire Minecraft en Python. Mais de le recréer vraiment, sans aucune aide, en partant de zéro, avec aussi peu de librairies annexes que possible. C'était vraiment ~~fou~~ ambitieux.

Nous, 3 lycéens de première voulions recréer Minecraft en Python et en 8 mois. Le fait que ce soit en python est un aspect assez important, principalement à cause des performances qui seront le nerf de la guerre pendant cette aventure.

Nous voulions donc faire un monde 3D généré procéduralement de manière infinie, dans lequel on pourrait se déplacer, placer et casser des blocs.

C'était notre objectif, et ce qui est bien est que peu importe notre avancée, il est toujours possible de faire plus, d'implémenter de nouvelles fonctionnalités, de nouvelles mécaniques. Mais nous n'allions pas simplement nous contenter de recopier Minecraft. Bien sûr, nous allions faire du relief, des plaines, des montagnes, des déserts, des arbres, des villages (qui se révélèrent assez techniques à implémenter) mais aussi des îles flottantes et même des volcans !

Il y a donc trois principales parties à ce projet : en premier, il y a la partie de génération de terrain, qui s'occupe de générer tout le terrain, c'est à dire le relief, les biomes, les îles flottantes, les arbres (normaux et palmiers), les océans, mais aussi les structures, à savoir les villages et volcans.

L'autre est la partie d'affichage des blocs générés. Cette partie s'occupe de prendre des blocs, et d'afficher individuellement chaque triangle qui les composent, avec des textures et ombres. Et ça fait ultimement des centaines de milliers de blocs affichés 120 fois par seconde, c'est dire du besoin d'optimisation et d'efficacité de cette partie.

La troisième partie est assez simplement la mise en commun des deux premières, en plus de toutes les mécaniques telles que le cassage et placement de blocs, ainsi que la physique et le déplacement du joueur. Cette partie a été particulièrement laborieuse par la nécessité de continuellement devoir mettre à jour parfois la génération de terrain, parfois l'affichage, bien sûr sans être épargné par les bugs, ce qui a fait que ces "merges" pouvaient quelquefois durer plusieurs semaines.

> ORGANISATION DU TRAVAIL :

Pouvez-vous présenter chaque membre de l'équipe et préciser son rôle dans ce projet ?
Comment avez-vous réparti les tâches et pourquoi ?
Combien de temps avez-vous passé sur le projet ? Avez-vous travaillé en dehors de l'établissement scolaire ?
Quels sont les outils et/ou les logiciels utilisés pour la communication et le partage du code ?
Vous veillerez au bon équilibre des différentes tâches dans le groupe. Chaque membre de l'équipe doit impérativement réaliser un aspect technique du projet (hors design, gestion de projet).

Voici la répartition des tâches :
- Arthur : Génération de Terrain
- Oscar : Unité centrale, lien entre la génération (Arthur) et le rendering (Stanley) + joueur (et la physique) et caméra
- Stanley : Rendering + GUI (affichage de cubes, textures, shaders, barre d'inventaire), côté GPU

Nous sommes tous dans la même classe.
Au moment de répartir les tâches, Arthur était très intéressé par la génération de terrain, et a donc pris le commandement de cette opération périlleuse dont il deviendra expert.

Au début, Stanley et Oscar travaillaient eux deux sur l'affichage (principalement) de blocs. Oscar était lui de son côté quand même plus intéressé par la partie mathématique du projet.

Mais très vite, ils se rendirent compte que les faibles performances de pygame/tkinter allaient nécessiter l'utilisation d'autres méthodes.

À ce moment, ils se sont divisés. Stanley aimait déjà bien tout ce qui était GPU et graphiques, il allait donc se concentrer sur le rendering("affichage") jusqu'à la fin du projet.
Oscar restait sur le cœur du projet, au début restant avec Stanley pour l'implémentation de l'affichage.

En effet, au début au moins, le travail était clairement coupé en deux. D'un côté, Arthur avait la génération de terrain et devait tester avec un sombre module annexe, car on ne pouvait pas se permettre d'attendre que le rendering de Stanley et Oscar marche (ce qui allait prendre du temps). De l'autre côté, Stanley et Oscar se débattaient avec le rendering en cherchant toujours plus d'optimisations pour être finalement capable d'afficher des millions de blocs par seconde.

Nous avons passé vraiment beaucoup trop de temps sur ce projet. En termes de lignes de codes, nous devons être à environ 2700 lignes pour Arthur, 1700 lignes pour Oscar et 1000 pour Stanley (échelle de conversion : 30 lignes -> 1h d'interaction sociale ~~perdue~~ employée à d'autres fins)

Mais ces nombres ne représentent pas tout le travail en amont de recherche, d'optimisations, de réécriture de code, de changement de méthode, ou même simplement de tests de fonctionnement. Au total, nous devons être à peu près à 200/250 heures chacun en moyenne, soit environ 600h pour arriver à ce résultat.

Un autre grand aspect qui nous a coûté beaucoup de temps est celui de la documentation. En effet, nous ne voulions pas nous contenter de simplement dire qui avait fait quoi et dans quel fichier, mais nous voulions faire en sorte que n'importe quelle personne avec une bonne maîtrise de python puisse, avec notre code et notre documentation, comprendre pourquoi et comment chaque choix avait été fait, et comment tout notre jeu fonctionne.

Nous avons donc évidemment dû travailler hors du lycée pour parvenir à ce résultat, même si le temps ensemble au lycée nous permettait de partager nos évolutions et surtout, vers la fin, de combiner régulièrement nos codes ensemble.

Nous avons utilisé GitHub pour le partage du code, principalement avec 1 branche main et *1 branche chacun en théorie (Arthur avait 4 branches, tellement il se gérait mal)*, même si ce nombre a évolué en fonction des combinaisons et des versions du code différentes.

LES ÉTAPES DU PROJET :

Présenter les différentes étapes du projet (de l'idée jusqu'à la finalisation du projet)

Etapes principales :
12/09 : Début du projet
14/09 : début du Renderer V0 (tkinter)
17/09 : Génération de terrain basique (Arthur)
19/09 : renderer V0 version stable (trop lent) --> Oscar et Stanley cherchent des optimisations
21/09 : début des expérimentations avec le module cython, Stanley se renseigne sur Opengl
23/09 : Arbres, terre, sable, neige, océans (Arthur)
11/10 : Iles flottantes (+ grottes, retirées car trop lentes) (Arthur)
11/10 : Début d'openGl avec Oscar et Stanley: création des vertex et fragment shaders
28/10 --> 15/12: Travail sur la physique et le stockage de blocs dans le jeu, entité du joueur (Oscar)
12/11 : Volcans, version stable (Arthur)
16/12 : Villages, version stable (Arthur)
21/12 : Batching (renderer V2, fonctionnel) (Stanley)
27/12 : Génération de terrain dynamique (générée en real-time) (Arthur+Oscar)
01/01 à 00:01: 1ere VERSION STABLE – 1ere fois que tous nos codes sont combinés et marchent ensemble (Oscar+Stanley+Arthur)
20/02 : Amélioration génération real-time grâce au multiprocessing et aux data manager (Oscar)
02/02 : Loading de mondes possible (Arthur+Oscar)
15/02 : Possibilité de casser/placer des blocs (Arthur + Oscar)
22/02 : Changement d'algorithme de génération des structures (Arthur)
29/02 : Prégénération des structures (Arthur+Oscar)
03/03 : Opacité, eau transparente (Stanley) (pas encore implémentée)
07/03 : Shaders (Stanley)
02/03 - 20/03 : Refactor du code, ajout des commentaires, écriture de la documentation
21/03 : Combinaison de tous les documents nécessaires, partage sur GitLab
25/03 : Rendu au professeur de NSI

> FONCTIONNEMENT ET OPÉRATIONNALITÉ :

Pouvez-vous présenter l'état d'avancement du projet au moment du dépôt ? (ce qui est terminé, en cours de réalisation, reste à faire)
Quelles approches avez-vous mis en œuvre pour vérifier l'absence de bugs et garantir une facilité d'utilisation de votre projet ?
Quelles sont les difficultés rencontrées et les solutions apportées ?

Nous avons pu implémenter tout ce que nous voulions jusqu'au placement et cassage de blocs, même si initialement, notre projet était plutôt orienté ~~objet~~ sur les catastrophes naturelles, comme les inondations, les sécheresses, les éruptions volcaniques etc. Le seul rescapé de ces idées farfelues a été le volcan, duquel nous sommes très fiers.

Il nous reste à améliorer l'opacité de blocs transparents, une génération plus agréable des structures. Dans un futur plus lointain, nous avions pensé peut-être ajouter ces fameuses catastrophes naturelles, peut-être plus de formes de blocs, ou même des entités. Et pourquoi pas, du multijoueur ?

Nous avons fait un Guide Utilisateur dans la Documentation, qui traite du fonctionnement et de l'utilisation du jeu, et aborde les problèmes (bugs/erreurs/ralentissements) possibles, leur cause et le moyen de les régler. (la documentation en elle-même traite de nettement plus de sujets que cela, nous vous incitons vivement à aller la consulter)

Pour vérifier l'absence de bugs, nous avons créé un fichier de test automatisé (non présent dans cette version). Nous avons donc lancé le jeu environ 2000 fois pour répertorier et corriger les erreurs.

Nous avons tout du long dû batailler avec des problèmes de performance, des bugs divers parfois stupides (Stanley par exemple avait oublié un «u » à « color » ce qui lui a valu une dizaine d'heures de souffrance). Quant à la performance, nous nous sommes assez vite rendus compte du fait que les listes python ainsi que leurs boucles allaient être trop lentes, et nous nous sommes donc tournés vers les numpy.array pour les opérations chronophages. C'est également pour des raisons de performance que nous avons utilisé OpenGL, et non tkinter.

> OUVERTURE :

Quelles sont les nouvelles fonctionnalités à moyen terme ? Avez-vous des idées d'amélioration de votre projet ? Pourriez-vous apporter une analyse critique de votre projet ? Si c'était à refaire, que changeriez-vous dans votre organisation, les fonctionnalités du projet et les choix techniques ?
Quelles compétences/appétences/connaissances avez-vous développé grâce à ce concours ?
En quoi votre projet favorise-t-il l'inclusion ?

A moyen terme, nous allons peut-être ajouter une sorte d'objectif à notre jeu / quelque chose à *speedrun,* car nos amis nous poussent à le faire pour que naisse une compétition et une *hype* autour du jeu. Cet objectif peut être par exemple récupérer un bloc spécial au cœur de 3 volcans, avec pourquoi pas l'ajout du mode Survie ?

PyCraft était un projet assez marrant à faire, car c'était vraiment le saut dans l'inconnu. Nous avions une vague idée de notre but, mais assez peu de certitudes sur la manière dont nous allions y parvenir.

Nous avons donc commencé sur certaines bases, que nous avons consolidées ensuite, tout en ajoutant par-dessus de nombreuses fonctionnalités.

En effet, si nous le refaisions avec toutes les connaissances que nous avons maintenant, nous pourrions avoir une architecture bien plus structurée (surtout du côté de la génération), et ainsi cela nous permettrait d'envisager beaucoup d'optimisations. Par exemple, pour l'instant, les structures sont stockées sous la forme de fichiers en local car notre architecture permet difficilement de faire mieux, mais nous pourrions les stocker en RAM (en variables globales par exemple) pour gagner beaucoup de performances.

Cependant, nous avons l'impression que nous n'aurions pas vraiment pu faire mieux sans connaissances préalables. Peut-être quand même nous aurions pu mieux nous organiser, de sorte à limiter le plus possible les "merges" qui sont assez chronophages pour pas grand-chose.

Ce projet a été l'occasion de nous améliorer dans de nombreux domaines. Stanley s'est largement amélioré dans tout ce qui touche au GPU, en manipulant et apprenant OpenGL. Oscar a

aussi appris les mécaniques d'affichage 3D, en communiquant avec lui. Il a aussi touché aux joies du multiprocessing.

Arthur a lui beaucoup appris sur tout ce qui touche aux arrays numpy, à la vectorisation, mais aussi certains algorithmes comme le Perlin Noise.

Globalement, nous avons tous appris les plaisirs de l'optimisations et les tests de performances, mais aussi quelque chose de très important qui est la coopération et la communication (heureusement nous nous voyions tous les jours).