RAPPORT DE MINI PROJET

MGA 802 – INTRODUCTION À LA PROGRAMMATION AVEC PYTHON

MINI PORJET A – EDITEUR DE G-CODE

AUTEURS : BERNABEU LEO – DA COSTA DIAS ANDRE – MULLINS MATHIEU

MONTREAL, LE 30 MAI 2023

ÉCOLE DE TECHNOLOGIE SUPERIEURE

1. Introduction

L'objectif de ce projet est de concevoir un éditeur de G-code. À partir de paramètres d'ajustement fournis par l'utilisateur, cet éditeur doit être en mesure de modifier adéquatement les instructions d'un fichier G-code préexistant généré, par exemple, par un logiciel de découpage et d'impression 3D comme *PrusaSlicer*. L'utilisateur doit pouvoir regrouper les couches successives en phases, puis pouvoir modifier la température et la vitesse d'impression ainsi que le débit du matériau extrudé sur chacune des phases. Enfin, l'utilisateur doit pouvoir décaler la position de la pièce sur le lit. De plus, nous avons décidé, comme tâche optionnelle, de proposer à l'utilisateur de rajouter une phase de réchauffement d'une couche venant d'être imprimé pour améliorer la soudure avec la prochaine couche.

2. Choix de programmation

Nous présentons ici nos principaux choix de programmation.

2.1. Fichier de paramétrage

Pour permettre à l'utilisateur d'entrer ses paramètres, nous lui exigeons qu'il mette à jour un fichier texte qui sera utilisé en entrée de notre éditeur de G-code. L'utilisateur doit éditer ce fichier texte en suivant un format précis. Ce choix s'explique par la quantité potentiellement très importante de paramètres entrés par l'utilisateur. En séparant l'objet en une dizaine de phases, l'utilisateur doit fournir plusieurs dizaines de valeurs. C'est pourquoi nous avons jugé qu'une console Python n'est pas suffisamment ergonomique pour entrer une grande quantité de valeur. Aussi, si l'utilisateur se rend compte d'une erreur en cours de route (par exemple d'une faute de frappe ou d'une mauvaise valeur), celui-ci doit annuler l'exécution du programme et le réexécuter intégralement. Avec un fichier texte, il est alors possible de corriger ses erreurs sans avoir besoin de tout réécrire. Ensuite, nous jugeons que le format est plus lisible, car les paramètres peuvent être écrits en colonne (une valeur par ligne) et des commentaires peuvent être ajoutés par l'utilisateur. Enfin, cela permet plus facilement de garder en mémoire et suivre les paramétrages qui ont été testés.

2.2. Gestion du flux de contrôle

La fonction principale de notre programme est *gcode_editor()*. C'est elle qui appelle toute les autres sous fonctions de notre programme. Voici une liste des étapes principales de notre programme.

- Aller chercher les paramètres provenant du fichier de paramètres avec *extract_values_from_file()*.

- Aller chercher les informations sur les couches du G-code. La fonction *find_layer_info()* nous permet d'aller chercher le nombre de couches dans notre G-code en consultant ce dernier ligne par ligne et en gardant en mémoire chaque changement de ligne jusqu'au dernier. Le dernier changement de ligne est retourné comme étant le nombre total de lignes. Cette variable nous permet plus tard de savoir dans quelle phase se trouve chaque ligne.
- Début de la boucle principale de lecture et modification du fichier G-code ligne par ligne :
    - Nous avons décidé d'identifier chaque ligne devant être modifiée par les caractères au début de la ligne (exemple : « G1 »).
    - Puisqu'un élément dans une ligne est séparé par des espaces, chaque ligne est ensuite séparée en une liste (*modified_line_parts*) d'éléments.
    - La liste d'éléments est ensuite envoyée à sa fonction de modification en fonction de la modification à appliquer (exemple : *modify_speed(...)*).

2.3. Modification ligne par ligne

La fonction *gcode_editor()* lie le fichier de G-code ligne par ligne. Pour chaque ligne qui doit être modifiée, nous appliquons la modification de la ligne et la retournons tout de suite dans le fichier de G-code modifié. Si la ligne ne doit pas être modifiée, nous la retournons tel quelle dans le fichier de G-code modifié.

Nous choisissons d'envoyer les lignes lues et modifiées ou non-modifiée tout de suite dans notre fichier de G-code modifié afin de diminuer la mémoire interne requise par l'ordinateur. Une alternative aurait été de sauvegarder chaque ligne dans une liste et d'envoyer la liste modifiée dans le fichier modifié uniquement lorsque le fichier aurait été lu et modifié au complet. Bien que le gain soit minime pour le fichier test fourni dans le cadre du projet (~7min d'impression), nous croyons qu'un gain important serait remarqué pour des fichiers de G-code plus volumineux.

3. Conclusion

En conclusion, ce projet d'introduction à Python a été une expérience pratique enrichissante, nous permettant d'appliquer nos compétences en programmation dans une équipe à l'aide d'un dépôt *GitHub* complet et de créer un code fonctionnel conforme aux exigences du cours. Les fichiers *readme* donnent plus de détails sur l'implémentation de notre code. En réalisant les tâches obligatoires et facultatives, cette expérience a renforcé nos compétences en Python, notre compréhension de l'impression 3D et nous a préparés pour des projets futurs nécessitant des compétences similaires.