LES TROPHÉES NSI

Édition 2023

DOSSIER DE CANDIDATURE
PRÉSENTATION DU PROJET

Ce document est l'un des livrables à fournir lors du dépôt de votre projet : 4 pages maximum (hors documentation).

Pour accéder à la liste complète des éléments à fournir, consultez la page **Préparer votre participation**.

Vous avez des questions sur le concours ? Vous souhaitez des informations complémentaires pour déposer un projet ? Contactez-nous à **info@trophees-nsi.fr**.

NOM DU PROJET : CalculaTreece

> PRÉSENTATION GÉNÉRALE :

Durant cette année de Terminale, deux éléments nous ont amenés à choisir de réaliser ce projet en particulier. La recherche d'une idée de projet, ~~quoi de mieux pour trouver une idée de projet~~ ainsi que la recherche de concret. En effet, nos contenus de cours de NSI (piles/files/arbres/…) étaient intéressants, mais sans vraiment d'exemple d'application.

À travers des exercices d'approfondissement donnés par notre professeur (basé sur le calcul post-fixe en utilisant des piles) nous est venue l'idée toute bête d'une calculatrice. *Sans doute la première application jamais créée dans l'histoire de l'informatique, on voulait peut-être juste recréer la roue…*

La structure la plus *« rigolote »* à utiliser pour des calculs était à nos yeux les **arbres**, à gros coups de récursivité. Le nom du projet a donc découlé tout naturellement de nos esprits : « **CalculaTreece** » *(tree pour arbre, calculatrice en phonétique française)*. Pour l'anecdote, l'icône de l'application a été générée à l'aide de DALL-E, « un arbre qui est une calculatrice », pas besoin de se casser la tête.

Au départ, nous n'avions nous même pas pris ce projet au sérieux, et au fil du temps, des idées de fonctionnalités nous venaient à l'esprit rendant ce projet un peu plus *pétillant*, comme par exemple le dessin d'arbres pour avoir une visualisation de notre calcul ~~et pas grand-chose d'autre, mais c'est rigolo les arbres alors ça a suffit.~~

À terme, nous nous sommes retrouvés avec une calculatrice capable de calculer des expressions complexes (*malgré peut-être quelques bugs qui nous ont échappés)*, de dessiner notre calcul, de le calculer *(oui, aussi)*, de résoudre des équations *(même si le second degré n'est pour l'instant qu'une température*, plus sérieusement, cette fonctionnalité n'est qu'expérimentale pour le moment*)*, de dériver des expressions, et tout ça dans une jolie interface avec des boutons !

Nous espérons que cette application vous plaira, et que vous y verrez un quelconque intérêt *(calculer par exemple ?)*.

> ORGANISATION DU TRAVAIL :

Nous étions trois membres, **Naël Juniot**, l'élément moteur du groupe *(c'est moi qui écrit cette partie, j'y dis un peu ce que je veux :))*, **Thomas Geffroy**, un fan de pygame et **Camille Gillé**, un mordu du web, ~~qui aurait été capable de vouloir ajouter une fonction multijoueur.~~ Nous aurions bien aimé avoir une certaine mixité dans le groupe mais la composition de notre classe ne le permettait par vraiment...

Voici la répartition des tâches pour laquelle nous avons opté :

- Naël : transformations des expressions en arbre et calcul (puis dérivation sur un coup de tête)
- Thomas : interface ~~*bien exploité par nos demandes*~~
- Camille : équations *(ça n'a pas dû être facile, aucune idée de la méthode)* et système d'historique

Dès que nous sommes passés à la partie code, nous avons tout de suite créé un répertoire sur **GitHub** (celui-ci) afin de pouvoir contribuer chacun depuis notre ordinateur. Au départ peu familiers avec Git, ceci nous a permis de nous familiariser avec son fonctionnement. Nous avons également créé un serveur **Discord** auquel nous avons lié le répertoire par un WebHook (afin d'être notifié des nouveaux commits). Un dernier outil que nous avons utilisé de temps à autre est Code With Me, un outil de **JetBrains** permettant de travailler sur un même projet en temps réel *(permettant également d'exécuter ce que l'on veut sur l'ordinateur de l'hôte, amenant quelques blagounettes plus ou moins catastrophiques)*.

L'outil le plus pratique restait de loin Code With Me. Nous l'utilisions dès que nos disponibilités concordaient, sinon nous nous contentions de nous notifier sur Discord en cas de bugs/besoin d'aide/idées. Il nous arrivait également de travailler sur ce projet en cours de NSI ~~discrètement~~ sur l'accord de notre professeur. Nous nous retrouvions parfois hébergés chez un camarade, pour pouvoir constituer le dossier de projet.

Pour la recherche de bugs, nous le faisions individuellement de temps en temps au grès des envies, *et forcions le responsable de cette horrible erreur à le corriger sous peine d'être exclu du projet.* Une dernière session de vérification a été faite avant de déposer ce projet pour les Trophées NSI, ce qui n'a malheureusement pas dû empêcher certains bugs d'échapper à nos tests.

LES ÉTAPES DU PROJET :

La première étape *(après le choix du projet)* est de faire un système permettant de calculer *(sans quoi on est mal barré avec notre calculatrice)*, pour être sûr d'aller vers quelque chose. Les classes d'arbres ont donc été réalisées en première, puis des fonctions de calculs, puis des fonctions pour convertir une chaîne de caractères en arbre (à travers plusieurs conversions string->list->tree).

L'étape suivante a été d'avoir une idée de l'apparence, des boutons disponibles. Les détails supplémentaires viendront pendant la création de l'interface. Vous pourrez voir dans la **DOCUMENTATION** le seul et unique croquis de l'application, et il est tellement plus pratique de s'imaginer des carrés que de les dessiner que l'idée de croquis a vite été abandonnée…

Il a donc fallu choisir une librairie graphique ~~*ou en coder une, mais bon trop facile pour nous, perte de temps*~~, et nous avons opté pour **pygame** . Sans doute pas la plus adaptée pour une application bureau, mais nous avions déjà une idée de comment faire avec celle-ci. Pour le dessin d'arbres, il était plus simple de le faire avec **turtle** (ce qui permet également d'ouvrir une seconde fenêtre), avec une fonction de dessin récursive.

Simultanément à la création de l'interface, Camille travaillait sur la partie équations, de manière à pouvoir tirer un nombre d'un arbre contenant un « = ». Une partie assez sombre, aucune idée de comment il a fait, ~~et je n'ai pas envie de fourrer mon nez dedans~~ et je lui fais entièrement confiance sur ce point.

Certaines autres améliorations imprévues ont eu lieu, tel que l'ajout de la dérivation. Nous ne pouvons pas vraiment considérer ces améliorations comme des étapes.

Certains éléments qu'il serait possible de considérer comme étapes : les commentaires / doc-strings et la recherche de bugs ont en réalité eu lieu tout au long du projet. Il est bien plus simple de se repérer dans un code bien documenté que de perdre quelques précieuses secondes à se souvenir de ce qu'un bout de code est censé faire. La recherche de bugs au fil du temps était aussi nécessaire qu'involontaire, c'est en testant nos fonctionnalités pour en voir le résultat qu'on trouve beaucoup de bugs, et avoir un code qui fonctionne est nécessaire ~~à notre plaisir~~ pour pouvoir travailler à plusieurs sur un même projet, surtout quand tout est lié.

Malgré cette avancée progressive, nous avons tout de même dédié une première étape à la recherche des derniers bugs suivie d'une grande vérification des commentaires / doc-strings *(il serait bête de documenter un code rempli de bugs)*.

> FONCTIONNEMENT ET OPÉRATIONNALITÉ :

À l'heure actuelle, l'application est fonctionnelle *(surprenant, même pour nous)*. Nous pouvons donc comme prévu utiliser notre application de calculatrice pour *calculer*, super ! Plus sérieusement, beaucoup des fonctionnalités prévues ont pu être implémentées (calculs, dessins, dérivation, historique, équations).

Il faudrait encore améliorer l'utilisation de l'historique, comme par exemple en ajoutant une barre de défilement *(et aussi corriger quelques soucis d'utilisation qu'on espère bien cachés)*.

La partie équation est encore à considérer comme en cours de réalisation. On ne peut pour l'instant résoudre que des polynômes de 1er degré développés. ~~Camille n'a~~ Nous n'avons pas eu le temps de pousser cette partie en raison de la difficulté de celle-ci sans module externe ~~*et du fait que nous n'avons aucune idée de comment faire*~~. Car oui, un des objectifs majeurs de ce projet était de le réaliser sans module non-natif (sauf pygame, que nous aurions en effet pu remplacer par tkinter*, mais pygame est tellement connu parmi les modules d'interfaces graphiques que nous nous excusons nous même*). Utiliser simpy nous aurait bien aidé pour cette partie, mais cela n'aurait pas été marrant.

Ok, ça fonctionne, mais comment ? Toutes les actions de l'utilisateur passent par les boutons et le clavier. Tous les boutons sont associés à une action, qui peut consister en la modification du texte ou l'appel d'une fonction (ou les deux). Nous avons essayé de donner à chaque fonction une utilité, par exemple, le calcul du texte donné par l'utilisateur passe par 4 fonctions distinctes jusqu'à l'obtention du résultat.

Pour ce qui est de l'ergonomie, nous ne pouvions pas faire tester cette application à n'importe qui. Naël aurait bien montré à son petit frère de 11 ans que c'est super de pouvoir dériver en cliquant sur un bouton, mais il aurait sans doute vite été lassé. Les commentaires de personnes externes découlaient donc souvent d'une démonstration de notre part à un parent/camarade (voir professeur) capable d'avoir un regard critique sur les

mathématiques. Une remarque très pertinente était de différencier les couleurs des boutons multiplier et « x » (la lettre), c'est tellement évident qu'on oublie d'y songer…

Palier à la présence de bugs n'a pas de secret : un peu de recherche ~~et beaucoup de chance~~. Cette recherche passait à la fois par des tests, en testant les fonctions une à une avec les cas les plus tordus possible, mais également par la relecture. Une lecture avisée peut parfois nous faire remarquer un soucis dans le code.

Dans la majorité des cas, les bugs étaient découverts par hasard lors du test d'une autre fonction utilisant la fonction défectueuse.

Malgré tout, cette longue aventure ne consistait pas qu'en l'ajout de nouvelles fonctions et la correction de petits bugs, comme dans le meilleur des mondes. Il nous est parfois arrivé de devoir refaire des morceaux de code entier pour pouvoir à nouveau progresser. Un des obstacles en ce genre était lié à l'interface graphique : tout était géré dans une unique classe, ce qui rendait le tout très brouillon et presque impossible d'utilisation. Il a fallu refaire entièrement cette partie en la segmentant en plusieurs petites classes, avec chacune leur utilité. Un autre a été de convertir une chaîne de caractère en… quelque chose d'utilisable, et l'idée d'analyser des chaînes de caractère à l'aide d'automates nous a permis de faire quelque chose de *relativement* fonctionnel.

> OUVERTURE :

Dans le cadre d'améliorations futures, plusieurs nouvelles fonctionnalités pourraient être envisagées pour enrichir notre application de calculatrice. Parmi celles-ci, l'intégration d'un solveur d'équations plus avancé pour résoudre des polynômes de degré supérieur, l'ajout d'un outil de calcul matriciel, et la mise en place d'un système de visualisation graphique pour représenter des fonctions en deux ou trois dimensions, ou encore permettre à la CalculaTreece de gérer les expressions post-fixes.

Pour toucher un large public, nous pourrions mettre en place une stratégie de diffusion originale en organisant un concours en ligne, invitant les participants à résoudre des problèmes mathématiques complexes en utilisant notre application. Les gagnants pourraient recevoir des récompenses telles que des abonnements gratuits à des plateformes éducatives, ou des stages auprès de professionnels des mathématiques. De plus, nous pourrions collaborer avec des influenceurs dans le domaine de l'éducation pour promouvoir notre application sur les réseaux sociaux et les plateformes de partage de vidéos.

En ce qui concerne une analyse critique du résultat, si c'était à refaire, nous pourrions apporter quelques changements dans notre organisation, les fonctionnalités du projet et les choix techniques. Au niveau de l'organisation, une répartition plus équilibrée des tâches et une meilleure planification pourraient permettre de gagner en efficacité et de réduire les retards. Concernant les fonctionnalités, nous pourrions envisager d'intégrer davantage de modules spécialisés pour répondre à des besoins spécifiques en mathématiques, comme la géométrie, l'algèbre et la trigonométrie. Enfin, sur le plan technique, nous pourrions reconsidérer l'utilisation de pygame et opter pour une autre bibliothèque graphique plus adaptée aux applications de bureau, comme tkinter, pour améliorer la compatibilité et l'efficacité de notre application.

DOCUMENTATION

Installation :

- Téléchargez python 3.10 à partir d'ici
- Téléchargez ce dépôt en utilisant git clone ou le bouton download (liens du dépôt)
- Ouvrez un terminal dans le dossier où vous avez téléchargé le dépôt
- Exécutez python -m pip install -r requirements.txt pour installer les dépendances
- Lancez python main.py pour démarrer CalculaTreece

Guide d'utilisation :

Pour utiliser la calculatrice, l'utilisateur doit saisir une expression mathématique en notation infixée. L'expression infixée est ensuite évaluée à l'aide d'un arbre binaire.

Spécifications techniques :

- Langage : Python ≥ 3.10
- Dépendances : pygame 2.1.2
- Architecture : La calculatrice est basée sur des arbres binaires pour évaluer les expressions mathématiques
- Format de stockage des données : La calculatrice ne stocke pas de données.

Spécifications fonctionnelles :

CalculaTreece est une calculatrice basée sur des arbres binaires qui permet aux utilisateurs de faire des calculs mathématiques simples.

Voici les différentes fonctionnalités de la calculatrice :

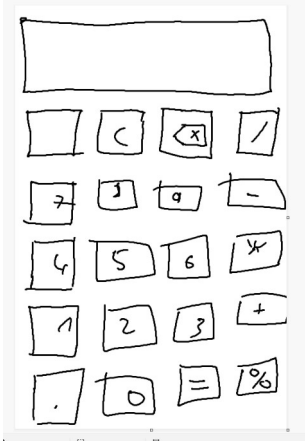
- Addition (+) : permet d'additionner deux nombres ensemble.
- Soustraction (-) : permet de soustraire un nombre d'un autre.
- Multiplication (*) : permet de multiplier deux nombres ensemble.
- Division (/) : permet de diviser un nombre par un autre.
- Modulo (%) : permet de trouver le reste de la division entre deux nombres.
- Puissance (^) : permet de calculer la puissance d'un nombre.
- Fonction exponentielle (sqrt) : permet de calculer la racine carré d'un nombre.
- Résolution d'équation simple à un inconnu
- Dérivation
- Représentation graphique à travers un arbre binaire

Déroulement des étapes d'exécution :

- L'utilisateur saisit une expression mathématique en notation infixée.
- L'expression infixée est convertie en liste.
- La liste est convertie en arbre afin d'être évalué par une fonction.
- Le résultat est affiché à l'utilisateur.

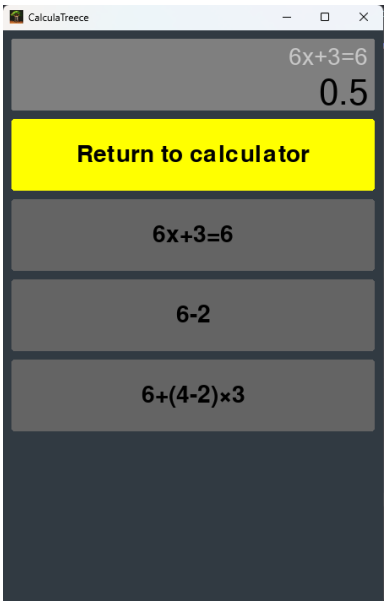
IMAGES

Voici donc les images liées à ce projet.
Pour commencer, voici le tant attendu **croquis** :
L'intention est en effet sympathique, mais ne nous a pas beaucoup aidé, le design ne consistait en réalité qu'en un choix de boutons et de couleurs.



Ici, une image de la **vue de base** avec un exemple de calcul :
Nous pouvons y voir en bas à gauche les boutons permettant :
- La dérivation
- Le dessin de l'expression
- Le passage à la vue d'historique

Ici, une vue de l'**historique** :
Le clic sur un des boutons ramènera le résultat correspondant dans l'entrée de texte.



Et ici, une vue d'un **arbre de calcul**, résultat de l'utilisation de la fonction *« Draw »* avec la même expression que pour l'image de la vue de base vue plus haut.