



Rapport final projet Python

Sujet n°4 : Jeux Olympiques

SOMMAIRE

Introduction	2
Rappel des spécifications et de la conception	2
Méthode générale de résolution.....	4
Détails des parties originales.....	6
Problèmes rencontrés et solutions apportées	7
Ecart par rapport au cahier des charges	8
Bilan	8
Conclusion.....	9
Sitographie	10

Introduction

Le présent rapport constitue le compte-rendu final de notre projet d'application de gestion des Jeux Olympiques. Ce projet avait pour objectif de créer une belle interface graphique sous forme d'application, avec navigation intuitive entre les différentes fonctionnalités, permettant d'afficher et de gérer la base de données des JO.

Dans ce rapport, nous allons présenter succinctement les principales étapes de réalisation du projet : Les objectifs initiaux, la méthode choisie, les problèmes rencontrés, ainsi que les résultats obtenus. Nous aborderons également les perspectives d'amélioration de notre travail.

Rappel des spécifications et de la conception

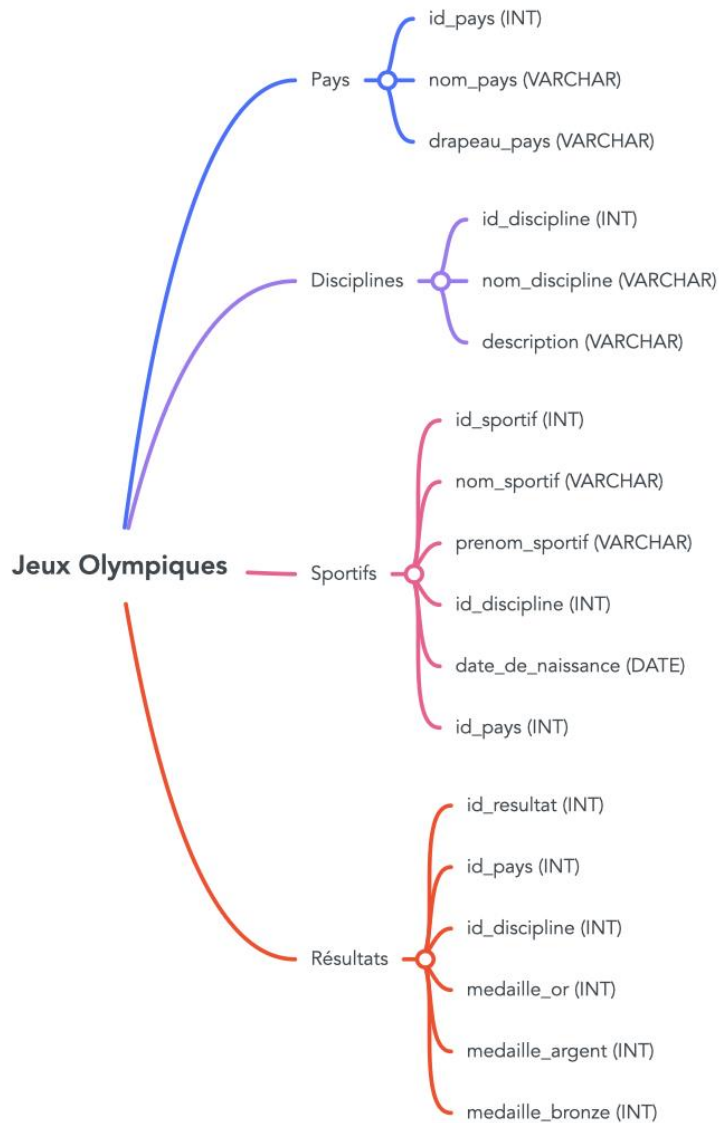
Le but était de concevoir une interface utilisateur attrayante et ergonomique, offrant une expérience utilisateur fluide et intuitive.

Cette interface nous permettant d'afficher et de gérer la base de données des JO, c'est-à-dire de :

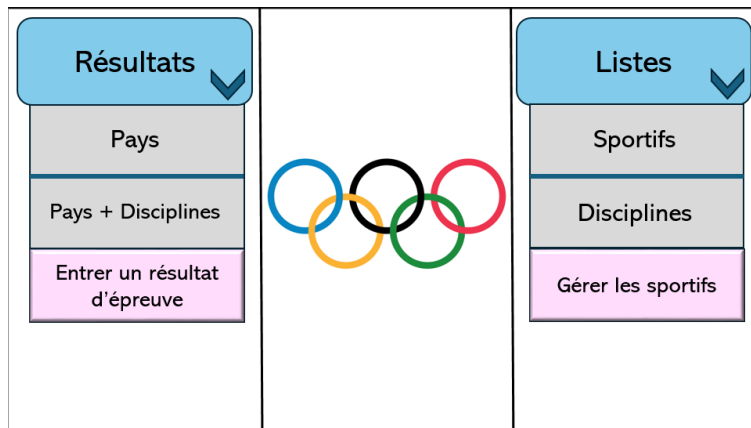
- Consulter la liste des pays participants, avec les drapeaux associés, le nombre de médailles (images) actuelles (or, argent, bronze).
- Consulter la liste des disciplines.
- Consulter les résultats par discipline et par pays.
- Enregistrer les résultats d'une épreuve.
- Gérer des sportifs :

- Ajouter ou supprimer des sportifs.
- Consulter la liste des sportifs.

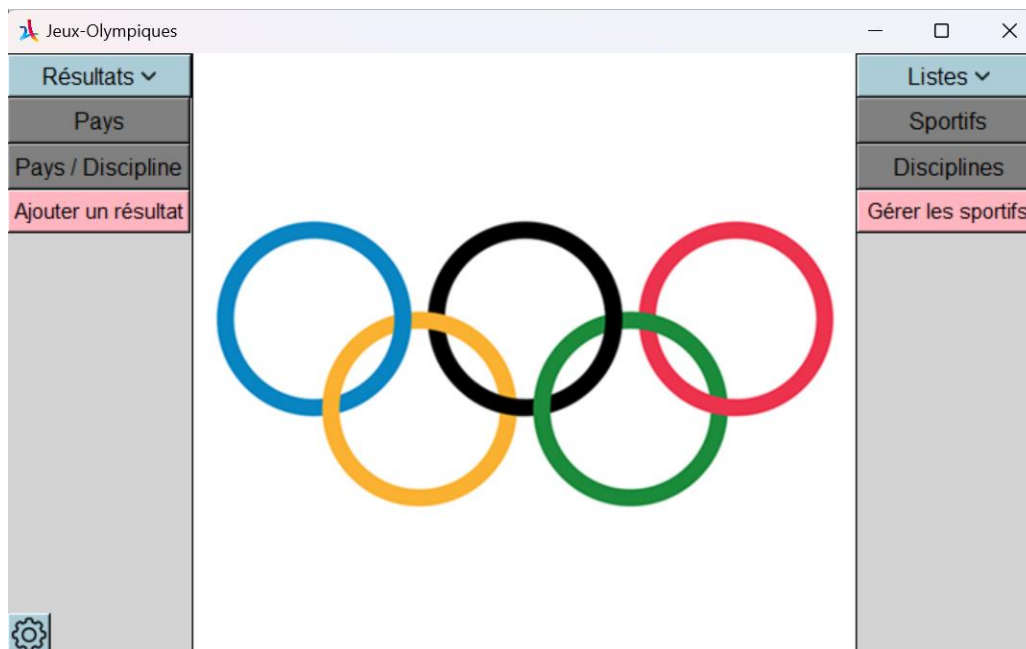
Pour ce faire, nous avons choisi de nous baser sur le schéma suivant pour la base de données :



L'interface graphique devait ressembler à cela :



Ce qui a été réusé car actuellement l'interface ressemble à ceci :



Nous avons donc bien respecté les objectifs initiaux, et nous avons même ajouté un onglet de paramètres, ce qui n'avait pas été prévu dans le document de conception.

Méthode générale de résolution

Notre méthode c'est d'abord basée sur l'ajout des fonctionnalités globales pour avoir une première version opérationnelle, puis l'ajout de fonctionnalités plus précises (exemple : barre de défilement, onglet paramètre, etc) au fur et à mesure.

À chaque nouvelle fonctionnalité ajoutée, nous vérifions qu'il n'y avait aucun dysfonctionnement. Et s'il y en avait, nous les résolvons au fur et à mesure. Cette phase de validation nous a permis de garantir la qualité et la stabilité de notre projet.

Nous avons codé en même temps la partie interface graphique avec Tkinter et la partie permettant d'interagir avec la base de données. Ce qui nous a permis d'afficher les résultats directement dans une interface graphique en construction, plutôt que d'afficher

d'abord les résultats dans le terminal, avant de les afficher dans l'interface graphique. Ce qui aurait rajouté une étape supplémentaire inutile.

Bien sûr pour ce faire nous nous sommes réparti les tâches. Nicolas s'occupait principalement de la partie interface utilisateur, tandis que Baptiste réalisait essentiellement l'interaction entre la base de données et l'interface graphique. Même si évidemment, nous ne nous sommes pas cantonnés à ce type de fonctionnement, chacun aidant l'autre sur n'importe quelle partie et à n'importe quelle étape du projet.

La répartition des tâches avait été initialement prévue (lors de la rédaction du document de conception) de telle sorte à ce que chacun travaille sur un menu du projet. En effet, étant donné que notre interface comporte un menu à gauche et à droite, nous avons pensé qu'il serait plus facile de se répartir les rôles ainsi. Ce qui impliquait d'ailleurs une phase d'harmonisation des deux parties dans le planning. Mais cette répartition a rapidement été abandonnée au profit de celle présentée ci-dessus, car jugée trop contraignante.


Pour suivre plus facilement l'avancement du projet et pour respecter notre planning prévisionnel, nous avons utilisé un tableur Excel :

Avant :

L19

</

Après :

Projet Python JO			Légende et bouton des couleurs d'état					
			Non commencé	En cours	Retardé	Terminé		
			Début du projet :		Thu, 3/21/2024		Fin du projet 17/05/2024 à renc	
TÂCHE	Statut	ATTRIBUÉE A	AVANCEMENT	DÉBUT	FIN supposée	FIN réelle	Ecart (en jours)	Début réel
Base de données	Terminé	Baptiste	100%	21/03/24	25/03/24	21/03/24	4	21/03/24
Interface graphique principale opérationnelle (version non finale)	Terminé	Nicolas + Baptiste	100%	22/03/24	29/03/24	26/03/24	3	22/03/24
Partie code Python « Listes » opérationnelle (version non finale)	Terminé	Nicolas	100%	29/03/24	08/04/24	06/04/24	2	26/03/24
Partie code Python « Résultats » opérationnelle (version non finale)	Terminé	Baptiste	100%	29/03/24	08/04/24	19/04/24	-11	26/03/24

Et nous utilisons GitHub pour nous partager le code au fur et à mesure. Il est d'ailleurs consultable à cette adresse : <https://github.com/Baptiste-micl/Projet-Jeux-Olympiques>

Détails des parties originales

La fonction du code permettant de créer un rectangle bleu arrondi est un peu complexe. Nous allons donc la détailler.

Cette fonction nous permet d'afficher le message : « Veuillez relancer le programme pour mettre à jour la base de données. » dans un beau rectangle bleu arrondis, après l'ajout d'un sportif par exemple.

Ce rectangle est créé dans cette fonction :

```
def arrondis(canvas, x1, y1, x2, y2, radius, **kwargs):
    points = [x1+radius, y1,
              x1+radius, y1,
              x2-radius, y1,
              x2-radius, y1,
              x2, y1,
              x2, y1+radius,
              x2, y1+radius,
              x2, y2-radius,
              x2, y2-radius,
              x2, y2,
              x2-radius, y2,
              x2-radius, y2,
              x1+radius, y2,
              x1+radius, y2,
              x1, y2,
              x1, y2-radius,
              x1, y2-radius,
              x1, y1+radius,
              x1, y1+radius,
              x1, y1]
    canvas.create_polygon(points, kwargs, smooth=True)
```

Cet ensemble d'instruction à l'intérieur de la liste « points » sont les coordonnées des points nécessaires pour dessiner un rectangle avec des coins arrondis. Ces coins arrondis sont calculés en ajoutant ou soustrayant le rayon *radius* aux coordonnées des coins du rectangle.

Ainsi, le programme commence par dessiner le rectangle en partant du coin supérieur gauche, où il fait l'arrondi. Puis une droite horizontale jusqu'au deuxième angle en haut à droite, qu'il fait arrondir. Puis une droite verticale vers le bas, et ainsi de suite jusqu'à avoir effectué le contour du rectangle en entier.

Le paramètre additionnel ***kwargs* est utilisé pour passer des options supplémentaires au rectangle, comme la couleur de remplissage *fill*, la couleur de contour *outline*, etc.

L'option *smooth=True* est utilisée pour rendre les bords du rectangle lisses, ce qui est particulièrement utile pour créer l'effet visuel de coins arrondis.

Le reste du programme est normalement assez bien commenté et les noms des éléments codés (variables, fonctions, etc.) ont été choisis de telle sorte à ce que ce soit compréhensible.

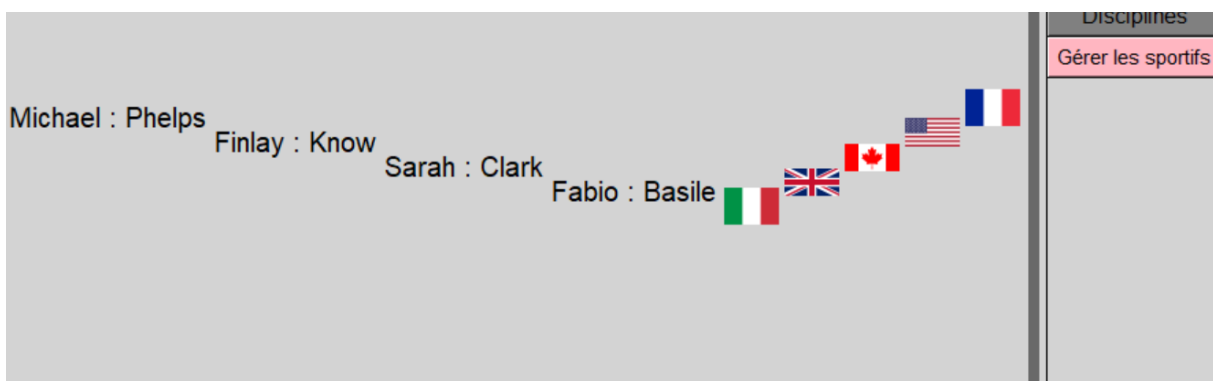
Problèmes rencontrés et solutions apportées

1 : Nous nous sommes trop rapidement lancés dans le programme sans penser à la structure du code. À partir de 300 lignes de code dans un seul fichier, cela devenait assez pénible à chaque nouvelle fonctionnalité ajoutée ou mise à jour. Nous avons essayé en vain de séparer le code en plusieurs fichiers, car la structure initiale de notre code nous en empêchait (nous étions obligés de faire des « importations circulaires » ce qui faisait dysfonctionner le programme). Nous avons donc fait différentes choses pour résoudre ce problème :

- Ajout d'un commentaire « # changement » à côté de chaque ligne de code ajoutée ou modifiée.
- Ajout de commentaires explicatifs à côté des lignes de codes
- Commentaires sur [GitHub](#) pour détailler ce qui a été fait à chaque mise à jour du code
- Code minimaliste pour ne pas alourdir le code et réduire le nombre de lignes
- Utilisation de noms de variables et fonctions facilement compréhensibles

2 : Nous avons passé beaucoup de temps sur l'affichage correct et cohérent d'images (drapeaux et médailles) correspondantes aux bons résultats, ou sportifs.

Ce qui a d'ailleurs donné lieu à des affichages assez étonnants :



3 : Nous avons aussi passé un certain temps à trouver la bonne solution pour avoir une barre de défilement (scrollbar) pour nos différentes listes (de sportifs par exemple).

En effet, Tkinter peut nous permettre de faire une barre de défilement, mais elle est assez difficile à utiliser, est peu esthétique, et ne fonctionne pas pour tous les types de widgets (boutons, images, blocs de textes, etc.).

Nous avons donc, après de nombreux essais, opté pour le module « Custom Tkinter » qui permet de faire facilement des cadres avec une barre de défilement. Et celle-ci est facilement modifiable au niveau de la couleur, etc.

À noter que la barre de défilement horizontale s'utilise avec la touche Majuscule enfoncée en plus de la molette.

4 : Enfin la dernière partie qui nous a causée de nombreux problèmes a été la possibilité d'ajouter et de supprimer des sportifs dans la base de données. Il a fallu gérer le fait que tous les champs (nom, prénom, disciplines, pays) soient bien renseignés par l'utilisateur. Avec des messages d'erreur, si ce n'était pas le cas.

La solution a été d'utiliser les éléments try, except et ValueError pour que le tout fonctionne du mieux possible et de manière relativement simple, et sans allonger le code inutilement.

Écarts par rapport au cahier des charges

Dans l'ensemble, nous avons très bien respecté les objectifs que nous nous étions fixés dans le document de conception.

Le seul point sur lequel nous nous sommes écartés a été le fait de ne pas prendre en compte les médailles par sportif.

En effet dans notre document de conception, on peut y lire « Tel sportif a eu telle médaille pour telle discipline ». Ce qui n'a pas été réalisé, car nous avons jugé cette fonctionnalité inutile et trop éloigné de ce que nous souhaitions faire à l'origine.

Nous avons aussi attribué à chaque sportif sa date de naissance dans la base de données, mais nous avons décidé de ne pas afficher les dates de naissance pour ne pas alourdir l'affichage de la liste des sportifs.

Bilan

Ce projet nous a été très formateur. Au-delà de la gestion de projet en elle-même, cela nous a poussés à faire de nombreuses recherches sur la documentation des librairies ainsi que sur des forums comme Stackoverflow, mais aussi de développer notre esprit créatif avec la création de l'interface graphique.

Baptiste : Ce projet m'a permis sur le plan technique, d'approfondir mes connaissances en Python, notamment grâce à un apprentissage plus poussé de la bibliothèque Tkinter ainsi que de son extension, Custom Tkinter. J'ai pu explorer en détail les fonctionnalités offertes par ces outils et les appliquer dans des contextes variés, ce qui m'a permis de développer des interfaces utilisateur plus sophistiquées et intuitives.

Bien que j'aie apprécié travailler sur ce projet et que cela m'ait permis de renforcer mes compétences techniques, cette expérience m'a également conduit à une réflexion plus personnelle sur mes aspirations professionnelles. J'ai réalisé que, malgré l'intérêt que je porte au développement logiciel, je ne souhaite pas poursuivre une carrière dans ce domaine en tant qu'ingénieur.

Nicolas : Cela m'a permis de développer une certaine aisance dans la gestion de projet.

Auparavant, je trouvais que la programmation était un domaine intéressant, mais qui ne passionnait absolument pas. J'ai néanmoins beaucoup apprécié travailler sur ce programme. Sûrement car il était sous forme de projet, où nous nous sommes fixé nos propres consignes et objectifs. Et nous avons pu travailler sur un projet plus grand et plus complet que si cela avait été un travail individuel.

Néanmoins, comme Baptiste, je ne souhaite pas poursuivre une carrière dans le domaine du développement informatique.

Conclusion

Nous sommes très fiers du rendu final de notre programme. Nous avons parfaitement respecté les consignes que nous nous étions nous-mêmes fixés. Tout le programme fonctionne sans aucun dysfonctionnement.

Et le pari de faire une interface utilisateur intuitive a été réussi. En effet, la grand-mère de Nicolas, qui n'avait jamais touché à un ordinateur de sa vie, a eu l'occasion de tester le programme et a réussi à utiliser toutes les fonctionnalités sans accros.

Prolongements possibles et améliorations :

Nous nous sommes volontairement bridés lors de la définition de nos objectifs lors de la rédaction du document de conception afin de ne pas avoir un trop gros projet infaisable au vu de la charge de travail.

Néanmoins, ces objectifs que nous avons choisi d'écarter peuvent être vus comme des pistes de prolongements possibles.

En effet, le but était de faire un programme pour la gestion des JO qui puisse s'adapter à n'importe quel JO. Donc pas forcément ceux de Paris 2024. C'est donc pour cette raison qu'il n'y a pas de billetterie, de gestion des lieux d'épreuves, ni tout autre élément qui place notre programme dans un cadre temporel, le rendant rapidement obsolète et non-universel.

Mais on peut envisager d'améliorer le programme pour le rendre plus spécifique aux JO de Paris 2024, et ainsi y ajouter une billetterie avec les lieux des épreuves, etc.

Au cours du projet, nous avons eu des idées pour compléter le projet plus pleinement :

- L'utilisation d'une API météo pour voir quel temps il fait sur les lieux d'épreuves.
- Un bouton permettant de basculer entre un thème sombre et un thème clair dans les paramètres.
- Pouvoir ajouter/supprimer un sportif et qu'il s'ajoute directement sans avoir à relancer le programme pour mettre à jour la base de données.
- Transformer l'entièreté de notre programme en un seul fichier .exe

Sitographie

Nomenclature utilisée : Nom de la source – auteur : (lien / partie utilisée)

- Scrollable Frames!! – Youtube Tkinter.com : (<https://youtu.be/Envp9yHb2Ho>)
- CTkScrollableFrame – CustomTkinter : (<https://customtkinter.tomschimansky.com/documentation/widgets/scrollableframe/>)
- Difference between the "fill" and "expand" options : (<https://stackoverflow.com/questions/28089942/difference-between-fill-and-expand-options-for-tkinter-pack-method>)
- Positionnement de widgets avec la méthode pack : (<https://koor.fr/Python/Tutoriel Tkinter/tkinter layout pack.wp#google vignet>)
- Notre Projet – Baptiste MICHEL et Nicolas THIERRY : (<https://github.com/Baptiste-micl/Projet-Jeux-Olympiques>)
- Python GUI's With TKinter | Codemy.com : (<https://youtube.com/playlist?list=PLCC34OHNcOtoC6GglhF3ncJ5rLwQrLGnV>)
- Tkinter.com | Youtube : (<https://www.youtube.com/channel/UCTsyH5vDt8RaRUD9ylhjX1w>)