

Roussel Alexis

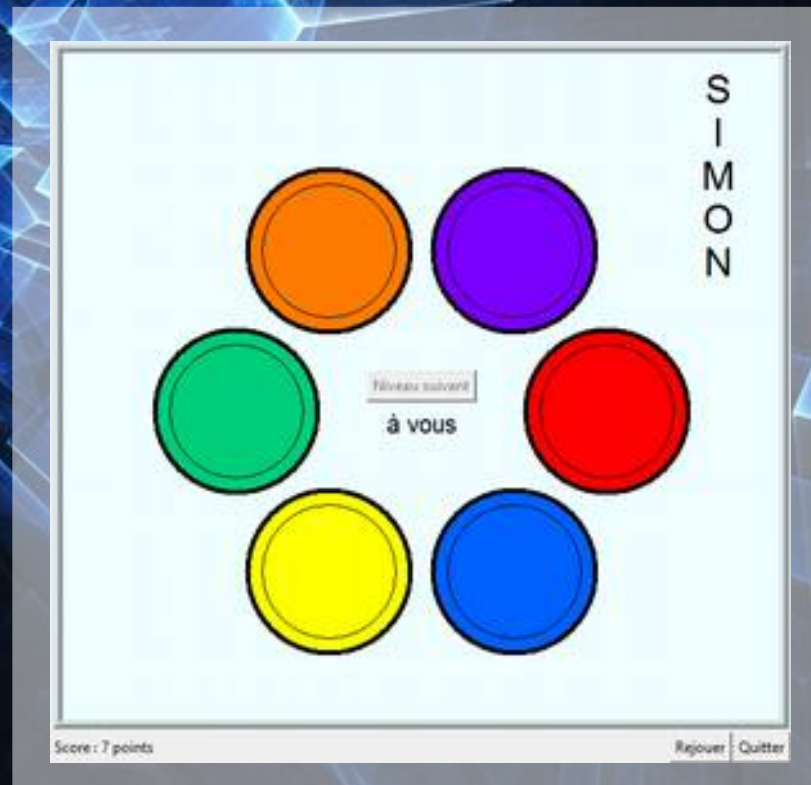


I.S.N.

Terminale S2

En groupe avec :
Vuillaume Alexandre
Helin Dorian

Projet final : Programmation d'un jeu de Simon



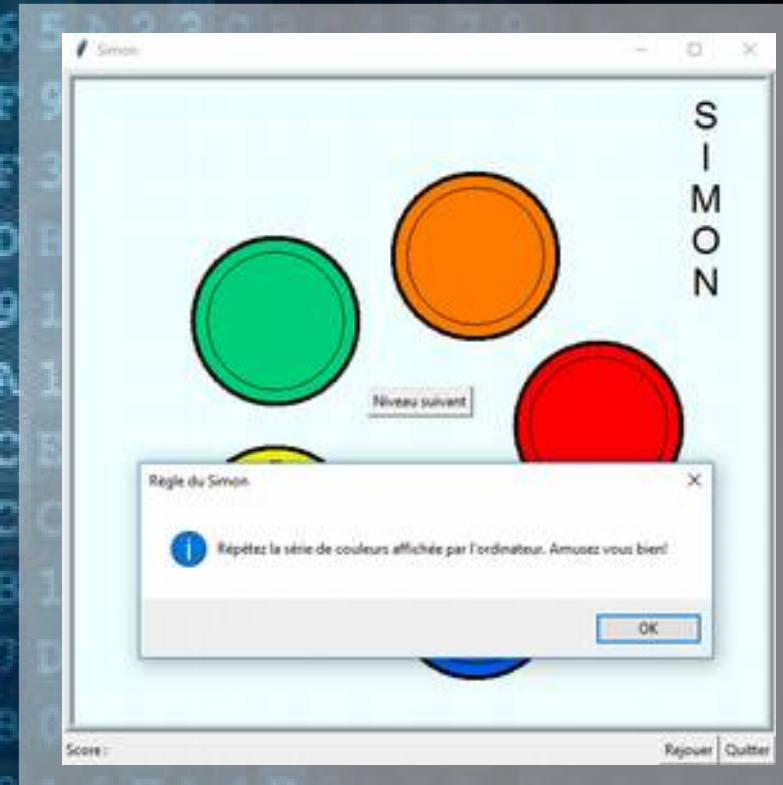
Sommaire

- **Description de notre projet**
 - Le jeu du Simon, qu'est-ce que c'est ?
 - Cahier des charges
- **Présentation du code de programmation**
 - Planning
 - Augmentation croissante de la difficulté
 - Affichage du score et possibilité de rejouer
- **Partie personnelle**
 - Affichage des interactions du joueur
 - Affichage et création de la série de couleur de l'ordinateur
 - Vérification de la suite du joueur
- **Difficultés rencontrées et améliorations possibles**
- **Conclusion personnelle**

Description de notre projet

Notre but :

réaliser à l'aide du logiciel python un jeu à la fois accessible, mais également amusant pour l'utilisateur. Cependant, nos connaissances étant limitées, notre choix s'est orienté vers un jeu avec des graphismes simples et des règles faciles à transcrire en programmation. C'est pourquoi nous nous sommes orientés vers la réalisation d'un jeu de type « Simon ».



Le jeu du Simon, qu'est-ce que c'est ?



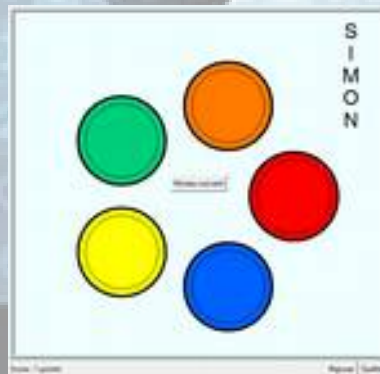
Amazon.fr

Le jeu du Simon est un jeu de mémoire, qui consiste à faire répéter au joueur une suite de couleurs que le jeu lui aura préalablement montré. L'objectif du jeu est de retenir la plus grande séquence possible de couleurs.

Cahier des charges

En plus des fonctions indispensables à notre Simon, nous voulions instaurer plusieurs difficultés et apporter différentes améliorations à notre jeu :

- augmenter la vitesse d'affichage des couleurs pour complexifier le jeu,
- augmenter le nombre de couleurs affichées en fonction du score pour varier la complexité,
- et nous avons également projeté de pouvoir jouer à plusieurs.



Exemple : passage de 5 à 6 couleurs au bout de 6 points

Présentation du code de programmation

```
DERNIERE VERSION.py - G:\DERNIERE VERSION.py (3.6.3)
File Edit Format Run Options Window Help

from tkinter import *
from tkinter.messagebox import showinfo
from random import choice
from math import sin
from math import cos
from math import pi

#Initialisation

Fenetre=Tk()
Fenetre.title("Simon")
c=150
r=81
points=0
Testpoints=True
Test=False
SolutionS=[]
SuiteJoueur=[]
couleur=10
erreur=0
b=10

def Difficulte(Testpoints):
    global ListeSombre, ListeClaire, j
    ListeSombre=['#ff0000','#0001ff','#ffff00','#000e7b','#ff7b00','#7b00ff','#0
    ListeClaire=['#db7093','#8ebdff','#ffffe6','#4682b4','#ffaa00','#b10c0f','#
    if points<5:
        j=5
    if points>5*(j-4) and j<9: # empêche d'obtenir j>9 et donc une erreur de
        j+=1
    Testpoints=True
    if Testpoints == True:
        zone_dessin.delete(ALL) # efface tout dans la zone de dessin
        Testpoints=False
        init()
```

Planning

Séance	Bilan / problèmes / travail réalisé	Objectifs
1ère séance	Recherche du projet	Réalisation du plateau de jeu
2ème séance	Mise en place du plateau de jeu, version carrée à quatre couleurs	développer la base
3ème séance	Mise en place des premiers boutons nécessaires (« quitter », « nouveau niveau », « rejouer »)	Visualisation de l'interaction du joueur (« clic » sur les disques)
4ème séance	Enregistrement dans la liste SuiteJoueur des actions du joueur sur les disques. Problèmes d'affichage : pas de visuel de l'action du joueur	Résoudre le problème d'affichage Création d'une séquence à reproduire
5ème séance	Utilisation du module <i>time</i> pour permettre l'affichage de l'action du joueur (ce module arrête le programme momentanément)	Prévision de différents niveau de difficulté Sur le conseil de notre professeur, préférer la méthode <i>after</i> au <i>time</i>
6ème séance	Changement d'interface pour permettre des niveaux de difficulté. Représentation sous formes de disques, jusqu'à neuf prévus Problème d'enregistrement des couleurs en cas de « clic » trop rapide	Affichage de la séquence à reproduire Limitation des actions du joueur pendant l'affichage
7ème séance	Création et affichage d'une séquence à reproduire. Problème quand à l'utilisation de la méthode <i>after</i> . L'affichage est correct dans la console, mais rien n'est visible dans la fenêtre graphique	Résoudre le problème d'affichage Fonction de vérification de la SuiteJoueur Comptage des points
8ème séance	Résolution des problèmes d'affichage avec la méthode <i>update</i> . Problème d'affichage de la séquence lorsque la même couleur est choisie à la suite (pas de visuel)	Affichage dans la fenêtre des points plutôt que dans la console Résoudre le nouveau problème d'affichage
9ème séance	Affichage dans la fenêtre résolu. Problème de réinitialisation avec le comptage des points	Résolution des problèmes survenus lors du comptage Modification de la vitesse d'affichage (difficulté croissante)
10ème séance	Comptage des points résolu	Élimination des variables inutiles Optimisation du code
11ème séance	Simplification des séquences d'affichage. L'affichage de la séquence à reproduire et des actions du joueur sur les disques utilisent les mêmes fonctions. Limitation de l'utilisation de variables (utilisation d'arguments dans les fonctions) Vitesse d'affichage de la séquence variable	Finaliser le programme

Nous avons décidé de gérer la partie graphique grâce à *Tkinter*, un module compris dans Python que nous avons déjà utilisé lors d'un autre projet. Structurellement, la programmation du jeu consiste essentiellement en plusieurs fonctions qui sont exécutées soit grâce à l'intervention de l'utilisateur, soit en remplissant des conditions prédéfinies.

```
from tkinter import *
from tkinter.messagebox import showinfo
from random import choice
from math import sin
from math import cos
from math import pi
```

```
#Initialisation
```

```
Fenetre=Tk()
Fenetre.title("Simon")
c=150
r=81
points=0
Testpoints=True
Test=False
SolutionS=[]
SuiteJoueur=[]
couleur=10
erreur=0
b=10
```

Voici quelques fonctions de notre programme qui vous seront expliquées plus précisément par mes camarades :

Augmentation croissante de la difficulté

```
def Difficulte(Testpoints):
    global ListeSombre, ListeClaire, j
    ListeSombre=['#ff0000','#0061ff','#ffff00','#00ce7b','#ff7b00','#7b00ff','#00ffa1','#707070','#9370db']
    ListeClaire=['#db7093','#8eb9ff','#ffffe6','#a4e8cc','#ffaa00','#b200ff','#91ffd6','#b2b2b2','#e791ff']
    if points<5:
        j=5
    if points>5*(j-4) and j<9:    # empêche d'obtenir j>9 et donc une erreur de liste (9 couleurs max)
        j+=1
        Testpoints=True
    if Testpoints == True:
        zone_dessin.delete(ALL)    # efface tout dans la zone de dessin
        Testpoints=False
        init()
```

```
def duree_affichage(n): # Augmentation de la difficulté durant la progression du jeu
    # Diminution du temps d'affichage de chaque couleur
    duree=int(30/n)    # la méthode after n'accepte qu'un entier (en millisecondes)
    if duree<1:
        duree=1
    return duree
```

Affichage du score et possibilité de rejouer

```
Affichscore=Label(Fenetre, text="Score :")
```

```
def Verification(n):    # Vérifie la suite réalisée par le joueur
    global points
    zone_dessin.unbind("<Button-1>") # désactiver la fonction attachée au clic {zone_dessin.bind("<Button-1>", fonction)}
    erreur=0
    for i in range(n):
        if SuiteJoueur[i]!=SolutionS[i]:
            erreur=erreur+1
    print(SuiteJoueur[i], "égale ou pas à", SolutionS[i])
    del (SuiteJoueur[:])
    if erreur==0:
        points+=1
        Affichscore.configure(text = 'Score : '+str(points)+' points') # pour affichage du nombre de points
        boutonniveau.config(state="normal")
    else:
        perdu()
        boutonniveau.config(state="disable")
```

```
def rejouer(): #Fonction qui relance le jeu
    global points
    points=0 # permettra de réinitialiser le score
    del (SolutionS[:])
    Affichscore.configure(text = 'Score : '+str(points)+' points') # initialise à zéro l'affichage des points
    boutonniveau.config(state="normal")
    Difficulte(Testpoints)
```

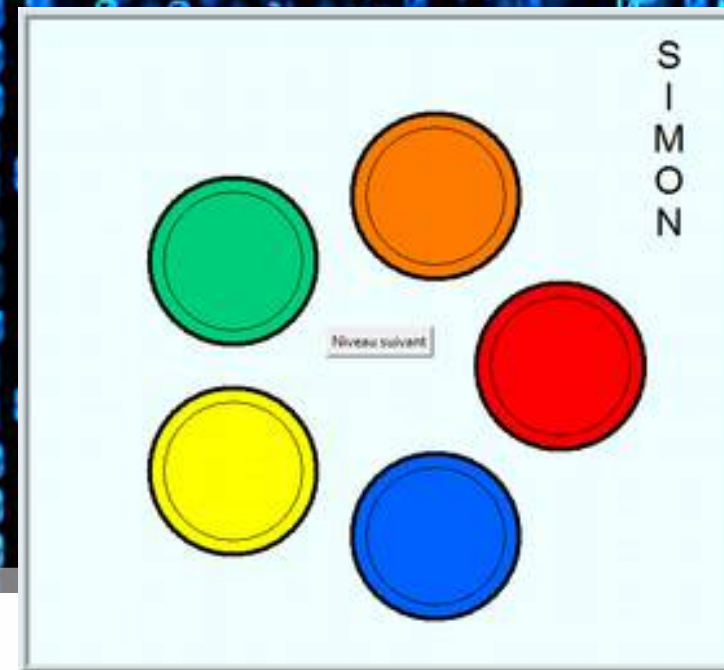


Partie personnelle

Pour ma part, je me suis particulièrement concentrer sur différentes parties :

- la création de la solution,
- l'affichage des changements de couleur dans la fenêtre d'affichage,
- la vérification de la série du joueur

Affichage des interactions du joueur



```
def fonction(evt): # Permet de mettre la couleur dans les cercles
    global elt, points
    elt=zone_dessin.find_withtag(CURRENT) # Prend l'élément sur lequel on a cliqué
    if 'grille' in zone_dessin.gettags(elt):# Vérifie si on a bien cliqué dans 'grille'
        couleur=zone_dessin.itemcget(elt,"fill") # Prend la couleur de l'élément désigné
        b=ListeSombre.index(couleur)
        print(couleur,' numero ',b,' Claire',ListeClaire[b],' Sombre',ListeSombre[b])
        duree=1
        zone_dessin.unbind("<Button-1>") # désactiver la fonction attachée au clic pour empêcher de prendre une autre couleur
        changecouleur(True,b,ListeClaire[b],duree*500) # le bouton prend sa couleur claire
        changecouleur(True,b,ListeSombre[b],duree*200) # retour à la couleur initiale
        zone_dessin.bind("<Button-1>",fonction) # réactiver la fonction attachée au clic
        SuiteJoueur.append(couleur) # ajoute une couleur choisie à la suite du joueur
    if len(SuiteJoueur)==len(SolutionS):
        zone_dessin.itemconfigure(invite,fill="") # cache le message "à vous"
        Verification(len(SuiteJoueur))
```

```
def changecouleur(Test,b,remplacecouleur, duree): # remplace par la couleur claire
    if Test==True:
        zone_dessin.itemconfigure(grille[b],fill=remplacecouleur) # Change la couleur de l'élément désigné
        zone_dessin.update() # On met à jour le canevas (sinon il ne se mettra à jour qu'à la fin de la fonction et on ne verra pas le changement)
        Test=False # Permet d'éviter une boucle infinie
        Fenetre.after(duree, changecouleur(Test,b,remplacecouleur, duree))
```



Affichage et création de la série de couleur de l'ordinateur

```
def Niveau():
    zone_dessin.unbind("<Button-1>") # désactiver la fonction fonction(evt)
    boutonniveau.config(state="disable") #on bloque les boutons le temps de l'affichage de la série
    Difficulte(False) # On donne à la variable "Testpoints" la valeur "Faux"
    creation_serie()
    lancement_affichserie()
```

```
def creation_serie(): # Création d'une solution de n couleurs
    global SolutionS, ListeSombre
    a=choice(ListeSombre[0:])
    SolutionS.append(a)
    print('longueur : ', len(SolutionS), "série de couleur à reproduire",SolutionS)
```

```
def lancement_affichserie():
    duree=duree_affichage(len(SolutionS))
    for i in range(len(SolutionS)):
        b=ListeSombre.index(SolutionS[i])
        changecouleur(True,b,ListeClaire[b],duree*50) # changement de la couleur en couleur claire
        changecouleur(True,b, ListeSombre[b],duree*50) # retour à la couleur initiale
        zone_dessin.update() # On met à jour le canevas (sinon il ne se mettra à jour qu'à la fin de la fonction et on ne verra pas le changement)
    zone_dessin.bind("<Button-1>",fonction) # réactiver la fonction attachée au clic (zone_dessin.bind("<Button-1>",fonction))
    zone_dessin.itemconfigure(invite,fill='black') # colore en noir "à vous" pour inviter à commencer la série
```

```
def changecouleur(Test,b,remplacecouleur, duree): # remplace par la couleur claire
    if Test==True:
        zone_dessin.itemconfigure(grille[b],fill=remplacecouleur) # Change la couleur de l'élément désigné
        zone_dessin.update() # On met à jour le canevas (sinon il ne se mettra à jour qu'à la fin de la fonction et on ne verra pas le changement)
        Test=False # Permet d'éviter une boucle infinie
        Fenetre.after(duree,changecouleur(Test,b,remplacecouleur,duree))
```

Vérification de la suite du joueur

```
def Verification(n):      # Vérifie la suite réalisée par le joueur
    global points
    zone_dessin.unbind("<Button-1>") # désactiver la fonction attachée au clic {zone_dessin.bind("<Button-1>",fonction)}
    erreur=0
    for i in range(n):
        if SuiteJoueur[i]!=SolutionS[i]:
            erreur=erreur+1
    print(SuiteJoueur[i],"égale ou pas à", SolutionS[i])
    del (SuiteJoueur[:])
    if erreur==0:
        points+=1
        Affichscore.configure(text = 'Score : '+str(points)+' points') # pour affichage du nombre de points
        boutonniveau.config(state="normal")
    else:
        perdu()
        boutonniveau.config(state="disable")
```

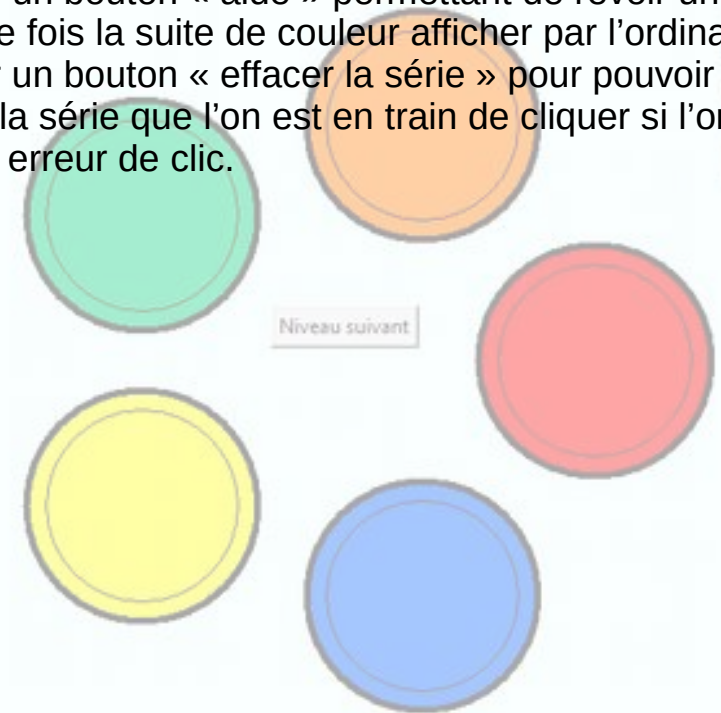
Difficultés rencontrées et améliorations possibles

Difficultés :

- comprendre les interactions provoquées par le programme ou l'utilisateur lors de l'exécution des fonctions, ce qui a pu bloquer la progression du projet,
- Problème d'affichage dans la fenêtre d'affichage, qui ont en particulier été longs à corriger,
- Problème de positionnement des disques,
- La mise en commun de toutes les parties de code a aussi posé de nombreux problèmes inattendus, qu'il a fallu comprendre avant de pouvoir les corriger.

Améliorations :

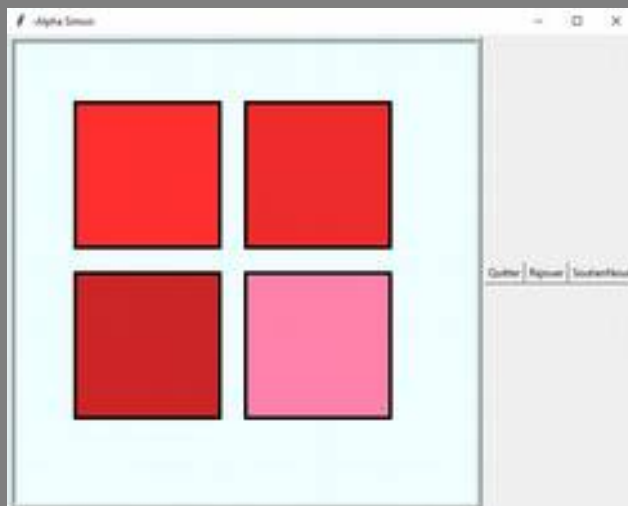
- ajouter une fonctionnalité pour jouer à plusieurs,
- ajouter un bouton « aide » permettant de revoir une seconde fois la suite de couleur à afficher par l'ordinateur,
- ajouter un bouton « effacer la série » pour pouvoir effacer la série que l'on est en train de cliquer si l'on a fait une erreur de clic.



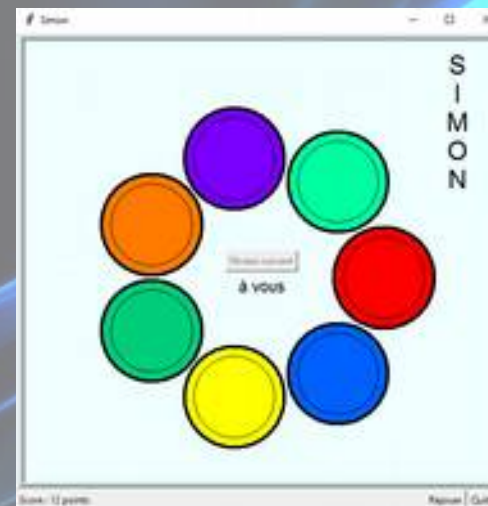
Conclusion personnelle

Personnellement ce projet m'a permis de découvrir le travail collaboratif sur un projet informatique, et les difficultés y afférant et ses avantages.

De plus ces échanges ont aussi permis de voir émerger des idées nouvelles provenant de l'addition de nos réflexions.



Première version du jeu



Dernière version du jeu