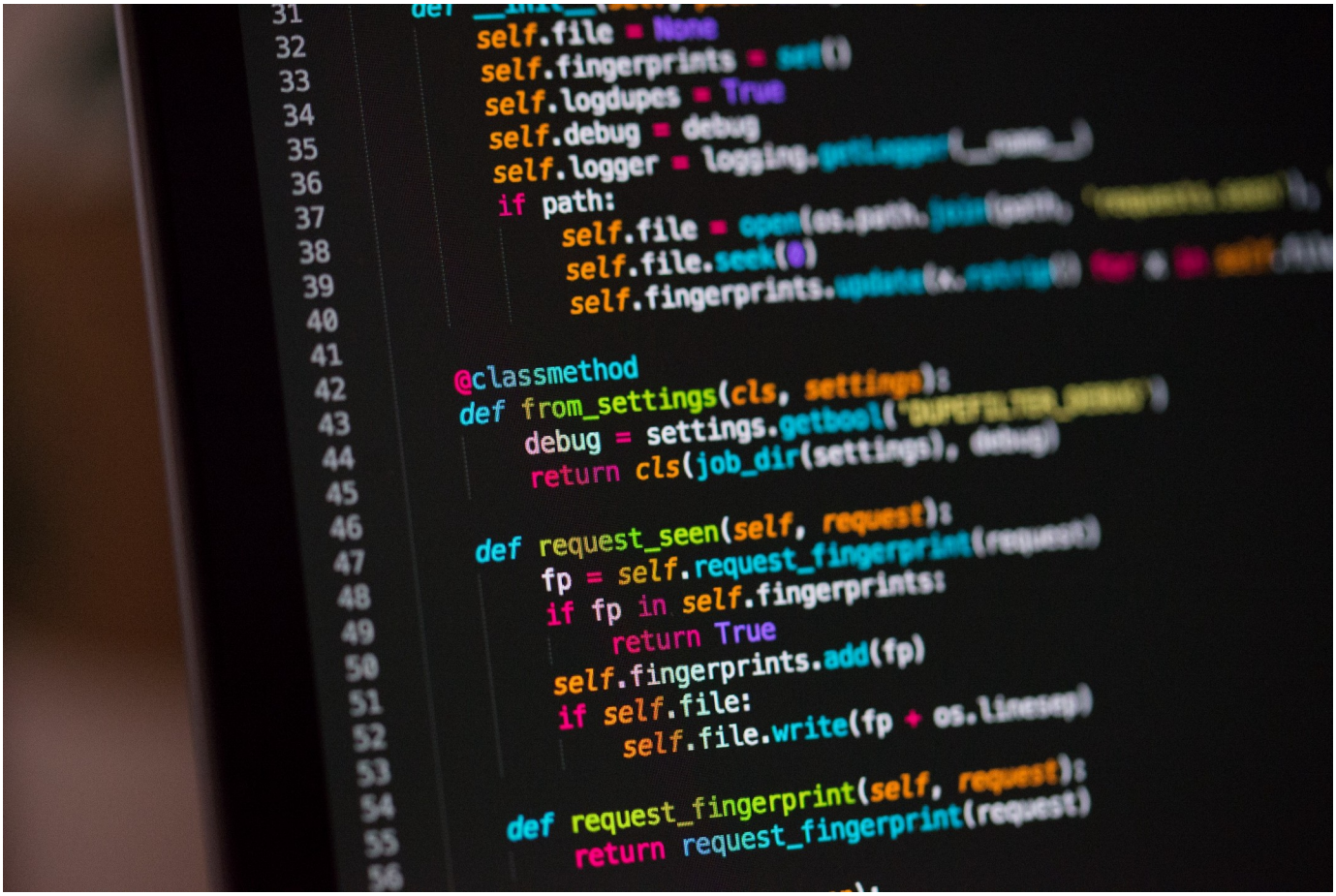
VERGEROLLE
Loicq
1G6

Dossier Technique

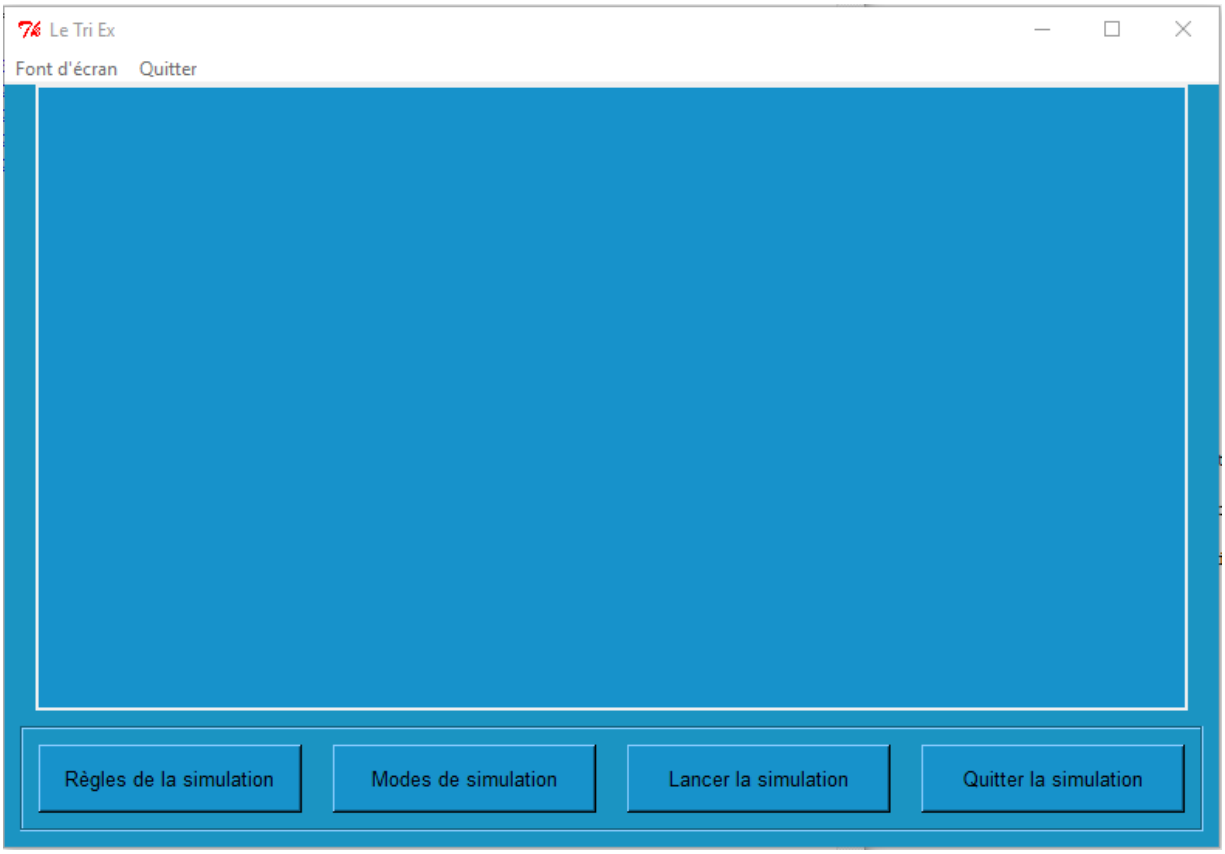
Le Tri Ex
(une simulation de Tri de cartes)



Sommaire

Description du mode console

Durant cette année de Terminal nous avons développé un programme informatique nommé « Le Tri Ex » dans le cadre de notre projet. Nous avons été deux dans la réalisation de ce projet, Lucas LAFONT et moi. Notre professeur nous à aidé à choisir le déroulement du programme, et avec les divers cours étudié en cours nous avons pu aboutir à une partie de la finalisation du programme.



Le « Le Tri Ex » est une simulation de tri de cartes, elle affiche un nombre de cartes pour ensuite les trier suivant le mode choisit préalablement par l'utilisateur.

Le programme met à disposition plusieurs modes de tri :

- le tri croissant
- le tri décroissant
- le tri à bulle
- le tri par insertion

Le tri croissant :

Le tri croissant consiste à sélectionner des cartes dans la liste pour ensuite les replacer dans l'ordre croissant suivant l'algorithme que python possède (la fonction sort()).

Nous avons dans un premier tester cette fonction sur le mode console de python, afin de visualiser son exécution. Ce type de méthode est très simpliste, on peut le voir sur cette capture d'écran.

```
>>> liste = {1,9,0,2,4,2,5}
>>> sorted(liste)
[0, 1, 2, 4, 5, 9]
```

Le Tri décroissant :

Le tri décroissant consiste à sélectionner des cartes dans la liste pour ensuite les replacer dans l'ordre croissant suivant l'algorithme que python possède (la fonction sort()). Une fois que les valeurs sont dans l'ordre croissant nous avons juste inverser l'ordre emplacement avec une autre fonction de python qui est reverse() et tout ceci sur console et non graphique. Puisque au niveau graphique il faut gérer le déplacement des cartes et pas juste des chiffres.

```
>>> liste = {1,9,0,2,4,2,5}
>>> a = sorted(liste)
>>> a.reverse()
>>> a
[9, 5, 4, 2, 1, 0]
```

Mais ces deux types de tri ne nous permettait pas de comprendre comment fonctionne réellement l'algorithme puisqu'il est déjà codé par python. Nous avons donc chercher différent type de tri et retranscrit leurs algorithmes en langage python. Dans un premier temps nous avons choisit la méthode du tri à bulle et ensuite une autre méthode qui est le tri par insertion.

Le Tri à bulle :

Le tri à bulle consiste à comparer la valeur de deux cartes côte à côte pour ensuite déplacer la plus petite valeur vers la gauche. Ceci afin d'obtenir un ordre croissant des cartes qui sont initialement mélangées. Ce mode de tri n'est pas le plus efficace puisqu'il arrive à terme au bout de plusieurs minutes. Mais nous avons pu trouver un algorithme pour ce mode de tri que nous avons retranscrit ensuite en python.

L’algorithme ce présente de cette façon:

```
PROCEDURE tri_bulle ( TABLEAU a[1:n])
passage ← 0
REPETER
    permut ← FAUX
    POUR i VARIANT DE 1 A n - 1 - passage FAIRE
        SI a[i] > a[i+1] ALORS
            echanger a[i] ET a[i+1]
            permut ← VRAI
        FIN SI
    FIN POUR
    passage ← passage + 1
TANT QUE permut = VRAI
```

Nous l’avons modifié pour qu’il puisse s’adapter à nos besoins vu que nous avons fait deux versions de celui-ci, une version sous mode console (squelette du code) et une autre version sous mode graphique(qui est la finalisation de notre code).

Sous forme console nous avons réussi à insérer le tri à bulle à notre programme ce qui donne la capture d’écran si dessous :

```
LE TRI A BULLE :

Variable comparé : 11 < 13
Variable comparé : 13 > 10
[11, 10, 13, 12, 6, 4, 1, 2, 7, 8, 5, 9, 3]
Variable comparé : 13 > 12
[11, 10, 12, 13, 6, 4, 1, 2, 7, 8, 5, 9, 3]
Variable comparé : 13 > 6
[11, 10, 12, 6, 13, 4, 1, 2, 7, 8, 5, 9, 3]
Variable comparé : 13 > 4
[11, 10, 12, 6, 4, 13, 1, 2, 7, 8, 5, 9, 3]
Variable comparé : 13 > 1
[11, 10, 12, 6, 4, 1, 13, 2, 7, 8, 5, 9, 3]
Variable comparé : 13 > 2
[11, 10, 12, 6, 4, 1, 2, 13, 7, 8, 5, 9, 3]
Variable comparé : 13 > 7
[11, 10, 12, 6, 4, 1, 2, 7, 13, 8, 5, 9, 3]
```

Les deux premières valeurs sont comparées et échangées si la première est plus petite que la deuxième.

Le tri par insertion:

A la suite des tri croissant et décroissant aux fonctionnement encore incompris nous avons choisit d'utiliser l'algorithme du tri par insertion disponible partout sur internet. Ce mode tri consiste à comparer deux valeurs de deux cartes pour ensuite placer la carte essayant le plus petite valeur à gauche. Ce processus est répété comme si l'on avait utilisé le tri par insertion à la seule différence que la carte choisis est comparé par toutes les cartes la précédant.

Nous avons choisi de programmer notre code suivant ce modèle d'algorithme :

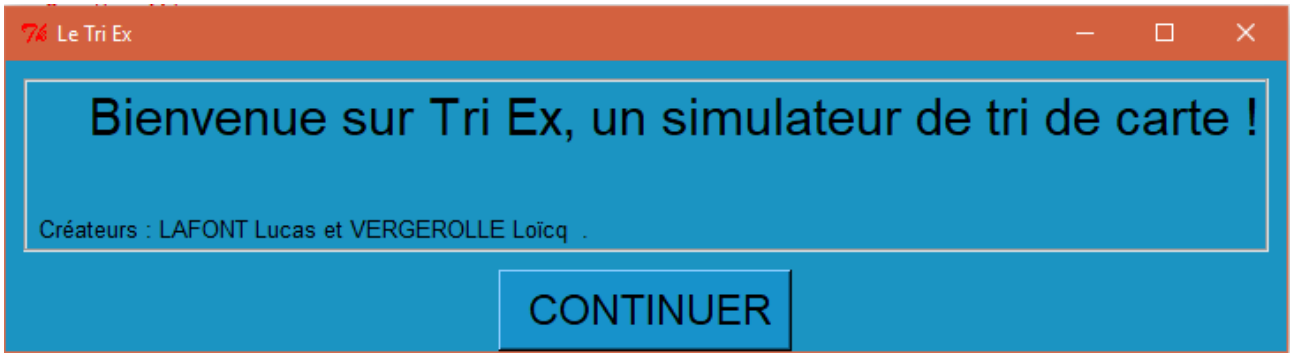
```
PROCEDURE tri_Insertion ( Tableau a[1:n])
    POUR i VARIANT DE 2 A n FAIRE
        INSERER a[i] à sa place dans a[1:i-1];
FIN PROCEDURE;
```

Ce qui à donner sous version console :

```
 LE TRI PAR INSERTION :


La comparaison est : 4 < 6
[4, 6, 10, 9, 13, 2, 5, 1, 11, 7, 8, 12, 3]
La comparaison est : 6 < 10
[4, 6, 10, 9, 13, 2, 5, 1, 11, 7, 8, 12, 3]
La comparaison est : 10 > 9
[4, 6, 10, 9, 13, 2, 5, 1, 11, 7, 8, 12, 3]
La comparaison est : 6 < 9
[4, 6, 10, 10, 13, 2, 5, 1, 11, 7, 8, 12, 3]
La comparaison est : 10 < 13
[4, 6, 9, 10, 13, 2, 5, 1, 11, 7, 8, 12, 3]
La comparaison est : 13 > 2
[4, 6, 9, 10, 13, 2, 5, 1, 11, 7, 8, 12, 3]
La comparaison est : 10 > 2
[4, 6, 9, 10, 13, 13, 5, 1, 11, 7, 8, 12, 3]
La comparaison est : 9 > 2
[4, 6, 9, 10, 10, 13, 5, 1, 11, 7, 8, 12, 3]
La comparaison est : 6 > 2
[4, 6, 9, 9, 10, 13, 5, 1, 11, 7, 8, 12, 3]
La comparaison est : 4 > 2
[4, 6, 6, 9, 10, 13, 5, 1, 11, 7, 8, 12, 3]
La comparaison est : 3 < 2
[4, 4, 6, 9, 10, 13, 5, 1, 11, 7, 8, 12, 3]
La comparaison est : 13 > 5
[2, 4, 6, 9, 10, 13, 5, 1, 11, 7, 8, 12, 3]
La comparaison est : 10 > 5
[2, 4, 6, 9, 10, 13, 13, 1, 11, 7, 8, 12, 3]
La comparaison est : 9 > 5
[2, 4, 6, 9, 10, 10, 13, 1, 11, 7, 8, 12, 3]
La comparaison est : 6 > 5
[2, 4, 6, 9, 9, 10, 13, 1, 11, 7, 8, 12, 3]
```

Description du mode graphique:

Dans un premier temps nous avons créé une fenêtre de présentation de la simulation qui sert à accéder au programme principale.



Détaille du code (python):

Afin de réaliser cette fenêtre nous avons utilisé une fonction menu() qui se compose :
- d'un Frame (un cadre qui entoure)
- de deux Labels (qui est le texte afficher dans le cadre)
- un bouton (qui permet de lancer le programme principal)

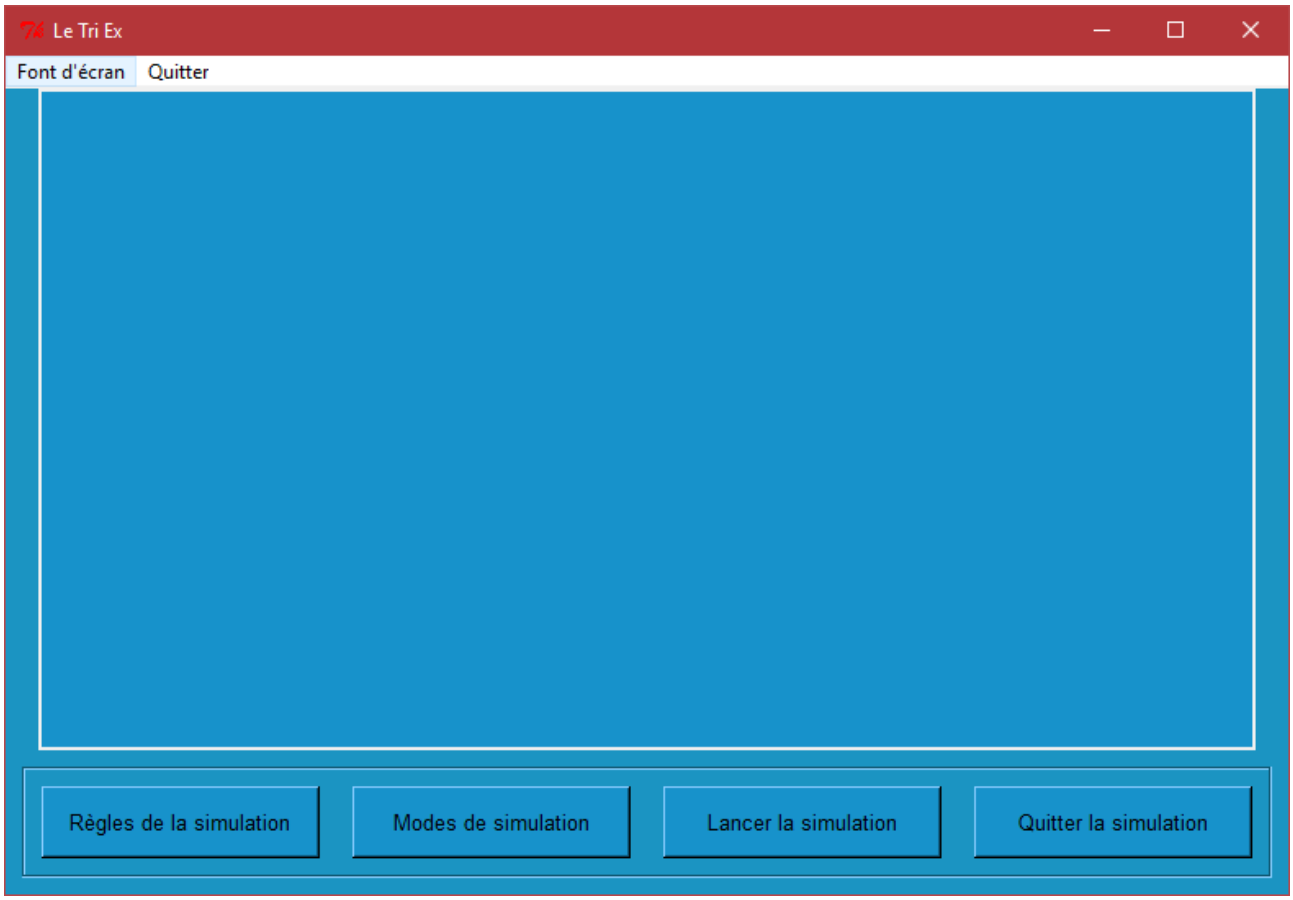
Le code que l'ont à fait pour cette partie est :

```
def lancer():
    ### Menu
    choix = ''
    #Creation d'un frame pour le texte principale
    frame1 = Frame(ui, bg = "white", borderwidth = 2, relief = GROOVE)
    frame1.pack(padx = 10, pady = 10)

    texte = "1 Lire les règles de la simulation \n" +\
            "2 Choix des différents mode de la simulation \n" +\
            "3 Lancer la partie\n" +\
            "4 Quitter la simulation"
    label1 = Label(frame1, text = "Bienvenue sur Tri Ex, un simulateur de tri de carte !\n   ", font = ("Purisa", 22), bg ="#1B94C2")
    label1.pack()
    label2 = Label(frame1, text = " Créateurs : LAFONT Lucas et VERGEROLLE Loïcq  .
    label2.pack(side = LEFT)
    #label2 = Label(frame2, text = texte, font = ("Purisa", 18), bg = "#1B94C2")
    #label2.pack()
    BoutonLancer = Button(ui, text = " CONTINUER", font = ("Purisa", 17), borderwidth = 2, bg = "#1792CB", width = 12, height = 1, command = menu)
    BoutonLancer.pack()
    ui.mainloop()
```

…

Menu principale du logiciel

Nous avons ensuite dû programmer la partie principale de la simulation qui permet à l'utilisateur de lancer la simulation mais aussi de lire les règles d'utilisation du logiciel, de choisir le mode de simulation voulut et pour finir de quitter l'application.



Détaille du code (python) :

Pour programmer cette interface nous avons utilisé:

- deux Frame (cadres)
- quatre Button (bouton avec des caractéristiques choisis spécifiquement, chacun des quatre bouton active une nouvelle fonction dans le programme qui exécute le code de la partie choisi)
- une barre de menu (qui contient des options sur la couleur du cadre centrale)

Le code que l'ont à fait pour cette partie est :

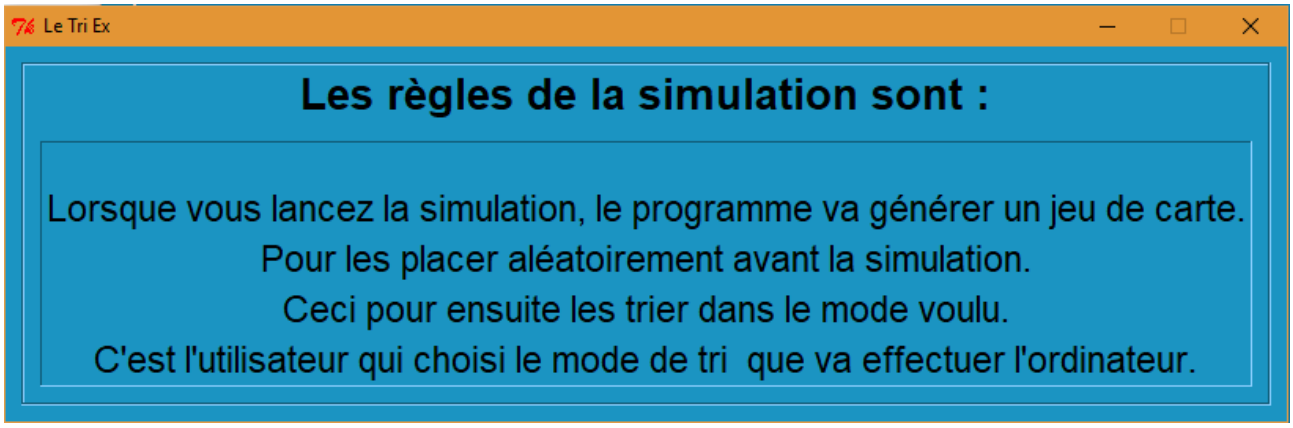
```
def menu():
    global uk, ui, fond, Bouton1, Bouton2, Bouton3, Bouton4
    ui.destroy()
    uk = Tk()
    uk.title("Le Tri Ex")
    uk.configure(bg = "#1B94C2")

    # Image de fond

    fond = Canvas(uk, width = 740, height = 400, bg = "#1792CB")
    fond.pack()
    # Frame et boutons...
    frame1 = Frame(uk, bg = "#1B94C2", borderwidth = 2, relief = GROOVE)
    frame1.pack(side = BOTTOM, padx = 10, pady = 10)

    def fond_bleu():
        global fond
        global Bouton1, Bouton2, Bouton3, Bouton4
        fond.destroy()
        Bouton1.destroy()
        Bouton2.destroy()
        Bouton3.destroy()
        Bouton4.destroy()
        buttons()
        fond = Canvas(uk, width = 740, height = 400, bg = "#2AA9BF")
        fond.pack(side = TOP)


    def fond_rouge():
        global Bouton1, Bouton2, Bouton3, Bouton4
        global fond
        fond.destroy()
        Bouton1.destroy()
        Bouton2.destroy()
        Bouton3.destroy()
        Bouton4.destroy()
        fond = Canvas(uk, width = 740, height = 400, bg = "#E7524B")
        fond.pack(side = TOP)
        buttons()

    def fond_vert():
        global fond
        global Bouton1, Bouton2, Bouton3, Bouton4
        fond.destroy()
        Bouton1.destroy()
        Bouton2.destroy()
        Bouton3.destroy()
        Bouton4.destroy()
        fond = Canvas(uk, width = 740, height = 400, bg = "#38AA21")
        fond.pack(side = TOP)
        buttons()


    #Menu bar..
    menubar = Menu(uk)
    menufichier = Menu(uk,tearoff=0)
    menufichier.add_command(label = "Fond bleu", command  = fond_bleu)
    menufichier.add_command(label = "Fond rouge", command = fond_rouge)
    menufichier.add_command(label = "Fond vert", command = fond_vert)
    menubar.add_cascade(label="Font d'écran", menu= menufichier)


    menubar.add_cascade(label = "Quitter", command = uk.destroy)

    uk.config(menu = menubar)
```

...

Les différentes fonctions du logiciel et options

Le bouton (Règle de la simulation) :

Nous avons réfléchis à une façon d'informer l'utilisateur du fonctionnement du logiciel. Pour palier à cette problématique nous avons créer un bouton (Règle de la simulation) qui ouvre une fenêtre expliquent le fonctionnement global du logiciel.



Le code utilisé pour créer cette fenêtre n'a pas été compliqué, il se compose : de deux Frames(cadres) et de plusieurs labels dont le premier est le titre de la fenêtre et les autres le texte explicatif.

Voici une partie du code qui à servi à cette fenêtre :

```
def règle_du_jeu():
    uk = Tk()
    uk.title("Le Tri Ex")
    uk.configure(bg = "#1B94C2")
    uk.resizable(width = False, height = False)
    frame1 = Frame(uk, bg = "#1B94C2", borderwidth = 2, relief = GROOVE)
    frame1.pack(padx = 10, pady = 10)

    frame2 = Frame(frame1, bg = "#1B94C2", borderwidth = 1, relief = GROOVE)
    frame2.pack(side = BOTTOM, padx = 10, pady = 10)

    texte1 = ("Les règles de la simulation sont :")
    texte2 = ("\nLorsque vous lancez la simulation, le programme va générer un jeu de carte.")
    texte3 = ("Pour les placer aléatoirement avant la simulation.")
    texte4 = ("Ceci pour ensuite les trier dans le mode voulu.")
    texte5 = ("C'est l'utilisateur qui choisi le mode de tri  que va effectuer l'ordinateur.")

    label1 = Label(frame1, text = texte1, font = ("Purisa", 20, 'bold'), bg = "#1B94C2")
    label2 = Label(frame2, text = texte2, font = ("Purisa", 17), bg = "#1B94C2")
    label3 = Label(frame2, text = texte3, font = ("Purisa", 17), bg = "#1B94C2")
    label4 = Label(frame2, text = texte4, font = ("Purisa", 17), bg = "#1B94C2")
    label5 = Label(frame2, text = texte5, font = ("Purisa", 17), bg = "#1B94C2")
    label1.pack(side = TOP)
    label2.pack()
    label3.pack()
    label4.pack()
    label5.pack()


    uk.mainloop()
```

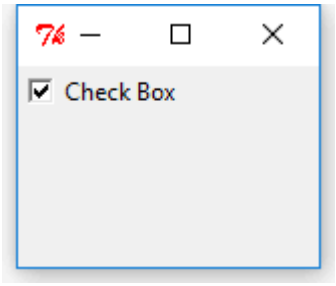
Le bouton (Mode de simulation) :

Pour que l'utilisateur puisse choisir le mode de simulation qu'il souhaite voir, nous avons créer une fenêtre permettant de donner à l'utilisateur le choix entre :

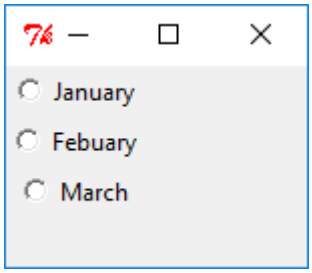
- Tri croissant
- Tri décroissant
- Tri à bulle
- Tri par insertion

Une fois que l'utilisateur à fait son choix, une variable l'enregistre. Elle est ensuite mise à disposition dans tout le programme afin de l'utiliser durant l'utilisation du logiciel.

Pour la partie programmation, nous avons dans un premier temps essayé d'utiliser des « Checkbutton », puis des « Radiobutton » pour finalement choisir des simples bouton moins compliquer d'utilisation. En effet pour les « Checkbutton » nous avons pas réussi à les associer au programme suite à de nombreux Bugs et problèmes de l'émulateur IDLE, de même pour les « Radiobutton ».

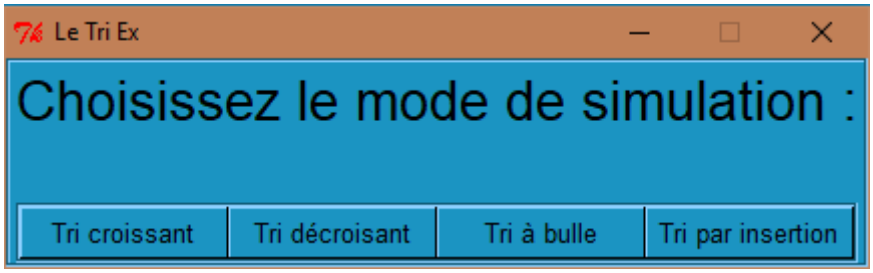


Exemple de : Checkbutton



Exemple de : Radiobutton

Voici la fenêtre illustrant les différents modes de simulation :



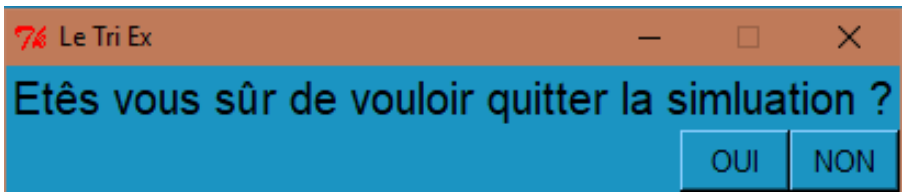
Le code permettant l'exécution de cette fenêtre à été un première partie compliqué mais simplifier par l'utilisation des boutons au lieu de « Checkbutton » et « Radiobutton ». Cette partie de code ce compose de deux Frames (cadres), avec un Label qui est le titre de cette fenêtre et quatre boutons.

Le code (la première partie):

```
def mode_simulation():
    global mode

    ib = Tk()
    ib.title("Le Tri Ex")
    ib.configure(bg = "#1B94C2")
    ib.resizable(width = False, height = False)

    frame1 = Frame(ib, bg = "#1B94C2", borderwidth = 2, relief = GROOVE)
    frame2 = Frame(frame1, bg = "#1B94C2", borderwidth = 2, relief = GROOVE)
    frame1.pack()
    frame2.pack(side = BOTTOM)

    label1 = Label(frame1, text = "Choisissez le mode de simulation :\n", font = ("Purisa", 20), bg = "#1B94C2")
    label1.pack()

    # Button

    def Button1():
        print("Vous avez choisi : le tri croissant !")
        choix = "tri croissant"
        recuperation_du_choix(choix)

    def Button2():
        print("Vous avez choisi : le tri décroissant !")
        choix = "tri décroissant"
        recuperation_du_choix(choix)

    ...
```

Le bouton (Quitter la simulation) :

Nous aussi créé un bouton quitter la simulation pas indispensable mais néanmoins utile car il demande une confirmation de la part de l'utilisateur, (oui ou non je souhaite quitter la simulation).



Pour cette fenêtre nous avons utilisé dans le code : un label (pour la question), et deux autres boutons (pour la confirmation de quitter l'application).

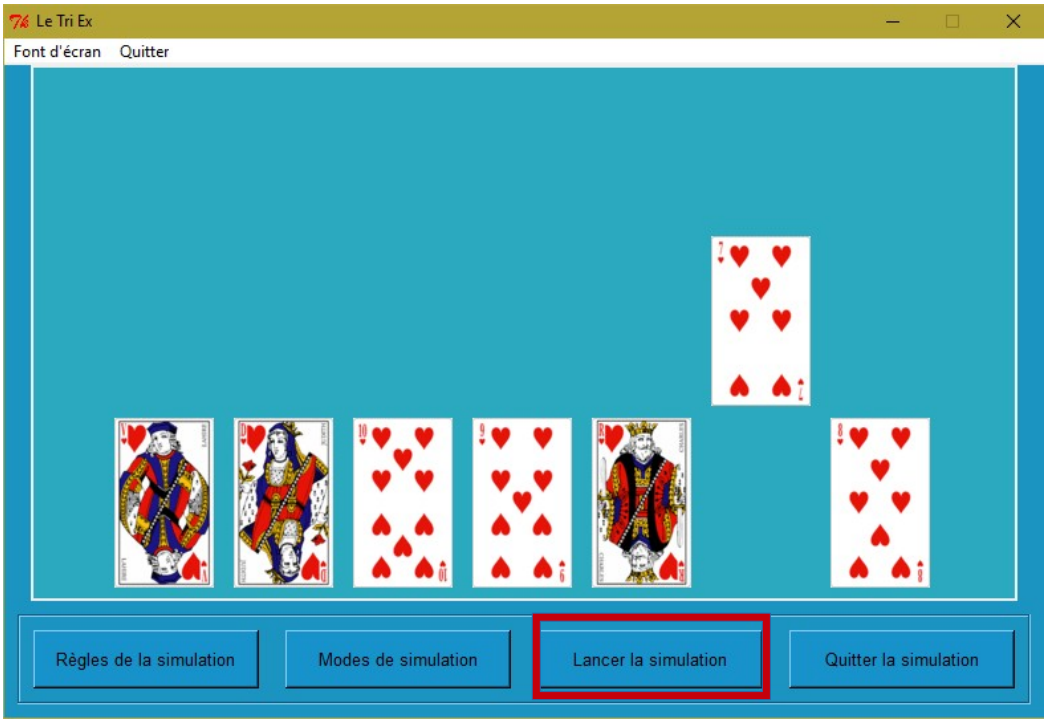
Voici le code de cette partie :

```
def quitter():
    global uk
    sur = Tk()
    sur.title("Le Tri Ex")
    sur.configure(bg = "#1B94C2")
    sur.resizable(width = False, height = False)
    label = Label(sur, text = "Etês vous sûr de vouloir quitter la simluation ?", font = ("Purisa", 15), bg = "#1B94C2")
    label.pack()

    def quitt():
        global uk
        uk.destroy()
        sur.destroy()

    def reste():
        sur.destroy()

    bouton1 = Button(sur, text = "OUI", font = ("Purisa", 10), bg = "#1B94C2", borderwidth = 2, width = 5, height = 1, command = quitt)
    bouton2 = Button(sur, text = "NON", font = ("Purisa", 10), bg = "#1B94C2", borderwidth = 2, width = 5, height =1, command = reste)
    bouton2.pack(side = RIGHT)
    bouton1.pack(side = RIGHT)
    sur.mainloop()
```

...

Le bouton (Lancer la simulation) :

Il nous ensuite fallut créer un bouton pour lancer la simulation. Celle ci ce compose en plusieurs parties : dans un premier le bouton « Lancer la simulation » affiche côte à côte et aléatoirement 8 cartes d'un jeu de cartes, ensuite en fonction du mode préalablement choisit par l'utilisateur l'application tri de façon soit : croissant, décroissant, avec l'algorithme à bulle ou avec l'algorithme par insertion.



Concrètement nous avons réalisé le partie tri croissant du programme, mais il nous à manqué les autres partie puisque le code demandé pour les effectuer nécessitait un algorithme particulier associant les déplacements cartes aux différents algorithme de tri, le tri à bulle et le tri par insertion.

Voici le code de la fonction « Lancer la simulation » :

```
def jouer():
    global mode, ui, fond, y, save1_x, save2_x, save3_x, save4_x, save5_x, save6_x, save7_x, x1, x2, x3, x4, x5, x

    # Initialisation des coordonnées

    x1 = 100
    x2 = 190
    x3 = 280
    x4 = 370
    x5 = 460
    x6 = 550
    x7 = 640
    y = 330

    #Met aléatoirement les images dans le cadre...
    i = 7
    l  = [x1, x2, x3, x4, x5, x6, x7]
    while i <= 13:

        x = random.choice(l)
        print("la variable choisi est : {} ".format(x))
        print("La valeur de la variable que l'on supprime : {}".format(l.index(x)))
        del l[l.index(x)]



        ##### Affiche les cartes : 7, 8, 9, 10, valet, Dame, roi . En fonction de la valeur de x choisit aléatoire
        if i == 7:
            img1 = PhotoImage(file = r'D:\Users\lycée-loicq\Desktop\Cours_ISN\Projet_Le_Trie_Ex\{}.gif'.format(i))
            image1 = fond.create_image(x, y, image = img1)
            save1_x = x

...
```

Conclusion :

Ce projet à été un des plus grand projet informatique que j’ai réalisé et l’un des plus compliqué. Il m’a permit d’acquérir une certaine aisance en informatique.

Annexes :

```
#!/bin/usr/env python3
# -*- coding : utf-8 -*-

from tkinter import *
from tkinter.messagebox import *
import random
import time

###### TOUTES les fonctions #####

def lancer():
    ### Menu
    choix = ''
    #Creation d'un frame pour le texte principale
    framel = Frame(ui, bg = "white", borderwidth = 2, relief = GROOVE)
    framel.pack(padx = 10, pady = 10)

    texte = "1 Lire les règles de la simulation \n" +\
           "2 Choix des différents mode de la simulation \n" +\
           "3 Lancer la partie\n" +\
           "4 Quitter la simulation"
    label1 = Label(framel, text = "Bienvenue sur Tri Ex, un simulateur de tri de carte !    \n  ", font = ("Purisa", 22), bg ="#1B94C2")
    label1.pack()
    label2 = Label(framel, text = " Créateurs : LAFONT Lucas et VERGEROLLE Loïcq  .
    label2.pack(side = LEFT)
    #label2 = Label(frame2, text = texte, font = ("Purisa", 18), bg = "#1B94C2")
    #label2.pack()
    BoutonLancer = Button(ui, text = " CONTINUER", font = ("Purisa", 17), borderwidth = 2, bg = "#1792CB", width = 12, height = 1, command = menu)
    BoutonLancer.pack()
    ui.mainloop()

def menu():
    global uk, ui, fond, Boutonl, Bouton2, Bouton3, Bouton4
    ui.destroy()
    uk = Tk()
    uk.title("Le Tri Ex")
    uk.configure(bg = "#1B94C2")
    uk.resizable(width = False, height = False)
    # Image de fond


        ...

    def gestionnaire(event):
        global x_inity
        if event.keysym == 'Left':
            x -= 10
        elif event.keysym == 'Right':
            x += 10
        else:
            x += 0
        jardin.coords(f_window, x, y)

    #jardin.bind('<Key>', gestionnaire)

    ##############


    if __name__ == "__main__":
        mode = 0
        var = ''
        mode_default = True
        ################################
        #Initialisation du module tkinter

        ui = Tk()
        ui.title('Le Tri Ex')
        ui.configure( bg = "#1B94C2" )
        ui.resizable(width = False, height = False)
        '''a = ui.winfo_screenwidth()
        b = ui.winfo_screenheight()
        jardin = Canvas(ui, width = a, height= b, bg='palegreen')
        jardin.pack()'''

        ###### Deplacement de la souris
        ui.bind('<Motion>', deplacement_souris)

        ####Lancement des fonctions
        principal()
```

PARTIE GRAPHIQUE

```
#!/usr/bin/env python3
#! -*- coding: utf-8 -*-

import os
import sys
import random
import time
from random import *

###### LES DEFINITIONS

def menu():
    print("\n" * 100)
    menu = " Bienvenue sur Trie-ex, un jeu qui défi vos facultées mentale !\n"+\
           "Cette platforme vous offre le choix entre :\n" +\
           "1 Lire les règles de la simulation \n" +\
           "2 Choix des différents mode de la simulation \n" +\
           "3 Lancer la partie\n" +\
           "4 Quitter la simulation"
    choix = 0
    try:
        while choix != 1 and choix != 2 and choix != 3 and choix != 4:
            print(menu)
            choix = int(input("\nEntrez l numéro de votre choix : "))
    except ValueError:
            print("\n" * 50 ,"Vous devez rentrez un chiffre entre 1 et 4 !")
            wait()
    return choix

def règle_du_jeu():
    regle = ("\n\n\n## --Les explication de la simulation :\n\n Lorsque vous lancer la simulation, le progra
    print(regle)
    touch = input("\nAppuyez sur une touche...")

def choix_des_modes_de_jeu():
    #on desactive le mode par default \ mode_default = True
    global mode_default
    mode = ''
    if mode_default == True:
        mode_default = False
```

•••

```
        i += 1
    print("Les valeurs des cartes : {}".format(new_a))
    wait()
    # inseration des valeurs des cartes dans les fonctions de triages
    tableau = new_a
    if mode == "trie_croissant":
        tri_croissant(tableau)
    elif mode == "trie_decroissant":
        tri_decroissant(tableau)
    elif mode == "trie_bulle":
        print("\n" *50,"LE TRI A BULLE : \n\n")
        tri_bulle(tableau)
    elif mode == "trie_insertion":
        print("\n" *50,"LE TRI PAR INSERTION : \n\n")
        tri_insertion(tableau)


def wait():
    touch = input("\nAppuyez sur une touche...")
    return touch

def principal():
    global repeat, mode
    repeat = False
    while repeat == False:
        choix = menu()
        if choix == 1:
            règle_du_jeu()
        elif choix == 2:
            mode = choix_des_modes_de_jeu()
        elif choix == 3:
            jouer()
        elif choix == 4:
            repeat = True

if __name__ == "__main__":
    mode = 0
    mode_default = True

    principal()
```

PARTIE CONSOLE