EXPLOSION DE VOITURES !

Règles du jeu :

C'est un jeu solitaire, le joueur joue contre l'ordinateur. C'est la course ! Le but du jeu est de finir la course, le dernier à pouvoir prendre des allumettes est le gagnant. Ici, nos voitures sur la ligne de départ représentent nos allumettes, mais attention, nous allons les faire exploser !

Le joueur doit jouer en suivant des règles, effectivement, celui-ci ne peut pas tirer le nombre d'allumettes qu'il veut, mais un certain nombre, parmi une règle, rentrée dans une liste. Par exemple, la règle r[1,2,3], lui permet de retirer 1, 2 ou 3 allumettes, ni plus ni moins.

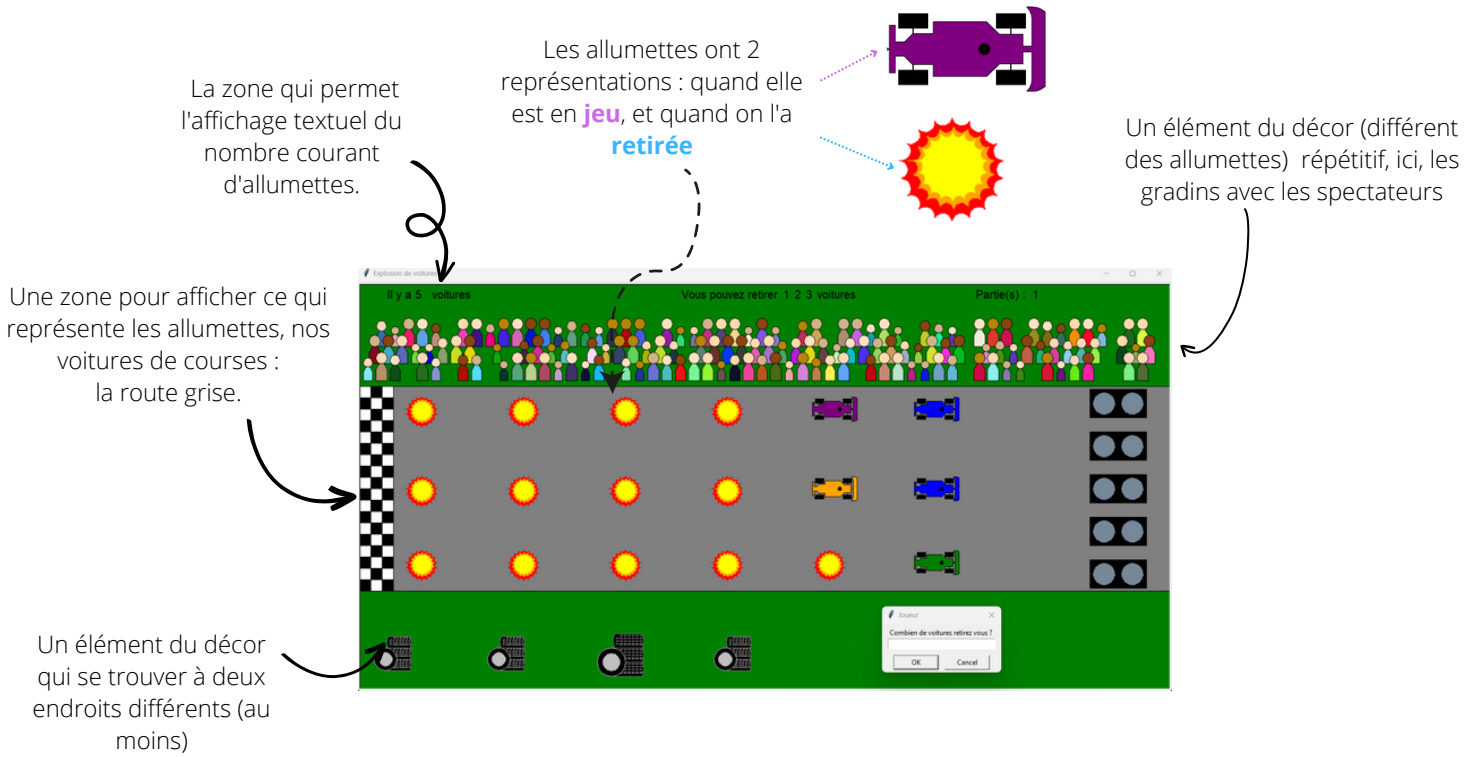
Structure du projet (décomposition en fonctions, en modules, organisation du code) :

Pour réaliser notre jeu nous avons fait 2 scripts différents, sur 2 fichiers différents. L'un d'eux, pour la partie graphisme, turtle etc et l'autre pour la partie jeu. Ainsi dans la partie jeu, nous faisons appel à l'autre fichier, pour intégrer le dessin dans le jeu.

Le fichier dans lequel il y a la partie graphique s'appelle "dessin.py".

L'autre fichier, code, depuis lequel on lance tout le jeu, s'appelle 'allumette.py".

```
from random import*
from dessin import*
import time
```



difficultés rencontrées :

Initialement nous avions prévu de remplacer nos voitures par des grilles (grille de départ de course de voiture), mais nous avons rencontrés un petit souci, lorsque le joueur ou l'ordinateur retirait des voitures, les grilles apparaissaient dans des sens aléatoires, nous n'avons pas pu déterminer d'où venait le problème.

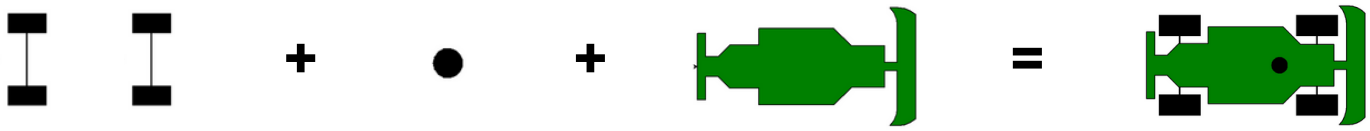
Solution : Nous avons donc remplacé les grilles par une forme symétrique pour que ça fonctionne, et c'est quand même plus drôle de faire exploser nos voitures

PARTIE GRAPHIQUE

Pour faire une belle partie graphique, nous avons fait des fonctions qui, seront appellées une fois le jeu lancé. Effectivement il est plus simple de faire appel a des fonctions qui chacunes dessinent un élément du décor.

LA VOITURE

Par exemple, rien que pour la voiture nous faisons appel à plusieurs fonctions.



```
def roue(t,x,y,taille):
    tracer(0)
    t.up()
    t.goto(x+15*taille,y+35*taille)
    t.setheading(0)
    t.down()
    t.color("black")
    t.begin_fill()
    for i in range(0,2):
        for i in range (0,2):
            t.forward(50*taille)
            t.left(90)
            t.forward(25*taille)
            t.left(90)
        t.up()
        t.goto(x+180*taille,y+35*taille)
        t.down()
    t.end_fill()
    t.up()
    t.goto(x+15*taille,y-35*taille)
    t.down()
    t.begin_fill()
    for i in range(0,2):
        for i in range (0,2):
            t.forward(50*taille)
            t.right(90)
            t.forward(25*taille)
            t.right(90)
        t.up()
        t.goto(x+180*taille,y-35*taille)
        t.down()
    t.end_fill()
    t.up()
    t.goto(x+40*taille,y+35*taille)
    t.width(2*taille)
    t.down()
    t.goto(x+40*taille,y-35*taille)
    t.up()
    t.goto(x+205*taille,y+35*taille)
    t.width(2*taille)
    t.down()
    t.goto(x+205*taille,y-35*taille)
```

```
def casque(t,x,y,taille):
    tracer(0)
    t.up()
    t.goto(x+150*taille,y)
    t.setheading(270)
    t.down()
    t.begin_fill()
    t.color("black")
    t.width(1)
    t.circle(10*taille)
    t.end_fill()
```

```
def chassis(t,x,y,c,taille,a=1):
    tracer(0)
    t.pencolor("black")
    t.width(taille)
    t.fillcolor(choice(c))
    t.up()
    for i in range (0,2):
        t.begin_fill()
        t.goto(x,y)
        t.setheading(0)
        t.down()
        t.left(90*a)
        t.forward(40*taille)
        t.right(90*a)
        t.forward(10*taille)
        t.right(90*a)
        t.forward(28*taille)
        t.left(90*a)
        t.forward(15*taille)
        t.left(45*a)
        t.forward(20*taille)
        t.right(45*a)
        t.forward(35*taille)
        t.left(90*a)
        t.forward(20*taille)
        t.right(90*a)
        t.forward(90*taille)
        t.right(45*a)
        t.forward(30*taille)
        t.left(45*a)
        t.forward(40*taille)
        t.right(90*a)
        t.forward(20*taille)
        t.left(90*a)
        t.forward(15*taille)
        t.left(90*a)
        t.forward(45*taille)
        t.circle(30*taille*a,45)
        t.setheading(0)
        t.forward(10*taille)
        t.circle(-30*taille*a,45)
        t.setheading(270*a)
        t.forward(63*taille)
        a=-a
        t.up()
        t.end_fill()
```

```
def voiture(t,x,y,taille):
    tracer(0)
    t.width(1)
    roue(t,x,y,taille)
    t.width(1)
    t.up()
    t.goto(x,y)
    t.down()
    chassis(t,x,y,c,taille)
    casque(t,x,y,taille)
```

LE PUBLIC

Afin de créer un peu de diversité au sein de notre public, nous avons créer une fonction bonhomme, qui dessine un bonhomme, avec une couleur de peau aléatoire, appelée dans une liste prédéfinie (histoire qu'il n'ait pas la tête bleue...), une taille en entrée, et une couleur de tenue différente. Ainsi, une fonction "tribune", dessine des rangées de bonhomme, en appelant la fonction "bonhomme" et en tirant aléatoirement des tailles, elles aussi depuis une liste, dans laquelle il y a aussi la taille 0, pour créer des places de libres dans le public.

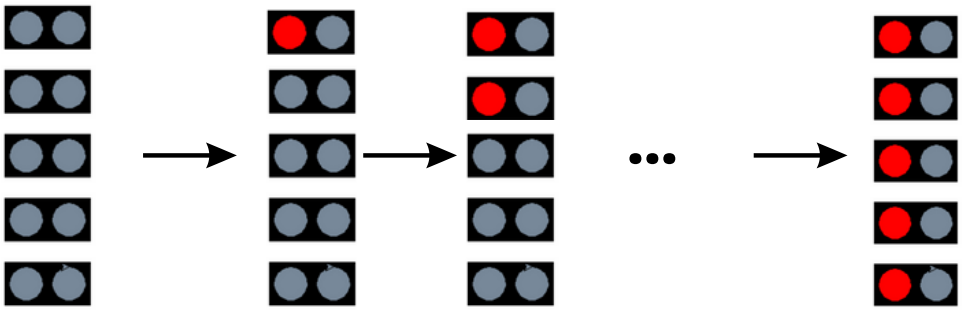
```
def bonhomme(t,x,y,f):
    tracer(0)
    t.up()
    t.goto(x,y)
    t.down()
    t.pencolor("black")
    t.fillcolor((randint(0,255)/255,randint(0,255)/255,randint(0,255)/255))
    t.setheading(0)
    t.down()
    t.begin_fill()
    t.forward(1*f)
    t.left(90)
    t.forward(2*f)
    t.left(30)
    t.forward(2*f)
    t.up()
    t.goto(x,y)
    t.down()
    t.setheading(180)
    t.forward(1*f)
    t.right(90)
    t.forward(2*f)
    t.right(30)
    t.forward(2*f)
    t.end_fill()
    t.up()
    t.goto(x,y+3*f)
    t.setheading(0)
    t.down()
    t.pencolor("black")
    t.fillcolor(choice(peau))
    t.begin_fill()
    t.circle(1*f)
    t.end_fill()
    t.up()
```

```
peau=["tan","bisque","navajowhite","darkgoldenrod","saddlebrown","peachpuff"]
f=[10,10,10,10,10,9,9,9,10,10,7,7,8,8,0,0]
y=window_width()
x=window_height()
```

```
def tribune(t,x,y):
    for j in range (0,3):
        for i in range (0,58):
            tracer(0)
            bonhomme(t,-(x-110)/2+24*i+12*(j%2)-40,y/2-110-30*j,choice(f))
```

BONUS

Afin de donner un peu de vie et surtout l'impression que l'on joue dans la vraie vie ! Nous avons ajouté une petite animation lorsque l'ordinateur joue, afin de créer une latence, qui donne l'impression que quelqu'un réfléchie, et surtout pour rappeler les vraies départs de course de voitures.



PARTIE CODAGE / JEU

Nous avons choisi de réaliser un jeu de niveau 1, dont les attentes étaient les suivantes :

- un seul tas d'allumettes,
- le nombre d'allumettes de départ est tiré aléatoirement,
- lecture des choix utilisateur avec numinput
- la règle du jeu est donnée dans une liste
- celui qui prend la dernière allumette a gagné

Pour le programme qui lance le jeu, on a fait une fonction pour le coup du joueur et une fonction pour le coup de l'ordinateur.

```
#Fonction pour le tour du joueur
def coup_joueur(a,r):
    j = numinput("Joueur","Combien de voitures retirez vous ? ")
    while j not in(r) or j > a :
        j = numinput("Joueur","Retirez un nombre de voiture permis par la regle

    return j

def coup_ordi(a,r):
    if a<min(r):
        gagner = False
    elif a>0 and a>=min(r):
        o=0
        i=0
        while a-o!=0 and i<len(r) and o<=a:
            o=r[i]
            i=i+1
        if a-o!=0:
            o=choice(r)
    return o
```

```
#Decor et initialisation des tortues
hideturtle()
tortue_decor = Turtle()
tortue_voiture = Turtle()
tortue_texte = Turtle()
tortue_cache = Turtle()
tortue_nombre = Turtle()
tortue_partie = Turtle()

tf = Turtle()
tf.hideturtle()

tortue_texte.hideturtle()
tortue_voiture.hideturtle()
tortue_decor.hideturtle()
tortue_cache.hideturtle()
tortue_nombre.hideturtle()
tortue_partie.hideturtle()
```

Nous avons fait plusieurs tortues, indépendantes pour dessiner des blocs différents du décor, ou encore du score à afficher.

Au début du jeu, le nombre de voiture est tiré aléatoirement entre 15 et 21, car nous avons fait une route qui peut accueillir jusqu'à 7 rangées de voiture, en 3 lignes, soit 21. Et 15 pour que le jeu ne soit pas trop court non plus.

```
#Mise en place des voitures
nb_voitures_depart = randrange(15,21)
nb_voitures_courant = nb_voitures_depart
dessin(tortue_voiture,largeur,hauteur,nb_voitures_depart,"voitures")
```

Pour effacer nos voitures une fois qu'on a choisi de les retirer, nous les avons en fait plutôt cachées. Effectivement dans la partie du programme qui suit, nous pouvons voir qu'un cache que nous avons préalablement dessiné, dans une fonction, vient se mettre par dessus la voiture, sur ce même cache, il y a le dessin de notre petite explosion. finalement on a vraiment l'impression que la voiture a disparu, et qu'elle a explosé ! Nous avons choisi de faire comme ceci et non de redessiner toutes nos allumettes (voitures) car initialement elles sont placées avec chacune une couleur aléatoire, or au court du jeu on ne veut pas que leur couleur change.

```
#Fonction pour redessiner les voitures et les remplacer après chaque coups
def Maj(t1,t2,l,h,a):
    #On peut mettre en commentaire la ligne suivante pour avoir la voiture combinée à l'explosion est pas seulement l'explosion
    dessin(t1,l,h,a,"cache")
    dessin(t2, l, h, a,"flammes")
    update()


def Maj_texte(t,n,l,h):
    t.clear()
    t.up()
    t.goto(-l/2 + 100,h/2 - 30)
    t.down()
    t.write(int(n),font=("arial",15,"normal"))
    update()
```

On a fait un mode où on peut choisir de continuer de faire, ou non des parties, le programme nous demande si on veut continuer de jouer ou non, et au fur et à mesure des parties, on peut voir le score du joueur.

```
c = textinput("Joueur","Voulez vous continuer ? o/n ")
if c == "n" :
    continuer = False
else:
    gagner = False
    partie = partie + 1

    tortue_partie.up()
    tortue_partie.goto(largeur/2 - 250,hauteur/2 - 30)
    tortue_partie.down()
    tortue_partie.write(partie ,font=("arial",15,"normal"))
    update()
```

```
#Affiche le score du joueur
speed(0)
decor(tortue_decor, largeur, hauteur)
tortue_texte.up()
tortue_texte.goto(-50,0)
tortue_texte.down()
tortue_texte.write("Vous avez gagné", font=("arial",20,"normal"))
tortue_texte.up()
tortue_texte.goto(-50,-30)
tortue_texte.down()
tortue_texte.write(score, font=("arial",20,"normal"))
tortue_texte.up()
tortue_texte.goto(-50,-60)
tortue_texte.down()
tortue_texte.write("partie(s) sur", font=("arial",20,"normal"))
tortue_texte.up()
tortue_texte.goto(-50,-90)
tortue_texte.down()
tortue_texte.write(partie, font=("arial",20,"normal"))
tortue_texte.up()
tortue_texte.goto(-50,-120)
tortue_texte.down()
tortue_texte.write("partie(s)", font=("arial",20,"normal"))
```

Répartition du travail entre les deux membres du binôme :
Anaïs : Partie Graphique / compte rendu
Yoan : Partie Intégration / codage du jeu
(Nous avons bien sûr tous les deux fait un peu de tout au final, mais dans l'ensemble et majoritairement le travail s'est divisé comme ci dessus entre nous.)

STRATÉGIE GAGNANTE

Pour implémenter la stratégie gagnante dans notre code, nous avons utilisé les fonctions créées dans le TD Allumettes. Dans cette configuration, l'utilisateur est toujours le premier à jouer.

```python
def mex(l):
    for i in range(len(l)) :
        if not(i in l):
            return i
    return len(l)


#La fonction prend en entrée un entier représentant le nombre d'allumettes de la pile
# et une liste représentant le nombre d'allumettes pouvant être retirées,
# et retourne la valeur de cette pile.
def valPileAllumettes(n,r):
    if n<= 0:
        return 0
    l = []
    for i in r:
        l.append(valPileAllumettes(n-i,r))
    return mex(l)

def sumNimXOR(a,b):
    return a^b

#prend en entrée un jeu d'allumettes (i.e. liste de piles)
# et la liste représentant le  nombre d'allumettes pouvant être retirées
# et qui retourne la valeur du jeu.
def valJeuAllumettes(j,r):
    val = 0
    for i in j:
        val = sumNimXOR(val,valPileAllumettes(i,r))
```

Grâce à ses fonctions nous pouvons calculer les coups menants à la victoire. Si aucun coup ne mène à la victoire l'ordinateur joue au hasard dans la règle. On remarque que dans le else la boucle qui parcours j n'a aucun intérêt puisque il n'y a qu'une seule valeur dans j, cependant cela permet d'implémenter plus facilement un jeu à plusieurs tas. On remarque aussi que l'on choisit le dernier des meilleurs coups, ce qui n'est pas optimal, mais en changeant le type de o en liste, on pourrait avoir la liste de l'ensemble des coups gagnant et de choisir celui qui nous fais gagner le plus rapidement possible. A noter que dans un jeu avec une règle 1,2,3 il n'y a qu'un seul coup gagnant s'il y en a 1.

```python
#Fonction pour le tour de l'ordinateur
# a représente le nombre d'allumettes courantes
# r représente la règle
# o représente le coup de l'ordi
def coup_ordi(a,r):
    #On regarde si on peut jouer
    if a<min(r):
        gagner = False
    # Calcul l'ensemble des premiers coups menant à la victoire. Et on en choisit 1
    else:
        j = [a]
        o = 0
        for i in range(len(j)):
            for k in r:
                if(j[i]-k>=0):
                    j[i]-=k
                    if valJeuAllumettes(j,r) == 0:
                        o = k
                    j[i]+=k

    # si pas de strat, on tire aléatoirement parmis les valeurs possibles dans la règle
    if o == 0:
        s = []
        for val in r :
            if val <= a:
                s = s + [val]
        o = choice(s)
    return o
```

ANALYSE DU PROGRAMME

LES FONCTIONS

Après une revue du programme, nous aurions pu mettre les instructions d'affichages textuelles dans des fonctions.
Par exemple : une fonction affiche_score affiche_regle affiche_nbvoiture etc... Ce qui aurait permis de rendre le main plus aéré. Nous avons cependant mis des commentaires pour éviter de perdre le lecteur.
Sinon le reste du programme apparait assez digeste.

CONCLUSION

Toute au long de cette année nous avons pu progresser à notre rythme, ce qui nous a permis d'améliorer le programme écrit en début d'année et d'y porter un jugement plus constructif et voir les "erreurs" ou mauvais choix que nous avons pu faire. Merci d'avoir pris le temps de lire ce rapport.