PROJET INITIATION SCIENCES NUMERIQUE

Année : 2012-2013

Programme : SNAKE V2.4

Par
BEAUVILLAIN DE MONTREUIL David
Et
SIDURON Arnaud

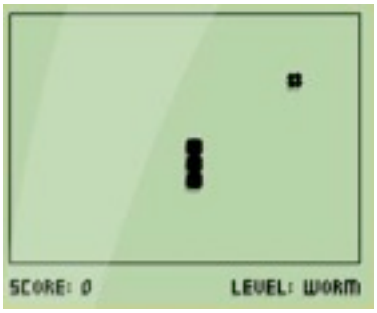
Dossier réalisé par David Beauvillain

Sommaire

I. Introduction Globale

a. Qu'est ce qu'un Snake ?

Un «Snake» signifie «Serpent» en français. De façon générale, pour avoir des points et augmenter sa grandeur, il suffit qu'il mange ses cibles. Ses cibles sont régénérées à chaque fois que le Snake les «mangent». Afin de poursuivre le jeu, il ne faut pas que le Snake se touche lui même ou qu'il rentre dans un mur sinon vous perdez et l'objectif final est d'avoir le plus de point et le Snake le plus grand possible.



b. Comment-a-t-on abordé le problème ?

Dans nôtre cas, le Snake est représenté par des 5 les uns derrière les autres dans un tableau rempli de 0, représentant l'espace ou le Snake peut se déplacer. Ses cibles sont représentées par des 2 générés aléatoirement dans tout le tableau. Les 1 représentent le mur, que le Snake ne peut franchir. Dans notre programme, vous ne perdez pas si vous rentré dans un mur ou si le Snake se touche lui même, il est tout simplement impossible d'effectuer ces actions et donc de perdre. A chaque fois que le Snake mange un 2, un cinq supplémentaire s'ajoute au Snake et 10 points supplémentaire vous sont attribués à chaque fois.

II. Présentation des Bibliothèques

a. Scipy

Codage : importé par : «from scipy import» (l,4)

=> A pour fonction de prendre en compte les tableaux de manière général afin qu'il sois afficher quand on le demande.

b. Time

Codage : importé par : «import time» (l,5)

utilisé en : «time.sleep(0.1)» (l,161)

=> A pour fonction la vitesse de régénération et d'affichage du tableau à chaque appel d'une fonction.

c. Random

Codage : importé par : «import random» (l,6)

utilisé en : « x = random.randint(3,17) » (l,19)

« y = random.randint(3,17) » (l,20)

=> A pour fonction la génération d'un 2 aléatoirement dans tout le tableau dans l'intervalle {3,17} sur x et y.

d. OS

Codage : importé par : «import os» (l,7)

utilisé en : « os.system(«cls») » (l,12)

=> A pour fonction l'effacement du tableau afin d'en avoir un seul et unique a chaque appel de fonction.

e. MSVRCT, KBHIT, GETCH

Codage : importé par : «from msvrct import kbhit, getch» (l,8)

utilisé en : «if kbhit( )» (l,162)

«touche = getch ( )» (l,163)

=> A pour fonction la prise en charge des touches du clavier dans le programme.

III. Présentation des diverses Fonctions

a. Affiche Tableau

Codage : «def affiche_tableau(x1,y1):
for y in range(y1):
for x in range(x1):
print t[y,x]
print»

=> Cette fonction permet de définir les variables x et y du tableau et est appelé à chaque fois que l'on a besoin que celui-ci s'actualise ou se rafraîchisse.

b. Objet

Codage : «def obj( ):
x = random.randint(3,17)
y = random.randint(3,17)
if t[y,x] == 5:
obj( )
print "remplacer"
t[y,x] = 2»

=> Cette fonction permet de définir la mise en place du 2 dans le tableau, à chaque appelle de celle-ci un nouveau 2 est généré aléatoirement dans le tableau. La condition lié a cette fonction permet de générer un autre 2 dans le tableau si celui généré précédemment tombe sur un 5 composant le Snake

c. Empiler

Codage : «def empiler( ):
global index
index = index + 1
pile[index,0] = x
pile[index,1] = y»

=> Cette fonction permet de prendre en compte un tableau définit dans le codage en temps que pile afin que les coordonnées du 5 y sois sauvegardées. A chaque fois qu'un 5 rencontrera un 2, une nouvelle case sera ajoutée à la pile et

les nouvelles coordonnées du 5 complétant le Snake y seront sauvegardé. Et ainsi de suite a chaque fois qu'un 5 rencontrera un 2.

c. Actualise Pile

Codage : «
```
def actualise_pile( ):
    global index
    for i in range(index):
        x = pile[i,0]
        y = pile[i,1]
        t[y,x] = 5
```
»

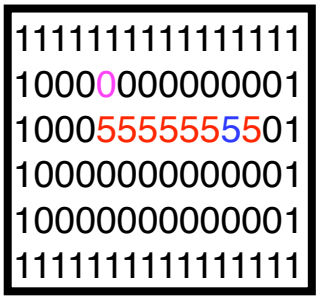
=> Cette fonction permet de définir la pile et sa composition, en autre les deux les colonnes x et y correspondant au coordonnées x et y du 5.

d. Déplacement Droit

Codage : «
```
def deplacement_droite( ):
    global xmin, ymin, xtete, ytete, index
    xmin = pile[0,0]
    ymin = pile[0,1]
    xtete = pile[index - 1,0]
    ytete = pile[index - 1,1]
    for i in range(index - 1):
        pile[i,0] = pile[i+1,0]
        pile[i,1] = pile[i+1,1]
    xtete = xtete + 1
    pile[index - 1,0] = xtete
    pile[index - 1,1] = ytete
    t[ymin, xmin] = 0

    actualise_pile( )
    affiche_tableau(20,19)
```
»

=> En premier lieu cette fonction permet d'incrémenter la pile d'un nouveau 5 lorsque la tête du snake(xtete, ytete) rencontre un 2 par la gauche, donc est en déplacement vers la droite.



On peut donc voir ici que lorsque la tête du Snake rencontre un 2 par la gauche, un 5 est ajouté à sa tête, et ce nouveau 5 admet pour coordonnée la nouvelle tête du Snake, donc il y a ici donc une incrémentation des coordonnées du 5 et ce ci est répété a chaque fois que le Snake rencontre un 2. Par la suite nous pouvons voir dans le codage que les coordonnées de la queue du snake correspondent a la première ligne de la pile, ses coordonnées sont bien sur incrémentées a chaque que le Snake se déplace. Cette méthode est de mise pour les trois fonctions qui vont suivrent.

e. Déplacement Gauche

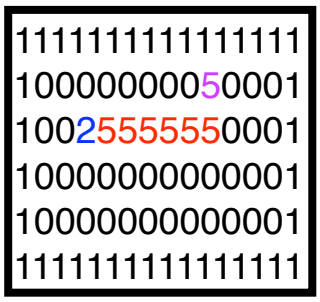
Codage : «

```
def deplacement_gauche( ):
    global xmin, ymin, xtete, ytete, index
    xmin = pile[0,0]
    ymin = pile[0,1]
    xtete = pile[index - 1,0]
    ytete = pile[index - 1,1]
    for i in range(index - 1):
        pile[i,0] = pile[i+1,0]
        pile[i,1] = pile[i+1,1]
    xtete = xtete - 1
    pile[index - 1,0] = xtete
    pile[index - 1,1] = ytete
    t[ymin, xmin] = 0

    actualise_pile( )
    affiche_tableau(20,19)
```

»

=> Cette fonction permet donc d'incrémenter la pile d'un nouveau 5 lorsque la tête du Snake rencontre un 2 par la droite, donc est en déplacement vers la gauche.



f. Déplacement Haut

Codage : «def deplacement_haut( ):

```
    global xmin, ymin, xtete, ytete, index
    xmin = pile[0,0]
    ymin = pile[0,1]
    xtete = pile[index - 1,0]
    ytete = pile[index - 1,1]
    for i in range(index - 1):
        pile[i,0] = pile[i+1,0]
        pile[i,1] = pile[i+1,1]
    ytete = ytete - 1
    pile[index - 1,0] = xtete
    pile[index - 1,1] = ytete
    t[ymin, xmin] = 0

    actualise_pile( )
    affiche_tableau(20,19)»
```

=> Cette fonction permet donc d'incrémenter la pile d'un nouveau 5 lorsque la tête du Snake rencontre un 2 par dessous, donc est en déplacement vers le haut.

g. Déplacement Bas

Codage : «
```
def deplacement_bas( ):
        global xmin, ymin, xtete, ytete, index
        xmin = pile[0,0]
        ymin = pile[0,1]
        xtete = pile[index - 1,0]
        ytete = pile[index - 1,1]
        for i in range(index - 1):
                pile[i,0] = pile[i+1,0]
                pile[i,1] = pile[i+1,1]
        ytete = ytete + 1
        pile[index - 1,0] = xtete
        pile[index - 1,1] = ytete
        t[ymin, xmin] = 0

        actualise_pile( )
        affiche_tableau(20,19)»
```

=> Cette fonction permet donc d'incrémenter la pile d'un nouveau 5 lorsque la tête du Snake rencontre un 2 par dessus, donc est en déplacement vers le bas.

```
1111111111111111
1055555500000 1
1000000500000 1
1000000200000 1
1000000000000 1
1111111111111111
```

```
1111111111111111
1000555500000 1
1000000500000 1
1000000500000 1
1000000500000 1
1111111111111111
```

IV. Présentation linéaire de la boucle «While...»

Codage : «
```
while fin == 0:
                    Px = xtete
                    Py = ytete

                    time.sleep(0.1)
                    if kbhit():
                        touche = getch( )
                        if touche == "d":
                            if t[Py,Px+1] !=1 and t[Py,Px+1] !=5:
                                if t[Py,Px+1] == 2:
                                    pile[index,0] = Px + 1
                                    pile[index,1] = Py
                                    index = index + 1
                                    compteur = compteur + 10
                                    obj( )
                                deplacement_droite( )
                            print "Point: ",compteur »
```

=> Cette partie de la boucle nous permet de voir la déclaration d'une nouvelle variable telle que Px en tant que (xtete) et Py en tant que (ytete) ainsi que la touche «d» pour le déplacement vers la droite. Par la suite deux conditions sont émises, pour que le snake puisse se déplacer vers la droite, il faut que la case suivante sois différente d'un 1 et d'un 5, sinon il ne bougera pas. La condition suivante est que si la case suivante est un 2, la coordonnée x de la tête sera incrémenté de 1 vu qu'on est en déplacement vers la droite. Donc une nouvelle case dans la pile est crée vu que par la suite la fonction Déplacement droite est appelée ainsi que la génération d'un nouveau 2.

Codage : « if touche == "q":
```
                if t[Py,Px-1] !=1 and t[Py,Px-1] !=5:
                    if t[Py,Px-1] == 2:
                        pile[index,0] = Px - 1
                        pile[index,1] = Py
                        index = index + 1
                        compteur = compteur + 10
                        obj( )
                    deplacement_gauche( )
                print "Point: ",compteur »
```

La suite de cette boucle est du même principe à part qu'ici nous somme en condition d'un déplacement gauche d'ou la pression sur la touche «q». La coordonnée x de la tête sera décrémenté de 1 vu que la coordonnée x se rapproche du 0 lors d'un déplacement vers la gauche.

Codage : « if touche == "z":
```
                if t[Py-1,Px] !=1 and t[Py-1,Px] !=5:
                    if t[Py-1,Px] == 2:
                        pile[index,0] = Px
                        pile[index,1] = Py - 1
                        index = index + 1
                        compteur = compteur + 10
                        obj( )
                    deplacement_haut( )
                print "Point: ",compteur »
```

On est ici en condition d'une pression sur la touche «z» donc d'un déplacement vers le haut, d'ou la décrémentation de 1 de la coordonnée y de la tête car on se rapproche du 0 en terme de coordonnées.

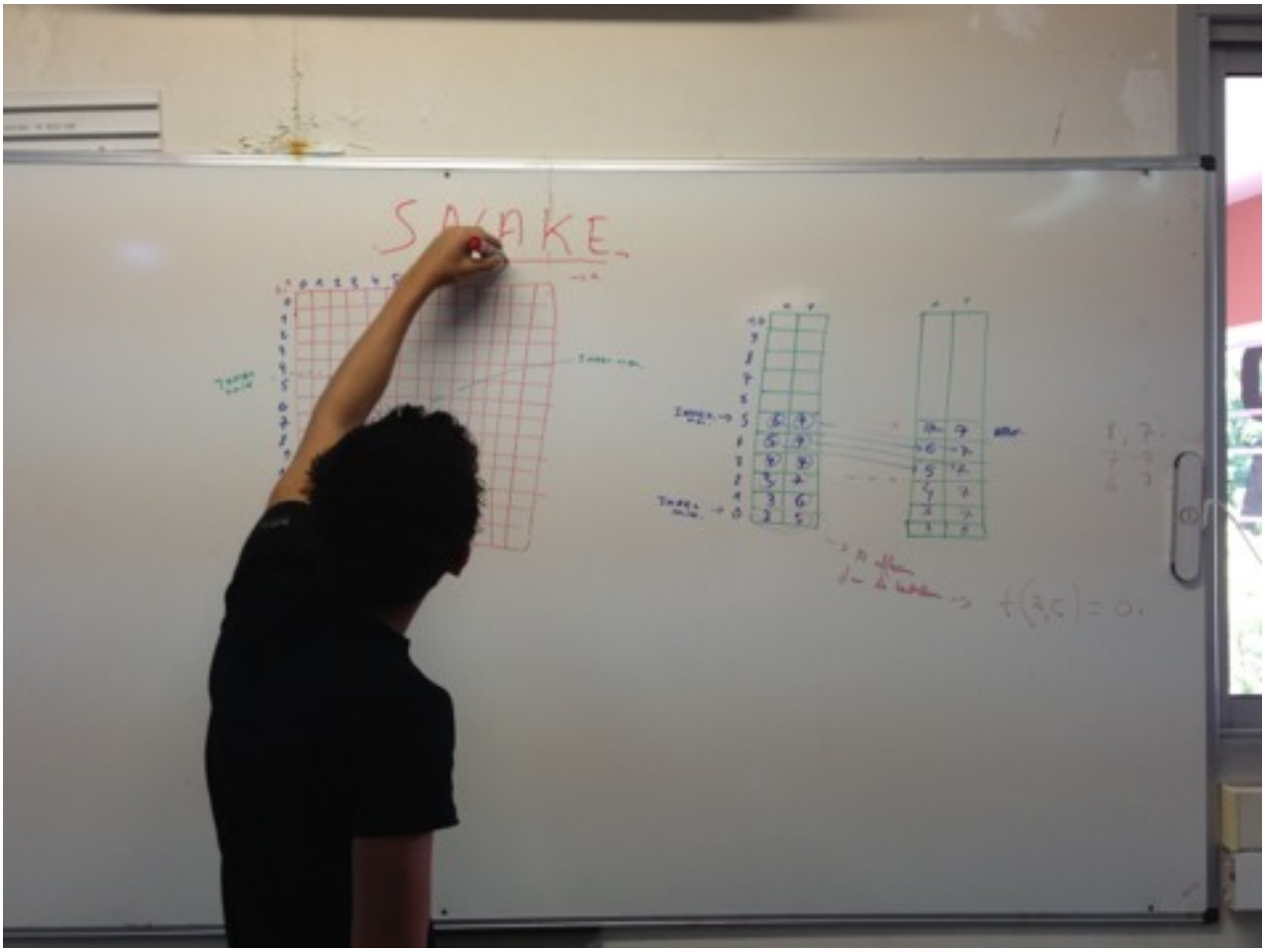
Codage : «
```
if touche == "s":
                    if t[Py+1,Px] !=1 and t[Py+1,Px] !=5:
                        if t[Py+1,Px] == 2:
                            pile[index,0] = Px
                            pile[index,1] = Py + 1
                            index = index + 1
                            compteur = compteur + 10
                            obj( )
                        deplacement_bas( )
                    print "Point: ", compteur
```
»

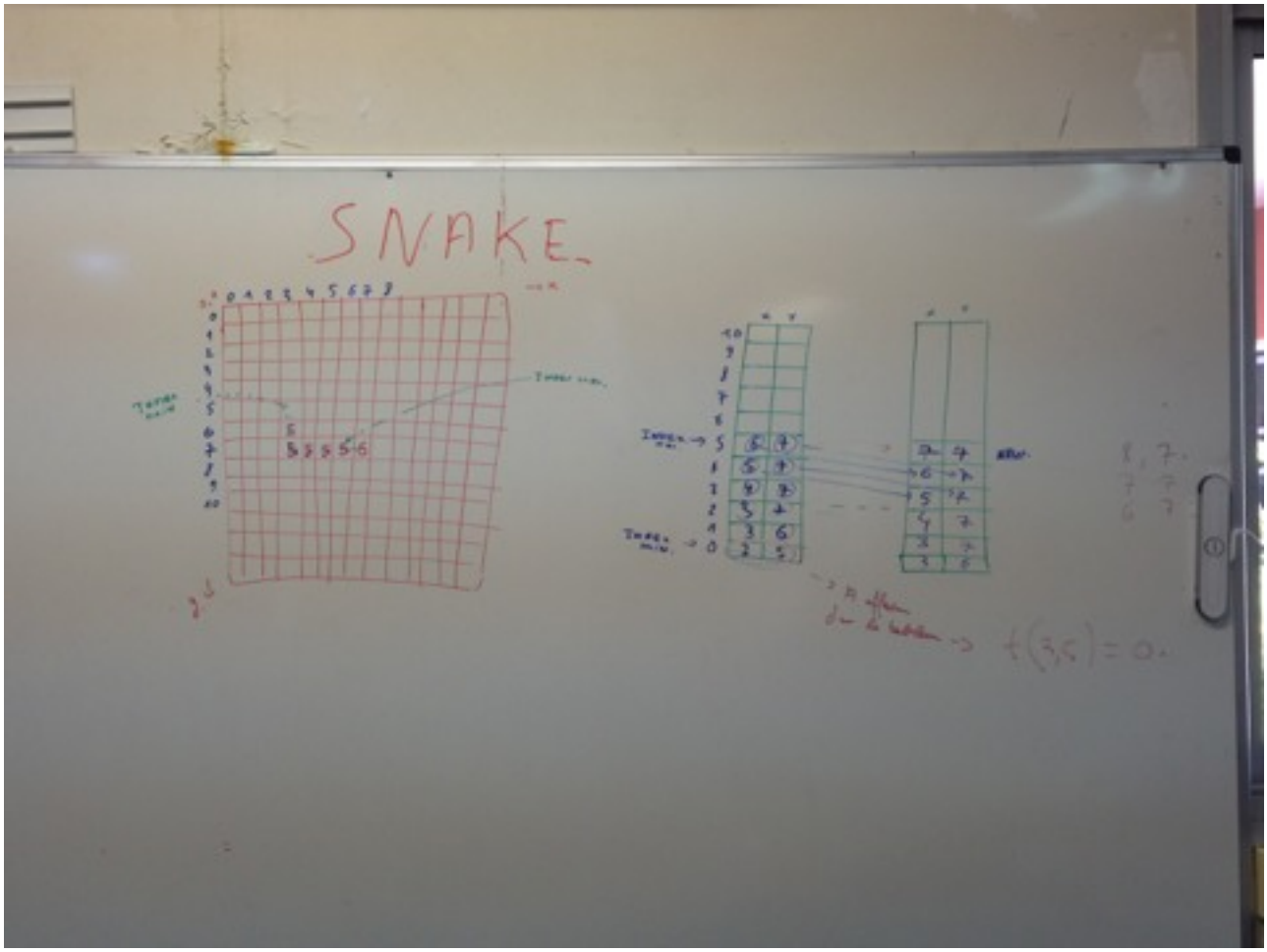
Pour terminer avec cette boucle, on est ici en condition d'une pression sur la touche «s» donc d'un déplacement vers la bas. On a donc une incrémentation de 1 de la coordonnée y de la tête

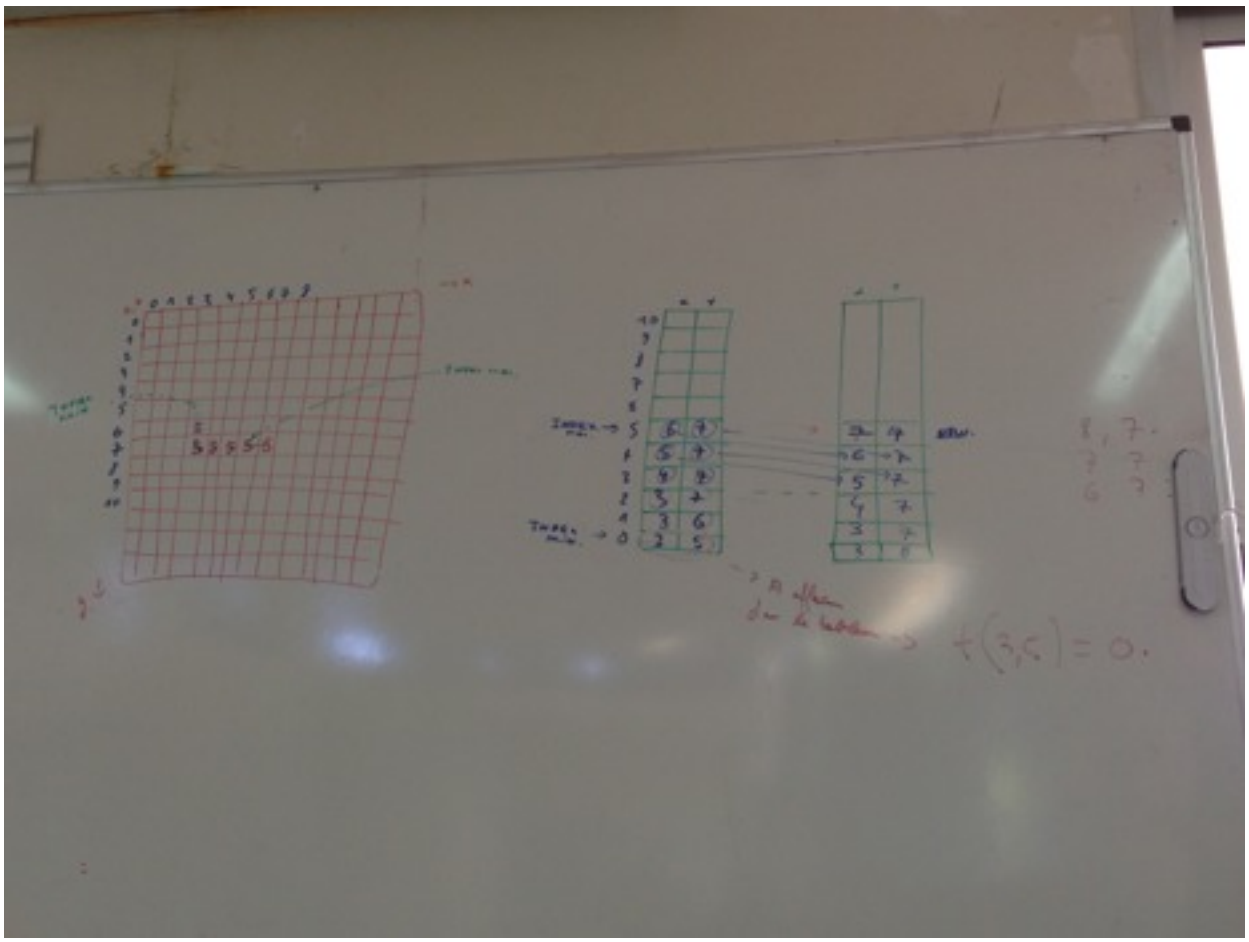
V. Les Problèmes rencontrés.

1. Nous nous somme attaqués au plus inutile au début, c'est à dire la partie graphique alors que nous n'avions pas traiter le problème de façon algorithmique, nous avons donc perdu un peu temps la dessus au début et nous sommes très vite repasser après sur l'algorithmique.
2. Problème du par où commencer.
3. Un problème qui nous a pris quand même pas mal de temps, la configuration de la pile pour que les 5 se déplace en ce suivant les uns dernière les autres.
4. Ajouter une case à la pile lorsque l'on mange un 2.
5. Problème des définitions des limites pour ce qui est du mûr et le Snake qui se mange lui même.
6. Problème de fenêtre pour l'apparition aléatoire du 2, qui parfois apparaissait dans le mûr.

Quelques Photos de notre réflexion à l'écrit







A travers ses photos nous pouvons voir notre réflexion sur le fonctionnement de la pile, ce qui allait devoir ce faire par la suite, en autre une mise en condition réel au tableau.

VI. Annexe (Variables + Programme)

«# -*- coding: utf-8 -*-» Prise en compte des caractères accentués afin d'éviter les erreurs.

«import...» Permet d'importer une bibliothèque.

«def + Nom de fonction» Définir une fonction de base qui pourra être ré-appeller par la suite dans le programme.

«(... ; ...):» Permet de définir la ou les variables de la fonction.

«for y in range(...)» Permet de définir par exemple un intervalle dans lequel sera définit la fonction. Dans le cas d'un tableau, cela permet de balayer les coordonnées du tableau.

«print...» Permet d'afficher un élément diverse... Que ce sois un commentaire ou même un tableau qui doit être rappeler ou même encore des coordonnées dans un tableau.

«global a» Permet d'utiliser la variable a en dehors de fonction ou elle se situe, elle n'est pas interne a la fonction.
«t = zeros([19,20],int)» Permet de définir un tableau et ses coordonnées.
«if touche == "d":» «if» Permet de définir une condition a laquelle répondra la fonction ou le programme.
«compteur = compteur + 10» Permet de comptabiliser quelque chose dans le programme, dans notre exemple, ce compteur permet d'additionner 10 points à chaque fois que l'on mange un 2.
«while ...» Permet de définir une condition de l'exemple «tant que...»
«compteur = 0» Permet de définir la valeur de base du compteur.