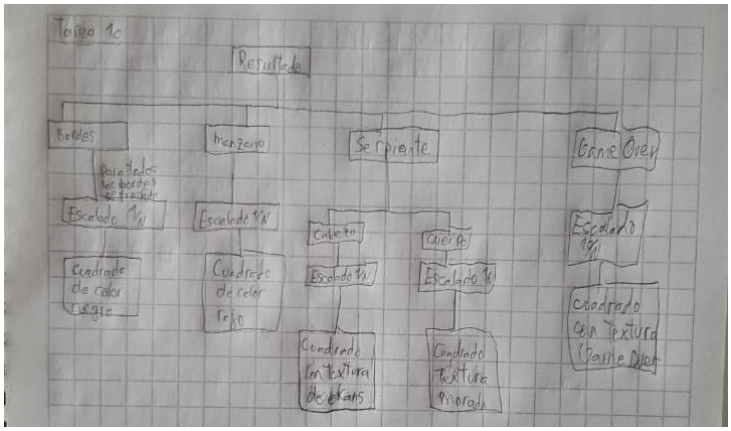
Curso: Modelacion y computacion grafica para ingenieros (CC3501-)
Nombre: Tomás Eduardo Marchessi Navarro
Fecha: 24.10.2020

Mi solucion consta de 4 directorios, uno donde estan las especifaciones para que el juego funcione(Game), otra para hacer los elementos del juego(Formas), una ultima para crear el mapa(Map) y una ultima para controlar la sepiente. En Formas cree 3 clases, una para la cola de la serpiente, una para las manzanas y una para la serpiente, las dos primeras simplemente tienen metodos para dibujar y algunas variables de utilidad, pero en la clase Ekans cree varios metodos para la relacion entre la manzana y la sepiente, entre la cabeza y la cola y una relacion con la clase Controller para poder controlar a la serpiente, en Map cree una clase para dibujar el limite y la textura al perder, y en la clase Game puse las condiciones para que el juego funcionara y se creara todo lo puesto en las clases anteriores, en la clase Controller defini las teclas que usariamos.

Al ejectuar el programa, este nos pedira introducir un numero(N), para crear las grillas del juego(requisito de la tarea), luego mientras el programa se ejecuta, podemos controlar lo que pasa mediante las flechas o las letras a,w,s,d(requisito de la tarea), ademas hay que clickear la ventana despues de definir el valor de N ya que no se abre sola.

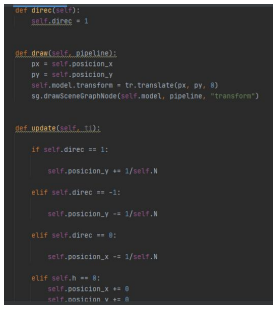



```
class Tale(object):
    N = Map.Map.N

    def __init__(self, p_x, p_y):
        self.o = 1
        gpu_tale_quad = es.toGPUShape(
            bs.createTextureQuad('cuerpo_new.png'), GL_REPEAT, GL_NEAREST)  # celeste

        # Creamos la cola
        tale = sg.SceneGraphNode('tale')
        tale.transform = tr.scale(1 / self.N, 1 / self.N, 1)
        tale.childs += [gpu_tale_quad]

        tale_tr = sg.SceneGraphNode("taleTR")
        tale_tr.childs += [tale]

        self.p_x = p_x
        self.p_y = p_y
        self.model = tale_tr

    def draw(self, pipeline):
        self.model.transform = tr.translate(self.p_x, self.p_y, 0)
        sg.drawSceneGraphNode(self.model, pipeline, "transform")

    def tale_direc(self):
        self.tale_direc = 1
```

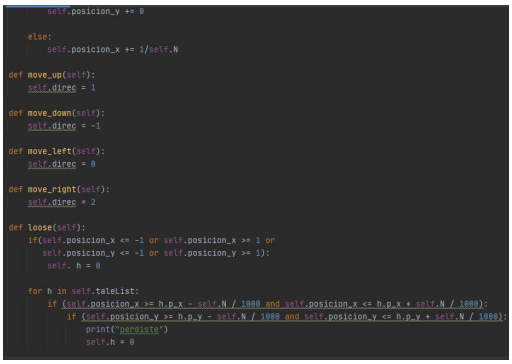
Esta clase no requiere de mucha explicacion, segui la idea del auxiliar al crear cuerpos y luego defini una funcion que lo dibujara.

```
class Ekans(object):
    N = Map.Map.N
    taleList: List['Tale']

    def __init__(self):
        self.h = 1
        self.k = 1
        self.x = 0
        self.taleList = []
        # Figuras basicas

        gpu_head_quad = es.toGPUShape(
            bs.createTextureQuad('cabeza.png'), GL_REPEAT, GL_NEAREST)  # morado


        # Creamos la cabeza

        head = sg.SceneGraphNode('head')
        head.transform = tr.scale(1/self.N, 1/self.N, 1)
        head.childs += [gpu_head_quad]

        # Armamos la serpiente

        ekans = sg.SceneGraphNode('ekans')
        self.transform_ekans = sg.SceneGraphNode('ekansTR')
        self.transform_ekans.childs += [head]
```

En esta parte de la clase hice lo mismo que en la anterior.



```
def direc(self):
    self.direc = 1


def draw(self, pipeline):
    px = self.posicion_x
    py = self.posicion_y
    self.model.transform = tr.translate(px, py, 0)
    sg.drawSceneGraphNode(self.model, pipeline, "transform")


def update(self, ti):

    if self.direc == 1:

        self.posicion_y += 1/self.N

    elif self.direc == -1:

        self.posicion_y -= 1/self.N

    elif self.direc == 0:

        self.posicion_x -= 1/self.N

    elif self.h == 0:
        self.posicion_x += 0
        self.posicion_y += 0
```



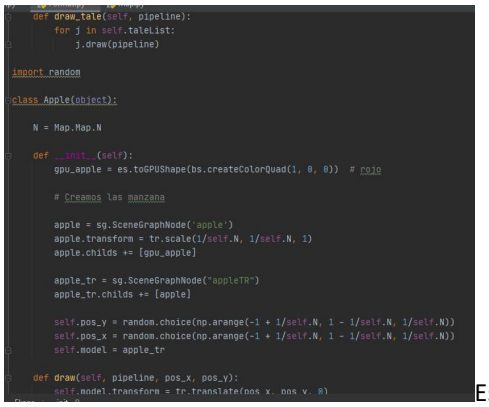
```
        self.posicion_y += 0

    else:
        self.posicion_x += 1/self.N

def move_up(self):
    self.direc = 1

def move_down(self):
    self.direc = -1

def move_left(self):
    self.direc = 0

def move_right(self):
    self.direc = 2

def loose(self):
    if(self.posicion_x <= -1 or self.posicion_x >= 1 or
       self.posicion_y <= -1 or self.posicion_y >= 1):
        self. h = 0

    for h in self.taleList:
        if (self.posicion_x >= h.p_x - self.N / 1000 and self.posicion_x <= h.p_x + self.N / 1000):
            if (self.posicion_y >= h.p_y - self.N / 1000 and self.posicion_y <= h.p_y + self.N / 1000):
                print("perdiste")
                self.h = 0
```

En esta parte empeze a definir funciones para la serpiente, pero todo todavia tiene que ver con la cabeza, usando el movimiento(todo esto con una clase como la que vimos en el auxiliar), y luego que pasa cuando la serpiente choca contra algo distinto a una manzana, lo cual genere que una variable cambia su valor a 0, luego definimos que cuando el valor sea 0 se detenga el movimiento.

```
def eat(self, pos_x, pos_y):
    self.x = 0
    if(self.posicion_x >= pos_x - self.N/1000 and self.posicion_x <= pos_x + self.N/1000):
        if(self.posicion_y >= pos_y - self.N/1000 and self.posicion_y <= pos_y + self.N/1000):
            self.x = 1
            if len(self.taleList) == 0:
                self.taleList.append((Tale(self.posicion_x, self.posicion_y)))
                self.k += 1

            else:
                q = self.taleList[len(self.taleList) - 1]
                self.taleList.append((Tale(q.p_x, q.p_y)))
    self.k = 0
    return self.x


def replace(self, ti):
    x = self.posicion_x
    y = self.posicion_y
    self.update(ti)
    for j in range(0, len(self.taleList)):
        save_x = self.taleList[j].p_x
        save_y = self.taleList[j].p_y
        self.taleList[j].p_x = x
        self.taleList[j].p_y = y
        x = save_x
```

```
def draw_tale(self, pipeline):
    for j in self.taleList:
        j.draw(pipeline)
```

En esta parte ya empezamos a relacionarnos con la cola, al colisionar la cabeza con la manzana, agregamos un elemento de la clase cola a una lista ya creada, luego creamos un metodo para ‘controlar’ la cola, la cual va reemplazando la posicion del elemento anterior a ella y luego la dibuja con esta nueva posicion



```
    def draw_tale(self, pipeline):
        for j in self.taleList:
            j.draw(pipeline)

import random

class Apple(object):

    N = Map.Map.N

    def __init__(self):
        gpu_apple = es.toGPUShape(bs.createColorQuad(1, 0, 0))  # rojo

        # Creamos las manzana

        apple = sg.SceneGraphNode('apple')
        apple.transform = tr.scale(1/self.N, 1/self.N, 1)
        apple.childs += [gpu_apple]

        apple_tr = sg.SceneGraphNode("appleTR")
        apple_tr.childs += [apple]

        self.pos_y = random.choice(np.arange(-1 + 1/self.N, 1 - 1/self.N, 1/self.N))
        self.pos_x = random.choice(np.arange(-1 + 1/self.N, 1 - 1/self.N, 1/self.N))
        self.model = apple_tr

    def draw(self, pipeline, pos_x, pos_y):
        self.model.transform = tr.translate(pos_x, pos_y, 0)
```

Esta clase es igual a la primera, con la diferencia que aqui creamos

Este es el grueso del programa, los otros 3 serian la clase controller, la cual esta sacada de la clase auxiliar, la clase juego que es implementar todo lo escrito en los otros 3 directorios y por ultimo la clase mapa que es la creacion del limite y de la textura de game over, ambas clases parecidads a la primera y ultima de las fotos, con la diferencia de que creamos cosas distintas