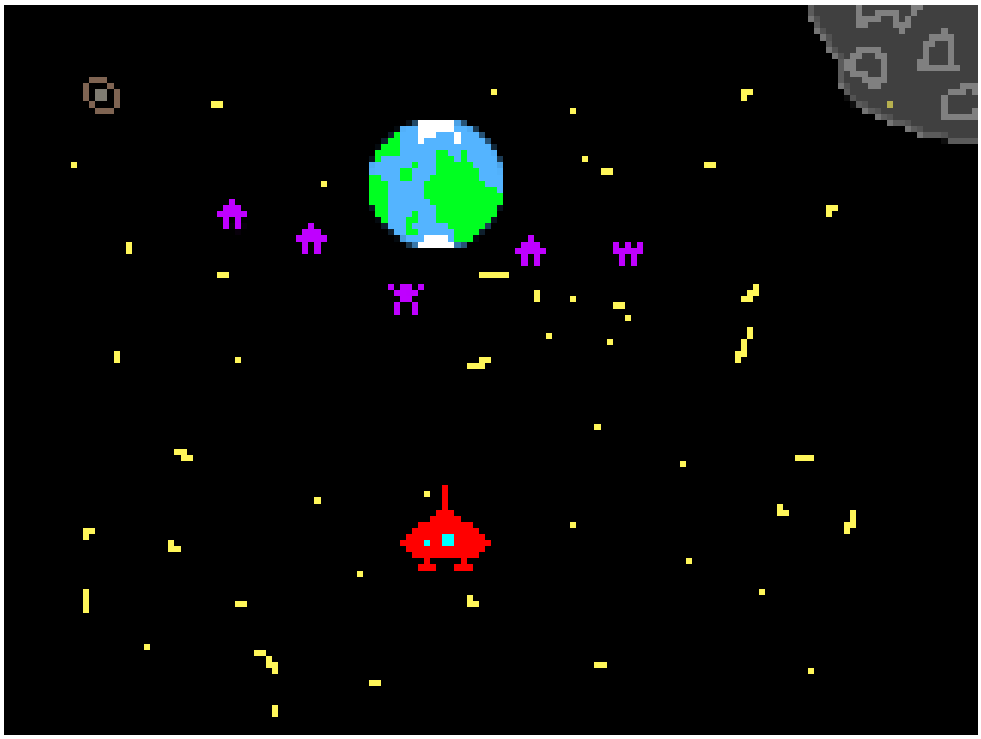
CONCEPTION



SHOOTER 3000

Type	**Conception**
Nom du projet	**Shooter 3000**
Commentaire	**Projet Ewen , S2, ENIB**
Auteur	**Ewen Daniel**
Version	**1.0**
Date	**23/01/24**

Table des matières

1 Rappel du cahier des charges

1.1 Contraintes techniques

- Le logiciel doit fonctionner sur les machines de TP de l'ENIB.
- Le développement devra donc se faire en python.
- Les notions de programmation orientée objet n'ayant pas encore été abordées, le programme devra essentiellement s'appuyer sur le paradigme de la programmation procédurale.
- Le logiciel devra être réalisé en conformité avec les pratiques préconisées en cours de IPI : barrière d'abstraction, modularité, unicode, etc...
- L'interface sera réalisée en mode texte dans un terminal de type linux en utilisant les séquence d'échappement ANSI.

1.2 Fonctionnalités

- F0: Démarrer une partie
- F1 : Déplacer le vaisseau
  - F1.1 : tuer un ennemi avec un tir
  - F1.2 : Faire rebondir les tirs sur les murs une seule fois
  - F1.3 : Déplacer le vaisseau vers la gauche
  - F1.4 : Déplacer le vaisseau vers la droite
  - F1.5:Déplacer le vaisseau vers l'avant
  - F1.6;Déplacer le vaisseau vers l'arrière
  - F1.7 : Arrêter le vaisseau
- F4 : Gagner
- F5 : Perdre
- F6 : Afficher le jeu
  - F6.1 : Afficher les ennemis
  - F6.2 : Afficher le vaisseau
  - F6.3 : Afficher le tir
  - F6.4 : Afficher le score
  - F6.5 : Afficher le numéro de niveau
  - F6.6 : Afficher le nombre de vies

2 Principes des solutions techniques

2.1 Langage

Conformément aux contraintes énoncées dans le cahier des charges, le codage est réalisé avec langage python. Nous choisissons la version 3.x du langage

2.2 Architecture du logiciel

Nous mettons en oeuvre le principe de la barrière d'abstraction. Chaque module correspond à un type de donnée et fournit toutes les opérations permettant de le manipuler de manière abstraite.

2.3 Interface utilisateur

L'interface utilisateur se fera via un terminal de type linux. Nous reprenons la solution donnée en cours de IPI en utilisant les séquence d'échappement ANSI et les modules :
termios, sys, select.

2.3.1 Boucle de simulation

Une boucle de une boucle de simulation gérant l'affichage, les événements clavier et le déplacement des tirs et du vaisseau. Les calculs de physiques devront avoir une fréquence élevé pour améliorer la qualité de la gestion des collisions. L'affichage pourra avoir une fréquence de mise à jour plus faible pour pour ne pas surcharger le terminal.

2.3.2 Affichage

L'affichage se fait en communicant directement avec le terminal en envoyant des chaînes de caractères sur la sortie standard de l'application, en utilisant les séquences d'échappement ANSI.

2.3.3 Gestion du clavier

L'entrée standard est utilisé pour détecter les actions de l'utilisateur.
Le module tty permet de rediriger les événements clavier sur l'entrée standard.
Pour connaître les actions de l'utilisateur il suffit de lire l'entrée standard.

2.3.4 Image ascii-art

Pour dessiner certaines parties de l'interface nous utilisons des « images ascii ».
Dans l'idée de séparer le code et les données, une image ASCII représentant le décor du jeu sera stockée dans le fichier texte : vaisseau.txt et vaisseau_ennemis.txt(pas encore sur)
Le jeu est affiché grâce a une liste qui associe les couleurs ASCII a des lettres qui vont lire les fichiers txt pour les afficher de manière coloré.

2.4 les niveaux

2.4.1 Grille d'entiers

Un niveau se définira par un tableau à 2 dimensions d'entier. (pas compris)

Chaque entier définit un ennemis:

0 = pas d’ennemis

1..9 = ennemis avec un certain nombre de vies restantes et une certaines vitesse plus ou moins rapide

Pour détecter et gérer les collisions entre les tirs, il suffira de convertir la position du tir en index dans la grille pour savoir si le tir touche un ennemis et modifier l'entier correspondant en conséquence

2.4.2 Description des niveaux

Les description des niveaux sera faite dans des fichiers texte qui contiendront un suite de caractère de 0-9. Ainsi, il sera facile d'ajouter ou éditer des niveaux sans modifier le code.

3 Analyse

3.1 Analyse noms/verbes :

- Verbes :

Démarrer, Déplacer, Détruire, Faire rebondir, Arrêter, Changer, Gagner,Perdre, Afficher le jeu

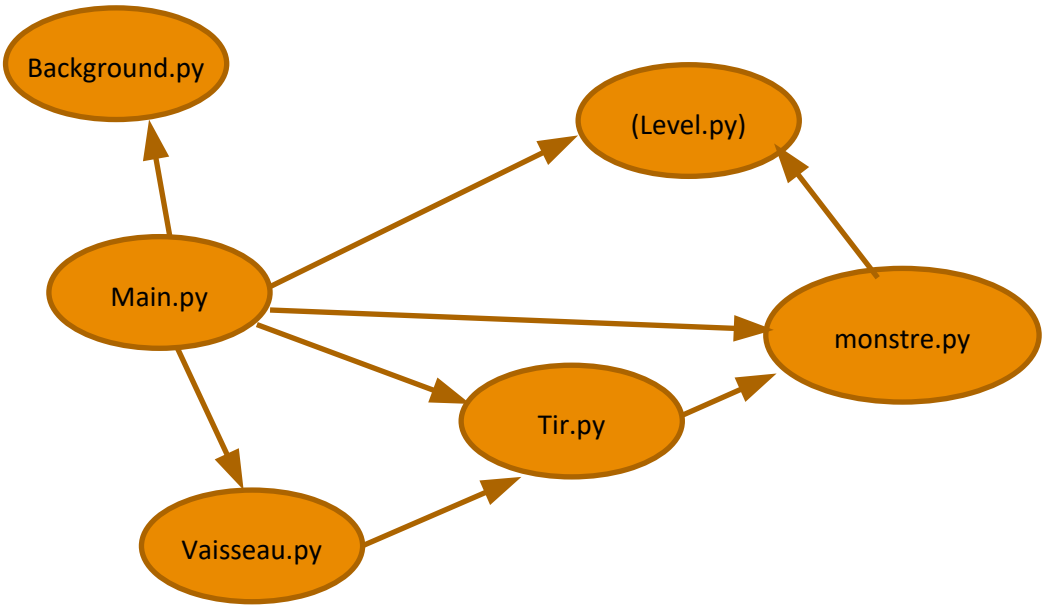
- Nom :

partie, vaisseau, niveau, jeu, vaisseau_ennemis, tir, score, numéro de niveau, nombre de vies

3.2 Types de donnée

```
type: Background = struct
            str : str
show_initialized : booléen

     fstruct

type: Tir = struct

          x     : réel
          y     : réel


     fstruct


type: Vaisseau = struct

          x     : réel
          y     : réel


     fstruct
```

3.3 Dépendance entre modules


Background.py
(Level.py)
Main.py
monstre.py
Tir.py
Vaisseau.py

3.4 Analyse descendante :

3.4.1 Arbre principal :

```
Main.main()
 +--  Main.init()
 |     +-- Background.create()
 |     +-- Vaisseau.create()
 |     +-- monstre.create()
 |     +-- Tir.create()
 |
 +-- Main.run()
       +-- Main.interact()
       +-- Main.live()
       +-- Main.show()
```

3.4.2 Arbre interaction

```
Main.interact()
      +-- Main.is_data()
      +-- Vaisseau.go_right()
      +-- Vaisseau.go_left()
      +-- Vaisseau.shot()
      +-- Vaisseau.go_forward()
      +-- Vaisseau.go_back()
```

3.4.3 Arbre simulation

```
Main.live()
  +-- Vaisseau.live()
  |      +-- Vaisseau.collide()
  |             +-- Vaisseu.test_collision()
  |             +-- Tir.normalise_speed()
  |
  +-- Tir.live()
  |      +-- Level.touch_ennemis()
  |
  +-- live = 0 :
      main.loose
```

3.4.4 Arbre affichage

```
Main.show()
  +-- affiche.image(h1.txt)
  +-- num.Num() pour afficher la vie
  +-- Vaisseau.show()
  +-- monstre.show()
  +-- Tir.show()
  - number.show() pour le score évolutif
```

4 Description des fonctions

4.1 Programme Principal : Main.py

```
data : dict {
        'timeStep' : float,
     'show_period' : float,
       'c' : int,
           'nombre_ennemi' : int,
           'vie' : Life,
           'info' : Info,
           'loose': Loose
           'score' : int,
           'game' : « menu »
           'score' : int,
          'life' : int
          'vaisseau' :Vaisseau,
            'tir' : Tir,
          'monstre':Monstre
          'affiche_score':Num}
```

- Main.**main()**
- Main.**init()**
- Main.**run()**
- Main.**show()**
- Main.**interact()**
- Main.**is_data()**
-

Main.**main()**->rien
Description : fonction principale du jeu
Paramètres : rien
Valeur de retour :aucune

Main.**init(data)**->rien
Description : initialisation du jeu
Paramètres : data
Valeur de retour :aucune

Main.**run(data)**->rien
Description : boucle de simulation
Paramètres : data
Valeur de retour :aucune

Main.**show(data)**->rien
Description : fonction d'affichage du jeu
Paramètres : data
Valeur de retour :aucune,,

Main.**interact(data)**->rien
Description : gère les événements clavier
Paramètres : data
Valeur de retour :aucune

Main.**is_data(rien)**->bool

Description : indique s'il y a quelque chose à lire sur l'entrée standard

Paramètres : rien

Valeur de retour : True si il y a quelque chose à lire, False sinon.

4.2 Vaisseau.py

- `def create(x, y, img) :`

```
    v = Vaisseau()
    v.x = x
    v.y = y
    v.img = affiche.image(img)
    return v

def get_x(v) : return v.x
def get_y(v) : return v.y
def get_img(v) : return v.img


def set_x(v,x) :
    v.x = x

def set_y(v,y) :
    v.y = y

def set_img(v, img):
    v.img = affiche.image(img)

def go_right(v) :
    if v.x < 175 :
        v.x += 3
        return v

def go_left(v) :
    if v.x > 1 :
        v.x -= 3
        return v

def go_up(v) :
    if v.y > 1 :
        v.y -= 1
        return v

def go_down(v) :
    if v.y < 38 :
        v.y += 1
        return v

def add(v,img) :
    x_de_base = v.x
```

Permet de se déplacer de gauche,droite,d'avant et en arrière

```
    x = v.x
    y = v.y
    for i in range(len(v.img)) :
        x = x_de_base
        for j in range(len(v.img[i])) :
            if v.img[i][j] != "h" and v.img[i][j] != " " :
                img[y][x] = v.img[i][j]
            x+=1
        y += 1
    return img
```

-

Permet d’afficher notre image et de la placer dans le terminal

4.3 Tir.py

- `def create(x,y,img, shoot= True ):`

```
    t = Tir()
    t.x = x
    t.y = y
    t.img = affiche.image(img)
    t.shoot = shoot
    return t

def move(t):
    if t.shoot :
        t.y -= 1
        if t.y == 0 :
            t.shoot = False
    return t

def erase(t):
    x = str(int(t.x))
    y = str(int(t.y))
    while t.y <= 0 :
        show_all("bullet.txt",liste_couleurs)
        t.shot = False
```

Supprime l'image

```
def set_img(t,img):
    t.img = affiche.image(img)
```

Permet d'afficher l'image du tir.

```
def est_sorti(t,hauteur):
    return t.y <= 0 or t.y >= hauteur-1
```

Vérifie si l'image du tir est sorti, si elle l'est, elle est donc effacé

-

4.4 Background.py

- `def create(colors, x, y) :`

```
bg = []
for i in range(y) :
    petite_liste = []
    for j in range(x) :
        petite_liste.append(colors)
    bg.append(petite_liste)
return bg
```

Permet de choisir la couleur d'une liste de couleur et de définir la taille du fond

```
def add(v,img) :
    x_de_base = v.x
    x = v.x
    y = v.y
    for i in range(len(v.img)) :
        x = x_de_base
        for j in range(len(v.img[i])) :
            if v.img[i][j] != "h" and v.img[i][j] != " " :
                img[y][x] = v.img[i][j]
            x+=1
        y += 1
    return img
```

4.5 Affiche.py

```
def image(file) :
    liste_final = []
    with open(file,"r") as fichier :
        lignes = fichier.readlines()
    # Supprimer les caractères de nouvelle ligne (\n) de chaque ligne
    lignes = [ligne.strip() for ligne in lignes]
    for element in lignes :
        petite_liste = []
        for elmt in element :
            petite_liste.append(elmt)
        liste_final.append(petite_liste)
    return liste_final
```

Permet de lire le txt et d'y associer les couleurs aux lettres. Sans cette fonction je ne pourrai rien afficher comme image

```
def add(a_ajouter, x, y, img) :
    x_de_base = x
    for i in range(len(a_ajouter)) :
        x = x_de_base
        for j in range(len(a_ajouter[i])) :
            if a_ajouter[i][j] != "h" and a_ajouter[i][j] != " " :
```

```
                img[y][x] = a_ajouter[i][j]
            x+=1
        y += 1
    return img
```

4.6 Autres fonctions :

Permettent d'afficher le menu et les scores, les informations, etc …
Exemple d'un de ces codes :

```
def inter(bienvenue, x, y, img):
    x = x - 20
    y = y - 15
    affiche.add(bienvenue[0], x + 15, y + 15, img)
    affiche.add(bienvenue[1],x + 15,y + 30 , img)
    affiche.add(bienvenue[2],x+45,y+45,img)
def show_space(bienvenue,x,y,img):
    x = x - 20
    y = y - 15
    affiche.add(bienvenue[1],x + 15,y + 30 , img)
```

Affiche les différents txt de la liste bienvenue

```
def is_data():
    return select.select([sys.stdin], [], [], 0) == ([sys.stdin], [], [])

def run(bienvenue, data):
    c = 0
    blink = True
    while c < 15 and data["game"] == "menu": # Mettez ici votre condition variable c
        img = background.create("g", 200, 45)
        show_space(bienvenue,46,5,img)
        inter(bienvenue, 46, 5, img)
        show_all(img, liste_couleurs, blink)
        blink = not blink # Inverser l'état de clignotement
        time.sleep(1) # Temps d'attente entre chaque inversion

        c += 1 # Augmenter la variable c à chaque itération

        if is_data():
            f = sys.stdin.read(1)
            if f == " ":
                data["game"] = "jeu"
            if f == "i":
                data["game"] = "info"
    show_all(img, liste_couleurs, blink=True) # Afficher le texte sans clignotement
```

Permet de changer pour rentrer dans le jeu ou les information

```
def bienvenue(data) :
    ecriture_bienvenue = affiche.image("bienvenue.txt")
    space = affiche.image("space.txt")
    info = affiche.image("info.txt")
    alphabet = [ecriture_bienvenue,space,info]
    run(alphabet, data)
```

Liste bienvenue avec les différentes images

5 Suivi de développement

5.1 P1 :

5.1.1 fonctions à développer

fonctions	codées	testées	commentaires

5.1.2 autre

Vaisseau.txt,
background.txt,0.txt,1.txt,2.txt ...9.txt,affiche_score.txt,coeur.txt,coeur1.txt,coeur2.txt,bullet.txt,h_1.txt,info.txt,info2.txt,monstre.txt,space.txt,vaisseau_explose.txt