

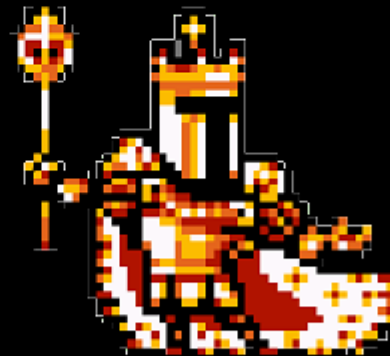
# ***Spécialité ISN 2015-2016***

# Présentation du jeu Half-Vie III

*Dans ce jeu, vous incarnez un personnage parmi quatre choix possibles :*



*Et vous affrontez un adversaire :*



*Votre but est de faire descendre la barre de vie de votre adversaire à zéro, sans que lui-même ne vous tue.*

exemple de barre de vie du personnage :



exemple de barre de vie de l'adversaire :



# Présentation du jeu Half-Vie III

*Pour cela, vous aurez le choix entre trois actions différentes à chaque tour :*

exemple:

**CHOISISSEZ VOTRE ACTION :**

1 - Coup d'épée
2 - Parade au bouclier
3 - Potion de vie

*Chacune d'entre elles a un impact différent sur le déroulement de la partie.*

*Par exemple, certaines actions peuvent occasionner des effets secondaires sur l'un des personnages.*

exemple de barre d'effets secondaires (dans les coins supérieurs de l'écran de jeu) :

PARADE 40 % ETOURDI

PARADE 30 % ENFLAMME 17 PV/SEC

# Comment le jeu a été réalisé

- À l'origine, Half-Vie 3 était un programme composé uniquement de texte.

```
Le boss possède 100 points de vie. FRAPPEZ!  
  
Quelle est la puissance de votre prochain coup ?  
75  
Dans votre rage, vous vous blessez à hauteur de 50 dommages collatéraux.  
Il vous reste 50 PV.  
Le boss possède encore 25 PV.  
Il attaque et vous subissez 40 dommages.  
Il vous reste 10 PV.  
<Attention, vous êtes grièvement blessé...  
  
Quelle est la puissance de votre prochain coup ?
```

- Puis, à l'aide du langage Python et de son module Pygame, des images et une interface graphique y ont été ajoutés au cours de l'année pour rendre le programme plus varié.

# I) Associer les différentes actions à chaque personnage

- En premier lieu, des personnages différents ont été ajoutés, chacun possédant des actions qui lui sont propres.
- Deux classes ont été créées : une pour le personnage, une pour l'adversaire.
- Dans chaque classe, les effets des différentes attaques sont détaillées et intégrées au jeu :

Classes :

Attaque 1 du personnage 1

```
def coup_epee(self, coup, coupmax, pv, ennemi, pve, randparadee):
    fenetre.blit(fontg.render(str("Quelle est la puissance de votre coup ? (dans console)"),1,(0,0,255)), (350,590))
    pygame.display.flip()
    self.coup = int(input())
    if (self.coup > self.coupmax):
        fenetre.blit(fontg.render(str("Vos capacités limitent votre puissance à . Réessayez."),1,(0,0,255)), (300,620))
        fenetre.blit(fontg.render(str(self.coupmax),1,(0,0,255)), (730,620))
        pygame.display.flip()
        self.coup = int(input())
    self.coup = self.coup * ennemi.randparadee
    self.coup = int(self.coup)
    testrand = random.randint(0,100)
    if testrand < ennemi.ecritiquee:
        self.coup = 0
        fenetre.blit(fontg.render(str("L'ennemi esquivé votre attaque."),1,(0,0,255)), (300,680))
    ennemi.pve = ennemi.pve - self.coup
    if (self.coup > self.limite):
        self.collateral = self.coup - self.limite
        self.pv = self.pv - self.collateral
        fenetre.blit(fontg.render(str("Dans votre rage, vous vous blessez à hauteur de dommages collatéraux."),1,(0,0,255)), (200,710))
        fenetre.blit(fontg.render(str(self.collateral),1,(0,0,255)), (715,710))
    pygame.display.flip()
```

Fichier principal :

Si le choix du joueur  
est l'attaque 1 :

```
if choixattperso == 1:
    if perso.choix == 'Guerrier':
        perso.coup_epee(perso.coup, perso.coupmax, perso.pv, ennemi, ennemi.pve, ennemi.randparadee)
    if perso.choix == 'Mage':
```

Le programme effectue la  
fonction décrite dans les classes

## II) L'affichage dans la fenêtre de jeu

- Pour afficher le déroulement de la partie directement sur la fenêtre, j'ai utilisé la fonction `font.render(str())` dans les classes d'action, permettant de transformer les variables textes ou nombres en textes affichables sur l'écran.

Dans la fonction définissant l'attaque du personnage 1 :

```
def coup_epee(self, coup, coupmax, pv, ennemi, pve, randparadee):
    fenetre.blit(fontg.render(str("Quelle est la puissance de votre coup ? (dans console)"),1,(0,0,255)), (350,590))
    pygame.display.flip()
    self.coup = int(input())
    if (self.coup > self.coupmax):
        fenetre.blit(fontg.render(str("Vos capacités limitent votre puissance à . Réessayez."),1,(0,0,255)), (300,620))
        fenetre.blit(fontg.render(str(self.coupmax),1,(0,0,255)), (730,620))
        pygame.display.flip()
        self.coup = int(input())
    self.coup = self.coup * ennemi.randparadee
    self.coup = int(self.coup)
    testrand = random.randint (0,100)
    if testrand < ennemi.ecritiquee:
        self.coup = 0
        fenetre.blit(fontg.render(str("L'ennemi esquivé votre attaque."),1,(0,0,255)), (300,680))
    ennemi.pve = ennemi.pve - self.coup
    if (self.coup > self.limite):
        self.collateral = self.coup - self.limite
        self.pv = self.pv - self.collateral
        fenetre.blit(fontg.render(str("Dans votre rage, vous vous blessez à hauteur de dommages collatéraux."),1,(0,0,255)), (200,710))
        fenetre.blit(fontg.render(str(self.collateral),1,(0,0,255)), (715,710))
    pygame.display.flip()
```

## II) L'affichage dans la fenêtre de jeu (part. 2)

- Au niveau de la fenêtre de jeu en elle-même, j'ai créé la fonction *reload()* permettant de recharger le modèle sur l'écran ainsi que les nombreux attributs annexes à afficher (personnages, barres de vie, barres d'effets secondaires...)

```
def reload():
    "recharger l'écran"
    global fenetre, font, textpv, textpve, image_jeu, pève, randparade, tparade, poison, tpoison, \
    tnpoison, arach, tarach, leth, tleth, camouflage, tcam, randparadee, tparade, \
    poisone, tfeu, tnfeu, peure, tpeur, textattp1, textattp2, textattp3
    fenetre = pygame.display.set_mode((1024, 768))
    textpv = font.render(str(perso.pv), 1, (255, 255, 255)) # afficher points de vie
    textpve = font.render(str(ennemi.pve), 1, (255, 255, 255))
    fenetre.blit(image_jeu, (0, 0))
    if perso.pv > 0: # barre de vie perso
        pygame.draw.rect(fenetre, (0, 0, 255), Rect((106, 61), (364 * perso.pv / pvmax, 37)))
    if ennemi.pve > 0: # barre de vie ennemi
        pygame.draw.rect(fenetre, (255, 0, 0), Rect((972 - 364 * ennemi.pve / pvmax, 61), (364 * ennemi.pve / pvmax, 37)))
    fenetre.blit(perso.image, (50, 200)) # afficher les sprites des personnages
    fenetre.blit(ennemi.image, (662, 200))
    fenetre.blit(textpv, (360, 70))
    fenetre.blit(tpvmax, (400, 70))
    fenetre.blit(pève, (390, 70))
    fenetre.blit(textpve, (620, 70))
    fenetre.blit(pève, (650, 70))
    fenetre.blit(tpvmaxe, (660, 70))
    if perso.randparade < 1: # afficher les effets secondaires actifs
        fenetre.blit(tparade, (5, 10))
        fenetre.blit(font.render(str(int(perso.randparade * 100)), 1, (0, 0, 255)), (80, 10))
    if perso.poisone:
        fenetre.blit(tpoison, (135, 10))
        fenetre.blit(font.render(str(perso.poisone), 1, (0, 255, 0)), (205, 10))
    if perso.arach:
        fenetre.blit(tarach, (5, 10))
    if perso.leth:
        fenetre.blit(tleth, (5, 10))
    if perso.camouflage:
        fenetre.blit(tcam, (5, 10))
    if ennemi.randparadee < 1:
        fenetre.blit(tparade, (670, 10))
        fenetre.blit(font.render(str(int(ennemi.randparadee * 100)), 1, (0, 0, 255)), (745, 10))
    if ennemi.poisone:
        fenetre.blit(tfeu, (800, 10))
        fenetre.blit(font.render(str(ennemi.poisone), 1, (255, 0, 0)), (900, 10))
    if ennemi.peure:
        fenetre.blit(tpeur, (800, 10))
    pygame.display.flip() # recharger l'image
```



### III) Perspectives d'améliorations

- Attribuer plus d'animations en fonction des événements du jeu
- Accroître le nombre de personnages et d'actions disponibles
- Permettre un mode de jeu en multijoueur