```python
import pygame
from pygame.locals import *


# Initialisation de la fenetre
pygame.init()
fenetre = pygame.display.set_mode((800, 600))


# Definition du fond
fond = pygame.Surface(fenetre.get_size())
fond.fill((0, 0, 0))

# Import d'une image a afficher
img = pygame.image.load("image.png").convert_alpha()

continuer = True
while continuer == True:
    for event in pygame.event.get():
        if event.type == QUIT:
            continuer = False

    fenetre.blit(fond, (0, 0))
    fenetre.blit(img,(100,200))
    pygame.display.flip()
```

Import des librairies utiles

Initialisation et création d'une fenêtre de 800 px par 600 px

Création d'une surface rectangulaire noire à la taille de la fenêtre

Chargement d'une image avec gestion de la transparence.

Analyse des événements

Mise en mémoire tampon des éléments à afficher

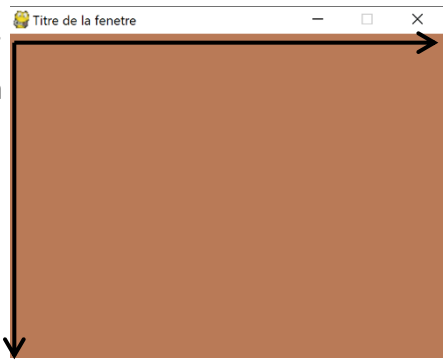
Mise à jour de l'affichage à partir de la mémoire tampon

Boucle « infinie » des événements

Exercice 1 : Création d'un cadre de jeu avec un personnage mobile

A partir du programme « squelette » : ***base_ex1.py*** :

1. **Créer une fenêtre** PyGame de dimension 800 par 600 avec un titre.



2. **Ajouter un fond** en créant une surface rectangulaire à la taille de la fenêtre (couleur du fond au choix)

```
fond = pygame.Surface(screen.get_size())
fond.fill((R, G, B))
fenetre.blit(fond, (0, 0))
```

R, G et B sont à remplacer par des entiers compris entre 0 et 255 qui représentent les composantes Rouge, Verte et Bleue de la couleur de fond choisie. (https://htmlcolorcodes.com/fr/)

3. **Ajouter un l'image d'un personnage** au centre de la fenêtre.

```
perso = pygame.image.load("perso.png").convert_alpha()
```

Ne pas oublier de « bliter » l'image dans la boucle infinie…

4. **Rendre le personnage mobile** à l'aide de flèche du clavier.
   Ajouter la récupération des événements de type « touche de clavier enfoncée » et ajouter le déplacement en fonction de la touche enfoncée.

   if event.type == KEYDOWN:

   if event.key == K_DOWN:

   Il est nécessaire de créer deux variables : par exemple *perso_x* et *perso_y* qui stockeront la position du personnage à afficher.
   Il suffit d'afficher l'image du personnage aux coordonnées *perso_x* et *perso_y* et de modifier ces deux variables lorsqu'une touche de déplacement est enfoncée.
   Attention au repère : l'axe des ordonnées est orienté du haut vers le bas.

   Liste des constantes pour les touches du clavier : https://www.pygame.org/docs/ref/key.html

5. Modifier le programme pour que le personnage ne puisse **pas sortir de la fenêtre de jeu**.
   Attention : Tenir compte de la hauteur et la largeur du personnage

6. Pour éviter d'avoir à appuyer plusieurs fois sur une touche et déplacer de manière continue votre personnage, utilisez la fonction :

   pygame.key.set_repeat(delay, interval)

Le premier paramètre correspond au délai pour déclencher l'évènement une fois la touche enfoncée. Le deuxième paramètre correspond à l'intervalle de temps pour réactiver l'évènement une fois la touche maintenue enfoncée.
***Exemple** : pygame.set_repeat(500,10) indique que si un personnage se déplace de 1 pixel lorsqu'on enfonce une touche, le premier déplacement commencera 500 ms après l'appui sur la touche et ensuite le déplacement s'effectuera toutes les 10ms*

Exercice 2 : Balle rebondissante

Programme « squelette » : ***base_ex2.py*** :

Dans ce programme « squelette », des fonctions ont été dédiées pour séparer les tâches :

- Gérer les événements (touches du clavier ou quitter)
- Faire évoluer le jeu (modifier les coordonnées, vitesses…)
- Dessiner les éléments (ici un fond, une balle, et du texte)

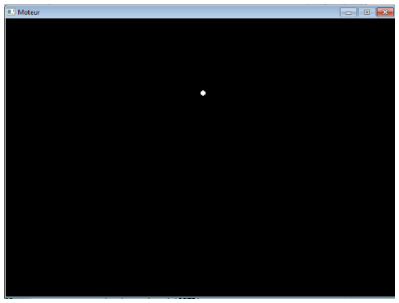
Question préliminaire

Dimension : 700x500)

Pour éviter d'avoir à gérer les coordonnées dans le repère assez inhabituel de la fenêtre PyGame (***voir figure de l'exercice 1***), nous allons utiliser une fonction ***change_repere*** définie dans le programme ***base_ex2.py*** pour travailler dans un repère plus naturel (comme sur l'image ci-contre).

L'idée est de travailler, au niveau du calcul, dans ce nouveau repère (x_1, y_1) plus naturel puis de convertir les coordonnées dans le repère PyGame (x_2, y_2) avant l'affichage des différents éléments (fonction *dessiner*).
Compléter la fonction ***change_repere*** pour que les coordonnées du repère de travail soient converties dans le repère de la fenêtre PyGame.

1. A partir du programme « squelette », **ajouter un fond noir et l'image d'une balle blanche** (disque de rayon 10)



pygame.draw.ellipse(screen,(255,255,255),(balle_x,balle_y,10,10))

balle_x et balle_y sont des variables représentant les coordonnées de la balle, et les valeurs 10,10 représentent la hauteur et largeur de la balle en pixels

***Attention** : lorsqu'on dessine une forme (rectangle, ellipse, cercle..etc) cela se fait directement sur la surface choisie(ici screen) donc inutile de « blitter » !*

2. **Ajouter de la gravité** pour faire chuter la balle.

Un peu de théorie…

On « discrétise » le temps « continue » en calculant, à intervalle régulier (ici $dt = 0{,}1$), des approximations des vitesses et des positions successives de la balle.

L'accélération étant $\boldsymbol{a_x = 0}$ (accélération horizontale) et $\boldsymbol{a_y = -10}$ (accélération verticale : approximation de g, accélération de la pesanteur terrestre)

- Les composantes de la vitesse de la balle sont approximativement :
  - $\boldsymbol{v_x = v_x + a_x * dt}$
  - $\boldsymbol{v_y = v_y + a_y * dt}$
- Les coordonnées de la balle sont approximativement
  - $\boldsymbol{x = x + v_x * dt}$
  - $\boldsymbol{y = y + v_y * dt}$

En s'inspirant des équations ci-dessus, compléter la fonction ***evoluer*** pour ajouter de la gravité sur la balle.

3. Associer à la touche « espace » du clavier la possibilité de **pousser manuellement la balle vers le haut.** Pour cela, modifier la **vitesse verticale** de la balle en ajoutant par exemple 50.

4. Afficher les coordonnées du vecteur-vitesse en haut de la fenêtre.

myfont = pygame.font.SysFont("Arial", 20)

label = myfont.render("Texte a afficher : " , 1, (255,255,255))

(255,255,255) : couleur blanche

***Attention** : ici **label** créé est en fait une surface sur laquelle on affiche le texte, il sera donc nécessaire de le « blitter » sur la surface voulue à la position souhaitée.*

5. Provoquer le **rebond de la balle** lorsqu'elle touche le sol.

Pour cela, inverser le sens du vecteur vitesse lorsque son ordonnée passe sous un certain seuil.

6. Provoquer le déplacement de la balle vers la droite ou vers la gauche lorsqu'un clic droit ou gauche a été effectué. Attention à ce que la balle ne sorte pas de la fenêtre !

```
if event.type == MOUSEBUTTONDOWN:
    if event.button == 1:
```

Pour ceux qui ont terminé :

Pour aller plus loin :

Chute sur différentes planètes

7. Vous pouvez **améliorer** en simulant la chute sur différentes planètes :
   On cherchera à changer le paysage du fond par appui sur une touche et on changera dans le même temps la valeur de l'accélération de la pesanteur
   - **sur terre g=10 m/s²**
   - **sur mars 0,38*g**
   - **sur la lune 0,162*g**
   - **sur jupiter 2,53*g**
   - **sur le soleil !! 27,96*g**

   Vous avez des images pour illustrer chacun des différents environnements à votre disposition

Tracé de la trajectoire :

8. Vous pouvez également faire apparaitre la trajectoire de la balle,
   Pour cela il suffit de **stocker** au fur et à mesure les valeurs des abscisses et des ordonnées de la balle dans 2 listes nommées **traceX** et **traceY**
   Vous **initialiserez** ces listes vides, puis on **ajoutera** à chaque tour de boucle les coordonnées de la balle dans ces listes. La taille de ces listes augmentera donc au cours du temps.
   Pour faire apparaitre la trajectoire, il faudra **relier par des lignes les points 2 à 2**, pour cela, par l'intermédiaire d'une boucle, on balaiera l'ensemble des points et on reliera le point d'indice i avec le point d'indice i+1.

Aide : pour tracer une ligne

pygame.draw.line(screen, (R,V,B), (xi , yi),(xi+1 , yi+1), 2)

On trace une ligne sur la surface « screen » de couleur (R,V,B) entre les points de coordonnées (xi , yi) et (xi+1 , yi+1). Le dernier paramètre correspond à l'épaisseur du trait

9. On pourra améliorer en effacement au moment voulu les trajectoires déjà affichées sur l'écran, il suffit pour cela par l'appui d'une touche d'assigner à traceX et traceY des listes vides.