Jeu et moteur physique

Présentation ISN 2017 - 30/03/2017

Vincent THOMAS – université de lorraine
Vincent.thomas@loria.fr

http://webloria.loria.fr/~vthomas/mediation/ISN_2015/

Point de départ

- Jeux et animation interessent étudiants

- Difficile envisager le continu (ex Mario)
  - animation (temps continu)
  - deplacement (espace continu)

- Mais une fois les ficelles montrées,
  - ==> accessible

Objectif de cet atelier

- Faire un moteur physique **basique** (mécanique du point) est simple.

- Un moteur physique **basique** permet de faire beaucoup de choses.

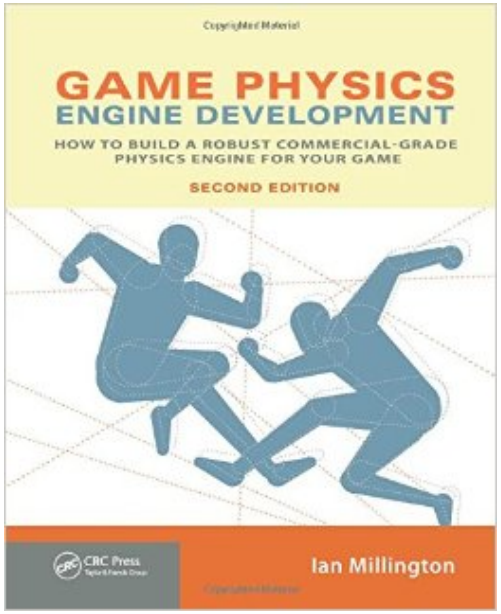
- Un moteur physique **basique** peut être une base pour de nombreux projets accessibles pour des èleves de terminale.

Références

- Encadrement Projets tutorés (DUT info)

- "pixar animation"
  - Tutoriel siggraph 2001 (a priori disparu)
  - http://www.pixar.com/companyinfo/research/pbm2001/index.html

- "Game physics engine dev"
  - Millington (2007 – reed 2010)

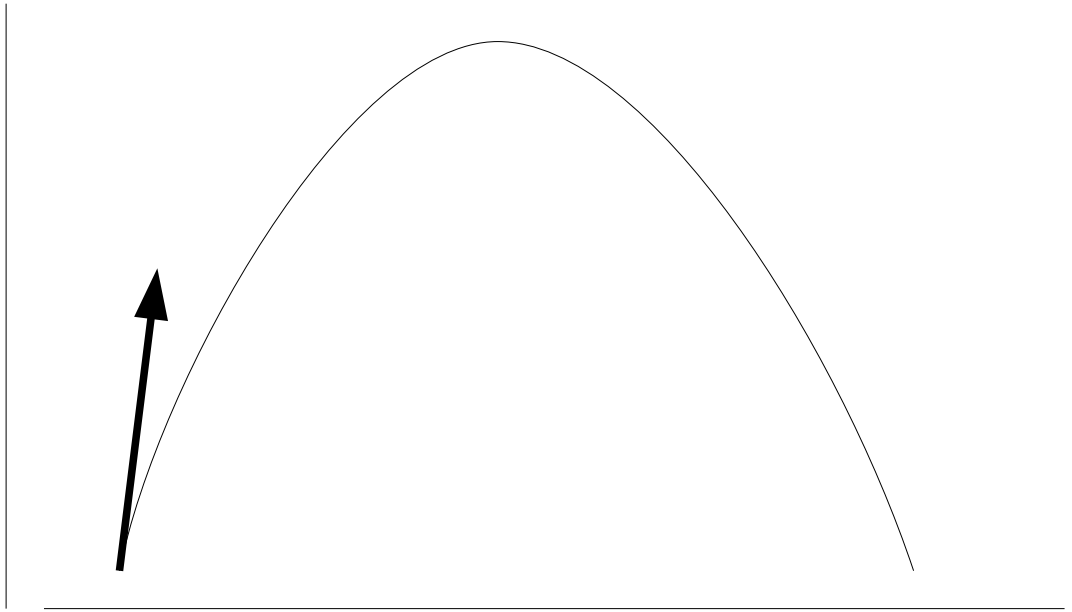


Plan

- Mecanique du point
- Moteur de jeu
- Mecanique du point (bis)
- Comportements collectifs
- Steering behaviour
- Probleme de controle

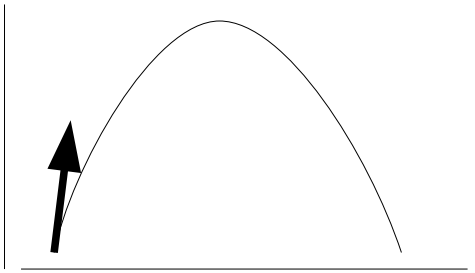
Physique du solide

- Gravité constante



Physique du solide

- Gravité constante



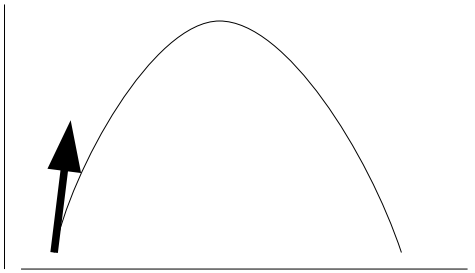
$$m\vec{a} = \sum \vec{F}$$

$$\vec{a} = \frac{d\vec{v}}{dt}$$

$$\vec{v} = \frac{d\vec{x}}{dt}$$

Physique du solide

- Gravité constante



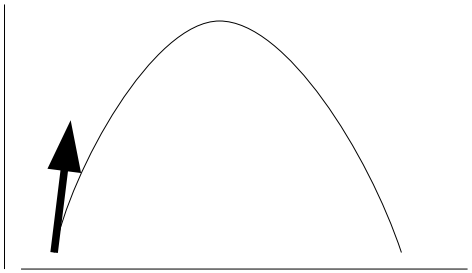
$$m\vec{a} = \sum \vec{F}$$

$$d\vec{v} = \vec{a}.dt$$

$$d\vec{x} = \vec{v}.dt$$

Physique du solide

- Gravité constante



$$m\vec{a} = \sum \vec{F}$$

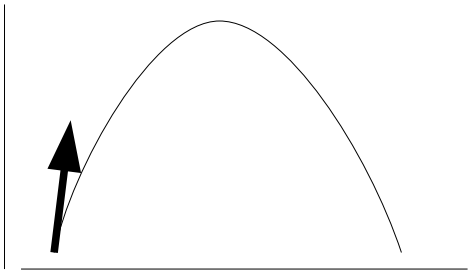
$$d\vec{v} = \vec{a}\,.\,dt$$

$$d\vec{x} = \vec{v}\,.\,dt$$

```
ax = fx / m
ay = fy / m

vx = vx0 + ax . dt
vy = vy0 + ay . dt

px = px0 + vx . dt
py = py0 + vy . dt
```

Physique du solide

- Gravité constante



$$m\vec{a} = \sum \vec{F}$$

$$d\vec{v} = \vec{a}\,.\,dt$$

$$d\vec{x} = \vec{v}\,.\,dt$$

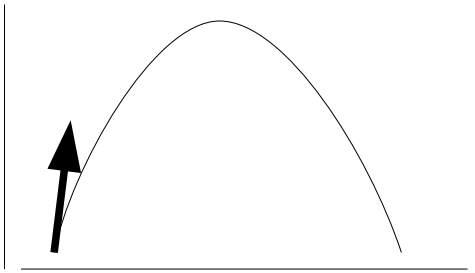
```
ax = fx / m
ay = fy / m

vx = vx0 + ax . dt
vy = vy0 + ay . dt

px = px0 + vx . dt
py = py0 + vy . dt
```

Physique du solide

```
class Jeu():

    def __init__(self):



    def evoluer(self):



    def afficher(self):




jeu=Jeu()
for i in range(1000):
    jeu.evoluer()
    jeu.afficher()
```

Physique du solide

```
class Jeu():
    def __init__(self): [/init/]
    def evoluer(self): [/dessous/]
    def afficher(self): [/print/]

jeu=Jeu()
for i in range(1000):
    jeu.evoluer()
    jeu.afficher()
```



$$m\vec{a} = \sum \vec{F}$$

$$d\vec{v} = \vec{a}.dt$$

$$d\vec{x} = \vec{v}.dt$$

```
ax = fx / m
ay = fy / m

vx = vx0 + ax . dt
vy = vy0 + ay . dt

px = px0 + vx . dt
py = py0 + vy . dt
```

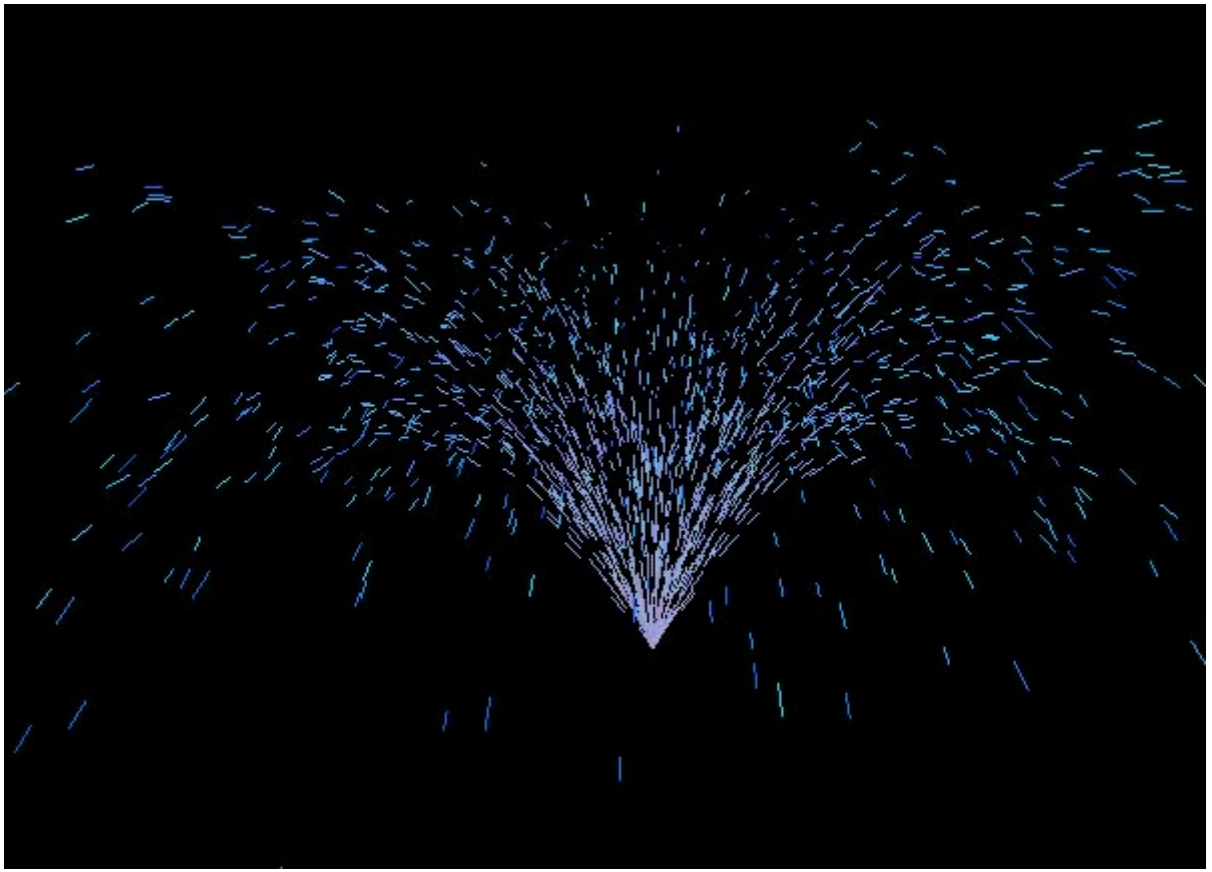
Ecrire code python (avec print)

Physique du solide

```
class Jeu():

    def __init__(self):
        self.x=50
        self.y=50
        self.vx=20
        self.vy=50
        self.ax=0
        self.ay=-9
        self.dt=0.1

    def evoluer(self):
        self.x = self.x + self.vx * self.dt
        self.y = self.y + self.vy * self.dt
        self.vx = self.vx + self.ax * self.dt
        self.vy = self.vy + self.ay * self.dt

    def afficher(self):
        print("(x,y): ", self.x, ",", self.y)
        print("(vx,vy):", self.vx, ",", self.vy)


jeu=Jeu()
for i in range(1000):
    jeu.evoluer()
    jeu.afficher()
```

Physique du solide

- Gravité constante
  - Systeme de particules



http://www.darwin3d.com/gamedev/articles/col0798.pdf

Plan

- Mecanique du point
- Moteur de jeu
- Mecanique du point (bis)
- Comportements collectifs
- Steering behaviour
- Probleme de controle

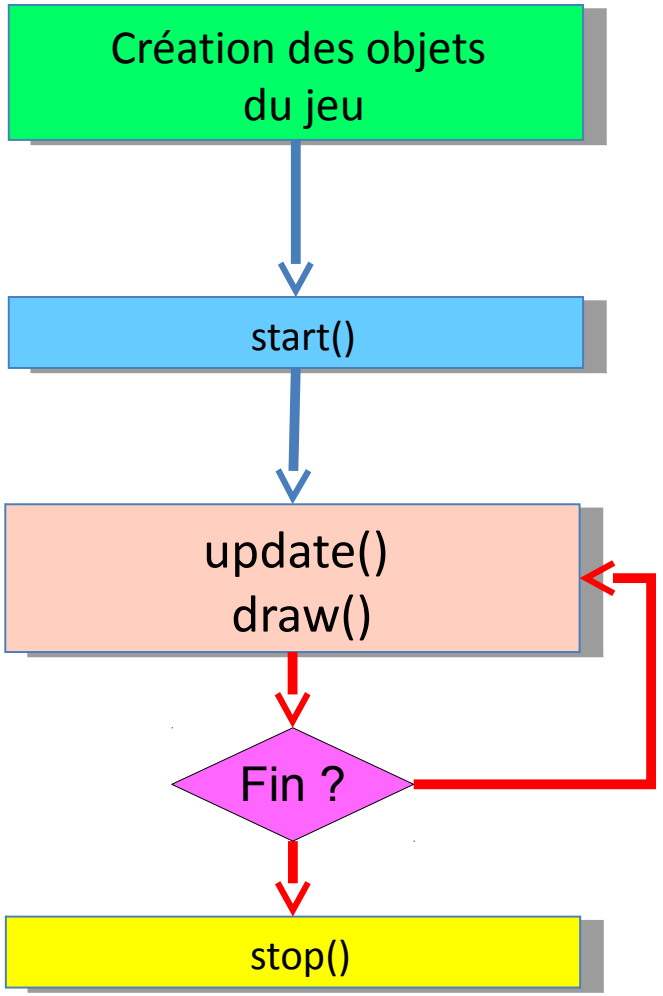
Moteur de jeu

- Éléments
  - Boucle de jeu
  - Gestion du temps
  - Gestion de l'affichage
  - Gestion du controle

- Possibilité
  - Construire from scratch
  - Utiliser libraries
    - Python : pygame, ...
    - Java: slick2D, ...

Moteur de jeu

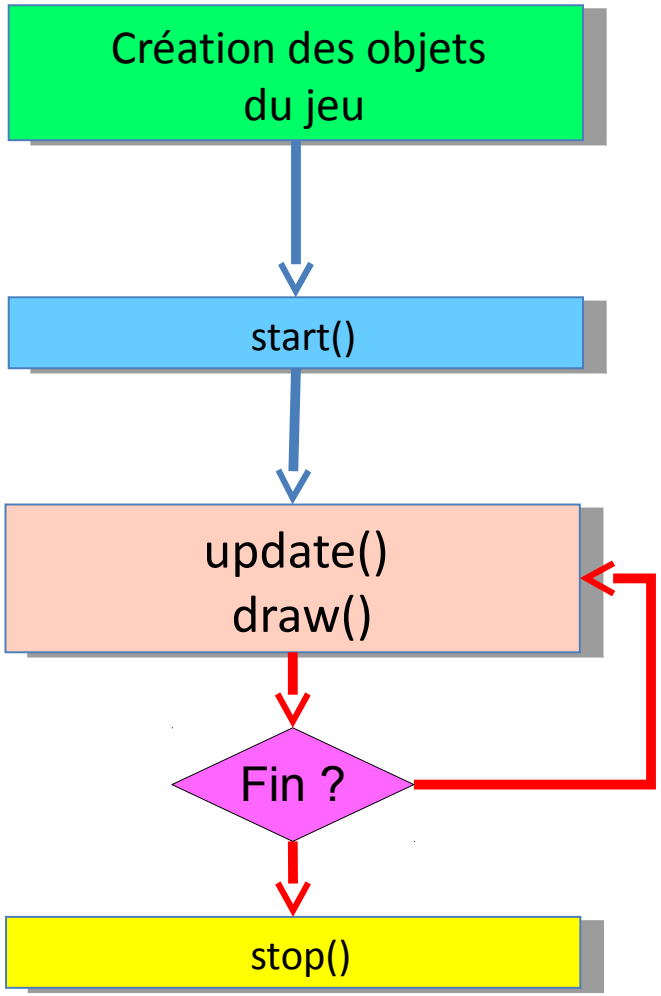
- Boucle de jeu



Boucle principale du jeu

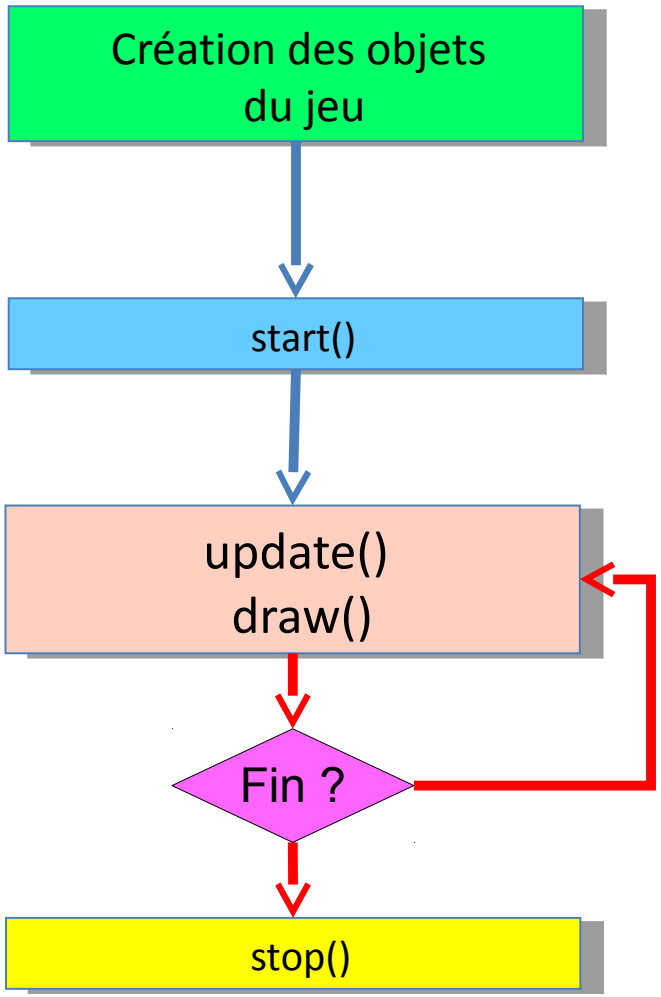
Moteur de jeu

- pygame



Moteur de jeu

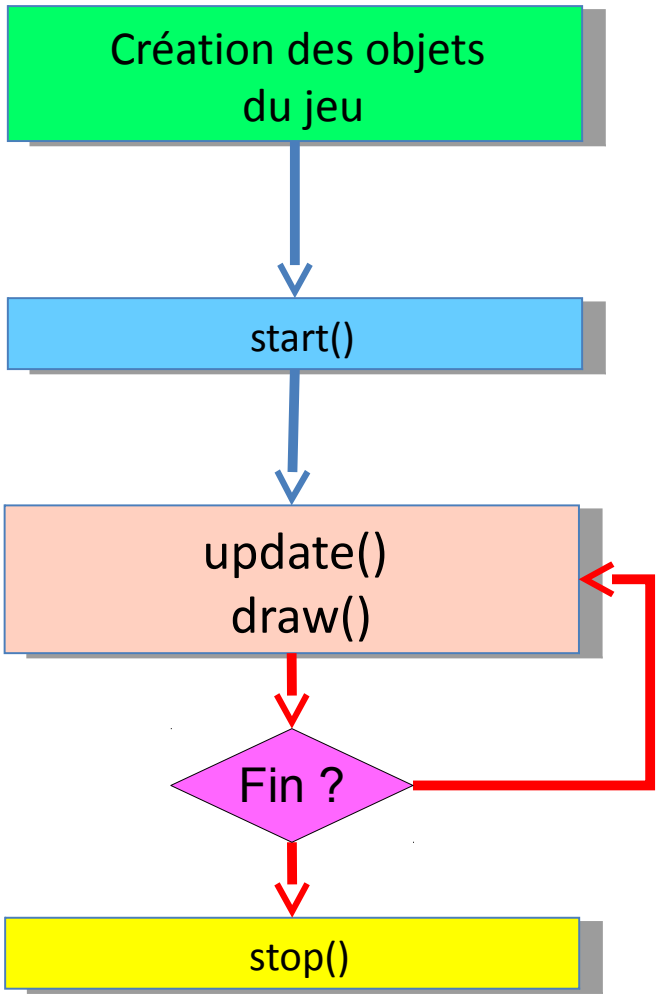
- pygame




```
# creer la fenetre
pygame.init()
screen = pygame.display.set_mode((700,500))
pygame.display.set_caption("Moteur")

# creer jeu
jeu = new Jeu()

# creer horloge pour vitesse de la boucle de jeu
clock = pygame.time.Clock()

# -------- Boucle principale -----------
# tant que le jeu n'est pas fini
while not jeu.etreFini():

    # --- on traite les evenements
    jeu.traiter_evenement()

    # --- on fait evoluer le jeu
    jeu.evoluer()

    # --- on dessine le jeu
    jeu.dessiner()
    pygame.display.flip()

    # --- on demande d'attendre pour un FPS de 60
    clock.tick(60)
# -------- Fin Boucle principale -----------
pygame.quit()
```

Moteur de jeu

- pygame




```
# creer la fenetre
pygame.init()
screen = pygame.display.set_mode((700,500))
pygame.display.set_caption("Moteur")

# creer jeu
jeu = new Jeu()

# creer horloge pour vitesse de la boucle de jeu
clock = pygame.time.Clock()

# -------- Boucle principale -----------
# tant que le jeu n'est pas fini
while not jeu.etreFini():

    # --- on traite les evenements
    jeu.traiter_evenement()

    # --- on fait evoluer le jeu (update)
    jeu.evoluer()

    # --- on dessine le jeu (draw)
    jeu.dessiner()
    pygame.display.flip()

    # --- on demande d'attendre pour un FPS de 60
    clock.tick(60)
# -------- Fin Boucle principale -----------
pygame.quit()
```

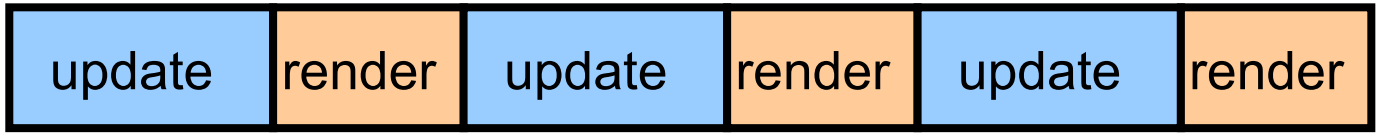
Moteur de jeu

- Éléments
  - Boucle de jeu
  - Gestion du temps
  - Gestion de l'affichage
  - Gestion du controle

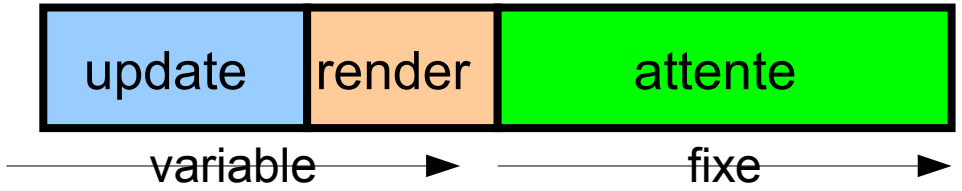
- Possibilité
  - Construire from scratch
  - Utiliser libraries
    - Python : pygame, ...
    - Java: slick2D, ...

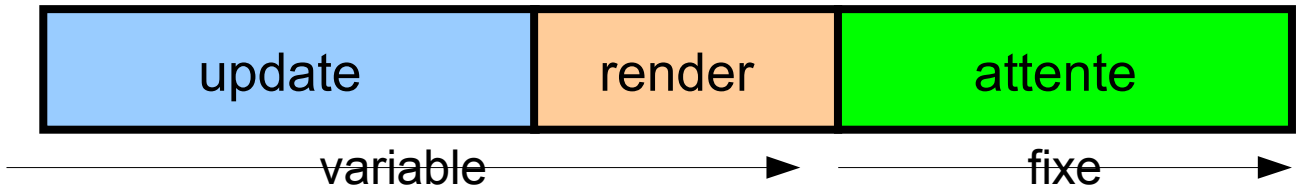
Moteur de jeu

- Gestion du temps
  - Sans attente
  - Probleme: vitesse du jeu indexée sur machine



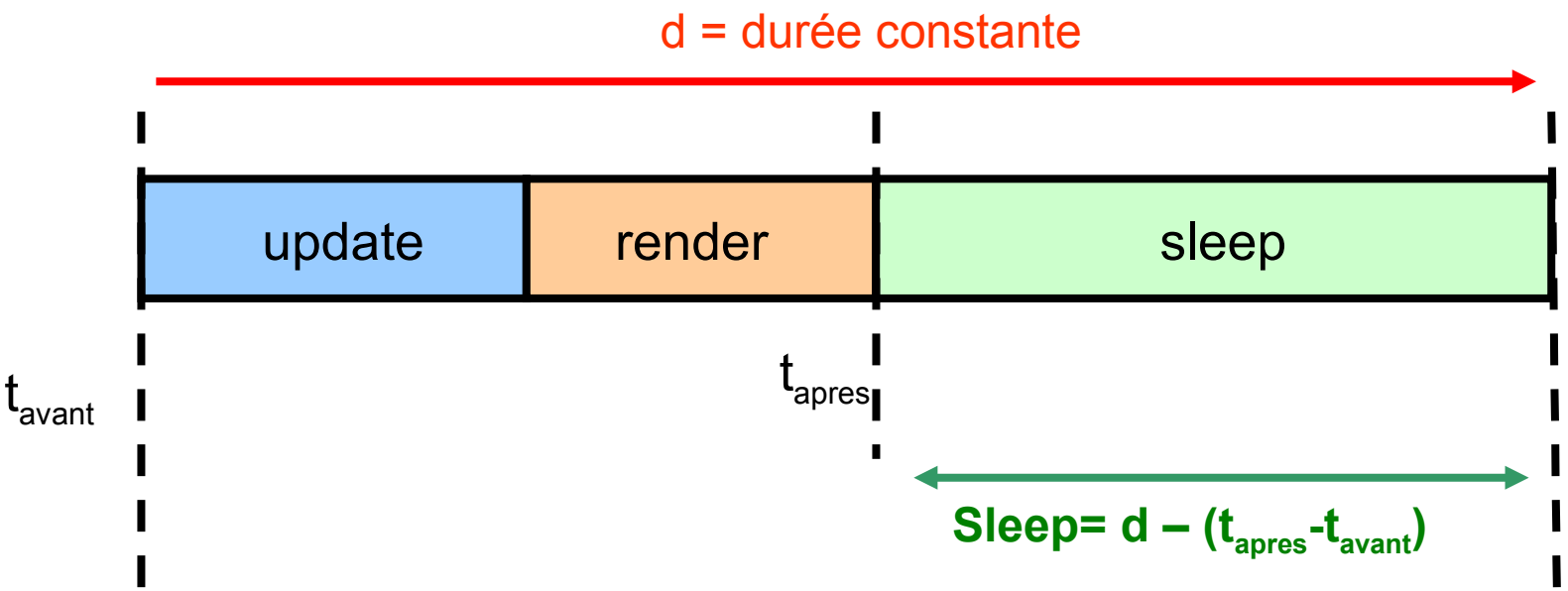
- Temps d'attente constant





Moteur de jeu

- Gestion du temps
  - Mise à jour reguliere

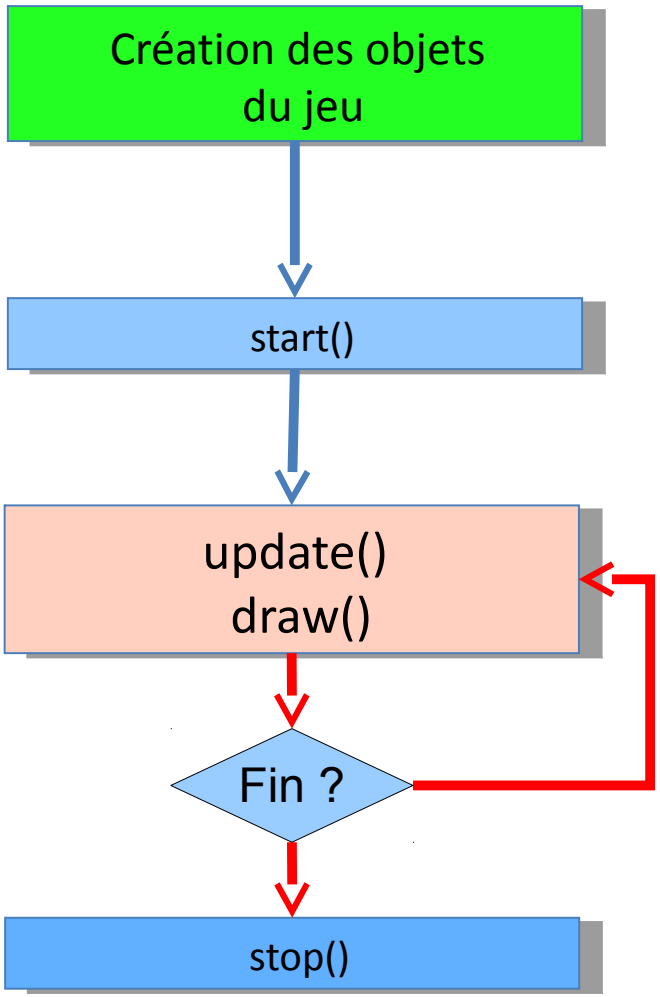


  - Sous pygame

```
clock.tick(60)
```

Moteur de jeu

- pygame




```
# creer la fenetre
pygame.init()
screen = pygame.display.set_mode((700,500))
pygame.display.set_caption("Moteur")

# creer jeu
jeu = new Jeu()

# creer horloge pour vitesse de la boucle de jeu
clock = pygame.time.Clock()

# -------- Boucle principale -----------
# tant que le jeu n'est pas fini
while not jeu.etreFini():

    # --- on traite les evenements
    jeu.traiter_evenement()

    # --- on fait evoluer le jeu
    jeu.evoluer()

    # --- on dessine le jeu
    jeu.dessiner()
    pygame.display.flip()

    # --- on demande d'attendre pour un FPS de 60
    clock.tick(60)
# -------- Fin Boucle principale -----------
pygame.quit()
```

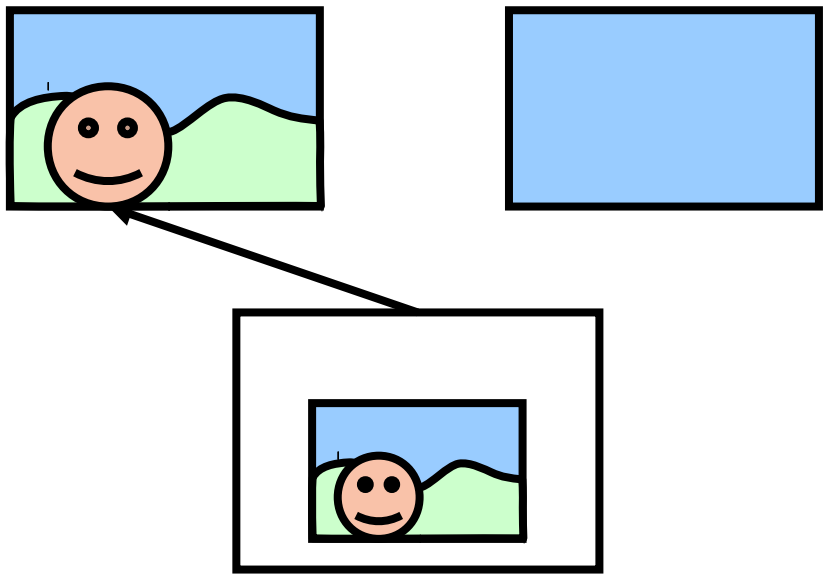
Moteur de jeu

- Éléments
  - Boucle de jeu
  - Gestion du temps
  - Gestion de l'affichage
  - Gestion du controle

- Possibilité
  - Construire from scratch
  - Utiliser libraries
    - Python : pygame, ...
    - Java: slick2D, ...

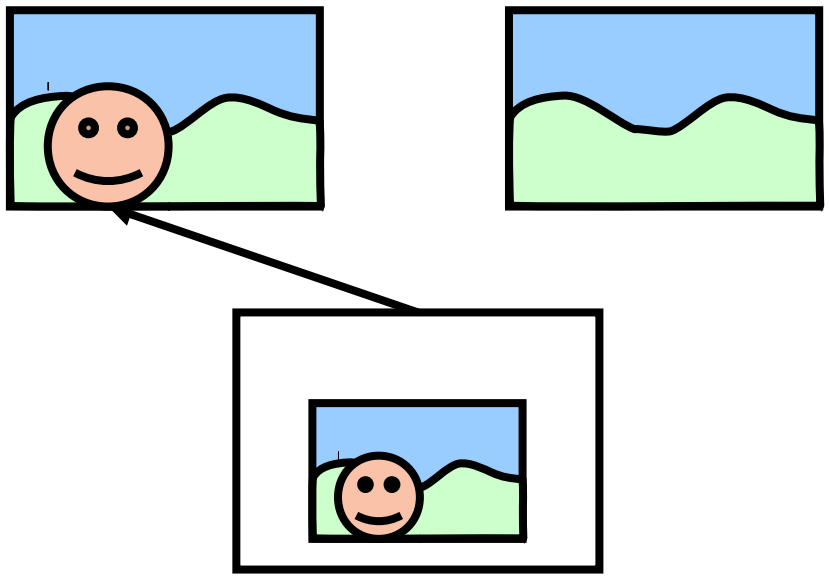
Moteur de jeu

- Double buffering
  - Une image affichée, une image cachée



Moteur de jeu

- Double buffering
  - Une image affichée, une image cachée

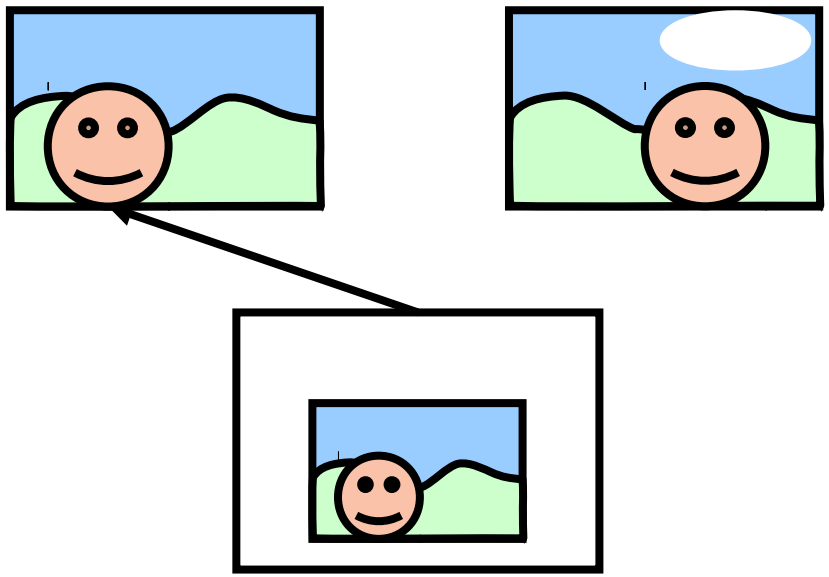


Méthode de dessin Pygame

```
pygame.draw.rect()
```

Moteur de jeu

- Double buffering
  - Une image affichée, une image cachée

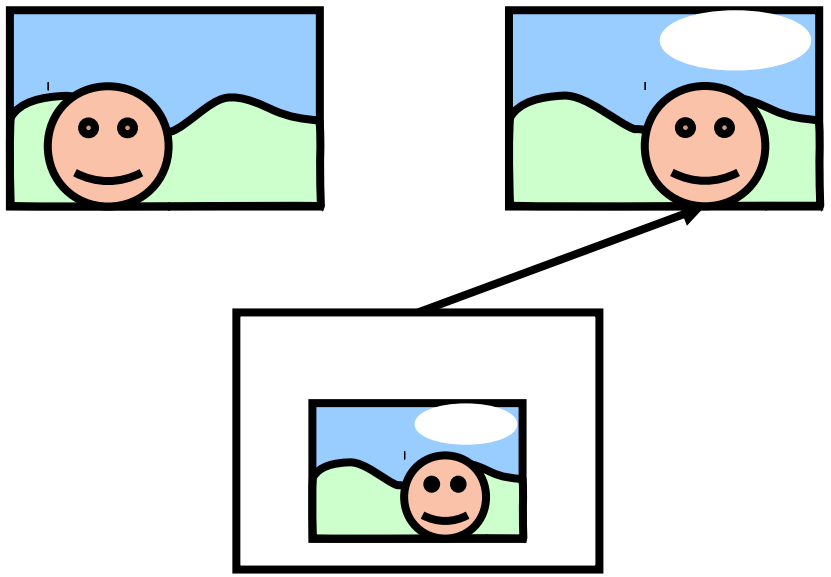


Méthode de dessin Pygame

```
pygame.draw.rect()
```

Moteur de jeu

- Double buffering
  - Une image affichée, une image cachée
  - On "inverse" les images

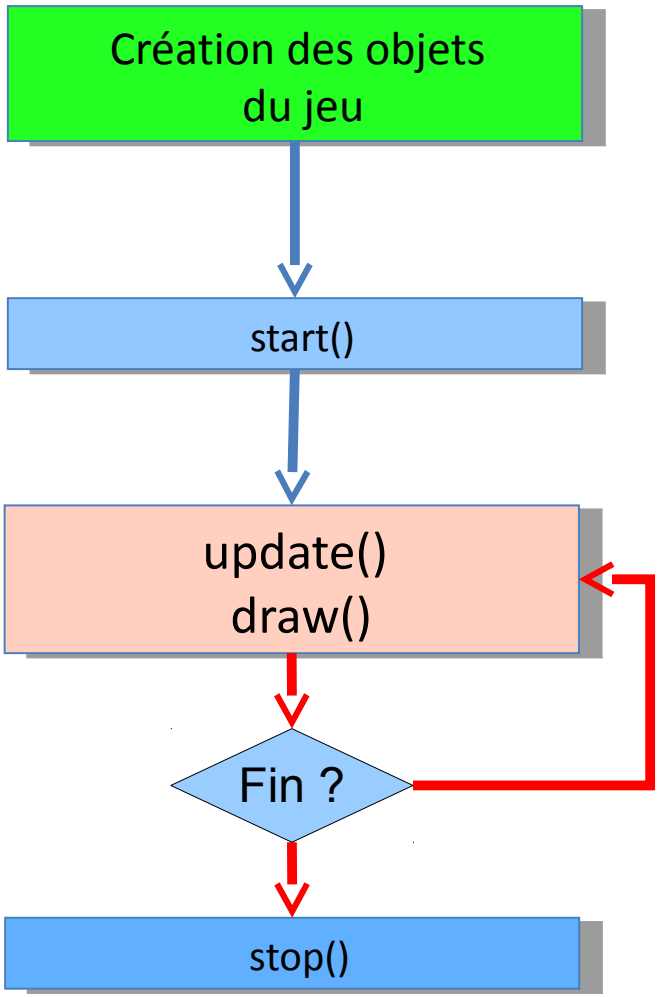


Inversion écrans Pygame

```
pygame.display.flip()
```

Moteur de jeu

- pygame




```
# creer la fenetre
pygame.init()
screen = pygame.display.set_mode((700,500))
pygame.display.set_caption("Moteur")

# creer jeu
jeu = new Jeu()

# creer horloge pour vitesse de la boucle de jeu
clock = pygame.time.Clock()

# -------- Boucle principale -----------
# tant que le jeu n'est pas fini
while not jeu.etreFini():

    # --- on traite les evenements
    jeu.traiter_evenement()

    # --- on fait evoluer le jeu
    jeu.evoluer()

    # --- on dessine le jeu
    jeu.dessiner()
    pygame.display.flip()

    # --- on demande d'attendre pour un FPS de 60
    clock.tick(60)
# -------- Fin Boucle principale -----------
pygame.quit()
```

Moteur de jeu

- Documentation pygame
  - Méthode fournies par la librairie pygame
  - http://pygame.org/docs/

- Méthodes de dessin
  - http://pygame.org/docs/ref/draw.html

```
pygame.draw.rect        -  draw a rectangle shape
pygame.draw.polygon     -  draw a shape with any number of sides
pygame.draw.circle      -  draw a circle around a point
pygame.draw.ellipse     -  draw a round shape inside a rectangle
pygame.draw.arc         -  draw a partial section of an ellipse
pygame.draw.line        -  draw a straight line segment
pygame.draw.lines       -  draw multiple contiguous line segments
pygame.draw.aaline      -  draw fine antialiased lines
pygame.draw.aalines
```

Moteur de jeu

```
def dessiner(self):

    # couleurs
    WHITE = (0xFF, 0xFF, 0xFF)
    BLUE = (0x00, 0x00, 0xFF)

    # affiche rectangle blanc
    pygame.draw.rect(screen,WHITE,(100,100,300,200))

    # affiche point en x cercle bleu
    dx = (self.x)%290
    pygame.draw.ellipse(screen,BLUE,(100+dx,250,10,10))

    # dessin texte
    myfont = pygame.font.SysFont("monospace", 12)
    label = myfont.render("Appuyer sur espace", 1, BLUE)
    screen.blit(label, (150, 150))

    # dessin num frame
    label = myfont.render("num frame = "+str(self.x), 1, BLUE)
    screen.blit(label, (200, 200))
```

Moteur de jeu

- Éléments
  - Boucle de jeu
  - Gestion du temps
  - Gestion de l'affichage
  - Gestion du controle

- Possibilité
  - Construire from scratch
  - Utiliser libraries
    - Python : pygame, ...
    - Java: slick2D, ...

Moteur de jeu

- Gestion des evenements

- Pygame cache aspect evenementiel
  - Evenements dans liste `pygame.event.get()`

```
# --- on traite les evenements
for event in pygame.event.get():

        #si l'utilisateur arrete
        if event.type == pygame.QUIT:
                [...]

        #si l'utilisateur appuie
        if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_UP:
                        [...]
```

Moteur de jeu

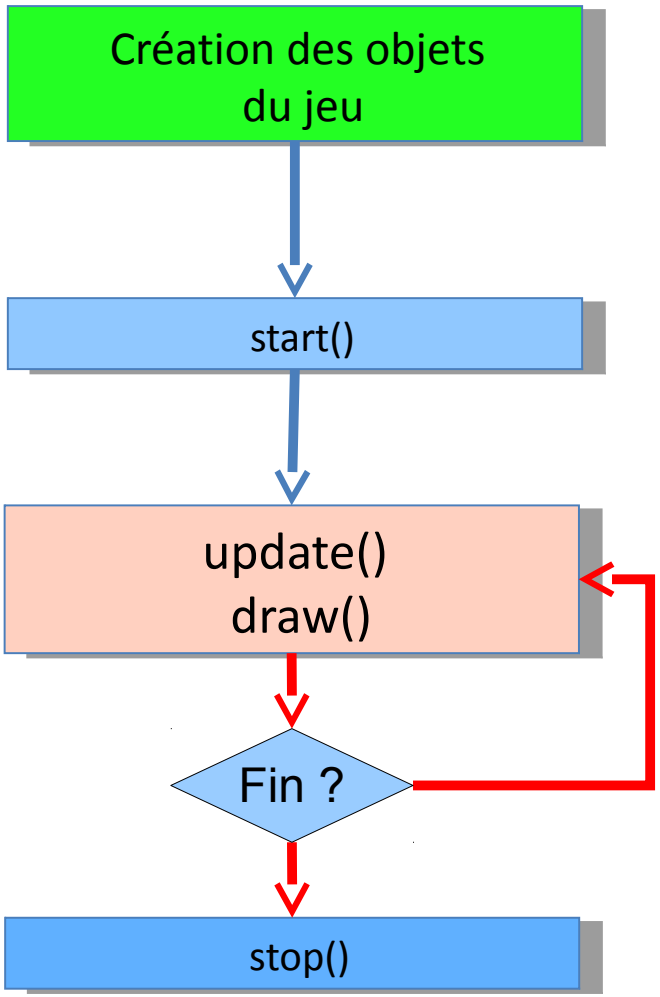
- https://www.pygame.org/docs/ref/event.html#pygame.event
- Fin
  - `pygame.QUIT`
- Souris
  - `pygame.MOUSEMOTION` `pos, rel, buttons`
  - `pygame.MOUSEBUTTONUP` `pos, button`
  - `pygame.MOUSEBUTTONDOWN` `pos, button`
- Clavier
  - `pygame.KEYDOWN` `unicode, key, mod`
  - `pygame.KEYUP` `key, mod`

Moteur de jeu

```
# --- on traite les evenements
for event in pygame.event.get():

    #si l'utilisateur arrete (click sur croix)
    if event.type == pygame.QUIT:
        [...]

    #si l'utilisateur appuie sur touche
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_UP:
            print("Haut")
        if event.key == pygame.K_RETURN:
            print("Entrée")

    #si appui sur la souris + souris_x < 100
    if event.type == pygame.MOUSEBUTTONDOWN:
        if (event.button == 1) and (event.pos[0]<100):
            print ("bouton gauche")

    #si deplace la souris en traversant x==100
    if event.type == pygame.MOUSEMOTION:
        if event.pos[0] == 100:
            print("bouton gauche")
```

Moteur de jeu

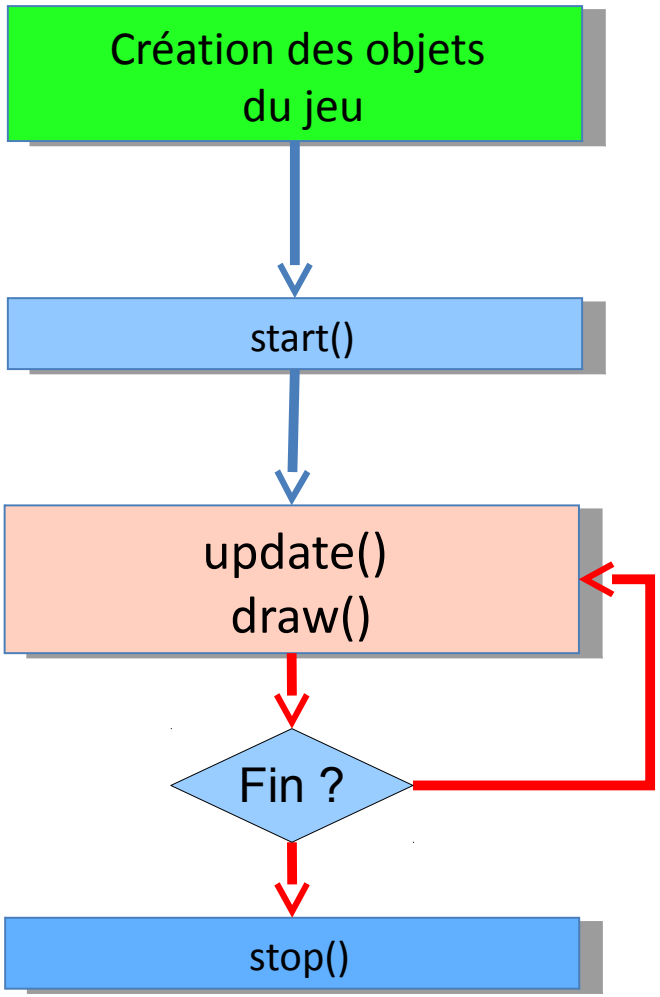
- pygame




```
# creer la fenetre
pygame.init()
screen = pygame.display.set_mode((700,500))
pygame.display.set_caption("Moteur")

# creer jeu
jeu = new Jeu()

# creer horloge pour vitesse de la boucle de jeu
clock = pygame.time.Clock()

# -------- Boucle principale -----------
# tant que le jeu n'est pas fini
while not jeu.etreFini():

    # --- on traite les evenements
    jeu.traiter_evenement()

    # --- on fait evoluer le jeu
    jeu.evoluer()

    # --- on dessine le jeu
    jeu.dessiner()
    pygame.display.flip()

    # --- on demande d'attendre pour un FPS de 60
    clock.tick(60)
# -------- Fin Boucle principale -----------
pygame.quit()
```

Moteur de jeu

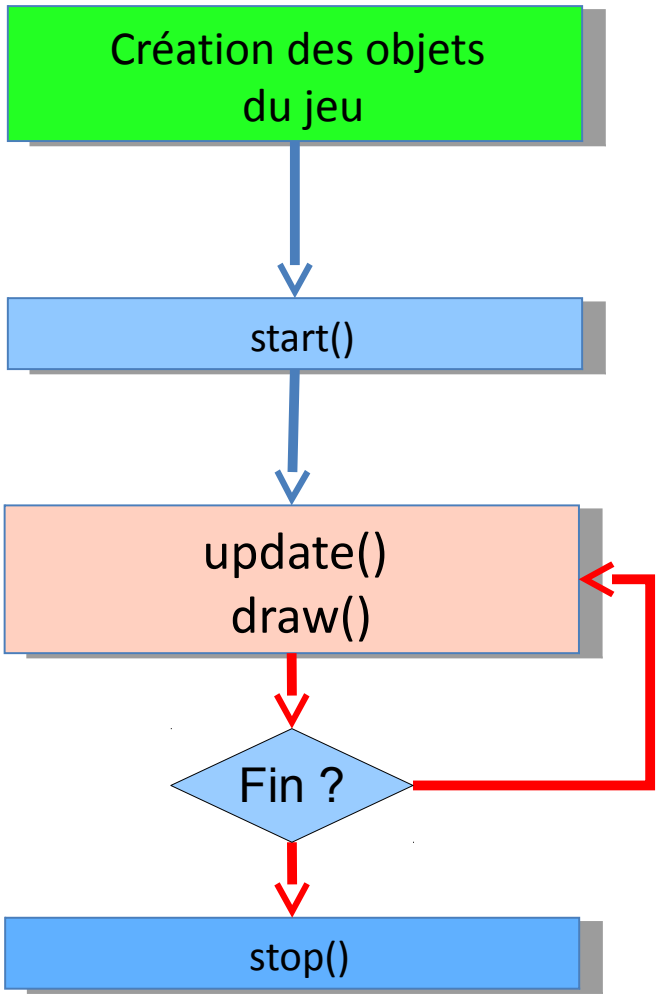
- pygame




```
# creer la fenetre
pygame.init()
screen = pygame.display.set_mode((700,500))
pygame.display.set_caption("Moteur")

# creer jeu
jeu = new Jeu()

# creer horloge pour vitesse de la boucle de jeu
clock = pygame.time.Clock()

# -------- Boucle principale -----------
# tant que le jeu n'est pas fini
while not jeu.etreFini():

    # --- on traite les evenements
    jeu.traiter_evenement()

    # --- on fait evoluer le jeu
    jeu.evoluer()

    # --- on dessine le jeu
    jeu.dessiner()
    pygame.display.flip()

    # --- on demande d'attendre pour un FPS de 60
    clock.tick(60)
# -------- Fin Boucle principale -----------
pygame.quit()
```

Moteur de jeu

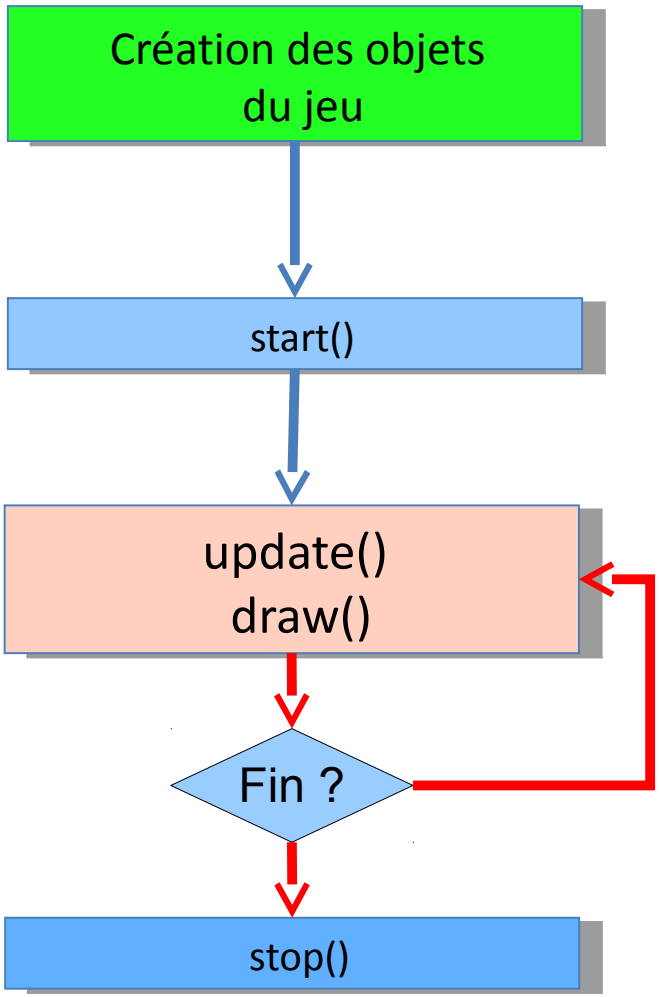
- pygame




```
# creer la fenetre
pygame.init()
screen = pygame.display.set_mode((700,500))
pygame.display.set_caption("Moteur")

# creer jeu
jeu = new Jeu()

# creer horloge pour vitesse de la boucle de jeu
clock = pygame.time.Clock()

# -------- Boucle principale -----------
# tant que le jeu n'est pas fini
while not jeu.etreFini():

    # --- on traite les evenements
    jeu.traiter_evenement()

    # --- on fait evoluer le jeu
    jeu.evoluer()

    # --- on dessine le jeu
    jeu.dessiner()
    pygame.display.flip()

    # --- on demande d'attendre pour un FPS de 60
    clock.tick(60)
# -------- Fin Boucle principale -----------
pygame.quit()
```

Moteur de jeu

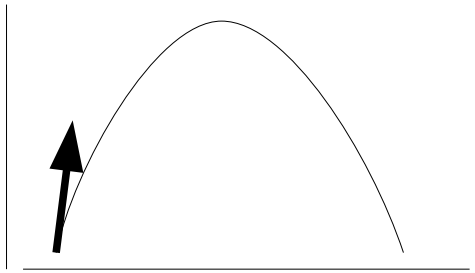
- pygame




```
# creer la fenetre
pygame.init()
screen = pygame.display.set_mode((700,500))
pygame.display.set_caption("Moteur")

# creer jeu
jeu = new Jeu()                                    ?

# creer horloge pour vitesse de la boucle de jeu
clock = pygame.time.Clock()

# -------- Boucle principale -----------
# tant que le jeu n'est pas fini
while not jeu.etreFini():                          ?

    # --- on traite les evenements
    jeu.traiter_evenement()

    # --- on fait evoluer le jeu
    jeu.evoluer()                                  ?

    # --- on dessine le jeu
    jeu.dessiner()
    pygame.display.flip()

    # --- on demande d'attendre pour un FPS de 60
    clock.tick(60)
# -------- Fin Boucle principale -----------
pygame.quit()
```

Moteur de jeu

- pygame

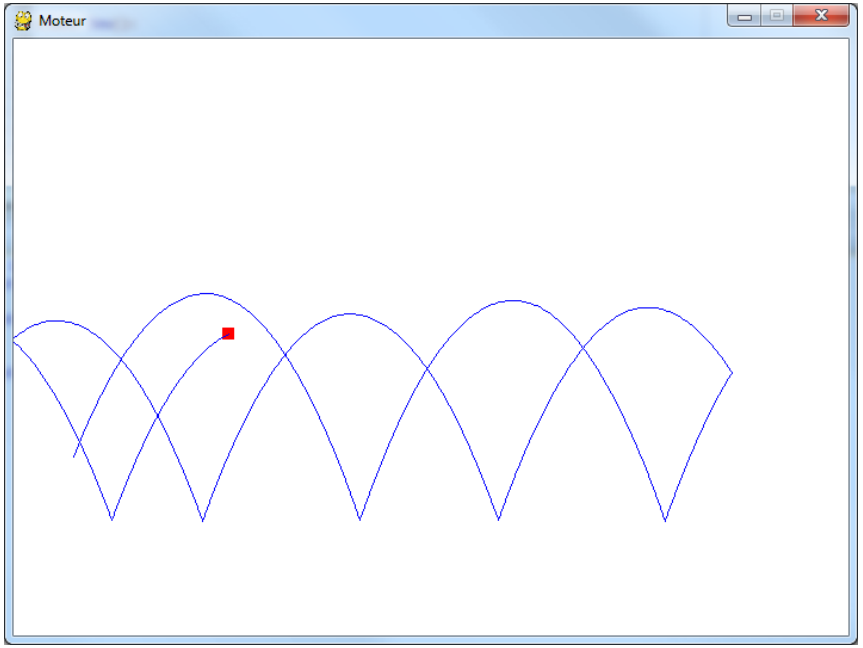


$$m\vec{a} = \sum \vec{F}$$
$$d\vec{v} = \vec{a}.dt$$
$$d\vec{x} = \vec{v}.dt$$

```
# creer la fenetre
pygame.init()
screen = pygame.display.set_mode((700,500))
pygame.display.set_caption("Moteur")

# creer jeu                                          ?
jeu = new Jeu()

# creer horloge pour vitesse de la boucle de jeu
clock = pygame.time.Clock()

# -------- Boucle principale -----------
# tant que le jeu n'est pas fini                     ?
while not jeu.etreFini():

    # --- on traite les evenements
    jeu.traiter_evenement()

    # --- on fait evoluer le jeu                     ?
    jeu.evoluer()

    # --- on dessine le jeu
    jeu.dessiner()
    pygame.display.flip()

    # --- on demande d'attendre pour un FPS de 60
    clock.tick(60)
# -------- Fin Boucle principale -----------
pygame.quit()
```

Exercice

- Objectif
  - Utiliser le squelette fourni
  - Faire le rendu du moteur physique



Je vous laisse faire (15 min)

Physique du solide

```
class Jeu():

    def __init__(self):
        self.x=0
        self.y=0
        self.vx=50
        self.vy=10
        self.ax=0
        self.ay=-9
        self.dt=0.01

    def evoluer(self):
        self.x=self.x+self.vx*self.dt
        self.y=self.y+self.vy*self.dt
        self.vx=self.vx+self.ax*self.dt
        self.vy=self.vy+self.ay*self.dt

    def afficher(self):
        print("(x,y): ",sel
        print("(vx,vy):",self.vx,",",s


jeu=Jeu()
for i in range(1000):
    jeu.evoluer()
    jeu.afficher()
```

Remplacer par dessiner

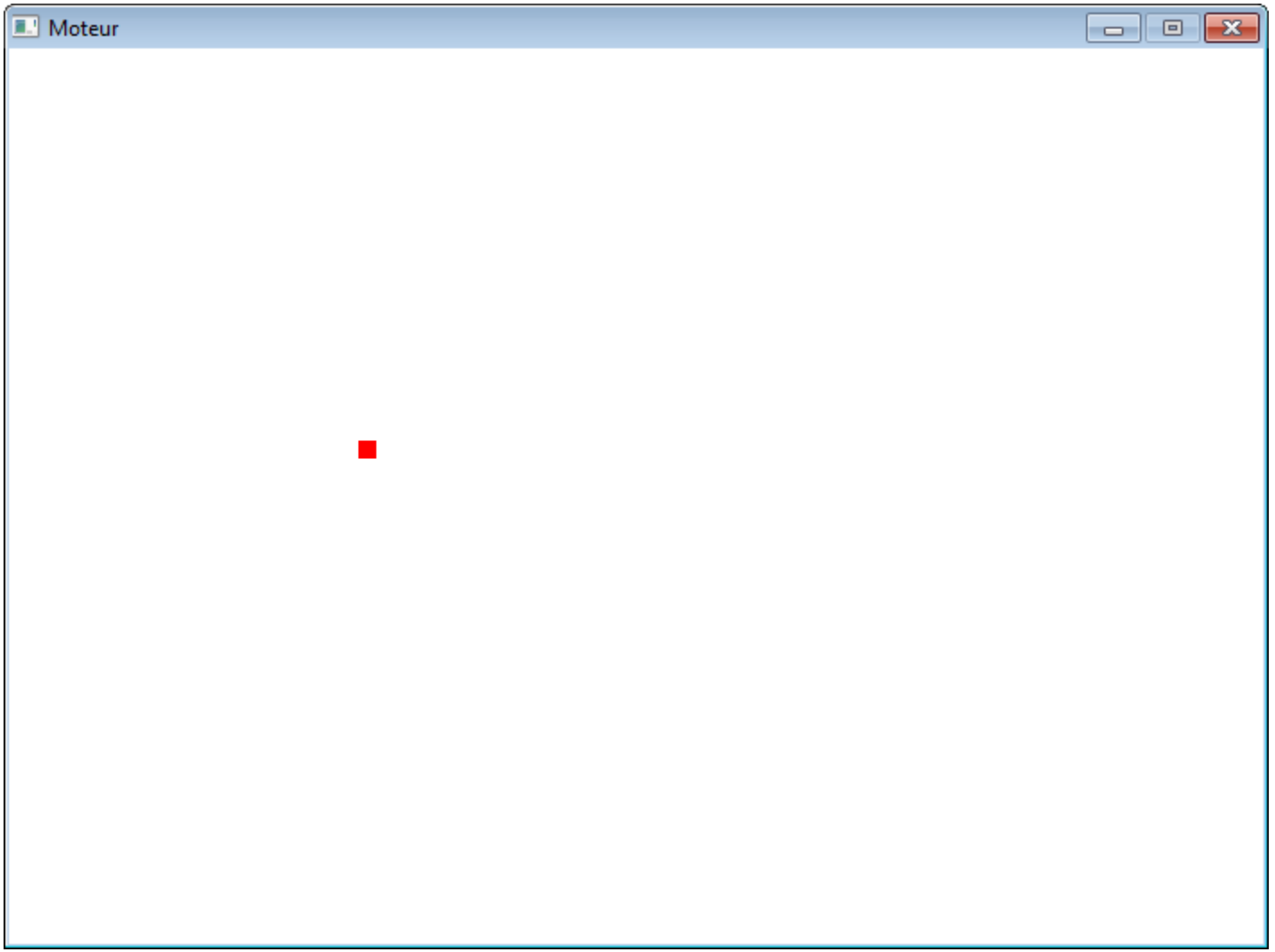
Remplacer par moteur jeu

Physique du solide

```
class Jeu():

    def __init__(self):
        self.x=0
        self.y=0
        self.vx=50
        self.vy=10
        self.ax=0
        self.ay=-9
        self.dt=0.01

    def evoluer(self):
        self.x=self.x+sel
        self.y=self.y+sel
        self.vx=self.vx+s
        self.vy=self.vy+s

    def afficher(self):
        print("(x,y): ",s
        print("(vx,vy):",


jeu=Jeu()
for i in range(1000):
    jeu.evoluer()
    jeu.afficher()
```

```
# creer la fenetre
pygame.init()
screen = pygame.display.set_mode((700,500))
pygame.display.set_caption("Moteur")

# creer jeu
jeu = new Jeu()

# creer horloge pour vitesse de la boucle de jeu
clock = pygame.time.Clock()

# -------- Boucle principale -----------
# tant que le jeu n'est pas fini
while not jeu.etreFini():

    # --- on traite les evenements
    jeu.traiter_evenement()

    # --- on fait evoluer le jeu
    jeu.evoluer()

    # --- on dessine le jeu
    jeu.dessiner()
    pygame.display.flip()

    # --- on demande d'attendre pour un FPS de 60
    clock.tick(60)
# -------- Fin Boucle principale -----------
pygame.quit()
```

Affichage balle



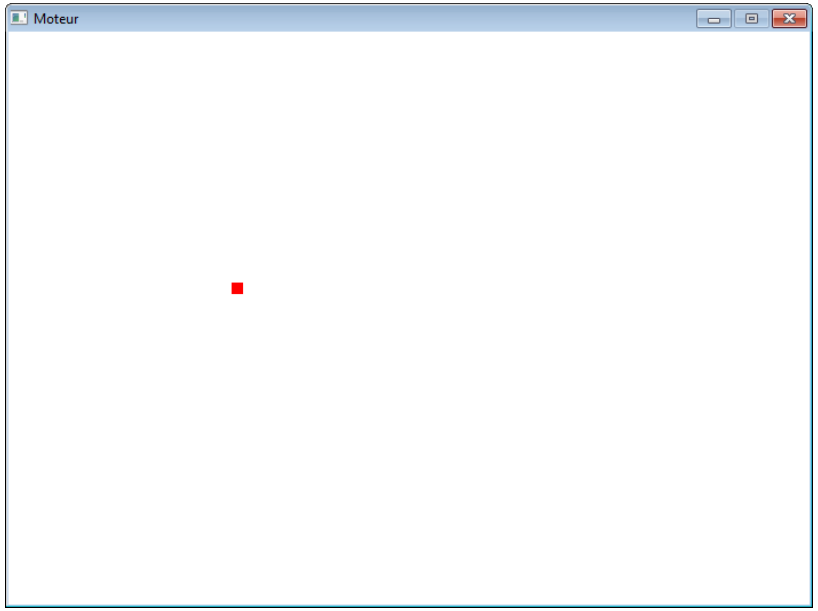
Affichage balle trajectoire

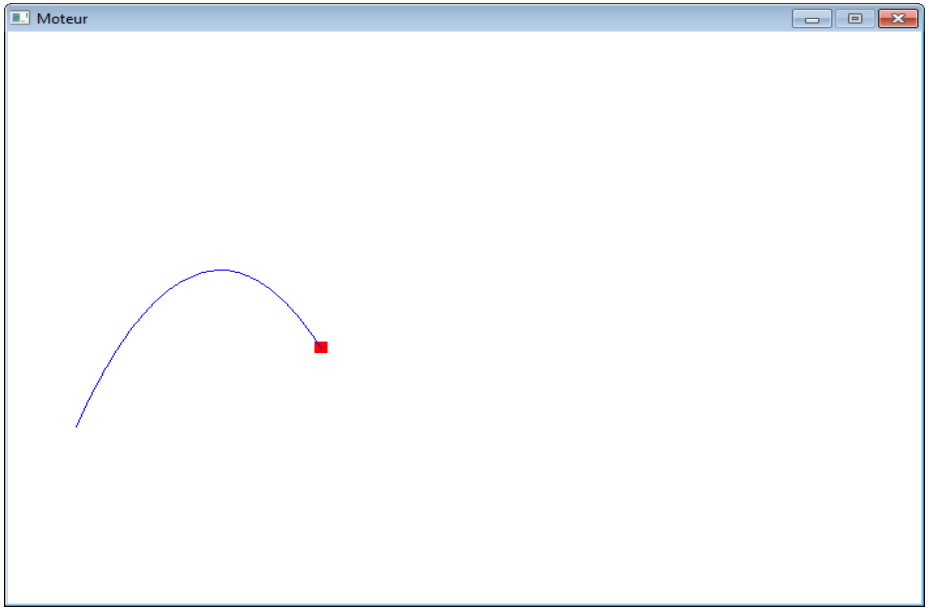
```
evoluer() :

#stocke nouvelles valeurs
self.trajX+=[self.x]
self.trajY+=[self.y]
```

```
__init__() :

#trajectoire vide
    self.trajX=[];
    self.trajY=[];
```

```
Dessiner() :

# pour chaque point
for i in range(0,len(self.trajX)-1):
    # segment
    x=self.trajX[i]
    y=self.trajY[i]
    coord=self.changerCoordonnes(x,y)
    x2=self.trajX[i+1]
    y2=self.trajY[i+1]
    coord2=self.changerCoordonnes(x2,y2)
    #trace segment
    pygame.draw.line(screen,BLUE,coord,coord2,1)
```

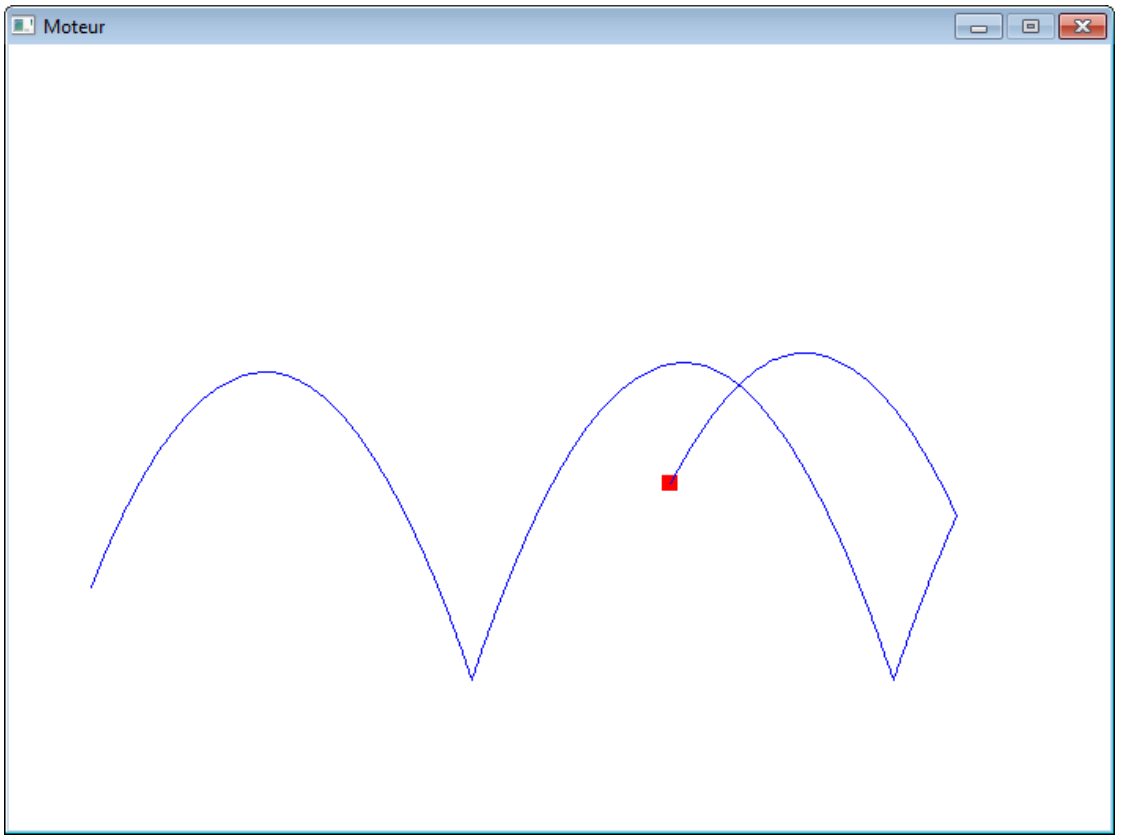
On peut faire mieux
(ici, on redessine tout)



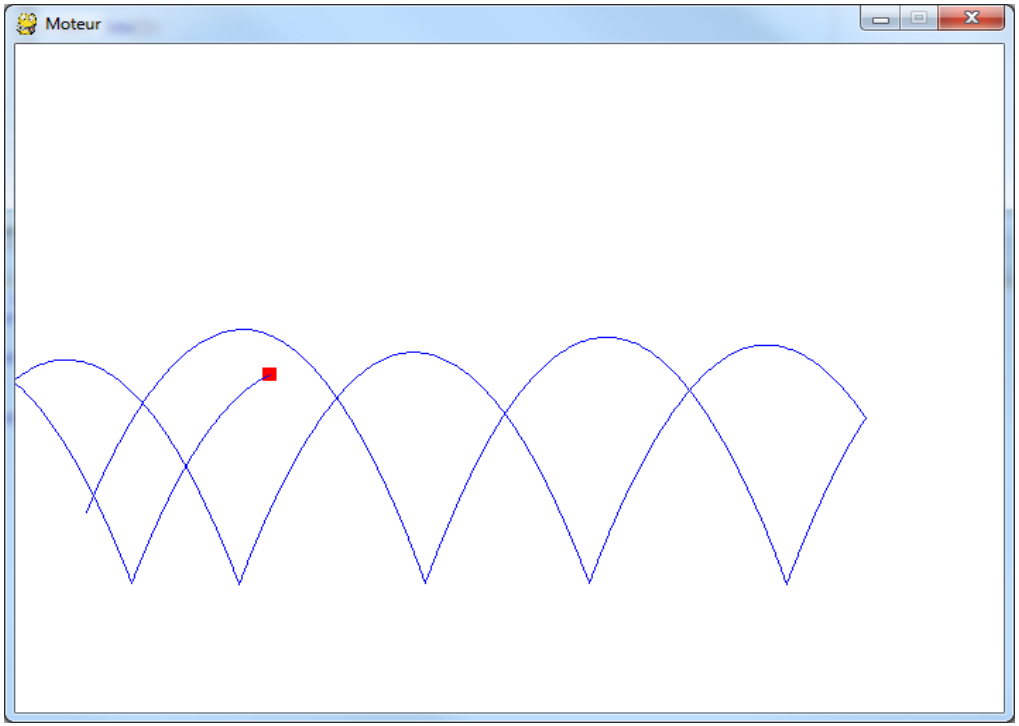


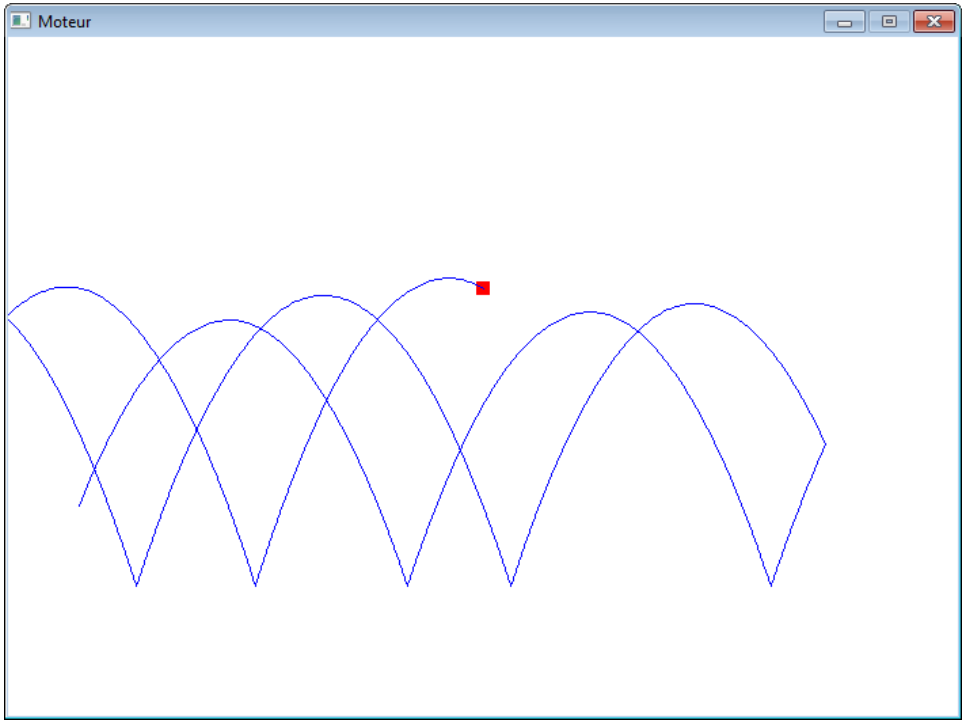
Faire un rebond

```
evoluer() :

#rebond sol
if (self.y<0):
    self.vy=-self.vy

#sortie horizontale
if(self.x>600):
    self.vx=-self.vx
if(self.x<00):
    self.vx=-self.vx
```



Probleme ?





Probleme ?

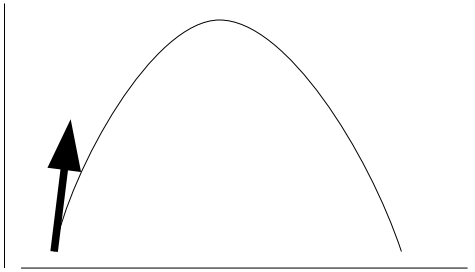
- Permettre accelerer le temps
  - Appui '+' ==> augmente FPS
  - Appui '-' ==> diminue FPS

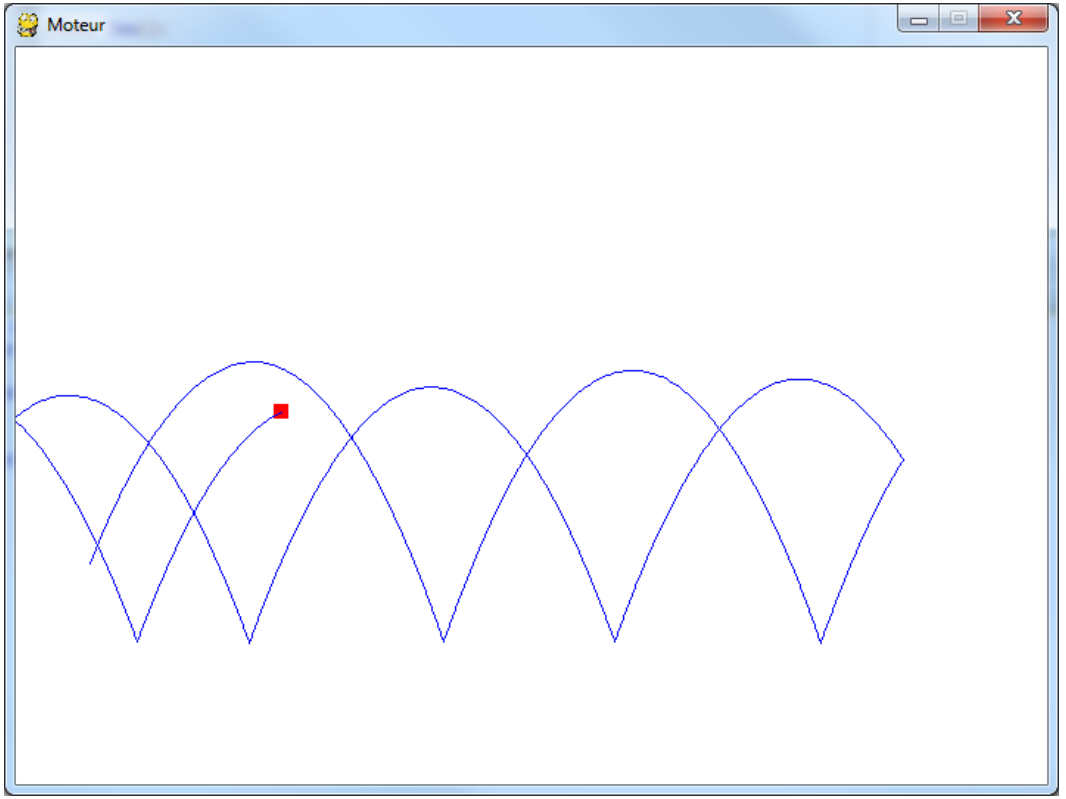
```
traiter_evenement() :

# si on appuie sur une touche
if event.type == pygame.KEYDOWN:

   # si on appuie sur +
   if event.key == pygame.K_KP_PLUS:
       self.fps=self.fps+10
       print("accelere FPS→",self.fps)

   # si on appuie sur +
   if event.key == pygame.K_KP_MINUS and self.fps>20:
       self.fps=self.fps-10
       print("ralenti FPS->",self.fps)
```

```
clock.tick(FPS)
```

Physique du solide

- Gravité constante
  - Erreurs approximation

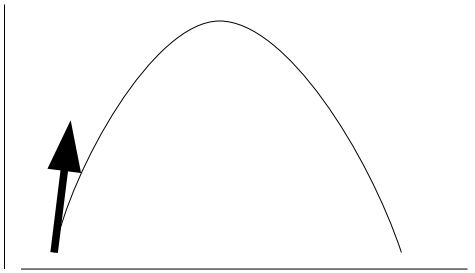


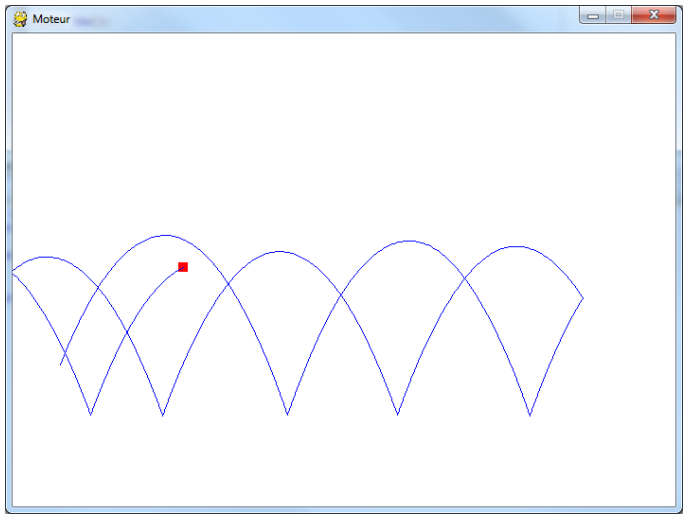


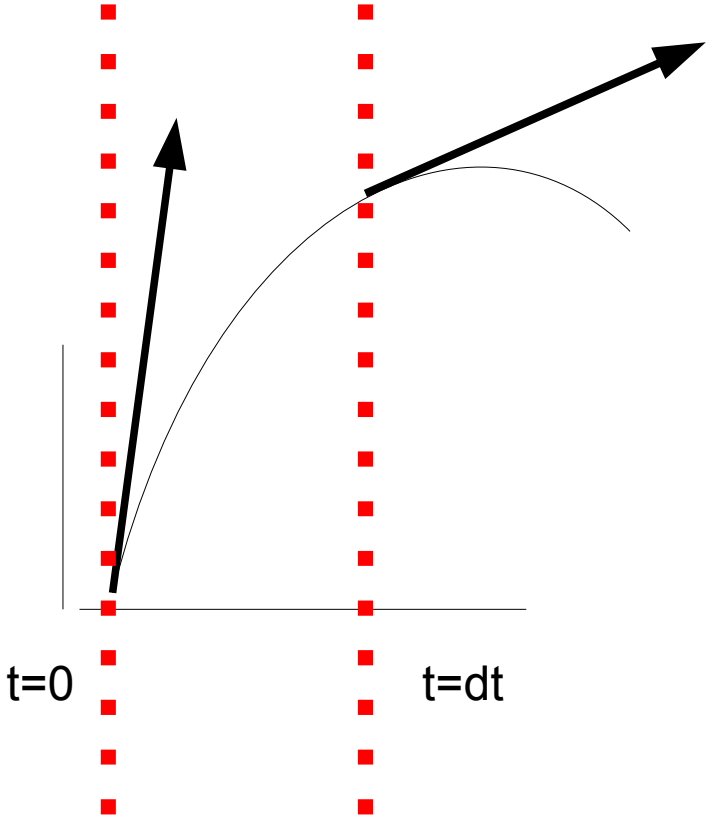
Perte d'énergie due aux approximations

Physique du solide

- Gravité constante
  - Erreurs approximation

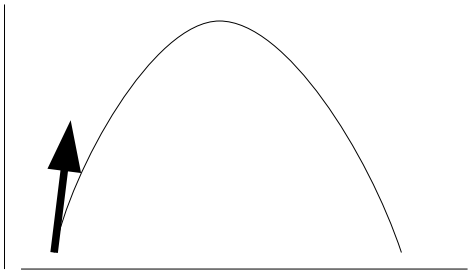


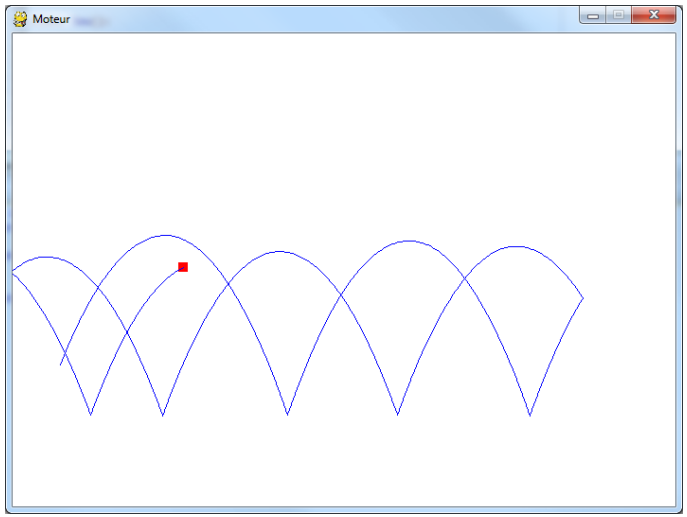


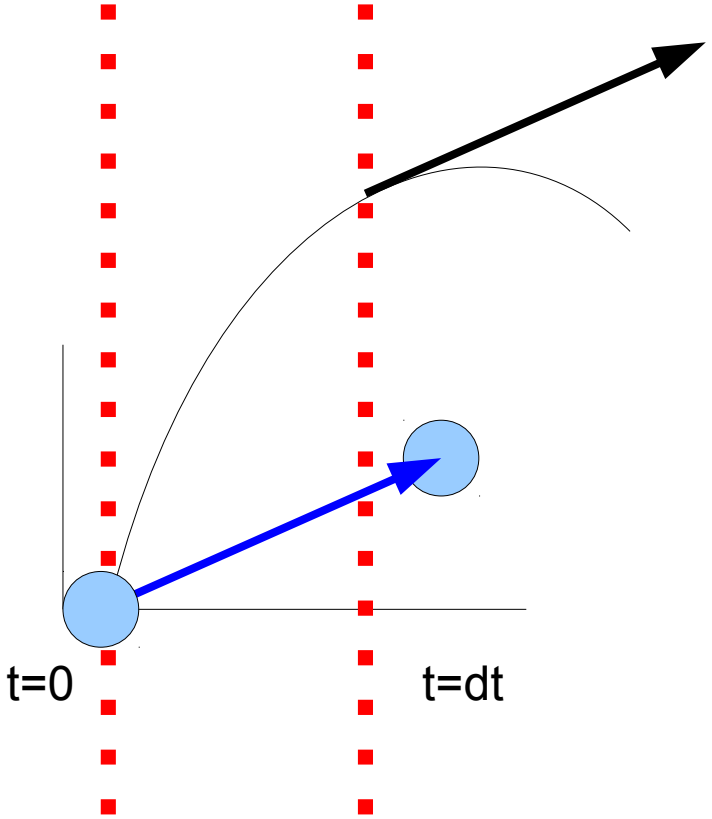


Physique du solide

- Gravité constante
  - Erreurs approximation



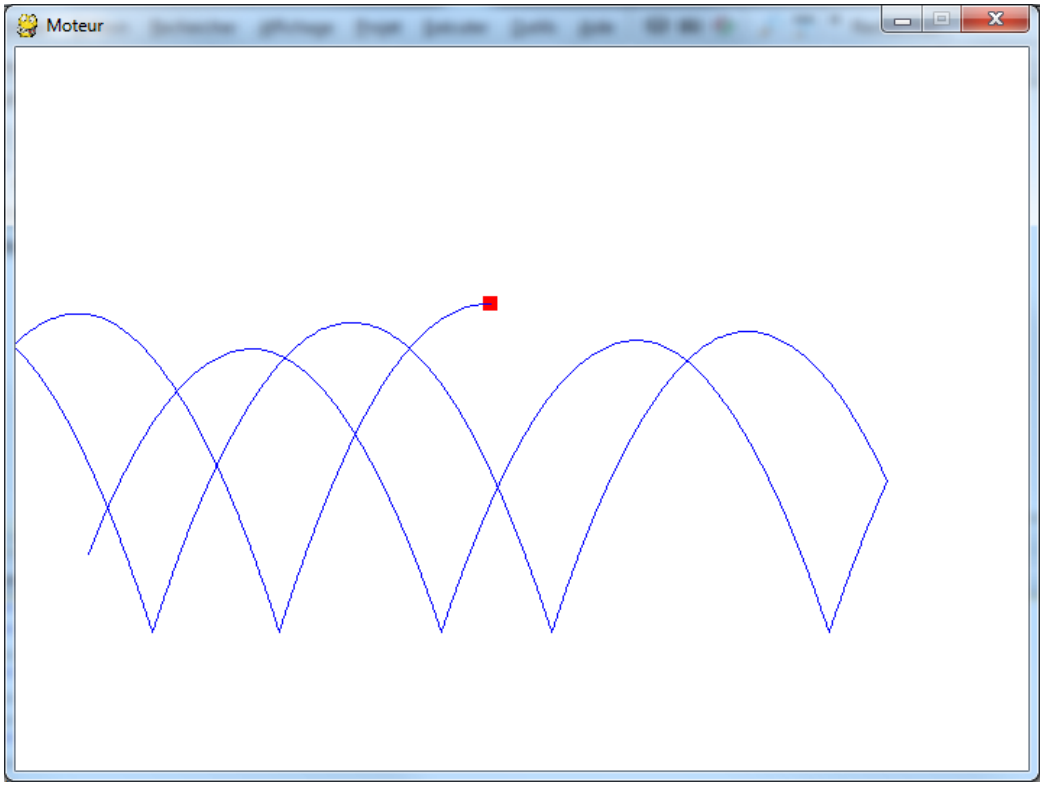


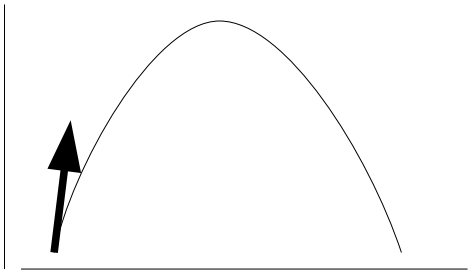


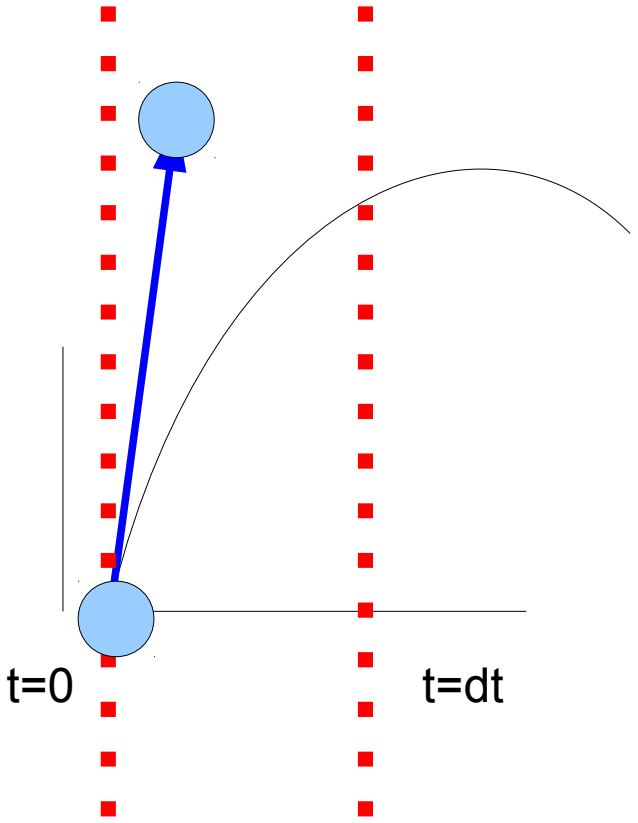
Vitesse mise à jour avant position

Physique du solide

- Gravité constante
  - Erreurs approximation



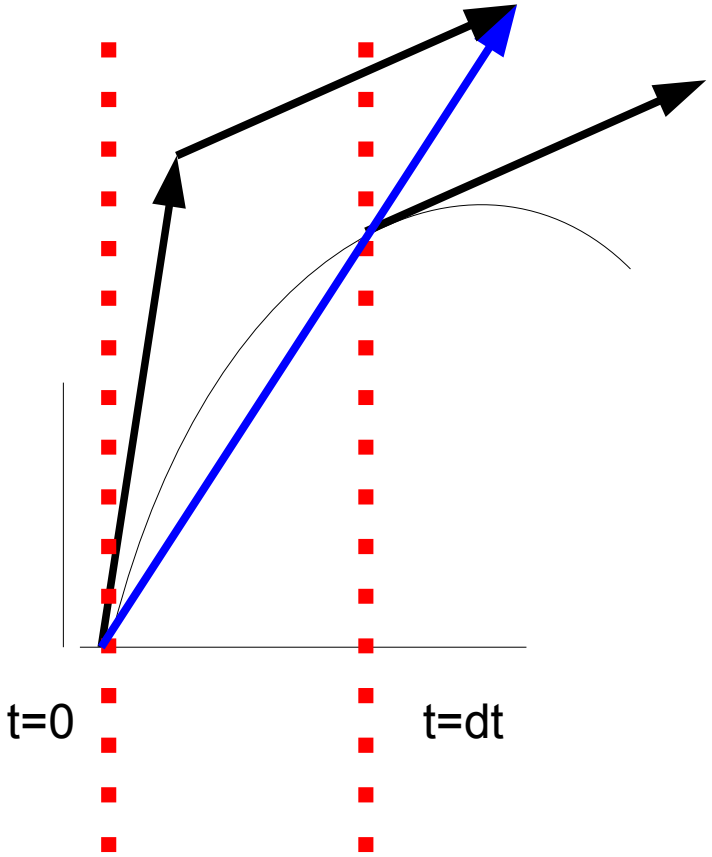




Position mise à jour avant vitesse
= augmentation energie

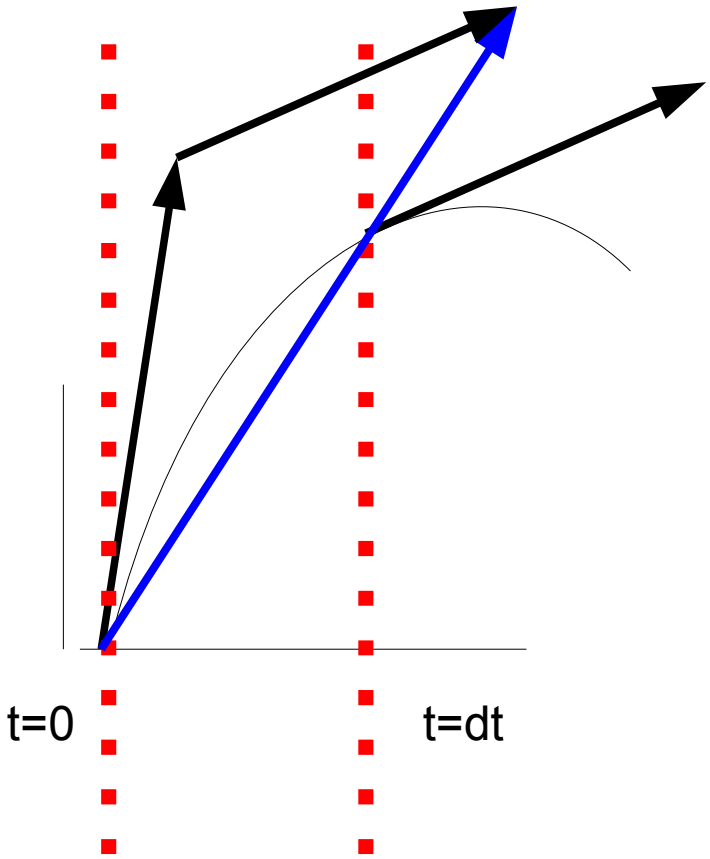
Physique du solide

- Gravité constante
  - Approximation d'ordre 2
  - Moyenne des vitesses



Physique du solide

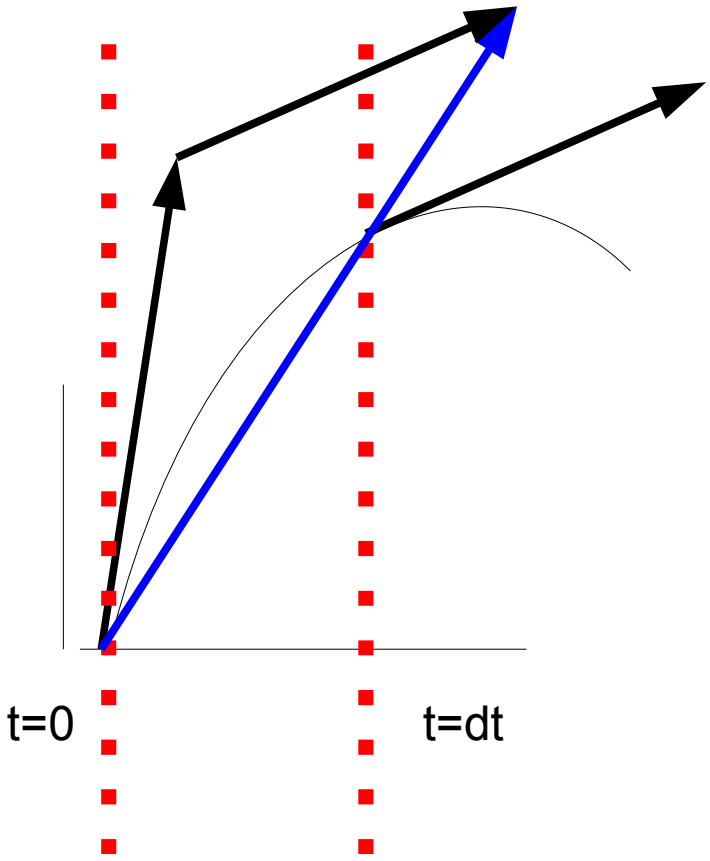
- Gravité constante
  - Approximation d'ordre 2
  - Moyenne des vitesses




```
old_vx=self.vx
old_vy=self.vy
self.vx=self.vx+self.ax*self.dt
self.vy=self.vy+self.ay*self.dt
self.x=self.x+(self.vx+old_vx)/2*self.dt
self.y=self.y+(self.vy+old_vy)/2*self.dt
```

Physique du solide

- Gravité constante
  - Approximation d'ordre 2
  - Moyenne des vitesses



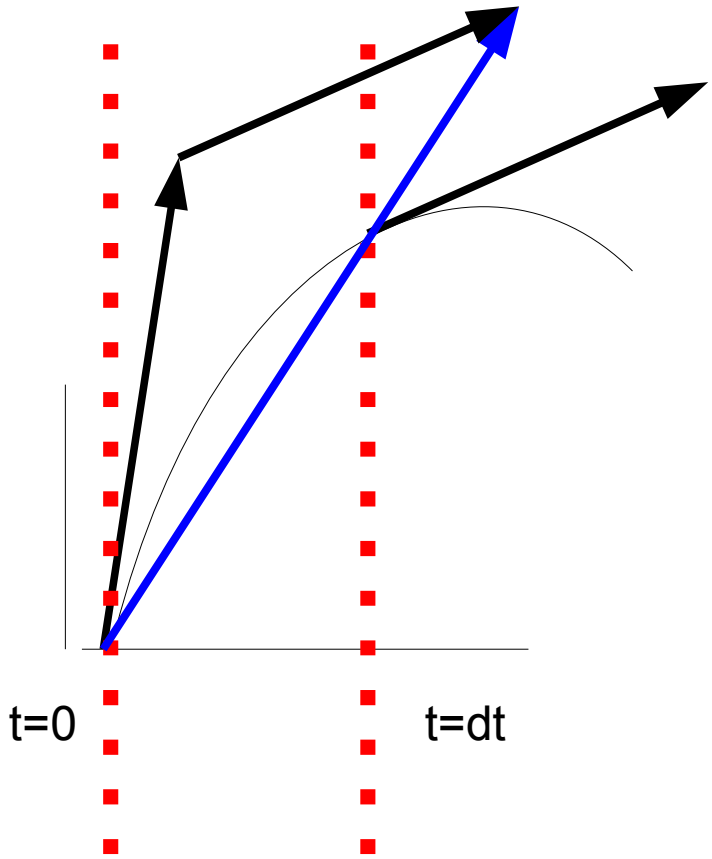

```
old_vx=self.vx
old_vy=self.vy
self.vx=self.vx+self.ax*self.dt
self.vy=self.vy+self.ay*self.dt
self.x=self.x+(self.vx+old_vx)/2*self.dt
self.y=self.y+(self.vy+old_vy)/2*self.dt
```

$$x = x_{old} + \left(Vx_{old} + Vx\right)/2 . dt$$

$$Vx = Vx_{old} + ax . dt$$

Physique du solide

- Gravité constante
  - Approximation d'ordre 2
  - Moyenne des vitesses




```
old_vx=self.vx
old_vy=self.vy
self.vx=self.vx+self.ax*self.dt
self.vy=self.vy+self.ay*self.dt
self.x=self.x+(self.vx+old_vx)/2*self.dt
self.y=self.y+(self.vy+old_vy)/2*self.dt
```

$$x = x_{old} + \left(Vx_{old} + Vx\right)/2 . dt$$

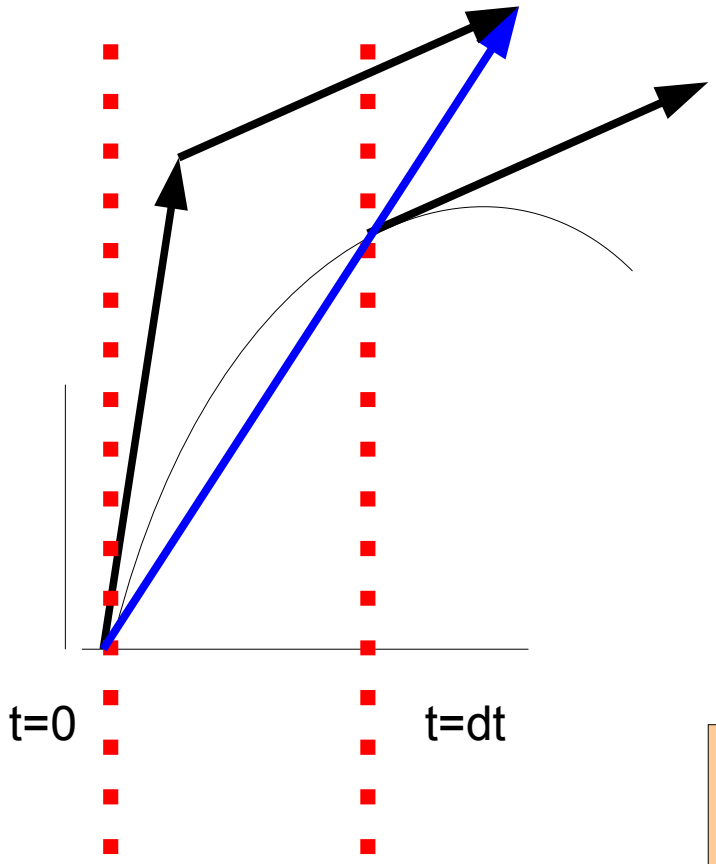
$$Vx = Vx_{old} + ax . dt$$

$$x = x_{old} + \left(Vx_{old} + Vx_{old} + ax . dt\right)/2 . dt$$

$$x = x_{old} + Vx_{old} . dt + ax/2 . dt^2$$

Physique du solide

- Gravité constante
  - Approximation d'ordre 2
  - Moyenne des vitesses




```
old_vx=self.vx
old_vy=self.vy
self.vx=self.vx+self.ax*self.dt
self.vy=self.vy+self.ay*self.dt
self.x=self.x+(self.vx+old_vx)/2*self.dt
self.y=self.y+(self.vy+old_vy)/2*self.dt
```

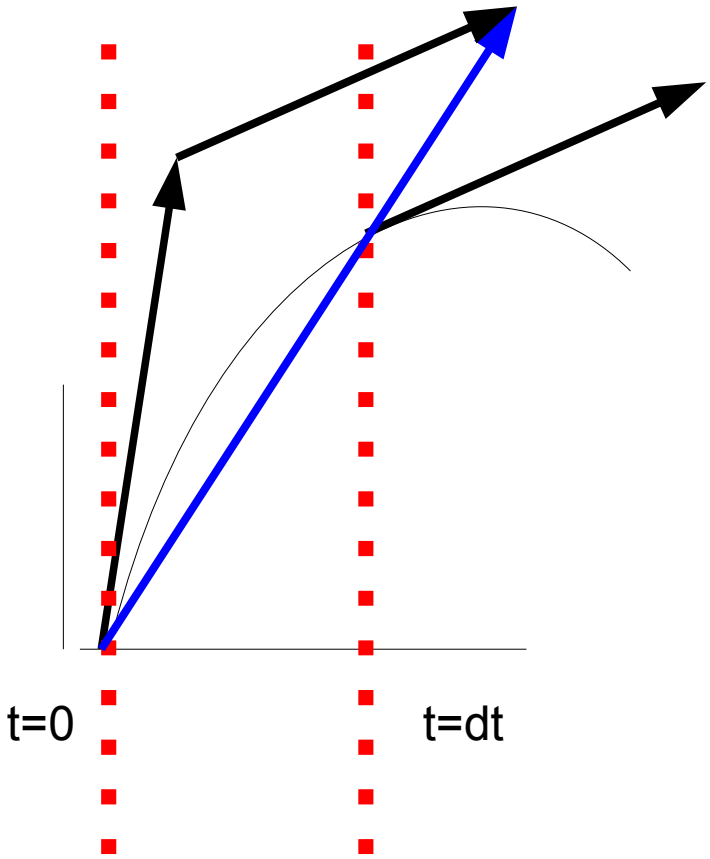
Intégration avec a supposé constant

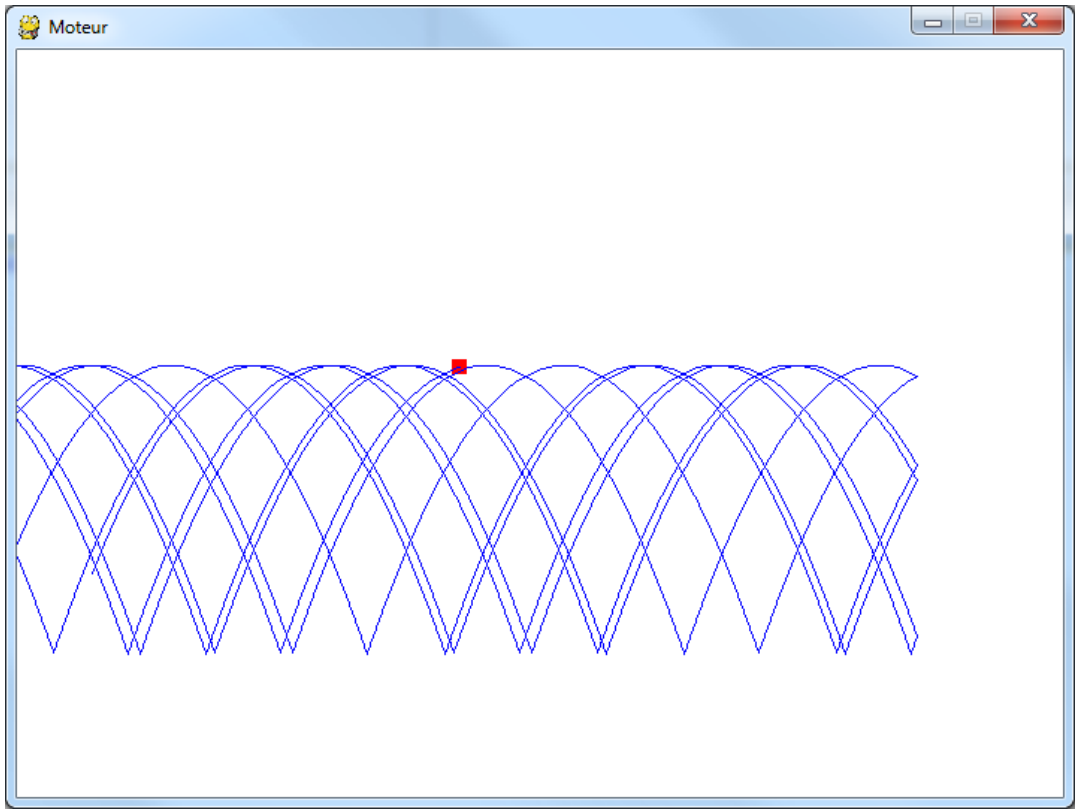
$$x = x_{old} + \left(Vx_{old} + Vx_{old} + ax \,.\, dt \right) / 2 \,.\, dt$$

$$x = x_{old} + Vx_{old} \,.\, dt + ax/2 \,.\, dt^2$$

Physique du solide

- Gravité constante
  - Approximation d'ordre 2
  - Moyenne des vitesses





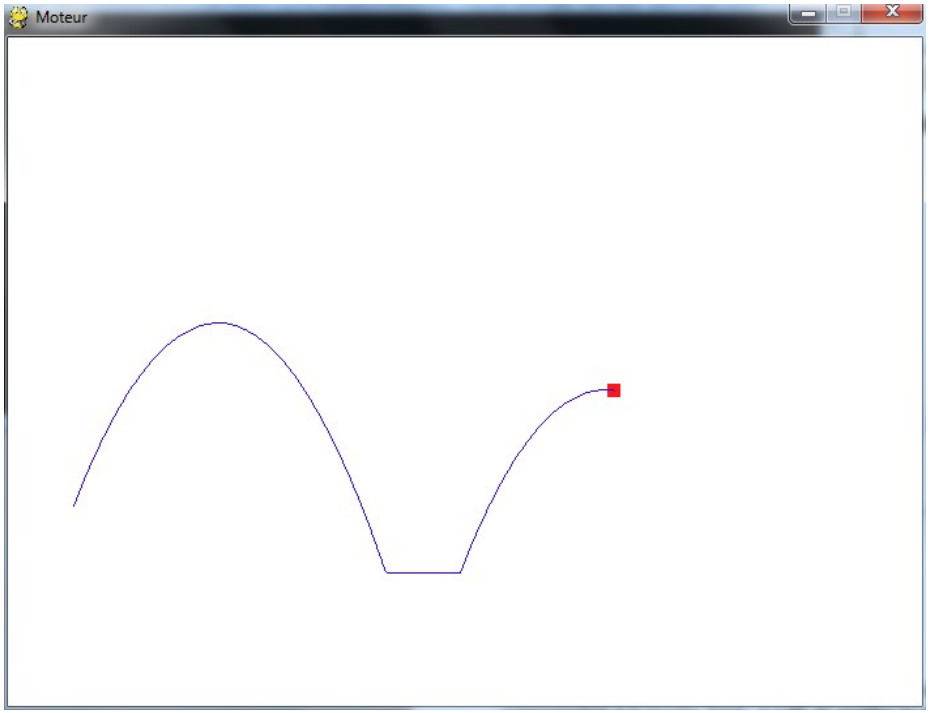
Plus d'erreur d'approximation (parce que g constant)

Physique du solide

```
class Jeu():

    def __init__(self):
        self.x=0
        self.y=0
        self.vx=50
        self.vy=10
        self.ax=0
        self.ay=-9
        self.dt=0.01

    def evoluer(self):
        self.x=self.x+self.vx*self.dt
        self.y=self.y+self.vy*self.dt
        self.vx=self.vx+self.ax*self.dt
        self.vy=self.vy+self.ay*self.dt

    def afficher(self):
        print("(x,y): ",self.x, "," , self.y)
        print("(vx,vy):",self.vx,",", self.vy)


jeu=Jeu()
for i in range(1000):
    jeu.evoluer()
    jeu.afficher()
```

Eventuellement
Ordre 2

Ajouter du controle

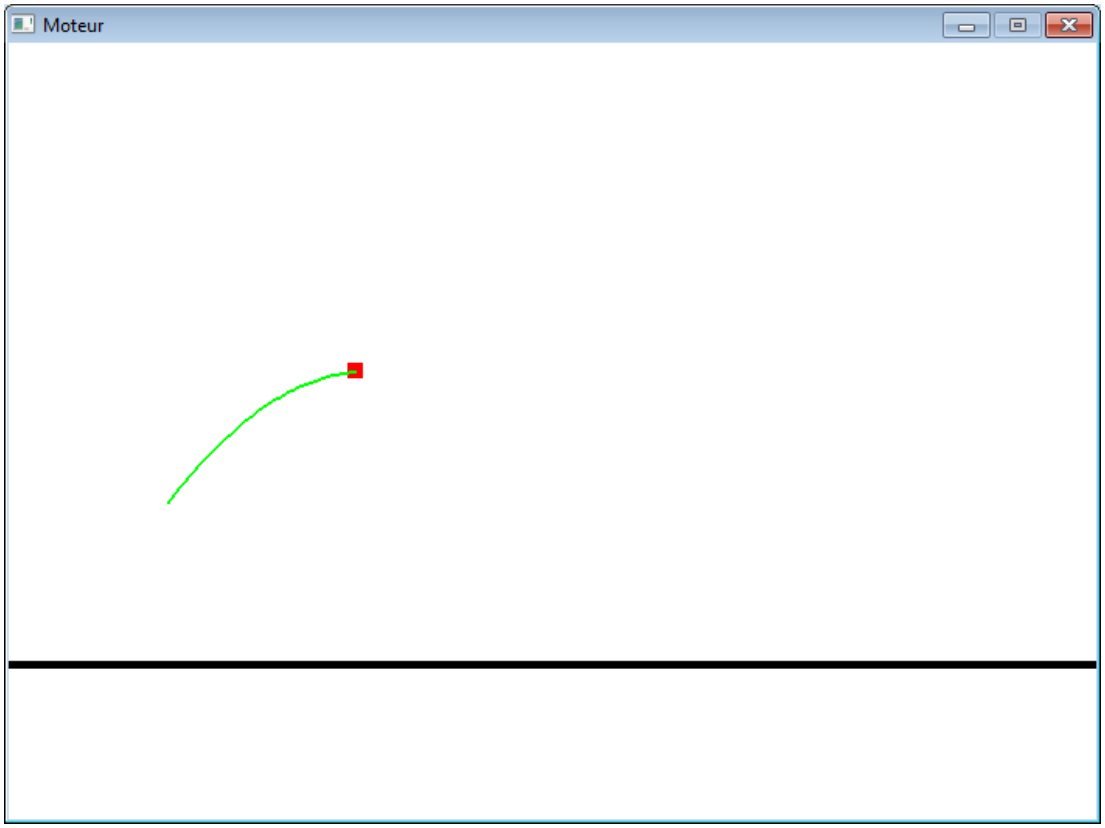



```
Evoluer() :

#plus de rebond
if (self.y<0):
    self.y=0
    self.vy=0;
```

```
Gestion_evenement() :

#gestion de l'appui d'une touche
if event.type == pygame.KEYDOWN:
    # si c'est la touche "haut"
    if event.key == pygame.K_UP:
        print("up")
        # si on est pas déjà en haut
        if jeu.y==0 :
            jeu.vy=50
```

Exemple python

- Angry birds like
  - Controle souris
  - Determiner vitesse initiale



Plan

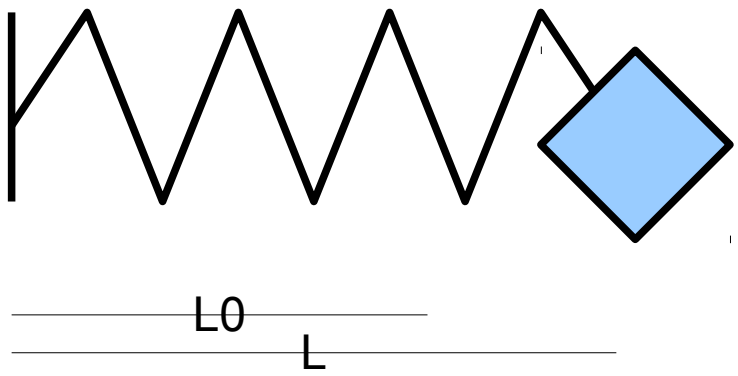
- Mecanique du point
- Moteur de jeu
- Mecanique du point (bis)
- Comportements collectifs
- Steering behaviour
- Probleme de controle

Suite

- Base solide
- Ajouter
  - Ressorts
  - Collisions

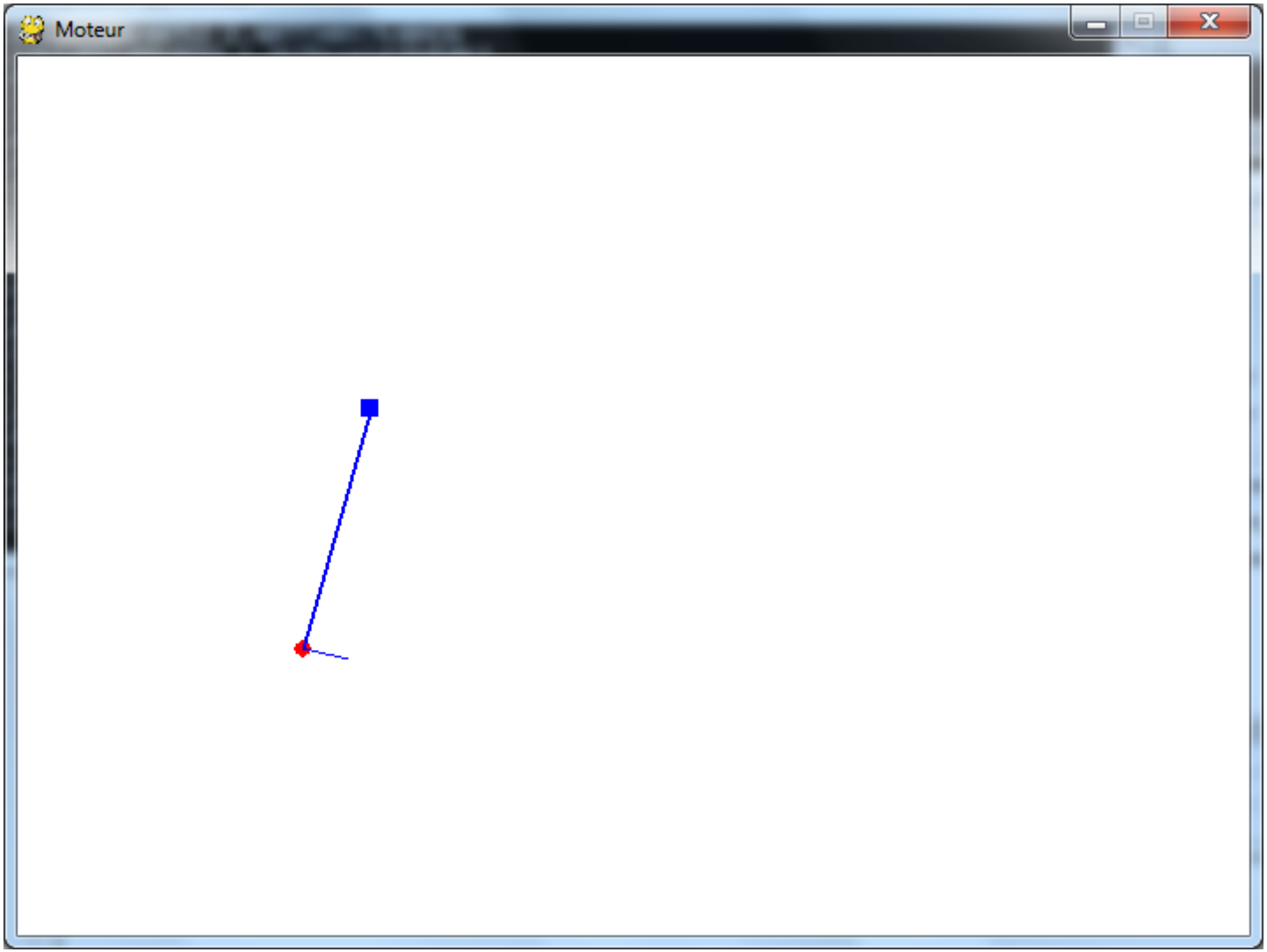
Physique du solide

- Modeles
  - Gravité constante
  - Systeme masse-ressort (raideur k)



  - Frottements
    - (- alpha.v)

Ex python

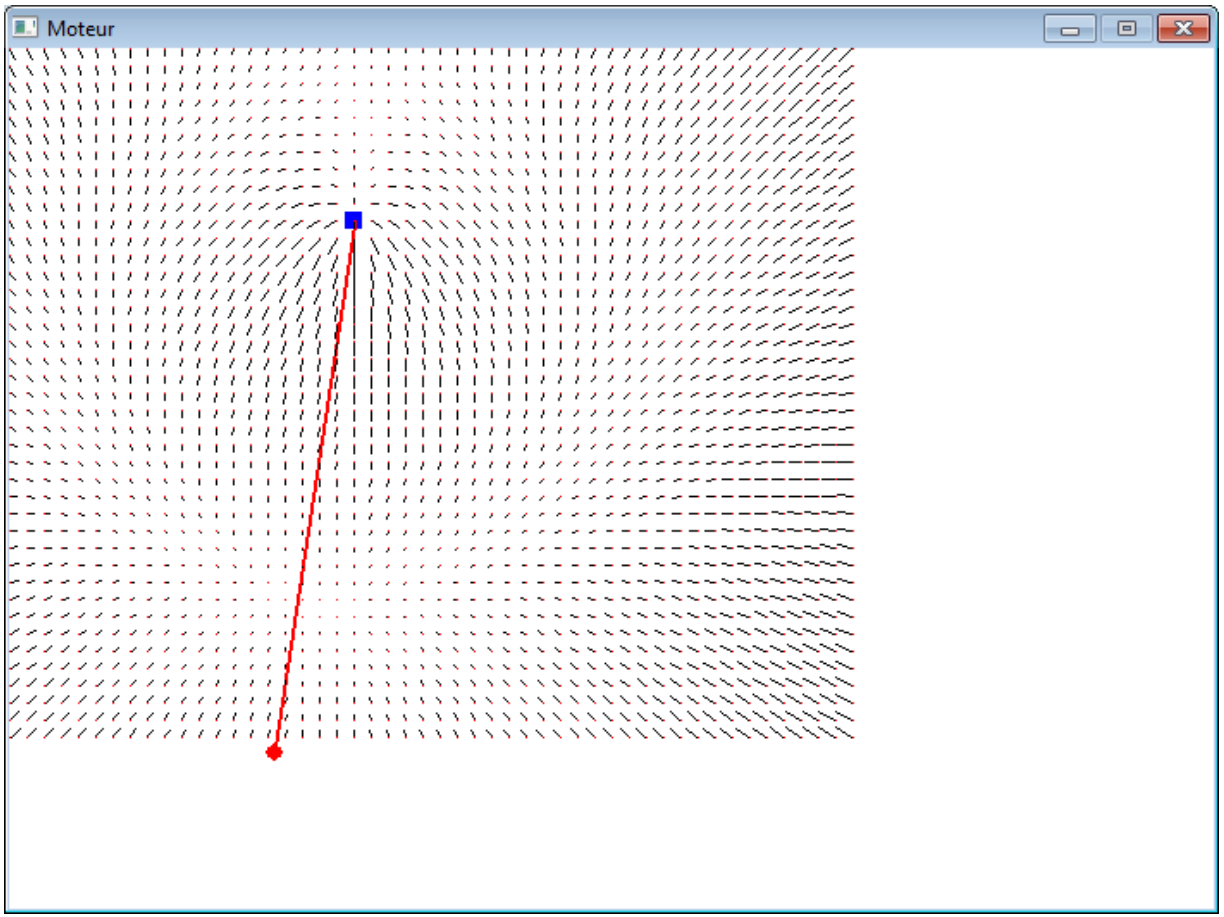


Physique du solide

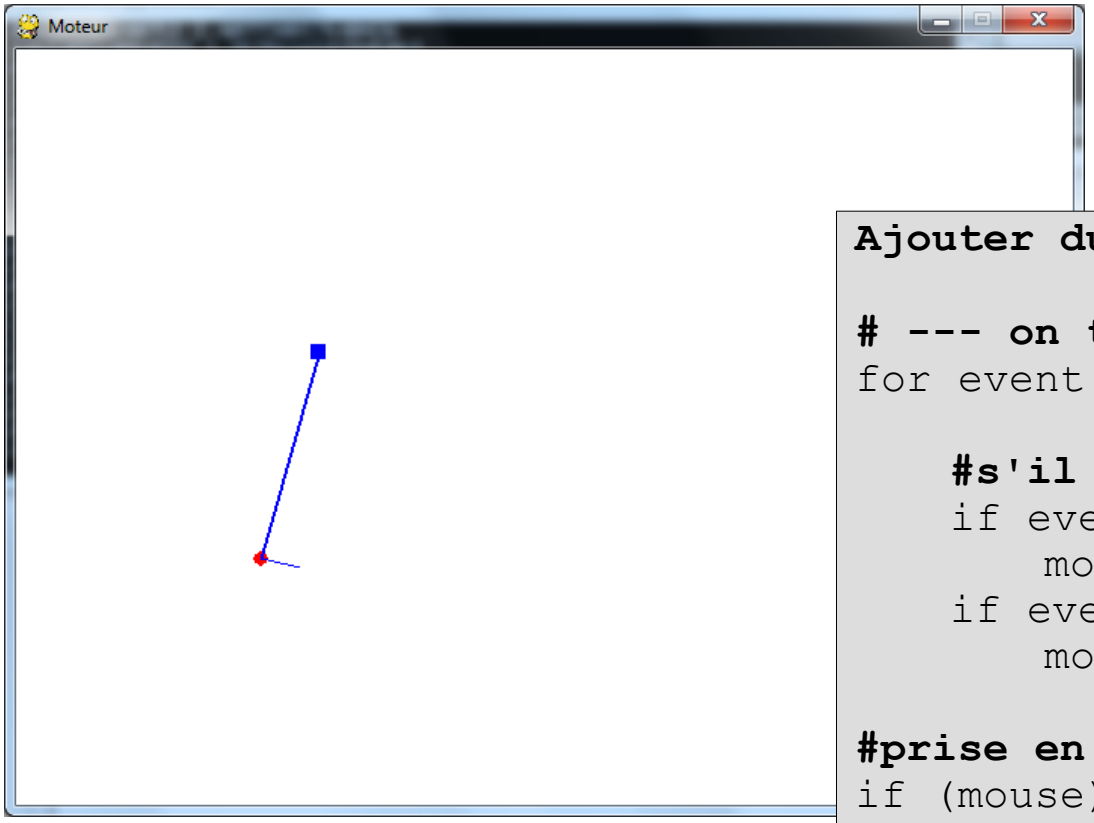
- Exemples
  - Dessin de graphes
  - Treillis de ressort
- Travail étudiant
  - Dessiner espace des phases
  - Afficher la carte des forces
- Possible de ne pas supprimer affichage entre frame

Affichage possible

- Champ de force
  - Équilibre stables et instables



Ex python

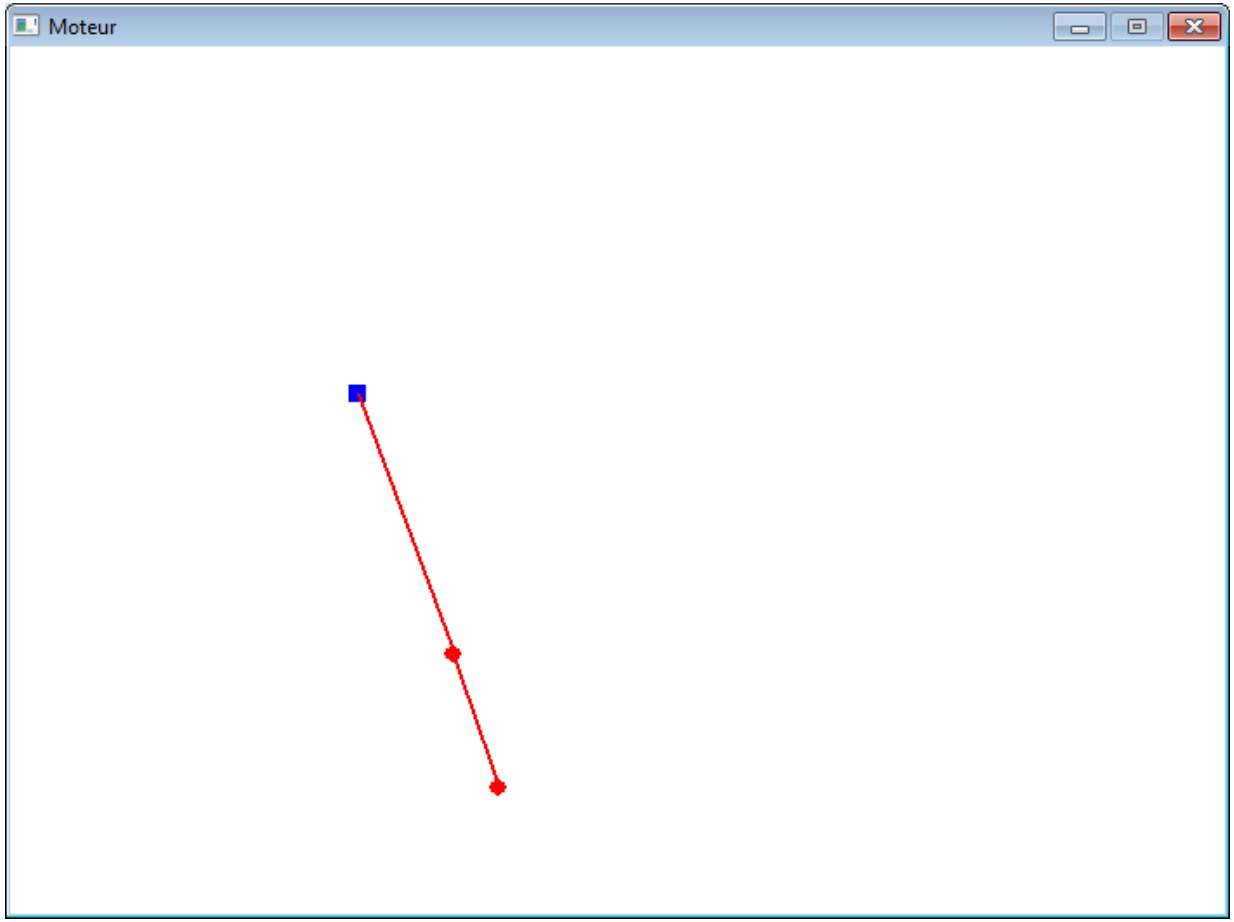


Ajouter du contrôle

```
# --- on traite les evenements
for event in pygame.event.get():

    #s'il a appuye sur la souris
    if event.type == pygame.MOUSEBUTTONDOWN:
        mouse=True
    if event.type == pygame.MOUSEBUTTONUP:
        mouse=False

#prise en compte souris
if (mouse):
    balle.x=pygame.mouse.get_pos()[0]
    balle.y=400-pygame.mouse.get_pos()[1]
    balle.vx=0
    balle.vy=0
```

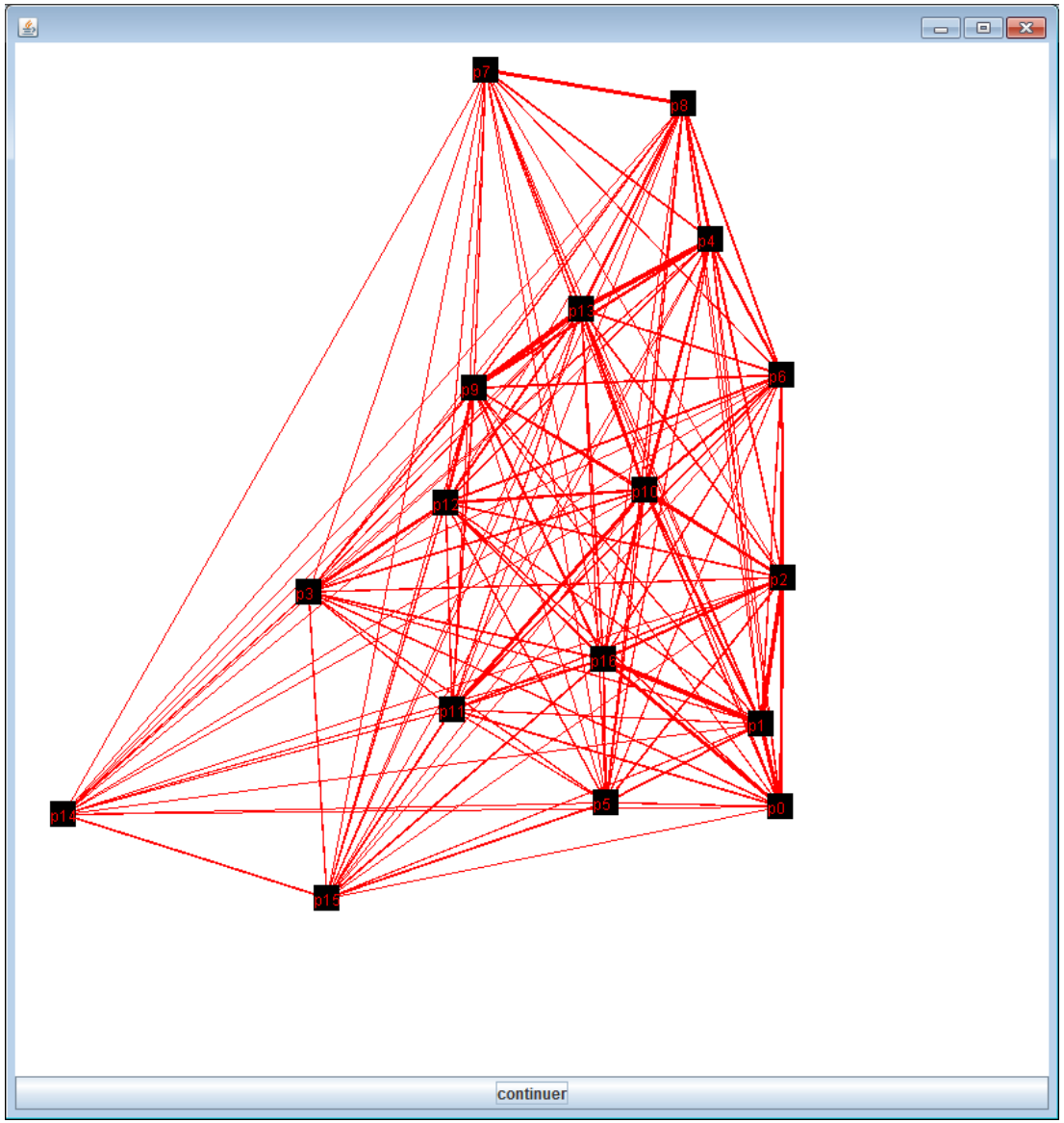
Exemple

- Multi-ressort
  - Avec/sans controle



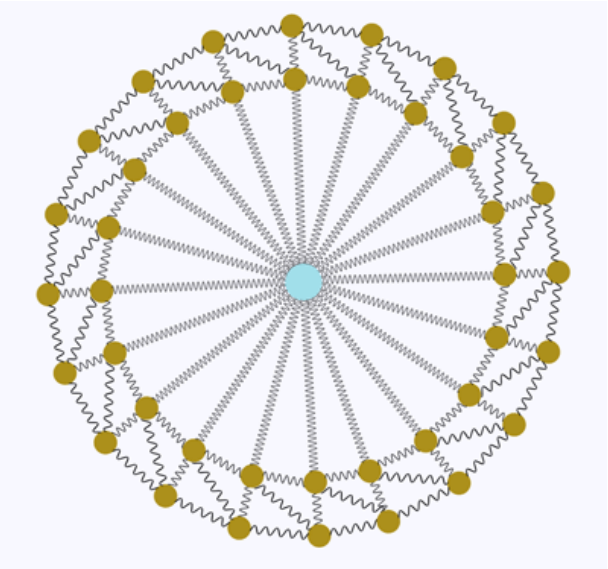
Treillis de ressort et graphe

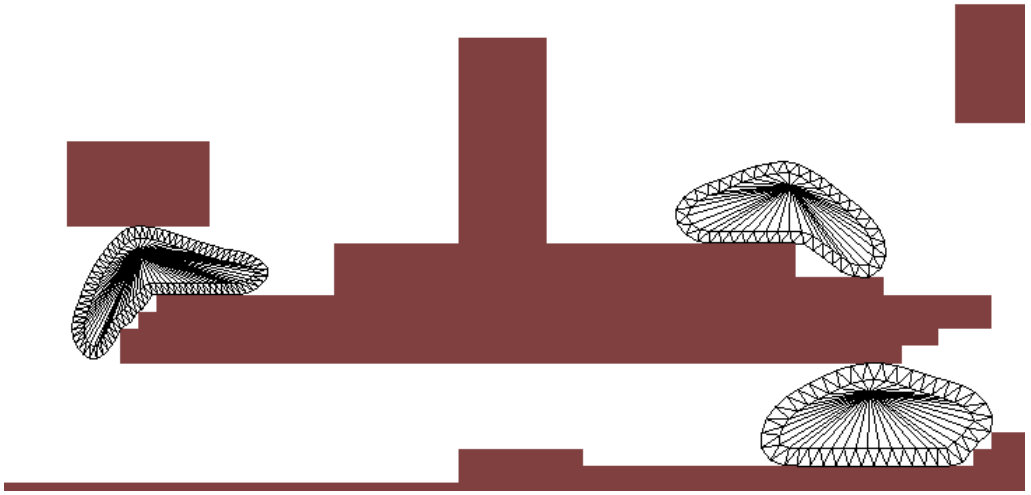
- Démo



Jeu video - Gish

- Système à base de ressorts





http://cowboyprogramming.com/2007/01/05/blob-physics/

Suite

- Base solide

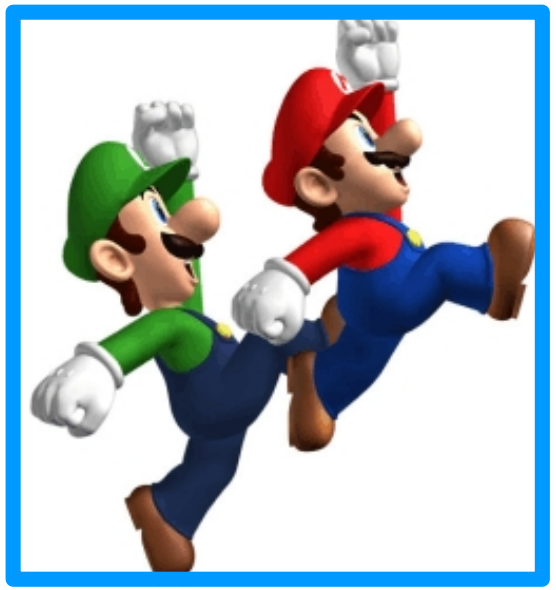
- Ajouter
  - Ressorts
  - Collisions

Gestionnaire de collision

- Principes
  - Utilisation de bounding box

Gestionnaire de collision

- Principes
  - Utilisation de bounding box

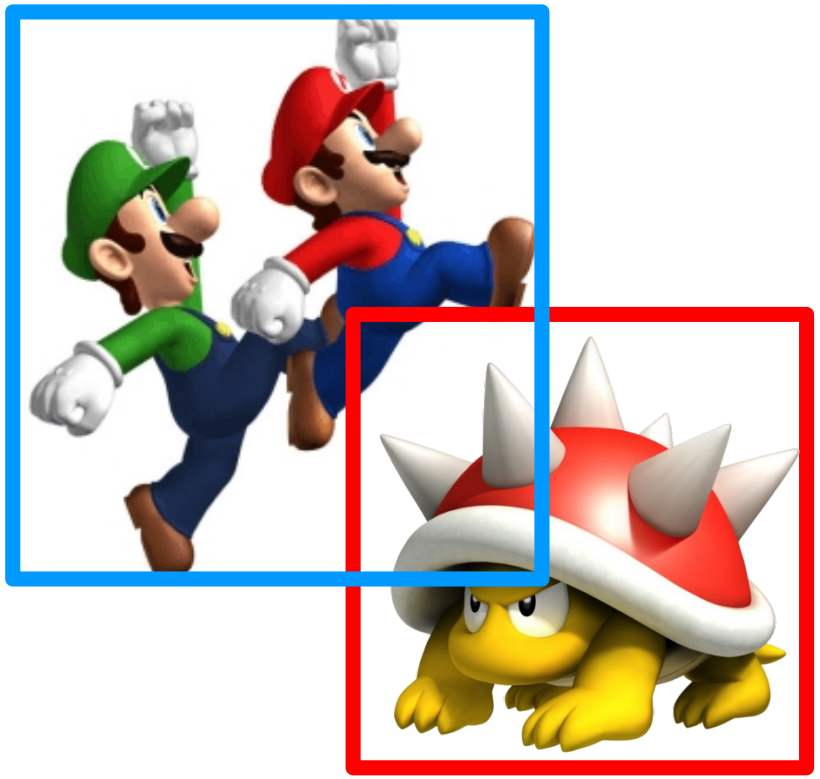


Gestionnaire de collision

- Principes
  - Utilisation de bounding box

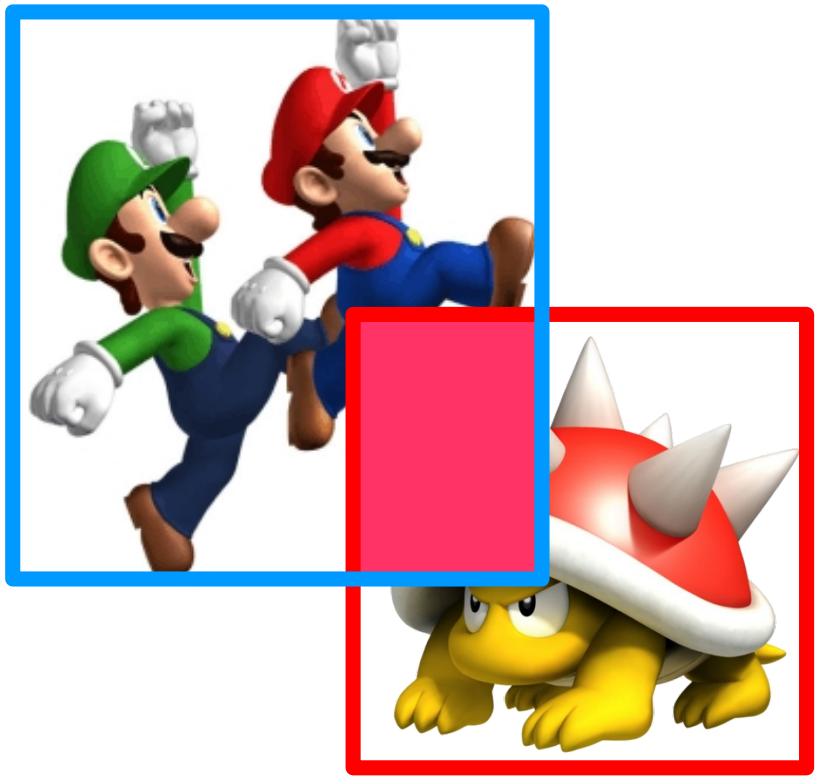
Gestionnaire de collision

- Principes
  - Utilisation de bounding box



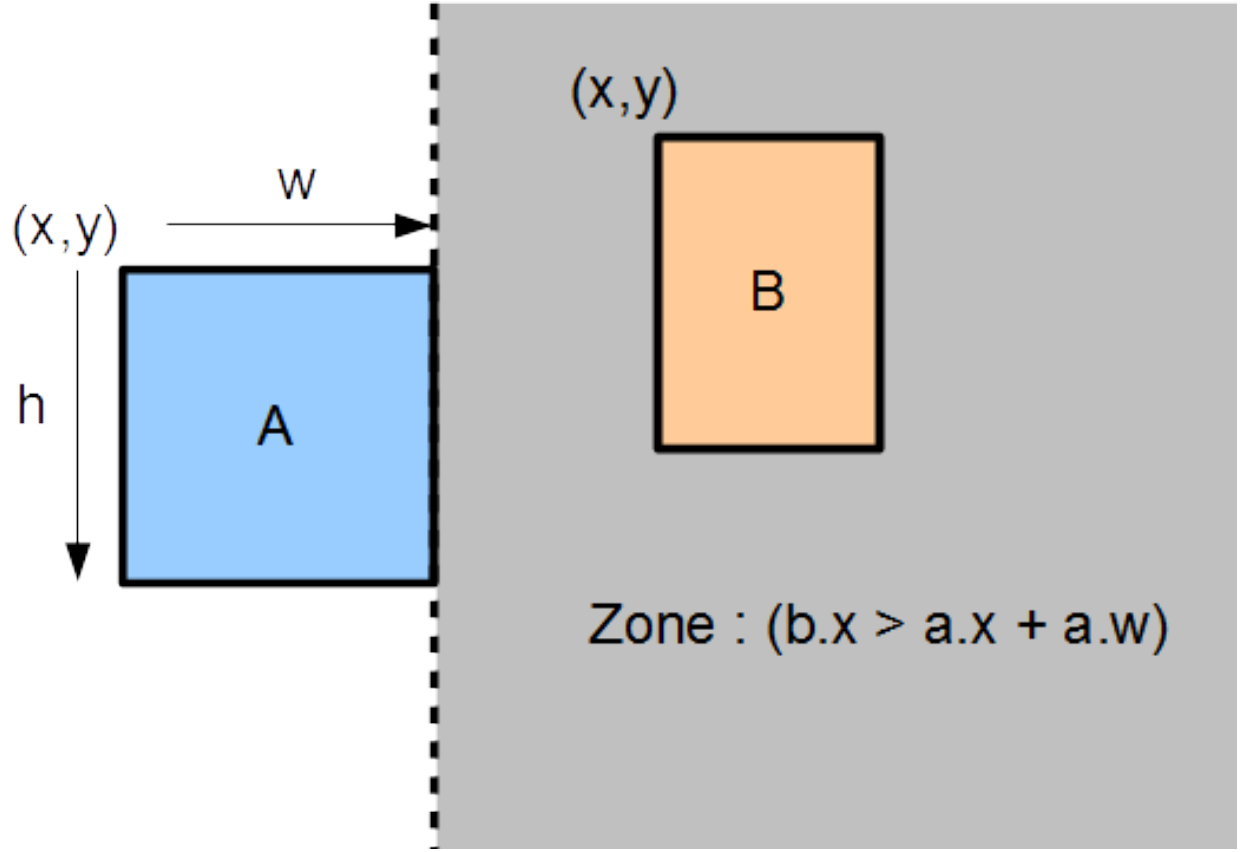
Gestionnaire de collision

- Principes
  - Utilisation de bounding box



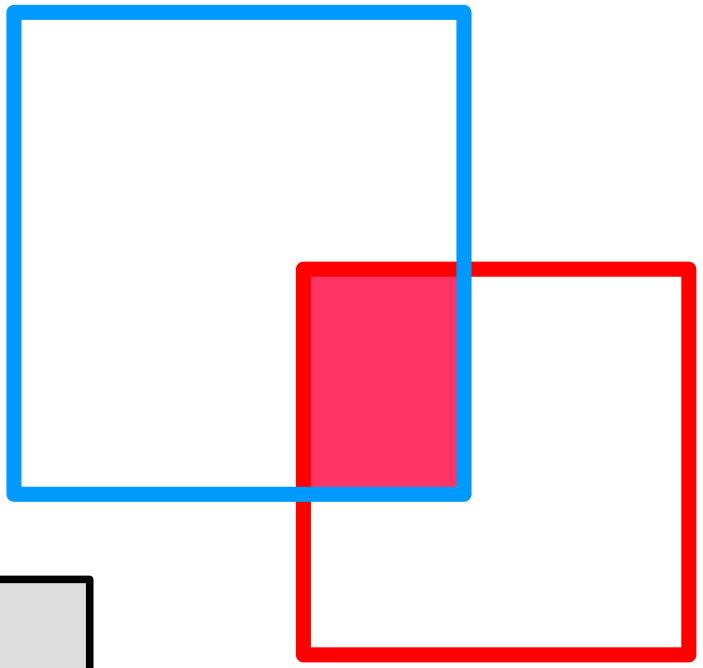
Physique du solide

- Gestion des collisions



Gestionnaire de collision

- Intersection de rectangles

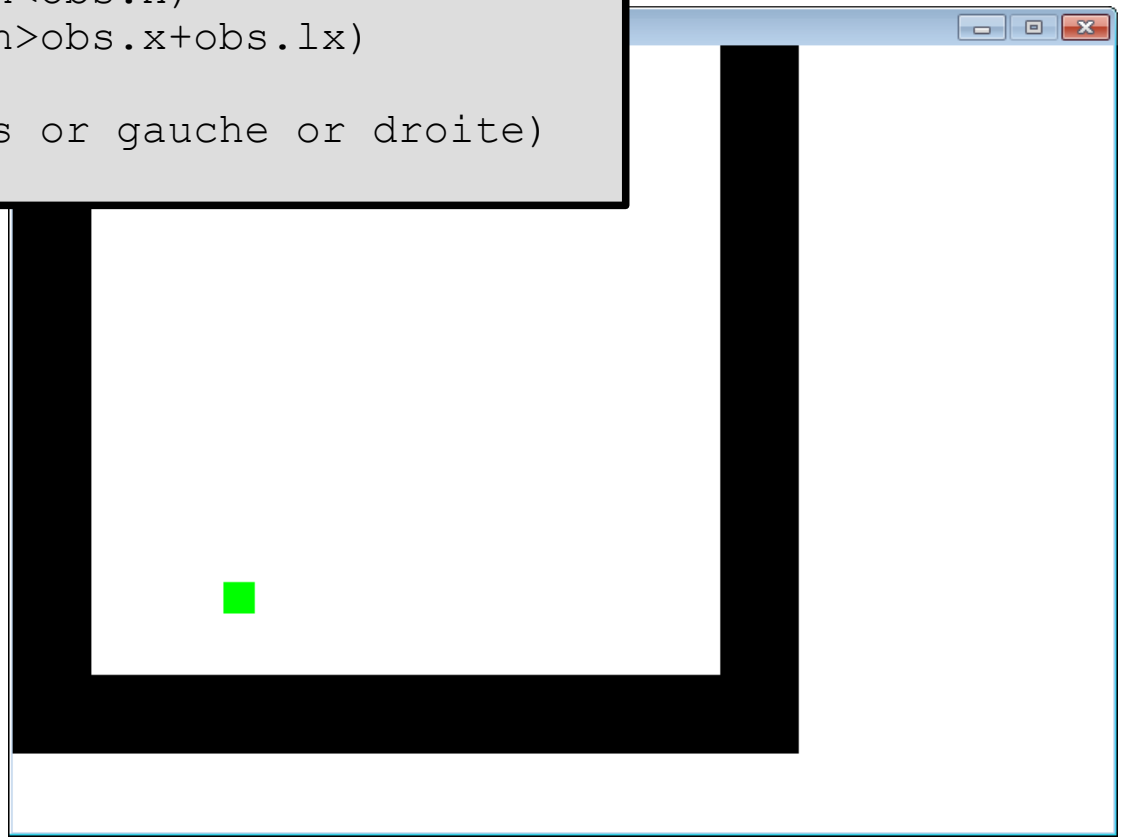


```
def collision(self,obs):
    #teste collision dans chaque direction
    haut=(self.y+self.rayon<obs.y)
    bas=(self.y-self.rayon>obs.y+obs.ly)
    gauche=(self.x+self.rayon<obs.x)
    droite=(self.x-self.rayon>obs.x+obs.lx)
    #teste collision
    collision=not(haut or bas or gauche or droite)
    return collision
```

Exemple python

- Detecter les collisions

```
def collision(self,obs):
    #teste collision dans chaque direction
    haut=(self.y+self.rayon<obs.y)
    bas=(self.y-self.rayon>obs.y+obs.ly)
    gauche=(self.x+self.rayon<obs.x)
    droite=(self.x-self.rayon>obs.x+obs.lx)
    #teste collision
    collision=not(haut or bas or gauche or droite)
    return collision
```

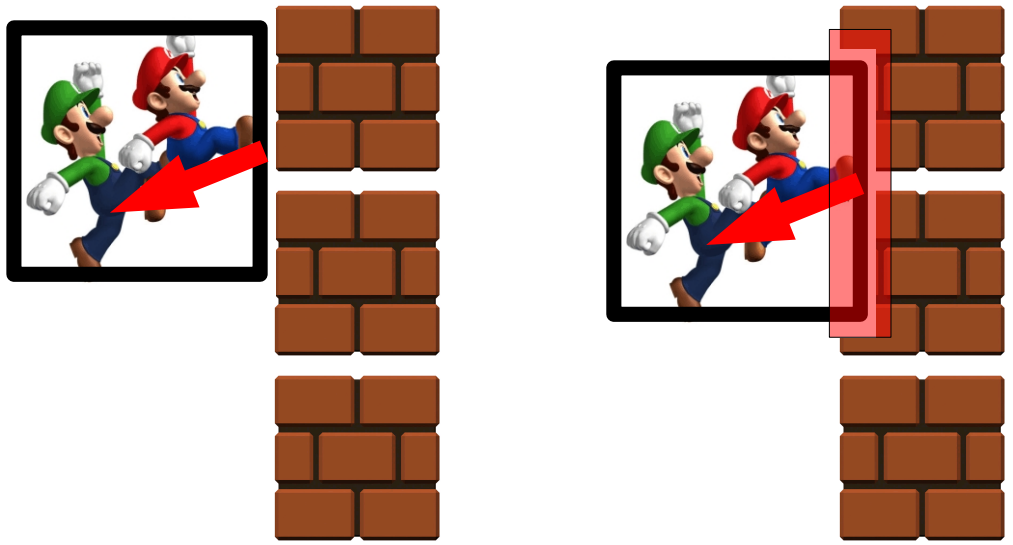


Gestionnaire de collision

- Prise en compte du résultat
  - Peu indétermination (t petit)

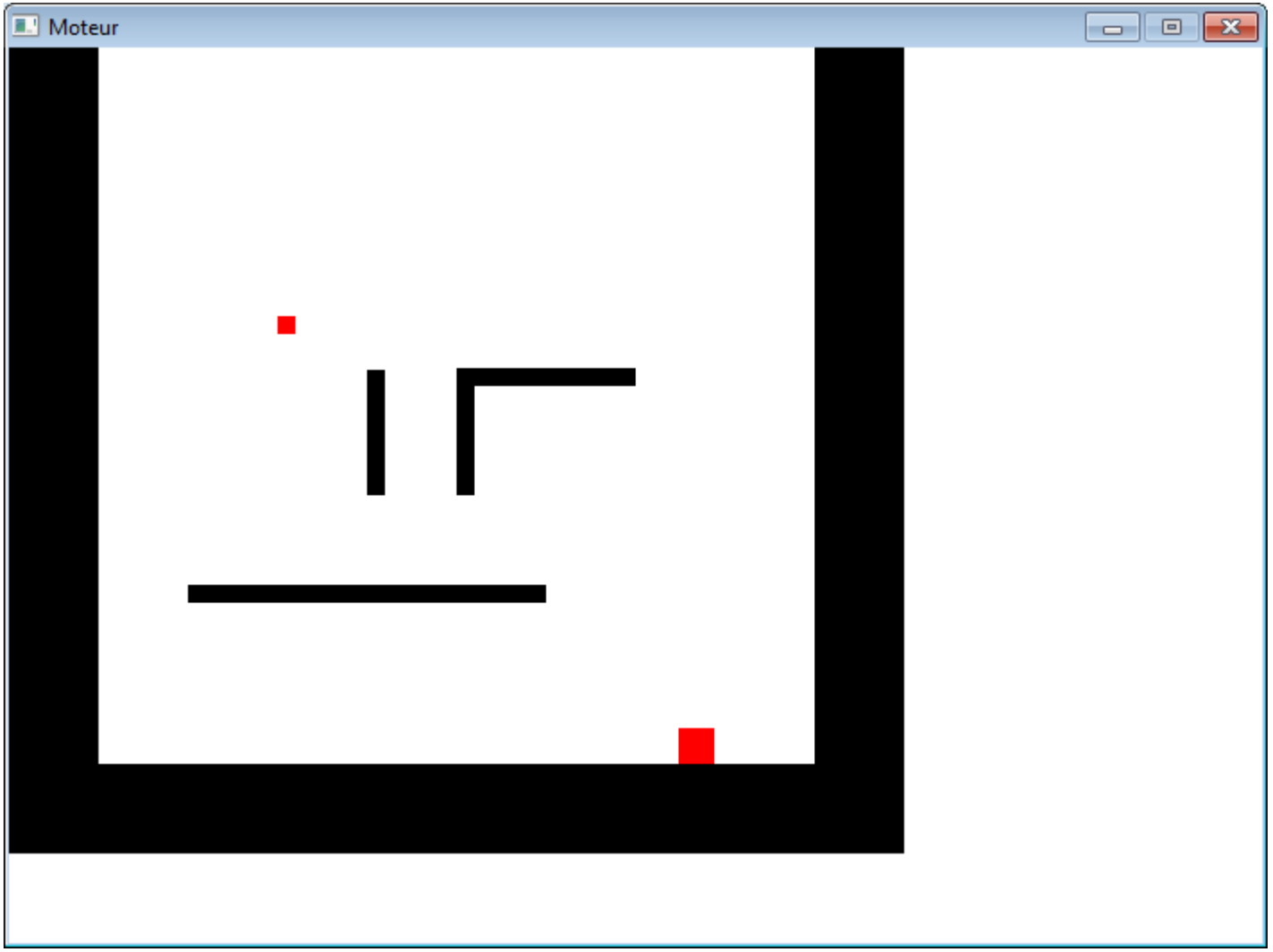
- Solution

1. Revenir passé
2. Approximation



Exemple python

- Simulateur avec collision



Jeu video

- Demonstration projet dut
  - Portal
  - Super street Melee
  - Save the train
  - Angry birds

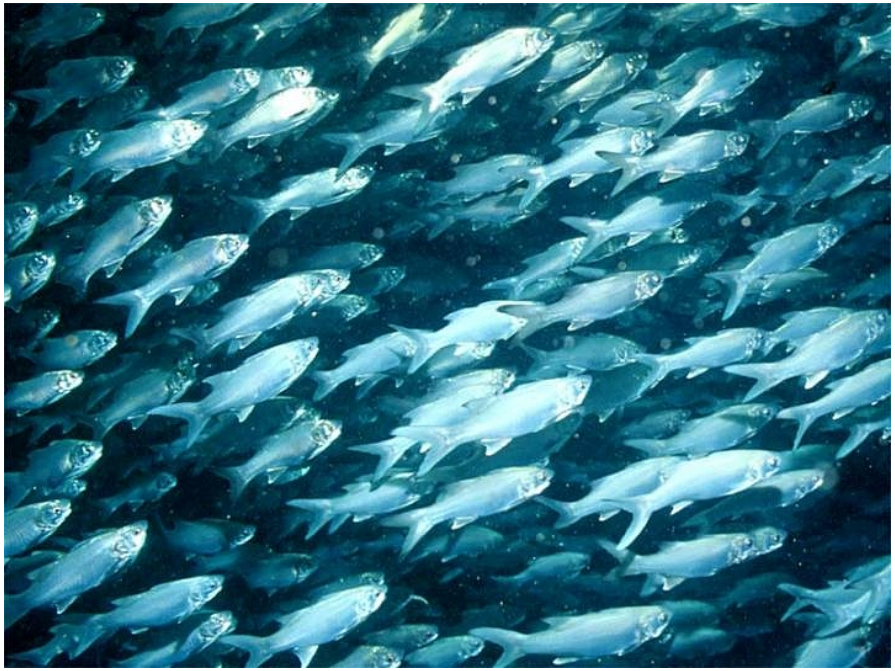
- Autre
  - Bang
  - Gish
  - Mario-like
  - Billard (collision elastique)

Plan

- Mecanique du point
- Moteur de jeu
- Mecanique du point (bis)
- Comportements collectifs
- Steering behaviour
- Probleme de controle

Comportement

- Flocking
  - Pas de leader



Comportement

- Flocking

- Foule de piétons
  - Couche de comportement supplémentaire
  - D. Helbling "social force model" (physical rev 95)
  - Helblin "Simulating dynamical features of escape panic" (nature 2000)

Système de forces sociales

Représenter relations avec les autres par des forces (proxemie)

http://www.larecherche.fr/savoirs/cognition/comment-pietons-marchent-foule-01-03-2011-87528

Plan

- Mecanique du point
- Moteur de jeu
- Mecanique du point (bis)
- Comportements collectifs
- Steering behaviour
- Probleme de controle

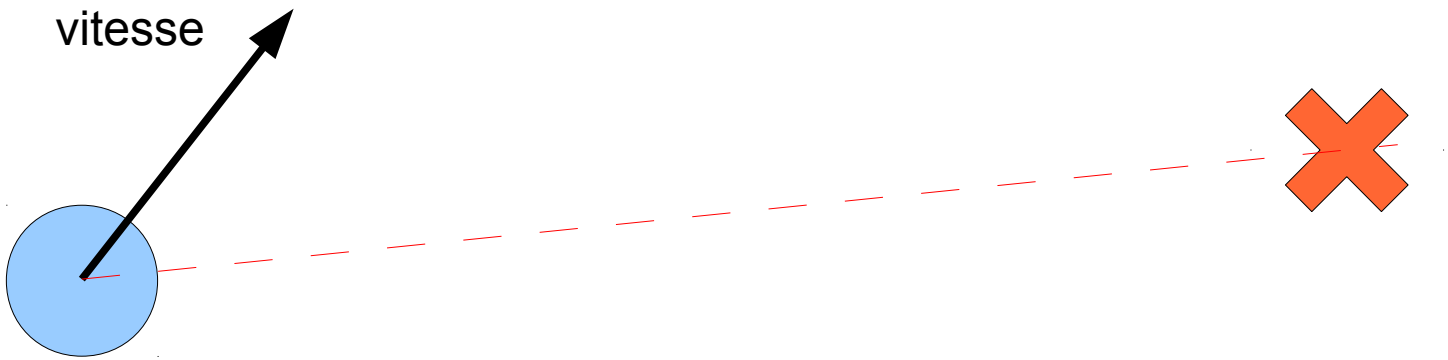
Steering behaviour

- Modeliser comportement
  - par acceleration/vitesse

- Article de référence
  - Steering Behaviors For Autonomous Characters (1999)
  - Craig Reynolds (http://www.red3d.com/cwr/)

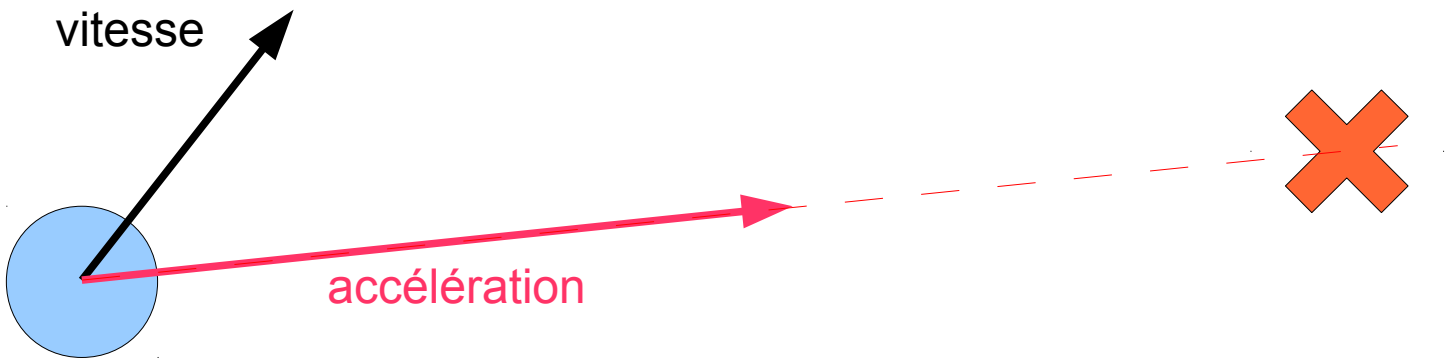
Steering beahviour

- Plein de modeles
  - Se diriger vers un point



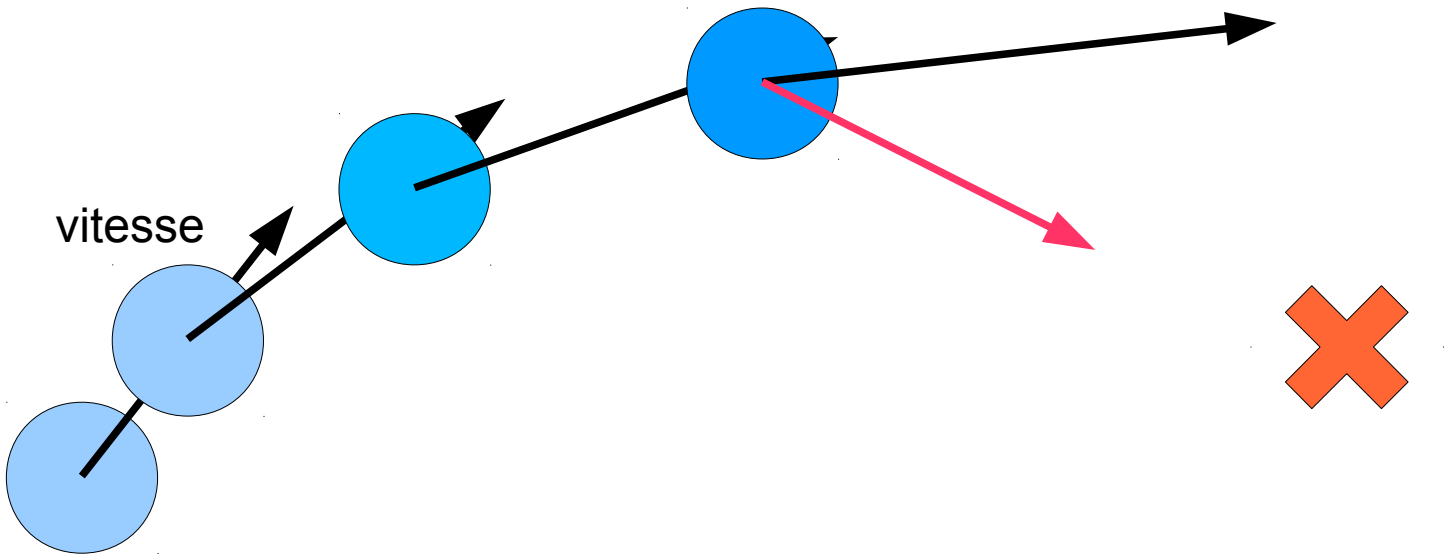
Steering beahviour

- Plein de modeles
  - Se diriger vers un point



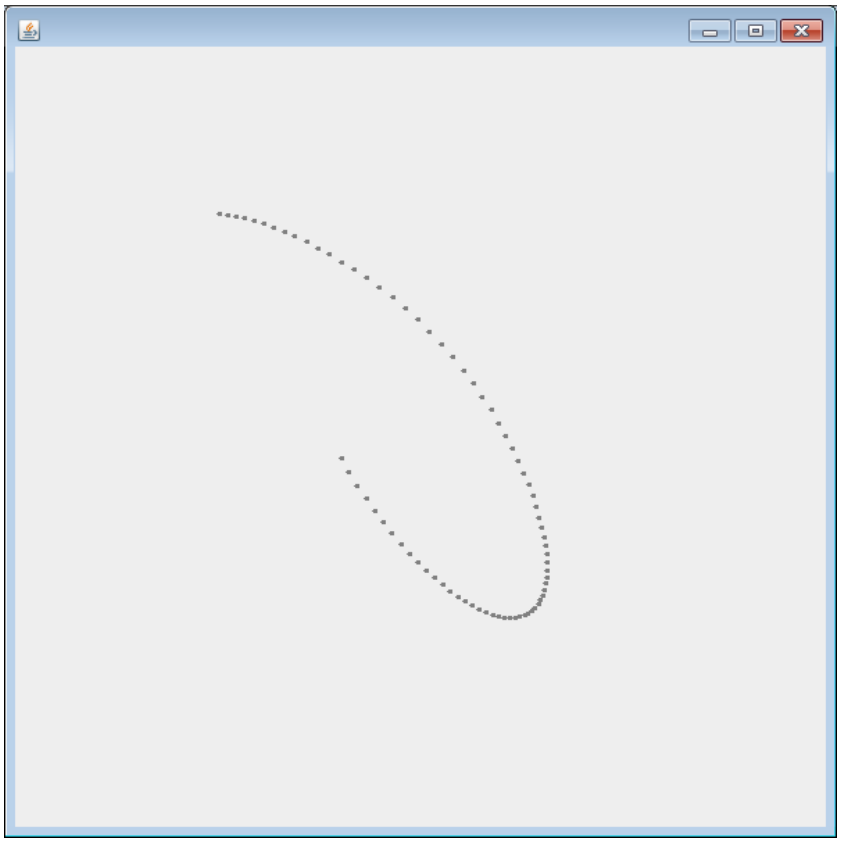
Steering beahviour

- Plein de modeles
  - Se diriger vers un point



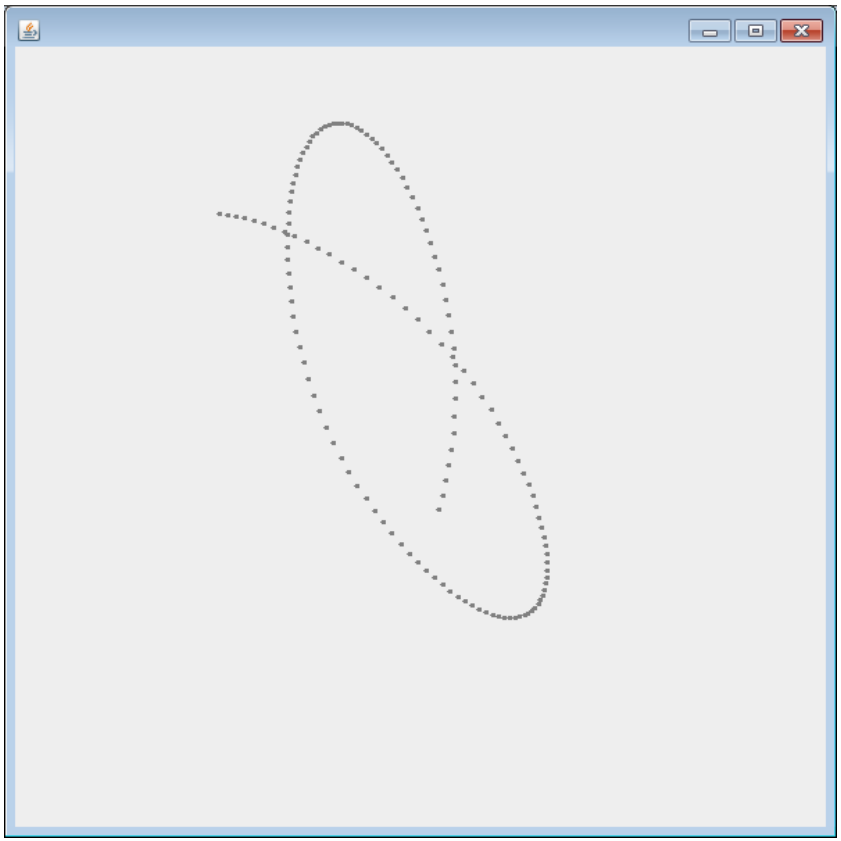
Steering beahviour

- Plein de modeles
  - Se diriger vers un point



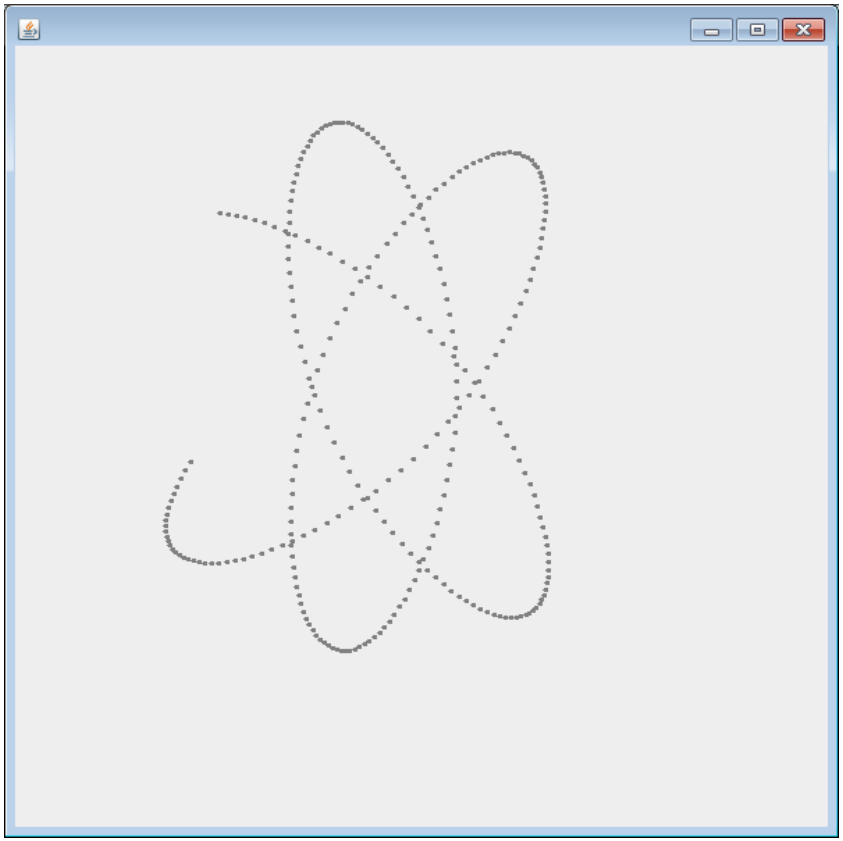
Steering beahviour

- Plein de modeles
  - Se diriger vers un point



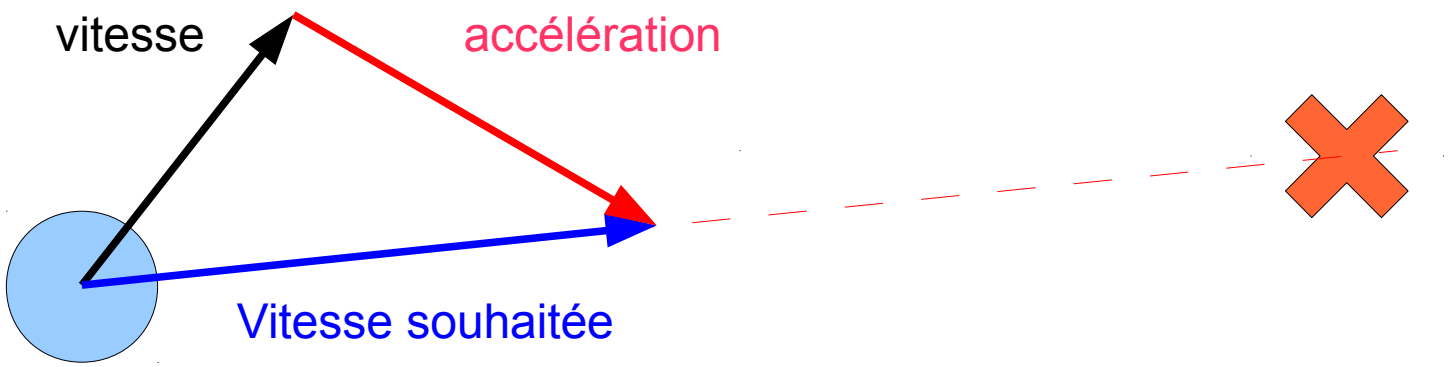
Steering beahviour

- Plein de modeles
  - Se diriger vers un point
  - Mauvaise solution : oscillateur

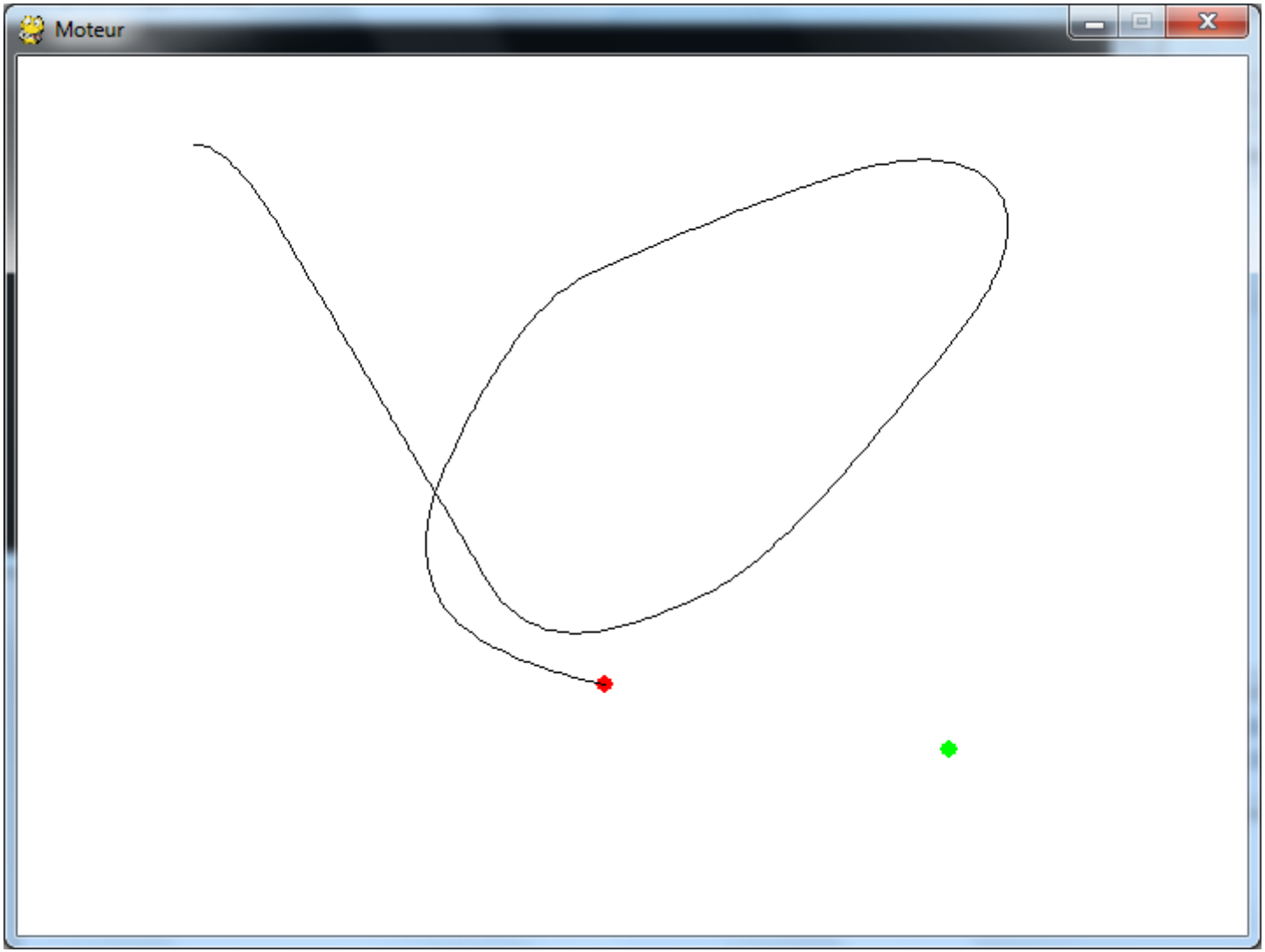


Steering beahviour

- Plein de modeles
  - Se diriger vers un point

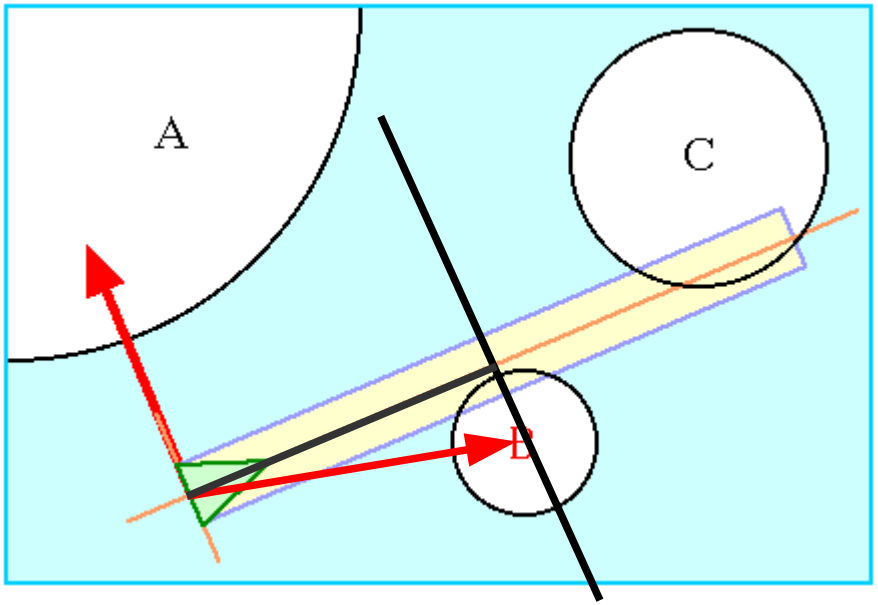


demo



Steering beahviour

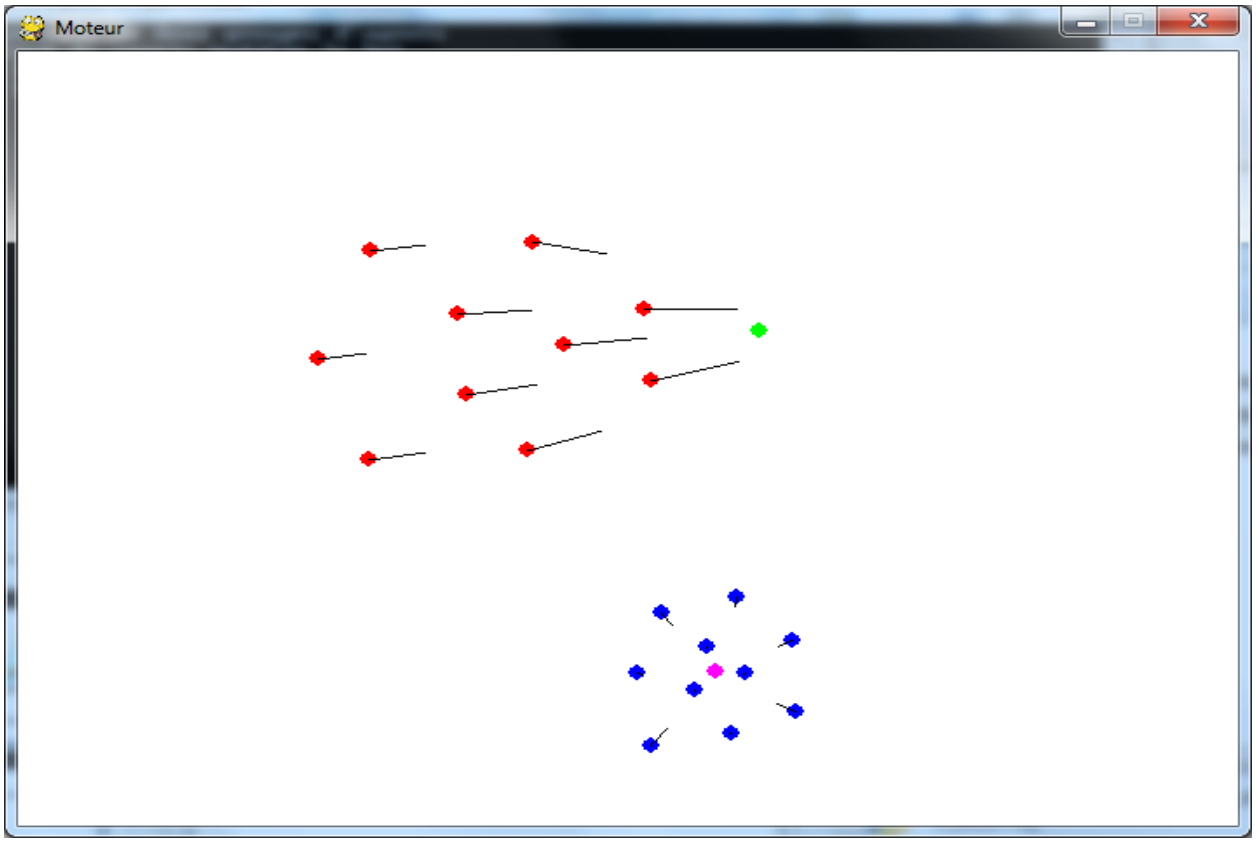
- Plein de modeles
  - Evitement



Craig Reynolds – Steering behaviours

Steering beahviour

- Plein de modeles
  - Force sociales
    - Répulsion entre agents proches



Steering beahviour

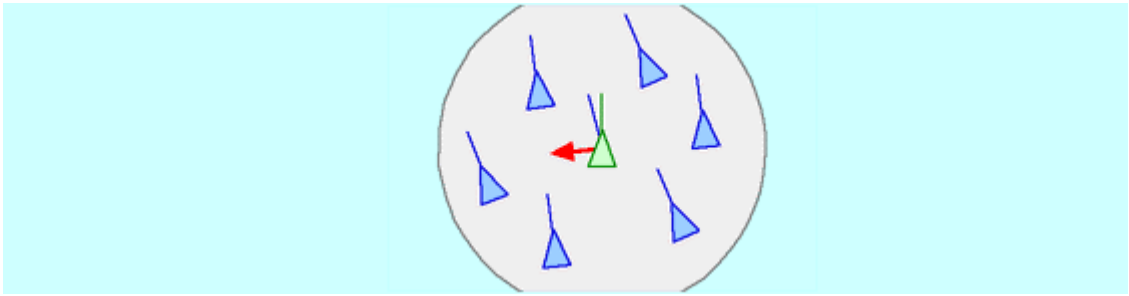
- Modèle de flocking
  - 3 comportements

http://gamedevelopment.tutsplus.com/tutorials/the-three-simple-rules-of-flocking-behaviors-alignment-cohesion-and-separation--gamedev-3444

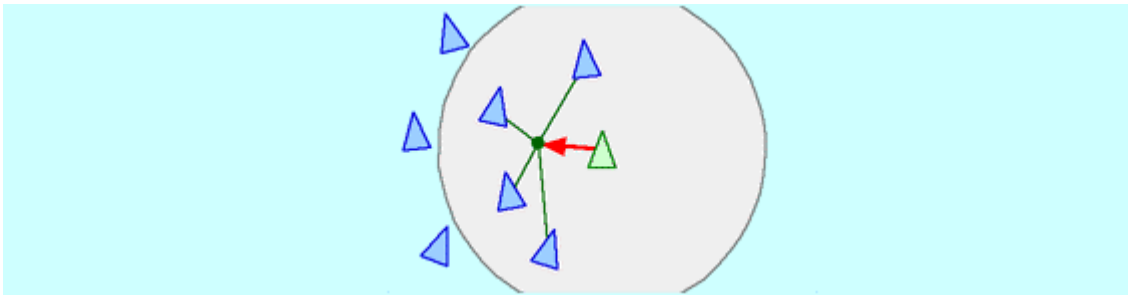
Steering beahviour

- Modèle de flocking
  - 3 comportements

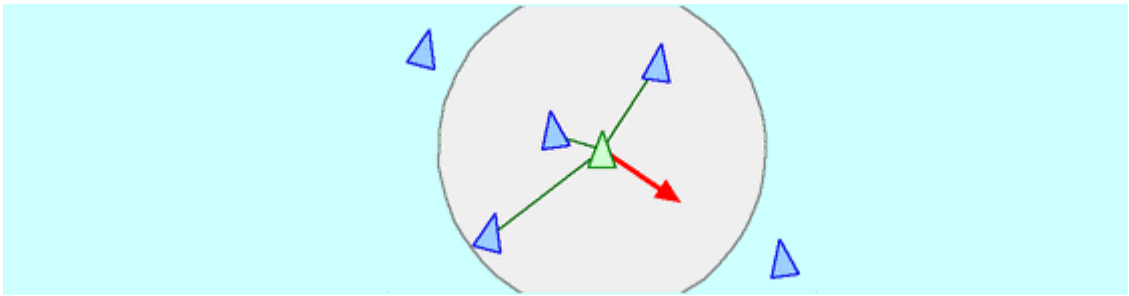
Alignement : oriente vitesse selon la moyenne des voisins



Cohésion : vitesse vers baricentre des voisins



Separation : les voisins proches repoussent l'agent



http://gamedevelopment.tutsplus.com/tutorials/the-three-simple-rules-of-flocking-behaviors-alignment-cohesion-and-separation--gamedev-3444

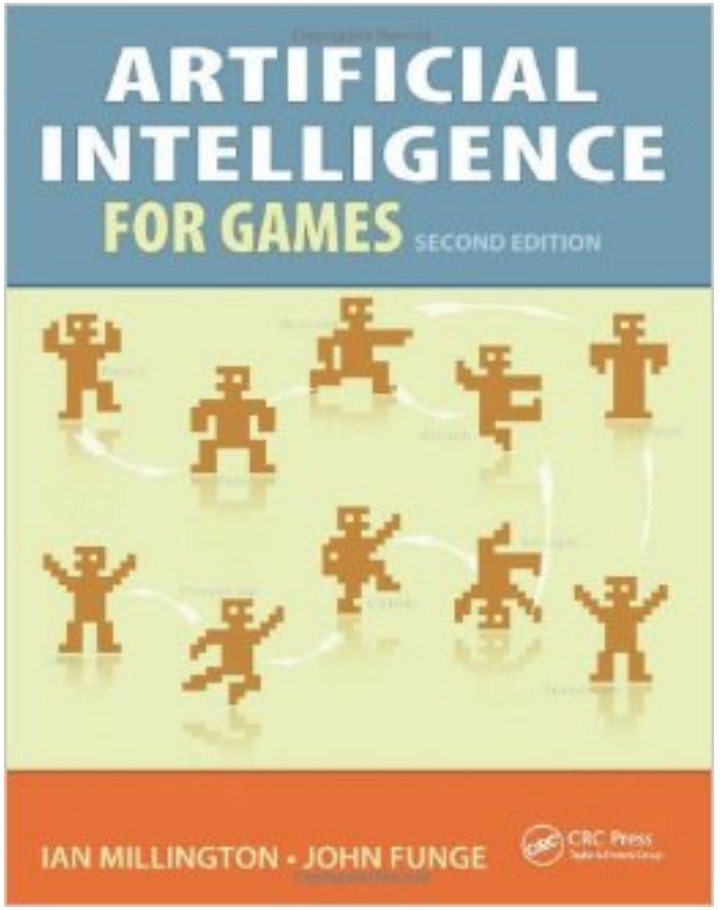
Steering behaviour

- Assassin's creed

Steering behaviour

- Ia dans les jeux vidéos
  - Millington, Funge (2006)

- http://ai4g.com/



Plan

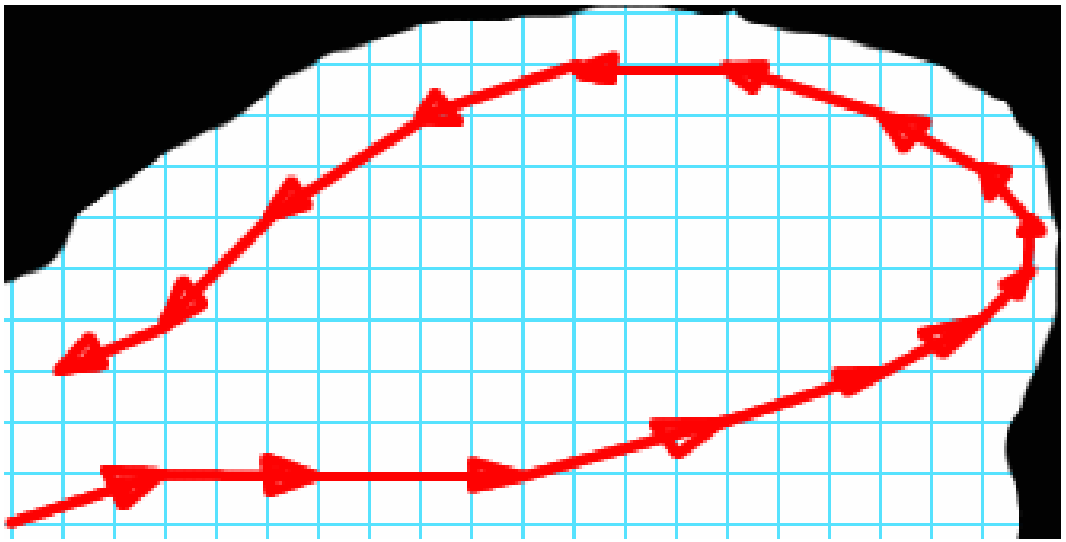
- Mecanique du point
- Moteur de jeu
- Mecanique du point (bis)
- Comportements collectifs
- Steering behaviour
- Probleme de controle

Probleme de controle

- Systeme dynamique
  - État interne du système
  - Actions possibles
- Loi d'évolution du systeme
  - $s_{t+1}=f(s_t,a_t)$
- Question
  - Quelle loi de controle a_t ?

Probleme de controle

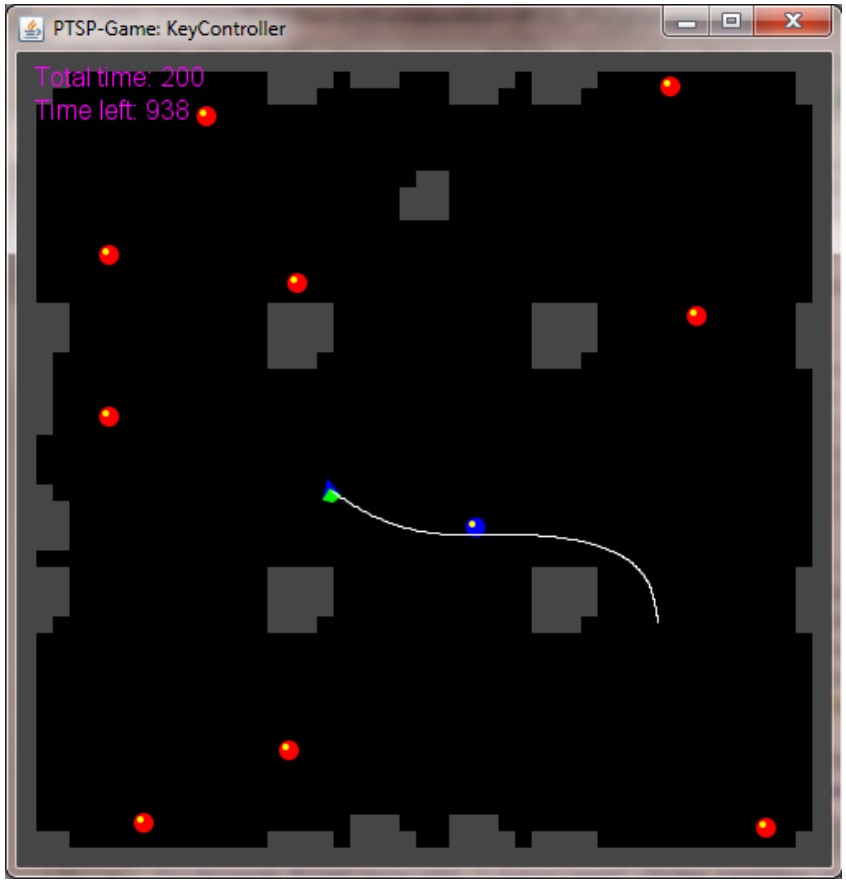
- Jeu de course papier



http://www.sjbaker.org/paper_and_pencil_games/graph_racers/

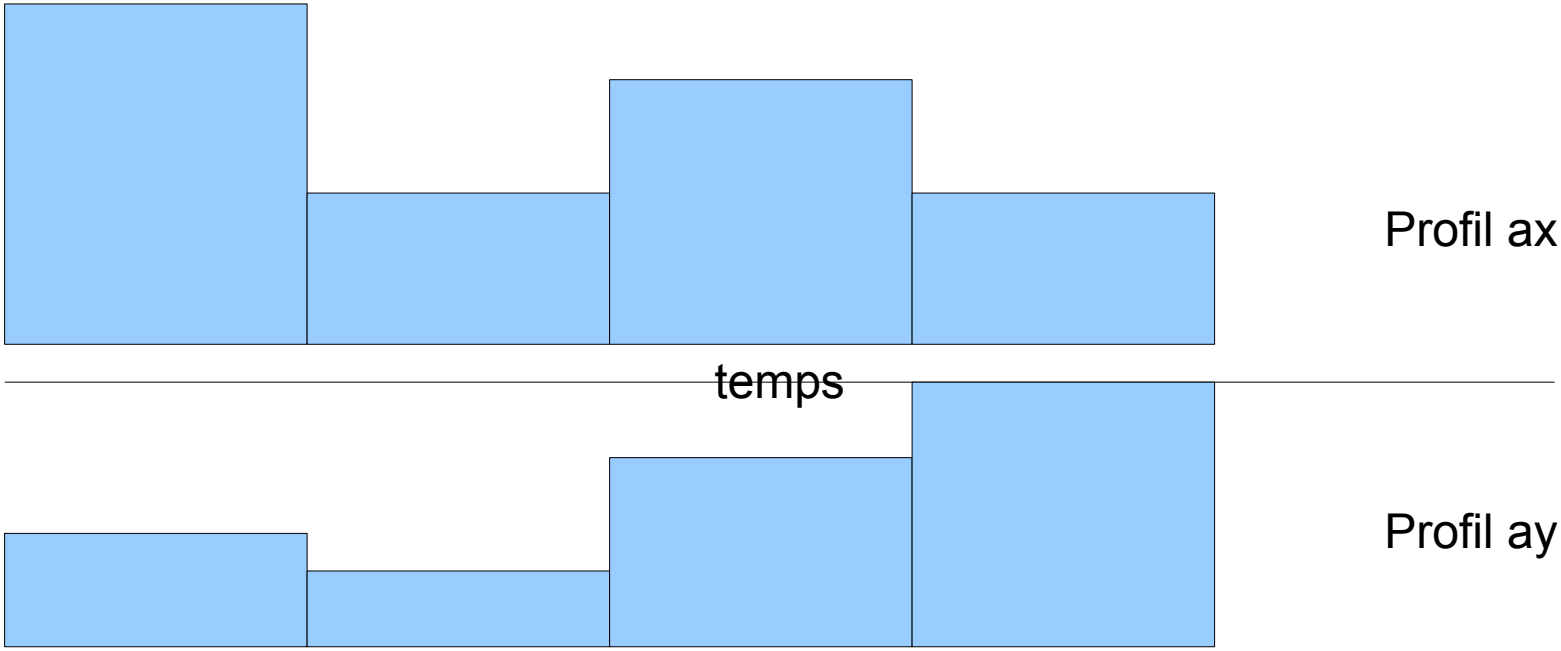
Voyageur de commerce

- Différences
  - Voyageur de commerce
  - Physical TSP (http://www.ptsp-game.net/)



Controle

- Raisonner sur des profils d'accélération



- Optimiser ces profils (recherche dans espace des fonctions)
  - Parcours aléatoire
  - Descente de gradient sur parametres
  - Algorithme genetiques

Controle

- Karl Sims
  - Evolved virtual creature (Siggraph '94)
  - http://www.karlsims.com/evolved-virtual-creatures.html

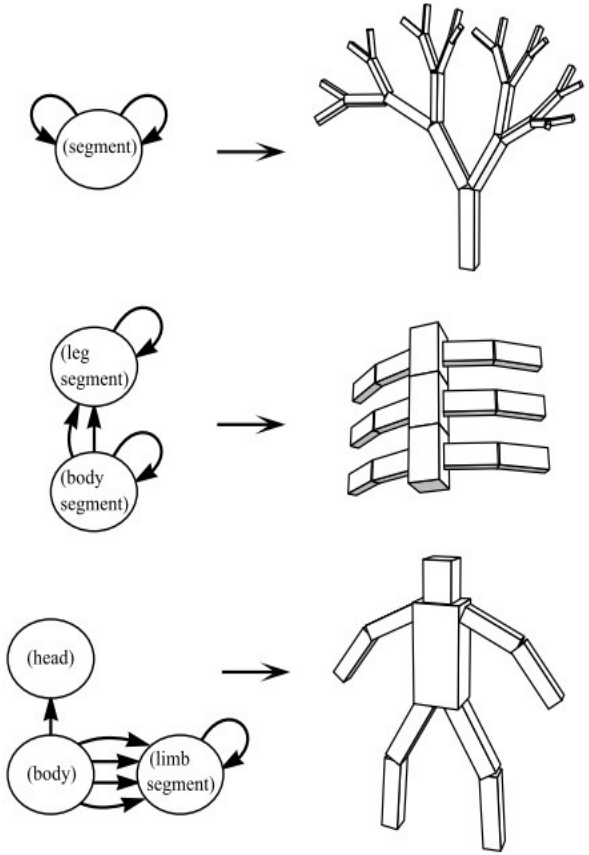
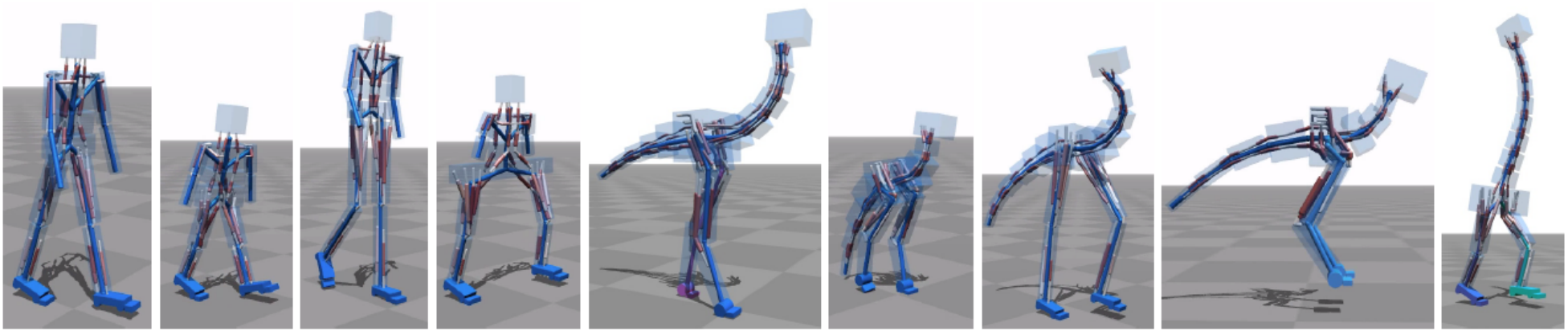


Figure 1: Designed examples of genotype graphs and corresponding creature morphologies.

Controle

- Flexible Muscle-Based Locomotion for Bipedal Creatures [Thomas Geijtenbeek et al. 2013]
  - http://goatstream.com/research/papers/SA2013/



  - https://www.youtube.com/watch?v=pgaEE27nsQw

Karl Sims

- Évolution forme + controle
  - Meilleures formes pour résoudre une tâche

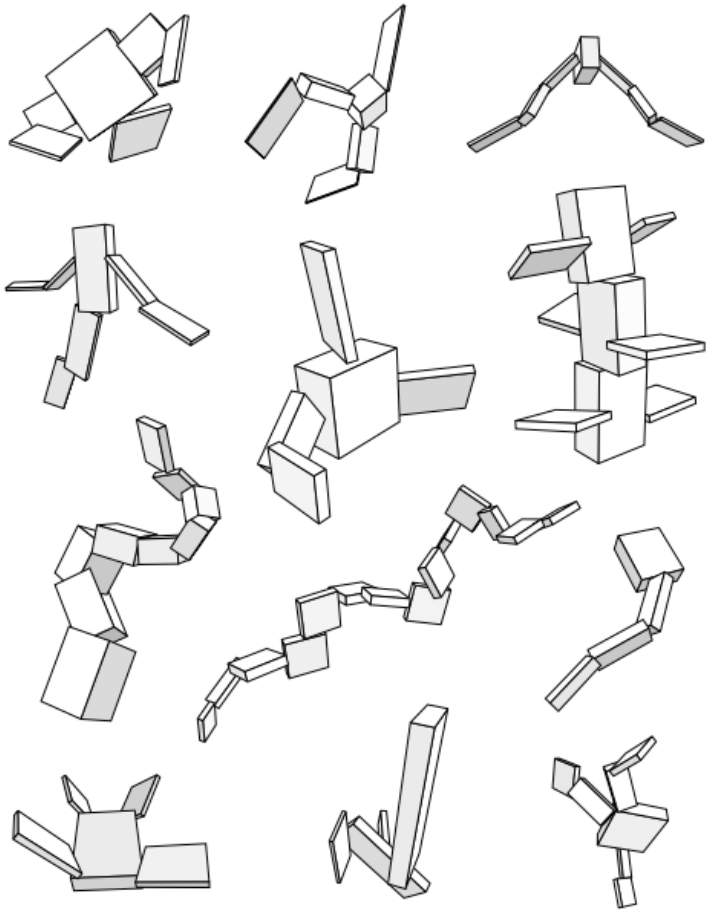


Figure 6: Creatures evolved for swimming.

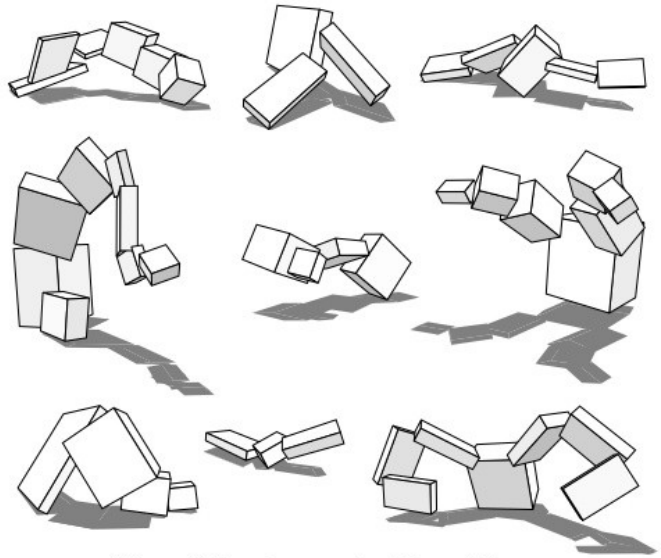


Figure 7: Creatures evolved for walking.

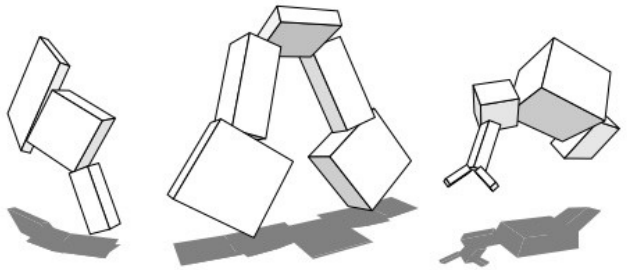
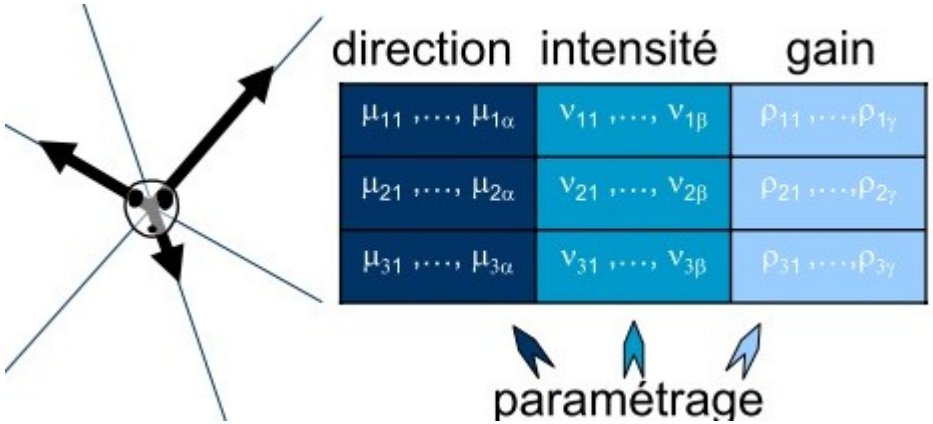
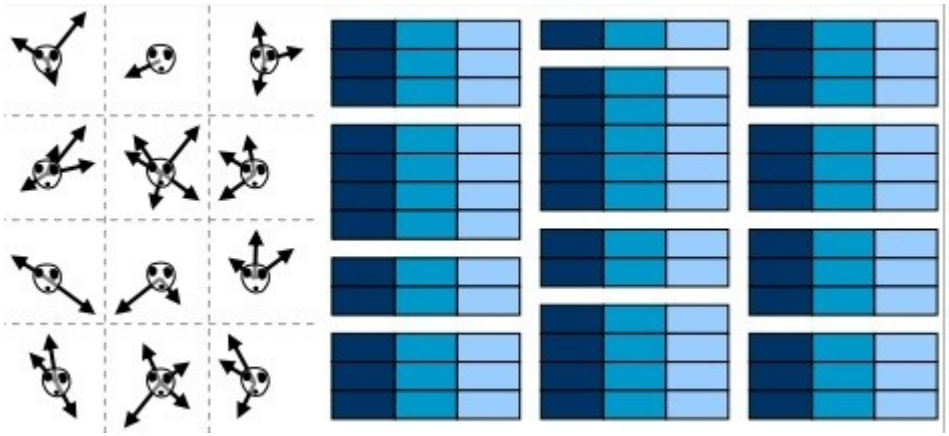


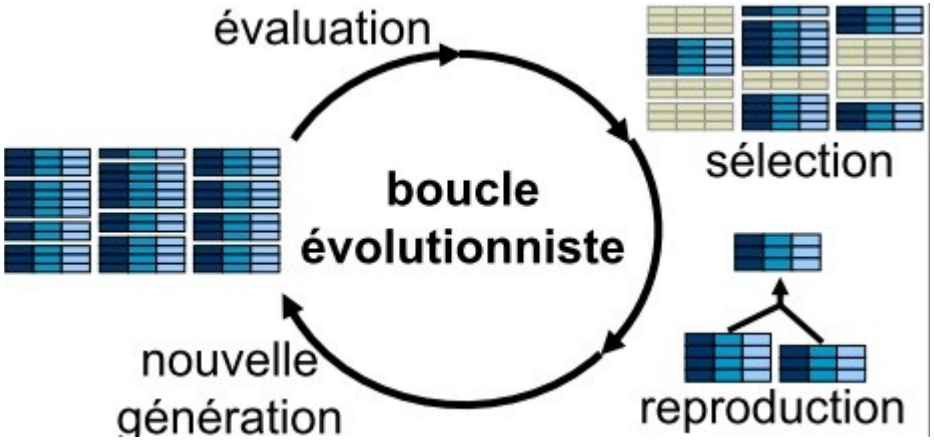
Figure 8: Creatures evolved for jumping.

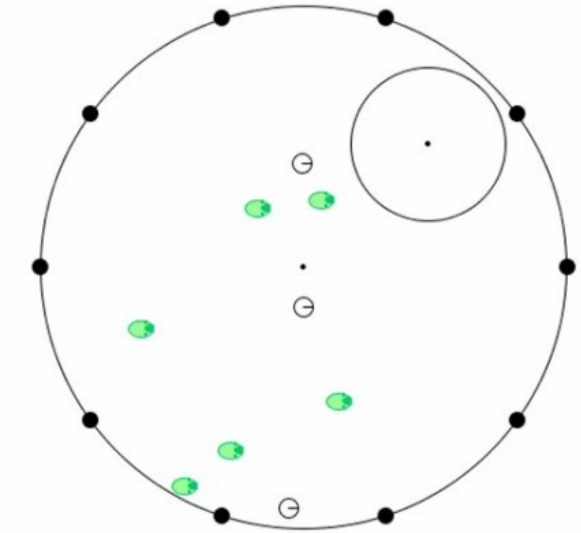
Gacs - Lois de controle

- GACS (Flacher 2005)









Probleme : capturer des moutons