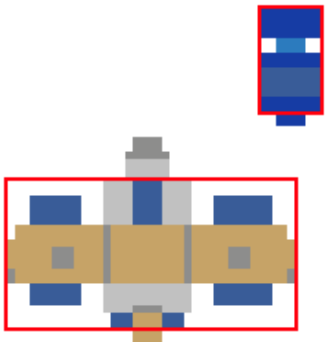
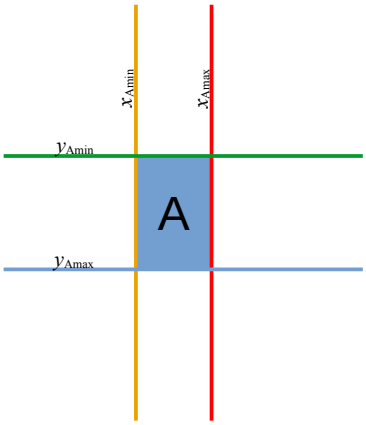
DÉTECTION DES COLLISIONS « AABB »

Qu'est-ce qu'une collision AABB ?



Ci-contre, un missile est en train de tomber sur un vaisseau spatial et, à chaque frame, on veut détecter s'il y a collision ou non.
Une façon simple et souvent suffisante de procéder est d'approximer chaque sprite par un rectangle (ici en rouge) dont les côtés sont parallèles aux axes et de déterminer si ces rectangles se chevauchent.
On parle alors de détection de collision « AABB » (Axis-Aligned Bounding Box).

Chevauchement de deux rectangles



On cherche donc à déterminer si deux rectangles A et B se chevauchent.

On a dessiné ci-contre le rectangle A **et on va raisonner à l'envers en identifiant plutôt les cas où le rectangle B ne touche pas A** :

- Soit B est complètement à gauche de la ligne jaune
- Soit B est complètement à droite de la ligne rouge
- Soit B est complètement au-dessus de la ligne verte
- Soit B est complètement en-dessous de la ligne bleue.

Fonction collision en Python

On définit les deux rectangles ci-dessus par 4 coordonnées :
Pour A : x_{Amin}, y_{Amin}, x_{Amax}, y_{Amax} et pour B : x_{Bmin}, y_{Bmin}, x_{Bmax}, y_{Bmax}.

Les 4 conditions de non chevauchement des 2 rectangles peuvent donc s'écrire :

- B est complètement à gauche de la ligne jaune : $x_{Bmax} < x_{Amin}$
- B est complètement à droite de la ligne rouge : ……
- B est complètement au-dessus de la ligne verte : ……
- B est complètement en-dessous de la ligne bleue : ……

On en déduit la fonction Python suivante :

```
def collision(xAmin, yAmin, xAmax, yAmax, xBmin, yBmin, xBmax, yBmax):
    return not (xBmax < xAmin  or  xBmin > xAmax  or  yBmax < yAmin  or  yBmin > yAmax)
```