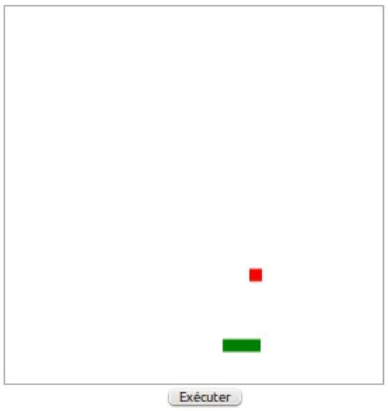
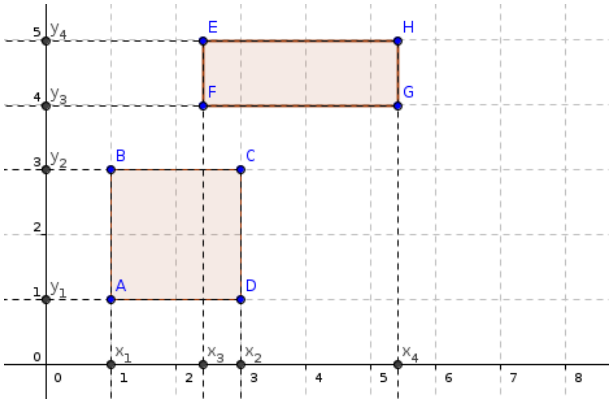
Voici ci-dessous une partie d'un programme informatique affichant à l'écran :
- un carré rouge qui se déplace du haut vers le bas
- et un rectangle vert qui se déplace de la droite vers la gauche.



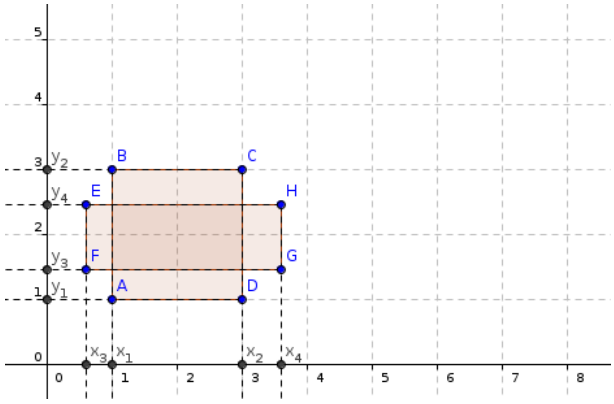
1. Introduction
2. Coordonnées, fonctions
3. Tests et conditions
4. Intervalles et collisions

```
var h = 2;
//centre du carré rouge
var x1 = 150;
var y1 = 300;
//centre du rectangle vert
var x2=100;
var y2=30;

function draw() {
  clear();
  rect(x1, y1, 10, 10,'red');
  if(y1 > 0)
    y1 = y1 - h;
  else{//si le carré rouge atteint le bord,
    y1 = 300; //on le remet en haut
    x1 = alea(0,300); //avec un "x" aléatoire
  }

  rect(x2,y2,30,10,'green');
  if(x2 > 0)
    x2 = x2 - h;
  else //si le rectangle vert atteint le bord,
    x2 = x2 + 300; //on le fait réapparaître sur le bord opposé
}

init(); //on initialise le jeu
loopdraw(); //on répète la fonction "draw" indéfiniment...
```

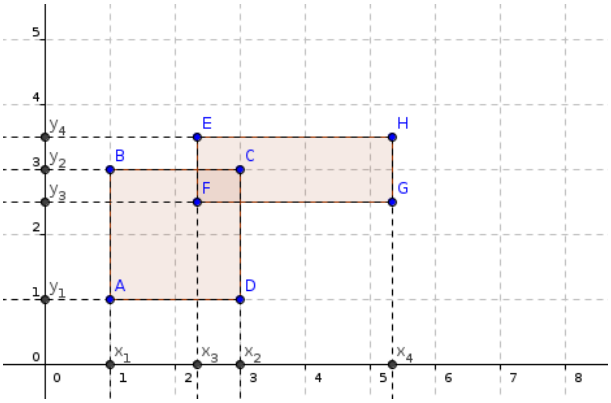
Parfois, le carré et le rectangle entrent en collision. On cherche à détecter ces collisions pour déclencher un événement, comme par exemple une explosion.



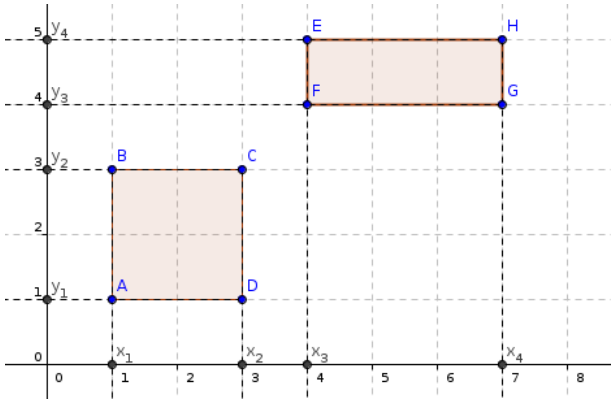
Exemple 1



Exemple 2



Exemple 3



Exemple 4

On suppose qu'on connaît seulement les coordonnées des points A, B, C, D et E, F, G, H :

$$x_1, x_2, x_3, x_4 \quad et \quad y_1, y_2, y_3, y_4$$

Comment peut-on savoir s'il y a une collision ?

Fiche prof de l'activité 1 :

Cette activité est faite pour :
1) voir un cas (autre que dans le contexte des inéquations) où le formalisme des intervalles est utile
2) manipuler un peu les intersections d'intervalles
3) faire un peu de logique
4) faire un peu d'algorithmique

- certains élèves veulent tout de suite écrire du code : il faut expliquer qu'on doit déjà résoudre le problème d'un point de vue théorique : quelle est la méthode qu'on va pouvoir utiliser sachant que la machine ne peut pas « voir » le dessin. Elle ne connaît que les 8 nombres qui sont les coordonnées.

- peut-être que certains élèves vont dire « il y a collision quand un point du rectangle vert est dans le carré rouge » ou quelque chose du même genre. L'exemple 2 permet de corriger cette intuition fausse.

- lorsque les élèves rentrent dans le problème, certains proposent des formulations du genre :
« il y a collision quand l'un des x du carré est entre les x du rectangle » : c'est une bonne piste… mais le professeur doit demander de l'exprimer en utilisant des inégalités ou alors en terme d'intersection d'intervalles : pour qu'il y ait collision, il faut que $[x_1;x_2]\cap[x_3;x_4]\neq\emptyset$

- attention, la condition précédente ne suffit pas :
il y a collision si $[x_1;x_2]\cap[x_3;x_4]\neq\emptyset$ ET si $[y_1;y_2]\cap[y_3;y_4]\neq\emptyset$
(Demander à un élève de venir dessiner un contre-exemple au tableau...)

- Si la classe est épuisée, on peut très bien en rester là : cela permet de signaler l'existence de la fonction « collision(x1,y1,w1,h1,x2,y2,w2,h2) » qui est rendue disponible dans la suite du parcours d'étude et de recherche et qui permet de rajouter des explosions dans les petits jeux vidéo des élèves..

- Si la classe a bien réagi et qu'on peut aller plus loin :
***Exercice* 1 :** formuler avec des intervalles « il n'y a pas collision » … (négation de la condition précédente)

- ensuite,
Exercice 2 : formuler cette négation en utilisant des inégalités à la place des intervalles. En effet, dans un langage de programmation on dispose en général des inégalités mais pas des signes « crochet » et «∩» .

Solution :
il n'y a pas collision si :

$$x_2<x_3 \quad ou \quad x_4<x_1 \quad ou \quad y_2<y_3 \quad ou \quad y_4<y_1$$

- Si on va jusqu'à une formulation en langage de programmation, voici un exemple en javascript (mais c'est beaucoup trop difficile pour rester exigible dans le cadre du programme de seconde).

```
function intersect(a1,b1,a2,b2){//intersection de deux intervalles
  //renvoie "false" si l'intersection de [a1 a2] et [b1 b2] est vide
  if(b1<a2 || b2<a1) return false;
  else return true;
}

function collision(x1,y1,w1,h1,x2,y2,w2,h2){
  //renvoie "true" si les deux rectangles se rencontrent
  if( intersect(x1-w1/2,x1+w1/2,x2-w2/2,x2+w2/2) && intersect(y1-h1/2,y1+h1/2,y2-h2/2,y2+h2/2) ) return
true;
  else return false;
}
```