

Complément mémoire:

```
int a = 5;
```

```
int* pA = &a;
```

```
float* pB = (float*) &a;
```

réinterprétation de pointeur

→ implique de la vigilance

car le compilateur ne pourra pas détecter les erreurs.

void* → Pointeur sans type associé

```
int a = 5;
```

```
void* pA = &a;
```

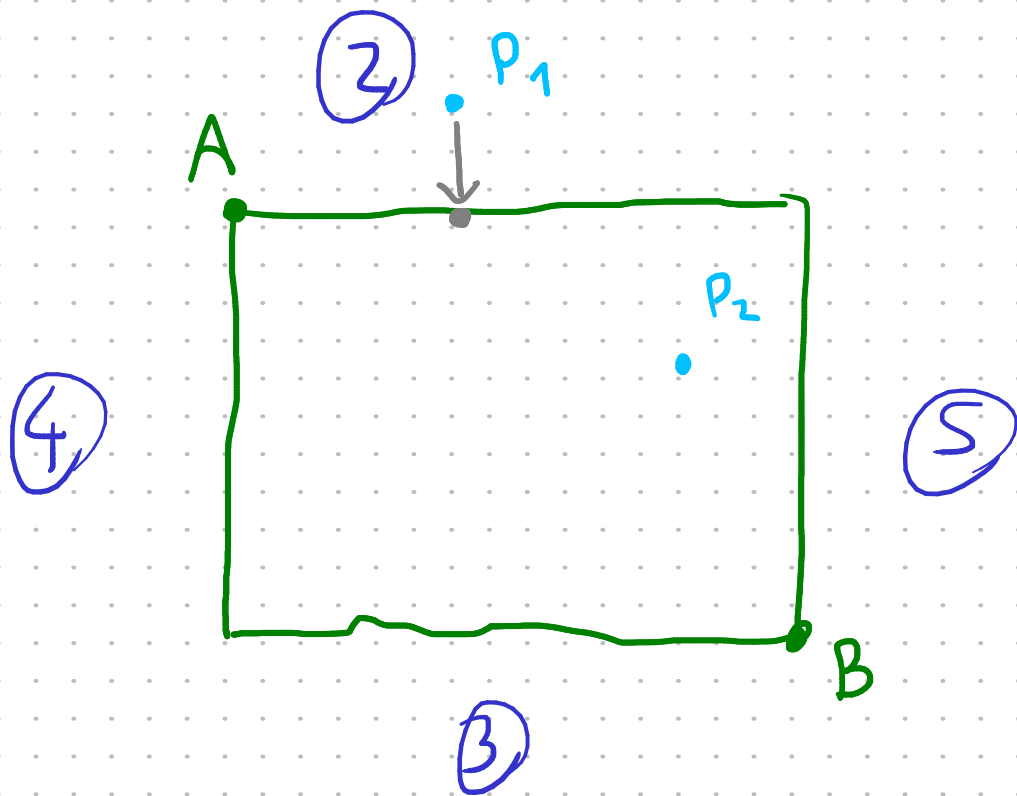
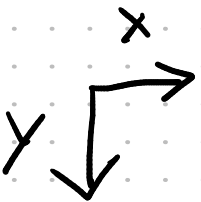
```
int* pB = pA;
```

Erreur : si le pointeur est sans Type, le compilateur ne peut pas vérifier.

```
int* pB = (int*) pA;
```

OK : Réinterprétation explicite

TD 71



(2) $P_1.y < A.y$
 $P_1.y = A.y$

(3) $P_1.y > B.y$
 $P_1.y = B.y$

(4) $P_1.x < A.x$
 $P_1.x = A.x$

(5) $P_1.x > B.x$
 $P_1.x = B.x$

Collisions

Dans le projet, $r = 8px$

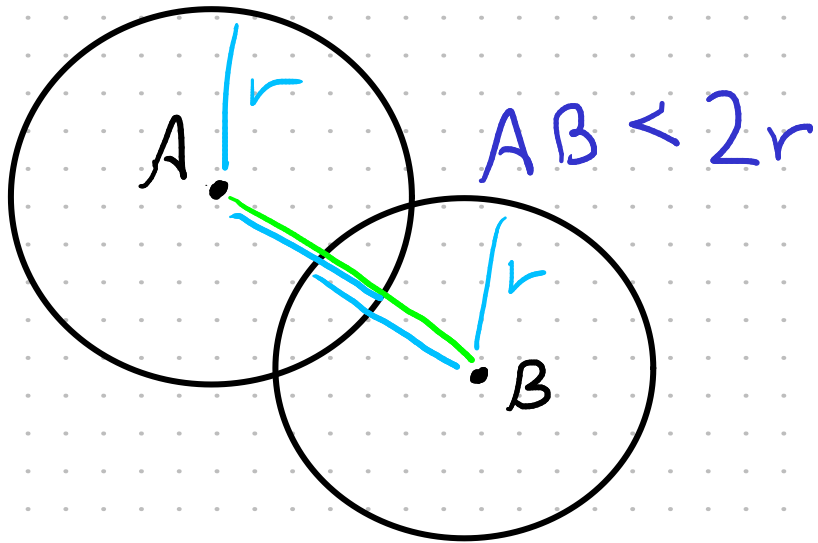
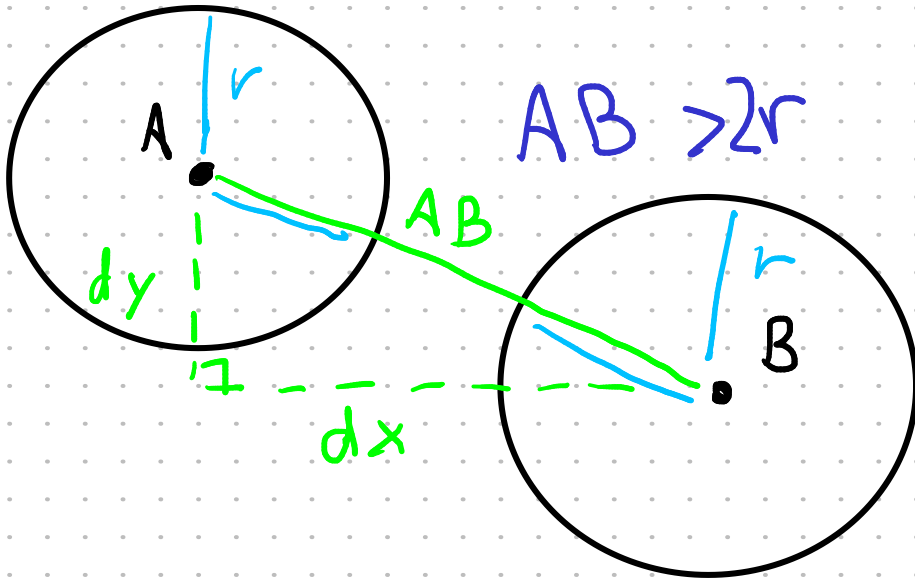
$$AB = \sqrt{dx^2 + dy^2}$$

$$AB = \sqrt{(A.x - B.x)^2 + (A.y - B.y)^2}$$

$$\sqrt{(A.x - B.x)^2 + (A.y - B.y)^2} < 2r ?$$

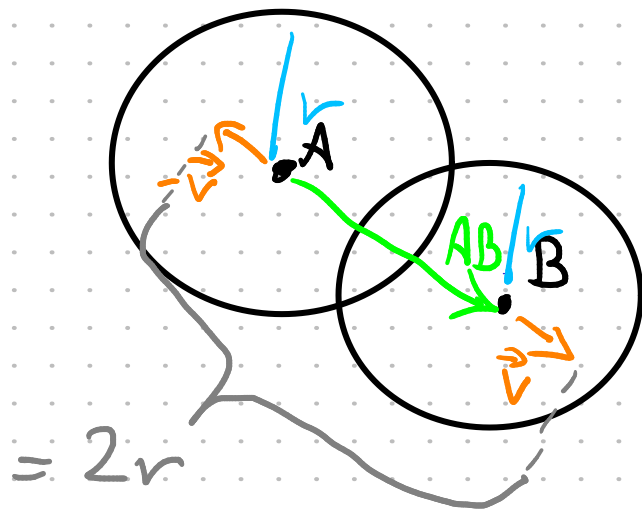
$$\square^2 \downarrow \square^2$$

$$(A.x - B.x) \times (A.x - B.x) + (A.y - B.y) \times (A.y - B.y) < 4 \times \frac{r^2}{8}$$
$$< 256$$



Résolution de collisions

$$\vec{v} = \alpha \vec{AB}$$



$$(\vec{A} - \vec{v}) \quad (\vec{B} + \vec{v}) = 2r$$

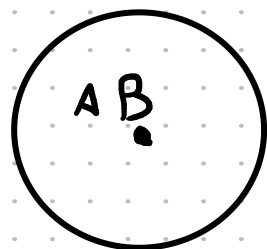
$$2v + AB = 2r$$

$$2\alpha AB + AB = 2r$$

$$2\alpha AB = 2r - AB$$

$$\alpha = \frac{2r - AB}{2AB}$$

$$\vec{v} = \alpha \vec{AB} = \frac{2r - AB}{2AB} \vec{AB}$$



$$ABsq = (A.x - B.x) \times (A.x - B.x) + (A.y - B.y) \times (A.y - B.y)$$

$$rsq = 8 \times 8$$

if (ABsq > 0 &&
ABsq < 4rsq) {

// Collision
// #include <math>

$$AB = \text{sqrt}(ABsq)$$

$$\alpha = \frac{2 \times 8 - AB}{2 \times AB}$$

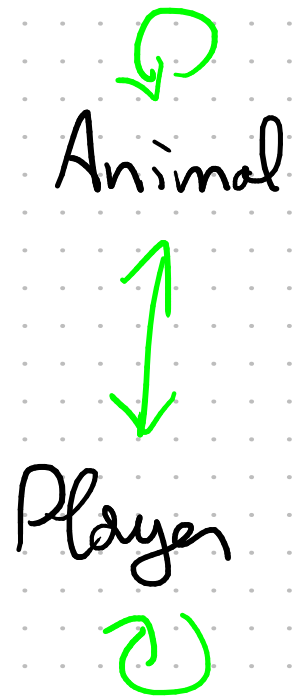
$$v.x = \alpha \times (B.x - A.x)$$

$$v.y = \alpha \times (B.y - A.y)$$

$$A.x -= v.x \quad A.y -= v.y$$

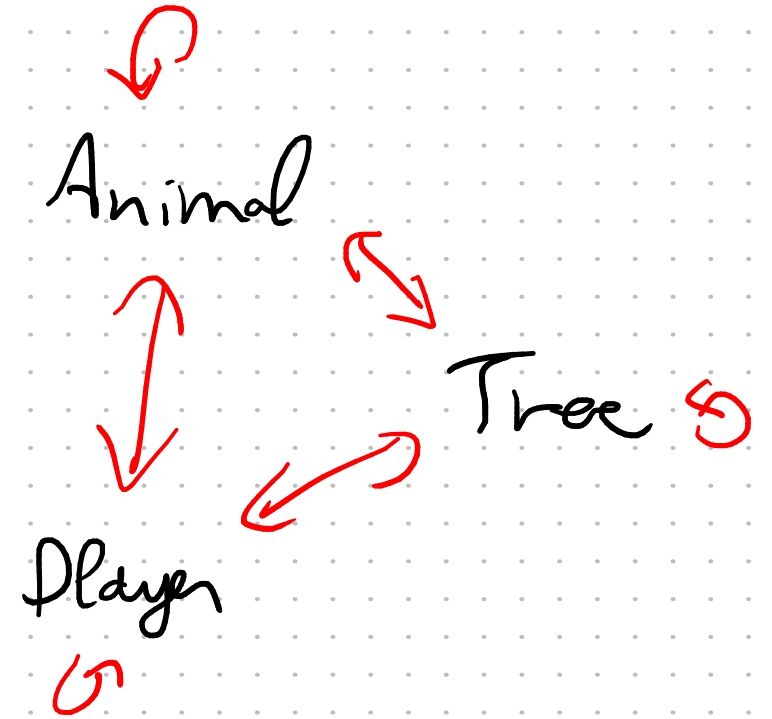
$$B.x += v.x \quad B.y += v.y$$

}



3 boucles pour gérer les collisions

6 boucles pour gérer les collisions



→ Problème de mise à l'échelle:
À chaque fois que j'ajoute un nouveau type d'élément,
Je double la taille du code (copier/coller de boucles)

class DinoEntity ← classe de base / basique

m_pos

SetPos() GetPos()

// gestion des collisions ?



Héritage

```
class DinoAnimal
: public DinoEntity
{
    ...
};
```

```
class DinoPlayer ← classe dérivée
: public DinoEntity
{
    ...
};
```

⇒ Gérer tous les cas de collisions en un seul endroit, dans une même liste

Imaginons

A1	A2	J1	A3	J2	J3	A4	...
----	----	----	----	----	----	----	-----

std::vector ne pourrait pas trouver l'emplacement du $n^{\text{ième}}$ élément
(pas de multiplication par sizeof(T) possible)

Solution

g-Player

J1	J2	J3	J4
----	----	----	----

g-Animals

A1	A2	A3	A4	A5
----	----	----	----	----

pEntities

--	--	--	--	--	--	--	--	--



Mise en commun → Réduire la taille de bug

On a parfois besoin de point "customizable"

Comportement à peu près identique dans les grandes lignes,
mais quelques détails différents.

⇒ Méthodes virtuelles / overridables