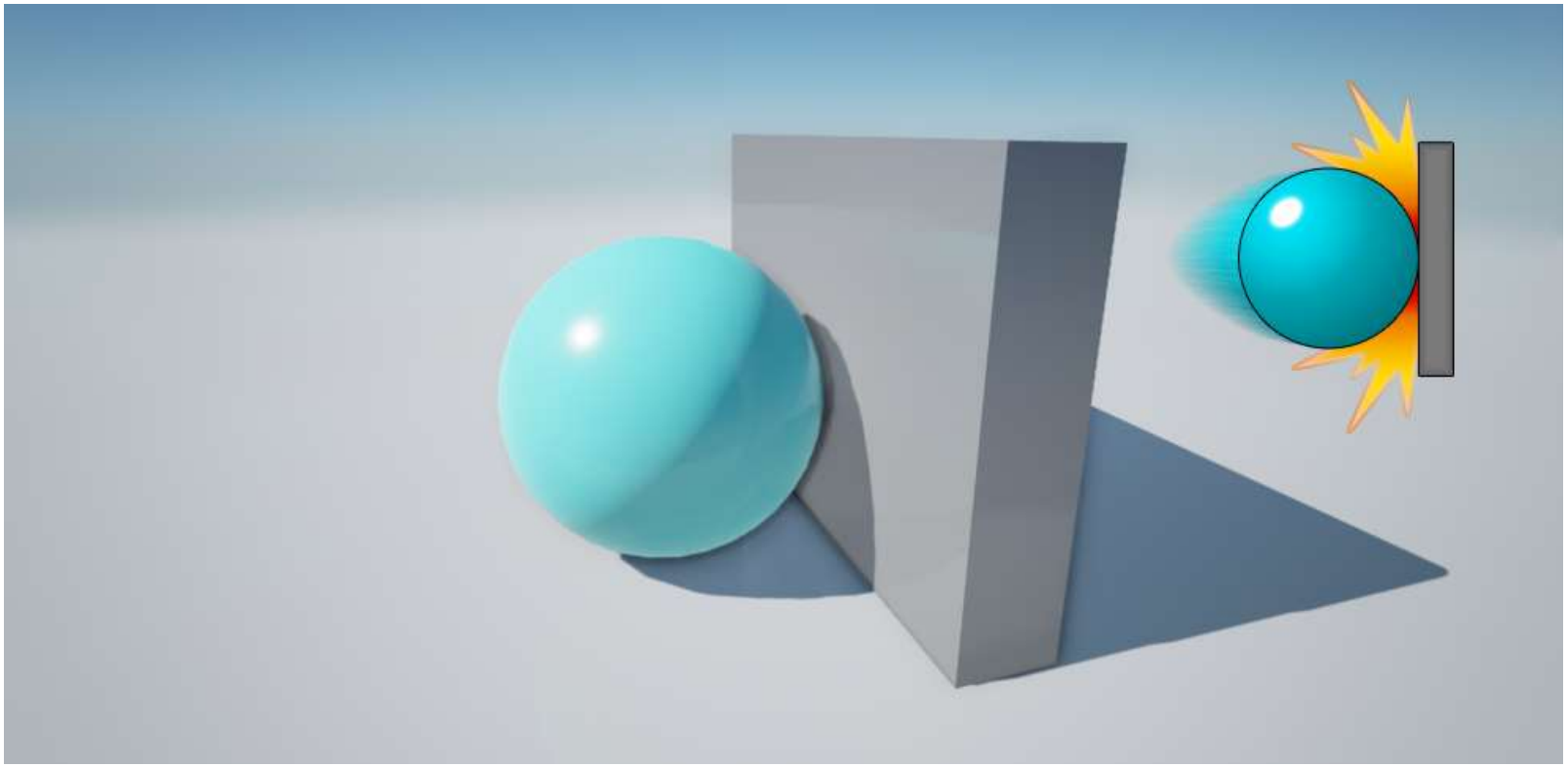
Colisiones

Dr. Ivan Sipiran

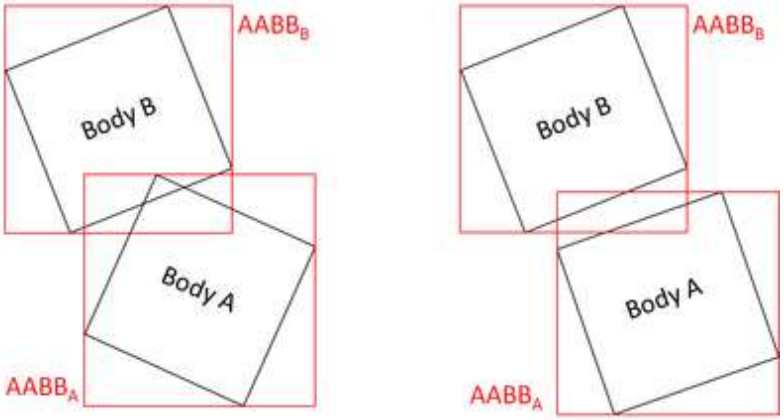
Introducción

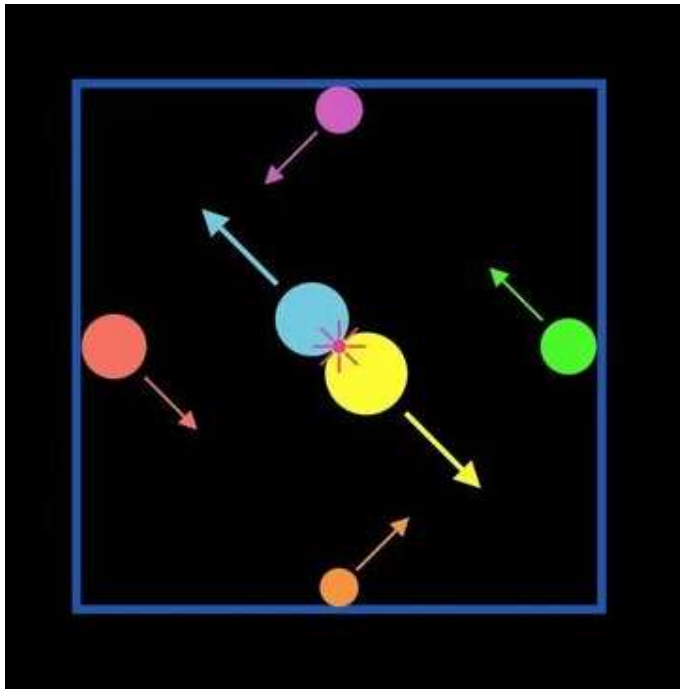
- Una forma de añadir realismo a una escena es simular la interacción entre los objetos.
- Un ejemplo es cuando dos objetos chocan y reaccionan al choque de la forma más parecida a la realidad



Introducción

- A nivel computacional, tenemos dos etapas que tenemos que resolver:
  - La detección de la colisión (saber si dos objetos chocaron)
  - La respuesta física de la colisión (qué pasa con los objetos luego de chocar)



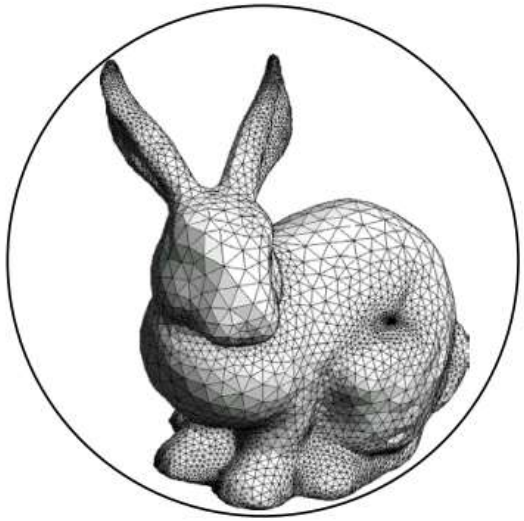


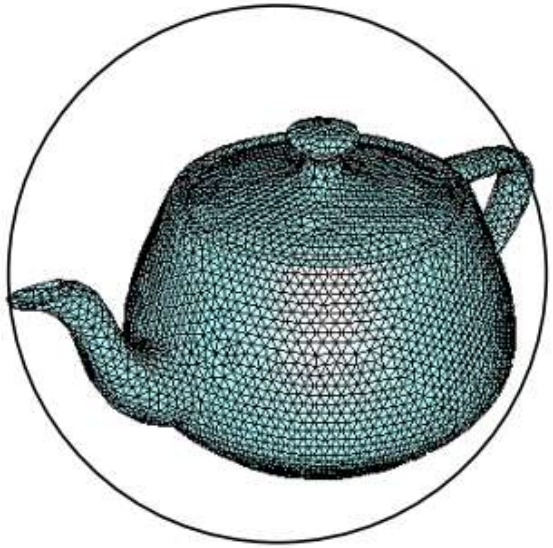
Detección de colisiones

- Los objetos de una escena podrían ser geométricamente muy complejos, por lo que realizar un análisis de colisiones exacto sería computacionalmente costoso.
- Una solución aproximada es usar una geometría proxy para cada objeto (esferas, rectángulos contenedores, etc)
- Es más fácil hacer la detección de colisiones entre formas simples que entre formas complejas

Colisiones de esferas

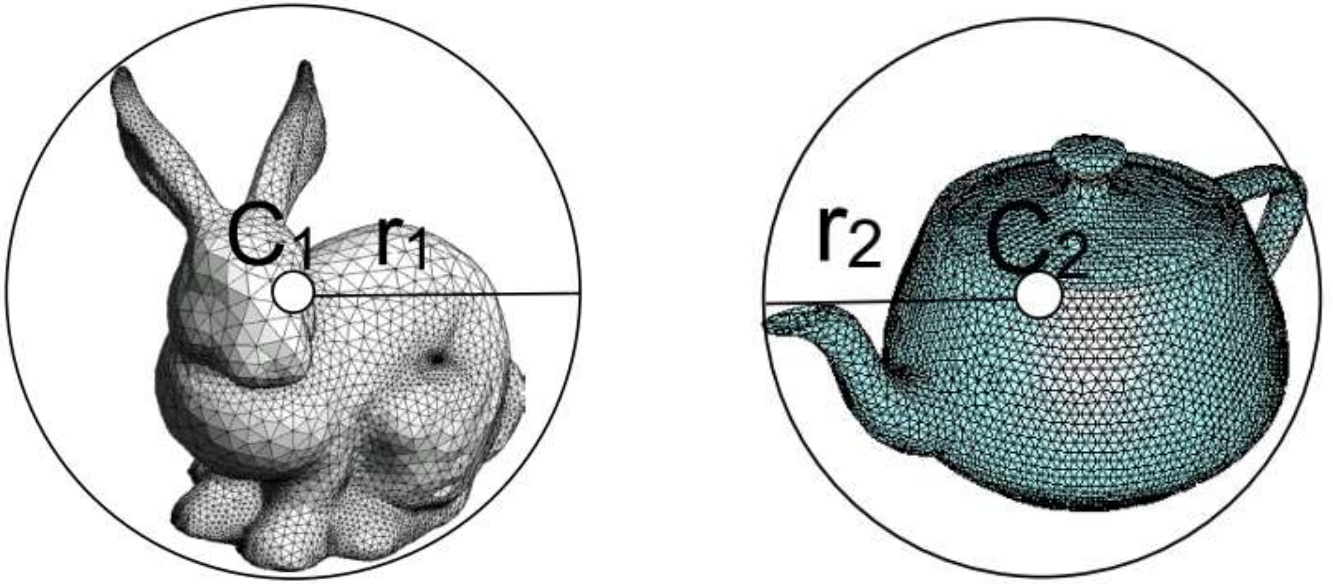
- Localizar esferas alrededor de los objetos
- Si las esferas no se intersecan, los objetos tampoco!
- Colisión esfera-esfera es computacionalmente fácil





Colisiones de esferas

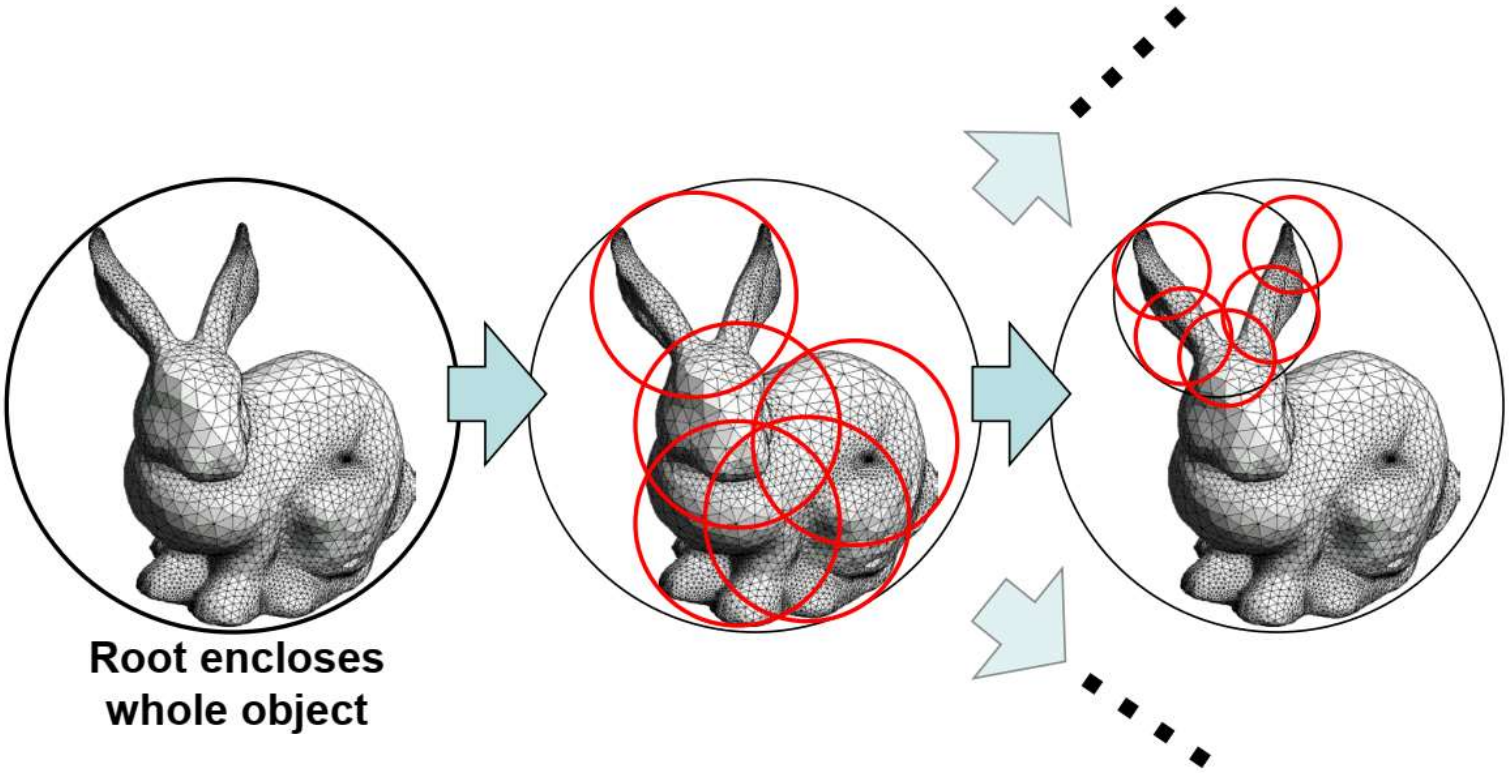
- Dos esferas con centros C_1 y C_2, y radios r_1 y r_2
- Las esferas intersecan solo si $\|C_1C_2\| < r_1 + r_2$



- Este tipo de test podría no dar un resultado muy fino. Algunas veces necesitamos mayor precisión.

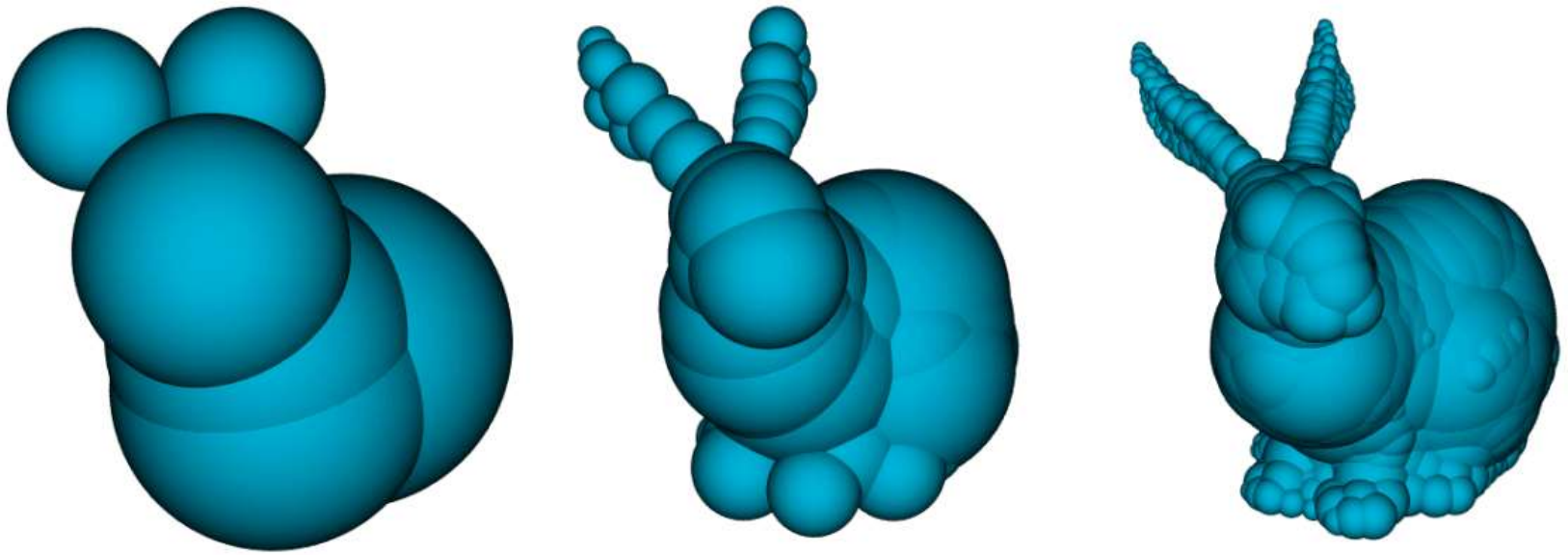
Análisis Jerárquico

- Jerarquía de esferas
  - Organizadas en un árbol
- Test recursivo con recorte temprano



Análisis Jerárquico

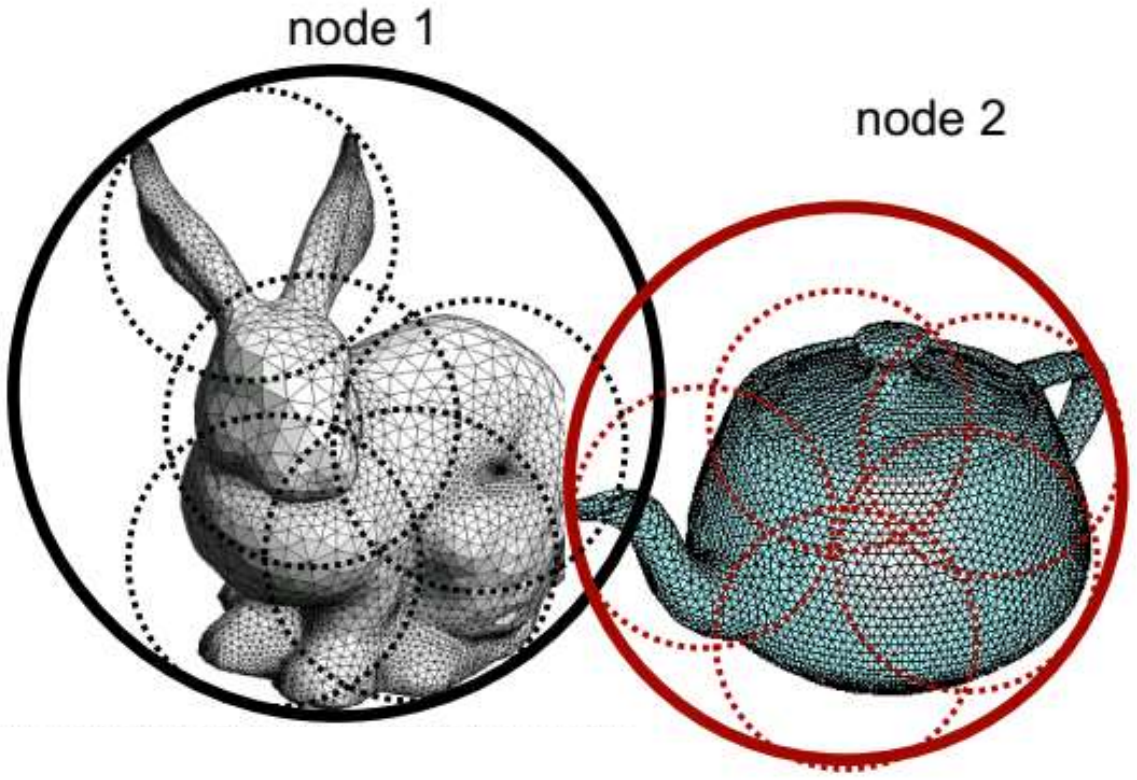
- Representación jerárquica de esferas



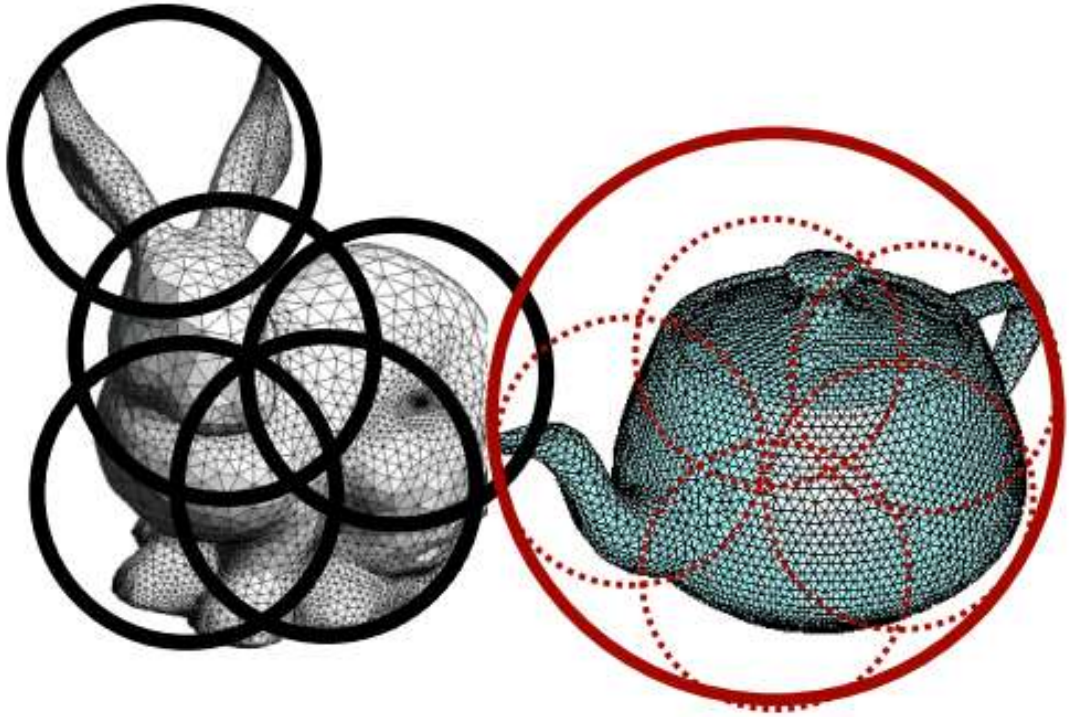
Análisis Jerárquico (versión simple)

```
boolean intersect(node1, node2)
   // no overlap? ==> no intersection!
   if (!overlap(node1->sphere, node2->sphere)
      return false

   // recurse down the larger of the two nodes
   if (node1->radius()>node2->radius())
      for each child c of node1
         if intersect(c, node2) return true
   else
      for each child c f node2
        if intersect(c, node1) return true

   // no intersection in the subtrees? ==> no intersection!
   return false
```

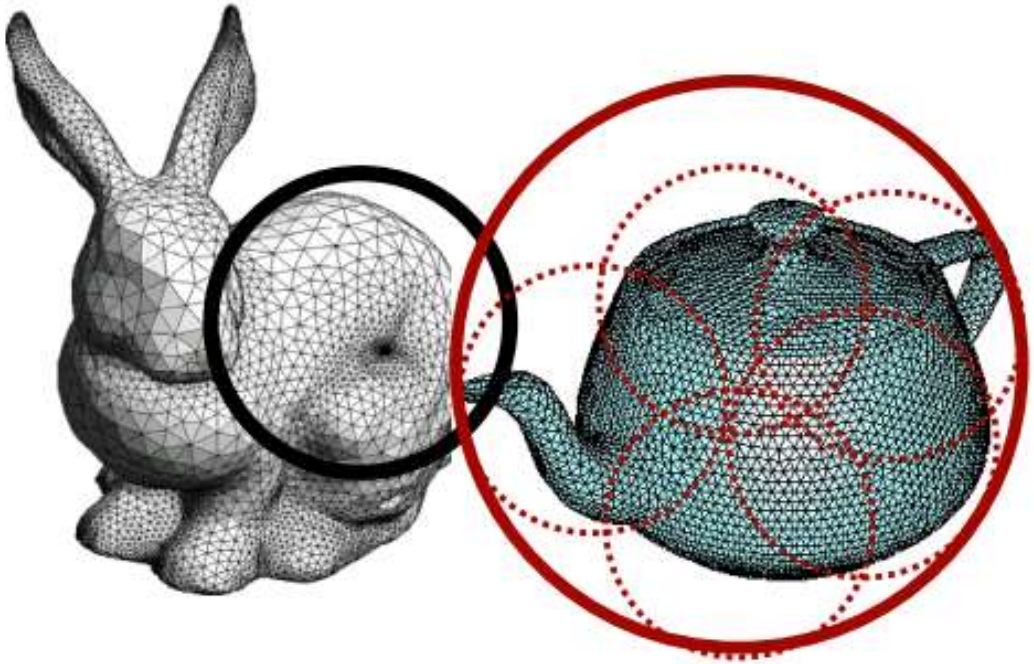
```
boolean intersect(node1, node2)
   if (!overlap(node1->sphere, node2->sphere)
      return false
   if (node1->radius()>node2->radius())
      for each child c of node1
         if intersect(c, node2) return true
   else
      for each child c f node2
        if intersect(c, node1) return true
   return false
```



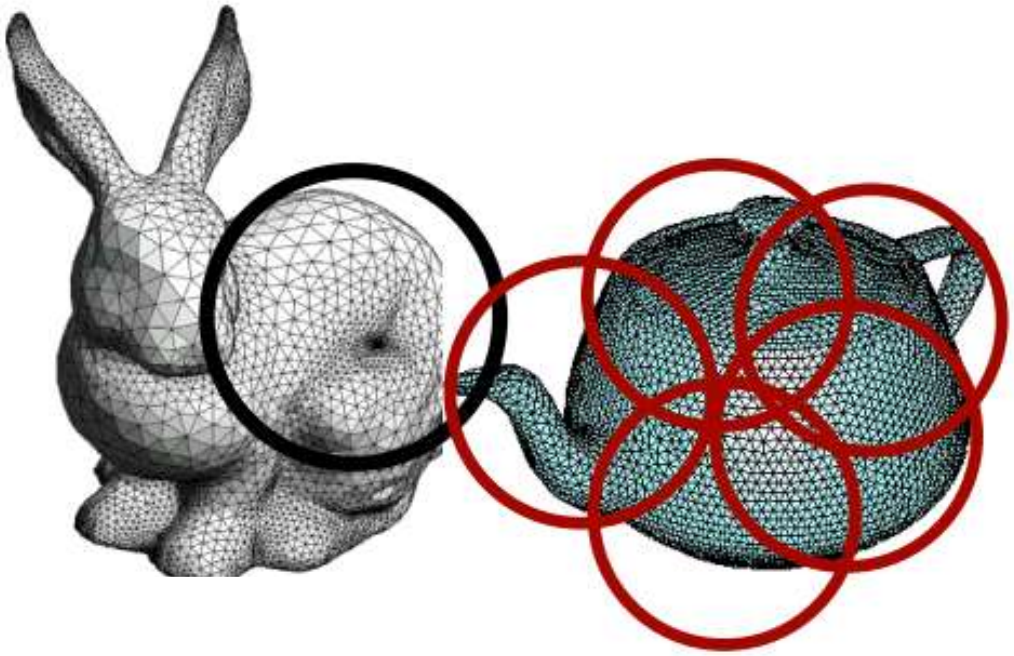
```
boolean intersect(node1, node2)
    if (!overlap(node1->sphere, node2->sphere)
        return false
    if (node1->radius()>node2->radius())
        for each child c of node1
            if intersect(c, node2) return true
    else
        for each child c f node2
            if intersect(c, node1) return true
    return false
```



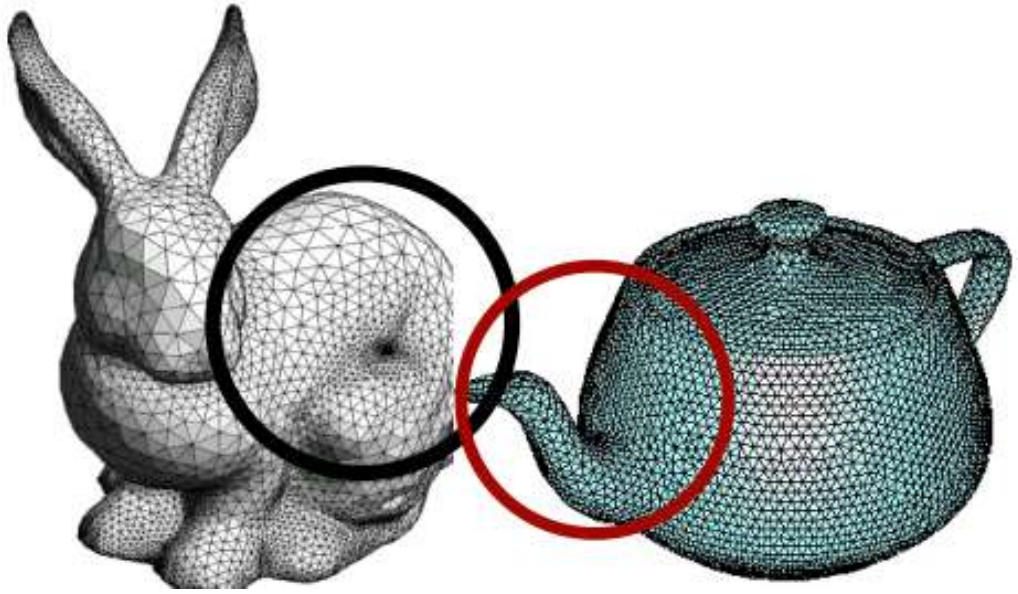
```
boolean intersect(node1, node2)
    if (!overlap(node1->sphere, node2->sphere)
        return false
    if (node1->radius()>node2->radius())
        for each child c of node1
            if intersect(c, node2) return true
    else
        for each child c f node2
            if intersect(c, node1) return true
    return false
```



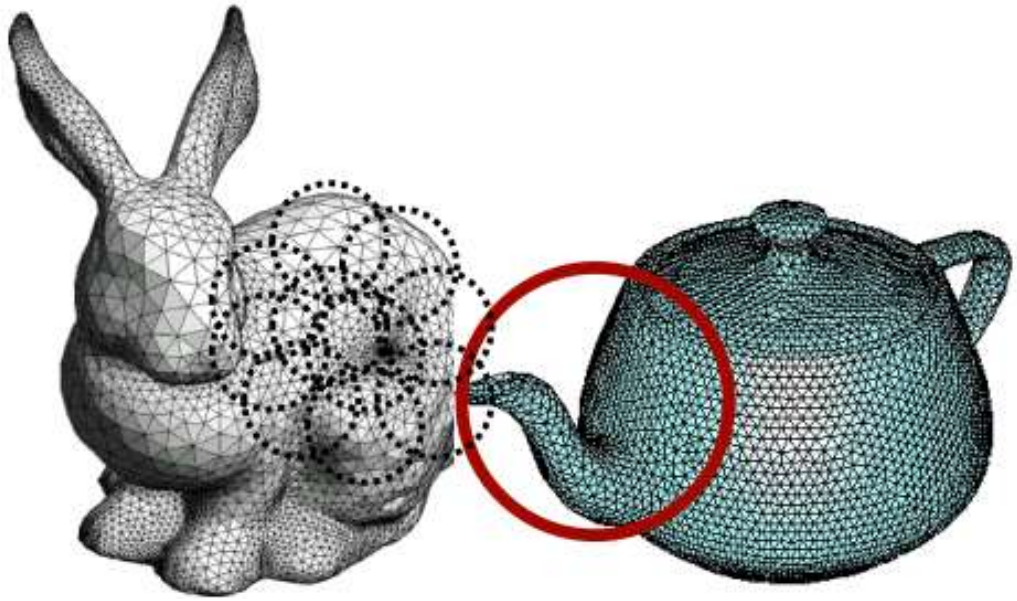
```
boolean intersect(node1, node2)
    if (!overlap(node1->sphere, node2->sphere)
        return false
    if (node1->radius()>node2->radius())
        for each child c of node1
            if intersect(c, node2) return true
    else
        for each child c f node2
            if intersect(c, node1) return true
    return false
```



```
boolean intersect(node1, node2)
   if (!overlap(node1->sphere, node2->sphere)
      return false
   if (node1->radius()>node2->radius())
      for each child c of node1
         if intersect(c, node2) return true
   else
      for each child c f node2
        if intersect(c, node1) return true
   return false
```



```
boolean intersect(node1, node2)
    if (!overlap(node1->sphere, node2->sphere)
        return false
    if (node1->radius()>node2->radius())
        for each child c of node1
            if intersect(c, node2) return true
    else
        for each child c f node2
            if intersect(c, node1) return true
    return false
```



Análisis jerárquico (con nodos hoja)

```
boolean intersect(node1, node2)
   if (!overlap(node1->sphere, node2->sphere)
      return false

   // if there is nowhere to go, test everything
   if (node1->isLeaf() && node2->isLeaf())
      perform full test between all primitives within nodes

   // otherwise go down the tree in the non-leaf path
   if ( !node2->isLeaf() && !node1->isLeaf() )
      // pick the larger node to subdivide, then recurse
   else
      // recurse down the node that is not a leaf

   return false
```

Otros tipos de primitivas

Caja vs Caja

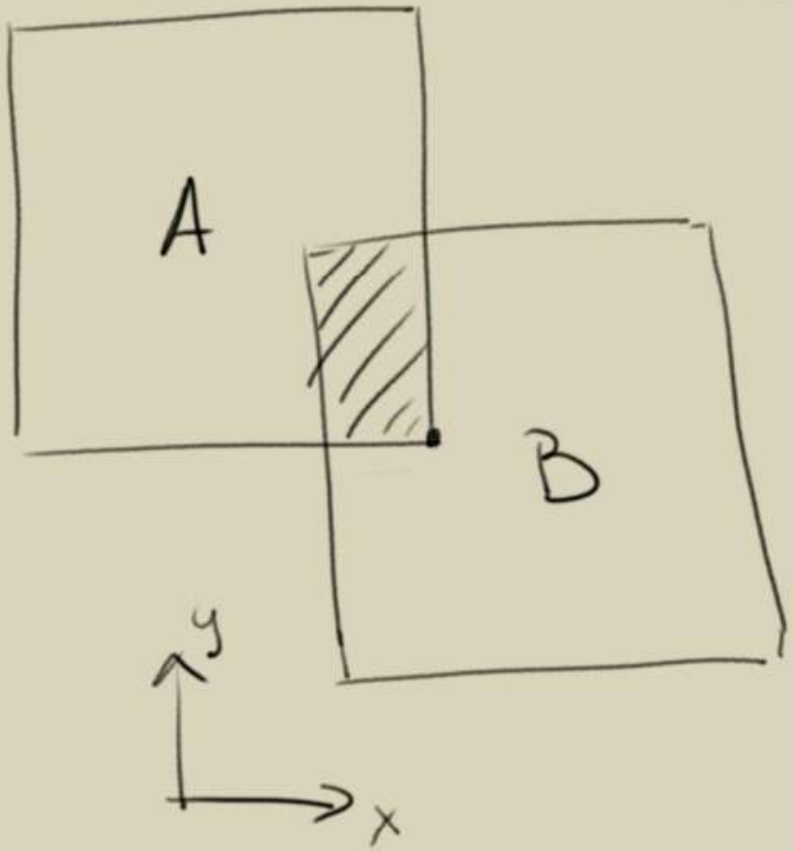
2D/3D

i.e.

AABox: Axis Aligned Box

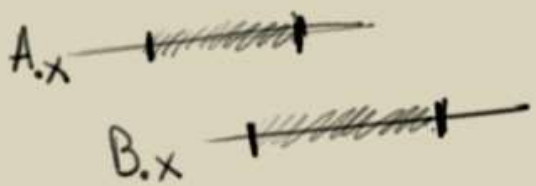
modela

- Bounding Box
- Sprite
- Rectángulo

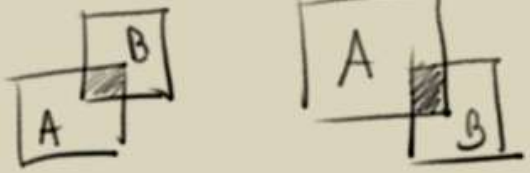


Intersección en X

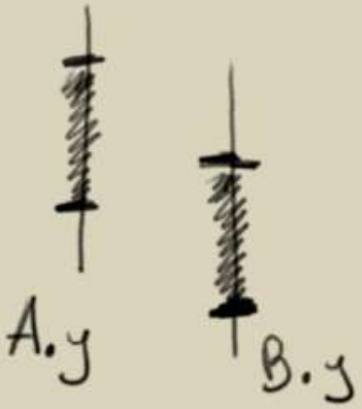
$$\wedge \begin{cases} A.x_{max} > B.x_{min} \\ A.x_{min} < B.x_{max} \end{cases}$$



Intersección en Y



$$\wedge \begin{cases} A.y_{max} > B.y_{min} \\ A.y_{min} < B.y_{max} \end{cases}$$

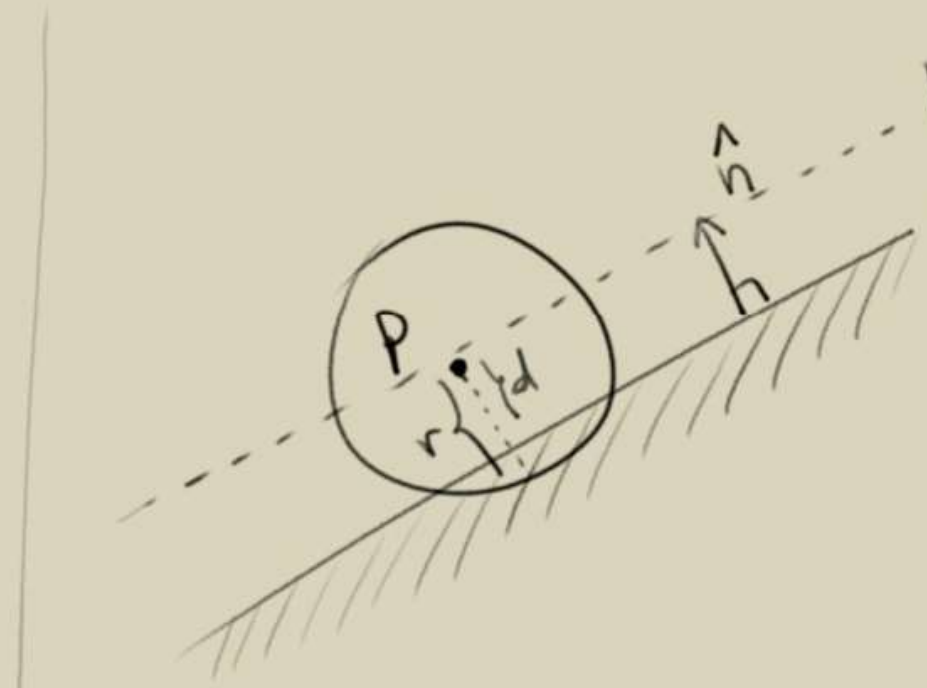


Intersección de AABox <=>

Intersección en ejes X e Y

- Barato pues solo necesita comparaciones

Esfera vs Plano



$d < r \Rightarrow$ colisión!

$P \cdot L < r$

$L = (\hat{n}, D)$ ← vector 4D representando el plano

$P \cdot \hat{n} + D < r$

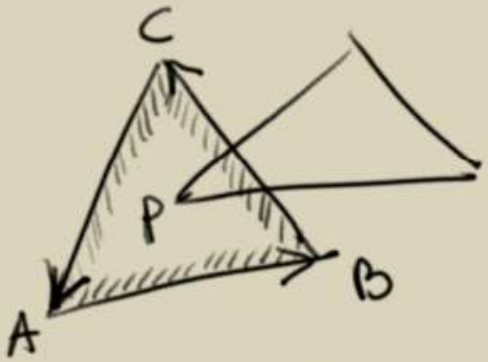
$\boxed{P \cdot \hat{n} + (D - r) < 0}$ ← colisión del punto P con el plano L'

$L' = (\hat{n}, D - r)$

→ Punto versus Plano !! ☺

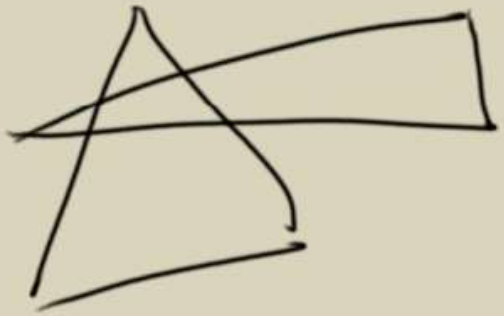
Triángulo vs Triángulo [2D]

~ Malla 2D vs Malla 2D.



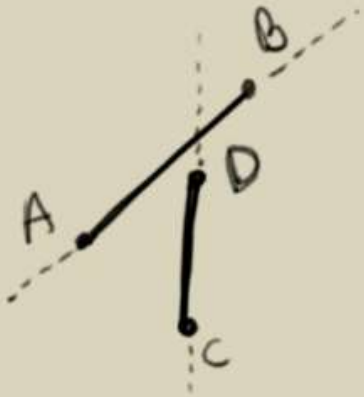
- Vértice en el interior
- P está a la derecha de todos los lados.

$$\|\vec{AB} \times \vec{AP}\| \geq 0$$



- Intersección de segmentos

Intersección de segmentos



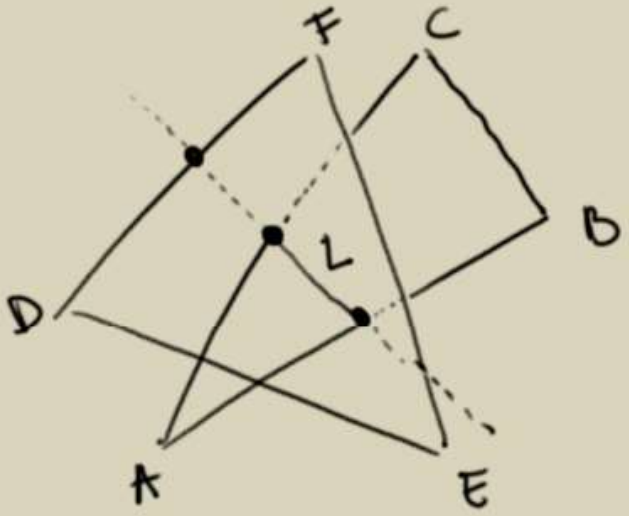
- Intersección de rectas no es suficiente.
- Puntos deben estar en lados diferentes de un segmento.

Triángulo vs Tr:ángulo [3D]

1.- Calcular la ecuación del plano del Δ_1. Si todos los vértices de Δ_2 están al mismo lado del plano
$\Rightarrow$ No hay intersección.

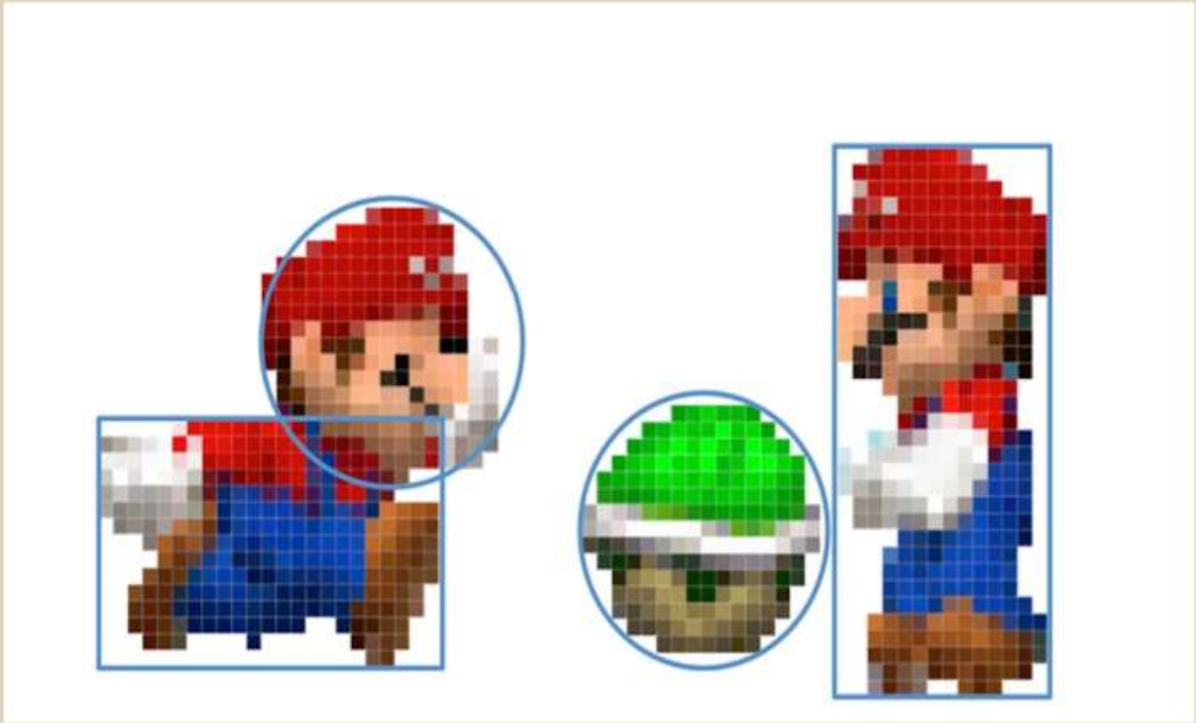
2.- Calcular la intersección de Δ_2 con el plano de Δ_1. Esto corresponde a un segmento

3.- Revisar si el segmento intersecta o está contenido en el Δ_1. Si está contenido, los Δ_s se intersectan. En caso contrario, No se intersectan.



Colisiones en videojuegos

- Formas básicas para disminuir el costo.
- Detección de interacciones.



Física de Colisiones

Colisión Elástica en 1D

In an elastic collision, both momentum and kinetic energy are conserved.[1] Consider particles 1 and 2 with masses m_1, m_2, and velocities u_1, u_2 before collision, v_1, v_2 after collision. The conservation of the total momentum before and after the collision is expressed by:[1]

$$m_1 u_1 + m_2 u_2 \ = \ m_1 v_1 + m_2 v_2 .$$

Likewise, the conservation of the total kinetic energy is expressed by:[1]

$$\tfrac{1}{2} m_1 u_1^2 + \tfrac{1}{2} m_2 u_2^2 \ = \ \tfrac{1}{2} m_1 v_1^2 + \tfrac{1}{2} m_2 v_2^2 .$$

These equations may be solved directly to find v_1, v_2 when u_1, u_2 are known:[2]

$$v_1 \ = \ \frac{m_1 - m_2}{m_1 + m_2} u_1 + \frac{2m_2}{m_1 + m_2} u_2$$

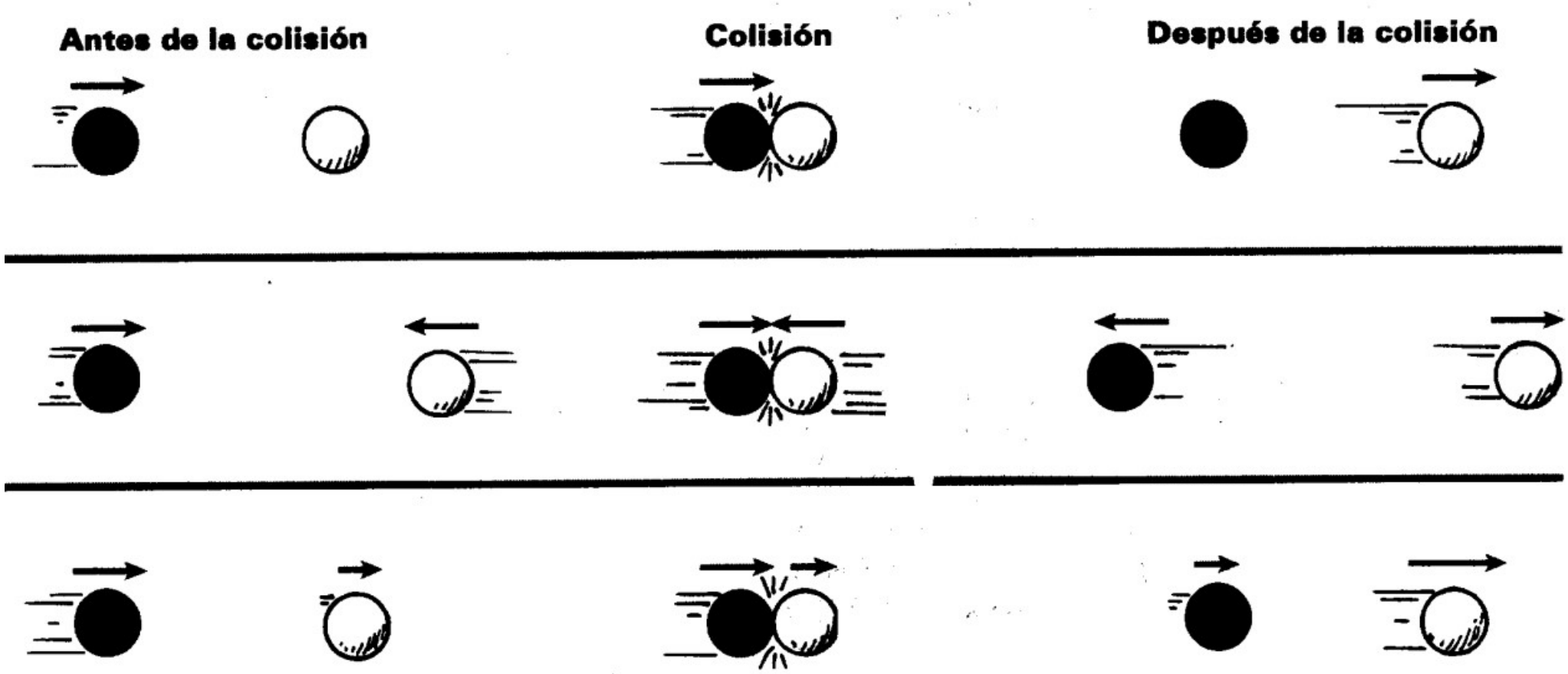
$$v_2 \ = \ \frac{2m_1}{m_1 + m_2} u_1 + \frac{m_2 - m_1}{m_1 + m_2} u_2$$

If both masses are the same, we have a trivial solution:

$$v_1 = u_2$$
$$v_2 = u_1 .$$

This simply corresponds to the bodies exchanging their initial velocities to each other.[2]

https://en.wikipedia.org/wiki/Elastic_collision


Antes de la colisión
Colisión
Después de la colisión

Colisión Elástica en 1D

$$m_1 u_1 + m_2 u_2 = m_1 v_1 + m_2 v_2$$
$$\tfrac{1}{2} m_1 u_1^2 + \tfrac{1}{2} m_2 u_2^2 = \tfrac{1}{2} m_1 v_1^2 + \tfrac{1}{2} m_2 v_2^2$$

Derivation of solution [edit]

To derive the above equations for v_1, v_2, rearrange the kinetic energy and momentum equations:

$$m_1(v_1^2 - u_1^2) = m_2(u_2^2 - v_2^2)$$
$$m_1(v_1 - u_1) = m_2(u_2 - v_2)$$

Dividing each side of the top equation by each side of the bottom equation, and using $\frac{a^2-b^2}{(a-b)} = a + b$, gives:

$$v_1 + u_1 = u_2 + v_2 \quad \Rightarrow \quad v_1 - v_2 = u_2 - u_1 \,.$$

That is, the relative velocity of one particle with respect to the other is reversed by the collision.

Now the above formulas follow from solving a system of linear equations for v_1, v_2, regarding m_1, m_2, u_1, u_2 as constants:

$$\begin{cases} v_1 - v_2 = u_2 - u_1 \\ m_1 v_1 + m_2 v_2 = m_1 u_1 + m_2 u_2 . \end{cases}$$

Once v_1 is determined, v_2 can be found by symmetry.

https://en.wikipedia.org/wiki/Elastic_collision

https://en.wikipedia.org/wiki/Elastic_collision

Colisión Elástica en 2D

For the case of two colliding bodies in two dimensions, the overall velocity of each body must be split into two perpendicular velocities: one tangent to the common normal surfaces of the colliding bodies at the point of contact, the other along the line of collision. Since the collision only imparts force along the line of collision, the velocities that are tangent to the point of collision do not change. The velocities along the line of collision can then be used in the same equations as a one-dimensional collision. The final velocities can then be calculated from the two new component velocities and will depend on the point of collision

Colisión Elástica en 2D