Algoritmos

Colisiones

Logro

Al finalizar la sesión, el estudiante elabora programas **haciendo uso de** colisión de objetos.

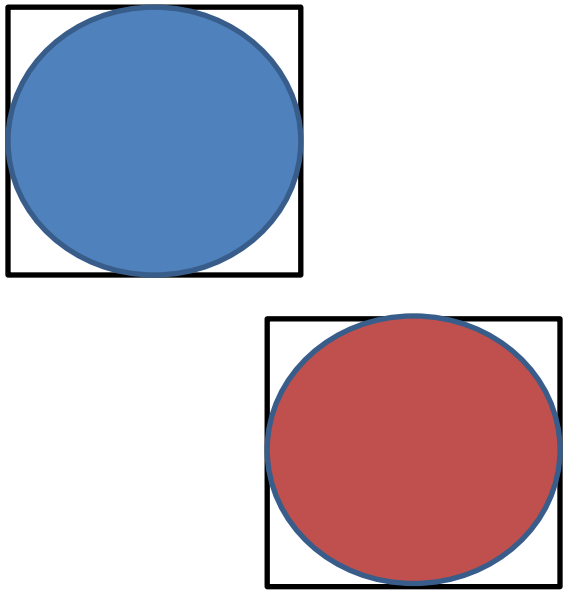
Colisiones

En el curso de Programación II, cuando pienses en colisiones, independiente de la figura, debes de pensar que lo que están colisionando son dos rectángulos.

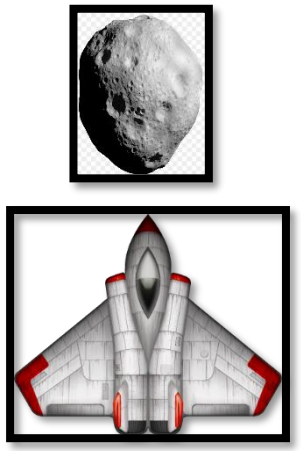
Colisiones

¿Por qué?

Porque independiente de la forma que veas, esta siempre tiene un contorno y el contorno es un rectángulo.



Colisión entre dos círculos



Colisión entre un asteroide y una nave

Colisiones

Si reducimos la colisión de las formas a la colisión de dos rectángulos, el método se simplifica
Podemos usar los siguientes métodos:

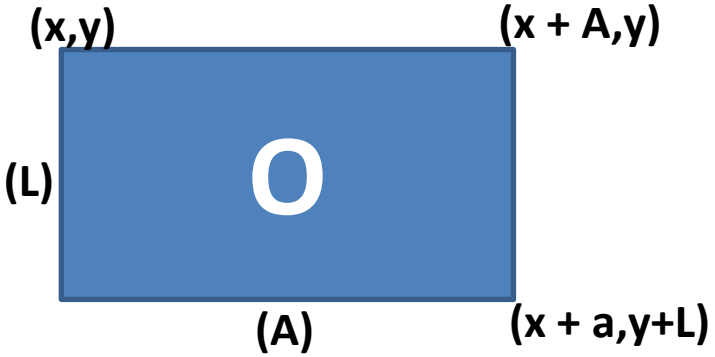
- ✓ Algoritmo Bounding Box
- ✓ Función Intersect de System:Drawing:Graphics
- ✓ Distancia entre dos puntos

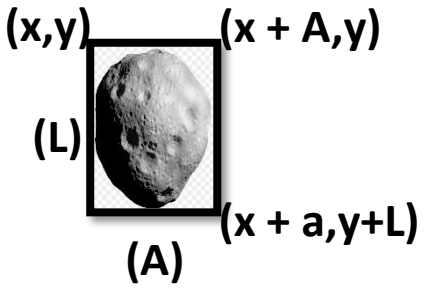
Colisiones

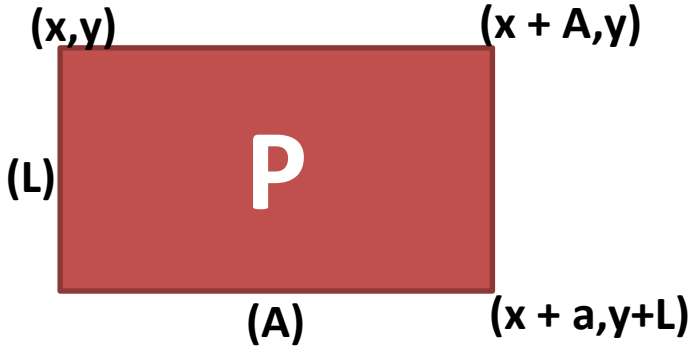
- ✓ Algoritmo Bounding Box

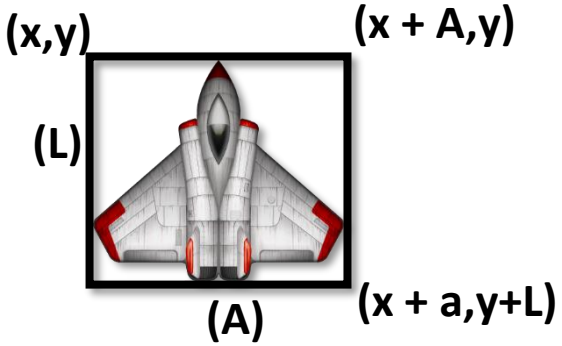
Este algoritmo es bastante sencillo. Busca, mediante comprobaciones, si dos rectángulos están cerca a partir de las esquinas que lo conforman. Por ejemplo:

- Sean O y P dos objetos cuyos atributos son X,Y, A y L.





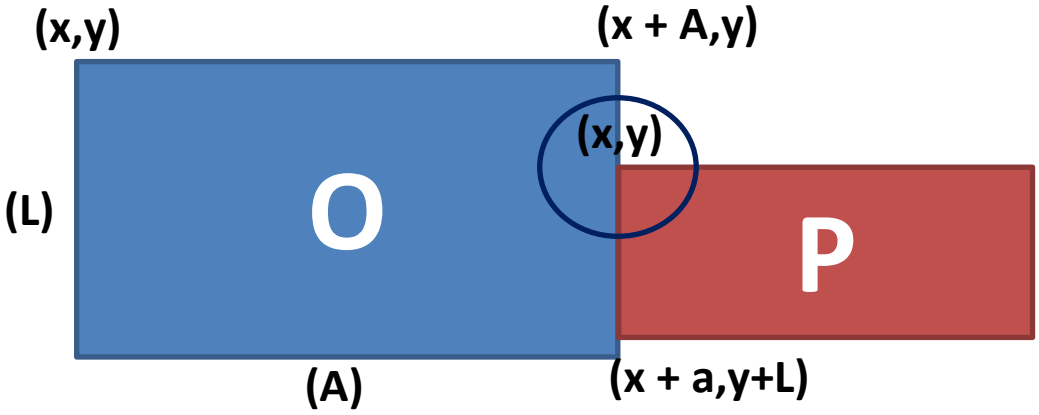


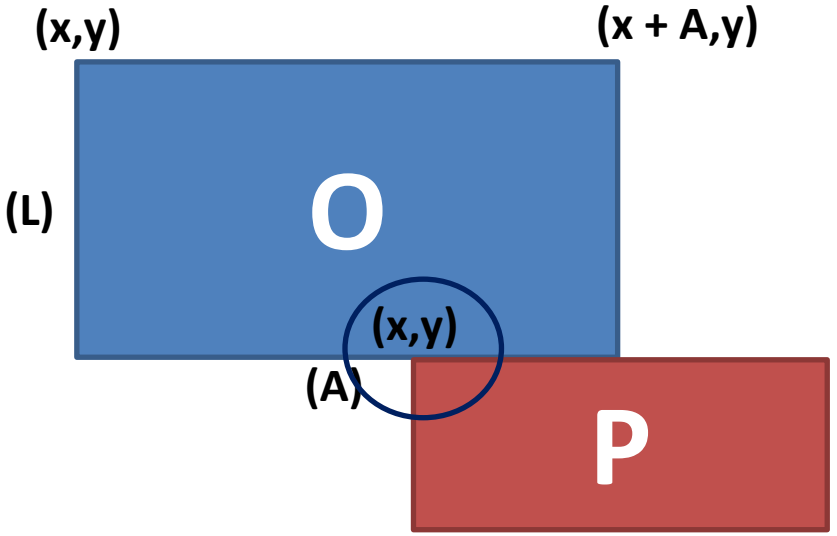


Colisiones

- ✓ Algoritmo Bounding Box

Se realiza cuando sucede lo siguiente





Colisiones

- ✓ Algoritmo Bounding Box

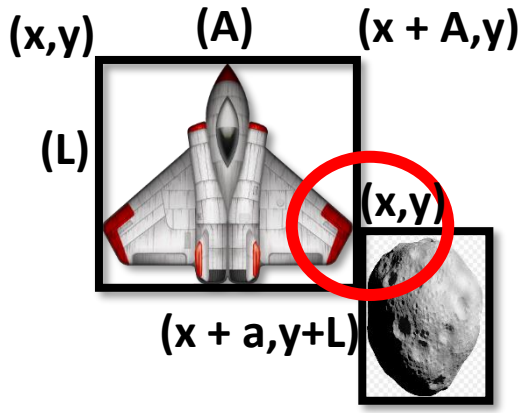
En código:

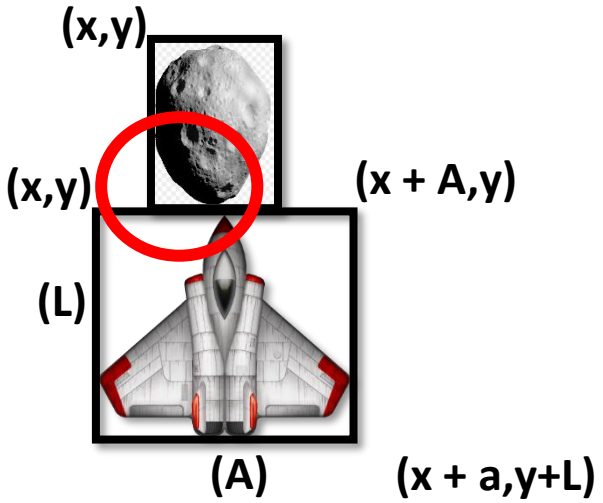
```
bool Clase_Colisiones::ColisionUsando_BoundingBox(int obj1X, int obj1Y, int obj1A,
                                                  int obj1L , int obj2X, int obj2Y, int obj2A, int obj2L)
{
        return (obj1X + obj1A > obj2X &&  obj1X < obj2X + obj2A &&  obj1Y + obj1L > obj2Y
                                                                  && obj1Y< obj2Y + obj2L);
}
```

Colisiones

- ✓ Algoritmo Bounding Box

Se realiza cuando sucede lo siguiente





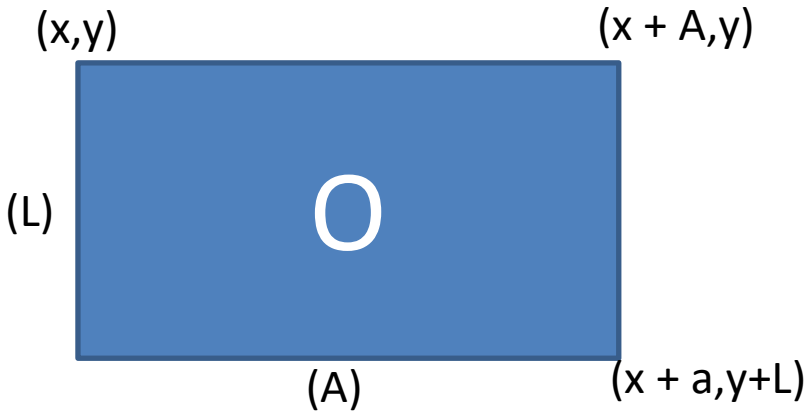
Colisiones

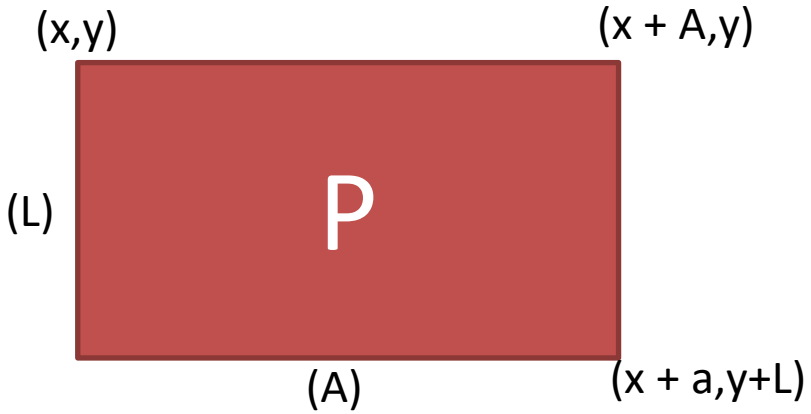
- ✓ Intersect

La función que nos permite obtener la región rectangular formada por la intersección.

Si la función retorna un valor falso es que los objetos no colisionaron y la región rectangular es nula (NULL)

Sean O y P dos objetos cuyos atributos son X,Y, A y L.





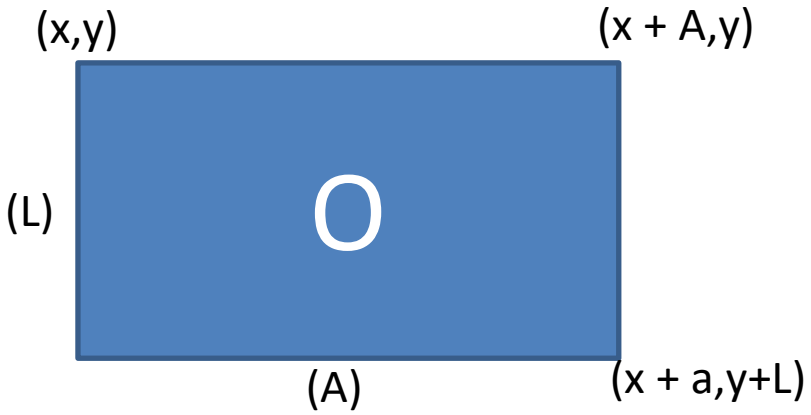
Colisiones

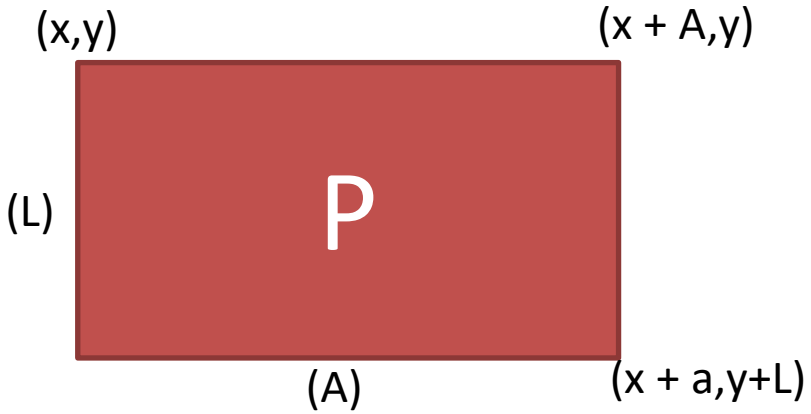
- ✓ Intersect

La función que nos permite obtener la región rectangular formada por la intersección.

Si la función retorna un valor falso es que los objetos no colisionaron y la región rectangular es nula (NULL)

Sean O y P dos objetos cuyos atributos son X,Y, A y L.

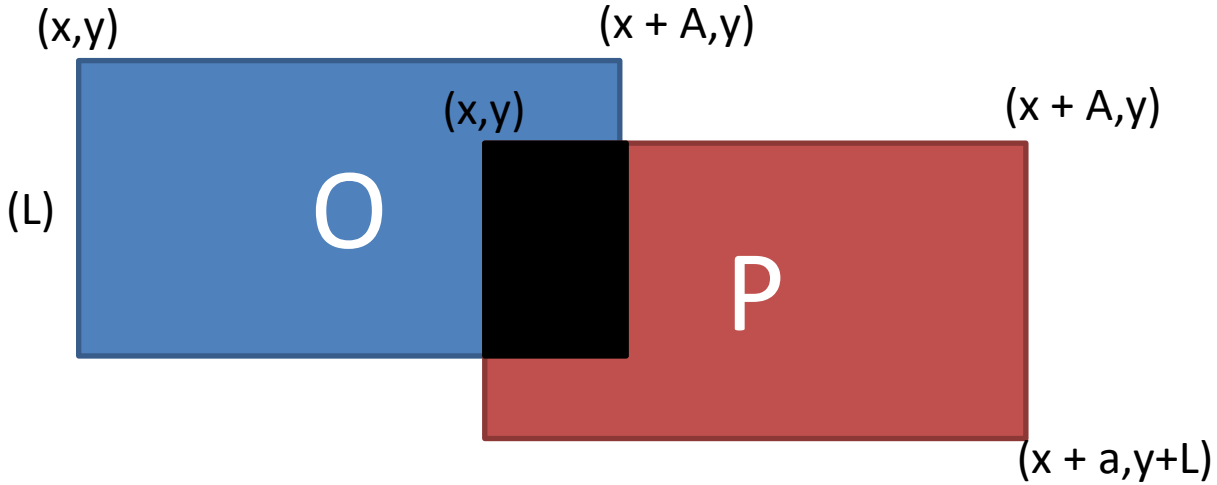




Colisiones

- ✓ Intersect

Cuando ambos objetos colisionan se forma una región rectangular de la siguiente manera



Colisiones

- ✓ Intersect

  En código sería

```
bool Clase_Colisiones::ColisionUsando_InsertectsWith(int obj1X, int obj1Y, int obj1A, int
obj1L ,    int obj2X, int obj2Y, int  obj2A, int obj2L)
{
    System::Drawing::Rectangle rectObj1 = System::Drawing::Rectangle(obj1X, obj1Y, obj1A, obj1L );
    System::Drawing::Rectangle rectObj2 = System::Drawing::Rectangle(obj2X, obj2Y, obj2A, obj2L );
    return rectObj1.IntersectsWith(rectObj2);
}
```

Colisiones

- ✓ Distancia entre dos puntos

La Coordenada del centro de la figura es : (x + A/2, y+L/2)

Colisiones

- ✓ Distancia entre dos puntos

  En código sería

```
bool Clase_Colisiones::ColisionDistancia(int obj1X, int obj1Y, int obj1A, int obj1L , int
                                                    obj2X, int obj2Y, int obj2A, int obj2L)
{
    float px= pow((obj1X+ obj1A/2) - (obj2X+ obj2A/2) , 2.0);
    float py= pow((obj1Y+ obj1L/2) - (obj2Y+ obj2L/2) , 2.0);
    return  sqrt ( px + py)<1.0;
}
```