Colisiones 2D en los videojuegos

Francisco José Gallego Durán
Sergio Pradas Rodríguez

Colisiones 2D en los videojuegos

- La base de la detección de colisiones es hacer que cuando los diferentes objetos que actúan en la pantalla se juntan lo suficiente entre sí parezca que están chocando y puedan reaccionar de alguna manera
- Necesidad de poca complejidad algorítmica
- Efectos “realistas”, no “reales”
- Comprobación de las colisiones antes de realizar el movimiento
- Gran diversidad de métodos

Colisiones 2D en los videojuegos

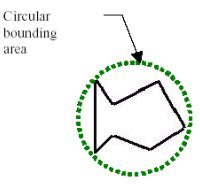
- *Métodos más usados*

- Máscaras de bits aplicadas a los sprites
- *Bounding Boxes*
- Particionamiento del espacio en zonas ocupadas y desocupadas

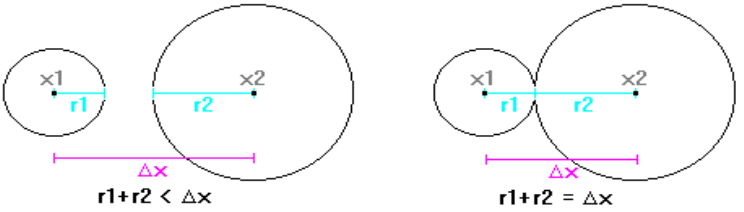
Colisiones 2D en los videojuegos

- *Bounding Boxes*

- Uso de círculos para cubrir el objeto
- El círculo representa las dimensiones del objeto en el algoritmo de detección
- Sencillo de calcular y de poco coste

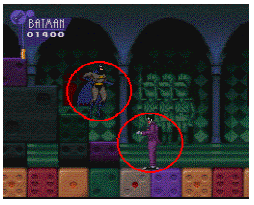


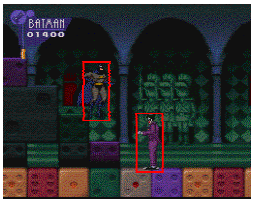
Colisiones 2D en los videojuegos



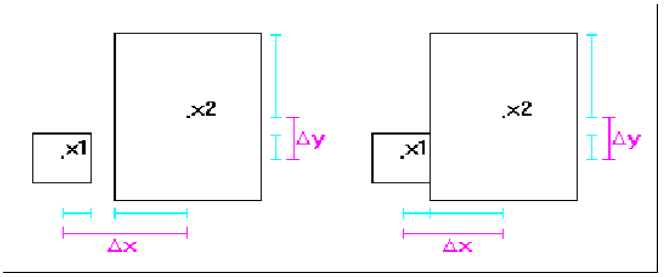
Colisiones 2D en los videojuegos

- Los círculos puedes no ser adecuados según la forma del objeto
- Uso de rectángulos pare representar formas humanas o cuya anchura difiera de la altura





Colisiones 2D en los videojuegos



Colisiones 2D en los videojuegos

- A tener en cuenta el coste del cálculo de la distancia entre los puntos. La raíz cuadrada tiene un coste demasiado elevado.

$$\text{distancia} = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

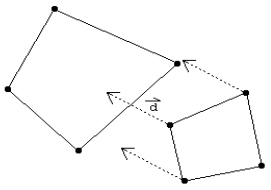
- Algoritmo basado en la Serie de Maclaurin : cálculo de distancias entre dos puntos mediante derivadas y aproximaciones matemáticas, el resultado no es totalmente exacto, sin embargo desde el punto de vista de un videojuego la aproximación es bastante buena, tiene una precisión del 97%

Colisiones 2D en los videojuegos

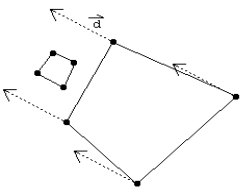
```
int distancia( int x1, int y1, int x2, int y2)
{
  int x = abs( x2 - x1 );
  int y = abs( y2 - y1 );

  int min = x < y? x : y;

  return ( x+y - (min>>1) - (min>>2) + (min>>4) );
}
```

Colisiones 2D en los videojuegos

- Otra forma es comprobar las intersecciones de los vectores que indican la dirección y el sentido del movimiento de los objetos con los lados de los objetos que pueden colisionar.



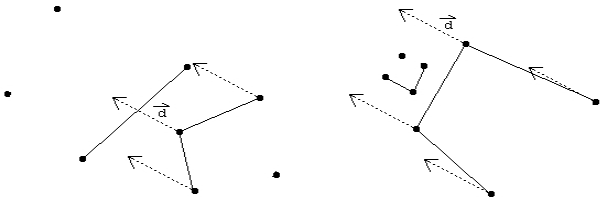
Average sized blocks collide given a displacement vector $\vec{d}$.



Large block collides with a small block given a displacement vector $\vec{d}$.

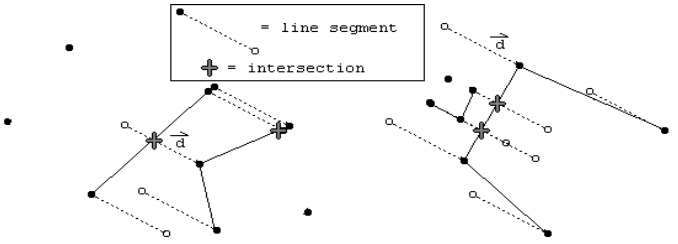
Colisiones 2D en los videojuegos

- Consiste en trazar dicho vector desde cada uno de los vértices de los lados que pueden colisionar con el objeto implicado, comprobando si intersectan con alguno de los lados de dicho objeto.



Cull the sides not involved in the collision using the displacement vector.

Colisiones 2D en los videojuegos



Produce the line segments using the displacement vector and then locate the points of collision.

Colisiones 2D en los videojuegos

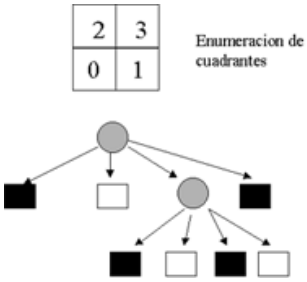
- Particionamiento del espacio

- Descomposición en celdas en la cual un sólido es descompuesto en celdas idénticas acomodadas en una malla uniforme.
- Estas celdas son llamadas *voxels* (elementos de volumen) como analogía a los *píxels*. La forma de celda mas común es el cubo y la representación del espacio como un arreglo de cubos se llama *cuberille*.
- Controlamos solamente la presencia o ausencia de una celda dentro de la malla, en este sentido solamente debemos determinar qué celdas se encuentran ocupadas y cuáles no. De esta manera podemos codificar un objeto como un lista única y sin ambigüedades de celdas ocupadas. Determinar si una celda se encuentra fuera o dentro de un sólido, y es igualmente sencillo de ver si dos objetos son adyacentes.

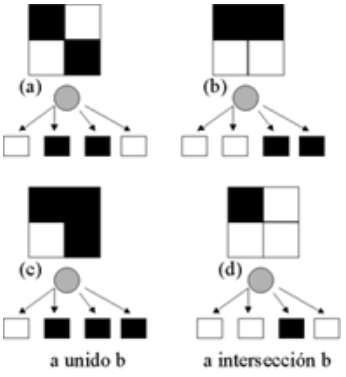
Colisiones 2D en los videojuegos

- *Quadtrees* : formato de representación 2D usado para codificar imágenes.
- La idea fundamental detrás los quadtrees es divide y vencerás. Un quadtree es derivado subdividiendo sucesivamente el plano 2D en ambas direcciones para obtener cuadrantes.
- Cuando un quadtree es usado para representrar un área en un plano , cada cuadrante puede estar lleno, parcialmente lleno o completamente vacío, dependiendo que tanto del área intersecta el cuadrante. Un cuadrante parcialmente lleno es recursivamente subdividido hasta obtener solamente cuadrantes homogéneos: completamente llenos o completamente vacíos , o hasta llegar a una altura predefinida.

Colisiones 2D en los videojuegos



Enumeracion de los cuadrantes

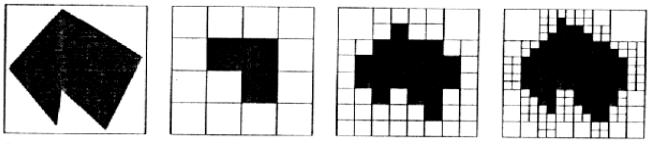


a unido b

a intersección b

Colisiones 2D en los videojuegos

- Un nodo de un *quadtree* puede tener ocho vecinos, cuatro que comparten un vértice: norte, sur este y oeste, y cuatro que comparten un lado: nor-este, nor-oeste, sur-este y sur-oeste.
- Una manera para encontrar el vecino en una dirección específica : se comienza con el nodo original en el árbol hasta encontrar el primer ancestro en común entre el nodo original y el nodo vecino. Luego comienza a bajar hasta encontrar el nodo vecino.



Colisiones 2D en los videojuegos

- Los árboles BSP dividen el espacio recursivamente en pares de subespacios, separados por un plano de orientación y posición arbitraria.

- Cada nodo interno del árbol BSP esta asociado a un plano y tiene dos punteros a los hijos, uno para cada lado del plano, asumiendo que las normales se encuentran apuntando fuera del objeto, el hijo izquierdo esta dentro o detrás del plano y el derecho se encuentra afuera o delante del plano.