Anwendung von Kollisionsabfragen

## Einfache Kollision

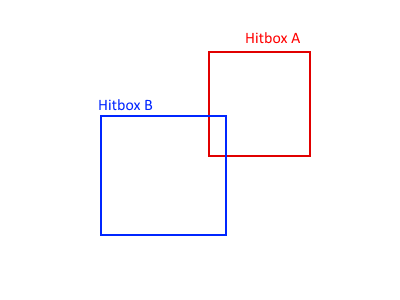
In SFML gibt es bereits eingebaute Hitboxen für jedes Shape. Auf diese können wir mit **Shape.getGlobalBounds()** zurückgreifen.

Abbildung : Zwei Hitboxen überlappen sich

Um zu prüfen, ob sich 2 Hitboxen überschneiden nutzen wir die "**Intersects(shape2.getGlobalBounds())**" Methode der Hitbox. Hier ein Beispiel:

if (BoxA.GetGlobalBounds().Intersects( BoxB.GetGlobalBounds() ))

{

// Wenn die Hitboxen sich überlappen

}

else

{

// Wenn sie sich nicht überlappen

}

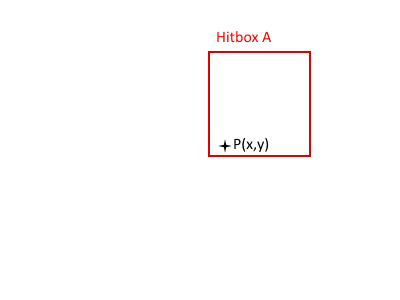
Jetzt können wir programmieren, was passieren soll, wenn sich die beiden Hitboxen überlappen.

Abbildung : Hitbox ist über Punkt(x,y)

Alternativ können wir auch prüfen, ob sich ein Punkt innerhalb einer Hitbox befindet, dazu nutzen wir die "**Contains(x,y)**" Methode.

Beispiel:

if (BoxA.GetGlobalBounds().Contains(200, 300))

{

// Hitbox befindet sich über Punkt 200,300

}

else {

// Hitbox befindet sich nicht über Punkt 200,300

}

## Bewegungskontrolle

Eine weitere Möglichkeit Hitboxen zu benutzen ist, die Bewegung eines Objektes einzuschränken, z.b. durch eine Wand, die ein weiteres Objekt darstellt.

RectangleShape Player;

RectangleShape Wall;

Player = new RectangleShape();

Player.Size = new Vector2f(50, 50);

Player.Position = new Vector2f(200, 400);

Player.FillColor = Color.Green;

Player.OutlineColor = Color.Red;

Player.OutlineThickness = 1;

Wall = new RectangleShape();

Wall.Size = new Vector2f(10, 400);

Wall.Position = new Vector2f(400, 200);

Wall.FillColor = Color.Blue;

Wall.OutlineColor = Color.Red;

Wall.OutlineThickness = 1;

Wenn wir nun unseren Player wie gewohnt mit **isKeyPressed** bewegen:

if (Keyboard.IsKeyPressed(Keyboard.Key.D))

{

Player.Position += new Vector2f(3, 0);

}

Dann kann er einfach durch unser "Wall" Objekt. Wir könnten jetzt auf Kollision prüfen:

if (Player.GetGlobalBounds().Intersects( Wall.GetGlobalBounds() ))

{

}

else

{

if (Keyboard.IsKeyPressed(Keyboard.Key.D))

{

Player.Position += new Vector2f(3, 0);

}

}

Und so nur Bewegung zulassen, wenn sie sich nicht überschneiden.

Leider führt das dazu, dass sobald eine Kollision ensteht, unser Shape am Wall "festklebt". Deswegen müssen wir prüfen ob eine Kollision nach der Bewegung entstanden ist, und wenn ja, die Bewegung rückgängig machen. Man beachte hierbei das "-=" !

if (Keyboard.IsKeyPressed(Keyboard.Key.D))

{

// Bewege Player

Player.Position += new Vector2f(3, 0);

if (Player.GetGlobalBounds().Intersects(Wall.GetGlobalBounds()))

{

// Kollision, bewege Player zurück

Player.Position -= new Vector2f(3, 0);

}

}

Jetzt wird eine ungültige Bewegung einfach rückgängig gemacht.

Jetzt kann der Player den "Wall" nichtmehr passieren.

Allerdings gibt es noch Sonderfälle zu beachten. Wenn die Geschwindigkeit des Players sehr hoch wird, dann kann es sein, dass er den "Wall" einfach überspringt und keine Kollision auslöst. Diese Fehler sind oft schwer zu verhindern. Es gäbe für uns zwei Möglichkeiten:

* Maximal Geschwindigkeit und mindest "Wanddicke" für Objekte(begrenzt die Möglichkeiten für das Spiel)
* Die Kollisionsabfrage erweitern, so dass die Fehler nichtmehr auftreten (kostet mehr Leistung als normale Abfrage)