

Rigid Body Dynamics

Collision generation algorithms

Tobias Scharpff

Gliederung

- Begriffe
- Allgemeine Überlegungen
- Kugel - Kugel Kollision
- Kugel - Ebene Kollision
- Kugel - Box Kollision
- Box - Ebene Kollision
- Box - Box Kollision

Begriffe

- collision detection vs. contact generation
- contact data
 - collision point
 - collision normal
 - penetration depth
- cases of contact
 - Überblick
 - point-face
 - edge-edge
 - face-face
 - edge-face
 - point-edge
 - point-point

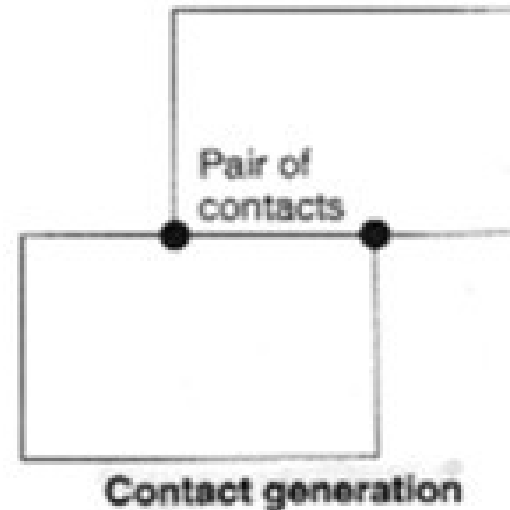
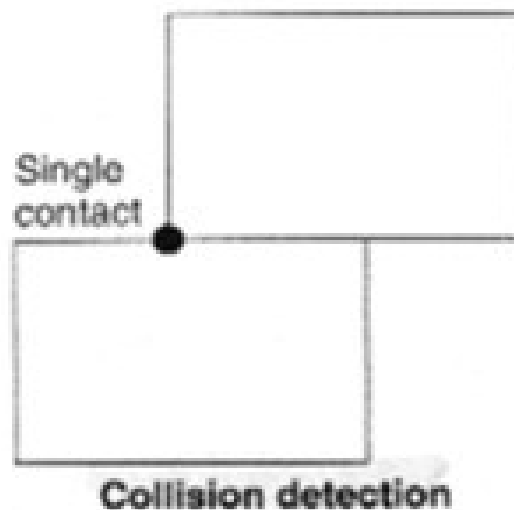
collision detection vs. contact generation

- collision detection - Kollisionserkennung:
 - Überprüft, ob sich zwei Objekte berühren oder überschneiden
 - Liefert häufig den Kontaktpunkt oder den Punkt des tiefsten Eindringens
- contact generation – Kontaktpunktgenerierung
 - Liefert die Menge aller Kontakt- und Überschneidungspunkte

collision detection vs. contact generation

Beispiel:

Zwei Boxen liegen aufeinander, sind aber minimal zueinander verdreht. Links wird nur der Punkt des tiefsten Eindringens erkannt, rechts alle Punkte, die den Kontakt herstellen.

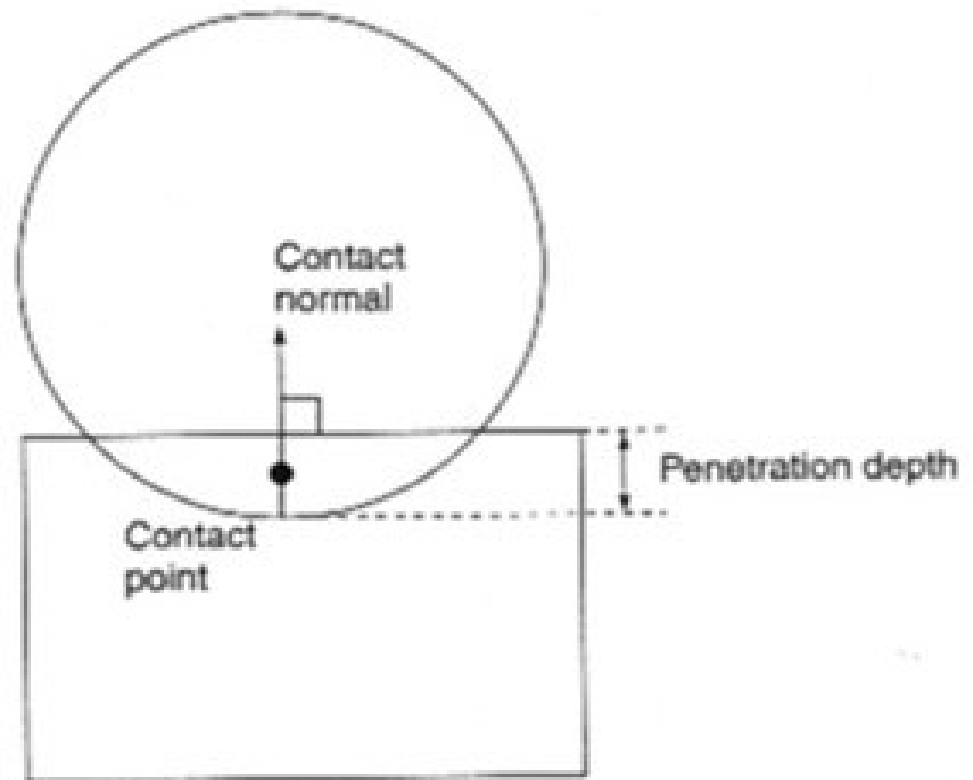


contact data - Kontaktdaten

- collision point - Kollisionspunkt
 - Kontaktpunkt zwischen den beiden Körpern (selten)
 - Bei Überschneidung wird häufig ein Punkt auf der Oberfläche eines der beiden Körper gewählt, oder ein Punkt auf dem Normalenvektor z.B. Mittelpunkt
- collision normal - Kollisionsnormalenvektor
 - Vektor, in dessen Richtung der Impuls zum Trennen der Objekte wirken soll
- penetration depth – Eindringtiefe
 - Bei überlappenden Objekten wird hier angegeben, wie tief der eine Körper in den anderen eindringt

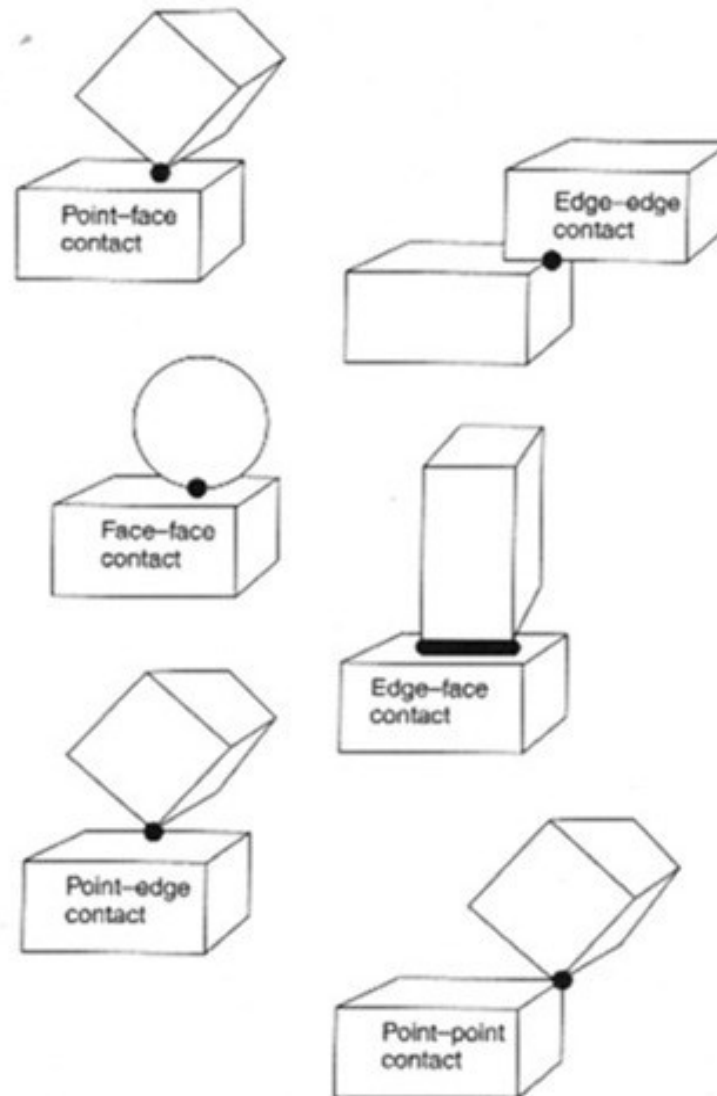
contact data - Kontaktdaten

```
class Contact {  
    /* Kollisionspunkt in Weltkoordinaten*/  
    Vector3 contactPoint;  
  
    /* Kollisionsnormalenvektor in Weltkoordinaten*/  
    Vector3 contactNormal;  
  
    /* Eindringtiefe am Kollisionspunkt*/  
    real penetration;  
};
```



cases of contact - Kontaktfälle

- Überblick

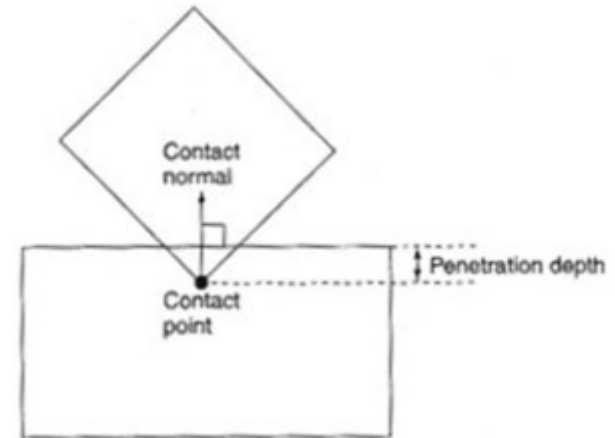


cases of contact - Kontaktfälle

- point-face contacts - Punkt-Fläche Kontakt

- Häufigster und wichtigster Fall

- Der Normalenvektor der Fläche am Punkt des Kontakts wird als Normalenvektor gespeichert

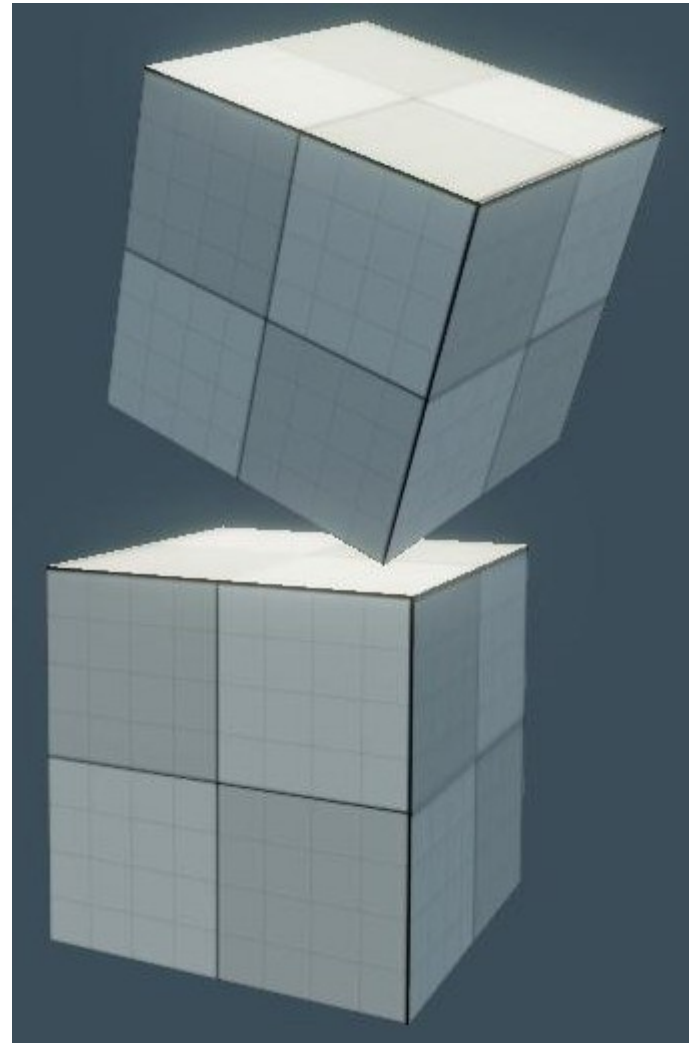
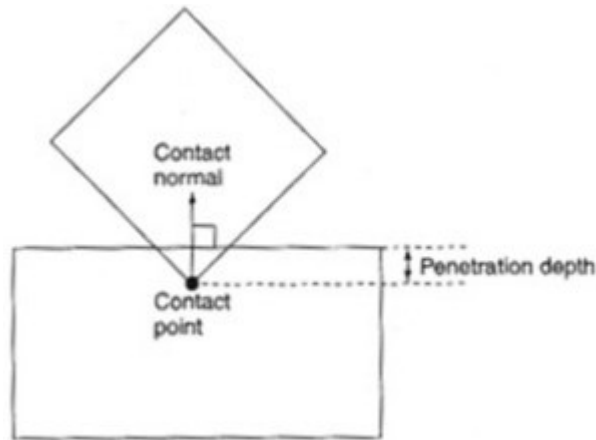


- Der Kontaktpunkt ist normalerweise der Punkt, der die Oberfläche berührt oder am tiefsten in sie eingedrungen ist. Manchmal auch der Punkt, auf halber Strecke zwischen der Oberfläche und dem tiefsten Punkt

- Eindringtiefe ist der Abstand des am tiefsten eingedrungenen Punktes zur Oberfläche

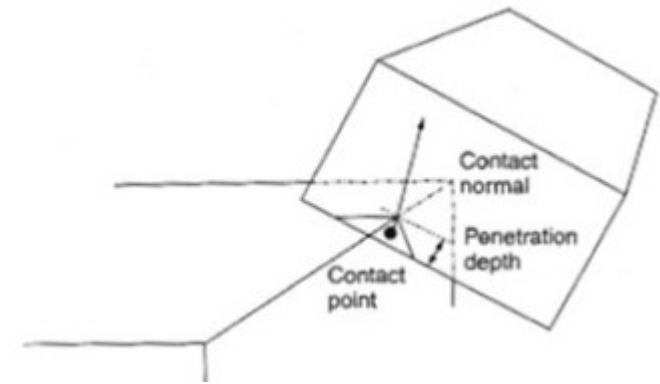
cases of contact - Kontaktfälle

- point-face contacts - Punkt-Fläche Kontakt



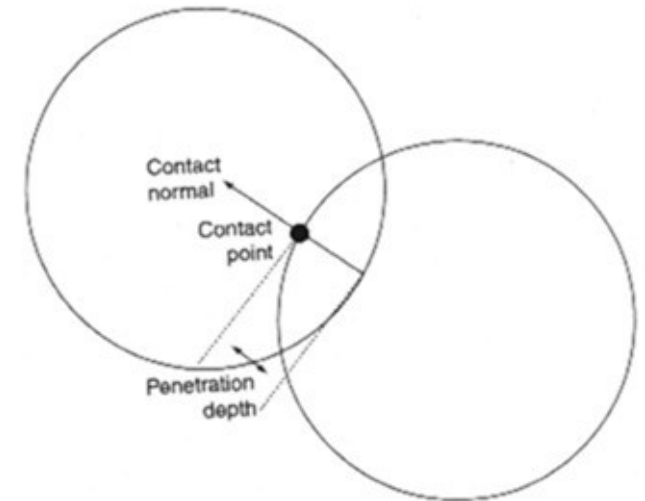
cases of contact - Kontaktfälle

- edge-edge contacts - Kante-Kante Kontakt
 - Zweitwichtigster Fall
 - Normalenvektor steht senkrecht auf den beiden Tangenten der Kanten
 - Als Kontaktpunkt wird häufig der Punkt auf einer der beiden Kanten gewählt der am nächsten zur anderen ist, oder der Punkte genau zwischen den Kanten
 - Eindringtiefe ist der Abstand der beiden Kanten



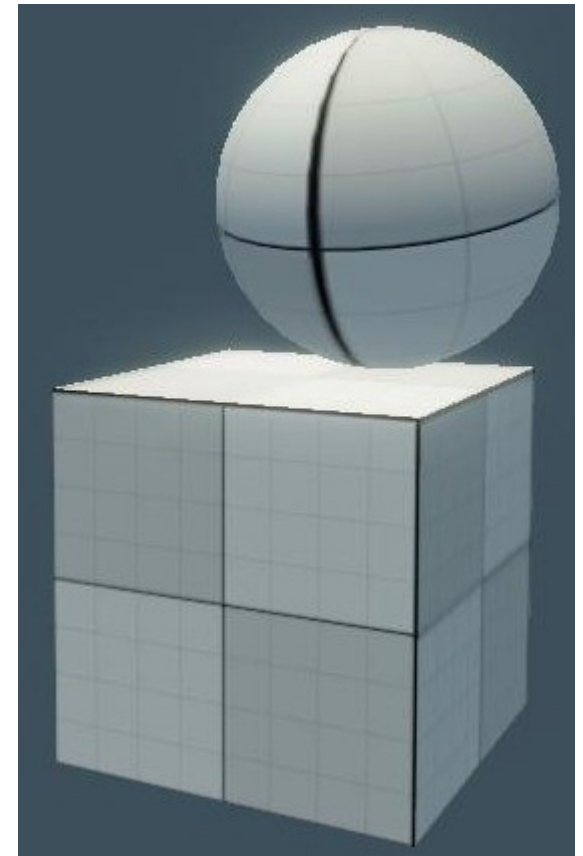
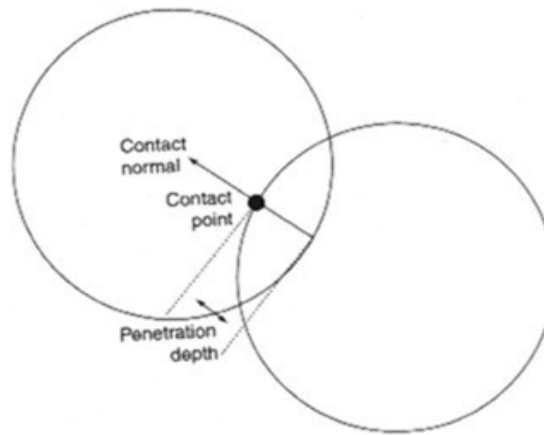
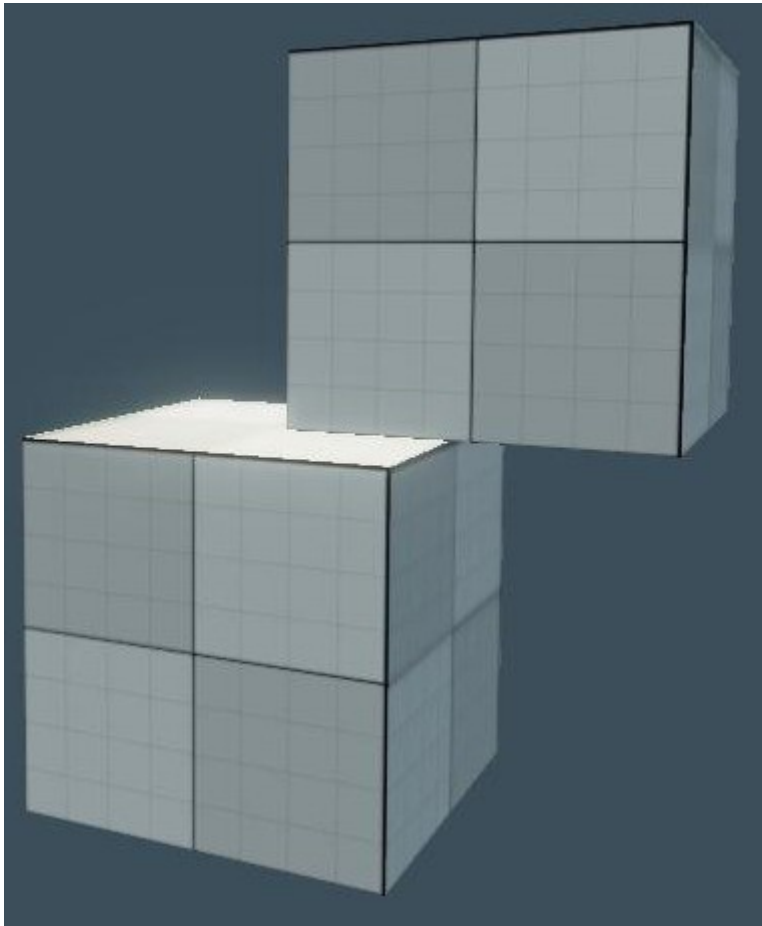
cases of contact - Kontaktfälle

- face-face contacts - Fläche-Fläche Kontakt
 - Wird meist nur bei gekrümmten Oberflächen betrachtet
 - Es wird die Normale einer der beiden Oberflächen als Normalenvektor gewählt
 - Als Kontaktpunkt wird meist der am tiefsten eingedrungene Punkt gewählt
 - Eindringtiefe ist der Abstand des am tiefsten eingedrungenen Punktes zur Oberfläche



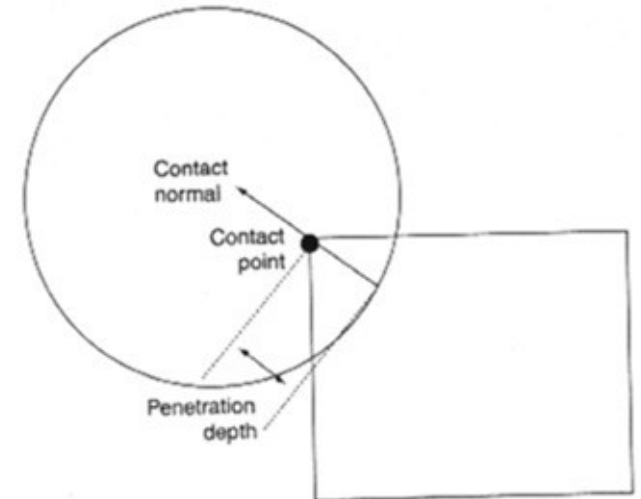
cases of contact - Kontaktfälle

- face-face contacts - Fläche-Fläche Kontakt



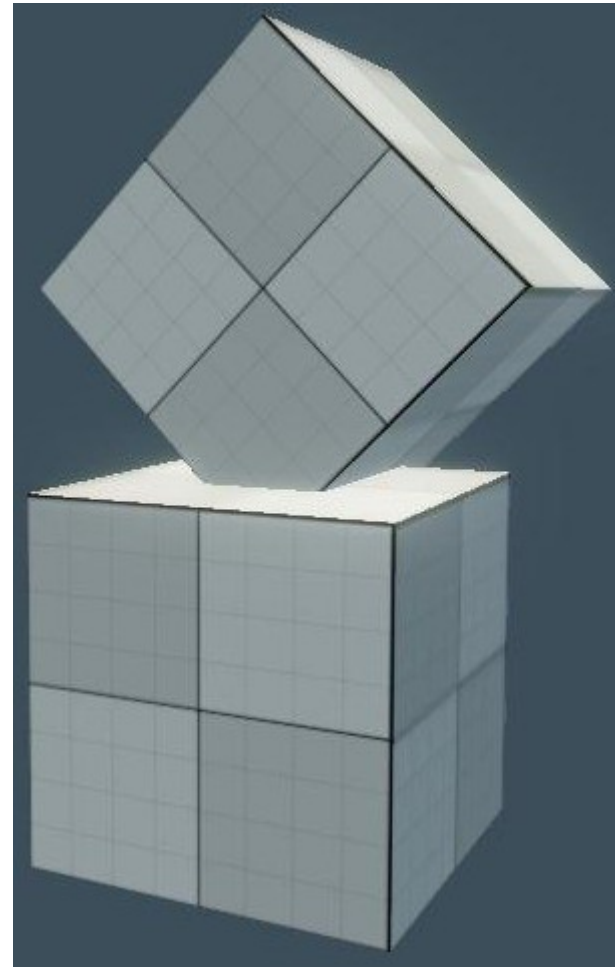
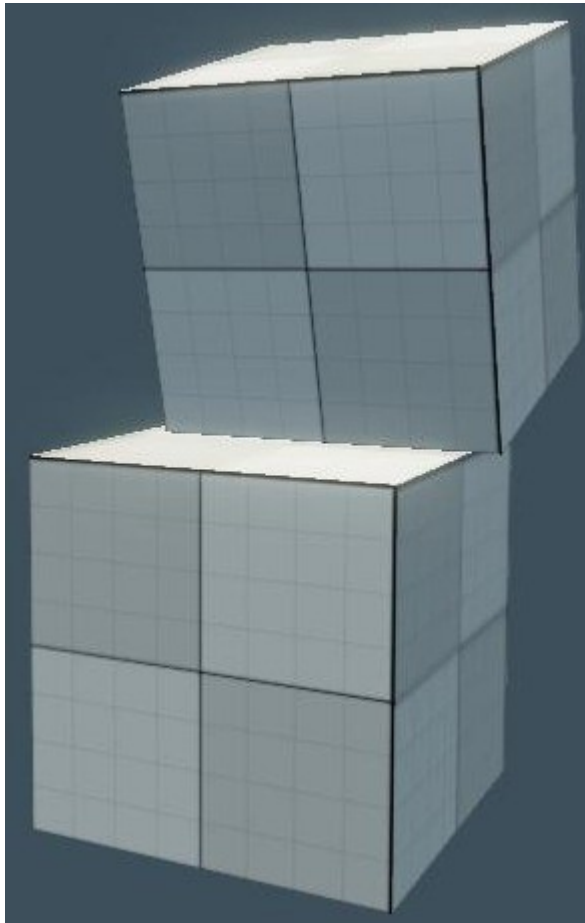
cases of contact - Kontaktfälle

- edge-face contacts - Kante-Fläche Kontakt
 - Ähnlich Punkt-Fläche Kontakt
 - Normale der Oberfläche wird als Normalenvektor benutzt
 - Kontaktpunkt ist wieder der am tiefsten eingedrungene Punkt
 - Eindringtiefe lässt sich als Abstand zwischen Kontaktpunkt und Oberfläche berechnen



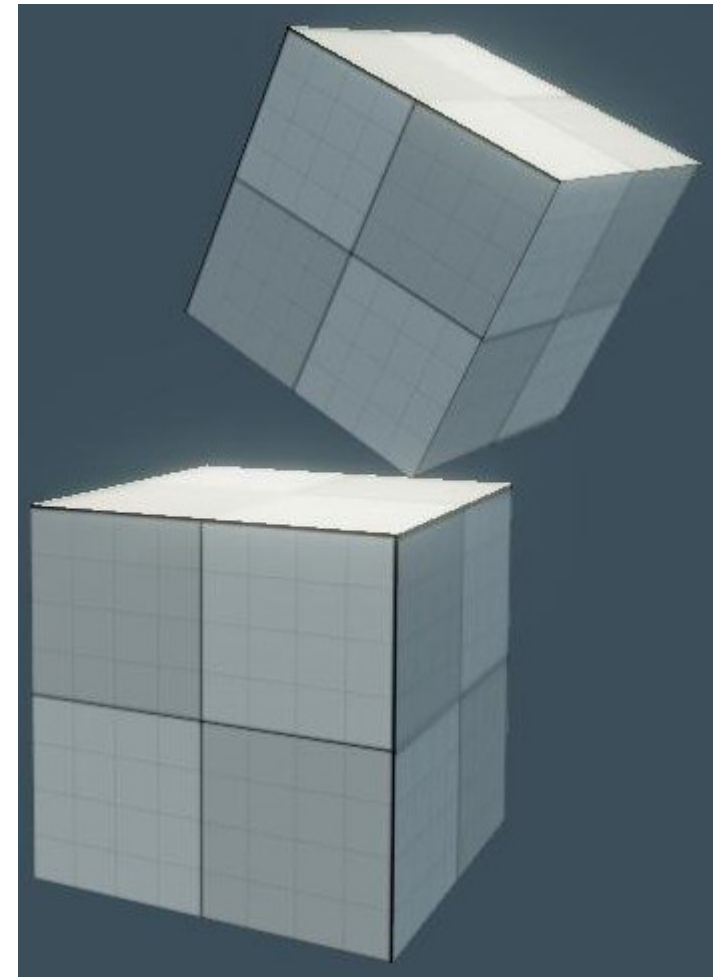
cases of contact - Kontaktfälle

- edge-face contacts - Kante-Fläche Kontakt



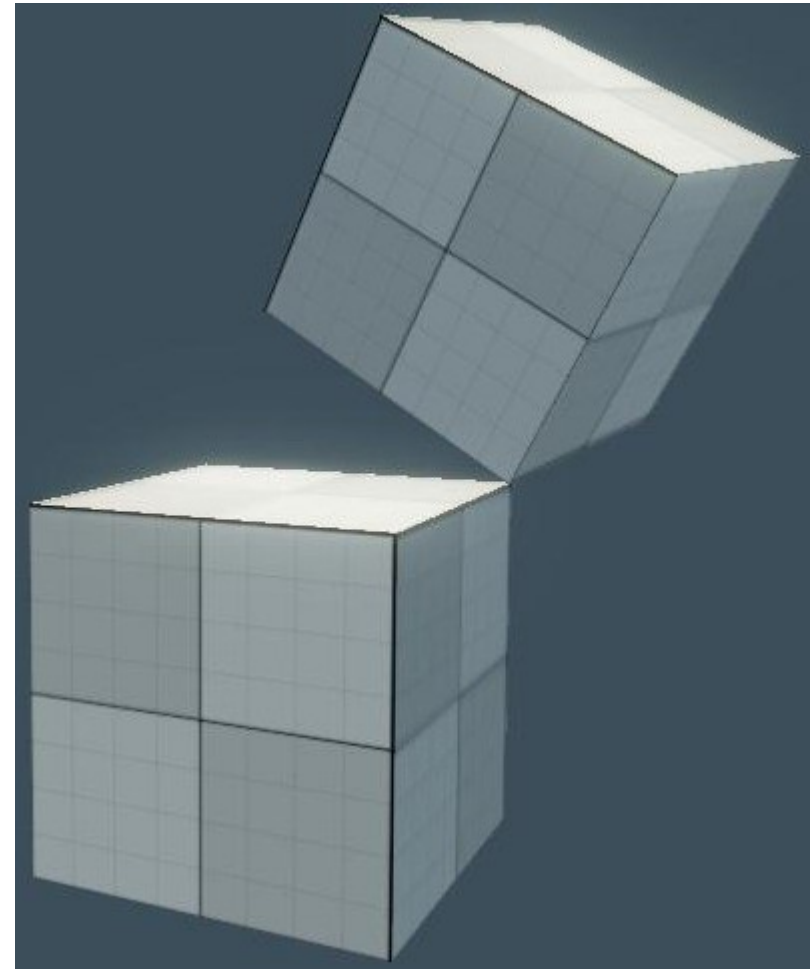
cases of contact - Kontaktfälle

- point-edge contacts - Punkt-Kante Kontakt
 - Sehr selten und wird deshalb meist ignoriert
 - Im nächsten Berechnungsschritt häufig als Punkt-Oberfläche oder Kante-Kante Kontakt erkannt



cases of contact - Kontaktfälle

- point-point contacts - Punkt-Punkt Kontakt
 - Passiert fast nie und kann deshalb auch ignoriert werden
 - Im nächsten Berechnungsschritt nicht mehr vorhanden oder dann einer der anderen Fälle



Allgemeine Überlegungen

- Naiver Ansatz
- Optimierung
 - Kontakterkennung und Punktberechnung kombinieren
 - early outs
 - Reihenfolge der Kontaktfälle
- Allgemeiner Code

Naiver Ansatz

- Überprüfe, ob die Objekte sich berühren
- Berechne die Kontaktpunkte

```
if (inContact()) {  
    findContacts();  
}
```

- Nicht performant, da oft Teilergebnisse aus `inContact()` in `findContacts()` wieder berechnet werden

Optimierung

- Nur eine Funktion die entscheidet, ob die Körper sich berühren und berechnet wo die Kontaktpunkte sind
- early outs - Frühes Ausscheiden
 - Überprüfen, ob die Objekte sich berühren und falls sicher ist, dass sie das nicht tun, sofort abbrechen
- Punkt-Fläche und Kante-Kante Kontakt zuerst überprüfen da sie sehr häufig sind

Allgemeiner Code

```
/* Hilfsstruktur die dem Detektor Informationen zum Berechnen der Kontaktpunkte zur
Verfügung stellt */
struct CollisionData {
    /* Array in das die Kontaktpunkte eingetragen werden */
    Contact* contacts;

    /* Maximale Anzahl der im Array speicherbaren Kontaktpunkte */
    unsigned contactsLeft;
}

/* Allgemeine Kontaktberechnungsfunktion */
void detectContacts( const Primitive &firstPrimitive,
                    const Primitive &secondPrimitive,
                    CollisionData *data);

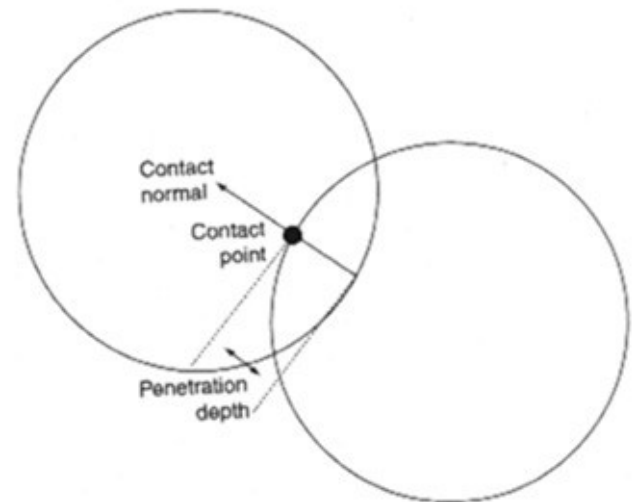
/* Nähere Beschreibung der kollidierenden Objekte */
class Primitive {
    RidgidBody *body;
    Matrix offset;
}
```

Kugel - Kugel Kollision

- Idee

- Zwei Kugeln berühren oder überschneiden sich wenn der Abstand ihrer Mittelpunkte gleich oder kleiner als die Summe ihrer Radien ist
- Fläche-Fläche Kontakt mit Kontaktpunkt genau zwischen den beiden Mittelpunkten

```
class Sphere : public Primitive {  
    public: /* Mittelpunkt ist offset in Primitive */  
        real radius;  
}
```



Kugel - Kugel Kollision

```
unsigned CollisionDetection::SphereAndSphere( const Sphere &one,
                                              const Sphere &two,
                                              CollisionData *data) {

    //Sicherstellen, dass wir noch Platz für die Kollision haben
    if (data->contactsLeft <= 0) return 0;

    //Mittelpunkte zwischenspeichern
    Vector3 positionOne = one.GetAxis(3);
    Vector3 positionTwo = two.GetAxis(3);

    //Vektor zwischen den Mittelpunkten und dessen Länge berechnen
    Vector3 midline = positionOne - positionTwo;
    real size = midline.magnitude();

    //early out
    if (size <= 0 || size >= one.radius+two.radius) return 0;

    //Berechnen des Normalenvektors
    Vector3 normal = midline*(((real)1.0)/size);
```

Kugel - Kugel Kollision

```
//Kontaktdaten eintragen
Contact* contact = data->contacts;
contact->contactNormal = normal;
contact->contactPoint = positionOne + midline * (real)0.5;
contact->penetration = one.radius + two.radius - size;

//Weitere Daten eintragen
contact->body[0] = one.body;
contact->body[1] = two.body;
contact->restitution = data->restitution; //Elastizität
contact->friction = data->friction; //Reibung

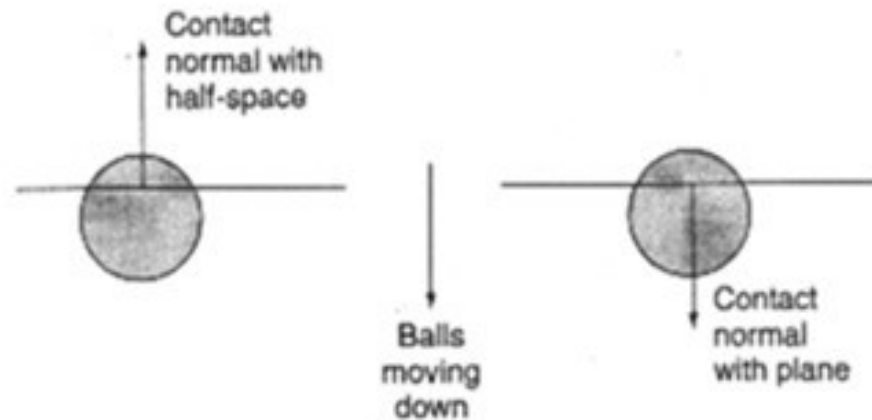
return 1;
}
```

Kugel - Ebene Kollision

- Idee

- Eine Kugel kollidiert genau dann mit einer Ebene, wenn der Abstand des Mittelpunkts zur Ebene kleiner wird als der Radius der Kugel
- Fläche-Fläche Kontakt mit Kontaktpunkt auf der Ebene, als erster Punkt des Kontakts

```
class Plane : public Primitive {  
    public:  
        Vector3 normal;  
        real offset;  
}
```



Kugel - Ebene Kollision

- Abstandsberechnung

- Abstand d , Mittelpunkt \vec{p} , Ebenennormale \vec{n} , Ebenenoffset l , Radius r

$$d = \vec{p} * \vec{n} - l - r$$

- Codeschnipsel für Halbebene

```
//Ermittle den Mittelpunkt der Kugel  
Vector3 position = sphere.GetAxis(3);
```

```
//Abstand zur Ebene berechnen  
real ballDistance = plane.direction * position – sphere.radius – plane.offset;
```

```
if (ballDistance >= 0) return 0;
```

```
...
```

```
contactPoint = position – plane.direction * (ballDistance + sphere.radius);
```

Kugel - Ebene Kollision

- Codeschnipsel für Ebene

```
//Ermittle den Mittelpunkt der Kugel  
Vector3 position = sphere.GetAxis(3);
```

```
//Abstand des Mittelpunkts zur Ebene berechnen  
real centerDistance = plane.direction * position - plane.offset;
```

```
//Überprüfen, ob die Kugel kollidiert  
if (centerDistance * centerDistance > sphere.radius * sphere.radius) return 0;
```

```
//Überprüfen, auf welcher Seite der Ebene die Kugel ist  
Vector3 normal = plane.direction;  
real penetration = -centerDistance;  
if (centerDistance < 0) {  
    normal *= -1;  
    penetration = -penetration;  
}
```

```
..
```

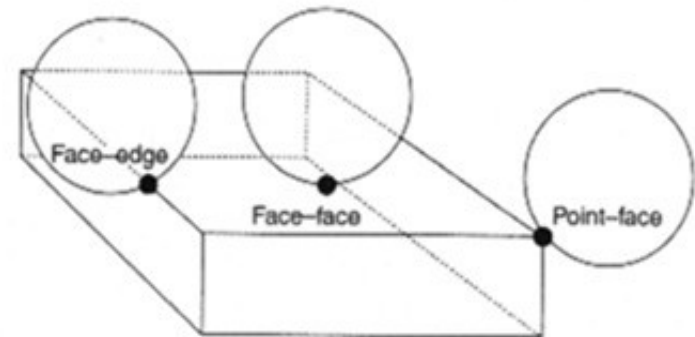
```
contactPoint = position - plane.direction * centerDistance;
```

Kugel - Box Kollision

- Idee

- Es gibt drei Möglichkeiten, wie eine Kugel mit einer Box kollidieren kann:

- mit einer Ecke
- mit einer Kante
- mit einer Seite



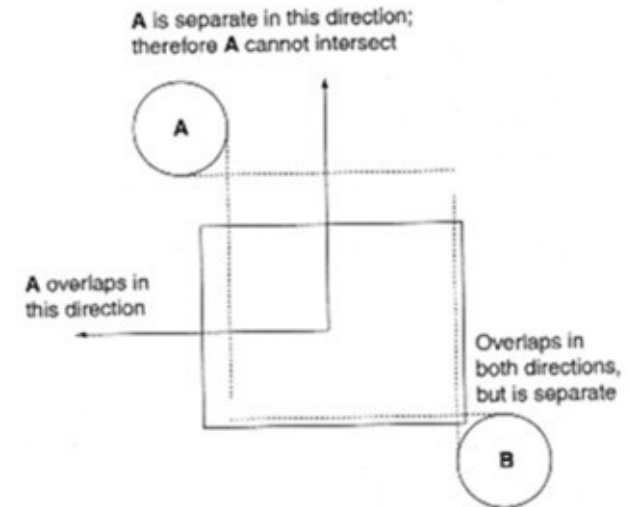
- Es muss der Punkt der Box gefunden werden, der am nächsten am Mittelpunkt der Kugel ist
- Mit diesem Punkt lassen sich die Kontaktdaten unabhängig vom Kollisionsfall berechnen

Kugel - Box Kollision

- Umsetzung

- early out mittels Achsenseparieren

- Berechne die relative Position des Kugelmittelpunkts zum Boxzentrum
 - Wenn der Abstand in einer Koordinatenrichtung abzüglich des Radius größer ist als die Ausdehnung der Box in dieser Koordinatenrichtung, haben wir keinen Kontakt

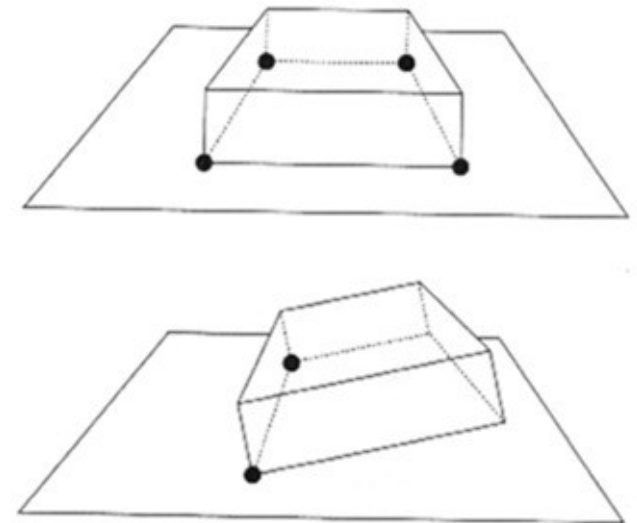


- Der nächste Punkt lässt sich leicht über die relative Position des Mittelpunkts bestimmen. Ist die Koordinate innerhalb des Ausdehnungsbereiches der Box so wird diese benutzt, sonst die maximale Ausdehnung

Box - Ebene Kollision

- Idee

- Es kann bis zu vier Punkt-Ebene Kollisionen geben
- Die Berechnung der Eckpunkte und deren eventuelle Kollision mit Daten geht sehr schnell
 - Als Normalenvektor wird der der Ebene benutzt
 - Als Kontaktpunkt der Eckpunkt
 - Die Eindringtiefe wird schon berechnet wenn man wissen will, ob eine Kollision vorliegt



Box - Ebene Kollision

- Anmerkung
 - Leicht parallelisierbar

- Codeschnipsel

```
//Für jeden Eckpunkt
```

```
//Berechne den Abstand der Ecke zur Ebene  
real vertexDistance = vertexPos * plane.direction;
```

```
//Überprüfe ob die Ecke ein Kontaktpunkt ist  
if (vertexDistance <= plane.offset + data->tolerance) {  
    ...  
    contact->contactPoint = vertexPos;  
    contact->contactNormal = plane.direction;  
    contact->penetration = plane.offset - vertexDistance;  
}
```

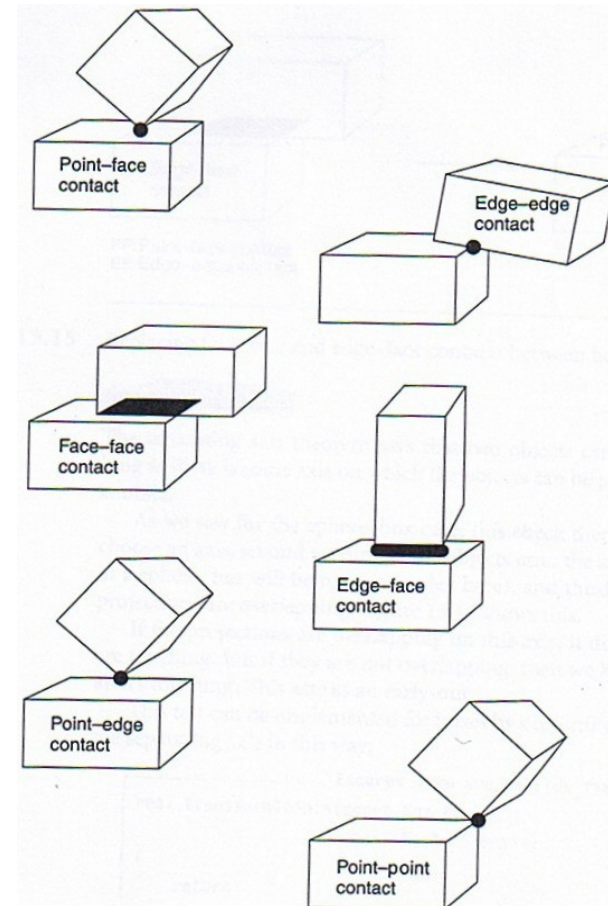
Box - Box Kollision

- Idee

- Es gibt sechs verschiedene Kontaktmöglichkeiten

- Punkt - Fläche
- Kante - Kante
- Fläche - Fläche
- Kante - Fläche
- Punkt - Kante
- Punkt - Punkt

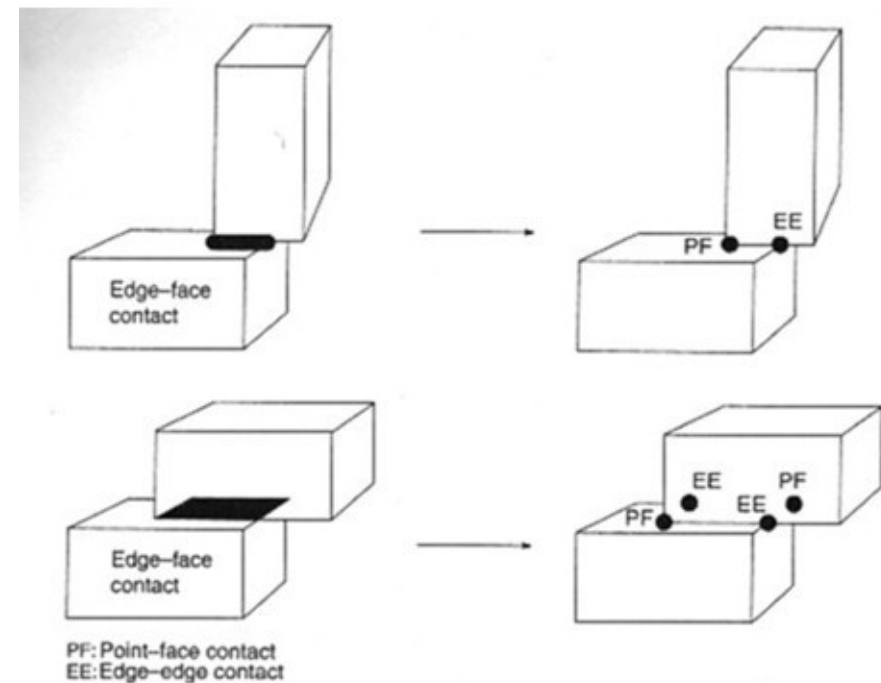
- Es gibt maximal acht Kontaktpunkte



Box - Box Kollision

- Idee

- Punkt-Kante und Punkt-Punkt Kontakt sind so selten, dass sie ignoriert werden können
- Fläche-Fläche und Kante-Fläche Kollisionen können durch die beiden verbliebenen Kollisionensarten modelliert werden

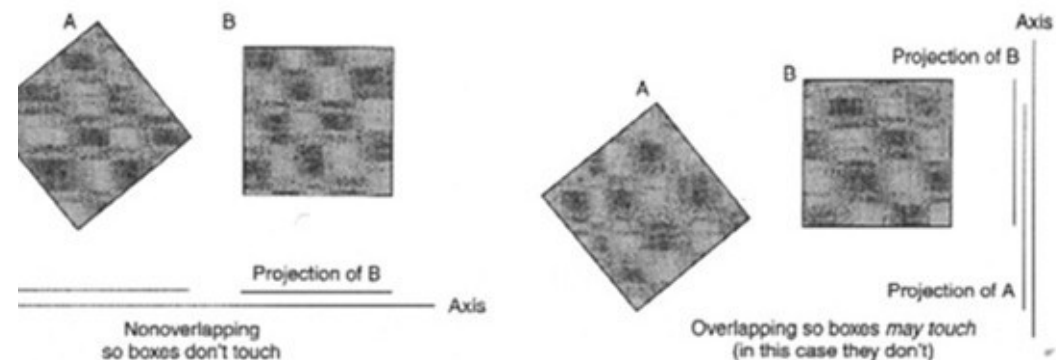


Box - Box Kollision

- Idee

- early out mittels Achsenseparieren

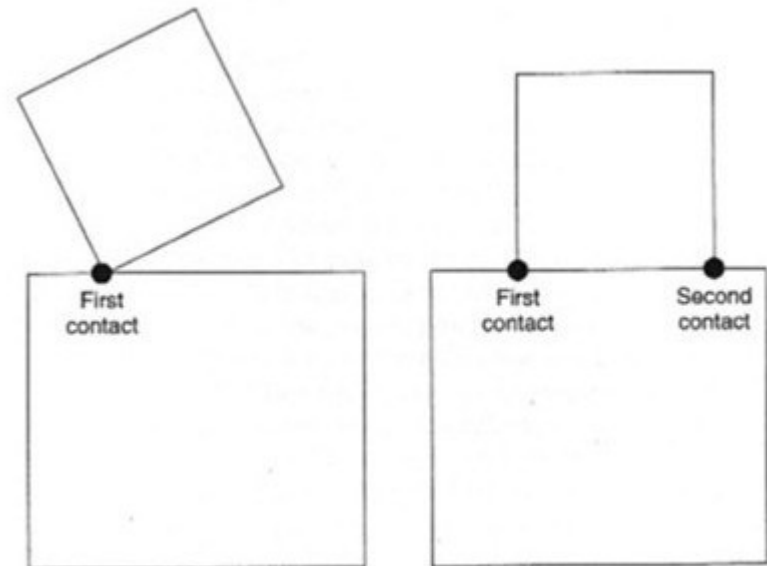
- Projiziere die beiden Boxen in Richtung einer Achse
 - Überschneiden sich die Projektionen nicht, können sich die Boxen nicht berühren
 - Es müssen 15 Achsen getestet werden. Jeweils die drei Richtungen in die sich die Boxen ausdehnen und die Kreuzprodukt aus deren Kombination



Box - Box Kollision

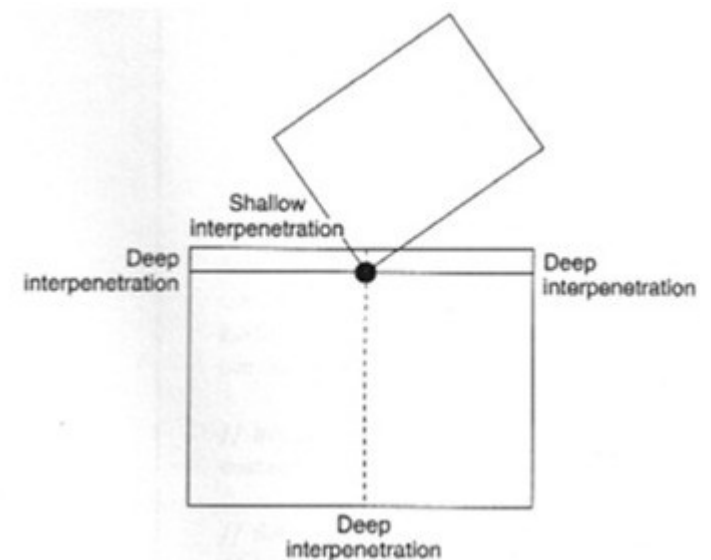
- Idee

- Kontaktpunkte können nacheinander zu einer Gesamtmenge hinzugefügt werden
- Zu jedem Kontaktpunkt zusätzlich seine Art speichern um schneller zu überprüfen, ob dieser Kontakt noch aktuell ist



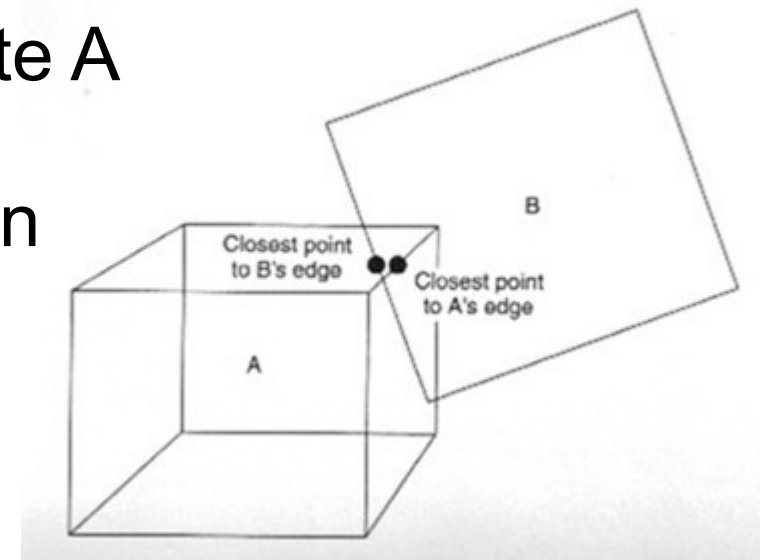
Box - Box Kollision

- Umsetzung
 - Jeder Eckpunkt der einen Box kann mit einer Fläche der andern kollidieren (Kugel-Box Kollision)
 - Überprüfe alle Kombinationen und liefere den am tiefsten eingedrungenen Punkt zurück
 - Für jeden Punkt muss nur seine jeweils minimale Eindringtiefe berücksichtigt werden



Box - Box Kollision

- Umsetzung
 - Jede Kante der Box A kann mit einer Kante der Box B im Kontakt sein
 - Berechnen den Abstand zwischen jeder dieser Kantenkombinationen
 - Ist der nächste Punkt auf Kante A näher am Mittelpunkt von B als der Punkt auf Kante B dann überschneiden sich die Boxen



Box - Box Kollision

- Umsetzung
 - Für jede Kante wird wieder nur der Kontakt mit der geringsten Eindringtiefe benötigt
 - Die Berechnung muss nur aus der Sicht einer Box gemacht werden, da so alle Möglichkeiten abgedeckt sind
 - Der Punkt, der nach dem Punkt-Fläche und dem Kante-Kante Test als am tiefsten eingedrungen berechnet wurde, wird zu der Punktmenge hinzugefügt

Quelle

- Game Physiks
Engin Development
 - ISBN-13: 978-0-12-369471-3
 - Kapitel 13

Danke für eure Aufmerksamkeit!