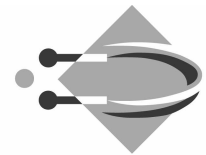


# Physics & animation

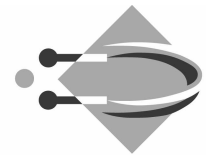


The **IT** University  
of Copenhagen

# Today

---

- 1 Collision detection and response
- 2 How panda is doing it.
- 3 Some Physics
- 4 Collision assignment



# Collision detection and response

---

Collision detection determines if two graphical objects have collide.

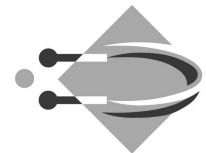
Ex. If two triangles "touch" each other then create a collision event.

Collision response is the reaction build into the game upon the event of a detected collision.

Ex. In case of a collision event then move the involved objects apart.



event example



The **IT** University  
of Copenhagen

# Detection : Broad phase/Narrow phase

---

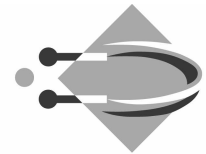
Broad phase for  $n$  objects:  $O(n^2)$

Fast (Crude) test on possible collisions.

Typical tests using bounding volumes.

Narrow phase for  $m$  faces :  $O(m^2)$

Scrutinize possible collisions by checking all combinations of faces from two objects.



# Time in Collision Detection

---

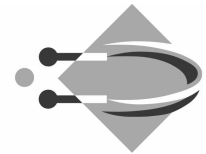
Intrinsic time sampling problems

Has fixed-timestep sampling

Would like fixed collision sampling in time

Ghost passing in the case of large relative velocity compared to sampling rate for collisions.

Exact impact is rare. Penetration is common.



The IT University  
of Copenhagen

# Broad phase strategies

---

## Temporal coherence

Check 4D space-time objects, estimate next time for collision detection

## Spatial coherence

Partition the world in a grid.

Check objects against grid.

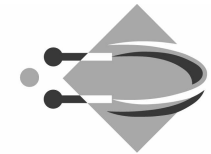
## Bounding volumes

Axis Aligned Bounding Boxes

Object Bounding Boxes (PCA)

Spheres

## Hierarchical optimization of the above



# Broad phase: OBB collision detection

---

If OBBs are disjoint they can possibly be separated by a plane containing a face.

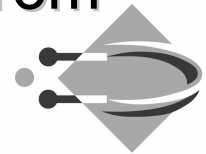
If they are disjoint and cannot be separated by a face then they can be separated with a plane parallel to an edge from each box.

A total of 15 possibilities :

3 faces + 3 faces + 3 edges x 3 edges

Drawing of seperating axis.

(Velocity of the projected objects can be calculated from the 3D velocity)

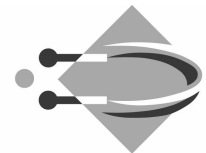


# Narrow Phase , the straightforward test

---

Two convex polyhedra (concave can be decomposed into convex).

1. Test : Verteces of X within Y and vice versa.
2. Test : Edges of X penetrating Y and vice versa.
3. Test : Centroid of faces of X within Y.  
(Identical and aligned objects)



# Single phase approach

---

Hierarchies are needed for optimisation.

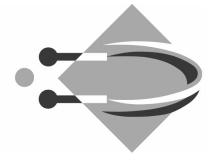
The environment could be done by octrees.

Object hierarchies

Makes hierarchical detection possible.

Ex. Sphere trees

Medial structure and skeletons



The IT University  
of Copenhagen

# Collision Solids in Panda (pusherexample.py)

---

```
#load a model. reparent it to the camera so we can move it.
```

```
s = loader.loadModel('smiley')
```

```
s.reparentTo(camera)
```

```
s.setPos(0, 25,0)
```

```
#create a collision solid for this model
```

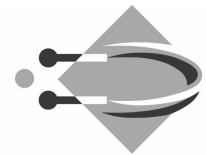
```
cNode = CollisionNode('s')
```

```
cNode.addSolid(CollisionSphere(0,0,0,1.1))
```

```
sC = s.attachNewNode(cNode)
```

```
sC.show()
```

(<Collision Nodepath>.setCollideGeom(1) does not work)



# Handler and traverser

---

#initialize traverser

base.cTrav = CollisionTraverser()

#add this object to the traverser

base.cTrav.addCollider(<collision node>, <collision handler event>)

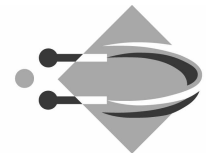
Goes through the list of collision nodes and checks for collisions!

#handlers (who to tell about a collision)

CollisionHandlerEvent()

CollisionHandlerPusher()

CollisionHandlerGravity()



# Collision bitmasks

---

"Into" mask is compared to "From" mask.

#this object can only collide into things

```
sColl[0].setIntoCollideMask( BitMask32().allOff() )
```

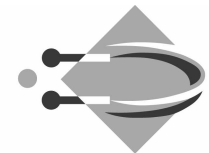
#this object will only collide with objects with this bitmask

```
sColl[0].setFromCollideMask( BitMask32.bit( 1 ) )
```

#this object can only be collided into and will collide with the other object

```
tColl[0].setIntoCollideMask( BitMask32.bit( 1 ) )
```

```
tColl[0].setFromCollideMask( BitMask32().allOff())
```



The **IT** University  
of Copenhagen

# Collision response (app. dependent)

---

Two spheres colliding:

$$M1(V1a - V1) = -P, \quad M2(V2a - V2) = P$$

Impuls preservation

$$M1 V1a + M2 V2a = M1 V1 + M2 V2$$

$$Vrel = V1 - V2, \quad Vrel\_a = V1a - V2a$$

Restitution :

$$Vrel\_after = -e Vrel$$

If the restitution coefficient  $e$  is smaller than 1

then energy is stored in the object causing deformation.



# Collision with a point of contact

---

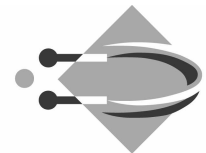
A vertex on one object has contact with a face on the other object.

Here we will only consider translating velocity (not rotational).

The relative velocity perpendicular to the normal of face stays the same.

In a perfect elastic collision the relative velocity in the normal direction is reverted.

In case of a non-elastic collision the relative velocity will be reduced.



# Physics

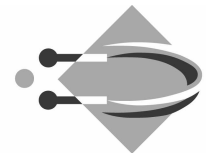
---

Newtons second law :

$$F = m a \quad , \quad a = dv/dt = d^2x/dt^2$$

Forces :

1. Gravity
2. Damping Force  
Opposite and proportional to velocity  
Viscosity linearly proportional  
Air resistance is app. quadratic proportional.
3. Elastic Springs  
Proportional to displacement
4. Geometric constraints, pendulum



# Translation and rotation.

---

The movement induced via a force can both be translation and rotation.

A force that induces rotation is called a torque.

Moment of inertia :

$$I_x = \text{Integrate } (y^2 + z^2) \, dm$$

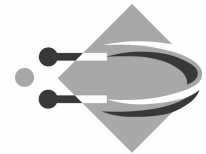
$$I_y = \text{Integrate } (x^2 + z^2) \, dm$$

$$I_z = \text{Integrate } (x^2 + y^2) \, dm$$

In case of asymmetric objects:

$$I_{xy} = \text{Integrate } xy \, dm, \dots$$

$$I = \begin{matrix} & I_x & -I_{xy} & -I_{xz} \\ -I_{xy} & I_y & & -I_{yz} \\ -I_{xz} & -I_{yz} & & I_z \end{matrix}$$



# More translation and rotation

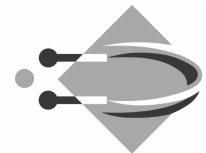
---

$$F = m a$$

$$t = dH/dt, \quad H = I w$$

t : torque, H: angular momentum,

I : Inertia, w: angular velocity



# Physics for a car

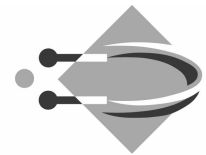
---

Control :

Acceleration = const (Vtarget – Vcurrent)

Limit :  $0 < a < k2/v1$ ,  $k2 = \text{horsepower/mass}$

White board



# Reading , Assignment of the week

---

Literature:

Watt : "3D Computer Graphics", Addison-wesley

Sec. 17.5-6, p.517-529

Sec. 17.4.1-17.4.5, p.505-515

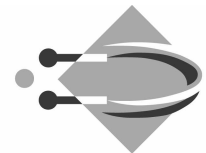
Check out :C:\Panda3d\lib\direct\src\physics

falltest.py rotationtest.py

Next friday

A little car physics in Panda.

A lot AI for car races.



# Assignment

---

Include collision detection in your car game

Include some collision response for instance using the pusher collision handler.

Deadline for handing in is Friday the 18th of April

