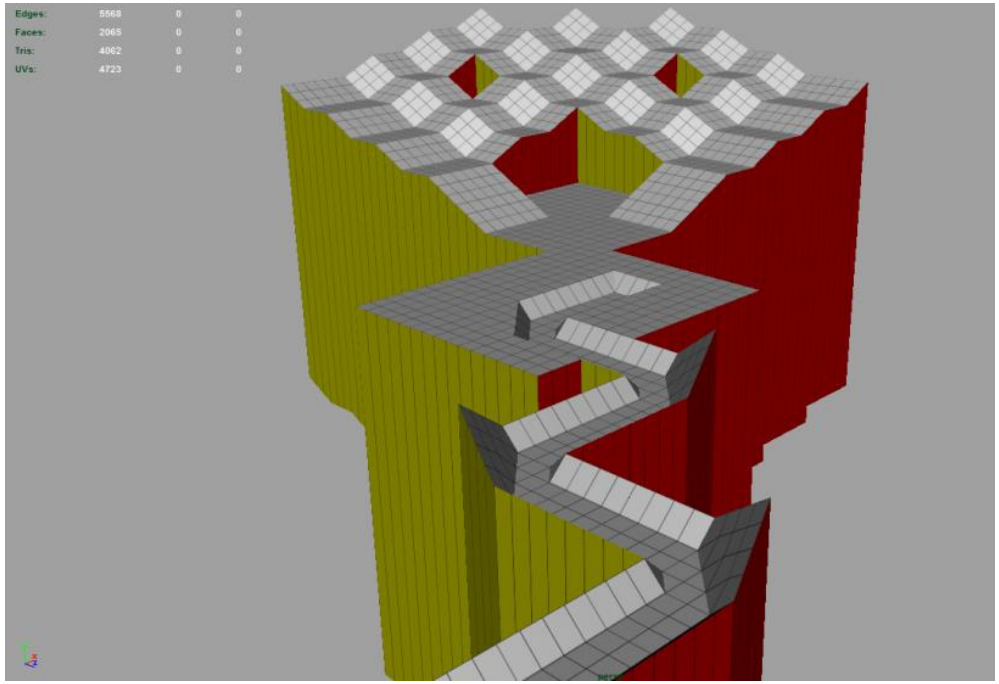
Game Technology

Lecture 9 – 12.12.2014
Physics 2

TECHNISCHE UNIVERSITÄT DARMSTADT



Dr.-Ing. Florian Mehm
Dipl-Inf. Robert Konrad

Prof. Dr.-Ing. Ralf Steinmetz
KOM - Multimedia Communications Lab

Preliminary timetable

TECHNISCHE UNIVERSITÄT DARMSTADT

Lecture No.	Date	Topic
1	17.10.2014	Basic Input & Output
2	24.10.2014	Timing & Basic Game Mechanics
3	31.10.2014	Software Rendering 1
4	07.11.2014	Software Rendering 2
5	14.11.2014	Basic Hardware Rendering
6	21.11.2014	Animations
7	28.11.2014	Physically-based Rendering
8	05.12.2014	Physics 1
9	**12.12.2014**	**Physics 2**
10	19.12.2014	Procedural Content Generation
11	16.01.2015	Compression & Streaming
12	23.01.2015	Multiplayer
13	30.01.2015	Audio
14	06.02.2015	Scripting
15	13.02.2015	AI

Announcement

TECHNISCHE UNIVERSITÄT DARMSTADT

Game Jams

- Game development contest
  - Vague theme (e.g. “10 seconds”)
  - Tight time constraints (e.g. 48 h)
  - Starting from scratch (design, assets, code, …)
- No excuses – just submit something…

The Head Wizards Course, 2014

Ludum Dare 30 @ TUD, 2014

A Maze Thing, 2013

10 Seconds to Apocalypse, 2013

Global Game Jam 2015@TUD

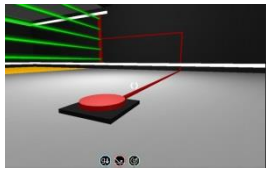
- Fr., 23.01.2015, 17:00 –
  So, 25.01.2015, 17:00
- Registration (first-come-first-serve):
  gamejam@kom.tu-darmstadt.de

10sion, 2013

10Up Experiments: Mountain Brew, 2014

The Most Important 10 Seconds Of Your Life, 2013

As We Are, 2014



Neon Multiverse, 2014

Announcement 2

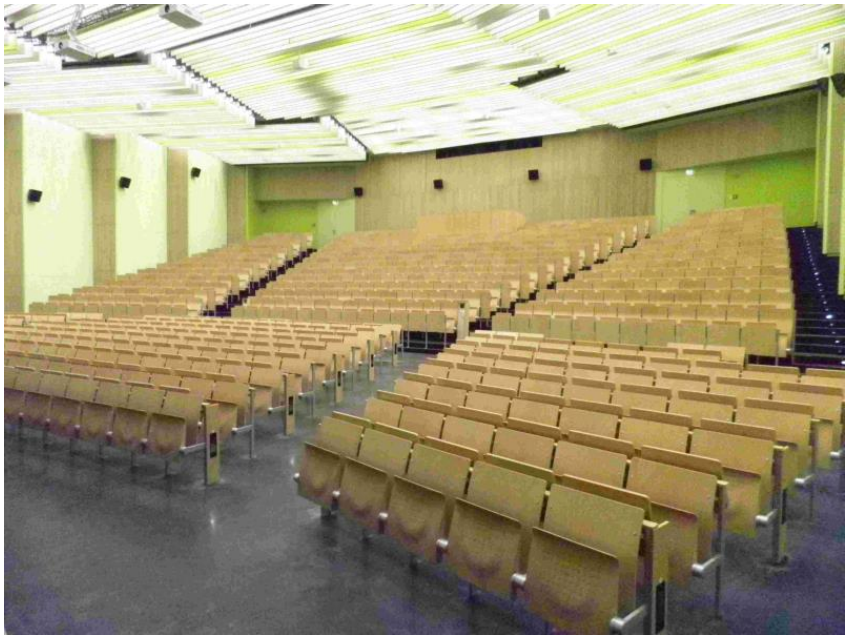
TECHNISCHE UNIVERSITÄT DARMSTADT

The exam is now scheduled for

Friday, 06. March 2015

09-11 AM

S101/A1

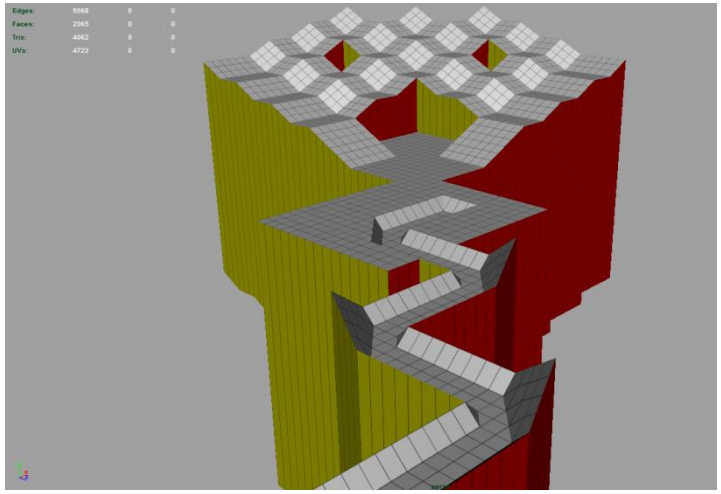


Adding Physics to Marbellous

TECHNISCHE UNIVERSITÄT DARMSTADT

Collision with the level

- Level supplied by artist as 3D mesh
- How to handle the collisions with the mesh?



Friction

- Handle rotations
- Add friction

Controls

- Apply forces when keys are pressed

(Camera control

- Keep the ball in view
- Don‘t follow every single movement)

Hand-placed colliders

TECHNISCHE UNIVERSITÄT DARMSTADT

Sometimes good placeholders for objects or level geometry

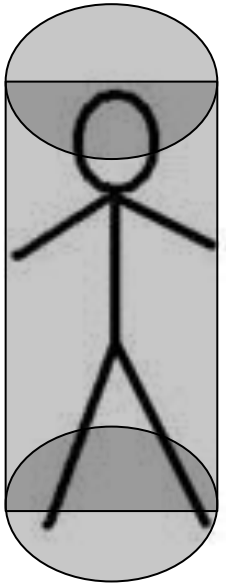
Planes

- Ground plane
- Simple intersection

Boxes

Spheres

Capsules



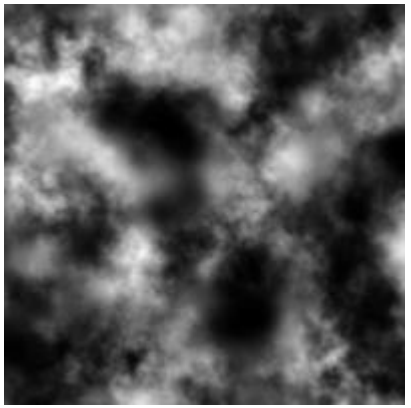
Height map

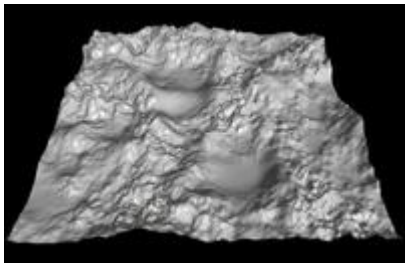
TECHNISCHE UNIVERSITÄT DARMSTADT

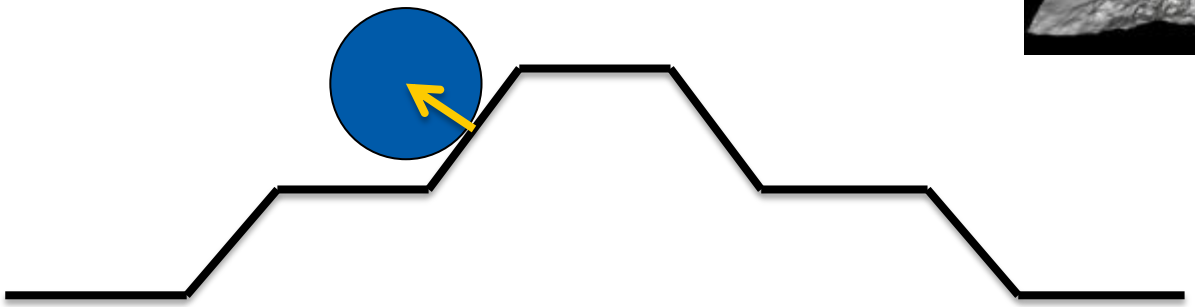
Supplied as a texture or generated

Gives height values at grid points

By interpolating, we can find the height of the mesh under the sphere and the normal







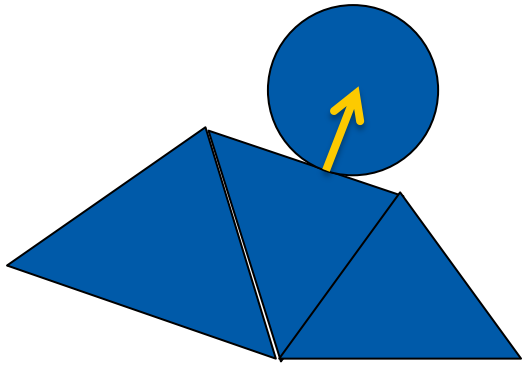
Using the mesh itself

TECHNISCHE UNIVERSITÄT DARMSTADT

Intersection with triangles

Check all triangles

If sphere intersects a triangle, handle the collision



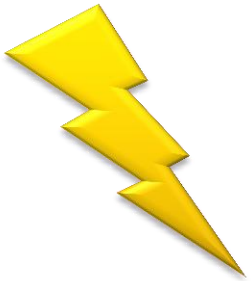
Using the mesh itself

TECHNISCHE UNIVERSITÄT DARMSTADT

Intersection with triangles

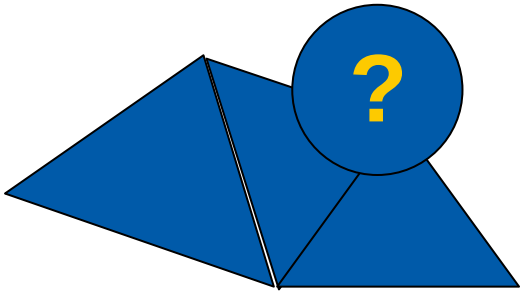
Check all triangles

If sphere intersects a triangle, handle the collision



If there are multiple collisions

- Handle only one (most prominent)
- Handle all



Separating Axis Test

TECHNISCHE UNIVERSITÄT DARMSTADT

If two objects are separated, there must be an axis which separates the two objects

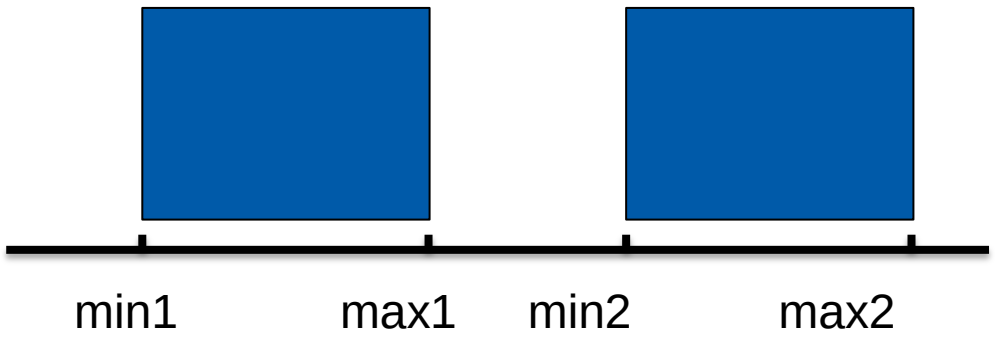
- („Separating Axis Theorem“ → Not a theorem – follows from Hyperplane separation theorem by Hermann Minkowski)
- First mentioned in computer graphics in 1995

Separating Axis Test

TECHNISCHE UNIVERSITÄT DARMSTADT

More exact

- There must be points P1 and P2 of objects 1 and 2 such that the normal resulting from P2 – P1 is a separating axis
- Separating Axis
  - Project all points of the objects onto the separating axis
  - We get the minimal and maximal points min1, min2 and max1, max2
  - The objects are separated iff max1 < min1 or max2 < min1



Separating Axis Test

TECHNISCHE UNIVERSITÄT DARMSTADT

Infinite set of possible points to test for

It can be proven that an upper boundary exists

- Only the relevant axes have to be tested for
- If separation exists on any axis, the test is done → early out for positive test result
- If no separation exists, we still have to test all combinations of features → no early out for negative tests
  - Can be more efficient to reject the test based on other information, e.g. bounding boxes

For polygonal objects, the features are

- Faces
- Edges
- Vertices

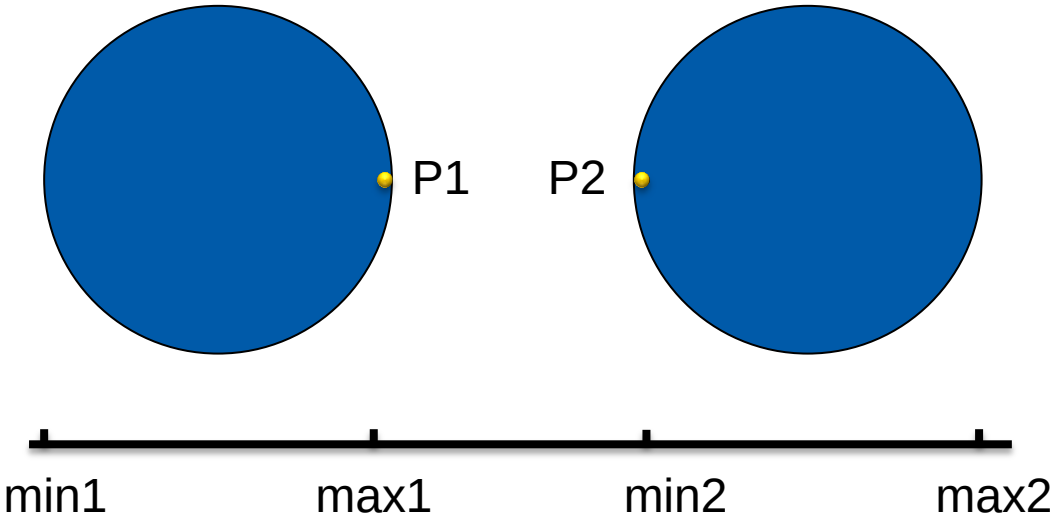
Separating axes for spheres

TECHNISCHE UNIVERSITÄT DARMSTADT

Spheres have no clear feature points

We have already used the separating axis test, though

- The relevant features for two spheres are the two closest points of the spheres
- We find them by finding the axis from one sphere‘s center to the other‘s center
- The intersection test in the previous lecture used this axis for testing intersections



Triangle-Sphere-Test

TECHNISCHE UNIVERSITÄT DARMSTADT

Relevant Features of the Triangle

- Face (x1)
- Vertices (x3)
- Edges (x3)

Relevant feature of the sphere

- The point on the surface closest to the feature of the triangle

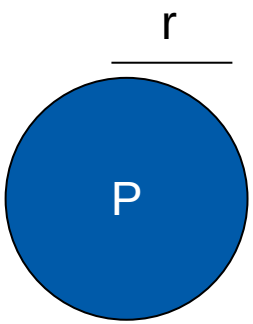
SAT: Testing the plane of the triangle

TECHNISCHE UNIVERSITÄT DARMSTADT

(We have done this test already – need to define the plane)

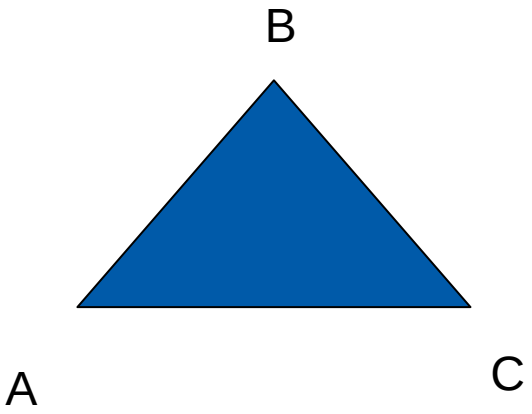
Normal: Use the cross product (very useful for finding normal vectors)

- n = normalize((B – A) x (C - A))



Distance

- Insert one of the points into the equation for distance
- n*A – d = 0 (since A lies on the plane of the triangle)
- →n*A = d



Test for separation

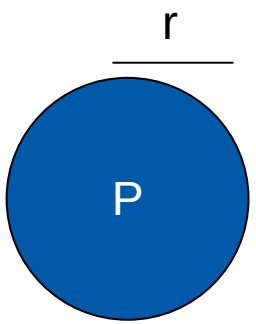
- Separation = distance(Plane, P) > r

SAT: Vertices

Here shown for A (similar for B and C)

Finding the sphere‘s feature

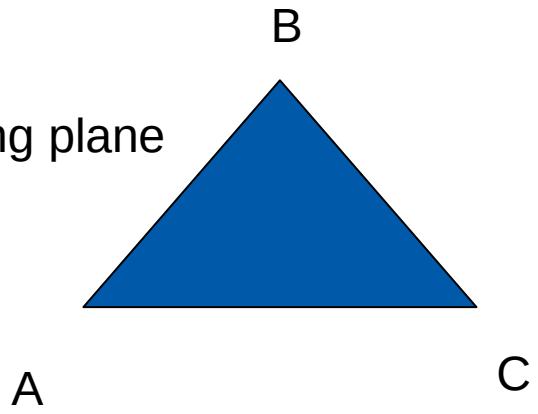
- Along the line from A to P



Compute distance from A to P

Separation (along this axis) iff

- Distance d > r
- And B and C lie on the opposite side of the separating plane

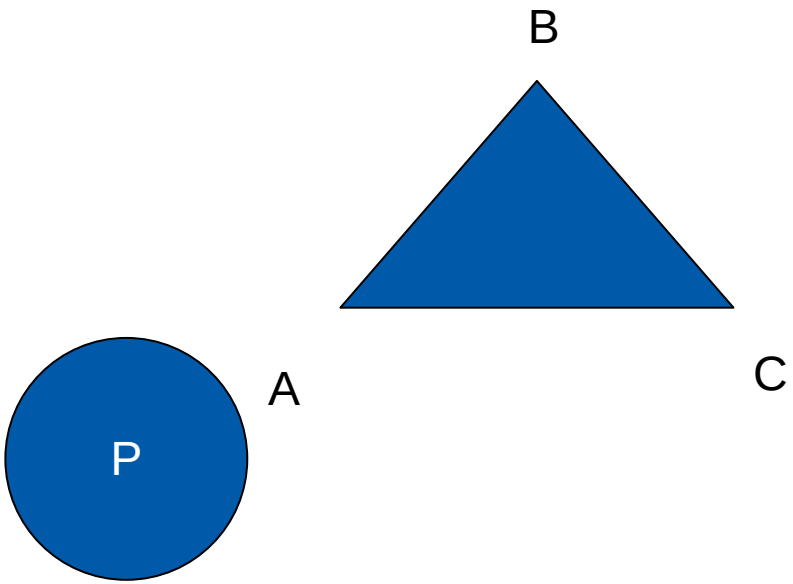


SAT: Vertices

TECHNISCHE UNIVERSITÄT DARMSTADT

Separation (along this axis) iff

- Distance d > r
- And B and C lie on the opposite side of the separating plane

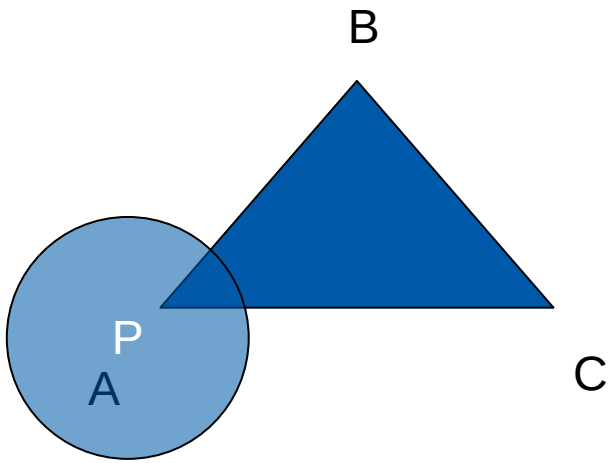


SAT: Vertices

TECHNISCHE UNIVERSITÄT DARMSTADT

Separation (along this axis) iff

- Distance d > r
- And B and C lie on the opposite side of the separating plane

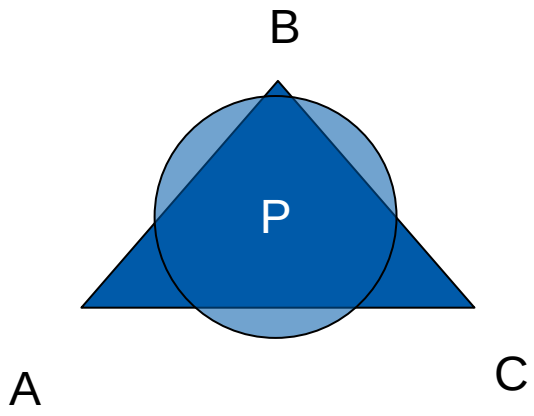


SAT: Vertices

TECHNISCHE UNIVERSITÄT DARMSTADT

Separation (along this axis) iff

- Distance d > r
- And B and C lie on the opposite side of the separating plane



SAT: Vertices

TECHNISCHE UNIVERSITÄT DARMSTADT

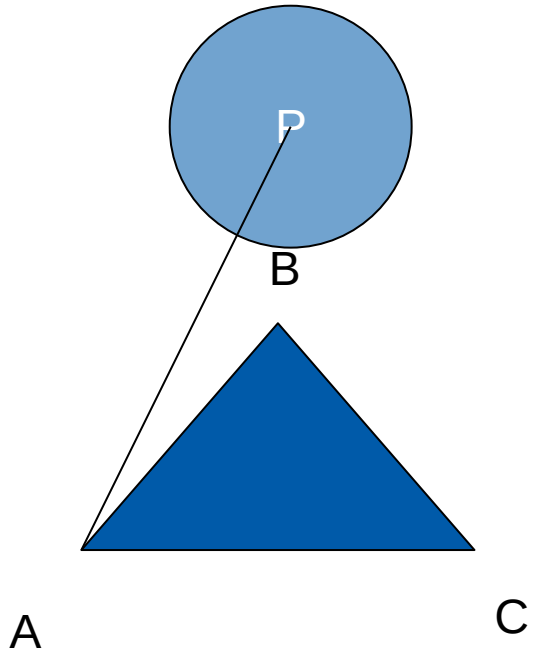
Separation (along this axis) iff

- Distance d > r
- And B and C lie on the opposite side of the plane

→ **We assume that A-P is the separating axis**

→ **No check if A is the closest point**

→ **B and C might be separating axes!**



SAT: Edges

Here shown: AB

Find a point for Q for which Q-P is a normal vector orthogonal to AB

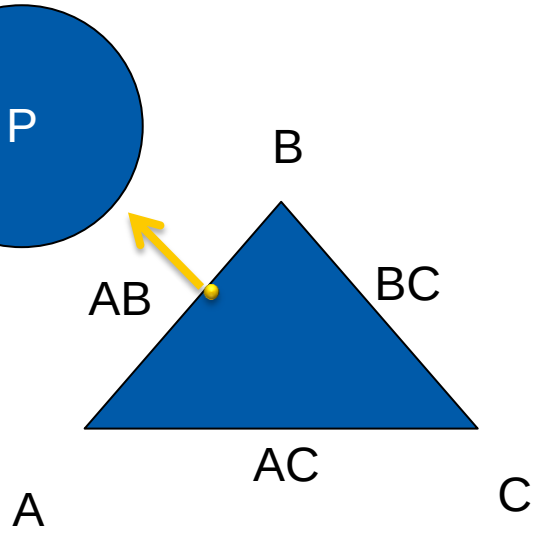
→ **Projection of P onto AB**

→ Use the dot product (ideal for projecting vectors onto each other)

Determine the distance d of Q to P

AB defines a separating axis iff

- Distance d > r
- C lies on the other side of the plane through AB with normal PQ

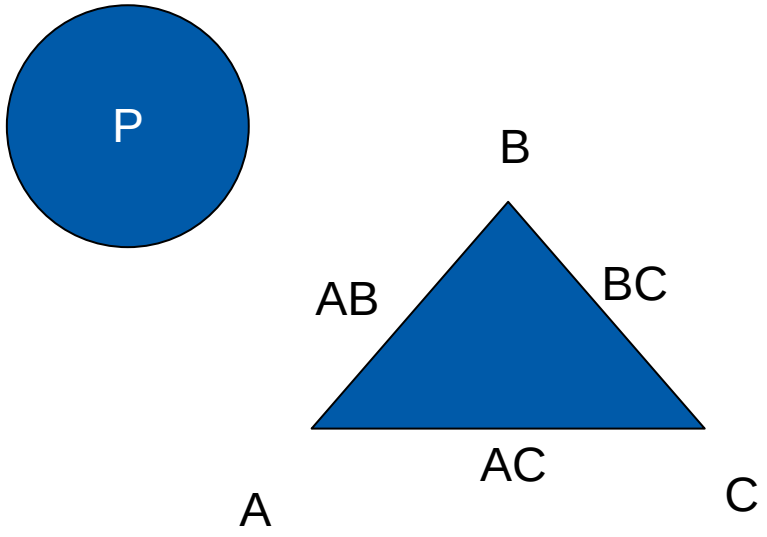


SAT: Edges

TECHNISCHE UNIVERSITÄT DARMSTADT

AB defines a separating axis iff

- Distance d > r
- C lies on the other side of the plane through AB with normal PQ

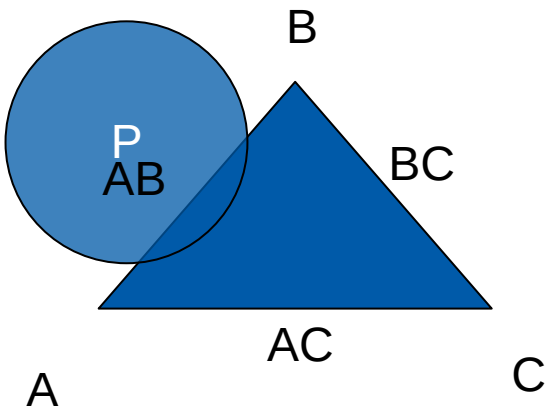


SAT: Edges

TECHNISCHE UNIVERSITÄT DARMSTADT

AB defines a separating axis iff

- Distance d > r
- C lies on the other side of the plane through AB with normal PQ

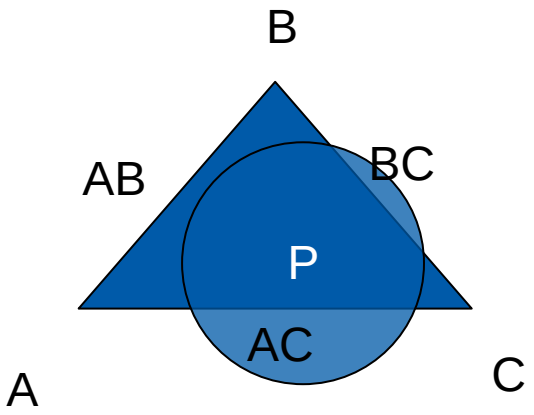


SAT: Edges

TECHNISCHE UNIVERSITÄT DARMSTADT

AB defines a separating axis iff

- Distance d > r
- C lies on the other side of the plane through AB with normal PQ



Speeding the calculation up

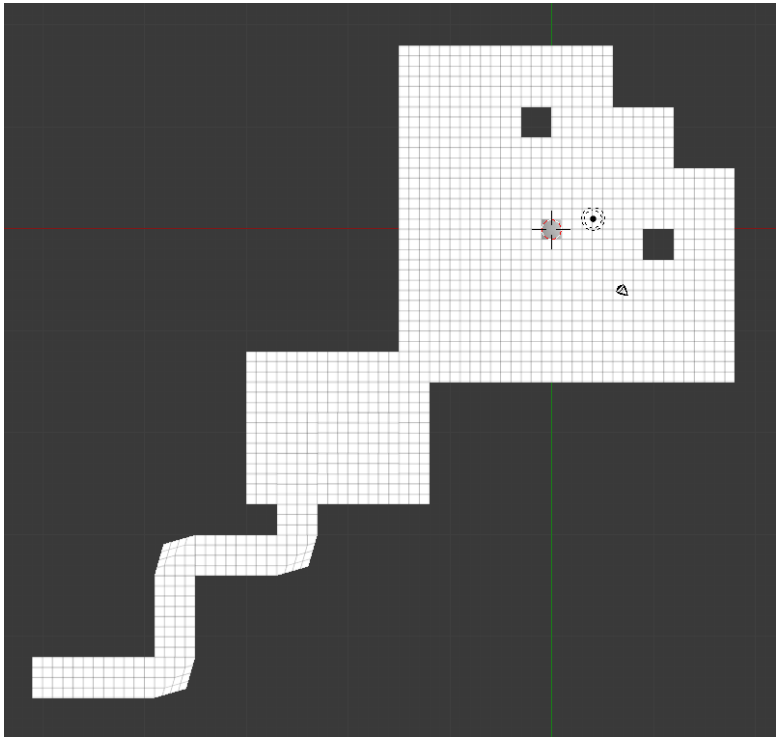
TECHNISCHE UNIVERSITÄT DARMSTADT

Note: In our case, the level is essentially 2D

- Most of our collisions will be from the top of the level

Use a space partition

- Regular Grid
- Quadtree
- KD-Tree
- BSP



Regular grid

TECHNISCHE UNIVERSITÄT DARMSTADT

Subdivide space regularly

E.g. specify

- Cell size in units
- Start point

For each cell

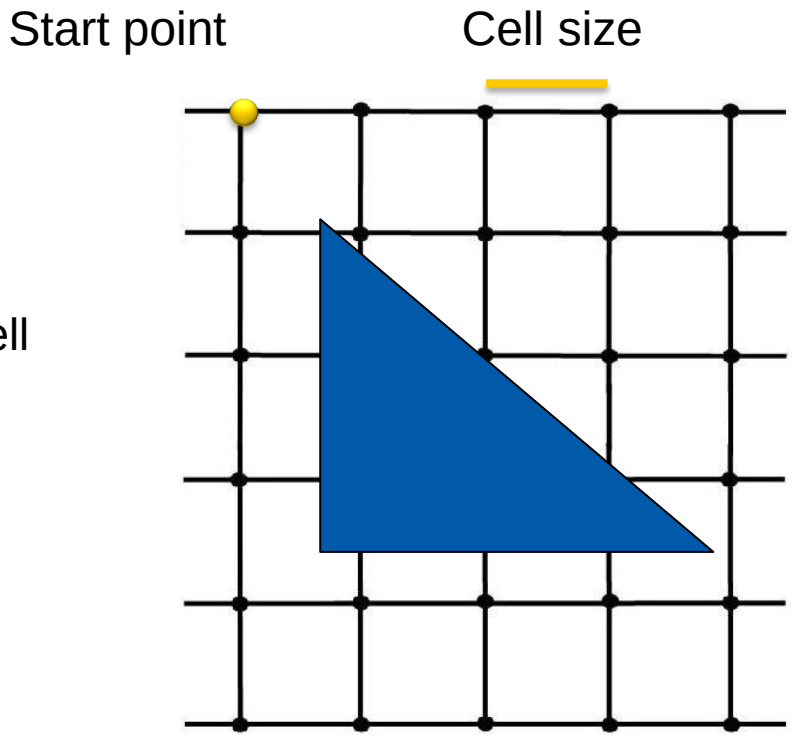
- Test if an object intersects (partly) with the cell
- If so, save a reference to this object
- (Objects can be in several cells)

Advantages

- Easy to compute
- Lookup of cells is trivial

Disadvantages

- Sparsity kills the performance
- Clusters



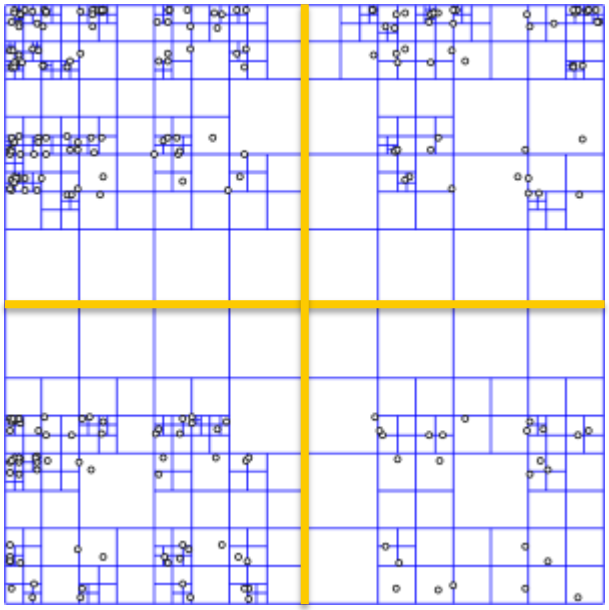
Quadtree(2D), Octree(3D)

TECHNISCHE UNIVERSITÄT DARMSTADT

Start with a rectangular shape

Subdivide the space into 4 or 8 subdivisions of equal size if the number of contained objects is too large

Until the required minimal number of objects per subdivision is found



Advantages

- Still simple lookup where an object is placed
- Can handle clusters better

Disadvantages

- Can cope less with changing number and position of objects

KD-Tree

TECHNISCHE UNIVERSITÄT DARMSTADT

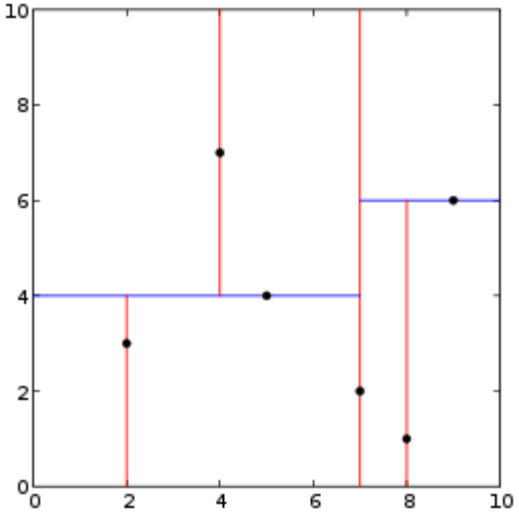
Similar idea to Quad/Octree

Subdivide starting from a rectangular shape

Choose the subdividing line

- E.g. median point of the contained objects (cutting them in half)

Alternate axes for subdivision



Advantages:

- Well suited for clusters

Disadvantages

- Lookup harder than octree

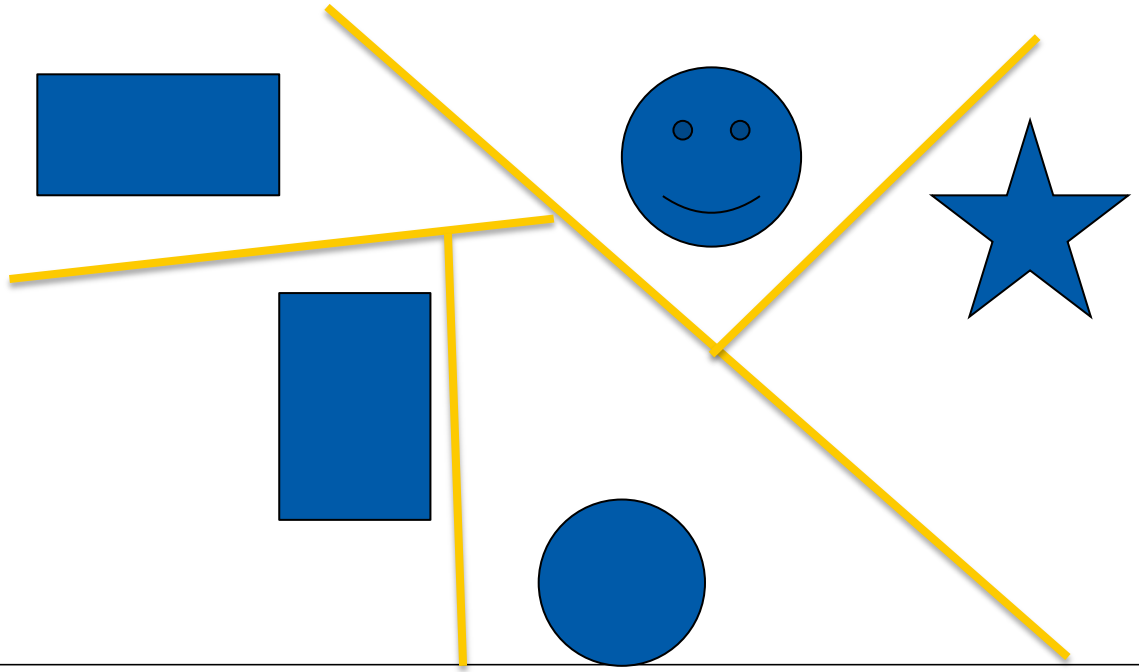
Binary Space Partition

TECHNISCHE UNIVERSITÄT DARMSTADT

Generalization of KD-Tree

Subdivide the space into half-spaces with arbitrary planes

Used previously to speed up rendering (Quake Engine)



Reducing the dimensionality

TECHNISCHE UNIVERSITÄT DARMSTADT

Many problems in 3D games are essentially 2D

- Heightmaps
- Top-down shooters
- Real-Time Strategy games
- ...

In Marbellous, we can expect that

- No overhangs are present in the level
- The sphere will stay close to the mesh at all times

If we look at the level from above, we can see that if we put a grid over the game world, only the triangles in the same 2D cell can be possibly colliding

→ During initial setup and the lookup, project everything into 2D

Lookup

TECHNISCHE UNIVERSITÄT DARMSTADT

Saving the triangles

- We should save only the triangles that are contained in the grid cell
- → We need to check intersection between a rectangle and a triangle

Minimizing storage

- Re-use the vertex and index buffer
- Save only the index of the triangle
  - (Ideally, we will not suffer from too many cache misses, since the goal in the first place is to reject most collision tests early)

Intersection between the triangles and the grid

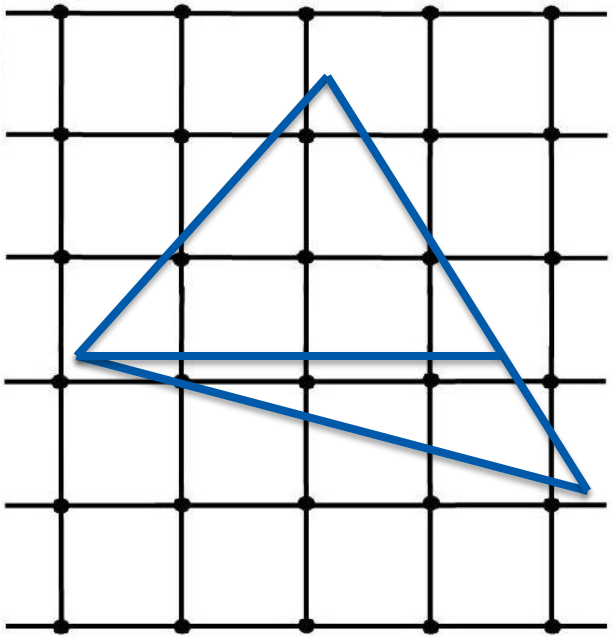
TECHNISCHE UNIVERSITÄT DARMSTADT

Re-use the scanline rasterization algorithm

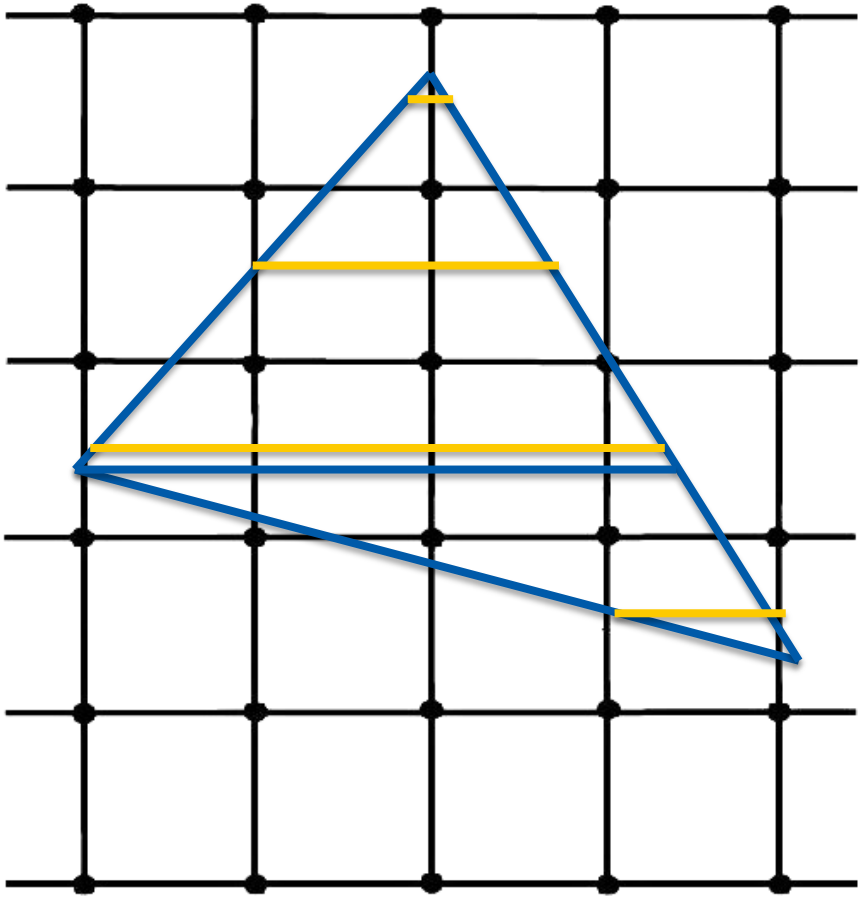
- Very similar task
- But have to watch out due to larger cell size

Original algorithm

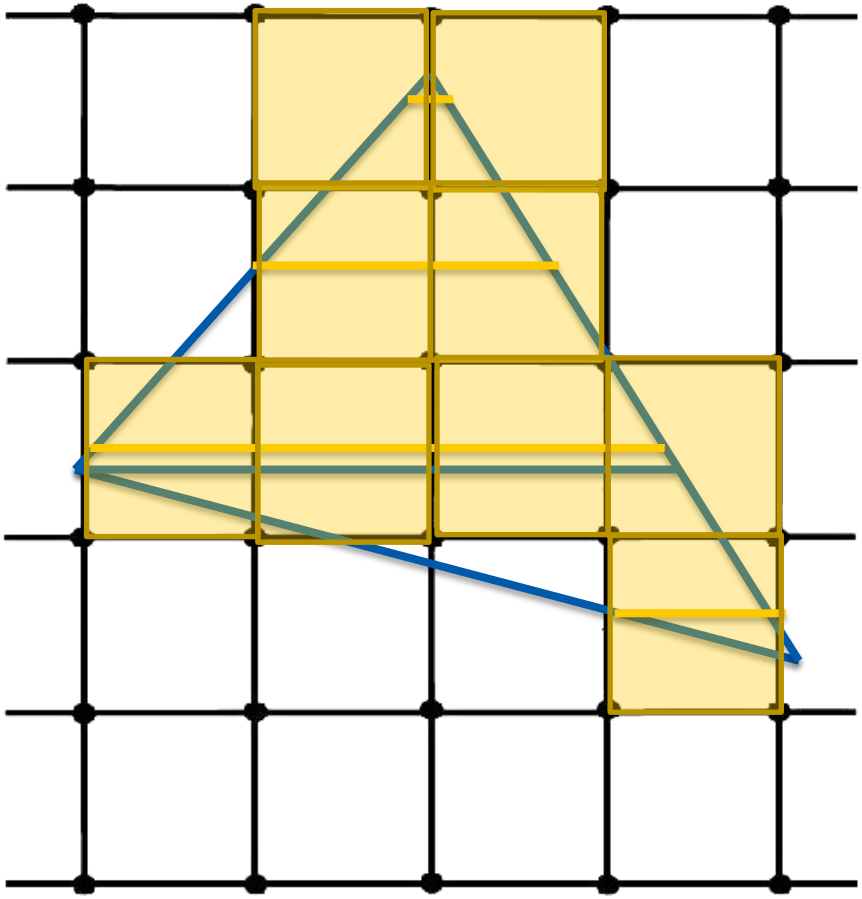
- Find edge longest with biggest ydif
- Fill lines between long edge and other edge 1
- Fill lines between long edge and other edge 2



Triangle Rasterisation



Triangle Rasterisation



New algorithm

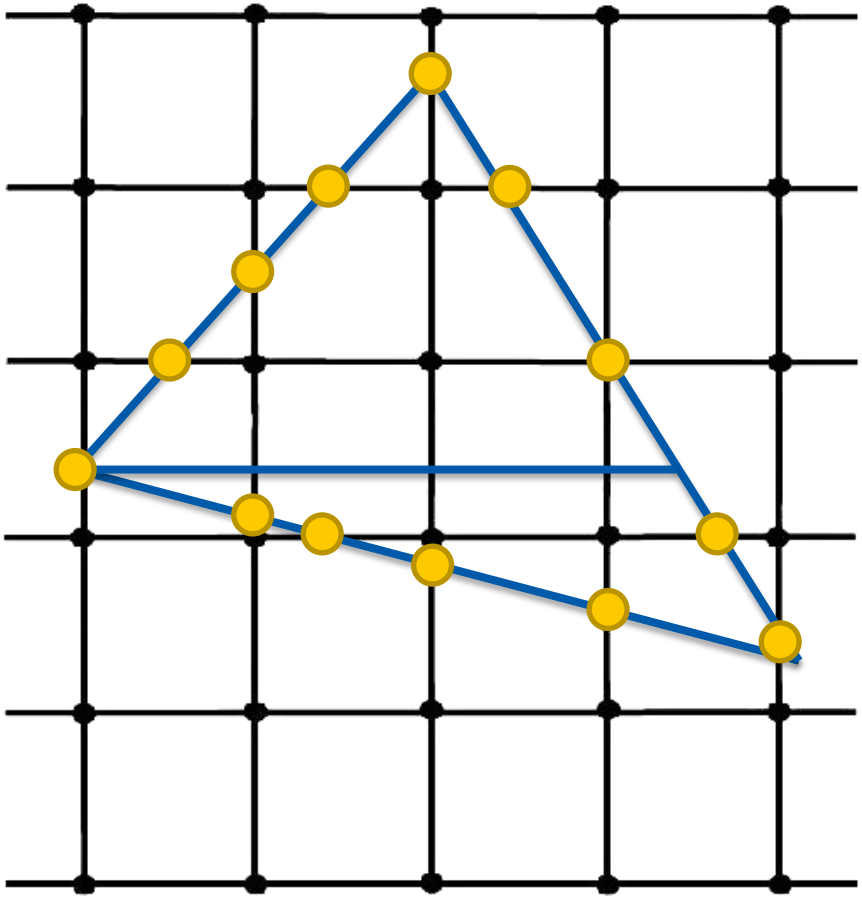
TECHNISCHE UNIVERSITÄT DARMSTADT

Calculate intersection with all grid lines

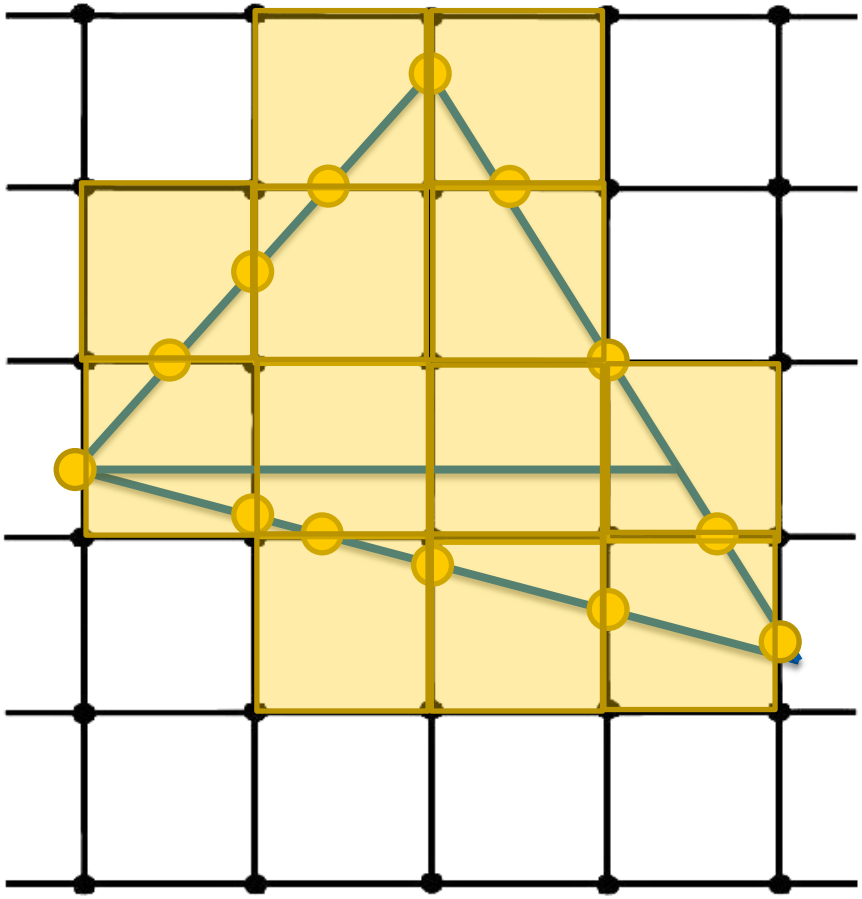
For each row

- Left extent is the minimal intersection point
- Right extent is the maximal intersection point

Triangle Rasterisation



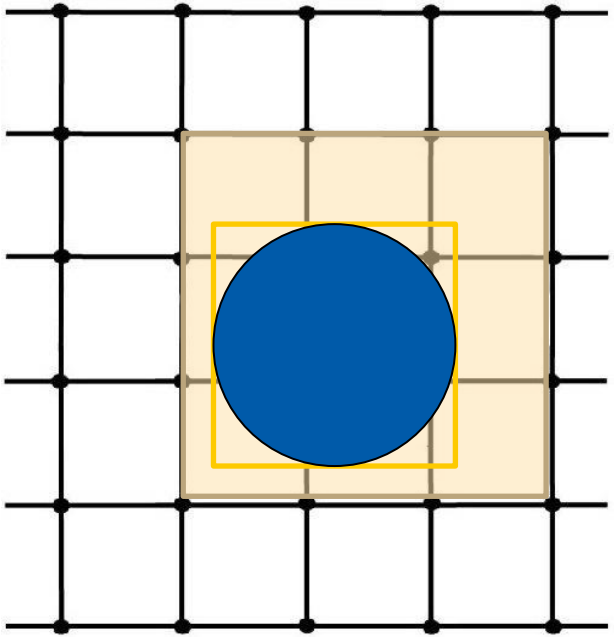
Triangle Rasterisation



Intersection between the sphere and the grid

TECHNISCHE UNIVERSITÄT DARMSTADT

- Use the bounding box of the sphere
- Defined by the extents in the x-z-Plane
- (Or implement rectangle-sphere intersection)



Is it worth it?

TECHNISCHE UNIVERSITÄT DARMSTADT

No (at least not for our exercise)

On a Core2 Duo @2.7 GHz, the intersection with the mesh takes about 0.908 ms in Release mode

But, for production code, larger meshes and more objects, it could become relevant

(Triangle-Sphere Intersection implemented with optimized code by Christer Ericson, http://realtimecollisiondetection.net/blog/?p=103)

Broad Phase vs. Narrow Phase

TECHNISCHE UNIVERSITÄT DARMSTADT

Broad Phase

Rule out as many possible collisions

Only call narrow phase if separation can not be proven here

Reduce the problem

- Use spatial data structures (grid, octree, etc.)
- Use bounding volumes (and bounding volume hierarchies)

Narrow Phase

Check for exact collisions

Use exact tests

- E.g. based on SAT

Should be much slower than broad phase and therefore seldomly called

Provide collision data to resolver

Time Handling

TECHNISCHE UNIVERSITÄT DARMSTADT

Fixed Time Step

- Explicit Time Step → Our method
- (Semi-)Implicit Time Step Method
  - Try to predict the times of collisions and handle them at the beginning

Adaptive Time Step

- Retroactive Detection
  - If there is interpenetration at t +deltaT, use deltaT *= 0.5 and retry
- Conservative Advancement
  - Predict the next time of collision
  - Advance to this time

Continuous Collision Detection

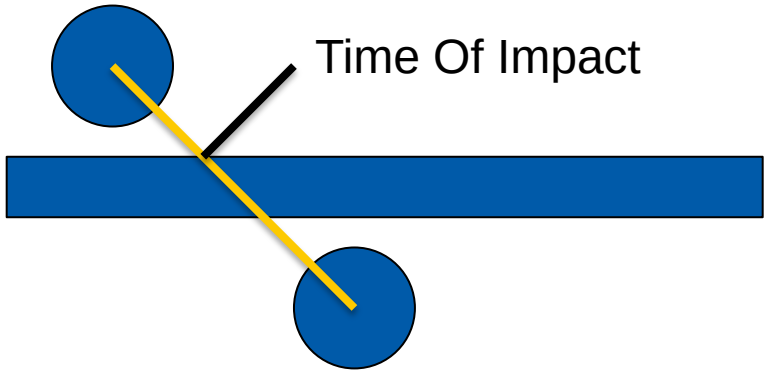
TECHNISCHE UNIVERSITÄT DARMSTADT

Check if an object moved through another in the frame

- On one side before, on one side after
- Swept shape algorithms

Time of impact ordering

Go to time of impact, resolve there

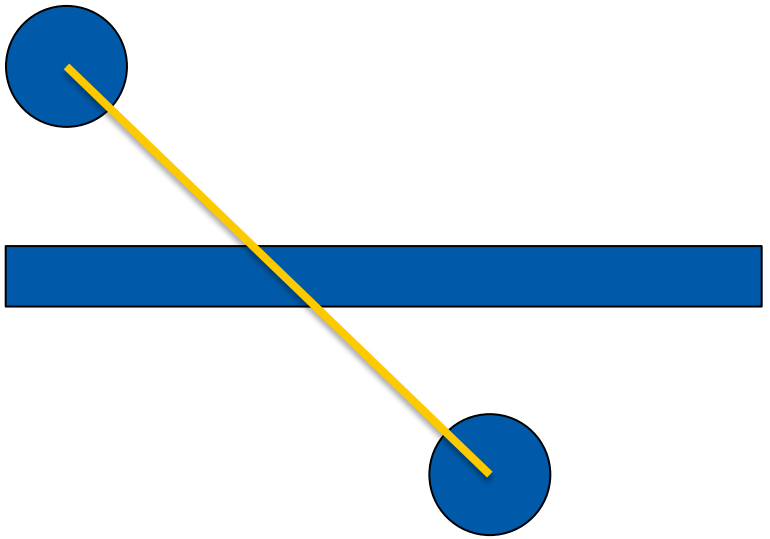


Speculative Contact

TECHNISCHE UNIVERSITÄT DARMSTADT

Calculate the distance to the collider

Remove just enough velocity so they touch in the next frame

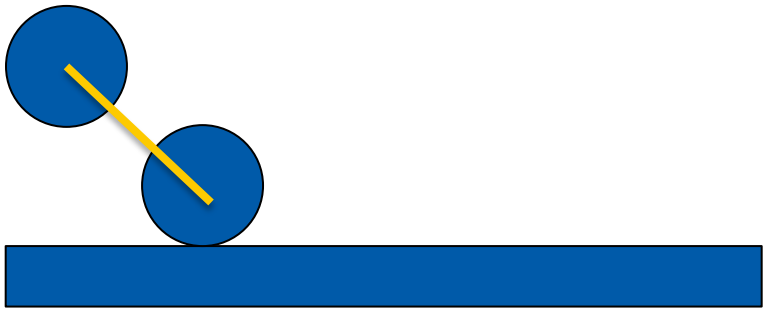


Speculative Contact

TECHNISCHE UNIVERSITÄT DARMSTADT

Calculate the distance to the collider

Remove just enough velocity so they touch in the next frame



Constraints

TECHNISCHE UNIVERSITÄT DARMSTADT

Stiff constraints

- Keep objects at an exact length compared to each other
- E.g. when attached to a steel cable

Springs

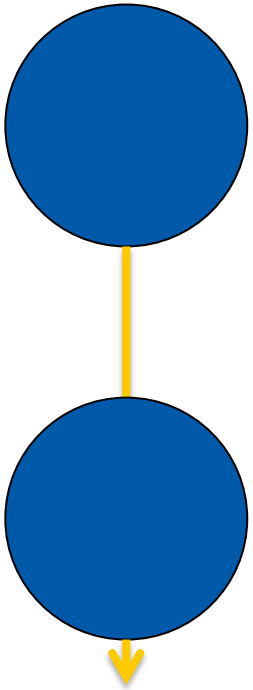
- Variable length between objects
- E.g. when attached to a bungee rope

Stiff Constraints - Rods

TECHNISCHE UNIVERSITÄT DARMSTADT

Distance between two objects is determined to stay constant

→ **Separating Velocity between the two objects along the vector from one to the other should be 0 at all times**

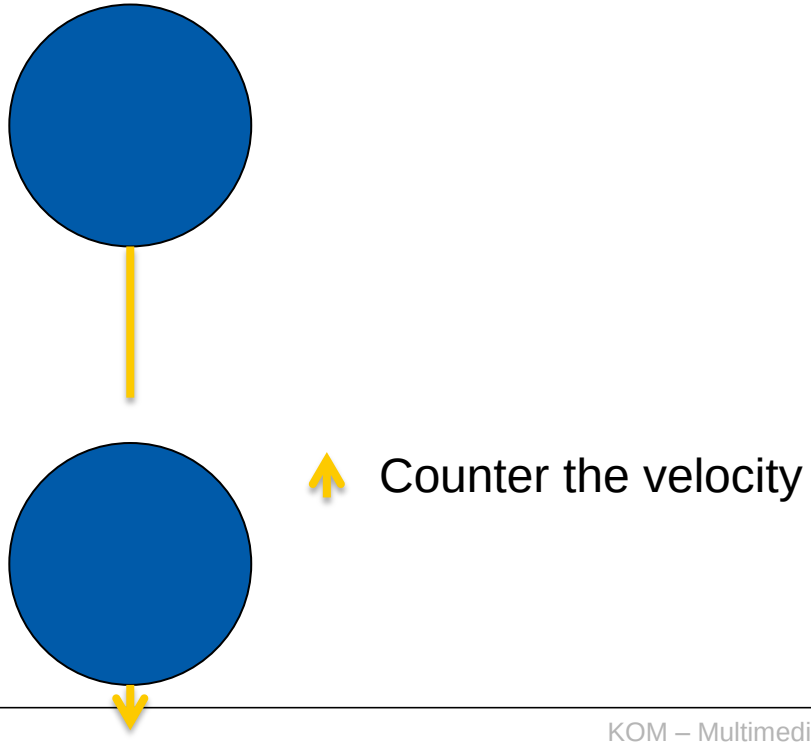


Stiff Constraints - Rods

TECHNISCHE UNIVERSITÄT DARMSTADT

Distance between two objects is determined to stay constant

→ **Separating Velocity between the two objects along the vector from one to the other should be 0 at all times**

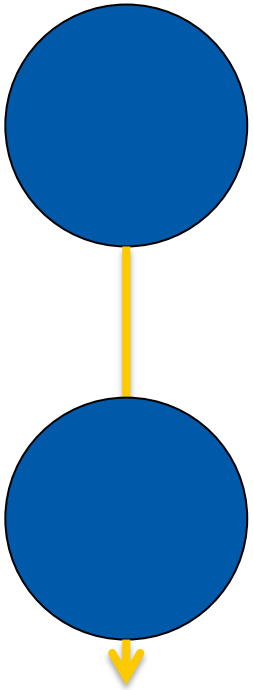


Stiff Constraints - Rods

TECHNISCHE UNIVERSITÄT DARMSTADT

Distance between two objects is determined to stay constant

→ **Separating Velocity between the two objects along the vector from one to the other should be 0 at all times**



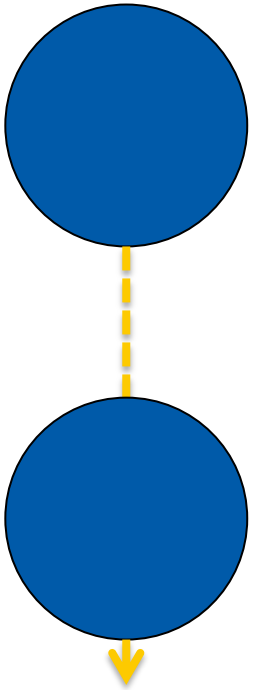
Spring Constraints

TECHNISCHE UNIVERSITÄT DARMSTADT

Model a spring between two objects (one might be stationary)

Spring force

- Rest length (no force)
- Stiffness
- (Breaking point)



Hooke‘s Law

TECHNISCHE UNIVERSITÄT DARMSTADT

F = -k * (l – l0)

F: Spring force

k: Spring constant (stiffness)

l: Current length of the spring

l0: Rest length of the spring

Apply the resulting force to the objects that are attached

(One might be immovable)

Stiffness

TECHNISCHE UNIVERSITÄT DARMSTADT

Also a property of numerical systems

The stiffer, the more problems we face → exploding systems

J. D. Lambert : "If a numerical method with a finite region of absolute stability, applied to a system with any initial conditions, is forced to use in a certain interval of integration a steplength which is excessively small in relation to the smoothness of the exact solution in that interval, then the system is said to be stiff in that interval."

Particle networks

TECHNISCHE UNIVERSITÄT DARMSTADT

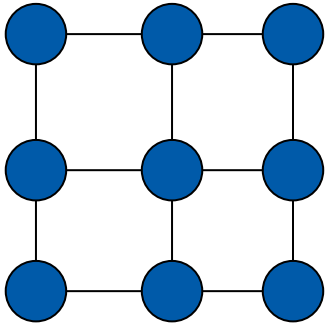
Connect multiple particles with springs

Approximation for deformable objects

Often used for cloth

Problems/Challenges

- Stiff constraints
- Self-intersections
- Stability



Deformable objects

TECHNISCHE UNIVERSITÄT DARMSTADT

Generalization of particle networks

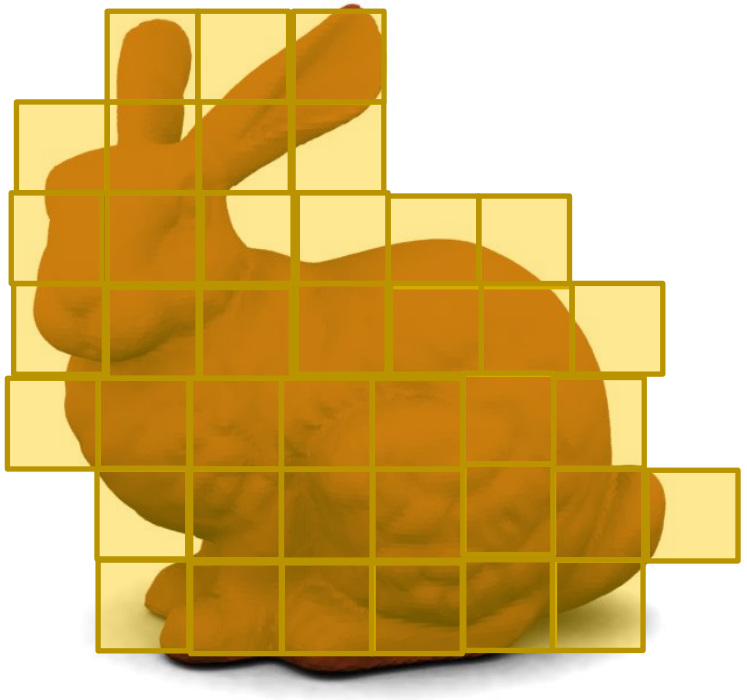
Finite Element Method from Mechanics

Model forces inside the object

- Stress
- Strain

Gasses, Liquids

- Discretize into a vector field
- Calculate flow by solving the Navier-Stokes-Equations



Collision handling schemes

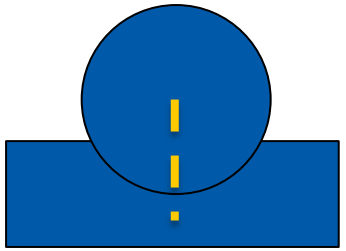
TECHNISCHE UNIVERSITÄT DARMSTADT

Impulse-based Micro-Collisions

- What we are using

Spring-Based

- Insert a spring at the point where the collision is detected
- Forces the objects out again



Constraint-Based

- Formulate the collisions as violations of constraints

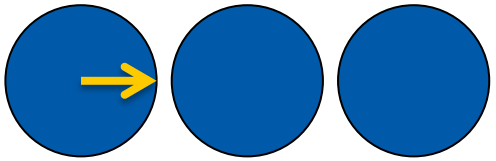
Sequential Impulses

TECHNISCHE UNIVERSITÄT DARMSTADT

Aka. Propagating Impulses

Stability

- Add iterations
- Solve impulses in order of importance



Adaptive schemes

- Few, „large“ contacts → need fewer iterations
- Many contacts → need more iterations

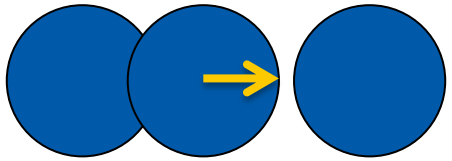
Sequential Impulses

TECHNISCHE UNIVERSITÄT DARMSTADT

Aka. Propagating Impulses

Stability

- Add iterations
- Solve impulses in order of importance



Adaptive schemes

- Few, „large“ contacts → need fewer iterations
- Many contacts → need more iterations

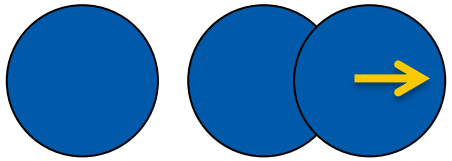
Sequential Impulses

TECHNISCHE UNIVERSITÄT DARMSTADT

Aka. Propagating Impulses

Stability

- Add iterations
- Solve impulses in order of importance



Adaptive schemes

- Few, „large“ contacts → need fewer iterations
- Many contacts → need more iterations

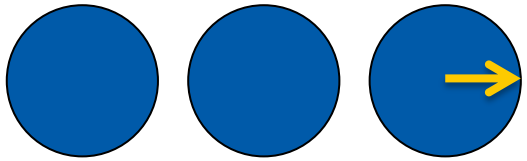
Sequential Impulses

TECHNISCHE UNIVERSITÄT DARMSTADT

Aka. Propagating Impulses

Stability

- Add iterations
- Solve impulses in order of importance



Adaptive schemes

- Few, „large“ contacts → need fewer iterations
- Many contacts → need more iterations

Constraint-based

TECHNISCHE UNIVERSITÄT DARMSTADT

Constraint Vector

- For each collision or constraint
- Equality constraint
  - Objects should stay at a fixed relative position
- Inequality constraints
  - E.g. for separating objects after collisions

For each collision, add a constraint to the constraint vector

Results in a large system of equations

Solve via Linear Complementarity Problem (LCP)

Other integrators

TECHNISCHE UNIVERSITÄT DARMSTADT

Runge Kutta 4th order (RK 4)

- Approximate from 4 values

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$
$$t_{n+1} = t_n + h$$

$$k_1 = f(t_n, y_n),$$
$$k_2 = f(t_n + \tfrac{h}{2}, y_n + \tfrac{1}{2}k_1 h),$$
$$k_3 = f(t_n + \tfrac{h}{2}, y_n + \tfrac{1}{2}k_2 h),$$
$$k_4 = f(t_n + h, y_n + k_3 h).$$

Velocity-less Verlet integration

- Uses no explicitly saved velocity
- Instead, uses position difference between this and previous calculation

- x(t + deltaT) = 2 * x(t) – x(t – deltaT) + deltaT^2 * a

Rotation

TECHNISCHE UNIVERSITÄT DARMSTADT

Angular Velocity, Acceleration

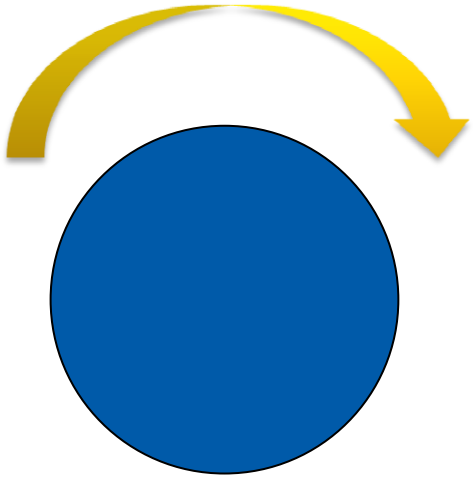
- Save as additional properties
- Velocity: 3-Vector, Rotations around x, y, z axis
- Accelaration: Change in angular velocity

Mass Moment of Inertia

- Property that resists the change in angular velocity

Torque

- Force acting off-center



Torque

TECHNISCHE UNIVERSITÄT DARMSTADT

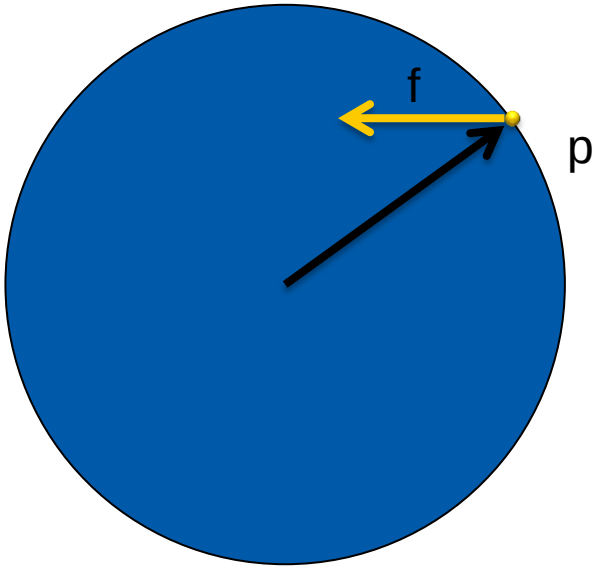
torque = p x f

p is the point of application

f is the force applied

Note: If p and f are in the same direction

→ No torque



Mass Moment of Inertia

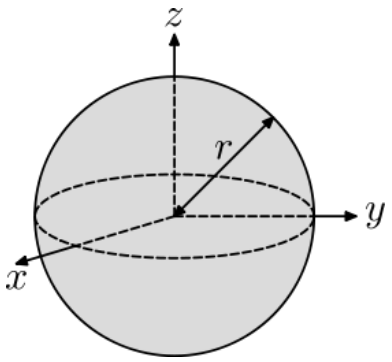
TECHNISCHE UNIVERSITÄT DARMSTADT

Inertia Tensor – Generalized version of a matrix

For purposes of games, most often 3x3

Diagonal Matrix for moments of inertia about x, y, z-axis
Off-center entries encode product of inertia

See http://en.wikipedia.org/wiki/List_of_moments_of_inertia



$$I = \begin{bmatrix} \frac{2}{5}mr^2 & 0 & 0 \\ 0 & \frac{2}{5}mr^2 & 0 \\ 0 & 0 & \frac{2}{5}mr^2 \end{bmatrix}$$

Inverse Inertia Tensor

TECHNISCHE UNIVERSITÄT DARMSTADT

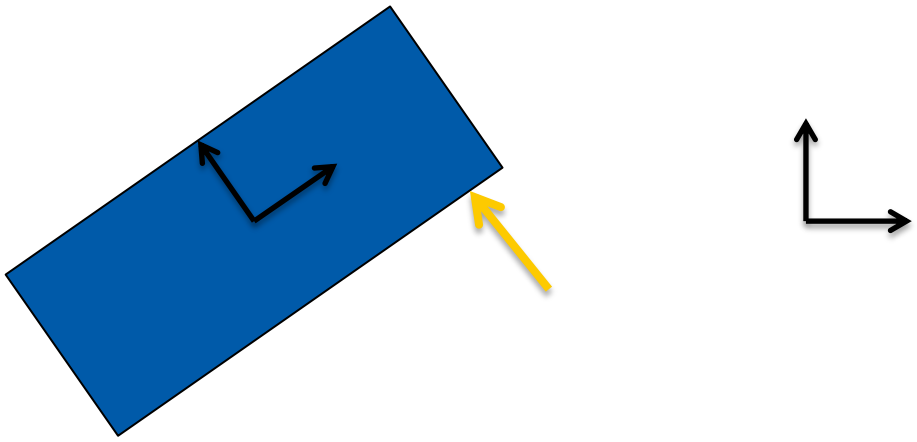
Remember the calculation of forces

F = m * a → a = F / m

We need the inverse of the inertia tensor for the equivalent formula

Additionally, need to transform to the world coordinate system

→ Torques given in world coordinates



Integration

TECHNISCHE UNIVERSITÄT DARMSTADT

Add an accumulator for Torque
(D‘Alambert‘s Principle also works here)

Add all forces to linear accumulator

Calculate torque for each force

Add torques to torque accumulator

Integration

- Multiply inverse mass moment of inertia with sum of torques

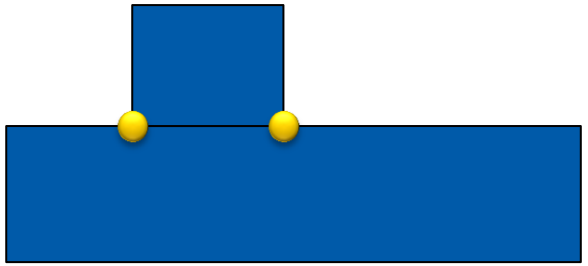
Handling non-spherical rigid bodies

TECHNISCHE UNIVERSITÄT DARMSTADT

E.g. a box

Can collide with any feature

- Face
- Vertex
- Edge



If we handle only one feature, the others would sink

Sequential impulses

- One part starts sinking into the floor
- Push up → Rotation
- Continue
- Needs iterations to get stable

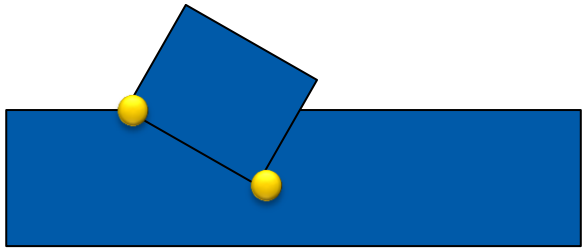
Handling non-spherical rigid bodies

TECHNISCHE UNIVERSITÄT DARMSTADT

E.g. a box

Can collide with any feature

- Face
- Vertex
- Edge



If we handle only one feature, the others would sink

Sequential impulses

- One part starts sinking into the floor
- Push up → Rotation
- Continue
- Needs iterations to get stable

Friction

TECHNISCHE UNIVERSITÄT DARMSTADT

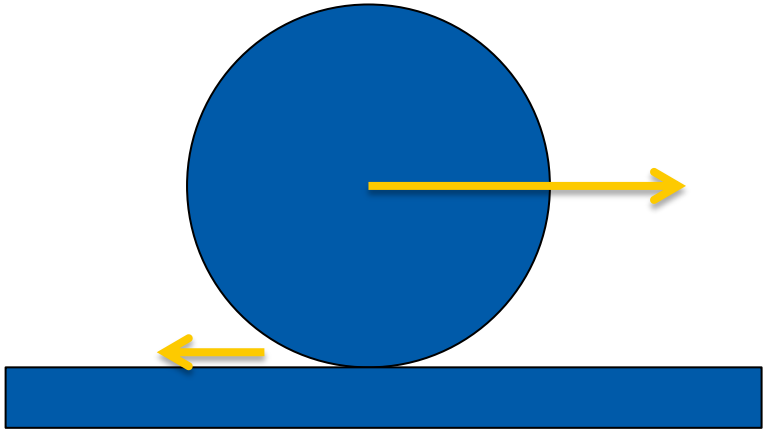
In the previous exercise, our spheres slided over the floor

→No rotation

→They came to rest because of dampening and not friction

Friction resists the spheres at the point of contact with the floor

- Rolling along the floor
- Different coefficients
  - Ice
  - Smooth floor
  - Sand
  - …



Coulomb‘s Law

TECHNISCHE UNIVERSITÄT DARMSTADT

Depends on

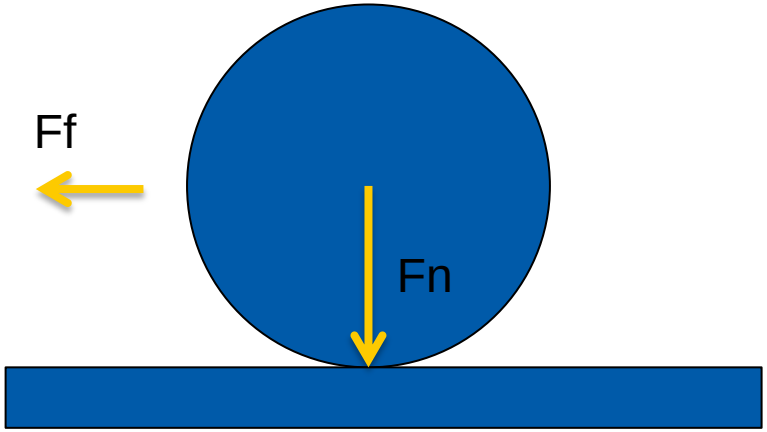
- normal force that presses the surfaces together
- coefficient of friction
  - Most dry materials have a coefficient of friction of 0.1 to 0.6

Ff: Friction force

μ: Coefficient of friction

Fn: Normal force

$$F_\mathrm{f} \leq \mu F_\mathrm{n}$$



In 3D

- Tangential plane
- Force lies in this plane

Friction

TECHNISCHE UNIVERSITÄT DARMSTADT

Static friction

- Keeps objects in place
- Start moving when the limit is overcome
- f_static <= k_static * |r|
- k_static: Constant for friction between the involved materials
- r: Reaction force of the ground at the point of contact

Dynamic friction

- Force between the objects while they are sliding across
- f_dynamic = -v_planar * k_dynamic * |r|
- v_planar: The velocity of the object across the surface
- k_dynamic: Constant for dynamic friction

Contact basis

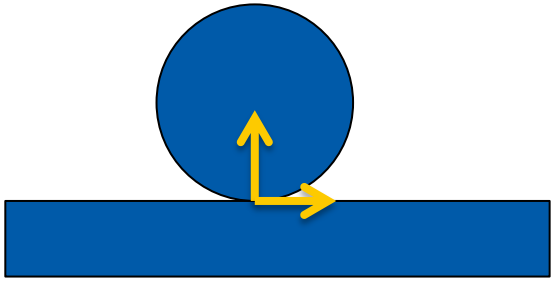
TECHNISCHE UNIVERSITÄT DARMSTADT

Calculating friction requires us to calculate the velocity along the contact

Handle collision with a collision basis

3 orthonormal vectors

- x: collision normal
- y, z: Perpendicular to x, define the plane of the contact



Calculating the contact basis

TECHNISCHE UNIVERSITÄT DARMSTADT

x: Contact normal

y: Choose a vector perpendicular to x

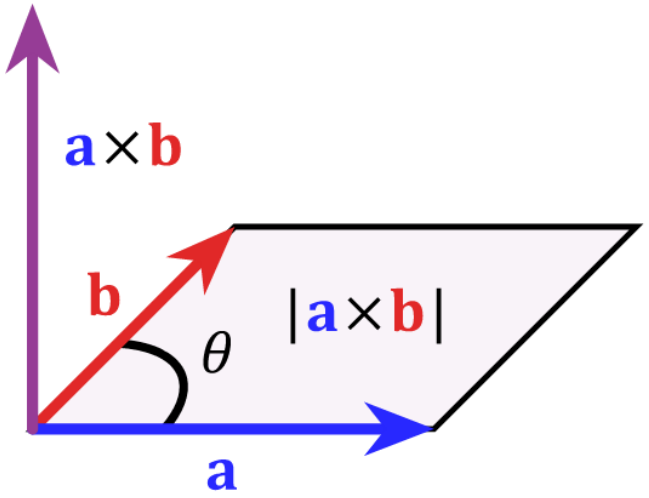
Cross product: A x B is perpendicular to A and B (unless they are parallel)

Use an axis, e.g. global z

y = x x (0, 0, 1)

Choose third vector to be perpendicular to x and y

z = x x y



Velocity resolution

TECHNISCHE UNIVERSITÄT DARMSTADT

Find contact basis

Calculate the change in velocity of the contact paint per unit impulse

Invert this to get a way to counter velocities

Calculate the x-term of the impulse (along the collision normal – our old calculation)

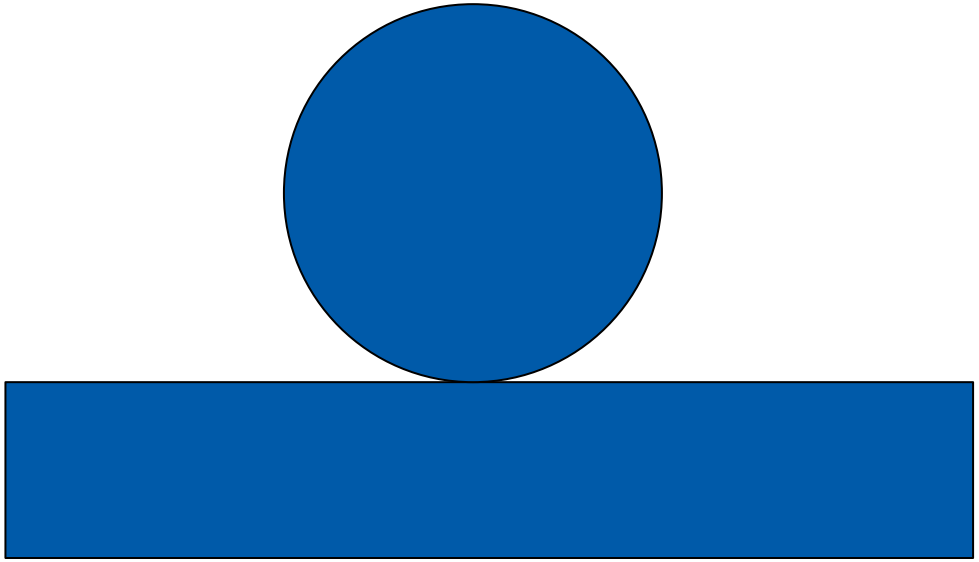
Calculate the y and z-terms of the impulse (for friction)

Apply the impulse

Velocity resolution

TECHNISCHE UNIVERSITÄT DARMSTADT

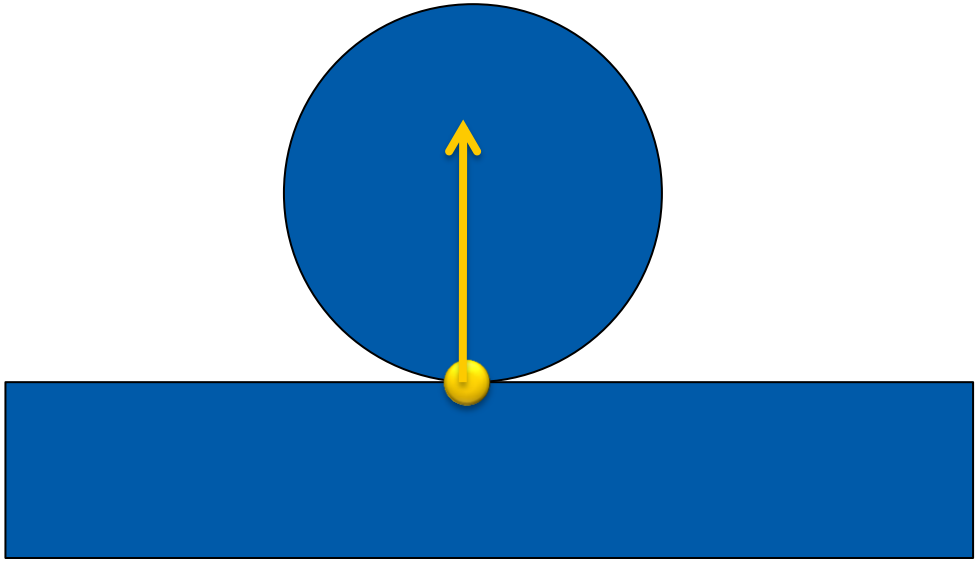
Find the collision collision normal and point of collision



Velocity resolution

TECHNISCHE UNIVERSITÄT DARMSTADT

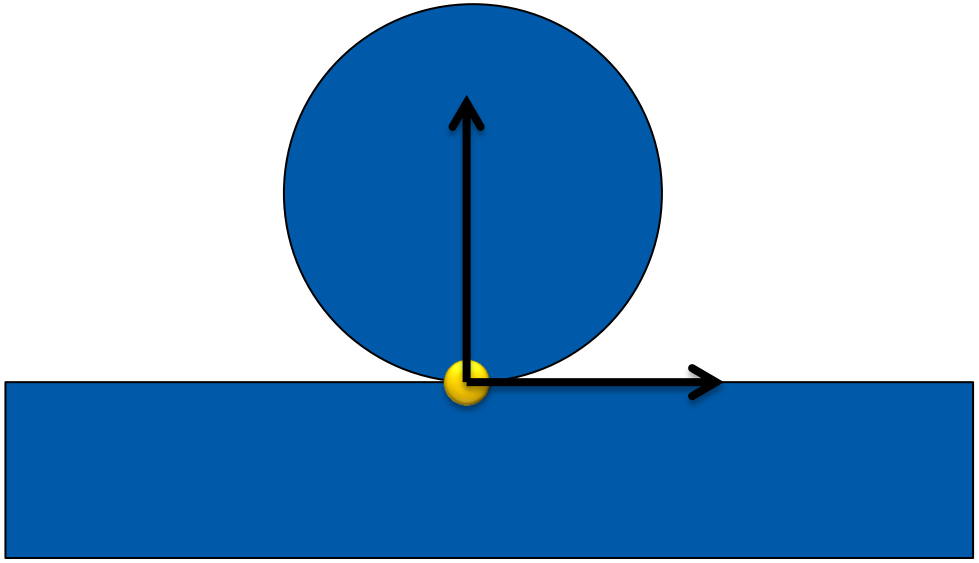
Find the collision collision normal and point of collision



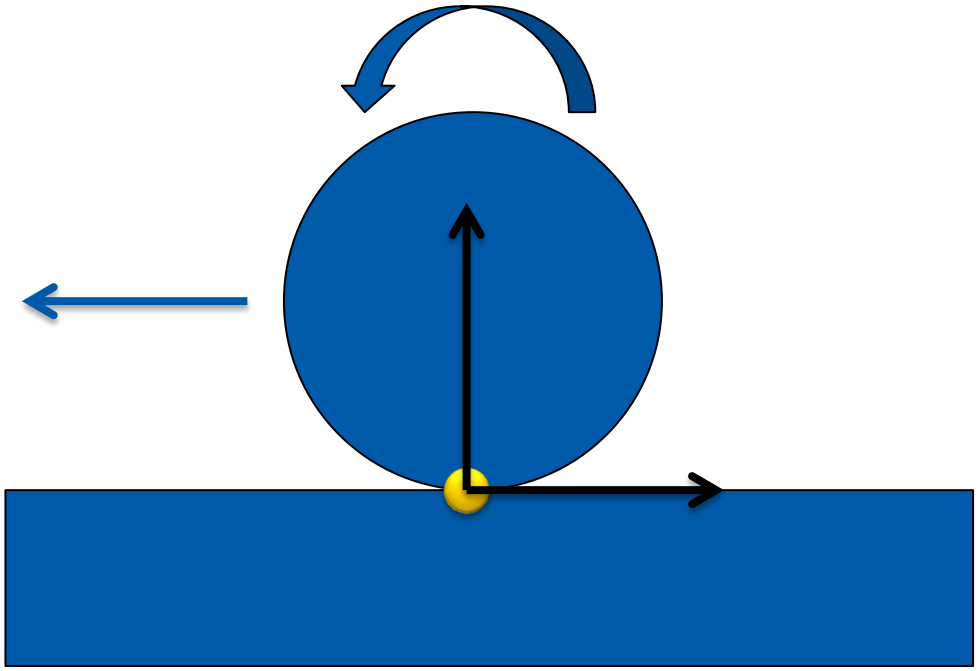
Velocity resolution

TECHNISCHE UNIVERSITÄT DARMSTADT

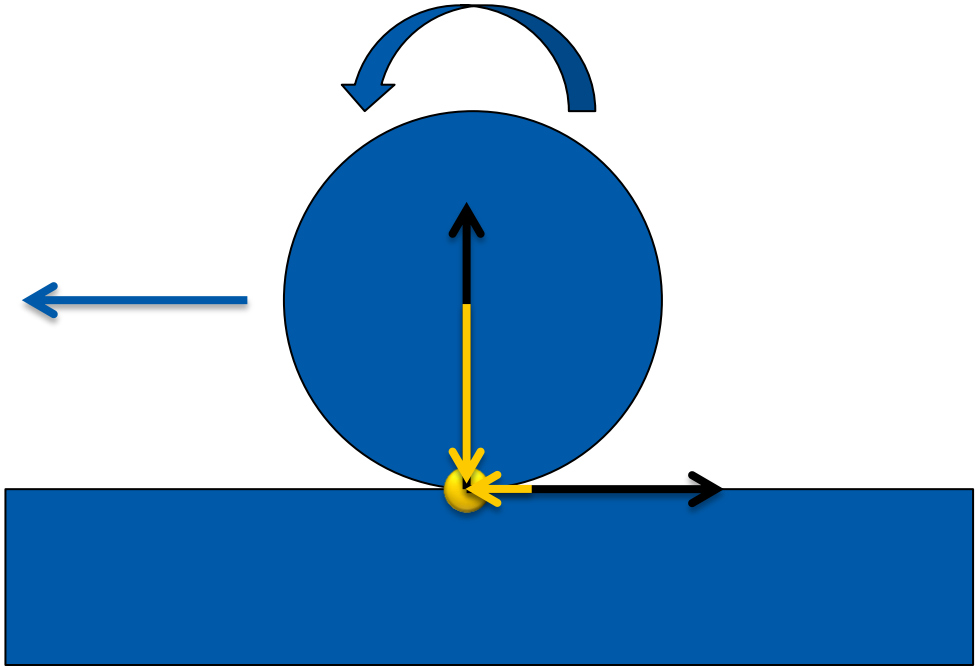
Find the collision basis



Identify the velocity of the collision point



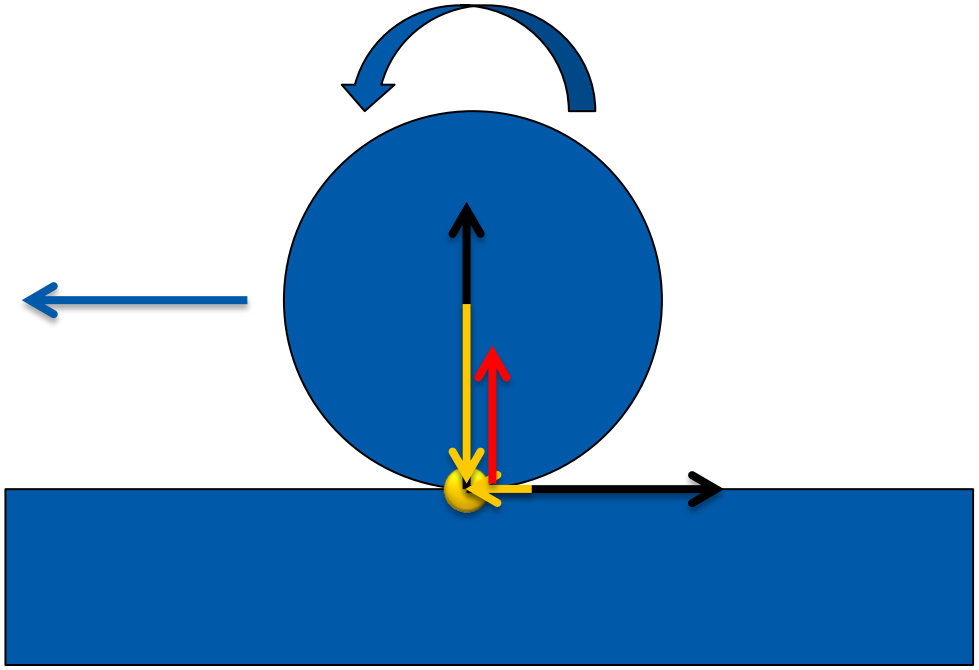
Map velocity into the collision basis



X-Axis (= collision Normal)

TECHNISCHE UNIVERSITÄT DARMSTADT

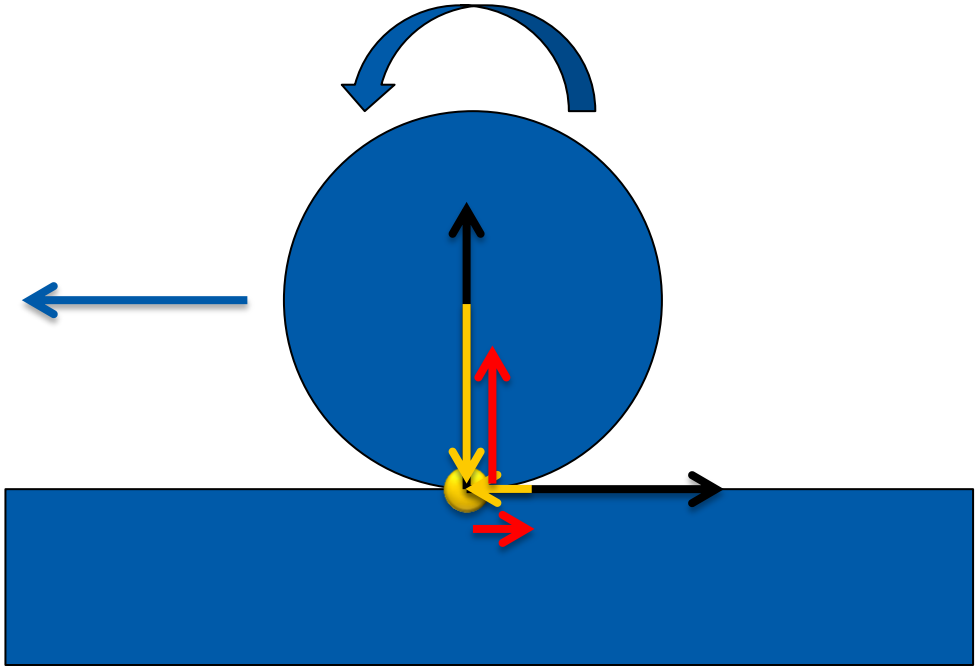
Separate the objects (what we did last lecture)



Y and Z-Axis

TECHNISCHE UNIVERSITÄT DARMSTADT

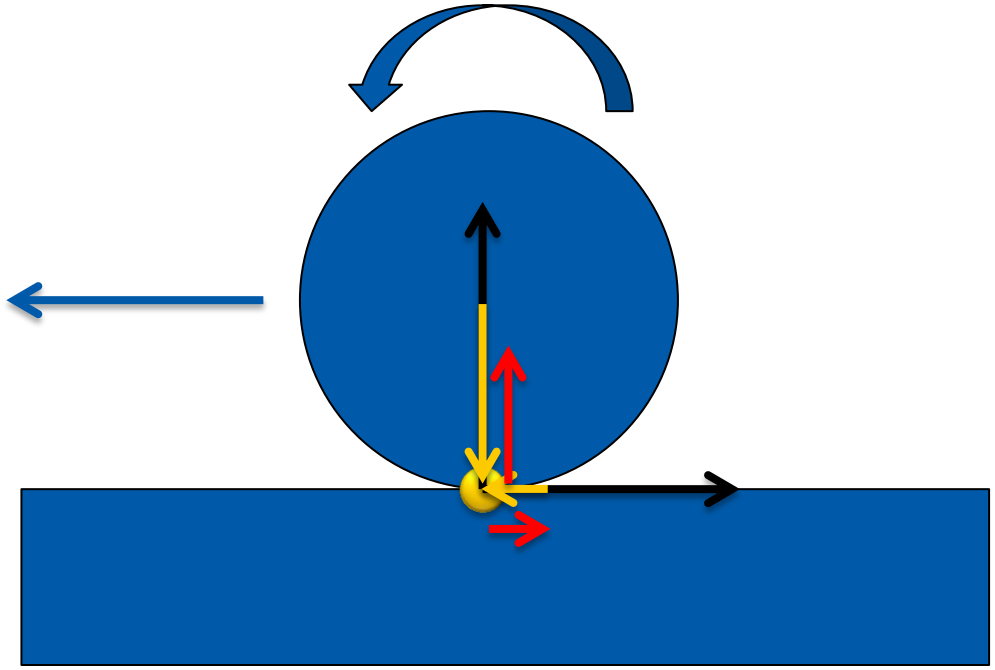
Handle Friction



Apply changes as impulses

TECHNISCHE UNIVERSITÄT DARMSTADT

Changes to velocity and rotation

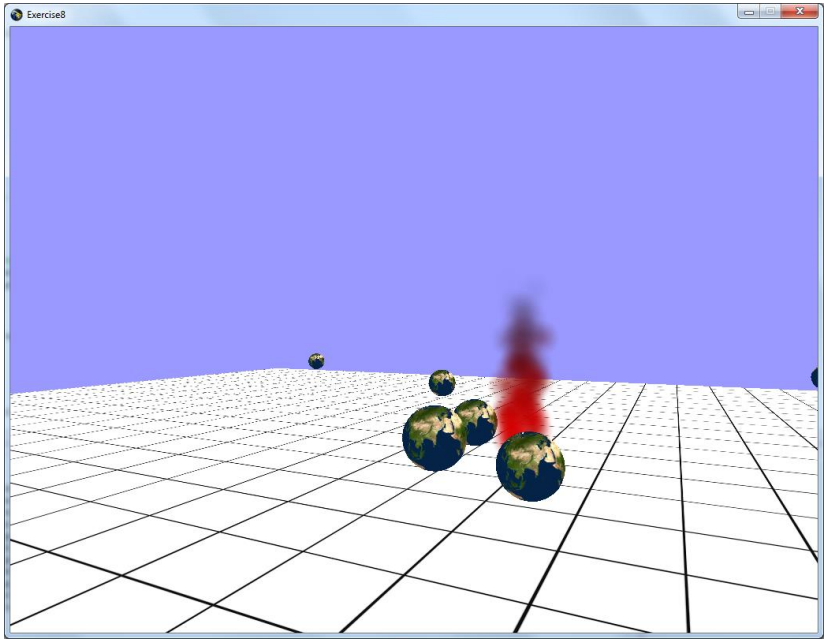


Exercise 8

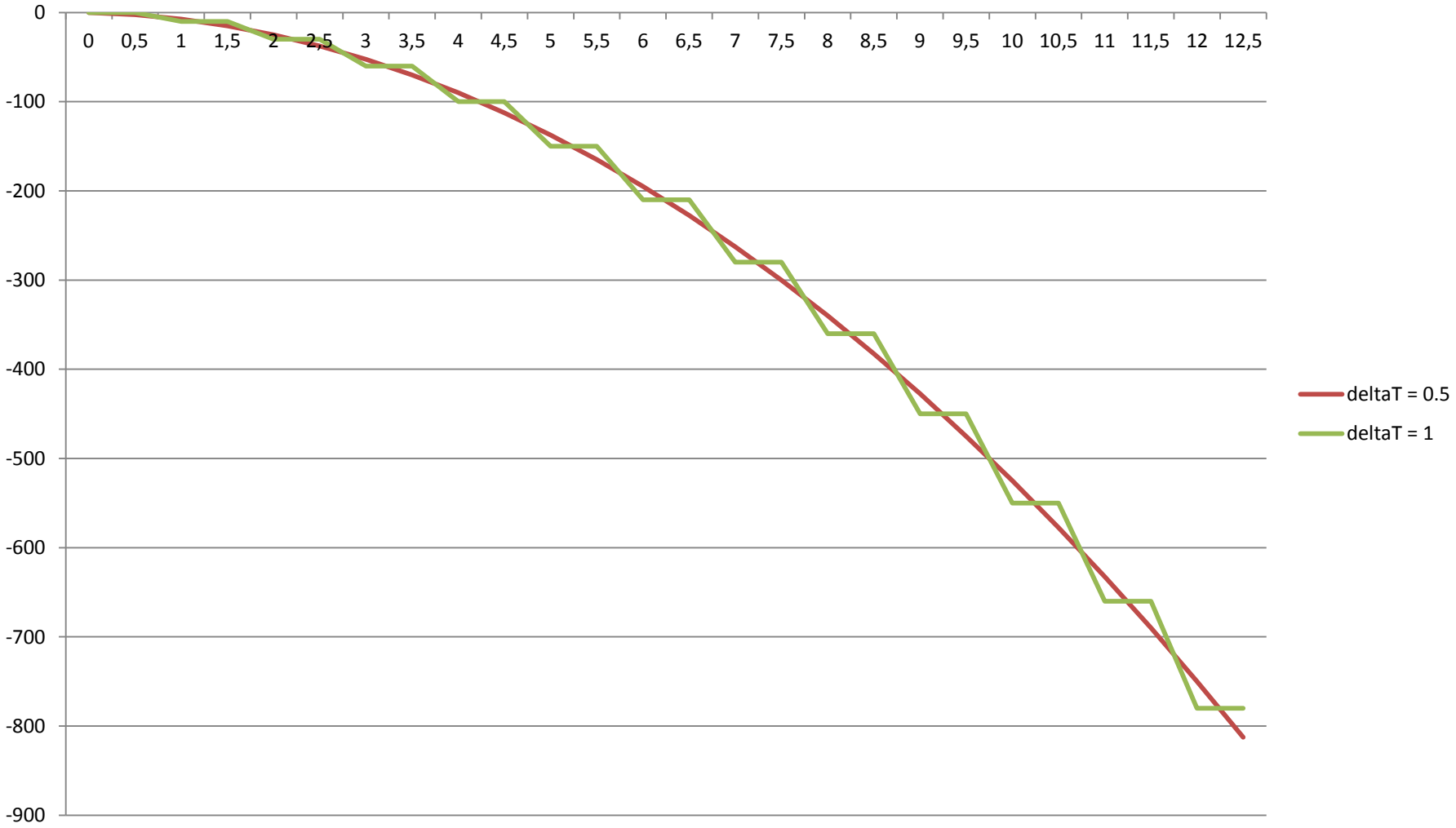
TECHNISCHE UNIVERSITÄT DARMSTADT

Practical Exercise

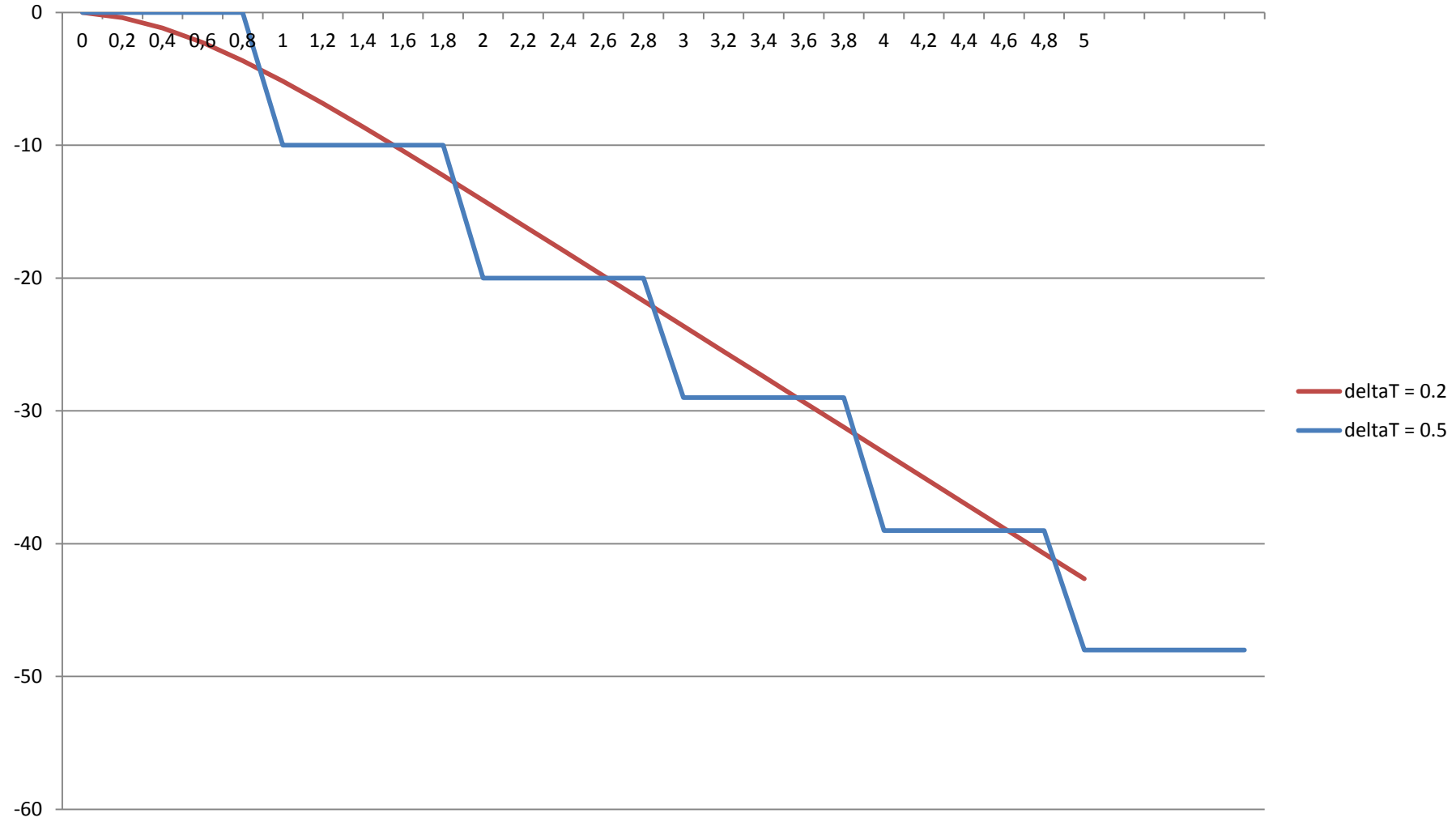
- Solution will be uploaded



Exercise 8 2.1



Exercise 8 2.2



Exercise 8 2.3

TECHNISCHE UNIVERSITÄT DARMSTADT

Build a matrix for billboarding that keeps the y-axis in place

- Using only the view matrix is a large restriction
- Easier to construct with knowledge of the object's position

Structure of the view matrix's rotation part

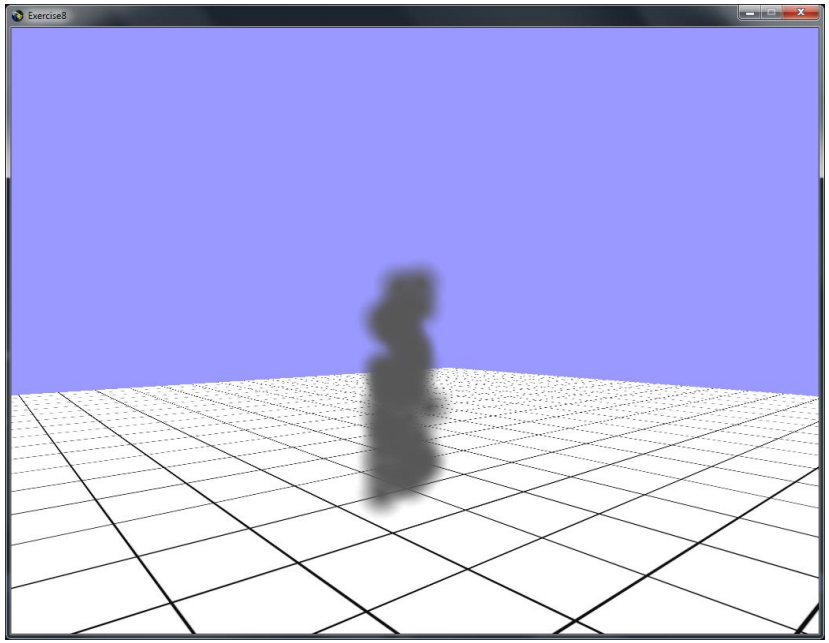
$$\begin{pmatrix} \boldsymbol{right} & \boldsymbol{up} & \boldsymbol{lookat} \\ \boldsymbol{right} & \boldsymbol{up} & \boldsymbol{lookat} \\ \boldsymbol{right} & \boldsymbol{up} & \boldsymbol{lookat} \end{pmatrix}$$

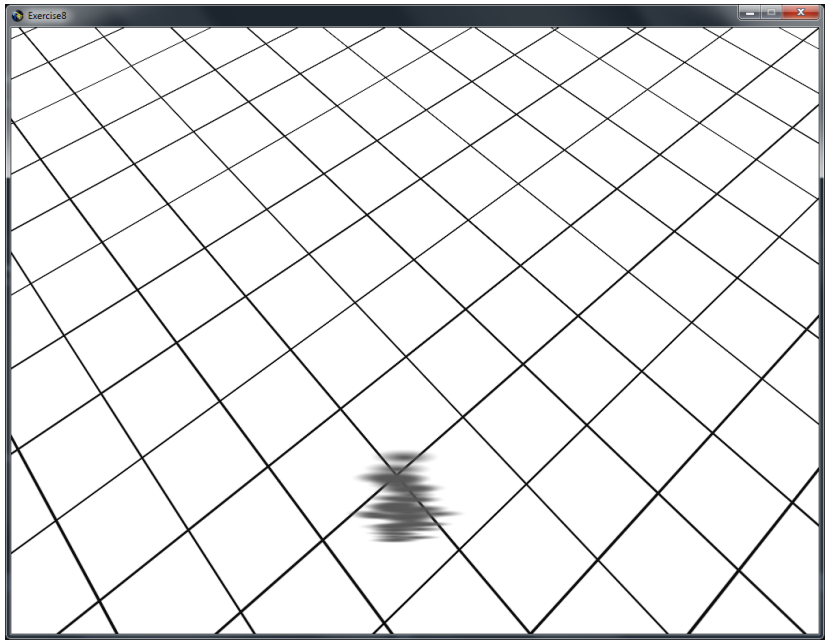
Change to

$$\begin{pmatrix} \mathbf{1} & \boldsymbol{up} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{up} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{up} & \mathbf{1} \end{pmatrix}$$

See also
http://www.lighthouse3d.com/opengl/billboarding/index.php3?billInt

Exercise 8 2.3





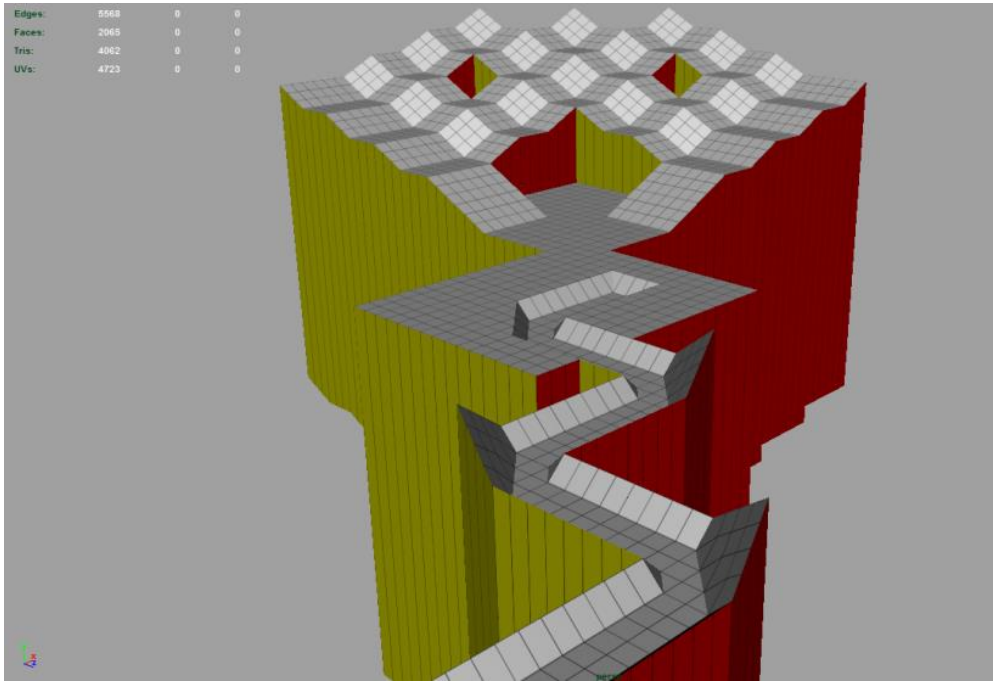
Exercise 9 – Practical Tasks

TECHNISCHE UNIVERSITÄT DARMSTADT

1.1 Triangle-Sphere-Intersection – Check one of the SAT axes

1.2 Construct the collision basis

1.3 Implement a goal area – See task 2.1



Exercise 9 – Theoretical Task

TECHNISCHE UNIVERSITÄT DARMSTADT

Build the foundation for task 1.3

Describe a box-sphere intersection test

Either

- Derive your own and describe it (does not have to be optimized)
- Describe one from the literature (name your source)

Questions & Contact

TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Electrical Engineering
and Information Technology
Multimedia Communications Lab - KOM

TECHNISCHE UNIVERSITÄT DARMSTADT

Dr.-Ing. Florian Mehm

Florian.Mehm@KOM.tu-darmstadt.de
Rundeturmstr. 10
64283 Darmstadt
Germany

Phone +49 (0) 6151/166885
Fax +49 (0) 6151/166152
www.kom.tu-darmstadt.de

game-technology@kom.tu-darmstadt.de