


## *Proximity Queries*

Simulation in Computer Graphics  
University of Freiburg

WS 04/05


## *Acknowledgements*



- parts of this slide set are courtesy of Bruno Heidelberger, ETH Zurich
- parts of this slide set are based on G. van den Bergen, "Collision Detection in Interactive 3D Environments," Elsevier, Amsterdam, ISBN: 1-55860-801-X, 2004.

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory


## *Outline*



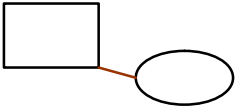
- introduction
- Minkowski sum
- distance computation  
Gilbert-Johnson-Keerthi algorithm (GJK)
- penetration depth computation  
expanding-polytope algorithm (EPA)
- approximate distance
- approximate consistent penetration depth
- demos

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

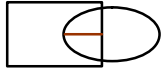
## *Proximity Query*



- for a pair of objects
  - compute their distance  
(find a pair of closest points)
  - compute their penetration depth  
(minimal translation to separate two interfering objects)



distance



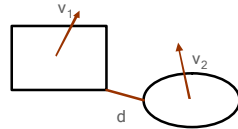
penetration depth

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

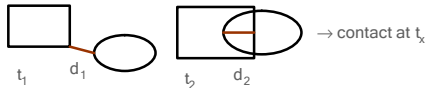
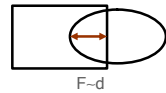
## Application



- distance
  - collision candidates
  - continuous collision detection



- penetration depth
  - penalty-based collision response
  - computation of time of contact



University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Outline



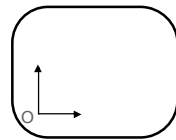
- introduction
- Minkowski sum
- distance computation
  - Gilbert-Johnson-Keerthi algorithm (GJK)
- penetration depth computation
  - expanding-polytope algorithm (EPA)
- approximate distance
- approximate consistent penetration depth
- demos

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Minkowski Addition



- A
- B
- $A + B = \{x + y : x \in A, y \in B\}$
- $(A + t_1) + (B + t_2) = (A + B) + t_1 + t_2$
- representation of swept objects

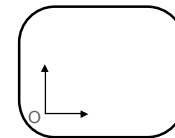


University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Configuration Space Obstacle



- A
- B
- $CSO(A, B) = A - B = A + (-B) = \{x - y : x \in A, y \in B\}$
- to realize A-B, the reflection of B is added to A



University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

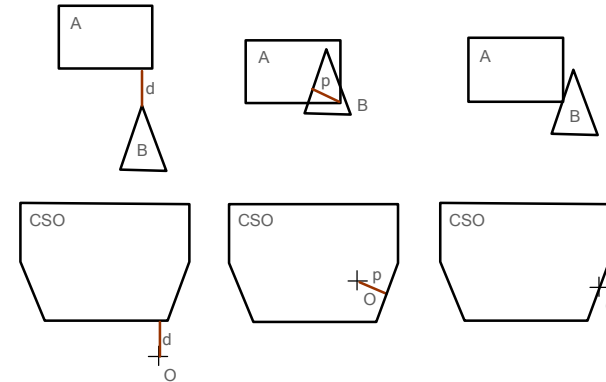
## CSO and Proximity Queries



- iff A and B intersect, they have a common point  $x_1 = y_1$  with  $x_1 - y_1 = 0$
- $\rightarrow O \in \text{CSO}(A,B)$  iff A and B intersect
- d (A,B) distance between A and B  
 $d(A,B) = \min \{ \|x - y\| : x \in A, y \in B \}$
- p (A,B) penetration depth of A and B  
 $p(A,B) = \inf \{ \|x\| : x \notin \text{CSO}(A,B) \}$

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Proximity Queries - Examples



University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Convex Objects



- if A and B are convex, then  $A+B$  and  $\text{CSO}(A,B)$  are convex
- proof:
  - let  $w_1 = x_1 + y_1, w_2 = x_2 + y_2, x_1, x_2 \in A, y_1, y_2 \in B, w_1, w_2 \in A+B$
  - $A+B$  is convex iff  $\lambda_1 w_1 + \lambda_2 w_2 \in A+B, \lambda_1 + \lambda_2 = 1, \lambda_1, \lambda_2 \geq 0$
  - A is convex  $\Rightarrow \lambda_1 x_1 + \lambda_2 x_2 \in A$
  - B is convex  $\Rightarrow \lambda_1 y_1 + \lambda_2 y_2 \in B$
  - $\lambda_1 x_1 + \lambda_2 x_2 + \lambda_1 y_1 + \lambda_2 y_2 = \lambda_1(x_1 + y_1) + \lambda_2(x_2 + y_2) = \lambda_1 w_1 + \lambda_2 w_2$
  - $\Rightarrow \lambda_1 w_1 + \lambda_2 w_2 \in A+B$
  - $\Rightarrow A+B$  is convex
- important for computing proximity queries on CSOs for convex objects

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Convex Polytopes



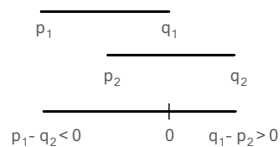
- A and B are polytopes, e. g. closed triangulated surfaces
- $\text{conv}(A)$  - convex hull of A
- $\text{vert}(A)$  - set of vertices of A
- $A+B = \text{conv}(\text{vert}(A) + \text{vert}(B))$
- computing the convex hull for all pair wise sums of vertices of A and B gives the Minkowski sum of A and B
- important for computing  $A+B$  for convex polytopes

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Proximity Queries - AABBs



- axis-aligned boxes  $A = [p_1, q_1]$ ,  $B = [p_2, q_2]$
- $CSO(A, B) = [p_1, q_1] - [p_2, q_2] = [p_1 - q_2, q_1 - p_2]$
- A and B intersect iff  $O \in [p_1 - q_2, q_1 - p_2]$
- intersecting AABBs in 1D



University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Proximity Queries - AABBs



- axis-aligned boxes  
 $A = [c_1 - h_1, c_1 + h_1]$ ,  $B = [c_2 - h_2, c_2 + h_2]$ ,  $h_1, h_2 > 0$
- $CSO(A, B) = [c_1 - c_2 - (h_1 + h_2), c_1 - c_2 + (h_1 + h_2)]$
- $O \in CSO(A, B)$  iff  $|c_1 - c_2| < h_1 + h_2$   
(see BVH slides)
- intersection test for spheres can be derived in a similar way

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Summary



- Minkowski sum or configuration space obstacle CSO can be used for proximity queries
- if origin is not contained in CSO, then the distance of two objects is given by the distance of the CSO to the origin
- if origin is contained in CSO, the penetration depth is given by the distance of the CSO to the origin
- useful characteristics for CSO of convex polytopes
- intersection tests for AABBs and other basic primitives can be derived from CSO

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Outline



- introduction
- Minkowski sum
- distance computation  
Gilbert-Johnson-Keerthi algorithm (GJK)
- penetration depth computation  
expanding-polytope algorithm (EPA)
- approximate distance
- approximate consistent penetration depth
- demos

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Overview



- for a given convex polytope  $C$  with  $O \notin C$ , GJK computes the point  $v(C)$  closest to the origin  $O$
- $\|v(C)\| = \min(\|x\| : x \in C)$
- iff  $C = \text{CSO}(A, B)$ , then GJK computes the distance  $d(A, B)$  of two non-intersecting convex objects  $A$  and  $B$
- $d(A, B) = \|v(\text{CSO}(A, B))\|$

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

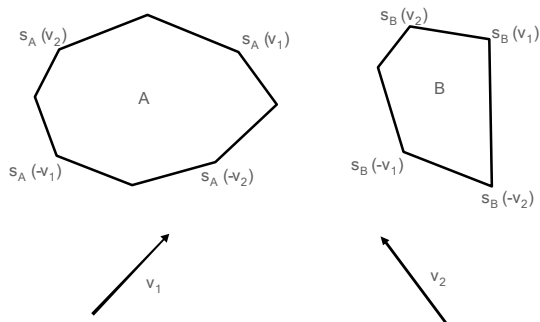
## Support Mapping



- A support mapping of a polytope  $A$  is a function  $s_A$  that maps a vector  $v$  to a vertex of  $A$ .
- $s_A(v) \in \text{vert}(A)$  with  $v \cdot s_A(v) = \max(v \cdot a : a \in \text{vert}(A))$
- The vertex  $s_A(v)$  is the support point of  $A$  with respect to  $v$ .

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Support Mapping - Example



University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Support Mapping for Convex Polytopes



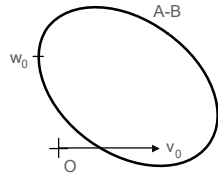
- represent the convex polytope as an adjacency graph
- start with an initial guess
- “climb the hill” by searching the adjacency graph for better solutions  $\Rightarrow$  hill climbing
- $p$  = cached support vertex
- repeat
  - optimal = true
  - for  $q \in \text{adj}(p)$  do
    - if  $v \cdot q > v \cdot p$  then  $\{ p = q, \text{optimal} = \text{false} \}$
- until optimal

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## GJK Initialization – Step 0



- iterative approximation of  $d(A, B)$
- GJK starts with an arbitrary  $v_0 \in A - B$  and a set of vertices  $W_0 = \emptyset$



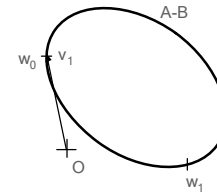
- $W_0 = S_{A-B}(-v_0)$

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Step 1



- $v_1 = v(\text{conv}(W_0 \cup \{w_0\})) = v(\text{conv}(w_0))$
- $w_1 = S_{A-B}(-v_1)$
- $W_1 =$  “smallest”  $X$  with  $X \subseteq W_0 \cup \{w_0\}$  such that  $v_1 \in \text{conv}(X)$
- $W_1 = \{w_0\}$

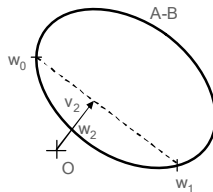


University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Step 2



- $v_2 = v(\text{conv}(W_1 \cup \{w_1\})) = v(\text{conv}(w_0, w_1))$
- $w_2 = S_{A-B}(-v_2)$
- $W_2 =$  “smallest”  $X$  with  $X \subseteq W_1 \cup \{w_1\}$  such that  $v_2 \in \text{conv}(X)$
- $W_2 = \{w_0, w_1\}$

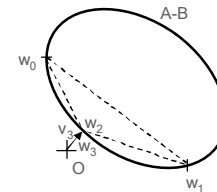


University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Step 3



- $v_3 = v(\text{conv}(W_2 \cup \{w_2\})) = v(\text{conv}(w_0, w_1, w_2))$
- $w_3 = S_{A-B}(-v_3)$
- $W_3 =$  “smallest”  $X$  with  $X \subseteq W_2 \cup \{w_2\}$  such that  $v_3 \in \text{conv}(X)$
- $W_3 = \{w_2\}$



University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

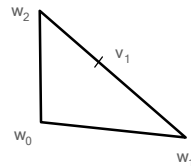
## “smallest” $X$



- $v_1 = v(\text{conv}(w_0, w_1, w_2))$   $X = \{w_0, w_1, w_2\}$
- $v_1 = \lambda_0 w_0 + \lambda_1 w_1 + \lambda_2 w_2$  with  $\lambda_0 + \lambda_1 + \lambda_2 = 1, \lambda_0, \lambda_1, \lambda_2 \geq 0$
- if  $\lambda_i = 0$  then the corresponding  $w_i$  can be removed from  $X$  such that  $v_1 = v(\text{conv}(X))$

### example:

- $v_1 = \lambda_0 w_1 + \lambda_1 w_2$
- $\Rightarrow v_1 = v(\text{conv}(w_1, w_2))$
- $\Rightarrow X = \{w_1, w_2\}$



University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## GJK Algorithm



- $v =$  arbitrary point in  $A - B$
- $W = \emptyset$
- $w = s_{A-B}(-v)$
- while  $v$  not close enough to  $v(A-B)$ 
  - $v = v(\text{conv}(W \cup \{w\}))$
  - $W =$  smallest  $X \subseteq W \cup \{w\}$  such that  $v \in \text{conv}(X)$
  - $w = s_{A-B}(-v)$
- return  $\|v\|$

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Convergence and Termination



- $\|v_{k+1}\| \leq \|v_k\|$
- if  $\|v_{k+1}\| = \|v_k\|$  then  $v_k = v(A-B)$
- for polytopes, GJK computes  $v_k = v(A-B)$  in a finite number of iterations
- for non-polytopes, the error of  $\|v_k\|$  is bound by  $\|v_k - v(A-B)\|^2 \leq \|v_k\|^2 - v_k \cdot w_k$

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Summary



- GJK computes the distance of two non-intersecting objects
- iterative process
- main loop performs three steps on a simplex
  - computation of the distance of the simplex to the origin
  - support mapping based on this distance
  - adaptation of the simplex based on the support point
- GJK converges to the correct solution
- GJK computes the distance in a finite number of iterations for polytopes

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Outline



- introduction
- Minkowski sum
- distance computation  
Gilbert-Johnson-Keerthi algorithm (GJK)
- penetration depth computation  
expanding-polytope algorithm (EPA)
- approximate distance
- approximate consistent penetration depth
- demos

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Introduction



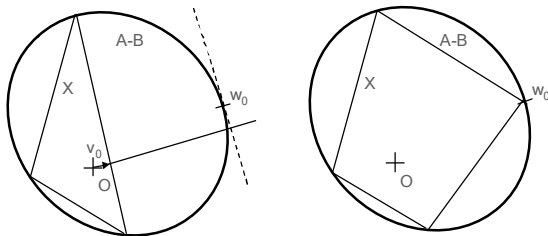
- EPA computes the penetration depth of two objects
- iterative process
- works with an CSO that contains the origin
- starts with a simplex (triangle in 2D, tetrahedron in 3D) that contains the origin and whose vertices are on the boundary of the CSO
- the initial simplex is subdivided (expanded) by EPA to approximate the CSO
- the distance of the expanded polytope to the origin corresponds to the penetration depth

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Step 0



- $v_0 = v(X)$
- $w_0 = s_{A-B}(v_0)$
- expand  $X$  such that it contains  $w_0$

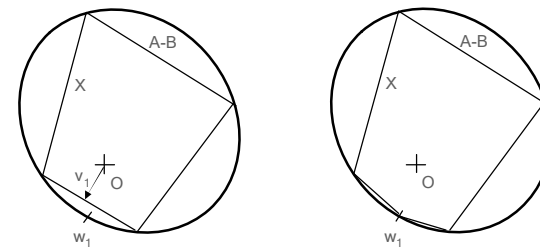


University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Step 1



- $v_1 = v(X)$
- $w_1 = s_{A-B}(v_1)$
- expand  $X$  such that it contains  $w_1$

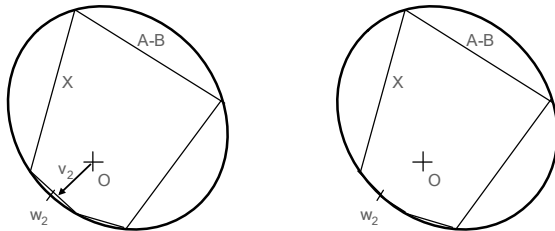


University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Step 2



- $v_2 = v(X)$
- $w_2 = s_{A-B}(v_2)$
- expand  $X$  such that it contains  $w_2$



University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Convergence and Termination



- $\|v_{k+1}\| \geq \|v_k\|$
- for polytopes, EPA computes  $v_k = v(A-B)$  in a finite number of iterations

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Outline



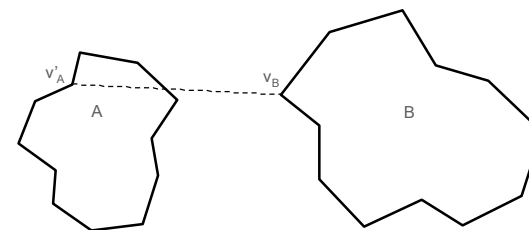
- introduction
- Minkowski sum
- distance computation  
Gilbert-Johnson-Keerthi algorithm (GJK)
- penetration depth computation  
expanding-polytope algorithm (EPA)
- approximate distance
- approximate consistent penetration depth
- demos

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Approximate Distance – Step 1



- two polytopes  $A$  and  $B$
- start with an arbitrary vertex  $v'_A$  with  $v'_A \in \text{vert}(A)$
- compute nearest vertex  $v_B$  with  $v_B \in \text{vert}(B)$

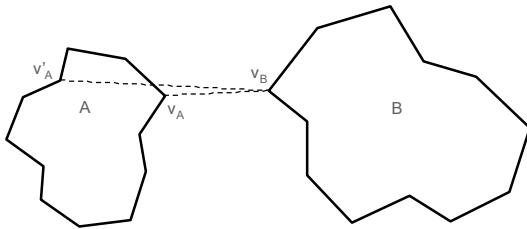


University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Approximate Distance – Step 2



- compute nearest vertex  $v_A \in \text{vert}(A)$  with respect to  $v_B$
- $\|v_A - v_B\|$  is the approximate distance of A and B

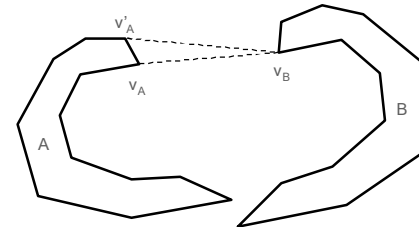


University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Characteristics



- better approximation for larger distances and convex objects
- bad approximation in case of concave objects



University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Outline



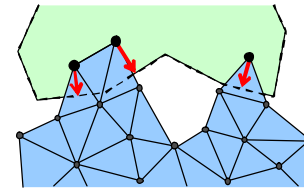
- introduction
- Minkowski sum
- distance computation  
Gilbert-Johnson-Keerthi algorithm (GJK)
- penetration depth computation  
expanding-polytope algorithm (EPA)
- approximate distance
- approximate consistent penetration depth
- demos

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Motivation



- compute consistent penetration depth information for all intersecting points of a tetrahedral mesh
- can be used to compute penalty forces which provide realistic collision response for deformable tetrahedral meshes

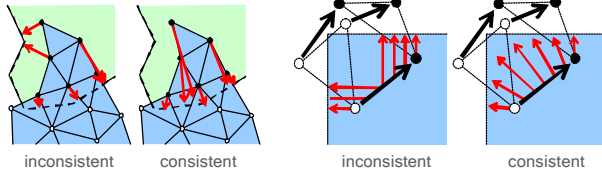


University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Challenges



- inconsistent penetration depth information due to discrete simulation steps and object discretization



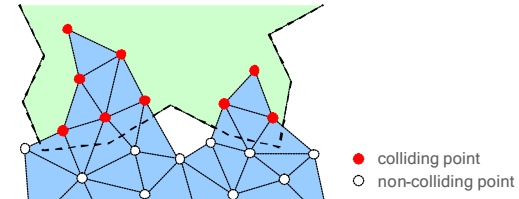
- inconsistent penetration depth results in oscillation artifacts or non-realistic collision response

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Algorithm – Stage 1



- object points are classified as colliding or non-colliding points → slides on spatial hashing



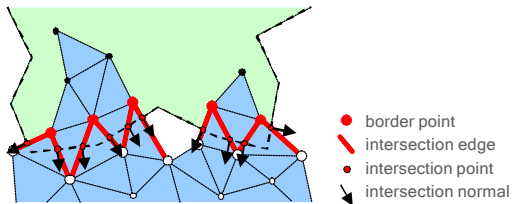
- colliding point
- non-colliding point

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Algorithm – Stage 2



- border points, intersecting edges, and intersection points are detected → extension of spatial hashing



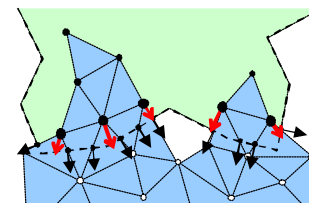
- border point
- intersection edge
- intersection point
- ↘ intersection normal

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Algorithm – Stage 3



- penetration depth  $d(p)$  of a border point  $p$  is approximated using the adjacent intersection points  $x_i$  and normals  $n_i$



$$\omega(x_i, p) = \frac{1}{\|x_i - p\|^2}$$

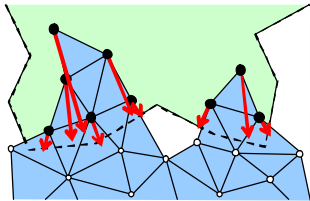
$$d(p) = \frac{\sum_{i=1}^k (\omega(x_i, p) \cdot (x_i - p) \cdot n_i)}{\sum_{i=1}^k \omega(x_i, p)}$$

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Algorithm – Stage 4



- consistent penetration depth information at points  $p_j$  is propagated to other colliding points  $p$



$$\mu(\mathbf{p}_j, \mathbf{p}) = \frac{1}{\|\mathbf{p}_j - \mathbf{p}\|^2}$$

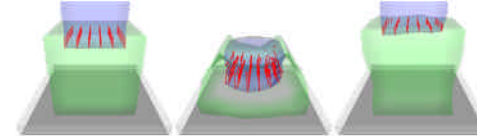
$$\frac{\sum_{j=1}^l (\mu(\mathbf{p}_j, \mathbf{p}) \cdot ((\mathbf{p}_j - \mathbf{p}) \cdot \mathbf{r}(\mathbf{p}_j) + d(\mathbf{p}_j)))}{\sum_{j=1}^l \mu(\mathbf{p}_j, \mathbf{p})}$$

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

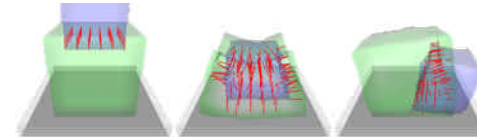
## Results



- consistent collision response



- inconsistent collision response



University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Results - Video



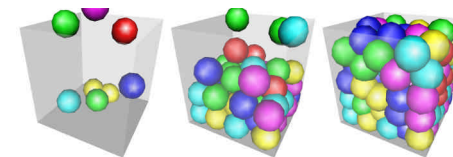
### Consistent Penetration Depth Estimation for Deformable Collision Response

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Summary



- consistent penetration depth information in case of
  - discrete object representation
  - discrete time simulation
- addresses the problem of discontinuities in magnitude and direction of the penetration depth
- provides realistic penalty-based collision response



University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Outline



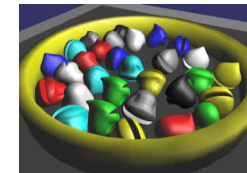
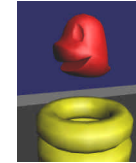
- introduction
- Minkowski sum
- distance computation  
Gilbert-Johnson-Keerthi algorithm (GJK)
- penetration depth computation  
expanding-polytope algorithm (EPA)
- approximate distance
- approximate consistent penetration depth
- demos

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## Interacting Deformable Objects



- deformable modeling based on constraints
- collision detection based on spatial hashing
- collision response based on consistent penetration depth computation



University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory

## References



- E. G. Gilbert, D. W. Johnson, S. S. Keerthi, "A Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space," IEEE Journal of Robotics and Automation, vol. 4, no. 2, pp. 193-203, 1988.
- G. van den Bergen, "Collision Detection in Interactive 3D Environments," Elsevier, Amsterdam, ISBN: 1-55860-801-X, 2004.
- B. Heidelberger, M. Teschner et al., "Consistent Penetration Depth Estimation for Deformable Collision Response," Proc. VMV, Stanford, USA, 2004.

University of Freiburg - Institute of Computer Science - Computer Graphics Laboratory