Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

ASSIGNMENT 3: BROAD-PHASE COLLISION DETECTION

Demo due 8nd Feb @ 3-4pm in LG37
Cumulative Submission (Report, Code and Video) due on 28nd Feb 2016 @ 23:59 via mymodule.tcd.ie

ASSIGNMENT 3: BROAD-PHASE COLLISION DETECTION

Demo due: 8nd February, 2016 @ 3pm – 4pm in LG37

There will be a 20% penalty for each day after this

- If late, **the onus is on you** to email me the code when you have finished and then we will arrange alternate time for demo. The date you email me the code will be taken as date of submission for purposes of the above penalty. It is expected that you will not make further modifications to the demo after this date.

This assignment is worth 10% of the module

Task: Implement the following broad-phase collision detection schemes and a demo that flags potentially colliding Rigid Bodies:

1. You MUST implement a Bounding sphere collision detection test

   AND

2. You MUST choose, read the related paper and implement ONE of the following:
   A. AABB with Sweep and Prune
   B. Hierarchical Spatial Subdivision
   C. OBB (note that you do not have to worry about how to fit the OBB automatically to an object for this assignment)

0. GENERAL REQUIREMENTS

Prerequisites: a Rigid Body System to update state of a large (> 10) collection of rigid bodies from previous lab. Apply simple linear and angular momentum to move the objects (to keep them within your scene you might reflect objects off edge of the scene)

Book keeping: Typically a list of potentially colliding pairs (the active list) is maintained. The broadphase culls trivially non-colliding cases before passing on to the narrow phase (next lab)

Visualisation: your demo should be able to pause the simulation and draw relevant bounding boxes/subdivisions (even though in practice this would be hidden), and flag intersecting objects e.g. by colouring the object or the bounding volume.

In addition each Broad-phase scheme will have the following components which you should implement (see following slides for details):

- Bounding Volume Fitting and update (insertion/binning for spatial subdivision)
- Intersection test (or proximity test)

1. BOUNDING SPHERE

Definition: the smallest sphere that encloses object

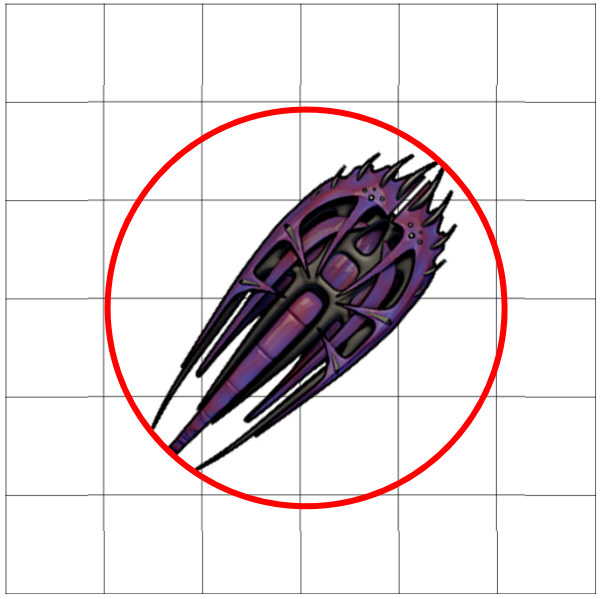
Representation: 4 floats (centre, radius)

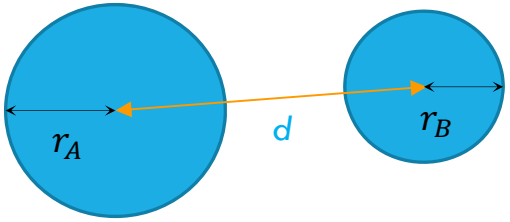
Creation: Find minimum enclosing sphere

- min+max (xyz); center; min radius

Collision check:

- Let $d = distance\ between\ centres\ of\ 2\ spheres$
- Then spheres are colliding $IF\ (d < (r_A + r_B))$





2(A). AXIS-ALIGNED BOUNDING BOX (AABB)

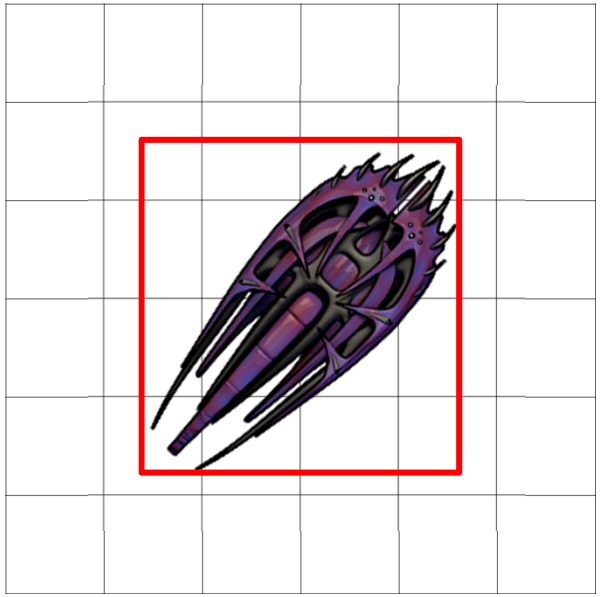
Definition: Box with edges that always align to the major axes of the coordinate system

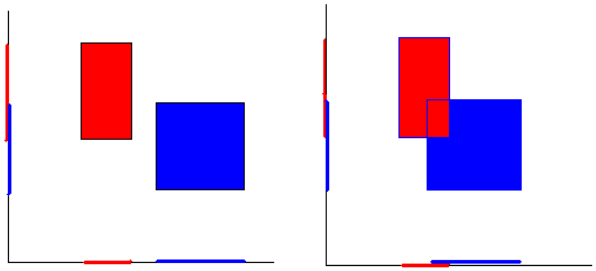
Representation: 6 floats (limits in each dimension)

Creation: Find min/max in x, y, z

Collision check:

- Three 1d interval overlap tests. Must overlap in all 3 directions
- Optimal solution is through sweep and prune (next slide)





Spaceship image: free sprite by millionthvector (http://millionthvector.blogspot.ie/2013/07/free-alien-top-down-spaceship-sprites.html)

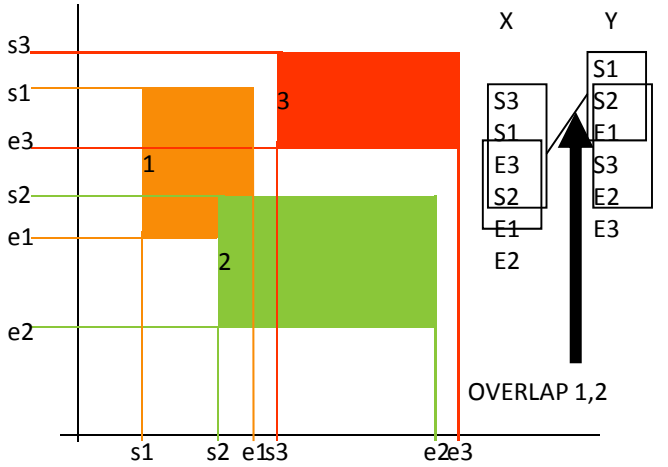
3(A). SWEEP AND PRUNE

For each axis,

- At each frame, create sorted list of start and end points of intervals for each box (use Insert Sort to exploit frame to frame coherency)

Traverse each list

- Each time a startpoint is reached insert into active list
- If endpoint is hit remove it from active list
- If 2 or more objects are active at the same time they overlap in the specified dimension
- Potential colliding pairs must overlap in all 3 lists



To keep track of confirmed overlaps increment a counter in a pairs list each time an overlap is found. 3 overlaps implies AABBs overlap. [N.B. this could be further optimised]

	1	2	3	4	5
6	X X				
5	X		X		
4		X			
3		X			
2					

Pairs table

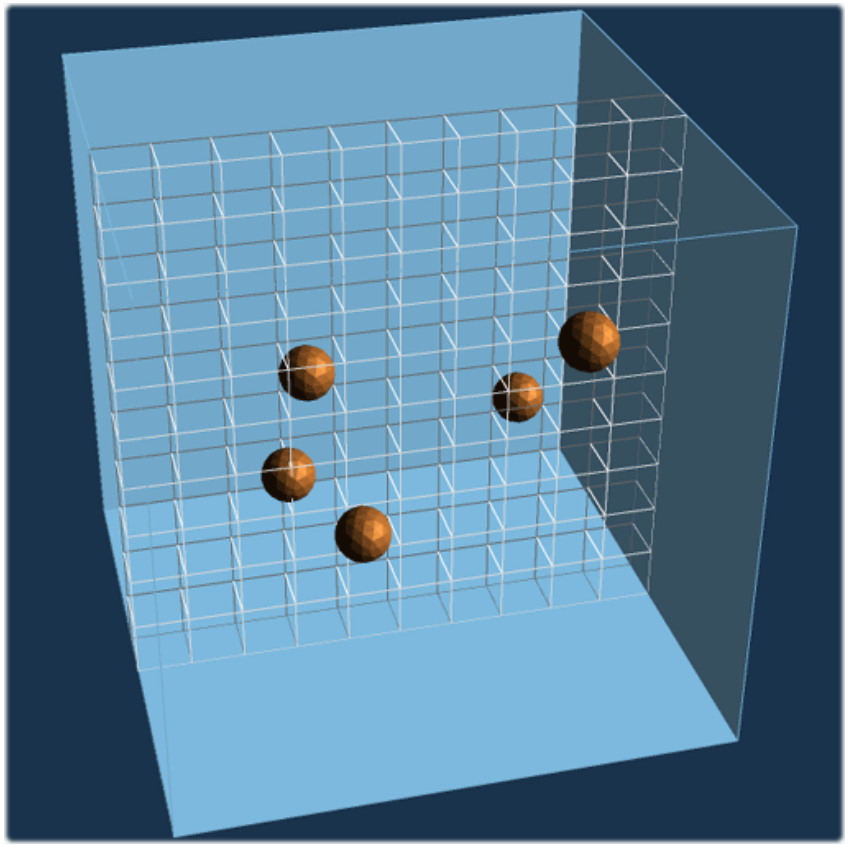
Required Reading [if you choose this option]: Cohen, J. D., Lin, M. C., Manocha, D., and Ponamgi, M. I-COLLIDE: an interactive and exact collision detection system for large-scale environments. In i3DG 1995 (http://www.cs.jhu.edu/~cohen/Publications/icollide.pdf)

2(B). SPATIAL SUBDIVISION

Definition: scene is divided up into (typically) uniform cells.

Update step: For each object keep a record of grid cells that it overlaps with

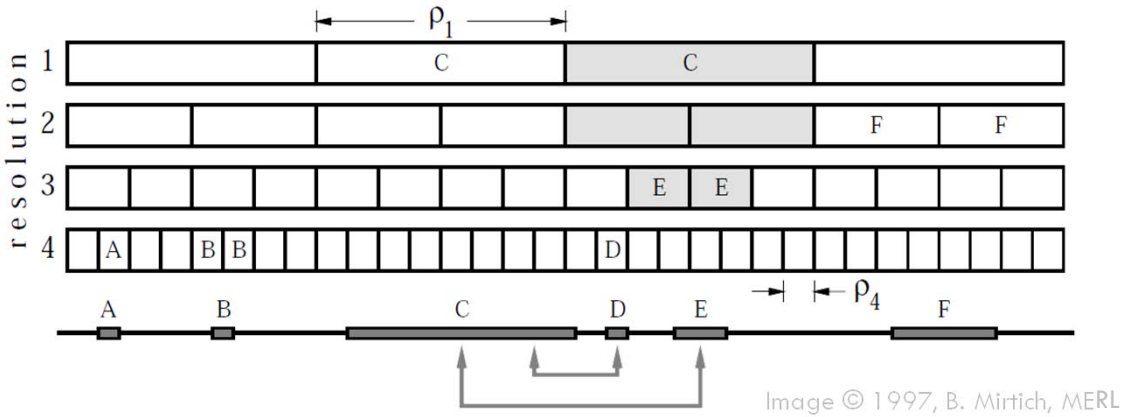
Collision test: Only perform pair-wise collision test with objects in neighbouring (or identical) grid cells



3(B). HIERARCHICAL SPATIAL SUBDIVISION

Adaptive Grid Sizes:

- Boxes are tiled at the minimum resolution where the cells are enough to cover the bounding volume of the object
- If res(X) and res(Y) are the tiling resolutions for X and Y, then Boxes X and Y are close iff they overlap a common cell at resolution min(res(X), res(Y))



Required Reading [if you choose this option]: Brian Mirtich, "Efficient Algorithms for Two-Phase Collision Detection"– TR9723 MERL, Dec, 1997. {**SEC 2. – 2.2 are relevant for this lab**}
http://www.merl.com/reports/docs/TR97-23.pdf

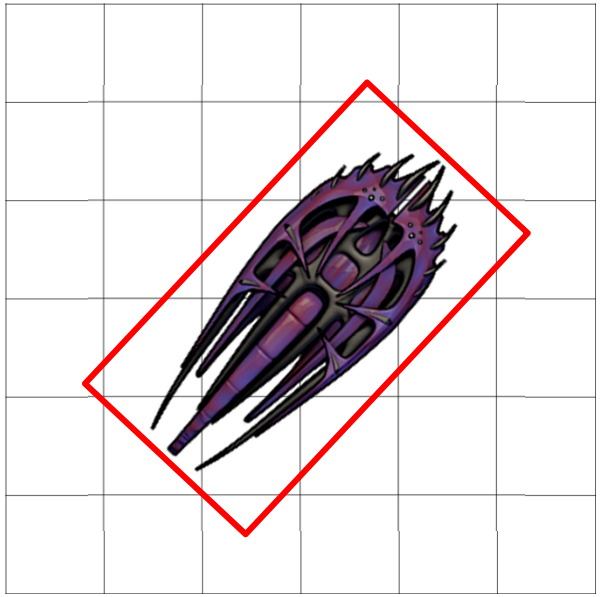
2(C). ORIENTED BOUNDING BOX (OBB)

Definition: Box with edges that align to object such that it fits optimally in terms of fill efficiency

Representation: 15 floats (position, orientation, extents)

Creation: For this lab you may simply fit this manually. Typically this is done based on Principal components analysis but you are not required to implement optimized OBB fitting for this lab

Collision test: Use separating axis theorem (next slide)



Required Reading [if you choose this option]: Gottshalk, S., Lin, M., Manocha, D. OBBTree: A Hierarchical Structure for Rapid Interference Detection. Siggraph 1996**.** (http://www.cs.unc.edu/~walk/papers/gottscha/sig96.pdf)

2(C). SEPARATING AXIS TEST (IN ONE SLIDE)

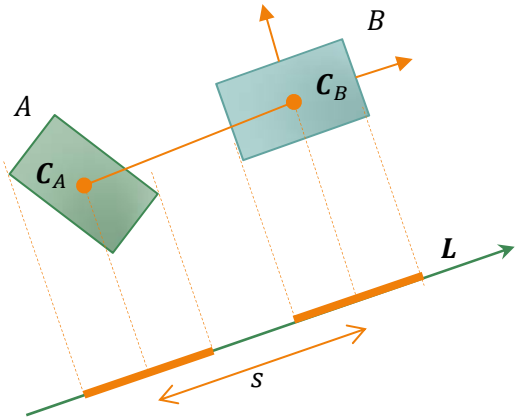
Determine Candidate Axes → Find Midpoint Separation → Find Half-length of projected intervals

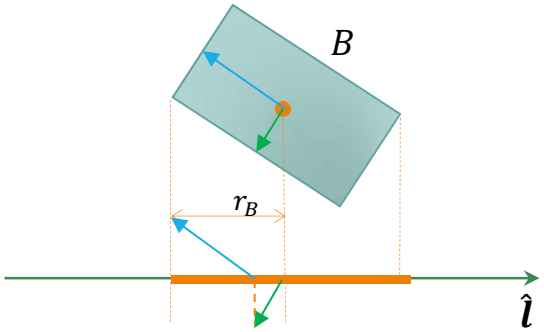
Face normal directions

$$\boldsymbol{u_0^A} \quad \boldsymbol{u_1^A} \quad \boldsymbol{u_2^A}$$
$$\boldsymbol{u_0^B} \quad \boldsymbol{u_1^B} \quad \boldsymbol{u_2^B}$$

Cross product of edge directions (for boxes these are orthogonal to face normals)

$$\boldsymbol{u_0^A \times u_0^B} \quad \boldsymbol{u_0^A \times u_1^B} \quad \boldsymbol{u_0^A \times u_2^B}$$
$$\boldsymbol{u_1^A \times u_0^B} \quad \boldsymbol{u_1^A \times u_1^B} \quad \boldsymbol{u_1^A \times u_2^B}$$
$$\boldsymbol{u_2^A \times u_0^B} \quad \boldsymbol{u_2^A \times u_1^B} \quad \boldsymbol{u_2^A \times u_2^B}$$

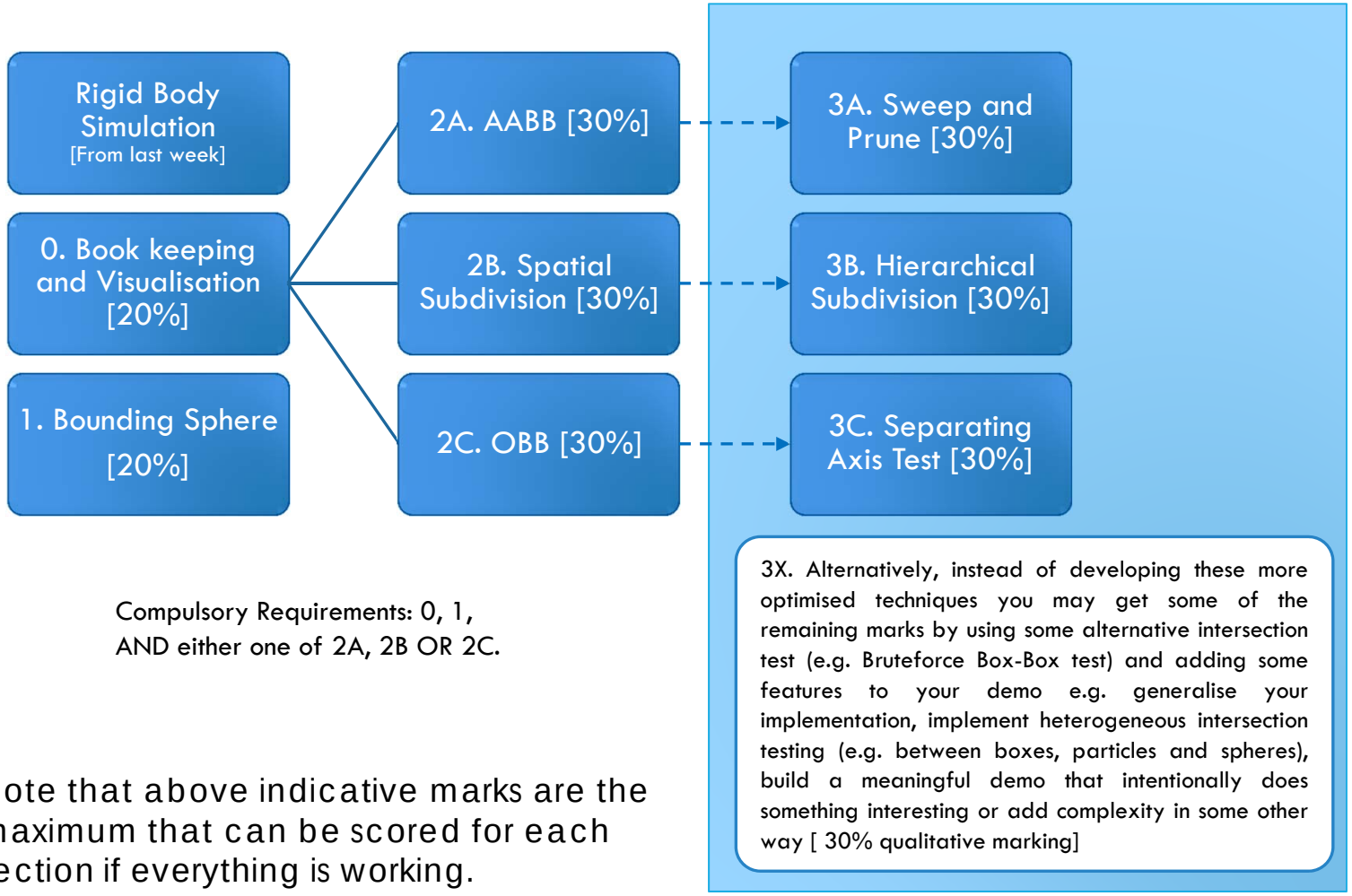




$$s = \left|(\boldsymbol{C_A} - \boldsymbol{C_B}) \cdot \hat{\boldsymbol{l}}\right|$$

$$r_B = e_1^B \left|\boldsymbol{u_1^B} \cdot \hat{\boldsymbol{l}}\right| + e_2^B \left|\boldsymbol{u_2^B} \cdot \hat{\boldsymbol{l}}\right| + e_3^B \left|\boldsymbol{u_3^B} \cdot \hat{\boldsymbol{l}}\right|$$

MARKING GUIDELINES (PLEASE NOTE OPTION 3X WAS ADDED)



Compulsory Requirements: 0, 1,
AND either one of 2A, 2B OR 2C.

Note that above indicative marks are the maximum that can be scored for each section if everything is working.

OTHER USEFUL RESOURCES

PLEASE NOTE: collision functions are ideally to be coded yourself for the learning experience in this lab assignment so the following sites are recommended for reference and for more generic math functions. Wherever code is directly used, you must acknowledge the original author.

Simple Intersection Tests for Games, Gamasutra article by Miguel Gomez:

- http://www.gamasutra.com/view/feature/131790/simple_intersection_tests_for_games.php

Geometric Tools (Wild Magic SDK by David Eberly):

- Free online resource with lots of (c++) code for various collision detection, math and physics functions including
  - PLEASE NOTE: collision functions are ideally to be coded yourself for the learning experience so this site is recommended as a reference and for more generic math functions. Wherever code is directly used, you must acknowledge the original author.
- http://www.geometrictools.com/LibPhysics/LibPhysics.html

Tristam MacDonald GAMEDEV.NET article on spatial hashing

- http://www.gamedev.net/page/resources/_/technical/game-programming/spatial-hashing-r2697

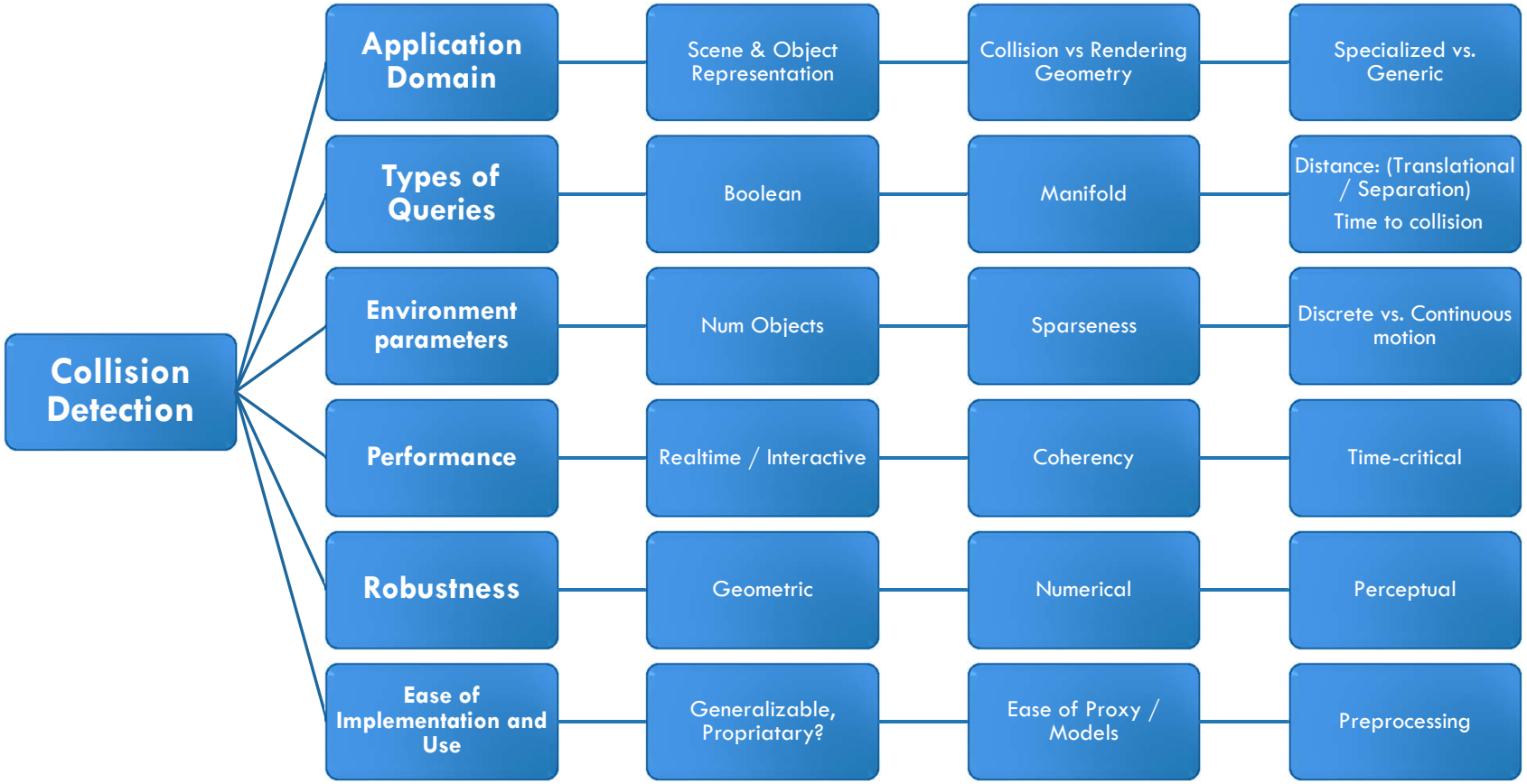
COMBINED REFERENCES

[Cohen] Cohen, J. D., Lin, M. C., Manocha, D., and Ponamgi, M. 1995. I-COLLIDE: an interactive and exact collision detection system for large-scale environments. In *Proceedings of the 1995 Symposium on interactive 3D Graphics*.
(http://www.cs.jhu.edu/~cohen/Publications/icollide.pdf)

[Gottshalk] Gottshalk, S., Lin, M., Manocha, D. OBBTree: A Hierarchical Structure for Rapid Interference Detection
(http://www.cs.unc.edu/~walk/papers/gottscha/sig96.pdf)

[Mirtich] Brian Mirtich – "Efficient Algorithms for Two-Phase Collision Detection" TR9723 MERL, Dec, 1997.
(http://www.merl.com/reports/docs/TR97-23.pdf)

ALGORITHM DESIGN ISSUES



CHALLENGES

From [Erricson05] Real-Time Collision Detection - By Christer Ericson. Elsevier, 2005.

Physics data complexity

- Multivariate, multi-dimensional
- Time-variant / transient
- Special cases / singularities

But require: Efficiency, Plausibility, Fidelity, Robustness … + relative ease of implementation

Tips for Debugging a Collision Detection System [Erricson]

- Keep a buffer of the last n collision queries: output data for further analysis
- Provide a means to visualise collision geometry
- Implement a simple reference algorithm: e.g. Brute force variant to isolate cause
- Maintain a test suite of queries; run code against different tests