dtecta

middleware solutions
for real-time 4D collision detection

SOLID 4D Collision Detection Library

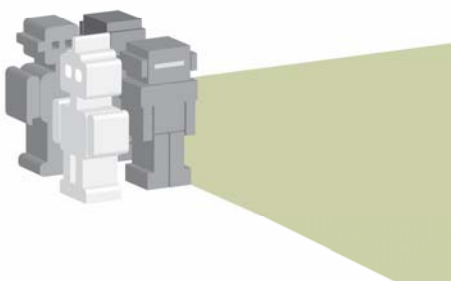
SOLID 4D is a software library for performing continuous collision detection of geometric objects in 3D space. Continuous collision detection is the process of detecting the first time of contact of moving objects. Besides continuous collision detection, SOLID 4D can also be used for static intersection tests and distance and penetration depth computation. A unique feature of SOLID 4D is the fact that it can perform all these queries on a wide variety of shape types, including: deformable triangle meshes, balls, boxes, cones, cylinders, ellipsoids, convex polyhedra, as well as Minkowski sums and convex hulls of arbitrary convex objects. Since it exploits temporal coherence in a number of ways, SOLID 4D is especially useful for detecting collisions between objects that move smoothly. The motions of objects are controlled by the client application, and are not determined or affected by SOLID 4D. Although it can be used for physics-based simulations, SOLID 4D is not a physics engine by itself. SOLID 4D leaves it up to the application programmer how the laws of physics are implemented.

SOLID 4D Application Areas

Implementing collision-detection routines for interactive 3D applications can be a daunting task, requiring advanced mathematical and programming skills. Furthermore, in real-time applications collision detection can easily take the lion's share of the available processing power. SOLID 4D offers a flexible framework for performing exact collision detection on an extensive set of shape types. SOLID 4D shows that flexibility not necessarily results in poor performance: SOLID 4D is one of the fastest solutions for performing collision detection currently on the market!

SOLID 4D is useful in the following areas:

- **Game Development**. Current 3D games feature a high degree of realism. Needless to say, maintaining a high degree of realism demands advanced collision detection. SOLID 4D offers the necessary queries for performing ray casts, distance and penetration-depth computation, as used for instance in 1st person shooters and racing games.
- **Virtual Reality**. Cyberspace has become a reality. VR technology is currently used, for instance, for military or medical training simulations. However, interacting with a highly-dynamic virtual environment in real time remains a big challenge. Haptic feedback devices require refresh rates of 500Hz or higher. SOLID 4D is capable of performing exact collision detection at these rates.
- **Research & Development**. Researchers in areas of Robotics, Automotive Industry, Chemistry, Molecular Biology, and Industrial Prototyping, often need to perform simulations that involve a lot of interactions between geometric objects. SOLID 4D offers a flexible toolkit of query types that can be used in these areas.



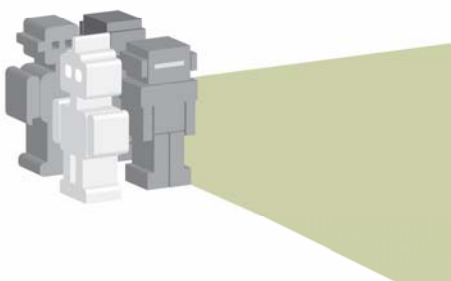
SOLID 4D Features

SOLID 4D incorporates a collision-detection pipeline that is composed of three phases:

- **4D Broad Phase**. In this phase the object pairs whose axis-aligned bounding boxes overlap at some point in the time interval are detected. The 4D broad phase is highly efficient both in processing time and memory usage. It uses fixed-point rather than floating-point arithmetic which makes it very fast and robust. Furthermore, it aggressively exploits frame coherence which keeps its computational cost low when configurations of objects do not change a lot. A unique feature of the SOLID 4D broad phase is that it is insensitive to fast movement of groups of objects all heading in the same direction at roughly the same velocity, such as fluids and complex characters.
- **Complex Narrow Phase**. Objects that are composed of many primitives, the so-called complexes, require a bounding-volume hierarchy for quickly culling the non-colliding pairs of primitives. This hierarchy is computed as a pre-processing step. Normally, bounding-volume hierarchies are static structures. They are computed once and cannot be modified. However, SOLID 4D allows for fast updating of bounding-volume hierarchies for deformable meshes, thus enabling fast collision detection of animated cloths and skin. SOLID 4D supports complexes of arbitrary shape types and is optimized for triangle meshes.
- **Convex Narrow Phase**. In this phase, pairs of potentially-colliding primitives are tested for exact collisions. Depending on the desired type of response, SOLID 4D will determine the first time of contact, the contact point and normal at the earliest contact, a witness point of an intersection, or the penetration depth vector and its witness points. The penetration depth vector is the shortest translation that brings the objects in touching contact. It is useful for determining contact data in case of interpenetration.

SOLID 4D offers a complete framework rather than a set of separate tools. Using SOLID 4D is easy. You simply define the shapes you want to use, define the collision response you would like to receive, place the objects in space by specifying each objects position and orientation, and attach them to a scene of objects. The rest is handled by the framework. SOLID 4D's interface is quite flexible:

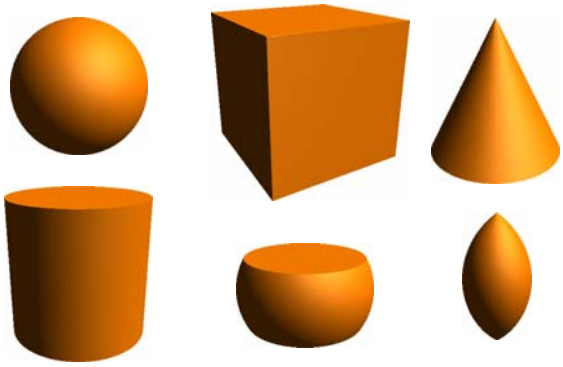
- Motion is specified using arbitrary affine transformations (translations, rotations, and non-uniform scalings).
- Polygonal complexes are specified using client-defined vertex arrays. A vertex array can be of any type and may contain any data besides xyz coordinate data. Deformations are specified by replacing or modifying a vertex array. This feature enables sharing of vertex data.

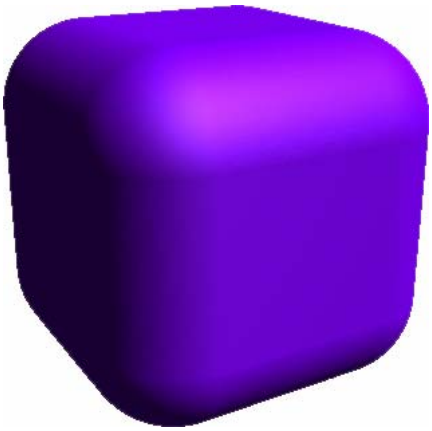


- Collision response is defined by the client by means of call-back functions. Response callbacks are defined per pair of response classes rather than per object pair. With each object a response class is associated. The response class of an object may change over time.
- Objects may be shared among multiple scenes. This feature is useful when collision detection is required for multiple tasks. For instance, it is possible to maintain at the same time a sound scene, a scene used for visibility culling, and a scene for physics simulations.

Any Shape You Desire

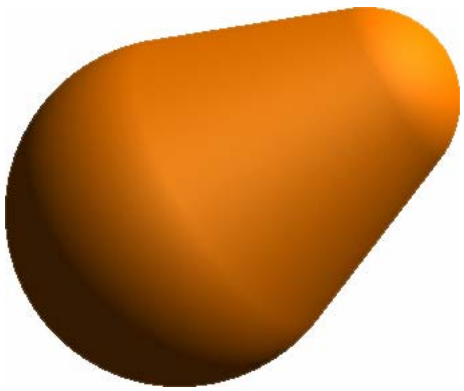
SOLID 4D offers you the freedom to represent your objects using virtually any convex shape type imaginable. It has an extensive set of basic primitive types, including balls, boxes, cones, cylinders, barrels, lemons, disks, line segments, triangles, and other polytopes[1]. Queries on convex polyhedra run in logarithmic time and are CPU-cache friendly for lightning-fast performance.

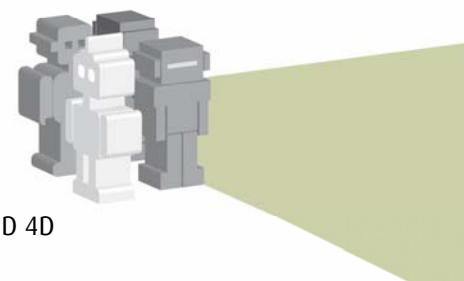




From these basic primitives more exotic convex shapes can be constructed using affine transforms, Minkowski sums, and convex hulls. Minkowski sums enable you to fatten objects by adding arbitrary convex shapes. For instance, objects can be expanded spherically, i.e., a ball can be 'added' to an arbitrary shape. The resulting object is the set of points whose distance to the shape is at most the ball's radius. This technique is useful for creating objects with rounded edges, or 'sensitive' areas around an object.

Another powerful tool for creating convex shapes is convex-hull construction. Any pair of convex objects can be 'shrink-wrapped' and used as a new object. For instance, the hull of a pair of balls of different sizes represents a tapered capsule that can be used, for instance, for limbs of an animated character.





[1] All images of shape types were rendered using the ray cast routine from SOLID 4D

middleware solutions
for real-time 4D collision detection

Any Platform

Controlling both performance and memory footprint is key in real-time applications. Many applications require special customization of the SOLID 4D library. SOLID has been designed with code-level performance-enhancements in mind:

- Any number type can be used within SOLID 4D, whether it be single-, double- or arbitary precision numbers. The SOLID 4D C++ code fully abstracts from the underlying number type. Numerical issues are resolved nicely for even the less accurate number formats. The use of a 32-bit floating-point format is often a requirement for games that run on current PC graphics hardware and consoles.
- Core routines can easily be replaced by custom routines. The generic design of SOLID allows for plugging in your own platform-dependent routines.
- Compressed AABB trees reduce the memory footprint of complex shapes considerably. This is useful for platforms that have limited memory, such as game consoles.
- The use of transcendental functions is kept to a minimum, and can be avoided for a limited number of shape types. This is useful for platforms with less powerful FPUs.

The only requirement is the availability of a modern C++ compiler. SOLID 4D currently compiles under GNU gcc version 2.95 and up, and Microsoft Visual C++ 7.1.

For inquiries about licensing SOLID 4D for your products, please contact:

DTECTA
Sterkenstraatje 4
5708 ZC Helmond
Netherlands

T+ 31 492663259
M+31 38305878
F+ 31 7431168
info@dtecta.com
www.dtecta.com

DTECTA is owned by Gino van den Bergen, the author of the book *Collision Detection in Interactive 3D Environments*, published by Morgan Kaufmann Publishers.

.