IFC-based clash detection for the open-source BIMserver

P. van den Helm, M Böhms & L van Berlo
TNO Built Environment and Geosciences, NL

Abstract

When designing construction objects, different disciplines have to collaborate intensively. Each discipline is responsible for its own contribution to the overall design. However, when the partial designs are combined, objects from different disciplines might occupy the same position in space or 'clash'. Such a clash will often be noticed during the construction process requiring a lot of extra time and cost.

Using 3D visualization, it is possible to combine the partial designs of all partners and find the clashing objects visually in an early stage. However, when the complexity of a construction project increases, a clash easily stays unnoticed. Besides, much more time is needed for inspecting a complex design. Therefore, an algorithm to discover clashes will be more reliable and efficient.

Clash detection techniques are already available in 3D graphics (referred to as 'collision detection'). There are a great number of methods described on the internet on how to perform this collision detection. All these methods are based on describing objects in terms of 3D shapes, but do not take into account the structure/hierarchy of objects that is common in construction projects.

In the AEC industry, the Industry Foundation Classes (IFC) open standard holds this structure next to the 3D shape of building objects. By using the IFC standard to perform clash detection on construction models, the structure of objects can be utilized, resulting in a much higher performance. This paper will demonstrate how to combine the best collision detection methods with the structure provided by IFC, making clash detection fast enough to be used inside a real time environment like a server. This clash detection is currently implemented in the IFC Engine viewer developed by TNO, and works on models opened from their open-source BIMserver (Building Information Model) where the collaboration starts. The next step would be integrating the clash detection inside the server.

Keywords: IFC, BIM, Clash detection

1 Introduction

When designing construction objects, different disciplines have to collaborate intensively. Each discipline is responsible for its own contribution to the overall design. However, when the partial designs are combined, objects from different disciplines might occupy the same position in space (called a 'clash'). When a design contained an unnoticed clash, it would eventually show up during construction requiring extra construction time and costs.

Before computers were common practice and paper drawings (mostly 2D) were used, finding a clash inside the design was almost impossible. The usage of computer graphics (Computer graphics wiki, Dec 2009) that started with 2D digital drawings, led to faster communication, but finding clashes was still almost impossible. Only when 3D graphics were introduced, clashes became visible

before construction within the 3D visualization of the design. Although finding a clash was still hard and time consuming for complex constructions it was a good start.

The next step is finding clashes automatically to speed up the process. On the internet there is already a lot describing how to do this (mainly via the term collision detection). The difference between these methods is speed versus accuracy. Some will perform faster, while others are more accurate. Although accuracy is the most important issue for clash detection, it is not wishful to have an algorithm taking hours before it knows which objects clash.

By using IFC models, additional information comes available next to the shapes of objects. This additional information makes it possible to filter useless comparisons directly resulting in a performance increase. By using the IFC structure as it is used in the open source BIMserver, these tools can be merged into a complete collaboration frame work. The BIMserver needs to be fast since no one wants to wait for their models coming from a server. With the speed improvement possible with IFC this criteria will be met.

This paper is intended for people interested in creating clash detection based on IFC, although it can even be interesting for people not using of IFC. The needed technologies to create the IFC based collision detection are described in chapter 2. Also the connection to the BIMserver is described in chapter 2. Chapter 3 goes into the details of the prototype of the IFC based collision detection is created. In chapter 4 some conclusions and recommendations are shown.

2 Survey of known technologies used for IFC clash detection

2.1 Collision detection

Collision detection is a method that verifies if two objects with geometry collide. A lot of research has been conducted on this level, leading to different approaches (Gottschalk, 2000)(Palmer and Grimsdale 1995)(Hubbard, 1996)(Klosowski et all, 1998)(Zachmann, 1998)(Bergen, 2005). Sites using these approaches often focus on gaming purposes where speed is most important since the algorithms are used real-time. Accuracy for predicting if objects collide is less important. By using simplified shapes around complex shapes the speed is increased, but accuracy decreases. Although this is not necessarily needed since many games already use simple geometry for most of the objects like the environment, and only a few shapes need to be compared. This differs from the building industry where the collision detection has to be accurate and performed on a large number of objects that might have a complex shape.

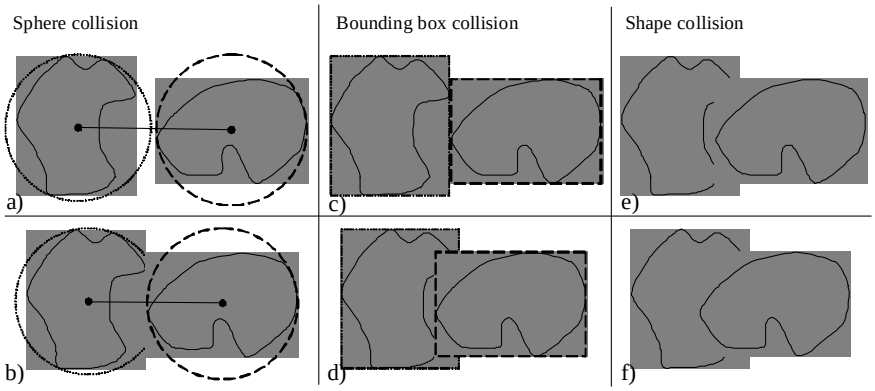

Figure 1 a 2D visualisation of different methods performing clash detection. Where a) and b) use a sphere and c) and d) use a axis aligned bounding box (AABB) as simplified shape. e) and f) are not using any simplified shape but are based on the complex shape. Objects that collide according to there used method are shown in b), d) and f), although only f) is colliding for real. To visualize the differences in accuracy between the various methods, b) uses the same scene as c) and d) uses the same scene as e).

The basics of algorithms predicting if objects collide/clash are similar. A surface description is created for each shape that has to be compared. This might be a surface consisting of the same triangles as the shape, but could also be a simplified geometry like a sphere or box. The next step will compare all shapes in pairs to check if they occupy the same space. The more complex the shape the more complex and time consuming this algorithm will be.

Comparing spheres is the fastest algorithm. Two spheres collide if the distance between the centres of the two spheres is smaller as the sum of the radiuses of the two spheres. Depending on how well the sphere surrounds the object, the result will predict if the actual objects collide as well. When all shapes fit completely within their equivalent sphere, not clashing spheres also indicate not clashing shapes. Spheres that do collide might have shapes that collide as well. The larger the difference between the sphere and the surface, the least accurate the indication of a collision will be.

Comparing "axis aligned" bounding boxes (AABB) is another fast algorithm. Axis aligned in this context indicates all the bounding boxes are using the same axis to describe their position, width, depth and height. The check if bounding boxes collide is rather simple. If in the width direction the position plus width of one bounding box is smaller as the position in width direction of another bounding box, they don't collide. Similar comparisons in the other directions (depth and height) will complete the comparison. Bounding boxes where none of the mentioned comparisons mark it is not colliding, do collide. Again the better the match between the complex shape and bounding box, the more accurate the prediction of a collision will actually indicate a collision of the shape.

A method used for detecting collisions that is accurate uses the surface of the shape. When two shapes collide, the surfaces of these objects will intersect. Since the surfaces are described by triangles in computer graphics, any intersection of a triangle from one shape with a triangle of another shape indicate a collision. The calculation for triangle-triangle intersection is complex and can be replaced by calculating the intersection between a line (edge of a triangle describing a surface) and a triangle of the other surface. The line triangle intersection calculation can be done via various techniques which next chapter will explain in more detail.

2.2 Ray triangle intersection

The most important calculation for accurately detecting collisions is the ray-triangle intersection calculation. To determine if a triangle is intersected by a line (edge) you can use standard mathematics. The triangle can be described as two vectors that form a plane. The ray is also a vector. Calculating the intersection point between a plane and a vector is basic mathematics. With the found intersection position can be determined whether this position is located within the triangle and is located within the bounds of the ray. If this is true the triangle and ray intersect.

There are some known algorithms to do these calculations. Most of them are optimized calculations derived from the basic mathematics method. Some claim to be faster as others, although this also depends on the expected outcome. For clash detection the intersection point is not important but only the indication that there is an intersection point or not. After a thorough check, the fastest possible method was described by (Möller and Trumbore, 2003).

Although the algorithm is much faster as implementing the basic mathematics method, it still uses a large number of multiplications making the algorithm slow to be performed millions of times. Each triangle-triangle intersection consists of three ray triangle intersection comparisons. Each triangle has to be compared to all triangles of the other shape for each shape inside the model, causing the millions of calls to this function.

2.3 IFC (Industry Foundation Classes)

IFC (International Alliance for Interoperability, Dec 2009) is a neutral data model structure developed for the construction industry. It allows users to exchange or share data among applications from various vendors. Within an IFC-file a building is not stored as geometry only, it will also contain

additional information about the building. A good example is the decomposition model used inside an IFC-file. This describes the hierarchy of which elements are parts of a higher level element. The hierarchy elements describing the structure of a building can consist of sites, buildings, wings, storeys and spaces.

The building elements like walls, doors, windows, etc. can be connected to only one structure element, but this is not mandatory. If this was mandatory it was possible to use this connection to structures to exclude some physical elements from the collision detection. The structures to which elements are linked describe where in the building objects are located. Via this method also elements in structures that are far apart can be ignored for clash detection. For instance walls on one floor will not collide with pipelines on another floor. Although the structure is not mandatory it can be used and would help, but applications creating IFC sometimes mess-up things as well what can cause unnoticed clashes.

Building elements like walls, doors, windows, etc. have a similar decomposition mechanism, but only for their representation. There are no doorknobs defined in IFC but you can create a geometry part for a doorknob that combined with the door panel and door frame represent the geometry of a door element. When checking if a door is clashing with another object the complete geometry would include the doorknob which might be a rather complex shape. In the next chapter is shown how this information from IFC can benefit the clash detection algorithm.

2.4 Open source BIMserver

The open source (BIMserver, Dec 2009), a development of TNO, can be used for sharing design data. Each partner can login to a project and add their revisions to the system. The merging functionality can bring designs of different project partners together. As mentioned earlier when merging designs of different partners it might result in clashes. Adding clash detection or a tool that can access the data from the BIMserver and perform clash detection would help the process of collaboration.

Internally the open source BIMserver uses a structure that matches close to IFC. This makes it possible to build the clash detection into the BIMserver, but this is not done in the prototype described in the next chapter. The prototype is a clash detection tool that can open files directly from the BIMserver and perform the clash detection.

To build the clash detection into the BIMserver, the server needs a method to get access to only the geometry. The BIMserver already uses such functionality for creating other formats as output like CityGML. Internally the geometry from the IFC-file is taken and converted to the other formats. For clash detection the same methods can be addressed to get the geometry, extending the capabilities of the server.

3 Prototype implementation of IFC clash detection

By combining the technologies mentioned above, the speed of the clash detection algorithm can be increased. Even without using IFC knowledge an increase of speed is possible. Combining the different clash detection methods an accurate prediction is achieved without using the ray triangle intersection for each shape. Shapes that are far apart will be filtered by the sphere or bounding box method. These methods will indicate that shapes will not clash, so no ray triangle calculation is needed. Only shapes where bounding boxes and/or spheres clash will be checked further using the ray triangle intersection calculation. In figure 2 this method is shown in a flowchart.

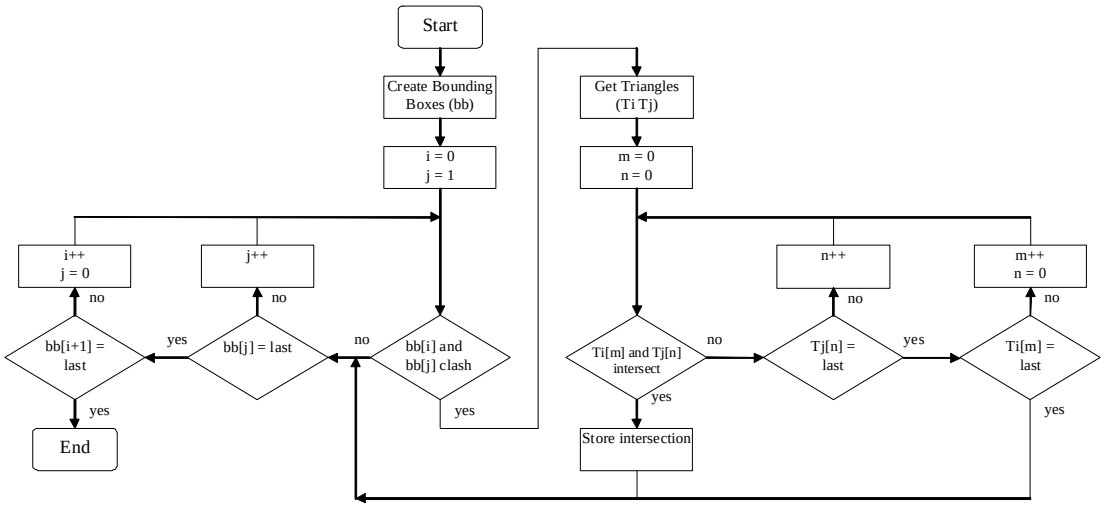

Figure 2 a flowchart of an accurate clash detection method not using IFC knowledge.

Because buildings are most often box shaped objects like walls and doors, I have chosen to use the bounding box method only to filter objects that are far apart. It is possible to even use the sphere method as well, although in many cases this will not increase speed much. However this depends on the shapes that need to be checked for clashes. If nearly round shapes are used often, the sphere check will increase speed more.

Before bounding boxes can be compared they need to be determined. For the comparison of bounding boxes the coordinates should use the same origin. Since IFC allows objects to be drawn relative to other objects (via the local placement), all coordinates of the objects should be converted to world coordinates. In an IFC file there is one root local placement that can be used as origin to define the absolute coordinates of all vertices.

After converting all vertices to absolute values, the bounding boxes can be determined. The first step in the two bounding box methods is retrieving the bounding boxes. This is a rather simple function that goes past all points used in the geometry of an object. Within this loop each point is checked if it has the lowest or highest x, y or z coordinate value till then and if so, stores this value. At the end of the loop the bounding box contains the minimum and maximum x, y and z coordinates used in the object. These values can be combined to describe the eight corners of the bounding box.

The speed increase is huge when using the bounding box collision detection to exclude further analyses of some object, but it still needs hours for complex models with a high level of detail. If detailed objects are split into smaller parts within the IFC file, which is often the case, the speed can be increased even more. A good example is a doorknob or window handle (a smooth, round shape) using a large number of triangles. The smoother the round shape is drawn the more triangles are used and thus more complex the object.

Using the bounding box collision detection on parts, the number of ray-triangle intersection calculations can be reduced further. When a clashing object contains parts, bounding boxes around these parts are compared first with parts of the other object. Only the parts within these objects that clash will be using the ray-triangle intersection calculation. The reduced number of ray-triangle intersections will speed up the process to seconds. Figure 3 shows a flowchart that specifies the process behind the combined clash detection using IFC knowledge. The steps used within the three different “levels” of clash detection are grouped within a gray box.

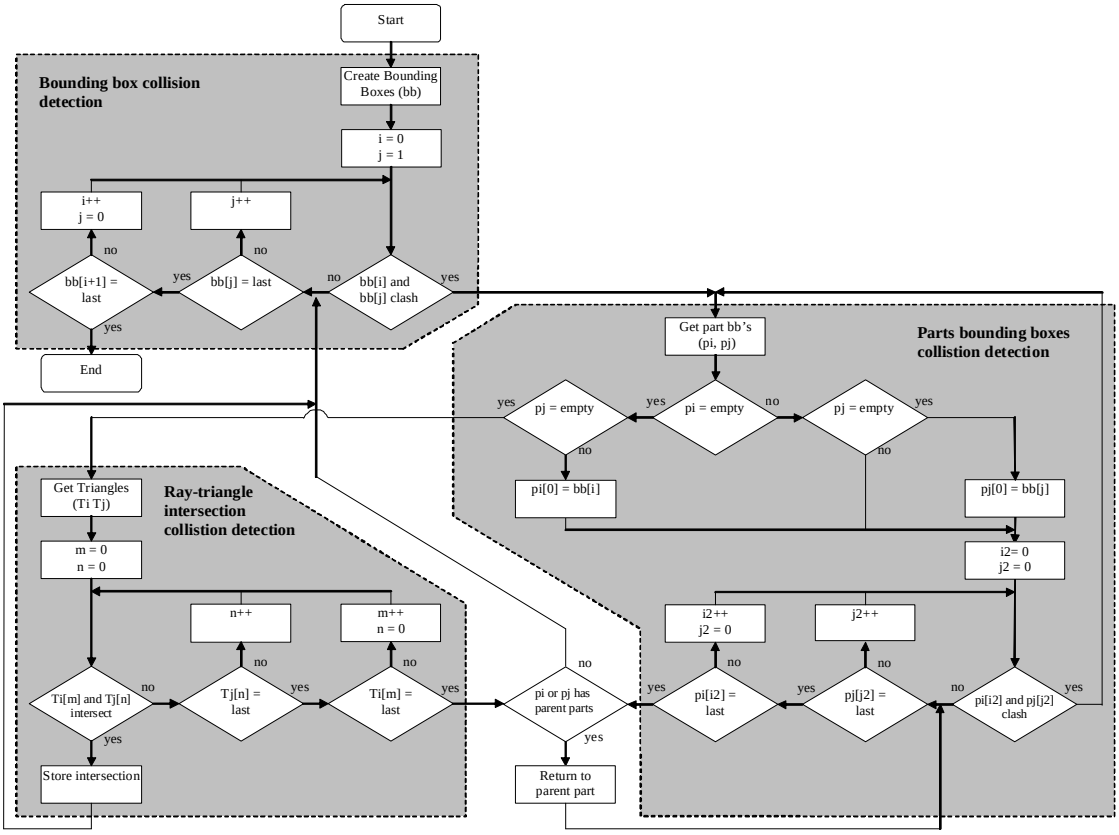

Figure 3 a flowchart of clash detection using IFC knowledge. The gray boxes indicate the different clash detection methods.

As is shown in the figure the collision detection consists of a check of the root objects bounding box, a part bounding box check and a ray-triangle intersection. The collision detection starts by comparing object bounding boxes. Objects that clash will try to use the bounding box clash detection on parts, but only when one of the two compared objects has parts. The part collision detection works similar as the bounding box collision detection but compares all parts of one object with the parts of the other object. Parts that clash and can be split into smaller parts will repeat the same steps but on the smaller parts. This makes the part bounding boxes collision detection a recursive method.

If two objects or parts that clash can't be split into smaller parts the ray triangle intersection is used to find the objects or parts that really clash. Finding triangles that intersect in the ray-triangle intersection indicates a clash. Clashes that are found can be reported and the clash detection can continue with the next objects. But without finding an intersection the ray-triangle intersection collision detection will keep calculating intersections between triangles till it does find an intersection or all triangles are used. When there are no intersections found, it will continue with comparing the next part or object depending on what 'level' started the ray-triangle intersection.

Detection of clashing bounding boxes uses a simple calculation and comparison. If the minimum value in any axis-direction (x, y or z) of one bounding box is larger as the maximum value in the same direction of the other bounding box, these objects can't clash. Only when in all directions this comparison does not result in true the bounding boxes compared clash.

The clash detection method as described is implemented inside the free IFC viewer developed by TNO and is available on www.ifcbrowser.com. This version of the clash detection tool can even filter clashes on the minimal overlap.

4 Conclusion

Designing a construction requires different disciplines to collaborate by working out their own objects and combining all designs into one total design. However, when the partial designs are combined, objects from different disciplines might occupy the same position in space (called a 'clash'). When a clash in the design stays unnoticed, it would eventually show up during construction requiring extra construction time and costs. This can be solved by a clash detection application. The usability of the clash detection depends on its accuracy but also speed especially when build inside a server.

The prototype created as described in chapter 3, does meet these goals. It is accurate in finding clashes and does this within a few seconds for complex models. During the creation of this prototype the performance gained speed by using different approaches:

- Using the ray triangle intersection algorithm described by Möller and Trumbore (2003) instead of the conventional mathematical methods or other ray-triangle intersection algoritms known.
- Using axis aligned bounding boxes around objects to select which objects need the time consuming but accurate ray-triangle intersection calculation and which objects not since the bounding boxes don't clash.
- Using bounding boxes around parts of objects from the IFC structure to select only the relevant parts that need to be compared via the ray-triangle intersection calculation. This again decreases the amount of ray-triangle intersection calculations needed.

Optional approach:

- Relying on correct IFC files, objects linked to different structures (buildings, wings or storeys) don't need the ray-triangle intersection calculation since they will not clash. This is not used in the implementation since many applications create IFC-files that contain mistakes.

References

Computer Graphics Wiki, http://en.wikipedia.org/wiki/Computer_graphics, last viewed: Dec. 2009

GOTTSCHALK, T., 2000, Collistion Queries using Oriented Bounding Boxes, Avialable online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.37.6529&rep=rep1&type=pdf

PALMER I. J., Grimsdale R. L., 1995, Collision Detection for Animation Using Sphere-trees, Computer Graphics Forum, 14(2):105-116.

HUBBARD, P. M., 1996, Approximating polyhedra with spheres for time-critical collision detection, ACM Transactions on Graphics (TOG), v.15 n.3, p.179-210

KLOSOWSKI, J. T., HELD, M., MITCHELL, J. S. B., SOWIZRAL, H., ZIKAN, K., 1998, Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs, IEEE Transactions on Visualization and Computer Graphics, v.4 n.1, p.21-36

ZACHMANN, G., 1998, Rapid Collision Detection by Dynamically Aligned DOP-Trees, Proceedings of the Virtual Reality Annual International Symposium, p.90, March 14-18

BERGEN G. van den, 2005, Efficient collision detection of complex deformable models using AABB trees.

MÖLLER, T. AND TRUMBORE, B., 2003. Fast, Minimum Storage Ray/Triangle Intersection, Available online: http://www.cs.virginia.edu/~gfx/Courses/2003/ImageSynthesis/papers/Acceleration/Fast%20MinimumStorage%20RayTriangle%20Intersection.pdf

IAI, International Alliance for Interoperability, www.buildingsmart.com, last viewed: Dec. 2009

LIEBICH, T., 2009, IFC 2x Edition 3, Model Implementation Guide, Available online: http://www.iai-tech.org/downloads/accompanying-documents/guidelines/IFC2x%20Model%20Implementation%20Guide%20V2-0b.pdf

BIMserver, www.bimserver.org, last viewed: Dec. 2009