Technological & Methodological Solutions for Integrated Wide Area Situation Awareness & Survivor Localization to Support Search & Rescue Teams

7th Framework Programme

FP7-SEC-2013-1 N. 607522

The open source software Blender and the bullet physics library. A DEM method to simulate collapsing building structures.

Workpackage	WP3	Simulation tool for structural damage analysis & casualties estimation	
Editor(s)	Laurea University of Applied Sciences LUAS,Oliver Walter, Kai Kostack		
Status	Final		
Distribution	CC BY		
Issue date	2015-04-30	Creation date	2015-04-30

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 607522

This project has received funding from the European Union’s Seventh Framework Programme for research, technological development and demonstration under grant agreement no 607522

TABLE OF CONTENTS

LIST OF IMAGES

INTRODUCTION

This report examines the available tools within the software Blender, that enable users to simulate physical interaction between objects in real time. It is written within the scope of the Inachus [1] research project in which methods are elaborated and compared that allow predictions of the behaviour of collapsing building structures. This document marks the baseline from which further research then will be conducted. This study brings clarity to the authors from Laurea university of applied science, what further tools need to be developed to get faster and better simulation setups and results. This report is also meant as an introduction to the other partners of the Inachus project and for interested readers of this topic in general. In the following sections we will state basic facts about Blender and the nature and characteristics of the bullet physics library. More in depth going information can be retrieved from relevant websites. We then will hint to available scripts for simulation setups and conduct simple build-ups to validate the accuracy of the physics integration in Blender. We will showcase a first simulation result, the Dom-Ino house by the architect Le Corbusier. In the end we will showcase some techniques in Blender how cavities in the debris can be visualized.

1. Section 1

1.1. What is Blender?

Blender is a free professional 3D modeling software. Active development has brought to this software a wide variety of functionalities including video editing, animation tools, sophisticated texture mapping, game logic, sculpting, path tracing renderer, real time physics simulation etc. Blender is considered by many as a valid alternative to commercially available 3D software. It has a rapidly growing user base, that is freely sharing content: tutorials, models and plug-ins. Beside a growing number of users there is also a growing number of programmers that add constantly to Blender´s functionality. Blender is foremost developed as a tool for artists but it is also often used for scientific purposes.

Blender started originally as a proprietary program owned by the Dutch animation studio NeoGeo in 1995 under the lead developer Ton Roosendaal. When NeoGeo was taken over by another company Ton Roosendaal and Frank van Beek decided to develop Blender further for commercial use under a new company called Not a Number (NaN). Investors decided to end the development of Blender in 2002 due to economic hardship. In the same year Ton Roosendaal founded the non-profit Blender Foundation whose goal it was to develop Blender further as a community based open source project [2].

1.2. Bullet physics

Bullet physics is a freely publicly available real time physics engine developed by Erwin Couman primarily for games and visual effects in movies. In the past years a number of similar engines have emerged in the market such as PhysX, True Axis, Newton etc. They all differ in type of license, available features, supported platforms and run-time performance. While most of the engines provide sufficient properties suitable for game development, where simulation accuracy is of minor importance, some supply a degree of stability that eventually allows them to be used in scientific simulations. Researchers and simulation engineers are typically more concerned with the accuracy of a physics simulation system then with visual effects. One example where bullet physics has been used in a nameable scientific project is the tensegrity robotics simulator made by NASA [3].

1.3. DEM vs FEM

The main challenge that every physics engine has to solve is the forward dynamics problem in a multybody system, where based on given forces and after collisions new trajectories and speeds as well as resulting reaction forces are calculated. This method is commonly referred to as DEM (discrete element method) multybody dynamics simulations assume ideal rigid bodies. The free motion of single rigid bodies moving through space is relatively simple to calculate. However, in the real world, bodies are deforming on impact , the numerics that underlay the DEM simulations do not take into account those deformations. This may distort the result of complex

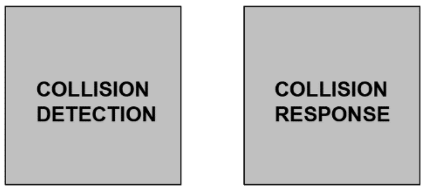
simulation, e.g. the collapse of a high rise building, to a degree that can easily undermine its validity. The DEM method does not include FEM (finite Element method) differential equation routines, which would be necessary to examine structural behavior during elastic modes of loading, the automatic yielding of reinforcement, detection and generation of buckling, crack propagation or separation of material under tensile forces. It is possible to link external FEM solvers through Blender´s Python-API. A list of cross platform open source FEM solvers is the following:
-Code_Aster
-Kratos [4]
-Elmer FEM solver
-FEniCS Project
-GetFEM++
To connect FEM to Blender is challenging due to lacking documentation and the requirement of a high programming skills set. It actually remains to be seen, if it will be possible to simulate finite element routines with genuine discrete element methods close enough to achieve satisfactory simulation results. One possible approach could be dynamic fracturing [5], where the expected highest load locus in an element is detected and where a cut is generated should the load exceed the maximum (see img 6).

2. Approach in Blender

The bullet physics library has been available in Blender´s game engine since 2005 (Erwin Couman had integrated it himself). Now its features can be handled in the modeling space directly. Its functionality can be activated from the main tool bar. Here rigid bodies can be given two main properties: Either a rigid body is passive or static. Passive objects are fixed in place, while active objects are affected dynamically by collisions. A large number of objects can be set up in a scene. Objects can be independent or interconnected with constraints. This assembly of objects then is influenced by gravity or strain inflicting forces over a set time span. The physical simulation is directly played back in the program´s viewport, objects in the scene can even be grapped with the pointer and moved around which dynamically impacts the other objects as well.

2.1. Collision detection

The simulator´s tasks can be subdivided in the two main actions: collision detection and collision response. While collision response deals with the physical aspect: how to alter the motion of objects so, that they do not fly through each other, collision detection is a purely geometrical problem and probably the most computationally intensive part in the simulation: Here it is evaluated, if the geometry of objects interfere at a single point in time or not. Blender has collision shape support that approximates the outline of geometries. Mesh and convex hull are most complex while primitives such as box, sphere, cylinder or capsule are fastest to calculate. A slight collision margin is usually applied for higher simulation stability. It is not yet possible to get information about accurate collision points during a simulation. This data is made available by bullet but Blender does not allow to take hold of it by not exposing it over its python API. This would be a precondition for dynamic fracturing [6].



img 1: The basic components of the DEM physics simulator

2.2. Materials

Blender has an extended list of material density definitions from which the mass of a body is calculated automatically. There are also parameters for friction and bounciness, which have to be numerically typed in. Those parameters do not have affinity with real world parameters and tests have shown that they do not have a linear simulation effect either [7]. We aim at extending the material properties with material specific strength values to be used in the breaking threshold evaluation of constraints

2.3. Softbody and cloth simulation

Several other independent physics integrators are already used in Blender, such as the cloth or the softbody simulator. They differ however from the bullet physics engine and are not compatible with it. According to Ton Roosendaal all Blender physics tools will be sooner or later unified under one unique physics library for example the bullet library which has actually softbody support already, which accuracy on the other hand is questioned in the above mentioned NASA project [3]. Other possible physics engines that could be integrated into Blender are PhysBAM or Nvidias FLEX, but with those might occur licence problems. Softbodies are the most obvious candidates for the simulation of deformable and breakable elements. Flawless interaction with rigid bodies assumed.

2.4. Constraints

1.1 There are a number of constraints, that can be used to connect objects with each other to simulate real world joints, hinges clamps, springs etc.

2.1 The available constraints are:

3.1 -point	-fixed
4.1 -slider	-hinge
5.1 -motor	-generic springs
6.1 -piston	-generic

7.1 Each of those constraints can be set to be breakable: a threshold defines, when the constraint shall dissolve.

2.5. Limitations and challenges

To setup a scene in Blender with several ten thousands of separate rigid bodies is straight forward and simple and the resulting simulations look also very plausible (img 2) . But as soon as objects are connected with constraints to achieve arrangements of higher complexity the procedure gets painstaking: Constraints have to be set one at a time and the program places constraints seldom correctly. It is very time consuming manual work to move constraints to their right location, for example the bearing of a beam or the foot of a pillar. Joints have particular definitions for the degrees of freedom (DOF), to set those parameters up is very labor-intensive as well.

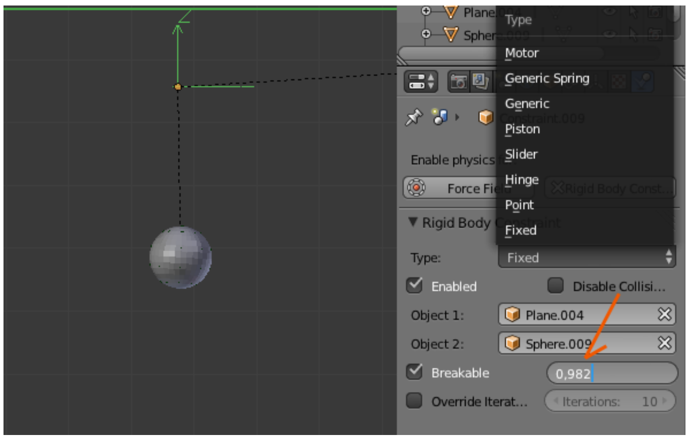
img 2: Simulation of 3000 KEVA planks by Phymec, source youtube

The “bullet constraints tools” add-on [5] automatizes certain steps like batch placing constraints over a selected amount of objects or batch updating constraint parameters. For civil engineering simulation purposes however the methods need considerable improvements.
Another tool is the fracture modifier [8] which according a benchmark test by Kai Kostack is up to 58´000 times faster in placing constraints. But this modifier needed optimization of Blender´s code base and is only available as a separate built downloadable from graphicall.org. Blender still sets limitations for managing sub-objects like mesh fragments that result from fracturing. But it is planned to bypass this restraints in future Blender releases.
This is the reason why the fracture modifier for now will not be implemented into Blender. Another unintentional big limitation is, that Blender doesn´t expose bullets force variables (linear and angular forces) in the python API. On the level of the game engine- on the other hand- this is already exercised. This lack should be investigated because picking up force variables would be helpful to evaluate forces within rigid bodies. The same applies for a list of colliding objects and their collision coordinates that should be exposed in the python API as well. These API

improvements would give more flexibility to physics simulation related add-ons in general.
Hereinafter we will learn from the short comings and reflect on how to improve the methods to help build more complex simulations for engineering purposes. Special attention will be paid, that the new functionality can be packed into an add-on, an optional script that extends Blender’s functionality. A special Blender built should be avoided because this would mean to hack Blender´s source code. Because Blender is primarily developed as a universal tool for 3D artists any highly specialized customization is seen critically. It therefore would be unlikely to get support from Blender´s core developers, too.

3. Experimental set-ups

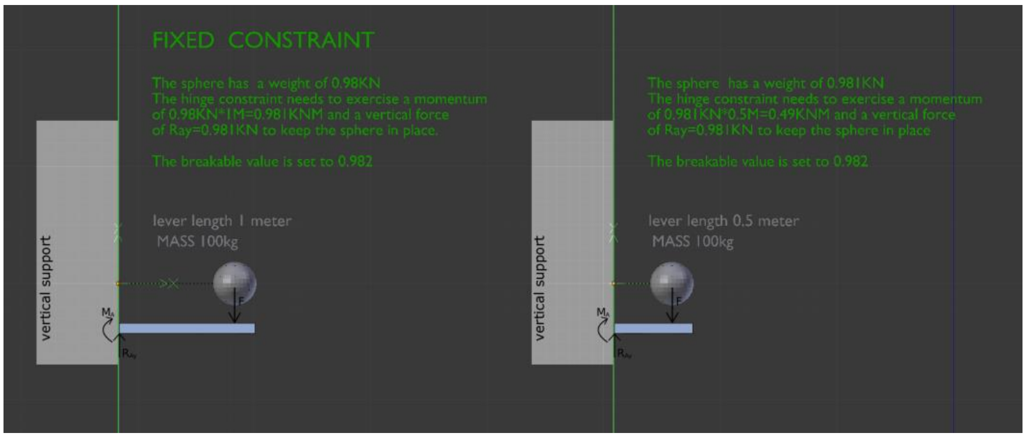
We built some simple experimental arrangements in Blender to see if the dynamic flux of forces are represented as expected.
a.) In a first set up we could confirm that threshold units correspond to K-Newton. For this a constraint kept a sphere with a mass of 100kg in place. When the threshold was lowered to 0,981 units the constraint dissolved. This is valid for 1000 steps/sec. increasing the step size will increase the necessary threshold value proportionally. All constraints work consistently in this manner except the generic constraint that behaves erratic and that therefore should not be used in the current stage.
Threshold (KN) = F   (N)/ Steps
Threshold (KN) = M (N*m)/ Steps



Type: Fixed
Enabled ✓ Disable Collisi...
Object 1: Plane.004
Object 2: Sphere.009
Breakable ✓ 0,982
Override Iterat... Iterations: 10

img 3: Setup for breaking threshold evaluation

b.) The problem with only one threshold value is that a joint/constraint can be exposed to more than only one force type. For example a joint can give up either if the momentum (N*m) exceeds or if the tensile force (N) passes over the limit. To illustrate this we have setup a cantilever (length 1m) with a weight (0,981KN) (img. 4). This cantilever breaks when the threshold is set lower then 0,982 units, which corresponds to the permissible momentum of 0,981KN*m. So far so good. But if we now shorten the cantilever to 0,5 meter the threshold value still triggers the breaking, even though the effecting momentum is now only 0,49KN*m.


FIXED CONSTRAINT

The sphere has a weight of 0.98KN
The hinge constraint needs to exercise a momentum
of 0.98KN*1M=0.981KNM and a vertical force
of Ray=0.981KN to keep the sphere in place.

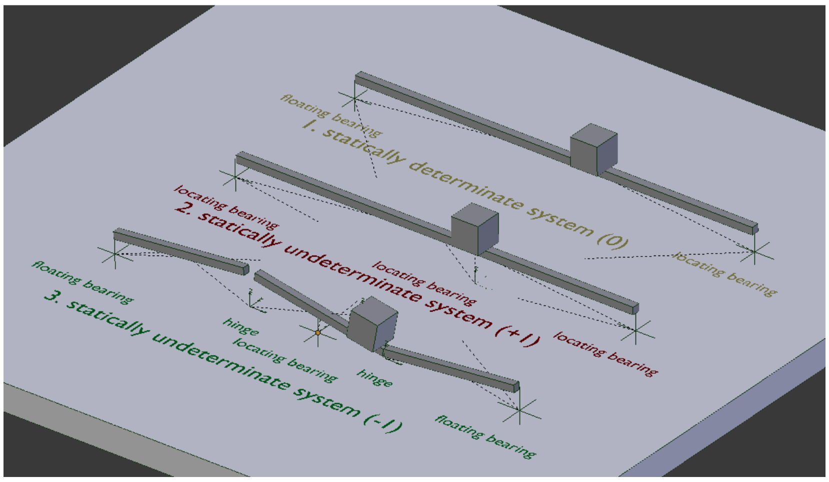
The breakable value is set to 0.982

The sphere has a weight of 0.981KN
The hinge constraint needs to exercise a momentum
of 0.981KN*0.5M=0.49KNM and a vertical force
of Ray=0.981KN to keep the sphere in place

The breakable value is set to 0.982

img 4: Setup with one threshold representing momentum and shear force

This means, that the threshold now automatically applies to the shear force (Ray). The admissible shear force at the joint in reality would most likely be higher. This is valid for all constraints except the point constraint. A threshold is always omnidirectional and does not distinguish between compressive and tensile forces which would be needed to simulate anisotropic material or composite microstructures such as armed concrete. In general the evaluation of forces ought to be more differentiated and allow multiple thresholds.
c.) Another arrangement consisted of statically determinate and statically indeterminate systems (see image 5). The particularity of a statically indeterminate system is that there are more bearing reactions then degrees of freedom (DOF) in opposition to determinate systems where bearing reactions equal the DOF. While the reaction forces in a determinate systems can be easily calculated with simple equilibrium equations the indeterminate system can´t be determined by the equations of statics alone. It requires simultaneous examination with additional equations characterizing deformations in the system.

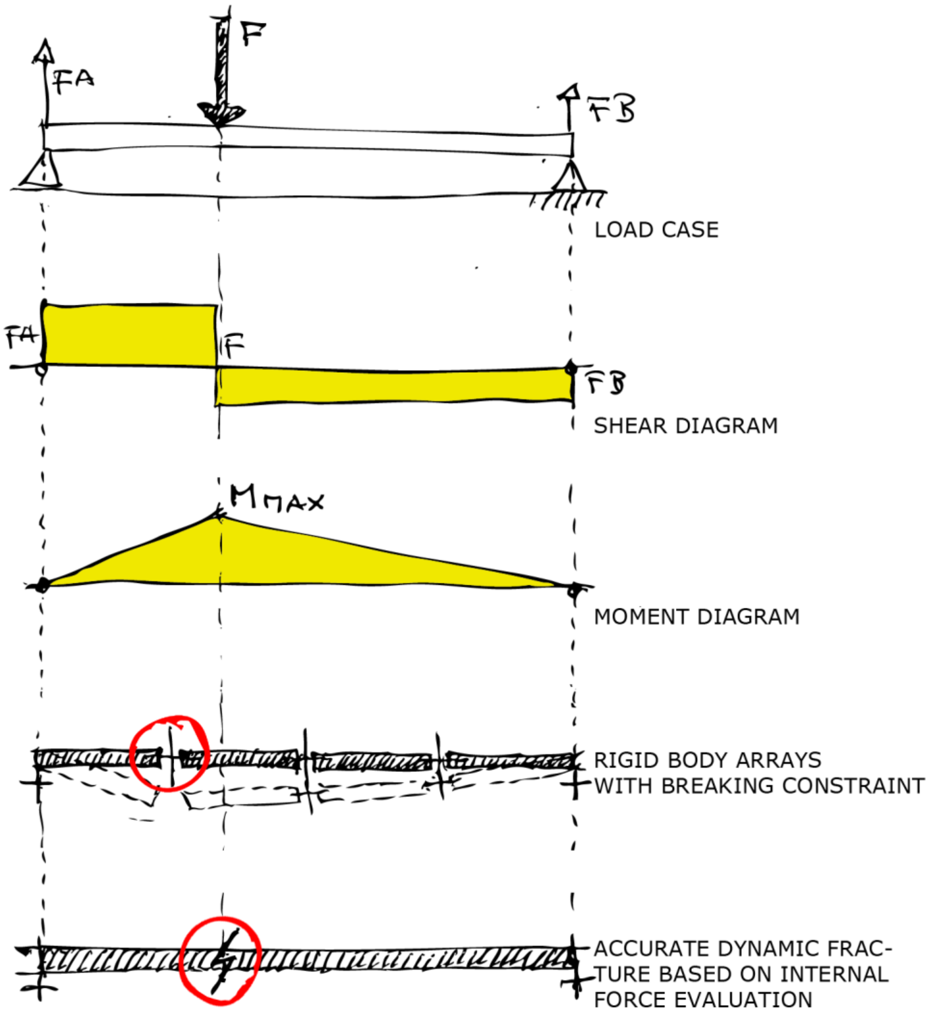


img 5: Setup with determinate and indeterminate beam systems

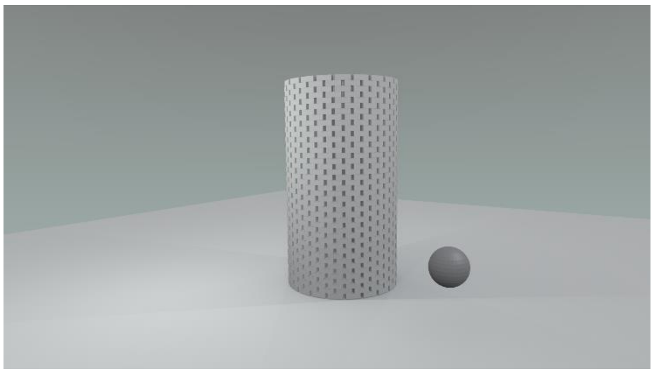
8.1 In the arrangement in image 5 cubes were placed on three beam structures.

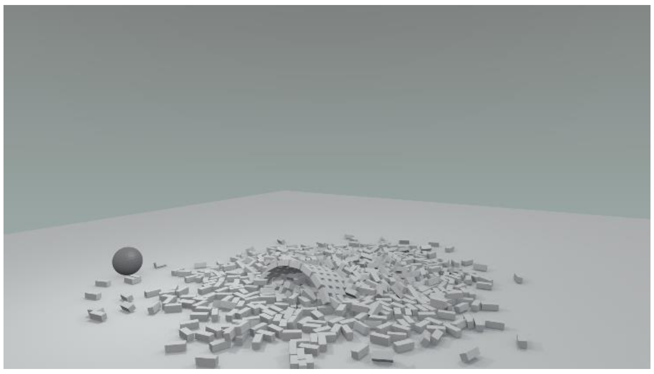
1. statically determinate system (0)
   with locating bearing, floating bearing
2. statically indeterminate system (+1)
   with 2 loacting bearings
3. statically indeterminate system (- 1)
   with 2 floating bearings and 2 hinges

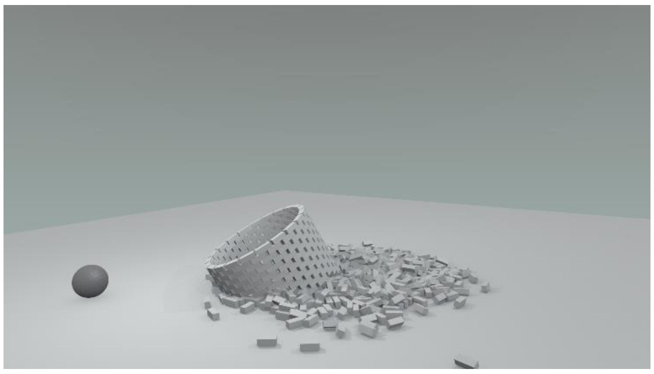
In reality the beam in system 2 would have a tendency to build up higher tension, that result from denying one DOF, which subsequently might result in stronger deformation or cracking. If the beam in the 3D model is represented as one rigid body, those tensions remain unnoticed during simulation.

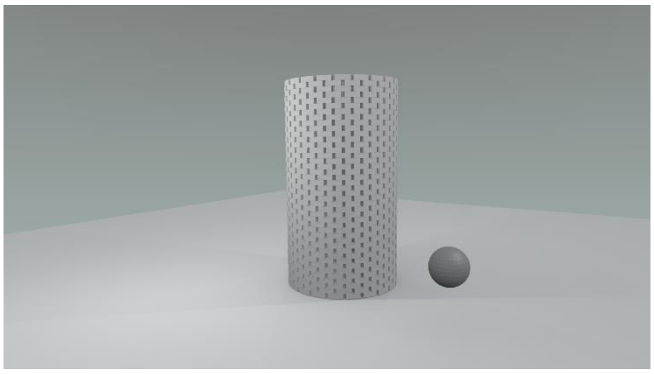


img 6: Discrete and finite method to detect locus of highest tension


img 7: Brick tower before impact of sphere


img 8: After impact with 100steps/sec, solver time 33 sec


img 9: After impact with 500 steps/sec time 76 sec


img 10: After impact with 1000steps/sec, solver solver time 164 sec

One possible way to detect those tensions and trigger cracking would be to subdivide the beam into smaller rigid bodies connected with constraints. In system 3 two DOF are released by adding 2 hinges. The resulting unstable system settles in the simulation as expected. Image 6 shows the principle dilemma of the DEM method to fail predict accurately the crack location.

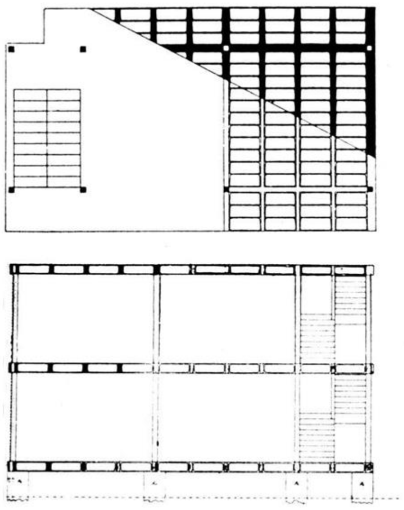
d.) In Blender´s scene panel the accuracy of the simulation can be adjusted via the steps/sec Paramter: Amount of simulation steps made per second. Higher values are more accurate but slower. The image sequenze on the left (img 7-10) illustrates the effect of changing steps/second values. The results differ to the extend that in the simulation with low steps/sec the tower collapses completely and with higher steps/sec still stands upright.

Earlier experiments have shown us, that the steps/sec value should be a multiple of the frame rate because of resulting asynchronism between bullet and blender, that else is noticable in periodical sudden jumps in movements.

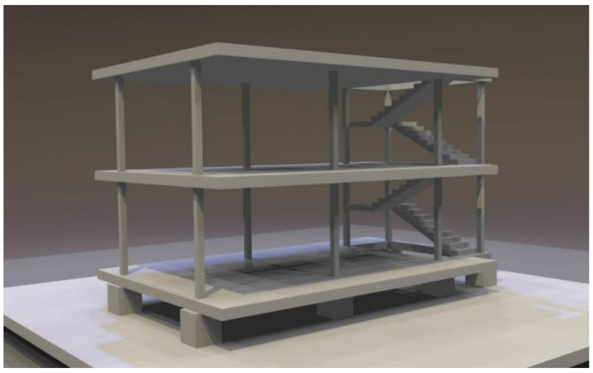
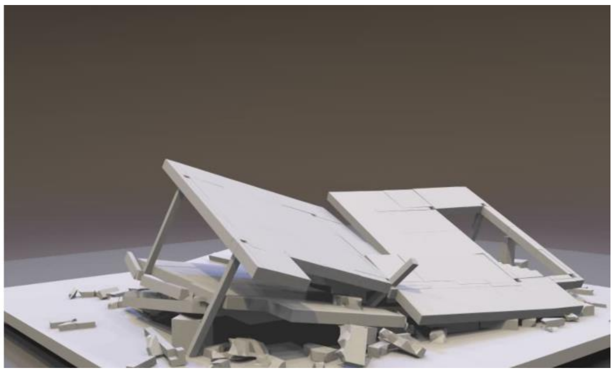
The ideal steps/sec rate and iteration amount still needs to be evaluated.

e.) Kostack studio provided a simulation of a simple steel reinforced concrete building, Le Corbusier´s Dom-Ino house (img 11). No accurate construction drawings were found. The steel armoring was estimated from the low resolution image on the left (img 12). The 3D model holds differently weight constraint groups to simulate the reinforcement. The model stands on a ground plate that simulates strong seismic activity.
This simulation (img13-14) is done with the Bullet engine based Dynamic Fracture system in Blender by Kai Kostack [6]. Here solid rigid bodies are broken on impact and scattered to smaller fragments using boolean operations. This fracturing system is still under development and will be further tested in upcoming building simulation cases.

We suggest to inclose this DomIno house type as one of Inachus´ test cases. It is very compact and could help to calibrate our software. It still has a size that can be handled with a FEM analysis for comparing purposes.



img 11, 12: the domino house from Le Corbusier serves as a very suitable study object: it consists of basic structural elements pillars and slabs from reinforced concrete.

img 13,14: The domino house built and simulated in blender. The collapse was triggered by techtonic shift of the base plate. (by Kai Kostack) https://www.youtube.com/watch?v=iE6XFlyopqY

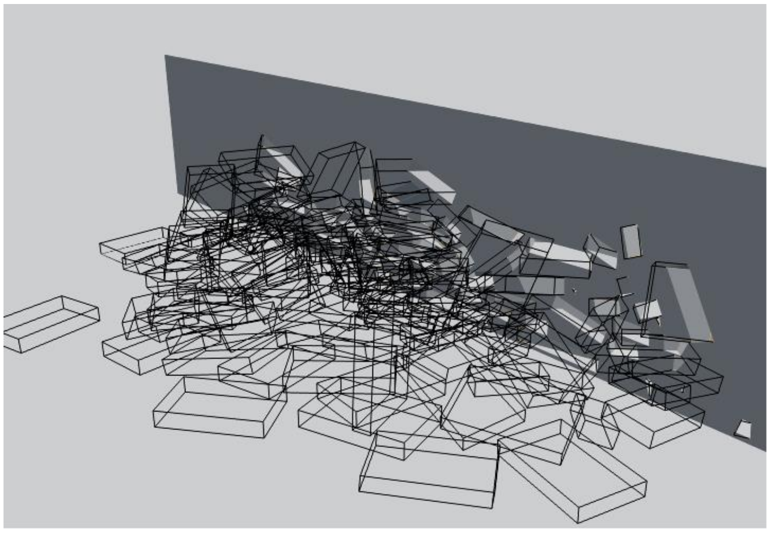
4. Cavity detection in debris

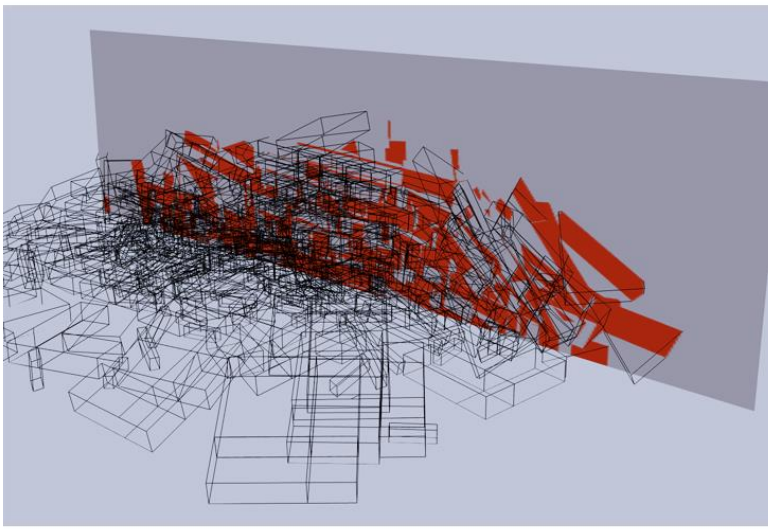
After the simulation results have been produced,
we have several possibilities to visualize cavities in the debris. Blender of course does not have a ready optimized routine for this, but a combination of several tools can be used to do exactly that. We made several tests with a calculable cluster of elements, a few hundred objects, which looked promising. Of course these techniques need to be evaluated in more realistic scenarios with ten thousands if not hundred thousends of debris elements.

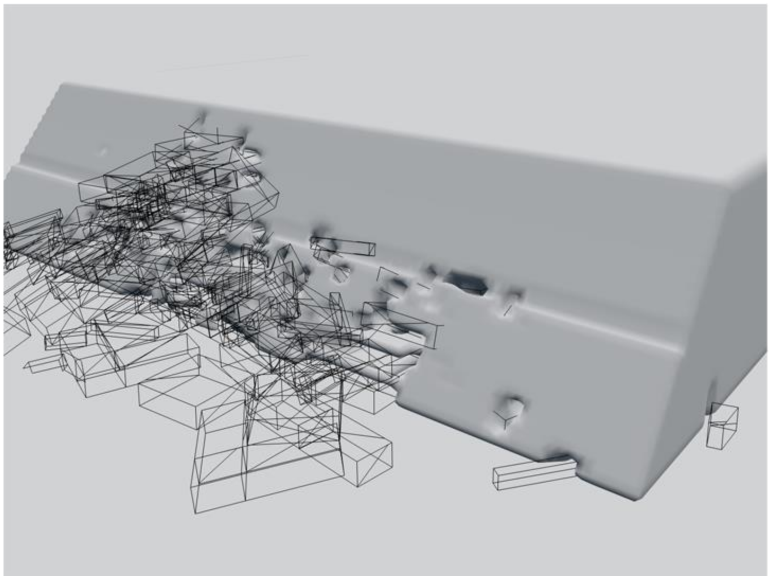
The first approach uses a boolean operation method in which the rubble was merged into one mesh object. This object was then substracted from a solid cube. The boolean operator in Blender is not the most advanced and the program was subjected to crashes to get the result. Blender will not be able to deal with meshes with high face count (img 15).

A second approch uses a sort of cell grid to detect void volumes. The void is filled with a bubble like mesh representation. The accuracy of the detection can be adjusted with thresholds. A section plane can be moved interactively through the rubble area. The cavity representation is updated with some couple of seconds delay. This technique is stable but is -depending on the resolution- quite slow (img 16).

A third very elegant method allows also interactiv investigation of voids in the debris by using the dynamic paint feature in Blender: A movable section plane acts as canvas while the debris mesh functions as a brush. Those areas of the canvas, that are within the brush volume are tinted with color. the uncolored areas then represent the voids (img 17).


img 15 cavity visualization by boolean


img 16 cavity visualization with cell grid, bubbles hint to voids of specific size


img 17 cavity visualization with dynamic paint, red colored is debris

5. Victim search

There are also several approaches to compute the position of trapped victims after a building collapse. One of them is to use Blender´s particle system. But we are not sure yet, if the particle physics interact well with the bullet physics simulator. Another more obvious way would be to use simplified human mesh representations as bullet rigid bodies. Their position would be simulated together with the main structural analyses.

6. References and links

[1] The INACHUS project is a European wide research project starting in 2015,-that examines measures to improve first response after building collapse in catastrophic events like earthquakes, storms, explosions etc.

[2] Open source licence:
Blender´s source code is published under GNU General Public License (GPL or GNU GPL) It means that the source code is freely accessible and can be changed, enhanced or complemented with new code as long as the licence of resulting code stays open as well.

[3] http://ti.arc.nasa.gov/tech/asr/intelligent-robotics/tensegrity/ntrt/

[4] http://www.cimne.com/kratos/

[5] “bullet constraints tools” is a Blender addon for creating constraints, current version 0.3.7 author:bashi
https://bwide.wordpress.com/scripts/bullet-constraints-tools/

[6] dynamic fracturing generates crack on the fly during simulation. No pre-fracturing needed. Development by Kai Kostack. https://www.youtube.com/watch?v=fcdltcdcbiE

[7] “Evaluation of real-time physics simulation systems” , page 285 by Adrian Boeing and Thomas Bräunl

[8] fracture modifier by Scorpion 81 http://df-vfx.de/fracturemodifier/
The benchmark test was done with 2500 Rigid Bodies and 4000 connecting constraints. The “bullet constraints tools” script placed the constraints in 23 minutes, the fracture modifier in only 0,024 seconds.