Investigation of the crash behavior of a LEGO® car

Playing with LEGO® bricks is something many engineers might have enjoyed during their childhood. And to some extend the creativity and complexity that LEGO® allows when building any kind of mechanical construction might have contributed to their fascination and finally in their decision of becoming engineers. It's interesting to see how many of them still stick with their fascination with LEGO® even in their adult live. In order to teach about mechanical engineering and FEA using LEGO® as a motivator might be a good idea. It could help young students to grasp some of that fascination and also aid them to see the fun of physics seems a logical thought.

Looking through the internet one can find seemingly endless resources of pictures, videos, construction instructions and also 3D CAD models on sites such as ldraw.org or bricklink.com. These 3D models are commonly made with special LEGO® CAD software and there is not just one but a whole bunch of different options to choose from. Among other LEGO® CAD softwares, there are LDCad, leocad, studio and complementary tools such as LDView just to name a few of them. Also there is a vivid community at ldraw.org maintaining a library of CAD Data for all the numerous bricks that are available by LEGO®.

However before starting to create a model in CAD it might be advisable to build the real thing with real LEGO® bricks. Especially with children, this is half the fun and gives room for fantasy and motivation. So the first prototype of my attempts before going virtual actually looked like this:

Figure 1: first physical prototype

This model definitely suffered from the fact, that one never has all the desired bricks of right colors but it gives a good first impression of how the model could work.

In order to get started with a CAD model, the software studio from BrickLink might be a good choice and even young children of age 10+ are easily able to understand the concept of this software and build easy models. For more advanced models or those preferring open source, LDCad might be the preferred choice.

My first attempt to recreate the simple prototype resulted in the following model:

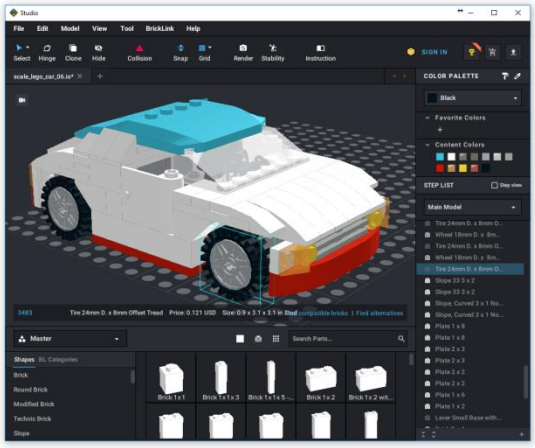

Figure 3: model opened in studio LEGO CAD software

Figure 2: photorealistic rendering generated with studio

The good thing about these LEGO® CAD programs is that you have an endless supply of all the bricks that ever existed. Because of which I was able to already apply some improvements. The model is publically available and can be downloaded at bricklink.com. So if you want to take a look, just go ahead and try yourself. You will also find models on bricklink.com, as well as on the OMR (Official Model Repository) in ldraw.org many other models of the same kind, just to investigate and play with.

Also on bricklink.com you can create a cart with all the needed physical bricks for the uploaded model and BrickLink will give you the possibility to order used or new bricks from numerous online stores that are specialized in selling LEGO® bricks. This is quite a convenient thing to do and in the end the actual physical model for my car looked like this:

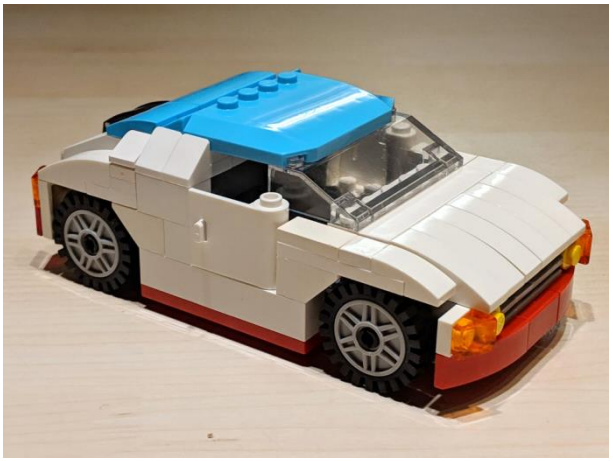

Figure 4: final build of scale car model

When trying to get from the ldraw format to actual LS-DYNA simulation models, in order to have some virtual fun with crashing LEGO® models, it seemed natural to use a similar approach as in the ldraw format. In this format basically every line is just providing the transformation information, a color and a reference to one brick in the standardized ldraw brick library.

The idea is, when having a library of meshed bricks in the same positions as the bricks in the ldraw library, it would be easy to create simulation models from any of the countless downloadable ldraw

models. Maybe at some point even easy enough for young students. With this approach each brick would have to be meshed only once and can then be used over and over again by using **`*INCLUDE_TRANSFORM`** cards to import the brick to their various locations.

Since I wanted to use the models also as a show case and later maybe as class example for our Simulation Data Management system LoCo, I wrote a small script for using LDCad directly from within LoCo and returning the individual bricks of the ldraw file as individual parts (components) in LoCo. This also allows me to structure the car model hierarchically and create a part structure just as I would probably do with real car models. In LoCo this looks like this:

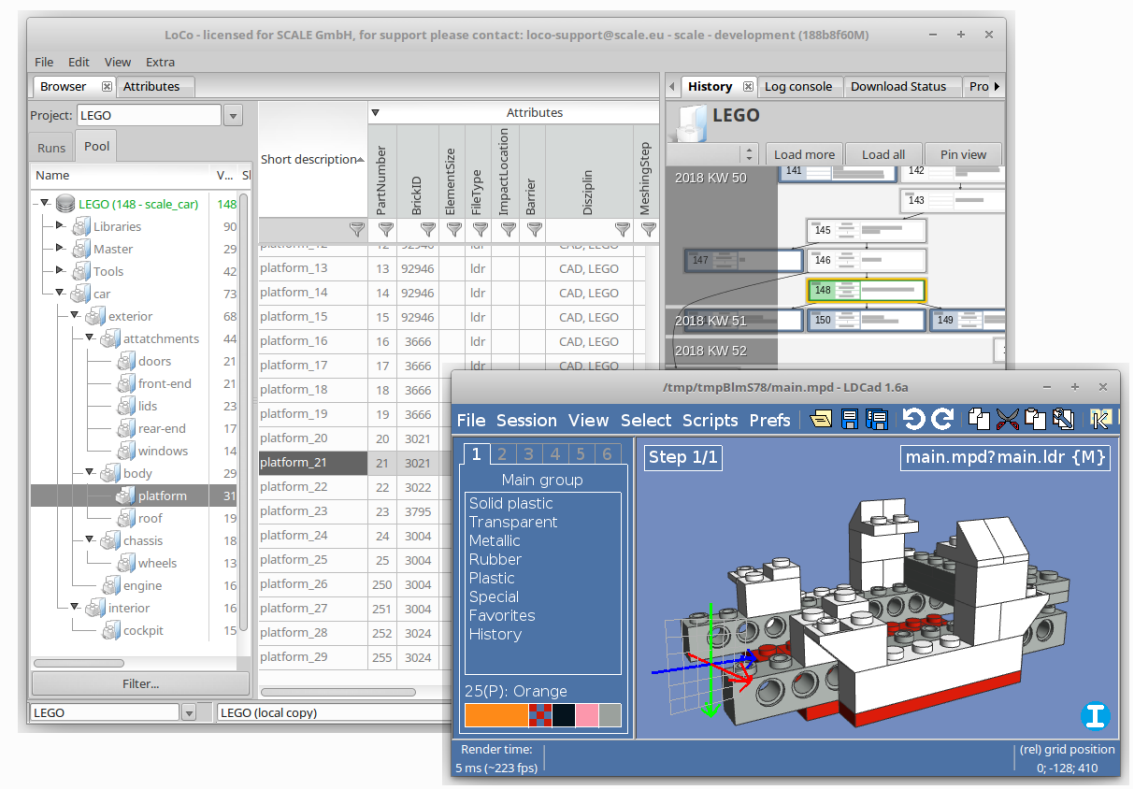

Figure 5: working with CAD data in LDCad directly in Simulation Data Management system LoCo

The different substructures of the model can be opened and worked with directly from LoCo in the CAD system (here LDCad) and upon saving the changes will be stored back to LoCo. New versions of all files and folders are created and synchronized automatically to all other team members working on the project. LoCo acts here also as a collaboration platform giving the possibility to try out different versions of the actual LEGO® model or create a model in cooperation with others.

Apart from the CAD data also the LS-DYNA models for each brick are stored in LoCo and mounted as a library:

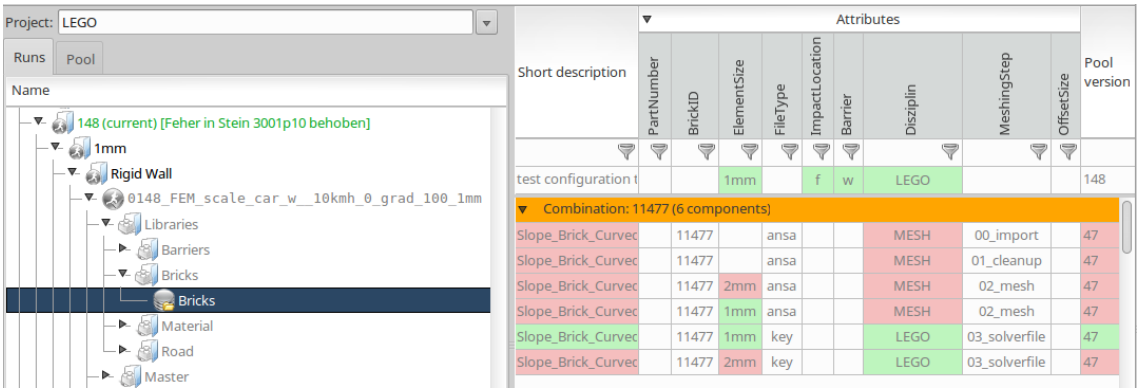

Figure 6: managing the library of meshed bricks in LoCo

Other files needed such as a master file containing all the control cards, barrier models, files with material cards and so on are also stored and maintained through LoCo.

Finally a script is kept together with all the other components of the model, which runs when assembling an actual LS-DYNA solver deck from LoCo. This script takes the transformation information for each brick and translates them to LS-DYNA `*INCLUDE_TRANSFORM` and `*DEFINE_TRANSFORMATION` cards such that each `*.key` file is imported over and over again.

```
$#########################################################################
$ Include - Transform for:
$ dashboard2_-_nnn_____________558_3070b_000000_----_3b16a7a2.ldr
$#########################################################################
$
*DEFINE_TRANSFORMATION
  20115001
$
$  rotation and translation from *.ldr file:
$
$  1 0 -0.567 -200 -274.567 -1 0 0 0 0 -1 0 -1 0 3070b.dat
$_____________________________________________________________________
$
$  Rotation:
$
    ROTATE       1.0       0.0       0.0       0.0       0.0       0.0      90.0
$
$ um X
    ROTATE       1.0       0.0       0.0       0.0       0.0       0.0       -90
$ um Y
    ROTATE       0.0       1.0       0.0       0.0       0.0       0.0        -0
$ um Z
    ROTATE       0.0       0.0       1.0       0.0       0.0       0.0       180
$
$_____________________________________________________________________
$
$  Translation:
$          x         y         z
    TRANSL   -0.2268       -80  -109.827
$
$ final rotations
$
    ROTATE       1.0       0.0       0.0       0.0       0.0       0.0     -90.0
    ROTATE       0.0       0.0       1.0       0.0       0.0       0.0     -90.0
$
$_____________________________________________________________________
$
*INCLUDE_TRANSFORM
Tile__1_x__1_with_Groove_____________________3070b_1mm03_.key
$#  idnoff    ideoff    idpoff    idmoff    idsoff    idfoff    iddoff
  20115000  20115000  20115000         0  20115000  20115000  20115000
$#  idroff
  20115000
$#  fctmas    fcttim    fctlen    fcttem    incout

$#  tranid
  20115001
$
```

Figure 7: automatically generated solver cards for including bricks with transformation on different positons

The same script also creates session files for LS-PREPOST, Animator and META for correct coloring when looking at the crash animations.

Having such a setup in place it makes it easy to apply changes to the LEGO® CAD model by using e.g. LDCad or importing other LEGO® models and run these as new simulations, provided the meshed bricks already exist for the bricks used in the model. And meshing these bricks is actually the most elaborate part when setting up such models.

The original CAD data for each brick from the ldraw brick library is mostly in the form of a simple mesh made out of triangles representing the surface of the brick. This can easily be converted to `*.stl` or `*.obj` files using leocad or LDView and imported to ANSA or Hypermesh. However these meshes are not suitable at all for conducting simulations. So they can only be used as a basis for creating meshes suited for FEA. Initial experiments trying to use the simple original STL meshes as rigid bodies have not been very promising and therefore we decided to create mostly tetrahedral volume meshes with a target edge length of 1mm and 2mm. However even with 1mm it is sometimes not possible to capture every detail of some bricks which however is crucial for the behavior when two bricks separate.

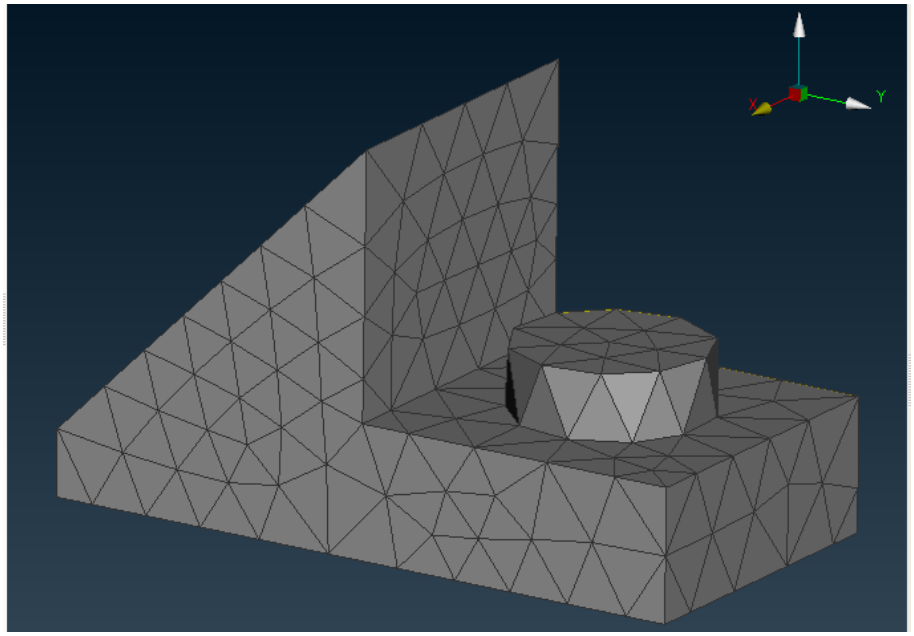

Figure 8: 1mm mesh for brick with ldraw ID:92946

For the moment the models are held together solely by contact and friction which is not quite capturing the physical truth because in reality the bricks are also hold together by some kind of clamping force. We are still investigating how to use tied break contacts to improve this behavior but even without it simulation results already look interesting.

In LoCo I set up many different load cases with different velocities, impact angles, and barrier offsets. This is actually quite easy once all the bricks where meshed. The results of a 17km/h 25% offset barrier crash can be seen in this video on YouTube.

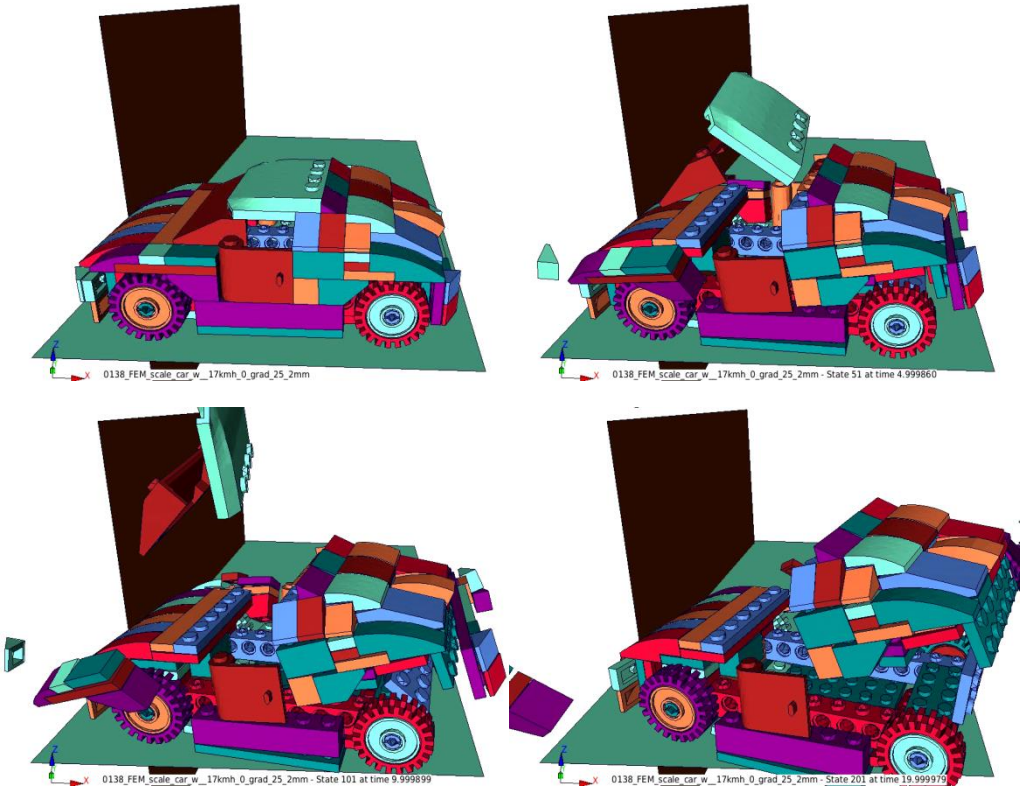

Figure 9: 25% offset crash visualized in LS-PrePost at 0, 5, 10 and 20ms

When seeing these results I've not been very convinced about them since the car is heavily bouncing backward. I expected that it would get a lot more spin when hitting the barrier only with one edge of the car. So I decided that it would need physical validation and I spend a weekend to actually construct a small wooden crash sled.

Figure 10: crash sled (left) and positioned car for 25% overlap frontal crash (right)

The sled is driven by a slingshot and can accelerate the car to approximately 15-20km/h which is way higher than one would think and made the bricks of the car fly all over the place. That's why I had to construct a little housing in front of the sled such that the bricks couldn't hurt anyone (especially my kids) and also I would be able to find all the bricks after each crash easily.

Figure 11: crash sled from below with slingshot mechanism

The barrier is made out of hard wood and can be placed in different positions to represent different load cases.

- 100% frontal
- 50% frontal offset
- 25% frontal offset
- 30° rotated 100% frontal

The test than is filmed using the slow motion mode of two smart phones. One was placed on top of the housing to create a video showing the crash from above. The other was placed on the left side of the crash sled to provide a side view video. Both cameras (Samsung S7 and Pixel 2) have been able to capture video at 240fps and produced reasonable results for such a low budget setup. The videos mostly where suffering from the vibrations in the sled but also from a heavy rolling shutter effect, which is actually also fun to explain to the kids.

Two videos comparing the side view results from simulation and crash as well as the top view are also posted on YouTube. And for a simulation with such little tweaking for validation it is already impressive how well the results of the simulation and the test compare.

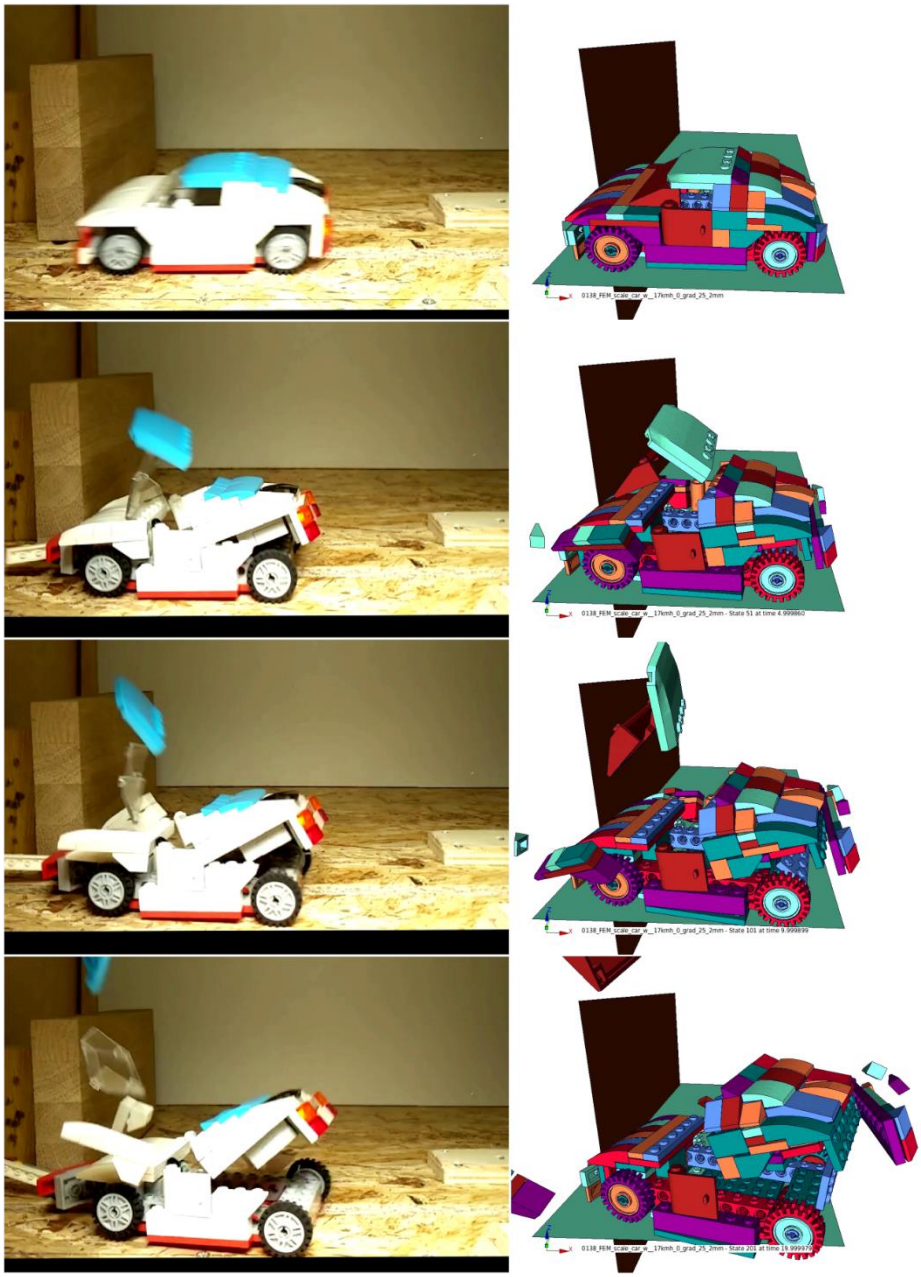

Figure 12: physical test compared to simulation for 25% offset crash at 0, 5, 10 and 20ms

The current setup with LoCo makes it now relatively easy to setup a simulation model from LEGO® models. However the usability of all this is not quite there yet such that young students can handle it. The final goal would be that they could download models from the OMR of Ldraw or bricklink.com, define easily some boundary conditions and run crash simulations. This will require further improvements of our Simulation Data Management System LoCo which we might achieve in our current development of LOCOX, which will be the successor of LoCo. At least this will be the benchmark for the usability and ease of use for the new user interface.

In the end… simulation has to become a child's play in order to influence new generations and pass on the fascination of physics and simulation.

(marko.thiele@scale.eu)