Assignment 5 – DSL for Custom-Made Lego Bricks

Submit Assignment

Due 23 Mar by 23:59 **Points** 1 **Submitting** a file upload
Available 5 Mar at 13:30 - 23 Mar at 23:59 18 days

Model-Driven Engineering, VT2019

Thorsten Berger, thorsten.berger@chalmers.se

Sergio Garcia, gsergio@chalmers.se

Additional remarks for all assignment deliverables

- *State the authors of the deliverable (the group members)*
- *Correct language use, no grammar or spelling errors*
- *Reference and describe all figures/tables in the text*
- *Figures and graphs should be readable from a quality perspective*
- *Reference literature in your text where appropriate*
- *Define non-obvious acronyms*
- *The deliverable should be easily readable, understandable and complete*
- *Give arguments for your decisions (also using references)*
- *Show critical thinking*
- *Be prepared to get frustrated if something does not work as you think it should. Rise to the challenge!*

Note

You will provide printable models for three showcase Lego bricks generated with your tool. We will print one of the three showcases with our 3D printer. You are more than welcome to observe the print or even to control the printer yourselves, if you like.

Assignment 5 – DSL for Custom-Made Lego Bricks

Hard deadline for Handing via Canvas: 23.3.14:00 (CET)

Domain-specific languages (DSLs) are often used in context of embedded systems and industrial machines to allow engineers, who have no higher education in software development, to nonetheless produce custom-made solutions. In assignment 6 you should create such a DSL to enable the creation of custom-made Lego bricks through 3D printing.

A. Lego Brick DSL

The goal of the Lego Brick DSL is to generate scripts that define valid input models for a 3D printer (see more about the tool chain below).

The DSL should allow to compose complex Lego bricks out of simple components.

- Define the maximum size for bricks that will be printed (Note: this is important, since printers are limited in the size of objects that can be printed and different printers have different limits.
- Define basic Lego bricks (i.e., blocks), by defining the size in terms of the number of studs in width and length. This can be an absolute number or it can be defined in relation to the size of another brick.
- Define complex Lego bricks as a combination of other (basic) bricks.
- Differentiate between bricks that only act as intermediate building blocks and bricks for which a model will be generated (e.g., in the example DSL in Figure 1, this is indicated by the presence or absence of the keyword *abstract*).

```
Limits{
length = 10
width = 10
}
abstract Brick I1
{ length = 2
  width = 2*I2.width
}
abstract Brick I2
{
  length = 3
  width = 2
}
Brick I3{
   assemble {
   I1.width, I2.width, central, central
}}
```

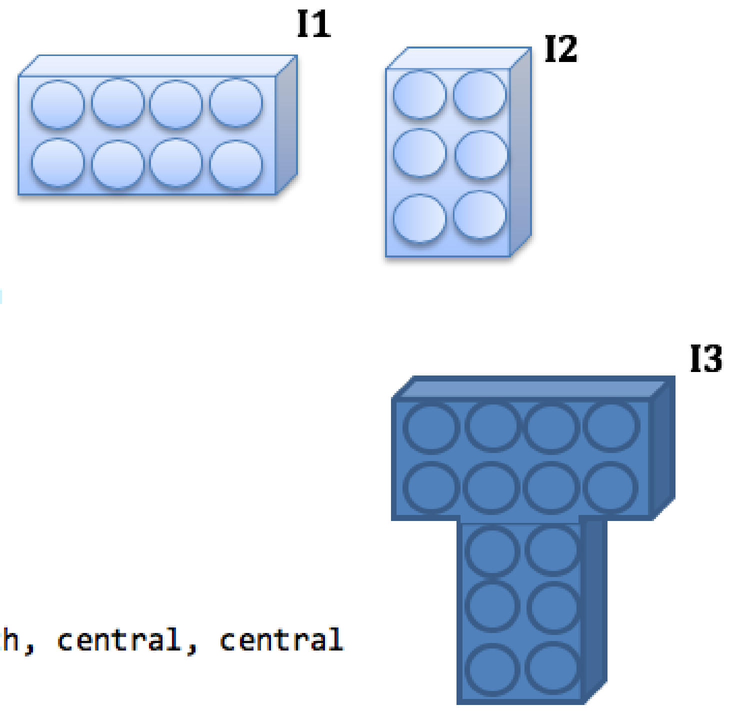


Figure 1 Example how a DSL could look like and sketches of abstract and concrete bricks defined with the DSL

Part of this task is to be creative, but also to be reasonable. There is a multitude of possibilities for combining bricks. Our suggestion is to start with a very simple language fulfills the requirements above, but has a simple assembly mechanism, which can be extended later. Note that the assembly mechanism illustrated in our example in Figure 1 is already quite complex. Discuss a good strategy with your supervisor.

B. Tool Chain

Figure 2 illustrates the tool chain for this task. Note that you only have to take care of the first part of the chain. However, it helps to understand the whole chain.

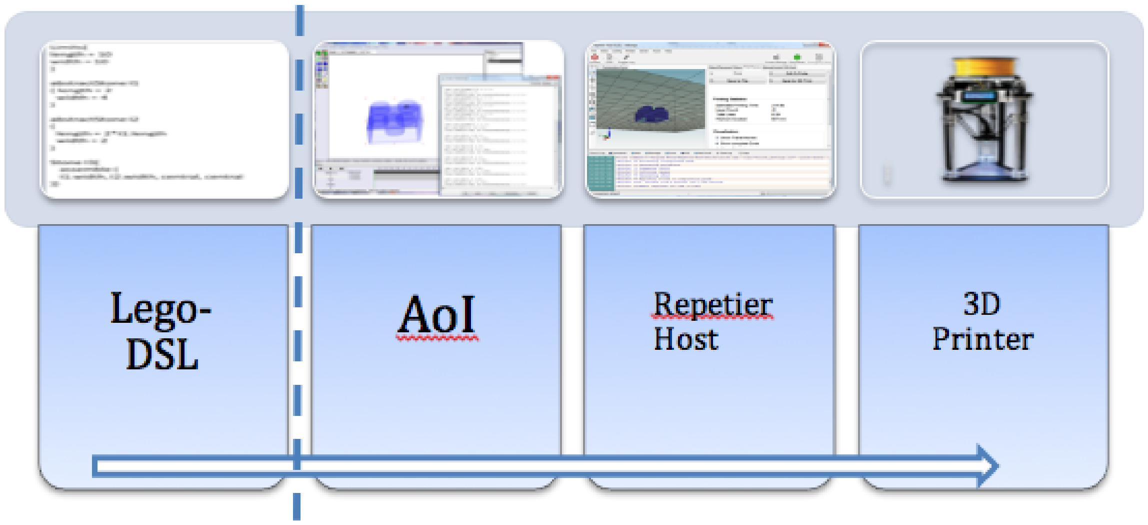
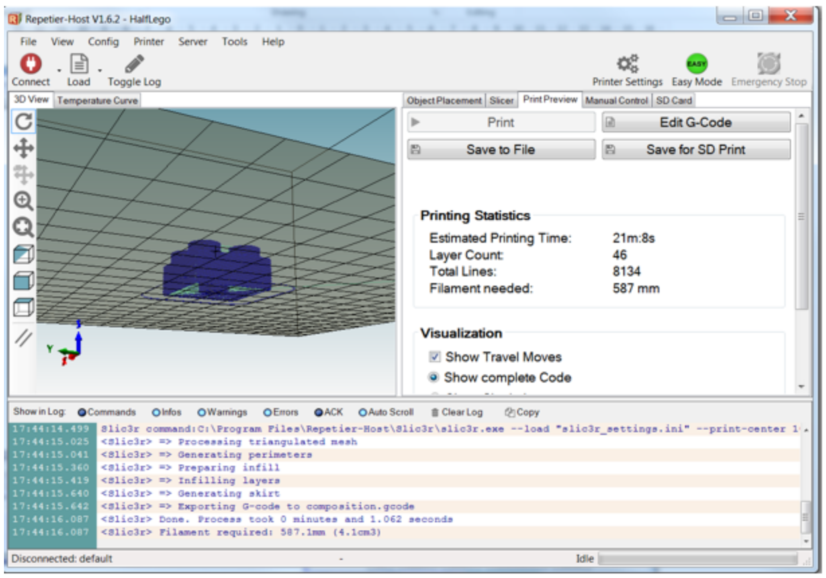


Figure 2 Tool Chain - The parts right to the dashed arrow are given. The Lego DSL shall create a Groovy script that creates the desired Lego brick in AoI.


Repetier-Host V1.6.2 - HalfLego
File View Config Printer Server Tools Help
Connect Load Toggle Log
Printer Settings Easy Mode Emergency Stop
3D View Temperature Curve
Object Placement Slicer Print Preview Manual Control SD Card
Print Edit G-Code
Save to File Save for SD Print
Show in Log: Commands Infos Warnings Errors ACK Auto Scroll Clear Log Copy

Printing Statistics

Estimated Printing Time:	21m:8s
Layer Count:	46
Total Lines:	8134
Filament needed:	587 mm

Visualization

[✓] Show Travel Moves

[✓ Show complete Code]

```
17:44:14.499 Slic3r command:C:\Program Files\Repetier-Host\Slic3r\slic3r.exe --load "slic3r_settings.ini" --print-center 1
17:44:15.025 <Slic3r> => Processing triangulated mesh
17:44:15.041 <Slic3r> => Generating perimeters
17:44:15.360 <Slic3r> => Preparing infill
17:44:15.419 <Slic3r> => Infilling layers
17:44:15.640 <Slic3r> => Generating skirt
17:44:15.642 <Slic3r> => Exporting G-code to composition.gcode
17:44:16.087 <Slic3r> Done. Process took 0 minutes and 1.062 seconds
17:44:16.087 <Slic3r> Filament required: 587.1mm (4.1cm3)
```

Disconnected: default - Idle

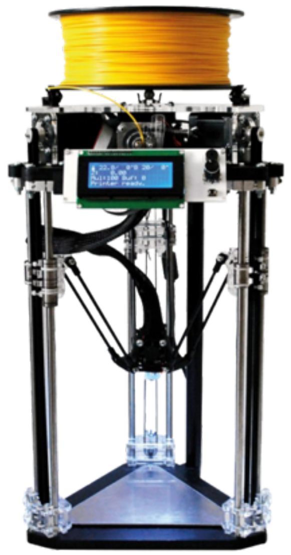
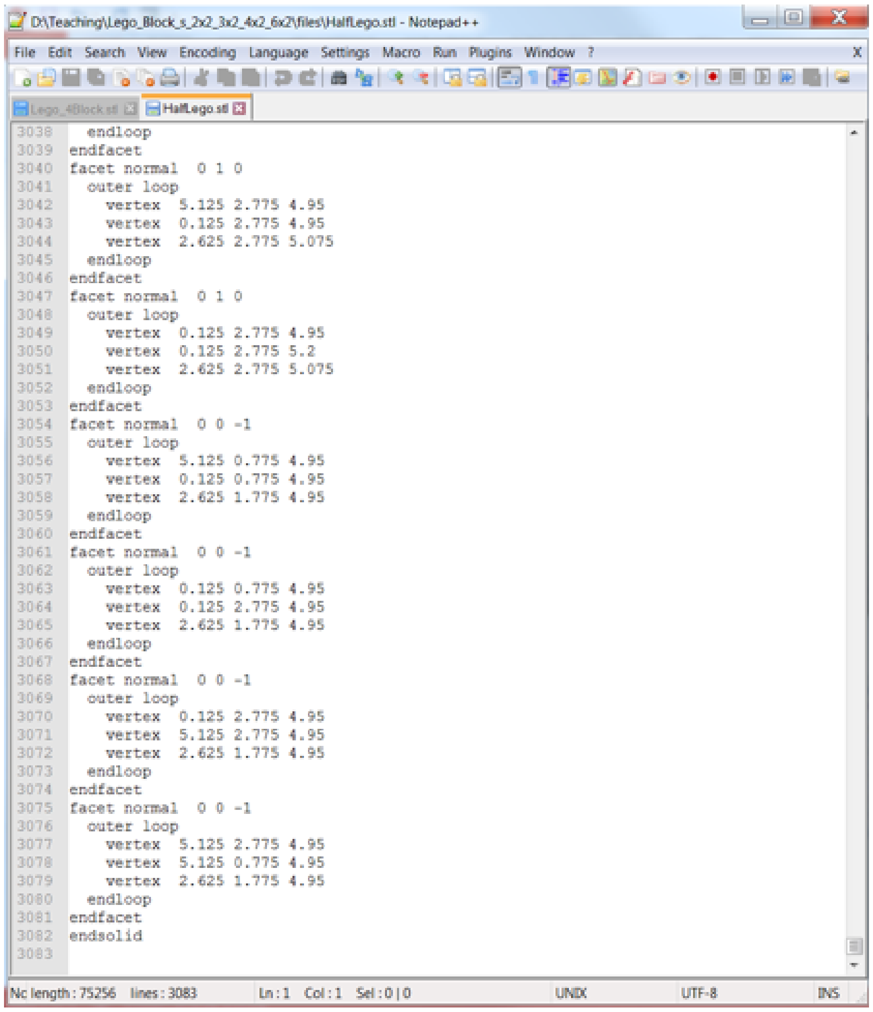


Figure 3 Screenshot of the Repetier-Host and μDelta 3D printer

The 3D Printer will be controlled with the open tool Repetier Host (see Figure 3).

Repetier Host inputs a 3D model in the STL (STereoLithography[1]) format, which describes 3D objects by means of triangles. In this format, already simple objects consist of hundreds of triangles. For example, a model of a simple 2*2 Lego brick consists of 434 triangles (making up for 3042 lines of code in the respective STL file, as shown in Figure 4). You can download the tool here[2].


D:\Teaching\Lego_Block_s_2x2_3x2_4x2_6x2\files\HalfLego.stl - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
Lego_4Block.stl
HalfLego.stl
Nc length : 75256 lines : 3083
Ln : 1 Col : 1 Sel : 0 | 0
UNIX
UTF-8
INS

```
  endloop
endfacet
facet normal  0 1 0
  outer loop
    vertex  5.125 2.775 4.95
    vertex  0.125 2.775 4.95
    vertex  2.625 2.775 5.075
  endloop
endfacet
facet normal  0 1 0
  outer loop
    vertex  0.125 2.775 4.95
    vertex  0.125 2.775 5.2
    vertex  2.625 2.775 5.075
  endloop
endfacet
facet normal  0 0 -1
  outer loop
    vertex  5.125 0.775 4.95
    vertex  0.125 0.775 4.95
    vertex  2.625 1.775 4.95
  endloop
endfacet
facet normal  0 0 -1
  outer loop
    vertex  0.125 0.775 4.95
    vertex  0.125 2.775 4.95
    vertex  2.625 1.775 4.95
  endloop
endfacet
facet normal  0 0 -1
  outer loop
    vertex  0.125 2.775 4.95
    vertex  5.125 2.775 4.95
    vertex  2.625 1.775 4.95
  endloop
endfacet
facet normal  0 0 -1
  outer loop
    vertex  5.125 2.775 4.95
    vertex  5.125 0.775 4.95
    vertex  2.625 1.775 4.95
  endloop
endfacet
endsolid
```

Figure 4 Example STL file for a simple Lego brick

To avoid creating that by hand, graphic tools like ArtOfIllusion (AoI)[3] can be used. You can download and install the tool here[4].

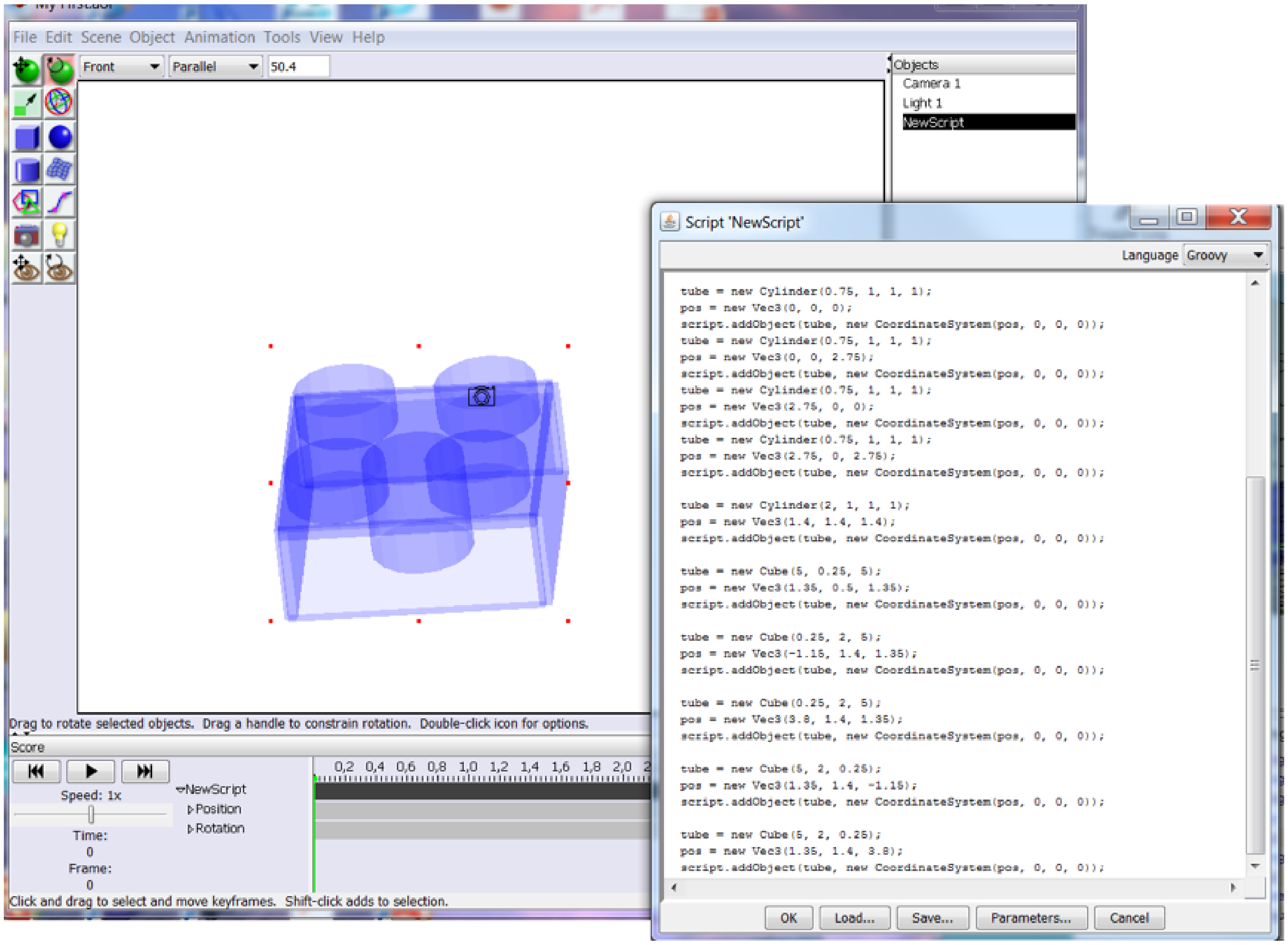
ArtOfIllusion To create an STL file from objects in ArtOfIllusions you have to install a standard plugin[5]:

- Open ArtOfIllusions and select from the menu: "tools->Scripts & Plugins Manager" and go to the tab "Install"
- The tool automatically searches for available plugins and shows them on the left side in the Plugin Folder.
- Select the STLTranslator plugin and press "Install Selected Files"
- As an alternative you might search for the STLTranslator.jar at the sourceforge page of ArtOfIllusions and copy it directly to the ArtOfIllusions/Plugins directory. Note that you will need the rights to write in that directory.

Note: make sure that you work with the latest version of AoI. Some older versions do not support that plugin.

To export the STL, select the object in the scene (!) and go to menu "File->Export->STL".

Create ArtOfIllusions objects: ArtOfIllusions allows to create objects by hand. However, we are interested in the option to use Groovy scripts. One such a script is shown in Figure 5 (the shown script is the complete code required to create the shown simple Lego brick; see Appendix). A simple tutorial on how to write such scripts by hand can be found here[6]. Furthermore, you can access an API Documentation on the ArtOfIllusions Website[7].



File Edit Scene Object Animation Tools View Help
Front
Parallel
50.4
Objects
Camera 1
Light 1
NewScript
Drag to rotate selected objects. Drag a handle to constrain rotation. Double-click icon for options.
Score
Speed: 1x
Time: 0
Frame: 0
NewScript
Position
Rotation
Click and drag to select and move keyframes. Shift-click adds to selection.
Script 'NewScript'
Language Groovy
OK
Load...
Save...
Parameters...
Cancel

```
tube = new Cylinder(0.75, 1, 1, 1);
pos = new Vec3(0, 0, 0);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));
tube = new Cylinder(0.75, 1, 1, 1);
pos = new Vec3(0, 0, 2.75);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));
tube = new Cylinder(0.75, 1, 1, 1);
pos = new Vec3(2.75, 0, 0);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));
tube = new Cylinder(0.75, 1, 1, 1);
pos = new Vec3(2.75, 0, 2.75);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));

tube = new Cylinder(2, 1, 1, 1);
pos = new Vec3(1.4, 1.4, 1.4);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));

tube = new Cube(5, 0.25, 5);
pos = new Vec3(1.35, 0.5, 1.35);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));

tube = new Cube(0.25, 2, 5);
pos = new Vec3(-1.15, 1.4, 1.35);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));

tube = new Cube(0.25, 2, 5);
pos = new Vec3(3.8, 1.4, 1.35);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));

tube = new Cube(5, 2, 0.25);
pos = new Vec3(1.35, 1.4, -1.15);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));

tube = new Cube(5, 2, 0.25);
pos = new Vec3(1.35, 1.4, 3.8);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));
```

Figure 5 ArtOfIllusion Lego Brick object with script

Your Lego DSL should build on top of the described tool chain, creating the Groovy scripts that build the correct 3D model for your custom-made Lego bricks in ArtOfIllustion. Therefore:

- Create a Meta-Model for your DSL. Reflect what concepts you need for definition and combination of bricks. For example, on which sides shall bricks be connected? If the sides of the bricks are not the same length, how can it be defined where the bricks' sides touch each other (e.g. should the centers of both bricks' sides meet)?
- Create a textual concrete syntax for your DSL using Xtext. Use it to generate an editor for you DSL.
- Create an OCL constraint to indicate to the user when concrete bricks are violating the size limit defined for the printer. Define further OCL constraints as needed.
- Create a model-to-text (M2T) transformation to generate the Groovy scripts from your DSL. You can use stringtemplate, which was discussed in a lecture, or any other M2T language, such as Xpand, which is relatively simple to use, or Xtend.

Deliverable

- Document (pdf) reporting about the results for assignment 6 as described above.
- Maximum 6 pages for the content from this assignment, 12p font size
- A ZIP-archive of all plugins you developed/generated.
- 3 printable showcases of Lego bricks created with your DSL, including for each showcase:
    - The model in your DSL that specifies the Lego stone.
    - The generated Groovy script that is input for ArtOfIllusions.
    - The printable .stl extracted from ArtOfIllusions (make sure that it can be opened on Repetier Host).

Appendix

Example Groovy Script for a simple 2*2 Lego Brick:

```
//2*2 cylinders on top of the brick
tube = new Cylinder(0.75, 1, 1, 1);
pos = new Vec3(0.75, 0, 0.75);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));

tube = new Cylinder(0.75, 1, 1, 1);
pos = new Vec3(0.75, 0, 3.5);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));

tube = new Cylinder(0.75, 1, 1, 1);
pos = new Vec3(3.5, 0, 0.75);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));

tube = new Cylinder(0.75, 1, 1, 1);
pos = new Vec3(3.5, 0, 3.5);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));


//1 cylinder on the inside of the brick
tube = new Cylinder(2, 0.95, 0.95, 1);
pos = new Vec3(2.125, 1.4, 2.125);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));

//1 cover plate of the brick carrying the 2*2 top cylinders
cube = new Cube(5, 0.25, 5);
pos = new Vec3(2.1, 0.5, 2.1);
```

```
script.addObject(cube, new CoordinateSystem(pos, 0, 0, 0));

//4 vertical sides on the brick
cube = new Cube(0.25, 2, 5);
pos = new Vec3(-0.4, 1.4, 2.1);
script.addObject(cube, new CoordinateSystem(pos, 0, 0, 0));

cube = new Cube(0.25, 2, 5);
pos = new Vec3(4.55, 1.4, 2.1);
script.addObject(cube, new CoordinateSystem(pos, 0, 0, 0));

cube = new Cube(5, 2, 0.25);
pos = new Vec3(2.1, 1.4, -0.4);
script.addObject(cube, new CoordinateSystem(pos, 0, 0, 0));

cube = new Cube(5, 2, 0.25);
pos = new Vec3(2.1, 1.4, 4.55);
script.addObject(cube, new CoordinateSystem(pos, 0, 0, 0));
```

[1]See **http://www.fabbers.com/tech/STL_Format** or **https://en.wikipedia.org/wiki/STL_(file_format)** for details

[2] https://www.repetier.com/

[3] http://www.artofillusion.org/documentation

[4] http://www.artofillusion.org/downloads

[5] http://reprap.org/wiki/Art_of_illusion#9._Hint:_Tidy_the_triangulation_before_writing_an_STL_file

[6] http://aoi.sourceforge.net/docs/scripttut/chapter2.html

[7] http://www.artofillusion.org/documentation#javadocs