DAT240 / DIT596
Assignment 5
DSL for Custom-Made Lego Bricks

Model-Driven Engineering

Group 1

Supriya Supriya (supe@student.chalmers.se)
Himanshu Chuphal (guschuhi@student.gu.se)
Kristiyan Dimitrov (gusdimkr@student.gu.se)

List of Figures

Introduction

Domain-specific languages (DSLs) are often used in context of embedded systems and industrial machines to allow engineers, In assignment 5, we are creating such a DSL to enable the creation of custom-made Lego bricks through 3D printing.

A. Lego Brick DSL

The DSL should allow composing complex Lego bricks out of simple components. To begin with, we have defined a metamodel as in Figure 1.

The BrickModel is the root class that comprises the bricks which are supposed to be limited in their length and width as defined in the Limits Class. The width and height of each brick refer to the Class Size via dimensions[0..1]. In addition to it, the getSize reference gives an option for the bricks size to be defined in accordance with the size of other bricks. The class Complex Brick is a combination of multiple intermediate Abstract bricks whose positions of the combination are assigned with reference hasPosition[0..*], to the Class Position. These positions can be either centre, left or right.

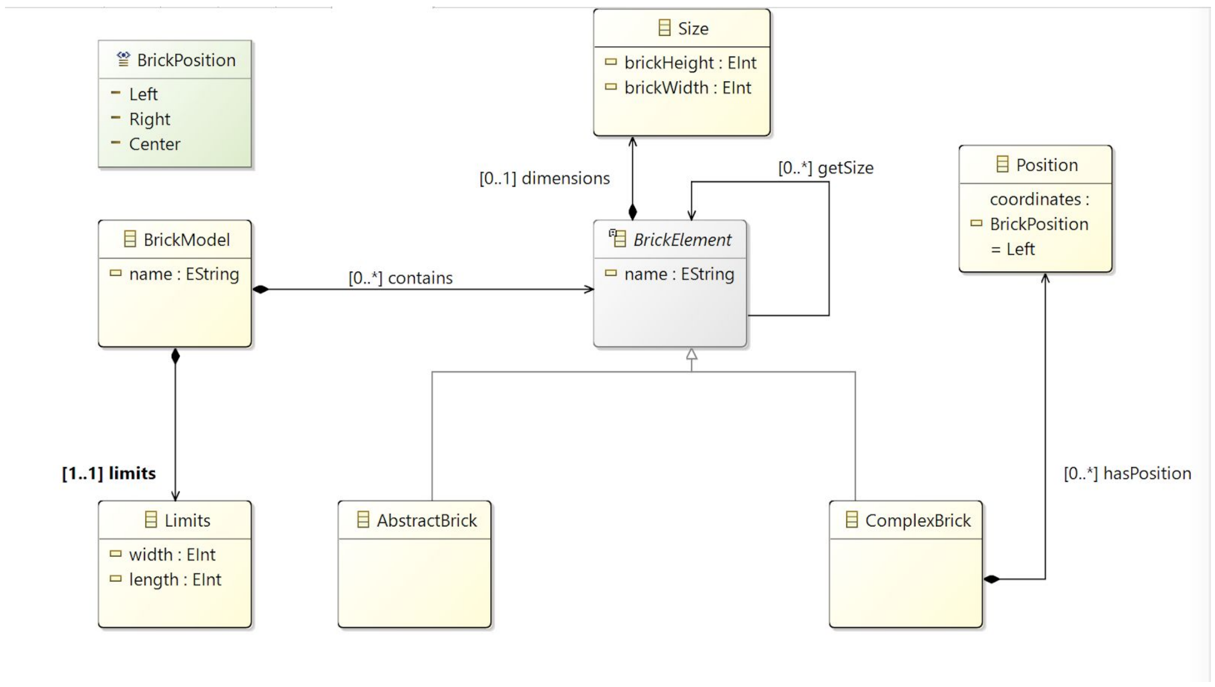
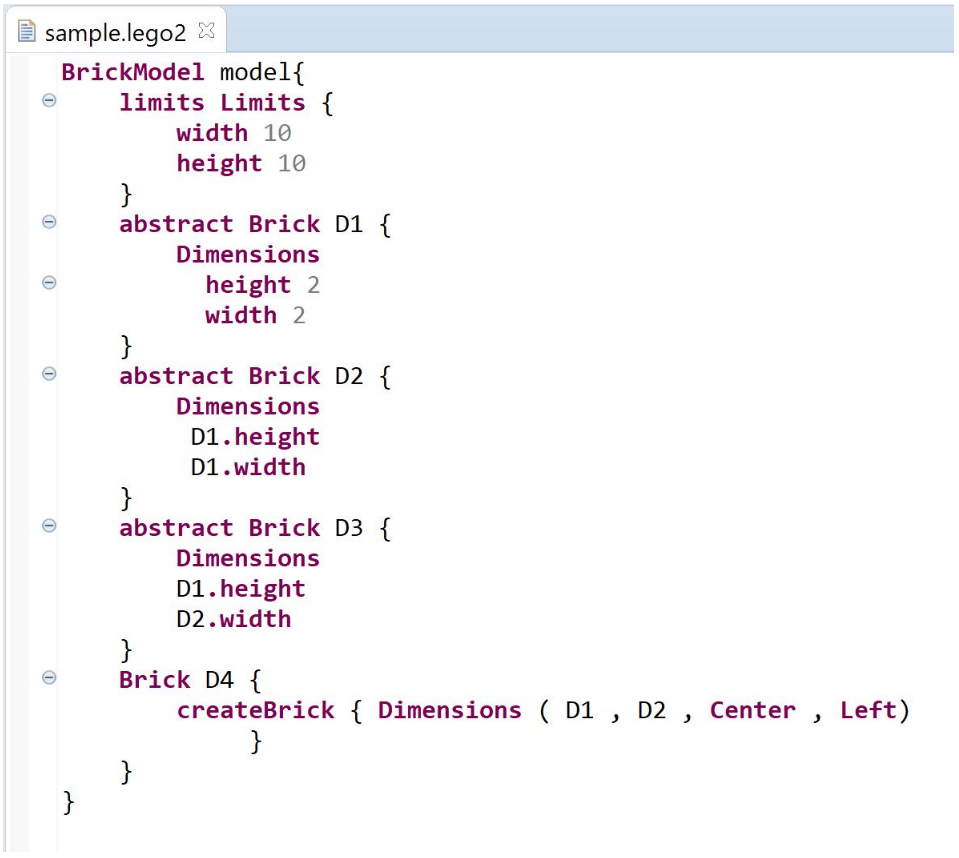


Figure1: MetaModel of the Lego Bricks

B. Textual Concrete Syntax

The metamodel is converted into Xtext nature and the grammar is generated from it. Using it, the concrete syntax is written as in Figure2.



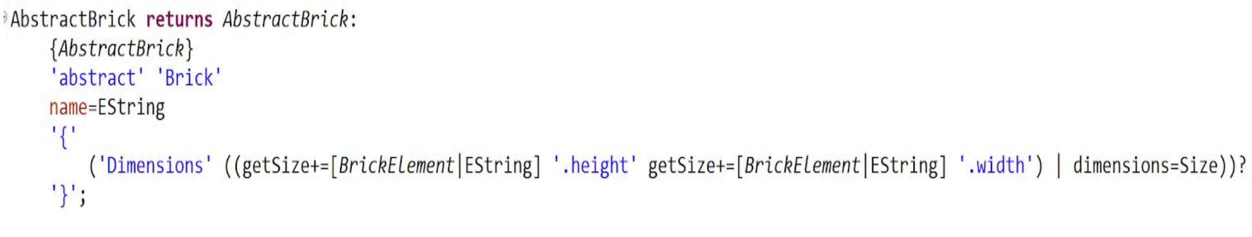
sample.lego2

```
BrickModel model{
    limits Limits {
        width 10
        height 10
    }
    abstract Brick D1 {
        Dimensions
          height 2
          width 2
    }
    abstract Brick D2 {
        Dimensions
         D1.height
         D1.width
    }
    abstract Brick D3 {
        Dimensions
        D1.height
        D2.width
    }
    Brick D4 {
        createBrick { Dimensions ( D1 , D2 , Center , Left)
             }
    }
}
```

Figure 2: Concrete syntax for the metamodel

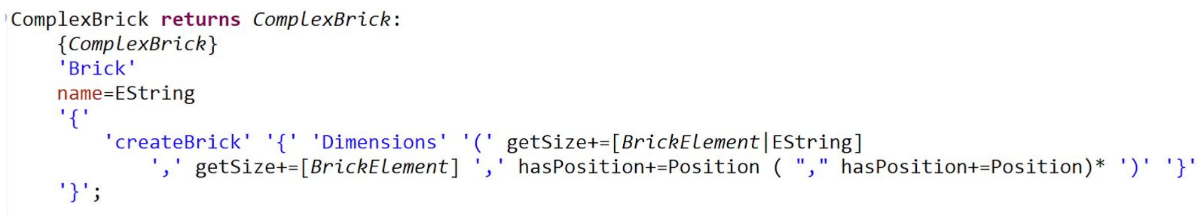
To generate the above concrete syntax, apart from the changes made in the original grammar generated from xtext through the modifications of terminals, the following two main modifications were made:

- Unlike the two separate statements for the dimensions of the Abstract brick as in original grammar, we used the ‘OR’ condition to merge into one. Now the dimensions of any Abstract Brick can have its size from the other brick ‘or’ can be an absolute number.



```
AbstractBrick returns AbstractBrick:
    {AbstractBrick}
    'abstract' 'Brick'
    name=EString
    '{'
        ('Dimensions' ((getSize+=[BrickElement|EString] '.height' getSize+=[BrickElement|EString] '.width') | dimensions=Size))?
    '}';
```

- Similarly, for the ComplexBrick, the grammar is modified to assemble all the abstract bricks involved in the formation of Complex Brick along with their positions.



```
ComplexBrick returns ComplexBrick:
    {ComplexBrick}
    'Brick'
    name=EString
    '{'
        'createBrick' '{' 'Dimensions' '(' getSize+=[BrickElement|EString]
            ',' getSize+=[BrickElement] ',' hasPosition+=Position ( "," hasPosition+=Position)* ')' '}'
    '}';
```

C. OCL Constraints

There are 2 OCL constraints defined for the ***BrickModel***. The context and expressions are defined as :

1. `Context:` **`BrickModel`**

```
Expression:
invariant limitHeight: self.contains ->
        forAll(e:BrickElement|
e.dimensions.oclAsType(Size).brickHeight <=
(self.limits.oclAsType(Limits).length / 2));
```

For all the items in the returned collection of BrickElement in the BrickModel, enforces that the height of each Brick should be less than or equal to the limit of the length and returns true if all the bricks height fall under the limit. Since the limit defined is on complex brick and since the complex brick in our model is made of 2 abstract bricks, we have divided the length by 2, to distribute the limited length as height among the abstract bricks.

2. `Context:` **`BrickModel`**

```
Expression:
invariant limitWidth: self.contains ->
        forAll(e:BrickElement|
e.dimensions.oclAsType(Size).brickWidth <=
(self.limits.oclAsType(Limits).width / 2));
```

For all the items in the returned collection of BrickElement in the BrickModel, enforces that the width of each Brick should be less than or equal to the limit of the width and returns true if all the bricks width fall under the limit. Since the limit defined is on complex brick and since the complex brick in our model is made of 2 abstract bricks, we have divided the width by 2, to distribute the limited width among the abstract bricks.

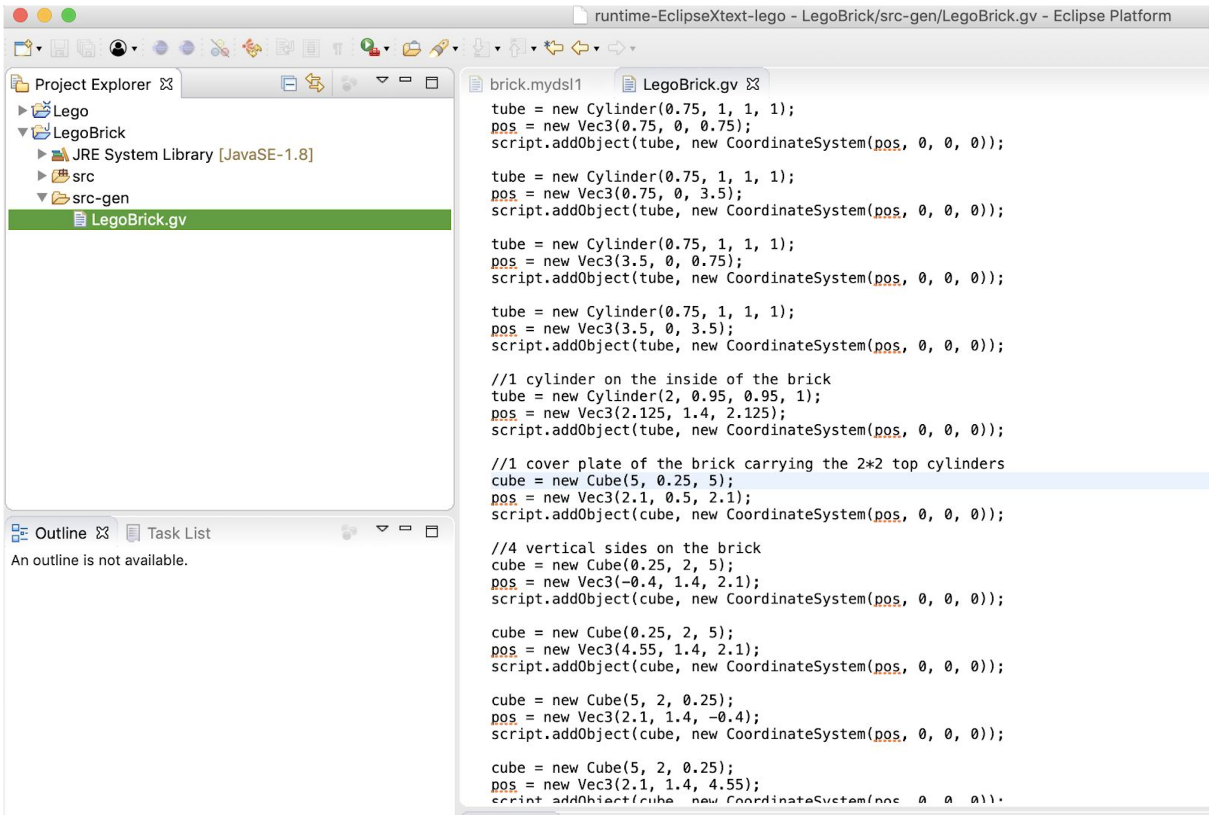
D. Model-To-Text Transformation

- ### Code Generation With XTend

For the Model to Text transformation we used a form of xtext/xtend templating. Similar to String Templates everything inside ‘’’ ‘’’ will be treated as a string. The only exceptions are simple operations like FOR loops and IF statements. Additionally, everything inside << >> similar to String templates will be replaced with the according method output that’s called or operation performed. In general we used FOR loops to iterate

- ### Generated code

Eclipse Running Application


runtime-EclipseXtext-lego - LegoBrick/src-gen/LegoBrick.gv - Eclipse Platform
Project Explorer
Lego
LegoBrick
JRE System Library [JavaSE-1.8]
src
src-gen
LegoBrick.gv
Outline
Task List
An outline is not available.
brick.mydsl1
LegoBrick.gv

```
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));
```

Figure 3: Generated code for the meta model

- ### Groovy Scripts (generated Using XTend)

```
//NEW BRICK CENTER
//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));


//NEW BRICK LEFT

//Cyllinders
tube = new Cylinder(0.75, 1, 1, 1);
pos = new Vec3(3.5, 0, 6);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));

tube = new Cylinder(0.75, 1, 1, 1);
pos = new Vec3(3.5, 0, 8.8);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));


tube = new Cylinder(0.75, 1, 1, 1);
```

```
pos = new Vec3(0.75, 0, 6);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));

tube = new Cylinder(0.75, 1, 1, 1);
pos = new Vec3(0.75, 0, 8.8);
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, 7.4);
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, 7.4);
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, 7.4);
script.addObject(cube, new CoordinateSystem(pos, 0, 0, 0));

cube = new Cube(0.25, 2, 5);
pos = new Vec3(4.55, 1.4, 7.4);
script.addObject(cube, new CoordinateSystem(pos, 0, 0, 0));

cube = new Cube(5, 2, 0.25);
pos = new Vec3(2.1, 1.4, 4.8);
script.addObject(cube, new CoordinateSystem(pos, 0, 0, 0));

cube = new Cube(5, 2, 0.25);
pos = new Vec3(2.1, 1.4, 9.8);
script.addObject(cube, new CoordinateSystem(pos, 0, 0, 0));

//NEW BRICK RIGHT

//Cyllinders
tube = new Cylinder(0.75, 1, 1, 1);
pos = new Vec3(3.5, 0, -1.75);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));

tube = new Cylinder(0.75, 1, 1, 1);
pos = new Vec3(3.5, 0, -4.55);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));

tube = new Cylinder(0.75, 1, 1, 1);
pos = new Vec3(0.75, 0, -1.75);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));

tube = new Cylinder(0.75, 1, 1, 1);
pos = new Vec3(0.75, 0, -4.55);
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, -3.15);
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, -3.2);
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, -3.2);
script.addObject(cube, new CoordinateSystem(pos, 0, 0, 0));

cube = new Cube(0.25, 2, 5);
pos = new Vec3(4.55, 1.4, -3.2);
script.addObject(cube, new CoordinateSystem(pos, 0, 0, 0));

cube = new Cube(5, 2, 0.25);
pos = new Vec3(2.1, 1.4, -0.65);
script.addObject(cube, new CoordinateSystem(pos, 0, 0, 0));

cube = new Cube(5, 2, 0.25);
pos = new Vec3(2.1, 1.4, -5.65);
script.addObject(cube, new CoordinateSystem(pos, 0, 0, 0));

//NEW BRICK TOP

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

tube = new Cylinder(0.75, 1, 1, 1);
pos = new Vec3(0.75, 2, 3.5);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));

tube = new Cylinder(0.75, 1, 1, 1);
pos = new Vec3(3.5, 2, 0.75);
script.addObject(tube, new CoordinateSystem(pos, 0, 0, 0));

tube = new Cylinder(0.75, 1, 1, 1);
pos = new Vec3(3.5, 2, 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, 3.4, 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, 3.5, 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, 3.4, 2.1);
script.addObject(cube, new CoordinateSystem(pos, 0, 0, 0));

cube = new Cube(0.25, 2, 5);
pos = new Vec3(4.55, 3.4, 2.1);
script.addObject(cube, new CoordinateSystem(pos, 0, 0, 0));

cube = new Cube(5, 2, 0.25);
pos = new Vec3(2.1, 3.4, -0.4);
script.addObject(cube, new CoordinateSystem(pos, 0, 0, 0));

cube = new Cube(5, 2, 0.25);
pos = new Vec3(2.1, 3.4, 4.55);
script.addObject(cube, new CoordinateSystem(pos, 0, 0, 0));
```

E.Usage and Limitations

Currently the model is capable of handling a narrow range of brick sizes.

- The model does not work with any brick with a height bigger than 2.
- It works only with even numbers for width: 2, 4 and 6. The reason behind this is due to how the positioning calculations work in artofillusion combined with a lack of time.

  Nevertheless the model can be extended by the use of more intricate mathematics to calculate brick positions and far more experience in artofillusion and 3D objects creation.

  For the actual creation of a complex brick it goes as follows: ***(brick 1, brick 2, position1, position2)***. Position 1 is which part of brick 1 will be connected to brick 2, similarly position 2 will be which part of brick 2 will connect to brick 1. For example if position 1 is center and position 2 is Left: the leftmost part of brick 2 will be connected to the center of brick 1 (above brick 1). Brick 2 always builds on top of brick 1.

  The M2T code generation using XTend has checks for above mentioned limitations and will generate final groovy scripts based on these limitations. After understanding the intricate mathematics to calculate brick positions, the code can be enhanced to make it work for any height and width of bricks.

F. Tools

1. ArtOfIllusions

Generated Groovy file (***LegoBrick.gv***) as input to the tool, we get following 3D output:

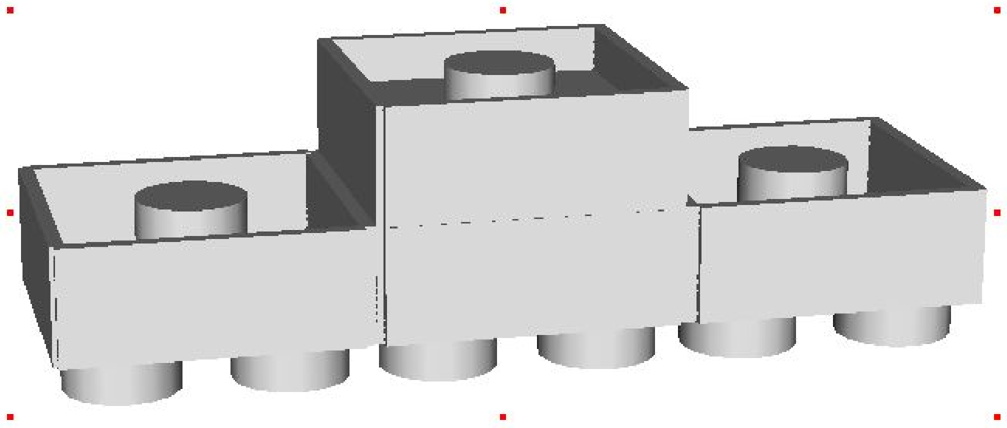


Figure 4: Generated 3D model of lego brick using ArtofIllusions tool

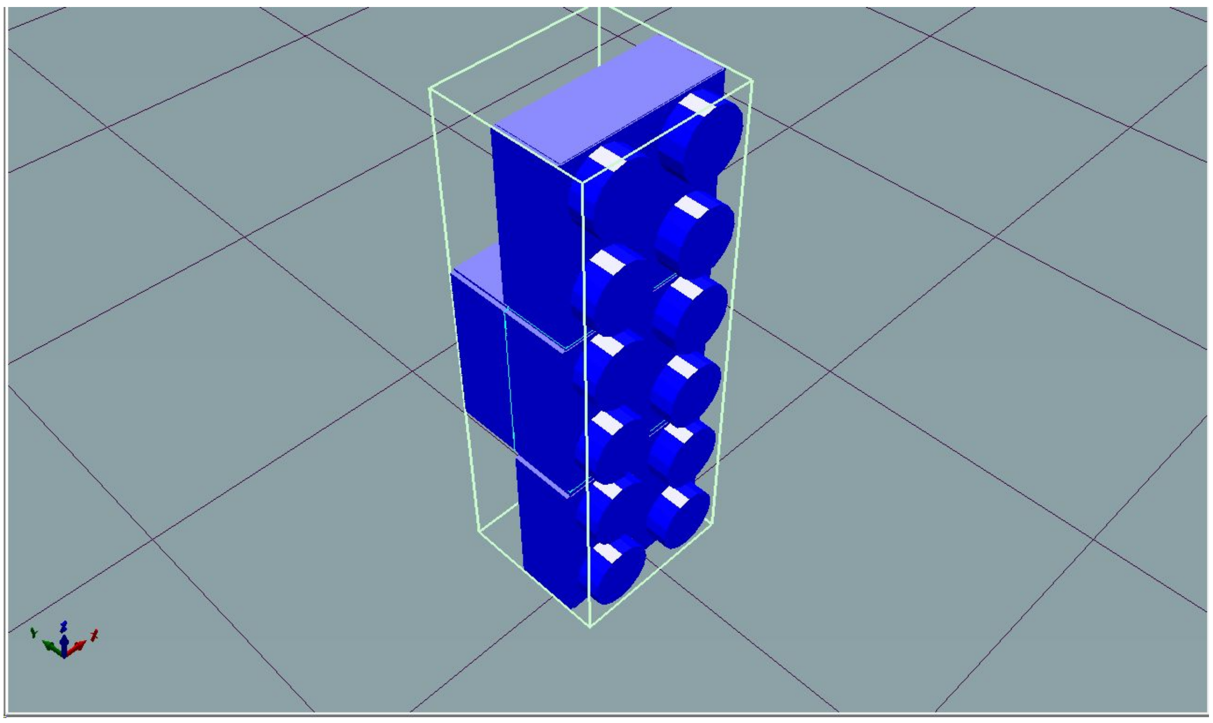


Figure 5: Generated 3D model for the Lego Brick using Repetier

References

1. http://en.wikipedia.org/wiki/DOT_graph_description_language (Links to an external site.)Links to an external site.
2. https://www.eclipse.org/xtend/documentation/203_xtend_expressions.html#switch-expression
3. https://bitbucket.org/itu-square/mdsebook/wiki/Home
4. Eclipse XTend https://www.eclipse.org/xtend/documentation/203_xtend_expressions.html#lambdas