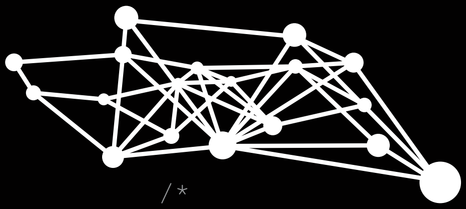


/*

interactive kinetic structures

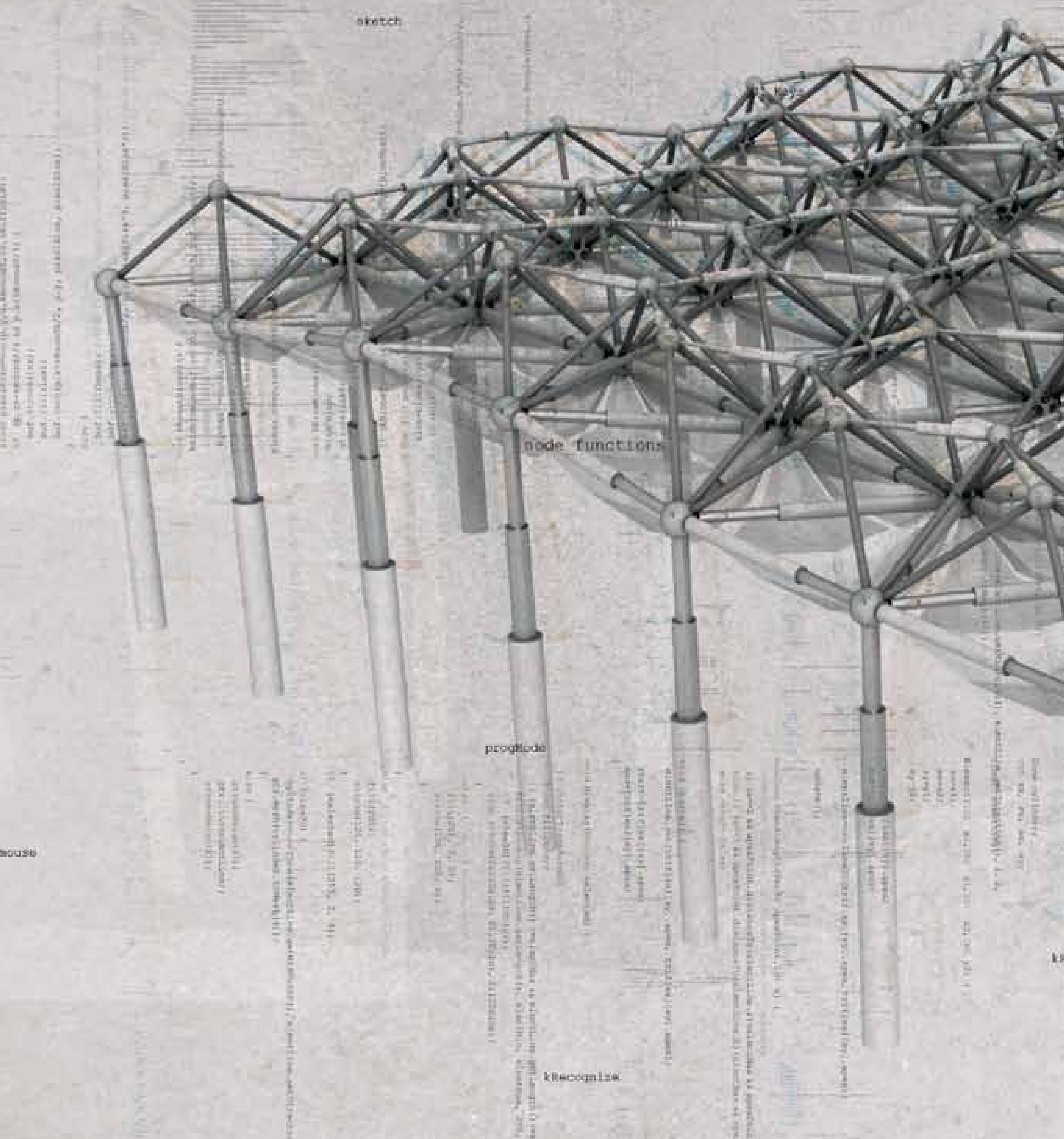
*/

/*
thesis project_
october 2012_

Architecture School_
National Technical University of Athens_

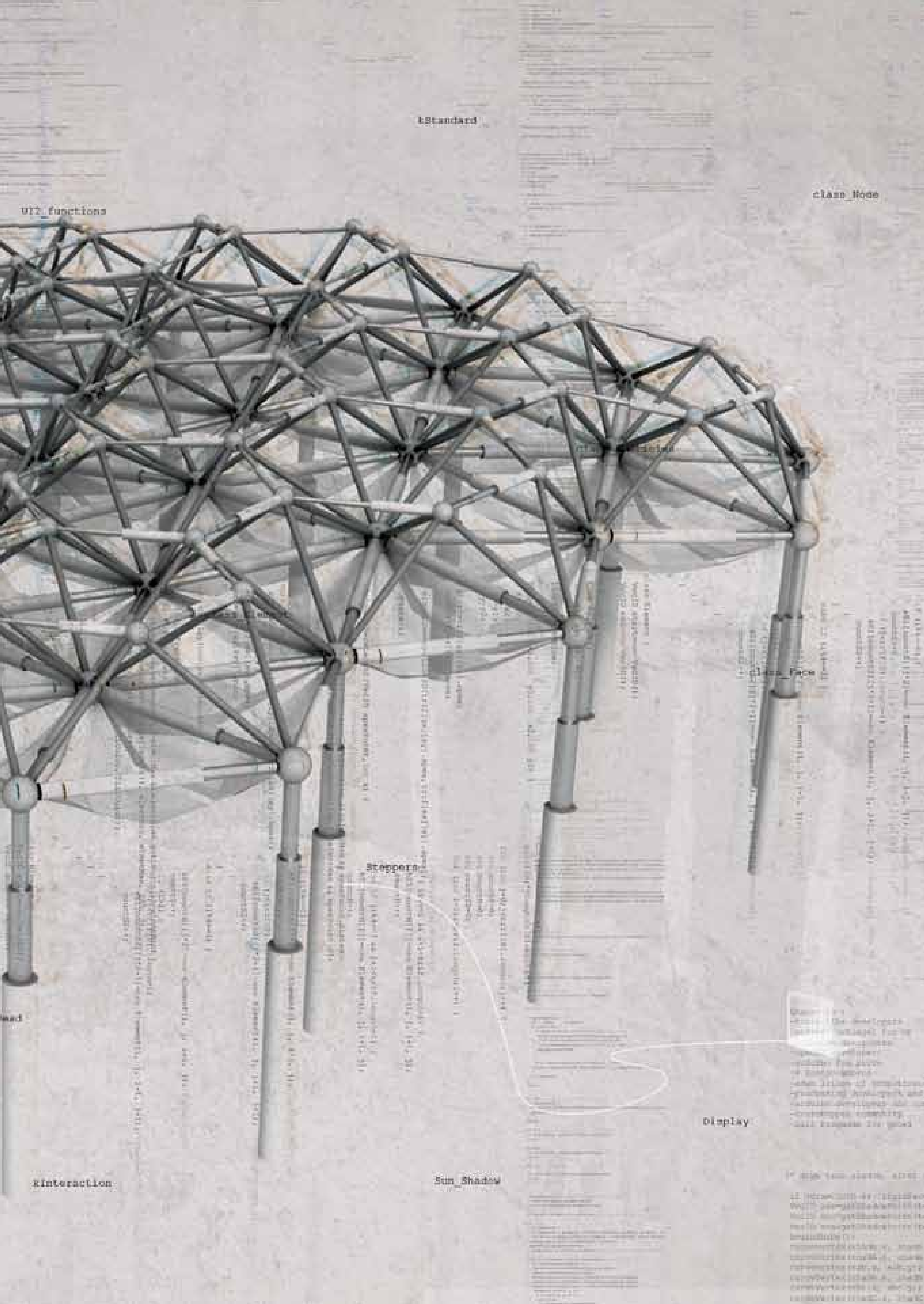
Miltiadis Constantinos_
supervisor: Papalexopoulos Dimitris_
*/

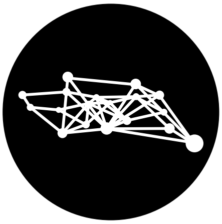
/*
κινητικες
διαδραστικες
κατασκευες
*/



This project was developed by Constantinos Miltiadis of the School of Architecture of the National Technical University of Athens, under the supervision of professor Dimitris Papalexopoulos and was presented at 31/10/12.

```
//kinetic space grid. Concept and development: Constantinos Miltiadis
//thesis project for Architecture School/NTUA, supervisor Dimitris Papalexopoulos, Athens 2012.
```


kStandard
class_Node
UI2_functions
class_Face
Steppers
ead
Display
kInteraction
Sun_Shadow



*Its quite impossible to present the function of an object in this printed medium, for this reason, it is limited to explaining how its organized, how it works, and how it can be built, through diagrams, renderings and details.

A video presentation of the functions of the program that controls the structure can be found here:
https://www.youtube.com/watch?v=XZqgqt0Jc28

prologue

The starting point for this project, was a fundamental problematic:

could we transform the object of architecture from the design of the defined to the design of the unpredictable?

Fallacies and Spandels

As architects, we are educated to design for equilibria; for situations, where we are called to produce the space for a given activity. We encounter a request (for a dwelling for example) as a totality, which we analyse to different entities (kitchen, bedroom etc) and attach distinctive traits to them. Doing this, helps us conceive the problem and solve it partially. We thus, realize a conceptual representation that disintegrates a whole to parts, that receive distinctive, absolute and idealized choreographies, so as to design an object that consists of articulated singularities.

Eventually, we design embeded in a twofold of representations; the first being the plan, which a third party, through a semiotic system -that of the 2d plan- is called to reproduce, through its own perception. The second level is the representation of an idealized function (a plot), that the architect constructs to be able to design. As a consequence, the produced space is inherently limiting, as it was designed to undertake and serve this solid plot, which will probably never materialize as the architect has choreographed it.

In practice, a bidirectional compression is been caused between the space, and the actors and actions. Space, as concurrently a passive and active system, interjects actors and actions suggesting towards the intentions of the designer (or its objective qualities). This makes up the inherent fallacy of architectural practice, and also offers, as a by-product, a relative inflexibility of space regarding its utilization potential.

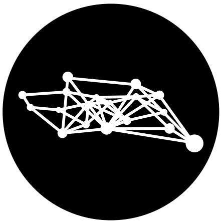
Even if, in the case of a dwelling for example, where one could argue that its functions are relatively defined, its not a good design approach that makes successful final product, but the ability of this planned space to be transformed, and thus utilized in ways that diverge from its initial purpose, towards the intentions of the user (architectural choreography versus practical signification).

In addition, this design methodology of creating spatial entities based on constructed subsets of activities, makes up an architectural convention; a house is not by definition a series of spaces that undertake different functions; this came of the history of architecture and the methods of design and fabrication. On the contrary, it could be said that a house is the space that follows and adjusts the need of its tenant in the dimension of time. Through this "definition" we could support that architecture today crystallizes and realizes the extreme phases of living, while the intermediate are declassified as irrelevant and inferior for the design, and purge. The produced object finally constitutes generic descriptions, and is actually a spandrel[1] (epiphenomenon) of the design methodology rather than an effective solution to a spatial problem.

In the case of public space, one cannot even approach design in the same manner, as its functions are neither finite nor belong to a defined and solid set that can be fragmented and solved partially. Thus, in the necessary presupposition of the existence of a plot and program, the architect undertakes the role of the storyteller; creating a fictional plot for the given space, that will render design feasible. In reality however, public space, is not something that can be predefined, because the functions that it will be called to undertake are in fact unpredictable, unmeasurable, unimaginable, open, and uncharted.

Trying to overcome these problems, we sought to create a different system, where the produced space is accountable to its users, reading every current situation so that it creates at any time what is considered required.

[1] Biologists Gould και Lewontin, have claimed that certain traits cannot be explained through the evolutionary theory, but on the contrary, are clearly epiphenomena of other choices taken in the course of evolution, lacking any reasoning or functionality on their own, and can only be given one through the examination of other traits; an architectural example they used was the one of spandrels that exist only as necessary epiphenomena of supporting a dome on arches. Spandrels thus, although acquiring certain functions and signification (in this case they are the place for the frescoes of the Evangelists) do not constitute architectural options. Stephen Jay Gould, Richard C. Lewontin, The Spandrels of San Marco and the Panglossian Paradigm: A Critique of the Adaptationist Programme, Proceeding of the Royal Society of London, 1979.



introduction

Parametric Procedural Design

Beginning from the parametric paradigm -where the architect instead of geometries, designs the relationships between objects- our intention was to maintain such capabilities, not only for the short period of the design process, but also for the lifetime of the structure; producing in this way a flexible architectural object, that has the capability to transform so as to undertake a range of functions.

The presupposition to reach this target, is the introduction of the dimension of time to the "architectural plan", so that we can design procedures that wil enable the transformation of the structure, depending on the circumstances. Thus, instead of the conventional design of dealing with objects, 3 dimensional relationships, the problem we intend to solve requires a system in which we would organize 2 dimensions: **time and information**; or the change of information over time -information as a function of time. The geometry of the structure, participates parametrically in the dimension of information, as does the environment which actuates the transformation.

What eventually is design in this case, besides the creation of the framework, is the development of parametric procedures that control the alteration of the structure.

The question of software

From the outset, one of the largest problems was that of software. How could we design the function of an object, since we have been taught to design its geometrical representation, and that is what our computer programs do?

The available media would allow as to design an object that could move, but not its movement, since they lack the dimension of time, thus halting at the assumption of both movement and function. On the contrary, we intended to design that function as well as its movement.

Considering the available software, our concept reaches a dead end...

We are nevertheless familiar with the suggestions of -between others- Manuel DeLanda[1], who recommended for architects to start developing their own software (or hack the existing) and Kostas Terzidis[2] who supported with his work the use of programming in architecture.

As it actually appears, (object oriented) programming not only allows for the design of such a structure, but also opens up a new world for architecture, of great capabilities that otherwise we would only think of theoretically and abstractedly. Programming operating as a blank piece of paper obligates us to create our own functional entities (objects) that could have a high degree of complexity, beyond the primitive point or line entities, as well as the relationships between them, that no longer are limited to geometrical proximity, but are modular and functional bonds. Furthermore, utilizing such design methodology today, we also take part in

[1] Manuel De Landa, Deleuze and the Use of the Genetic Algorithm in Architecture, http://www.cddc.vt.edu/host/delanda/pages/algorithm.htm

[2] Kostas Terzidis, Algorithmic Architecture, Architectural Press 2006.

a greater creative unity (developers) and the technological evolution, that offer us through the use of microntrollers the ability to realize movement of a structure, and also design its interaction, importing information about the environment, such as people's movement (Kinect sensor), in our plan and process them in real time.

The programming language we chose was Processing (java based) that was developed at the MIT Media Lab in 2001. The fundamental reasons for using processing are two. The first is that it was designed to be used by people and in fields that do not traditionally have any programming education or experience, and so it is quite easy to learn and focuses directly on practical applications. The second reason is that it is open-source, and therefore its users, operating through the open source mentality are constantly trying to expand it towards divergent directions and fields as well as help new users get familiar with it.

Therefore, Processing is a perfect choice for architects because it has both developed a huge user base in a short period of time, and also a great volume of expantion libraries, for a plethora of applications, accessible to everybody to utilize; facts that other professional programming languages would envy.

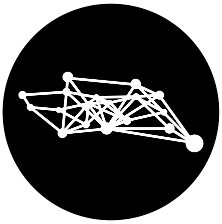
*Scripting

Beyond DeLand and Terzidis, another important personality for architecture that supports programming is Mark Burry[3], who has furthermore examined the capabilities of "scripting" -the method of specialized code-writting inside an environment of a conventional CAD software (rhinoscript, maya MEL, embedded python script etc). Despite that this method provides a range of important conveniences for the user -such as a fair amount of tools and geometrical entities that the main program offers- it also comes with significant disadvantages that prevent us from using this method for the current project; such as sufficiently limited processing power, limited expansion potential, limited capabilities for communication with microcontrollers and other hardware, and limited educational resources.

Design in the 21st century

As shown by the present project, architects today, can use computers, not as mere representational tools, but as mediators between their concepts and and the actual physical object they design, gaining in this way more control over the final product on the one hand, and more importantly a different degree of control on the other. The technological advancements that offer these capabilities, suggest a paradigm shift, where architecture is no more a static object, but in fact a complex set of relations; an interface between the environment, its subjects and a structure, rendering the latter able to **communicate, adjust and adapt**, in any situation and circumstances. In this system, the role of architecture is different, not only historically, but also as a social and urban encounter, where its potential as an actor in this system is unprecedented.

[3] Daniel Davis, Mark Burry and Jane Burry: The flexibility of logic programming, Proceedings of the 16th International Conference on Computer Aided Architectural Design Research in Asia / The University of Newcastle, Australia 27-29 April 2011, σ. 29-38



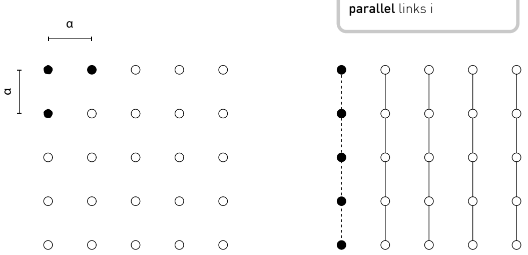
study of grid & movement

spaceframe **grid evolution**

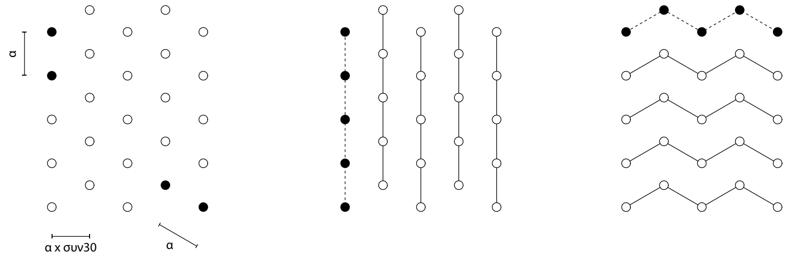
parallel links i

parallel links ii

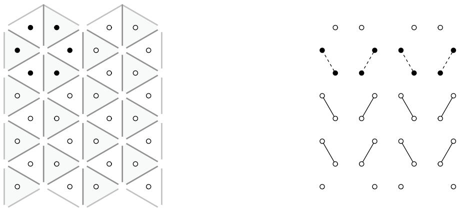
rectangular grid



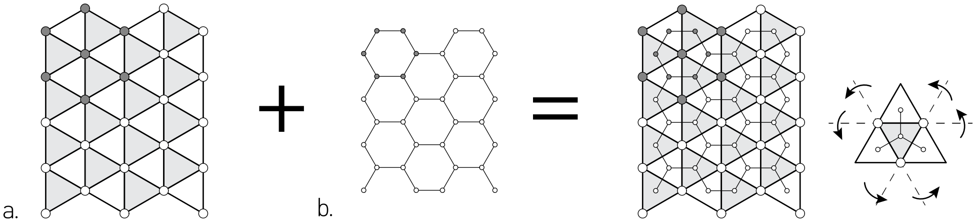
triangulated grid



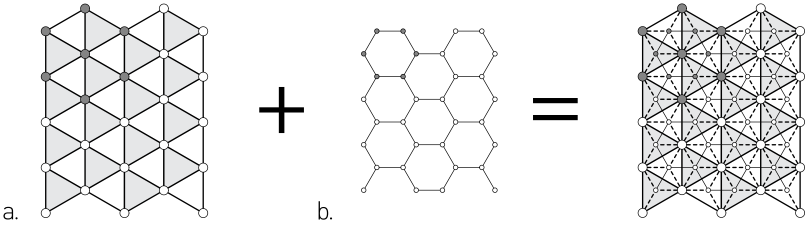
lever grid

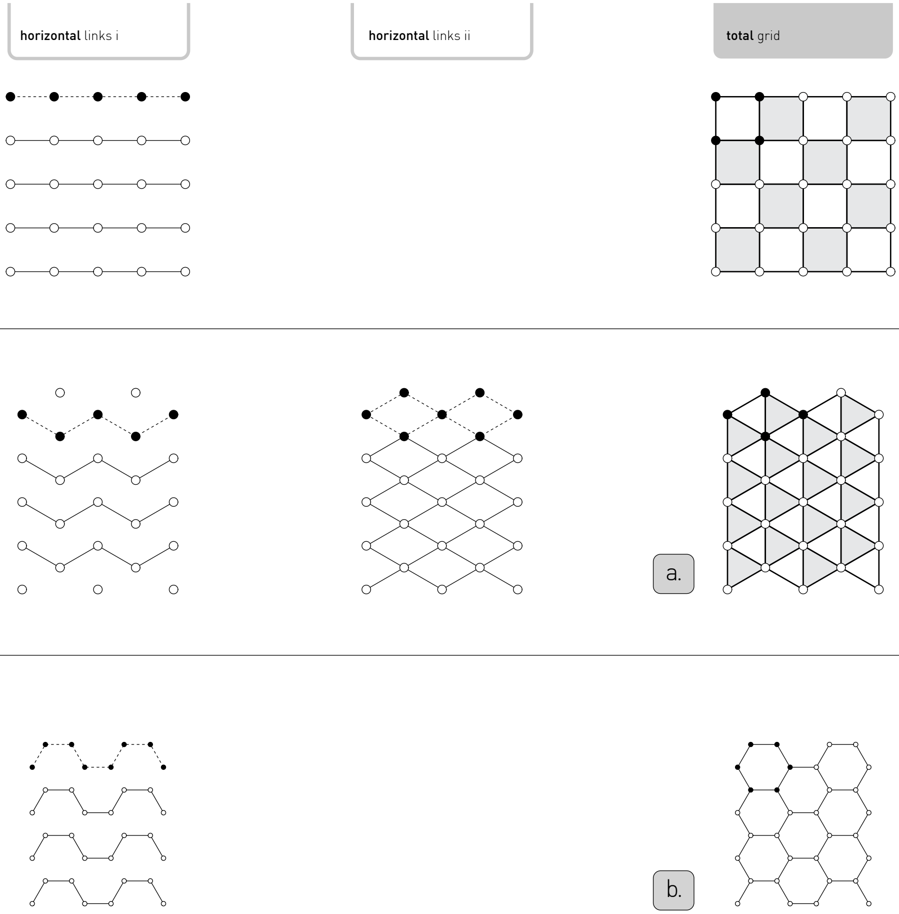


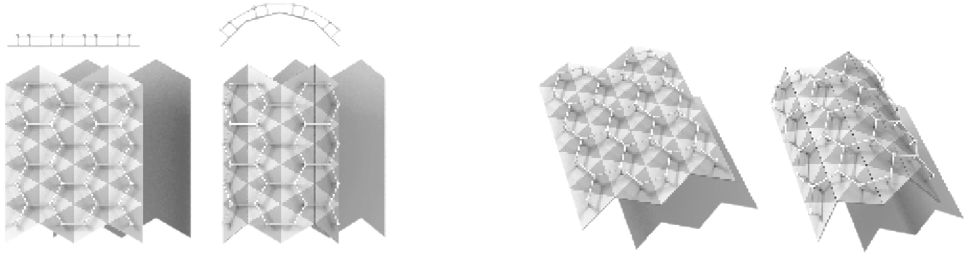
lever **grid link** i

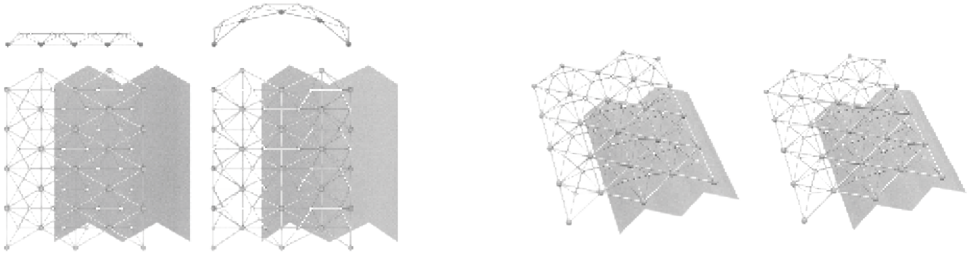


lever **grid link** ii
pyramidal spaceframe



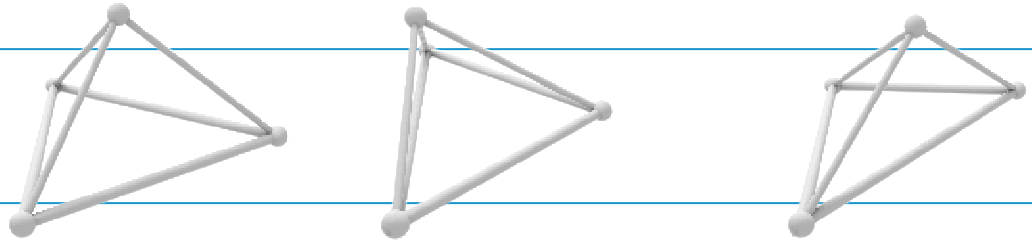

horizontal links i
horizontal links ii
total grid
a.
b.





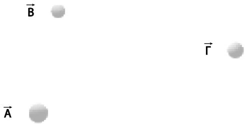
finding the **tetrahedron apex**

transformation study

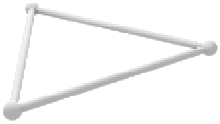


geometrical solution

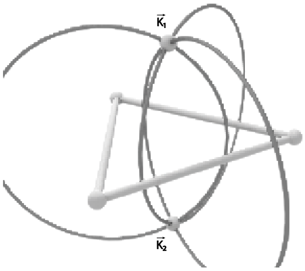
γ01 given:
triangle vertices: ΑΒΓ
beam length: R



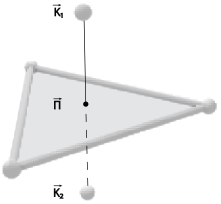
γ02 drawing the triangle



γ05 circle intersection

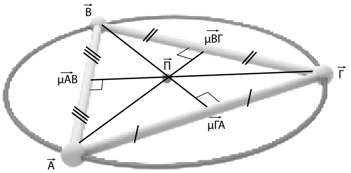


γ06 finding the two solutions

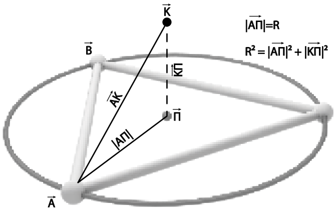


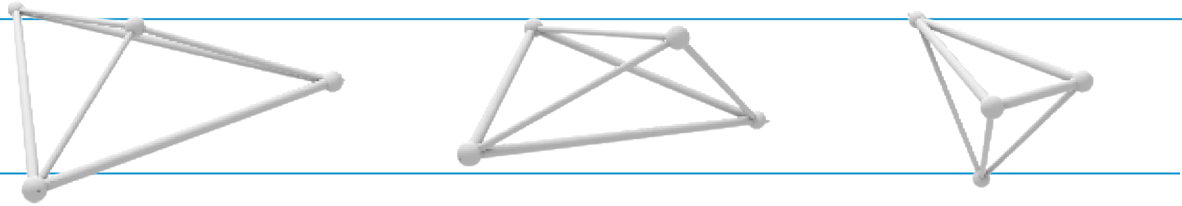
mathematical solution

μ01 finding the projection of the apex to the plane of the triangle

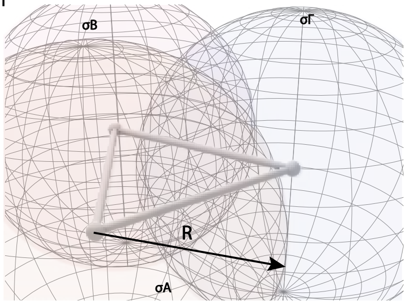


μ02 calculating transposition

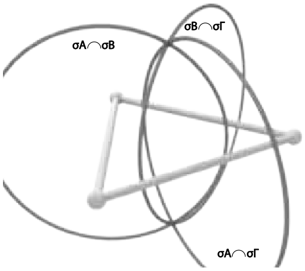




γ03 drawing the spheres

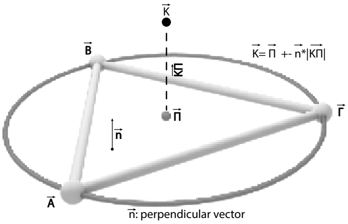


γ04 sphere intersection



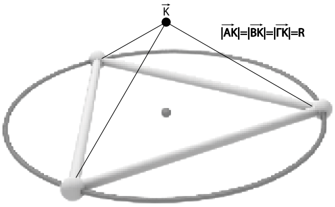
γ07 drawing the tetrahedra

μ03 finding the solutions



$\vec{n}$: perpendicular vector

μ04 drawing the tetrahedron


$\vec{K}$
$|\overrightarrow{AK}|=|\overrightarrow{BK}|=|\overrightarrow{\Gamma K}|=R$

algorithmic solution

$A(x_1,y_1,z_1)$, $B(x_2,y_2,z_2)$, $\Gamma(x_3,y_3,z_3)$, $K(x,y,z)$

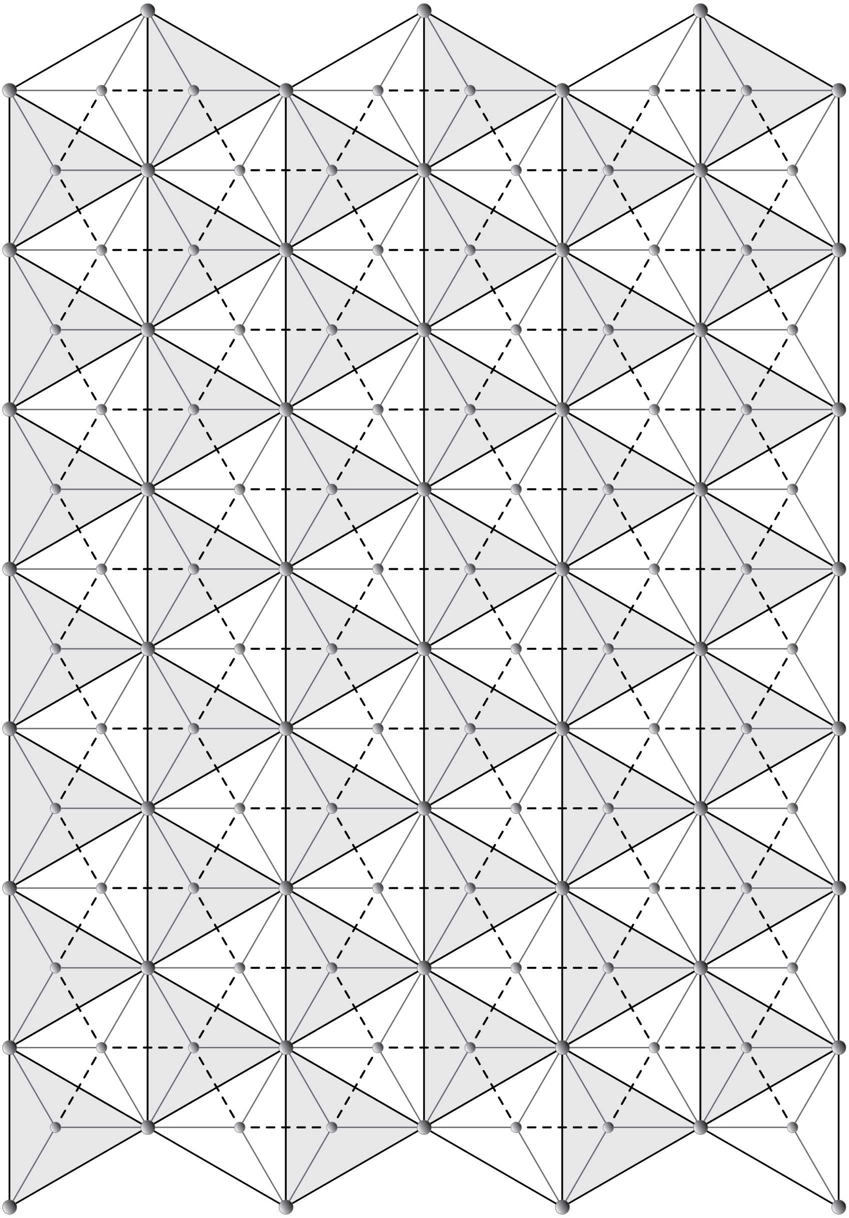
$$\alpha=|B-\Gamma|^2(A-B)\cdot(A-\Gamma)\ /\ 2|(A-B)x(B-\Gamma)|^2$$

$$\beta=|A-\Gamma|^2(B-A)\cdot(B-\Gamma)\ /\ 2|(A-B)x(B-\Gamma)|^2$$

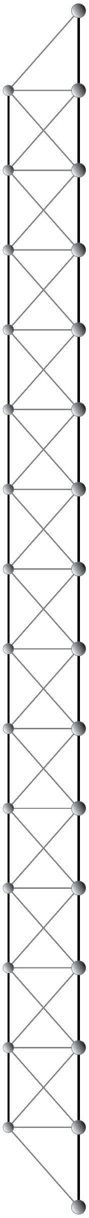
$$\gamma=|A-B|^2(\Gamma-A)\cdot(\Gamma-B)\ /\ 2|(A-B)x(B-\Gamma)|^2$$

$$K=\alpha A+\beta B+\gamma\Gamma$$

```
Vec3D getCircumCenter() {
   double aMult;
   double bMult;
   double cMult;

   aMult=(fTriangle.a.sub(fTriangle.b)).dot(fTriangle.a.sub(fTriangle.c));
   aMult*=sq((fTriangle.b.sub(fTriangle.c)).magnitude());
   aMult/=sq(((fTriangle.a.sub(fTriangle.b)).cross(fTriangle.b.sub(fTriangle.c))).magnitude());
   aMult/=2;

   bMult=(fTriangle.b.sub(fTriangle.a)).dot(fTriangle.b.sub(fTriangle.c));
   bMult*=sq((fTriangle.a.sub(fTriangle.c)).magnitude());
   bMult/=sq(((fTriangle.a.sub(fTriangle.b)).cross(fTriangle.b.sub(fTriangle.c))).magnitude());
   bMult/=2;

   cMult=(fTriangle.c.sub(fTriangle.a)).dot(fTriangle.c.sub(fTriangle.b));
   cMult*=sq((fTriangle.a.sub(fTriangle.b)).magnitude());
   cMult/=sq(((fTriangle.a.sub(fTriangle.b)).cross(fTriangle.b.sub(fTriangle.c))).magnitude());
   cMult/=2;

   Vec3D ma=new Vec3D(fTriangle.a);
   Vec3D mb=new Vec3D(fTriangle.b);
   Vec3D mc=new Vec3D(fTriangle.c);
   ma.scaleSelf((float)aMult);
   mb.scaleSelf((float)bMult);
   mc.scaleSelf((float)cMult);
   circumc=new Vec3D();
   circumc.addSelf(ma);
   circumc.addSelf(mb);
   circumc.addSelf(mc);

   return (circumc);
}
```

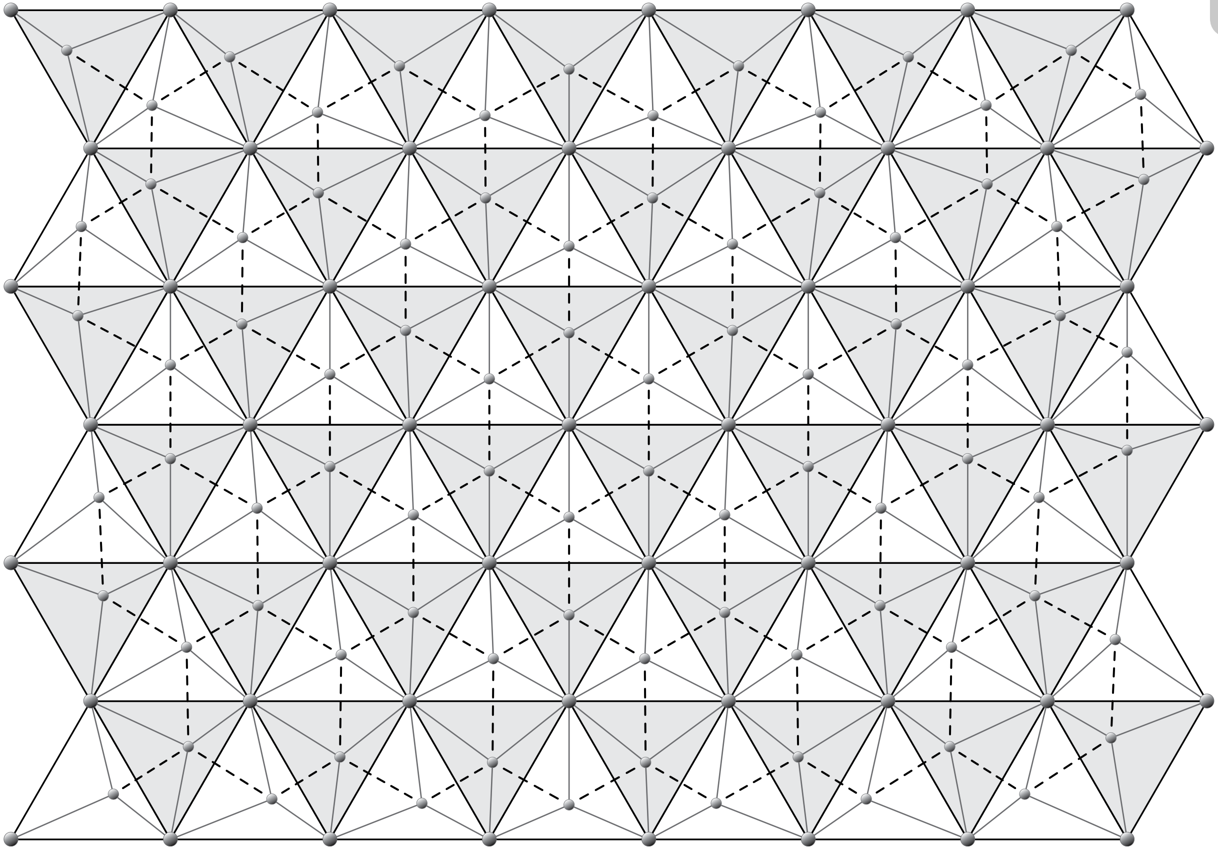


plan view

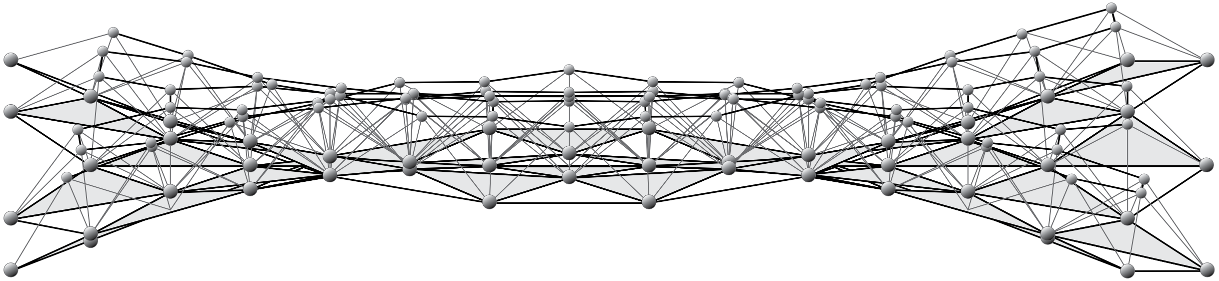


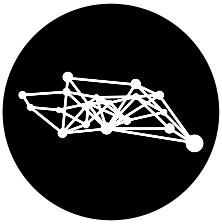
side view

plan view

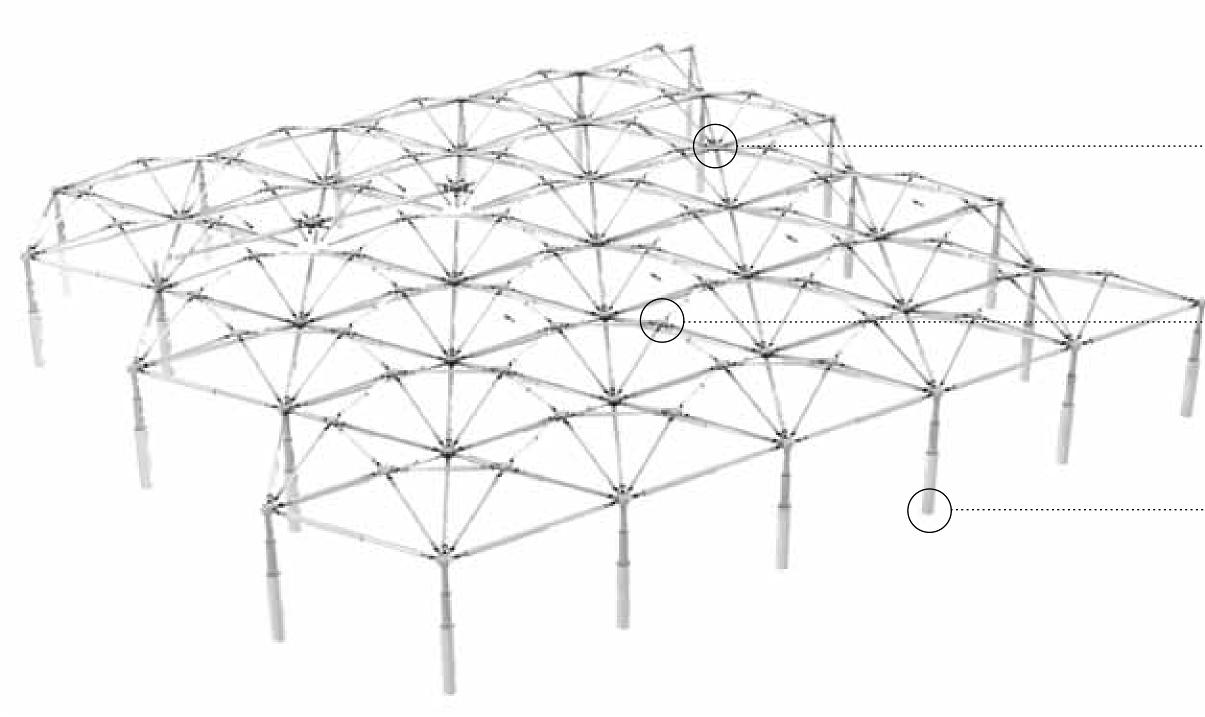


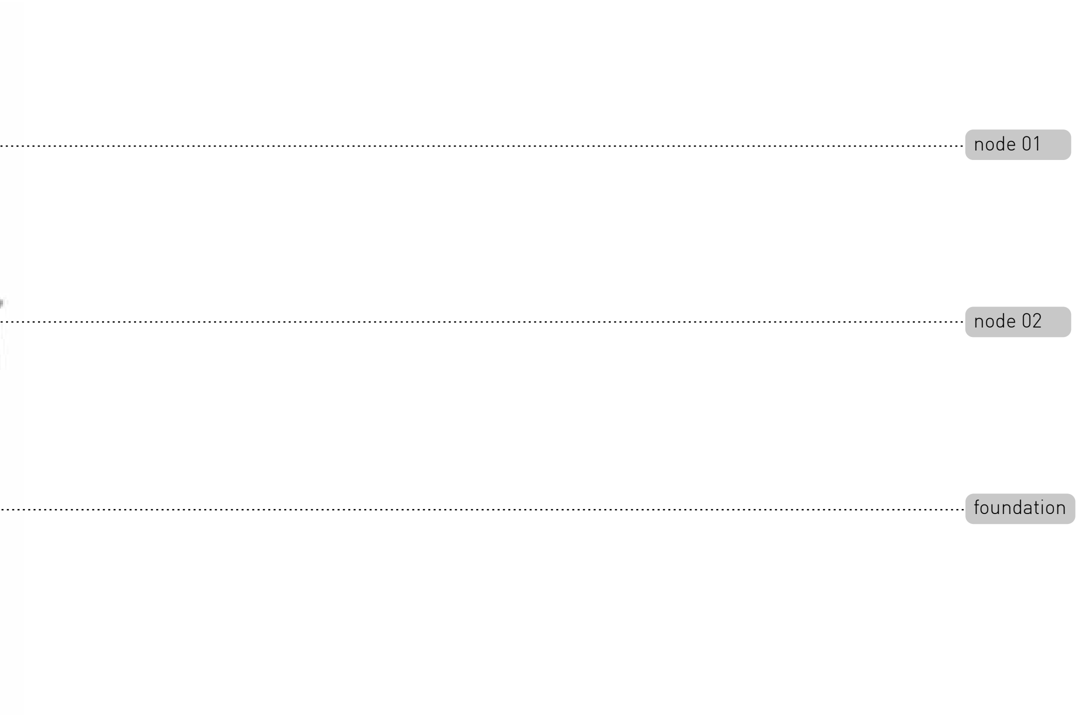
side view

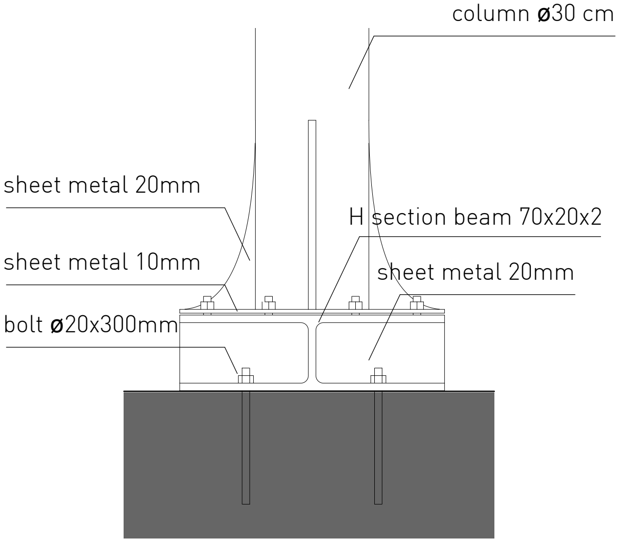


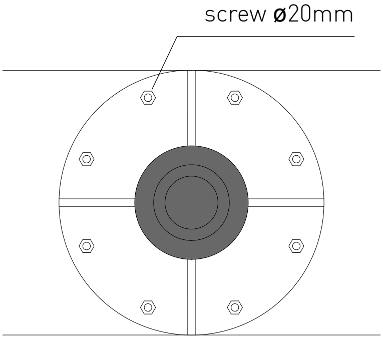


structural details



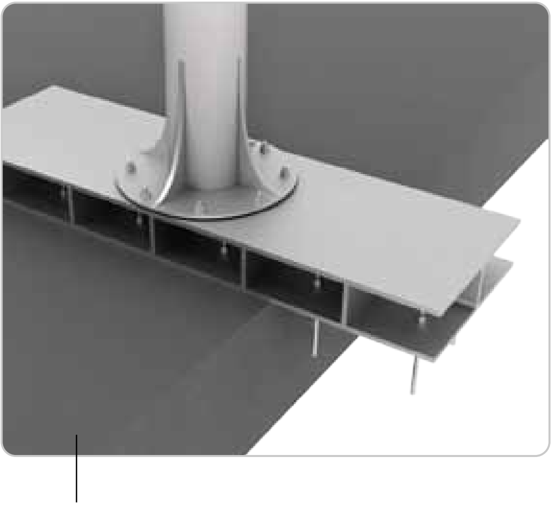

node 01
node 02
foundation

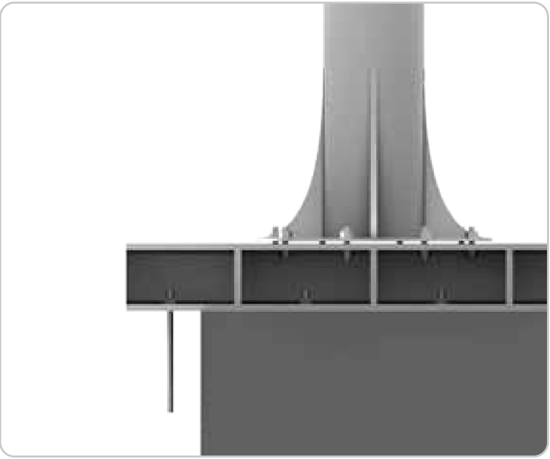


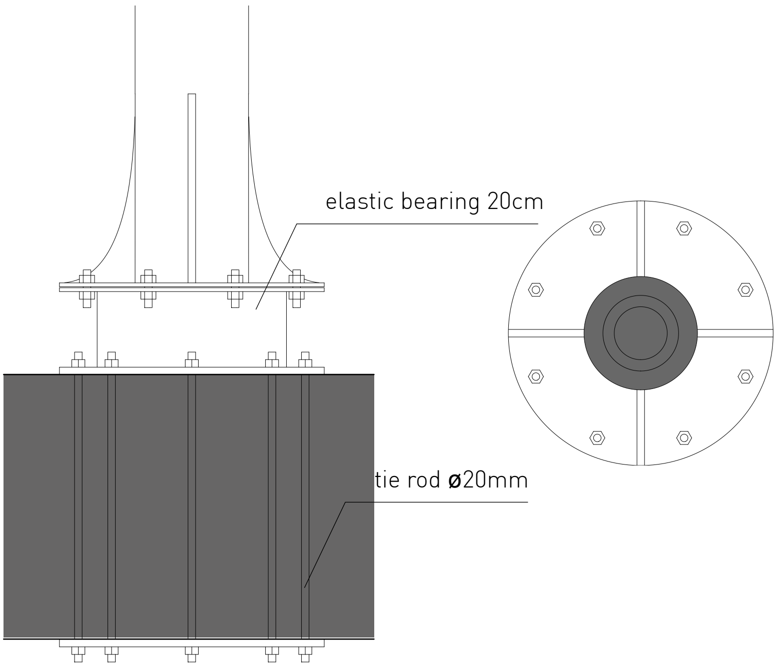


01 | **section**

| **plan view**

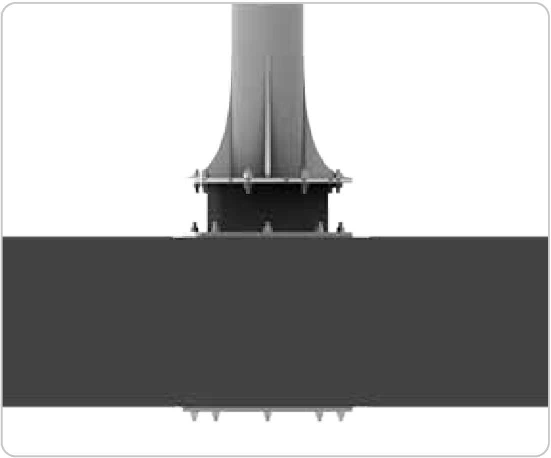






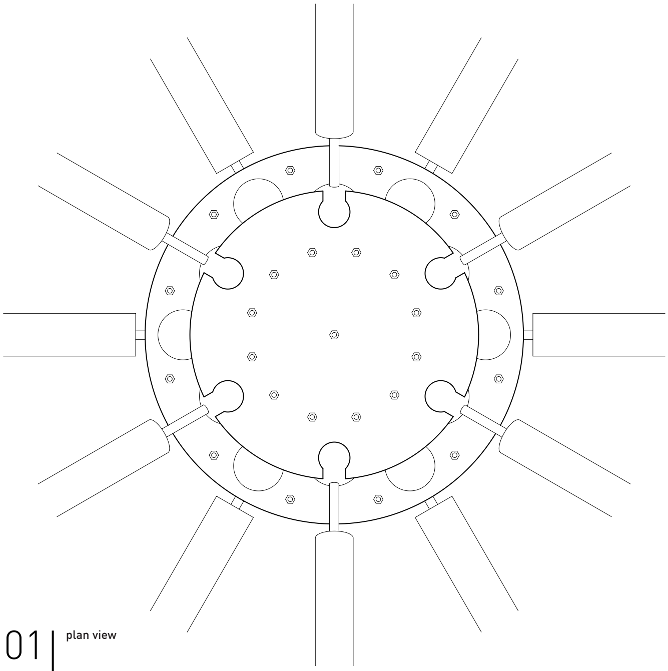
02 **side view**

plan view

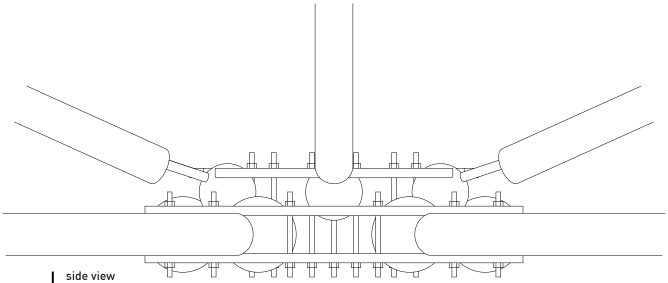


node with **3D spherical joints**
(3dof)

node study

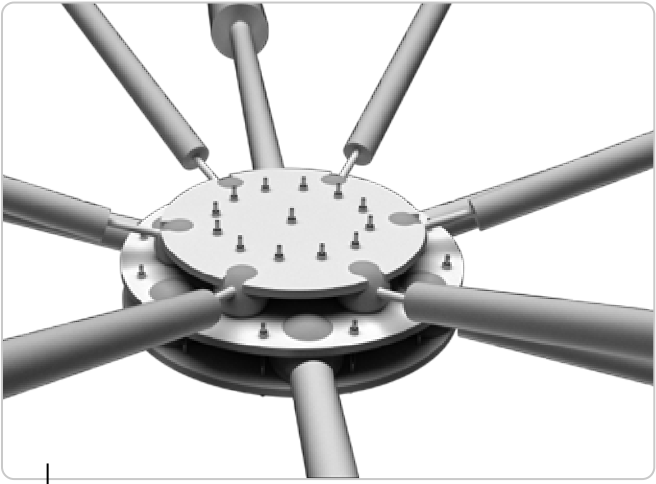


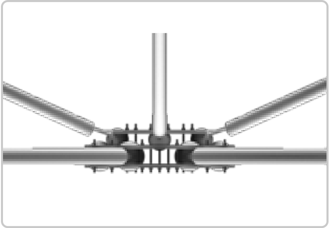
01 | plan view

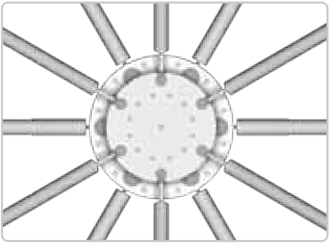


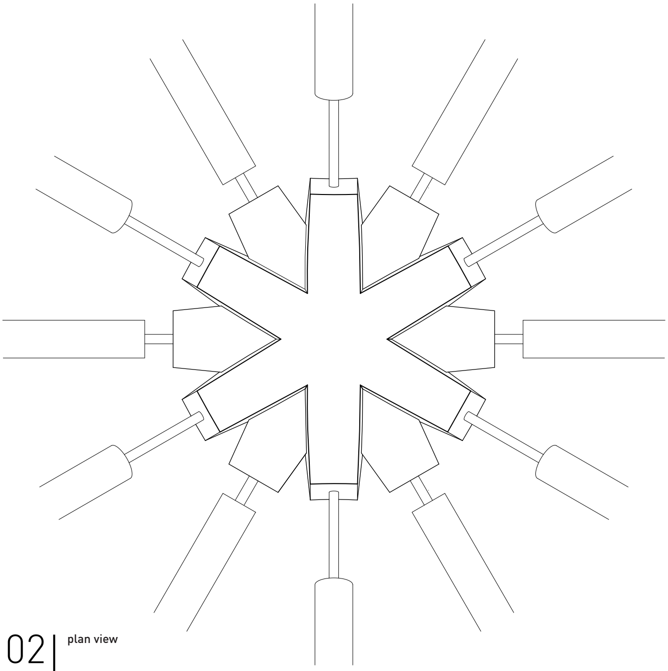
side view

renderings

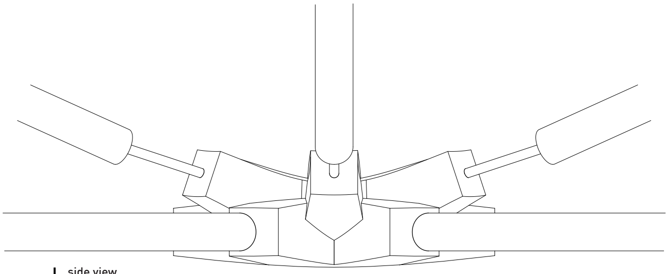




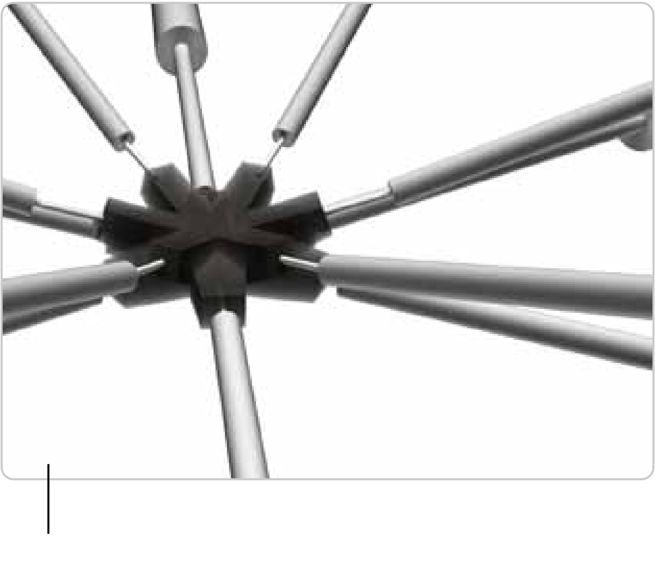


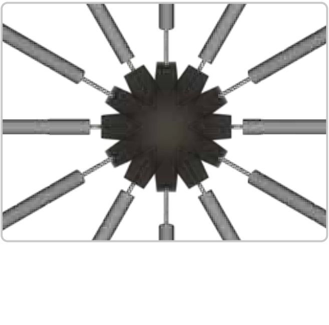


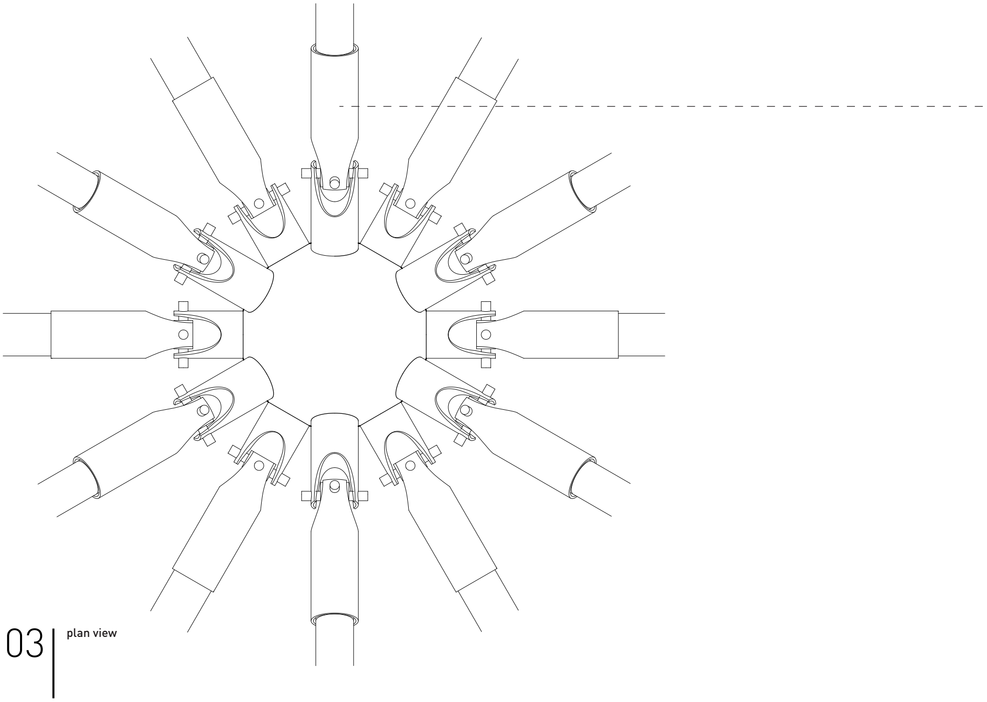
02 plan view



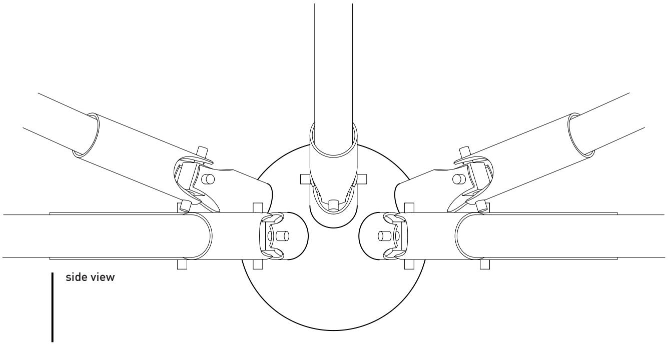
side view



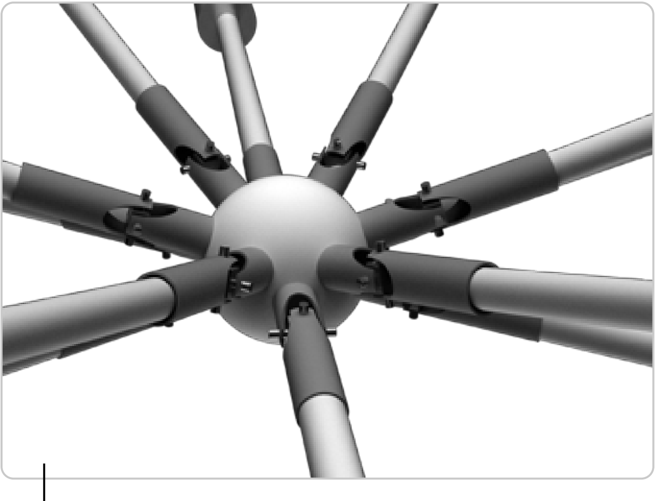


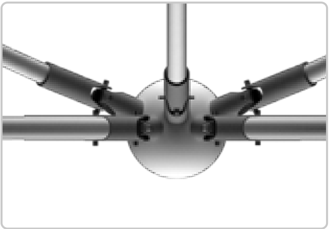


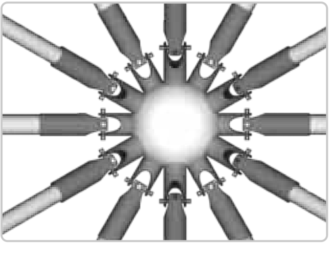
03 plan view



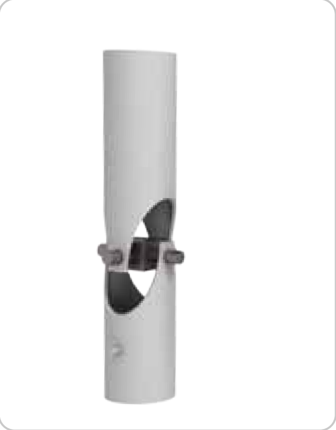
side view

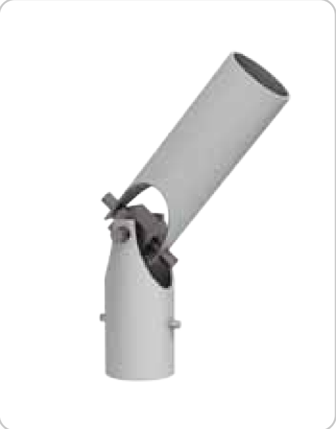


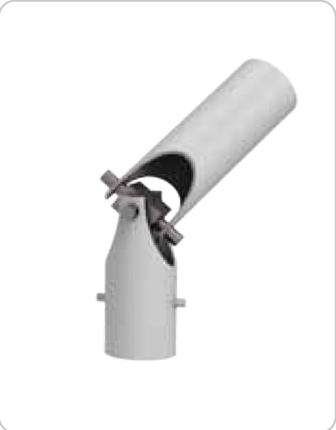


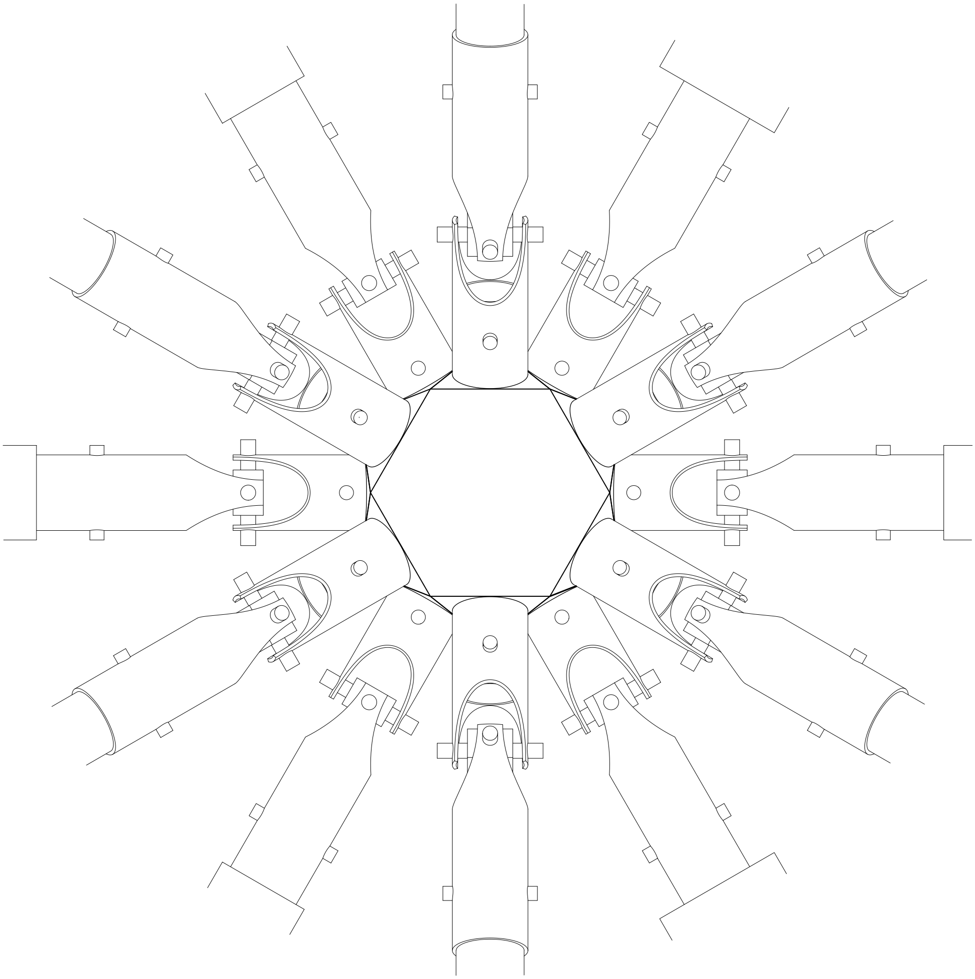


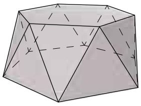
joint detail



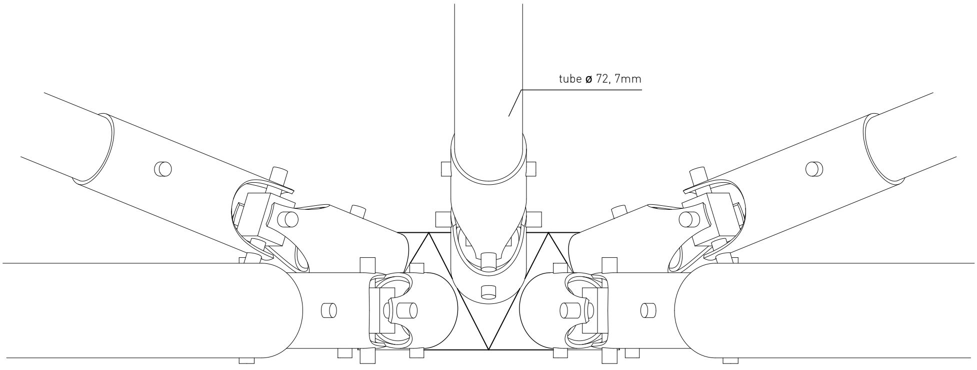




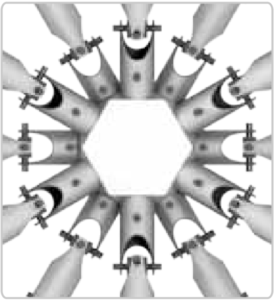


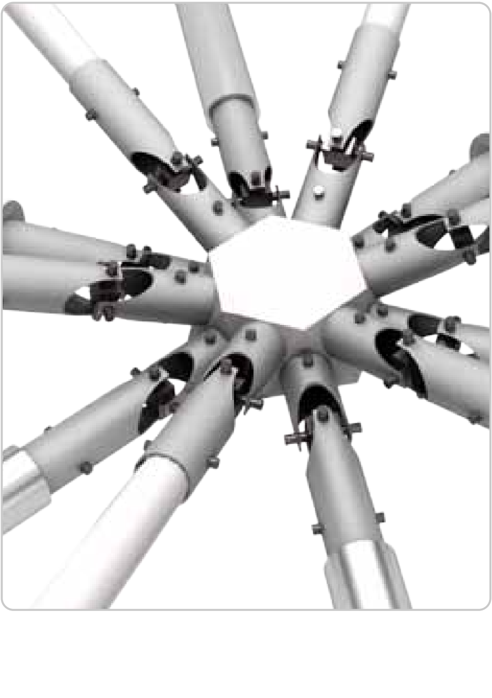


01 plan view

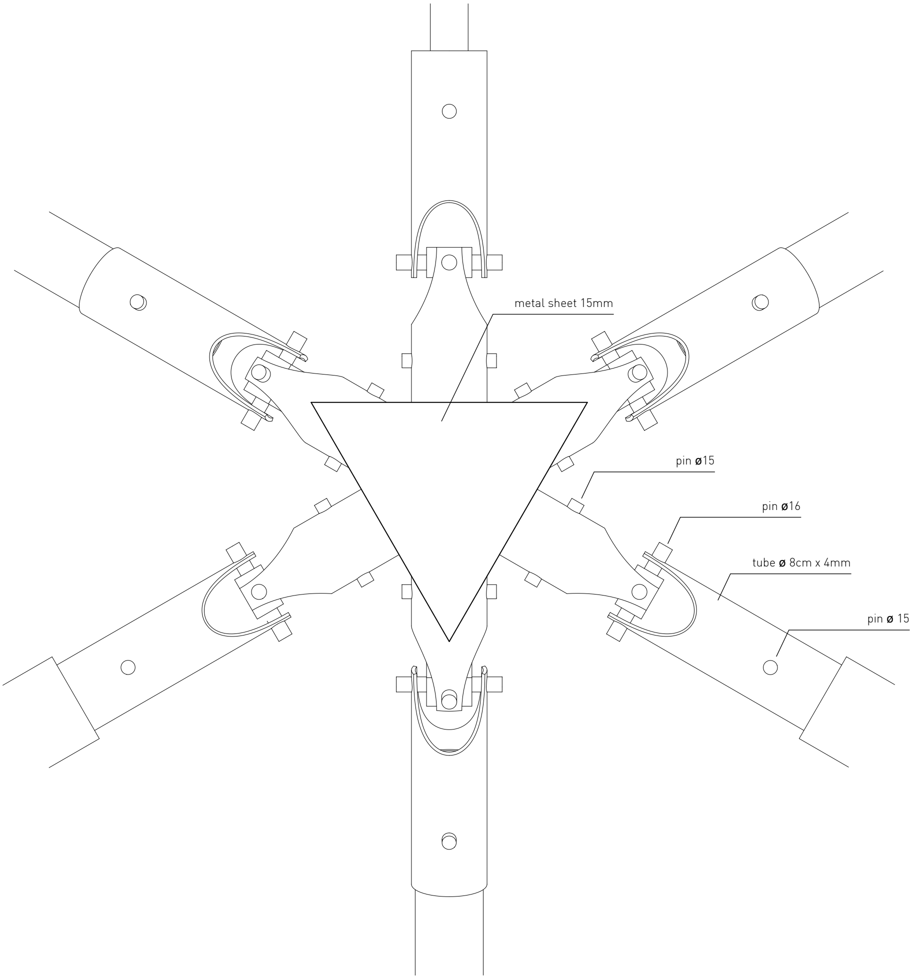


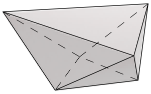
side view



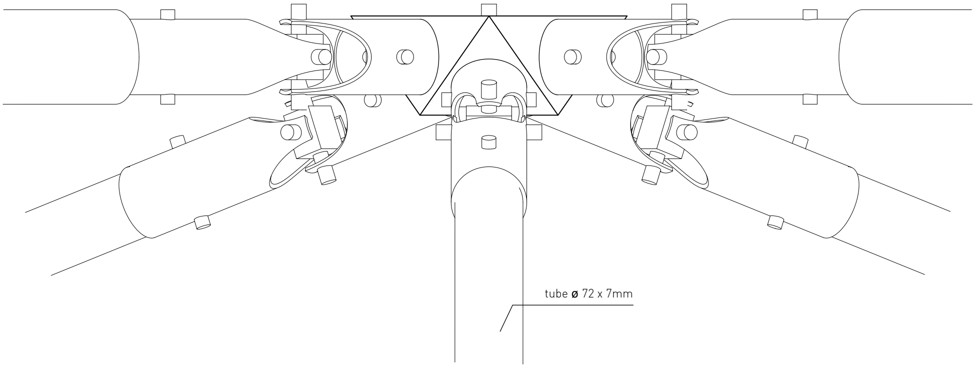


01

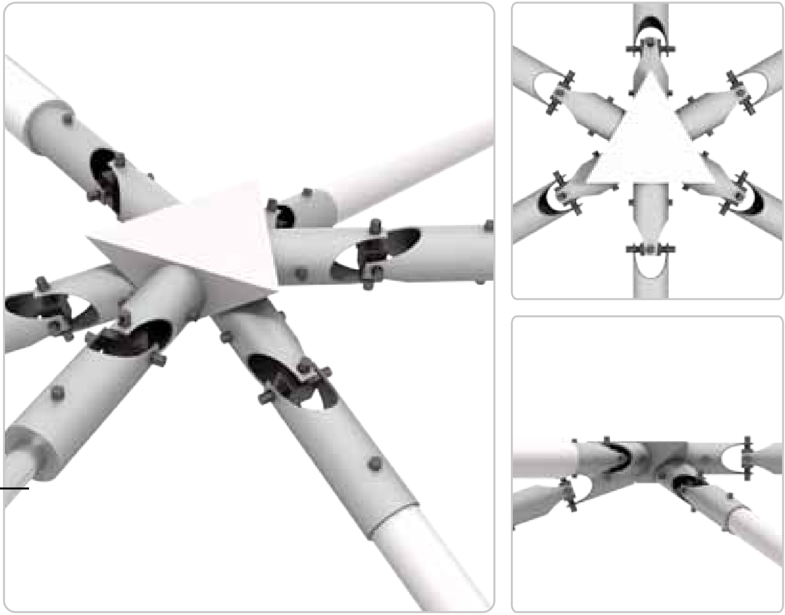




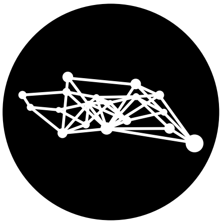
02 plan view



side view

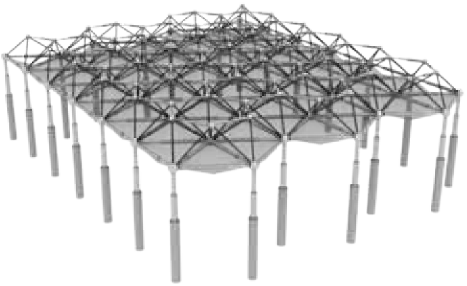


02



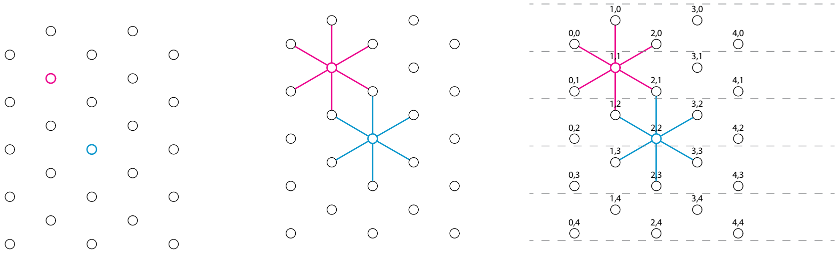
ontology

structure ontology



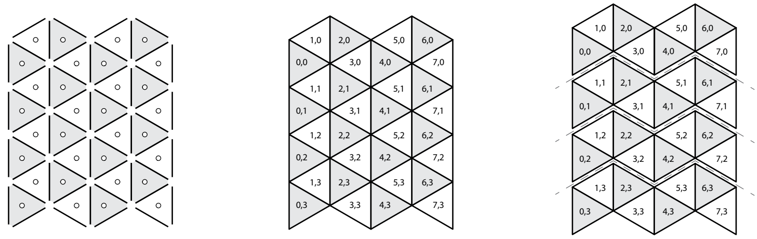
```
nX=5;
nY=5;
```

primary node relations



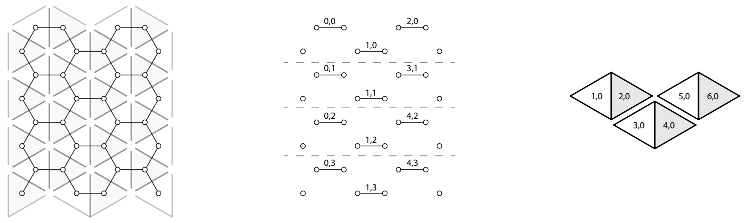
```
Node[ ][ ] nodes=new Node[nX][nY];
```

primary - secondary node relations

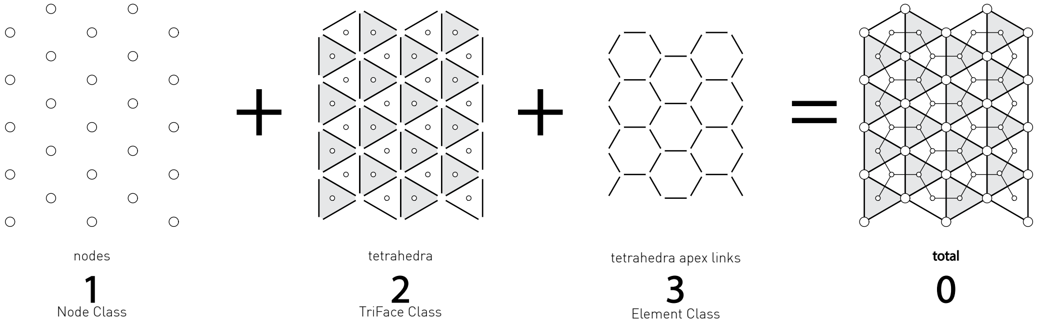


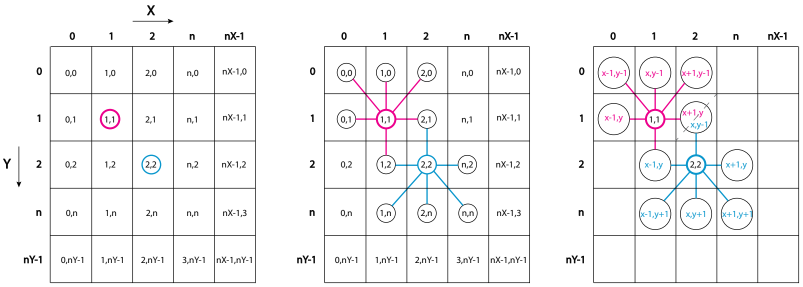
```
TriFace[ ][ ] trif=new TriFace[2*nX-2][nY-1];
```

secondary node relations



```
Element[ ][ ] hEl=new Element[nX-2][nY-1];
Element[ ][ ] sEl=new Element[nX-1][(nY-2)*2+1];
```

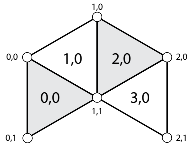


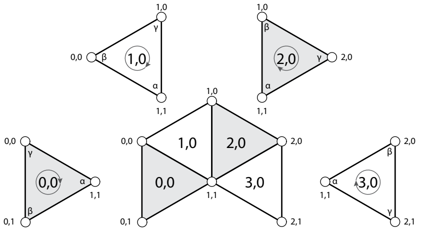


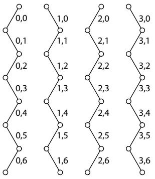
	0	1	2	n	nX-1
0	0,0	1,0	2,0	n,0	nX-1,0
1	0,1	1,1	2,1	n,1	nX-1,1
2	0,2	1,2	2,2	n,2	nX-1,2
n	0,n	1,n	2,n	n,n	nX-1,3
nY-1	0,nY-1	1,nY-1	2,nY-1	3,nY-1	nX-1,nY-1

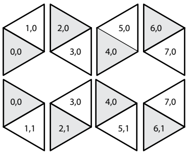
	0	1	2	n	nX-1
0	0,0	1,0	2,0	n,0	nX-1,0
1	0,1	1,1	2,1	n,1	nX-1,1
2	0,2	1,2	2,2	n,2	nX-1,2
n	0,n	1,n	2,n	n,n	nX-1,3
nY-1	0,nY-1	1,nY-1	2,nY-1	3,nY-1	nX-1,nY-1

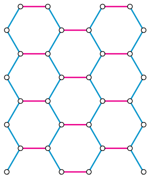
	0	1	2	n	nX-1
0	x-1,y-1	x,y-1	x+1,y-1		
1	x-1,y	1,1	x+1,y x,y-1		
2		x-1,y	2,2	x+1,y	
n		x-1,y+1	x,y+1	x+1,y+1	
nY-1					











Global variables

nX :
nY : }grid dimensions in nodes

minBeam :
maxBeam : }distance thresholds between nodes / beam length

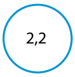
beamLen : initial beam length

colMin:
colMax: }column length thresholds

startZ: initial column length

pyramidBeam: tetrahedron sides length

Node Object



x:
y: }relative position

pos: position vector

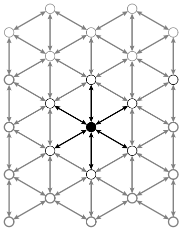
isColumn: true if connected to a column

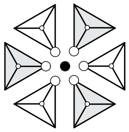
colPos: column base vector

neighbours: number of neighbors

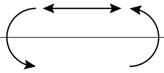
neighIndx[ ]: relative positions of neighboring nodes

faceIndx[ ]: relative positions of related tetrahedra

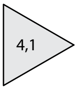




```
if (i%2==0)nodes[i][j]=new Node(50+i*startLen*cos(PI/6), 50+startLen*j+(startLen*sin(PI/6)), startZ, i, j);
else nodes[i][j]=new Node(50+i*startLen*cos(PI/6), 50+startLen*j, startZ, i, j);
```



Tetrahedron Object



x:
y: }relative position

x1,y1:
x2,y2: }related nodes
x3,y3:

fTriangle: 3D triangle between nodes

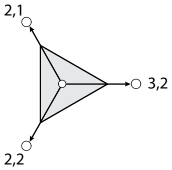
fCent: triangle center vector

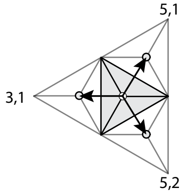
fNorm: normal to the triangle plain

fCircumc: cercumcenter vector

apex: apex vector

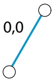
selIndx[ ]:
helIndx[ ]: }relative positions of neighboring tetrahedra





```
trif[countF][j-1]=new TriFace(i, j, i-1, j, i-1, j-1, countF, j-1);
countF++;
```

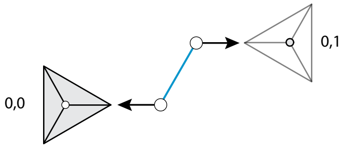
Secondary beam Object



sx,sy: start tetrahedron relative position

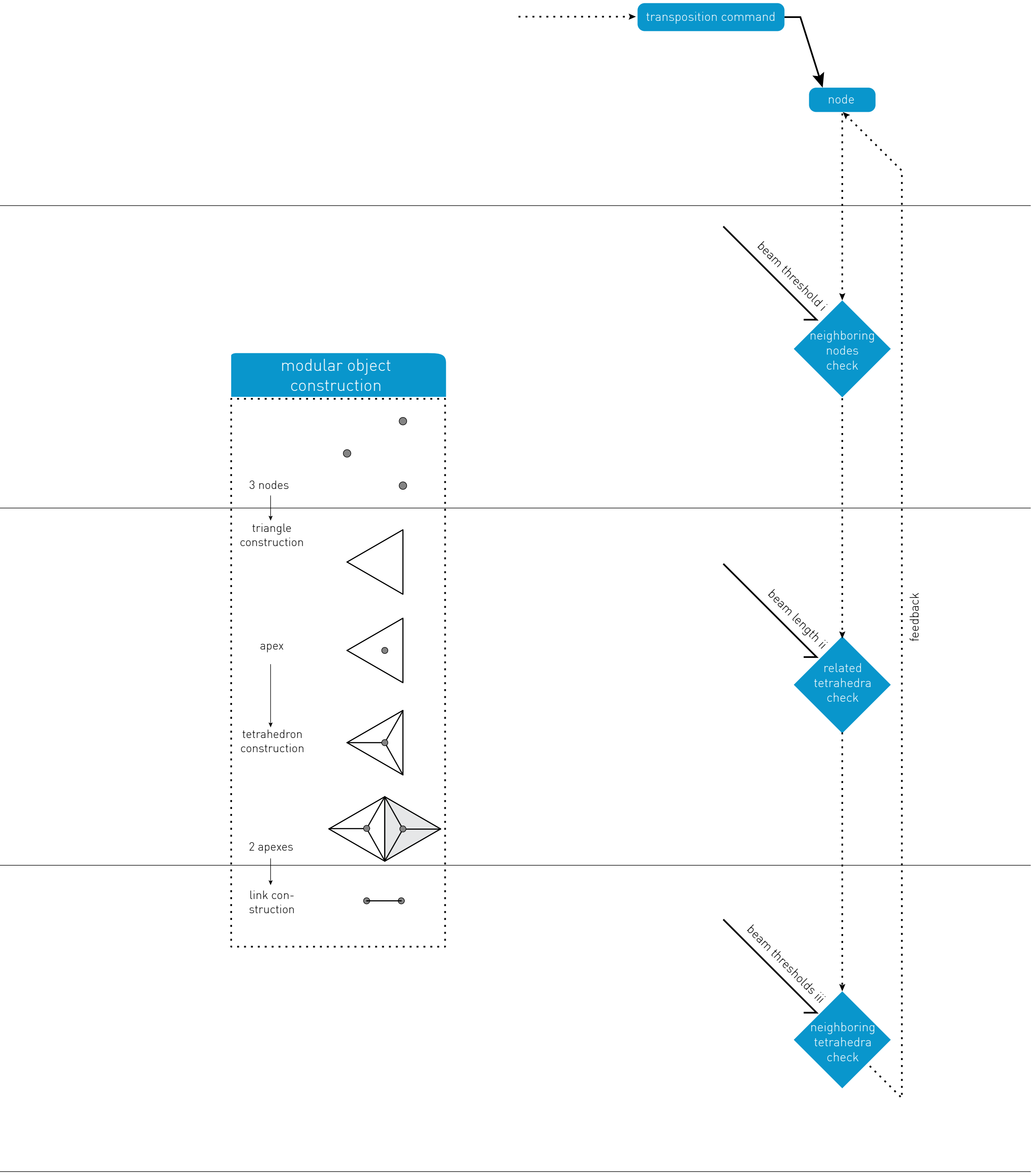
ex,ey: end tetrahedron relative position

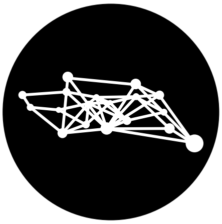
start:
end: }start and end vectors



```
sEl[countS][j*2]=new Element(i, j, i+1, j);
countS++;
```

modular function





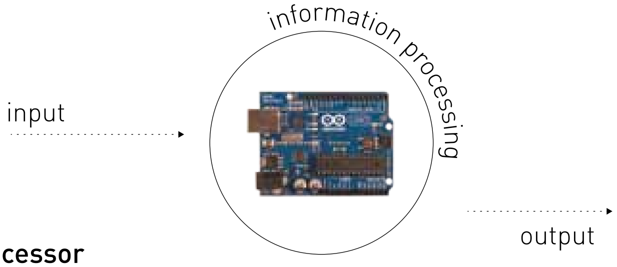
information management

Utilized libraries:

toxic Libraries - Karsten Schmidt / 2009
oscP5 - Andreas Schlegel / 2011
arduino Library / 2011
simpleOpenNI - Interaction Design Department Zurich / 2011
blobDetection Library - Julien Gachadoat / 2004
peasyCam-Jonathan Feinberg / 2011

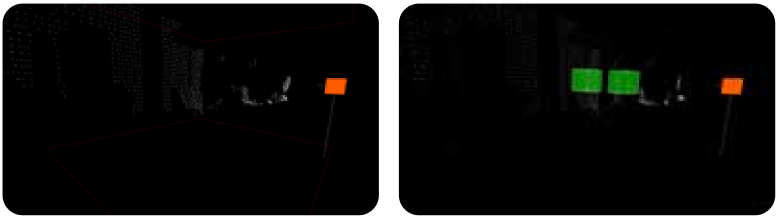
Processing
Ben Fry, Casey Reas / media lab, MIT / 2001

open hardware

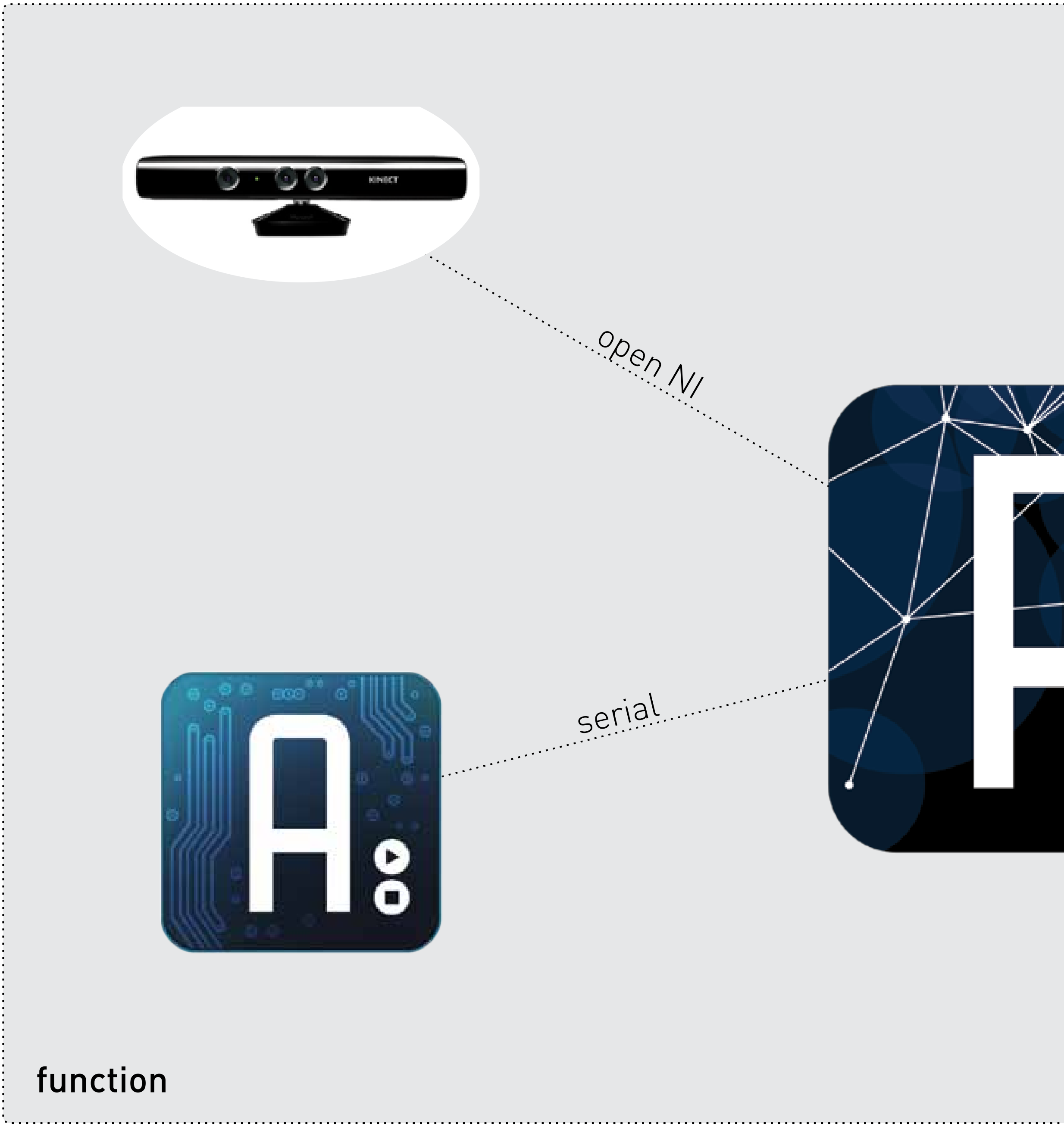


Arduino microprocessor
Massimo Banzi / IVREA institute / 2005

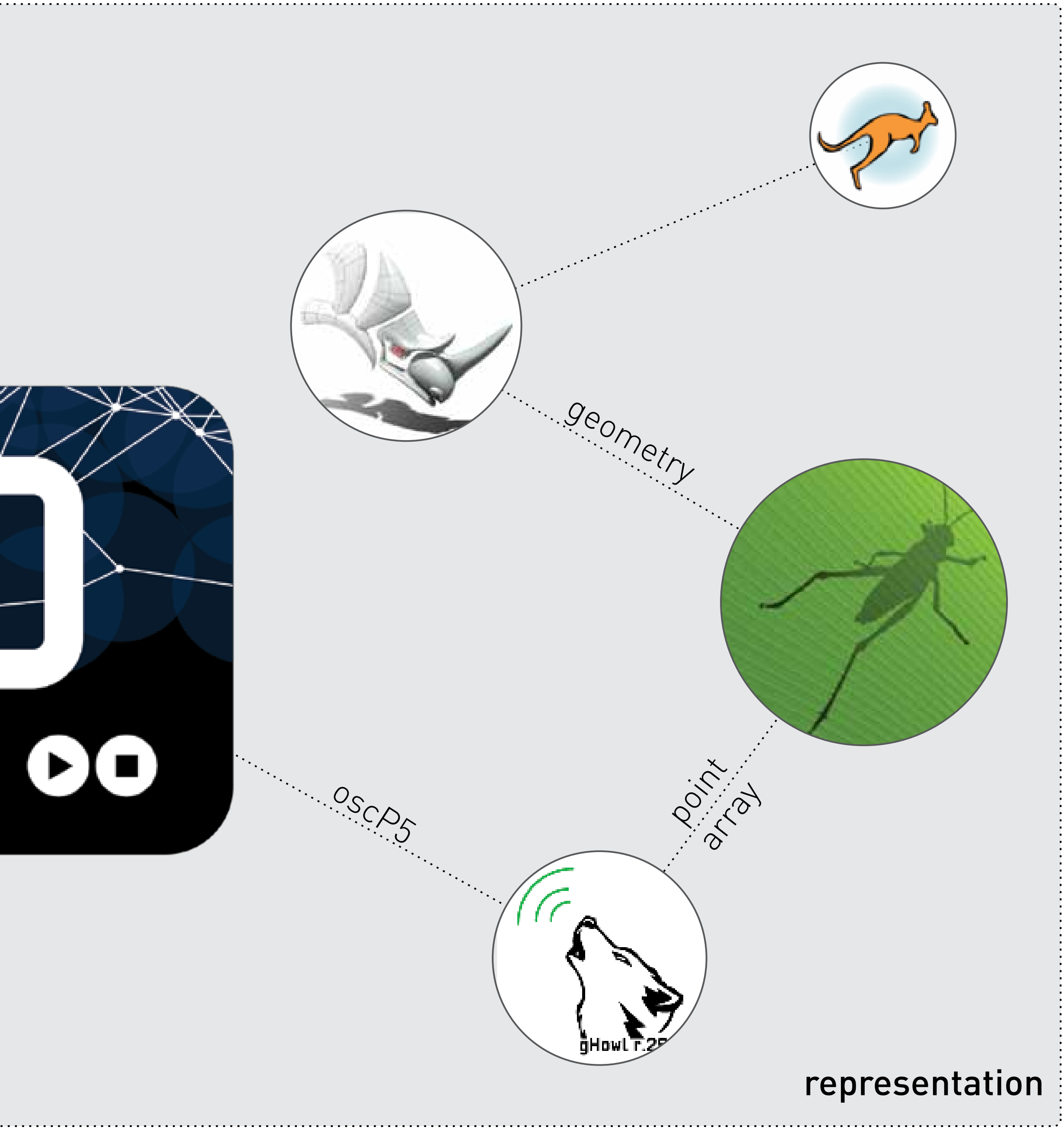
hardware **hacks**

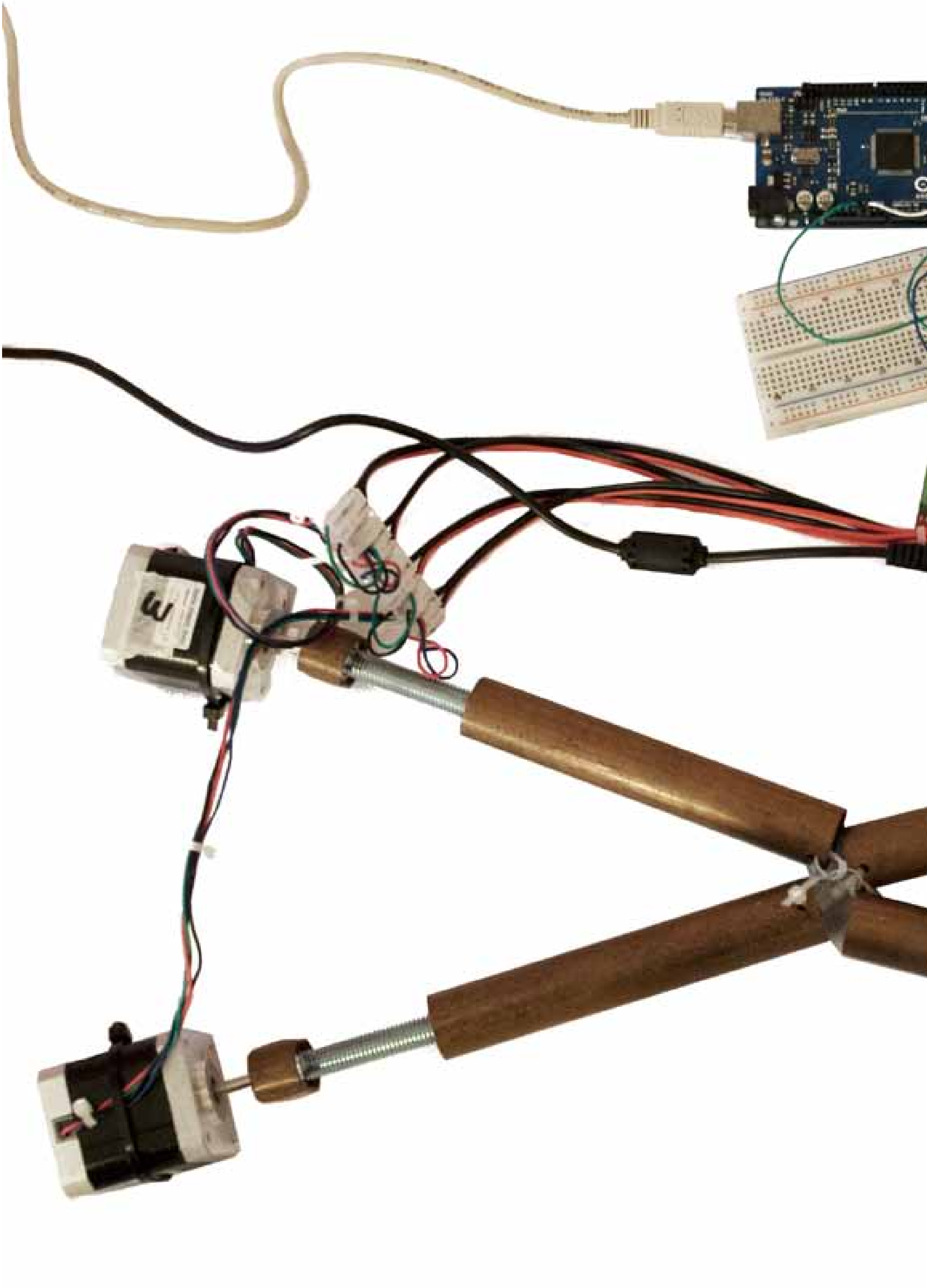


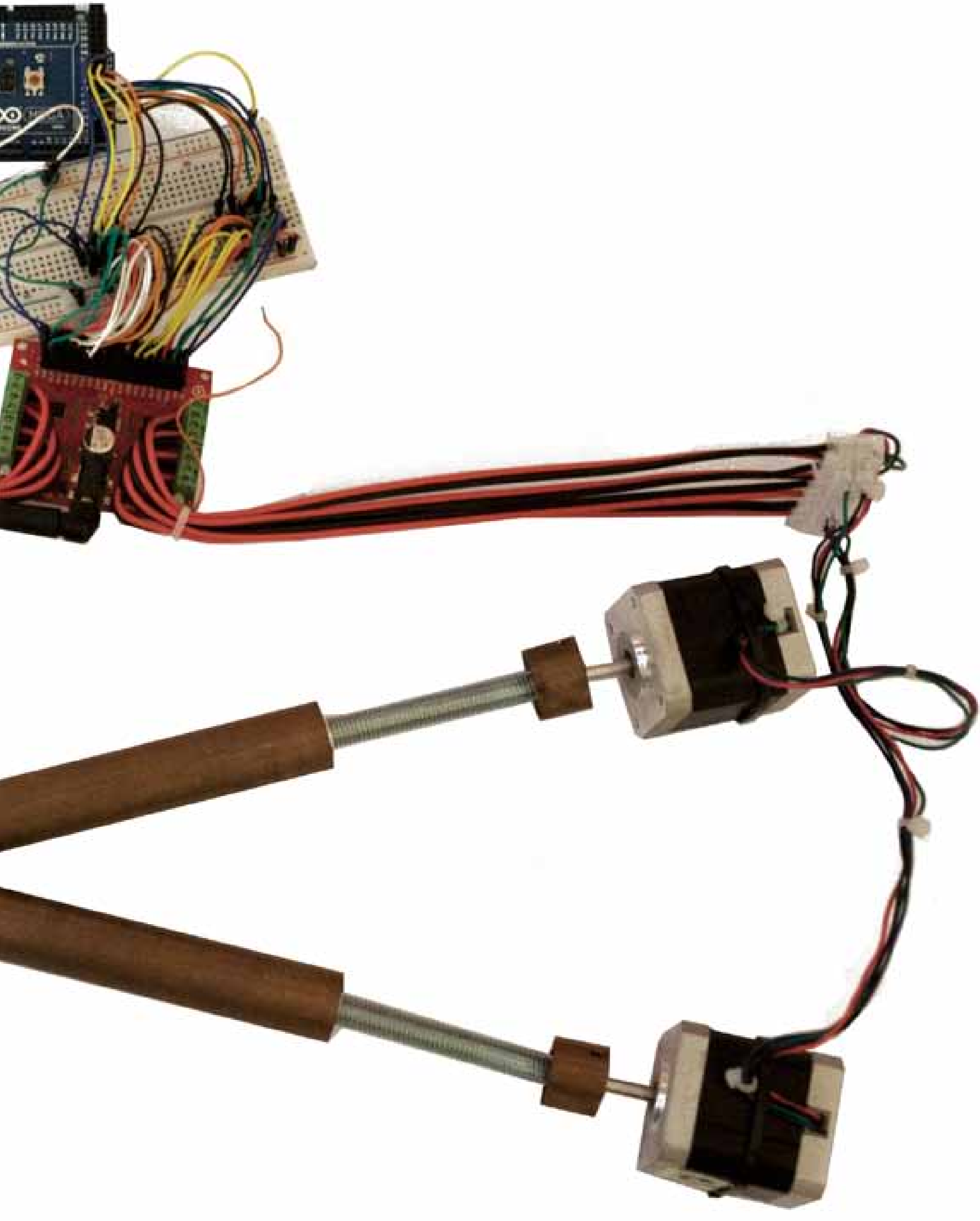
Kinect
Microsoft XBOX 360 / 2010



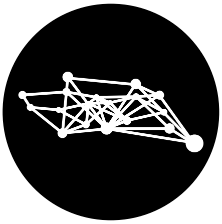
function


geometry
point
array
oscP5
gHowl r.2
representation

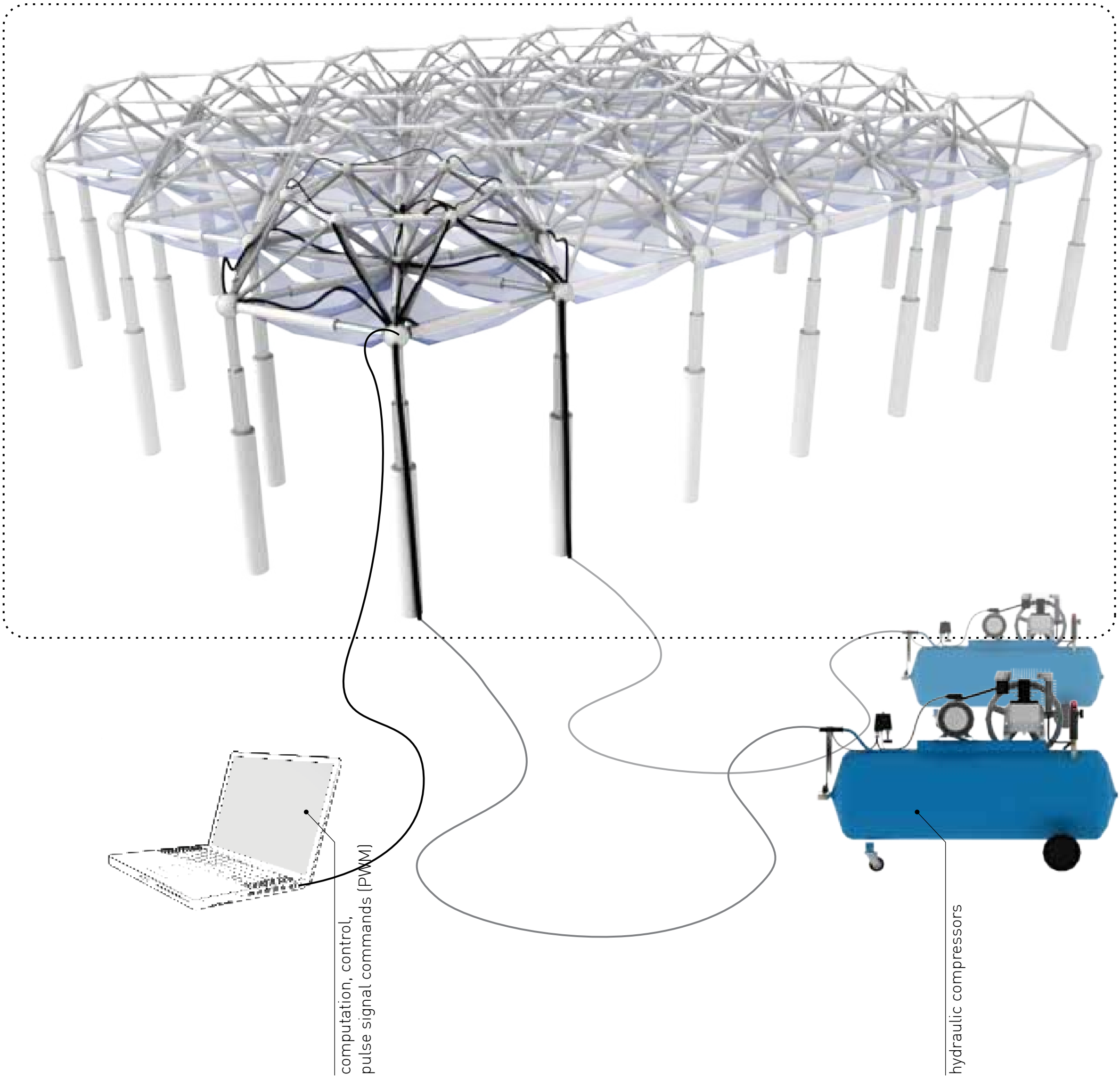




Functional model of mechanical pistons controled by stepper motors

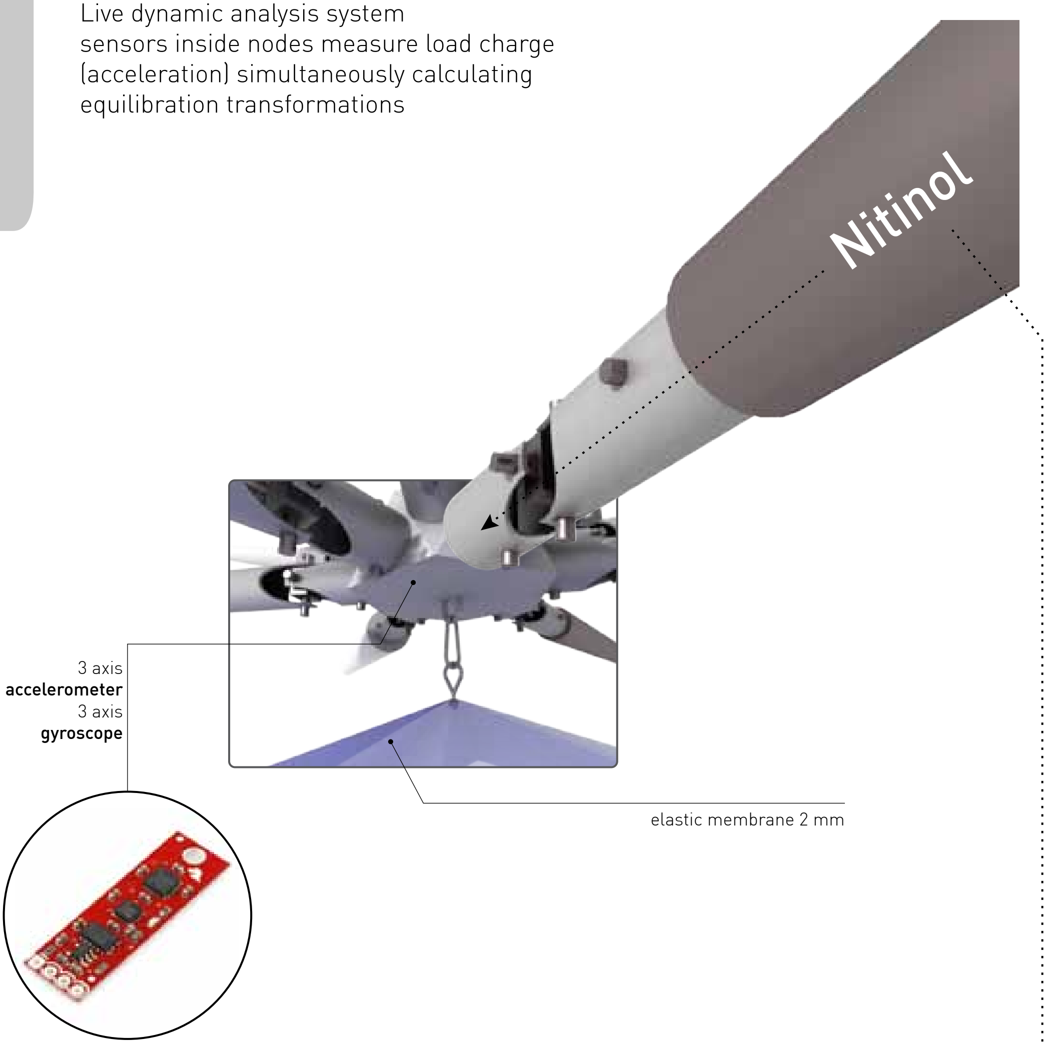


application details


computation, control,
pulse signal commands [PWM]
hydraulic compressors

Live **dynamic analysis**

Live dynamic analysis system
sensors inside nodes measure load charge (acceleration) simultaneously calculating equilibration transformations

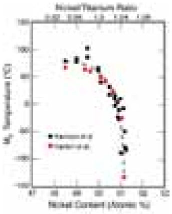


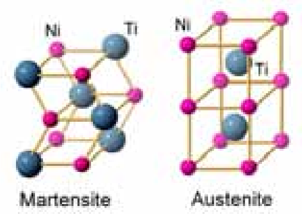
3 axis
accelerometer
3 axis
gyroscope

elastic membrane 2 mm

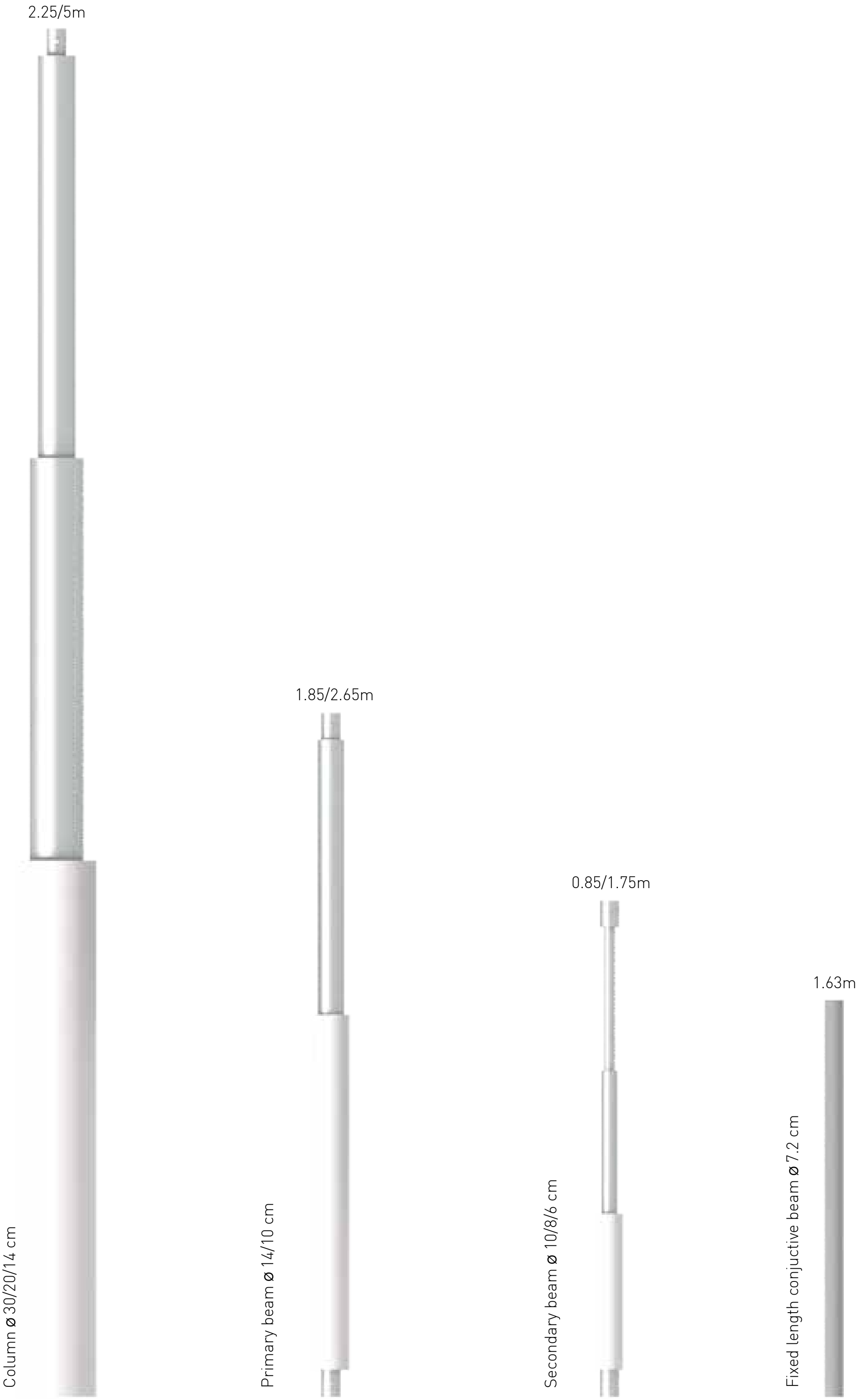
Nitinol alloy (NiTi)

_Superelastic behavior at temperature change
_Shape memory

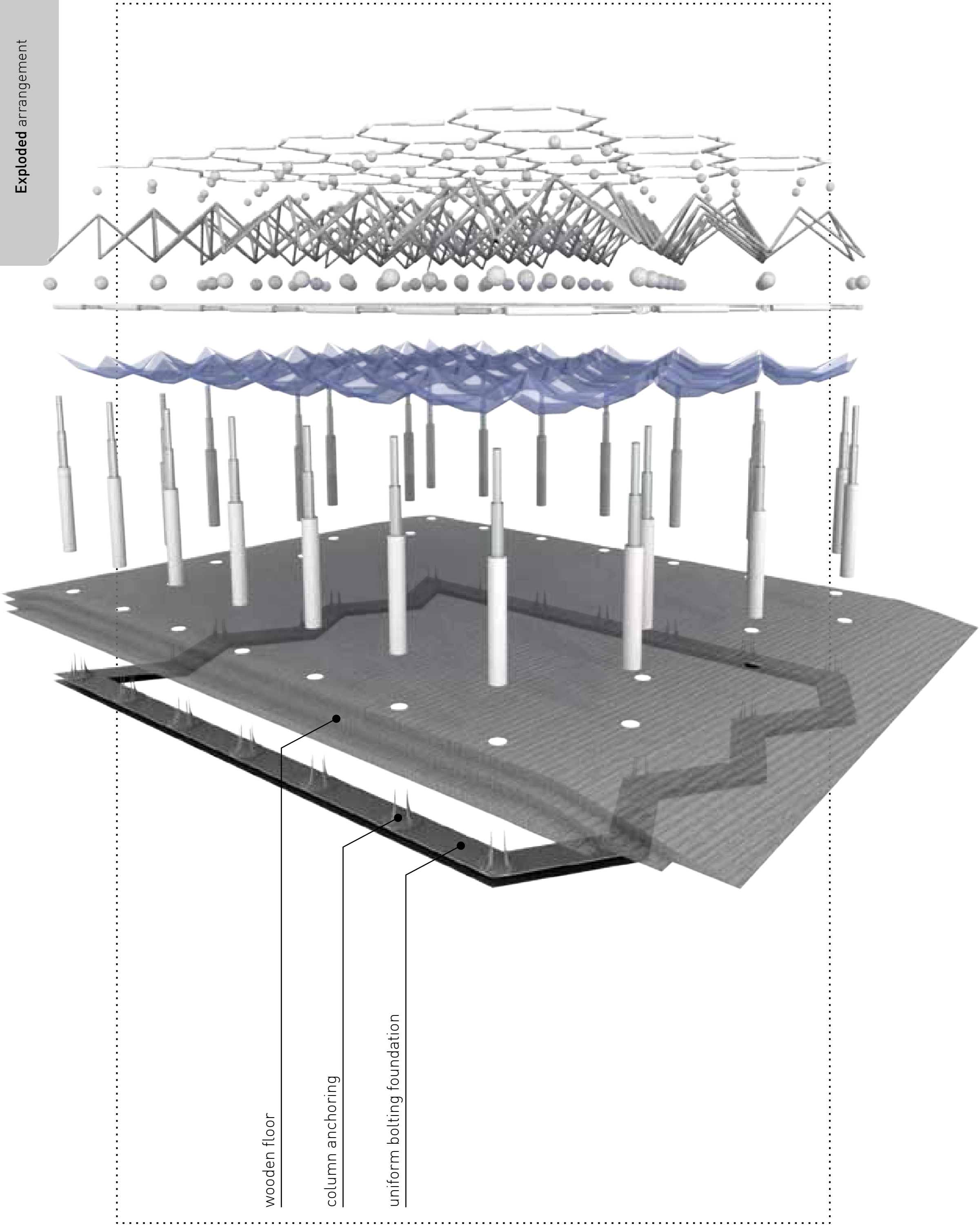


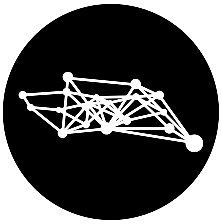


Element **dimensions**



Exploded arrangement





transformations

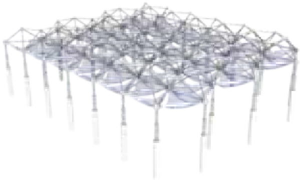
*The main functions (or modes) explored are two. The first is a performative one, where the structure reads the position and intensity of the sun, and transforms itself to produce the maximum shade, as well as the best shade density for the particular moment. The second is purely interactive, with the structure transforming constantly according to the movement of the people under it.

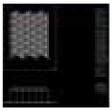
We want to stress the second mode (interaction) which is the one that the concept of this project is more interested in. Though it doesn't serve or solve any particular spatial problem, we find it very important through a different viewpoint; that of the social. The structure here, has the ability to do something in relation to what people do, be actuated by them, and also provoke them.

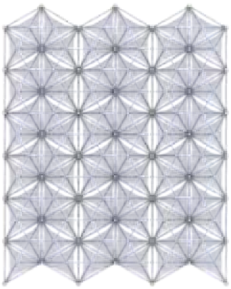
The structure sees people through a sensor. It doesn't see color, age, class or any other characteristics of that sort, and thus offers everybody a common capability to affect it. In this way it creates a social bond that did not exist before in that place, through a platform of collective morphogenesis. Furthermore, in the process of interaction with this mechanical structure, we do gain an exo-architectural surplus, that of interaction between people themselves; This social production by people unrelated to each other, that otherwise would never meet or interact, brought about by an architectural object, is what we aim for. This way placing the structure (or platform) at a city square, we want to encourage, through architecture, this kind of interpersonal interaction that aims towards the experience and possibly the reclamation of the sense of commons in the public realm that could also reflect in other aspects of urban and civic life beyond strictly spatial ones.

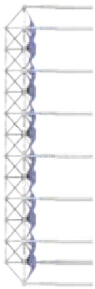
inertia state

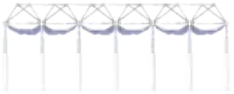
00**0**







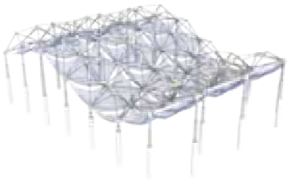


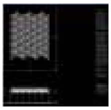


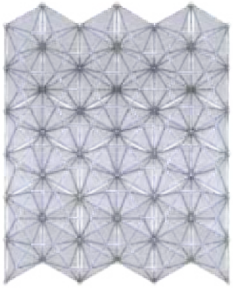
```
if (i%2==0)nodes[i][j]=new Node(50+i*startLen*cos(PI/6), 50+startLen*j+(startLen*sin(PI/6)), startZ, i, j);
        else nodes[i][j]=new Node(50+i*startLen*cos(PI/6), 50+startLen*j, startZ, i, j);
```

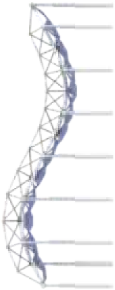
sine surface

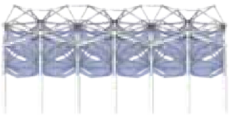
T0**1**







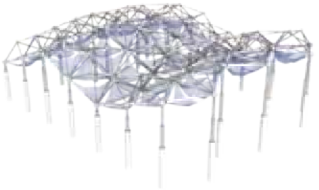


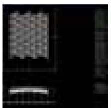


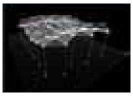
```
if (currIter<1)  mover= sin(py*1*PI/pcy);
```

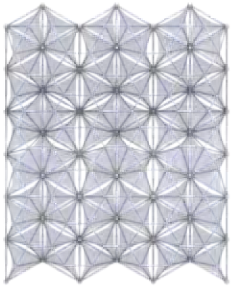
asynchronous sine - cosine surface

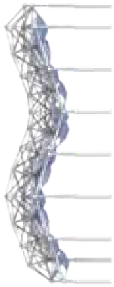
T02

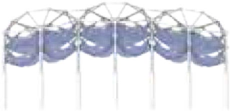








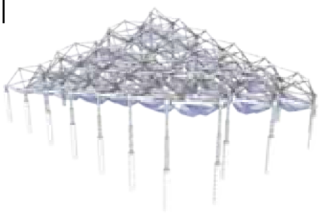


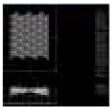


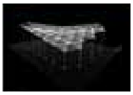
```
if (currIter<1) mover=sin(1.5*PI*py/pcy)+cos(+(0.5)*PI*px/pcx);
```

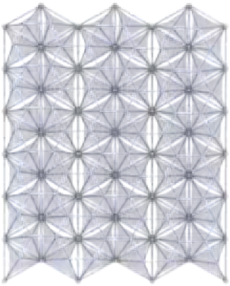
inverse tangent surface

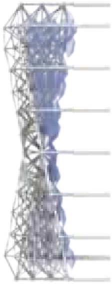
T03

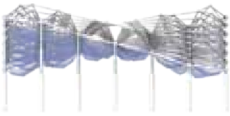








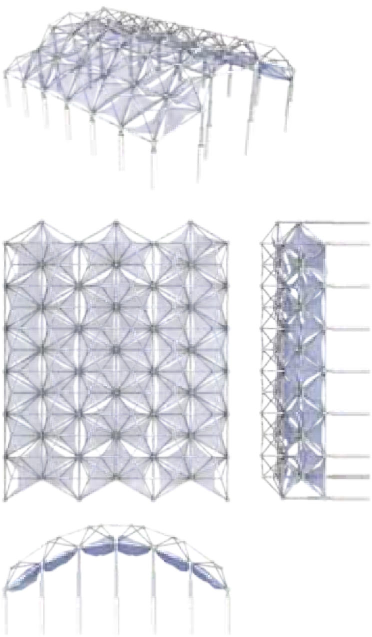


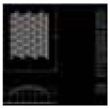


```
if (currIter<1) mover= atan((py*px)/pHeight);
```

cylindrical dome

θ01

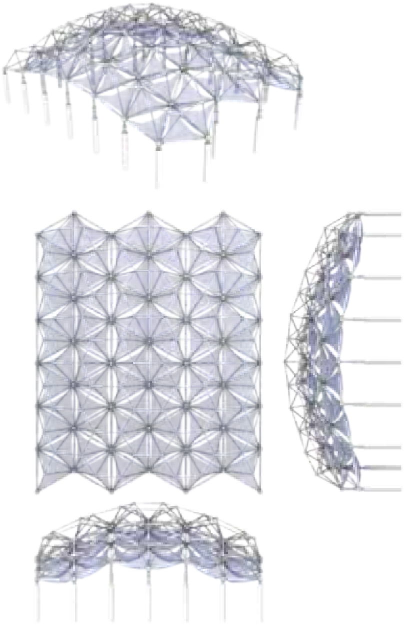


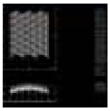


```
if (currIter<1) mover=-sq(x-pcx);
```

spherical dome

θ02

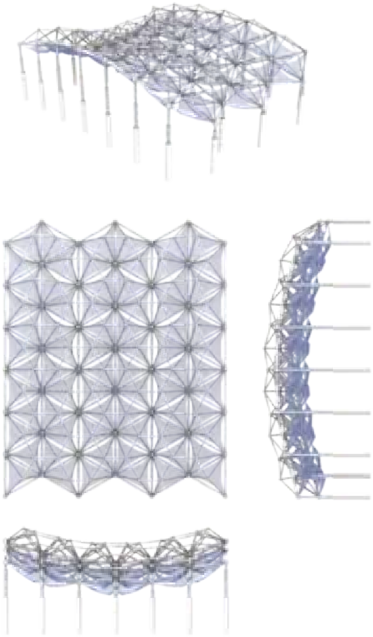


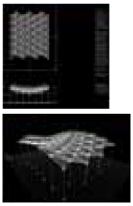


```
f (currIter<1) mover=-sqrt(sq(px)+sq(py)+5);
```

hyperbolic paraboloid surface

q02

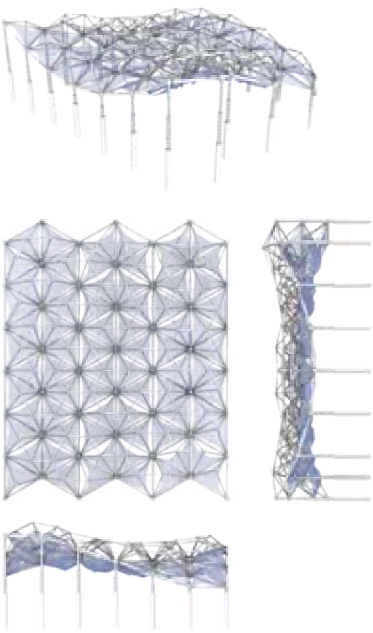


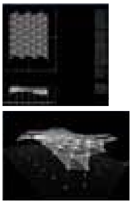


```
if (currIter<1)  mover=0.5*sq(px)-0.5*sq(py);
```

monkey saddle surface

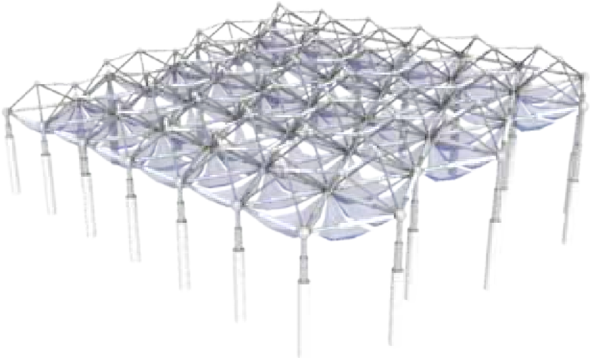
q02

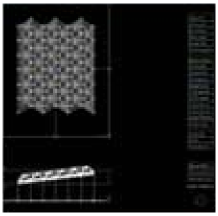


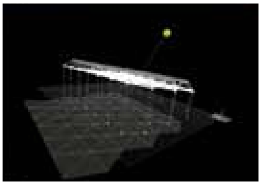


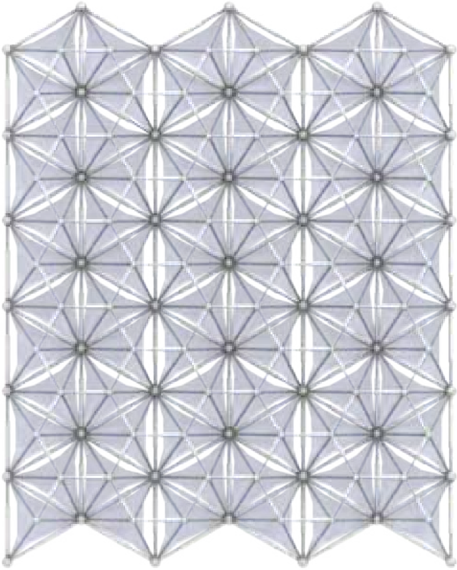
```
if (currIter<1)  mover=px*(sq(px)-3*sq(py));
```

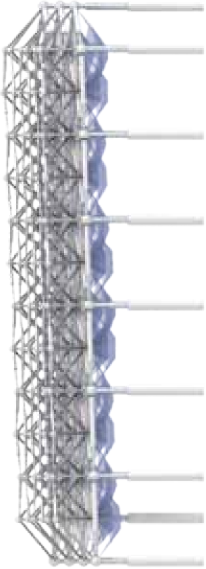
δ01 | reaction to sun movement

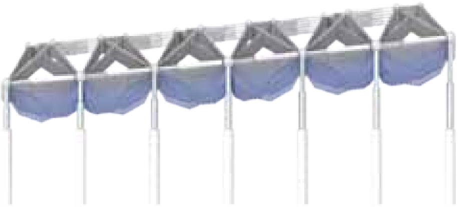






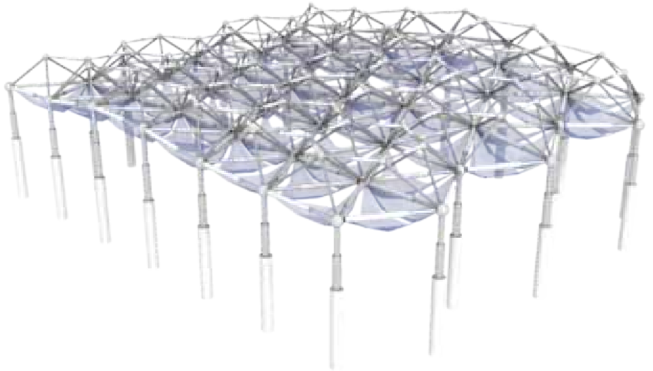


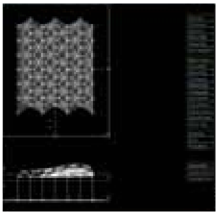


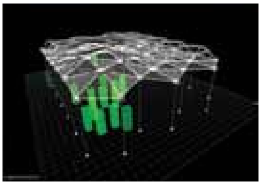


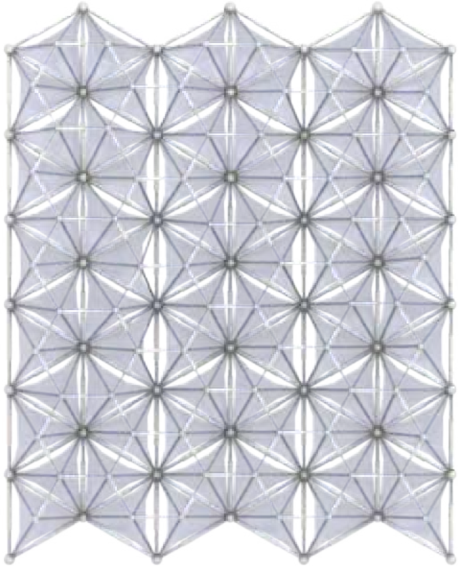
```
float sAngle=sunProjected.angleBetween(sunV);
float angleY=vAtan2(sunV, new Vec3D(0, 1, 0));
mover= 1/(sq(0.5+ (x)*0.008*sin(angleY) + cos(angleY)*(y)*0.008));
```

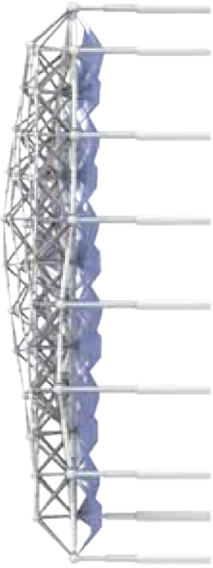
δ02 | interaction with people





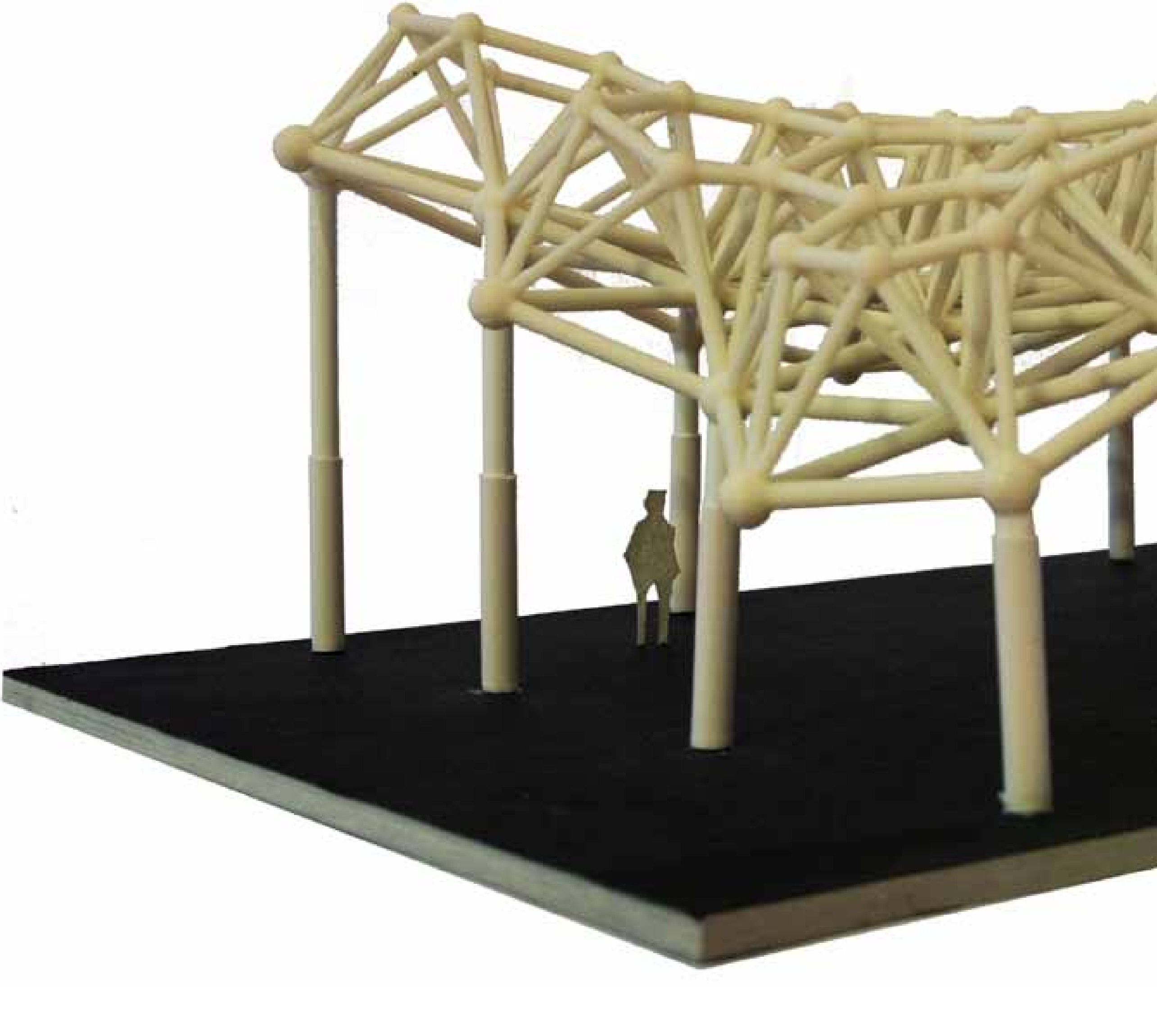




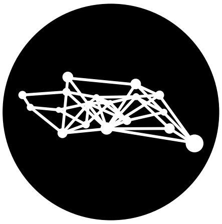


normal (gaussian) distribution

```
nodes[x][y].progVec.z+=pHeight*exp(-
sq(npDist)/2*sq(2))/2+sqrt(sq(particles[i].pos.x-nodes[x][y].pos.x)+
 sq(particles[i].pos.y-nodes[x][y].pos.y))/interactDistance
```

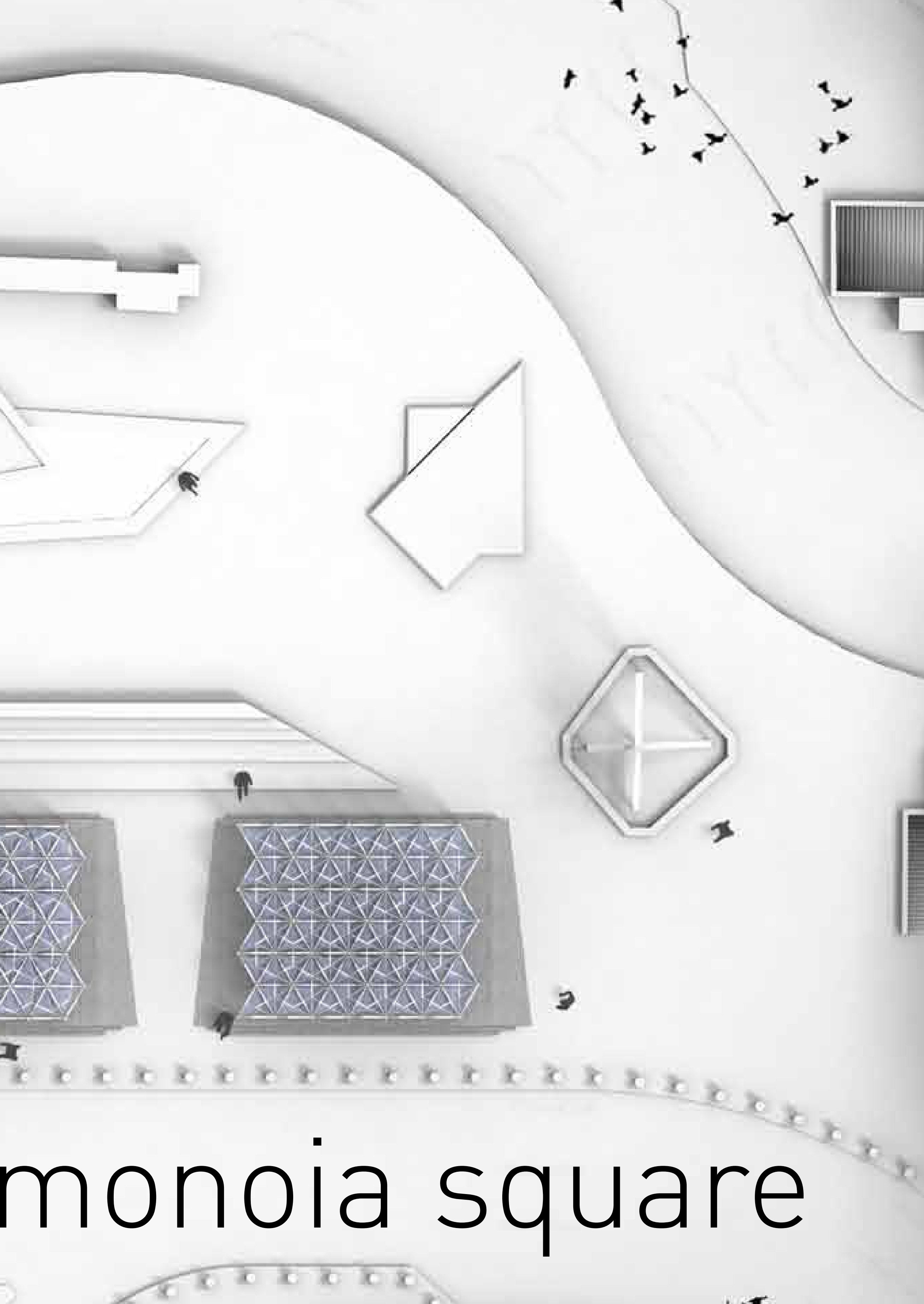


3D print model (monkey saddle surface)



experimental installation

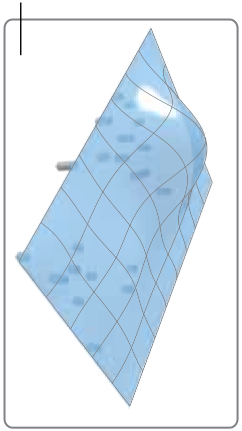
O



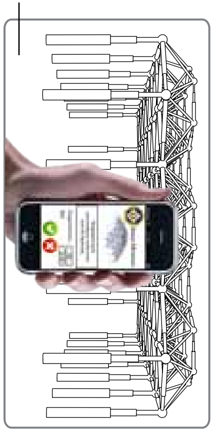
monoia square

Omonoia Square

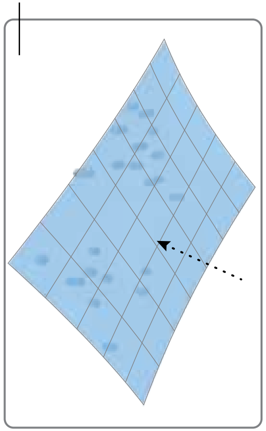
social interaction



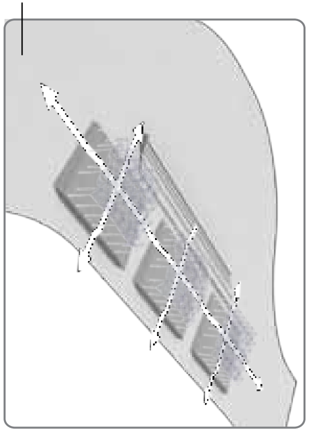
gatherng organization and reception

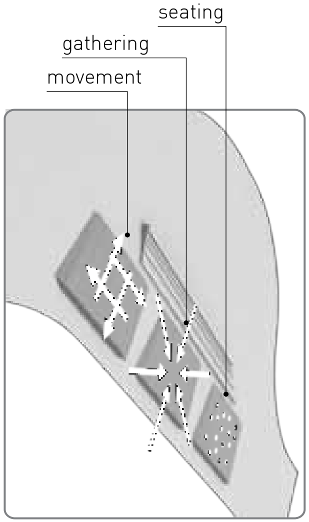


maximum shade production

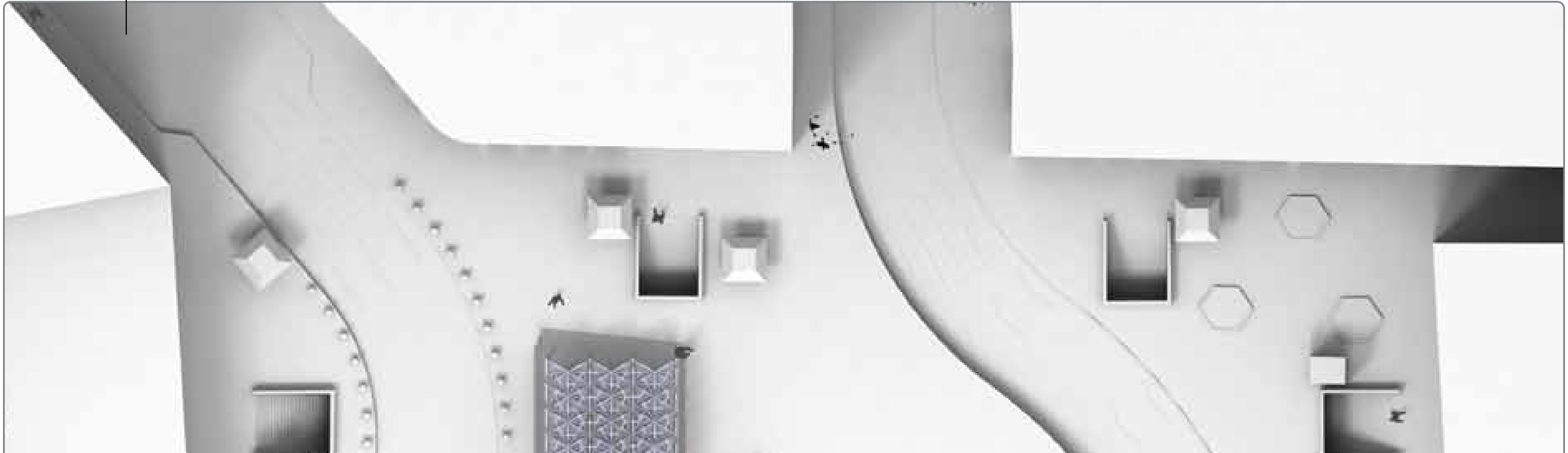


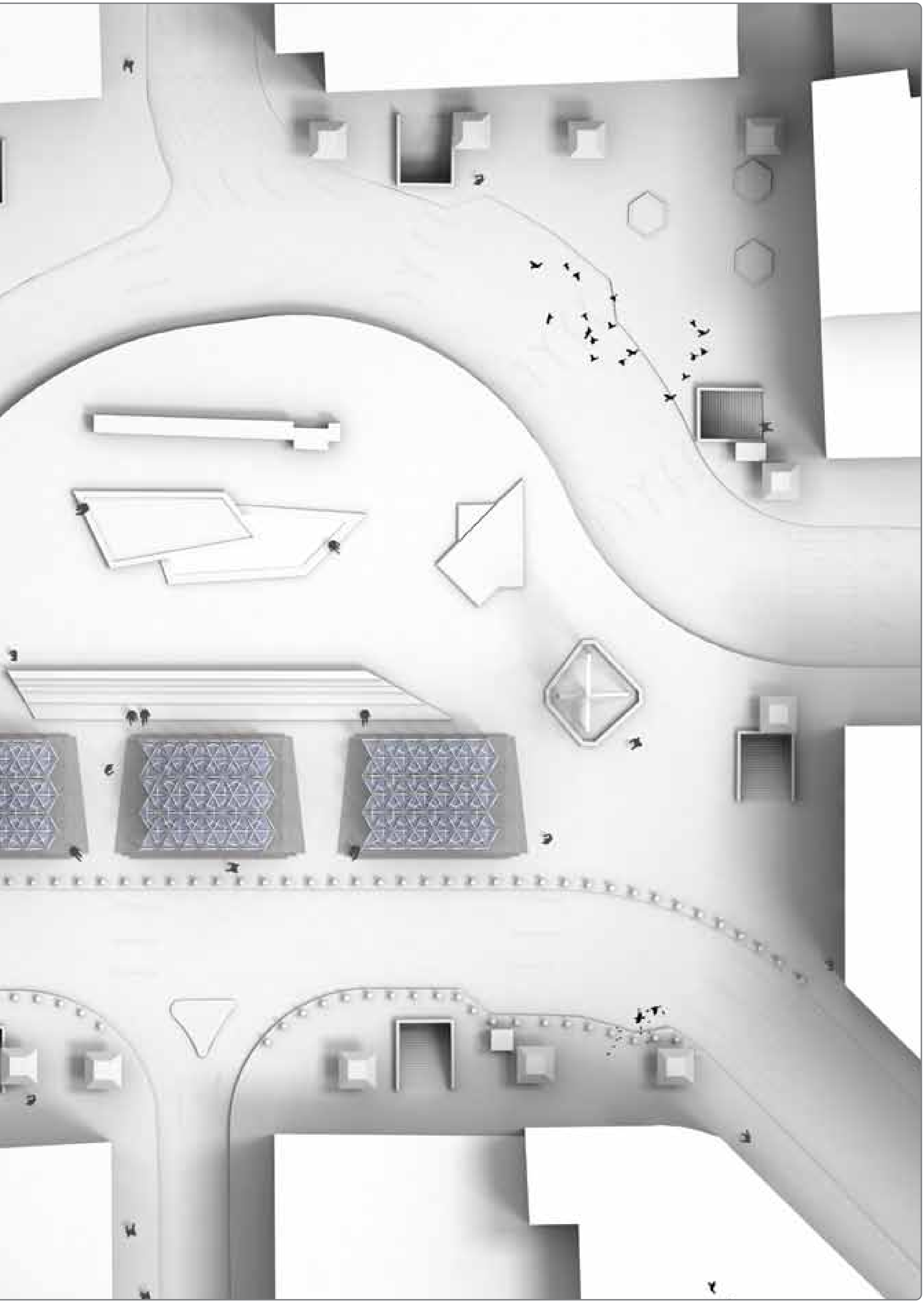
movement axes

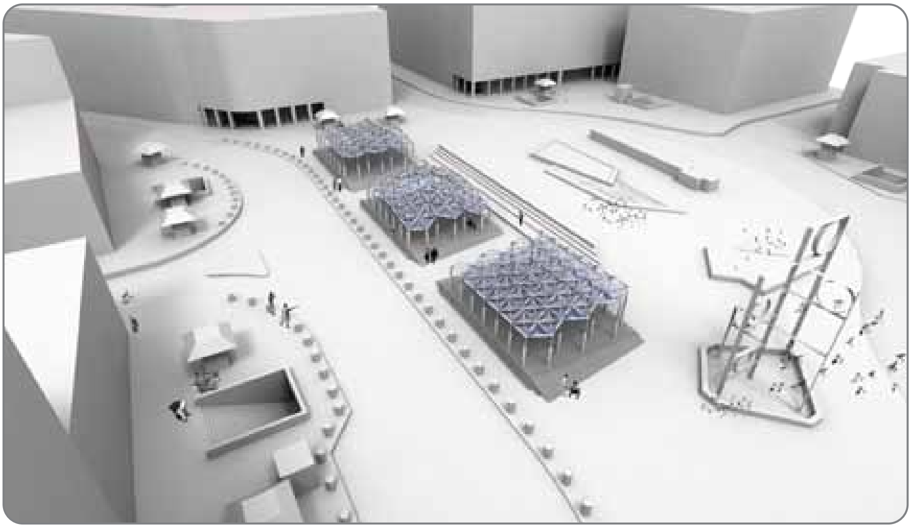


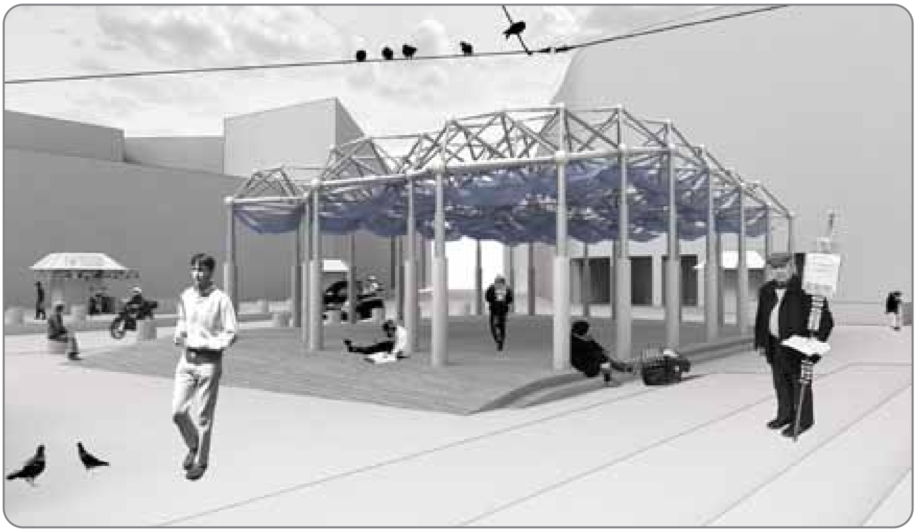


plan view









More of this project, such as videos, as well as open source chunks of code can be found at my blog:

http://at-nowhere.tumblr.com/