JOSHUA PARKER

SELECTED WORK

2007 -

JOSHUA PARKER, M. Arch., Syracuse University School of Architecture (SUSOA), B.S. Electrical Engineering, University of Washington (UW), Certification, Institute of Advanced Architecture of Catalonia (IaaC)

CONTENTS

ZHUHAI PORT PAVILION 5

Project Scope: Architectural Design of City planning museum for Zhuhau Port Authority | Collabortors: Ingame Architecture, Yang Zhang (principal in charge) | Role: Consultant for Ingame Architecture, Lead designer | Contribution: Concept/schematic design, computational design, design development

UPLIFT AIRSHIP HANGER 9

Project Scope: Design Competition, Airship Hanger, Anhui, China | Collabortors: OPEN Architecture, Chinese Academy of Building Research (CABR), LI HU (principal in charge) | Role: Consultant for OPEN, Lead designer | Contribution: Concept/schematic design, form-finding, computational design

PROTOCELL 15

Project Scope: metadesign, collective housing system | Collabortors: OPEN Architecture, Li Hu | Role: Lead designer, researcher

CHENJIACI CULTURAL VILLAGE AND PARK REDEVELOPMENT 21

Project Scope: Redevelopment of Cultural park & Square of Chan's Ancestral Temple, Guangzhou, China | Collabortors: Node Architecture and Urbanism, Doreen Liu (principal in charge), Jiebin, Huang, Shaoli Lin | Contribution: Concept/schematic design, masterplan for landscape/park nodes | Role: consulting architect for Node Architecture and Urbanism, in charge of landscape nodes

GUANGXI MUSEUM OF FINE ARTS 25

Project Scope: Design / Analysis of Building envelope | Collabortors: NODE Architecture and Urbanism | Roles: Design Consultant for NODE | Contributions: Concept/schematic design, massing, incharge of landscape and building envelope.

SPIRALLING: PARKING GARAGE 27

Scope: Custom parametric Tool for parking garage study | Collabortors: OPEN Archtiecture, LI HU | Contribution: Consultant for OPEN Architecture, Computational designer, form-finding

WEAVING: URBAN FURNITURE / SHADING 29

Scope: Custom parametric Tool | Collabortors: Kristiana Leniart | Contribution: Computational design

SAMPLING: JAZZ TOWERS, BAOTOU 31

Scope: residential towers | Collabortors: OPEN Archtiecture, LI HU (Principal in charge) | Role: Consultant for OPEN | Contribution: Custom design tool

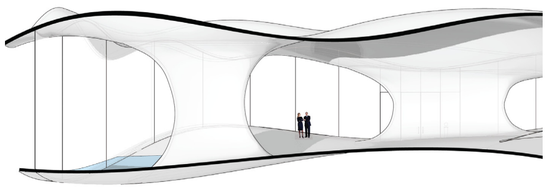
SOURCE CODE, UPLIFT HANGER 33

ZHUHAI PORT PAVILION

Pavilion to Showcase the 5-10 year development plan for the 7 port areas in Zhuhai. The plan of Gaolan Port is at the center of the display with the Wanshan Port being a supplementary. Requirements include dark rooms with circular model spaces 15m and 4.5m in diameter, Large display wall, cinema, lounge and general public space. Design to invoke the ocean and port and Zhuhai's position as a international port.

Project asked for a formal response to "sea and port". Continuous free-form surface provide language and medium to produce spaces evokative of flowing sea water while guiding people through continuous sequences of soft expansions and contractions that serve as exhibition space. the visitor experience is of being underwater. the language and meaning of port is also incorporated to describe moments where the undulating ground plane connects with ceiling to enclose ovular spaces that become oportunities for office space, bathrooms, cinema, etc.

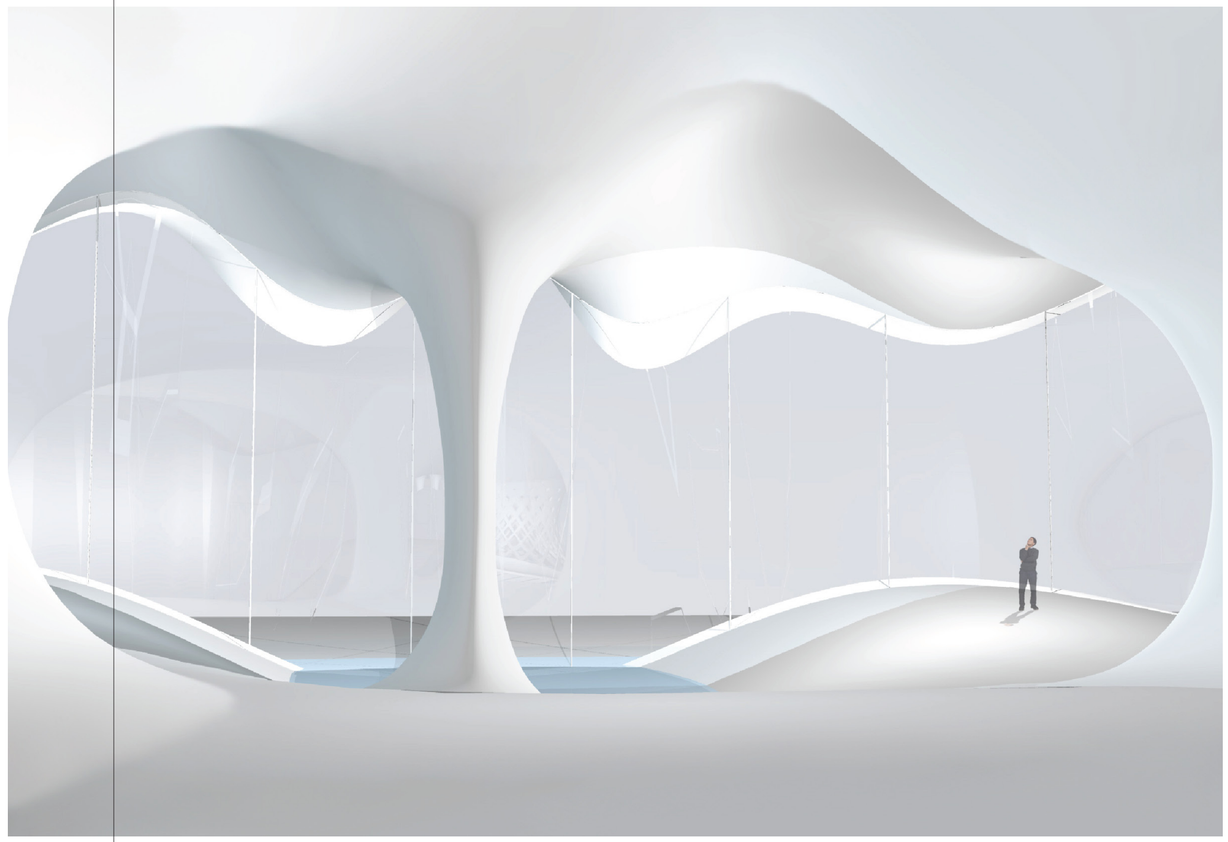
Project Scope: Architectural Design of City planning museum for Zhuhau Port Authority | Collabortors: Ingame Architecture, Yang Zhang (principal in charge) | Role: Consultant for Ingame Architecture, Lead designer | Contribution: Concept/schematic design, computational design, design development



1.

1. diagramatic building section
2. contour in plan, site area is approx. 4,000 square meters.
opposite: Renderings from street and interior view.

images and text courtesy of Ingame office

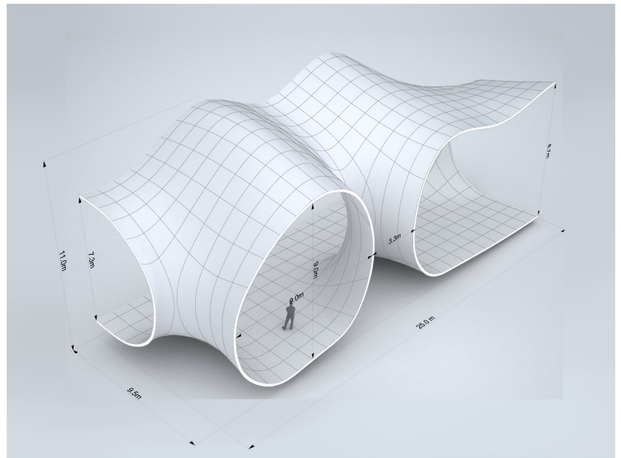


Program was divided into two groups. For the most part public activity space, lounge and exhibition occupies the main interstitial space. Remaining program: offices, bathroom, cinema, dark rooms, etc. occupies ovular spaces created by surface deformations. Design was a process of placing circular program in plan, taking the centers, and radius, as location and size of surface ports. A port pulls ground and roof planes together, effectively disturbing the surface as it self adjusts according to structural criteria. The surface, in turn, pulls back stretching and elongating the circles, creating ovular spaces. The cumulative effect of all ports is a heavily undulated surface pair. The frequency and amplitude of irregular waveforms depends on incremental spacing of ports and ground ceiling height.

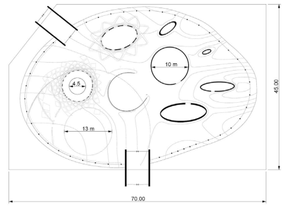
The thin-shell structure is realizable with reinforced concrete shells and plywood formwork. This has proved to be an economical and efficient solution used by free-form structures in recent years, most notably by Mutsuro Sasaki working in collaboration with Toyo Ito. Of particular relavance as design and structural precident was the Kakamigahara Crematorium. Based on design precidents and structural heristics, we can anticipate surface thickness of 200 mm. This creates an exceptionaly thin and light structure (see img below). The large overhang is additionally supported by steel columns that interupt otherwise continuous glass infill that wraps the structure at the edge minus an offset that also varies according to structural requirments.

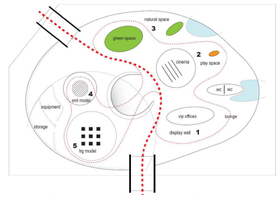
The structure would require variable reinforcement to support the distributed vertical load of approximately 1 KN/m2, as well as the horizontal wind and seismic loads. The arragement of reinforcement can by determined with calculus-based optimization software, which calculates incremental compression and bending forces along surface, then produces a reinforcemnt map. According to Sasaki, the ideal construction solution also includes steel plates to better hold the form and counter the loss of rigidity due to cracking. He describes a “hybrid construction of steel plates and cncreete”. Plates would act both as formwork for concrete and then fixed in place for a hybrid cross section.

A custom paneling tool was built with java/processing using open source geometry library (igeo, by Saturo Sugihara) to interogate form and explore the possibility of perforating the shell in places to create visual and physical access through vertical surfaces. Tool defines a uv diagrad and maps performation patterns onto surface using simple algorithms and exposing design parameters and enable real-time collaborative use by architects and engineers.

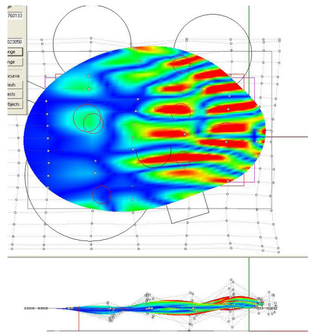


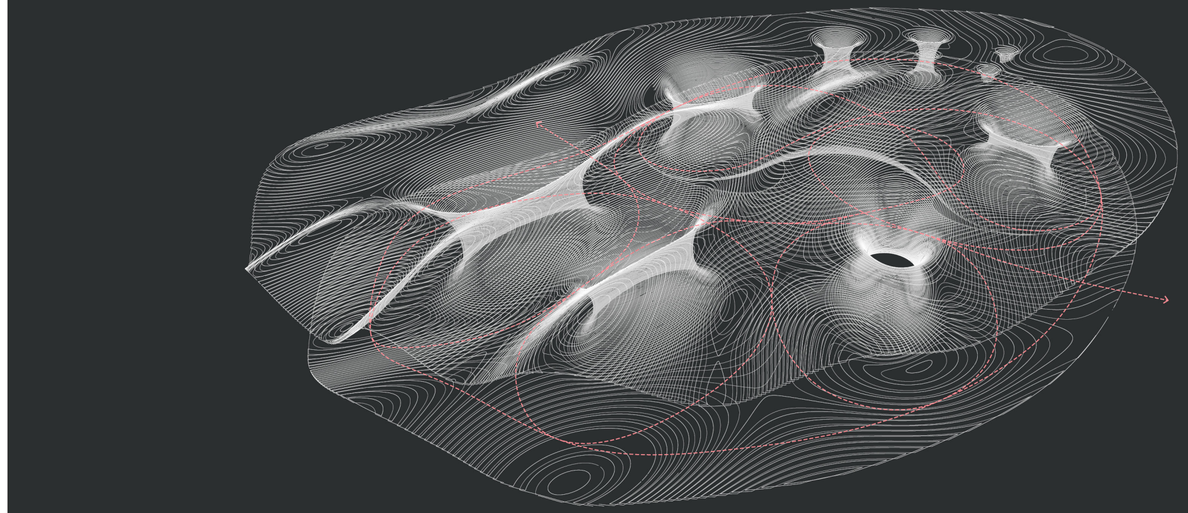
Below: floor plan contours, and programatic relation to primary circulation paths.

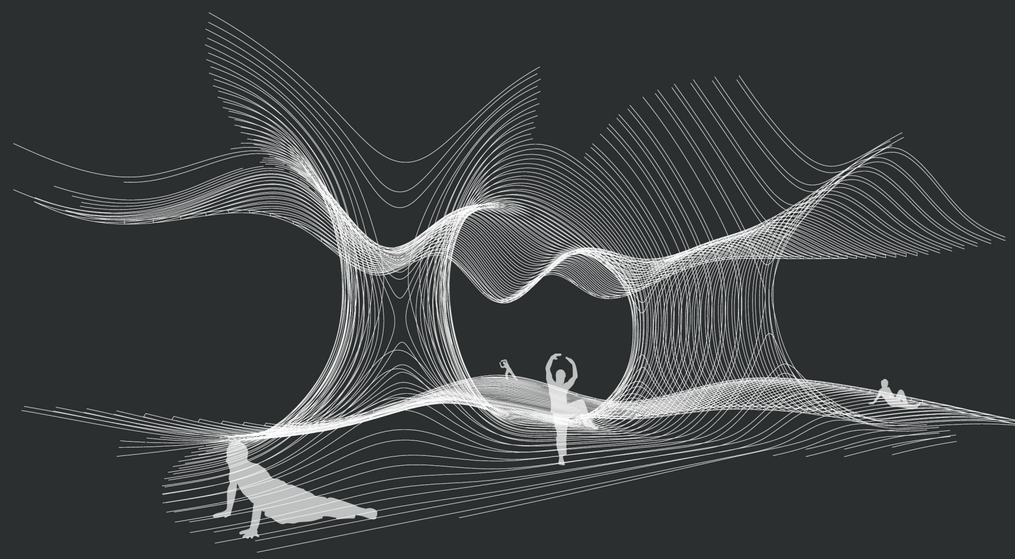




Left: Sectioned axon showing scale and formwork.
Below: Curvature analysis, blue indicating areas of minimal curvature, and red, areas of maximum.







Top: Contours and interior circulation.

Below: lateral contours express form and indicate possible structural approach.

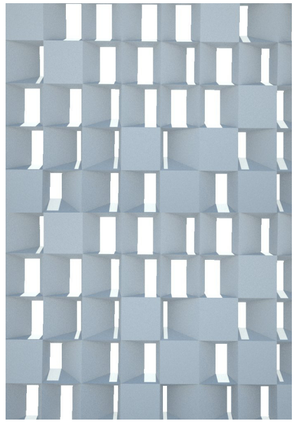
UPLIFT AIRSHIP HANGER

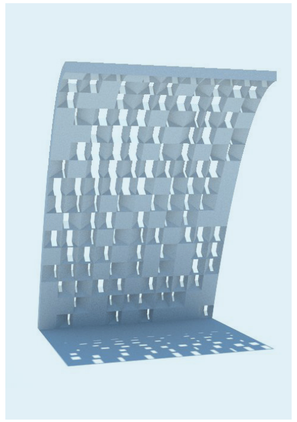
Layers of earth are thrust upward by the geological process of uplift exposing a spatial pocket in the earth for airship operations while generating unexpected oportunities for secondary function as form becomes manifest as a “natural” landform that seemlessly integrates with site both visually and systemically as a member of a self-sustaining networked ecology of tree-covered mountains and landscapes.

Prefabricated concrete frame and panel system realizes an innovative and fully integrated structural skin that carries vertical loads of structure, roof instalations and ceiling crane while oposing lateral forces with highly optimized shape that minimizes material while expressing the combined effi ciency of structure, economy and form.

Flexible and highly effi cient structural framework minimizes maintenance and construction costs while permitting a highly differentiated and modular system of cuts and openings that modulate natural light, facilitate ventilation and rain water management, as well as generate suffi cient energy to realize a zero mainanence green roof and fully integrated building system that produces all of the energy it consumes.

Project Scope: Design Competition, Airship Hanger, Anhui, China | Collabortors: OPEN Architecture, Chinese Academy of Building Research (CABR), LI HU (principal in charge) | Role: Consultant for OPEN, Lead designer | Contribution: Concept/schematic design, form-finding, computational design

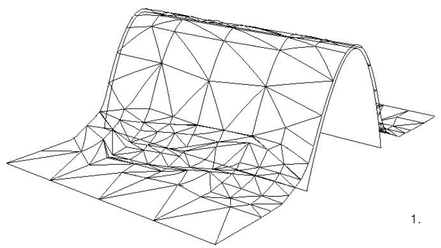




1. uplift formation concept
opposite: arial view of landform

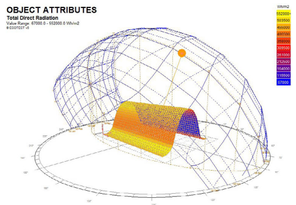
2-3. possible lighting variation acheived with integrated paneling system.

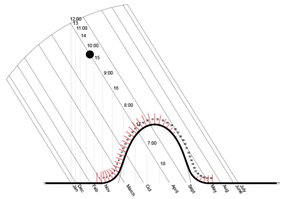
text and images courtesy of OPEN Architecture
more at http://openarch.com/task/1454



1.

above: site photos of tree covered land-scape





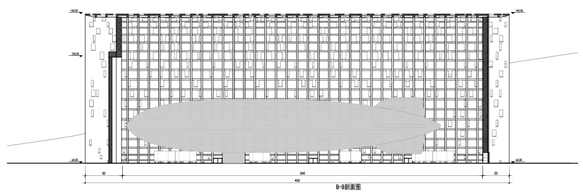
Above: seasonal solar exposure of build skin.

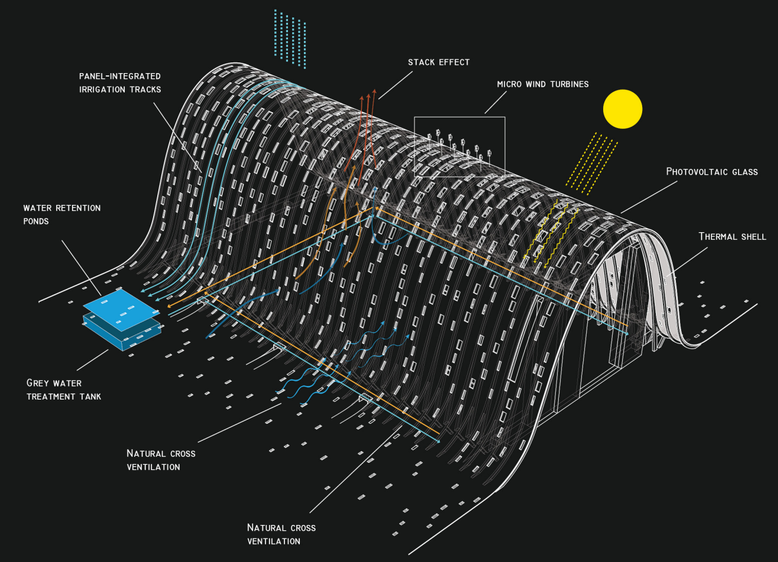
Primary structure consists of 64 concrete ribs that are arrayed along the length, and which follows catenary curve. This hightly efficient form allowed us to minimize material and cost, while supporting an exceptionally thin and light-weight skin. It was important that the skin be expressed as lifting up and not visually oppress the space below so this apparent lightness was critically important architectural criteria met with structural intellegence.

Lateral support beams run the length of the structure, stabalize it against lateral forces and establish gridded framework for prefabriacted panels and integrated energy systems. Framework and panel system comprise the structural skin, which follows catenary form of load-bearing ribs and peals away at base to express itself as a continuation of the ground plan, ie. layer of earth lifted up from it. This separation of skin from primary structure also creates an interstitial space that runs the lenth of the hanger to be occupied by offices, storage and secondary program.

Structure at ends also frame the hanger doors, consisting of four segments, which run along tracks, supported by truss at the top and in ground at the bottom. As doors open they penatrate the skin, where ommissions of laterial supports in the framework can be seen. This did not compromise structure as framework was sufficiently redundant to allow for the ommission of beams in places where door penetrates skin.

Right: Sections. Primary structure can be seen to follow a catenary form. Skin can be seen to separate at base creating interstitial spaces.

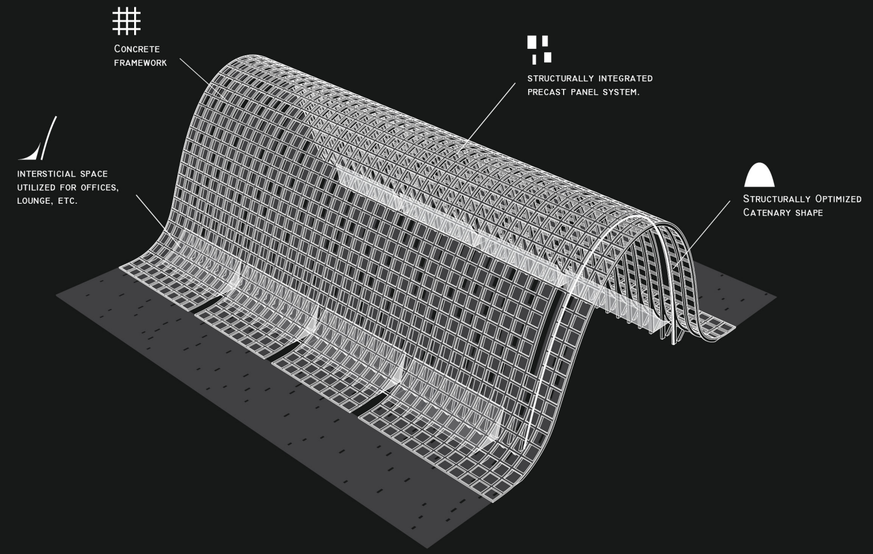




INTEGRATED STRUCTURE /
PERFORMATIVE SKIN

pasive energy systems

modular system of cuts and openings that modulate natural light, facilitate ventilation and rain water management, as well as generate suffi cient energy to realize a zero mainanence green roof



structural framework

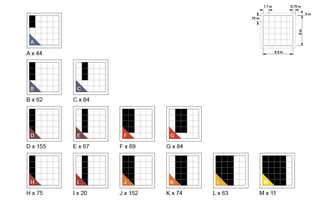
minimizes costs while permitting a highly differentiated and modular system of cuts and openings.

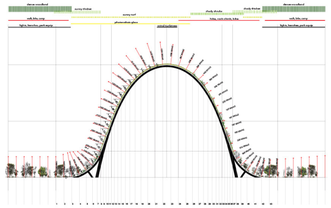
Using solar analysis software we determined the annual direct and indirect exposure of each panel. This data informed the arrangment of solar collection, diffuse light control and green roof programing. Modulation of light was acheived with 13 discrete panels types, which permit direct and diffuse light to penetrate skin. Penetration pattern responds most directly to solar profile and grade, and is then adjusted to satisfy programatic mapping of recreational trails, activity patches, and maintenance tracks.

Indigenous trees and ground cover are specified for landscape at the base and provide shading for trail network that extends outward and up the structure from the base, connecting greenroof ecology and public park services with surrounding forest and hills. Trees extend partway up structure, and ground covers the entire surface. Top soil and grass is supported by ecoweb mesh installed on top of structural skin.

Interior lighting requirements are met during day with system of cuts and opening in the skin. This also produces a controlled and highly dramatic lighting effect that registers time of day and quality of natural light. During night artificial lights produce a light pattern on the exterior skin.

Custom software tool was developed to search design space specified by parameters describing the panelized system of cuts and opening to meet lighting requitements and expore lighting effects in real-time. Alternate system of apperatures, shown above, produce slight more dynamic expression of light control that better registered the relationship between skin and environment, however effect came at a cost: added conmplexity to the prefabriced paneling system.

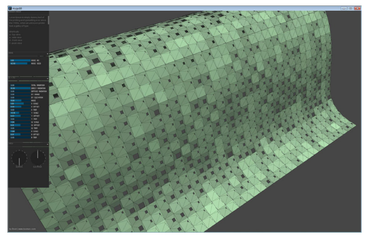




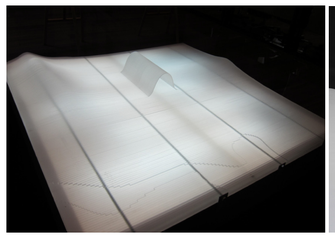
Above: Modules and encoding sceme based on solar exposure and program mappings.

1. Bottom Left: screenshot of custom paneling tool generates perforation pattern based on solar data, as well as light and program constraints.

2. Bottom Right: images of landform model



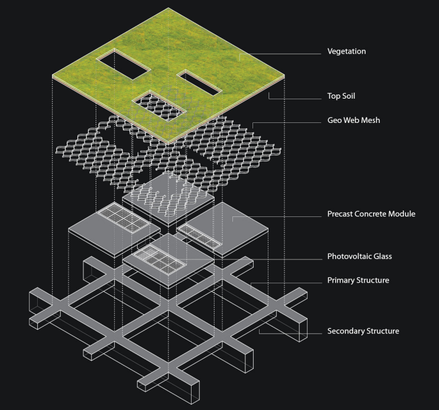
1.

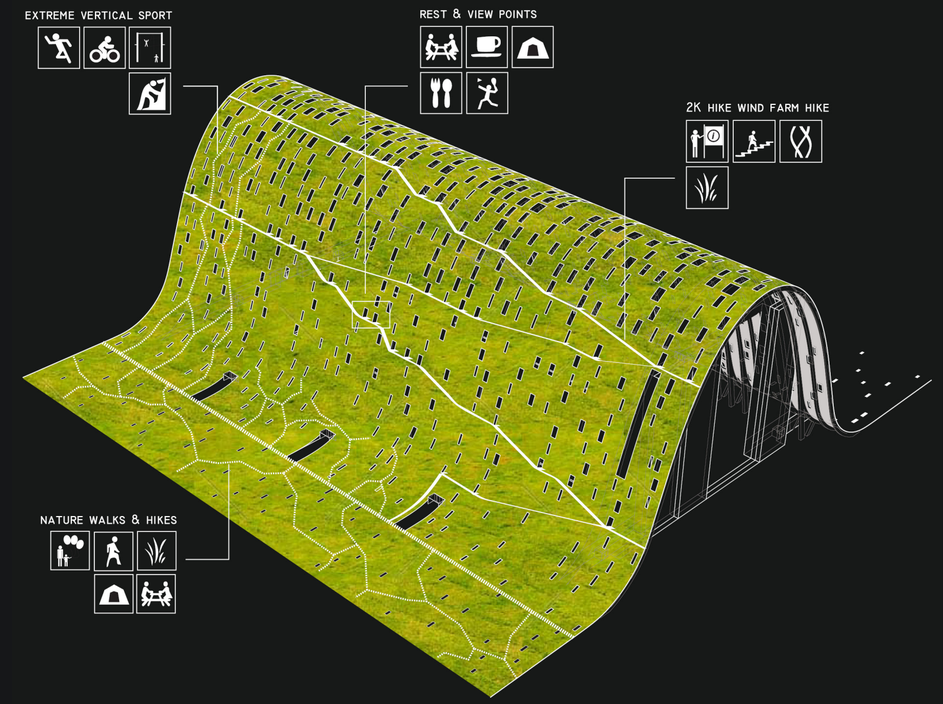


2.

SELF-SUFFICIENCY, PROGRAMMED GREEN ROOF

left: diagramatic detail of integrated construction layers. prefabricated panels with pre-installed solar glass fit onto frame. eco-web mesh is installed ontop of frame/panel system. hardware not shown.





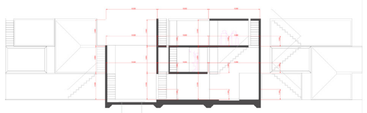
above: programatic diagram of greenroof. program is mapped on to surface according to grade, plantings, and sun exposure.

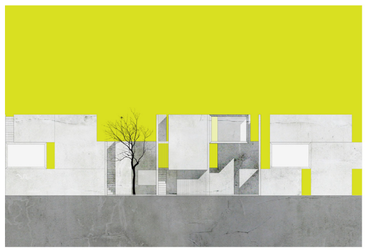
PROTOCELL

Research project initiated by OPEN Architecture proposal for Netdragon commune At the virgin beach front area where Min River meets the Pacific Ocean, and adjacent to Fuzhou Changle Airport, Netdragon Websoft Company's new headquarters building is currently under construction, embedded in it, the grand visions for the future of the company. Nearly 700 employees and their families will move from the center of Fuzhou city to this new campus. A 50,000m² land which used to be an eel farm will become their new living quarters. This is a very interesting and young community of people. Their work is to create wonderful virtual worlds that entertain millions of people. In real life and on this land of their new dreams, they need a unique living area and a brand new collective life style.

The Netdragon Village project is an architectural experiment investigating how to restructure relationships between people and nature, and also relationships amongst people themselves. The project seeks to create a ecological habitat for the future at a comparatively low cost.

Scope: Metadesign, collective housing system | Collabortors: OPEN Architecture, Li Hu | Role: Lead designer, researcher

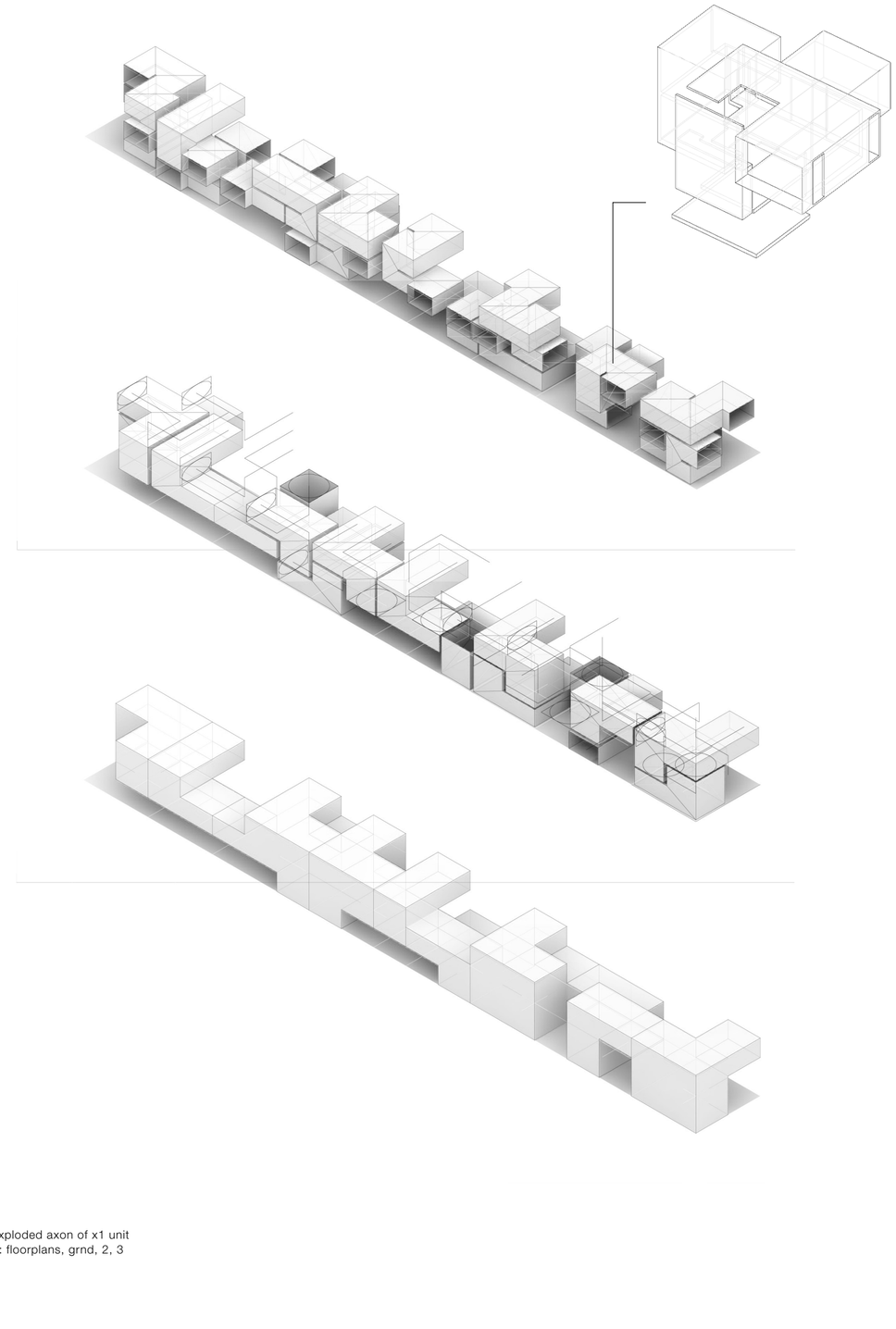




1. self-organized arrangement of approximately 100 four-cell housring units.

1.

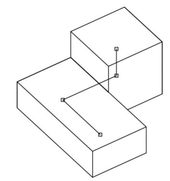
x100 habitat



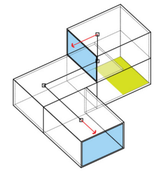
above: exploded axon of x1 unit
opposite: floorplans, grnd, 2, 3

The algorithm, initially implemented in rhino-script, can be divided into three isolated stages. The first distributes cellular mass according a CA logic and optimizer that adjusts massing for circulation paths and other encoded constraints like site, topographic features, etc. The first stage requires as input an initial massing condition as cellular automata procedes as incremental refinement.[1] First stage is also responsible for constructing a cell network, ie. each cell maintains state and keeps track of neighbooring cells. This network is passed to stage two along with further building constraints like program, floorspace allocation, etc. The second stage subdivides single-cell mass into multi-cell clusters or linked cell chains. This is accompolished by establishing an initial condition of randomly placed conditional units and defining a fitness function that evaluates the placement and situation of the unit and simply discards losers and ignores winners. This is a weak genetic optimization algorithm that is more brute force than anything else, but it does find solutions in reasonable amount of time. The final stage embeds building intellegence into cell clusters, places utility core, and adapts unit to basic building constrains... All stages are intended to be a guided meta-design processes, in which designers participate in real time by making subjective decisions and feeding them back into the system.

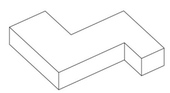
1. boundary condition
2. massing after 1st pass
3. cellular chains form
4. cells self adapt to local context to improve access and views.
5-7. schematic diagram and psuedo-code
7. resultant housing unit



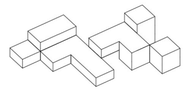
3



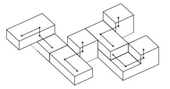
4



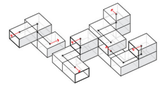
1



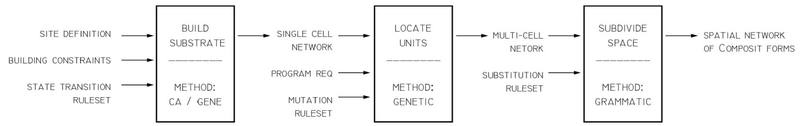
2



3



4



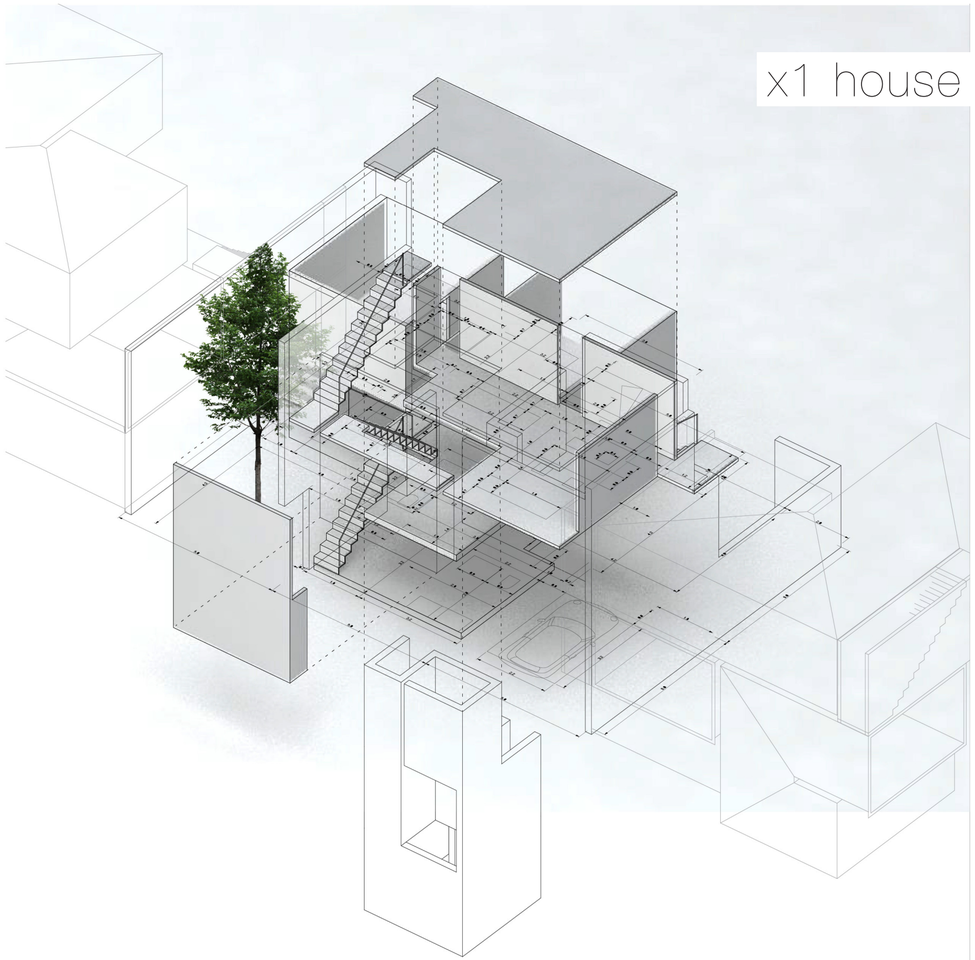
5 6 7

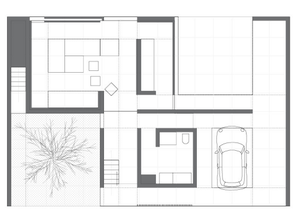
```
//input
site boundary
module size
far
porosity
avg height

ruleset obj

//calc 2nd order params
site area from bounds
floor area as site_area*far
mods as flr_area/mod_size
seeds as mods/porocity
levels as avg height
spl as seeds/levels
cvg as spl*mod_size/site_area

//build
create 2d pt grid
create cells from pts
assign random type

link cells to neighbors
define neighborhoods

//refine
```

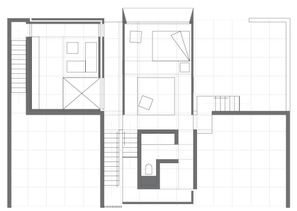
```
//input
array of linked cells
num of 6-cell units
num of 12-cell units
max height

ruleset obj

//solve
find candidate snake
Apply ruleset
(neccessary conditions)
repeat until all found

//optimize
pick snake at random
calculate fitness
apply ruleset
repeat

//output
for each ordered cell set
connect pts with polyline
sweep polyline
```

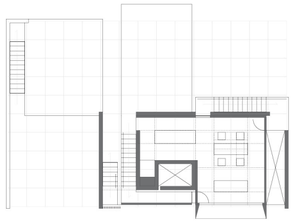
```
//input
array of spines

ruleset obj

for each snake:
apply ruleset
```

x1 house

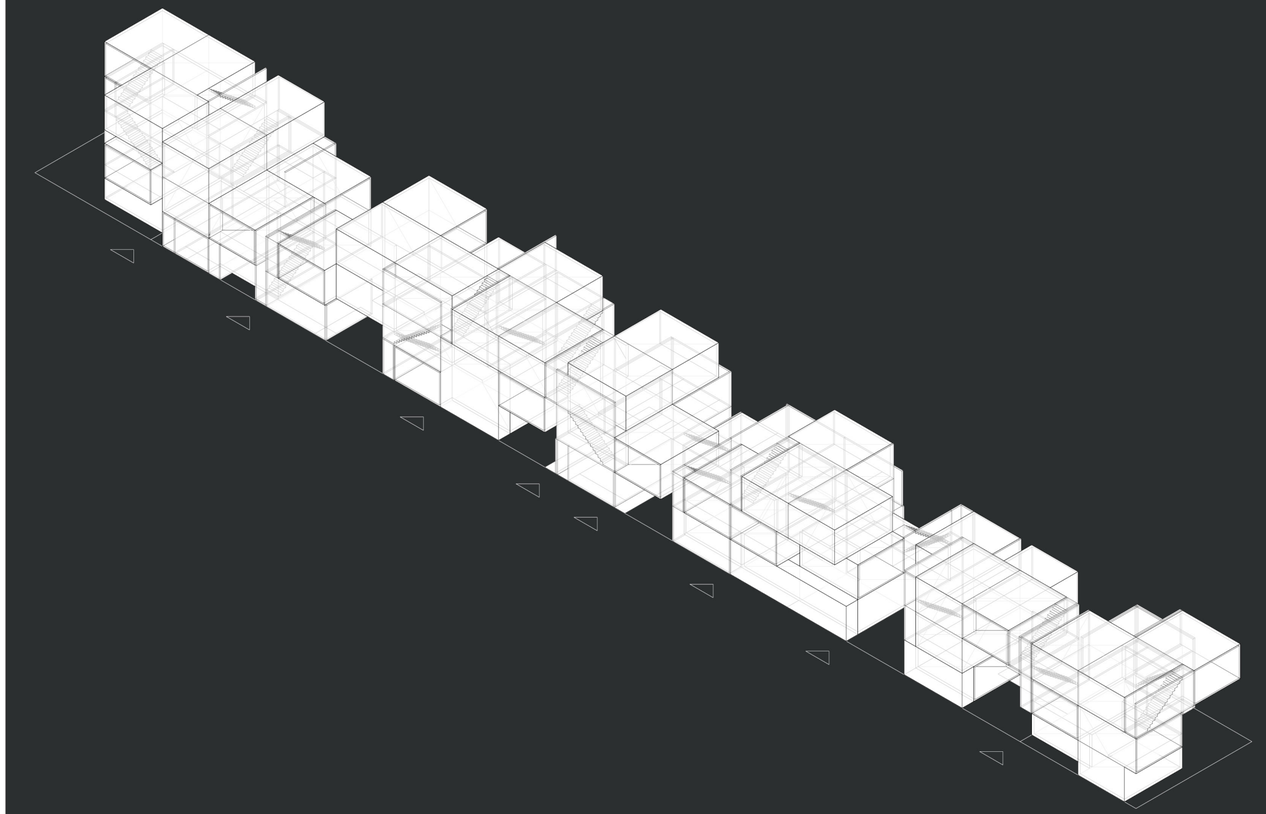


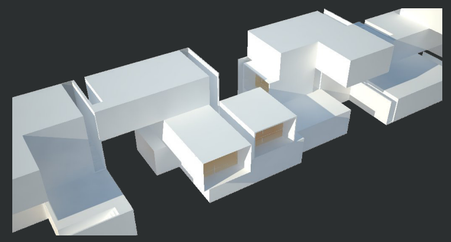






rightt: x10 housing block with x1 unit indicated with white box.





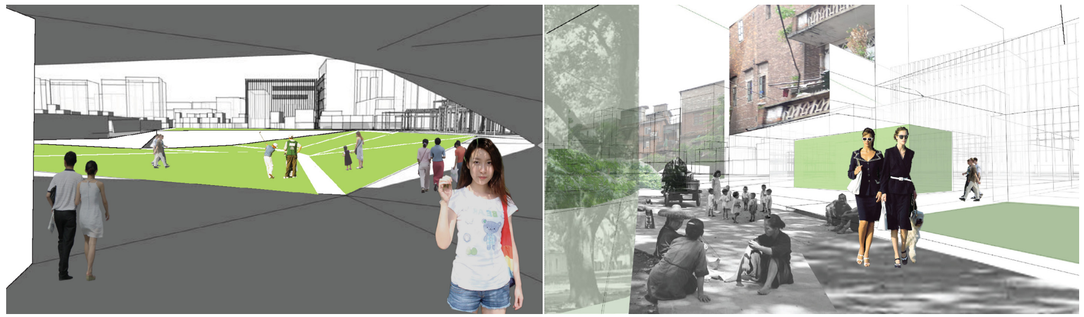
Above: wireframe of 10-unit case study after placeement of circulation, program and utility cores. Units at this point are clustered and structural walls are placed provisionally around prefab modules.

Opposite: x1, Single-family unit case study suggests possible organizational logic to be encoded in shape grammer: composit cellular forms wrapped by structural wall minimizing structural redundancies of prefabrication, programatic adjacencies, central utility core, externalized circulation follows structural walls and arranged to separate public and private routes.

CHENJIACI CULTURAL VILLAGE AND PARK REDEVELOPMENT

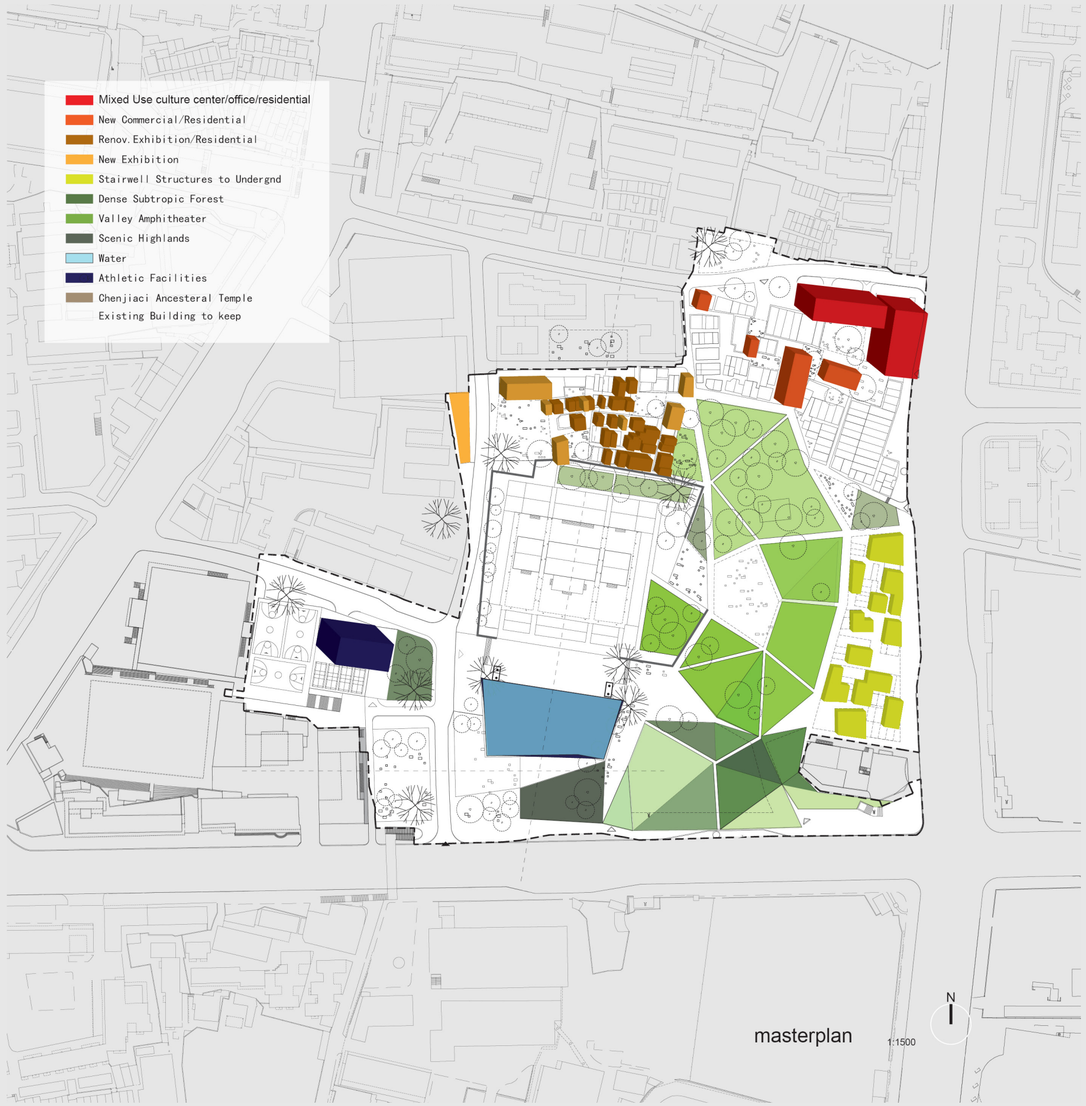
The cultural square of Chan's Ancestral Temple (CAT) is intended to be a multi-programmed tourist site that is rich both in historical architecture and civic culture. Our design proposes to divide the land around the CAT, the core of the region, into five spaces with different programs: history space, exposition space, culture space, sports space and civic activities space. Through rebuilding, revitalizing and transforming the five spaces, we aim at creating a new spatial organization that consists of one core, one museum, one stripe, one street and multiple points, restoring the historical features of the CAT to the largest extent and revitalizing this area through tourist and commercial propositions. [text by node]

Scope: Redevelopment of Cultural park & Square of Chan's Ancestral Temple, Guangzhou, China | Collabortors: Node Architecture and Urbanism, Doreen Liu (principal in charge), Jiebin, Huang, Shaoli Lin | Contribution: Concept/schematic design, masterplan for landscape/park nodes | Role: consulting architect for Node Architecture and Urbanism, in charge of landscape nodes

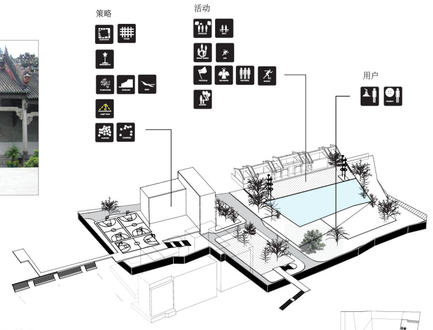


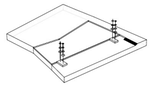
top: diagram of park zones
right: views from park
opposite: masterplan

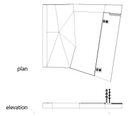
images courtesy of NODE Architecture & Urbanism



no.1







高地

升起的石⿰地⿰下形成了以⿰家祠⿰⿰的框景，并引⿰途人入内。

在地⿰上游人可以享受着⿰⿰无阻的景色，更可一⿰⿰略⿰家祠的滋味。

在⿰⿰的地方，游人可放⿰事、打羽毛球等活⿰。小路引⿰着人⿰到⿰野餐和休息的理想地点。

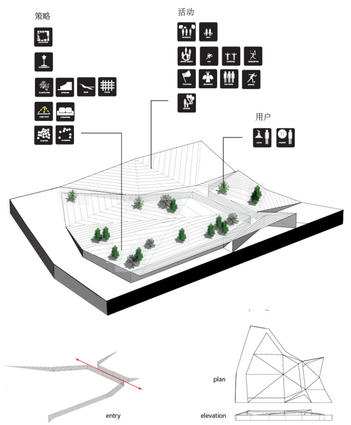
no.2

高地

升起的石⿰地⿰下形成了以⿰家祠⿰⿰的框景，并引⿰途人入内。

在地⿰上游人可以享受着⿰⿰无阻的景色，更可一⿰⿰略⿰家祠的滋味。

在⿰⿰的地方，游人可放⿰事、打羽毛球等活⿰。小路引⿰着人⿰到⿰野餐和休息的理想地点。

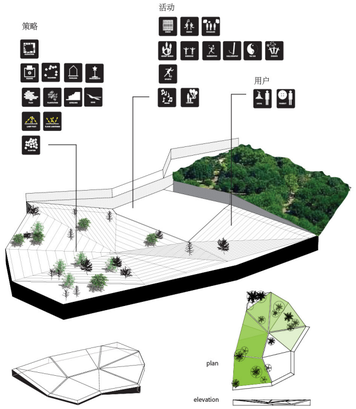


no.3

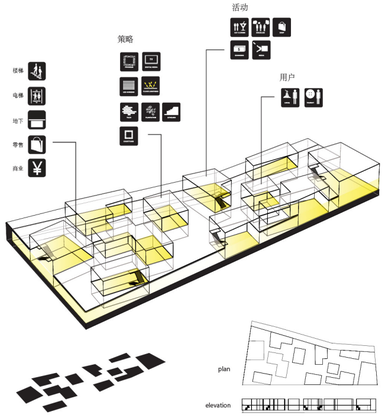
自然剧场

坡地自然景观形成了一个以陈家祠为中心的大型广场。广场可以举办大型活动，集体运动和游戏，是一个理想的社交场所。

陈家祠顿变成一个美丽的背景衬托着这个社交场地。围墙借用园林中的框景手法，不光可让祠外行人窥视陈家祠的，也可让祠中游人观看广场的情景



no.4



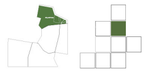
广场

通透的结构穿过地块形成地面与地下的连接。

晚上，地底下的照明使这结构变成一个景观中的发光体，在日间自然光投进临时的地下购物中心。 在街道上结构阻挡途人的视线，惹起途人的好奇心。

绿化屋顶把结构融入景观中。历史的痕迹使这些结构与临近的文化区内的建筑物形成关系。

no.5

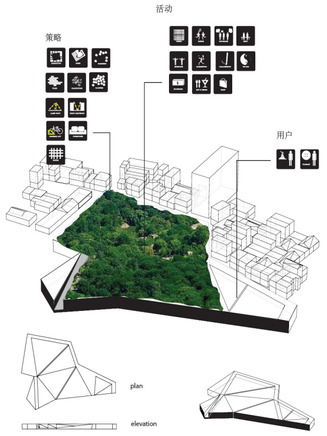


现有森林

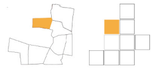
树冠提供最佳的遮阴处。

这里是退休人员聚集、交际、运动和下棋的好地方。

弯曲的小路穿过丛林形成自然步行径并穿插于各个休闲空间中



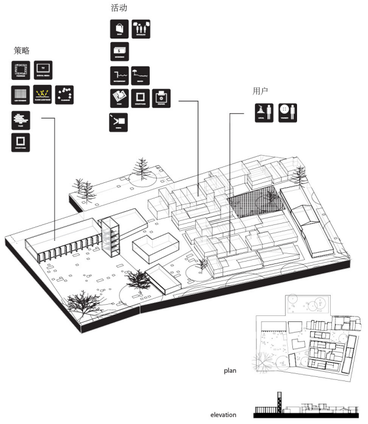
no.6



自然剧场

坡地自然景观形成了一个以陈家祠为中心的大型广场。广场可以举办大型活动，集体运动和游戏，是一个理想的社交场所。

陈家祠顿变成一个美丽的背景衬托着这个社交场地。围墙借用园林中的框景手法，不光可让祠外行人窥视陈家祠的，也可让祠中游人观看广场的情景



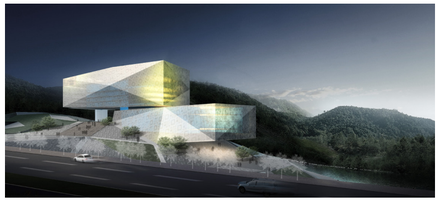
Catalog of space and activity. Each zone is given a thematic designation and program and activities are mapped onto it.

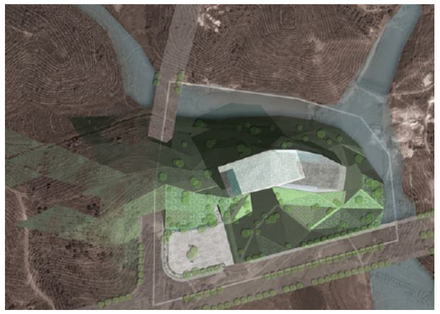
GUANGXI MUSEUM OF FINE ARTS

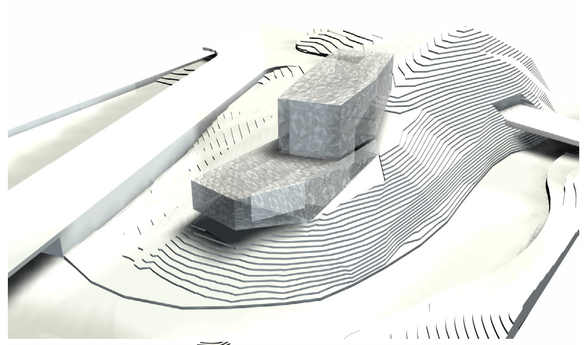
36,000 m2 museum of cultural art comprising 18,000 m2 of Exhibition Space and just under 3000 m2 of interior public space. Remaining area split between public facilties, office space and storage. 35% of total volume was buring and 65% above ground. The Museum is designed to integrate with the terrain, i.e. to embed part of the museum functions into the mountain, then follow the moutain slope to organize the circulation from the waterfront to the mountain top, thus create the impressive cantilevered building feature and form the corresponding axis at the head of the “dragon vein”.

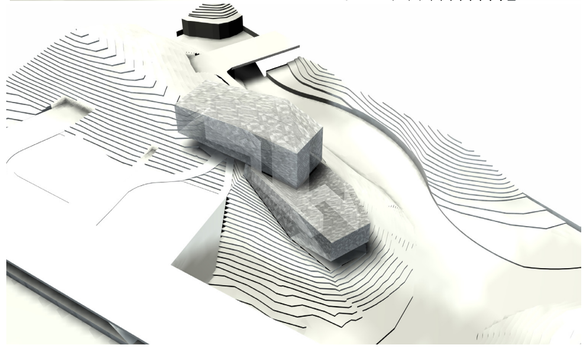
I participated in development of concept, schematic design and initial massing, In charge of design and development of building Envelope and landscape as single continous triangulated skin. Perforation pattern derived from traditional rug patterns and constructed with 2.5 x 4m perforated aluminum panels. Quilted effect from triangulation also recalls traditional rug pattern. pattern tiles are approx. 8x8m and consist of 8 perforated aluminum panels, 2 of which are unique. 3 unique pattern tiles of varying transparencies then yield a total of 6 unique aluminum panels.

Project Scope: Design / Analysis of Building envelope | Collabortors: NODE Architecture and Urbanism | Roles: Design Consultant for NODE | Contributions: Concept/schematic design, massing, incharge of landscape and envelop design



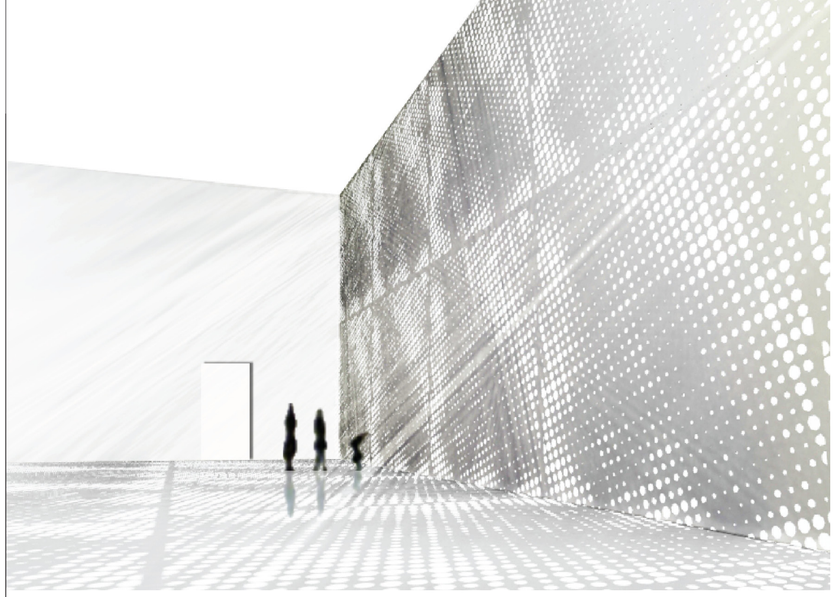


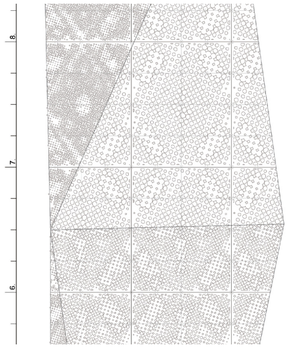


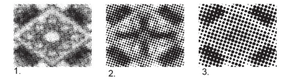


right: bird’s eye views of massing and skin
opposite: skin unfold with panel and pattern mappings.

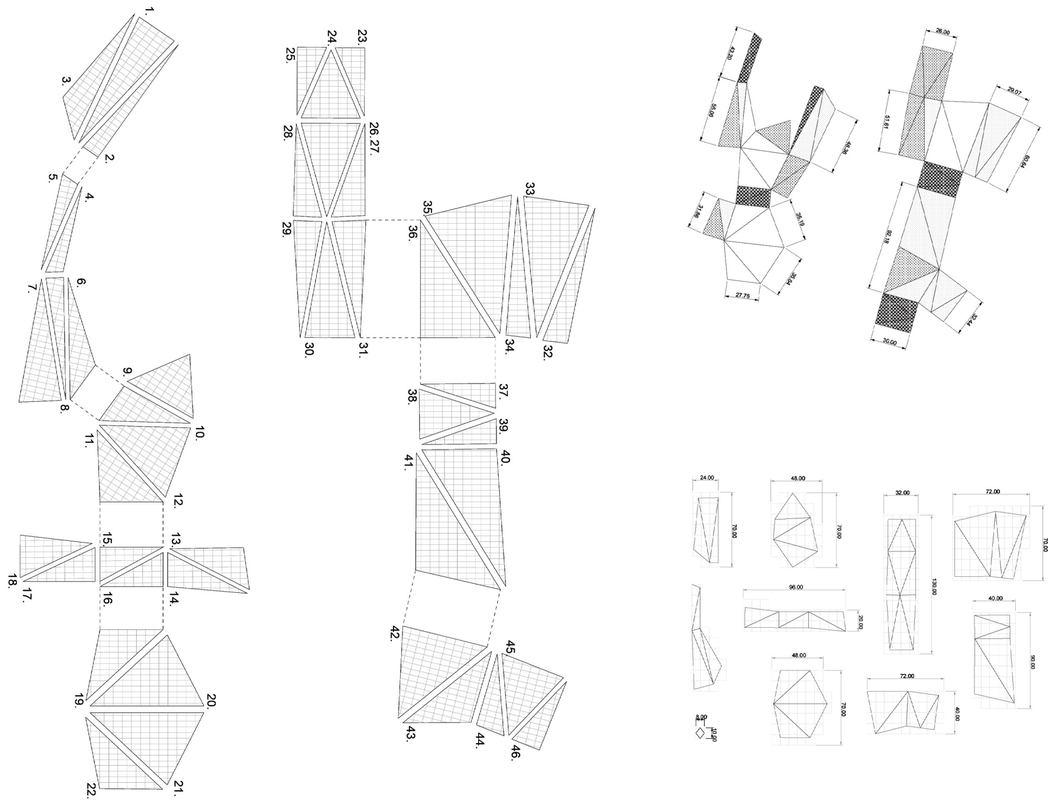
images courtesy of NODE Architecture & Urbanism







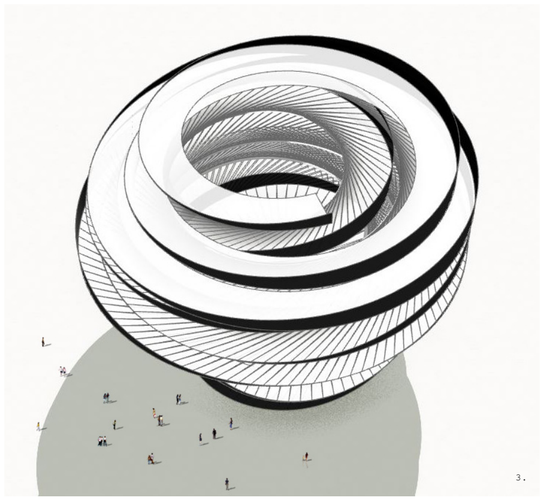
三个独特的穿孔模块来自于纺织品上的图案。 这些铝模块从表述性、文化性和标志性平铺有序地附在建筑表面上。 这三个模块，每块需要由8块2x4米的面板组成。 每个图案主要由两种独一无二的图案组成，三个穿孔的模板就共有六种不同的图案组成。



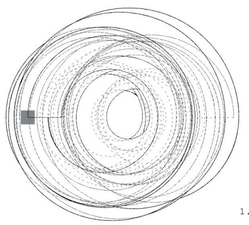
SPIRALLING: PARKING GARAGE

Custom spiralling tool developed for parking garage Form-finding excercise with grasshopper visual programming rhino plugin. Algorithm based on simple Heuristic spiraling algorithm by Aranda Lasch: Specify radius. Then, At each increment, Rotate point around center by x Degrees and step in z direction by distance Y. Connect Ordered set of points with arc segments or spline. Road-like surface is Defined by offseting each point radially producing a second set of points corresponding to inner edge of the spiralling surface. Agerage radius is constrained as symbolic function of height. Randomness is introduced by offseting points along radius at each increment. Vertical wall surfaces also follow the leading edge, the height of which varies as a function of sine. Other exposed parameters include initial radius, vertical step, rotation increment, and randomness, as well as material thickness. Structural intellegence was not requested. model was used primarily to explore design space and comminicate to client possibilities of spralling form.

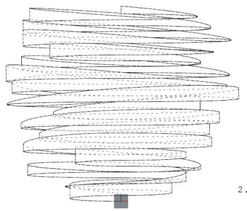
Scope: Custom parametric Tool for parking garage study | Collabortors: OPEN Archtiecture, LI HU | Contribution: Consultant for OPEN Architecture, Computational designer, form-finding



3.



1.

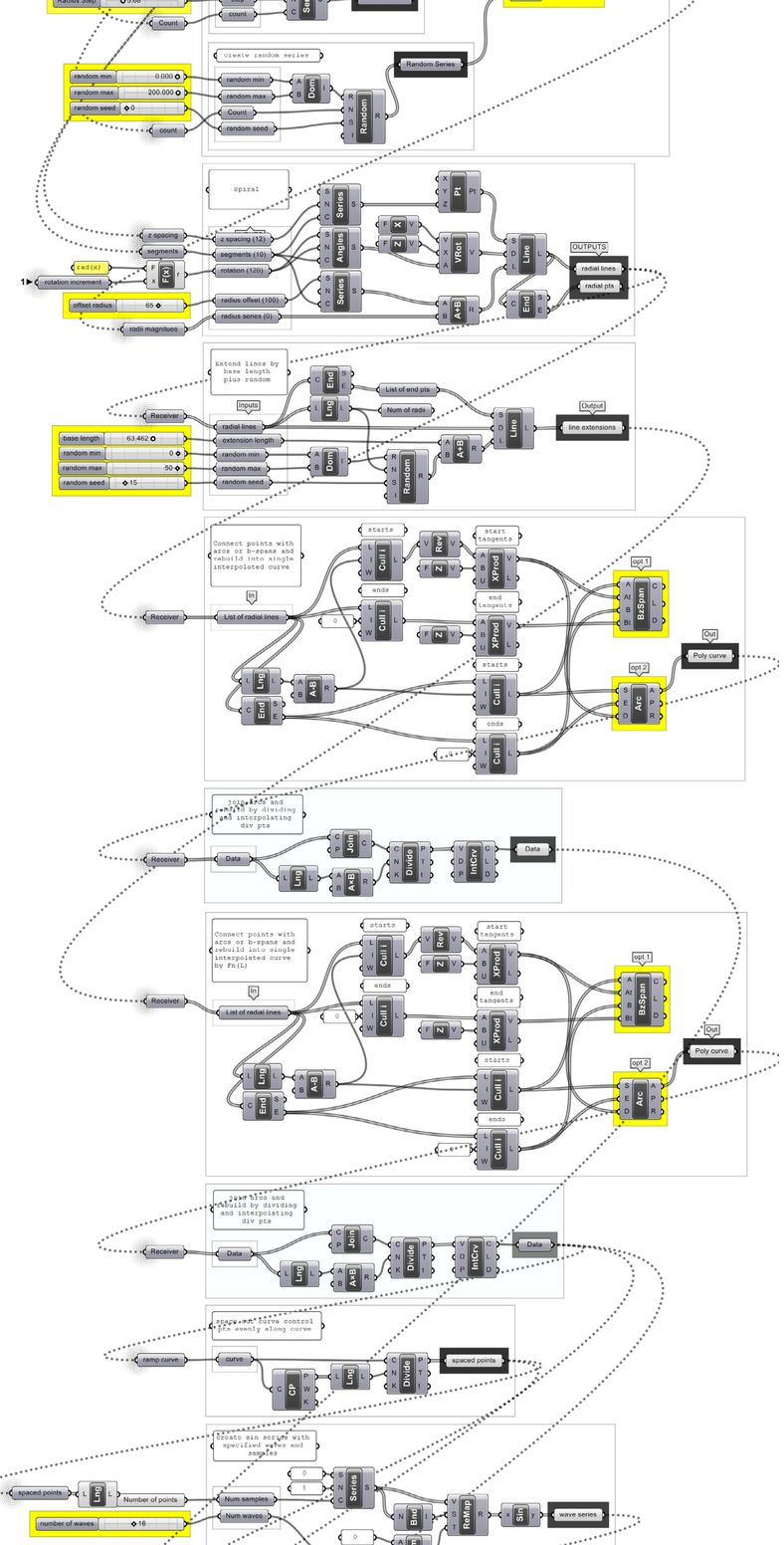


2.

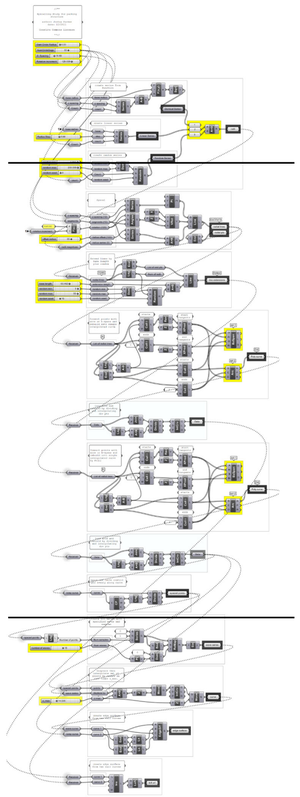
4.

1-2. Above: plan and elevation of test spiral
3. Concept rendering of test form
4. Compression and wave frequency of edge surface ensures high degree of structural overlapping.

opposite: grasshopper screenshot illustrating inner and outer asynchronous spiralling.



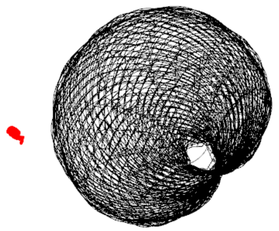
Left: Image of cropped grasshopper component graph
Below: Image of full definition

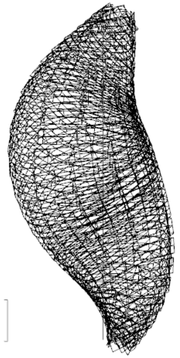


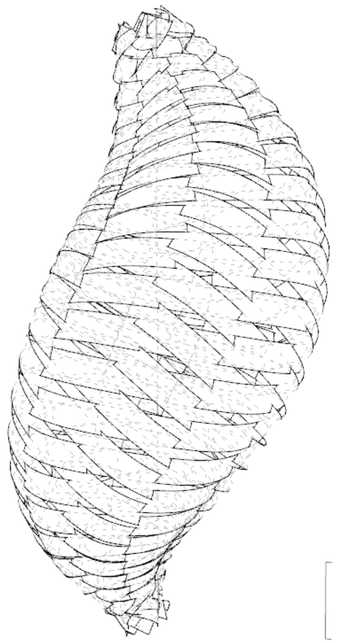
WEAVING: URBAN FURNITURE / SHADING

Weaving component Designed/implemented with grasshopper, visual programming rhino plugin for urban design furniture charrette in collaboration with Kristiana Leniart (AECOM), and later generalized as multi-scalar generic weaving tool. Custom Algorithm builds on spiralling algorithm used in early study. Defines x number of spirals starting every 360/x degrees in both directions. spirals offset radially at points of overlap to avoid intersection. This approximates a basic basket weave pretty well, however fails to orient strip surface normals perpendicular to base form. This functionality was added and Algorithm was further developed to allow for symbolic and geometric input to define base form.

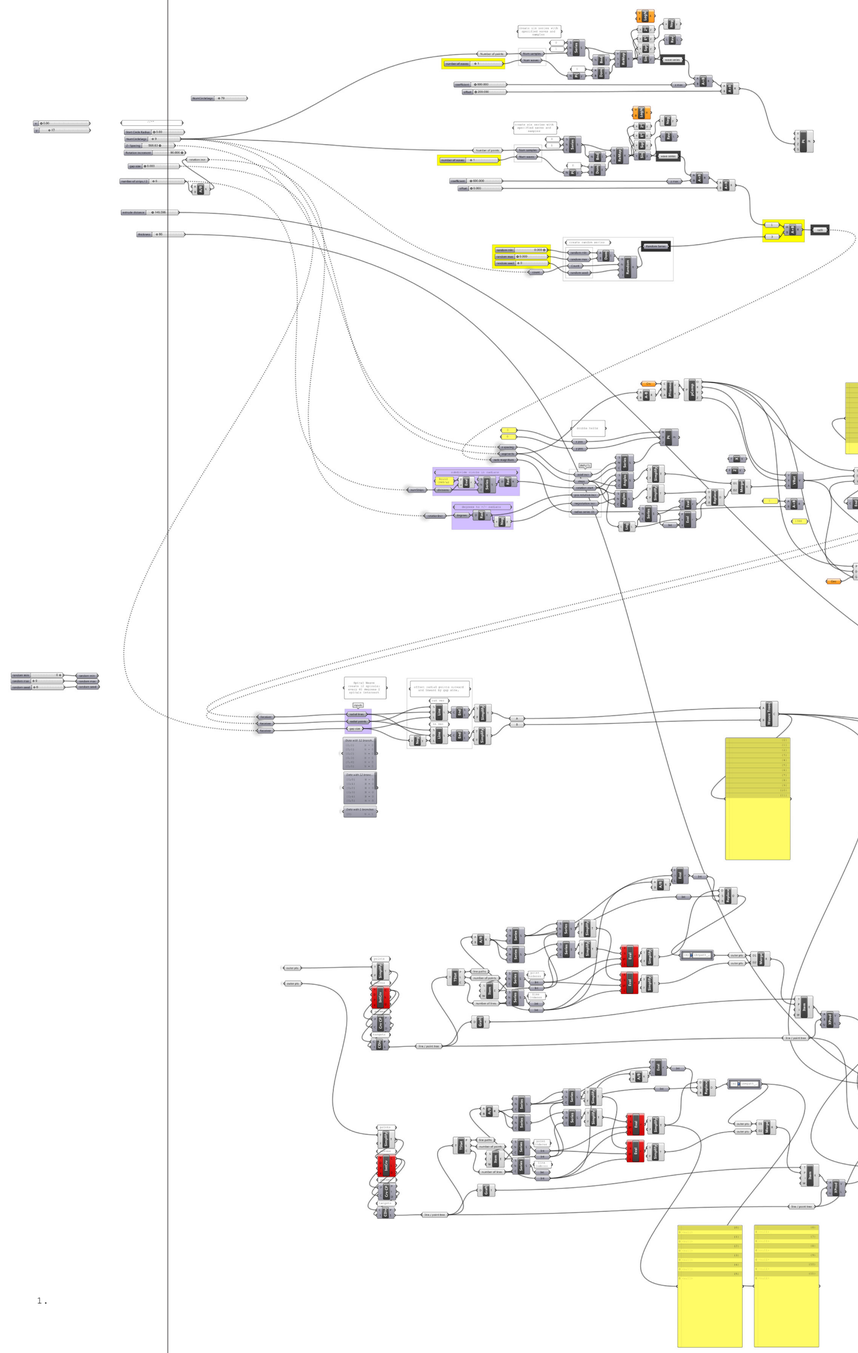
Scope: Custom parametric Tool | Collabortors: Kristiana Leniart | Contribution: Computational design







1. Test form following s-curve with sinusoidal expansion and interlocking weave
2. Test form for shading device with non-overlapping strips



1.

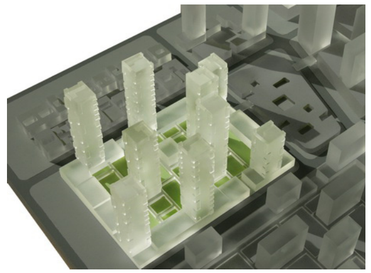
2.

SAMPLING: JAZZ TOWERS, BAOTOU

Jazz Towers is one of the many urban typologies within the BTCBD City of Plazas. The project responds to issues of gated development by placing a horizontal commercial perimeter block to surround the ten vertical residential towers, which allows both protection and accessibility. An open plaza harnessed by the horizontal band and on the roof of ground floor parking structure is inserted in the center. It provides public space for the residents on a higher ‘ground’, as well as linking to the social network of open spaces of the master plan.

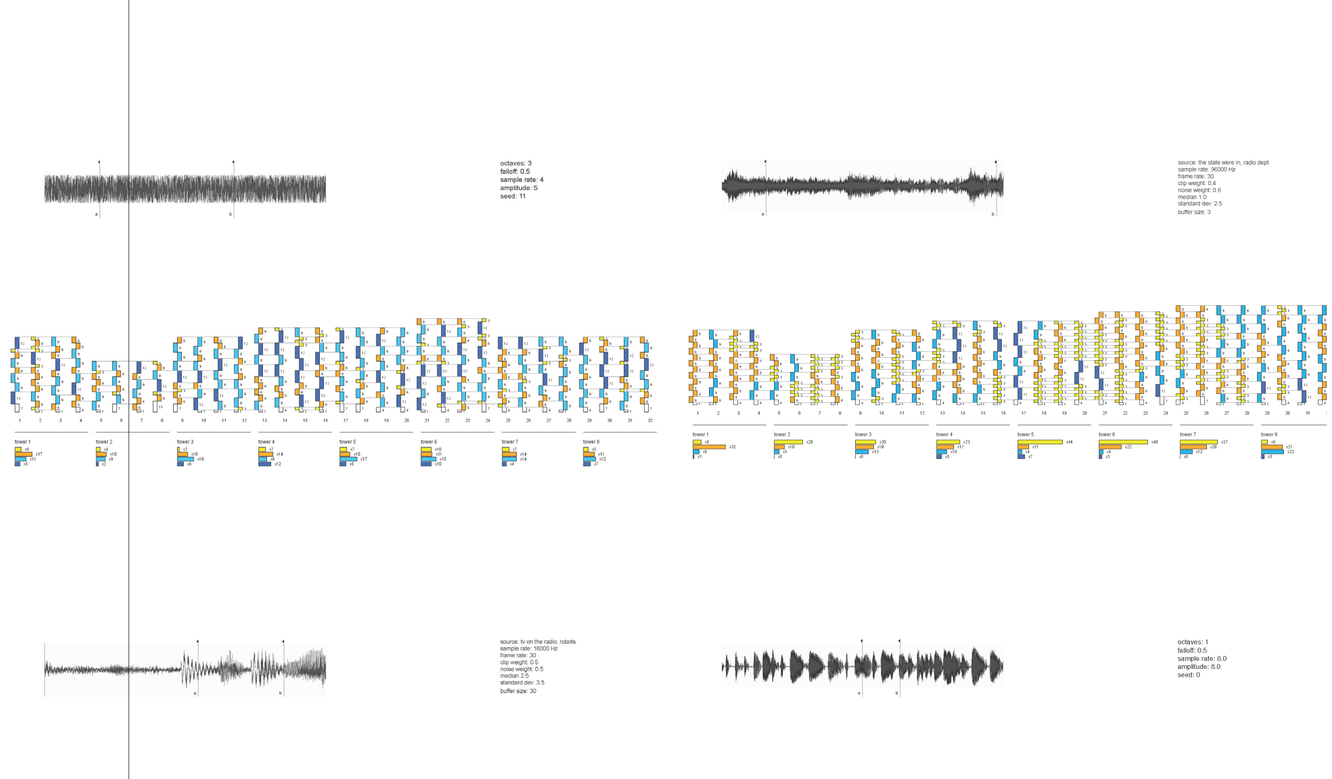
The towers, measured 18m x 18m with central cores, are proposed with open plan for flexible unit layouts. The corners of the towers cantilever out 1.5m to provide each space with its own unique perspective increasing connection from the interior space to the city. Inspired by the spontaneous yet systematic structure of Jazz music, the cantilever view-boxes are composed on the elevations of the towers through parametric calculation of Jazz musical sound wave. The musical rhythmic façade allows spacio-temporal movement by placing strategic architectonic elements in space to influence human moving and visual patterns. When one meandering around the “Towers of Interlocking Views”, it evokes a sequence of harmonic spaces as well as the dynamic and free energy of urban life.

Scope: residential towers | Collabortors: OPEN Archtiecture, LI HU (Principal in charge) | Role: Consultant for OPEN | Contribution: Custom design tool



1. building elevations
2. architectural model

text and images courtesy of OPEN Architecture
more at http://openarch.com/task/1456

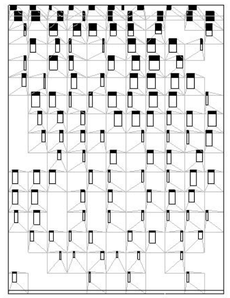


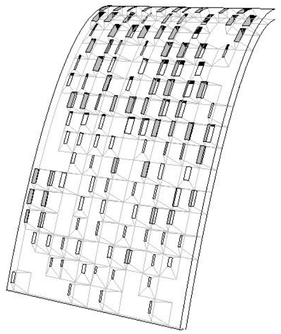
SOURCE CODE SAMPLE, UPLIFT HANGER

Code written for Interactive paneling tool developed for use in design of structurally integrated prefabricated concrete panel system for design competition of airship hanger in anhui, china for competition submission completed in collaboration with OPEN Architecture and the Chinese Academy of Building Research (CABR)

Uses the following processing libraries:

controlP5 - gui toolkit
- http://www.sojamo.de/libraries/controlP5/
Proscene - 3d scene library
- http://code.google.com/p/proscene/
0bjloader - 3d object loader
- http://code.google.com/p/saitoobjloader/





right: perforation pattern on skin segment.


```
/**************************************************

Aper[na]tures
Author: Joshua Parker
Date: 2011

Interactive paneling tool developed for use in
design of structurally integrated prefabricated
concrete panel system for design competition
of airship hanger in anhui, china for competition
submission completed in collaboration with OPEN
Architecture and the Chinese Academy of Building
Research (CABR)

Implementated with Simple Model-View-Controller
(MVC) pattern:

Gui - is a wrapper for controlP5, which supplies
  2d user interface toolkit
MyControlListener - recieves ui events, ids and
  routes them to controller
Controller - contains methods for updating data
  model & controlling view(s)*
View - contains the scene(empty space and camera),
  imported 3d base model, and data model defining
  panel system. View calls draws to screen via
  viewState interface, though only one viewState
  is implemented: View3d, a basic 3d view.
Hanger - data model defining panel system, composed
  of subclass skin, which is composed of panels.
  Panels are basically two sets of parameters: one
  which defines the panel's size and position in
  space, and another that defines the size and
  relative position of a single rectangular void
  in the panel. Panel also contains data from ecotect
  about the amount of direct and diffuse radiation
  each panel recieves.

Uses the following processing libraries:

controlP5 - gui toolkit
- http://www.sojamo.de/libraries/controlP5/
Proscene - 3d scene library
- http://code.google.com/p/proscene/
0bjloader - 3d object loader
- http://code.google.com/p/saitoobjloader/

Inheritted License: GPL, V3

Fourth Natures Lab | www.fnl.com

**************************************************/
import processing.core.*;
import controlP5.*;
import processing.opengl.*;
import controller.Listener;
import controller.Controller;
import model.Hanger;
import view.*;
import gui.*;

public class Apernatures extends PApplet{

        private static final long serialVersionUID = 1L;

        //CONSTANTS
        int U = 37; //number of modules in U direction
        int V = 42; //number of modules in V direction
        float D = 10f; //panel lengh/width dimension, D
(meters)
        float T = .5f; //thickness, T of frame (meters)
        float SUBDIVS = 20f; //subdivide panel for snapping

        //path to tabular data from ecotect analyses
        String PATH = "data/ecotectData.txt";

        //objects
        Listener myListener;
        Controller myController;
        Hanger myhanger;
        View myview;
        Gui mygui;

        public void setup(){
          size(1280, 1020, PGraphicsOpenGL.OPENGL);
          background(0);
          noStroke();
          //smooth();

          //MODEL: Parametric Data for panel system
          float snap = (float)((D-(T*2))/SUBDIVS);
          myhanger = new Hanger(this, U, V, D, T, snap,
PATH);

          //VIEW: draws a representation of model to
screen
          myview = new View(this, myhanger);

          //GUI: needs to access view scene for drawing 2d
gui on top of 3d scene
          mygui = new Gui(this, myview);

          //CONTROLLER: updates model & views in response
to ui events routed to it
          myController = new Controller(this, myhanger,
mygui, myview);

          //LISTENER/ROUTER: recieves ui events, ids and
routes them to controller
          myListener = new Listener(this, myController);

        }

        public void draw(){
          background(50);

          //Continually redraw view and gui via controller
          myController.update();
        }

        //required to run as application
        public static void main(String args[]) {
          PApplet.main(new String[] { "--present", "Apernat-
ures" });
        }

        //send ui events to listener, so it route them to
controller
        public void controlEvent(ControlEvent theEvent)
{myListener.controlEvent(theEvent);}
        public void mousePressed(){myListener.
mousePressed(mouseX, mouseY);}
        public void keyPressed(){myListener.
keyPressed(key);}

}

/**************************************************

Controller Class

MVC Controller contains methods for updating data
model & controlling view(s)*

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

**************************************************/
package controller;

import processing.core.PApplet;
import processing.core.PVector;
import model.Hanger;
import view.View;
import gui.Gui;

public class Controller{

        //The parent PApplet
        PApplet p;

        //PROPS
        private Hanger myhanger;
        private View myview;
        private Gui mygui;

        //CONSTRUCTOR
        public Controller(PApplet parent, Hanger Myhanger,
Gui Mygui, View Myview){
                p = parent;
          myhanger = Myhanger;
          mygui = Mygui;
          myview = Myview;
        }

        // PUBLIC METHODS ////////////////////////////////
//////////////////

        //gui noise increment
        public void setNoiseInc(float v){
                myhanger.setInc(v);
        }

        //set noise seed
        public void setNoiseSeed(int v){
                myhanger.setSeed(v);
        }

        //set coefficient with index i
        public void setPanelXcf(int i, float C){
                myhanger.setXcf(i, C);
        }

        //
        public void moveCameraByTheta(float v){
          PVector v0;
          float r = 500;
          float phi = 0;
          float theta = 0;
          //r = myview.scene.camera().sceneRadius();
          //r = 500;
          v0 = myview.scene.camera().position();
          phi = (float)Math.atan(v0.y/v0.x);
          theta = PApplet.radians(v); //azimuth 0-2PI
          setCameraPosition(phi, theta, r);
        }

        //
        public void moveCameraByPhi(float v){
          PVector v0;
          float r = 500;
          float phi = 0;
          float theta = 0;
          //r = myview.scene.camera().sceneRadius();
          //r = 500;
          v0 = myview.scene.camera().position();
          theta = (float)Math.acos(v0.z/r);
          phi = PApplet.radians(v); //azimuth 0-2PI
          setCameraPosition(phi, theta, r);
        }

        //
        public void setCameraPosition(float phi, float
theta, float r){
                float x = r*(PApplet.sin(theta)*PApplet.
cos(phi));
                float y = r*(PApplet.sin(theta)*PApplet.
sin(phi));
                float z = r*PApplet.cos(theta);

                PVector v = new PVector(x,y,z);
                PVector v2 = new PVector(0,0,0);
                PVector v3 = new PVector(0,0,-1);

                myview.scene.camera().setPosition(v);
                myview.scene.camera().setOrientation(theta,
phi);
                myview.scene.camera().setUpVector(v3,true);
                myview.scene.camera().lookAt(v2);
        }

        //
        public void focusGui(){
                myview.scene.disableMouseHandling();
                myview.scene.disableKeyboardHandling();
        }

        //
        public void focusScene(){
                myview.scene.enableMouseHandling();
                myview.scene.enableKeyboardHandling();
        }
```

```
  //
  public void toggleTexture(){
if(!myview.bTexture){
  showTexture();
} else {
  hideTexture();
}
  }

  //
  public void toggleMaterial(){
          if(!myview.bMaterial) {
                  showMaterial();
          } else {
                  hideMaterial();
          }
  }

  //
  public void toggleFill(){
          if(!myview.getFill()) {
                  showFill();
          } else {
                  hideFill();
          }
  }

  //
  public void showTexture(){
          myview.model.enableTexture();
          myview.bTexture = true;
  }

  //
  public void hideTexture(){
          myview.model.disableTexture();
          myview.bTexture = false;
  }

  //
  public void showMaterial(){
          myview.model.enableMaterial();
          myview.bMaterial = true;
  }

  //
  public void hideMaterial(){
          myview.model.disableMaterial();
          myview.bMaterial = false;
  }

  //
  public void showFill(){
          myview.enableFill();
  }

  //
  public void hideFill(){
          myview.disableFill();
  }

  //
  public void update(){
          myhanger.update(); //update the model
          myview.draw(); // draw the 3d scene
          mygui.draw(); // then draw the gui on
top
  }
}
/*************************************************

Listener Class

Listener recieves ui events, parses them and routes
them to controller for handling.

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

*************************************************/
package controller;

import controlP5.ControlEvent;
import controlP5.ControlListener;
import processing.core.PApplet;

public class Listener implements ControlListener{

          //The parent PApplet
          PApplet p;

          //PROPS
          Controller myController;

          //CONSTRUCTOR
          public Listener(PApplet parent, Controller MyCon-
troller){
                  p = parent;
                  myController = MyController;
          }

          // PUBLIC METHODS /////////////////////////////////
////////////////////

          //route slider events to controller via private
handlers
          public void controlEvent(ControlEvent theEvent) {

                  switch(theEvent.getController().getPar-
ent().getId()){
                  case(1):
                          //route g1 (shortcut info)
event!
                          //empty so break
                          break;
                  case(2):
                          //route g2 (noise sliders)
event!
                          g2ControlEvent(theEvent);
                  break;
                  case(3):
                          //route g3 (effect sliders)
event!
                          g3ControlEvent(theEvent);
                  break;
                  case(4):
                          //route g4 (camera dials)
event!
                          g4ControlEvent(theEvent);
                  break;
                  }
          }

          //route mouse events to controller
          //this should be replaced with focus on hover
          public void mousePressed(int mx, int my) {
                  if (mx < 200) {
                          myController.focusGui();
                  } else {
                          myController.focusScene();
                  }
          }

          //route key press events to controller
          public void keyPressed(char key) {

                  if(key == 't') {
                          //toggle the texture listed in
.mtl file
                          myController.toggleTexture();
                  }
                  else if(key == 'm') {
                          //toggle the material listed
in .mtl file
                          myController.toggleMaterial();
                  }
                  else if(key == 'f') {
                          //toggle the fill for new
shapes
                          myController.toggleFill();
                  }
          }

          // PRIVATE METHODS /////////////////////////////////
////////////////////

          //parse group 2 (noise sliders) event!
*****************************

          private void g2ControlEvent(ControlEvent theEvent)
{

                  float v;
                  switch(theEvent.getController().getId())
{
                  case(1):
                          v = theEvent.getController().
getValue();
                  myController.setNoiseInc(v);
                  break;
                  case(2):
                          v = theEvent.getController().
getValue();
                  myController.setNoiseSeed((int)(v));
                  break;
                  }
          }

          //parse group 3 (effect sliders) event!
*****************************
          private void g3ControlEvent(ControlEvent theEvent)
{

                  float v;
                  switch(theEvent.getController().getId())
{
                  case(5):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(0,v);
                  break;
                  case(6):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(1,v);
                  break;
                  case(7):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(2,v);
                  break;
                  case(8):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(3,v);
                  break;
                  case(9):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(4,v);
                  break;
                  case(10):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(5,v);
                  break;
                  case(11):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(6,v);
                  break;
                  case(12):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(7,v);
                  break;
                  case(13):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(8,v);
                  break;
                  case(14):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(9,v);
                  break;
                  case(15):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(10,v);
                  break;
                  case(16):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(11,v);
                  break;
                  case(17):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(12,v);
                  break;
                  case(18):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(13,v);
                  break;
                  case(19):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(14,v);
                  break;
                  case(20):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(15,v);
                  break;
                  case(21):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(16,v);
                  break;
                  case(22):
                          v = theEvent.getController().
getValue();
                  myController.setPanelXcf(17,v);
                  break;

                  }
          }

          //parse group 4 (camera dials) event! ***********
*******************
          private void g4ControlEvent(ControlEvent theEvent)
{

                  float v;
                  switch(theEvent.getController().getId())
{
                  case(13):
                          v = theEvent.getController().
getValue();
                  myController.moveCameraByPhi(v);
                  break;
                  case(14):
                          v = theEvent.getController().
getValue();
                  myController.moveCameraByTheta(v);
                  break;
                  }
          }

}

/*************************************************

Gui Class

Gui is a wrapper for controlP5 library, which
provides user interface toolkit. more at:
http://www.sojamo.de/libraries/controlP5/

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

*************************************************/
package gui;

import processing.core.PApplet;
import processing.core.PFont;
import controlP5.*;
import remixlab.proscene.*;
import view.*;

public class Gui
{
          //CONSTANTS
          private static final int DISABLE_DEPTH_TEST = 4;
          private static final int ENABLE_DEPTH_TEST = -4;

          //parent PApplet
          PApplet p;

          //cp5 objs
          private ControlP5 cp5;
          private Accordion accordion;

          //composit view
          private View myview;

          //CONSTRUCTOR
          public Gui(PApplet parent, View Myview){

                  //initialize
                  p = parent;
                  cp5 = new ControlP5(p);
                  cp5.setAutoDraw(false);
                  myview = Myview;

                  //setup gui elements
                  setupP5Gui();
          }

          // PUBLIC METHODS /////////////////////////////////
////////////////////

          public void draw() {

                  //save scene (so you can draw 2d gui on
top)
                  saveState(myview.scene);

                  //draw gui
                  cp5.draw();

                  //restore scene
                  restoreState(myview.scene);
          }

          // PRIVATE METHODS /////////////////////////////////
////////////////////

          //setup p5 gui elements
          private void setupP5Gui(){

                  int BarH = 20;
                  int BgClr = p.color(50); //bg of block
                  int BarClr = p.color(50); //bar color
                  int HoverClr = 30; //bar color on hover
                  int LabelClr = p.color(255);

                  int slideH = 10;
                  int slideW = 100;
                  int slideX = 10;
                  int slideSpc = 5;
                  int slideDY = slideH+slideSpc;

                  PFont arial = p.createFont("ArialMT-30",
11);
                  PFont arial10 =
p.createFont("ArialMT-30", 10);

                  //add footer
                  cp5.addTextlabel("label")
                                  .setText("by Fourth
Natures Lab | www.fnl.com")
                                  .setPosition(2,p.
height - 20)
                                  .setFont(arial10)
                                  .setHeight(20)
                                  .setWidth(200)
                                  ;

                  // ADD GROUPS *************************
************************************************

                  // group 1 (shortcut info)
                  cp5.Group g1 = cp5.
addGroup("myGroup1")
                                  .setId(1).
setLabel("Aper[na]ture")
                                  .setBackground-
Height(80)
                                  .setBarHeight(BarH)
.setBackgroundColor(BgClr)
                                  .setColorBackground(BarClr)
                                  .setColorForeground(HoverClr)
                                  .setColorLabel(LabelClr)
                                  ;

                  // group 2 (noise sliders)
                  controlP5.Group g2 = cp5.
addGroup("myGroup2")
                                  .setId(2)
                                  .setLabel("Noise")
                                  .setBackground-
Height(40)
                                  .setBarHeight(BarH)
.setBackgroundColor(BgClr) //bg of block
.setColorBackground(BarClr) //bar color
.setColorForeground(HoverClr) //bar color on hover
.setColorLabel(LabelClr) //label text color
                                  ;

                  // group 3 (effect sliders)
                  controlP5.Group g3 = cp5.
addGroup("myGroup3")
                                  .setId(3)
                                  .setLabel("x = f()")
                                  .setBackground-
Height(300)
                                  .setBarHeight(BarH)
.setBackgroundColor(BgClr) //bg of block
.setColorBackground(BarClr) //bar color
.setColorForeground(HoverClr) //bar color on hover
.setColorLabel(LabelClr) //label text color
                                  ;

                  // group 4 (camera dials)
                  controlP5.Group g4 = cp5.
addGroup("myGroup4")
                                  .setId(4)
                                  .setLabel("view")
                                  .setBackground-
Height(200)
                                  .setBarHeight(BarH)
.setBackgroundColor(BgClr) //bg of block
.setColorBackground(BarClr) //bar color
.setColorForeground(HoverClr) //bar color on hover
.setColorLabel(LabelClr) //label text color
                                  ;

                  // ADD CONTROLLERS: sliders and dials **
************************************************

                  //add group 1 controllers

                  cp5.addTextarea("txt")
                  .moveTo(g1)
                  .setPosition(5,5)
                  .setSize(195,90)
                  .setFont(arial)
                  .setLineHeight(14)
                  .setColor(p.color(128))
                  .setColorBackground(p.color(50))
                  .setColorForeground(p.color(255,100))
                  .setText("shortcuts: \n"
                                  +"f - toggle fills
\n"
                                  +"e - toggle per-
spective \n"
                                  +"c - close controls
\n"
                                  +"o - open controls
\n");
```

```
//add group 2 controllers

cp5.addSlider("noise inc")
.setId(1)
.moveTo(g2)
.setPosition(slideX,slideDY*1)
.setSize(slideW,slideH)
.setRange(0.0f,1.0f)
.setValue(.4f);

cp5.addSlider("noise seed")
.setId(2)
.moveTo(g2)
.setPosition(slideX,slideDY*2)
.setSize(slideW,slideH)
.setRange(0,50)
.setValue(8);

//add group 3 controllers

cp5.addSlider("total radiation")
.setId(5)
.moveTo(g3)
.setPosition(slideX,slideDY*1)
.setSize(slideW,slideH)
.setRange(0,10)
.setValue(0);

cp5.addSlider("direct radiation")
.setId(6)
.moveTo(g3)
.setPosition(slideX,slideDY*2)
.setSize(slideW,slideH)
.setRange(0,10)
.setValue(0);

cp5.addSlider("diffuse radiation")
.setId(7)
.moveTo(g3)
.setPosition(slideX,slideDY*3)
.setSize(slideW,slideH)
.setRange(0,10)
.setValue(0);

cp5.addSlider("by grade")
.setId(8)
.moveTo(g3)
.setPosition(slideX,slideDY*4)
.setSize(slideW,slideH)
.setRange(0,10)
.setValue(0);

cp5.addSlider("by elevation")
.setId(9)
.moveTo(g3)
.setPosition(slideX,slideDY*5)
.setSize(slideW,slideH)
.setRange(0,10)
.setValue(0);

cp5.addSlider("noise")
.setId(10)
.moveTo(g3)
.setPosition(slideX,slideDY*6)
.setSize(slideW,slideH)
.setRange(0,20)
.setValue(0);

cp5.addSlider("x scale")
.setId(11)
.moveTo(g3)
.setPosition(slideX,slideDY*7)
.setSize(slideW,slideH)
.setRange(-1.0f,1.0f)
.setValue(1.0f)
.setNumberOfTickMarks(21)
.snapToTickMarks(true)
.showTickMarks(false);

cp5.addSlider("x offset")
.setId(12)
.moveTo(g3)
.setPosition(slideX,slideDY*8)
.setSize(slideW,slideH)
.setRange(-1,1)
.setValue(.1f)
.setNumberOfTickMarks(100)
.snapToTickMarks(true)
.showTickMarks(false);

cp5.addSlider("x trim")
.setId(13)
.moveTo(g3)
.setPosition(slideX,slideDY*9)
.setSize(slideW,slideH)
.setRange(0,3)
.setValue(0)
.setNumberOfTickMarks(11)
.snapToTickMarks(true)
.showTickMarks(false);

cp5.addSlider("w scale")
.setId(17)
.moveTo(g3)
.setPosition(slideX,slideDY*13)
.setSize(slideW,slideH)
.setRange(-3,3)
.setValue(1)
.setNumberOfTickMarks(100)
.snapToTickMarks(true)
.showTickMarks(false);

cp5.addSlider("w offset")
.setId(18)
.moveTo(g3)
.setPosition(slideX,slideDY*14)
.setSize(slideW,slideH)
.setRange(-1,1)
.setValue(0)
.setNumberOfTickMarks(100)
.snapToTickMarks(false)
.showTickMarks(false);

cp5.addSlider("y scale")
.setId(14)
.moveTo(g3)
.setPosition(slideX,slideDY*10)
.setSize(slideW,slideH)
.setRange(-3,3)
.setValue(1)
.setNumberOfTickMarks(100)
.snapToTickMarks(false)
.showTickMarks(false);

cp5.addSlider("y offset")
.setId(15)
.moveTo(g3)
.setPosition(slideX,slideDY*11)
.setSize(slideW,slideH)
.setRange(-1,1)
.setValue(.1f)
.setNumberOfTickMarks(100)
.snapToTickMarks(true)
.showTickMarks(false);

cp5.addSlider("y trim")
.setId(16)
.moveTo(g3)
.setPosition(slideX,slideDY*12)
.setSize(slideW,slideH)
.setRange(0,3)
.setValue(0)
.setNumberOfTickMarks(11)
.snapToTickMarks(true)
.showTickMarks(false);

cp5.addSlider("w scale")
.setId(20)
.moveTo(g3)
.setPosition(slideX,slideDY*16)
.setSize(slideW,slideH)
.setRange(-3,3)
.setValue(1)
.setNumberOfTickMarks(100)
.snapToTickMarks(true)
.showTickMarks(false);

cp5.addSlider("h offset")
.setId(21)
.moveTo(g3)
.setPosition(slideX,slideDY*17)
.setSize(slideW,slideH)
.setRange(-1,1)
.setValue(0)
.setNumberOfTickMarks(100)
.snapToTickMarks(true)
.showTickMarks(false);

cp5.addSlider("h trim")
.setId(22)
.moveTo(g3)
.setPosition(slideX,slideDY*18)
.setSize(slideW,slideH)
.setRange(0,3)
.setValue(0)
.setNumberOfTickMarks(11)
.snapToTickMarks(true)
.showTickMarks(false);

//add group 4 controllers

cp5.addKnob("azimuth")
.setId(13)
.moveTo(g4)
.setPosition(20,20)
.setRadius(35)
.setAngleRange(2*PApplet.PI)
.setStartAngle((float)(.5*PApplet.PI))
.setRange(0,360)
.setValue(0)
.setNumberOfTickMarks(16)
.snapToTickMarks(true)
.setColorForeground(p.color(255))
.setColorBackground(p.color(40))
.setColorActive(p.color(255,255,0))
.setDragDirection(Knob.HORIZONTAL);

cp5.addKnob("elevation")
.setId(14)
.moveTo(g4)
.setPosition(110,20)
.setRadius(35)
.setAngleRange(PApplet.PI)
.setStartAngle((float)(1.5*PApplet.PI))
.setRange(1,179)
.setValue(0)
.setNumberOfTickMarks(8)
.snapToTickMarks(true)
.setColorForeground(p.color(255))
.setColorBackground(p.color(40))
.setColorActive(p.color(255,255,0))
.setDragDirection(Knob.HORIZONTAL);


//Add Sidebar Accordion:  **************************************
accordion = cp5.addAccordion("acc")
                .setPosition(0,0)
                .setWidth(200)
                .addItem(g1)
                .addItem(g2)
                .addItem(g3)
                .addItem(g4)
                ;


//initially open all groups
accordion.open(0,1,2,3);


//allow multiple groups open at a time.
accordion.setCollapseMode(Accordion.MULTI);


//Add keyboard shortcuts **************************************
cp5.mapKeyFor(new ControlKey() {public void keyEvent() {accordion.open(0,1,2,3);}}, 'o');
cp5.mapKeyFor(new ControlKey() {public void keyEvent() {accordion.close(0,1,2,3);}}, 'c');

        }//end init()
```



```
        //function from https://forum.processing.org/topic/proscene-and-2d-drawing
        private void saveState(Scene scene) {

                // Disable depth test to draw 2d on top
                p.hint(DISABLE_DEPTH_TEST);

                //  Set processing projection and modelview matrices to draw in 2D:
                // 1. projection matrix:
                float cameraZ = ((p.height/2.0f) / PApplet.tan(PApplet.PI*60.0f/360.0f));
                scene.pg3d.perspective(
                                        PApplet.PI/3.0f,
                                        scene.camera().aspectRatio(),
                                        cameraZ/10.0f,
                                        cameraZ*10.0f);

                // 2 model view matrix
                scene.pg3d.camera();
        }

        //function from https://forum.processing.org/topic/proscene-and-2d-drawing
        private void restoreState(Scene scene) {

                // 1. Restore processing projection matrix
                switch (scene.camera().type()) {
                case PERSPECTIVE:
                        scene.pg3d.perspective(
                                                scene.camera().fieldOfView(),
                                                scene.camera().aspectRatio(),
                                                scene.camera().zNear(),
                                                scene.camera().zFar());
                        break;
                case ORTHOGRAPHIC:
                        float[] wh = scene.camera().getOrthoWidthHeight();
                        scene.pg3d.ortho(
                                                -wh[0], wh[0], -wh[1], wh[1],
                                                scene.camera().zNear(),
                                                scene.camera().zFar());
                        break;
                }
                // 2. Restore processing modelview matrix
                scene.pg3d.camera(
                                scene.camera().position().x,
                                scene.camera().position().y,
                                scene.camera().position().z,
                                scene.camera().at().x,
                                scene.camera().at().y,
                                scene.camera().at().z,
                                scene.camera().upVector().x,
                                scene.camera().upVector().y,
                                scene.camera().upVector().z);
                //Re-enble depth test
                p.hint(ENABLE_DEPTH_TEST);
        }

}
/*************************************************

Hanger Class

Hanger is data model defining panel system, composed
of subclass skin, which is composed of panels.
Panels are basically two sets of parameters: one
which defines the panel's size and position in
space, and another that defines the size and
relative position of a single rectangular void
in the panel. Panel also contains data from ecotect
about the amount of direct and diffuse radiation
each panel recieves.

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

 *************************************************/
package model;

import processing.core.PApplet;
import model.Skin;

public class Hanger{

        //The parent PApplet
        PApplet p;

        //composit skin
        public Skin myskin;

        //CONSTRUCTOR
        public Hanger(PApplet parent,
                        int U,
                        int V,
                        float D,
                        float T,
                        float snap,
                        String PATH){

                p = parent;

                //create skin
                myskin = new Skin(p, U, V, D, T, snap);

                //pass data path to skin so it can load data into panels
                myskin.loadData(PATH);

                //update skin pattern
                update();

        }//end constructor

        // PUBLIC METHODS //////////////////////////////////
        ////////////////////

        //set noise increment and update model
        public void setInc(float Inc){
                myskin.setInc(Inc);
        }

        //set noise seed and update model
        public void setSeed(int Seed){
                myskin.setSeed(Seed);
        }

        //set pattern coeffecients and update model
        public void setXcf(int i, float C){
                myskin.setXcf(i, C);
        }

        //update Hanger Model
        public void update(){
                //update skin pattern
                myskin.update();
        }

}
/*************************************************

Panel Class

Panel is subclass of Hanger data model defining panel
system, composed of subclass skin, which is composed
of panels.
Panels are basically two sets of parameters: one
which defines the panel's size and position in
space, and another that defines the size and
relative position of a single rectangular void
in the panel. Panel also contains data from ecotect
about the amount of direct and diffuse radiation
each panel recieves.

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

 *************************************************/
package model;

import processing.core.*;

public class Panel
{
        //The parent PApplet
        PApplet p;

        //panel properties
        public PVector[] coords;
        public float x,y,w,h;
//normalized
        public float px, py, pd;                //centerline of frame
        public float t;
//thickness of frame
        public float ix, iy, id;                //inside line, ie. panel
        public float hx, hy, hw, hh;    //perforations, ie. holes
        public float snap;
//panel subdivisions
        public int pClr;
//panel color

        //imported data
/*      private float avgDailyTotal;
        private float avgDailyDirect;
        private float avgDailyDiffuse;
        private float grade;
        private float elevation;*/

        //normalized parameters
        private float totalNorm;
        private float directNorm;
        private float diffuseNorm;
        private float gradeNorm;
        private float elevationNorm;
        public float noiseVal;

        //parameter weights
        private float[] xcf;

        //constructor
        Panel(PApplet parent,
                        float px,
                        float py,
                        float pd,
                        float ix,
                        float iy,
                        float id,
                        float t,
                        float snap){

                p = parent;

                //println("new panel");

                this.px = px;
                this.py = py;
                this.pd = pd;
                this.ix = ix;
                this.iy = iy;
                this.id = id;
                this.t = t;
                //hx = this.hx;
                //hy = this.hy;
                //hw = this.hw;
                //hh = this.hh;
                this.snap = snap;
```

```
                coords = new PVector[4];
                xcf = new float[18];

        }

        // PUBLIC METHODS /////////////////////////////////
////////////////////

        //load data into panel
        public void loadData(String avgDailyTotal,
                             String avgDailyDirect,
                             String avgDailyDiffuse,
                             String grade,
                             String elevation,
                             String totalNorm,
                             String directNorm,
                             String diffuseNorm,
                             String gradeNorm,
                             String elevationNorm){

/*              this.avgDailyTotal = Float.
parseFloat(avgDailyTotal);
                this.avgDailyDirect = Float.
parseFloat(avgDailyDirect);
                this.avgDailyDiffuse = Float.
parseFloat(avgDailyDiffuse);
                this.grade = Float.parseFloat(grade);
                this.elevation = Float.
parseFloat(elevation);*/

                this.totalNorm = Float.
parseFloat(totalNorm);
                this.directNorm = Float.
parseFloat(directNorm);
                this.diffuseNorm = Float.
parseFloat(diffuseNorm);
                this.gradeNorm = Float.
parseFloat(gradeNorm);
                this.elevationNorm = Float.
parseFloat(elevationNorm);

        }//end loadData()

        //
        public void setXcf(int i, float C){
                xcf[i]=C;
        }

        //find hole
        public void perforate(){

                //name coefficients
                float totalWeight = xcf[0];
                float directWeight = xcf[1];
                float diffuseWeight = xcf[2];
                float gradeWeight = xcf[3];
                float elevationWeight = xcf[4];
                float noiseWeight = xcf[5];
                float xScale = xcf[6];
                float xOff = xcf[7];
                //float xTrim = xcf[8];
                float yScale = xcf[9];
                float yOff = xcf[10];
                //float yTrim = xcf[11];
                float wScale = xcf[12];
                float wOff = xcf[13];
                //float wTrim = xcf[14];
                float hScale = xcf[15];
                float hOff = xcf[16];
                //float hTrim = xcf[17];

                //calc smth like a weighted avg of pat-
tern components
                float pattern =  (totalNorm * totalWeight
+
                                  directNorm * direct-
Weight +
                                  diffuseNorm * dif-
fuseWeight +
                                  gradeNorm  * grade-
Weight +
                                  elevationNorm * el-
evationWeight +
                                  noiseVal * noise-
Weight +
                                  1)
                                                  /(xcf[0] +xcf[1]
+xcf[2] +xcf[3] +xcf[4] +xcf[5] +1);

                //calc normalized scale effects and off-
set x, y, w, z
                //VALUE RANGE: 0.0 - 1.0
                float wv = (float)((((pattern-.5) * wS-
cale) + .5) + wOff);
                float hv = (float)((((pattern-.5) * hS-
cale) + .5) + hOff);
                float xv = (float)((((pattern-1) * xScale)
+ 0) + xOff);
                float yv = (float)((((pattern-1) * yScale)
+ 0) + yOff);

                //calculate real hole dimensions and
displacement in meters
                hw = wv * id;
                hh = hv * id;
                hx = xv * id + t;
                hy = yv * id + t;

                //snap and normalize coords to subgrid
                w = (int)(PApplet.round(hw/snap));
                h = (int)(PApplet.round(hh/snap));
                x = (int)(PApplet.round(hx/snap));
                y = (int)(PApplet.round(hy/snap));

                //convert back to absolute coords
                hw = w*snap;
                hh = h*snap;
                hx = px + x*snap;
                hy = py + y*snap;

                //load coordinate array (relative to
basegrid)
                //adjust order for vertice mapping
                coords[3] = new PVector(x,y);
                coords[2] = new PVector(x+hw, y);
                coords[1] = new PVector(x+hw, y+hh);
                coords[0] = new PVector(x, y+hh);


        }//end perforate

        //
        public void colorize(){

                //calc smth like a weighted avg of pat-
tern components
                float pattern =  (totalNorm * xcf[0] +
                                  directNorm * xcf[1]
+
                                  diffuseNorm * xcf[2]
+
                                  gradeNorm  * xcf[3]
+
                                  elevationNorm *
xcf[4] +
                                  noiseVal * xcf[5] +
                                  1)/
                                  (xcf[0] +xcf[1]
+xcf[2] +xcf[3] +xcf[4] +xcf[5] +1);

                int c1 = p.color(50, 50, 50);
                int c2 = p.color(200, 255, 200);
                float v = pattern * 255;

                float deltaR = p.red(c2)-p.red(c1);
                float deltaG = p.green(c2)-p.green(c1);
                float deltaB = p.blue(c2)-p.blue(c1);

                int c = p.color(
(p.red(c1)+(v)*(deltaR/255)),
(p.green(c1)+(v)*(deltaG/255)),
(p.blue(c1)+(v)*(deltaB/255))
                                  );
                pClr = c;
        }

}//end class
/************************************************
```

Skin Class

Skin is subclass of Hanger data model defining panel system, composed of subclass skin, which is composed of panels.
Panels are basically two sets of parameters: one which defines the panel's size and position in space, and another that defines the size and relative position of a single rectangular void in the panel. Panel also contains data from ecotect about the amount of direct and diffuse radiation each panel recieves.

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

```
************************************************/
package model;

import processing.core.PApplet;
import model.Panel;

public class Skin
{
        //The parent PApplet
        PApplet p;

        //registration pt
        public float x, y;

        //composit panels
        public Panel[] panels;

        //skin properties
        int u;                          //number of modules
in U direction
        int v;                          //number of modules
in V direction
        float dim;                      //panel lengh/width
dimension, D (meters)
        float thick;        //thickness, T of frame (me-
ters)

        //noise vars
        float inc;
        int seed;

        //CONSTRUCTOR
        Skin(PApplet parent,
                        int U,
                        int V,
                        float D,
                        float T,
                        float snap){

                p = parent;

                //noise vars
                inc = 0.4f;
                seed = 8;

                this.u = U;
                this.v = V;
                this.dim = D;
                this.thick = T;

                //initialize ld array for u x v panels
                panels = new Panel[u*v];

                //create u x v panels in ld array
                int n = 0;
                for(int i=0; i<v; i++) {
                        for(int j=0; j<u; j++) {

                                //calc panel dimen-
sion and reg pts
                                float pd = D;
                                float px = j*D;
                                float py = i*D;

                                //calc inner frame
dimension and reg pts
                                float id = pd-(T*2);
                                float ix = px+T;
                                float iy = py+T;

                                //frame thickness
                                float thick = T;

                                //create panel
                                panels[n] = new
Panel(
p,
px,
py,
pd,
ix,
iy,
id,
thick,
snap);
                                //incr counter
                                n++;
                        }
                }

                //generate initial pattern
                update();

        }//end Skin()

        // PUBLIC METHODS /////////////////////////////////
////////////////////

        //load data
        public void loadData(String PATH){

                //holders for rows and items in a row
respectively
                String loadedStrings[];
                String splitString[];

                //load data into an array of rows
                loadedStrings = p.loadStrings(PATH);

                int n = 0;
                for(int i=0; i<v; i++) {
                        //get data for first row, i+1
b/c first row of text file
                        //is column headers
                        splitString = PApplet.
split(loadedStrings[i+1], '\t');

                        for(int j=0; j<u; j++) {
                                //load up panel
                                panels[n].loadData(
splitString[2],
splitString[5],
splitString[8],
splitString[10],
splitString[12],
splitString[3],
splitString[6],
splitString[9],
splitString[11],
splitString[13]);
                                n++;
                        }
                }
        }//end loadData()

        //set a coefficient
        public void setXcf(int id, float C){
                int n = 0;
                for(int i=0; i<v; i++) {
                        for(int j=0; j<u; j++) {
                                panels[n].setXcf(id,
C);
                                n++;
                        }
                }
        }

        //set noise increment
        public void setInc(float Inc){
                inc = Inc;
        }

        //set noise seed
        public void setSeed(int Seed){
                seed = Seed;
        }

        //update pattern
        public void update(){
                //generate noise pattern
                generateNoise();
                //assign colors to panels
                colorize();
                //generate perforation from ecotect data
and noise
                perforate();
        }

        //generate noise
        public void generateNoise(){

                //noise constants
                float vx = 0.0f;
                float vy = 0.0f;
                //float inc = 0.4;
                //float k = 0;
                p.noiseSeed(seed);

                int n = 0;
                for(int i=0; i<v; i++) {
                        for(int j=0; j<u; j++) {
                                //calc noise
                                panels[n].noiseVal =
p.noise(vx, vy);
                                //increment panel
                                n++;
                                //increment noise x
param
                                vx = vx + inc;
                        }
                        //increment noise y param
                        vy = vy + inc;
                }
        }//end generateNoise()

        //find holes
        public void perforate(){
                int n = 0;
                for(int i=0; i<v; i++) {
                        for(int j=0; j<u; j++) {
                                panels[n].perfo-
rate();
                                n++;
                        }
                }
        }//end perforate

        //find holes
        public void colorize(){
                int n = 0;
                for(int i=0; i<v; i++) {
                        for(int j=0; j<u; j++) {
                                panels[n].color-
ize();
                                n++;
                        }
                }
        }//end perforate

}//end class
/************************************************
```

View Class

View - contains the scene(empty space and camera), imported 3d base model, and data model defining panel system. View calls draws to screen via viewState interface, though only one viewState is implemented: View3d, a basic 3d view.

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

```
************************************************/
package view;

import processing.core.PApplet;
import remixlab.proscene.*;
import saito.objloader.*;
import model.Hanger;
import view.View3d;
import view.ViewState;

public class View{

        //parent PApplet
        PApplet p;

        //composit objs
        public Scene scene;
        public OBJModel model;
        public Hanger myhanger;

        //OBJModel flags - not used
        public boolean bTexture = true;
        public boolean bStroke = false;
        public boolean bMaterial = true;

        //states
        //private ViewState myView3d;
        //private ViewState myViewUnroll;

        public ViewState viewState;

        public View(PApplet parent, Hanger Myhanger){
                p = parent;

                //3d scene
                this.scene = new Scene(parent);
                this.scene.setAxisIsDrawn(false);
                this.scene.setGridIsDrawn(false);
                this.scene.enableFrustumEquationsUp-
date();
                //imported 3d model; no need to use mtls
or specify shapemode b/c
                //not drawing model, but just using the
face vertices and normals
                this.model = new OBJModel(p, "bs10.
obj");
                //this.model = new OBJModel(p, "bs10.
obj", "relative", LINES);
                //this.model.shapeMode(LINES);
                //this.model.disableMaterial();
                //this.model.enableDebug();
                this.model.scale(1);
                this.myhanger = Myhanger;

                //view state
                setViewState(new View3d(p, this));

        }

        // PUBLIC METHODS /////////////////////////////////
```

```
////////////////////

	//set view state
	public void setViewState(ViewState newState){
		this.viewState = newState;
	}

	//get current fill
	public boolean getFill(){
		return this.viewState.getFill();
	}

	//turn off strokes, turn on surface fill
	public void enableFill(){
		this.viewState.enableFill();
	}

	//turn off surface fill, turn on strokes
	public void disableFill(){
		this.viewState.disableFill();
	}

	//draw the view
	public void draw() {
		this.viewState.draw();
	}
}
/************************************************

View3d Class

View3d implements a basic 3d view

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

************************************************/
package view;

import processing.core.*;
import saito.objloader.*;
import model.*;

public class View3d implements ViewState{

	//The parent PApplet
	PApplet p;

	//composit view
	public View myview;

	//flags
	boolean bFill = false;

	//CONSTRUCTOR
	View3d(PApplet parent, View Myview){
		p = parent;
		this.myview = Myview;
	}

	// PUBLIC METHODS /////////////////////////////////
////////////////////

	//turn off strokes, turn on surface fill
	public void enableFill(){
		bFill = true;
	}

	//turn off surface fill, turn on strokes
	public void disableFill(){
		bFill = false;
	}

	//get current fill
	public boolean getFill(){
		return bFill;
	}

	//draw the 3d view
	public void draw() {

		//setup lighting
		p.directionalLight(126, 126, 126, 0, 0,
-1);
		//p.ambientLight(102, 102, 102);
		p.lights();
		p.stroke(10,10,10);

		//set camera position and target
/*		PVector v = new PVector(0,0,-1);
		PVector v2 = new PVector(0,0,0);
		myview.scene.camera().
setUpVector(v,true);
		myview.scene.camera().lookAt(v2);*/

		//just want vertices so do not draw the
3d model!!! this will slow sketch way down
		//myview.model.draw();

		//load faces in array
		Segment segment = myview.model.getSeg-
ment(0);
		Face[] faces = segment.getFaces();

		//for each face...
		for (int i = faces.length-1; i >= 0;
i--) {

			//get face and corresponding
panel
			Face nextFace = faces[i];
			Panel nextPanel = myview.my-
hanger.myskin.panels[i];

			//get vertices and normals of
face and relative coordinates
			//of panel opening
			PVector[] vs = nextFace.get-
Vertices();
			PVector[] ns = nextFace.get-
Normals();
			PVector[] cs = nextPanel.co-
ords;

			//get panel color
			int clr = nextPanel.pClr;

			//map panel space coordinates
to world space coordinates.
			PVector[] pts = mapFace(vs,
ns, cs);

			//stroke normal
			//line(vs[0].x, vs[0].y,
vs[0].z, vs[0].x+ns[0].x, vs[0].y+ns[0].y, vs[0].z+ns[0].z);

			//set fill and stroke
			if(bFill){
				p.fill(clr);
				p.noStroke();
			}else{
				p.noFill();
				p.stroke(10, 10,
10);
			}

			//draw four quad polygons
			p.beginShape();
			p.vertex(vs[0].x, vs[0].y,
vs[0].z);
			p.vertex(vs[1].x, vs[1].y,
vs[1].z);
			p.vertex(pts[1].x, pts[1].y,
pts[1].z);
			p.vertex(pts[0].x, pts[0].y,
pts[0].z);
			p.endShape();

			p.beginShape();
			p.vertex(vs[1].x, vs[1].y,
vs[1].z);
			p.vertex(vs[2].x, vs[2].y,
vs[2].z);
			p.vertex(pts[2].x, pts[2].y,
pts[2].z);
			p.vertex(pts[1].x, pts[1].y,
pts[1].z);
			p.endShape();

			p.beginShape();
			p.vertex(vs[2].x, vs[2].y,
vs[2].z);
			p.vertex(vs[3].x, vs[3].y,
vs[3].z);
			p.vertex(pts[3].x, pts[3].y,
pts[3].z);
			p.vertex(pts[2].x, pts[2].y,
pts[2].z);
			p.endShape();

			p.beginShape();
			p.vertex(vs[3].x, vs[3].y,
vs[3].z);
			p.vertex(vs[0].x, vs[0].y,
vs[0].z);
			p.vertex(pts[0].x, pts[0].y,
pts[0].z);
			p.vertex(pts[3].x, pts[3].y,
pts[3].z);
			p.endShape();

		}//end for loop

	}//end draw

	// PRIVATE METHODS /////////////////////////////////
////////////////////

	//find hole vertices on face from a face
vertices(vs),
	//face normals(ns), and face-space coordinates(cs)
	private PVector[] mapFace(PVector[] vs, PVector[]
n, PVector[] cs){

		PVector[] pts = new PVector[0];

		//for each corner, map 2d coord to 3d pt
		for (int i = vs.length-1; i >= 0; i--) {
			PVector pt = mapPt(vs[0],
n[i], cs[i]);
			pts = (PVector[])PApplet.
append(pts, pt);
		}
		return pts;
	}

	//find singe 3d pt from a vertice(v), normal(n),
	//and 2d relative coordinate(c)
	private PVector mapPt(PVector v, PVector n, PVec-
tor c){

		//get angle of normal
		float a = (float)Math.atan(n.z/n.y);
		a = PApplet.radians(90-PApplet.
abs(PApplet.degrees(a)));

		float x,y,z;

		if(n.y < 0){
			x = v.x + c.x;
			y = v.y - c.y*PApplet.cos(a);
			z = v.z - c.y*PApplet.sin(a);
		}else{
			x = v.x + c.x;
			y = v.y - c.y*PApplet.cos(a);
			z = v.z + c.y*PApplet.sin(a);
		}

		PVector pt3d = new PVector(x, y, z);

		return pt3d;
	}

}
/************************************************

ViewState Class

ViewState is an interface for various views

Part of:

Aper[na]tures
Author: Joshua Parker
Date: 2011

Inheritted License: GPL, V3

************************************************/
package view;

public interface ViewState{

	public void enableFill();
	public void disableFill();
	public boolean getFill();
	public void draw();
}
```