04 ARCHITECTURE

VERTICAL CITY

Ever since the beginning of the history, the communities of human have been growing.

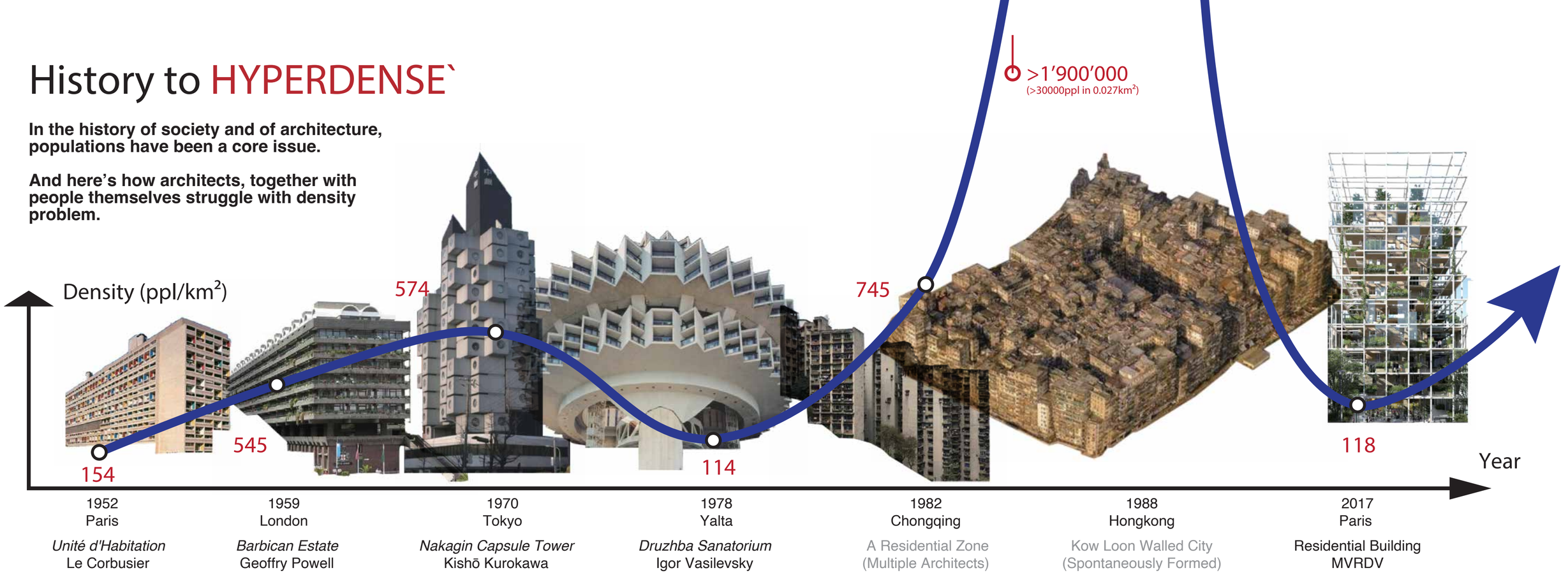
In the era of population explosion, we'll be witnessing the emergence of hyperdense cities.

This is a project of future residential community aimed at accommodating as many as possible, at the same time provide a city space to live in.

History to HYPERDENSE`

In the history of society and of architecture, populations have been a core issue.

And here's how architects, together with people themselves struggle with density problem.



"Social Housing" --
To accomodate the baby boom and industrialized production...

"Brutalism"
Human gets used to environment. Formfollows function.

"Metabolism"
Buildings are like creatures, inhales and grows.

"Residential Experiment"
How much space do a man need to survive?

"Vertical Community"
Visions and flows through heights

We set 3 principles as to imagine the future city:

1. Indoor and outdoor within the building.

2. Mixed usage space

3. Interconnected units.

Trilogy to Vertical City

Step1: Site Analyze

Our experiment site locates in the center of ***Nanjing, Jiangsu, China***. As a high rise rensidential complex, it is now facing an ***overcrowded*** situation.
Facing two major roads, it demands much about ***transportations***. All conditions below bring us great challenges and opportunities.

We're going design a rebuild project.

Site View

Original Interior

Conditions

Density: 842 ppl/km²;
Temperature: -1~33C°;

Building Area: 50000 m²;
Site Area: 19000 m²;
Main Entrance: East;

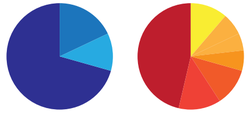
Legislated regulations:
>2h of sunshine in winter solstice;
<81m & >27m building height;
<12m from door to exit stair;
<100m circumference;

Basic factors:
Number of bedrooms;
Sunshine & Ventilation;
Urban texture;

Step2: Questionnaire

We focused on the demands of the residence. The community is largely relying on the ***commuters***, however, they make the minority.
People go in and out in nearly 24 hours in the day, for different purpose; Parking spaces severely crush pedastrian; The southward sunshine attracts everyone.

Family Composition

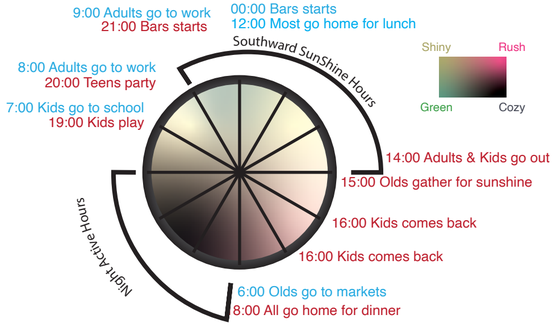


1 generation
(Adults with no kids)

2 generations
1 (Adult)
2 (Couple)

3 generations
3 (0 Kid,1 old)
3 (1 Kid)
4 (2 Kids)
5+ (2+2+1~2)

Behavior and Trends

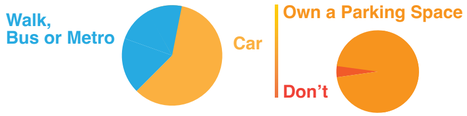


Complaints

In our 62 questionaires, 47 of them are valid, and they are:

No space to park cars(14);
Too much space occupied by cars(11);
No green space(7);
No sunshine(6);
No space to gather comfortably(3);
Too much elevator waiting(3);
Scary(3);

Transportations



In Conclusion, We Need:

1. Enormous house units
2. Southward sunshine
3. Open spaces

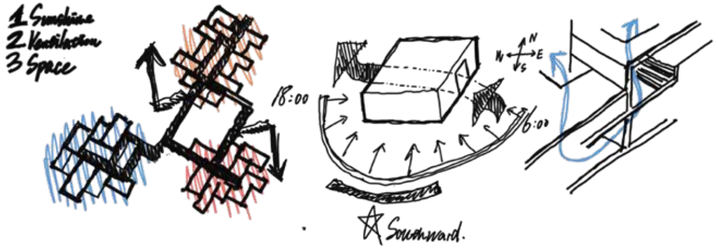
Step3: Idea Generation

In our opinion, to meet the demand of the site, we choose to turn the clear standard into architectual rules, then, compute.

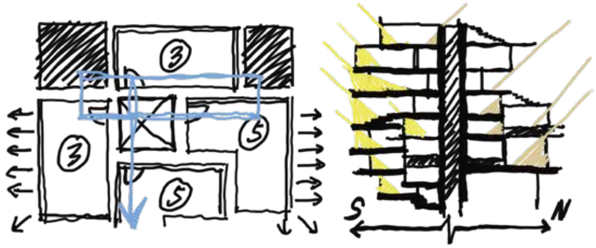
Question: **the building is over crowded, people get stucked on the way in and out.**

Our Solution: **Live, work, do sport in the building and get out less frequently.**

Step 3.1: **Analyze the needs, define the spaces.**



Step 3.2: **Design the house units, and set rules for compositions**



Step 3.3: **Judge the outcomes, select.**

Generation

Step 4.1: Unit Design

In the design of units, ***ventilation***, ***afforest*** and ***sunshine*** are the core issues to be focused.
After fulfilling the need of rooms, private side and public side is distinguished, to fit the inward-outward relationship.

Thus, we have 3 units now.

1Gen
1~2

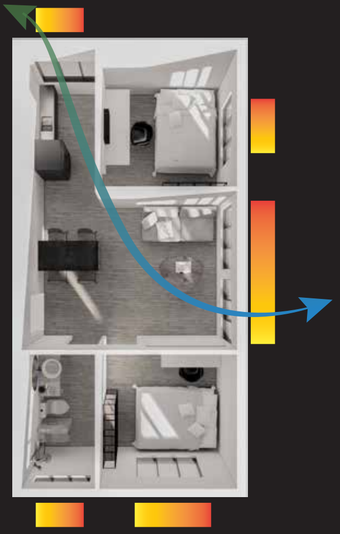
1 Bedroom
1 Living Room
1 Bathroom

The Unit1 is a 6*6m grid, fit in the need of the ***TENANTS***: small; quiet; private, and Eastward Sunshine in 7:00 for work.

2Gens
2~4

2 Bedrooms
(1 for kids)
1 Living Room
1 Bathroom

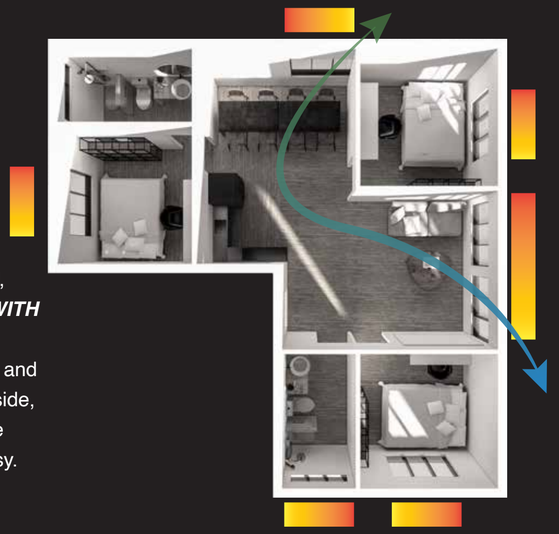
The Unit2 is a 12*6m grid, designed for ***FAMILIES WITH KIDS.***
More than working moms and dads, thanks to the 12m side, kids can be playing by the interior wall: Safe and easy.



2Gens
3~6

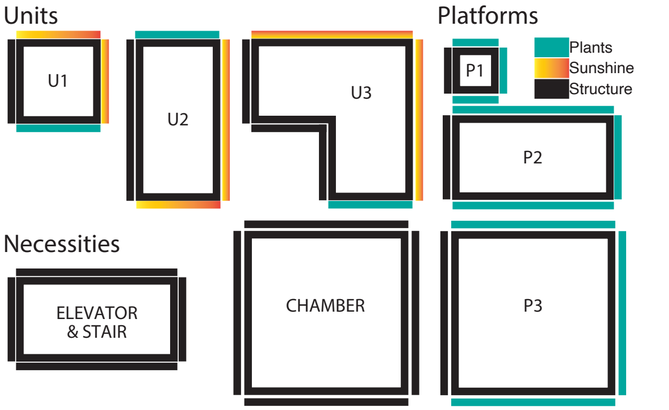
1 Bedroom
1 Living Room
1 Bathroom

The Unit2 is a 12*6m grid, designed for ***FAMILIES WITH KIDS.***
More than working moms and dads, thanks to the 12m side, kids can be playing by the interior wall: Safe and easy.

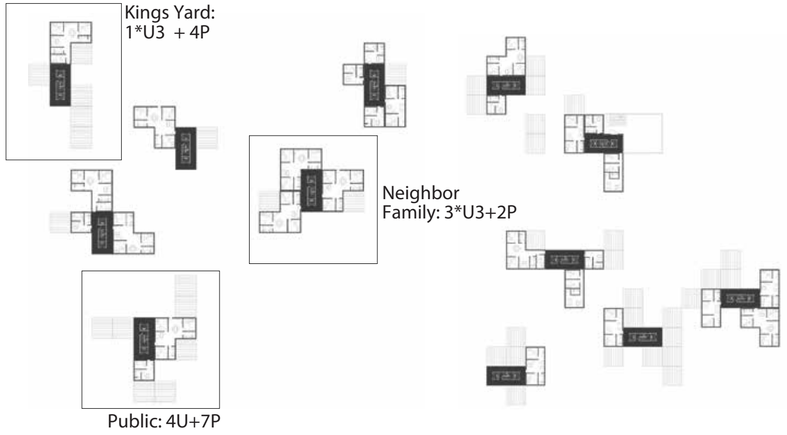


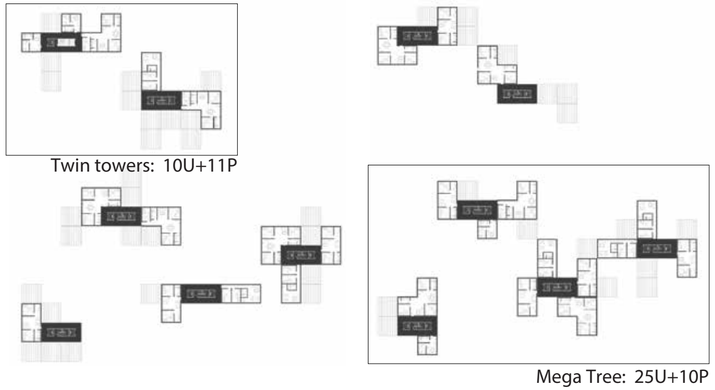
Step 4.2: Rules of Composition

After units themselves, it's time to align them in sequence, with care of each units.



Here's some of the possible compositions. We define the sun facades, plants facades and structures.
And the distance to the transportations are checked.



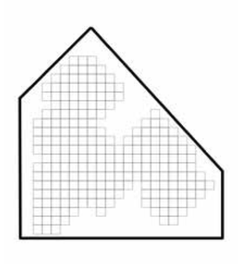


Next step, we translate the connection definitions in computational rules in ***Processing(Java).***

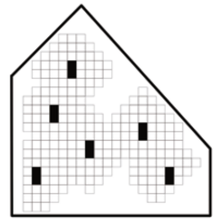
Optimization

Step 4.3: Compute

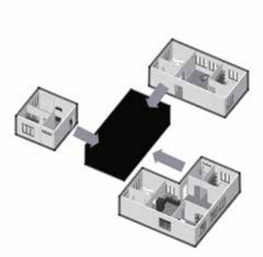
After setting the rules and choosing parameters, we used Processing(Java) to compute and do the massive detail design.



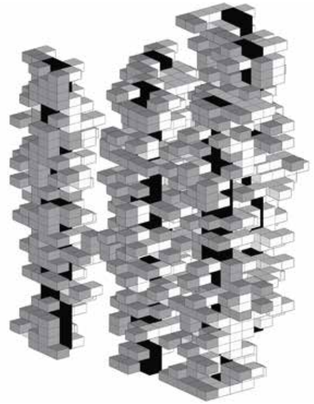
A.Site to Grids
Input the site and devide the site into 6*6m grids, for units to put in.



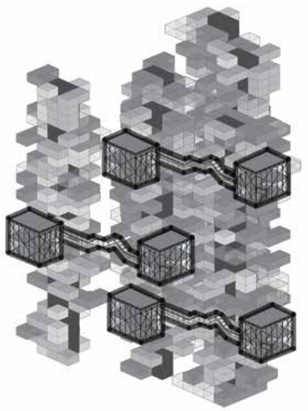
B.Composition
Set grid for elevators and stairs, as a starting point for units to be put next to.
Stairs and elevators are put evenly and connected to the whole site.



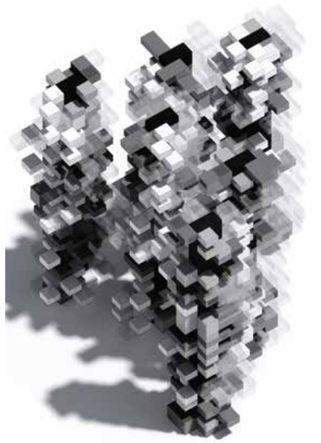
C.Composition
Fulfill the grid with units and platforms.
Transportation and Ventilation are Examined and used.



D.Scale
Compute plan and structure for each floor.
SunTime, Platform and Suspention are checked and computed.



E.Major Spaces
Insert public space to units and add connection.
Publicity works here to determine how much major public spaces would be. And Platform is also effected.



F.Evaluate
Check the connections.
and virtually, by analyzing and 3 values:
Vision Composition, Landscape Ratio and Suspention

G.Choose
After all above, it's time to choose one from multiple ready projects.
Here we judge how it correspond to our original principles.

Parameters

Physical
(must obey)
SunTime (S=1.0>E=0.6>W=0.4>N=0.15)
Transportation (Distance to elevator=4>Entrance=3.2)
Ventilation (Outward=1.0>Inward=0.4)

Structural
(better higher)
Publicity (Rate of shared neighbor ∈ (0.0,1.0))
Platform (Offset area rate to upper floor ∈ (0.0,0.2))
Load (Offset distance to column/Grid edge ∈ (0.0,0.194))

Values

Vision Composition (Public/Private)
Landscape Ratio(Different height/Sum)
Suspention (Way to grid/Edge)

Codes

```
void setup() {
 size(displayWidth, displayHeight, P3D);
 new PeasyCam(this, 200);
 Unit[]g=reproduct(rl);
 Unit[]t=get_the_transport(g);
 gross=new Unit[fmax][g.length];
 int units_of_the_floor=0;
 for (int i=0; i<fmax; i++) {
   Unit[]h=new Unit[g.length];
   for (int j=0; j<gross[i].length; j++) {
     h[j]=new Unit(new PVector(g[j].x, g[j].y, 0));
   }
   gross[i]=h;
   units_of_the_floor=g.length;
   for (int j=0; j<gross[i].length; j++) {
     gross[i][j].tvalue=transportvalue(t, gross[i][j]);
     boolean istran=false;
     for (int l=0; l<t.length; l++) {
       if (t[l].x==gross[i][j].x&&t[l].y==gross[i][j].y) {
         istran=true;}}
     if (istran) {
       gross[i][j].type=1;
     } else if (gross[i][j].tvalue>0&&
gross[i][j].tvalue<3) {
       gross[i][j].type=0;
     } else if (gross[i][j].tvalue==3&&
         random(1)>differrate) {
       gross[i][j].type=3;
       units_of_the_floor--;
     } else if (gross[i][j].tvalue>3) {
       gross[i][j].type=3;
       units_of_the_floor--;}
```

```
if (issawtooth(gross[i], gross[i][j])) {
      if (gross[i][j].type!=3) {
        units_of_the_floor--;
      }
      gross[i][j].type=3;
    }//clear sawtooth

 if (i>k_of_the_skyline*(gross[i][j].x+s/2)
+b_of_the_skyline-1) {
      if (gross[i][j].type!=3) {
        units_of_the_floor--;
      }
      if (gross[i][j].type!=3) {
        gross[i][j].type=3;
      }
    }//skyline2
    gross[i][j].hmin=(hd*i);}}}
float deleterate(float t, float x, float z) {
 float d=0;
 float a, b, c;
 a=map(x, ox+1126, ox, lowestrate,
highestrate);
 b=map(z, 0, maxheight, 0, 1);
 c=map(t, 0, 5, -mountainrate, mountainrate);
 if (b>a+(float)hd/maxheight) {
   d=1;
 } else if (b>a-2*(float)hd/maxheight) {
   d=mountainrate+c;
 } else {d=0;}
 return d;
}
```

```
Unit[] reproduct(PVector[] field) {
 ArrayList<Unit> totallist = new ArrayList<Unit>();
 Unit start=new Unit(new PVector(0, 0, 0));
 while (!start.check(field)) {
   start=new Unit(new PVector(random
(displayWidth), random(displayHeight), 0));}
 int x_overscreen=int(displayWidth/s)+1;
 int y_overscreen=int(displayHeight/s)+1;
 Unit[][]unitsoverscreen=new Unit
[x_overscreen][y_overscreen];
 for (int i=0; i<x_overscreen; i++) {
   for (int j=0; j<y_overscreen; j++) {
     unitsoverscreen[i][j]=new Unit(new PVector
(start.x%s+i*s, start.y%s+j*s, 0));
   if (unitsoverscreen[i][j].check(field)) {
       totallist.add(unitsoverscreen[i][j]);}}}
void play_box(Unit a,color b){
 pushMatrix();
 //noStroke();
 fill(b);
 translate(a.x,a.y,a.hmin+hd/2);
 box(s,s,hd);
 popMatrix();
}
Unit[]total=new Unit[totallist.size()];
 for (int l=0; l<totallist.size(); l++) {
   total[l]=totallist.get(l); }
 return total;}
```

```
void play_border(PVector[]a, color b) {
 fill(b);
  beginShape();
  for (int i=0; i<a.length; i++) {
   vertex(a[i].x, a[i].y);}
  endShape(CLOSE);}
void play_points(PVector[]a, color b) {
 fill(b);
 for (int i=0; i<a.length; i++) {
   point(a[i].x, a[i].y);
   text("("+str(a[i].x)+
   ","+str(a[i].y)+")", a[i].x, a[i].y);}}
class Unit {
 int type;
 //0-residential  1-transport  2-public  3-void
 PVector center;
 PVector nw, ne, se, sw, nw2, ne2, se2, sw2;
 float tvalue, svalue, avalue;
 float hmin, hmax, xmin, xmax, ymin, ymax, x, y;
 PVector[]border;
 Boolean airchecked;
 Unit (PVector a) {
   airchecked=false;
   tvalue=-1;svalue=-1;avalue=-1;type=0;
   y=a.y;x=a.x;center = a;hmin=0;
```

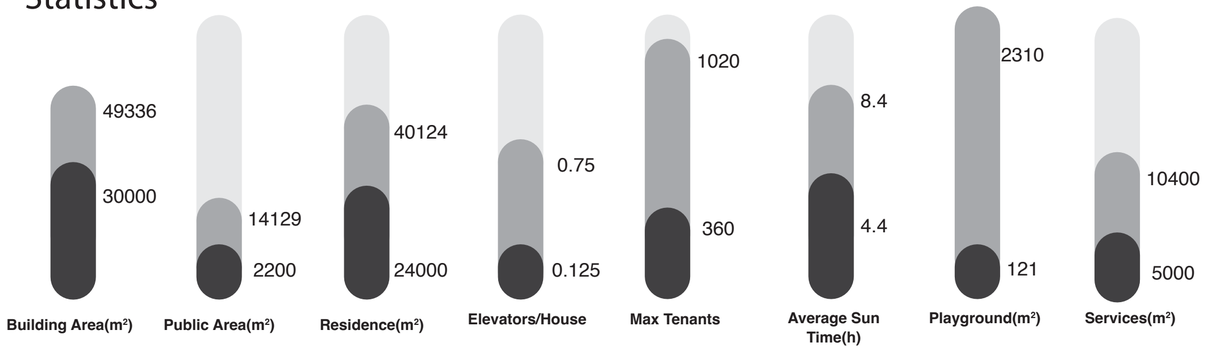
```
nw=new PVector(a.x-s/2, a.y-s/2, hmin);
    ne=new PVector(a.x+s/2, a.y-s/2, hmin);
    se=new PVector(a.x+s/2, a.y+s/2, hmin);
    sw=new PVector(a.x-s/2, a.y+s/2, hmin);
    border=new PVector[]{nw, ne, se, sw};
    xmin=a.x-s/2;xmax=a.x+s/2;ymin=a.x-s/2;
    ymax=a.x+s/2;}
  boolean check (PVector[]b) {
    boolean result =false;
    if (inside(b, nw)&&inside(b, ne)&
&inside(b, se)&&inside(b, sw)) {
      result=true;}
    return result;}
  Unit findnext(int c) {
    float r=s;
    float theta=PI*(c/2.0);
    PVector next_center=new PVector
(center.x+cos(theta)*r, center.y+
sin(theta)*r, center.z);
    Unit next= new Unit(next_center);
    return next;}}
boolean issawtooth(Unit[]a, Unit b) {
 boolean is=false;
 int i=0;
 int j=0;
 for (Unit c : a) {
   if (c.type==3) {
     if (c.x==b.x&&abs(c.y-b.y)==s) {
       i++;
     } else if (c.y==b.y&&abs(c.x-b.x)==s) {
       i++;}}}
```

```
for (Unit c : a) {
   if (c.type!=3) {
     if (abs(c.x-b.x)==s&&abs(c.y-b.y)==s) {
       j++;
     }}}
 if (i>2)
   is=true;
 if (i>1&&j==4)
   is=true;
 return is;
}

void draw() {
 background(255);
 play_border(rl, color(155, 155, 155));
 for (int i=0; i<fmax; i++) {
   for (int j=0; j<gross[i].length; j++) {
     //gross[i][j].hmin=(hd*i);
     if (gross[i][j].type==0) {
       play_box(gross[i][j], color( map
       (gross[i][j].tvalue, 0, 5, 0, 255), 155, 155));
     } else if (gross[i][j].type==1) {
       play_box(gross[i][j], color(15, 15, 15));
     } else if (gross[i][j].type==2) {
       play_box(gross[i][j], color(255, 255, 255));
     } else if (gross[i][j].type==4) {
       play_box(gross[i][j], color(5, 5, 255));
     }}}}
```

Vertical City

Statistics

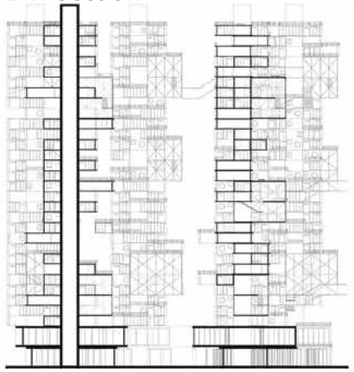


Conclusion

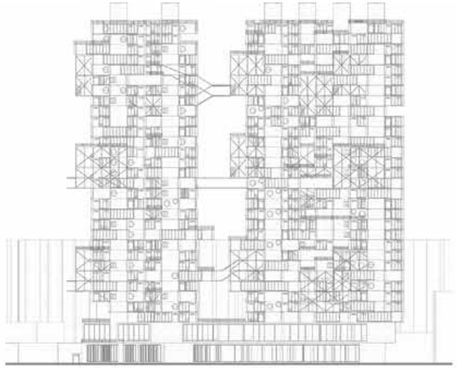
1. In the limited building area of Nanjing, we've optimized the composition and made maximum residence with improved physical properties.

2. In the Vertical City, we fulfilled ordinary life cycle with mixed spaces of services. Though we don't know when would it be, the customized residence trend is inevitable.

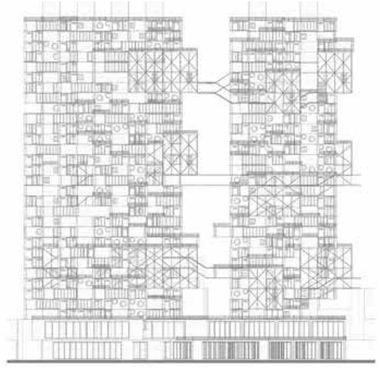
E-W Section



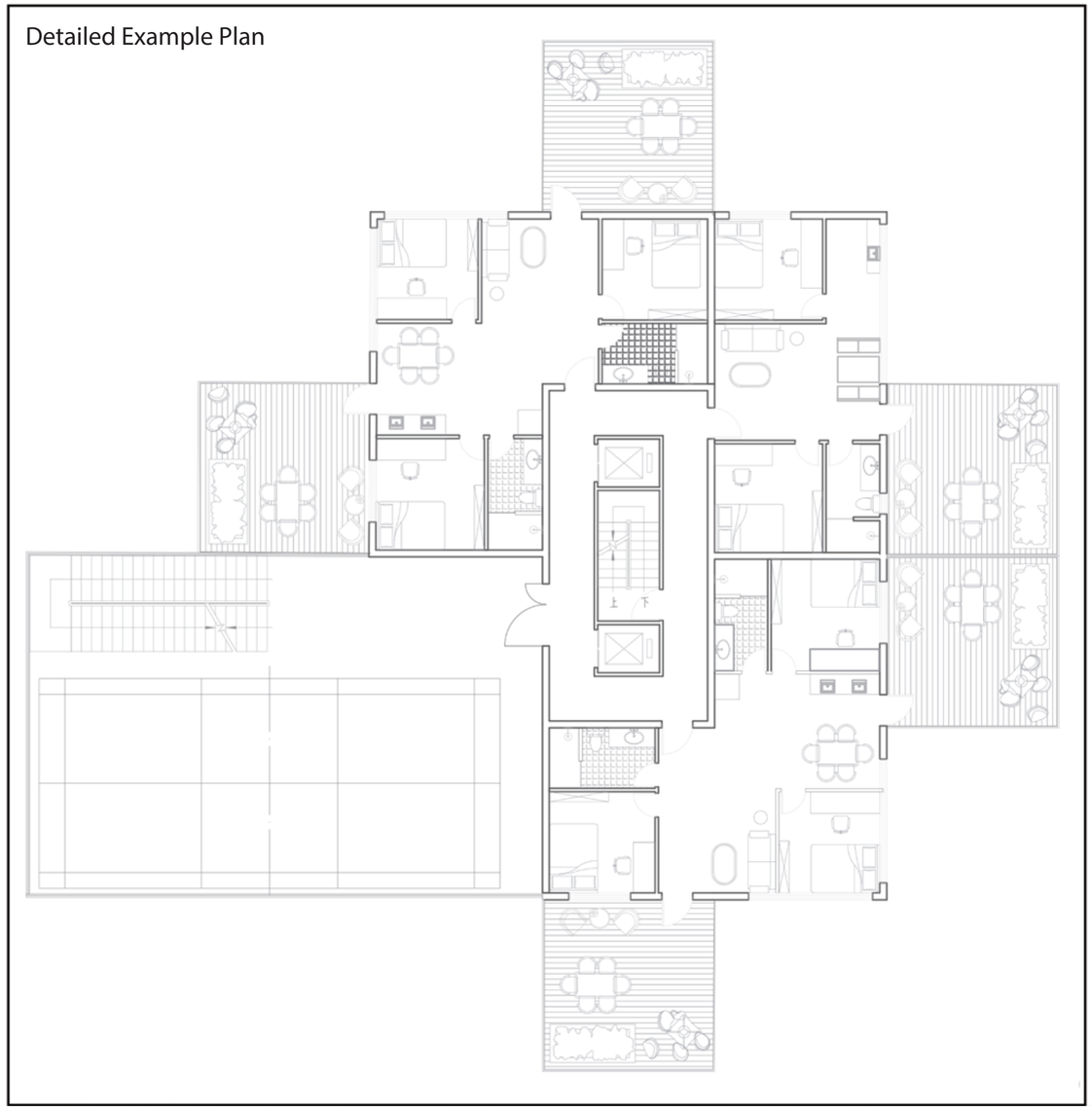
East Facade



North Facade



Detailed Example Plan



Outcomes

Community Balcony

Window View

Public Space

Bird View