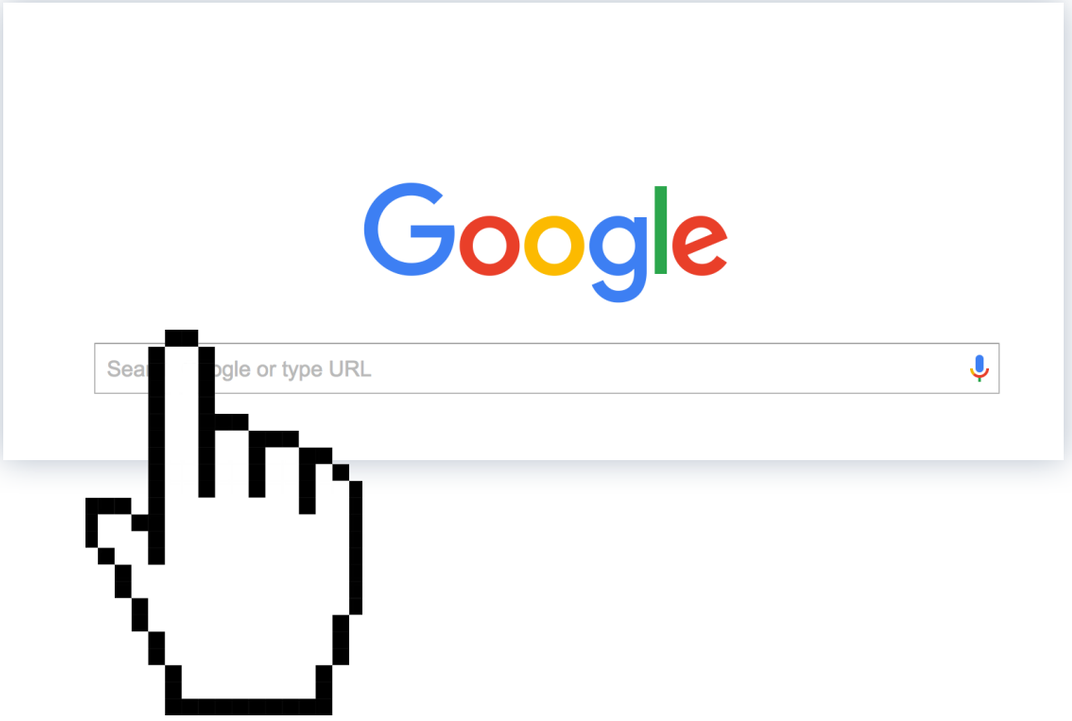
SURFACE TRACE :

AN EXPERIMENTAL SEARCH INTERFACE

TYPOGRAPHY AND INTERFACES

The typography that most of us consume most frequently is coded typography linked to computational systems. The type that constitutes our digital world watches and consumes our behaviour and habits.



“BEING ON THE INTERNET MEANS BEING TRACEABLE”

– Critical Atlass of the Internet

TYPOGRAPHY AND INTERFACES

ONE SIZE FITS NONE : TOWARDS CRITICAL INTERFACE DESIGN BY ROSA LLOP

→ Interfaces should be challenging rather than accessible

Interfaces should be challenging for advanced users and for the mainstream, so that interaction with the system is an intellectually motivating experience.

→ Interfaces should be tolerant but also sceptical

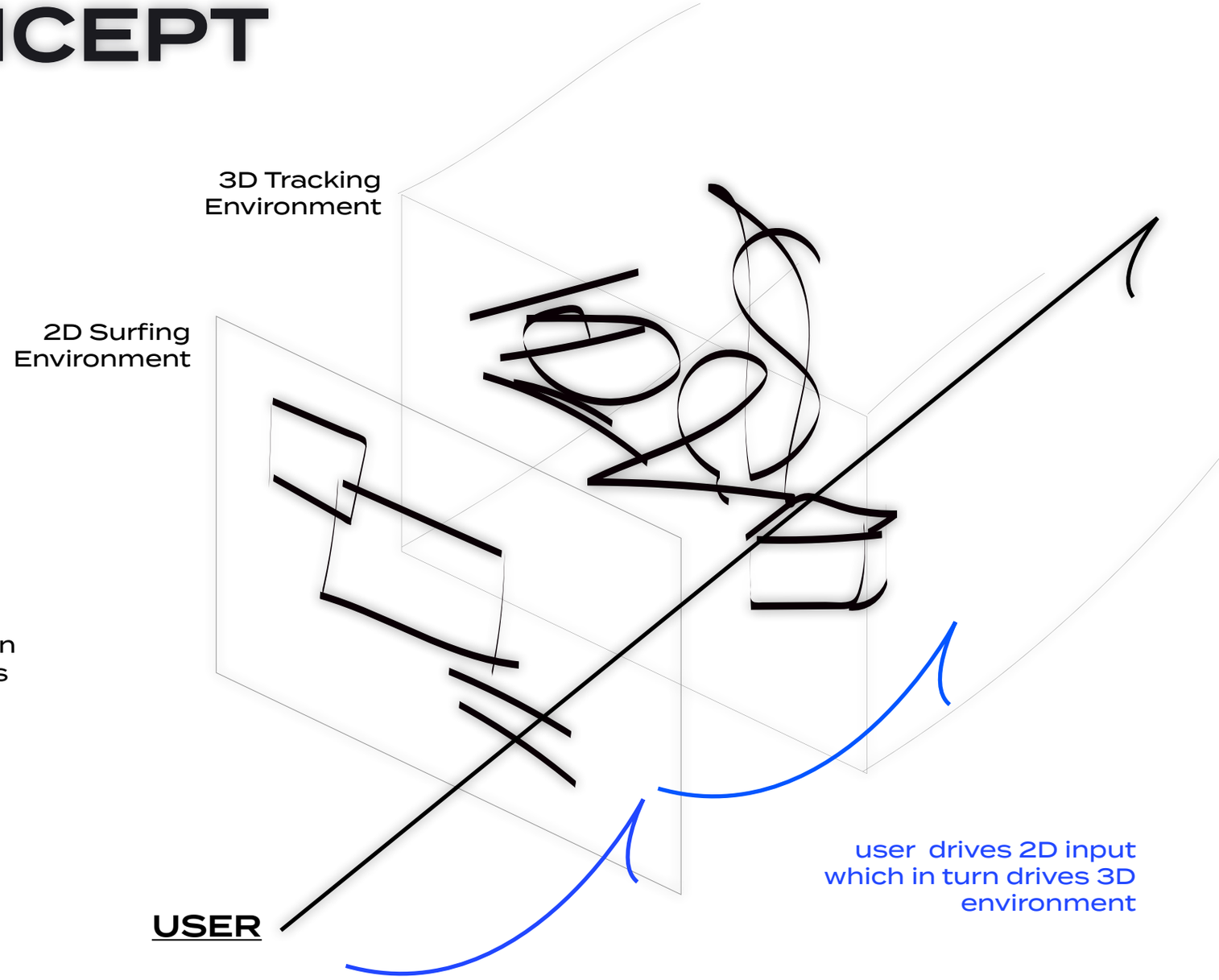
I propose that interfaces be tolerant, but also sceptical. That they encourage users to be curious about how they interact with them, help to disclose the values that arise from their use, and promote more conscious and responsible behaviour.

→ Interfaces should offer a sense of empathy rather than control

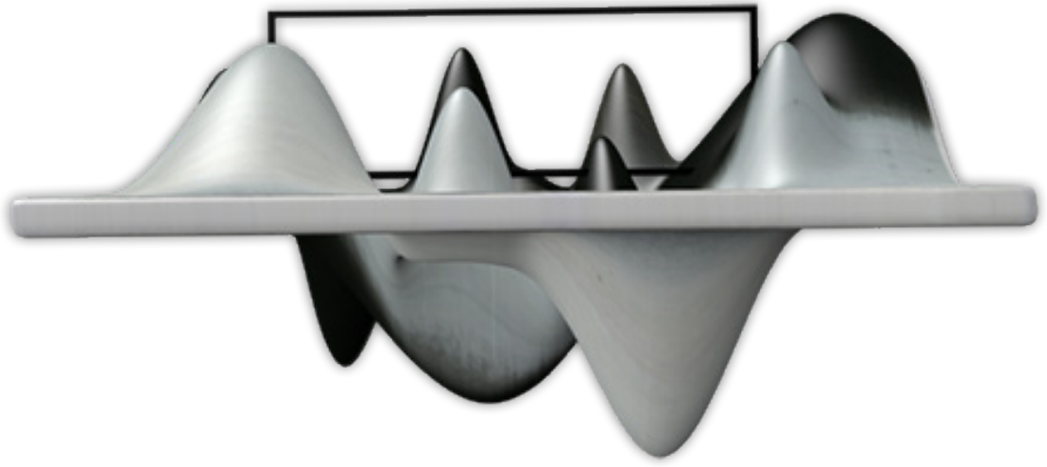
That instead of giving signs that they are executing the requested action they send signs that show the complexity of the system, generate curiosity, and encourage an attitude of exploration and learning.

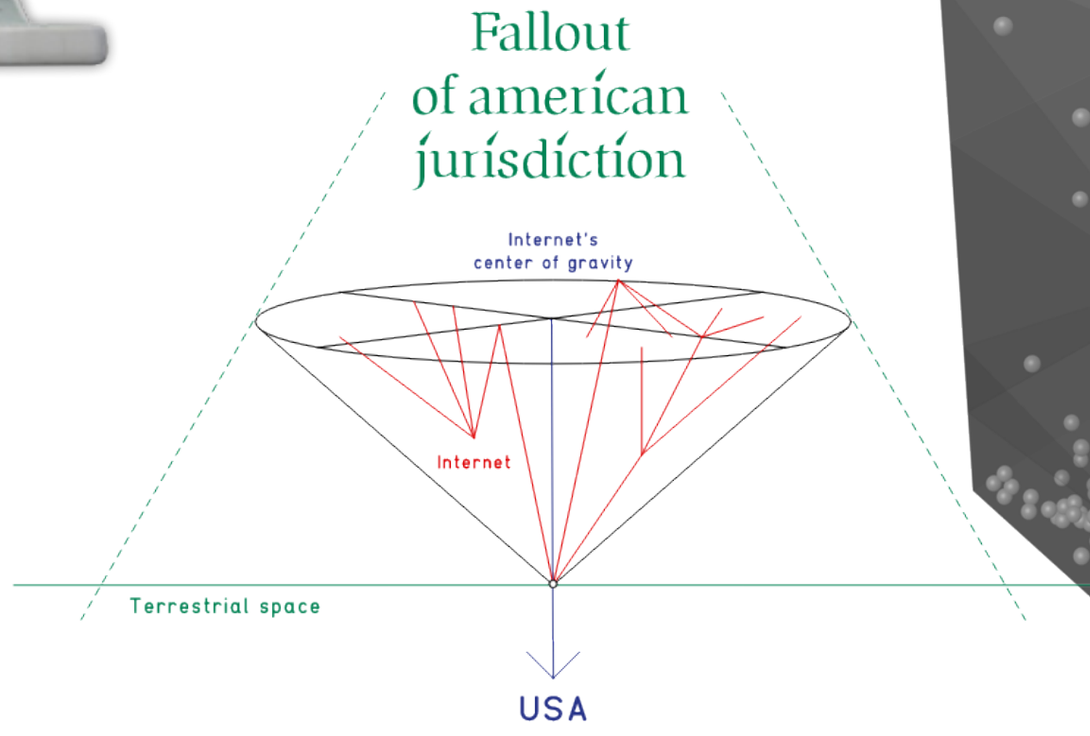
INITIAL CONCEPT

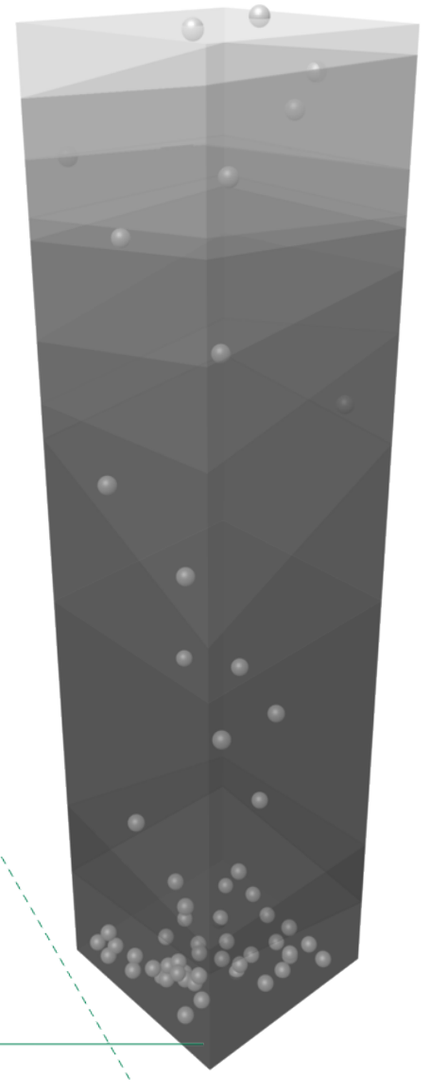
An experimental generative search interface that encourages user interaction and questions user interaction and stores the matadata of this interaction to be displayed.



MAPPING AS A TOOL FOR UNDERSTANDING







Representing ideas and processes in a visual or spacial way helps us to connect to those things and understand them in a deeper way.

PROCESS

EXPERIMENTING WITH METHODS OF TRACKING

```
  this.y = y;
  this.z = z;


  this.history = [];

this.update = function() {
  this.y = mouseY - windowHeight/2;
  this.x = mouseX - windowWidth/2;
  angle += 0.3;
  this.z = sin(frameCount*0.001)*800;

  var v = createVector(this.x, this.y, this.z);
  this.history.push(v);

  if (this.history.length > 10000) {
    this.history.splice(0, 1);
  }

}
this.show = function() {
  noStroke();
  fill(0);
  strokeWeight(10);
  beginShape();

  fill(0, 0, 255);
  stroke(0);
  for (var j = 0; j < 10; j++ ) {
  for (var i = 0; i < this.history.length; i++) {
    var pos = this.history[i];
    vertex(pos.x + 2, pos.y, pos.z);


  }
```

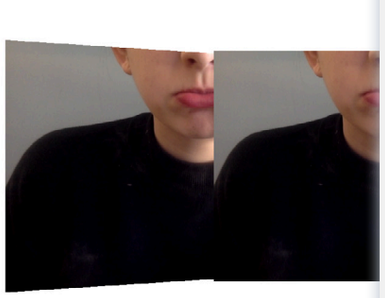
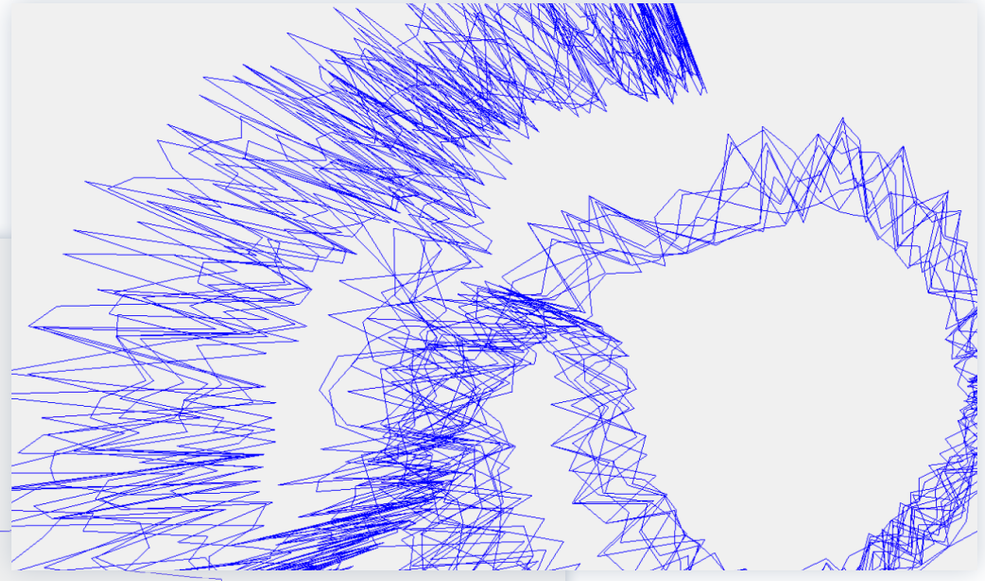


Photo taken and displayed each time mouse clicked:

Mouse position logged in 3D space:



PROCESS

METHODS OF TRACKING

→ Mouse Position
Mouse position logged and mapped in 3D space as well as mouse position affecting camera position / angle.

→ Images (giphy)
Returned and diaplyed on objects

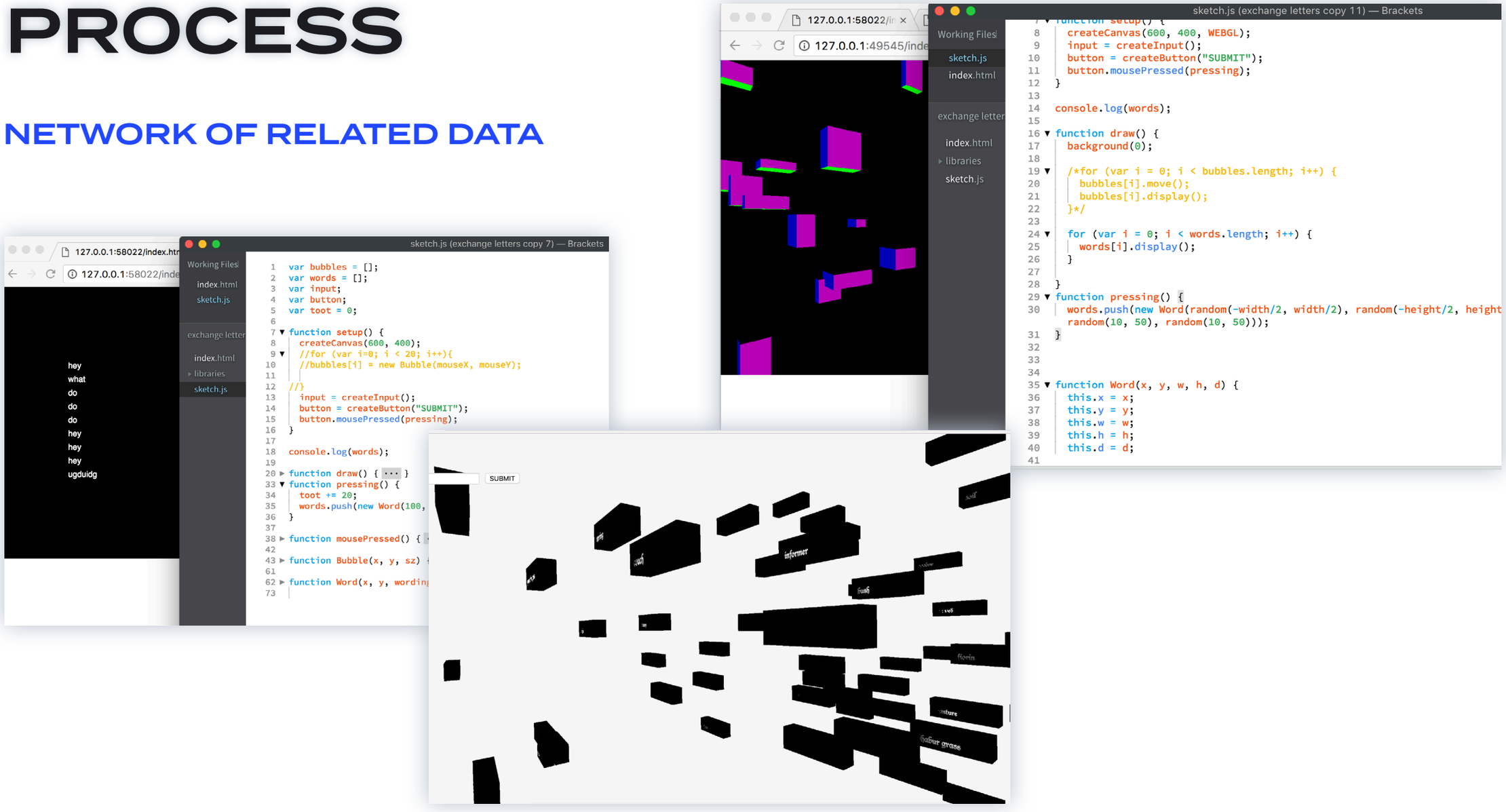
→ Network of extra data
related words (from wordnik) produced and displayed in sky

→ Searched Terms
Displayed on ground, constrcuting the walls of the environment

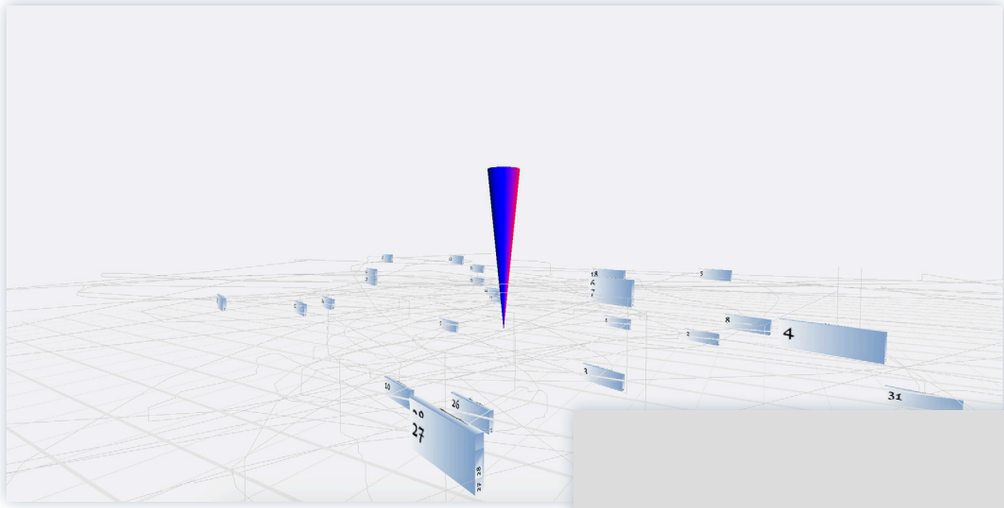
My design involved a number of different complex parts so I contructed each individually, gradually combining them to form the whole.

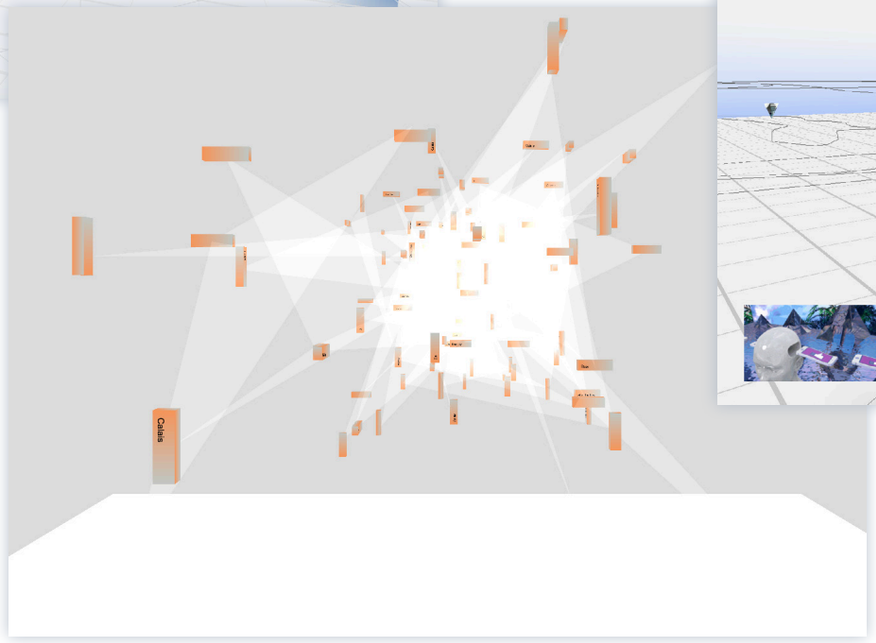
PROCESS

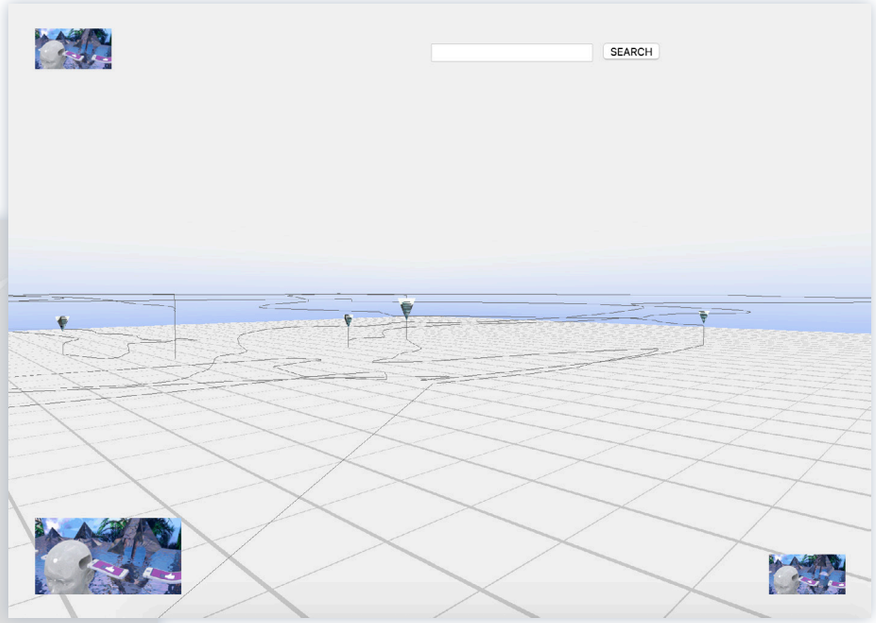
NETWORK OF RELATED DATA



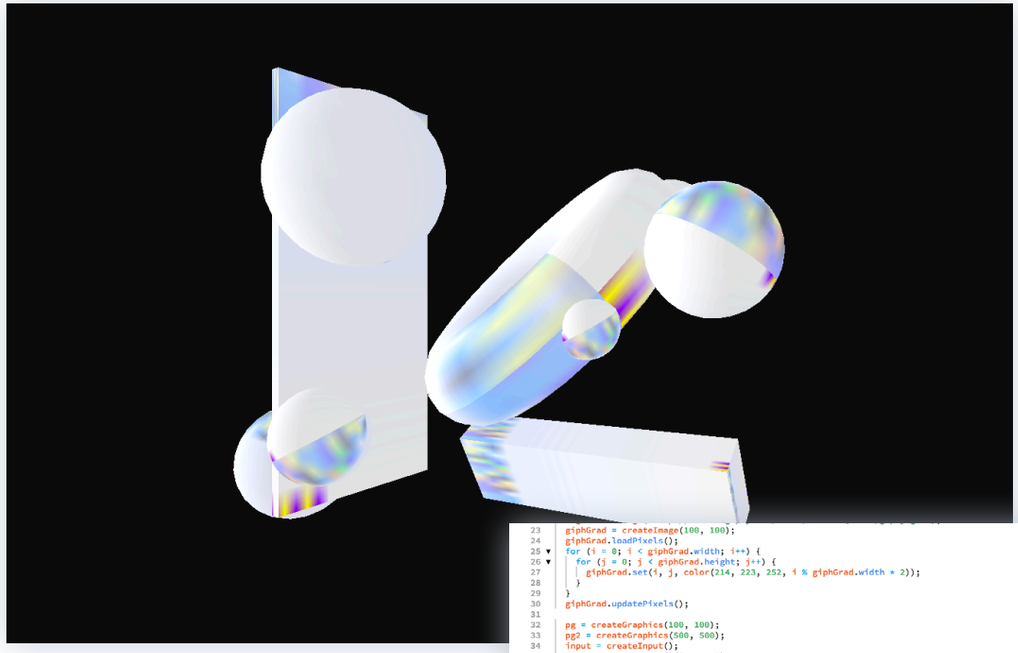
PROCESS





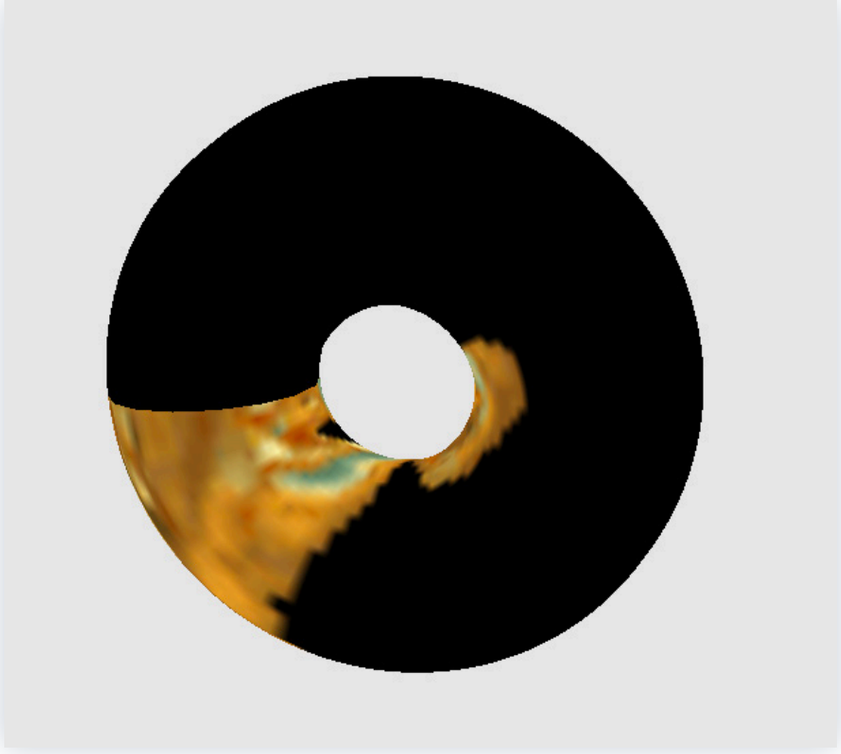


PROCESS



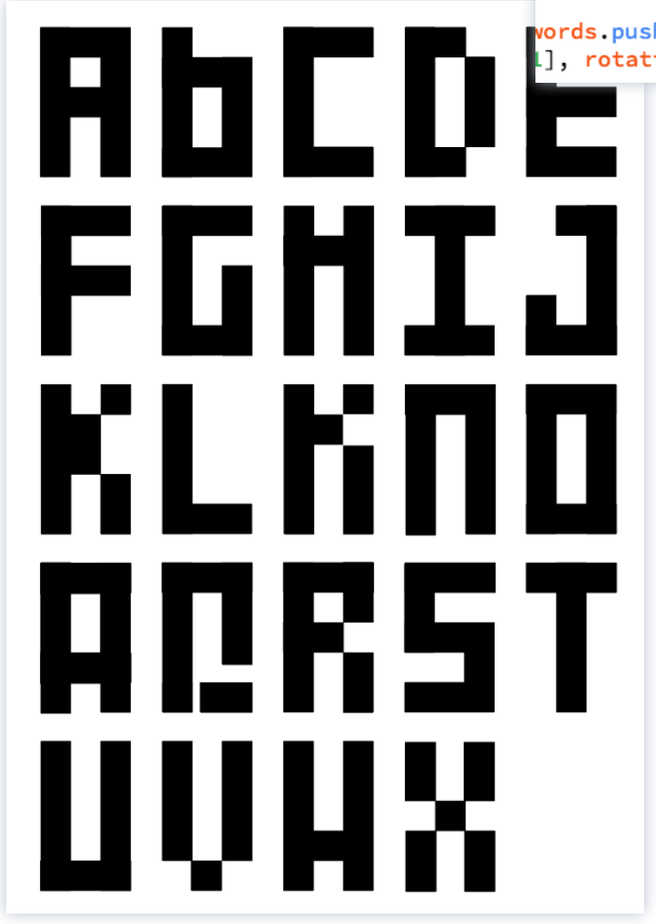
```
  giphGrad = createImage(100, 100);
  giphGrad.loadPixels();
  for (i = 0; i < giphGrad.width; i++) {
    for (j = 0; j < giphGrad.height; j++) {
      giphGrad.set(i, j, color(214, 223, 252, i % giphGrad.width * 2));
    }
  }
  giphGrad.updatePixels();

  pg = createGraphics(100, 100);
  pg2 = createGraphics(500, 500);
  input = createInput();
  button = createButton("SUBMIT");
  button.mousePressed(askGiphy);
}
function gotData(giphy) { ··· }
function askGiphy() {
  var url = api + apiKey + input.value();
  loadJSON(url, gotData);
}


function draw() {
  ambientLight(250);
  directionalLight(60, 60, 60, -1, 0, 0);
  background(10);
  orbitControl();
  angle += 0.3;
  if (gifss.length <= 0) {
    pg.background(243);
    pg.imageMode(CENTER);
    pg.imageMode(CORNER);
    pg.image(giphGrad, 0, 0);
    giphyShapes(0, 0, -150, pg, pg, pg, pg, pg, pg, pg, pg, pg);

  } else if (gifss.length > 0) {
    pg.background(230);
    pg.imageMode(CENTER);
    pg.image(img2, 0, 0, 1600, 1600);
    pg.imageMode(CORNER);
    pg.image(giphGrad, 0, 0);
    giphyShapes(0, 0, -150, pg, pg, pg, pg, pg, pg, pg, pg, pg);
  }

}


function giphyShapes(x, y, z, giph1, giph2, giph3, giph4, giph5, giph6, giph7, giph8,  giph9) { ··· }
```

IMAGE REPSONSE TO INPUT AS TEXTURE



PROCESS

INPUTS AS ARCHITECTURAL STRUCTURE



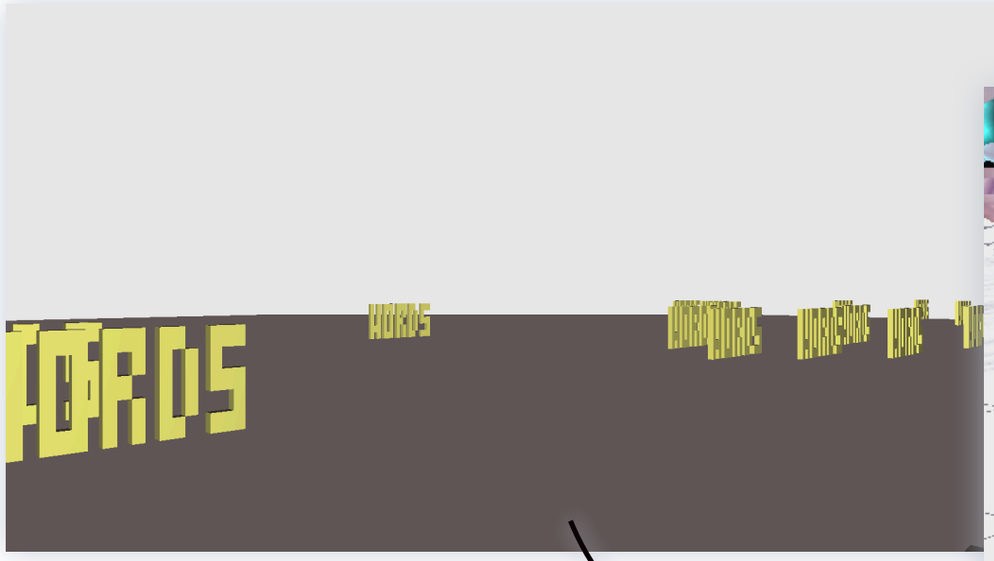
```
inputCollect.push(input.value()); //add imputs to an array for 3D text

words.push(new InputsData(random(-boxSize/2.3, boxSize/2.3 - 400), random(0, 0), random(-boxSize/2.3, boxSize/2.3), inputCollect[inp
1], rotations[floor(random(0, rotations.length))]));
```

```
for (var j = 0; j < words.length; j++) {

  var char = words[j].wording.split("");
  for (var i = 0; i < char.length; i++) { //split all arrays into array of individual letters
    if (words[j].x > -300 && words[j].x < 300 && words[j].z > -300 && words[j].z < 300) {
      var wordX = words[j].x + 500;
      var wordZ = words[j].z + 500;
    } else {
      var wordX = words[j].x;
      var wordZ = words[j].z;
    }
```

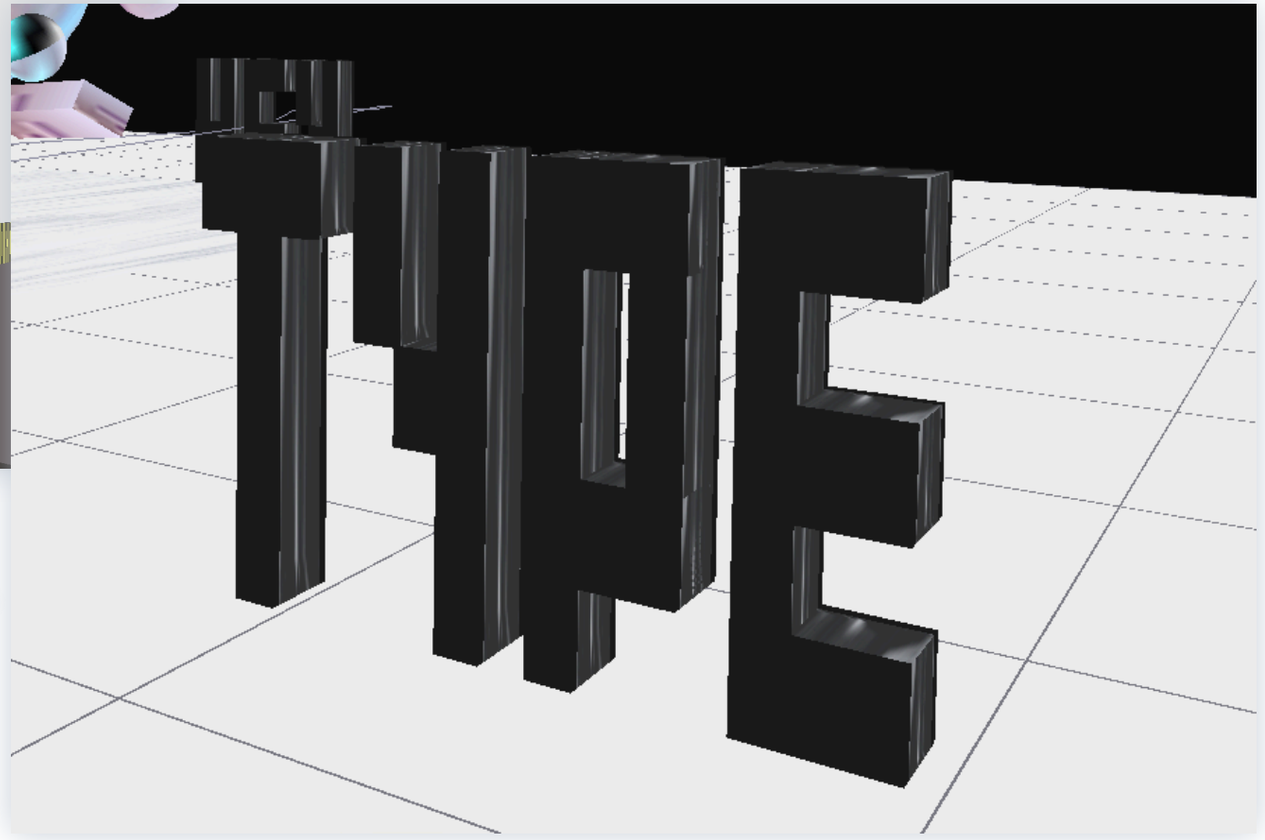
```
switch(char[i].toLowerCase()) { //change all letters to lowercase
//switching text for designed 3D letters
 case 'a':
  for (var lA = 0; lA < 40; lA ++) {
    if (words[j].wording.charAt(lA) == 'a') {
      var xa = letterSpacing * lA;
      drawA(wordX + xa, 0 + words[j].y + height/2 - heightoffset, 0 + wordZ, stretch, words[j].ystretch, letterScale, words[j].letterRo
      marbleTextureB);
    }
  }
  break;
```

PROCESS

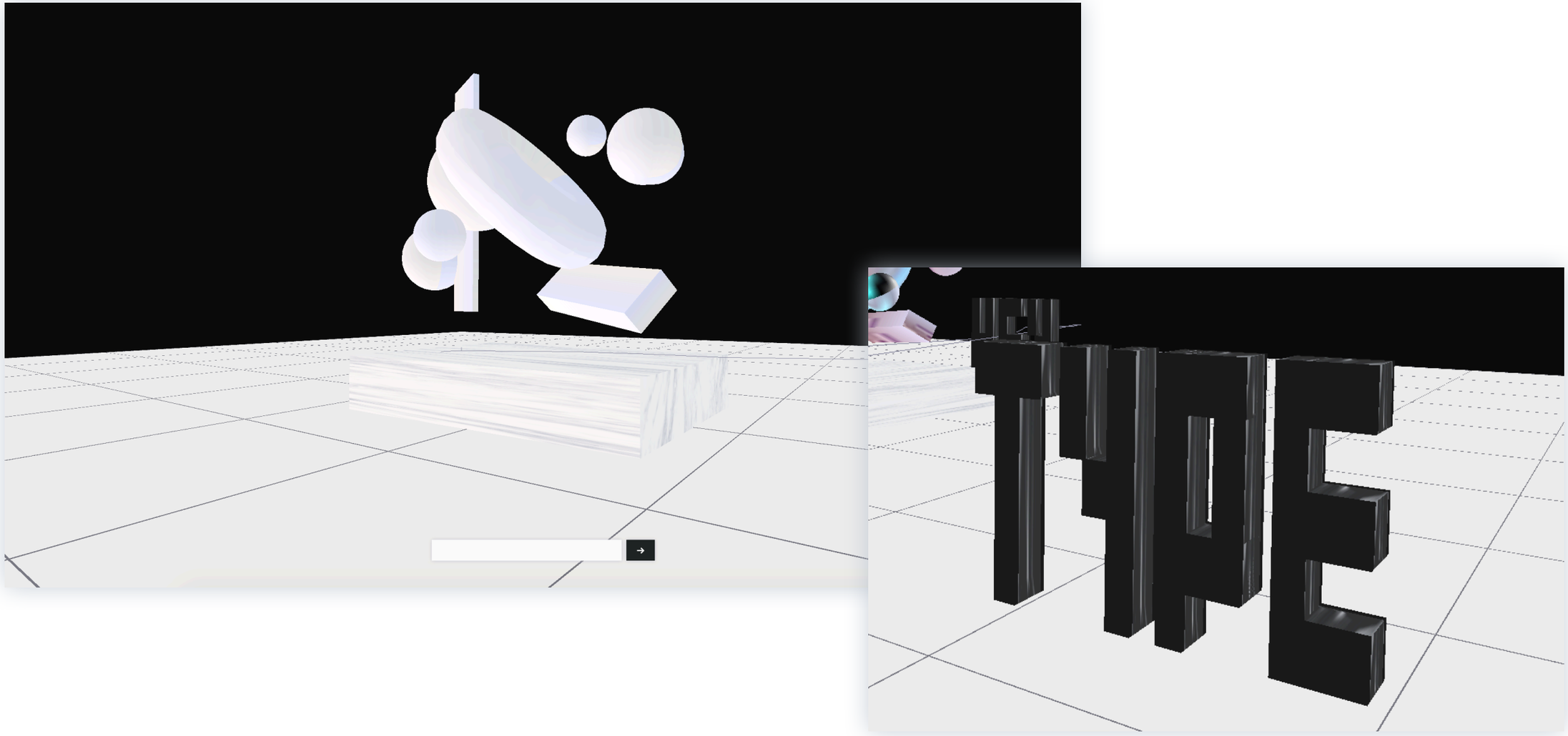
SETTING THE TYPE



yay.



FINAL OUTCOME



FINAL OUTCOME