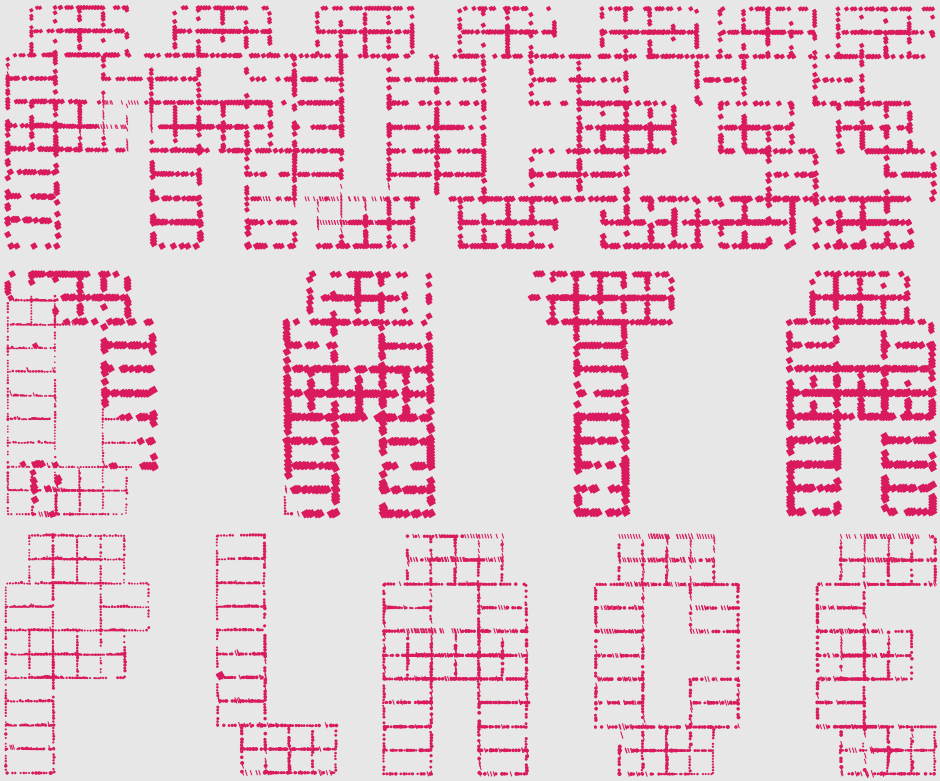


Process Data Place

A Generative Framework for Multisensory Type Forms.

Vol. 1 - Surrey

Development

Monika Sowa

Process Data Place

Process Data Place
A Generative Framework for Multisensory Type Forms
Vol. 1 - Surrey

DEVELOPMENT

Part two of a two-part project consisting of a research volume and a development volume.

This publication documents the development process of a design project examining the role of systems in typography. It traces the progression from initial research and experimentation to the construction of generative typographic systems and final outcomes.

Written and designed by Monika Sowa

BA (Hons) Graphic Design [Visual Communication and Illustration]
University of West London
2026

Typography set in Gotham Narrow (titles), Gotham (body)
Designed using Adobe InDesign
Printed by Mixam, London
A5 [printed on silk 130gsm, cover 300 gsm]

Design tools: Processing, Glyphs, Illustrator, Figma

All images are produced by the author unless otherwise stated.

The project is further extended through an online platform:
process.monikasowa.com

Process Data Place

A Generative Framework for Multisensory Type Forms.

Vol 1 - Surrey

Monika Sowa

01

Aims and Scope

Scope and Argument

This publication documents the development of a design project exploring typography as a system shaped by rules, processes, and variation. Building on the accompanying research volume, the project translates theoretical concepts into practice through a series of structured experiments and iterations.

The process focuses on the construction of a generative typographic framework, in which letterforms are not individually drawn, but emerge from defined rules and parameters. Through material exploration, system development, and computational experimentation, the project investigates how visual and sonic data can drive typographic behaviour.

By shifting emphasis from form-making to system design, it traces how a consistent set of rules can produce multiple outcomes. It presents the development of a process in which typography operates as a responsive and adaptive system rather than a fixed visual artefact.

Aims

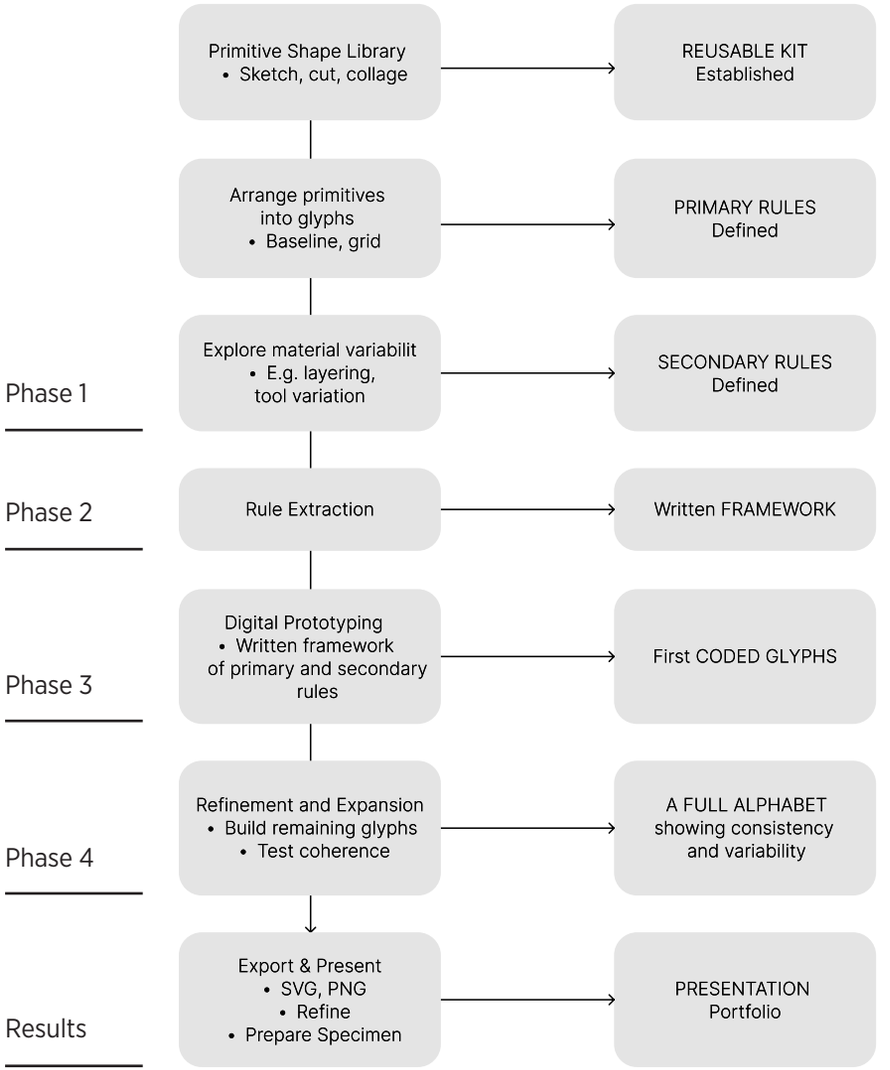
The project aims to examine the role of typography in shaping identity and place, and explore how visual and sonic data influence typographic behaviour. It also seeks to evoke cultural reflection through multisensory systems and create an immersive framework connecting design, sound, and narrative.

Audience

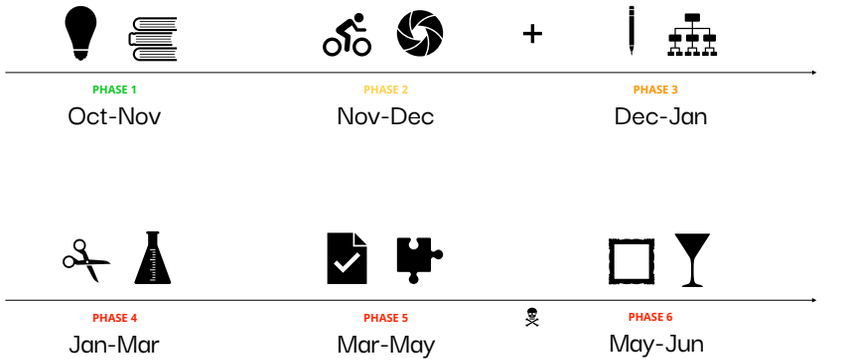
Primary:	Design Students Educators
Secondary:	Visual Artists Cultural Institutions General Public

02

Methodology



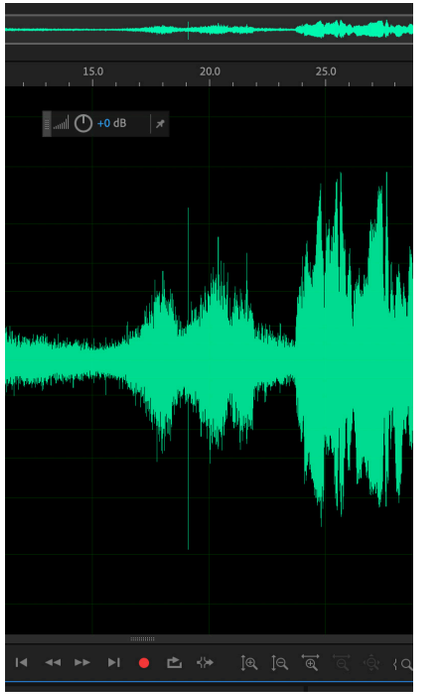
Workflow Diagram

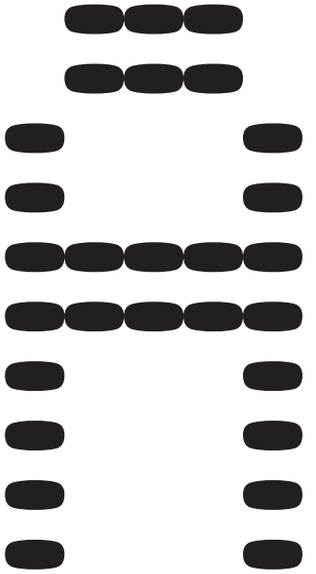


Timeline

METHODS

These diagrams outline the methodological framework of the project, mapping the progression from initial exploration and rule extraction to digital prototyping and final outputs. Each phase builds on the previous through defined processes, demonstrating how a rule-based typographic system produces consistent yet variable outcomes.

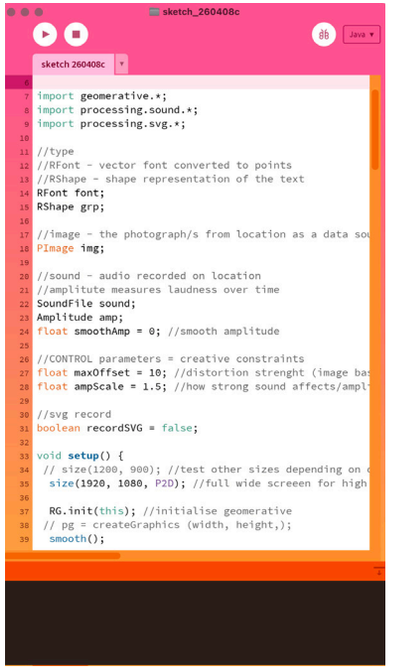


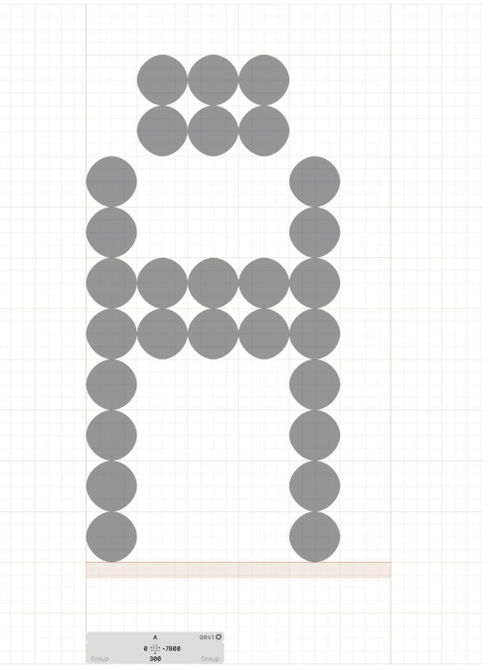


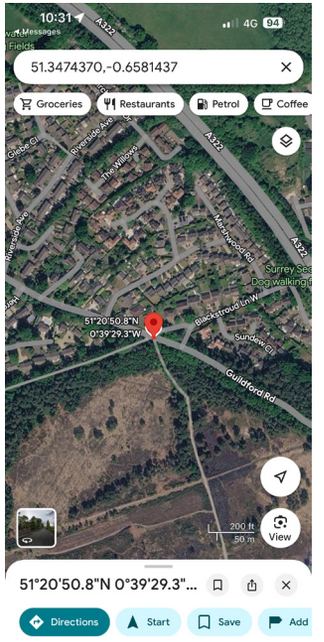
APPROACH

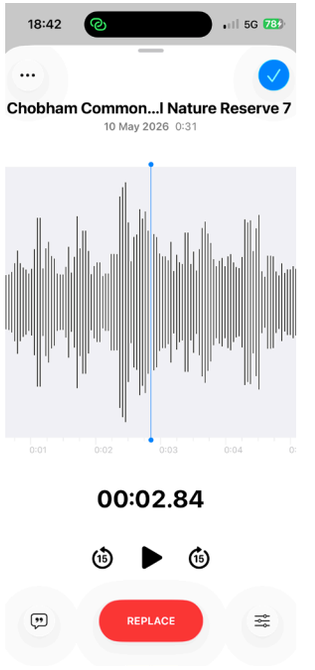
Multisensory input combining field recording, photography, material experimentation and custom typeface design.

Images:
Sound recording
Pixelfont Flat
Typographic sketches
Location Photography









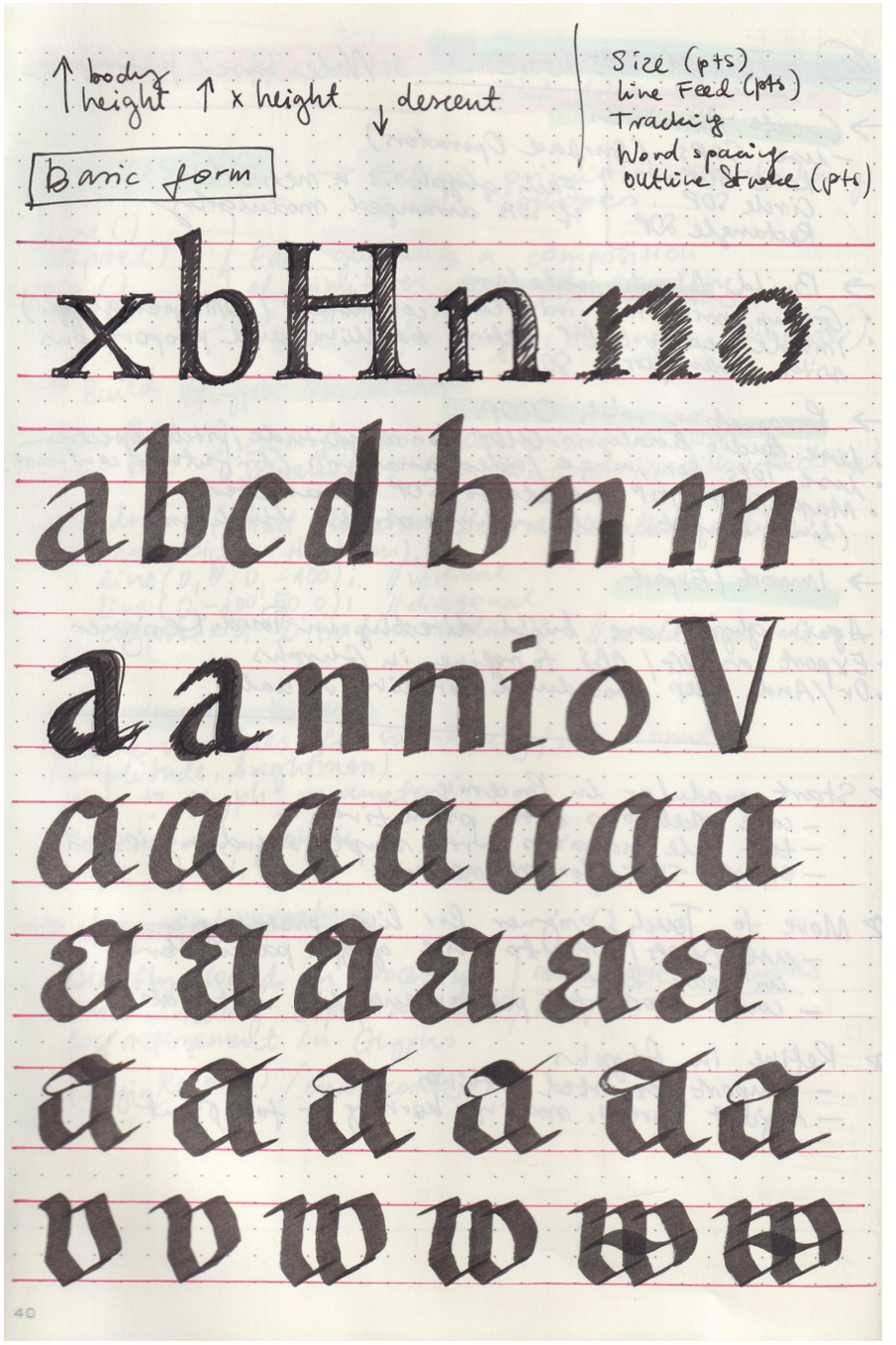
TOOLS

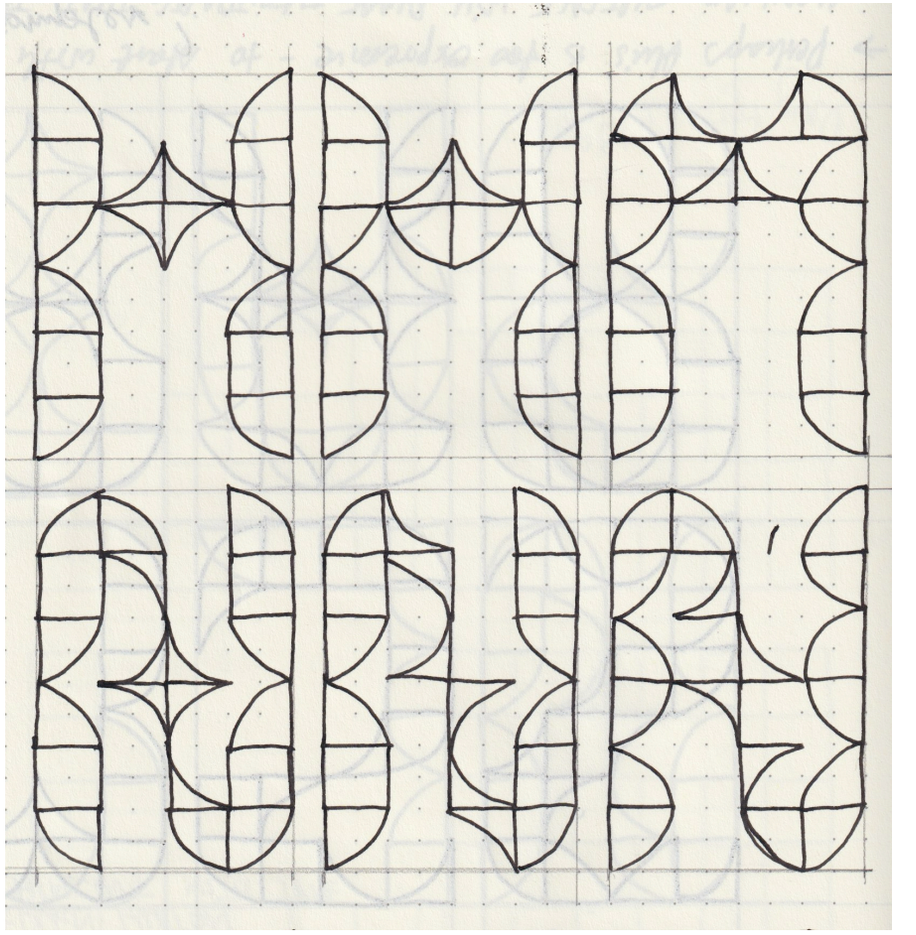
Software and tools supporting the development of the generative typographic system, enabling rule definition, data mapping, and visual output.

Images:
Processing
Glyphs
Stylosette
Google Maps
Voice Memos App
Lumix GX9 + fixed focal lenght

03

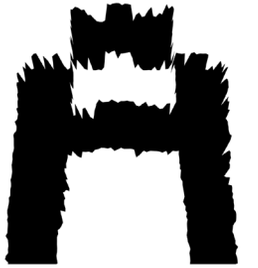
Research Translation

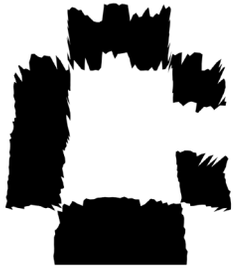


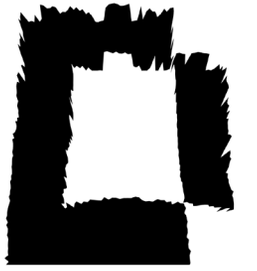


HISTORICAL SYSTEMS

Sketch-based exploration of letterforms informed the development of a structured typographic system. The transition from analogue mark-making to grid-based modular construction established consistency through proportion, alignment, and repeatable geometric elements.

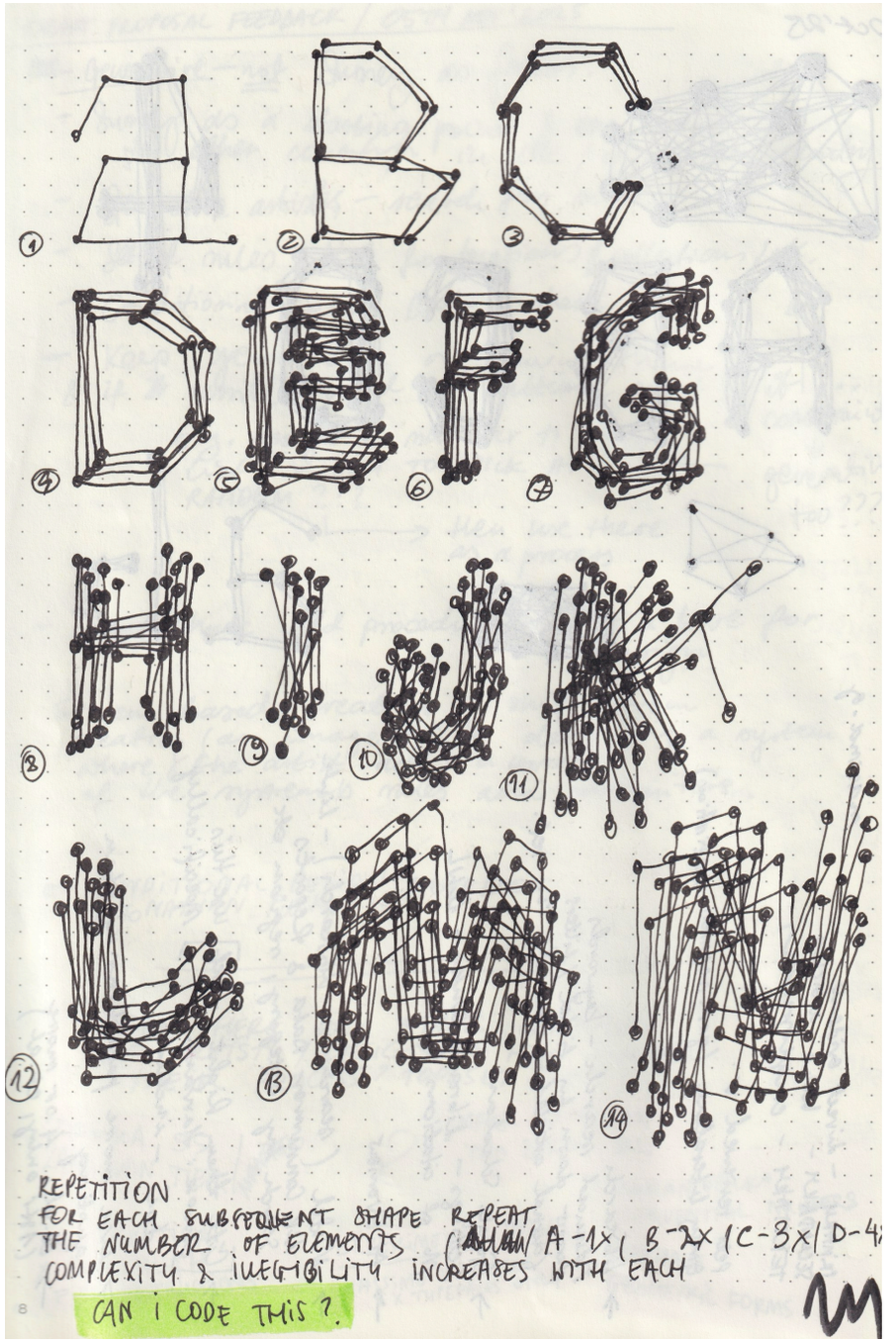


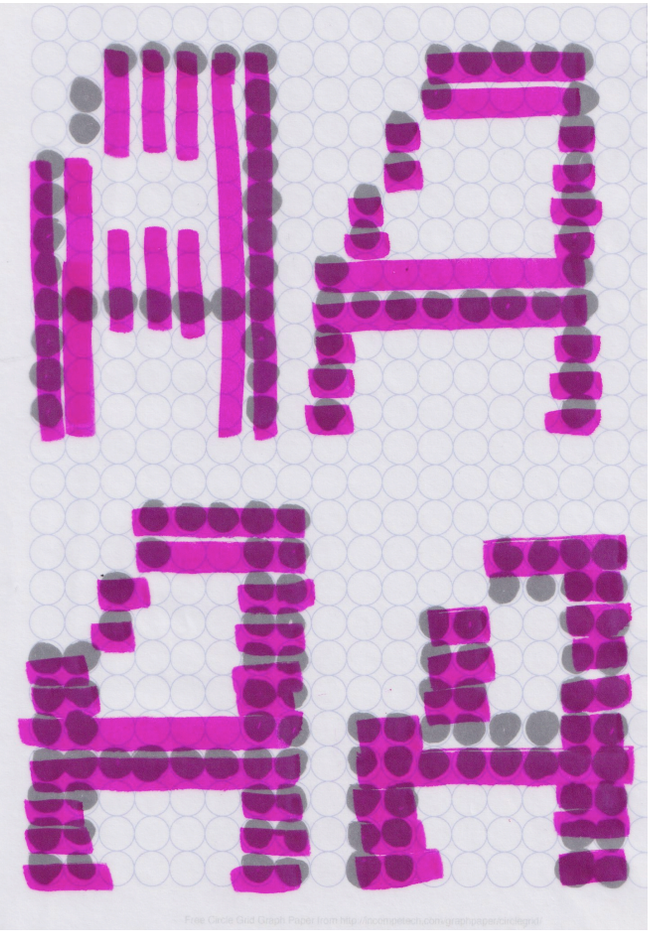




CYBERNETICS

Data-driven experiments introduced feedback and iteration, where typographic behaviour is shaped through relationships between input and output.



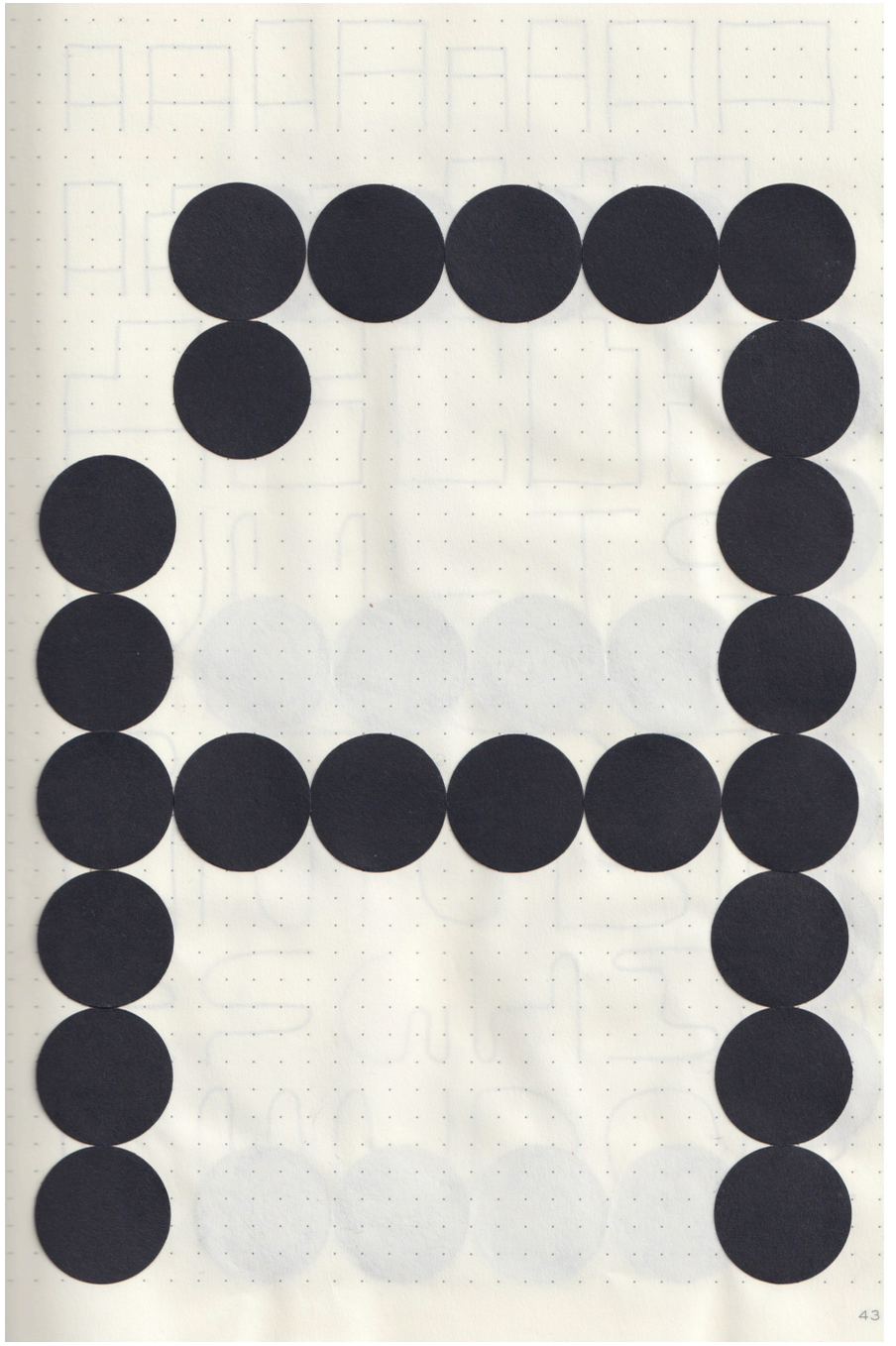


EXPERIMENTAL TYPOGRAPHY

Node-based and iterative sketches investigated repetition, variation, and system behaviour, exploring how increasing distortion and complexity could operate within a rule-based typographic framework.

04

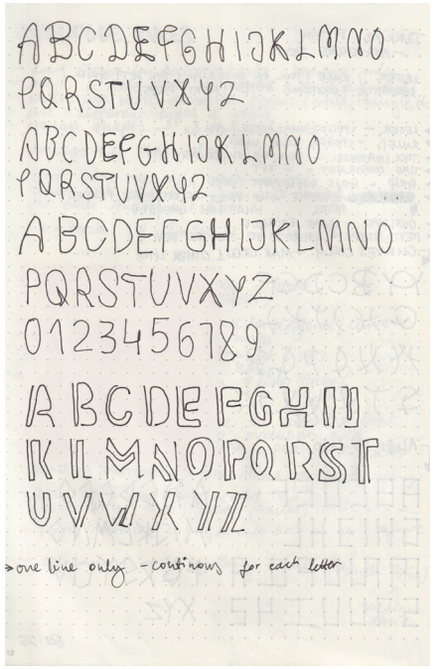
Phase 1
Exploration

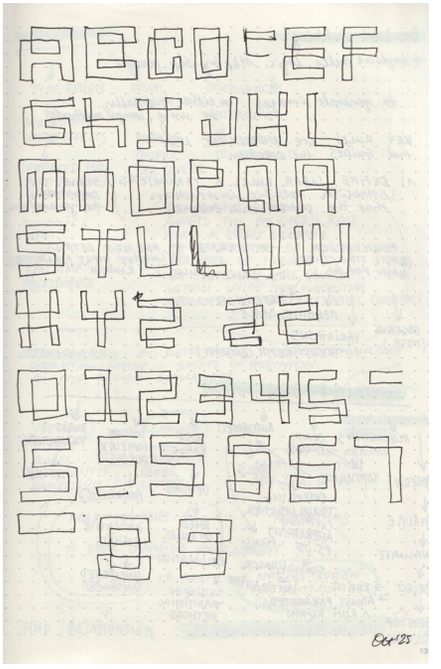


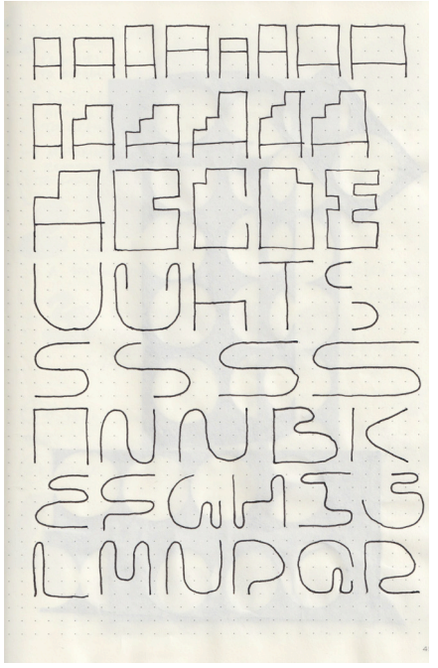
DISCOVERING RULES

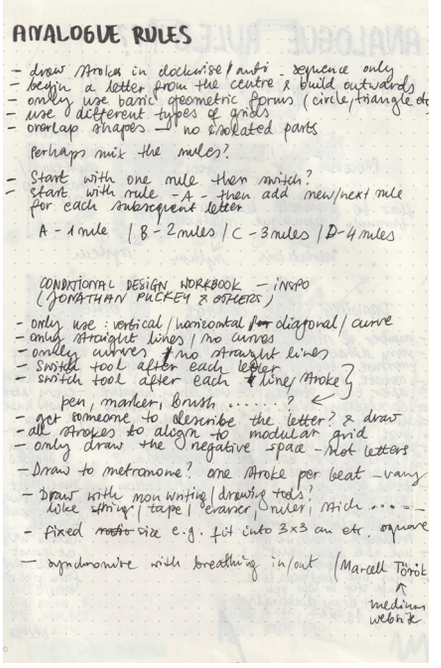
Sequence of progressive exercises were conducted from manual exploration to system development. These experiments established a material foundation for the typographic system, where rules emerged through repetition, constraint, and variation.

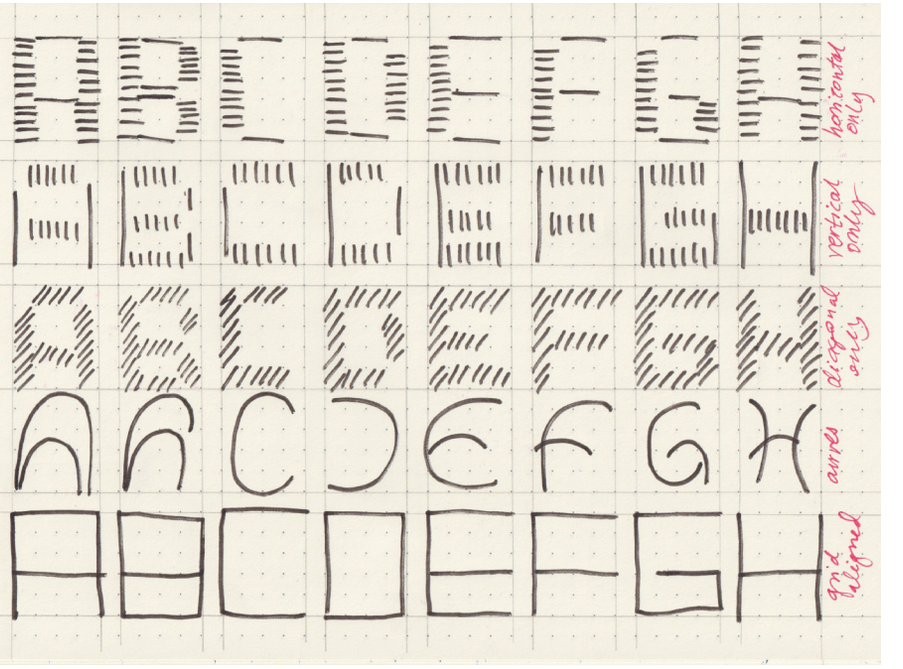
Through sketching and material exploration, structure and behaviour were discovered iteratively. This process ensured the system evolved from physical experimentation to scalable digital logic.



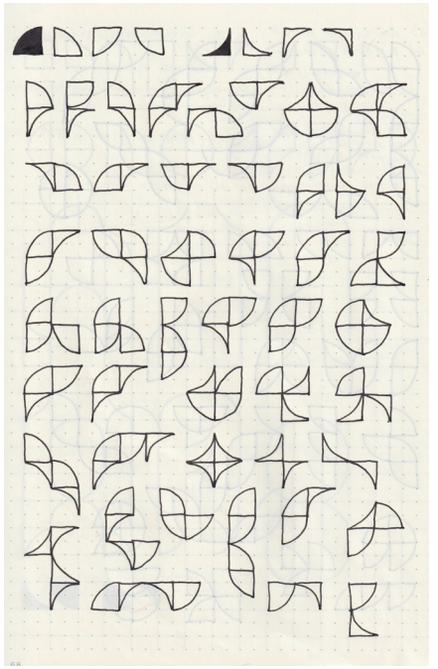


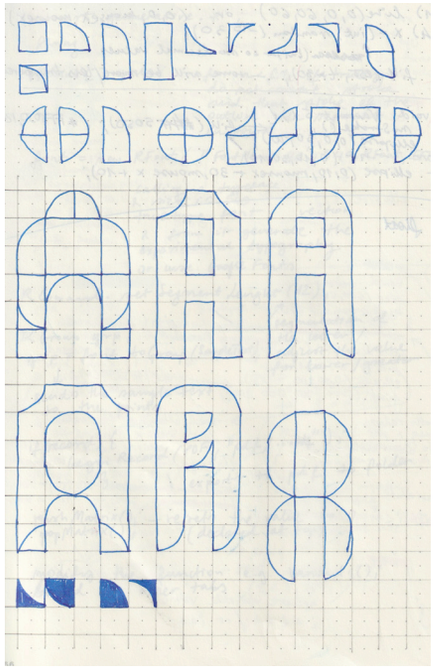


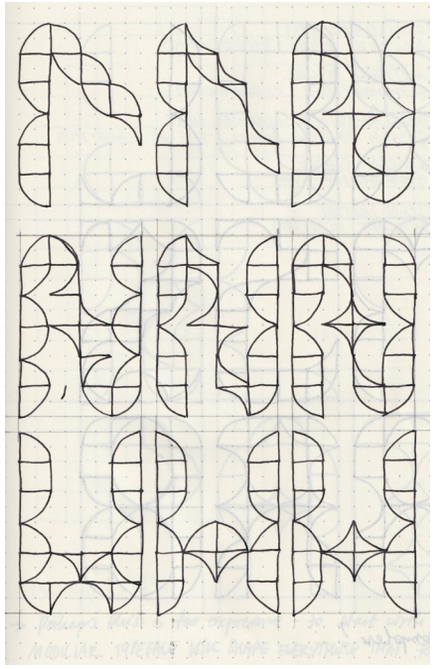


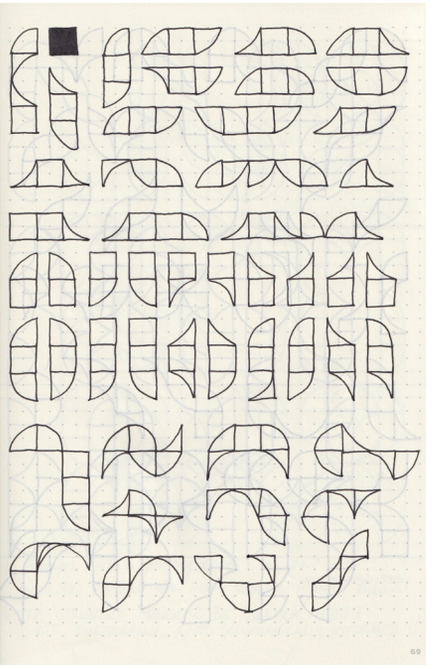


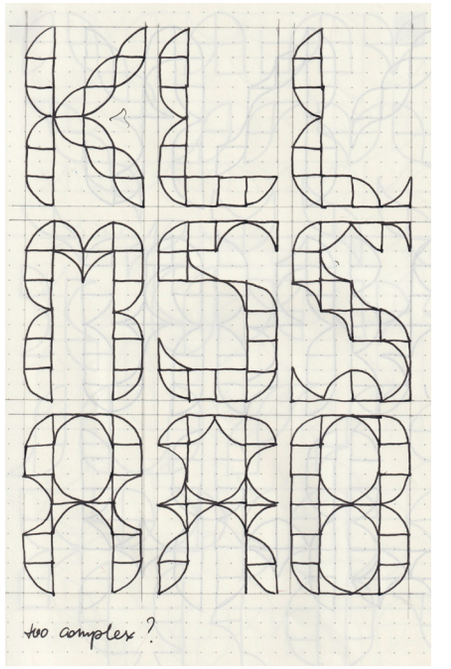
Letterform sketches were generated through analogue constraints, including continuous line drawing and directional limitations. These rules introduced structure into the process, shifting from intuitive drawing towards a controlled system in which typographic form emerges through restriction.

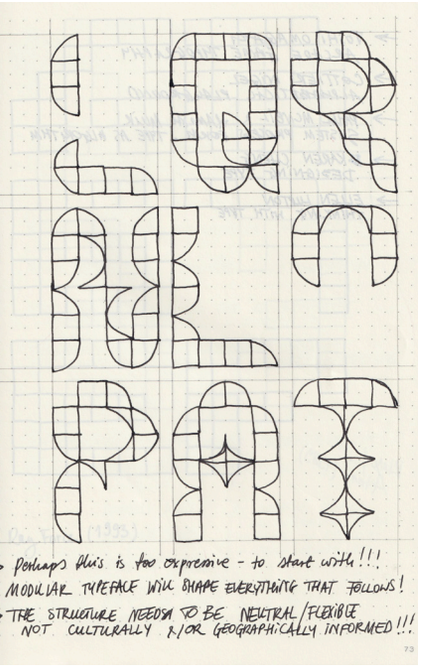






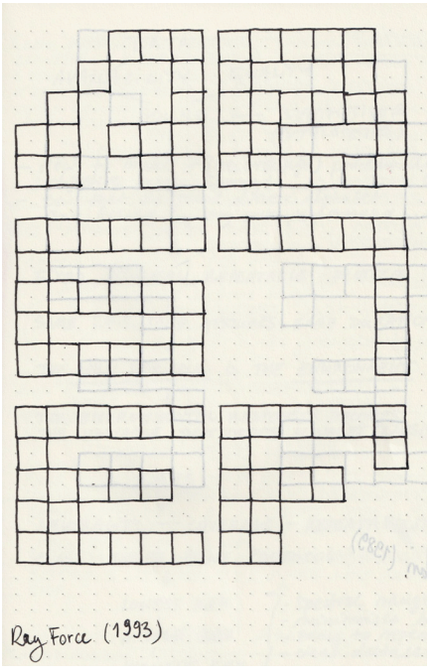


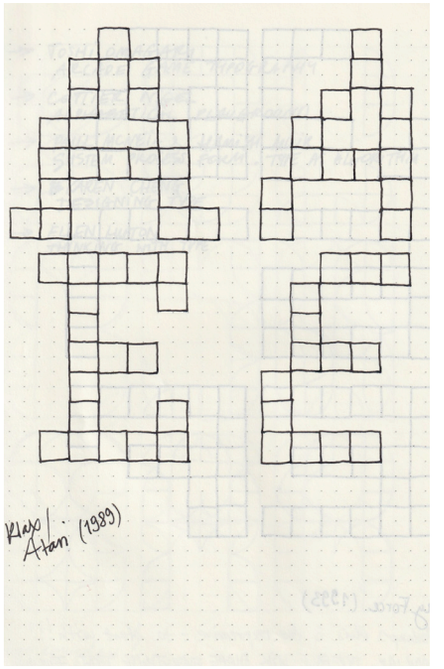


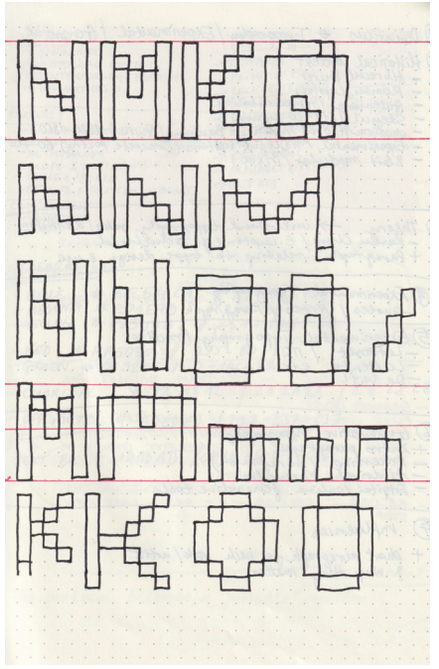


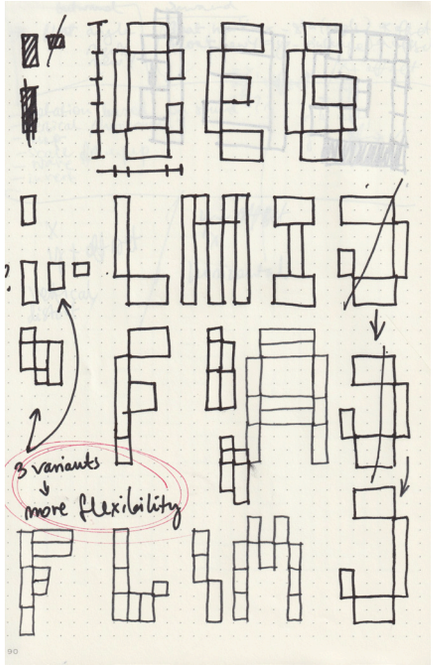
PRIMITIVE SHAPES

Modular experiments explored the use of a limited set of reusable components to construct letterforms. While this approach established consistency, it produced overly expressive and stylistically distinct results. This led to the recognition that the typographic system must remain neutral and flexible, avoiding cultural or geographical influence in order to maintain adaptability across different contexts.









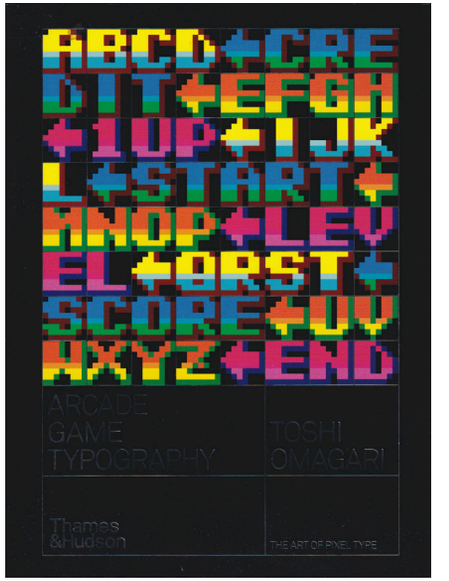


Fig. 1

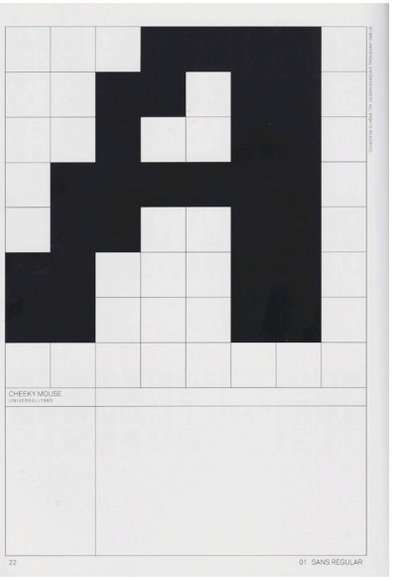


Fig. 2

Fig. 1

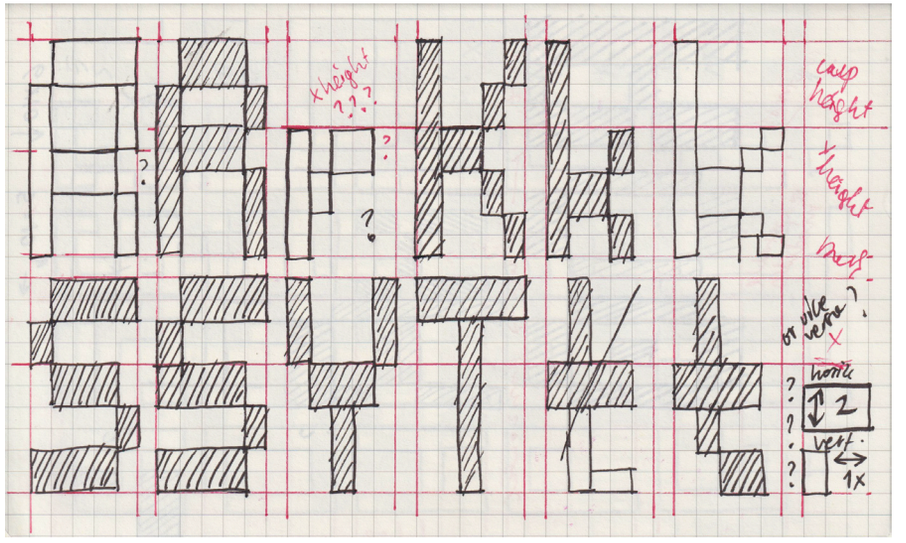
Toshi Omagari (2021) *Arcade Game Typography: The Art of Pixel Type*, London, Thames and Hudson

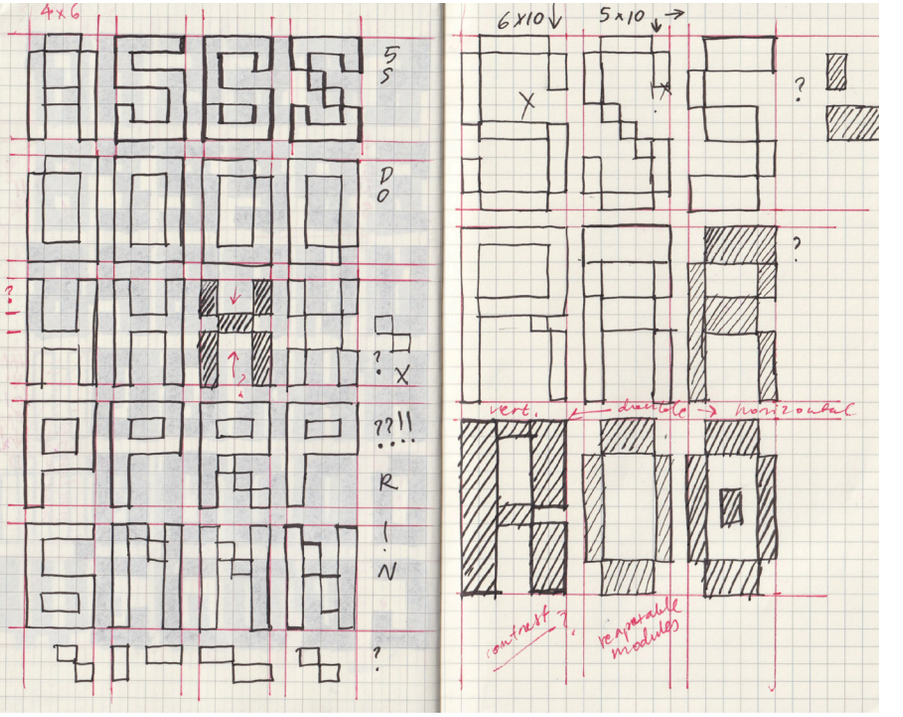
Fig. 2

Cheeky Mouse, Universal (1980)
Sans Regular

[As in: Toshi Omagari (2021) Arcade Game Typography, p. 22]

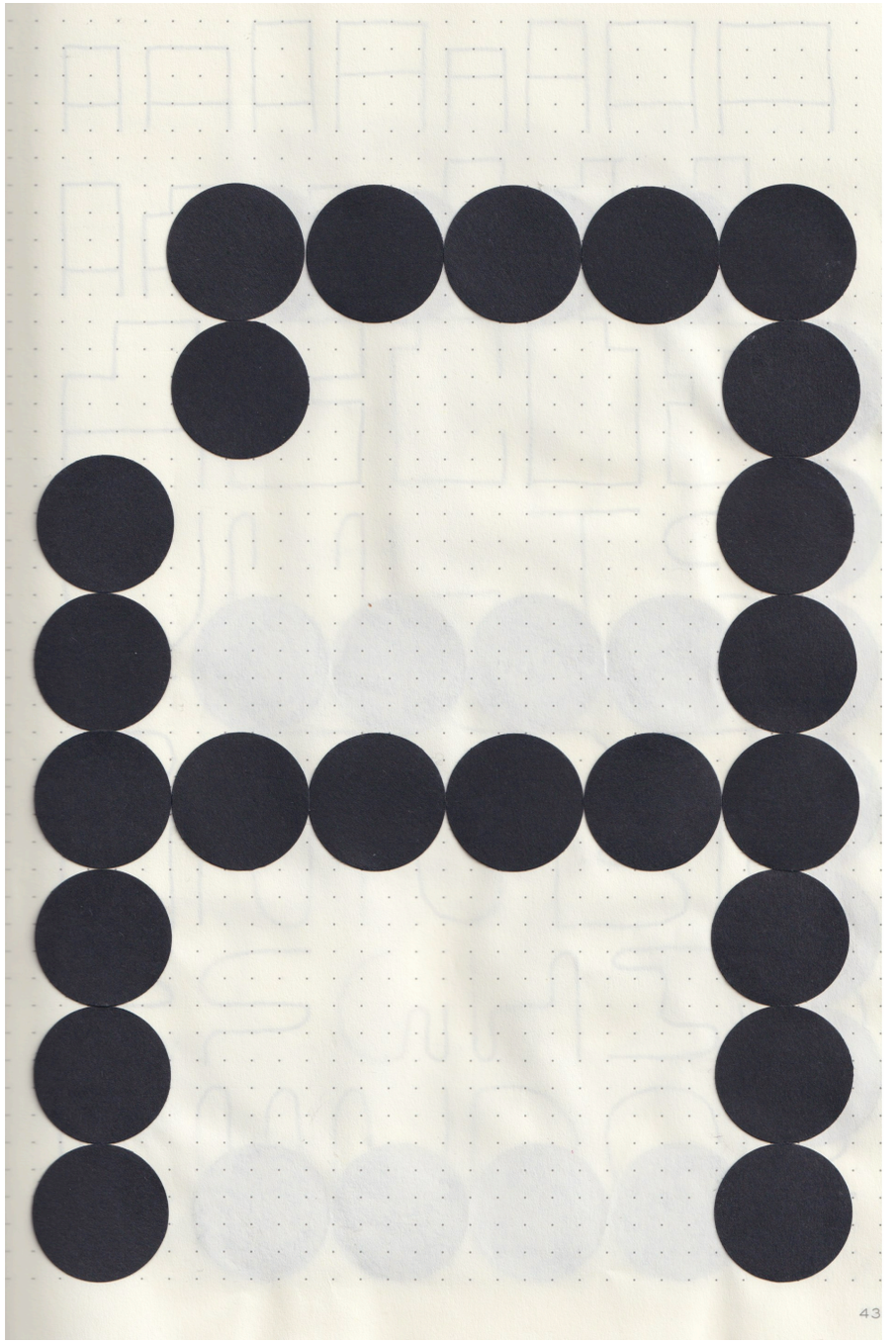
Following the need for a more neutral and adaptable system, pixel-based typography was explored through references such as Toshi Omagari's *Arcade Game Typography*. The modular grid-based construction and restricted visual language provided a clear, consistent framework, reinforcing the use of simple, repeatable units as a foundation for a flexible typographic system.

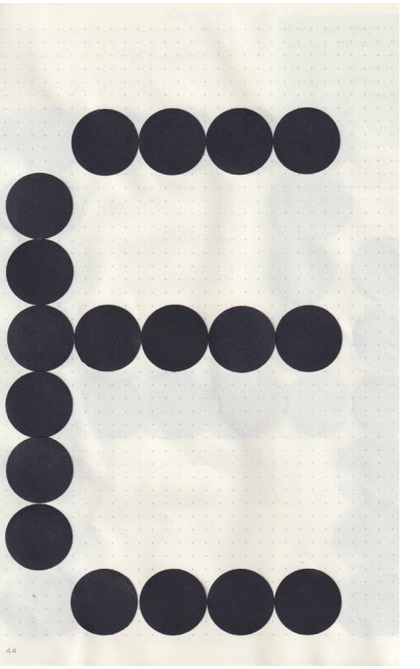


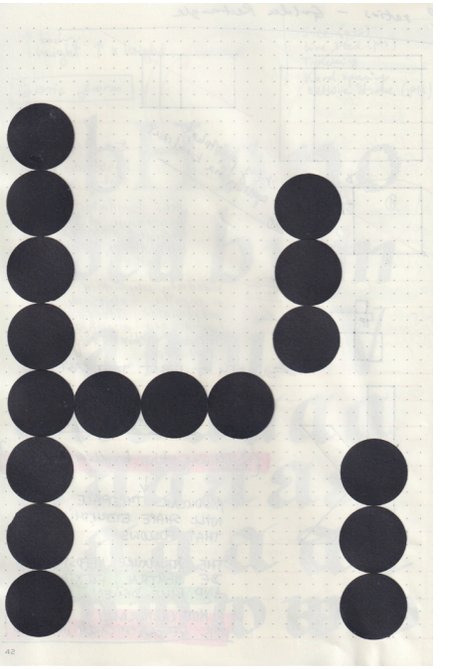


BASELINE AND GRID

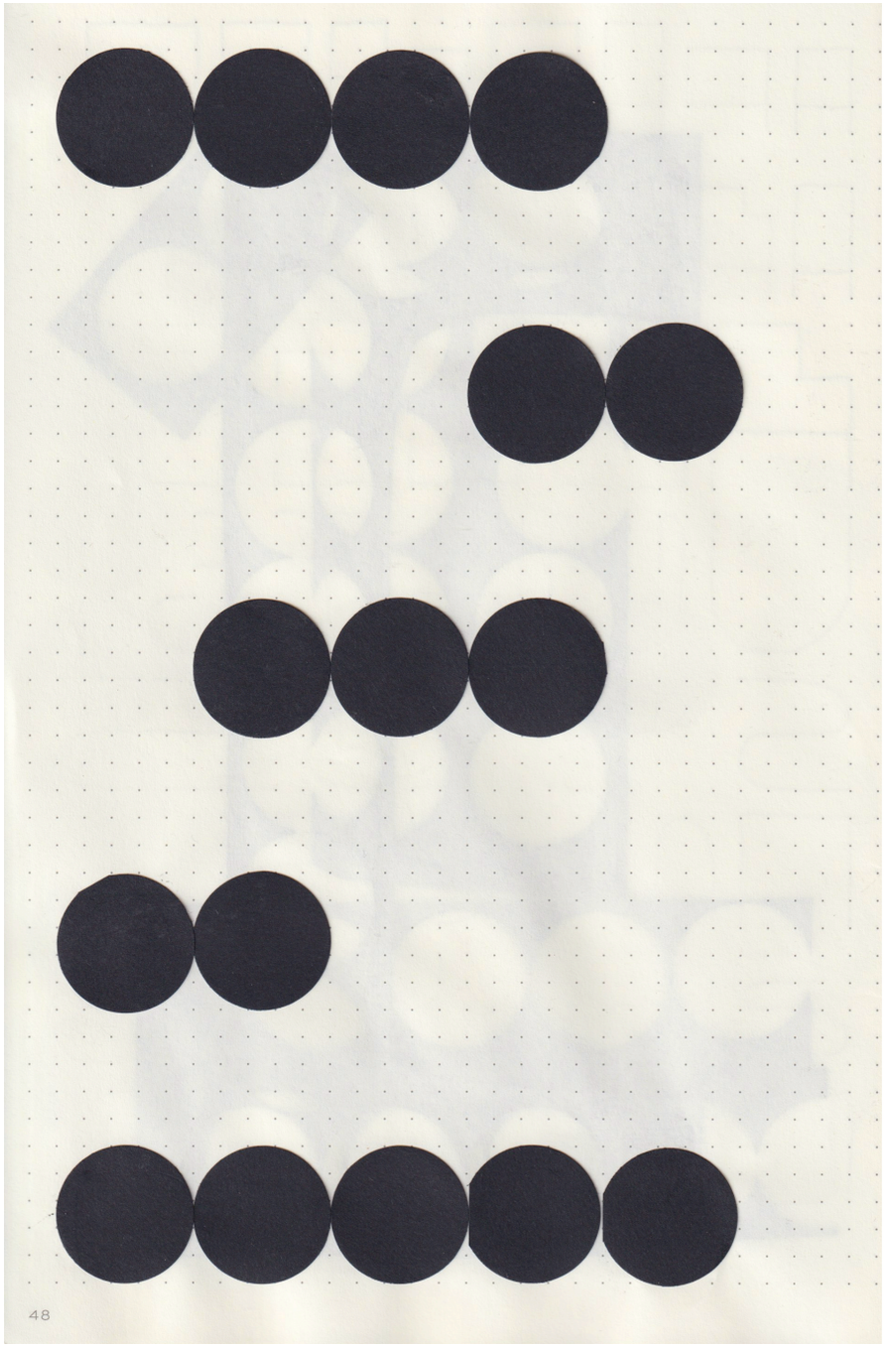
Applying grid introduced consistent alignment and proportional control, allowing variations to be tested within fixed structural constraints while maintaining coherence across the system.

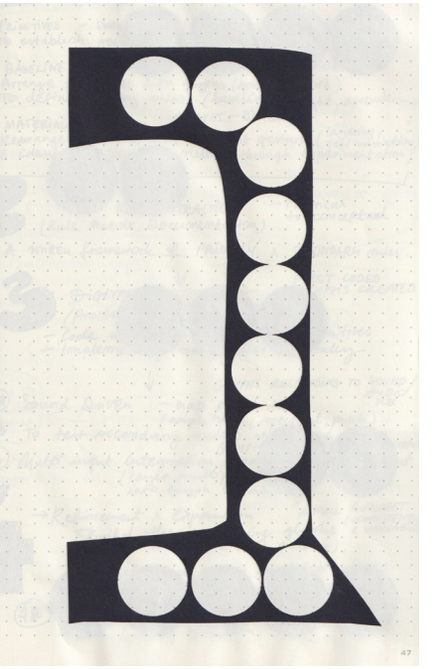


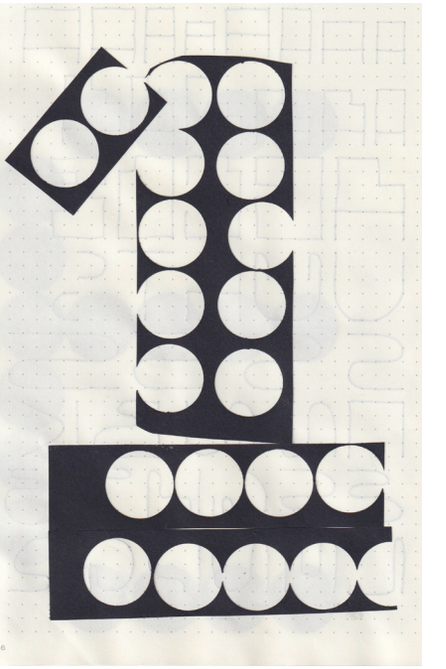




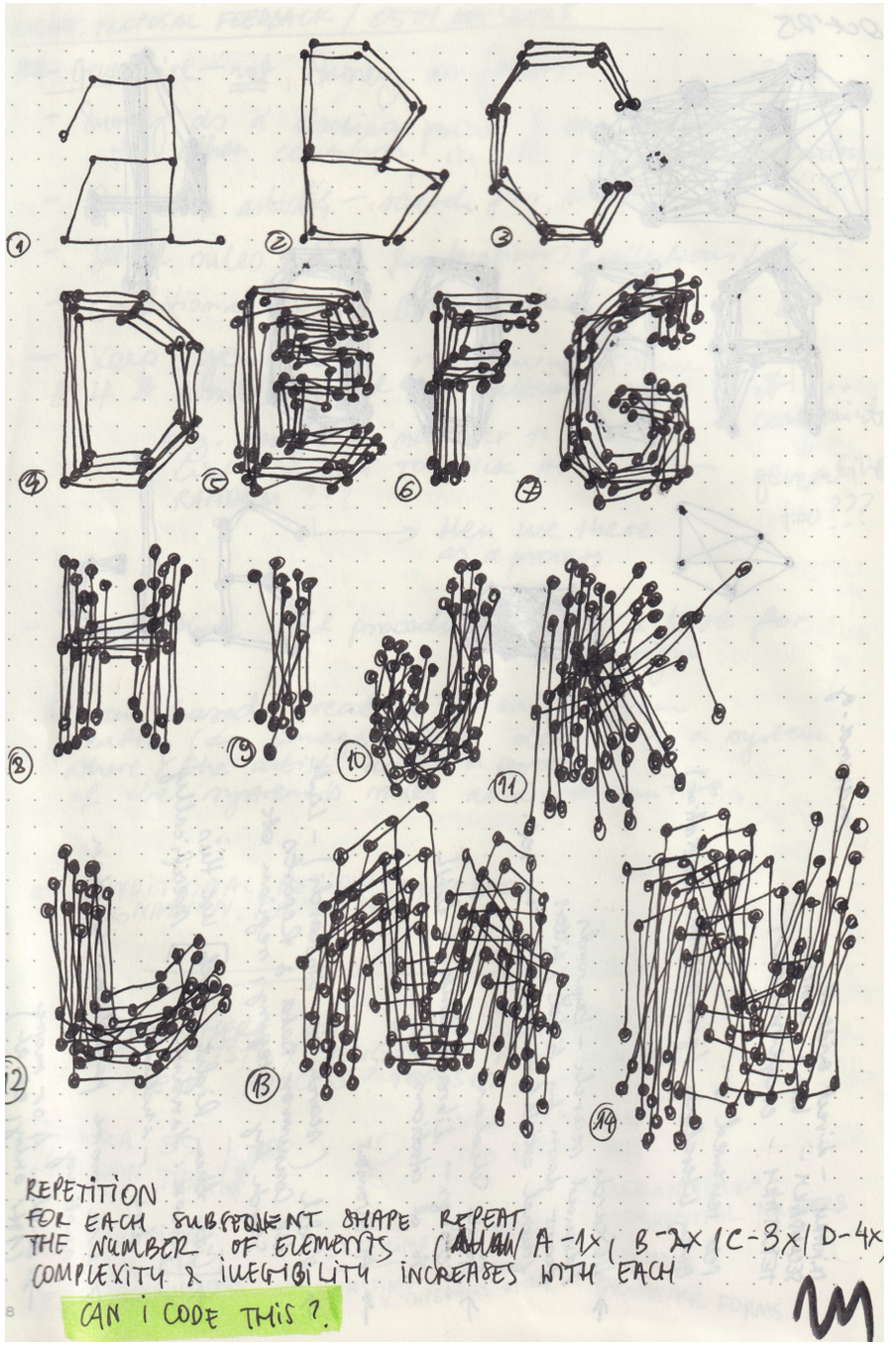
Circular primitives were systematically arranged on a grid to form complete structures through repetition and alignment, establishing consistency through the arrangement of identical modular units.

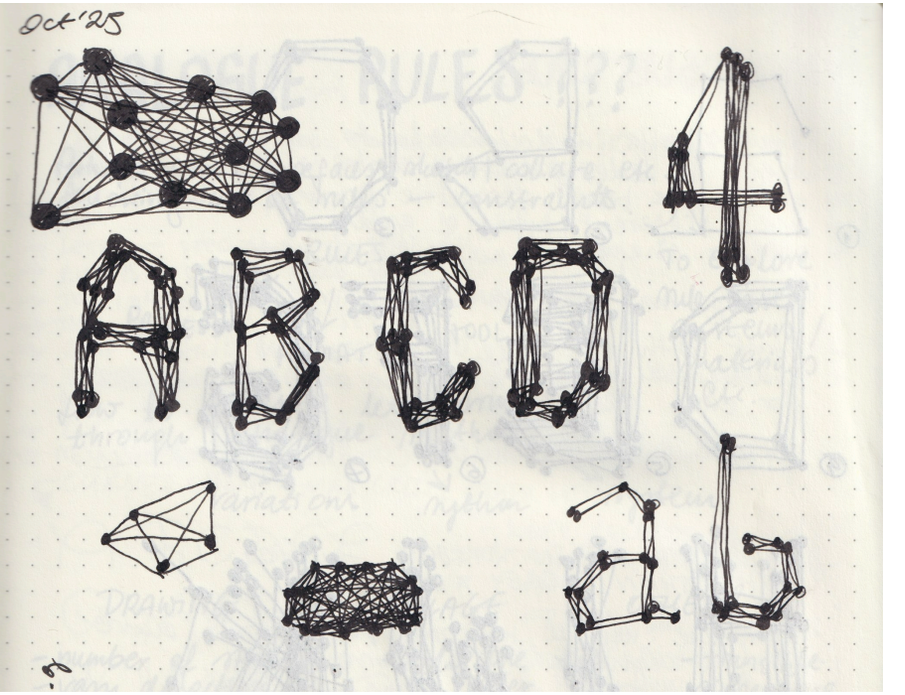






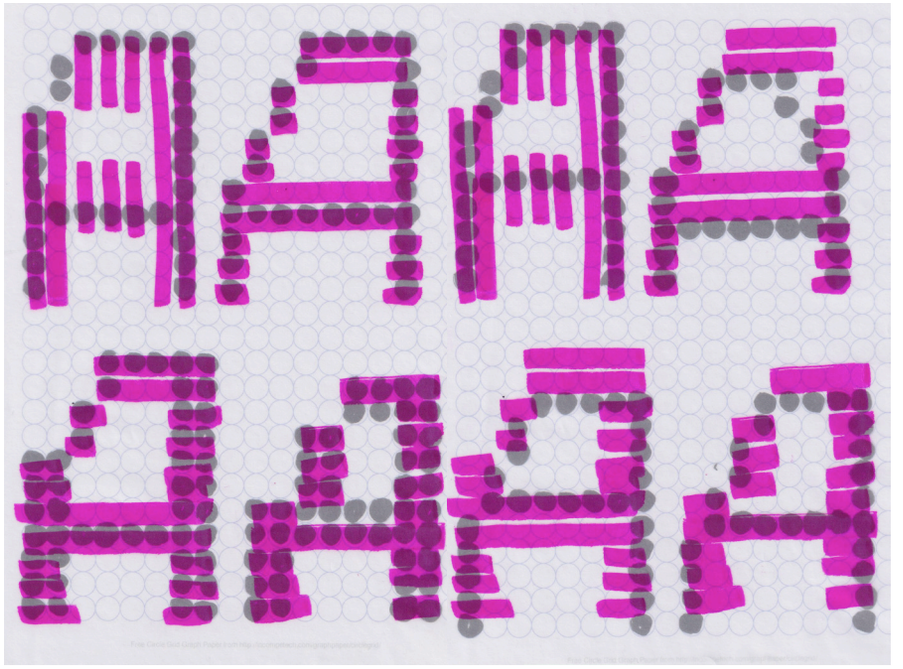
Selective omission of grid elements allowed form to emerge through negative space, introducing a secondary system where absence and spacing become active components in defining typographic structure.

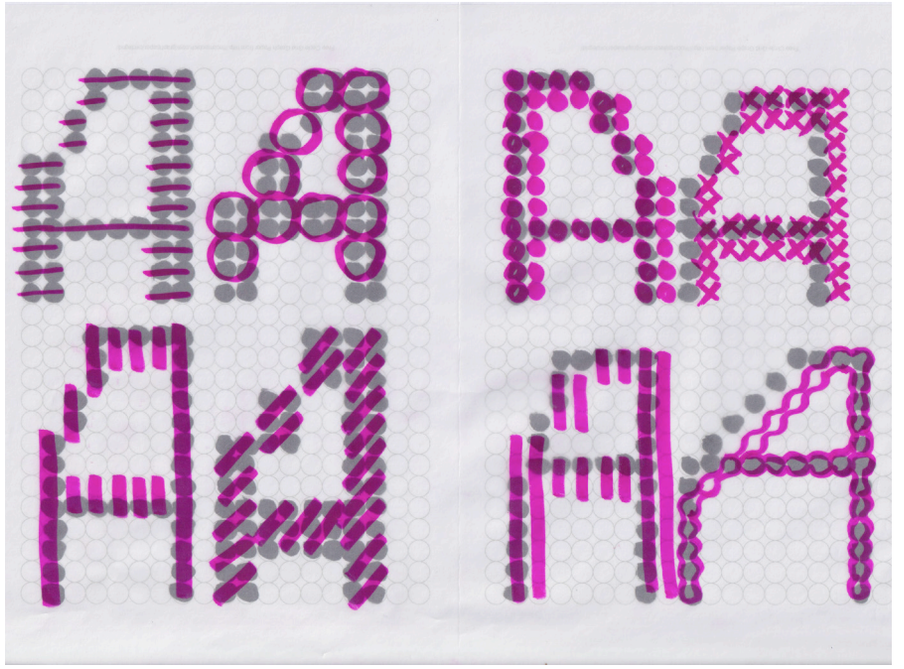


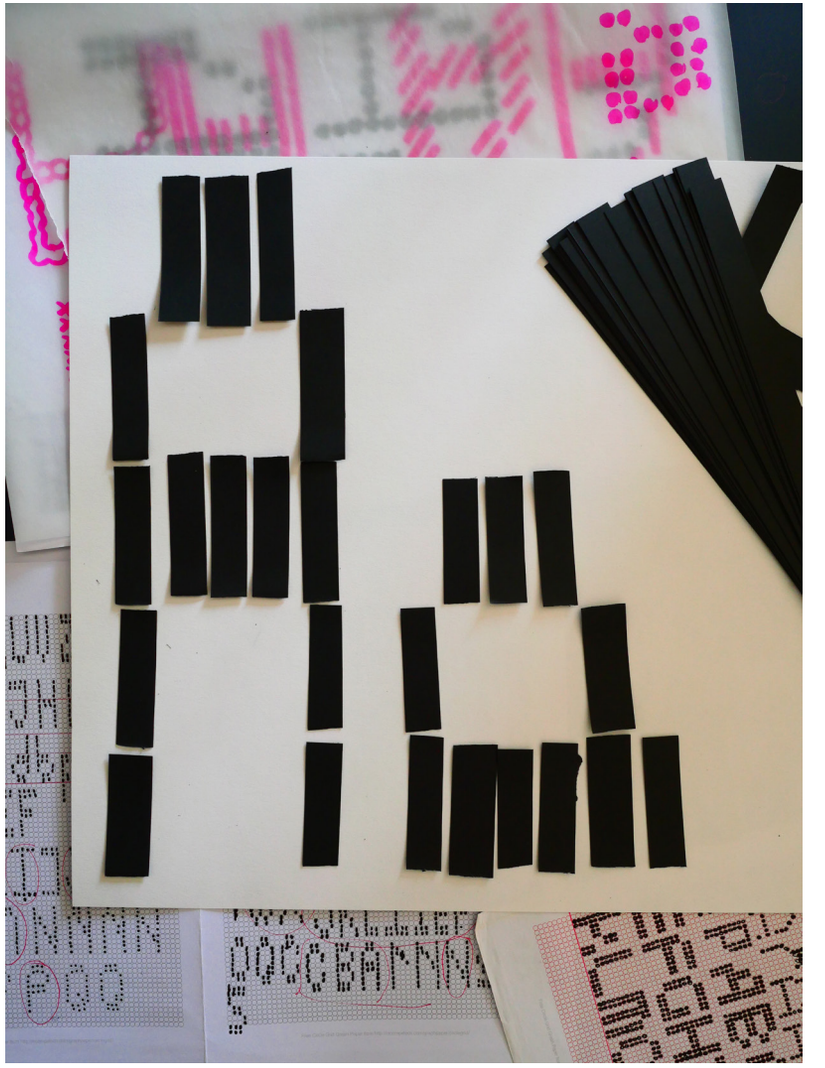


GESTURE AND MATERIALITY

Iterative node-based constructions tested how increasing connections within a system affects form. As repetition and density increased, legibility diminished, highlighting the balance between complexity and clarity and suggesting potential for algorithmic generation.



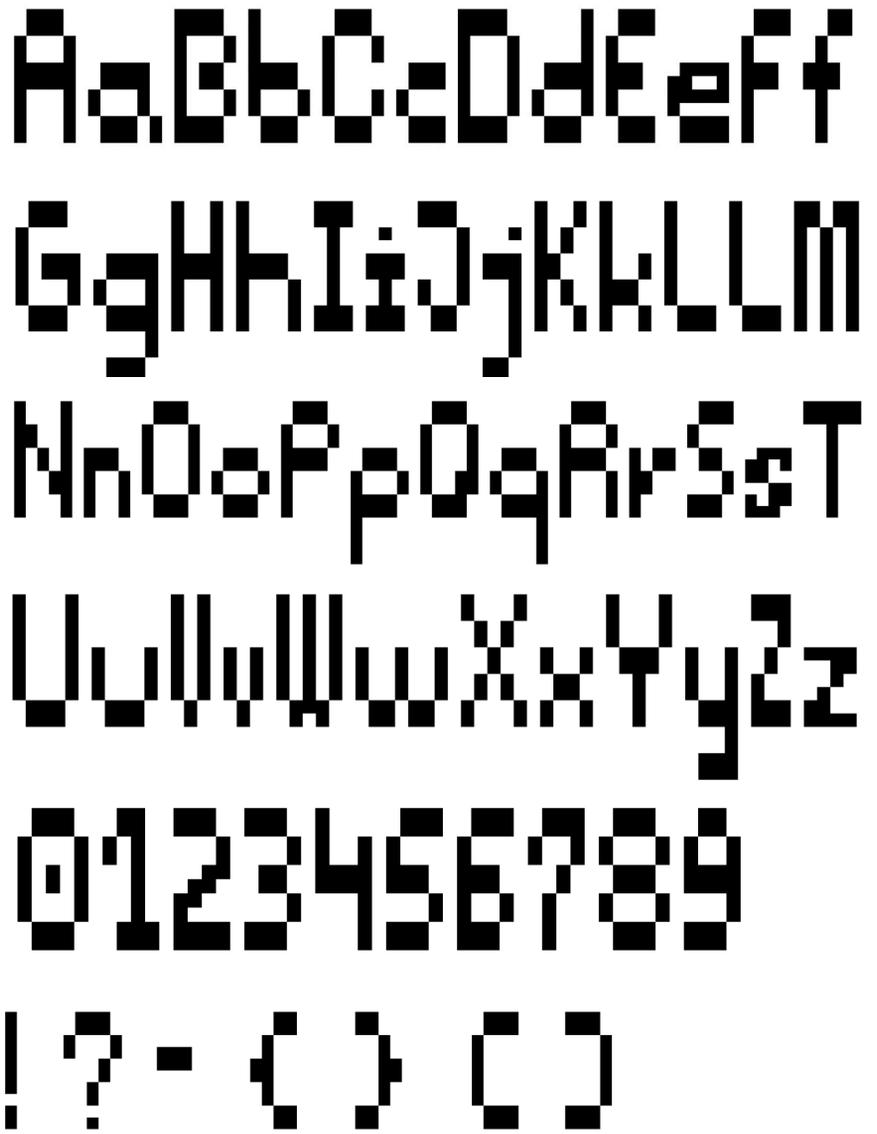




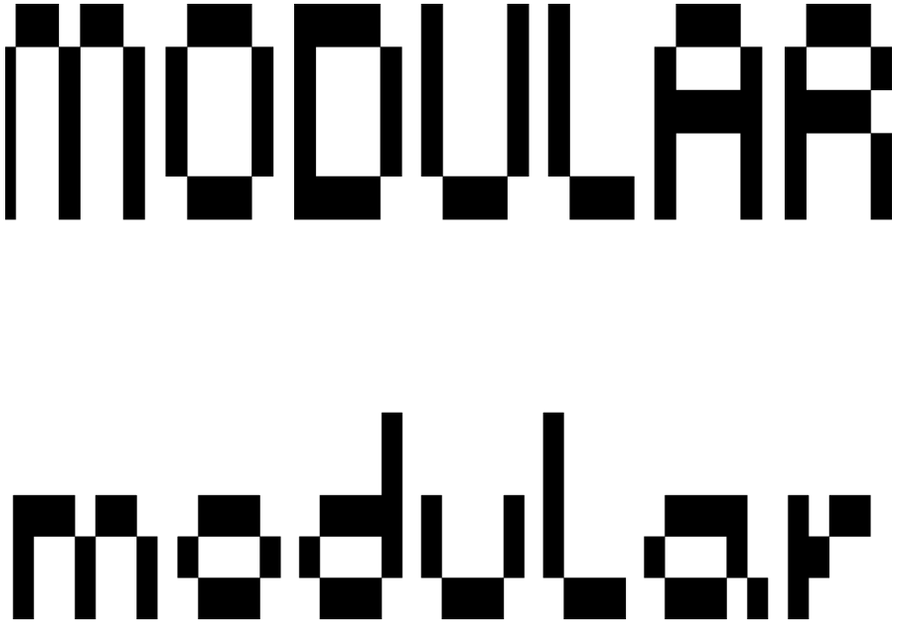
Layered gestures disrupted the underlying grid structure, introducing variation through repetition and misalignment, demonstrating how material interaction and repetition can produce controlled irregularity.

05

Phase 2 System Development



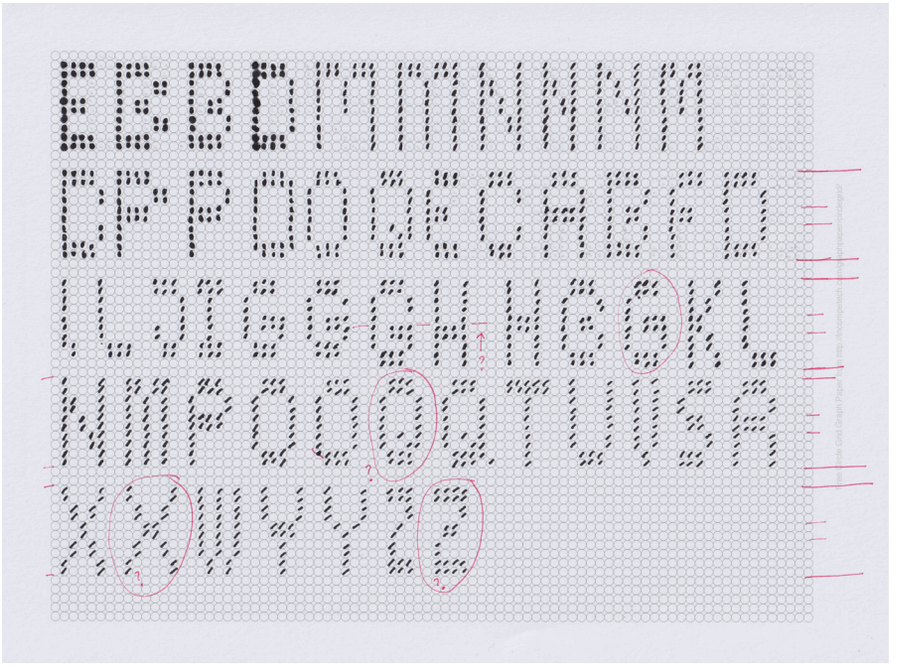
Preview of Pixelfont Square

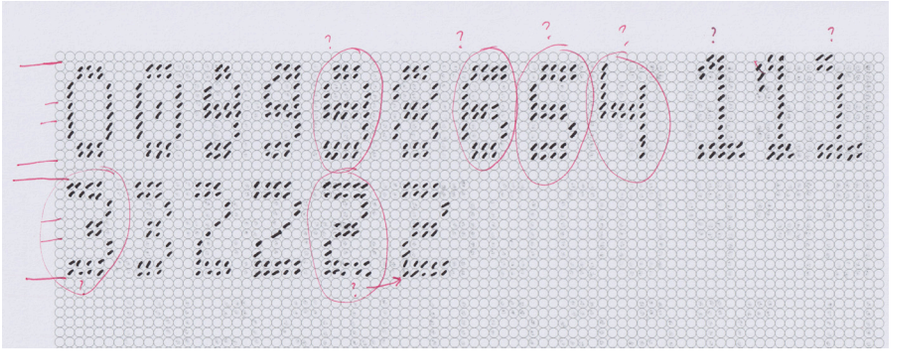


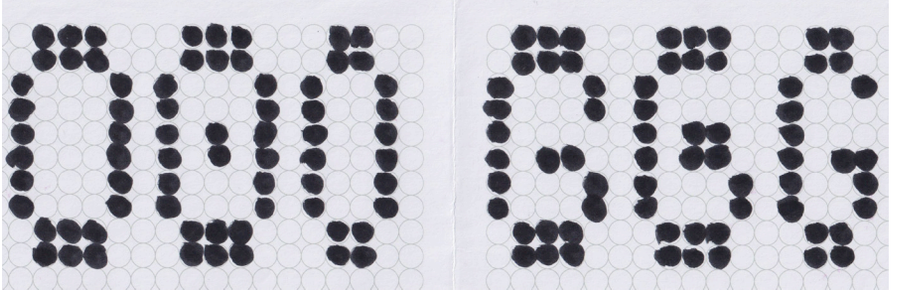
MODULAR TYPEFACE

A custom modular type system was developed as the structural foundation for all outputs. Built from a limited set of geometric modules, letterforms follow fixed rules while allowing controlled variation.

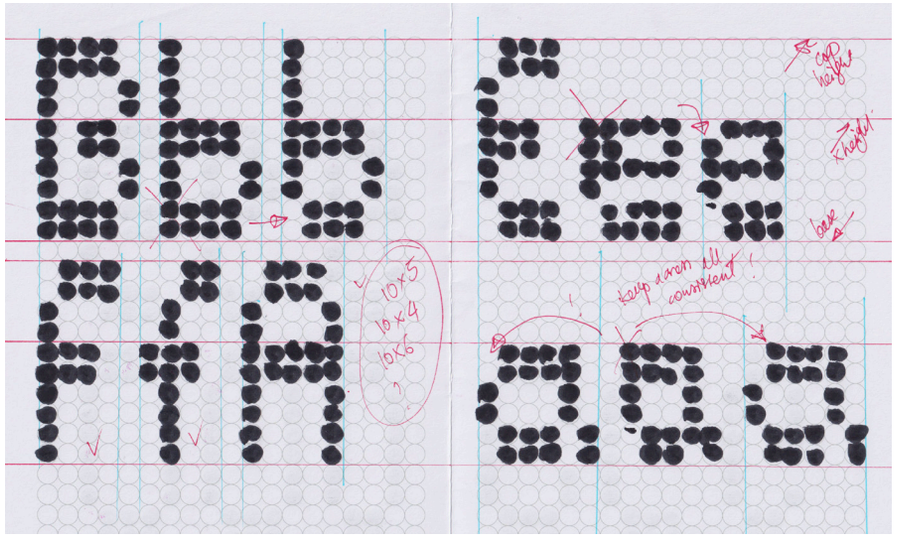
The system prioritises neutrality and consistency, avoiding stylistic influence while remaining flexible. Forms are generated through constraints rather than aesthetic decisions, enabling transformation driven by data.

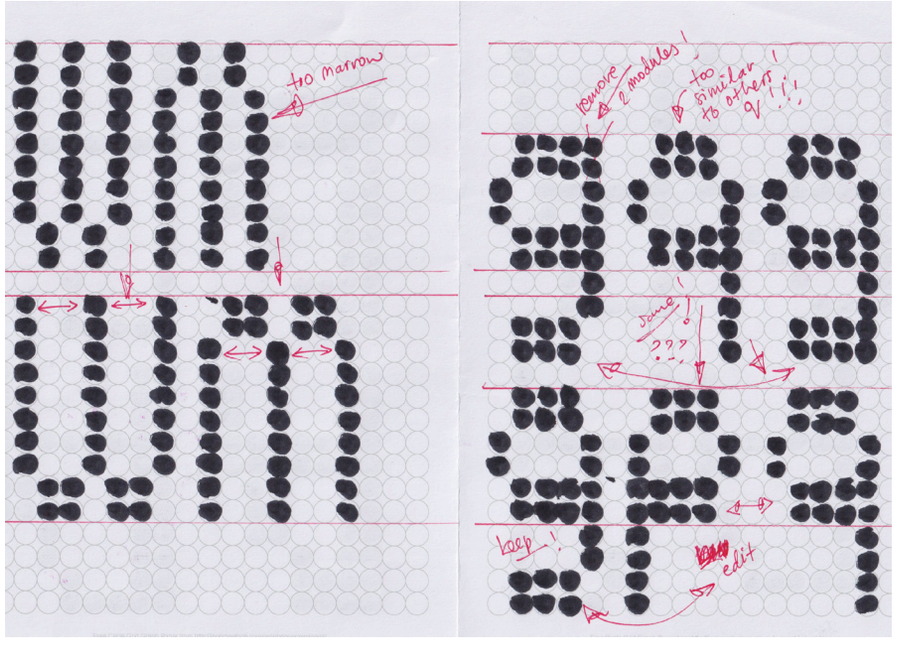




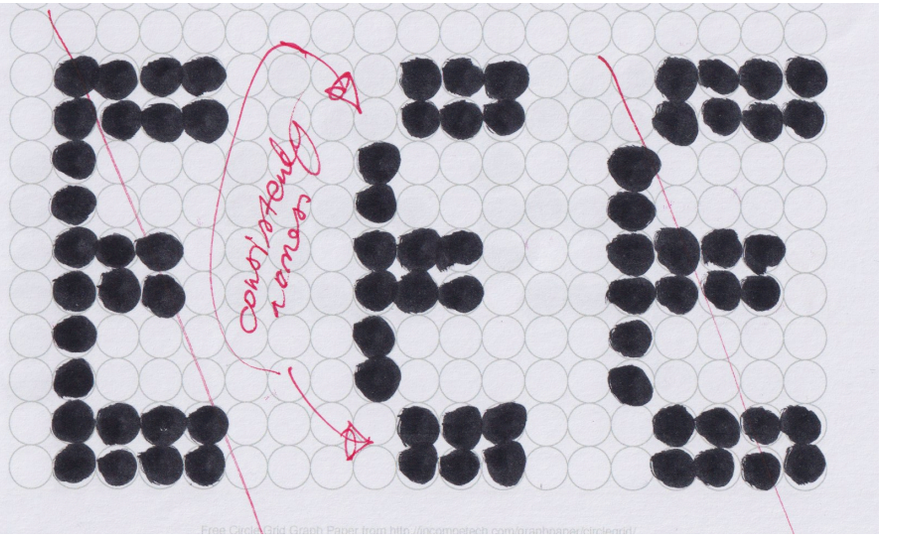


Grid testing across multiple densities revealed inconsistencies in diagonals, spacing, and counters. These findings informed a shift from to a rule-based system. Developed further in Glyphs, the typeface operates through modular constraints, allowing variation to emerge from structure rather than design decisions.



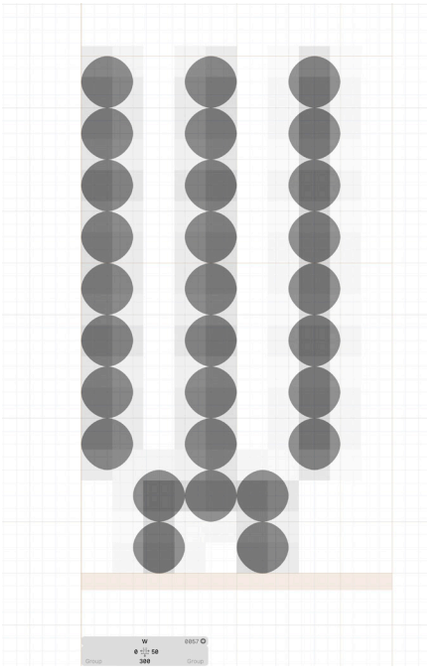


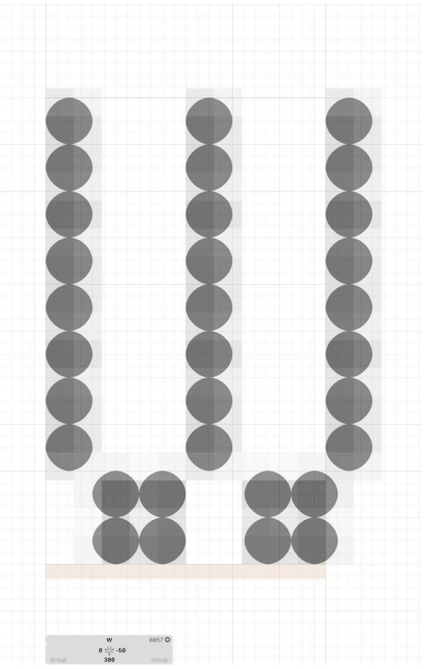
Further comparative testing revealed structural inconsistencies and informed refinements in module placement, forming the basis of a unified system.

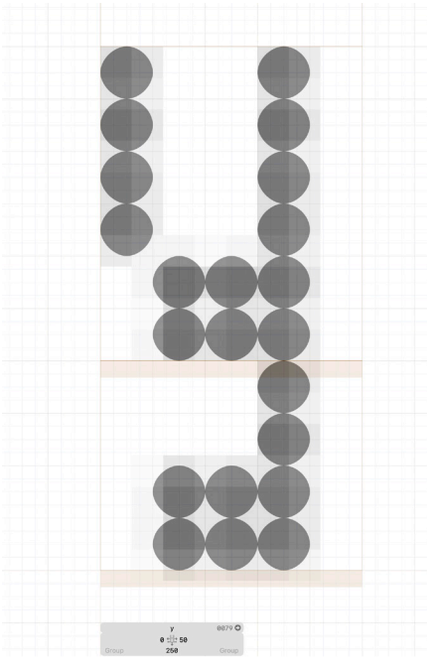


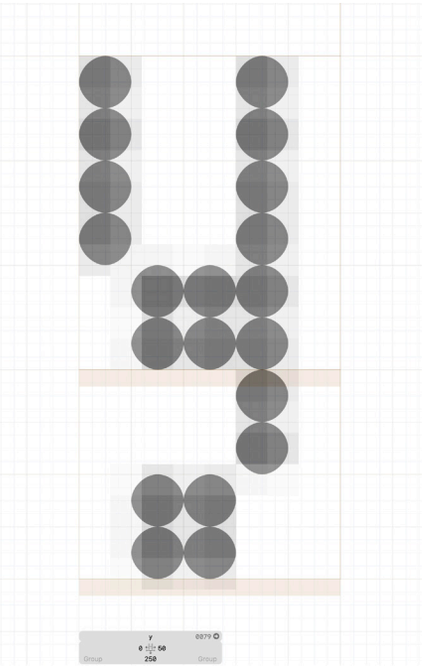
RULES

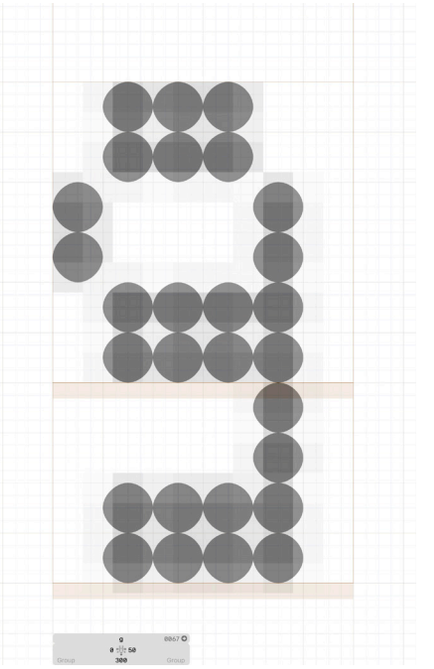
- Fixed modular grid
- Single repeated geometric unit
- Horizontal strokes constructed from two modules
- Minimum stroke thickness of one module
- Consistent x-height and baseline alignment
- Uniform weight and spacing
- Variation through alternate geometric modules

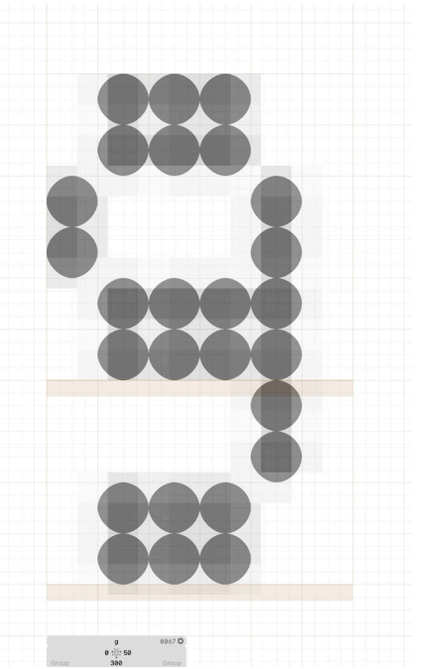








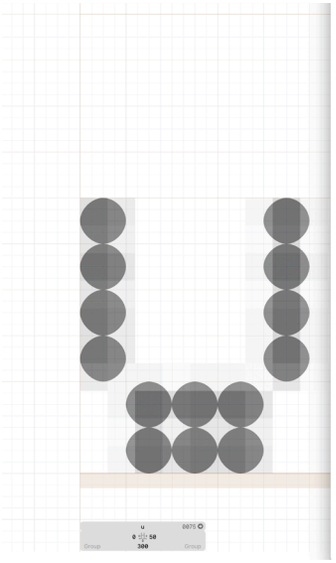


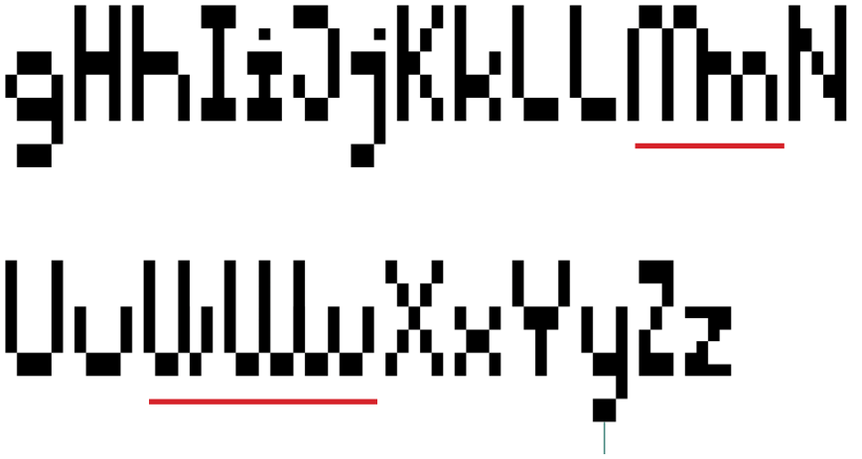


Glyphs

Following the definition of the system rules, letterforms were constructed and refined within Glyphs. This stage focused on translating modular logic into precise, repeatable digital forms.

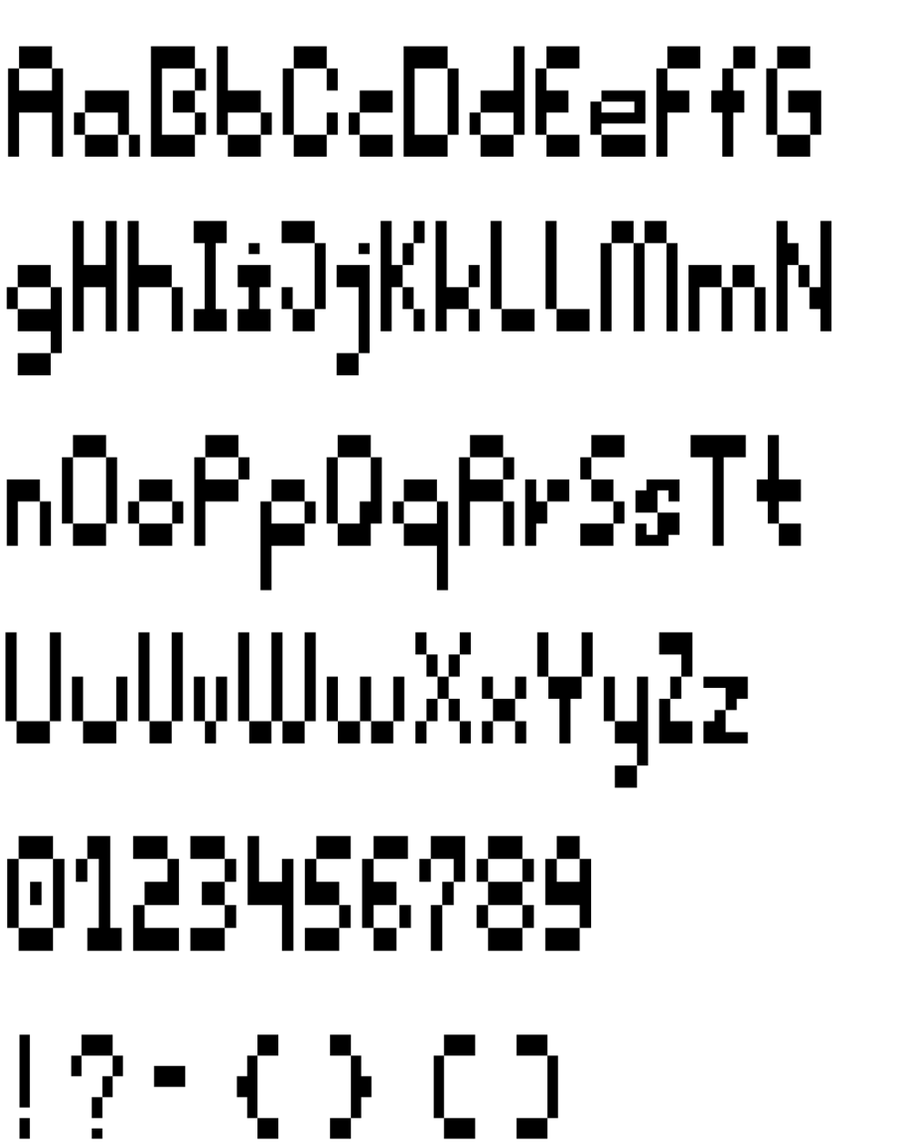
Multiple iterations of individual glyphs were tested to refine alignment, spacing, and proportional relationships





The typeface was tested across the full character set to evaluate overall cohesion and spacing consistency. Underlined text above shows incorrect spacing between characters. Refinements were made to improve balance.

Adjustments improved visual balance and consistency, ensuring even rhythm across all letterforms.

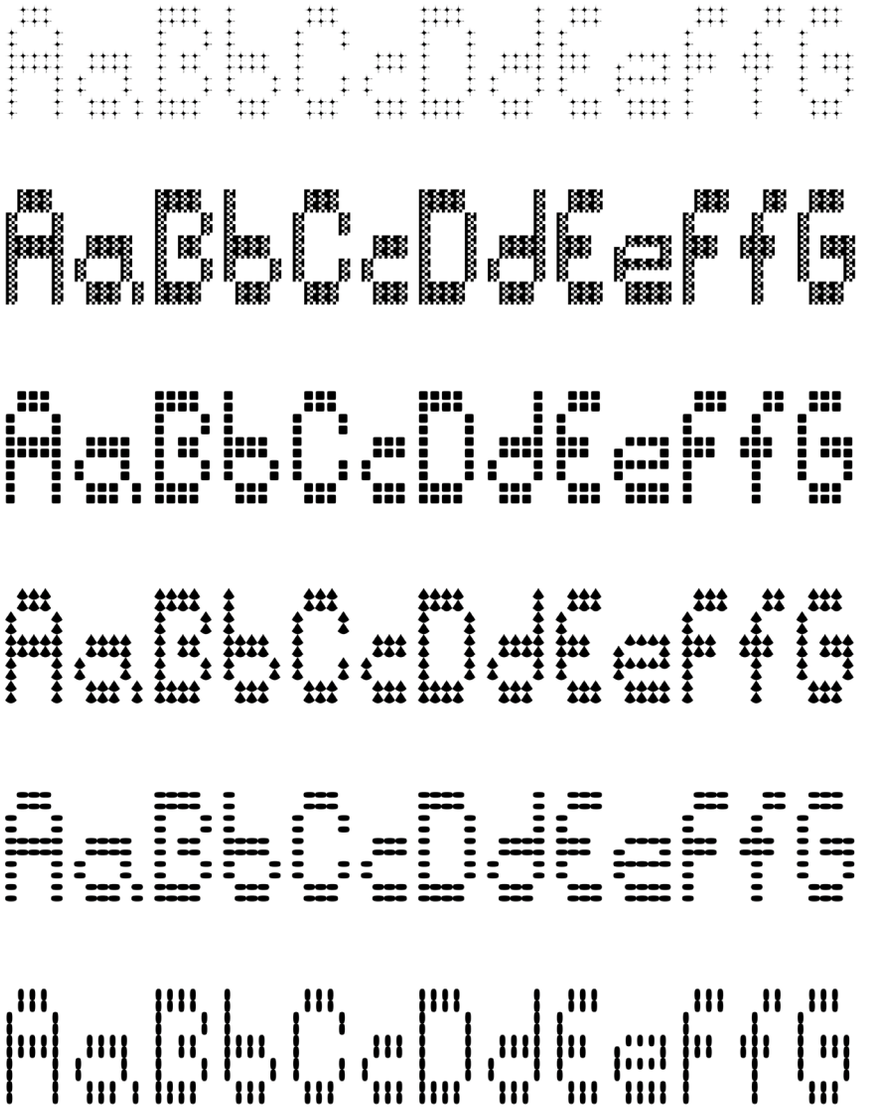


Pixelfont preview
Pixelfont-Square variant

Challanges:
Hinting and overlaping shapes on export.
Resolved:
Removed overlap and no autohint.

Pixelfont variants from top:

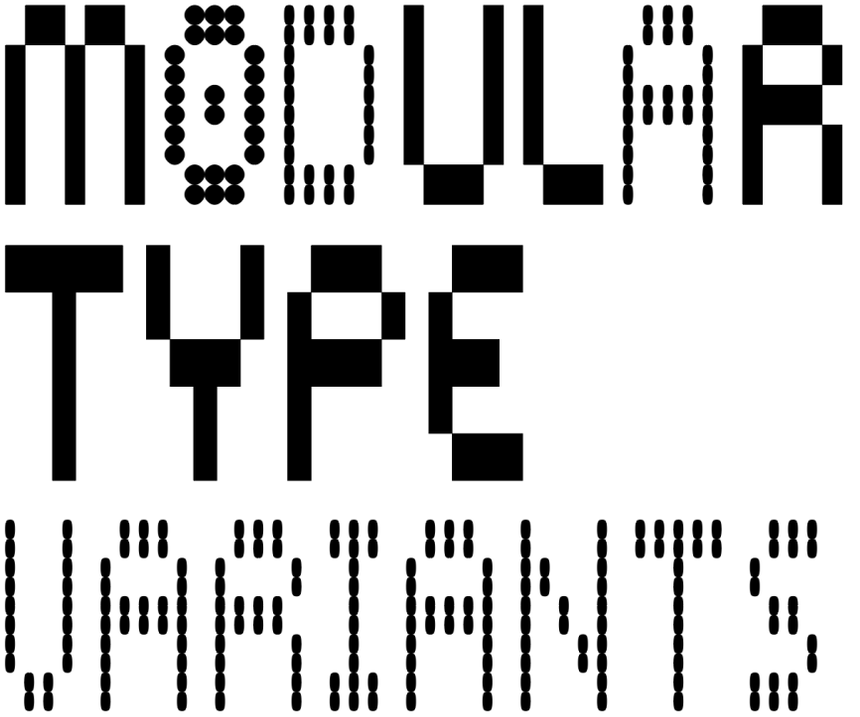
Circle, Square, Check, Half Square, Dot, Line



Pixelfont variants from top:

Star, Pattern, Rounded, Triangle, Flat, Tall

Pixelfont Square:
leading and spacing tests



Application and composition testing

Initial tests of the exported modular typeface were conducted in InDesign to evaluate its behaviour in text. Variants were applied across different scales and compositions to assess legibility, spacing, and visual rhythm.

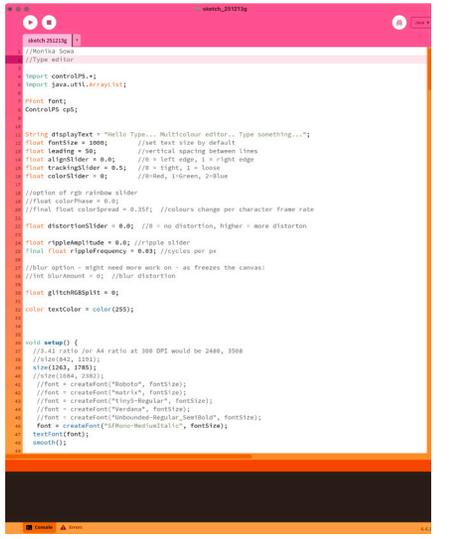
06

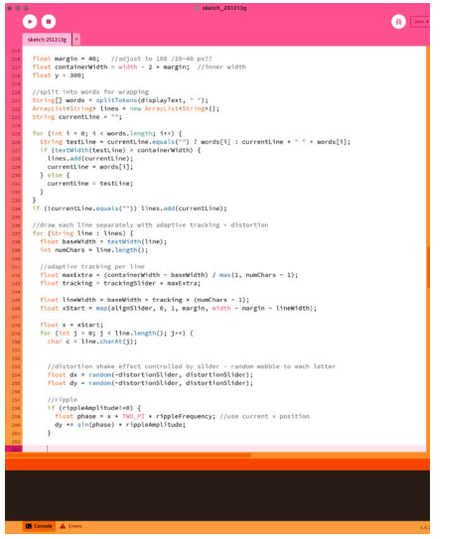
Phase 3

Iteration and Variation

Preview of the tool and interactive output generated through Processing controls (left)

Controls:
font size, alignment, tracking, leading, colour, distortion, ripple, RGB split, background colour.





Code excerpts
(top)

PROCESSING:
TYPE EDITOR TOOL

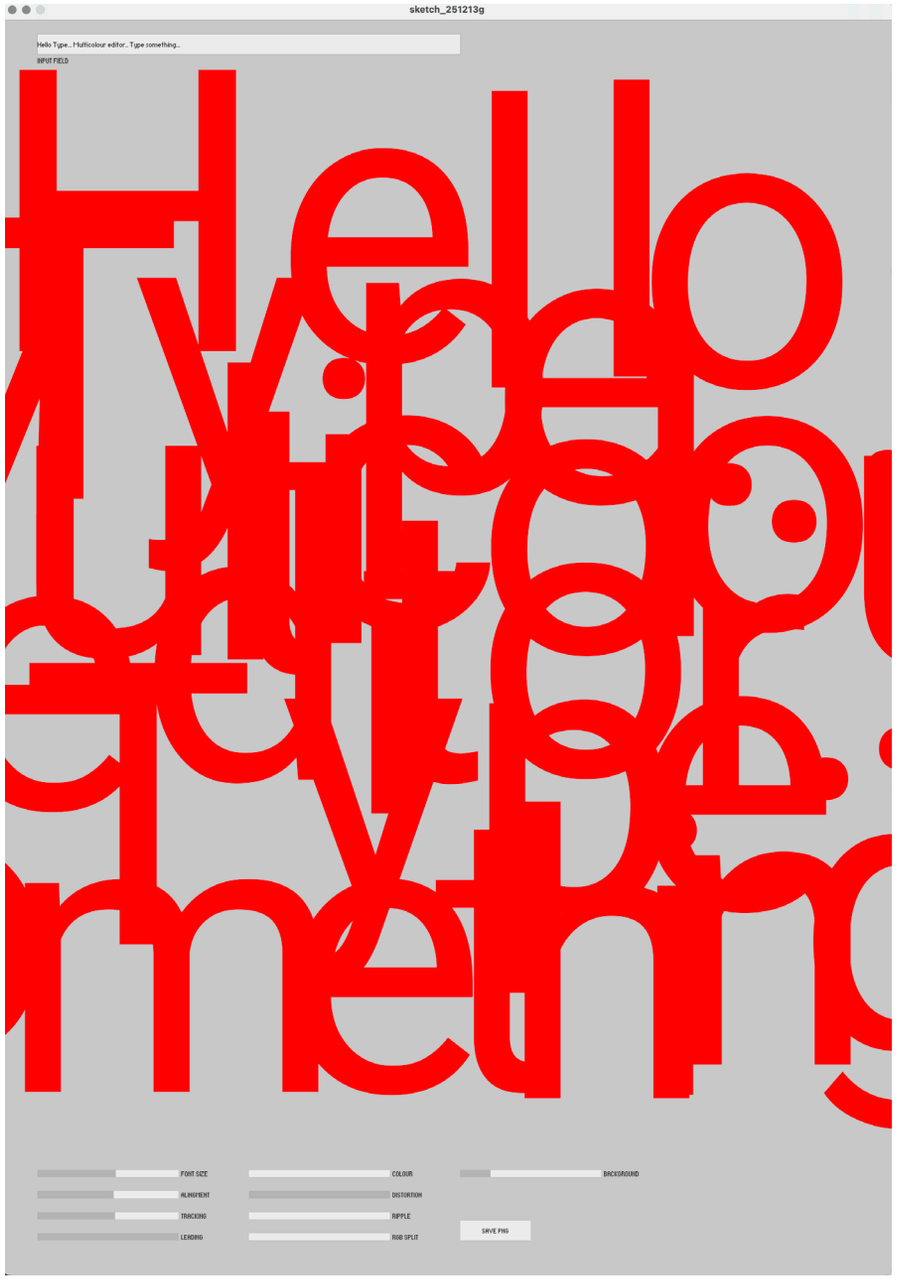
A custom type editor was developed in Processing to extend typography into a dynamic, interactive environment. The tool enables real-time manipulation of text through parameters such as size, tracking, alignment, distortion, and colour.

This exploration translates the static typeface into a generative system, where user input and coded behaviours directly influence typographic output. It demonstrates how glyph data can be interpreted computationally, enabling variation, transformation, and responsive typographic behaviour.

Controls used:
Left aligned, maximum tracking and leading.
(left)

Controls used:
Left alighed, maximum tracking and leading, colour change and slight distortion.
(top)

Typeface:
SFMono Medium Italic



Controls used:
Centre aligned, medium tracking, high leading, high distortion, background change. (left)

Controls used:
Left aligned, low tracking, maximum leading, slight ripple distortion.

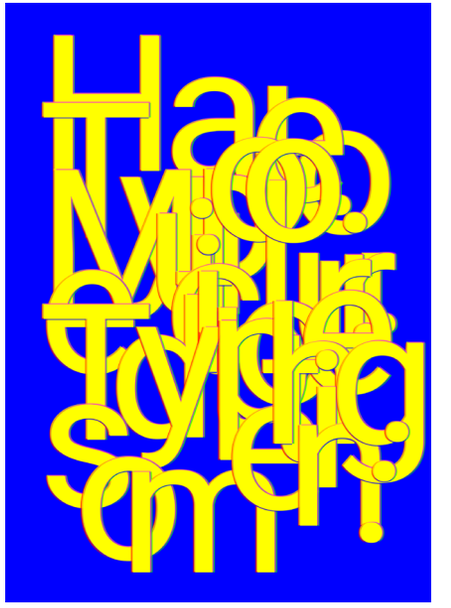
(top)

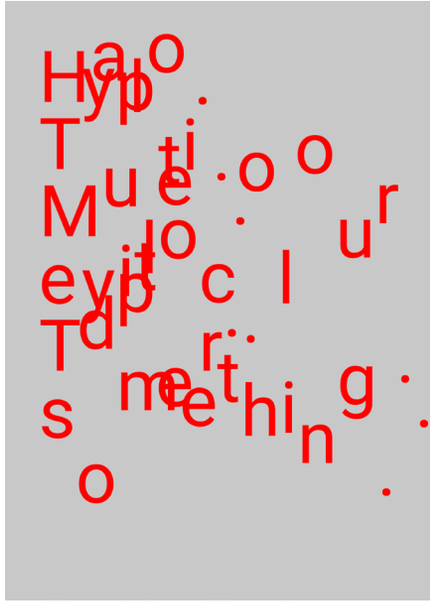
Typeface:
Roboto

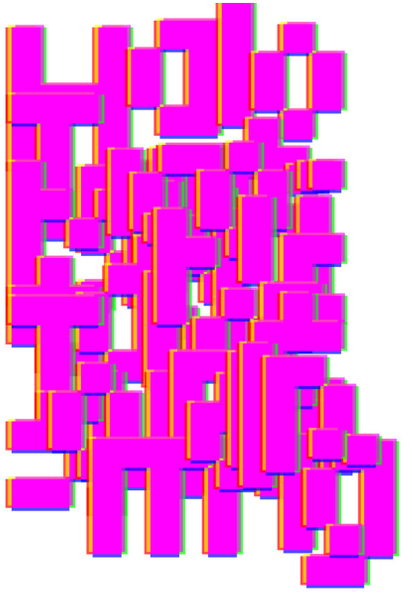
Controls used:
Left aligned, maximum tracking and leading, background change.
(left)

Controls used:
Centre aligned, lowest tracking, maximum leading.
(top)

Typeface:
tiny5-Regular



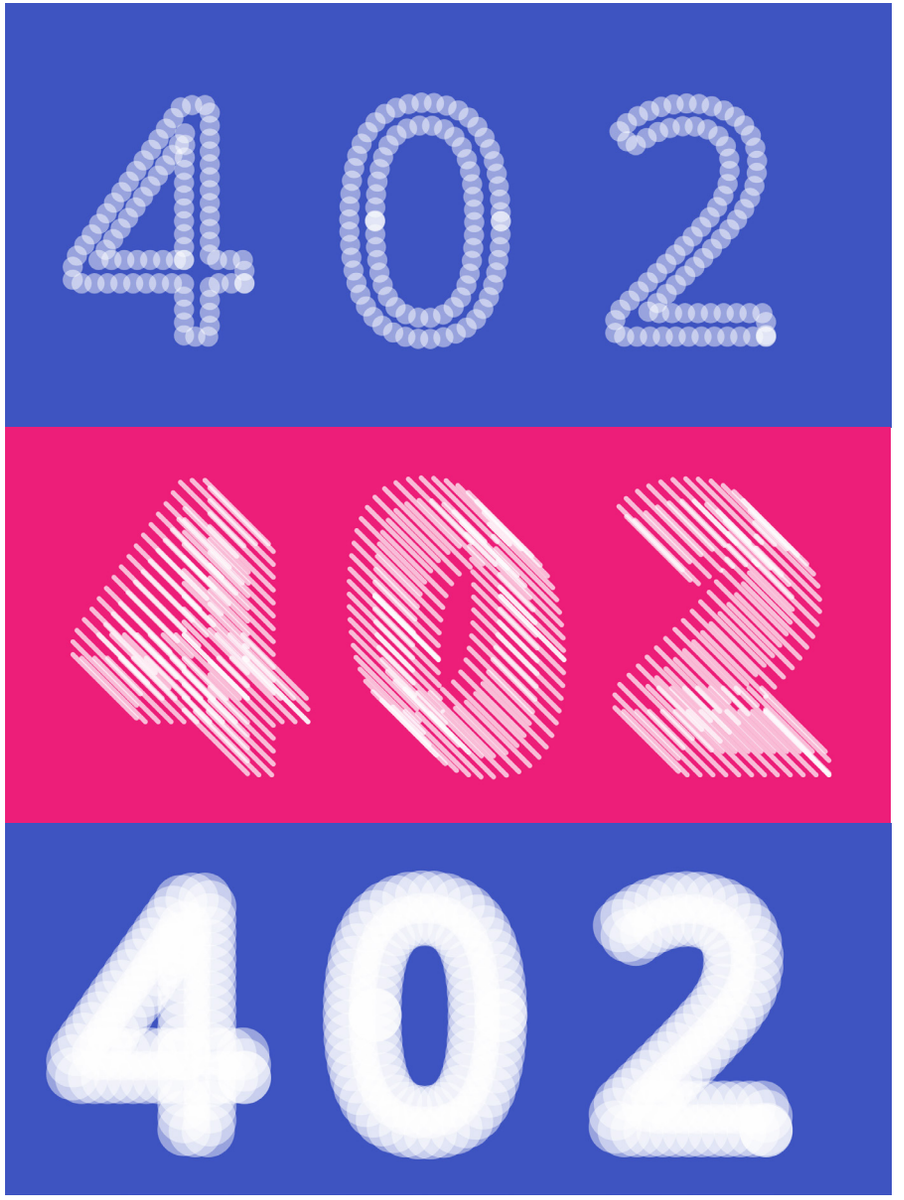




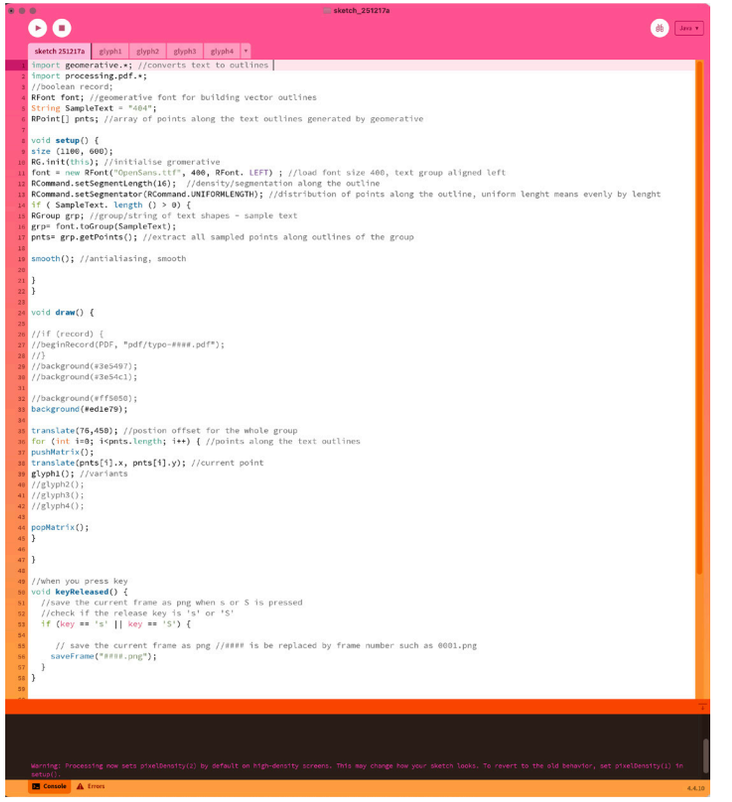
Further examples demonstrate variations generated through parameter adjustements and user interaction.

DATA MAPPING:
POSITION AND SPACE

Position-based inputs were mapped to geometric forms, where repetition, duration, and spatial interaction drive typographic transformation. These tests explore how different inputs influence density, structure, and legibility.



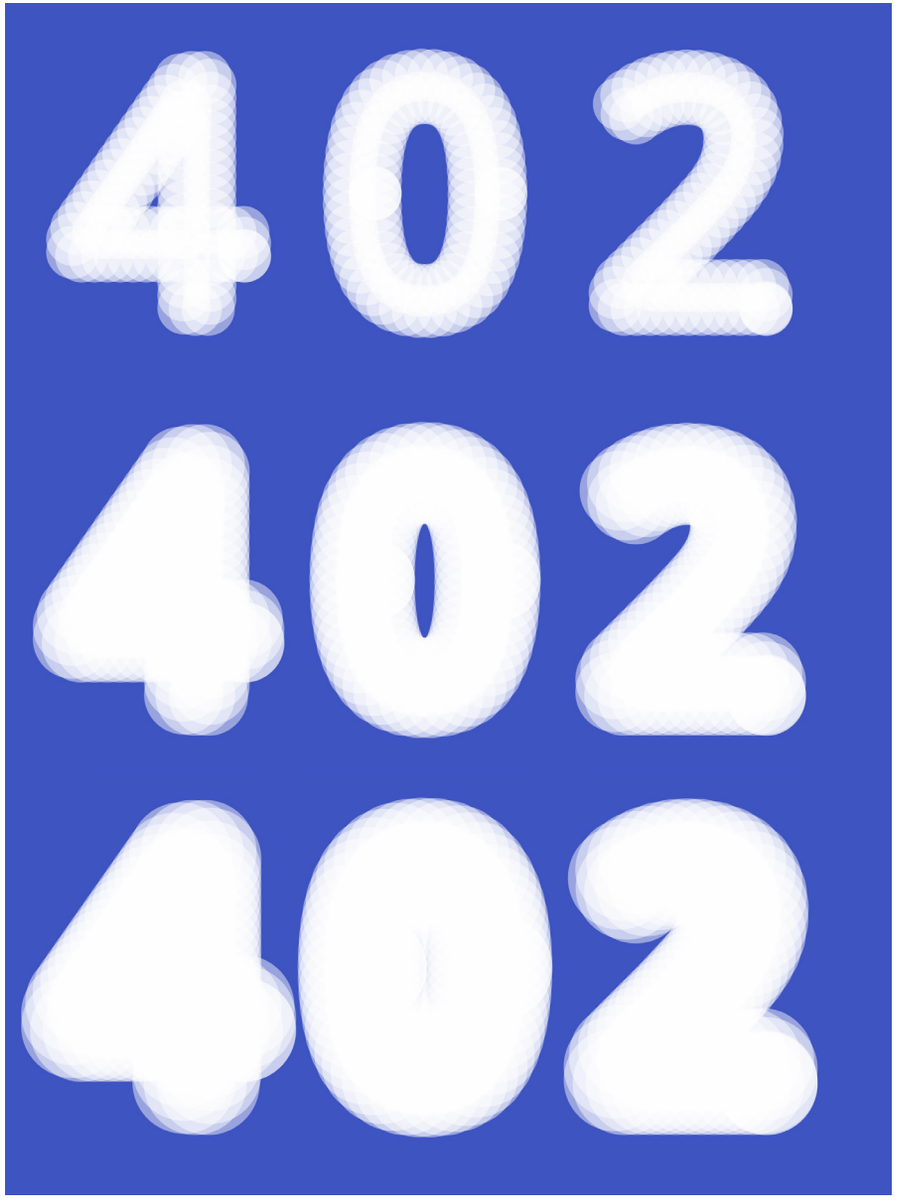
Position-based mapping controlled by mouse x and y coordinates



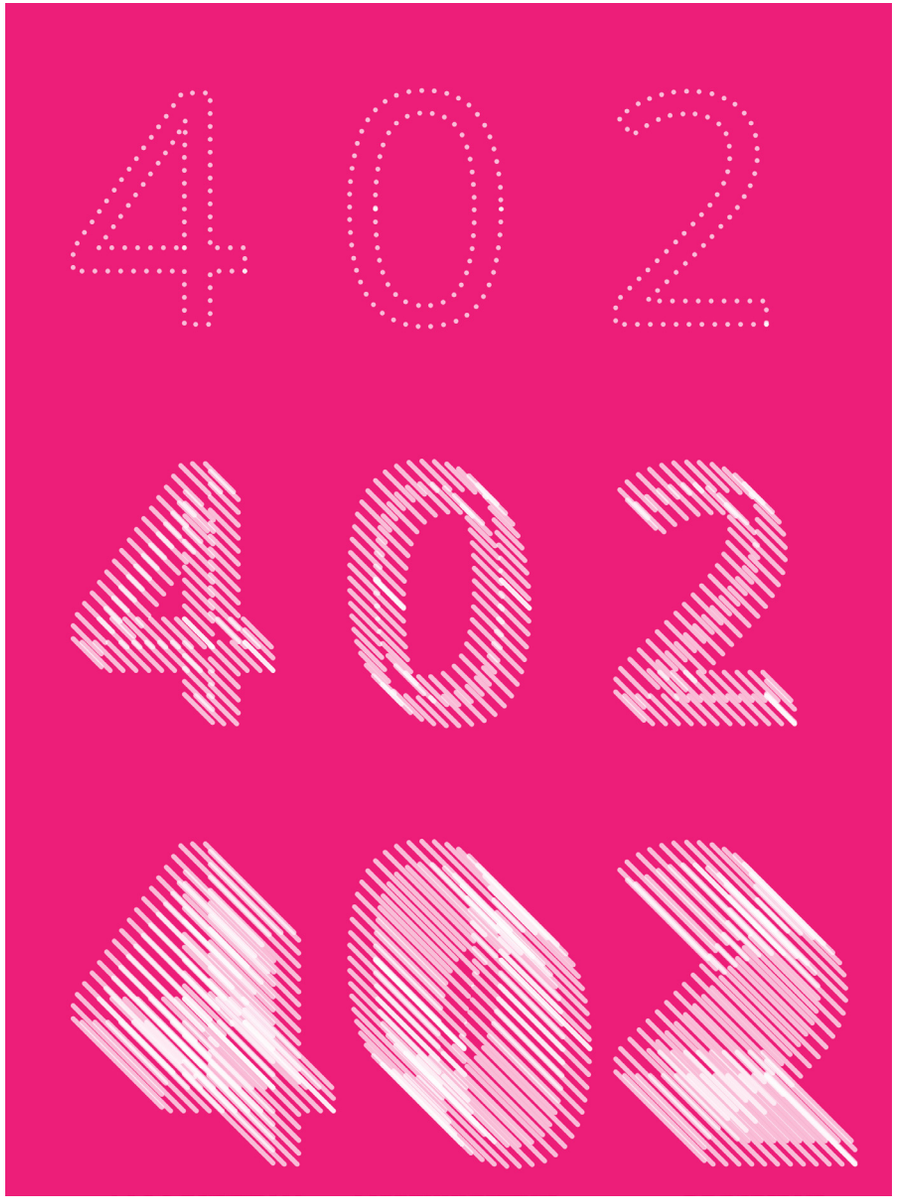
```
import geomerative.*; //converts text to outlines
import processing.pdf.*;
//boolean record;
RFont font; //geomerative font for building vector outlines
String SampleText = "404";
RPoint[] pnts; //array of points along the text outlines generated by geomerative

void setup() {
size (1100, 600);
RG.init(this); //initialise gromerative
font = new RFont("OpenSans.ttf", 400, RFont. LEFT) ; //load font size 400, text group aligned left
RCommand.setSegmentLength(16);  //density/segmentation along the outline
RCommand.setSegmentator(RCommand.UNIFORMLENGTH); //distribution of points along the outline, uniform lenght means evenly by lenght
if ( SampleText. length () > 0) {
RGroup grp; //group/string of text shapes - sample text
grp= font.toGroup(SampleText);
pnts= grp.getPoints(); //extract all sampled points along outlines of the group

smooth(); //antialiasing, smooth

}
}

void draw() {

//if (record) {
//beginRecord(PDF, "pdf/typo-####.pdf");
//}
//background(#3e5497);
//background(#3e54c1);

//background(#ff5050);
background(#ed1e79);

translate(76,450); //postion offset for the whole group
for (int i=0; i<pnts.length; i++) { //points along the text outlines
pushMatrix();
translate(pnts[i].x, pnts[i].y); //current point
glyph1(); //variants
//glyph2();
//glyph3();
//glyph4();

popMatrix();
}
}

//When you press key
void keyReleased() {
  //save the current frame as png when s or S is pressed
  //check if the release key is 's' or 'S'
  if (key == 's' || key == 'S') {

    // save the current frame as png //#### is be replaced by frame number such as 0001.png
    saveFrame("####.png");
  }
}
```

Code excerpt

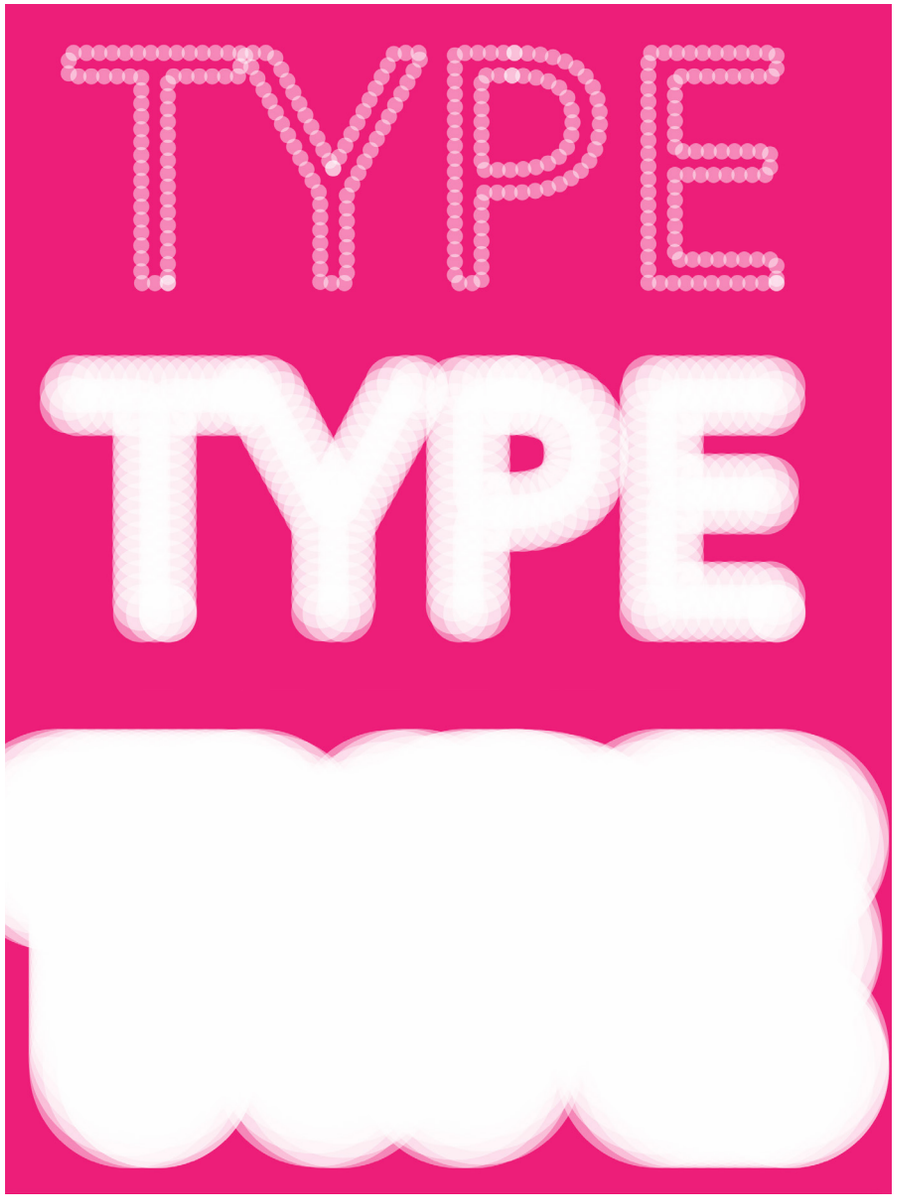
Letterforms were converted into vector outlines using Geomerative and sampled as point data. Each point was used to generate repeated geometric elements, allowing typographic forms to be reconstructed and transformed through controlled iteration.



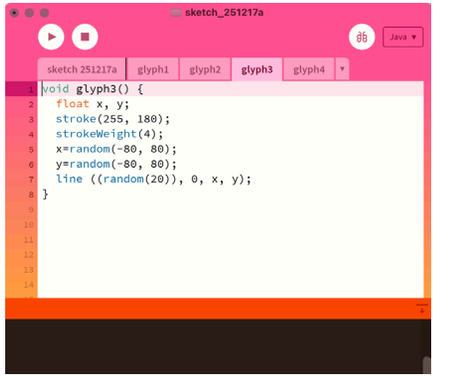
Position-based mapping
controlled by mouse x and y
coordinates

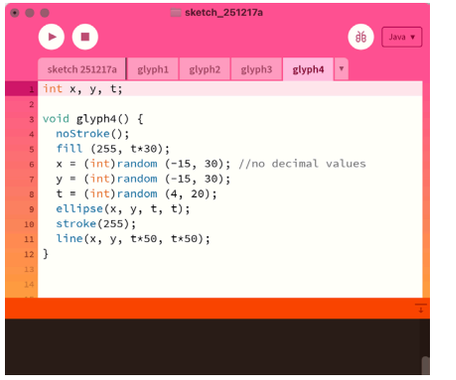


Position-based mapping
controlled by mouse x and y
coordinates

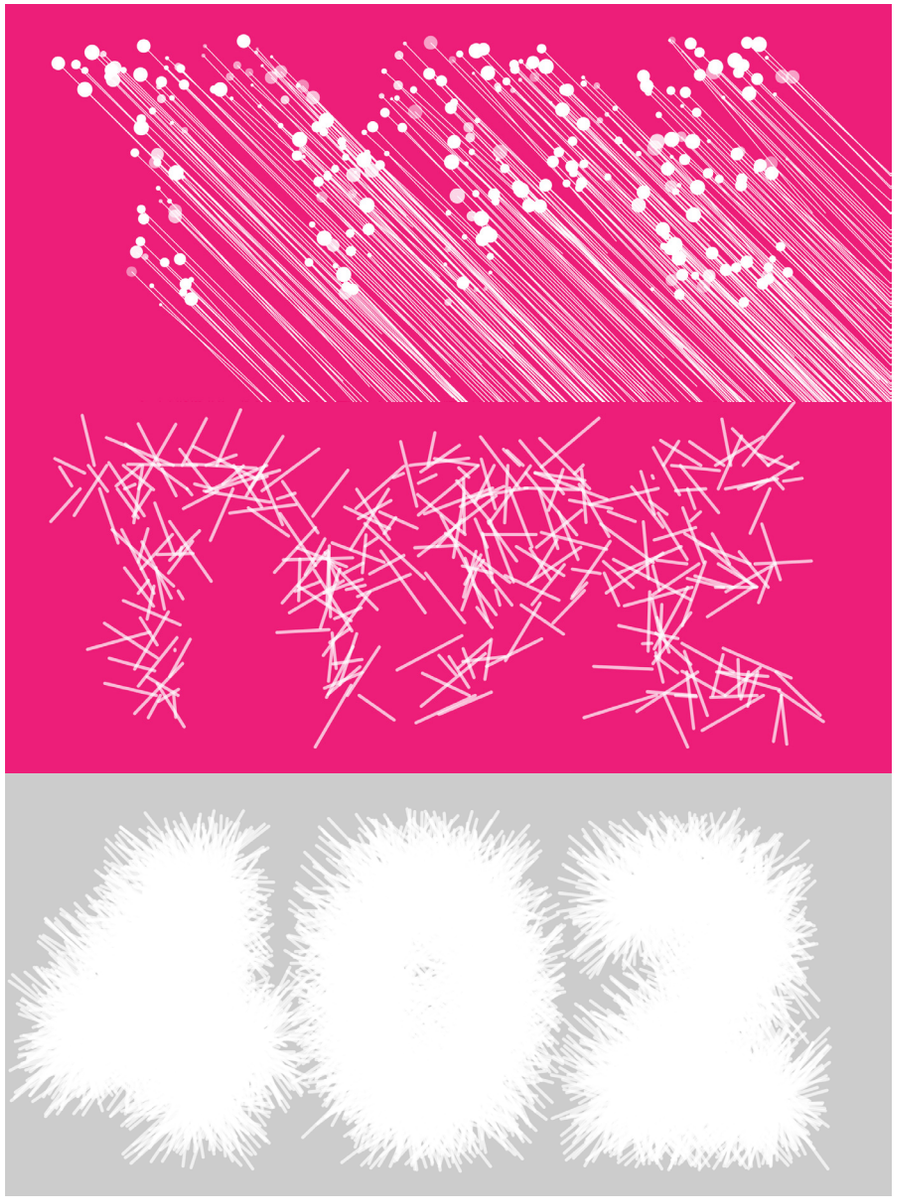


Position-based mapping controlled by mouse x and y coordinates





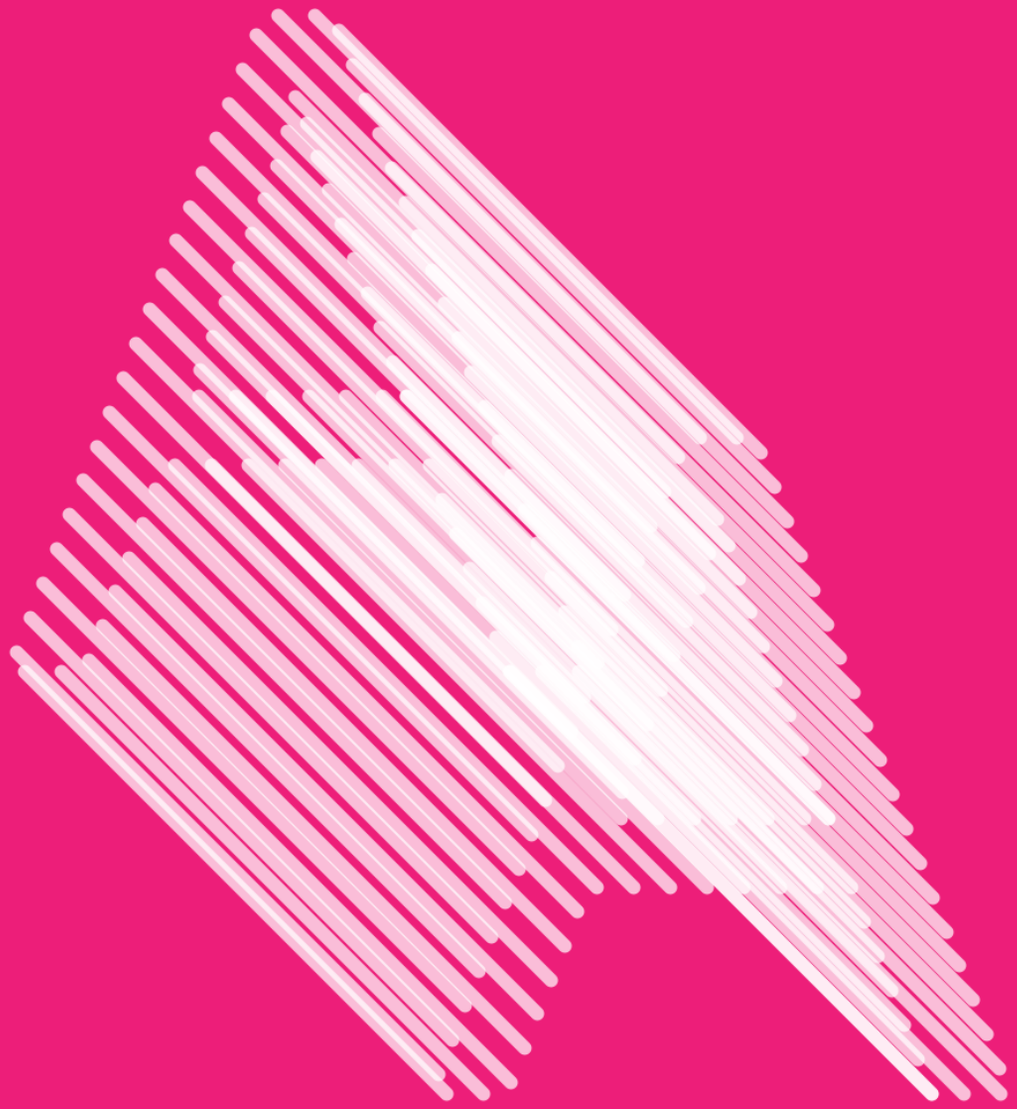
Different variants were structured as separate code modules using Processing tabs. Each defines a distinct visual behaviour, allowing systematic variation through controlled procedural rules.

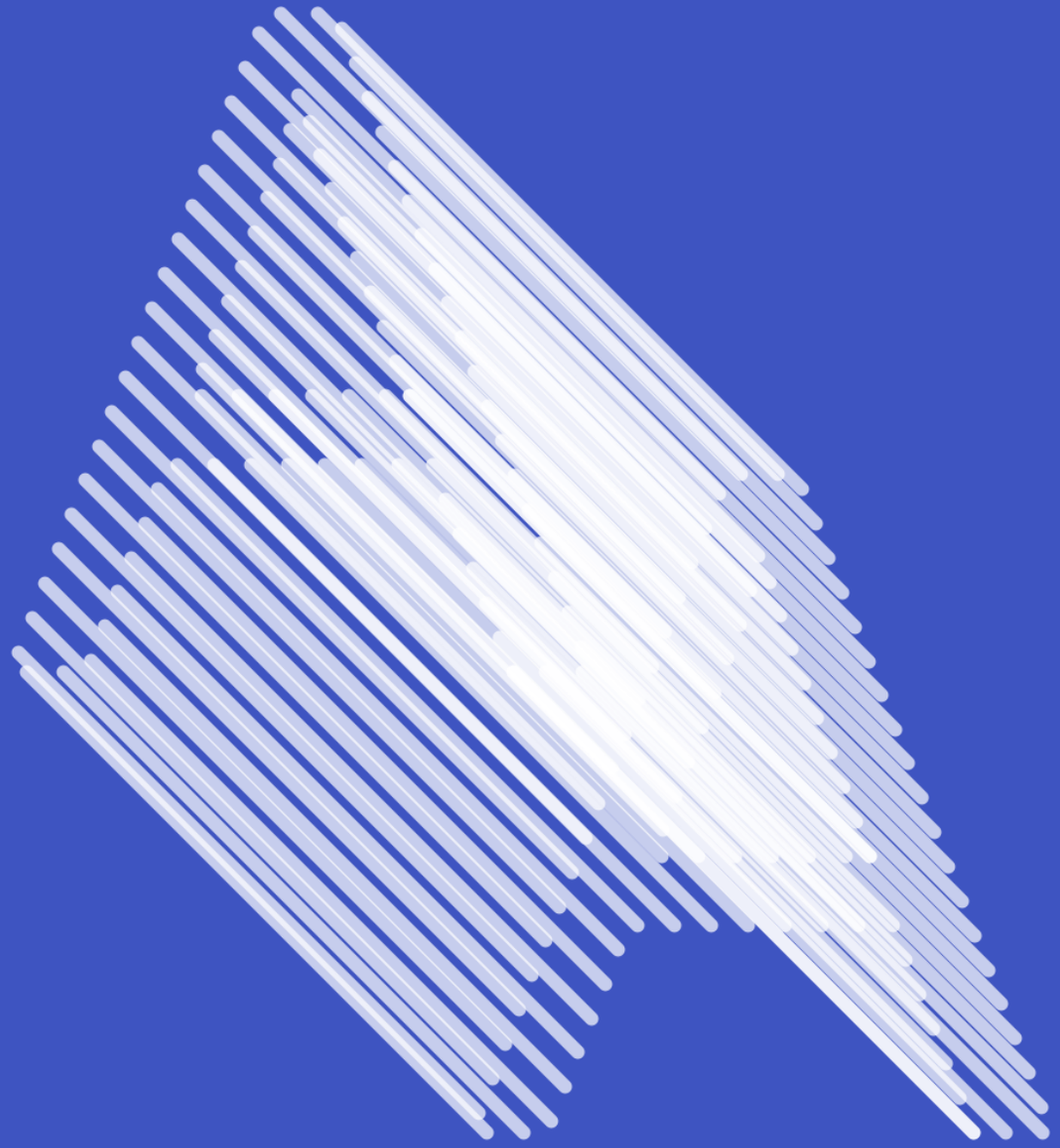


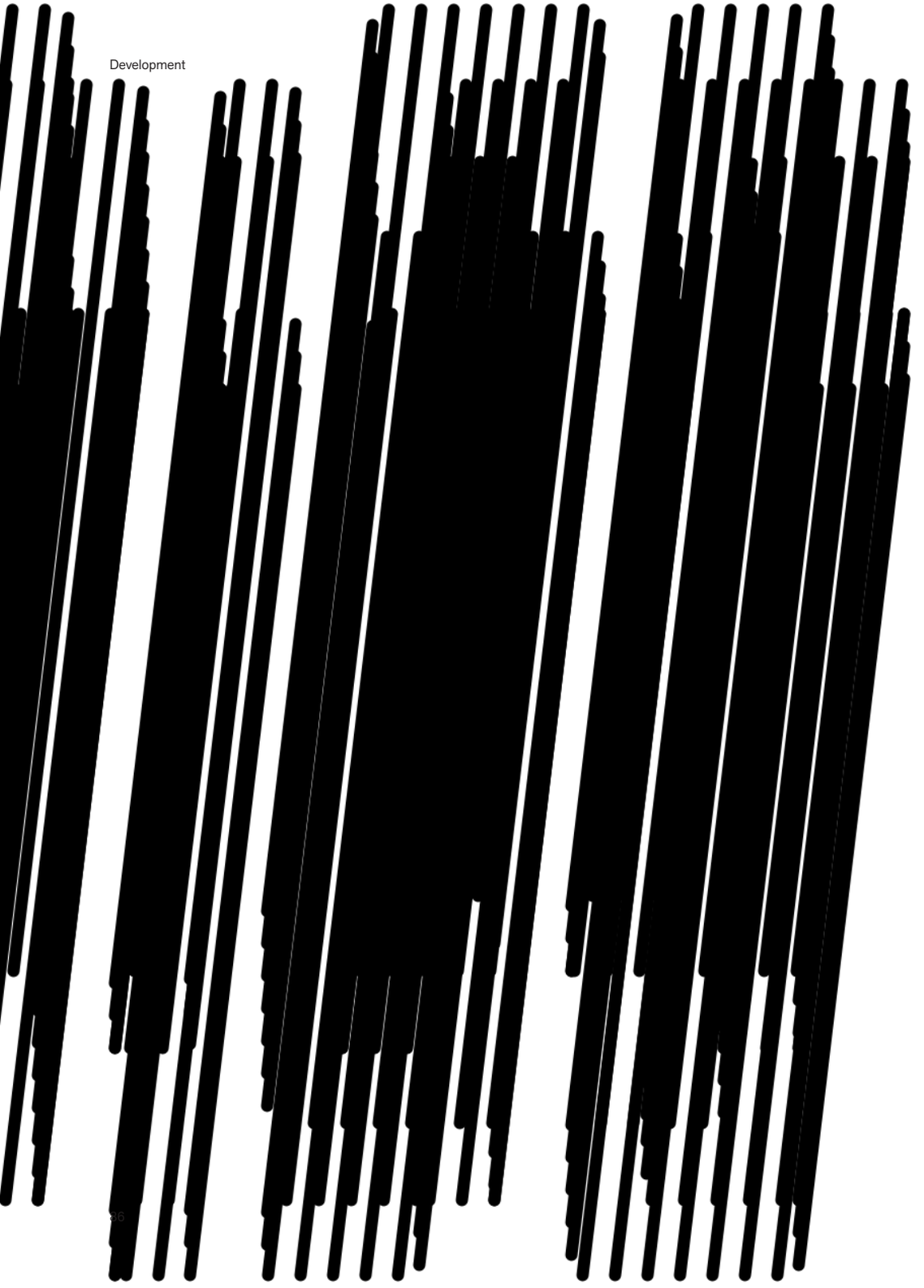
Each sampled point generates a circle and directional line, with randomised parameters controlling position and scale (top)

This variant uses randomised line direction (middle)

Radial dispersion from point data (bottom)

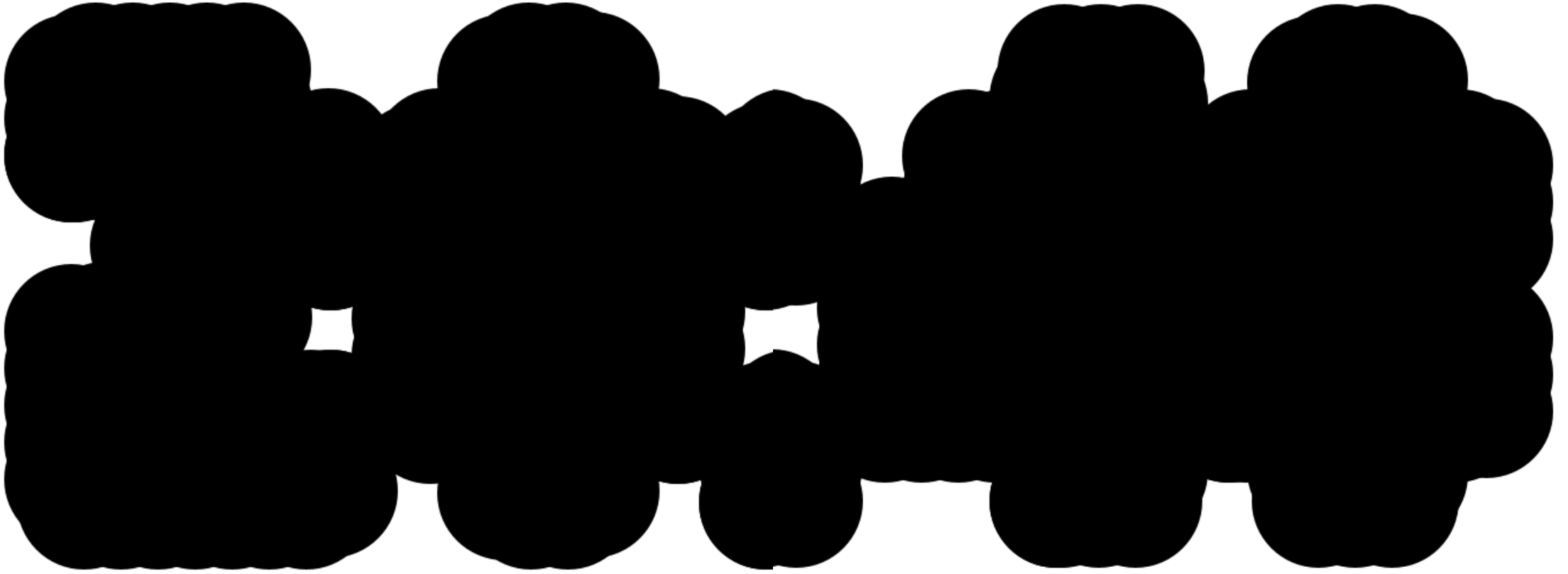


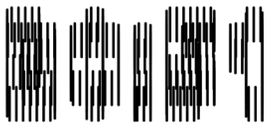


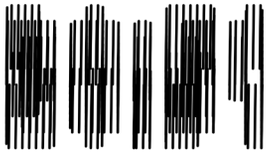


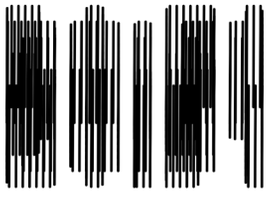
DATA MAPPING:
SHAPE AND TIME

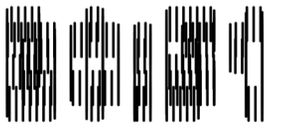
Time-based inputs were mapped to simple geometric forms, transforming repetition and duration into visual structure. These tests explore how temporal variation influences density, form, and legibility.

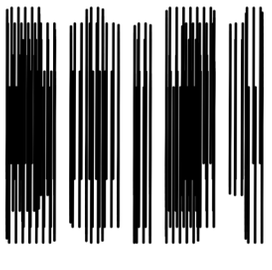


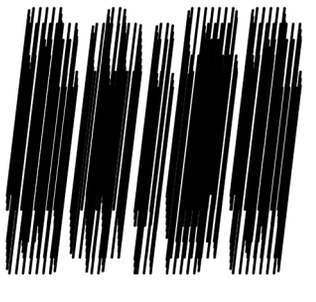


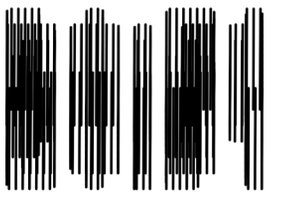


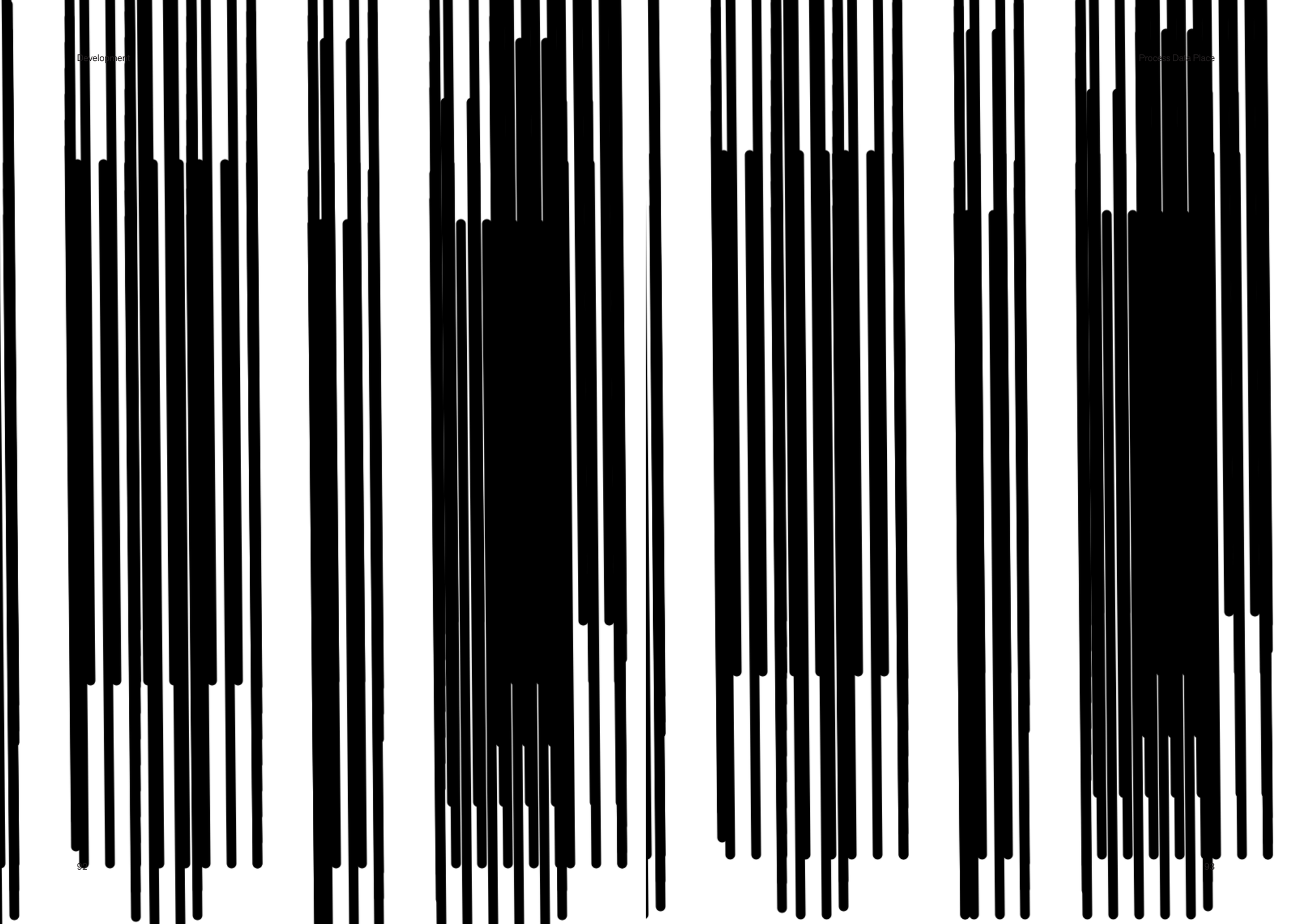












```
//testing geomerative
//major project processing work

import geomerative.*; //converts text to outlines
//import processing.pdf.*;
//boolean record;
RFont font; //geomerative font for building vector outlines
RPoint[] pnts;

String SampleText = "";
//RPoint[] pnts; //array of points along the text outlines generated by geomerative

void setup() {
//size (1100, 600);
size (1000, 1000);
smooth(); //antialiasing, smooth
frameRate(25);

RG.init(this); //initialise gromerative

font = new RFont("tiny5-Regular.ttf", 300, RFont.LEFT) ; //load font size 400, text group aligned left

RCommand.setSegmentLength(16);  //density/segmentation/point sampling along the outline
RCommand.setSegmentator(RCommand.UNIFORMLENGTH); //distribution of points along the outline, uniform lenght
}

void draw() {
 //background (#ed1e79);
 //background(237, 30, 121, 120); //magenta
 //background(#fbb03b); //yellow
// background(#3e54c1); //blue
// background(#000000); black
 background (#ffffff);

 //get current time
 int h = hour();
 int m = minute();
 //int s = second();
 SampleText = nf(h, 2) + ":" + nf(m,2);  //for seonds display edit

 //time hh:mm
 //text(nf(h, 2) + ":" + nf(m, 2), width/2, height/2);

 //convert text to group and points
 RGroup grp = font.toGroup(SampleText).toPolygonGroup(); //group/string of text shape
 pnts = grp.getPoints();

 pushMatrix();
// translate (76, 400);
 translate (200, 400);  //centre  - edit x, y position as required

 for (int i=0; i<pnts.length; i++) { //points along the text outlines
 pushMatrix();
 translate(pnts[i].x, pnts[i].y); //current point
 glyph3();   //change the number here for respective variants
 popMatrix();
 }

 popMatrix();
```

Code excerpt

Time data (seconds) is mapped to line length, generating typographic variation through proportional changes over time.

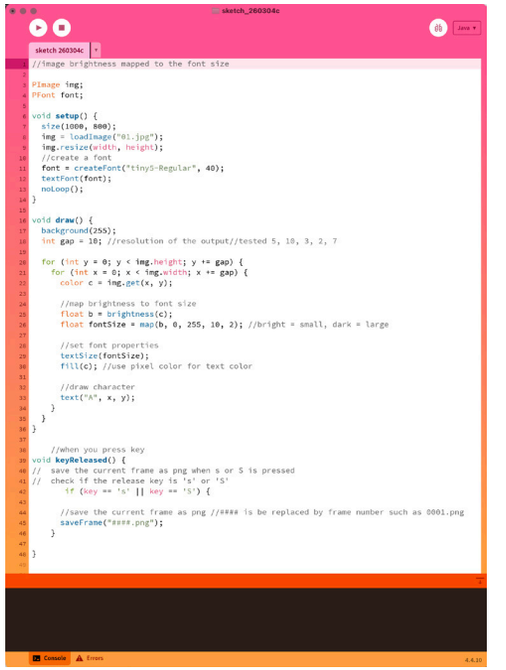
ABCDEFGHIJ
KLMNOPQRS
TUVWXYZ
abcdefghij
klmnopqrs
tuvwxyz
0123456789

Typeface: tiny5-Regular

Used as a placeholder while the custom typeface was being developed

DATA MAPPING:
IMAGE AND SOUND

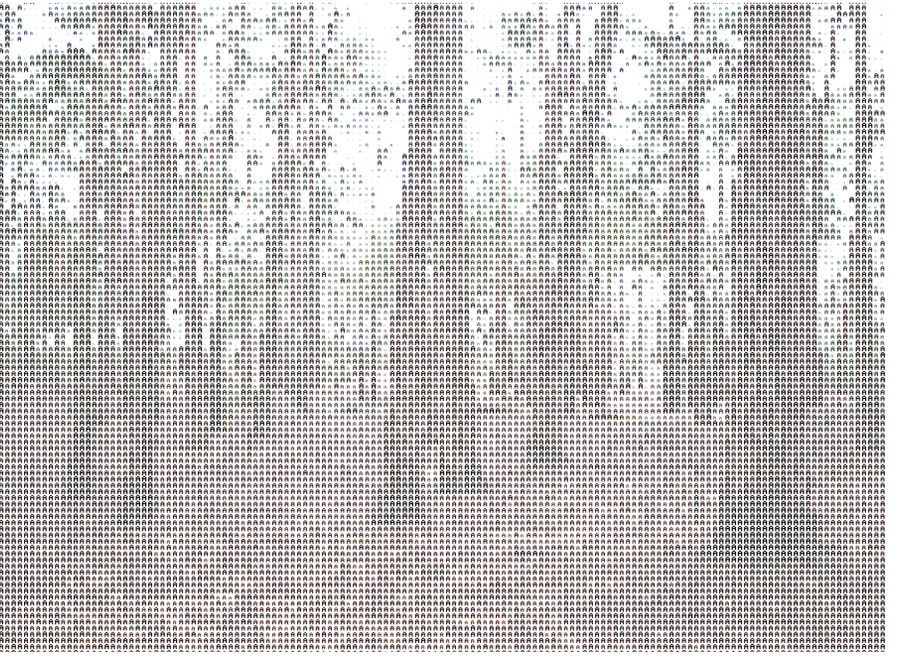
Image data was mapped to typographic systems through multiple approaches, translating pixel brightness and colour into variations in structure, density, and form. These tests explore how visual information can be reconstructed at both system and individual letter levels.

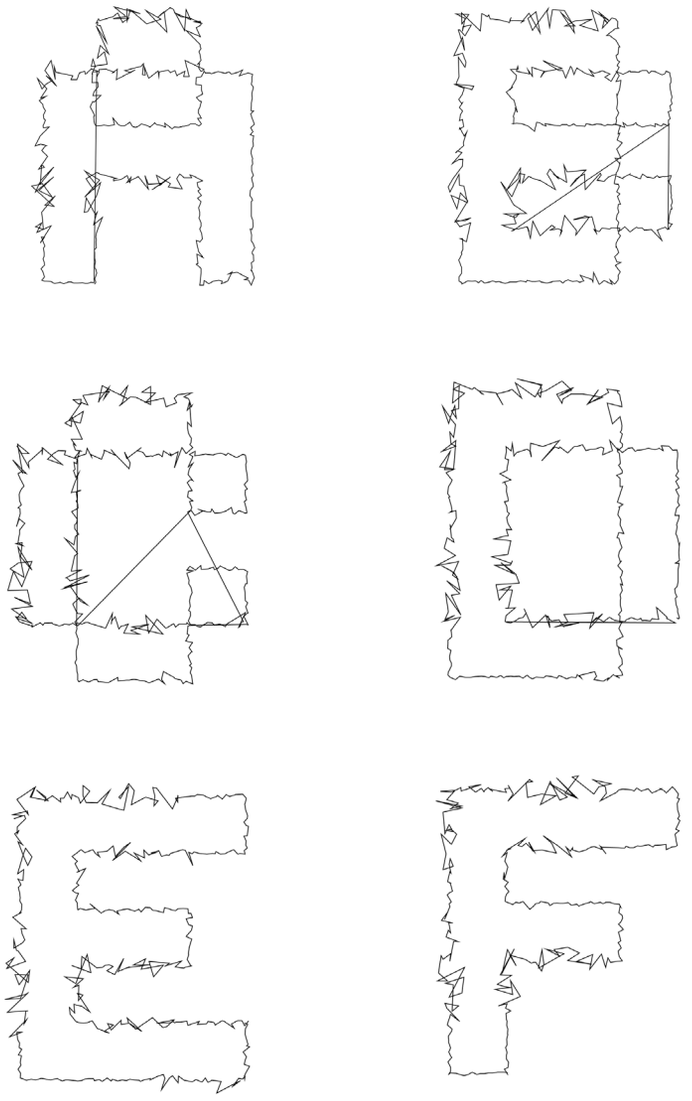


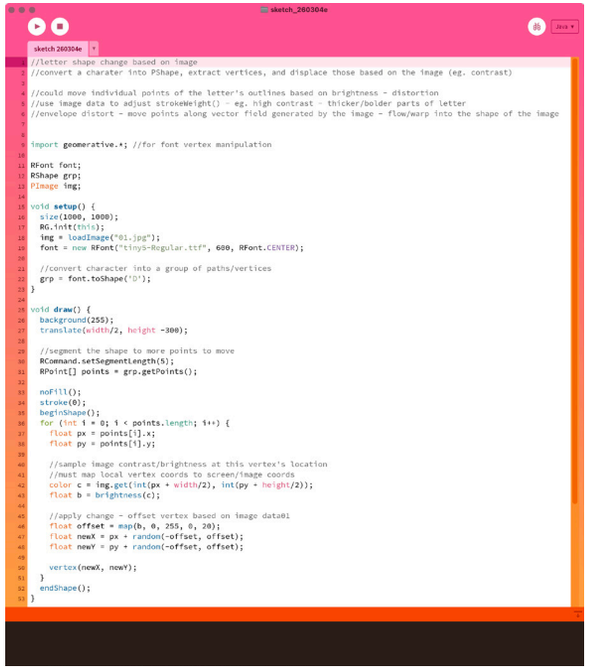
```
//Image brightness mapped to the font size

PImage img;
PFont font;

void setup() {
  size(1000, 800);
  img = loadImage("01.jpg");
  img.resize(width, height);
  //create a font
  font = createFont("tiny5-Regular", 40);
  textFont(font);
  noLoop();
}

void draw() {
  background(255);
  int gap = 10; //resolution of the output//tested 5, 10, 3, 2, 7

  for (int y = 0; y < img.height; y += gap) {
    for (int x = 0; x < img.width; x += gap) {
      color c = img.get(x, y);

      //map brightness to font size
      float b = brightness(c);
      float fontSize = map(b, 0, 255, 10, 2); //bright = small, dark = large

      //set font properties
      textSize(fontSize);
      fill(c); //use pixel color for text color

      //draw character
      text("A", x, y);
    }
  }
}

  //when you press key
void keyReleased() {
//  save the current frame as png when s or S is pressed
//  check if the release key is 's' or 'S'
    if (key == 's' || key == 'S') {

      //save the current frame as png //#### is be replaced by frame number such as 0001.png
      saveFrame("####.png");
    }

}
```

Code excerpt

Image data was mapped to typographic form by translating pixel brightness and colour into variations in character size, density, and colour, reconstructing visual information through text.



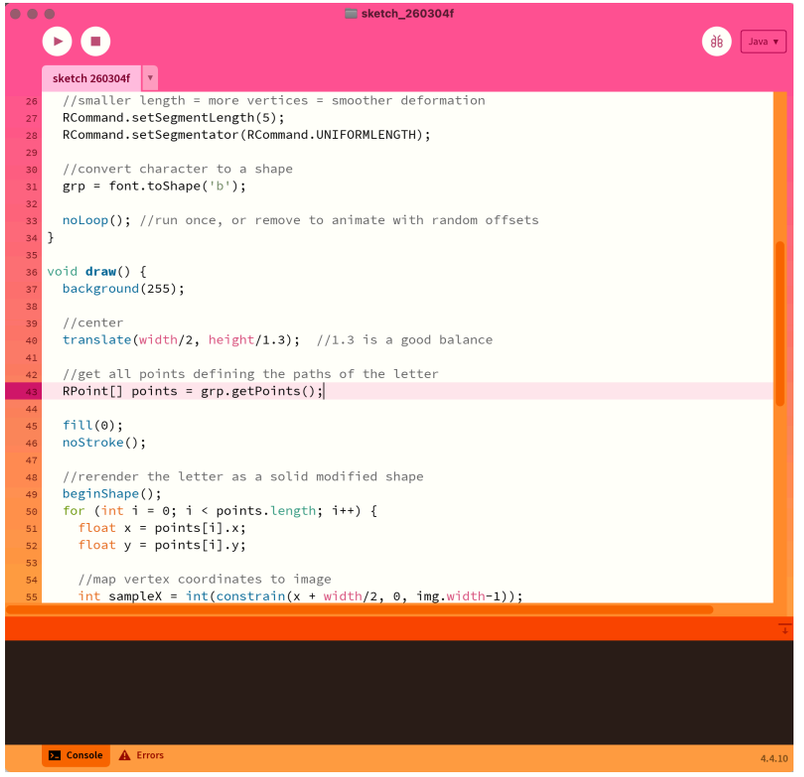




```
//letter shape change based on image
//convert a charater into PShape, extract vertices, and displace those based on the image (eg. contrast)

//could move individual points of the letter's outlines based on brightness - distortion
//use image data to adjust strokeWeight() - eg. high contrast - thicker/bolder parts of letter
//envelope distort - move points along vector field generated by the image - flow/warp into the shape of the image


import geomerative.*; //for font vertex manipulation

RFont font;
RShape grp;
PImage img;

void setup() {
  size(1000, 1000);
  RG.init(this);
  img = loadImage("01.jpg");
  font = new RFont("tinyS-Regular.ttf", 600, RFont.CENTER);

  //convert character into a group of paths/vertices
  grp = font.toShape('D');
}

void draw() {
  background(255);
  translate(width/2, height -300);

  //segment the shape to more points to move
  RCommand.setSegmentLength(5);
  RPoint[] points = grp.getPoints();

  noFill();
  stroke(0);
  beginShape();
  for (int i = 0; i < points.length; i++) {
    float px = points[i].x;
    float py = points[i].y;

    //sample image contrast/brightness at this vertex's location
    //must map local vertex coords to screen/image coords
    color c = img.get(int(px + width/2), int(py + height/2));
    float b = brightness(c);

    //apply change - offset vertex based on image data01
    float offset = map(b, 0, 255, 0, 20);
    float newX = px + random(-offset, offset);
    float newY = py + random(-offset, offset);

    vertex(newX, newY);
  }
  endShape();
}
```

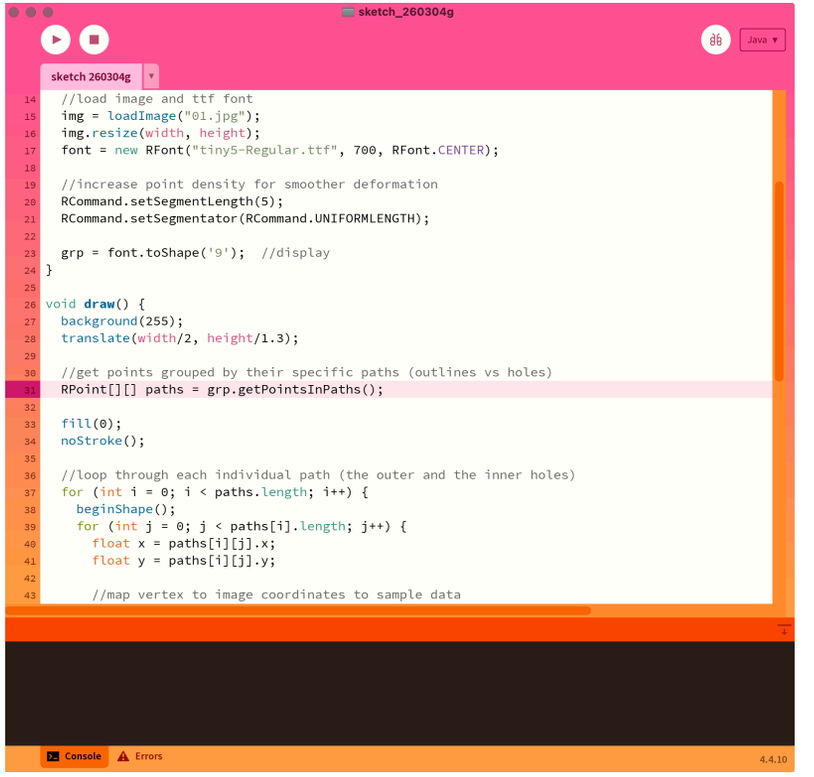
Code excerpt

Image brightness was mapped to individual outline vertices, generating localised distortion and irregular edge variation within each letterform.



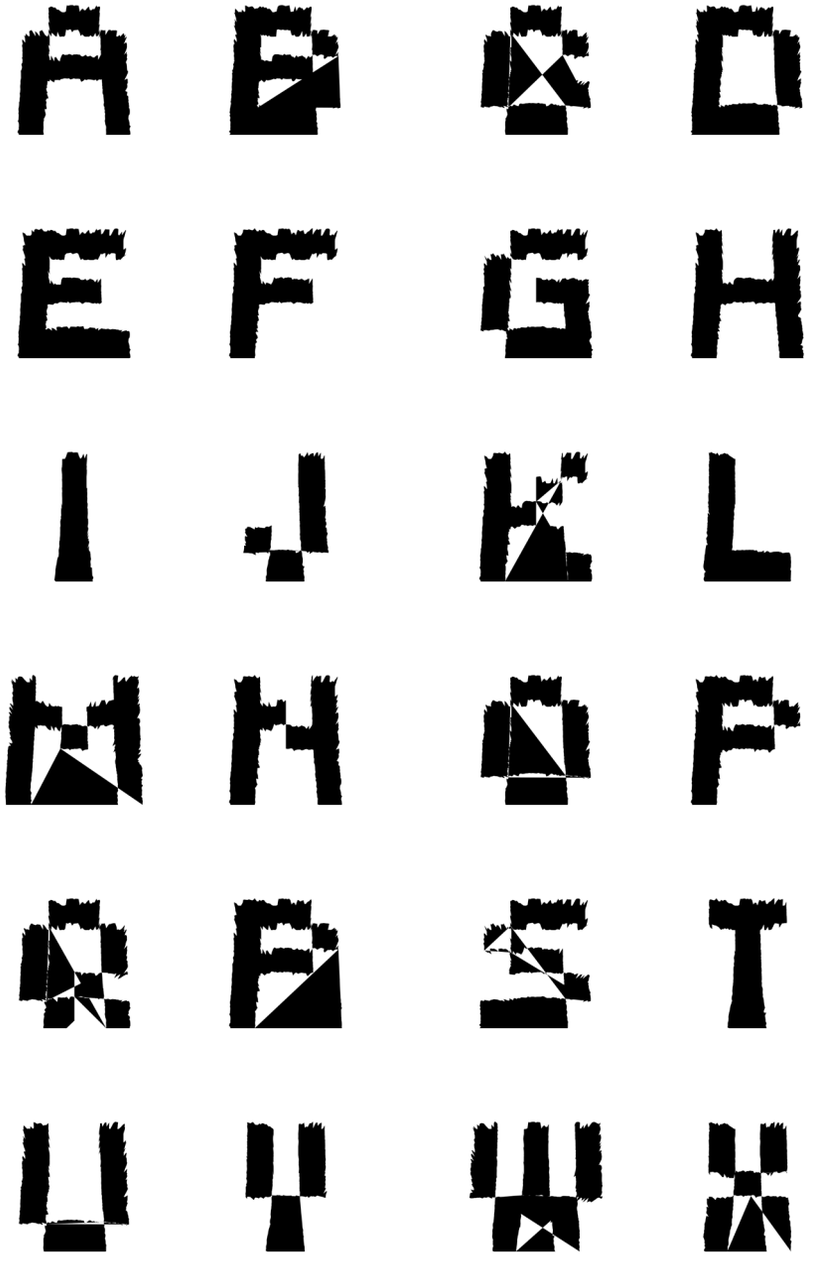
Process 1

Brightness-driven displacement applied to all vertices equally, resulting in uncontrolled deformation, broken counters, and inconsistent letter structure.

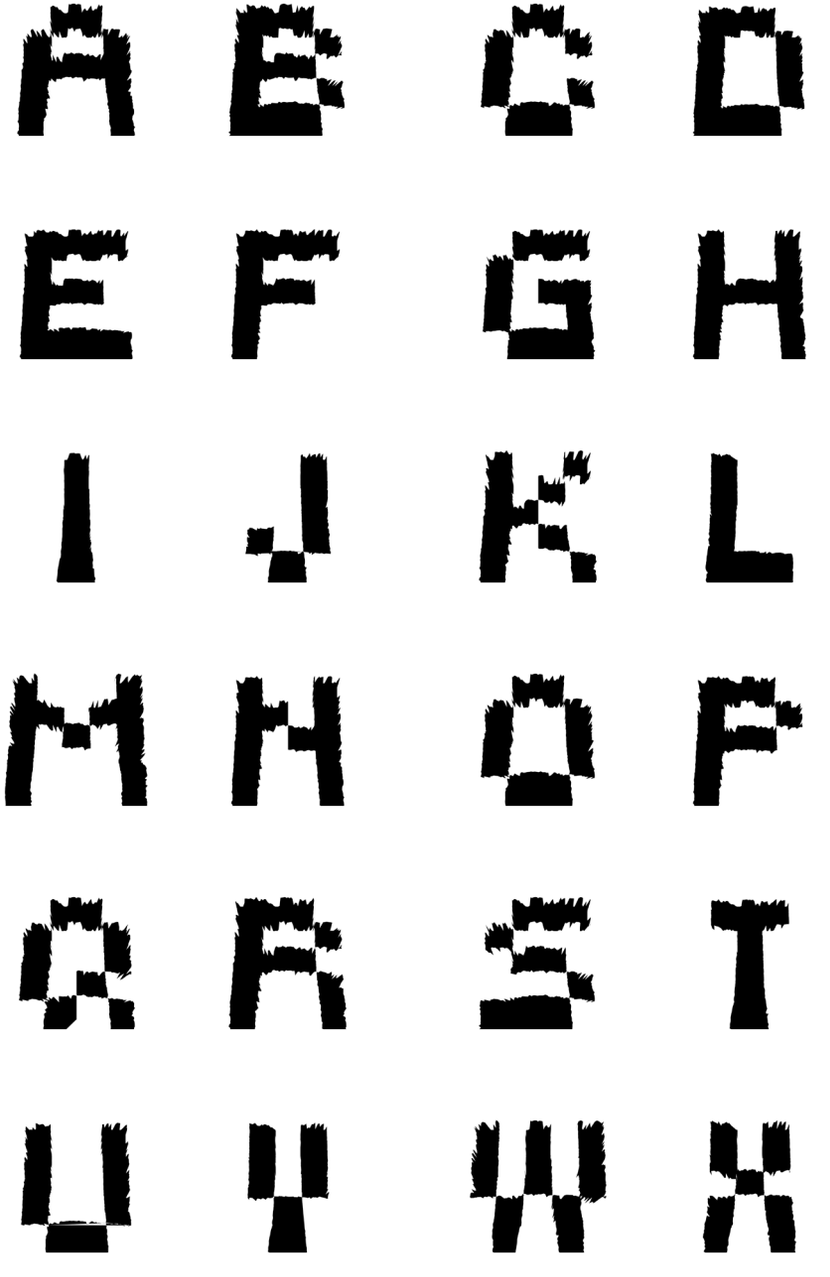


Process 2

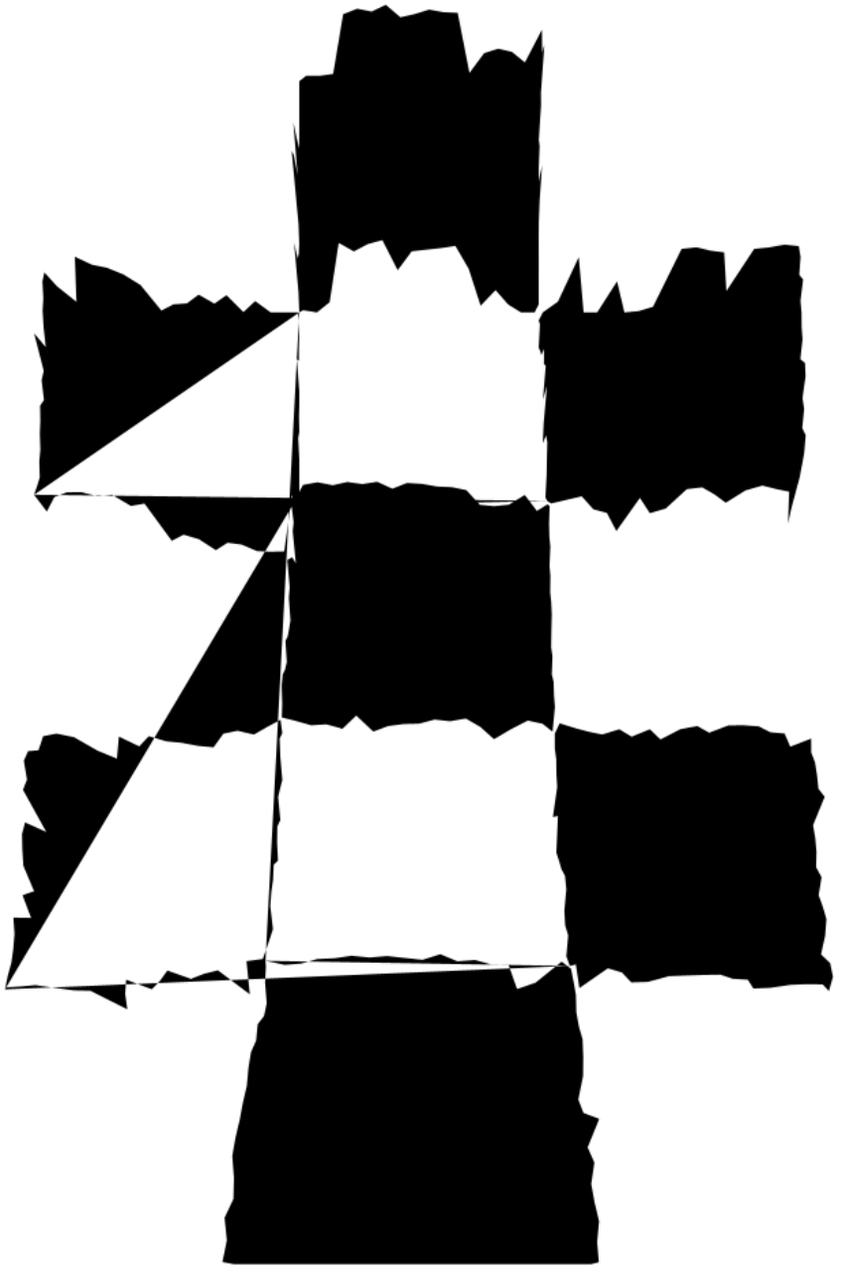
Brightness-driven displacement applied to separated outer and inner paths, producing controlled deformation, preserved counters, and clearer typographic structure.



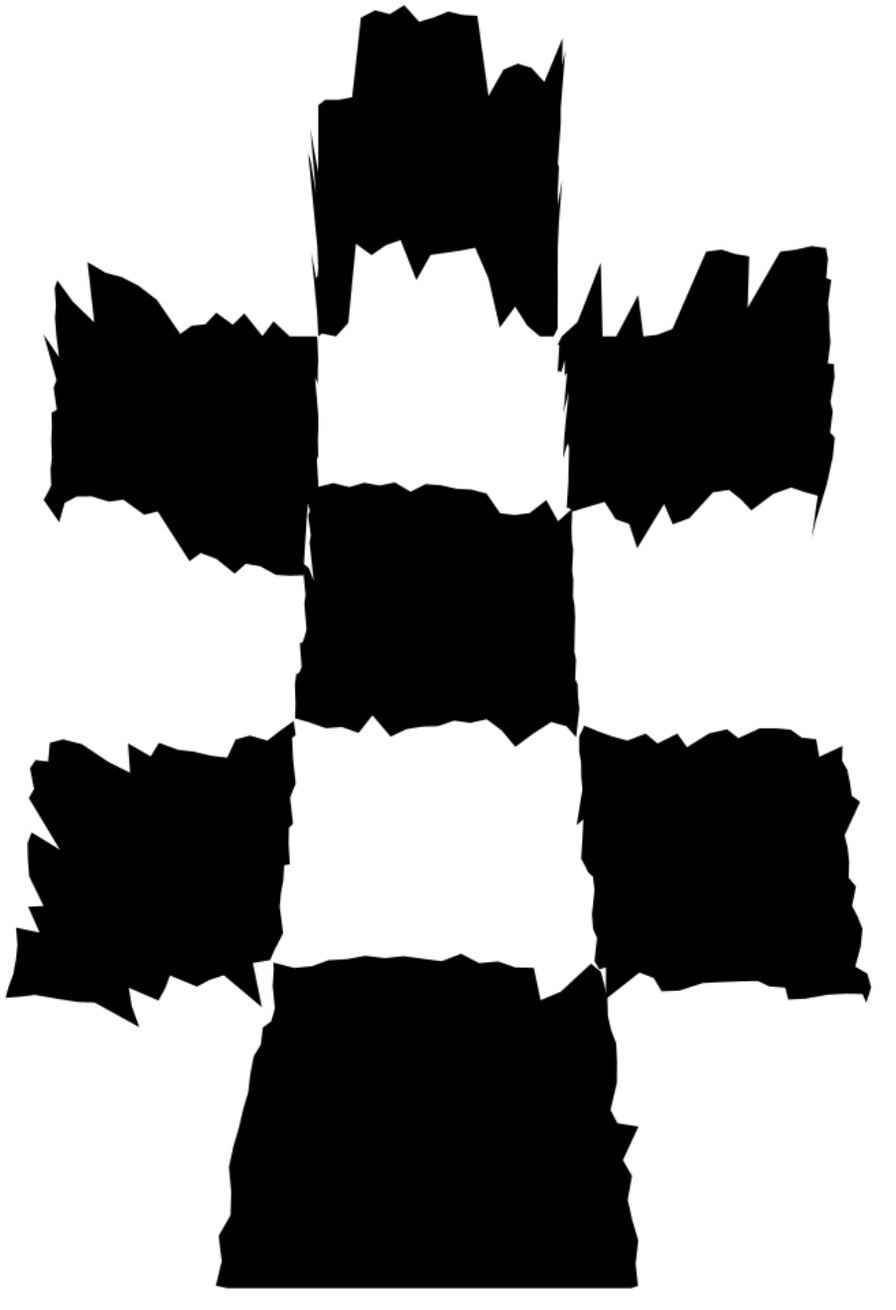
Process 1



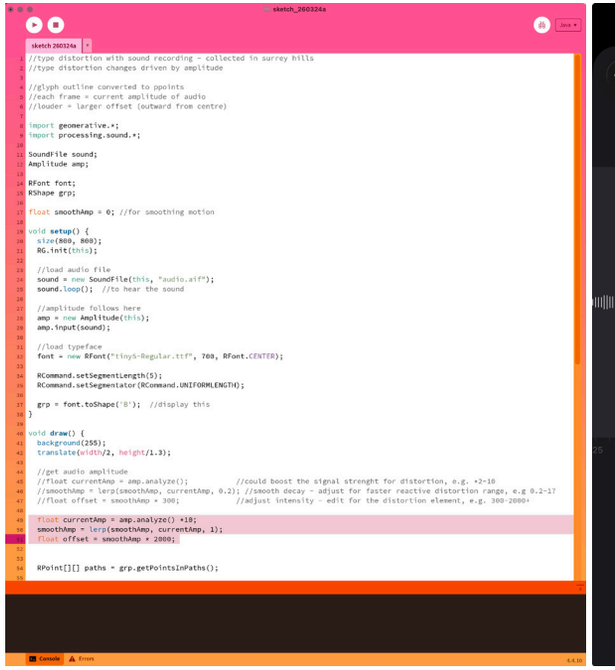
Process 2



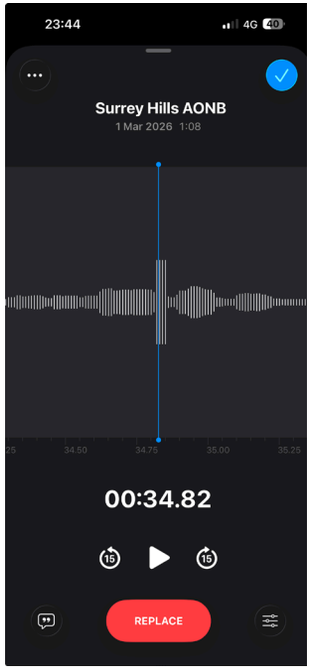
Process 1



Process 2



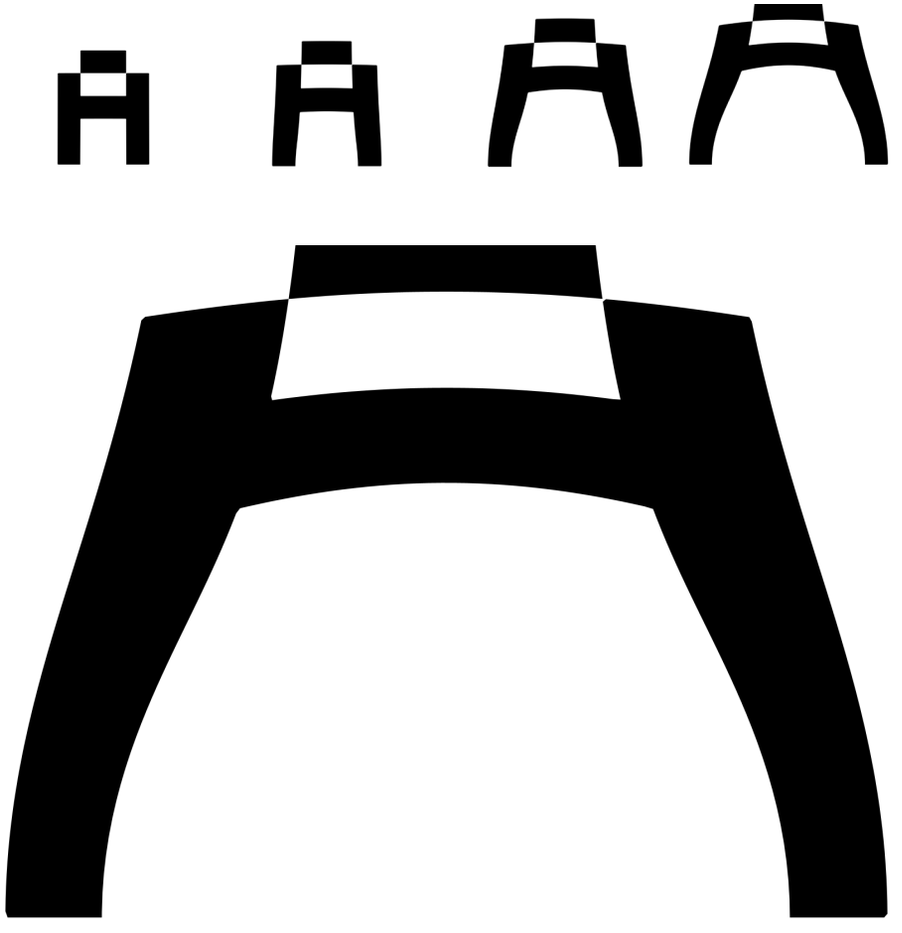
Code excerpt



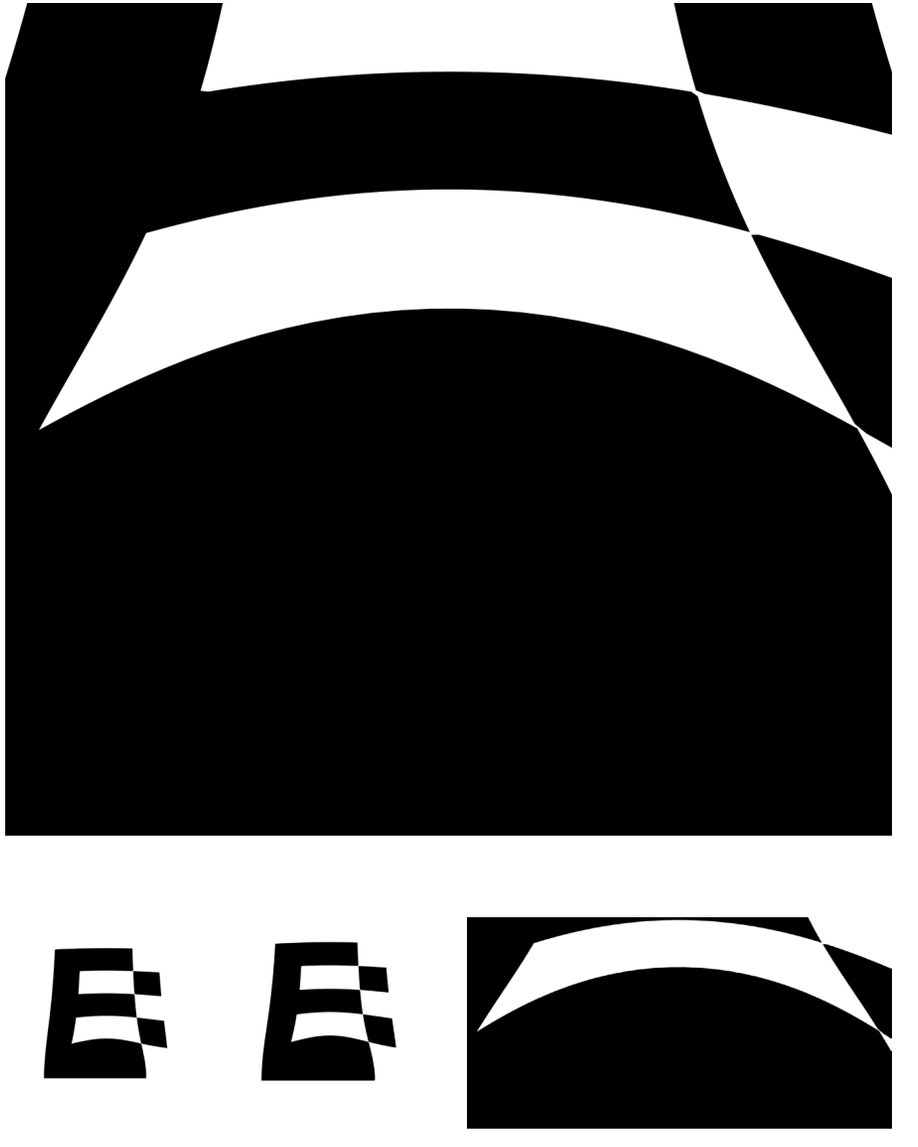
Audio

DATA MAPPING:
SOUND

Sound amplitude was mapped to vertex displacement, transforming letterforms in real time through audio-driven expansion and distortion.



Progression of letterform distortion driven by increasing audio amplitude.

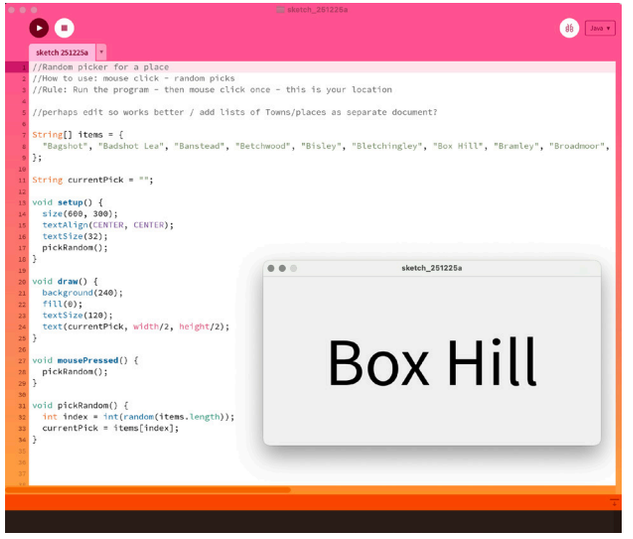


High amplitude generates maximum displacement, producing expanded and deconstructed typographic form.

07

Phase 4

Data Collection Protocol



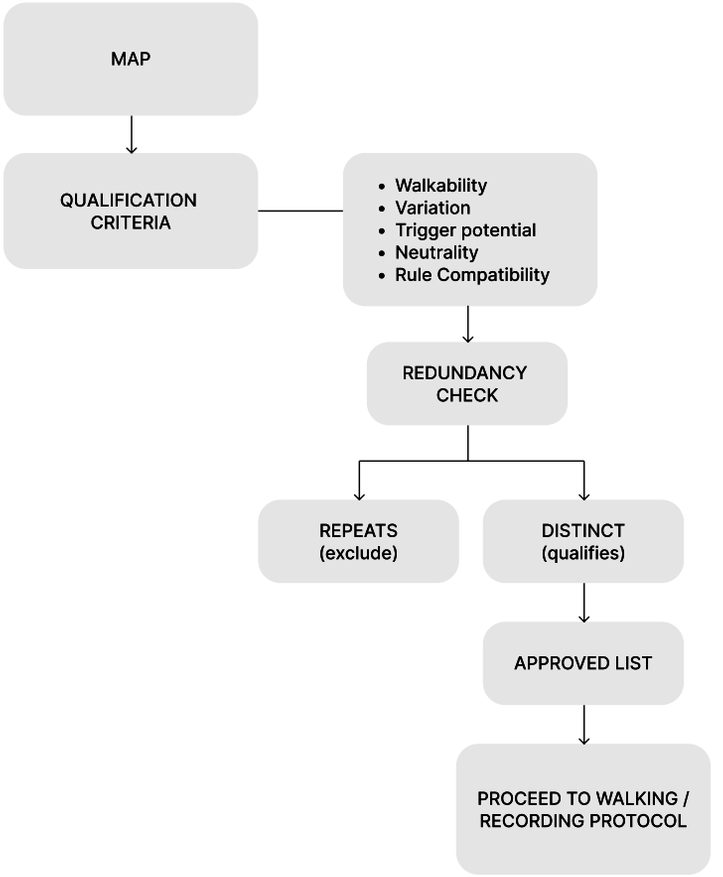
RANDOM PICK TOOL
(Processing)

Initial testing of a custom random selection tool for choosing collection locations. Place names are drawn from a predefined list, introducing a structured yet non-biased approach to data collection.

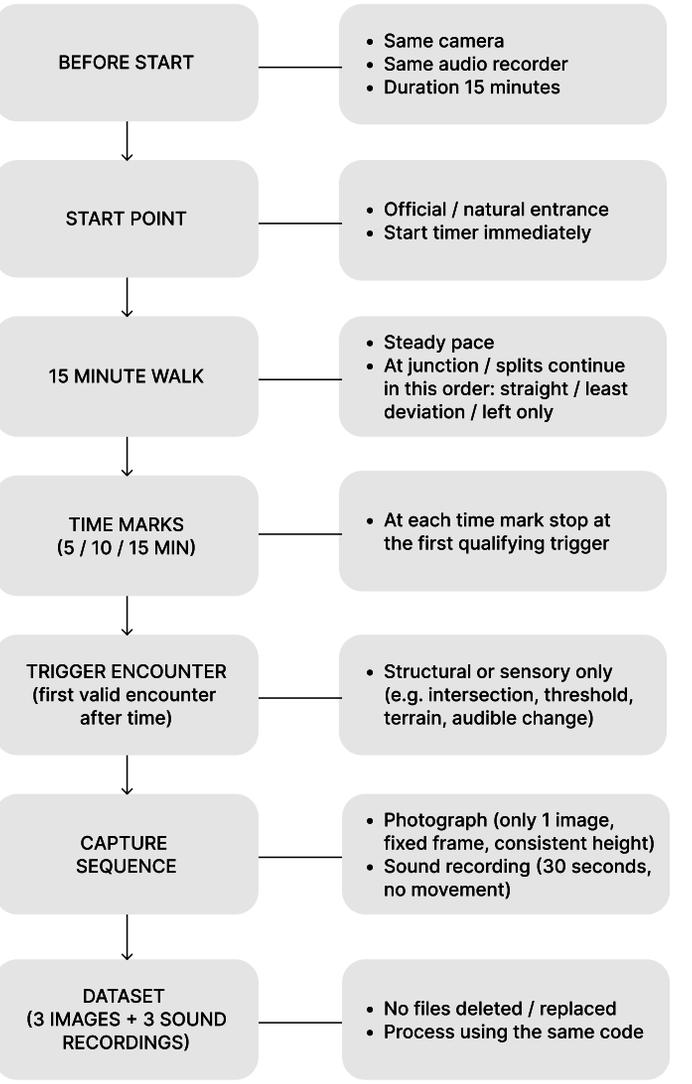
SOUND CONDITIONS PROTOCOL
(Stylosette)

Exploratory field-testing of a contact-based sound collection method using a stylosette. This approach was developed as a contingency to capture sonic data in environments where ambient sound was limited or inconsistent.

STEP 1
QUALIFICATION LIST



STEP 2
WALKING / RECORDING PROTOCOL



DATA COLLECTION PROTOCOL REFINED

Finalised data collection protocol establishing a controlled and repeatable system for gathering visual and sonic data. Locations are pre-qualified using map-based criteria, then activated through a fixed walking and recording procedure that ensures consistency while allowing environmental variation to define the outcome.

08

Test and Refinement

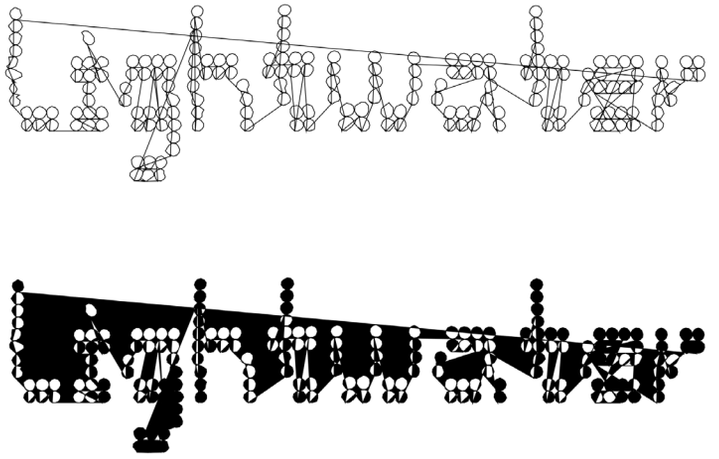
Output variations from
sky based input (left)

Typeface: Pixelfont-Circle
Structure: Image brightness mapping
Modulation: Sound amplitude

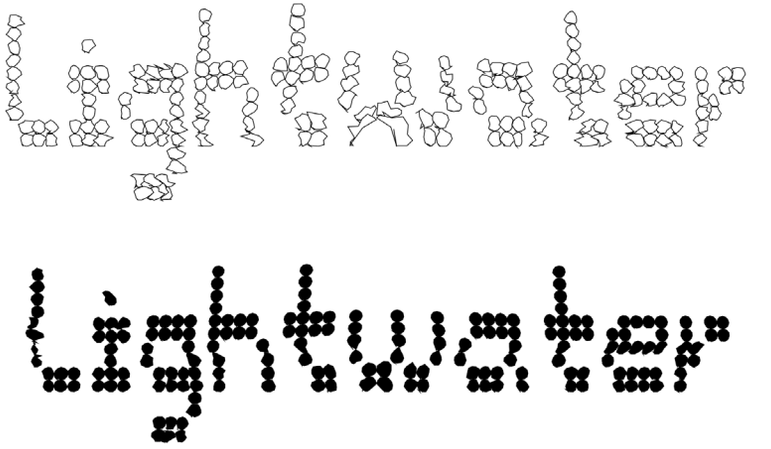
SYSTEM TESTING

The system was first applied to a controlled dataset from Lightwater Country Park to evaluate its behaviour in practice. This stage functions as a validation process, testing the consistency, responsiveness, and range of typographic outcomes generated by environmental data.

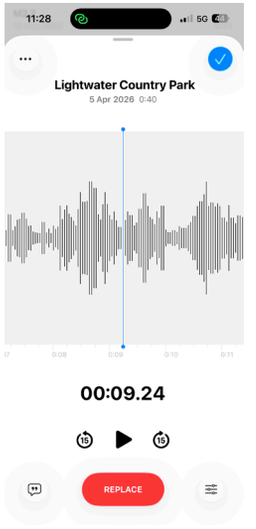
All outputs are generated using the same typeface and algorithm, with variation emerging solely from differences in environmental data conditions, including woodland, sky, infrastructure and open landscape.



Initial mapping
(inconsistent)



Revised mapping
(path grouping applied)

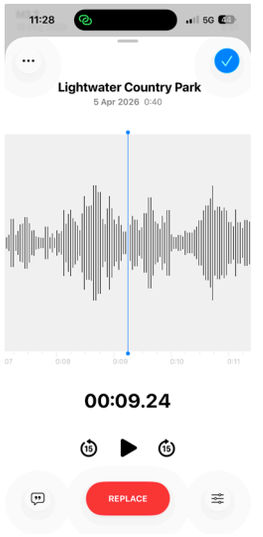


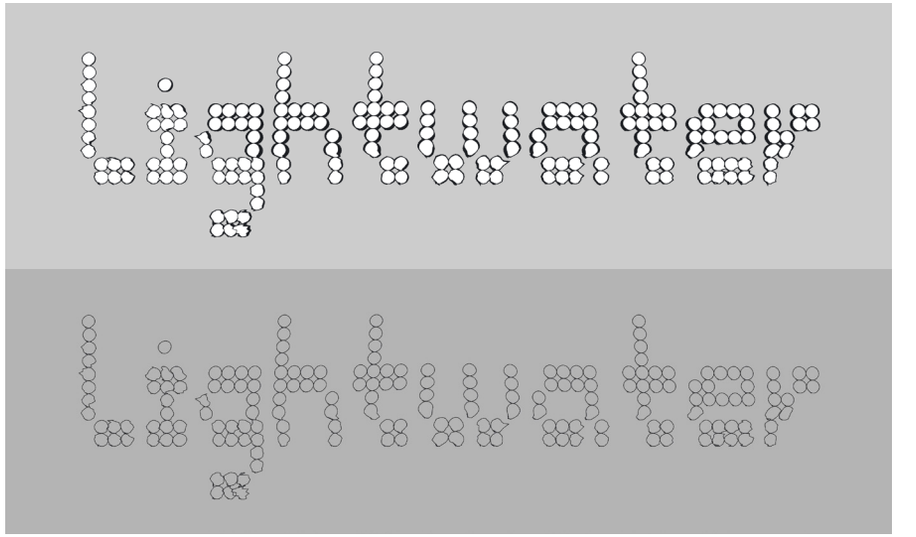
Input (left)

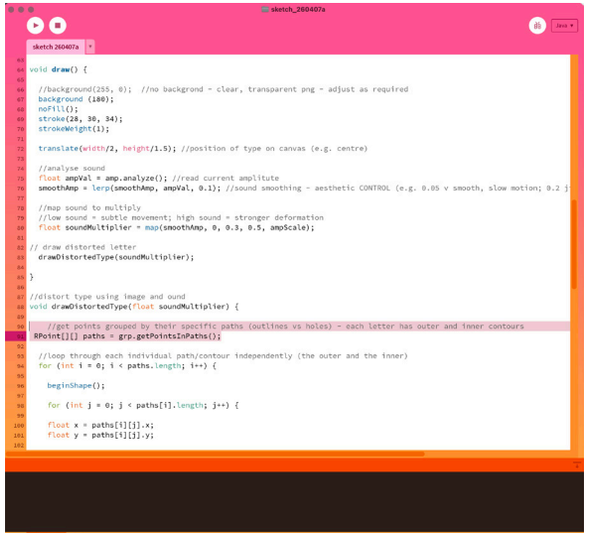
Typeface: Pixelfont-Circle
Image: P1101569.jpg
Audio: Lightwater_01.aif

ALGORITHM REFINEMENT

Initial mapping produced inconsistent typographic structures due to unstable path relationships. This was resolved by grouping inner and outer contours, resulting in more coherent and stable letterforms.







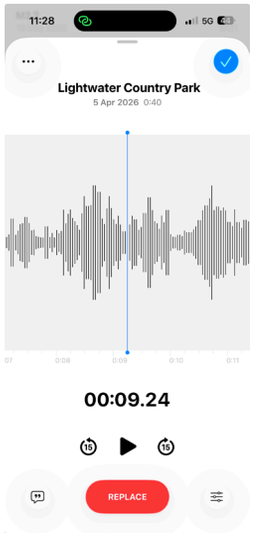
Input (from top)
Typeface: Pixelfont-Circle
Image: P1101548.jpg
Audio: Lightwater_01.aif

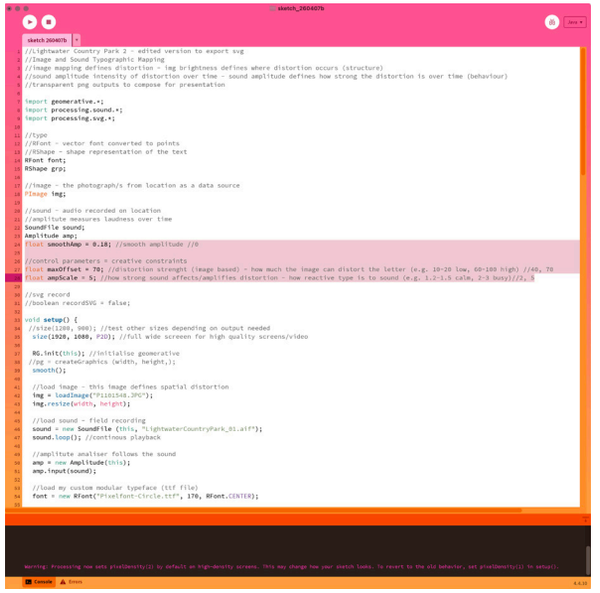
Code excerpt

Contour representation

Top: initial combined
Middle: outline
Bottom: filled

The system operates on vector paths, treating inner and outer contours independently. Testing different render strategies highlighted how outline-based structures emphasise distortion, while filled forms reinforce overall legibility.





Input (from top)
Typeface: Pixelfont-Circle
Image: P1101548.jpg
Audio: Lightwater_01.aif

Code excerpt

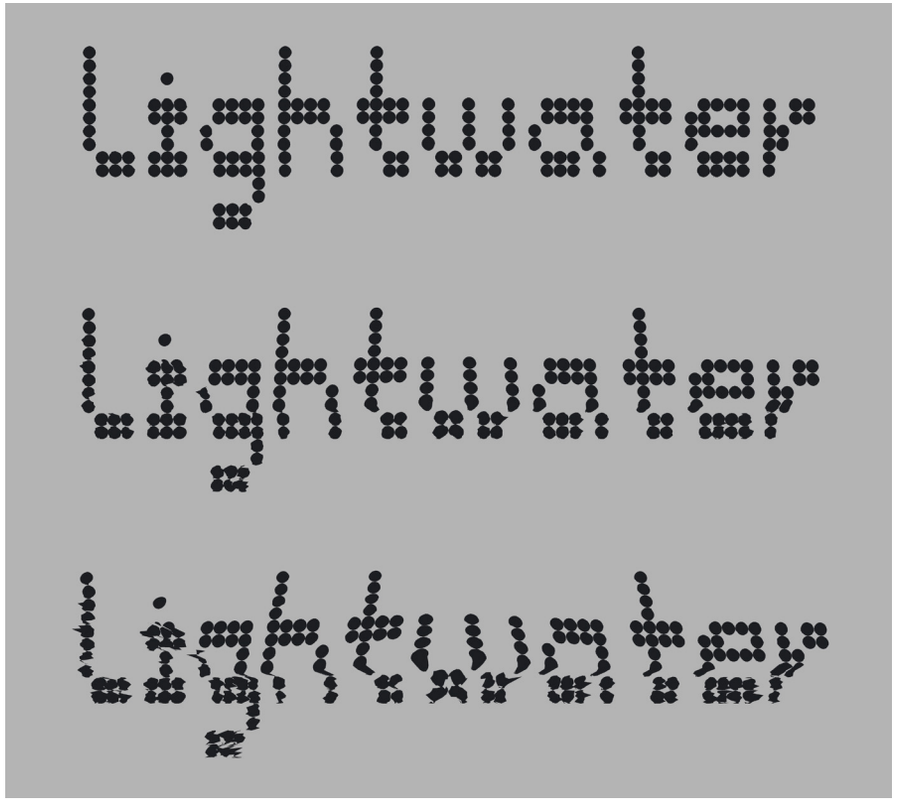


Image defines structure, sound defines behaviour.

Distortion (top)
maxOffset: 20
ampScale: 1.2
smoothAmp: 0.05

Distortion (middle)
maxOffset: 40
ampScale: 2
smoothAmp: 0.1

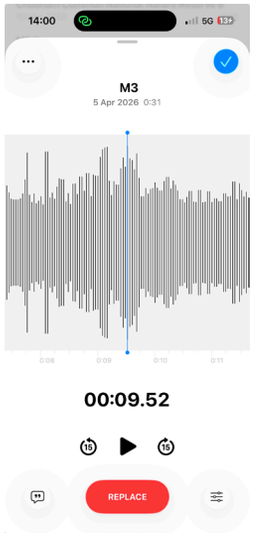
Distortion (bottom)
maxOffset: 70
ampScale: 5
smoothAmp: 0.18

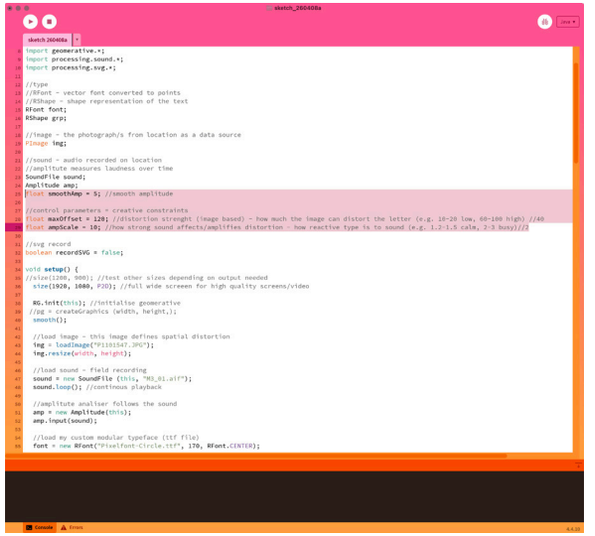
The same system behaves differently under adjusted parameter conditions. Modifying key variables such as maximum offset, amplitude scaling, and smoothing directly influences the intensity, responsiveness, and stability of typographic distortion.

Higher maxOffset:
increased spatial displacement

Higher ampScale:
stronger audio-driven modulation

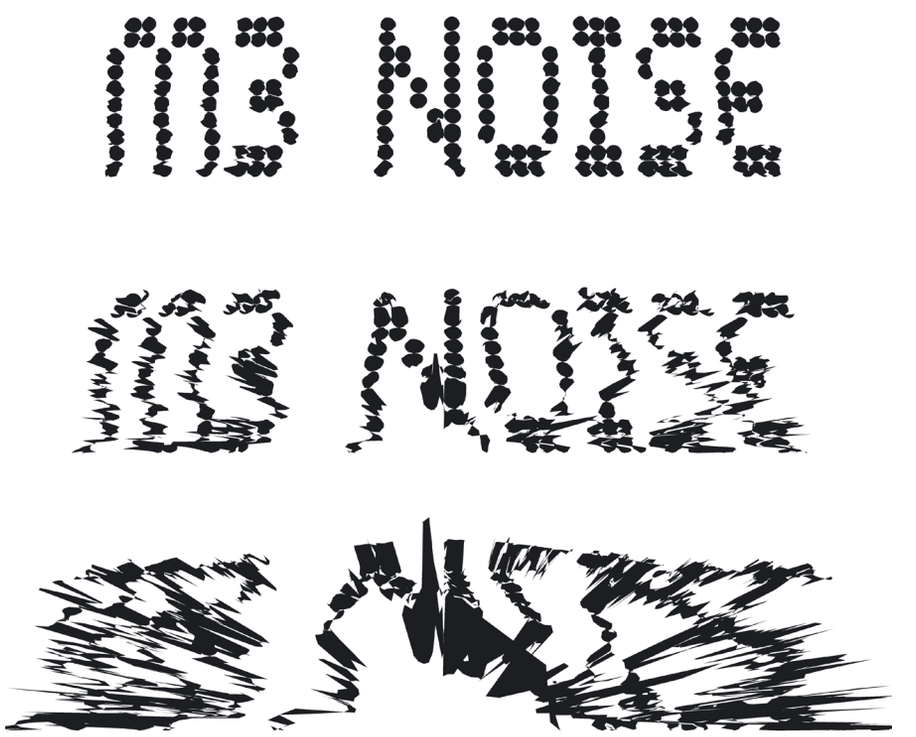
Lower smoothing:
more reactive, unstable behaviour





Input (from top)
Typeface: Pixelfont-Circle
Image: P1101547.jpg
Audio: M3_01.aif

Code excerpt



Distortion (top)

maxOffset: 30
ampScale: 3
smoothAmp: 0.12

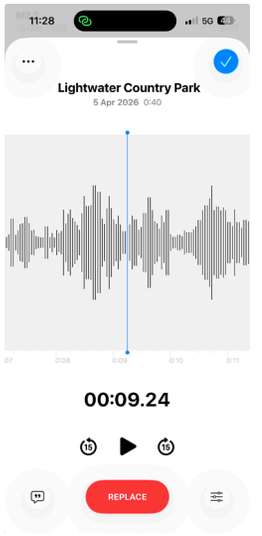
Distortion (middle)

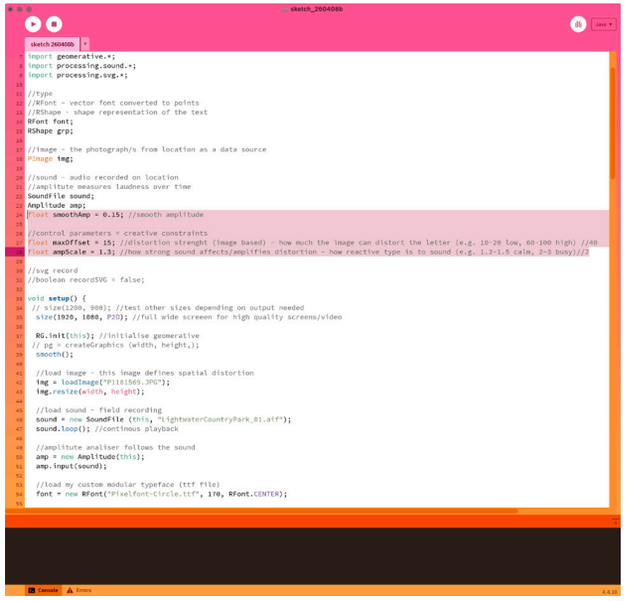
maxOffset: 80
ampScale: 5
smoothAmp: 0.08

Distortion (bottom)

maxOffset: 120
ampScale: 10
smoothAmp: 0.18

Recorded at the bridge over the M3, this dataset introduces directional and rhythmic distortion influenced by passing traffic. Unlike organic environments, the system responds with more linear, repetitive, and disrupted behaviours aligned with the structured and mechanical nature of infrastructure.





Input (from top)
Typeface: Pixelfont-Circle
Image: P1101569.jpg
Audio: Lightwater_01.aif

Code excerpt

Distortion (top)

maxOffset: 15
ampScale: 1.2
smoothAmp: 0.15

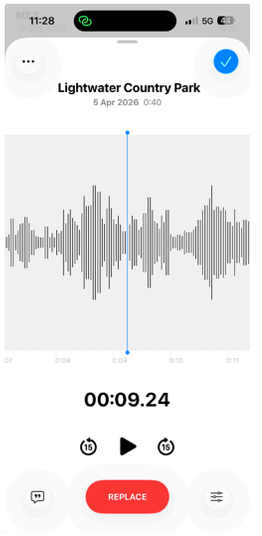
Distortion (middle)

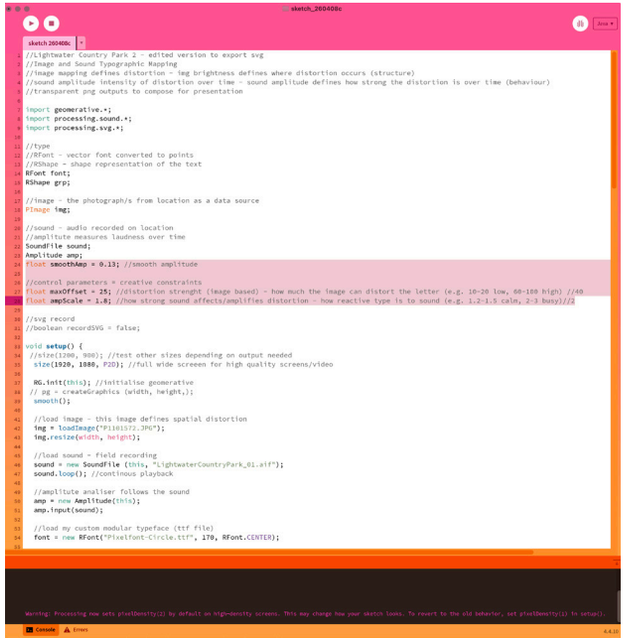
maxOffset: 25
ampScale: 1.8
smoothAmp: 0.12

Distortion (bottom)

maxOffset: 40
ampScale: 3
smoothAmp: 0.1

In contrast to infrastructure contexts, sky inputs produce subtle and continuous distortion. Lower parameter values and increased smoothing reduce visual disruption, resulting in calm, atmospheric typographic outputs.





Input (from top)
Typeface: Pixelfont-Circle
Image: P1101569.jpg
Audio: Lightwater_01.aif

Code excerpt

Distortion (top)

maxOffset: 25
ampScale: 1.8
smoothAmp: 0.13

Distortion (middle)

maxOffset: 40
ampScale: 2.5
smoothAmp: 0.1

Distortion (bottom)

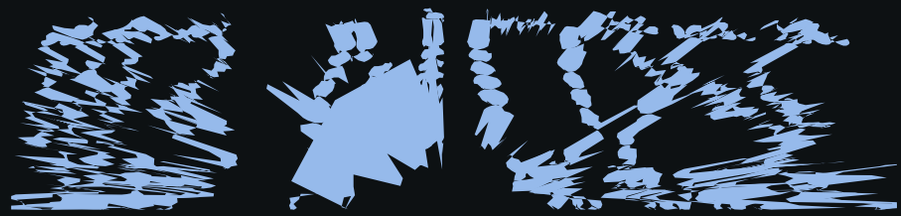
maxOffset: 65
ampScale: 3.5
smoothAmp: 0.08

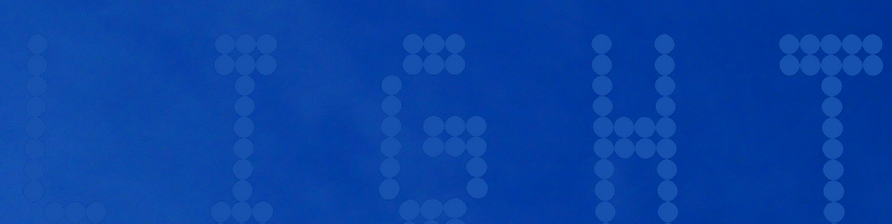
Woodland inputs introduce irregular and locally varied distortion, driven by the high contrast between light and shadow. Moderate parameter values and reduced smoothing produce responsive yet continuous typographic behaviour, reflecting the layered and dynamic nature of the forest. Variations in image brightness result in uneven deformation across the letterforms, reinforcing the fragmented but cohesive reading of the environment.

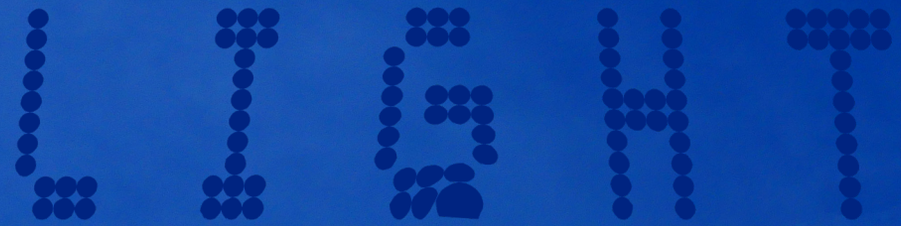
PRESENTATION TESTING

Presentation testing explored colour application derived directly from image data. While this approach strengthened the connection between input and output, it reduced typographic clarity and consistency. As a result, this method was rejected in favour of a more controlled and legible presentation system.

Colour outputs were tested as an extension of the system, introducing layered variations. While these emphasised dynamic behaviour, they introduced visual noise and fragmentation.



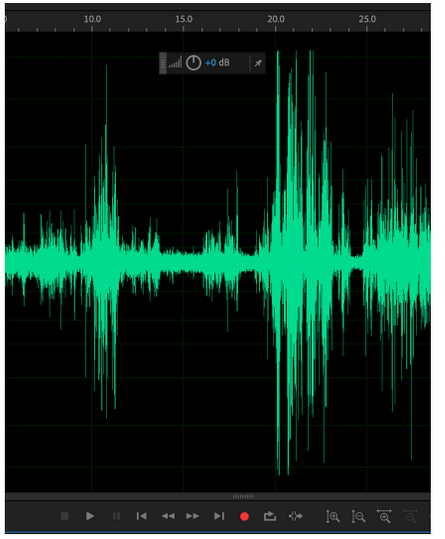





Hiking
Lapland

09

System Logic



ALGORITHM:
SYSTEM RULES AND CREATIVE CONTROL

The system operates through a set of consistent rules that define how image and sound data are translated into typographic form.

Image brightness determines spatial distortion, while sound amplitude controls temporal intensity. Typeface and dataset selection introduce variation without altering the underlying logic of the system. This distinction allows the process to remain structured while producing diverse visual outcomes.

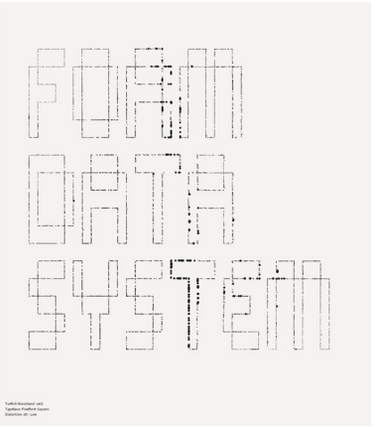
While the core mapping remains constant, parameters such as distortion intensity, smoothing, and representation mode act as points of creative control, enabling adjustments in density, legibility, and visual behaviour.

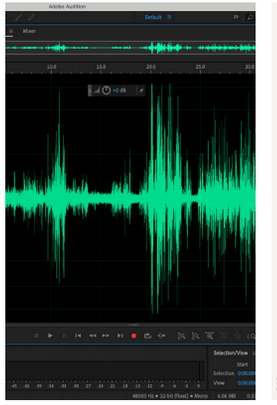
```
mp 260527b
250   for (int i = 0; i < lines.length; i++) {
251
252     RShape grp = font.toShape(lines[i]);
253     float yPos = startY + i * lineSpacing;
254
255     pushMatrix();
256     translate(startX, yPos);
257
258     drawDistortedType(grp, soundMultiplier, startX, yPos);
259     popMatrix();
260   }
```

```
mp 260527b
302 //core distortion system
303
304 void drawDistortedType(RShape grp, float soundMultiplier, float offsetX, float offsetY) {
305
306   RPoint[][] paths = grp.getPointsInPaths(); //outline only
307
308   //distortion strength depending on mode (0-Low, 1-Med, 2-High)
309   float distortionStrength;
310
311   if (distortionMode == 0) distortionStrength = 10;
312   else if (distortionMode == 1) distortionStrength = 25;
```

```
mp 260527b
338
339     float b = brightness(img.get(sampleX, sampleY));
340
341     //distortion
342     float imgOffset = map(b, 0, 255, 0, distortionStrength);
343     float offset = imgOffset * soundMultiplier;
344
345     float newX, newY;
346
347     if (distortionMode < 2) {
348       float angle = atan2(y, x);
```

```
mp 260527b
241 //sound analysis
242
243   float ampVal = amp.analyze();
244   smoothAmp = lerp(smoothAmp, ampVal, 0.08);
245   float soundMultiplier = map(smoothAmp, 0, 0.3, 0.5, ampScale);
246
247 //.....................................................................
248 //text
249
250   for (int i = 0; i < lines.length; i++) {
251
```



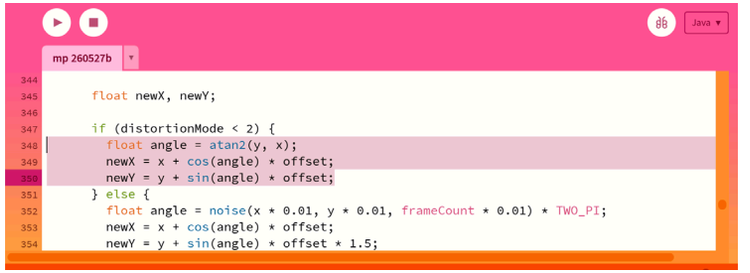


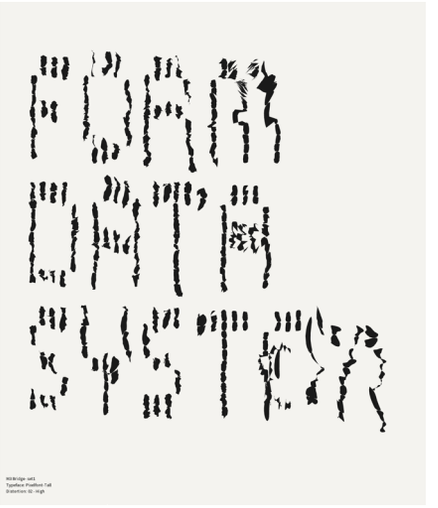
STRUCTURE

Typography is converted into vector shapes and sampled into point-based paths, forming the underlying structure through which distortion is applied.

DATA INPUT

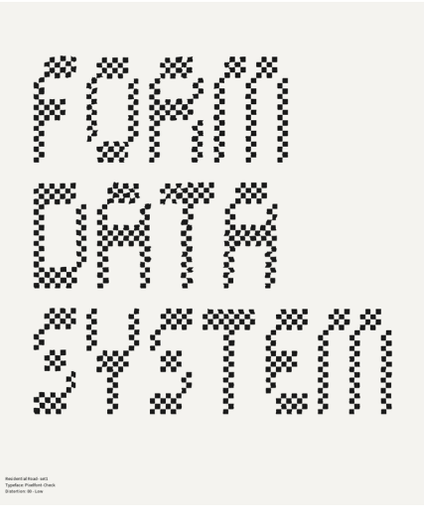
Image brightness defines spatial variation, while sound amplitude controls the intensity and behaviour of distortion.





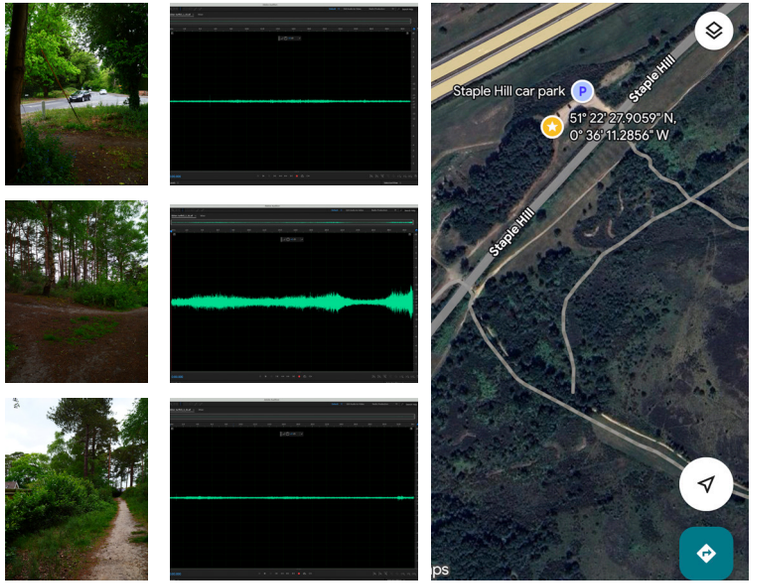
DISTORTION BEHAVIOUR

Each point is displaced based on mapped input values, generating spatial distortion that reflects environmental conditions.



REPRESENTATION

The system supports multiple output modes, separating point-based structure from continuous typographic form.



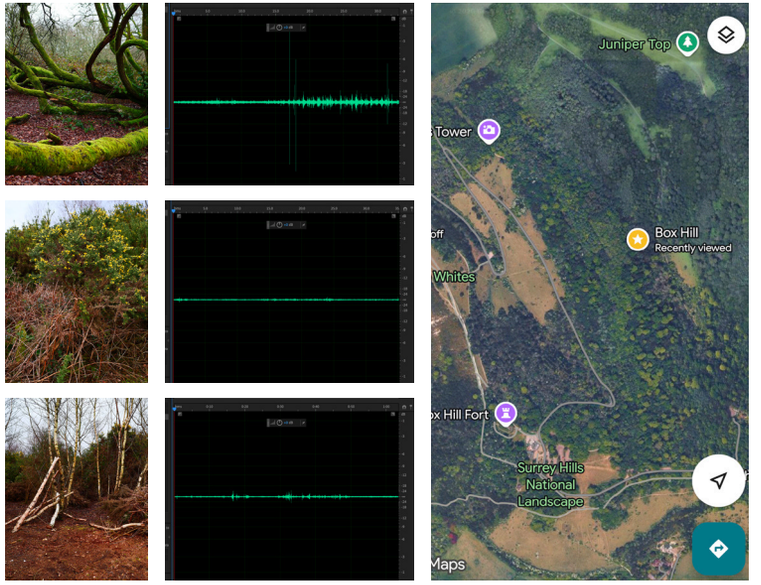
Location: Turfhill (Woodland)
Data set: turfhill_1, turfhill_2, turfhill_3

Location: Simplemarsh Road, Addlestone (Residential Road)
Data set: resident_1, resident_2, resident_3

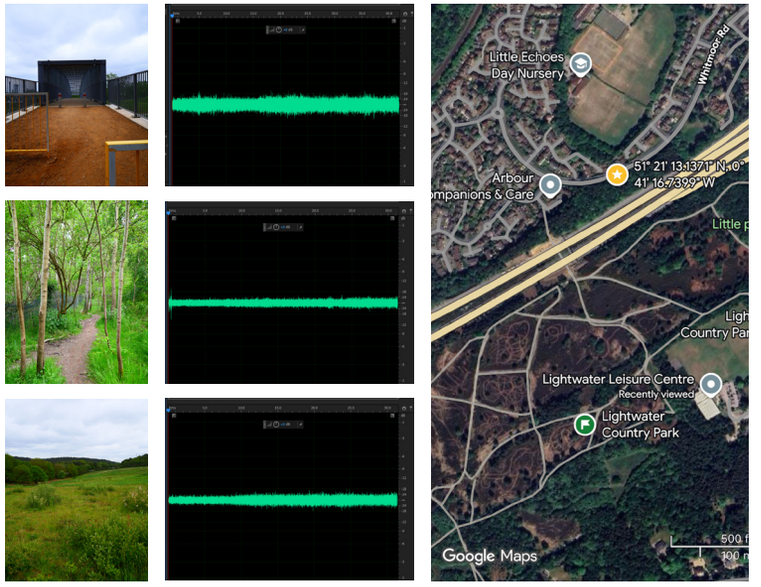
INPUT DATA

The system operates across multiple locations, each defined by a set of image and sound inputs. These datasets establish the environmental conditions that drive typographic variation in subsequent outputs.

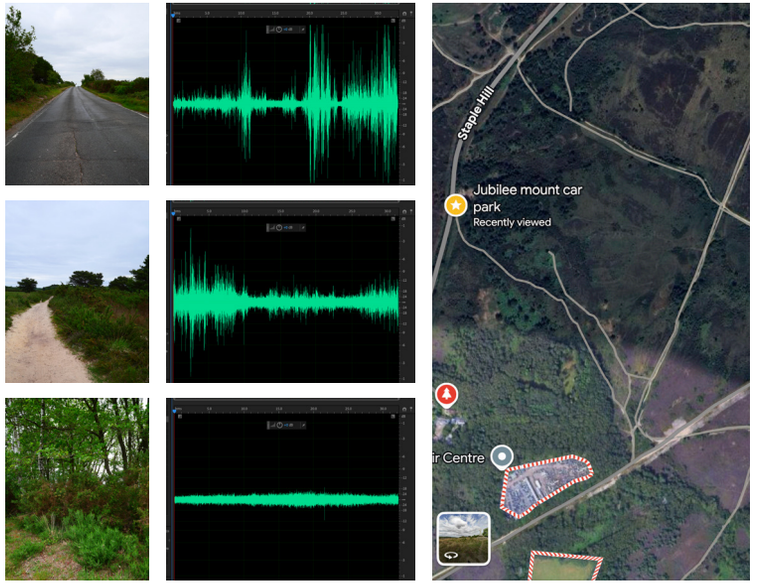
Input data sets and location maps (pp. 154, 156-157)



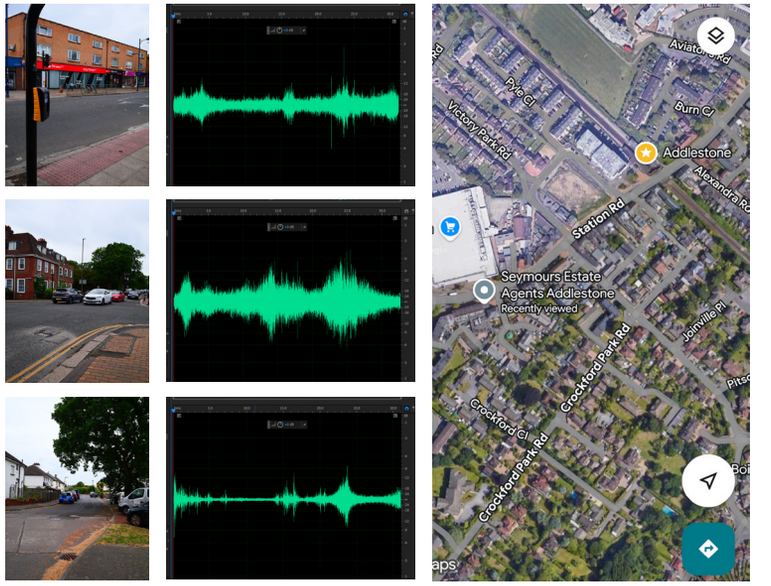
Location: Box Hill lower (Footpath Network - vertical variation)
Data set: boxhill_1, boxhill_2, boxhill_3



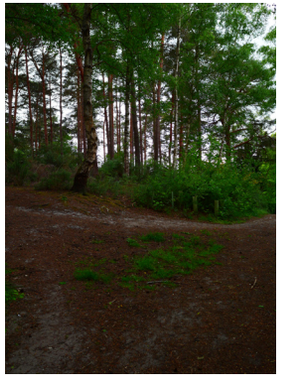
Location: M3 Pedestrian Bridge (Infrastructure)
Data set: m3_1, m3_2, m3_3

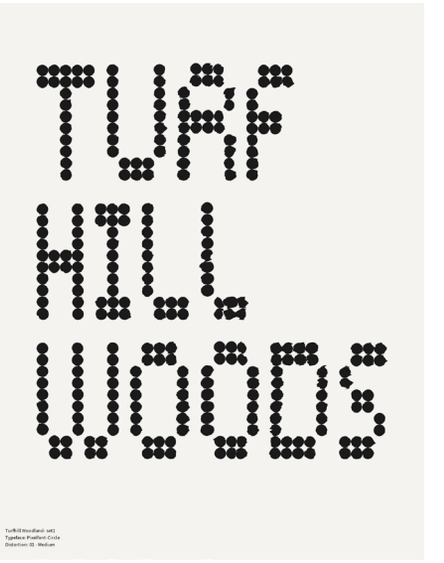


Location: Chobham Common (Open Heath)
Data set: chobham_1, chobham_2, chobham_3



Location: Station Road, Addlestone (Pedestrian Route)
Data set: station_1, station_2, station_3





TYPEFACE VARIATION

Different modular typefaces produce varied visual density and texture under identical system conditions, while overall distortion behaviour remains consistent.

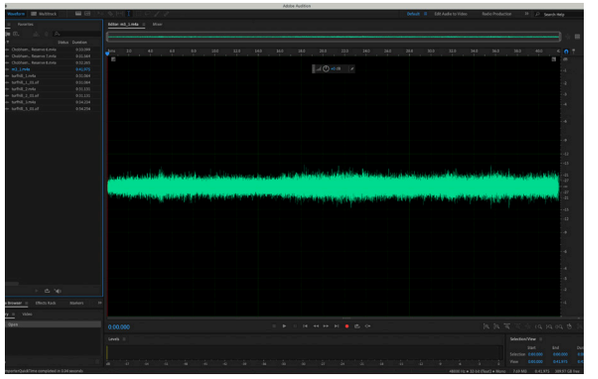
Inputs (from top):

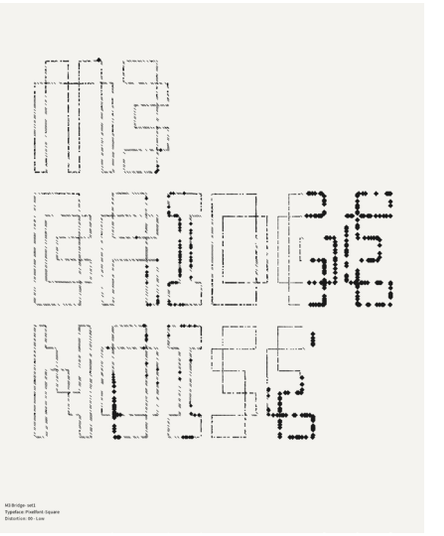
Typeface: Pixelfont-Circle, Pixelfont-Flat, Pixelfont-Square, Pixelfont-Pattern
Location: Turfhill (woodland)
Data set: turfhill_1

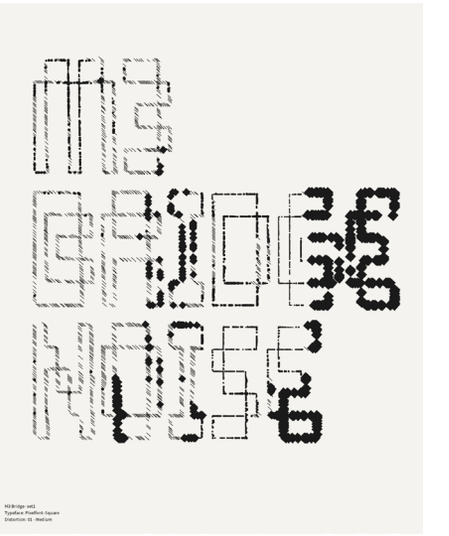
Outputs:

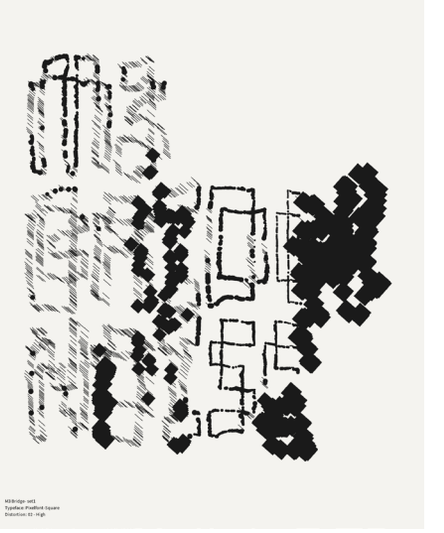
Distortion mode: Medium
Per typeface

M3 BRIDGE NOISE









DISTORTION MODES

Increasing distortion intensity alters spatial displacement and structural coherence, shifting outputs from stable to highly expressive.

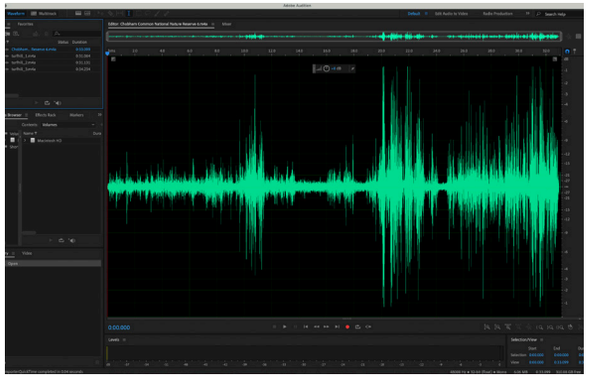
Inputs (from top):

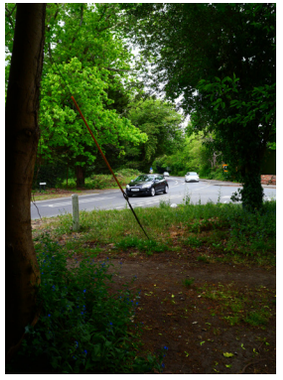
Typeface: Pixelfont-Square
Location: M3 Bridge (infrastructure)
Data set: m3_1

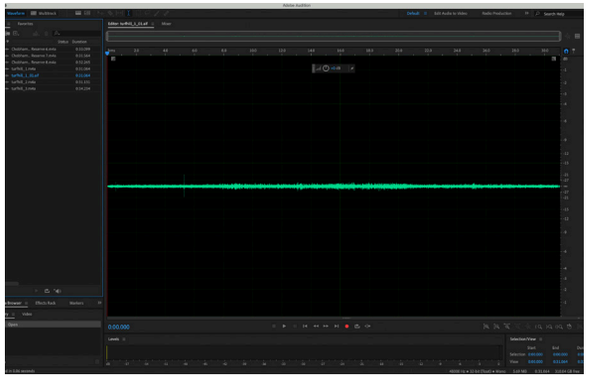
Outputs:

Distortion modes:
Low, Medium, High

LOC TION







Inputs (left, from top)

Typeface: Pixelfont-Circle
Location 1 set: m3_1
Location 2 set: chobham_1
Location 3 set: turfhill_1

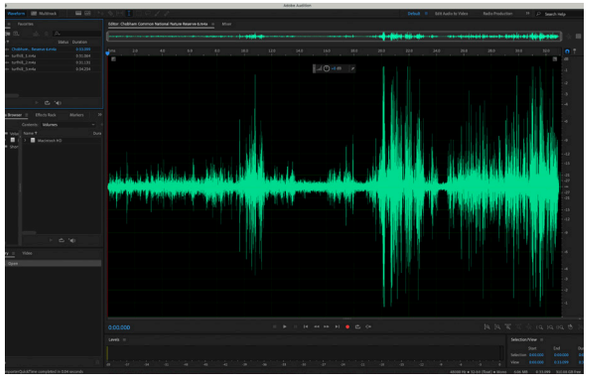
Outputs per location (top)

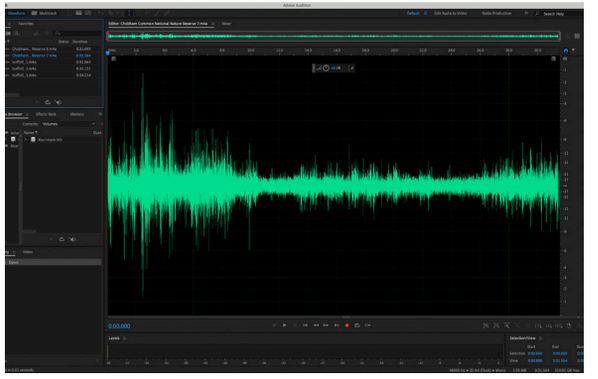
Distortion mode: High

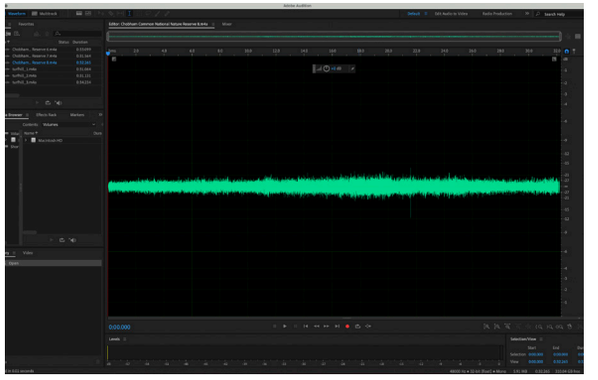
LOCATION

The system produces distinct typographic behaviours across environments, reflecting differences in spatial composition and audio input.

PROCESS D T FORM







Inputs (left, from top)

Typeface: Pixelfont-Dot
Location: Chobham Common (open heath)
Data sets: chobham_1, chobham_2,chobham_3

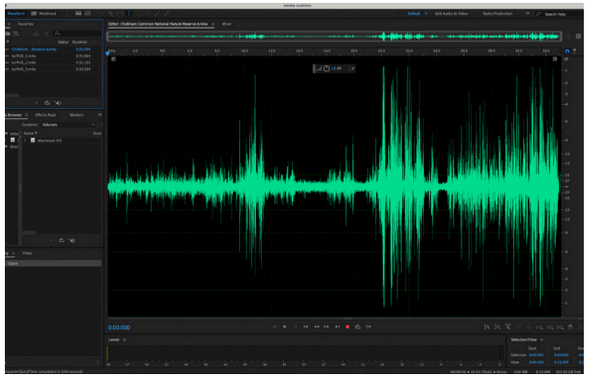
Outputs:

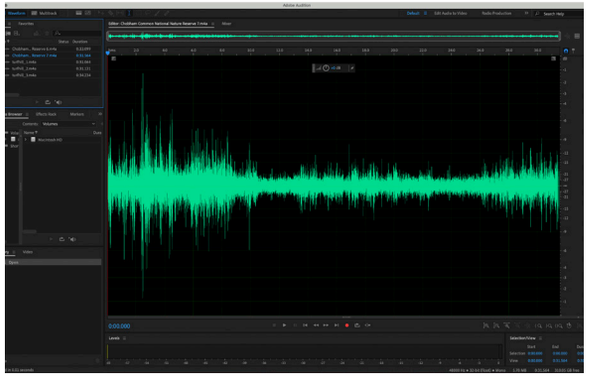
Distortion mode: High

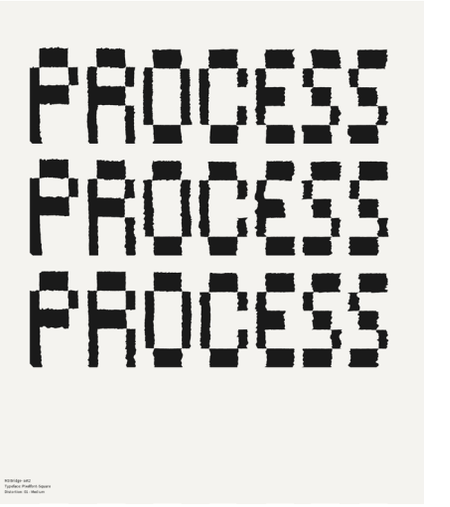
WITHIN-LOCATION

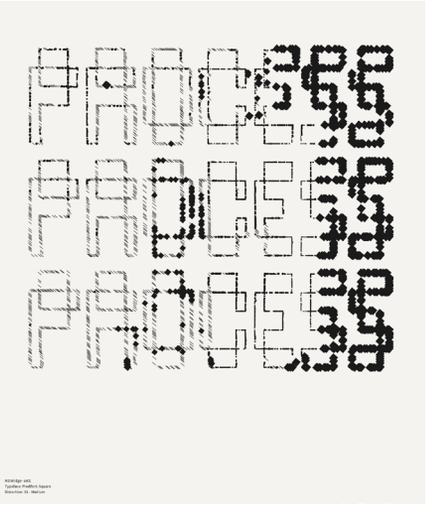
Variations within a single location reveal how small changes in image and sound conditions influence localised distortion behaviour.

PROCESS









REPRESENTATION MODES

Representation modes distinguish between structural and perceptual readings of the system, with outline outputs revealing underlying geometry and filled forms emphasising visual density and legibility.

Inputs (from top)

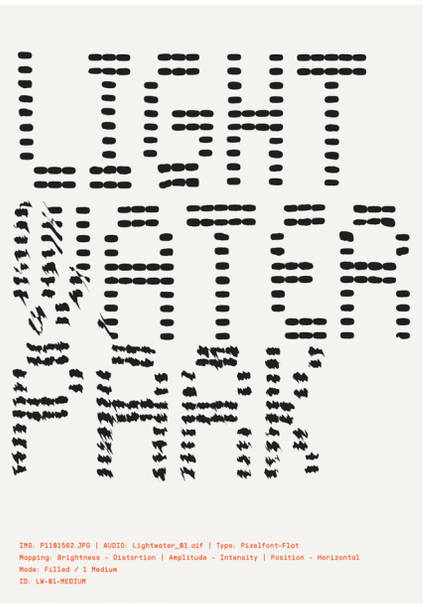
Typeface: Pixelfont-Square
Location: M3 Bridge (infrastructure)
Data sets: m3_1, m3_2

Outputs (top left)
Outline: m3_2 and m3_1

Outputs (top right)
Filled: m3_2 and m3_1

10

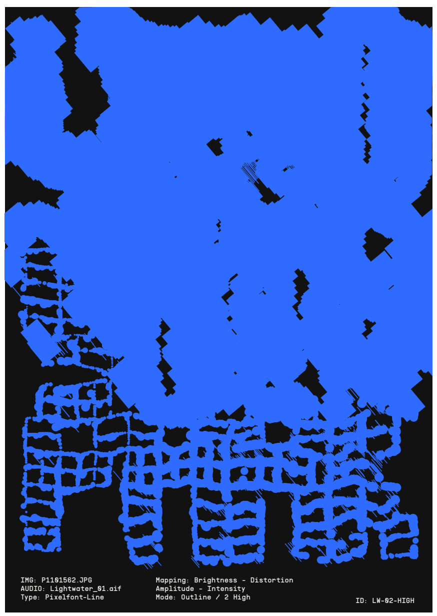
Results and Testing

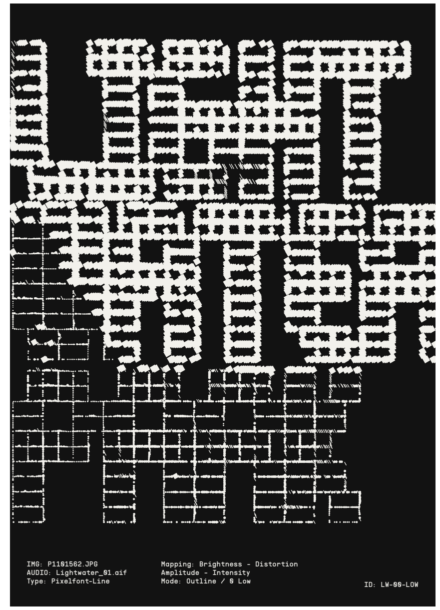


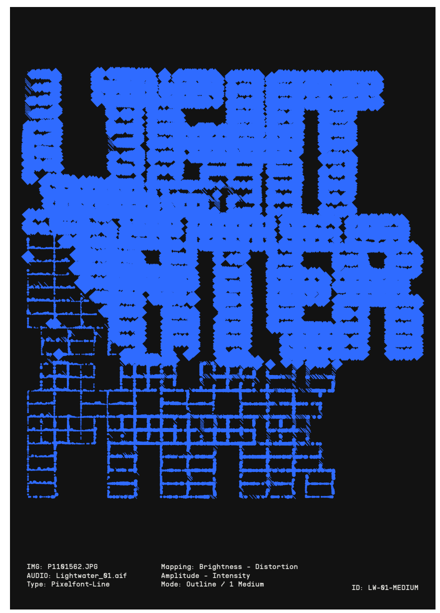
Testing variation of distortion, colour, supporting text.

LAYOUT DEVELOPMENT

A series of test posters were produced to explore variations in composition, typographic density, colour, and distortion.



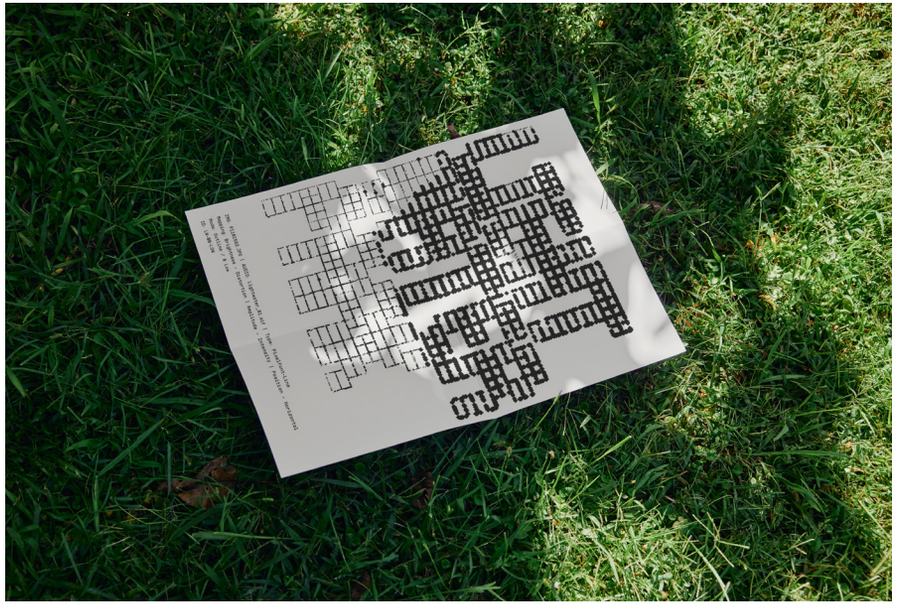




Testing limits of distortion, layouts, colour inversion, legibility, supporting text.

At higher intensities, typographic forms begin to collapse into abstract textures, reducing legibility while increasing expressive potential.

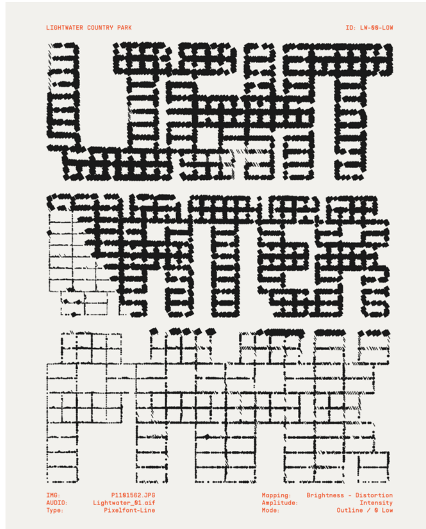
These experiments allowed evaluation of where the system transitions from readable typography to purely visual form, informing the selection of outputs for final compositions.

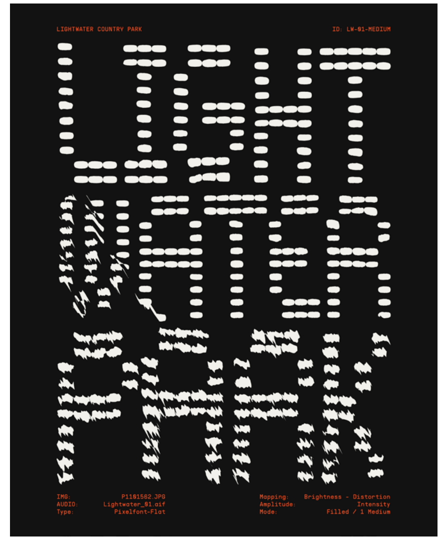


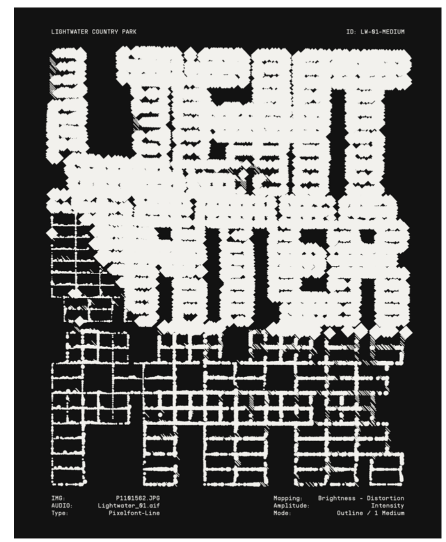
CONTEXT TESTING

Early poster iterations were tested within environmental contexts to assess scale and visibility. These tests revealed limitations in legibility and compositional balance, informing subsequent refinements in layout and typographic clarity.

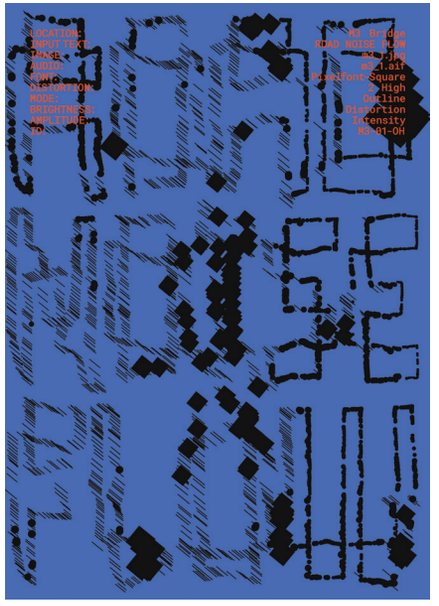
These outcomes were not taken forward, but informed later design decisions.

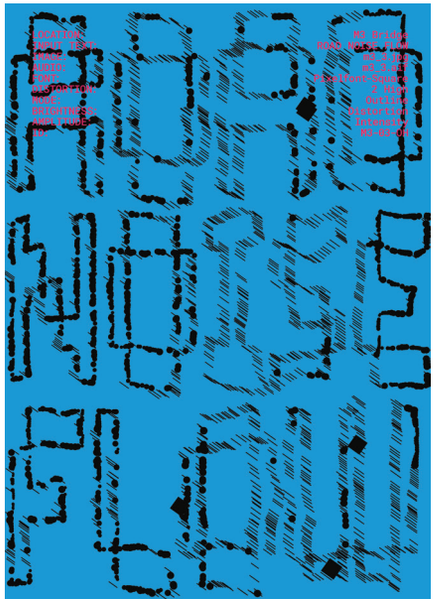






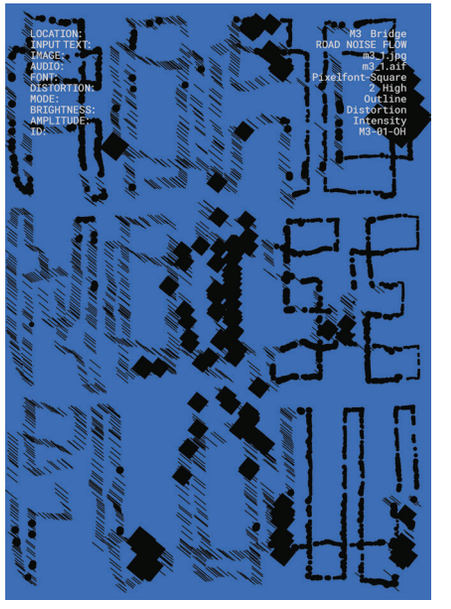
Iterative development allowed refinement of scale, spacing, and contrast and contextual information, leading to more coherent compositions.

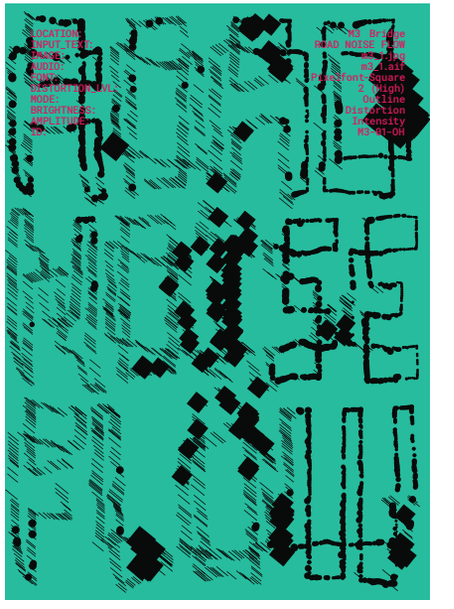




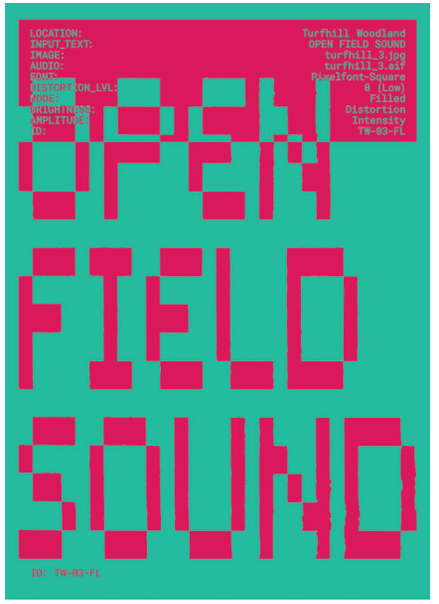
Further colour testing.

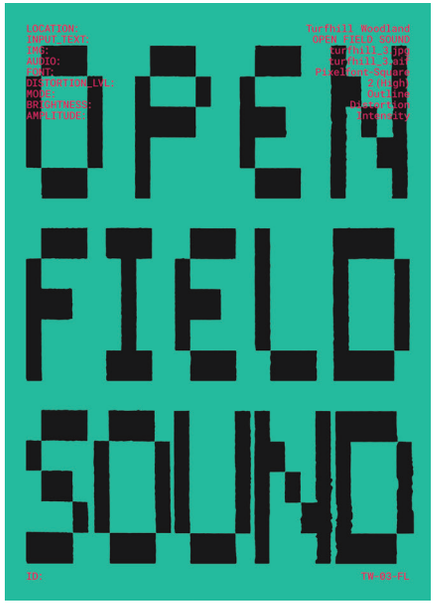
Metadata becomes a visual interface, not a caption.





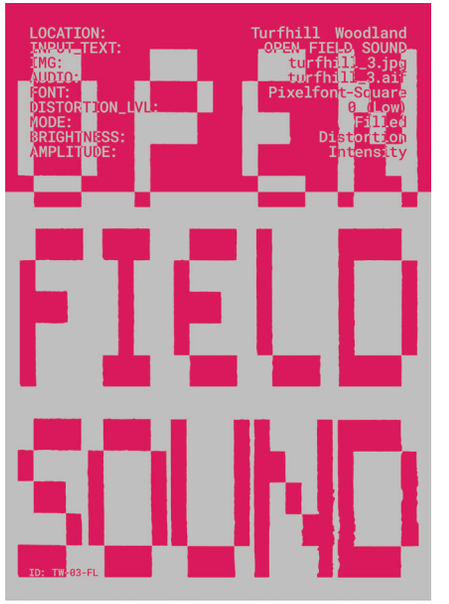
Stronger colour combinations (green, blue, magenta) were selected for final outcomes to maintain contrast under varying distortion levels

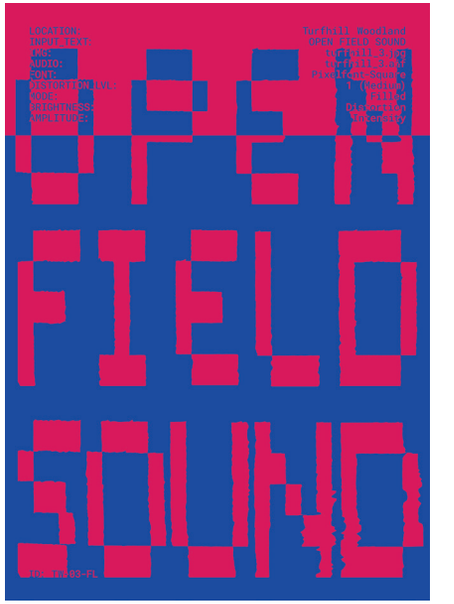


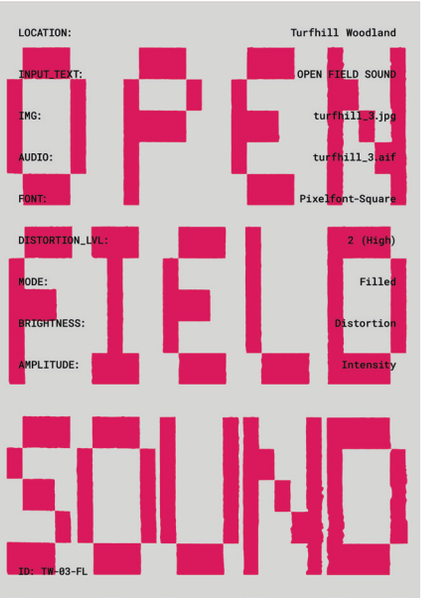


Colour tests explored inversion, saturation, and foregroun and background relationships.

Refinement focused on establishing a controlled, limited colour system to support the input–process–output structure.







Alignment testing explored how the positioning of metadata could reinforce the relationship between input and output.

By distributing labels and values across the composition, the metadata operates as a structured system layered onto the generative forms, rather than a separate caption.

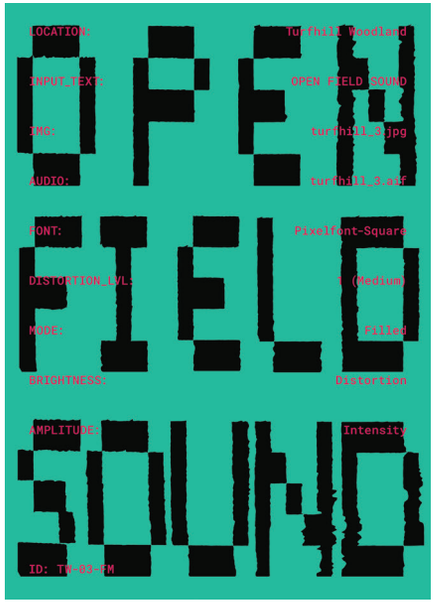
11

Selected Outputs

OUTPUT EXAMPLES

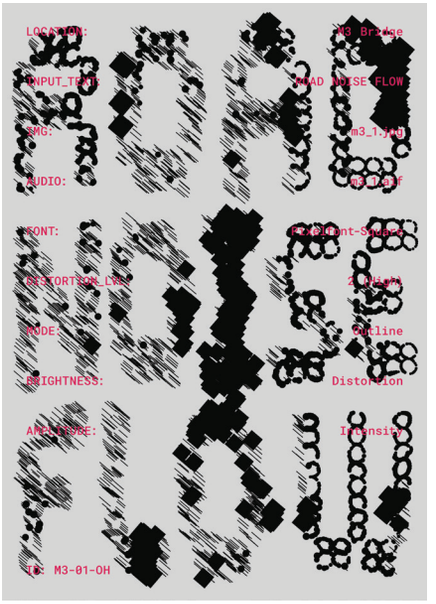
A consistent layout system was established to maintain a stable relationship between input and output across all variations. By fixing the position of metadata each composition operates as part of a larger system rather than an isolated design.

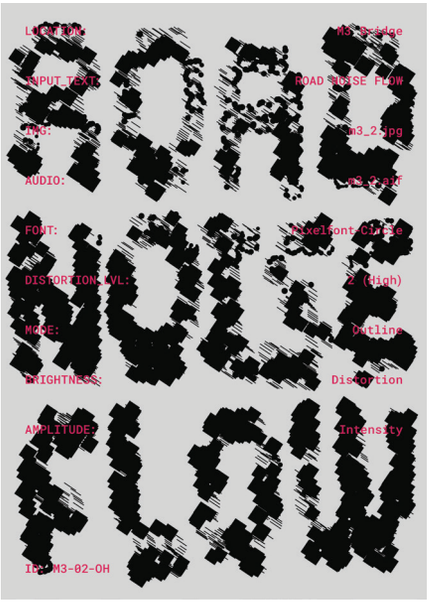
This approach enables controlled variation, where differences are driven by data while the overall structure remains constant.

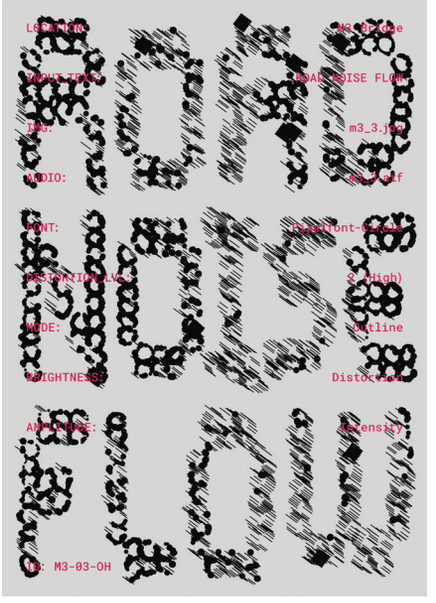


DISTORTION MODES

One dataset is translated through three distortion levels (low, medium, high), showing how variation is produced through changes in system parameters rather than input.

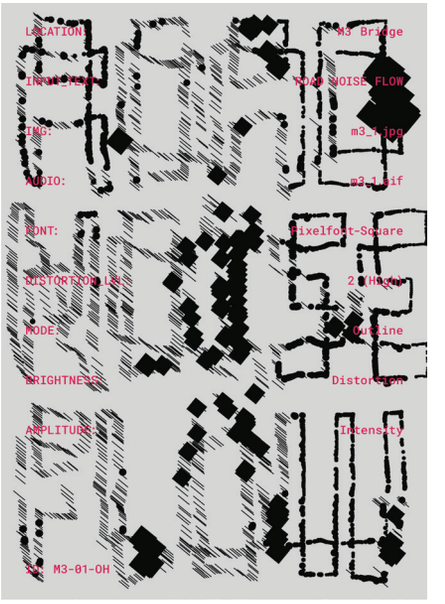


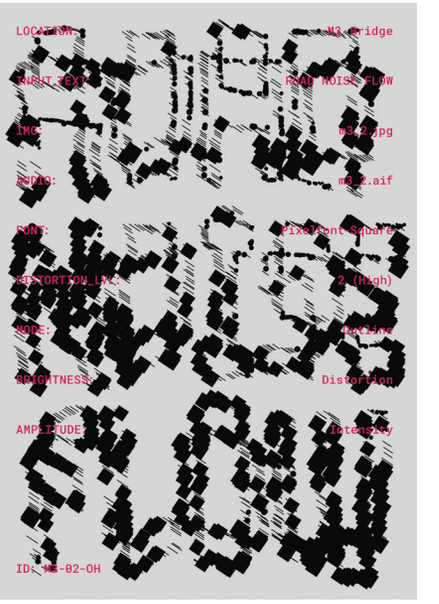


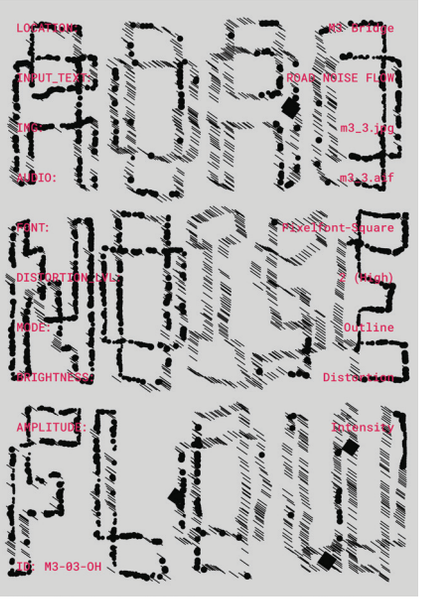


VARIATIONS WITHIN A SINGLE LOCATION

Three datasets from the same location were processed using identical parameters (high distortion mode) while applying the same typeface, PixelFont Circle. Variations in the output are driven by differences in the input data (image and audio), demonstrating how the system produces distinct typographic forms from multiple recordings within a single environment.







REPRESENTATION MODES AND TYPEFACE VARIATION

The same three datasets were processed under identical conditions (high distortion mode), with the only variable being the typeface, PixelFont Square, showing its impact on structure and visual density.

LOCATION

Three different locations are processed using identical parameters, isolating location as the only variable within the system. Despite using the same distortion settings and typographic structure, each output generates a distinct visual result, demonstrating how variations in recorded data influence the transformation of form.

12

Conclusion

CRITICAL REFLECTION: PROCESS DATA PLACE

The development of this project demonstrates how typographic design can operate as a system in which variation emerges through defined rules, parameters, and data-driven processes. From the outset, the aim was not to design individual letterforms as fixed outcomes, but to construct a system capable of generating multiple forms through execution.

Throughout the process, the focus shifted from form-making to system-building, where letterforms are produced as consequences of structure rather than direct aesthetic decisions. This approach allowed typography to function as an evolving framework, shaped by interaction between rules and data inputs. In doing so, the project negotiates the tension between legibility and abstraction, where clarity is not fixed but emerges through carefully defined constraints.

The integration of visual and sonic data established a relationship between place and form, where typographic behaviour became a translation of environmental conditions. Iterative experimentation revealed that maintaining coherence required carefully defined constraints, ensuring that variation remained controlled and meaningful.

As a result, authorship shifts from producing individual forms to designing the conditions from which form emerges. The designer defines the system, while the system generates the outcomes. This reflects a broader shift in contemporary design practice, where processes and behaviours take precedence over static compositions. However, this reliance on data also introduces limitations related to availability, interpretation, and legibility, requiring careful framing to maintain clarity and ethical responsibility.

Ultimately, the project demonstrates that generative typography extends established typographic systems, positioning structure, interaction, and process as the primary drivers of form.

13

Appendix

PROCESSING CODE

```
// Monika Sowa - Major Project - Process Data Place
// Process/System: Modular Type + Image + Sound
// Location: Turfhill, Surrey

// 1. Converts text into points (structure)
// 2. Distorts points using image + sound (behaviour)
// 3. Rebuilds typography using geometric glyphs (visual language)

//    L - switch location
//    N - switch dataset within location n
//    F - change typeface
//    D - change distortion mode
//    C - change colour

//    S - save SVG
//    P - save PNG

import geomerative.*;
import processing.sound.*;
import processing.svg.*;

//type system
RFont font;
int fontIndex = 0;

//Custom font variants
String[] fonts = {
  "Pixelfont-Check.ttf",
  "Pixelfont-Circle.ttf",
  "Pixelfont-Dot.ttf",
  "Pixelfont-Flat.ttf",
  "Pixelfont-Half.ttf",
  "Pixelfont-Pattern.ttf",
  "Pixelfont-Rounded.ttf",
  "Pixelfont-Square.ttf",
  "Pixelfont-Star.ttf",
  "Pixelfont-Tall.ttf",
  "Pixelfont-Triangle.ttf"
};

//location data sets
String[] locationNames = {
  "Turfhill Woodland",
  "Chobham Heath",
  "M3 Bridge",
  "Station Road",
  "Residential Road",
  "Surrey Hills"
};

//images per location
String[][] images = {
  {"turfhill_1.jpg", "turfhill_2.jpg", "turfhill_3.jpg"},
  {"chobham_1.jpg", "chobham_2.jpg", "chobham_3.jpg"},
  {"m3_1.jpg", "m3_2.jpg", "m3_3.jpg"},
  {"station_1.jpg", "station_2.jpg", "station_3.jpg"},
  {"resident_1.jpg", "resident_2.jpg", "resident_3.jpg"},
  {"boxhill_1.jpg", "boxhill_2.jpg", "boxhill_3.jpg"}
};

//sounds per location
String[][] sounds = {
  {"turfhill_1.aif", "turfhill_2.aif", "turfhill_3.aif"},
  {"chobham_1.aif", "chobham_2.aif", "chobham_3.aif"},
  {"m3_1.aif", "m3_2.aif", "m3_3.aif"},
  {"station_1.aif", "station_2.aif", "station_3.aif"},
  {"resident_1.aif", "resident_2.aif", "resident_3.aif"},
  {"boxhill_1.aif", "boxhill_2.aif", "boxhill_3.aif"}
};

int locationIndex = 0;
int dataIndex = 0;

//image and sound
PImage img;

//sound
SoundFile sound;
Amplitude amp;
float smoothAmp = 0;

//export
boolean recordSVG = false;
boolean savePNG = false;

//text
//"Phrase" 'Letter'

String[] lines = {

//  "TURF",
//  "HILL",
//  "WOODS"

//  "PROCESS",
//  "DATA",
//  "PLACE"
};
```

```
//layout (edit as required)
float margin;
float startX;
float startY;
float lineSpacing;

//system controls
float maxOffset = 12;
float ampScale = 2.0;
float pointDensity = 4;

//distortion modes
int distortionMode = 0;
// 0 = subtle
// 1 = medium
// 2 = high

//labels
String[] distortionLabels = {“00 - Low”, “01 - Medium”, “02 - High”};
String[] renderLabels = {“Outline”, “Filled”};
//String[] colourLabels = {“Black”, “Blue”, “Orange”};

//render modes - outline and fill
int renderMode = 0;
//0 - outline, 1 - filled

//colour modes
int colourMode = 0;
// 0 = soft black
// 1 = blue
// 2 = orange

//set up
void setup() {
 //size(1240, 1754, P2D); // A3 Poster
 //size(1920, 1080, P2D); // wide
 //size(1080, 1350, P2D); // IG Post

 //size(1300, 1600, P2D); // book dev images 1200 1500
  size(1400, 1600, P2D); // book dev images 1200 1500

 //size(540, 675, P2D); // IG Post/2
  smooth();

  RG.init(this);
  font = new RFont(fonts[fontIndex], 200, RFont.LEFT); //200

//layout controls
  margin = 110; //110, 80
  startX = margin;
  startY = margin + 400; //400
```



```
  lineSpacing = 400; //360

//point sampling density
  RCommand.setSegmentLength(pointDensity);
  RCommand.setSegmentator(RCommand.UNIFORMLENGTH);

//sound system
  amp = new Amplitude(this);

//load first data set
  loadData();
}

//load image and sound
void loadData() {
  img = loadImage(images[locationIndex][dataIndex]);
  img.resize(width/2, height/2);

  if (sound != null) sound.stop();

  sound = new SoundFile(this, sounds[locationIndex][dataIndex]);
  sound.loop();
  amp.input(sound);

  println(“Location: “ + locationNames[locationIndex] + “ | Set: “ + (dataIndex + 1));
}

void draw() {
  if (recordSVG) {
    beginRecord(SVG, “poster_” + fonts[fontIndex] + “_mode” + distortionMode + “.svg”);
  }

  //background off white
  background(244, 243, 239);

//sound analysis
  float ampVal = amp.analyze();
  smoothAmp = lerp(smoothAmp, ampVal, 0.08);
  float soundMultiplier = map(smoothAmp, 0, 0.3, 0.5, ampScale);

//text
  for (int i = 0; i < lines.length; i++) {

    RShape grp = font.toShape(lines[i]);
    float yPos = startY + i * lineSpacing;

    pushMatrix();
    translate(startX, yPos);

    drawDistortedType(grp, soundMultiplier, startX, yPos);
    popMatrix();
  }
```

```
  drawUI();
}

//display current data set  - on screen
void drawUI() {

  fill (0);
  textSize(14);

  String currentFont = fonts [fontIndex].replace(".ttf", "");

//left
  text (locationNames[locationIndex] + "- set" + (dataIndex +1), 20, height - 80);
  text ("Typeface: " + currentFont, 20, height -60);
  text ("Distortion: " + distortionLabels[distortionMode], 20, height -40);

//right side
 //text ("Render: " + renderLabels[renderMode], width -180, height -60);
 //text ("Colour: " + colourLabels[colourMode], width -180, height -40);

//export - inside draw!
  if (recordSVG) {
    endRecord();
    recordSVG = false;
    println("SVG exported");
  }

  if (savePNG) {
    saveFrame("poster_" + fonts[fontIndex] + "_mode" + distortionMode + ".png");
    savePNG = false;
    println("PNG saved");
  }
}

//core distortion system
void drawDistortedType(RShape grp, float soundMultiplier, float offsetX, float offsetY) {

  RPoint[][] paths = grp.getPointsInPaths(); //outline only

  //distortion strength depending on mode (0-Low, 1-Med, 2-High)
  float distortionStrength;

  if (distortionMode == 0) distortionStrength = 10;
  else if (distortionMode == 1) distortionStrength = 25;
  else distortionStrength = 50;

  for (int i = 0; i < paths.length; i++) {

//start shape if filled mode - needs colour to fill
 if(renderMode == 1) {
  if (colourMode ==0) fill (26);
```



```
  else if (colourMode ==1) fill (44, 74, 110);
  else fill (228, 87, 46);

  noStroke();
  beginShape();
}

for (int j = 0; j < paths[i].length; j++) {

      float x = paths[i][j].x;
      float y = paths[i][j].y;

      //image sampling
      int sampleX = int(map(x + offsetX, 0, width, 0, img.width - 1));
      int sampleY = int(map(y + offsetY, 0, height, 0, img.height - 1));

      sampleX = constrain(sampleX, 0, img.width - 1);
      sampleY = constrain(sampleY, 0, img.height - 1);

      float b = brightness(img.get(sampleX, sampleY));

      //distortion
      float imgOffset = map(b, 0, 255, 0, distortionStrength);
      float offset = imgOffset * soundMultiplier;

      float newX, newY;

      if (distortionMode < 2) {
        float angle = atan2(y, x);
        newX = x + cos(angle) * offset;
        newY = y + sin(angle) * offset;
      } else {
        float angle = noise(x * 0.01, y * 0.01, frameCount * 0.01) * TWO_PI;
        newX = x + cos(angle) * offset;
        newY = y + sin(angle) * offset * 1.5;
      }
      if (renderMode ==0) {
      drawGlyph(newX, newY, offset, soundMultiplier, b);
    }
    else {
      vertex(newX, newY);
    }
  }

  //close filled shape
  if (renderMode ==1) {
    endShape(CLOSE);
  }
 }
}

//glyph language/system
```

```
void drawGlyph(float x, float y, float offset, float sound, float brightness) {

  //density
  if (random(1) > map(brightness, 0, 255, 0.9, 0.35)) return;

  pushMatrix();
  translate(x, y);

//colour system
  if (colourMode == 0) fill(26);   //soft black
  else if (colourMode == 1) fill(44, 74, 110);   //blue
  else fill(228, 87, 46);   //orange

  noStroke();

  float size = offset * 1.2 + 2;

//shape selection
  if (brightness < 85) ellipse(0, 0, size, size);
  else if (brightness < 170) {
    rotate(sound);
    rectMode(CENTER);
    rect(0, 0, size * 1.2, 1.5);
  } else {
    rotate(sound);
    rectMode(CENTER);
    rect(0, 0, size, size);
  }
  popMatrix();
}

//controls
void keyReleased() {
 // saveFrame("frames/frame-####.png");  //save png sequence to folder

//export
  if (key == 's' || key == 'S') recordSVG = true;  //key S - export SVG
  if (key == 'p' || key == 'P') savePNG = true;    //key P - export PNG

//change font
  if (key == 'f' || key == 'F') {                  //key F - change font
    fontIndex = (fontIndex + 1) % fonts.length;
    font = new RFont(fonts[fontIndex], 200, RFont.LEFT);
    println("Font: " + fonts[fontIndex]);
  }

//change distortion mode
  if (key == 'd' || key == 'D') {                  //key D - change distortion
    distortionMode = (distortionMode + 1) % 3;
    println("Distortion mode: " + distortionMode);
  }
```



```
//change colour
  if (key == 'c' || key == 'C') {              //key D - change colour
    colourMode = (colourMode + 1) % 3;
    println("Colour mode: " + colourMode);
  }

//change render mode - filled vs. outline      //key R - change render - filled vs. outline
 if (key == 'r' || key == 'R') {
   renderMode = (renderMode + 1) % 2;
 }

//change location
 if (key == 'l' || key == 'L') {
  locationIndex = (locationIndex + 1) % locationNames.length;
  dataIndex = 0;
  loadData();
}

//change data set
 if (key == 'n' || key == 'N') {
  dataIndex = (dataIndex +1) % images[locationIndex].length;
  loadData();
  }
}
```

14

Bibliography

Books

Acrylicode (2023) *AudioReactiveSetUp + AudioReactiveRotation (TouchDesigner)*, Gumroad, [Design file], Available at: https://gumroad.com/d/ea8ef9d7b1e376207c39328f457ad984 (online) Accessed: 10.10.2025

Cinelli, M. (2023) *Giving Type Meaning*, 1st edn. Bloomsbury Visual Arts, Available at: https://www.perlego.com/book/4277538 (online) Accessed: 22.10.2025

Cheng, K. (2026) *Designing Type*, second edition, London, Laurence King Student & Professional

Conditional Design (nd.) *Conditional Design Workbook*, Conditional Design, Available at: https://workbook.conditionaldesign.org (online) Accessed: 06.11.2025

Cottier, N. (2025) *Alphabetical Playground*, 1st ed., Karlsruhe (Germany), Slanted Publishers

Counter-Print Books (2024) *Expressive Type Today*, London, Counter Print

Currie, M. (2010) *Postmodern Narrative Theory*, 2nd edn., Bloomsbury Academic, Available at: https://www.perlego.com/book/2997604 (online) Accessed: 28.02.2026

D&AD (2018) *The Copy Book*, London, Taschen

Derivative (2025) *Books*, Derivative, Available at: https://derivative.ca/UserGuide/Books (online) Accessed: 06.11.2025

Elam, K. (2019) *Geometry of Design*, Studies in Proportion and Composition, edition 1, Krakow, d2d.pl

Evans, J. Hall, S. (1999) *Visual Culture: The Reader*, London, Sage

Generative Gestaltung (2025) *Generative Design*, Generative Gestaltung, Available at: http://www.generative-gestaltung.de/2/ (online) Accessed: 09.11.2025

Experimental Jetset (2015) *Automatically Arranged Alphabets*, Roma Publications

Gerstner, K. (2007) *Designing Programmes: Instead of Solutions for Problems Programmes for Solutions*, Lars Muller Publishers

Gross, B. et al. (2018) *Generative Design*, [edition unavailable], Princeton Architectural Press, Available at: https://www.perlego.com/book/1100223 (online) Accessed: 26.12.2025

Houston, K. (2013) *Shady Characters*, London, Penguin Books

Hubner, P. (2025) *The Generative Mind. A New Approach to Creative Thinking*, Switzerland, Braun Publishing AG

Interactive Immersive HQ (2025) *An Introduction to Touchdesigner 099*, Interactive Immersive HQ, Available at: https://interactiveimmersivehq.github.io/touchdesigner-book/#book (online) Accessed: 06.11.2025

Lupton, E. (2014) *Thinking with Type*, Princeton Architectural Press, Available at: https://www.perlego.com/book/1099923 (online) Accessed: 28.02.2026

Lupton, E. (2014) *Type on Screen*, Princeton Architectural Press, Available at: https://www.perlego.com/book/1099517 (online) Accessed: 28.02.2026

Lupton, E. (2017) *Design is Storytelling*, London and New York, Cooper Hewitt

L., Reas, C., Fry, B. (2016) *Make: Getting Started with p5.js*, first ed., San Francisco, Maker Media

McNeil, P. and Muir, H. (2024) *System Process Form. Type as Algorithm*, London, Thames & Hudson

Moller Hansen, S. (2025) *Learning Creative Coding*, Stig Moller Hansen, Available at: https://stigmollerhansen.dk/resume/learning-creative-coding/ (online) Accessed: 23.12.2025

Pearson, M. (2011) *Generative Art*, [edition unavailable], Manning, Available at: https://www.perlego.com/book/2682581 (online) Accessed: 09.12.2025

Reas, C. and Fry, B. (2015) *Make: Getting Started with Processing*, second ed., San Francisco, Maker Media

Shiffman, D. (2008) *Learning Processing. A Beginner's Guide to Programming Images*, Animation, and Interaction, Burlington, Morgan Kaufmann

Tools and Resources

Acrylicode (2023) *AudioReactiveSetUp + AudioReactiveRotation (TouchDesigner)*, Gumroad, [Design file], Available at: https://gumroad.com/d/ea8ef9d7b1e376207c39328f457ad984 (online) Accessed: 10.10.2025

Derivative (2025) *Touch Designer*, Derivative, [software], Available at: https://derivative.ca (online) Accessed: 10.09.2025

Generative Gestaltung (2025) *Download Code Package*, Generative Gestaltung, [Code files], Available at: http://www.generative-gestaltung.de/2/ (online) Accessed: 09.11.2025

Generative Gestaltung (2025) *Generative Design*, Generative Gestaltung, [code package], Available at: http://www.generative-gestaltung.de/2/ (online) Accessed: 05.11.2025

Gross, B. et al. (2018) *Generative Design*, [edition unavailable], Princeton Architectural Press. Available at: https://www.perlego.com/book/1100223 (online) Accessed: 05.11.2025

Interactive Immersive HQ (2025) *Project Files*, Interactive Immersive HQ, [project files], Available at: https://interactiveimmersivehq.github.io/touchdesigner-book/#book (online) Accessed: 09.11.2025

Processing (2025) *Processing 4.4.10 for macOS*, Processing, [software], Available at: https://processing.org/download (online) Accessed: 23.10.2025

Shiffman, D. (2025) *Downloads*, Daniel Shiffman, Learning Processing, [book code exercises], Available at: http://learningprocessing.com/downloads/ (online) Accessed: 08.11.2025

Sorkhabi, E. (2019) *An Introduction To TouchDesigner 099, Derivative*, [open-source book], Available at: https://derivative.ca/UserGuide/Books (online) Accessed: 05.11.2025

Van Der Sanden, M. (2025) *What Melodies Can be Found in Typography?*, Mike Van Der Sanden, [code example], Available at: https://mikevandersanden.com (online) Accessed: 23.10.2025

John McCaffrey (2020) *How to Add Sound to a Processing Projec*t, YouTube, Available at: https://youtu.be/l6fG1wneXWo?si=u33pkrD8RV6PThXC (online) Accessed: 23.10.2026

The Coding Train (2015) *10.1: Intro to Images - Processing Tutorial*, YouTube, Available at: https://youtu.be/-f0WEitGmiw?si=KoGhGQWU2vJoe0o_ (online) Accessed: 23.10.2026

The Coding Train (2015) *7.3: Modularity with Functions - Processing Tutorial*, YouTube, Available at: https://youtu.be/j_XyeWg_3EE?si=FF8Xz4L4OdS_8Qi9 (online) Accessed: 23.10.2025

The Coding Train (2023) *Creative Coding for Beginners - Full Course!*, YouTube, Available at: https://youtu.be/4JzDttgdILQ?si=1bPc2cagkwfY_Wxj (online) Accessed: 28.10.2025

P5.js (2025) *p5.js*, p5.js start coding, [code editor], Available at: https://p5js.org (online) Accessed: 21.11.2025

Gross, B. et al. (2018) *Generative Design*, [Code Package], Generative Gestaltung, Available at: http://generative-gestaltung.de (online) Accessed: 09.11.2025

Interactive Immersive HQ (2025) *Project Files*, Interactive Immersive HQ, [project files], Available at: https://interactiveimmersivehq.github.io/touchdesigner-book/#book (online) Accessed: 09.11.2025

Processing (2025) *Examples*, Processing, [code files], Available at: https://processing.org/examples (online) Accessed: 23.10.2025

Shiffman, D. (2025) *Downloads*, Daniel Shiffman, Learning Processing, [book code exercises], Available at: http://learningprocessing.com/downloads/ (online) Accessed: 08.11.2025

Magazines

Slanted Publishers (2023) *Slanted Magazine #40—Experimental Type 2.0*, Issuu, [magazine online], Available at: https://issuu.com/slanted/docs/slanted_experimentsl_issuuu (online) Accessed: 16.10.2025

Slanted (2025) *Experimental Type 3.0*, Slanted

Fukt Magazine For Contemporary Drawing (2025) *Fukt Sound*, Fukt Magazine, Issue 23

TypeOne (2023) *Issue 07*, TypeOne Ltd, UK

Podcasts

Creative Characters (2023) *Roger Black and Charles Nix Trace the History of Type*, Apple Podcasts, Available at: https://podcasts.apple.com/gb/podcast/creative-characters/id1554900091?i=1000635608916 (online) Accessed: 21.10.2025

Design Practice (2025) *068: Rola Typographii w Projektowaniu – Kulisy Pracy Projektanta I Typografa | Mateusz Michalski*, Design Practice, Available at: http://designpractice.pl/068 (online) Accessed: 07.05.2026

Design Practice (2025) *081: Jak Pracuje Typografka? | Ania Wielunska*, Design Practice, Available at: http://designpractice.pl/081 (online) Accessed: 19.04.2025

Design Practice (2026) *093: O Szukaniu Wlasnego Stylu w Projektowaniu | Ada Zielinska*, Design Practice, Available at: http://designpractice.pl/093 (online) Accessed: 03.05.2026

The Angry Designer (2025) *The Real Reason Graphic Design is Undervalued by Everyone*, Apple Podcasts, Available at: https://podcasts.apple.com/gb/podcast/the-angry-designer-graphic-design-freelancing/id1531777489?i=1000732753306 (online) Accessed: 30.10.2025

The Creative Boom Podcast (2025) *Brian Collins on Fighting Second Dragons, the Power of Orange Ties and the Future of Creativity*, Apple Podcasts, Available at: https://podcasts.apple.com/gb/podcast/the-creative-boom-podcast/id1497680408?i=1000733609532 (online) Accessed: 29.10.2025

Events and Exhibitions

Boyd, J. (2025) *Strange Fascinations*, London, The Crypt Gallery, 11-16 October 2025 [exhibition] Attended: 16.10.2025

Experimental Jetset (2026) *Circuits*, Stedelijk Amsterdam [exhibition] Attended: 28.01.2026

TypoCircle (2025) *TypoCircle: Supercondensed*, Typographic Circle [event] Attended: 22.10.2025

Typocrcle (2025) *TypoCircle Presents: Harriet Richardson – What Did You Just Say to Me?*, Typocircle [event] Attended: 30.10.2025)

THANK YOU

Thanks to my lecturers for their guidance and support throughout this project, and to my partner for their encouragement.

Process Data Place explores a generative approach to typography shaped by image and sound. Using site-specific recordings from Surrey, the project transforms letterforms into responsive systems, where environmental data drives form, variation, and behaviour.

Through a rule-based framework, typography shifts from fixed glyphs to adaptive structures, proposing a model in which design emerges through the definition of conditions from which form is generated.