Contextualizing Programming with Algorithmic Art Practices Using Computational Thinking Principles for Undergraduate Design Students

Özyeğin University Design, Technology and Society Department
Alp Tuğan

15.05.2025

START →

Introduction

•

Rapid advancements in technology necessitate programming skills and computational thinking (CT) for everyone.

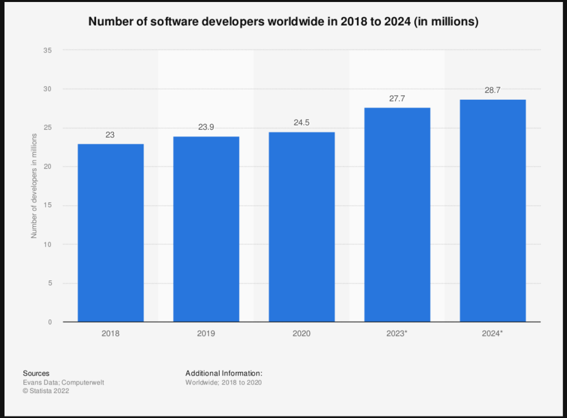
Growing programmer population

Paradigmatic Shift on Human-computer Interaction

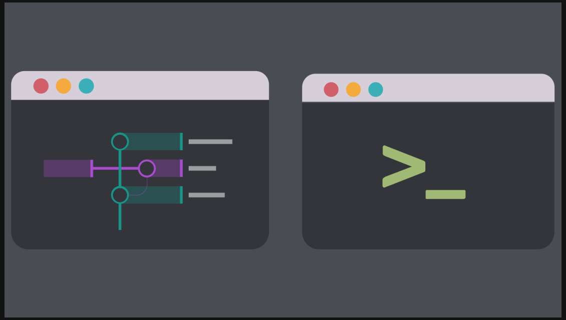
GUI to Command-based Interface (CBI)

Cases of The Paradigmatic Shift

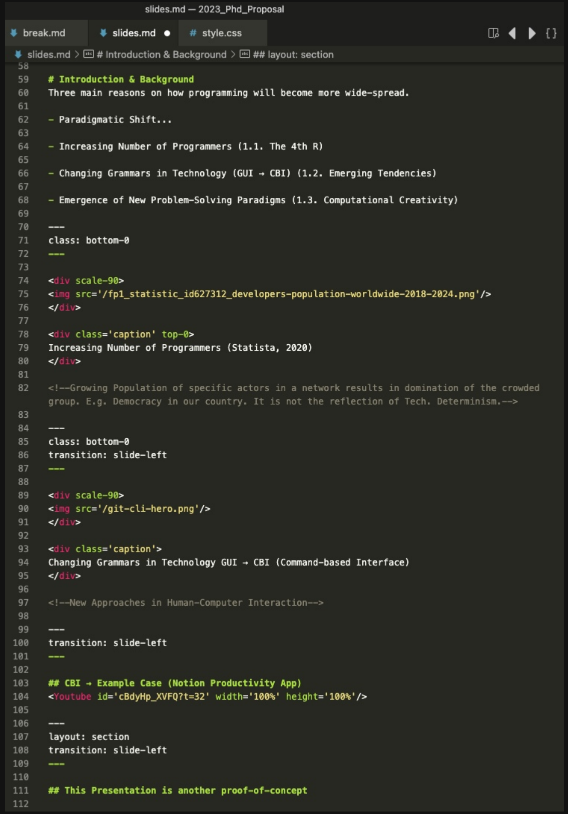
Increasing Number of Programmers (Statista, 2020)



GUI → CBI (Command-based Interface)



This Presentation is another proof-of-concept



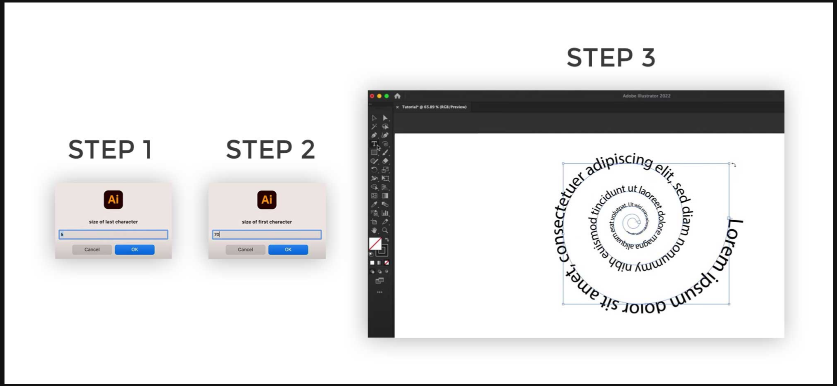
Existing Software Follow The Trend

`CBI AI Tools`

Eg. Mid Journey, DALL-E

`Scripting`

Eg. Adobe Illustrator



Problems

- Traditional programming courses present challenges for design students:
  - Abstract concepts: Programming fundamentals often feel disconnected from real-world applications.
  - Lack of relevance: Traditional approaches struggle to resonate with non-context preferences, leading to disengagement.
- Students may hold prejudice against coding, perceiving it as tedious or only relevant to technical fields.
- Translating natural language understanding into formal programming languages is a key difficulty for novices.
- High dropout rates: Programming courses in higher education face a rising trend of dropouts and failures.

Research Gap & Goal

Literature supports context-based learning for engagement but lacks specific, actionable methodologies tailored for visual learners.

- There is a need for elaborate approaches on programming education.
- Lack of computational thinking tools for learning.
- The scope and methods used when suggesting contextualizing programming with "art" remain unclear.
- Contextualizing programming with art has positive impacts. But how?

Research Goal: Propose practical methods and contribute to debates on developing CT skills through contextualized programming paradigms by analyzing algorithmic art.

Develop and evaluate a methodology using algorithmic art to facilitate programming education that is both educationally rigorous and creatively relevant to design students

Research Question

How can we contextualize programming fundamentals through algorithmic art practices to improve students’ computational thinking skills and engagement in higher education?

SRQ1: *"What are the common practices used in algorithmic works of art related to programming fundamentals, especially in creative coding?"*

SRQ2: *"How can we relate common practices used in algorithmic art with computational thinking to provide hands on tools that can be used as teaching and learning material in a visual context?"*

Theoretical Foundations

Constructionist Learning + ANT + Semiotic Representations

Seymour Papert's Constructionist Theory: Students learn best when actively creating personally meaningful artifacts. This underpins the hands-on, creative coding approach.

Actor-Network Theory (ANT): Useful for understanding complex relationships between human and non-human actors.

Duval's Theory of Semiotic Representation: Addresses the cognitive challenges of converting representations between different systems (visual, verbal, code). Non-congruent conversions and cognitive dissociation are key challenges the method aims to address.

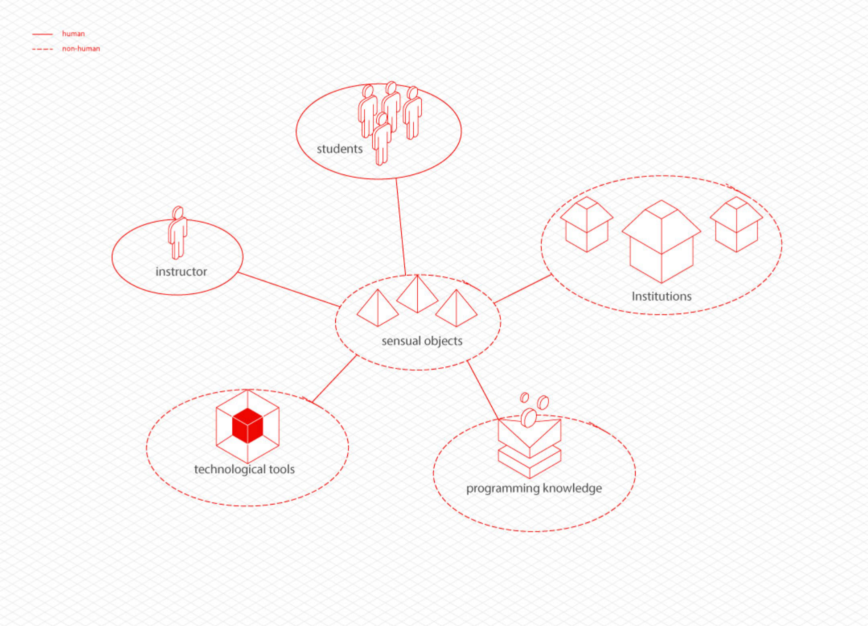
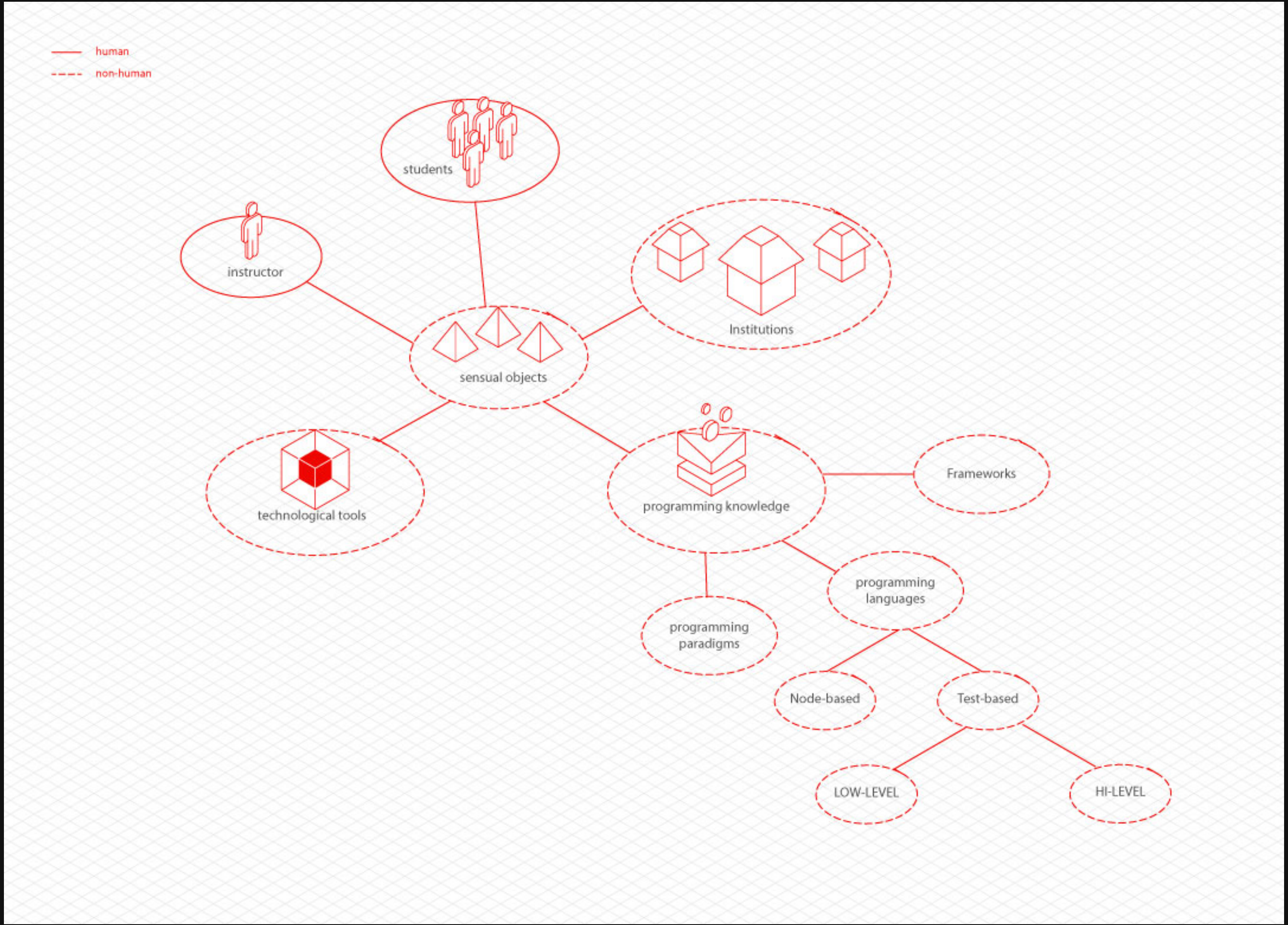


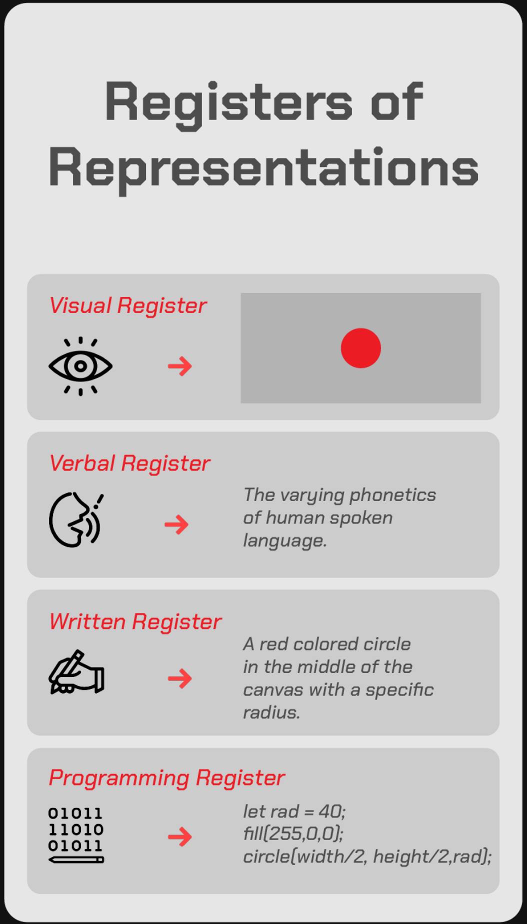
Figure: Research Network Actors


human
non-human
students
instructor
Institutions
sensual objects
technological tools
programming knowledge
Frameworks
programming languages
programming paradigms
Node-based
Test-based
LOW-LEVEL
HI-LEVEL

What is register?

A register of representation refers to a specific semiotic system used to express mathematical concepts, such as natural language, symbolic notation, graphical representations, or visual displays. (Duval, 2006)

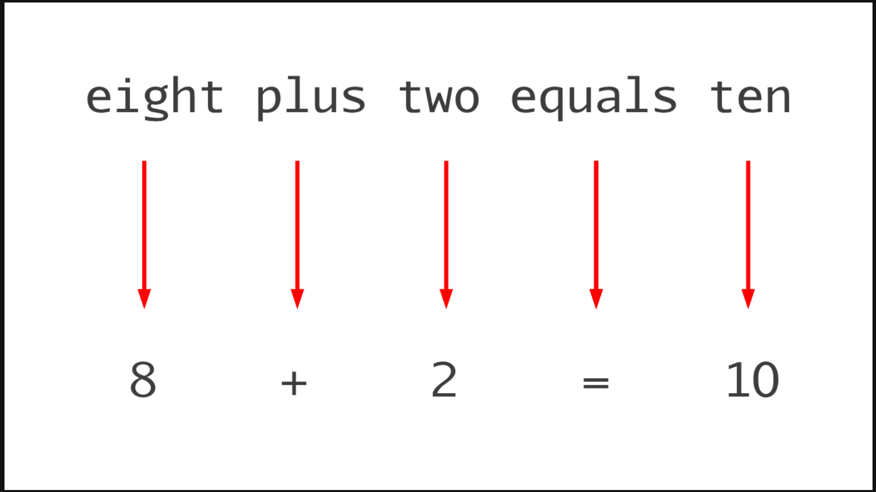
1. Visual Register
2. Verbal Register
3. Written Register
4. Programming register



Register Conversion

Congruent and Non-congruent registers

Congruent Register Conversion



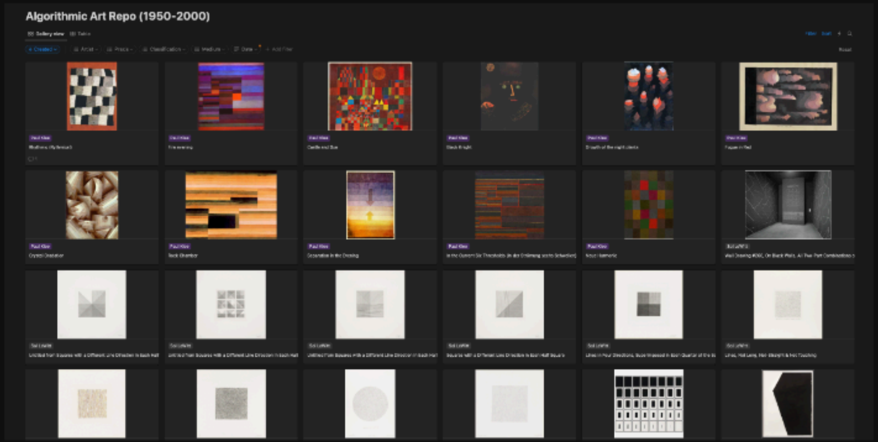
Non-congruent Register Conversion

`Adding eight and half of four gives the sum of ten`, the word "adding" comes before the numbers, two is derived from the half of four and the conversion becomes non-congruent.

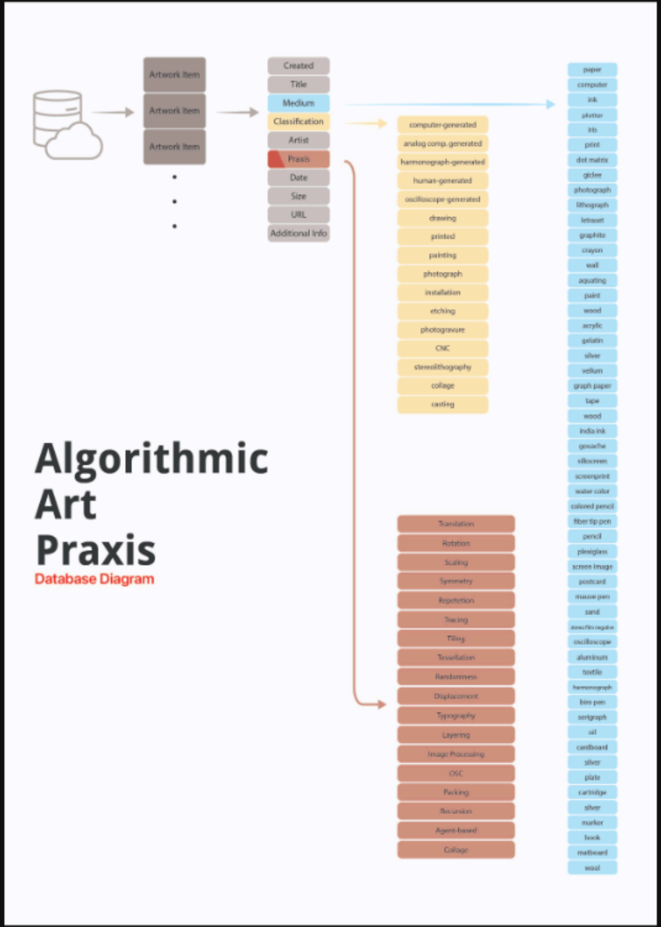
Methodology

- Phase 1: ALAP Database, Categories and Cheat Sheet
- Bridge: Theory of Semiotic Registers
- Phase 2: De-scription/In-scription Method

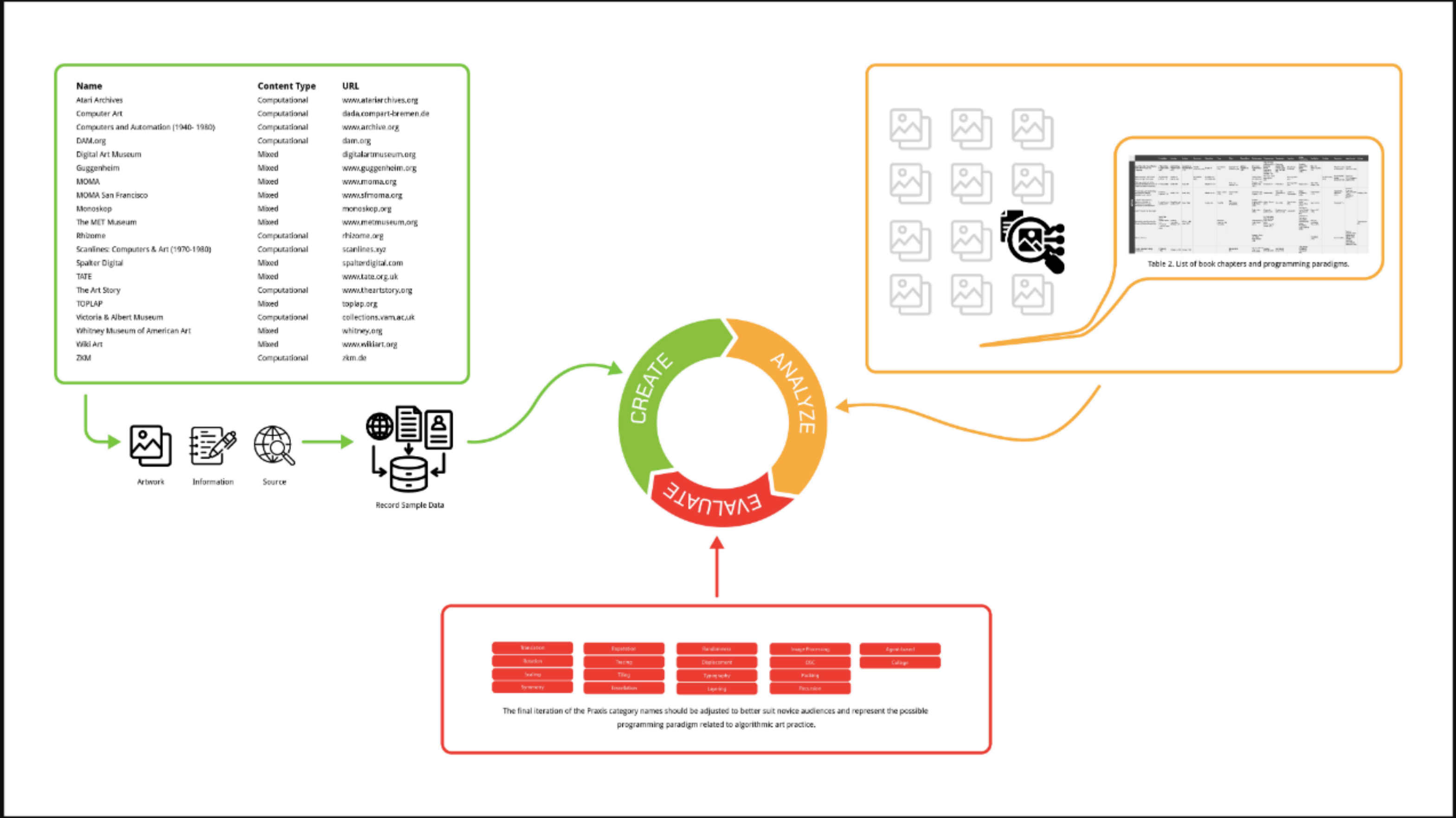
Phase1: Algorithmic Art Praxis



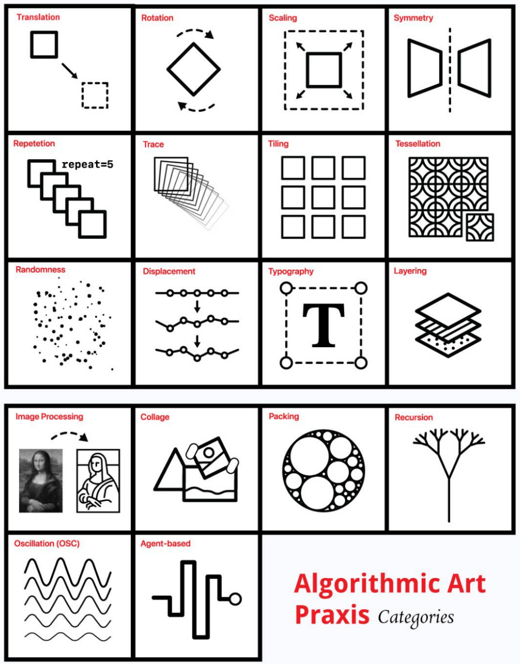
Link to ALAP Database



Categorization Rationale



Outcomes: ALAP Categories



Algorithmic Art Praxis

Cheat Sheet

ALAP Category	Icon	Description	Sample Artworks
Translate		Represents positional changes of the elements relative to each other or the canvas.	
Rotate		Represents orientational changes of the elements relative to each other or the canvas.	
Scale		Represents dimensional changes of the elements relative to each other or the canvas.	
Symmetry		Represents mirrored elements in vertical, horizontal, or custom axis relative to each other or to a point on the canvas.	
Repetition		Represents occurances of a single or group of elements with or without formalistic modifications on the canvas.	
Traces		Represents occurances of graphical elements (lines, curves) along with or without a continous path. The opacity of the repeating pattern may vary on the canvas.	
Tiles		Represents a grid-based distribution of elements on the canvas. Individual graphical objects in the grid do not have to be continuous, mixed, or same with each other.	
Tessellation		Represents a continous distribution of the elements on the canvas. Each tile must have a relational and formal connection in order to create unique patterns. In short, every Tessellation involves Tiling, but not every Tiling can be considered a Tessellation.	
Randomness		Represents the graphical elements as if they were randomly positioned, scaled, or colored on the canvas. The random behavior may be observed as displacement of points on lines or curves.	
Displacement		Represents the positional change of the contour points of graphical elements on the canvas. For example, a straight line consisting of four points can be transformed into a zig-zag shape by moving the points in different directions.	
Typography		Represents the use of typographic elements on the canvas.	
Layers		Represents stacked or redrawn graphical elements on to each other using different colors.	
Image Processing		Represents the recreation of a preloaded image in different styles on the canvas.	
Oscillation (OSC)		Represents occurrence of sinusoidal abstract forms and wave-like shapes on the canvas.	
Packing		Represents fitting the objects into a limited space (a.k.a space-filling or packing algorithm). The rule is that objects must not interfere with each other.	
Recursion		Represents a process that repeats itself to create patterns and shapes, which can continue indefinitely or become highly intricate. For example, a tree starts with a trunk, then splits into two main branches. Each branch further splits into two smaller branches, and so on...	
Agent-based		Represents the creation of a graphical composition showcases continous formalistic features. For example, drawing a sketch without holding up the pencil.	
Collage		Represents the traditional style collage in art. Images can be cropped manually and then transferred to the computer, and using programming practices, they can be positioned on the canvas.	

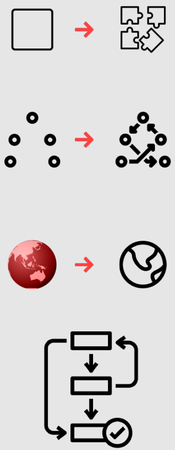
Bridging Phase1 and Phase 2

Semiotic Registers

The Methodological Framework

Computational Thinking

Computational Thinking



Decomposition

Pattern Recognition

Abstraction

Algorithm Design

ALAP Categories

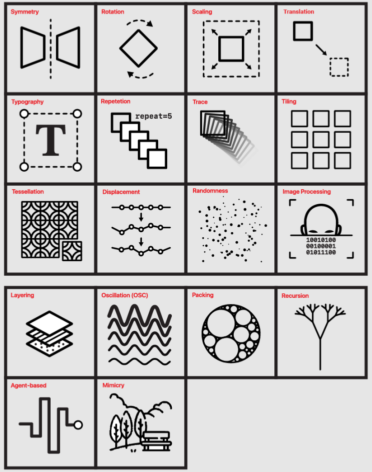
Algorithmic Art Praxis

Categories identified in the previous research;

Symmetry
Rotation
Scaling
Trace
Layering
Tiling
Tessellation
Image Processing
Collage
Typography
Translation
Displacement
Repetition
Recursion
Packing
Randomness
Agent-based
Oscillation (OSC)

Multiple Registers

Semiotic Representations



Register Conversion

Registers of Representations

Visual Register

Verbal Register

The varying phonetics of human spoken language.

Written Register

A red colored circle in the middle of the canvas with a specific radius.

Programming Register

let rad = 40;
fill(255,0,0);
circle(width/2, height/2,rad);

Phase 2: De-scription / In-scription Method

1. Choose an image from the database
2. Analyze it using pen and pencil or any other tool like drawing tablets familiar to the learner.
3. Determine the instruction order.
4. Register Conversion Stage.
   - Use the cheat-sheet.
   - Research using the cheat-sheets (web-sites, previous assignments, ALAP codes).
5. Algorithm Design

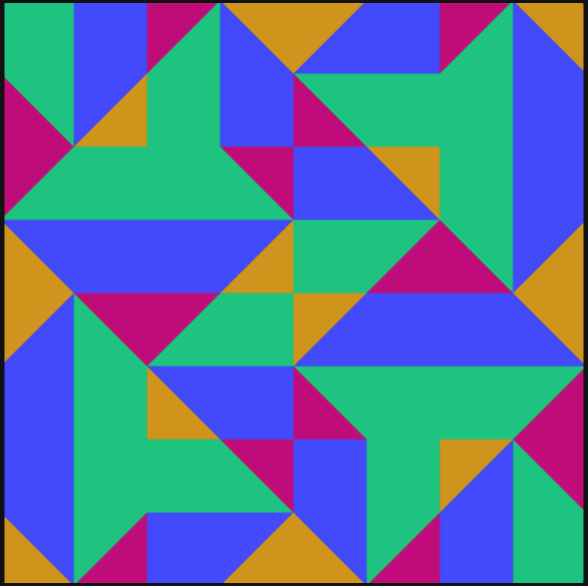
De-scription / In-scription Method

Computational Thinking Framework

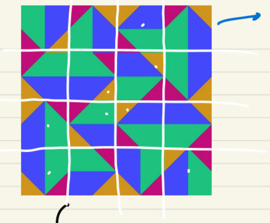
	Step	Name	Registers
De-scription	1	CHOOSE	Visual
	2	ANALYSIS	Visual, Verbal, Written
	3	PROCEDURAL FLOW	Visual, Verbal, Written
In-scription	4	REGISTER CONVERSION	Programming ⇄ Visual, Verbal, Written
	5	ALGORITHM DESIGN	Programming

D/I Method Applied

1. Choose



2. Analysis

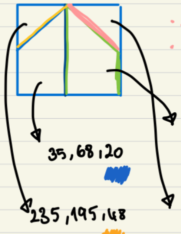


the image can be decomposed in the grid

8 different squares.

worked grid by grid, each rect carries, triangles

there are 8 squares and are all the same



· 2 rect inside the grid (square)
· Create two triangles on top

36, 222, 143

35, 68, 120

239, 23, 121

235, 195, 48

3. Procedural Flow

Flow

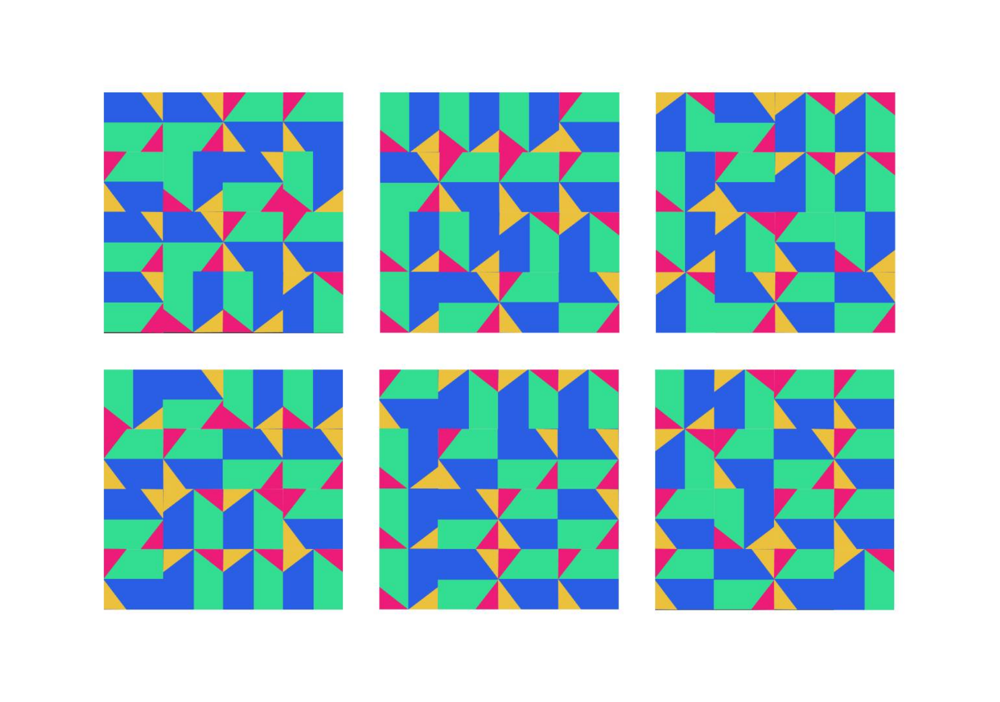
1) draw the grid
2) draw the shape inside one square
3) rotate it

① draw two rect
② draw triangles on top of each grid as it is in the image
③ Fill the color
④ use rotate, translate

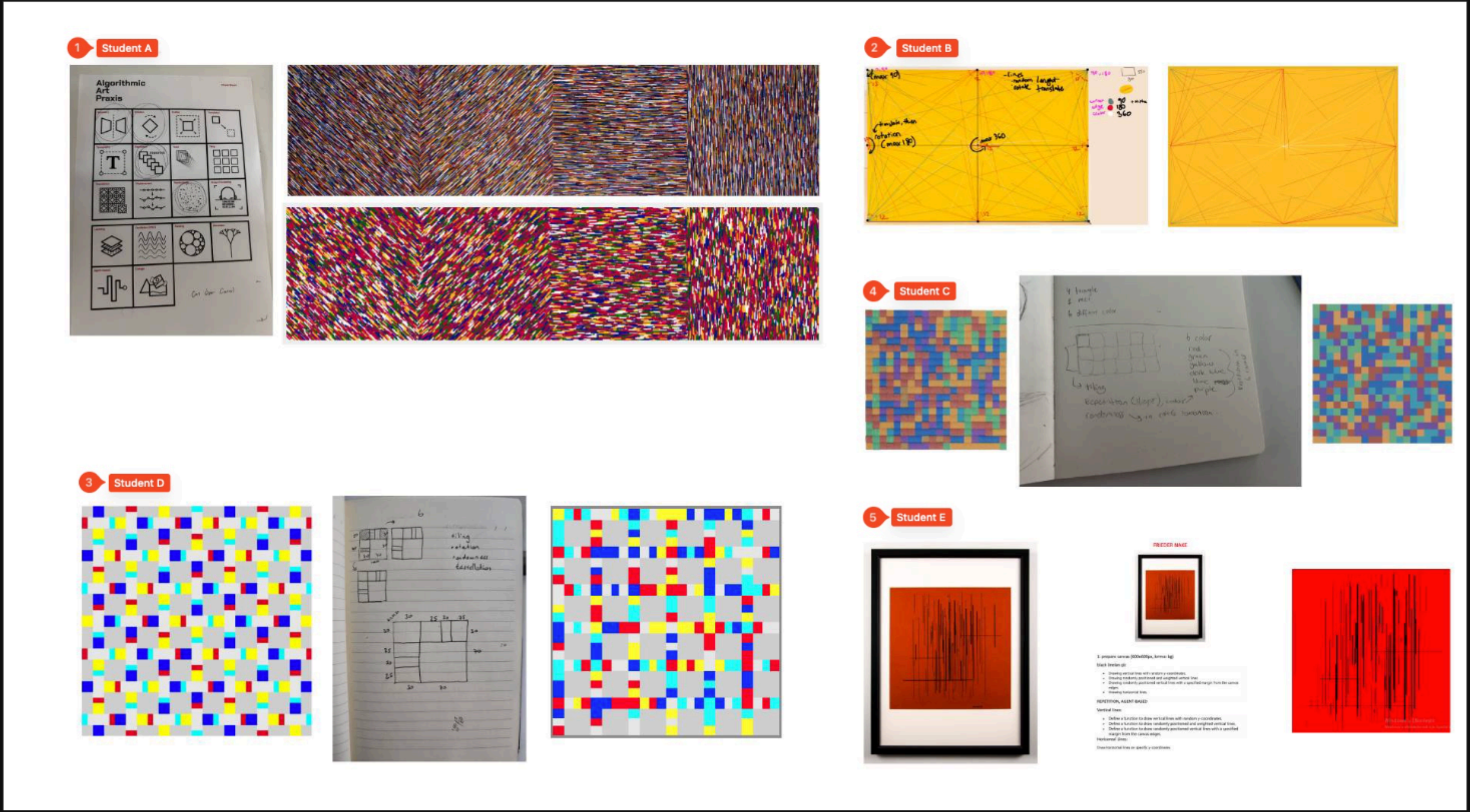
4. Register Conversion

Written Register	Programming Language Register
Green color →	`let green; green = color(36, 343, 143);`
Triangle →	`triangle(x1,y1,x2,y2,x3,y3);`
Square →	`rect(xr1, yr1, sr1, sr1);`
Create canvas 800 by 800 px →	`createCanvas(800, 800)`
…	`...`

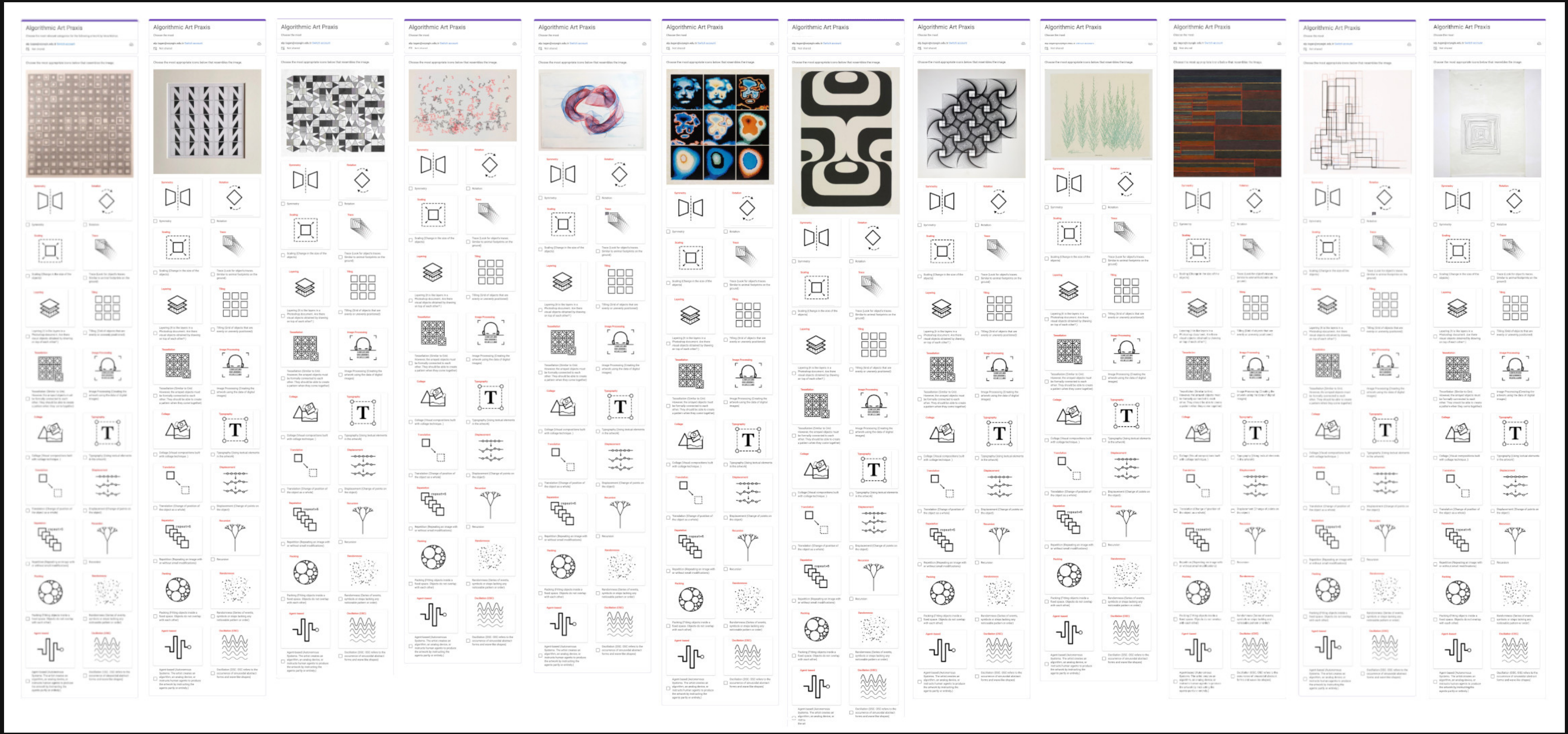
5. Algorithm Design



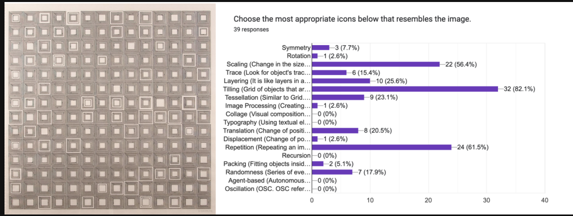
Other Students

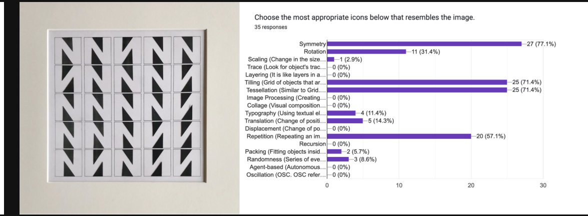


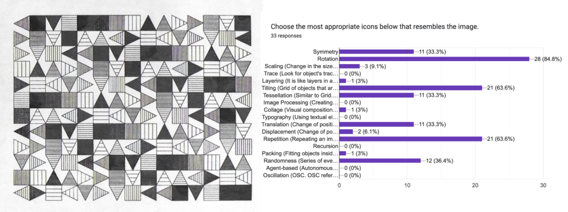
Survey 1 - ALAP Categories

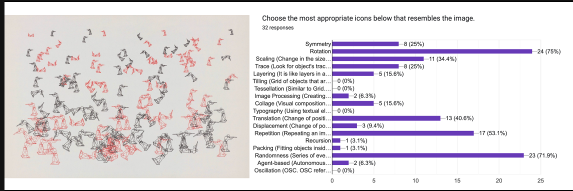


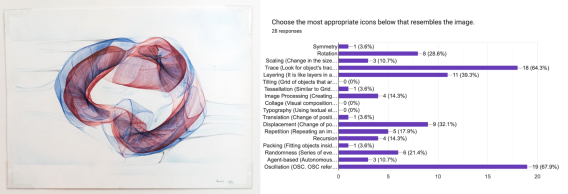
Results (Survey 1)

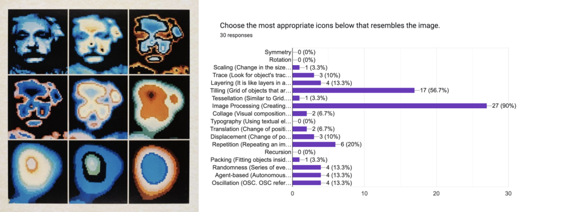


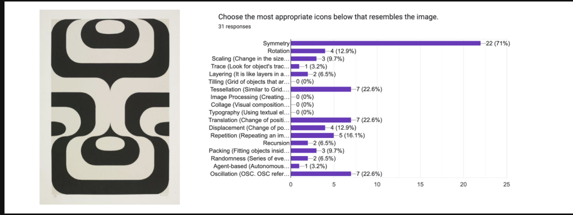


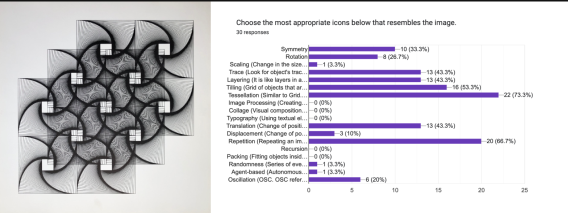


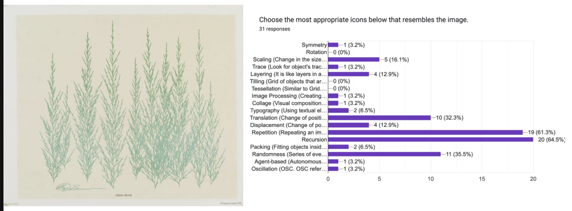


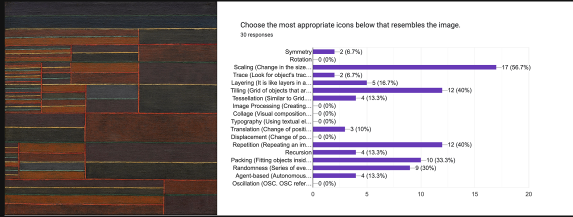


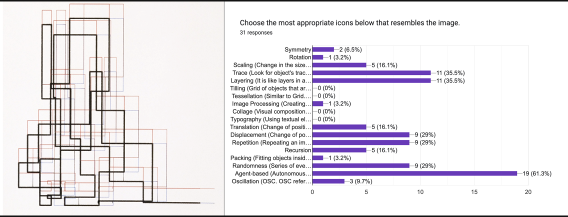


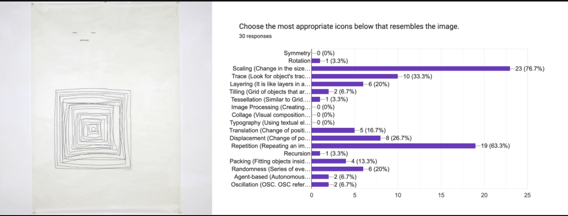




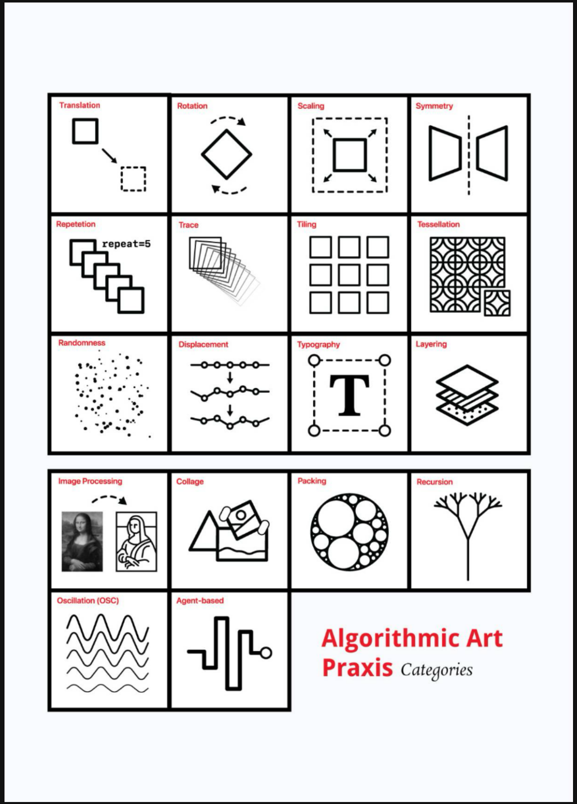








Conclusion: ALAP Survey



Results show that;

- Participants comprehend categories.
- Some of the selected artworks take time to identify. The artworks, including abstract and natural forms (e.g., Desmond Paul Henry’s), become more challenging than those generated with simple shapes, such as Vera Molnar’s geometric works.
- Increased engagement.
- Peer assessment.
- In-class discussions increased.
- Participants use cheat-sheets while asking questions.

Results (Survey De-scription/In-scription)

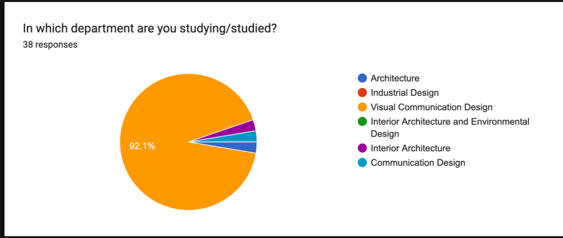
De-scription / In-scription Method

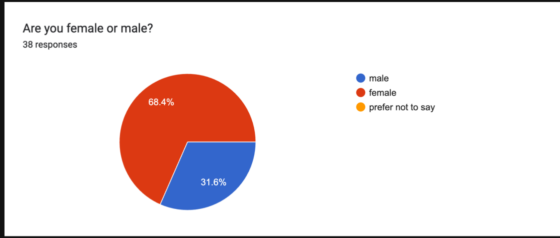
Computational Thinking Framework

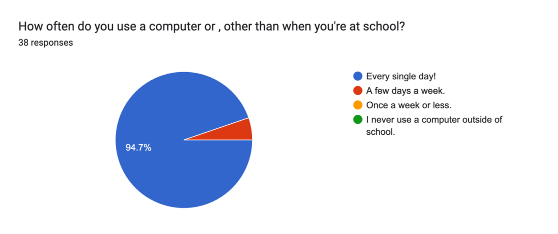
	Step	Name	Registers
De-scription	1	CHOOSE	Visual
	2	ANALYSIS	Visual, Verbal, Written
	3	PROCEDURAL FLOW	Visual, Verbal, Written
In-scription	4	REGISTER CONVERSION	Programming ⇄ Visual, Verbal, Written
	5	ALGORITHM DESIGN	Programming

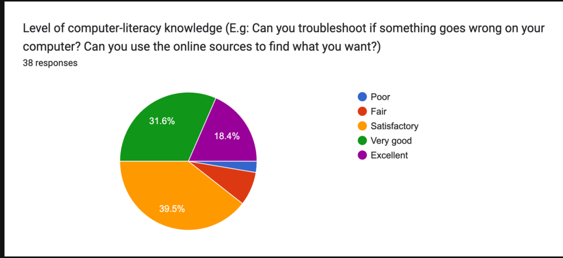
- The survey is divided into two main sections.
- The first section, related to participants' backgrounds, providing essential demographic information.
- The second section, focuses on the efficiency of the De-scription/In-scription method.

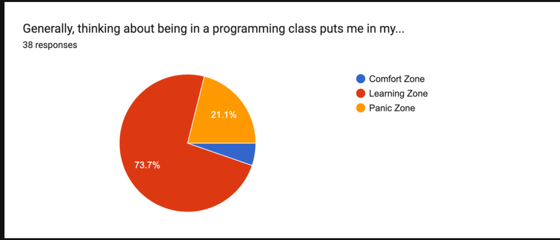
Participants’ Background

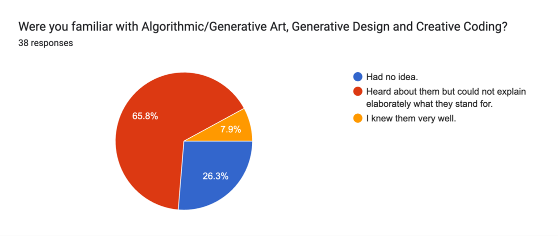


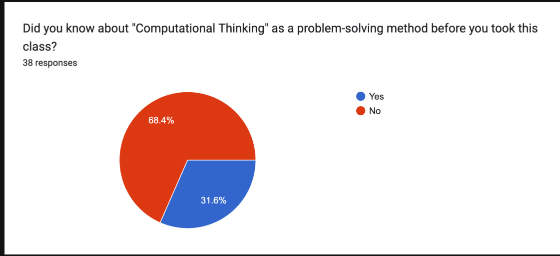




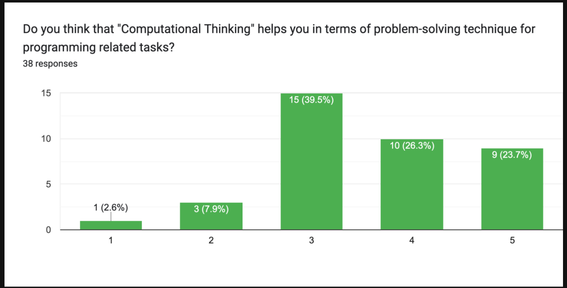


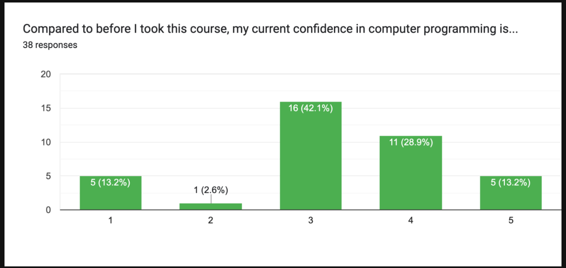


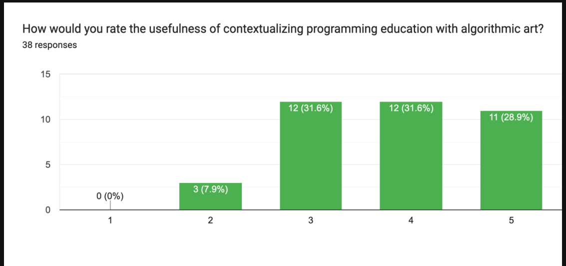


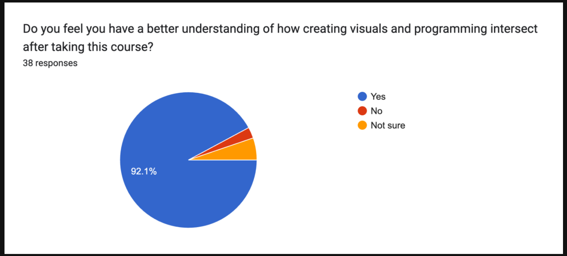


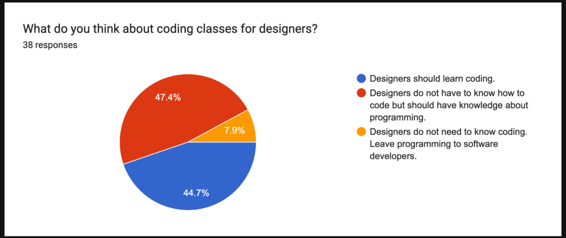
Is The Method Effective?











Contributions of the Research

·

The De-scription/In-scription Method: A validated experimental method specifically for teaching creative coding to visual learners in design.

The ALAP Framework and Resources: Algorithmic Art Praxis (ALAP) database: A unique online repository (1940-2000) providing historical context and examples.

ALAP Categories: 18 distinct categories linking algorithmic art practices to programming paradigms.

ALAP Cheat Sheet: A practical, visual tool aiding register conversion and CT activities.

Theoretical Contribution: Contributes to the discourse on CT in higher education, reframing it as essential for interdisciplinary creativity, not just technical fields. Empirically grounds Papert's constructionism and utilizes concepts from ANT and Duval's semiotics in a practical teaching method.

Empirical Analysis: Provides in-depth quantitative and qualitative data on design majors' perspectives towards programming

Limitations and Future Directions

Limitations

Restricted Accessibility of the ALAP Database: Currently only accessible to the author, hindering wider use and sustainability.

Study conducted with a specific group of design students at one university.

Workshop duration (four weeks) is relatively short for comprehensive programming proficiency.

Future Research Directions

Establish a dedicated, accessible online platform for the ALAP database, allowing collaboration and expansion (including contemporary works).

Integrate interactive elements allowing direct code experimentation within visual contexts.

Add more case studies within the ALAP framework.

Explore the application of the method in other non-technical disciplines.

Further clarify distinctions between similar ALAP categories (e.g., Tiling/Tessellation) and provide more references

Conclusion

Equipping students with opportunities to express themselves through visual aids like the ALAP categories can significantly enhance the learning experience, comprehending cognitive processes, and fostering self-assurance and willingness to articulate views.

- ALAP Database provide source material for contextualization.
- ALAP categories eases the process of register conversion.
- ALAP cheatsheet improves communication between the instructor and the learner (CT Tools).
- D/I Method provides an explicit, step-by-step, problem-solving approach.
- Improvement of student ↔ instructor communication.
- Increased self-confidence results in higher engagement.

Thank you.

The End