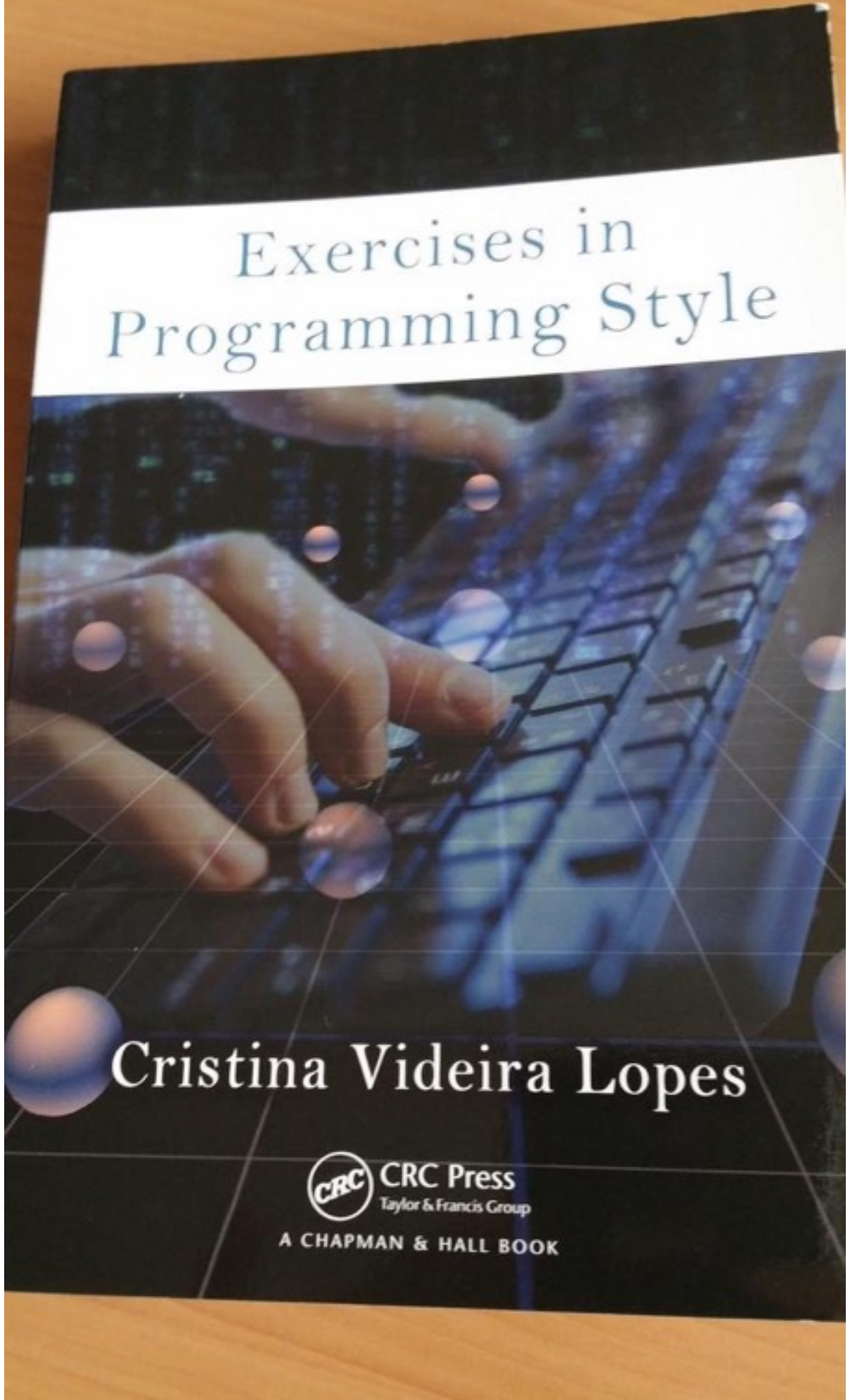


Programming Styles

Tijs van der Storm

Why think about styles?

- Acknowledge there’s more thar way to do it
- Recognize them
- Evaluate trade-offs in us
- Evaluate fitness for purpo
- Decouple from program language

The book isolates the concept of design itself!

Drivers

- Readability
- Changeability
- Robustness
- Correctness
- Performance
- …

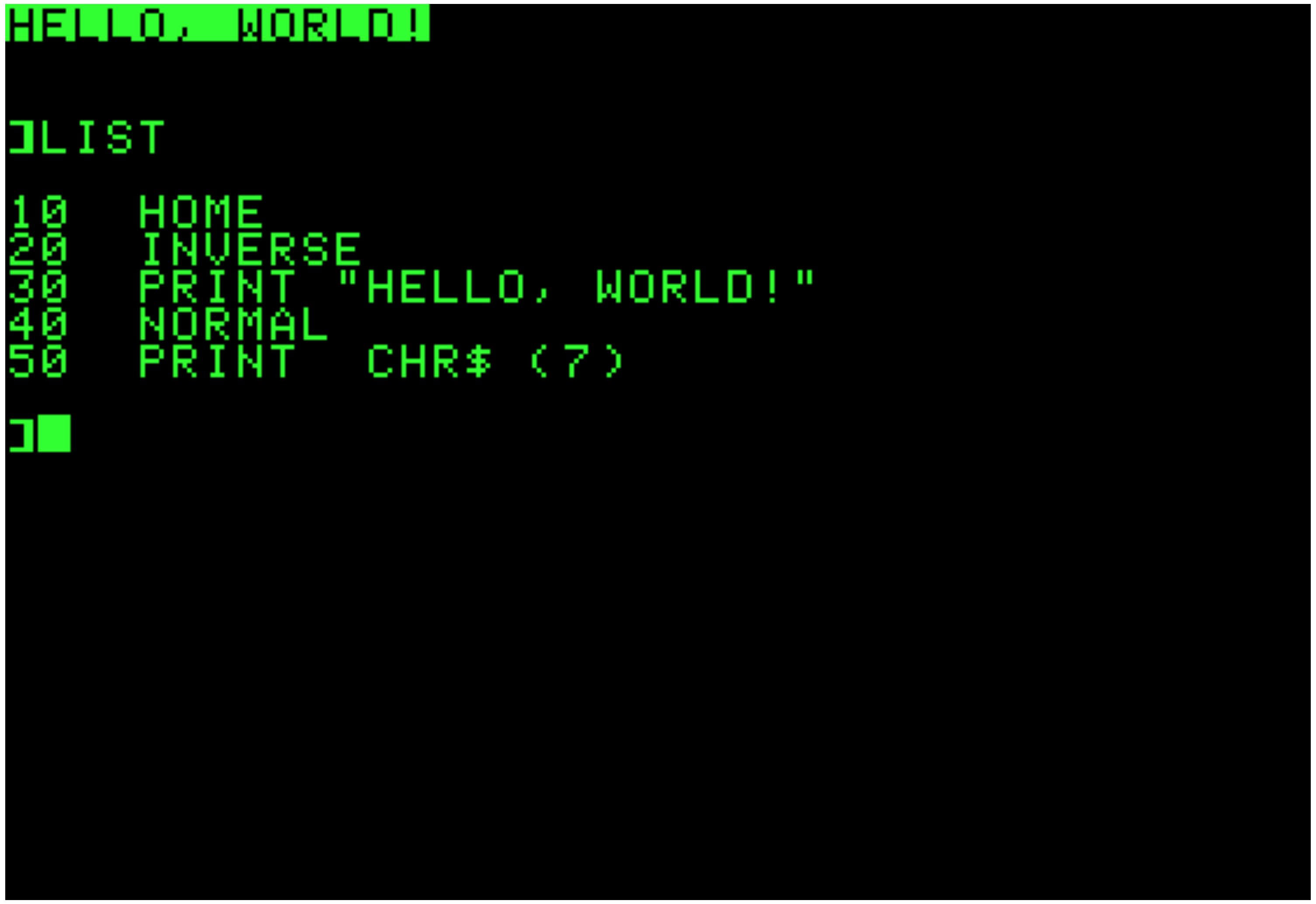
- Idiomatic
- Concision
- Reuse
- Extensibility
- Usability
- …

Today

- Monolith
- Cookbook
- Pipeline
- Things
- Abstract things
- Hollywood
- Constructivism
- Tantrum
- Passive aggressive
- Trinity

Basic styles

- Monolith
- Cookbook
- Pipeline



```
HELLO, WORLD!


]LIST

10  HOME
20  INVERSE
30  PRINT "HELLO, WORLD!"
40  NORMAL
50  PRINT  CHR$ (7)

]
```

Monolith

Monolith

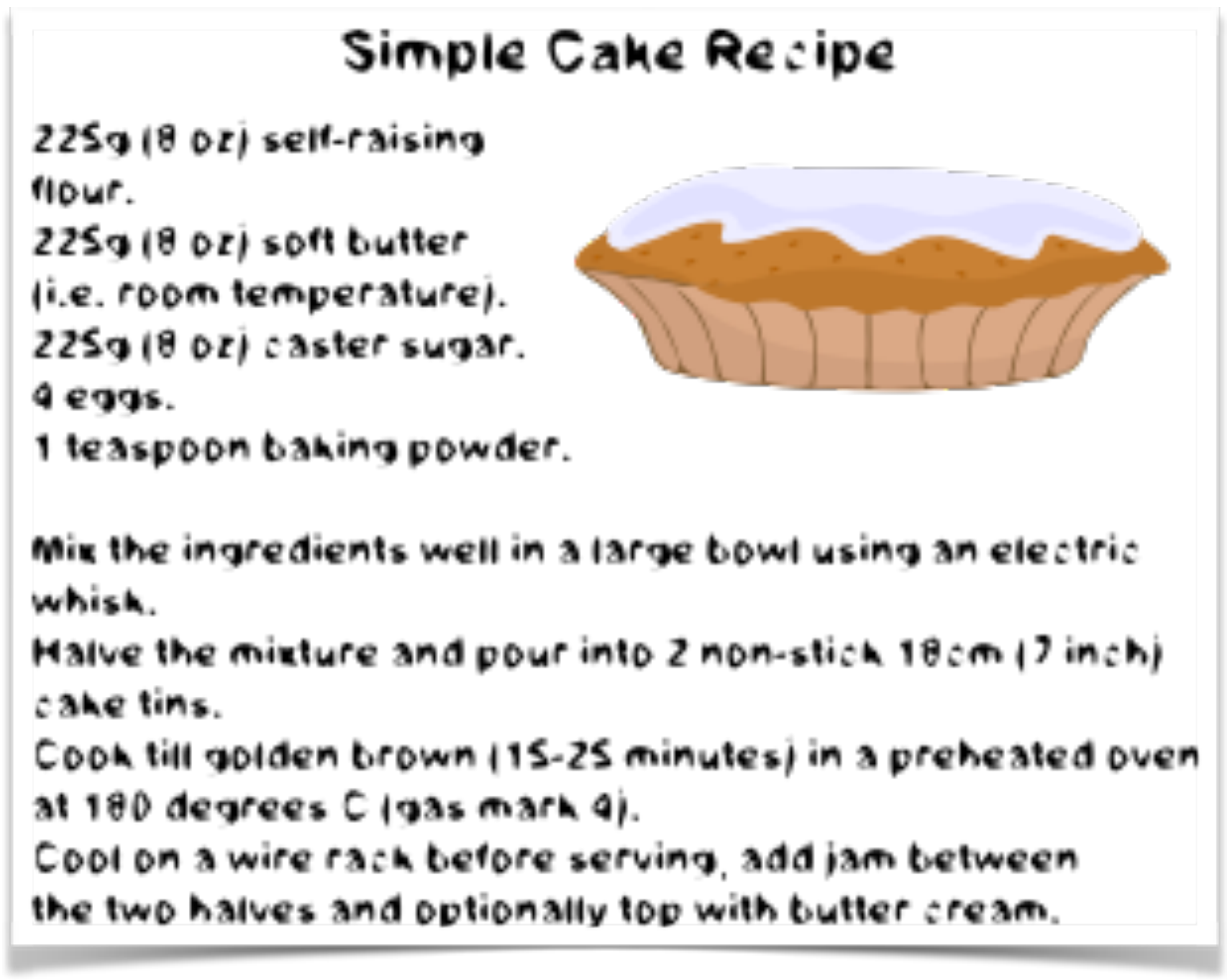
- Linear
- No names
- No libraries
- Global variables

Hard to read

“Bad”

“Fast”

Cookbook



Simple Cake Recipe

225g (8 oz) self-raising flour.
225g (8 oz) soft butter (i.e. room temperature).
225g (8 oz) caster sugar.
4 eggs.
1 teaspoon baking powder.

Mix the ingredients well in a large bowl using an electric whisk.
Halve the mixture and pour into 2 non-stick 18cm (7 inch) cake tins.
Cook till golden brown (15-25 minutes) in a preheated oven at 180 degrees C (gas mark 4).
Cool on a wire rack before serving, add jam between the two halves and optionally top with butter cream.

Cookbook

- Procedures/subroutines
- Shared, global state
- *Structured programming*

Names as sign-posts

Side-effects

Non-idempotent:
"Asking twice can change the result"

Pipeline

Pipeline

- (Pure) Functions!
- No side-effects
- Immutable data
- Function composition

Idempotence!

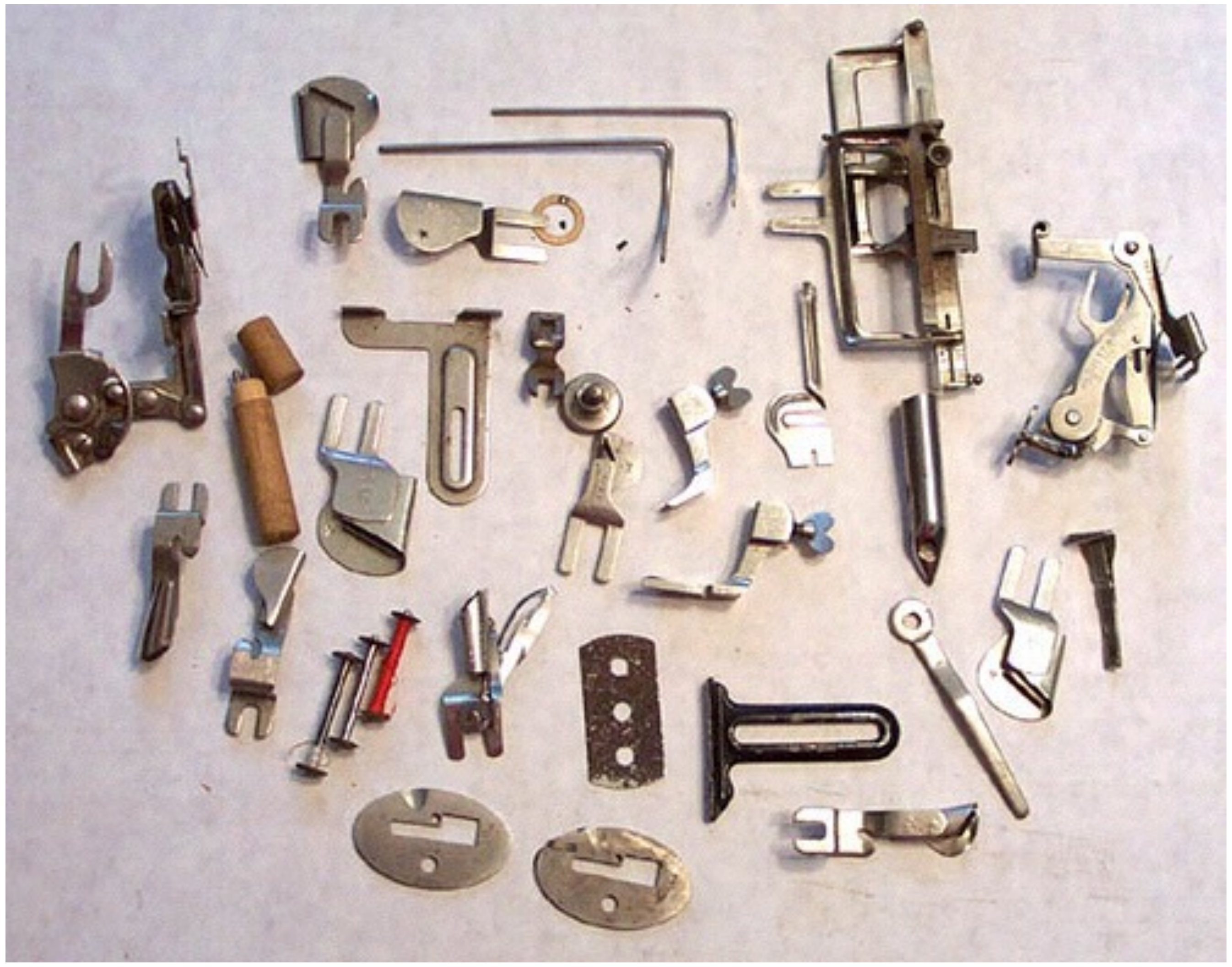
Unit testing is easy.

Concurrency is easier.

Equational reasoning (Substitute equals for equals).

Objects & object interaction

- Things
- Abstract Things
- Hollywood



Things

Fil mjölk
Mjölk
Kaffe
ISO
Öboy
Nalle RIISI
Talous-sokeri
Uncle Ben's
5 min
GEVALIA
Daim
KEX
Colgate

Things

- Domain objects
- Data + behavior
- Encapsulation

Information hiding.

Encapsulate representation.

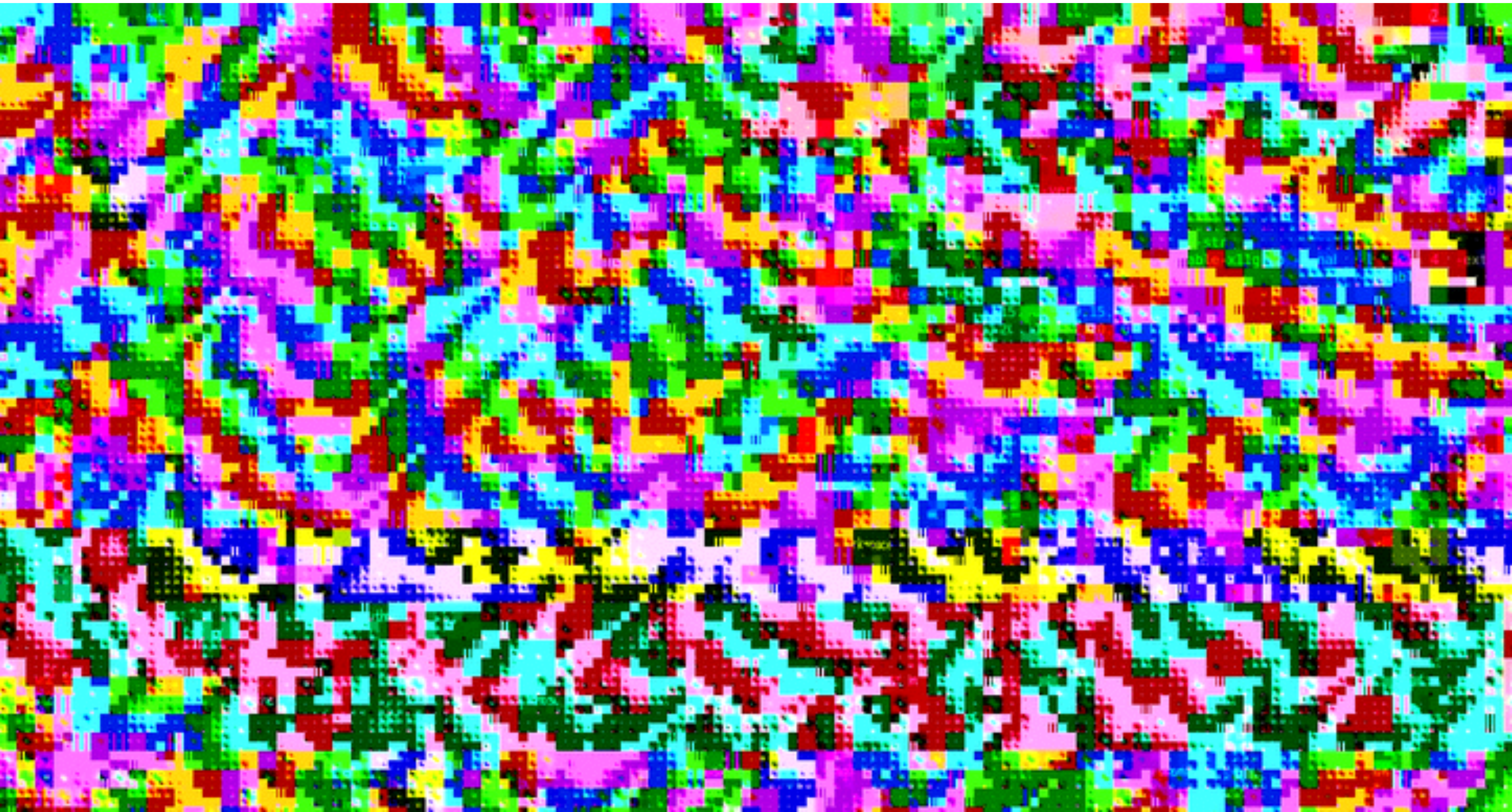
Local state.

“Modeling the real world.”

Oscar Nierstrasz

Inheritance:
-Domain analysis Conceptual hierarchies
-Design: polymorphism
-Implementation: reuse

Abstract things



Abstract things

- **Interfaces**
- Binding to implementation
- Mediates communication
- Types/contracts

Polymorphism: multiple impls can serve same interface

Again: information hiding.

HOLLYWOOD

- “Don’t call us, we’ll call you”
- Call-backs
- Inversion of control

Frameworks (e.g. GUI).

One-to-many calls.

Can be hard to understand!

Dealing with adversity

- Constructivism
- Tantrum
- Passive aggressive

Constructivism

Constructivism

- Check for errors
- Try to continue, always

Can be good for user experience.

Can also lead to weird behavior (Misinformation).

Tantrum

Tantrum

- Check for errors
- “Handle” and propagate

Can be verbose:
Try-catch everywhere

Checking of return values in C.

Cf. ASML: 30% error

Passive aggressive

- Check for errors
- IFF you can handle them
- Otherwise: propagate

Allows you to centralize error handling

Extreme: **assert**

Conclusion

- 9 different styles for the same program
- Basic styles, Object styles, Dealing with adversity
- Not tied to programming language
- Design is about **decisions by you**!!!!
- Different trade-offs, benefits, drawbacks

Change is the only constant

- Stopwords not from file but URL?
- Different sorting of frequencies?
- Show different top-N number?
- Don’t count stop-words?

W.r.t. the lab assignment

- AST hierarchy
- Type checking
- Expression evaluation
- Rendering
- Event handling

Ask yourself: which style would fit this component best?

What are the trade-offs?

How to structure the interaction between components?

How do I deal with errors?