Algorithmic Sustainable Design: The Future of Architectural Theory.

Nikos A. Salingaros
University of Texas at San Antonio

Lecture 5

Harmony-seeking computations.

A. Architectural harmony.

B. Alexander's theory of centers.

C. Design as computation.

D. Computational reducibility

A. Architectural harmony

- GOAL OF COMPUTATION: improve coherence of the design by successive steps
- Mathematical model of "harmony" given in "*A Theory of Architecture*"
- Harmony estimates density of symmetries, connections, scaling coherence, universal scaling, universal distribution, etc.

San Miniato al Monte, Florence

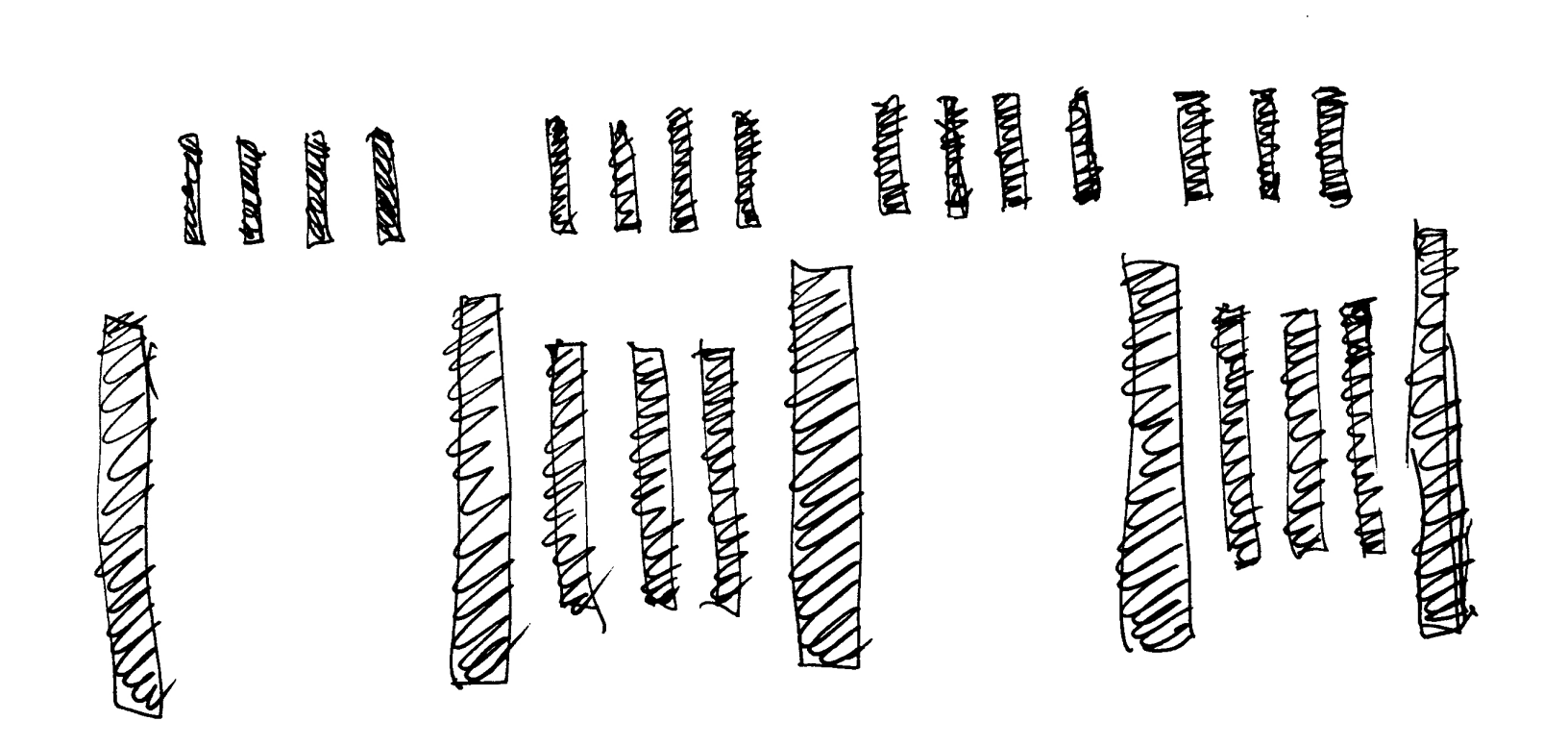
Estimate the harmony

- Reflectional symmetries on all scales = 2/2
- Translational and rotational symmetries on all scales = 2/2
- Scaling symmetries = 1/2
- Geometrical connections = 2/2
- Color harmonization = 1/2
- *Sum to get **total harmony = 80%***

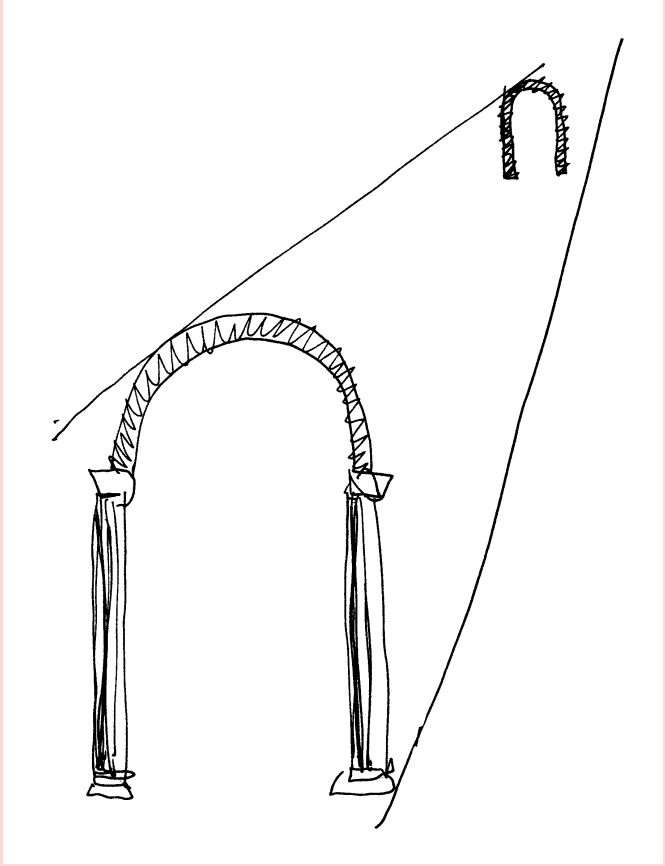
Method of estimation

- Simplest estimate for each property seen in obvious design characteristics:
- *NONE = 0*
- *SOME, NOTICEABLE = 1*
- *A GREAT DEAL = 2*
- Each of the 5 components of the architectural harmony adds up to give a percentage measure

Translational symmetries



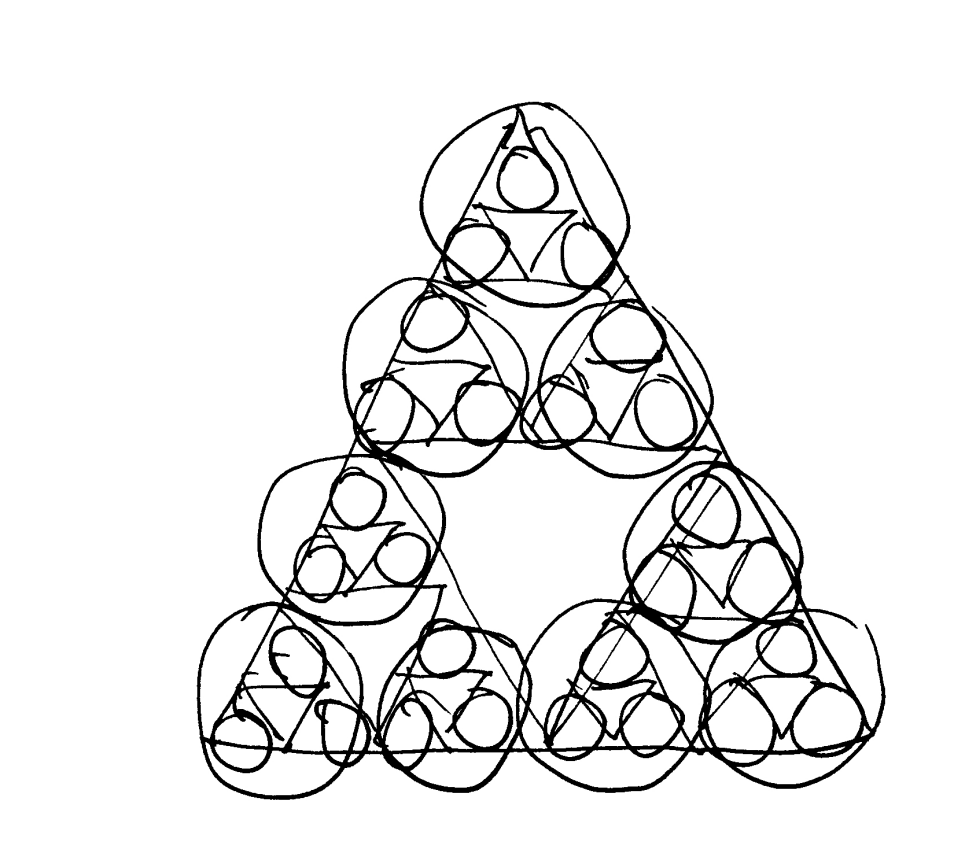
Scaling symmetries



B. Christopher Alexander's theory of centers

- Basic notion describing the ordering process in nature (and in architecture)
- The geometry of mutually reinforcing focal points
- Independent from patterns already obtained via interaction between geometry and social structure

Recursive points of focus (circles) in the Sierpinski gasket



Focus and condensation in fractals

- Self-similarity and the universal distribution require that the details in fractals are not uniformly distributed
- Smaller scales focus in particular regions of a fractal where subdivision occurs

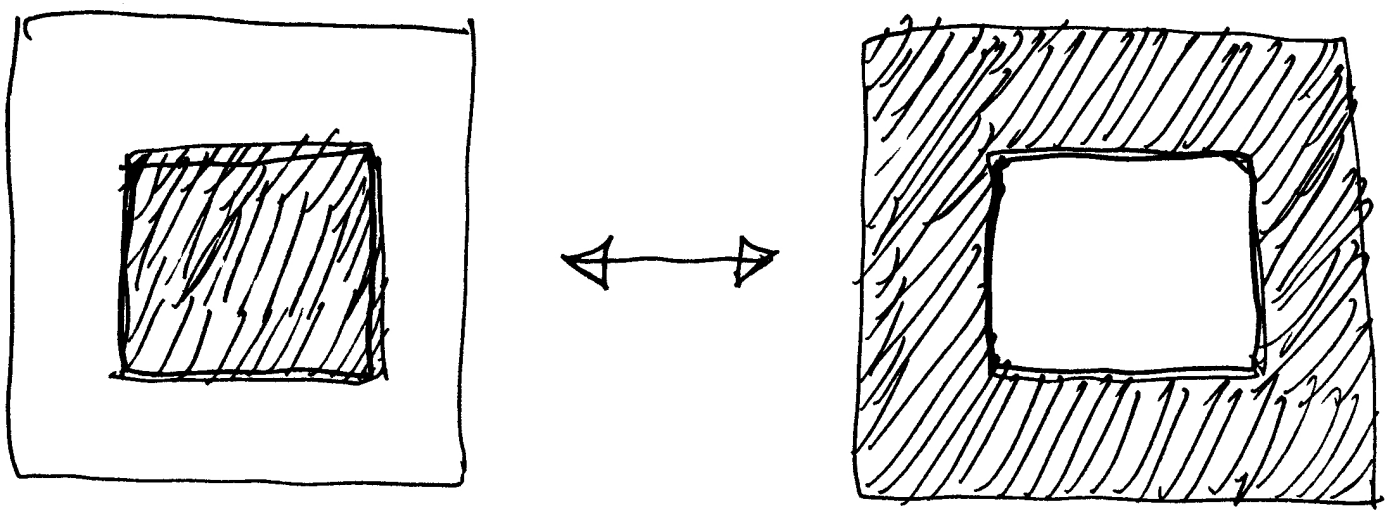
The theory of centers

- A “center” is a visual field that is the focus of a region
- The region that focuses on a “center” can be of any size
- Centers help to tie the space together by reinforcement
- Recursion leads to fractal properties

Centers — structure-void duality

- Two types of centers: “*defined*” and “*implied*” (my own terminology)
- Either a well-defined structure in the middle is surrounded by a looser boundary, or a void is surrounded by a structured boundary
- Mathematically, these two types are dual to each other

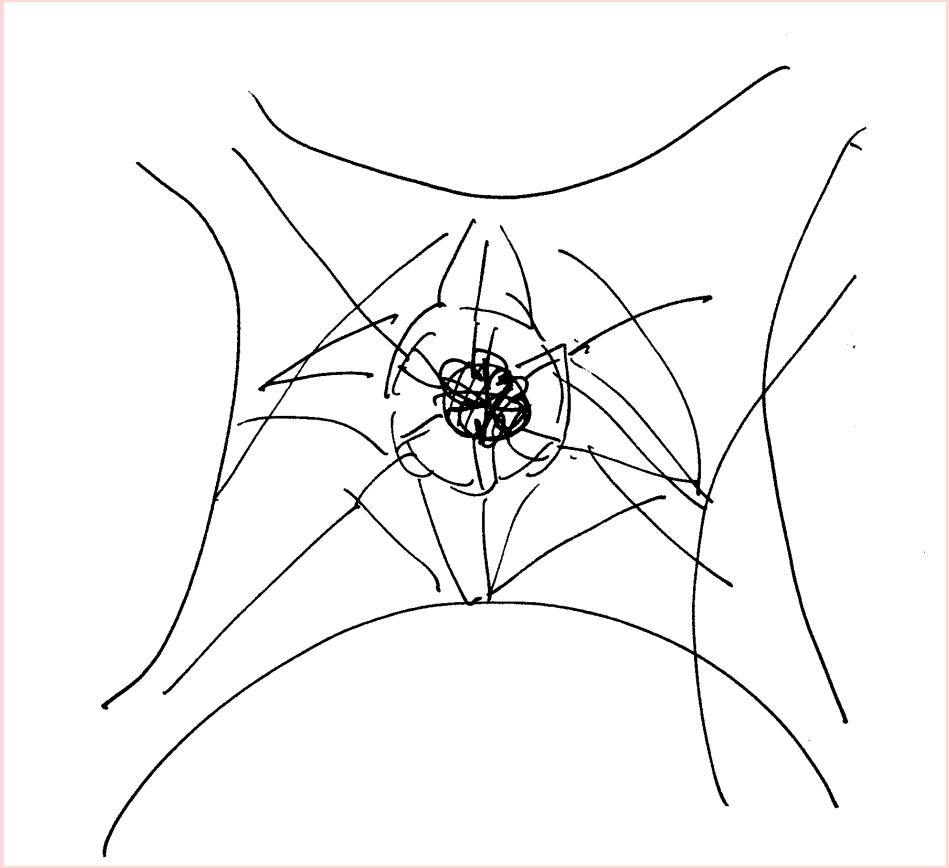
Figure-ground duality



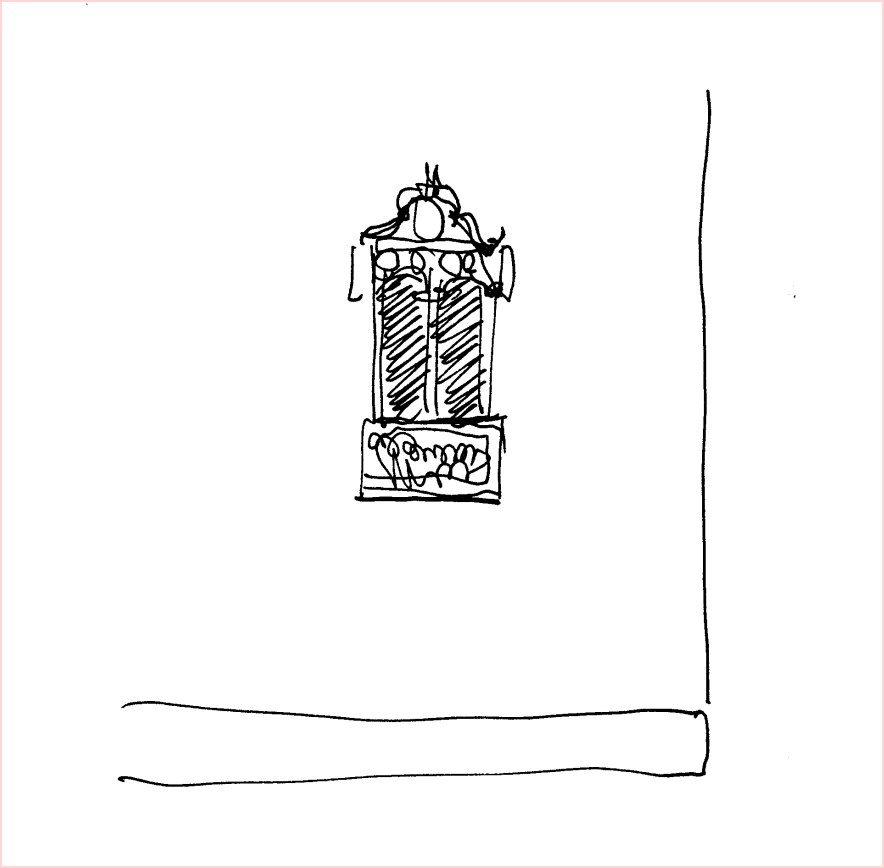
1. “*Defined*” or “*explicit*” centers

- A region in which something right in the middle focuses the structure
- The focal point draws attention to the actual center of a region
- Examples: fountain or sculpture in the middle of plaza; window or door centered in the middle of a wall; light fixture in the center of a ceiling; medallion in paving

Medallion is focal point of ceiling design



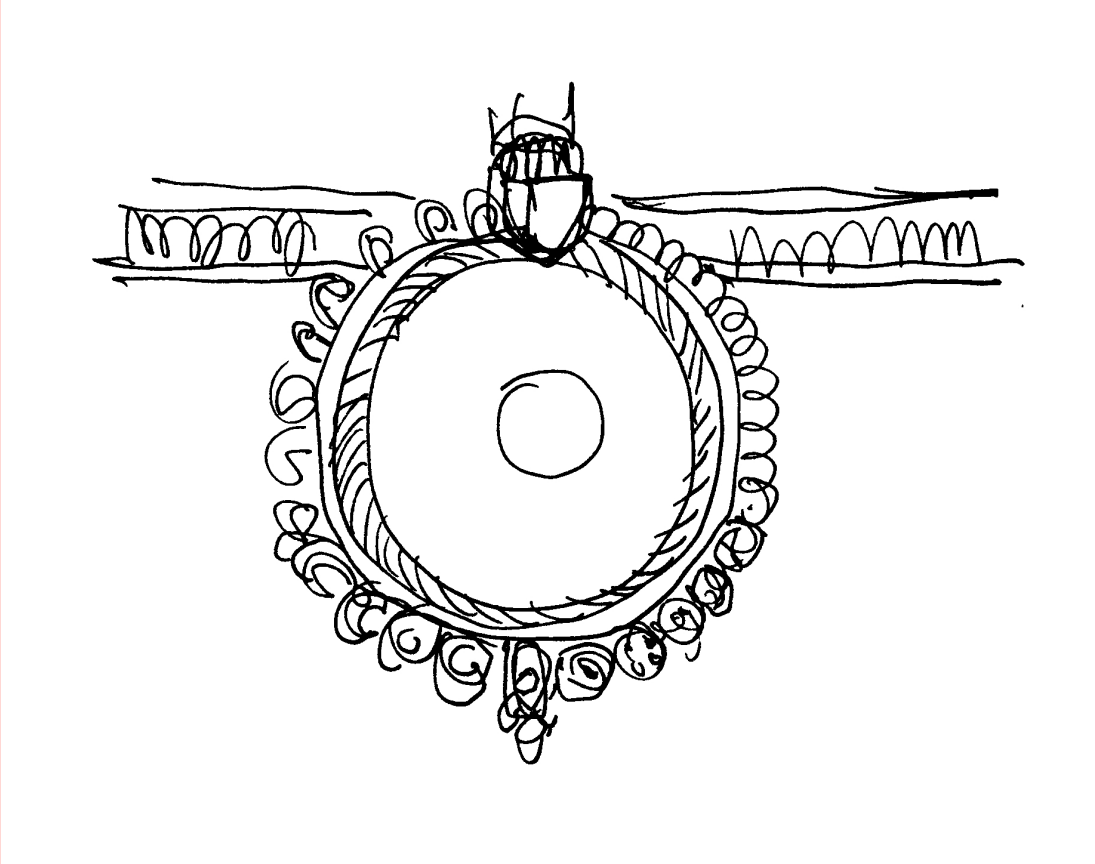
Window is focal point of plain wall



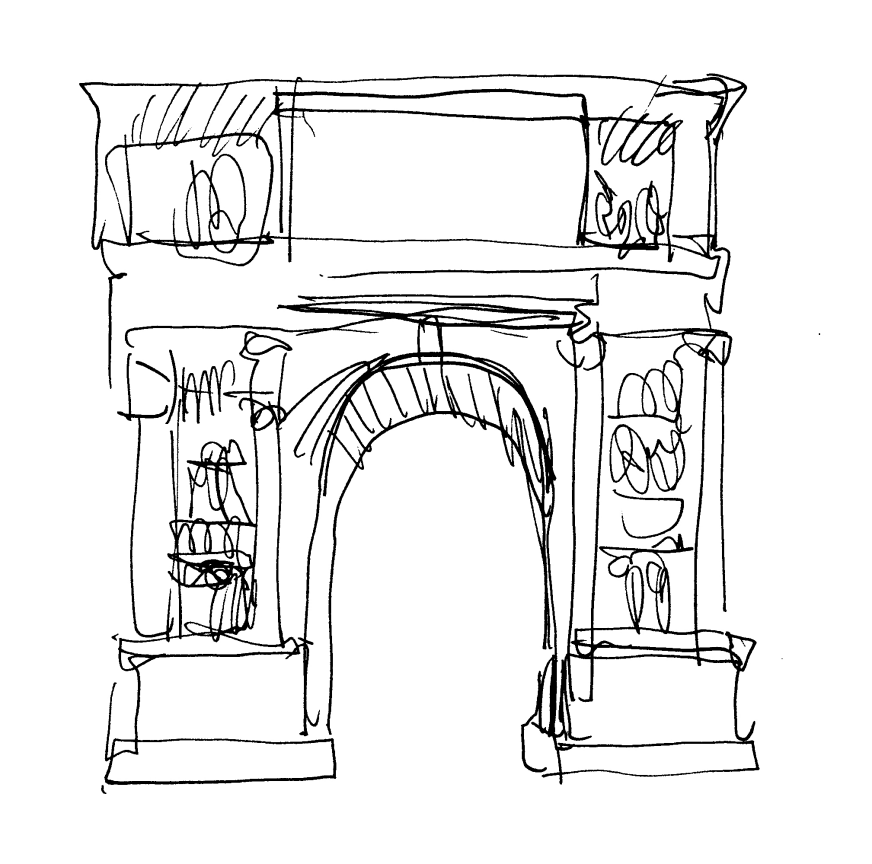
2. "*Implied*" or "*latent*" centers

- A region that focuses on its central point, but where the middle is empty
- Surrounding structure is helping to focus attention towards the interior
- This is a boundary effect — the boundary is focusing on the implied center
- Examples: courtyard enclosed by decorated walls; cloister; decorated arch

Highly ornamented window frame focuses on center



Monumental arch focuses on passageway



Geometrical focus

- Both “defined” and “implied” centers are the foci for their surrounding structures
- “Defined” and “implied” centers can overlap, thus helping each other
- In a coherent design, all the centers cooperate to reinforce each other
- Smaller centers combine to form larger centers — recursive property

Algorithm for generating centers

- Create both strong “defined” and “implicit” centers on a particular scale
- Place/create smaller centers so that they are nested within larger centers
- Use symmetries to make centers cooperate so they support each other geometrically
- Success means that centers blend together

Adaptivity and asymmetry

- We are encouraging the formation of a high density of local symmetries, not an overall symmetry
- *Asymmetry* arises from adaptation, usually seen on larger scales
- But *there needs to be a reason for asymmetry*, not just personal whim

Alexander's first algorithm

- "***Every time you create a center on a particular scale, make sure that it reinforces the centers on the immediately smaller scale, and the centers on the immediately larger scale***"
- From Alexander's "The Nature of Order", Book 1

Alexander's second algorithm

- "***Begin by visualizing the whole. Then identify the scale that is the weakest, or is missing. Create or intensify a center on that scale. The new center must reinforce all existing centers on its own scale, as well as follow rule 1.***"
- From Alexander's "The Nature of Order", Book 3

Example: find a weakness

- *Problem*: some part of your design feels wrong
- Don't just adjust that piece, but look at that SCALE in the entire design
- Ask: WHAT IS THE BEST CENTER THAT REINFORCES THIS SCALE?
- *Solution*: implement that center, rather than adjusting the original faulty piece

Starting from weakness

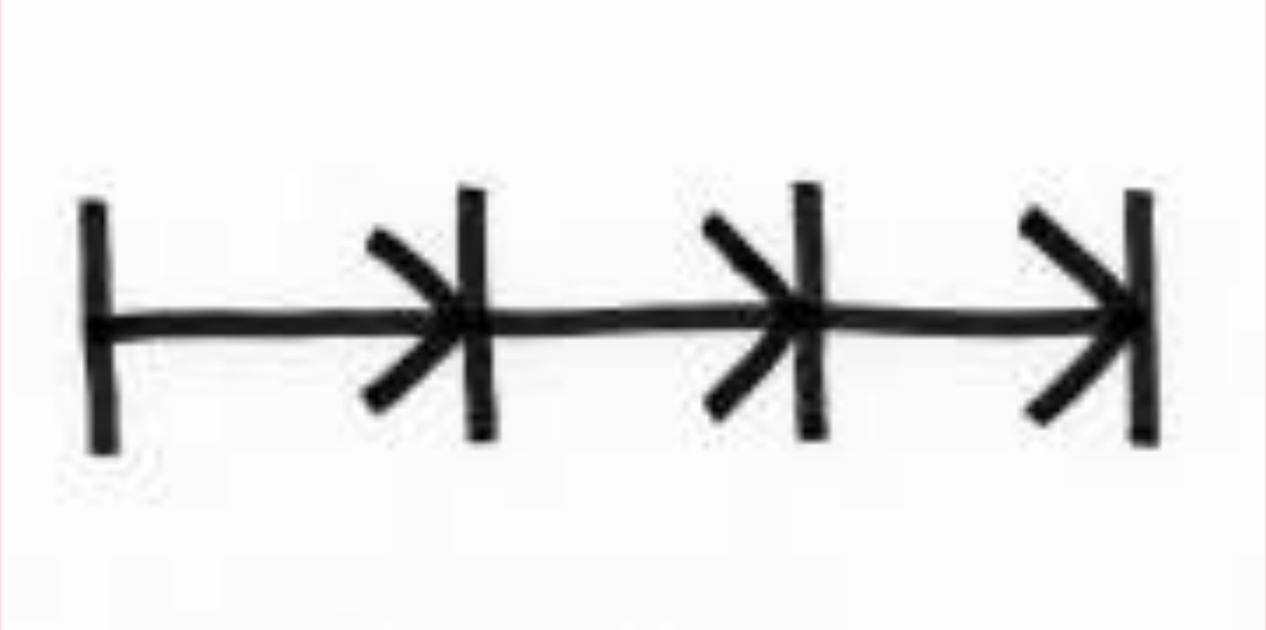
- Usually start from the site, which may contain a weak system of centers
- Apply successful transformations
- Each step creates new centers, or reinforces existing weak centers
- All centers reinforce each other to create a coherent whole

The first set of Leitner diagrams

- Helmut Leitner uses simple visuals to grasp the center-generating transformations
- **1. Stepwise**
- **2. Reversible**
- **3. Structure-preserving**
- **4. Design from weakness**
- **5. New from existing**

1. Stepwise

Perform one step at a time

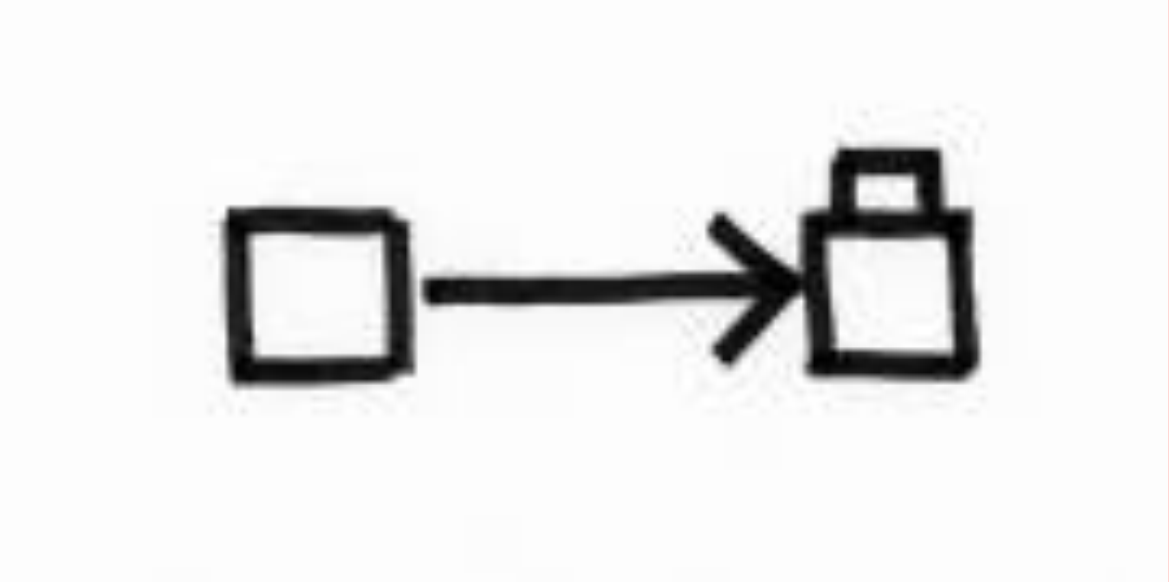


2. Reversible

Test design decisions using models; "trial and error"; if it doesn't work, undo it

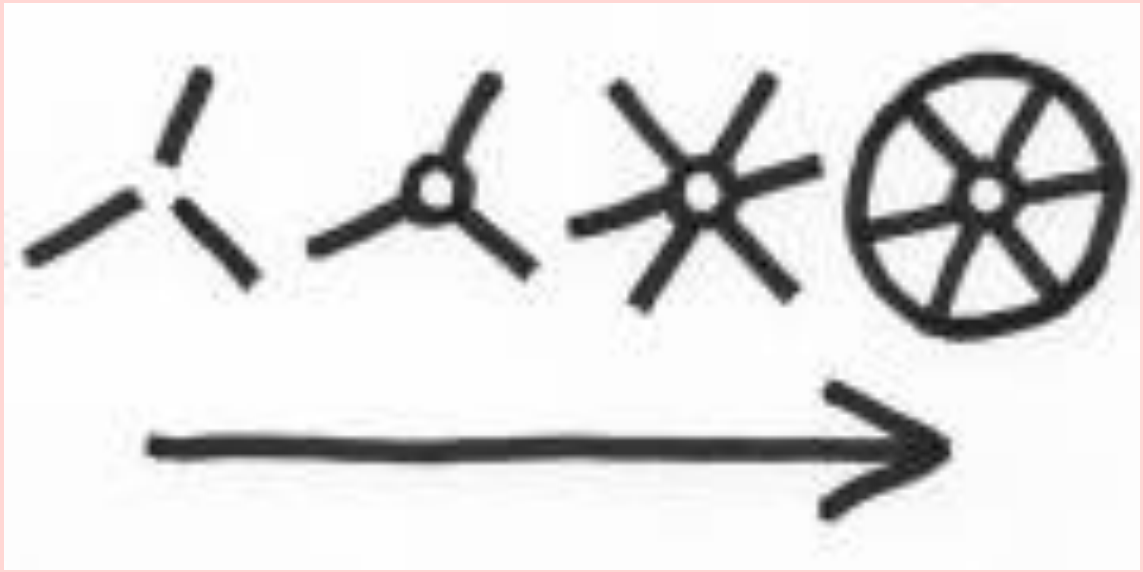
3. Structure-preserving

Each step builds upon what is already there



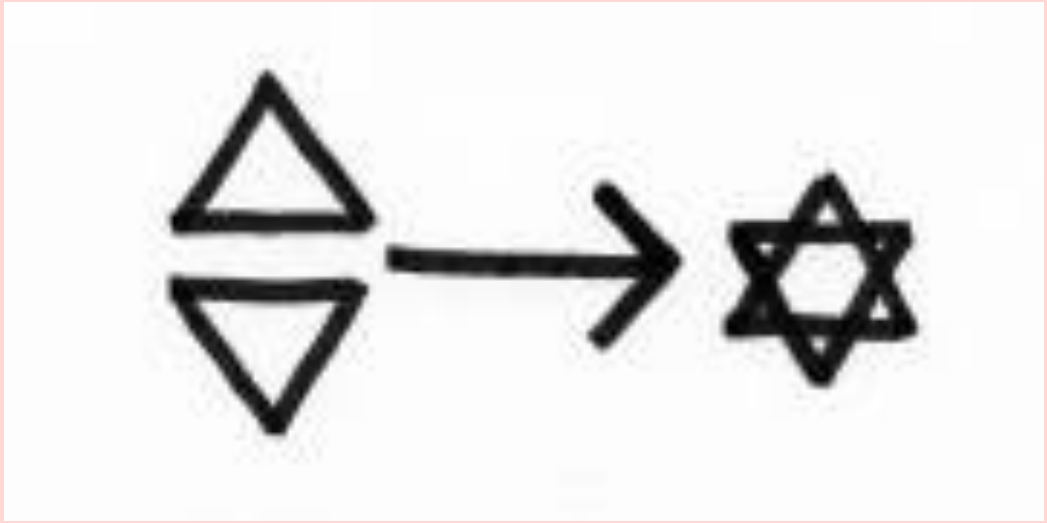
4. Design from weakness

Each step improves coherence



5. New from existing

Emergent structure combines what is already there into new form



Future software

- With time, we can program these rules
- Pattern recognition is a problem of major interest in computer intelligence and vision
- Model for estimating the coherence or "life" of structures is developed in "*A Theory of Architecture*"

Incompleteness theorem

- Software will never substitute for a human designer
- “Living structure” is not possible just from a mathematical algorithm
- Not enough cognitive capacity!
- Computer algorithm is interesting and will be very useful for saving effort

Universal distribution merges to become a field effect

- Centers obey universal distribution: *few large ones, some of intermediate size, many smaller ones*
- Achieving harmony, however, blurs the identity of each center
- Coherence is a "field effect" — the secret of our greatest architecture

C. Design as computation

- Christopher Alexander views successive steps of adaptive design as steps in a complex computation
- Take initial condition as defined by the site, and by successive steps transform it into the final coherent design
- Computation of finite number of steps

Algorithms are recursive

- Algorithm is repeated until a desired level of harmony in achieved, or until the resources run out
- With each succeeding step, coherence of total design is improved
- Next step locates (makes obvious) new bottleneck to coherence

What is our algorithm?

- Alexander's first and second algorithms
- 1. *Identify the weakest or missing center that forms a bottleneck in the harmony of the configuration*
- 2. *Intensify that center*
- 3. *Act both locally and globally*

… but there are more

- These are just two of several algorithms acting together
- More process principles are needed for computation
- Process concepts are not yet as well developed as structural concepts
- Refer to Leitner's first set of diagrams

What are the constraints?

- 1. Brief of project (a) — functions
- 2. Brief of project (b) — human needs
- 3. Biophilic considerations — human feelings of wellbeing
- 4. Patterns from a Pattern Language
- 5. Connecting to the surroundings

Patterns as complex socio-geometric “centers”

- Socio-geometrical ways of behavior
- Repeated rediscovery of useful configurations in buildings and cities
- Classified in Alexander’s book: “*A Pattern Language*”
- Come from participatory design
- Not a pure geometrical concept

What are the programming tools?

- ***1. Alexander's 15 fundamental properties***: provide the "code" in which the algorithm is written and implemented (next lecture)
- ***2. Process principles***: to be developed more
- ***3. Connecting concepts***: universal scaling, universal distribution, wide boundaries, architectural harmony, centers, etc.

Goal of computation

- Goal is not what one would expect!
- Algorithm does ***not*** compute the typology of the building (e.g. house)
- Algorithm computes ***harmony***, and each step proceeds by improving the harmony
- Function of building lies in the constraints!

Formal decomposition

- Algorithm broken up into specific computational loops (in theory)
- But this decomposition does not even touch the implementation problems!
- How do we achieve “living structure”?
- Not only geometrical harmony
- Need to incorporate patterns

High-level description

- Algorithm: *larger main loop computes architectural harmony*
- Several nested secondary iterative loops act as constraints:
- *— project brief*; *patterns from "A Pattern Language"*; *universal scaling*; *universal distribution…*

Non-adaptive architectural design

- A drawing based on images has nothing to do with an adaptive building
- An adaptive design must be computed!
- Human mind is the best pattern computer
- *The number of computations is proportional to the complexity of the desired result*
- There can be no shortcuts to final form

Most design is memory-based

- No computation at all
- Retrieval from a memory bank
- Even if architect is convinced he/she is being totally innovative, design is usually coming out of subconscious memory
- Harmony-seeking computations are rarely applied by architects in the industrial world

Good and bad memory

- Stored proven patterns are good
- Evolved over generations, tested and survived by adaptive selection
- But recycling of faulty design patterns gives bad designs
- Therefore: ***need periodic checks for the correctness of stored patterns***

Algorithmic checks

- Coherence and cooperation of different elements among different levels of scale
- Analogous to the coherence of a fractal
- Alexander's fifteen fundamental properties help achieve living quality
- Global-local geometrical property

Emergence

- A very simple algorithm acting on the smallest scale generates a complex pattern with long-range geometrical features
- Complex geometrical properties are emergent
- They are not obvious in the initial code

Alexander's harmony-seeking process is more than emergent

- Emergence is only a two-way process
- Smaller components cooperate to create a larger whole — link small with large
- Harmony-seeking computations have an additional element — three-way process
- Whole interacts with an even larger external entity — ***small, with large, with outside***

D. Computational reducibility

- General misunderstanding of how much work is required to create a complex system
- Design generates complex systems
- Everyone wants shortcuts
- Some shortcuts compromise system coherence and functionality

Computational processes

- All processes can be viewed as computations (Stephen Wolfram)
- Both human and natural processes
- Form develops by changing its state on various different levels
- Life continuously changes materials of organism, but maintains form template

Computational reducibility

- Adaptive systems evolve, with each step being a computation
- In simple physical systems, we don't need to duplicate the amount of computational effort, but can shortcut to final state — i.e., use a formula
- Simple case is *computationally reducible*

Computational **ir**reducibility

- In irreducibly complex systems, there are no formulas for finding the final state
- Computation of final state requires the same effort as the system has gone through to create itself — no reduction
- Stephen Wolfram's "computational irreducibility"

The reducibility fallacy

- Design that is adaptive needs to compute a large number of steps
- The algorithm is usually recursive
- Such a process is ***computationally irreducible***
- It is therefore impossible to make a top-down design so that it is adaptive

General procedure

- Decompose design problem into more tractable subunits or components
- Decomposition is dictated by experience
- Employ known methods (relying upon precedent) to evaluate subroutines
- Re-assemble partial results into final result
- Initial decomposition determines re-assembly

General procedure (cont.)

- Require selection criteria to be able to eliminate false positives
- How do you recognize false steps?
- Again, this relies upon precedent
- Process is successful if large scale structure is adaptive, not if it is strange or irrelevant

Conclusion: computational equivalence

- Classical and traditional architects follow part of our algorithm for design
- From computational irreducibility, ***all adaptive design algorithms are computationally equivalent***
- Any inequivalent algorithm cannot be adaptive