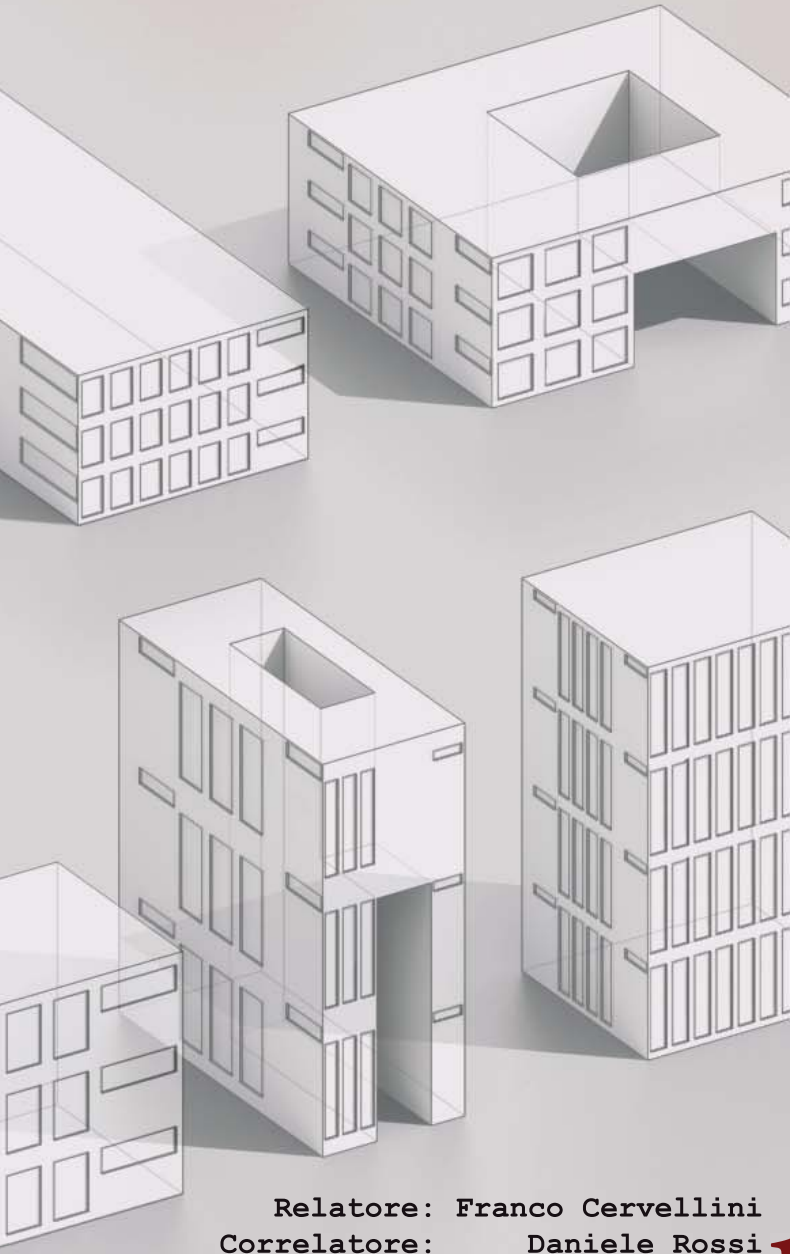
Università degli studi di Camerino
Facoltà di Architettura
di Ascoli Piceno
a.a. 2004/2005

Indagini sperimentali intorno alla forma architettonica

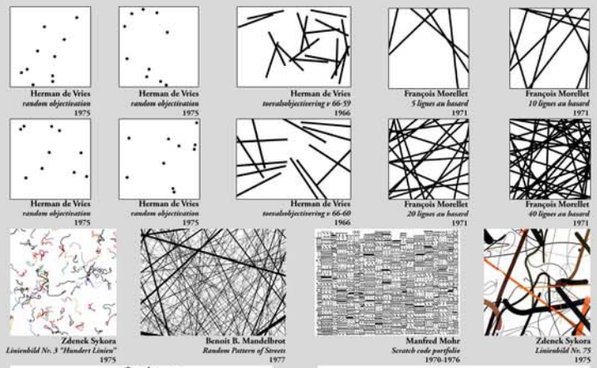


Relatore: Franco Cervellini
Correlatore: Daniele Rossi
Laureando: Filippo Sicuranza

Tavole

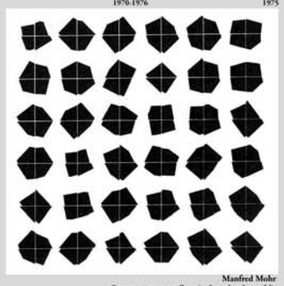
Algorithmic Art

AA. VV.



Herman de Vries — *random objectivation* 1975
Herman de Vries — *random objectivation* 1975
Herman de Vries — *toevalsobjectivering v 66-59* 1966
François Morellet — *5 lignes au hasard* 1971
François Morellet — *10 lignes au hasard* 1971
Herman de Vries — *random objectivation* 1975
Herman de Vries — *random objectivation* 1975
Herman de Vries — *toevalsobjectivering v 66-60* 1966
François Morellet — *20 lignes au hasard* 1971
François Morellet — *40 lignes au hasard* 1971
Zdenek Sykora — *Linienbild Nr. 3 "Hundert Linien"* 1975
Benoit B. Mandelbrot — *Random Pattern of Streets* 1977
Manfred Mohr — *Scratch code portfolio* 1970-1976
Zdenek Sykora — *Linienbild Nr. 75* 1975

Manfred Mohr
Opera appartenente alla serie: *Scratch code portfolio*
1970-1976

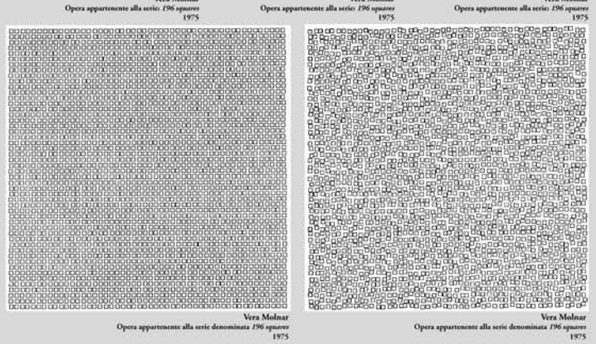


Manfred Mohr
Opera appartenente alla serie: *Scratch code portfolio*
1970-1976

Vera Molnar
Opera appartenente alla serie: *196 squares*
1975

Vera Molnar
Opera appartenente alla serie: *196 squares*
1975

Vera Molnar
Opera appartenente alla serie: *196 squares*
1975



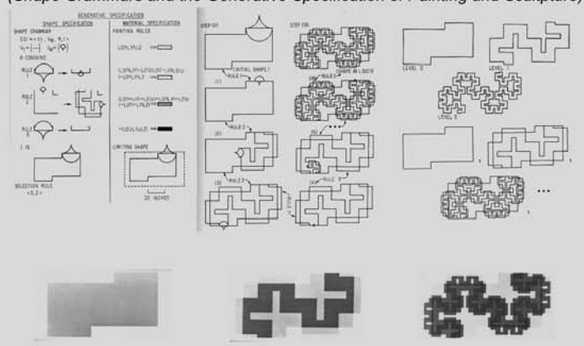
Vera Molnar
Opera appartenente alla serie denominata *196 squares*
1975

Vera Molnar
Opera appartenente alla serie denominata *196 squares*
1975

Shape Grammar

George Stiny e James Gips

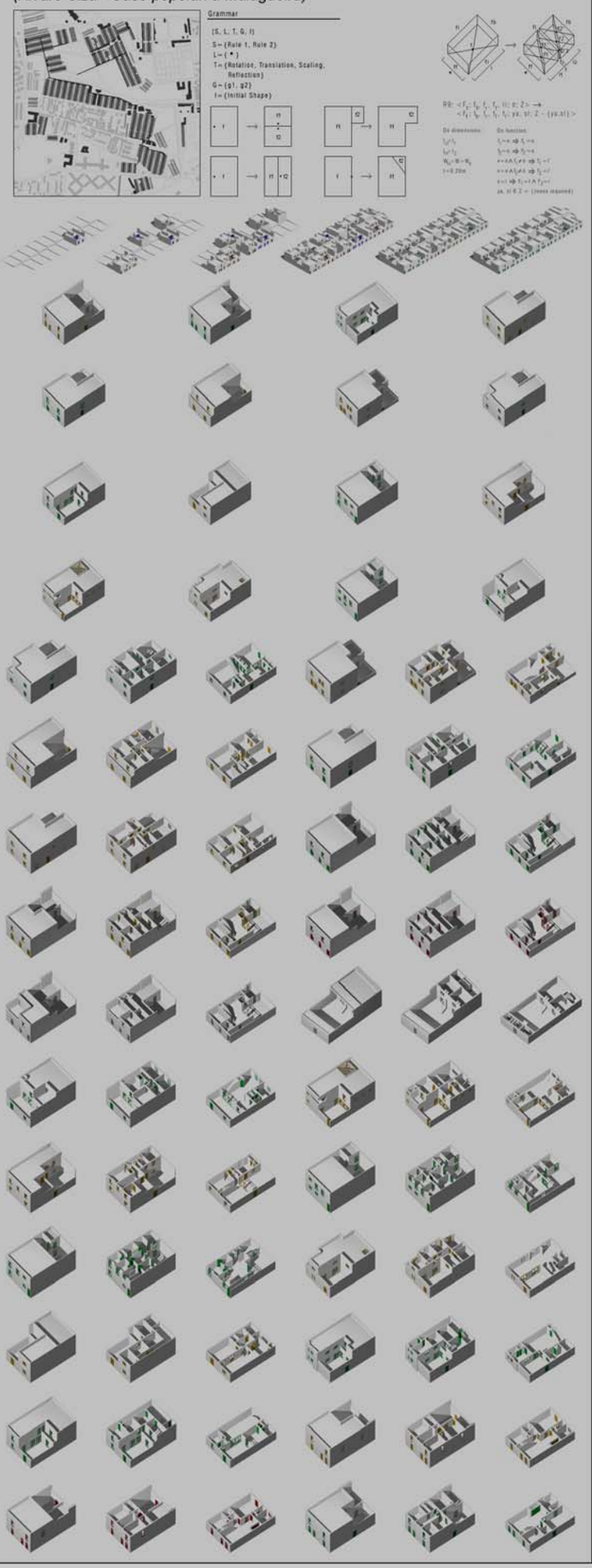
(Shape Grammars and the Generative Specification of Painting and Scultpture)



Shape Grammar

Jose Pinto Duarte

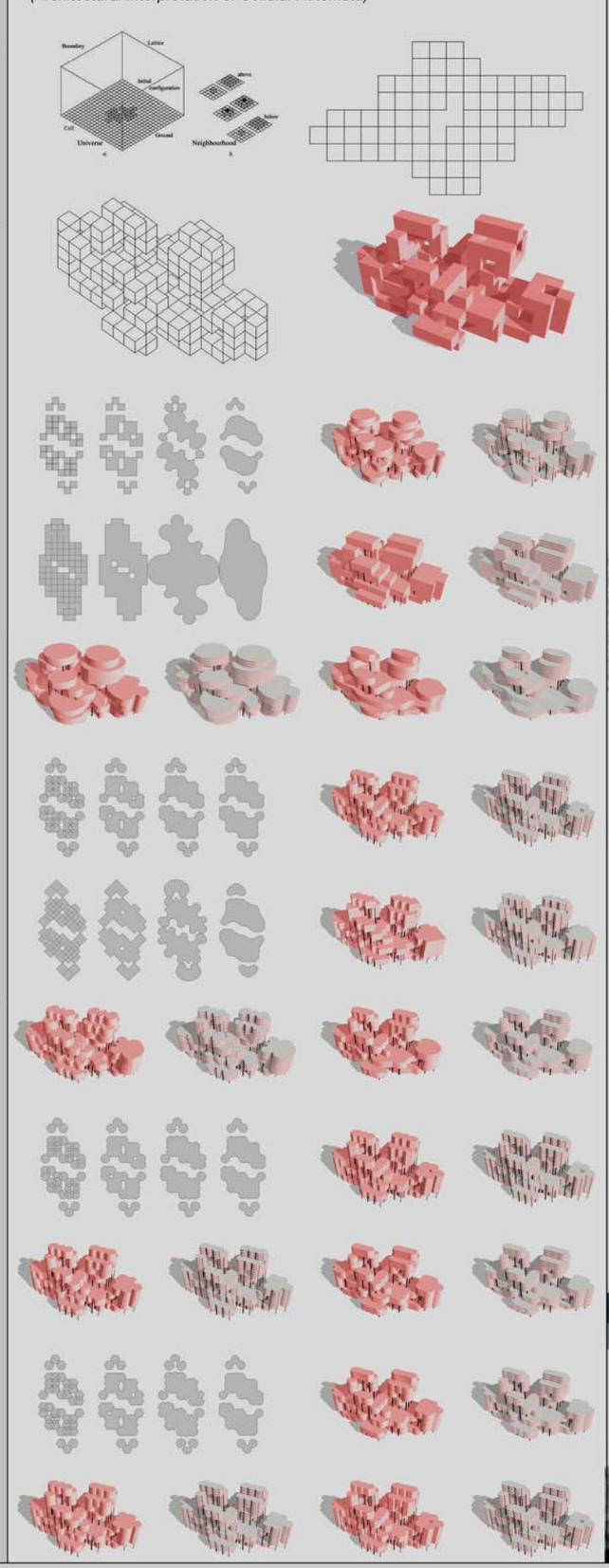
(Alvaro Siza - Case popolari a Malagueira)



Cellular Automata

Robert J. Krawczyk

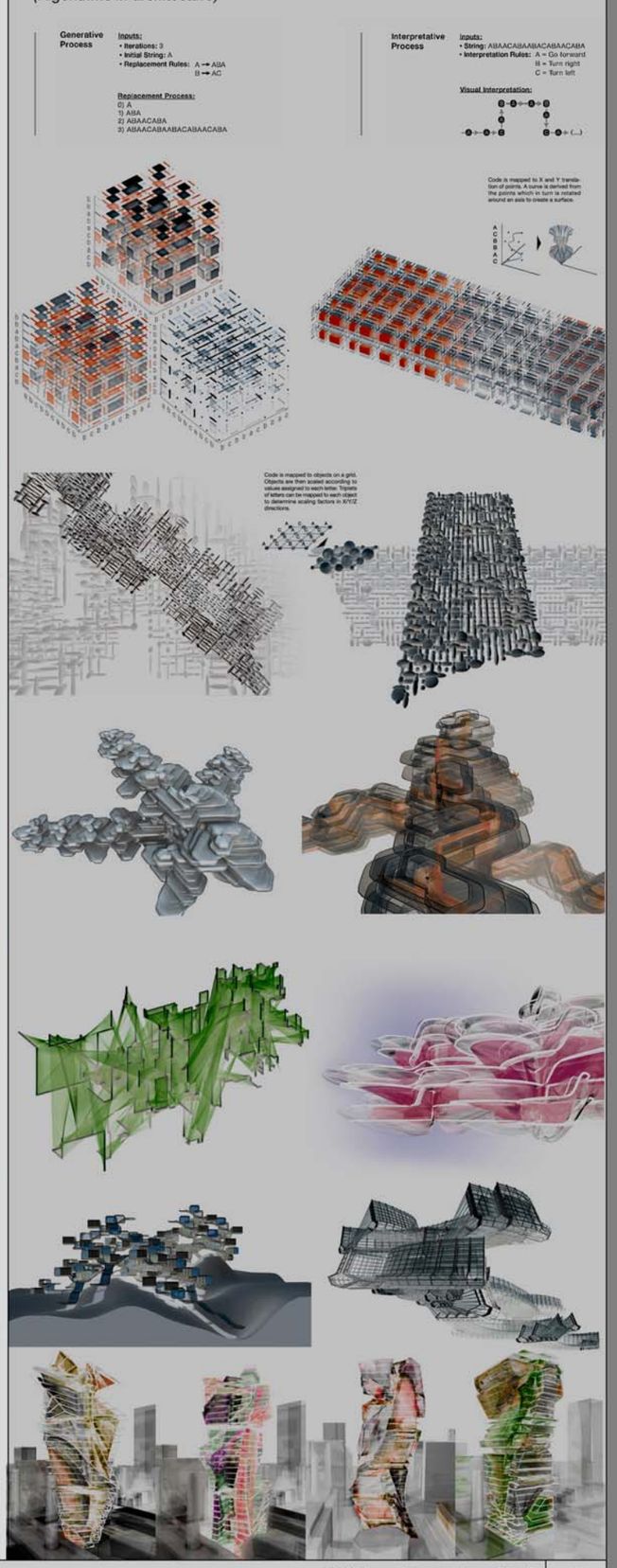
(Architectural Interpretation of Cellular Automata)



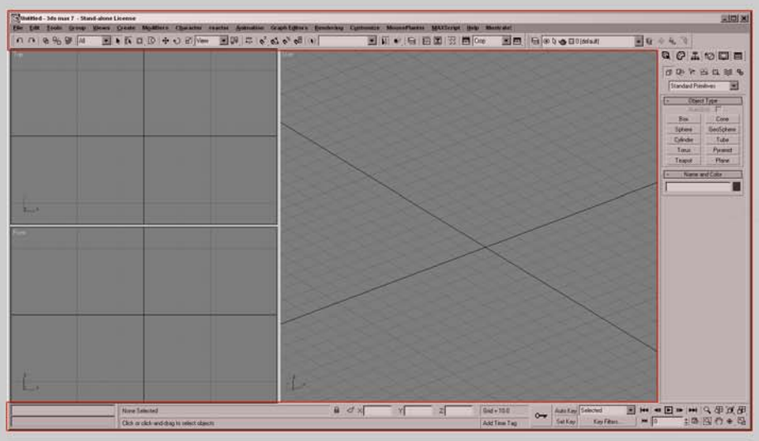
L-System

Michael Hansmeyer

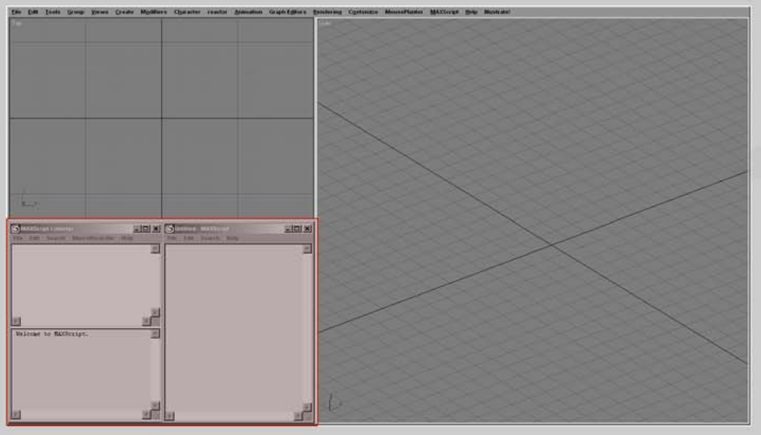
(Algorithms in architecture)



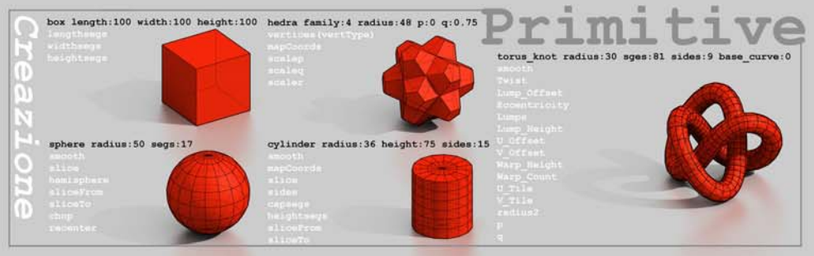
Standard



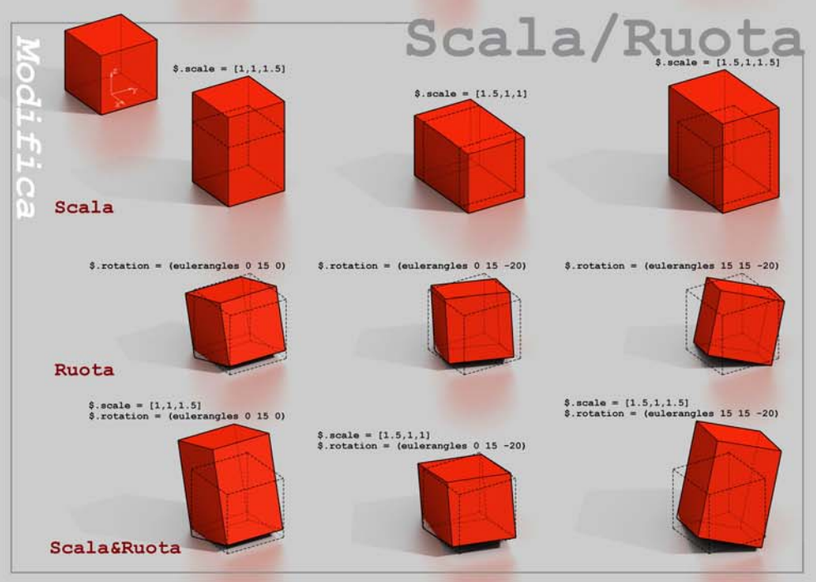
Expert



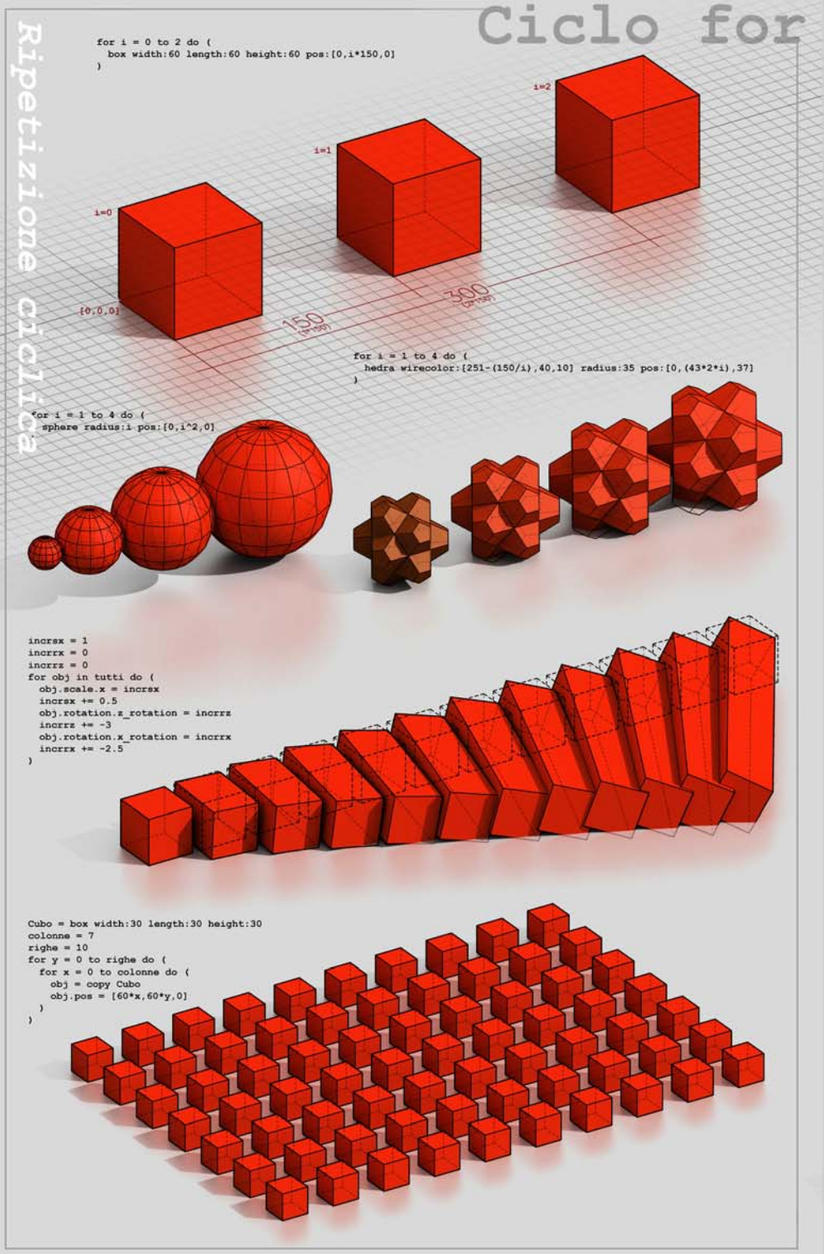
Creazione Primitive



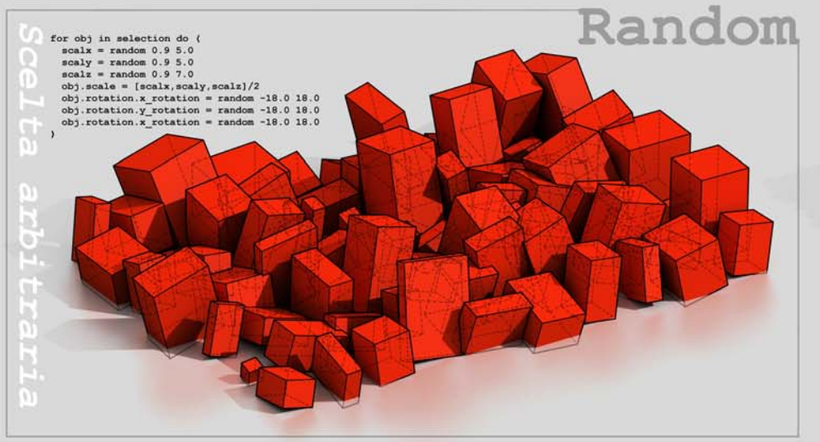
Modifica Scala/Ruota



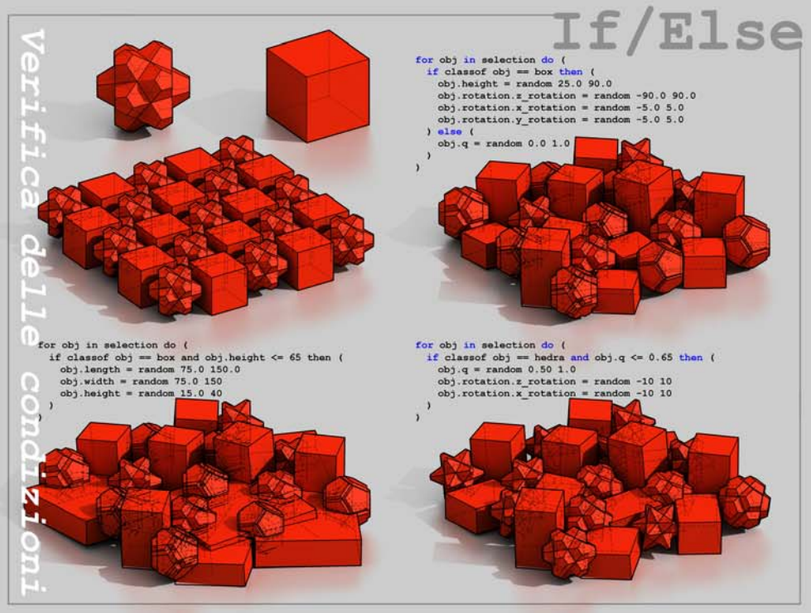
Ripetizione ciclica — Ciclo for



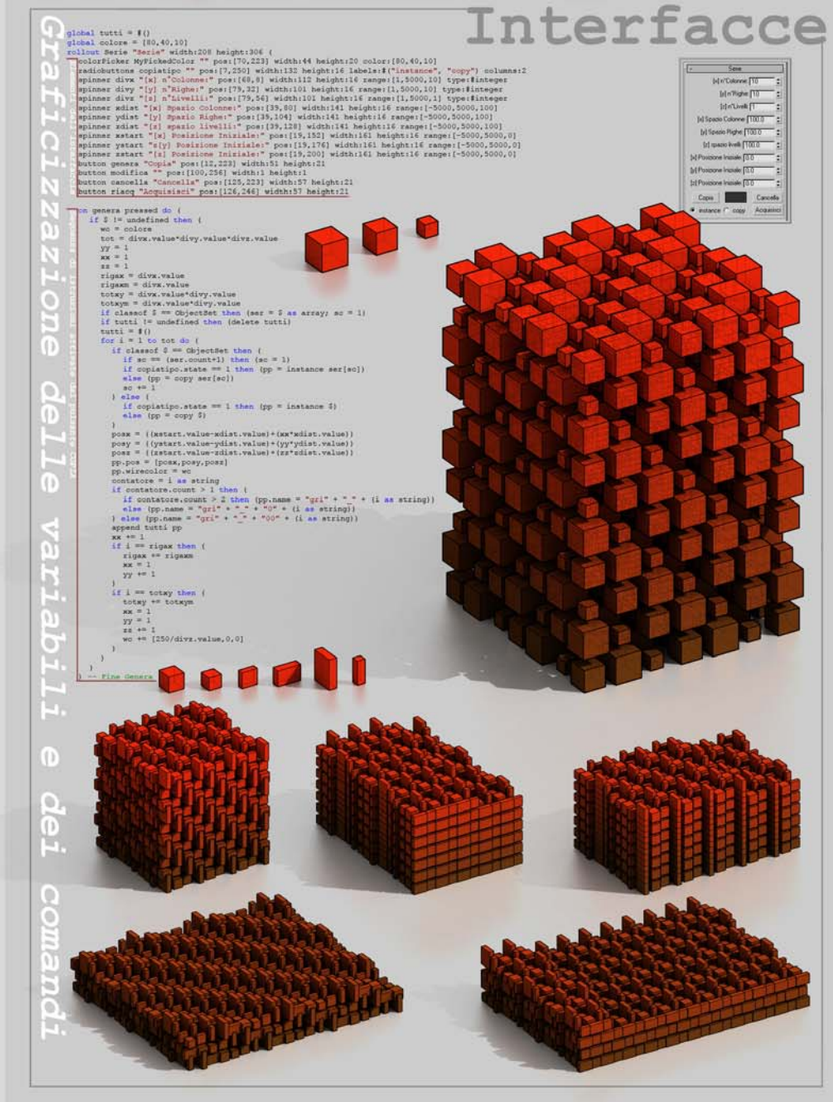
Scelta arbitraria — Random



Verifica delle condizioni — If/Else



Gestione delle variabili e dei comandi — Interfacce



Trasformazioni incrementali

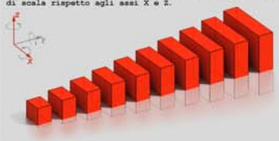
Per trasformazione incrementale si intende un cambiamento di dimensione (scala o stretch) o di orientasento (rotazione) rispetto ad uno o più assi specificati, la cui quantità q, applicata ad un elemento, è pari a q sul primo elemento, a q^2 sul secondo, a q^3 sul terzo e così via.
Qui di seguito vi sono degli esempi che illustrano alcune trasformazioni incrementali applicate ad una serie di 10 elementi.

Nell'esempio sottostante è stata applicata una rotazione incrementale intorno all'asse Z.

Nell'esempio sottostante è stata applicata una rotazione incrementale intorno all'asse Y ed intorno all'asse Z.

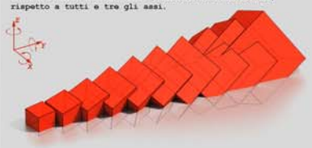
Nell'esempio sottostante è stata applicata una trasformazione di scala rispetto all'asse X.

Nell'esempio sottostante è stata applicata una trasformazione di scala rispetto agli assi X e Z.

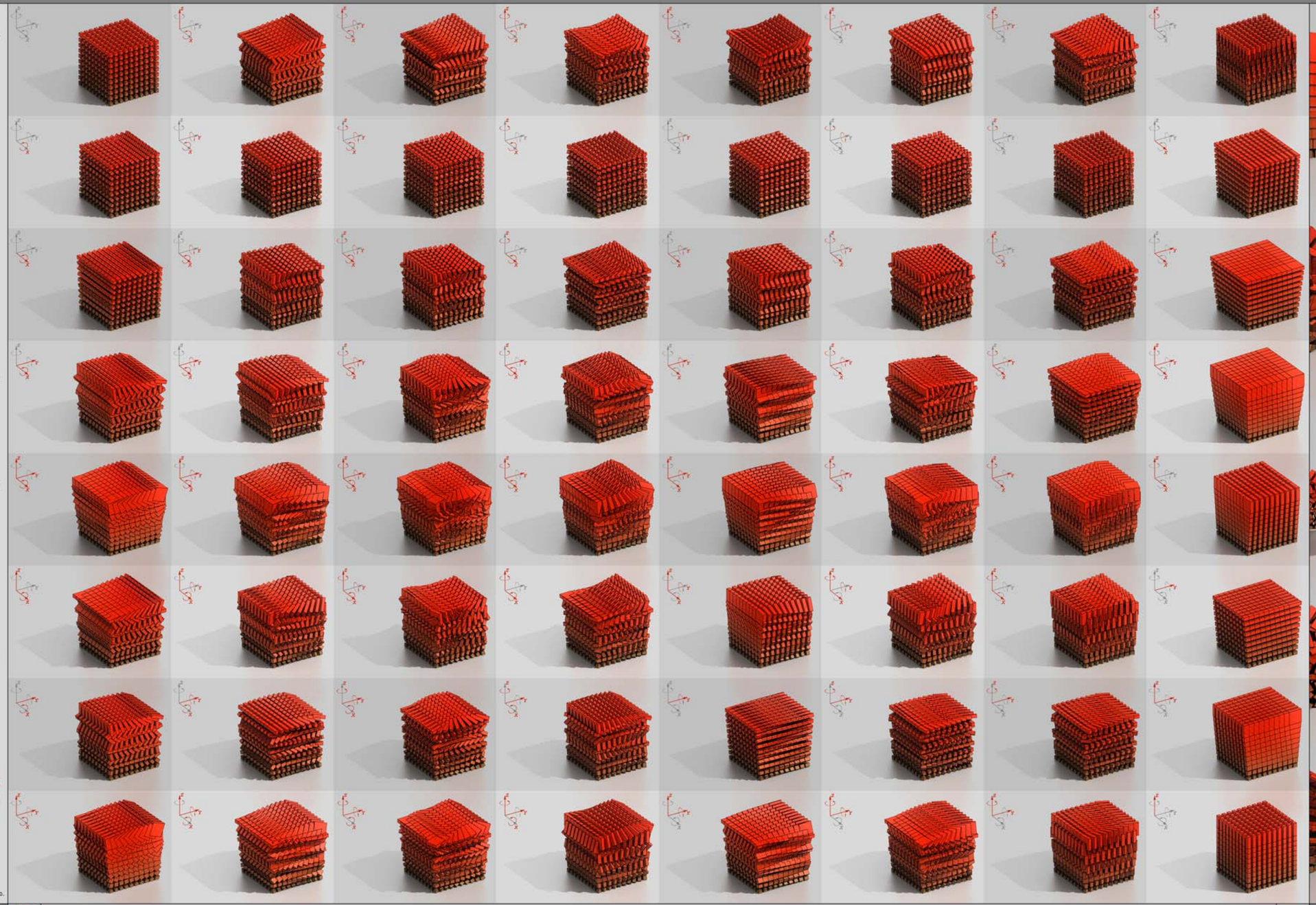


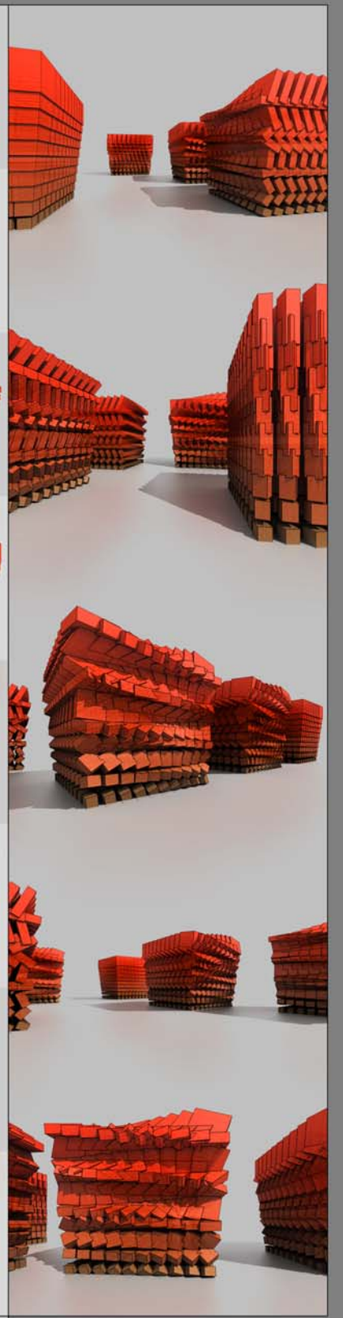
L'esempio sottostante mostra il risultato dell'applicazione simultanea delle trasformazioni degli esempi precedenti.

L'esempio sottostante mostra l'applicazione delle trasformazioni incrementali (rotatorie e dimensionali) rispetto a tutti e tre gli assi.



La tavola mostra lo sviluppo di tutte le possibili trasformazioni (rotatorie e dimensionali) incrementali rispetto ai tre assi fondamentali.
Esse sono state applicate ad una serie di mille elementi disposti in una griglia tridimensionale e ortogonale 10x10x10.





Trasformazioni incrementali dipendenti da un intervallo

L'intervallo, specificato per ogni trasformazione (rotatoria e dimensionale) e per ogni asse, influenza l'applicazione della trasformazione incrementale.
L'intervallo *i* specifica il numero di oggetti al quale la quantità *q* viene aggiunta (incremento). Una volta raggiunto il numero di elementi specificati dall'intervallo *i*, la quantità *q* viene sottratta (decremento).
Qui di seguito vi sono alcuni esempi che illustrano delle trasformazioni incrementali applicate ad una serie di 10 elementi.

Nell'esempio sottostante è stata applicata una rotazione incrementale intorno all'asse Z. L'intervallo che specifica gli incrementi e i decrementi è uguale a 3.

Nell'esempio sottostante è stata applicata una rotazione:
- intorno all'asse Z con un intervallo uguale a 3
- intorno all'asse X con un intervallo uguale a 4

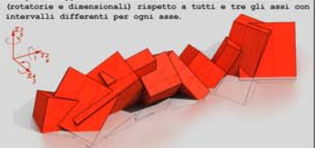
Nell'esempio sottostante è stata applicato un cambiamento di scala rispetto all'asse X con un intervallo uguale a 4

Nell'esempio sottostante è stata applicata una trasformazione di scala rispetto:
- all'asse X con un intervallo uguale a 4
- all'asse Z con un intervallo uguale a 3

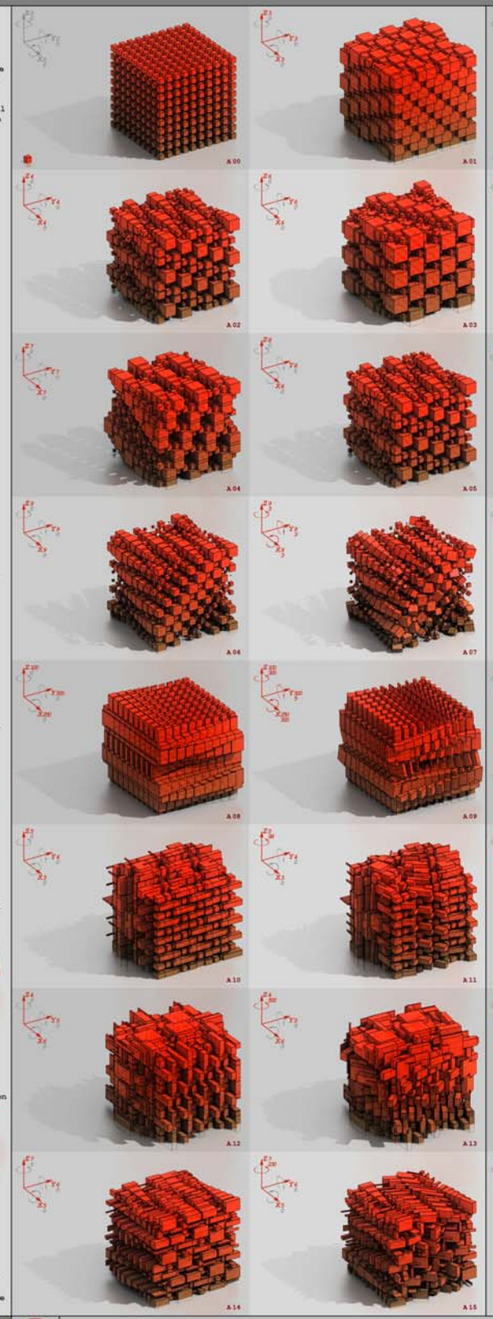
Nell'esempio sottostante è stata applicata una trasformazione di scala rispetto:
- all'asse X con un intervallo uguale a 4
- all'asse Z con un intervallo uguale a 3

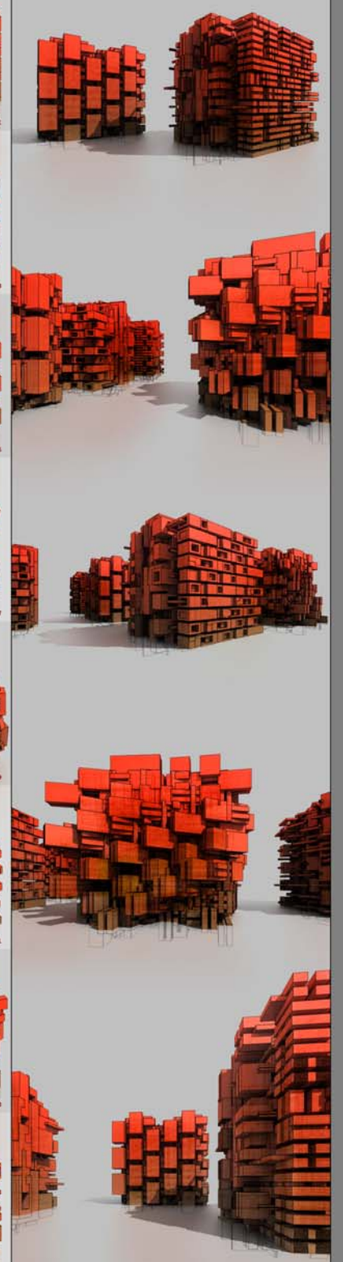
e di rotazione rispetto:
- all'asse X con un intervallo uguale a 4
- all'asse Z con un intervallo uguale a 3

Esempio di applicazione delle trasformazioni incrementali (rotatorie e dimensionali) rispetto a tutti e tre gli assi con intervalli differenti per ogni asse.



La tavola mostra lo sviluppo di alcune possibili trasformazioni (rotatorie e dimensionali) incrementali, dipendenti da intervalli, rispetto ai tre assi fondamentali. Esse sono state applicate a quattro serie di mille elementi modulari disposti in una griglia tridimensionale e ortogonale 10x10x10.





Università degli Studi di Camerino
Facoltà di Architettura
di Ascoli Piceno ~ a.a. 2004/2005

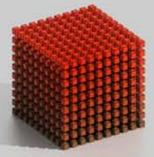
Indagini sperimentali intorno alla forma architettonica

Relatore: Franco Cervellini
Correlatore: Daniele Rossi
Laureando: Filippo Sicuranza

Ridistribuzione

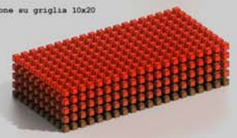
I mille elementi appartenenti ad alcune combinazioni della tavola precedente sono stati ridistribuiti in griglie differenti dalla 10x10x10, mantenendo le loro trasformazioni e cambiando soltanto la posizione rispetto al nuovo reticolo. Fissate la larghezza e la profondità della griglia, l'altezza viene conseguentemente determinata dal numero finito degli elementi.

Griglia originaria 10x10x10

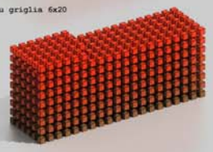


Ridistribuzione su griglia 7x7

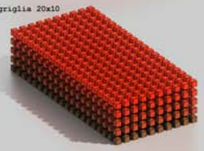
Ridistribuzione su griglia 10x20



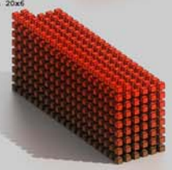
Ridistribuzione su griglia 6x20



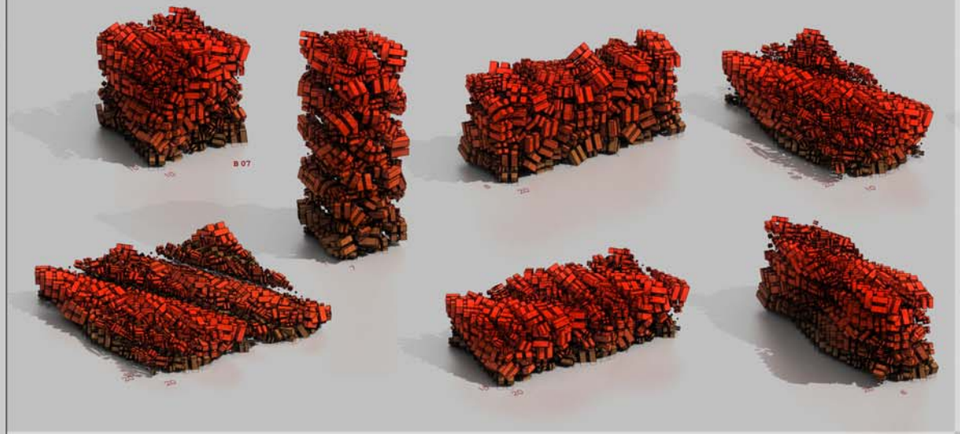
Ridistribuzione su griglia 20x10

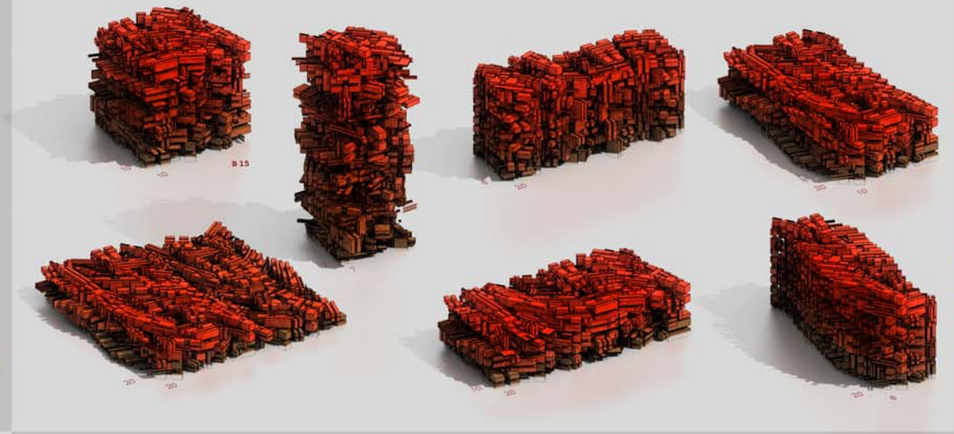


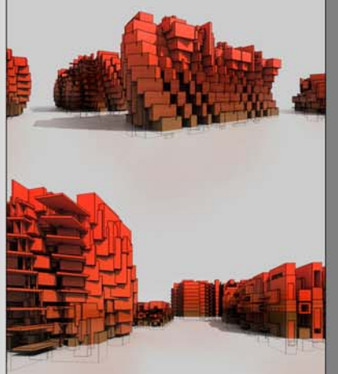
Ridistribuzione su griglia 20x6

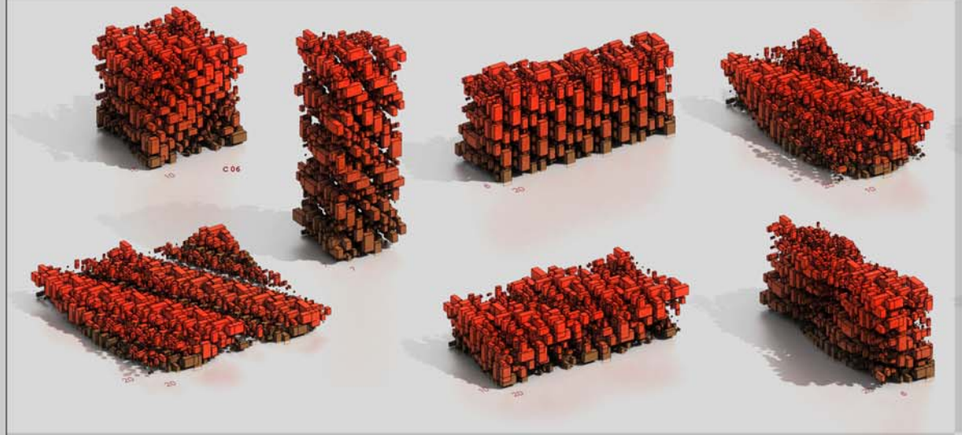


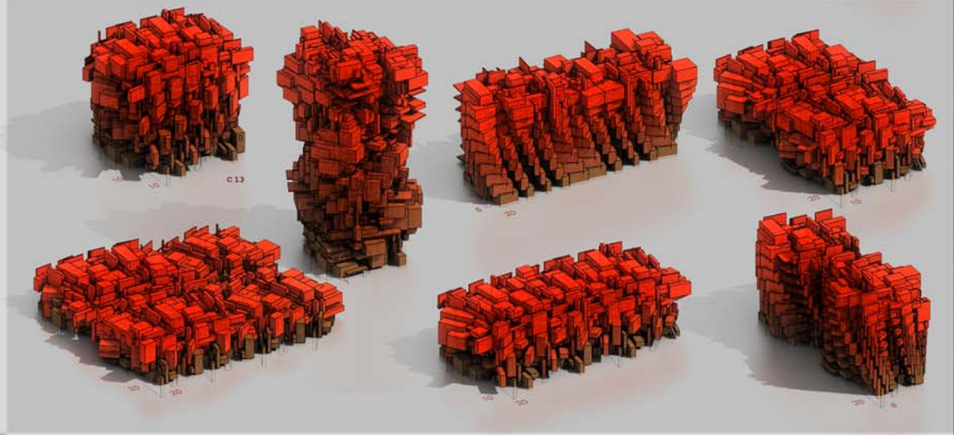
Ridistribuzione su griglia 20x20

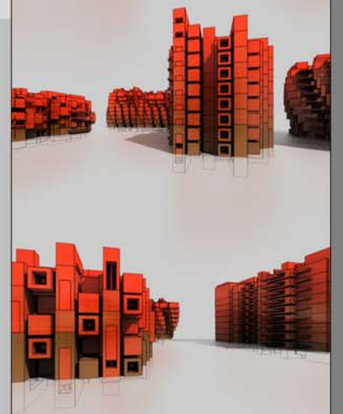


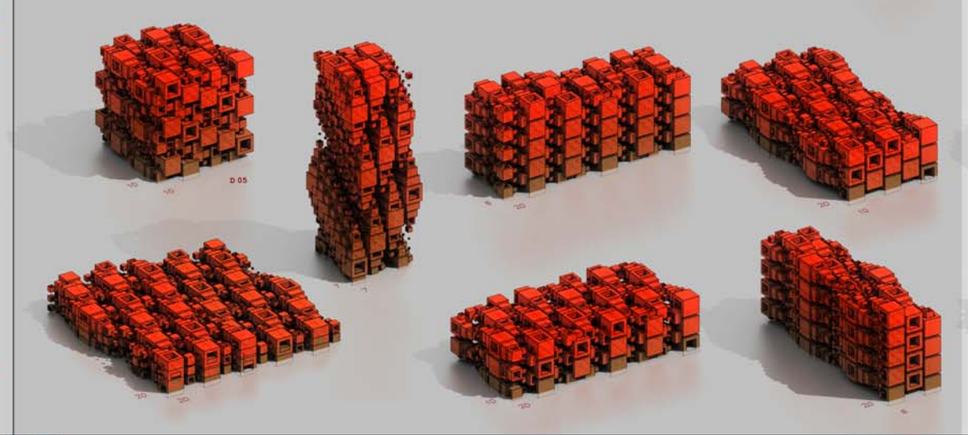


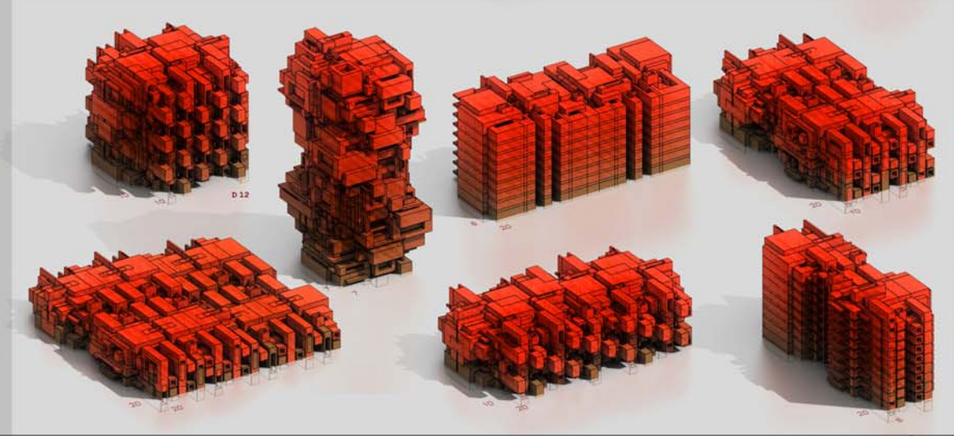






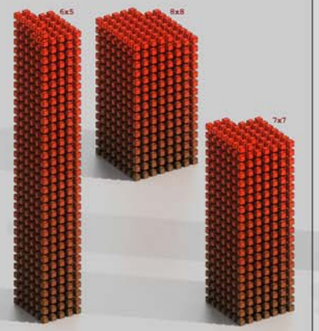


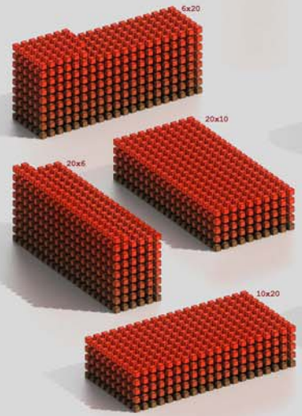


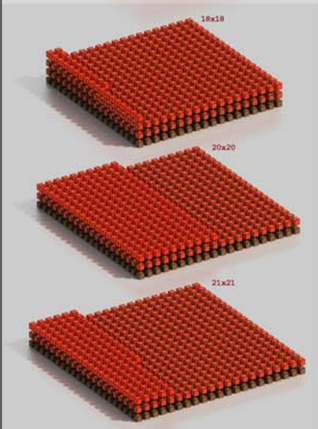


Memorie tipologiche

Sviluppo, mediante ridistribuzione, di alcune soluzioni accomunate nella tavola precedente.
Le soluzioni ricavate sono state ordinate in base a degli schemi che ricordano alcunetipologie edilizie quali la "torre", la "linea" e la "piastra".

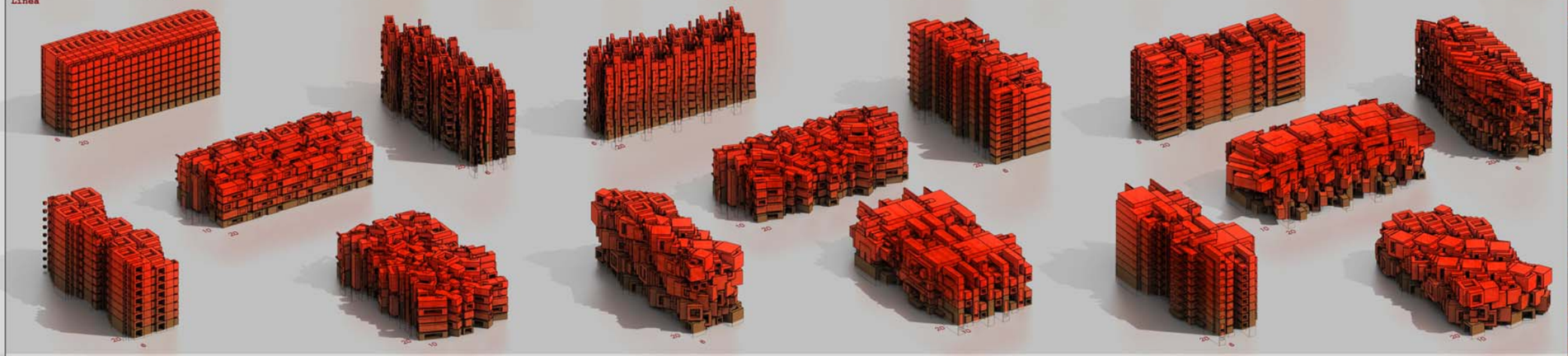




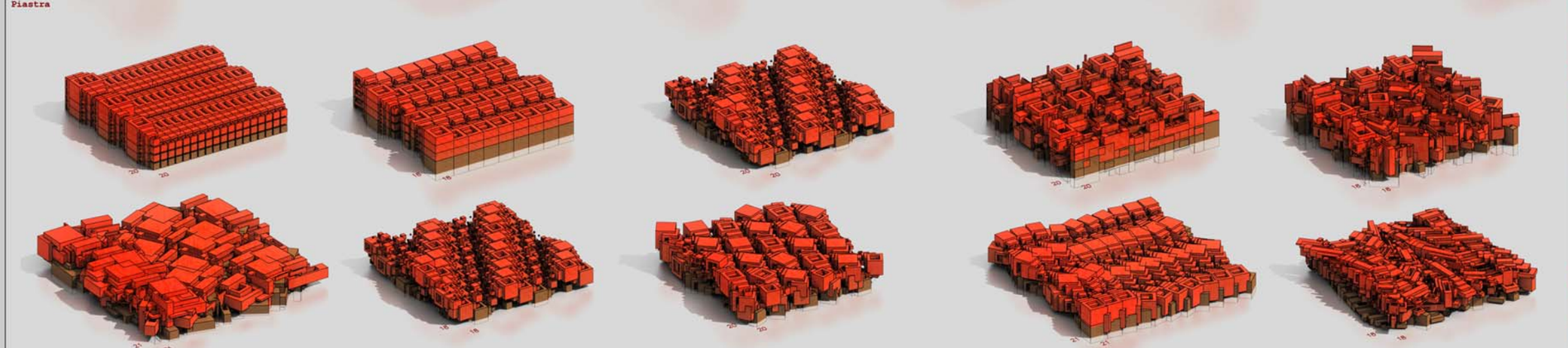


Torre

Linea



Piastra



Università degli Studi di Camerino
Facoltà di Architettura
di Ascoli Piceno ~ a.a. 2004/2005

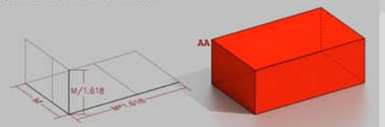
Indagini sperimentali intorno alla forma architettonica

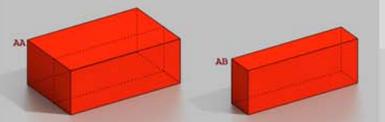
Relatore: Franco Cervellini
Correlatore: Daniele Rossi
Laureando: Filippo Sicuranza

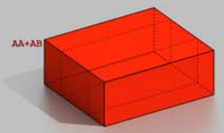
Oggetto ben formato per controllo dimensionale

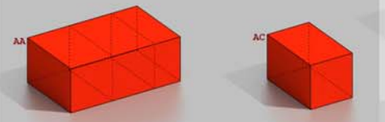
Le variazioni sperimentali fin qui esaminate hanno ruotato principalmente intorno al tema morfologico della ripetizione seguendo varie declinazioni. L'ultima esercitazione sperimentale che viene proposta affronta un tema diverso, di notevole complessità, di possibili ed interessanti sviluppi. Quello della costruzione di un oggetto ben formato a partire da procedimenti "automatici" (fortemente controllati) per calibratura dimensionale.

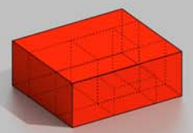
Partendo dalla misura M si ricava l'altezza (M/1.618) e la larghezza (M*1.618) con le quali si costruisce l'oggetto AA.
Dall'oggetto AA si ricava l'oggetto AB avente altezza e larghezza uguali ad AA e profondità pari a M/1,618.
Sempre dall'oggetto AA viene ricavato l'oggetto AC avente altezza e profondità pari ad AA e larghezza pari a M/1.618.
L'oggetto AC viene a sua volta diviso in due oggetti:
- ACA avente larghezza e altezza pari ad AC e profondità pari a M/1.618
- ACB avente larghezza pari ad AC, altezza pari a (M/1.618)/1.618 e profondità pari a M-(M/1.618).



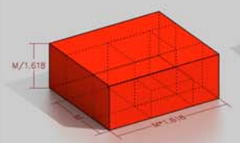




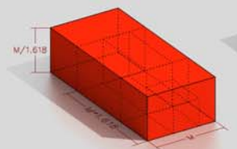




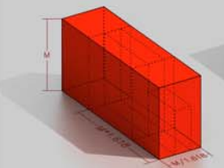
Schema a



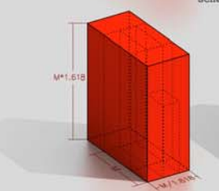
Schema b



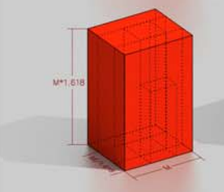
Schema c



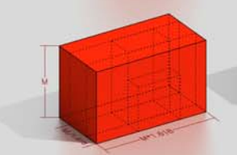
Schema d



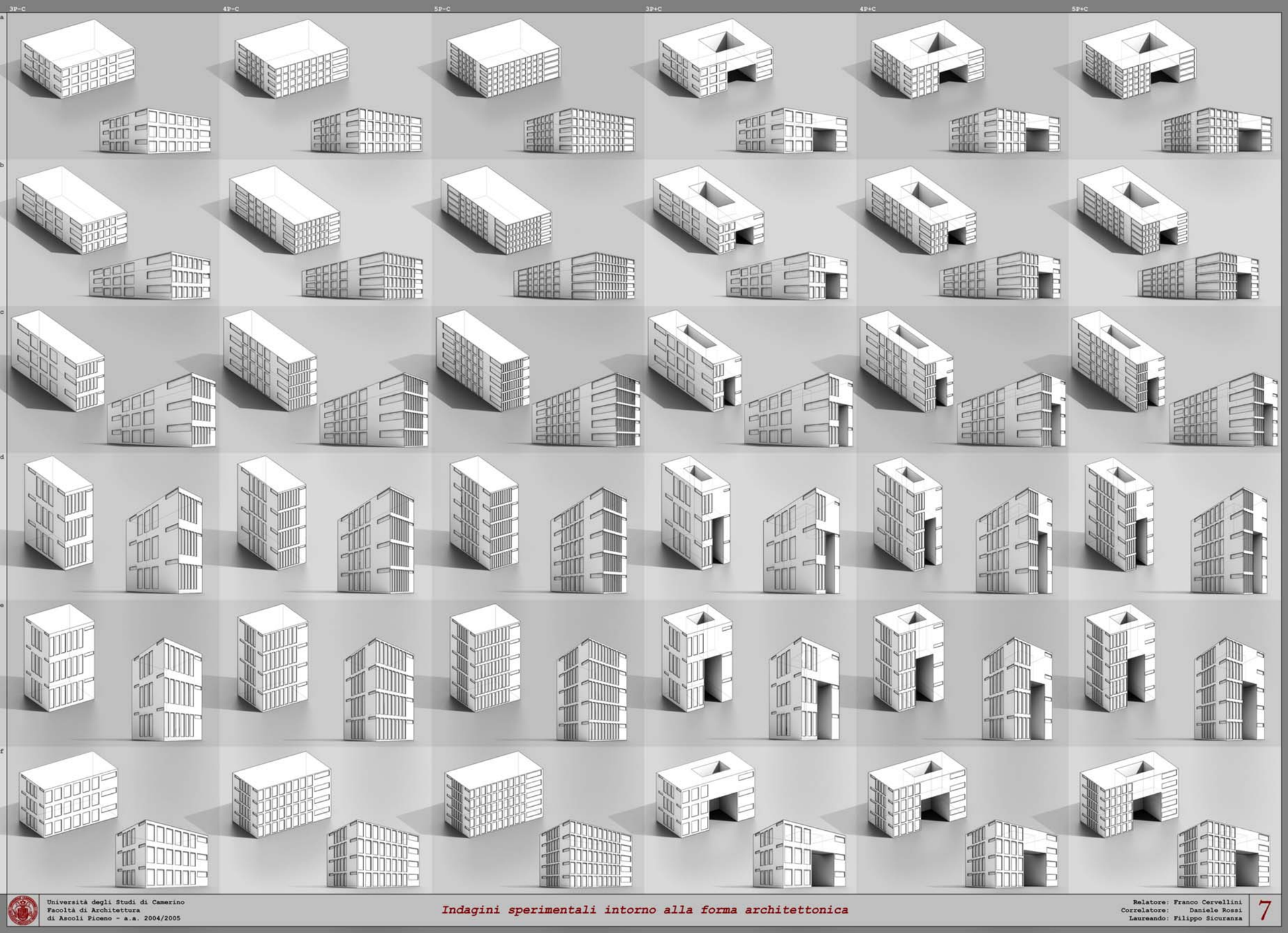
Schema e



Schema f



3P-C 4P-C 5P-C 3P+C 4P+C 5P+C



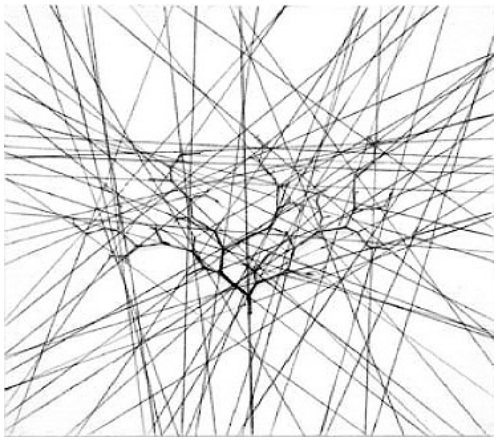
Università degli Studi di Camerino
Facoltà di Architettura
di Ascoli Piceno - a.a. 2004/2005

Indagini sperimentali intorno alla forma architettonica

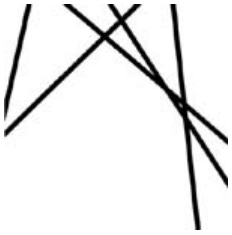
Relatore: Franco Cervellini
Correlatore: Daniele Rossi
Laureando: Filippo Sicuranza

Relazione

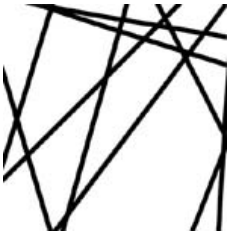
Indice



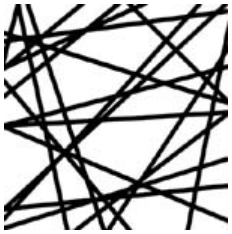
François Morellet: Troquons n°1
Rami e pastelli su plexiglass - 39x39cm - 2005



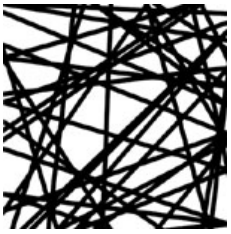
François Morellet:



François Morellet:



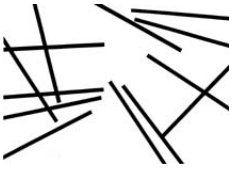
François Morellet:



François Morellet:

Ogni opera: Olio su tela, 140 cm x 140 cm. 1971.
Collezione Manfred Wandel, Stiftung für konkrete Kunst, Reutlingen.

Herman de Vries:
toevalsobjectivering
v 66-59, 1966.



Herman de Vries:
toevalsobjectivering
v 66-60, 1966

In questi ultimi anni la ricerca che si occupa della progettazione unita all'utilizzo dell'informatica è tesa allo studio di nuovi metodi di interazione tra progettista e macchina; si sta lentamente allontanando dalla concezione che abbiamo avuto fino ad oggi. Concezione che vede il computer (inteso sia come hardware, con la sua enorme capacità di elaborazione ed immagazzinamento dei dati, che come software) come solo strumento di disegno e di controllo del progetto. Si stanno sperimentando le modalità e gli strumenti che vedono il progettista posto ad un livello più alto nel processo progettuale; un livello dove egli traccia i principi generali e le regole che devono governare il progetto, li traduce in algoritmi e lascia al computer il compito di svilupparli. Questa metodologia è nota come *Algorithmic design*.

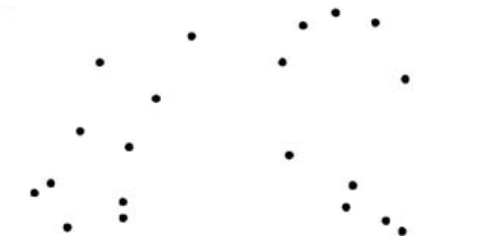
Per comprendere meglio l'*Algorithmic design* bisogna però fare un passo indietro.

Negli anni '60 molti movimenti artistici iniziarono ad abbandonare la pittura, impiegando, in sostituzione, altri tipi di materiali e media artistici. Prese corpo la raffigurazione degli *stati visuali* [1] anziché la produzione di quadri o sculture. L'esecuzione della rappresentazione (*visual score*) poteva avvenire in maniere differenti. L'artista non era in grado di predire il risultato finale; egli era in grado di fissare esclusivamente alcune proprietà e lasciare la definizione delle restanti all'esecutore o al caso. Quando tali proprietà venivano definite e specificate da un algoritmo[2] matematico si entrava nel campo dell'*Algorithmic Art*. Sostanzialmente un semplice algoritmo può essere eseguito manualmente da una persona. Esso può definire un insieme di immagini diverse con assoluta precisione enumerando, una ad una, tutte le variazioni all'interno di un determinato motivo oppure, nel caso il numero delle scelte o delle possibilità fosse troppo elevato, prendere delle decisioni (casuali o basate su altre regole) all'interno di un insieme di possibilità. L'algoritmo, in questo modo, divie-

---

[1] Remko Scha, *Towards an Architecture of Chance,* In: Hans Konstapel, Gerard Rijntjes and Eric Vreedenburgh, *De Onvermijdelijke Culturele Revolutie,* Stichting Maatschappij en Onderneming, 1998, pp. 105-114. [In Olandese]

[2] *"Nella sua definizione più semplice ed intuitiva un algoritmo è una sequenza ordinata di passi semplici che hanno lo scopo di portare a termine un compito più complesso"* - http://it.wikipedia.org/wiki/Algoritmo
In informatica, invece, è definito come: *"Sistema di regole e procedure che conducono alla risoluzione di un problema attraverso un numero finito di operazioni"* - http://www.spysystem.it/glossario/a.htm

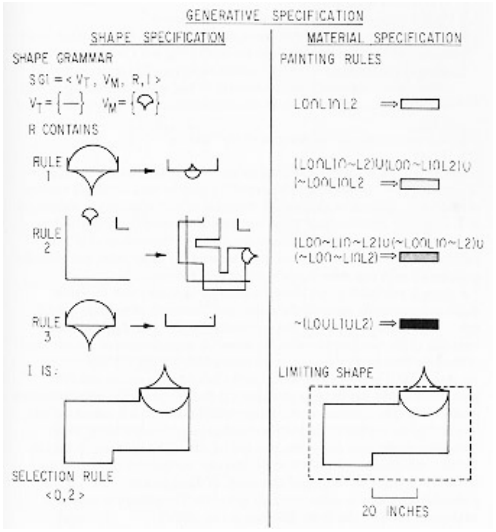


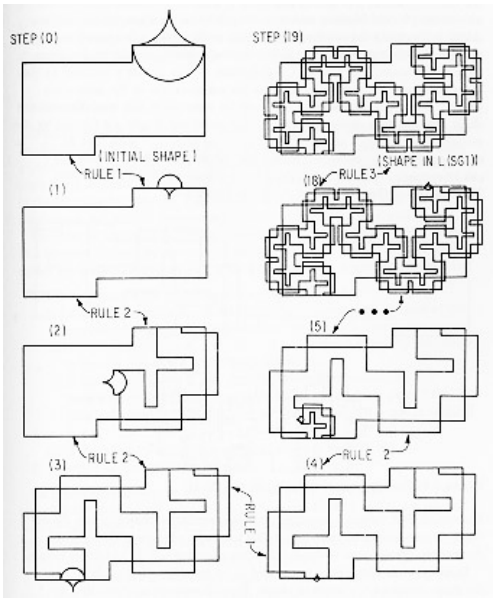
Herman de Vries: random objectivation,

Herman de Vries: random objectivation,

Herman de Vries: random objectivation,

Herman de Vries: random objectivation,





Schemi illustrativi pubblicati su: "*Shape Grammars and the Generative Specification of Painting and Scultpture"*

ne un *meta-artwork*, ovvero una descrizione matematica di una serie di possibili opere d'arte.

Quando le scelte sono operate dall'algoritmo in maniera casuale si ha la *chance art*. Con questi principi venivano realizzate le opere casuali dei neocostruttivisti François Morellet e Herman de Vries i quali impiegavano dei semplici algoritmi per generare delle forme (un quadrato, un cerchio o un segmento) e disporle in maniera casuale in un piano. Tali algoritmi venivano eseguiti "a mano" operando le scelte casuali mediante un tiro di dadi o consultando delle tabelle contenenti numeri arbitrari.

Un prototipo degli algoritmi utilizzati negli anni '60 disponeva delle forme determinate su di un piano scegliendo in maniera random la posizione; un altro prototipo generava casualmente dei poligoni chiusi mediante dei segmenti. Combinando questi due tipi di algoritmi se ne poteva creare un altro che determinava sia la forma che la posizione in maniera casuale. In questo modo era possibile realizzare infinite immagini che rispettavano le regole dell'artista e allo stesso tempo erano sempre diverse.

Al giorno d'oggi è possibile eseguire, con l'ausilio del computer, algoritmi molto più complessi. Un esempio di queste potenzialità lo si è avuto con Artificial [3]. Un software, o meglio, una famiglia di softwares che generano immagini casuali mediante l'impiego di una "grammatica visuale" all'interno della quale sono state definite alcune regole e immagini. L'idea di utilizzare un metodo (grammatica) che ha come principio l'impiego di regole che modificano delle forme di partenza è stata formalizzata e pubblicata nel 1971 da George Stiny e James Gips con il titolo "*Shape Grammars and the Generative Specification of Painting and Scultpture"*[4].

Remko Scha, uno dei creatori di Artificial scrive:

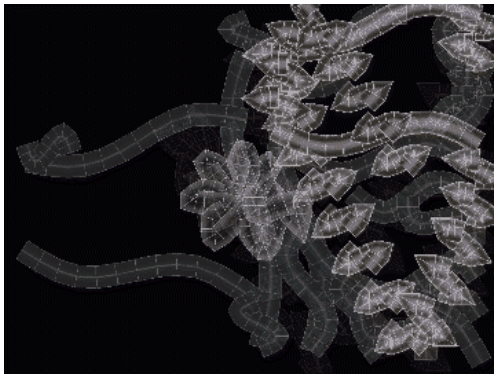
"*Artificial costituisce un contributo costruttivo all'arte autonoma contemporanea dove tutto è creato con la consapevolezza che non ci sono più buone ragioni per creare una cosa invece che un'altra completamente diversa. Artificial esplora questo vicolo cieco.*"[5]

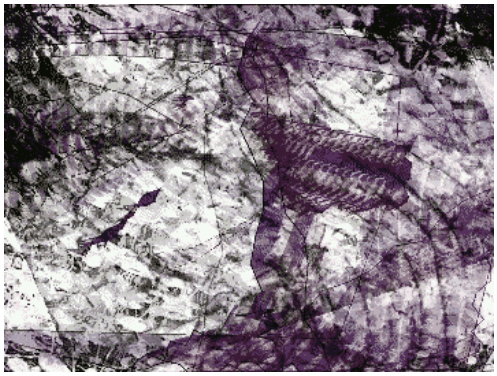
Scha è convinto che in ambito architettonico sia in gioco un problema analogo; in molte situazioni progettuali, imporre il proprio gusto personale risulta improprio anche se inevitabile. Ispirandosi all'approccio di Artificial, egli propone un'*Architecture of Chance* (un'architettura del caso o meglio delle possibilità) dove l'architetto lavora al giusto livello di astra-

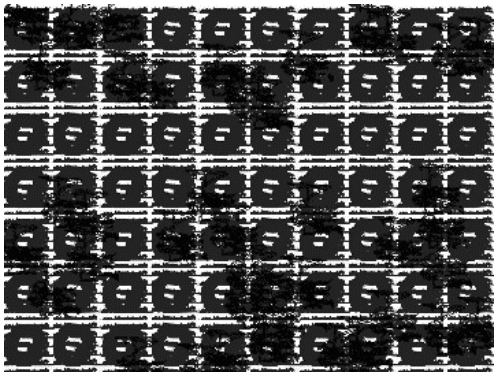
[3] http://iaaa.nl/artificial/main.html

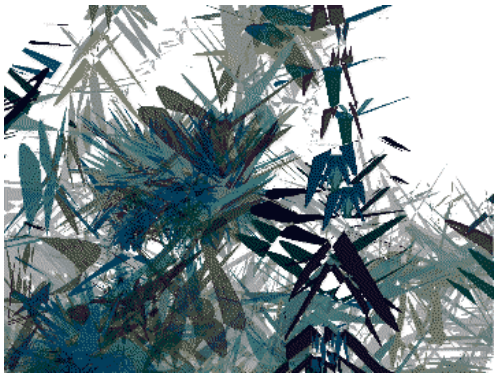
[4] http://www.shapegrammar.org/ifip/ifip1.html

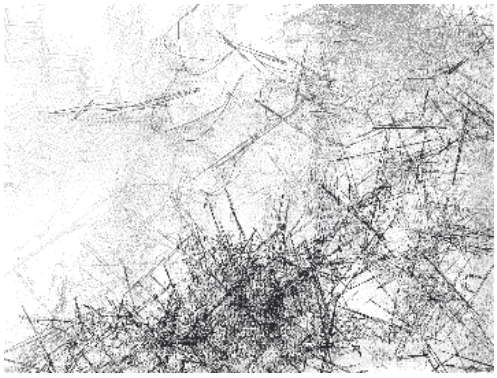
[5] Remko Scha, *Towards an Architecture of Chance,* In: Hans Konstapel, Gerard Rijntjes and Eric Vreedenburgh, *De Onvermijdelijke Culturele Revolutie,* Stichting Maatschappij en Onderneming, 1998, pag. 109. [In Olandese]











Immagini generate da Artificial

zione; dove il progettista non è più interessato ai dettagli espressivi e definisce le "regole del gioco" che determinano quale situazione sia possibile tra le tante. Regole che, come negli scacchi, stabiliscono la grandezza del "campo" in cui si opera, quali pezzi vi sono impiegati e come sia possibile combinarli.

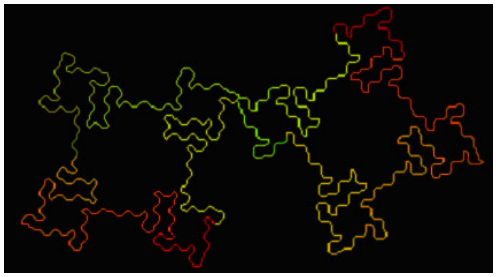
Remko Scha prende in esame anche l'aspetto funzionale. Egli pensa che la funzione di un edificio sia variabile e non vada scelta come punto di partenza fisso sul quale basare il processo progettuale. Il concetto di funzione è stato sostituito dal concetto di potenziale e, per far sì che anche il potenziale possa essere preso in considerazione, bisogna concepire ed impiegare algoritmi molto più complessi rispetto a quelli utilizzati nella *Canche Art.*

La grande sfida nel campo della progettazione architettonica e della pianificazione urbanistica consiste nel riuscire a raggiungere la necessaria flessibilià funzionale mantenendo l'attrattiva data da una forma multipla. Ecco perché il processo costruttivo non dev'essere basato su una eccessiva precisione nella definizione della funzione abitativa. Il progettista basa le proprie scelte sui suoi gusti; gusti che non sono percepiti come una limitazione. Dunque diviene utile suddividere le decisioni progettuali in livelli dove ad ogni livello superiore vengono fissate le condizioni e i limiti per il livello inferiore. Il modo in cui queste condizioni vengono soddisfatte può essere determinato dal caso (chance) o dal committente e riuscire così ad andare al di là dei gusti del progettista.

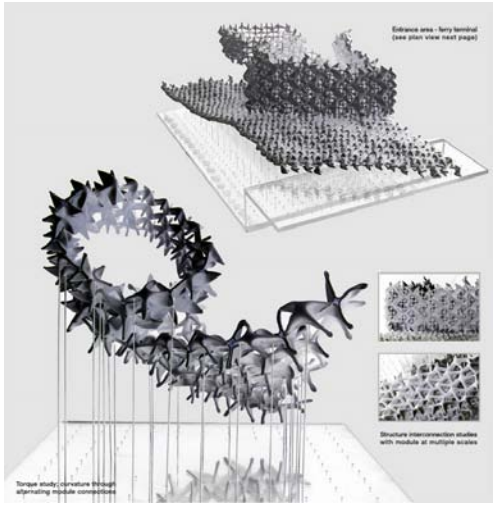
L'idea di una progettazione basata su regole non è nuova. È apparsa implicitamente nei testi teorici sull'architettura scritti in passato e, in questo contesto, è di particolare rilevanza la riscoperta di Vitruvio e l'interpretazione che ne fece Palladio.

Palladio fu il primo importante architetto che progettò secondo una grammatica della forma. Tale grammatica gli permise di produrre un vastissimo numero di variazioni sul tema della villa. George Hersey e Richard Freedman hanno effettuato uno studio sulle regole e sugli elementi compositivi caratteristici delle ville palladiane[6], implementando un software che prende gli elementi e li combina secondo queste regole generando delle ville che sono palladiane a tutti gli effetti ma non sono state progettate da quest'ultimo. Le regole applicate da Palladio e dai suoi contemporanei furono formulate in termini di trasformazioni elementari (traslazione, rotazione e riflessione).

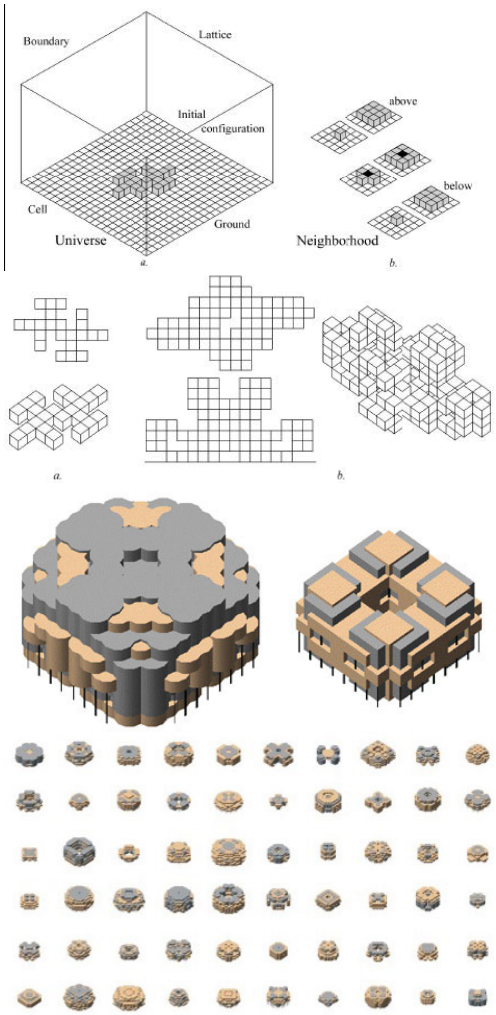
[6] George Hersey e Richard Freedman, *Possible Palladian Villas (Plus a Few Instructively Impossible Ones),* Cambridge, Mass, The MIT Press, 1992.



Schema generato dall'algoritmo L-System



Progetto sviluppato mediante L-System da Michael



Sperimentazione del metodo Cellular Automata applicato all'edilizia - Dr. Stephen Wolfram's, Robert J. Krawczy e Ying-Chun Hsu

In natura possiamo osservare che semplici trasformazioni possono portare a risultati molto complessi.

Per simulare la natura, o meglio, per simulare il processo di crescita biologico, sono stati implementati diversi algoritmi complessi basati su di una semplificazione delle regole che governano la crescita delle forme viventi. È così che sono venuti alla luce i sistemi chiamati **Shape Grammar** (già descritto prima), **L-System** e **Cellular Automata** (automazione cellulare).

"L-System è l'acronimo di Lindenmayer-Systems, dal nome di Aristide Lindenmayer (1925-1989), un biologo olandese che per primo sviluppò la tecnica usata per generare questi frattali. Lo scopo di Lindenmayer era di riprodurre in modo virtuale la crescita di svariati tipi di organismi.

L-System non è perciò un tipo di frattale, ma è un metodo che permette di ritrovare i frattali, anche i più noti, come Koch, Sierpinski, alberi etc., che si possono costruire per altra via, purchè lineare. Il metodo adottato da Lindenmayer è molto suggestivo. Si parte da un disegno iniziale (che può essere, ad esempio, un segmento o anche una poligonale). Questo disegno viene riprodotto al computer usando delle regole ben precise."[7]

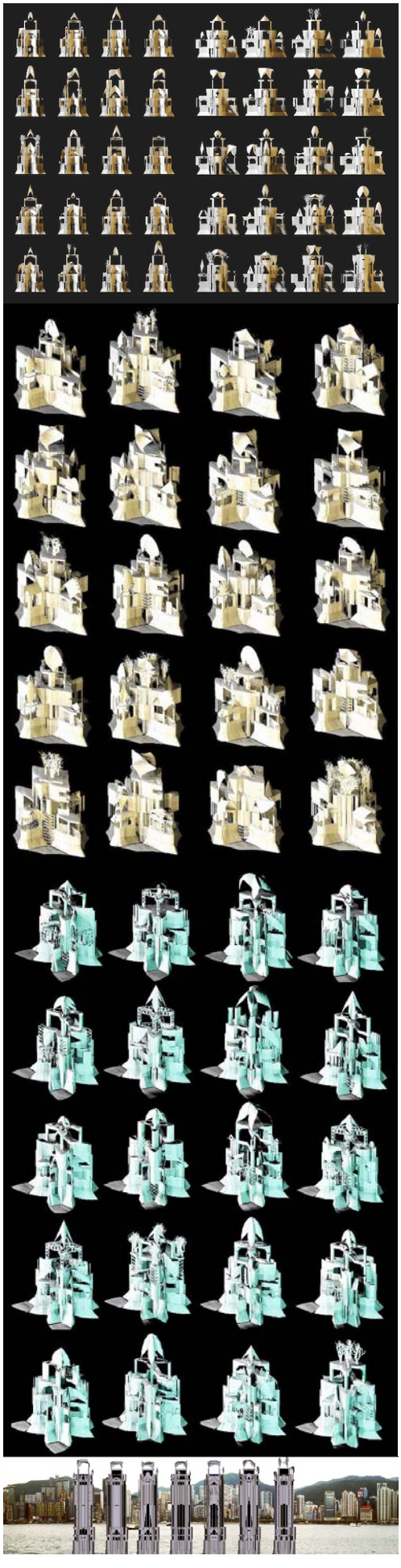
L'architetto Michael Hansmayer sperimenta da tempo l'applicazione del metodo L-System all'architettura, mostrando, nel suo sito[8], una serie di possibili applicazioni ed esempi.

Per definizione Cellular Automata[9] sono dei sistemi dinamici discreti (in spazio e tempo) che operano su di una griglia uniforme e sono caratterizzati da un'interazione locale. Cellular Automata è un metodo di computazione che riesce a simulare un sistema complesso, come il processo di crescita biologica, mediante l'applicazione di semplici regole. Tale concetto, introdotto da John von Neumann (1915) e successivamente sviluppato da Ulam (1962), ha acquistato popolarità quando, nel 1970, Martin Gardner realizzò un gioco che generava motivi grafici bidimensionali. L'universo in cui si sviluppa la vita simulata dal cellular automata è composto da un reticolo in cui ogni quadrato rappresenta una cellula. Ogni cellula ha uno stato, pieno o vuoto. Il processo inizia quando al sistema vengono applicate una serie di regole che determinano la morte, la duplicazione o la nascita delle cellule. Tali regole consistono nel prendere in esame una singola cellula e applicare delle modifiche (nascita, morte o duplicazione) alle cellule attorno ad essa. Una volta applicate le modifiche si passa alla cellula successiva. Robert J. Krawczyk ha prodotto una serie di esempi (sia teorici che software) che mostrano l'interpre-

[7] http://www.webfract.it/FRATTALI/lsystem.htm

[8] http://www.mh-portfolio.com/indexF.html

[9] http://www.iit.edu/~krawczyk/rjkga02.pdf - rjkga03.pdf - rkecad02.pdf



l'interpretazione e l'applicazione, dal punto di vista architettonico, dell'automazione cellulare.

La Generative Art utilizza ampiamente gli algoritmi precedentemente descritti.

"*Scopo di questo tipo di ricerca artistica è quello di creare strutture complesse a partire da una combinazione di elementi formali semplici.*"

"Attraverso la "programmazione morfogenetica" dell'arte generativa si arriva a creare da forme elementari strutture molto complesse come volumetrie architettoniche[…]Possiamo parlare di arte generativa nei termini elaborati dalla biogenetica. Come la biologia ha scoperto che lo sviluppo dell'organismo umano - il processo che si definisce epigenesi - è concepita come lo sviluppo di un programma genetico contenuto nel Dna, allo stesso modo possiamo considerare l'architettura generativa come lo sviluppo nello spazio di volumi che nascono dalla composizione e ricombinazione di elementi formali elementari, che funzionano come codice."[10]

Uno dei massimi esponenti italiani di tale Arte applicata all'architettura e al design è il prof. Celestino Soddu del Politecnico di Milano. Egli, assieme al Dipartimento di Scienze del Territorio del Politecnico di Milano, ha elaborato un software capace di "progettare geneticamente" forme architettoniche. A tale proposito Soddu scrive:

"Il design generativo e' l'idea espressa come codice genetico degli oggetti artificiali. Il progetto generativo e' un software-idea che opera generando eventi tridimensionali unici e non ripetibili come espressione plurima e aperta dell'idea generante, visioni immaginifiche di un mondo nuovo. Quest'idea / processo opera un'espansione imprevedibile, sorprendente e pressoche' infinita della creativita'. In tutto cio' il computer e' solo uno strumento che memorizza ed esegue. Cio' apre una nuova era nel design e nella produzione industriale: la sfida di una nuova naturalita' dell'oggetto industriale come evento unico ed irripetibile, specchio dell'unicita' ed irripetibilita' dell'uomo e della natura. Ancora una volta l'uomo emula la natura, come e' proprio del fare arte. Argenia e' il termine che ho coniato per questo codice genetico dell'artificiale che, come avviene in natura con il DNA, identifica non solo un oggetto ma una specie di oggetti. Il design industriale non sara' piu' l'idea e la realizzazione di un solo oggetto ma l'idea di una specie di oggetti e la sua generazione industriale.[…] Oggi queste Argenie sono diventate estremamente complesse e direttamente operative in quanto interfacciabili con i sistemi produttivi. Cio' ha consentito di avviare prime sperimentazioni pratiche, come la realizzazione di libri con copertine tutte diverse, di gioielli unici ed irripeti-

[10] http://www.mediamente.rai.it/home/tv2rete/mm9899/99030105/n990304.htm

Alcuni esempi di Architetture generative

bili, di scenari di spazi sempre diversi che si trasformano incessantemente rincorrendo il possibile, di oggetti che, sebbene generati con il computer, ritrovano l'unicita' irripetibile dell'evento artistico e di performance generative e interattive tra spazi, poesie, musica e danza." [11].

[11] http://www.celestinosoddu.com/design/GA_soddu_i.htm

Inizialmente i programmi per il disegno assistito al computer vengono concepiti come strumenti di disegno emulanti gli strumenti reali. Tali software nascono come una sorta di super tecnigrafo che, mediante equazioni, permette la realizzazione di disegni più o meno complessi; una linea tracciata con uno di questi software è soltanto un'equazione rappresentante un segmento.

La loro evoluzione è dovuta all'invenzione (con relativo inserimento e connessione) di nuovi strumenti e concetti informatici:

- i database relazionali;
- la parametrizzazione;
- gli script.

Con l'evoluzione dei database[12] in database relazionali[13] si è avuta la possibilità di associare ad ogni elemento presente nel disegno una serie di dati (testuali e numerici). In questo modo è stato possibile realizzare disegni (ad es. piante di edifici) con linee che, una volta clickate, restituiscono informazioni quali: funzione dell'elemento, materiale, dimensioni, relazioni con gli altri elementi etc. Molti software sono in grado di generare un albero delle proprietà mostrando con un'unica immagine le relazioni dirette e indirette tra tutti gli elementi appartenentei al progetto.

Un ulteriore passo in avanti è stato fatto quando, agli elementi, si è aggiunta la parametrizzazione. Tale funzione ha permesso di lavorare su elementi grafici con associati dei valori che, se variati, comportano la modifica in tempo reale della forma (e di altre proprietà visibili dell'oggetto). Il modello (inteso come l'insieme degli elementi costituenti un progetto di lavoro), inizialmente definito soltanto da coordinate di punti

---

[12] "*Un database non e' altro che una specie di "contenitore" che ci permette di gestire grossi quantitativi di informazioni simili in maniera ordinata e (si spera) più semplice e veloce che con grossi libroni cartacei o documenti di tipo foglio di calcolo o testo*" - *http://www.harrrdito.it/corsi/sql_002.asp*

[13] Nel modello relazionale i dati sono organizzati in tabelle che rappresentano sia le entita', sia le relazioni tra di esse: esistono quindi tabelle di entita' e tabelle di relazioni. Nel modello relazionale, a differenza dei precedenti, non c' e' alcun meccanismo esplicito per rappresentare i legami logici tra i diversi tipi di record che non sia la relazione. La modifica di un dato o di un legame comporta la manipolazione di un solo record di una tabella.Nel modello relazionale, a differenza dei precedenti, si realizza l' indipendenza logica, e' possibile modificare le strutture senza dover modificare i programmi.
http://www.archesis.it/white/oracle/intro_ora.htm

ed equazioni di linee, si è trasformato in un entità costituita da una parte grafica e da una parte informatica unite da molteplici connessioni.

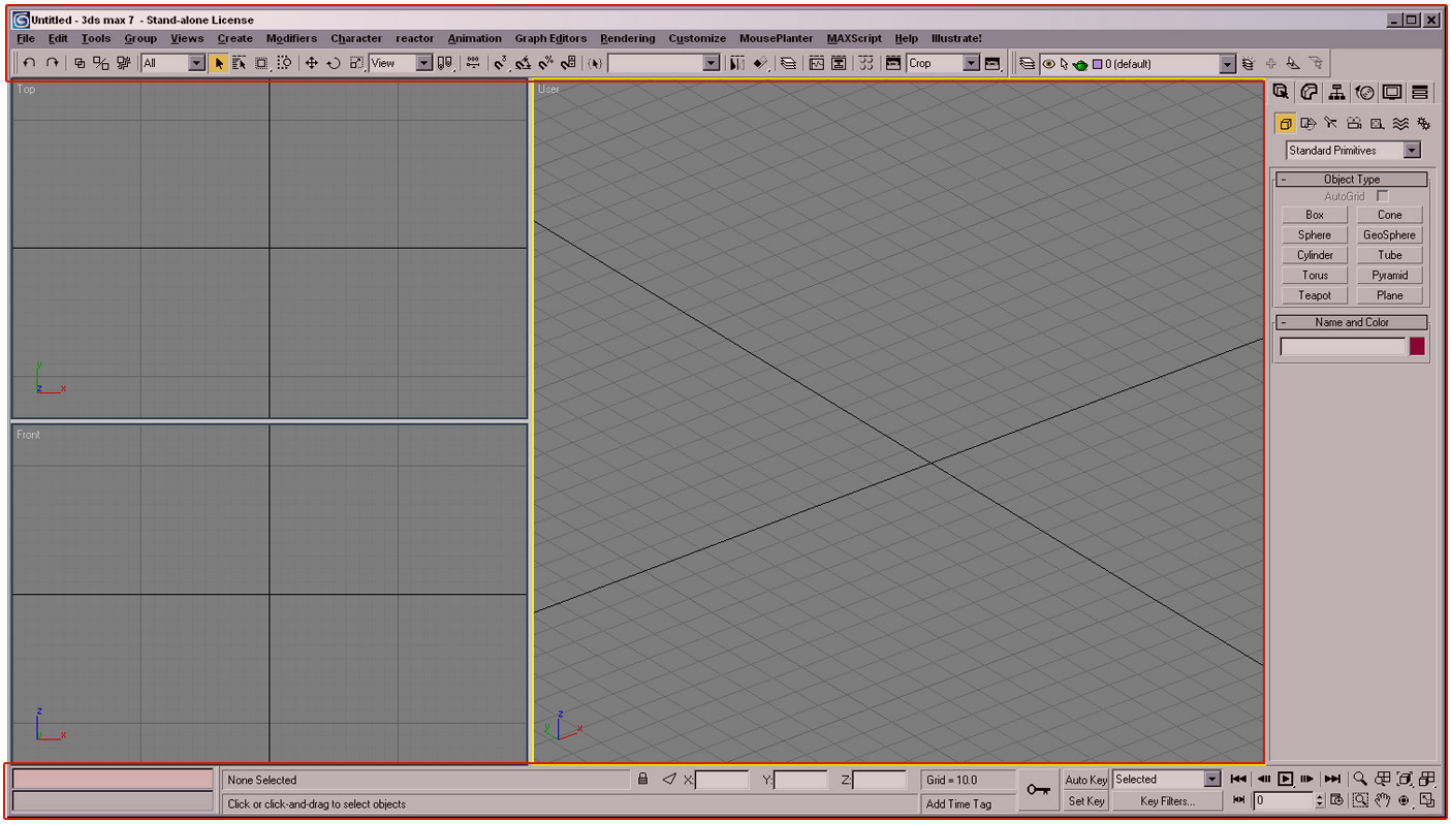
Si è così passati dalla concezione bidimensionale e lineare del disegno ad una struttura più complessa costruita su livelli separati e, a volte, sovrapposti.

Tutti i più recenti software che permettono la creazione e la gestione di modelli tridimensionali, sia *poligonali* sia *NURBS*, hanno, al loro interno, un linguaggio di programmazione noto come linguaggio di scripting. Tale linguaggio permette la gestione del software mediante il solo utilizzo di linee di codice; grazie ai linguaggi di scripting si ha la possibilità di governare un software senza l'utilizzo degli input del mouse, della tastiera e della loro interazione con l'interfaccia grafica.

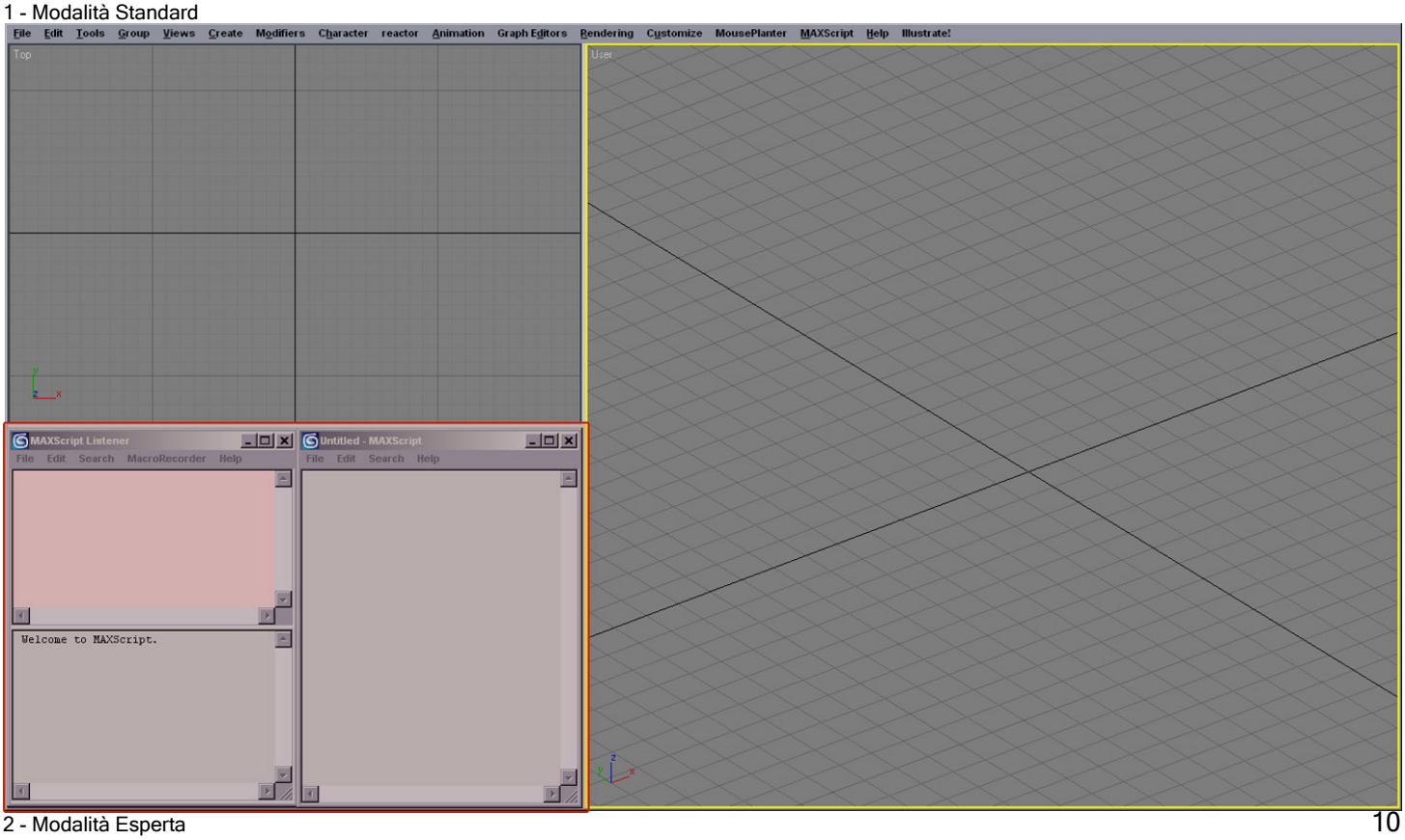
Mediante gli script è possibile andare oltre l'interfaccia e gli strumenti ad essa correlati. Non si è più limitati dalla modifica "meccanica" delle entità mediante il click del mouse, ma l'accesso alle proprietà e alle entità appartenenti al disegno avviene numericamente. Se da un lato si ha una complicazione nell'interazione con il software, dall'altro aumentano le potenzialità riguardanti la generazione e la modifica delle entità. Inoltre, grazie allo scripting, è possibile rimodellare ed estendere l'interfaccia aggiungendo nuove funzionalità e nuovi strumenti[14].

[14] Vedere Appendice B – Cenni sugli elementi delle interfacce grafiche

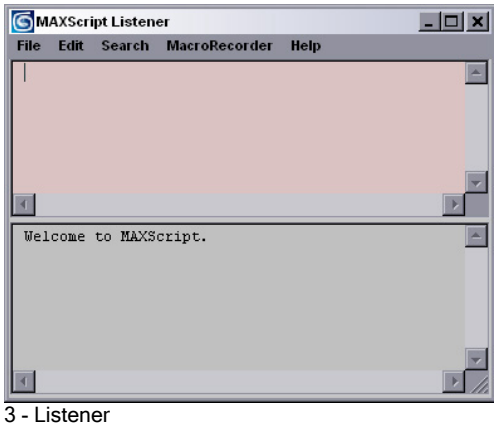
Maxscript è il nome del linguaggio di scripting presente in 3d Studio Max (3dsMax). Prima di andare ad analizzare ed esaminare alcuni aspetti operativi del Maxscript è però necessario presentare la GUI di 3dsMax e osservare come si trasformi in base alla modalità operativa che si sceglie di utilizzare. Ecco come si presenta 3dsMax all'apertura [Fig.1]. In rosso



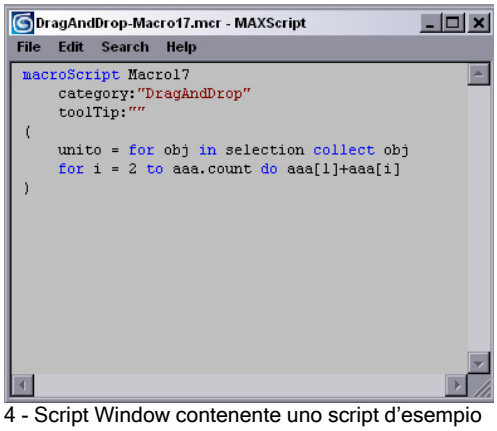
1 - Modalità Standard



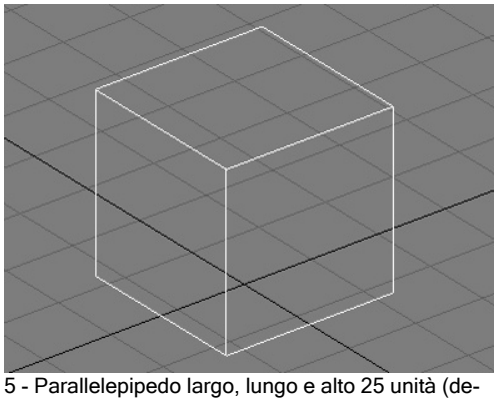
2 - Modalità Esperta

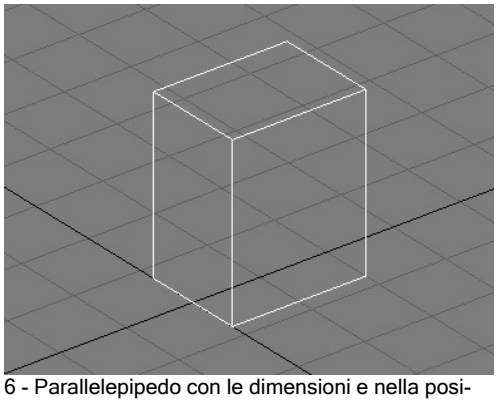


3 - Listener



4 - Script Window contenente uno script d'esempio


5 - Parallelepipedo largo, lungo e alto 25 unità (default) e posizionato sull'origine del modello [0,0,0]


6 - Parallelepipedo con le dimensioni e nella posi-

è evidenziata l'area dell'interfaccia che permette l'utilizzo del software mediante il mouse e la sua interazione con le icone e i menù a cascata. Area che scompare quando si passa in Modalità Esperta [Fig.2]. In tale modalità rimangono visibili soltanto i menù a tendina e l'interazione con 3dsMax è resa possibile solo mediante i menù a tendina, le scorciatoie da tastiera e due o più finestre all'interno delle quali è possibile inserire dei comandi (in linguaggio Maxscript).

Le finestre sono di due tipi differenti:

il **Listener** è diviso in due parti; un'area rosa all'interno della quale è possibile digitare singole righe di codice ed un'area grigia (contrassegnata all'apertura di 3dsMax dal testo *"Welcome to MaxScript"*) dove viene visualizzato l'esito della riga di codice digitata. L'esito può essere il risutato di un'operazione matematica, la conferma della corretta esecuzione del comando oppure un messaggio d'errore che segnala la digitazione non corretta del codice. Il listener è presente anche nella modalità Standard ed è ubicato nell'angolo in basso a sinistra del porgramma;

Script Window è una finestra di testo utilizzata per scrivere script più complessi (su più rughe), visualizzarli, salvarli e mandarli in esecuzione.

Grazie al MaxScript è possibile "comandare" 3dsMax mediante il solo inserimento di linee di codice; senza l'utilizzo dell'input del mouse e dell'interazione con l'interfaccia si riesce a creare/importare delle geometrie, trasformarle, cambiargli proprietà, definirne i materiali.

Per creare una primitiva mediante MaxScript si può inserire nel Listener la riga:

```
box()
```

il risultato sarà la creazione di un parallelepipedo le cui proprietà (lunghezza, larghezza, altezza, posizione etc.) non essendo state specificate, vengono scelte da 3dsMax in base a dei valori di default.

Se invece specifichiamo le proprietà mediante la riga:

```
box width:20 length:15 height:25 pos:[10,0,0]
```

avremo un parallelepipedo largo 20 lungo 15 alto 25 inserito a 10 unità di distanza dall'origine sull'asse x.

In questo modo è possibile creare diverse primitive (sfere, coni, cilindri). Ogni primitiva in 3dsMax ha delle proprietà che dipendono dalla tipologia della primitiva e che possono essere specificate o meno. Selezionando il cubo appena creato e digitando nel listener la riga:

```
showproperties $
```

vedremo comparire nella porzione di finestra sottostante, quella deputata alla visualizzazione dell'esito della riga immessa:

```
.height : float
.length : float
.lengthsegs : integer
.width : float
.widthsegs : integer
.mapCoords : boolean
.heightsegs : integer
```

la prima parte della riga (ad esempio .height) indica la proprietà della geometria mentre la seconda, quella a destra dei due punti (:) indica il tipo di valore che è possibile specificare. Per tipo di valore si intende un numero che, nel caso del *float* indica una quantità numerica decimarle (ad es. 1.5 - 2.015 - 0.314 etc.). L'altra tipologia *integer* indica un valore intero (ad es. 1 - 5 - 8 - 554) mentre la tipologia *boolean* ammette solo due modalità, true o false.

Una volta create (o importate), le geometrie, possono essere trasformate. Sempre mediante righe di codice è possibile ruotare, cambiare scala o spostare una o più geometrie. Queste trasformazioni vengono eseguite attorno ad un punto base chiamato pivot. Selezionando il cubo creato precedentemente ed inserendo, sempre nel listener, la riga:

```
$.scale = [1,1,1.5]
```

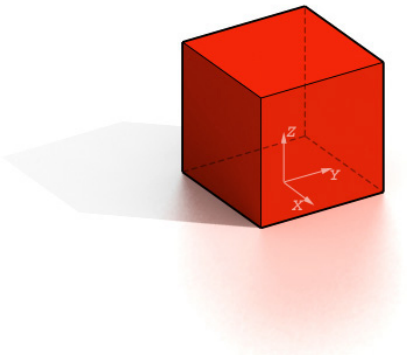
si otterrà l'allungamento del box lungo l'asse Z; la proprietà Scale accetta un valore chiamato point3, ovvero formato da 3 valori racchiusi in una parentesi quadra e separati da virgole [x,y,z], che indica la variazione di scala lungo i 3 assi. In questo caso il valore [1,1,1.5] indica una variazione di scala pari a 1 (scala invariata) lungo gli assi X e Y e una variazione di 1.5 (ingrandimento della metà) lungo l'asse Z. Le variazioni di scala lungo i 3 assi sono indipendenti e possono essere sommate.

Gli stessi principi si applicano alla rotazione dove i valori di rotazione vengono espressi in Eulerangles nel formato (eulerangles x y z) dove ogni valore indica la rotazione attorno ad un asse del pivot. Quindi scrivendo il codice:

```
$.rotation = (eulerangles 0 15 0)
```

otterremo la rotazione del box di 15 gradi attorno all'asse Y.

Come nel cambiamento di scala anche nella rotazione è possibile sommare gli effetti.



7 - Pivot di un box

Inserendo due righe di codice, una che scala e una che ruota, nella script window e mandando in esecuzione lo script otterremo la somma delle due trasformazioni:

```
$.scale = [1,1,1.5]
$.rotation = (eulerangles 0 15 0)
```

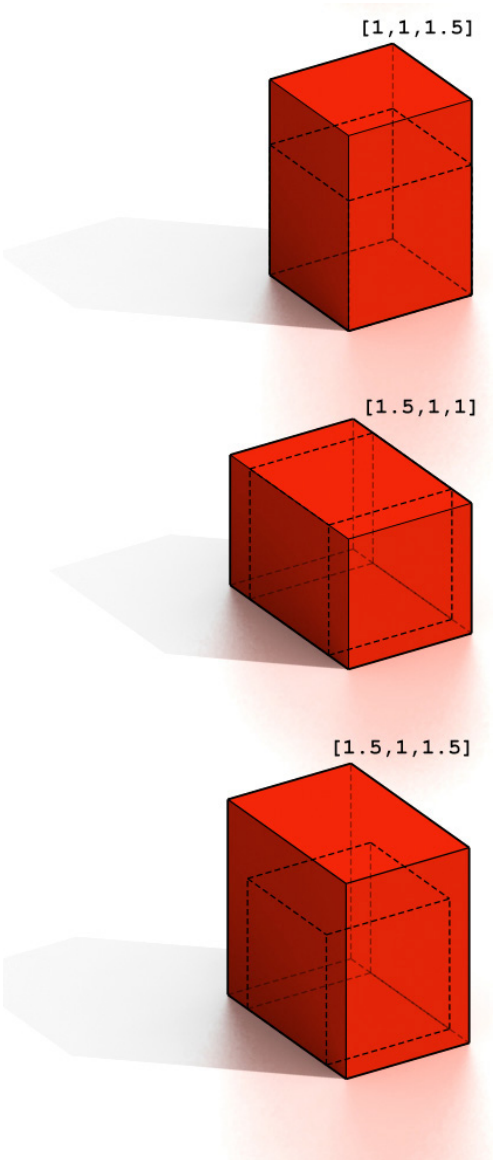
Fino ad ora abbiamo trattato operazioni di creazione e trasformazione a riga singola che se effettuate nel modo tradizionale, ovvero mediante l'input del mouse, risulterebbero sicuramente più pratiche e veloci. Quello che però ci interessa è la possibilità di effettuare queste operazioni in maniera ripetitiva, o meglio, ciclicamente e applicando delle semplici regole matematiche.

Prendiamo ad esempio queste tre righe di codice:

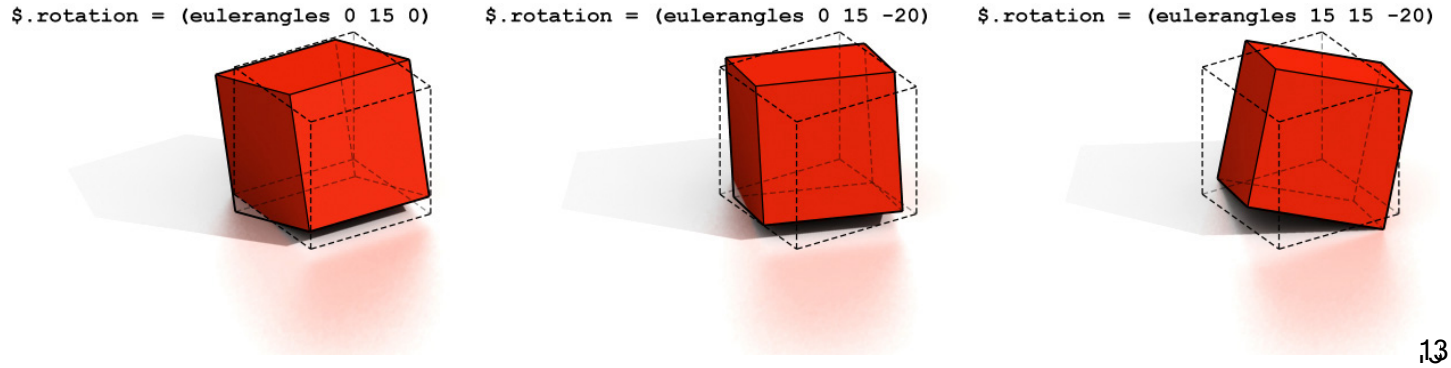
```
box width:60 length:60 height:60 pos:[0,0,0]
box width:60 length:60 height:60 pos:[0,150,0]
box width:60 length:60 height:60 pos:[0,300,0]
```

Queste righe creano tre cubi di lato 60 unità, ognuno distante dall'altro 90 unità (150-60). Otterremo lo stesso risultato scrivendo:

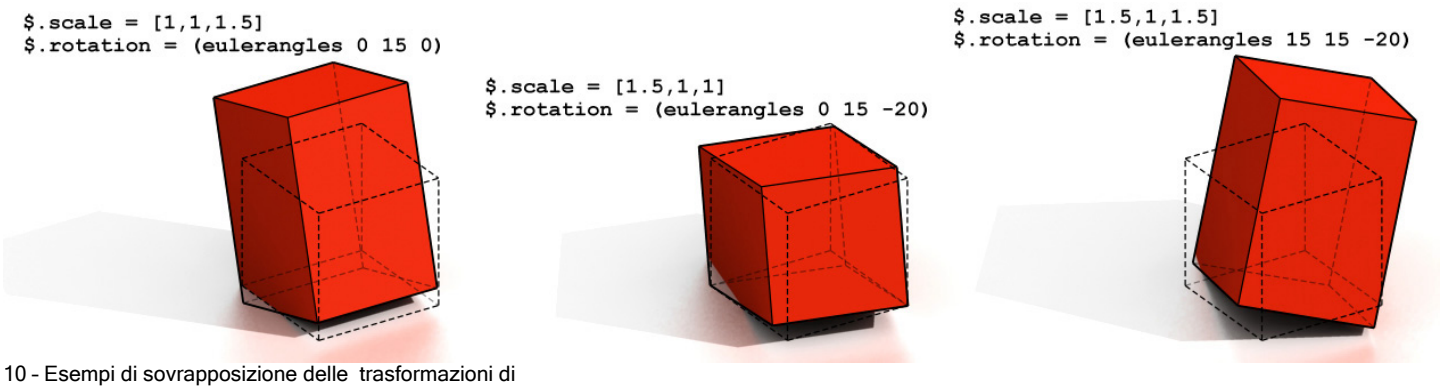
```
for i = 0 to 2 do (
box width:60 length:60 height:60 pos:[0,i*150,0]
)
```



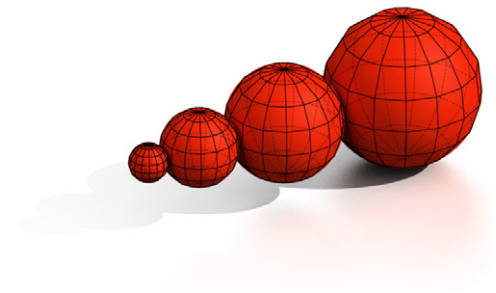
8 - Esempi di cambiamento di scala con diversi pa-



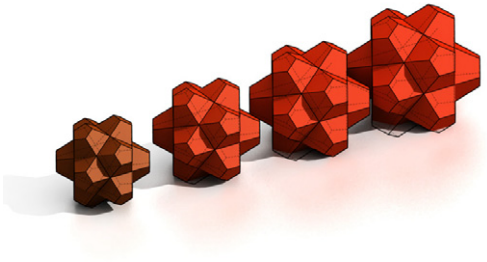
9 - Esempi di rotazione con diversi parametri



10 - Esempi di sovrapposizione delle trasformazioni di scala e rotazione



12 - Sfere create mediante il ciclo for



13 - Variazione posizione, forma e colore mediante il ciclo for

Il codice appena scritto prende il nome di **ciclo for.**

Il *ciclo for* permette di ripetere uno o più comandi posti tra le parentesi. L'indice `i` è un contatore che permette di specificare il numero di ripetizioni da effettuare. La stringa estrapolata dal codice:

```
i = 0 to 2
```

Indica che il *ciclo for* verrà ripetuto 3 volte (da 0 a 2) e l'indice `i` assumerà il valore relativo alla ripetizione (0, 1 e 2). Ecco che alla prima ripetizione, ovvero quando `i` sarà uguale a 0 la riga che crea il box sarà:

```
box width:60 length:60 height:60 pos:[0,0*150,0]
```

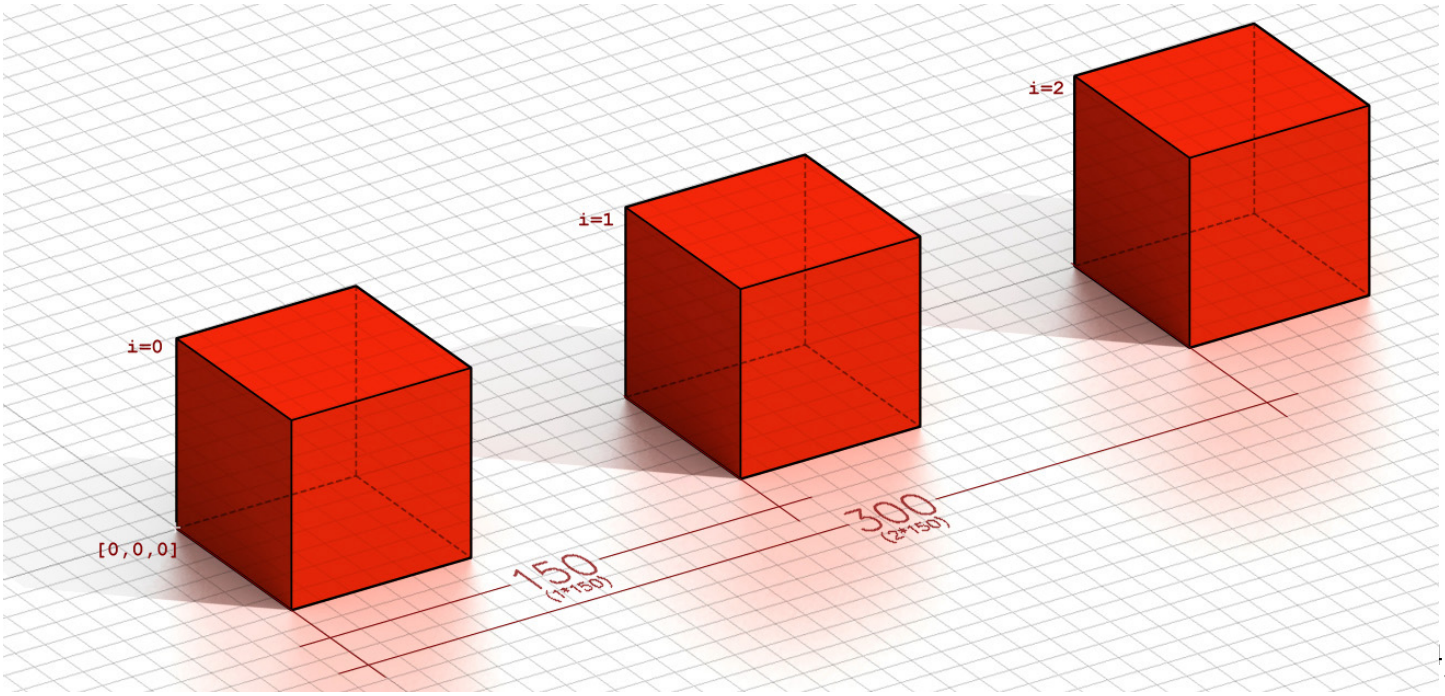
0 moltiplicato 150 è uguale a 0 e quindi il box verrà posto sull'origine alle coordinate [0,0,0]. Alla ripetizione successiva l'indice `i` assumerà il valore 1 e quindi la riga che verrà eseguita sarà:

```
box width:60 length:60 height:60 pos:[0,1*150,0]
```

Il box che verrà creato sarà posto alle coordinate [0,150,0] e così via.

Ecco che grazie al *ciclo for* è possibile creare o modificare una serie di geometrie mediante semplici regole matematiche. Scrivendo:

```
for i = 1 to 4 do (
    sphere radius:i pos:[0,i^2,0]
```



11 - Schema esplicativo del ciclo for

```
)
```

otterremo quattro sfere dove la prima sarà posta con il centro sull'origine e le altre avranno raggio e distanza proporzionali all'indice `i` dove la posizione sarà uguale al raggio elevato al quadrato (`i^2`).

Quest'altro esempio:

```
for i = 1 to 4 do (
    hedra wirecolor:[251-(150/i),40,10] ra-
dius:35*i pos:[0,(43*2*i),37]
)
```

è simile a quello delle sfere ma, oltre ad agire sul raggio e sulla posizione della primitiva creata, varia il colore (proprietà *wirecolor*) assegnando, mediante un valore point3 nel formato [R,G,B], ovvero le 3 componenti Red, Green e Blue, un colore sempre diverso dipendente dal contatore `i`.

Prima di proseguire è necessario definire il concetto di **variabile**.

Le variabili sono stringhe di testo all'interno delle quali vengono memorizzati dei dati; come nelle banali equazioni si può scrivere:

```
a = 5
b = 2
c = a+b
```

`a` e `b` sono delle variabili a cui sono stati assegnati dei valori; `c` è allo stesso modo una variabile all'interno della quale viene memorizzata la somma di `a+b`, ovvero 7. Oltre a memorizzare valori numerici possiamo memorizzare anche nomi di entità. Scrivendo:

```
cubo = $
cubo.scale =  [1,1.5,1]
```

la variabile cubo assume il valore della selezione e scrivere:

```
cubo.scale =  [1,1.5,1]
```

equivale a scrivere:

```
$.scale =  [1,1.5,1]
```

Come la creazione anche le trasformazioni possono essere eseguite mediante il *ciclo for*. Il seguente script:

```
incrsx = 1
incrrx = 0
incrrz = 0
for obj in selection do (
    obj.scale.x = incrsx
    incrsx += 0.5
    obj.rotation.z_rotation = incrrz
    incrrz += -3
    obj.rotation.x_rotation = incrrx
    incrrx += -2.5
)
```

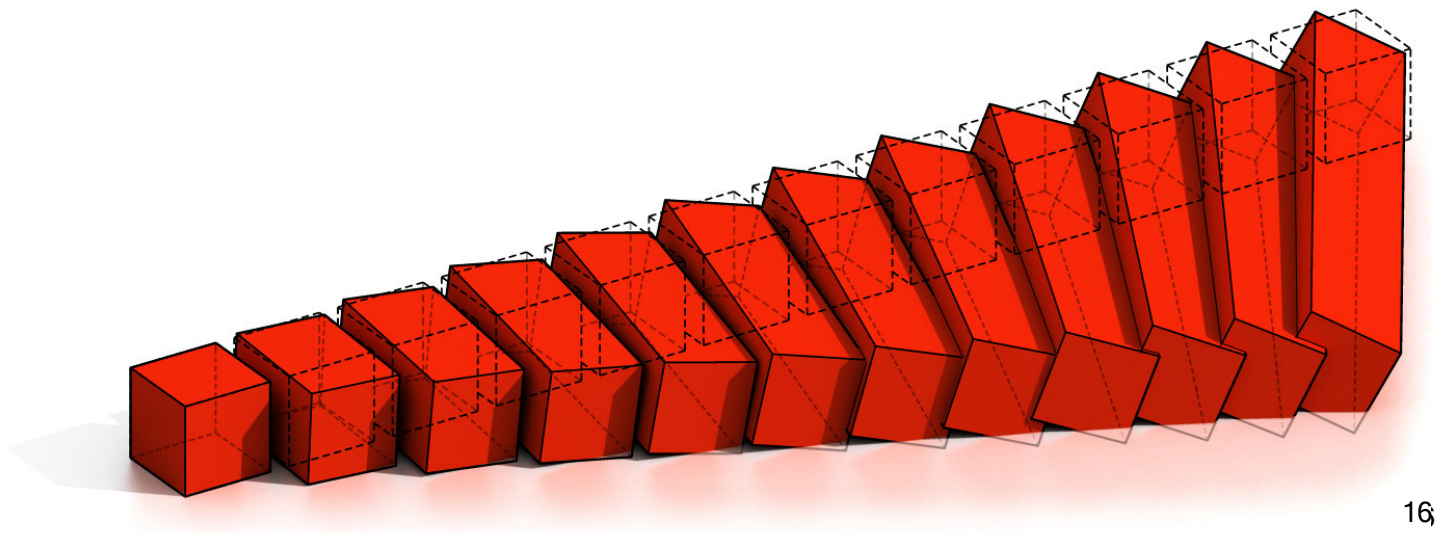
applica una serie di trasformazioni incrementali ad una serie di cubi precedentemente creati. Prima di analizzare le trasformazioni applicate e il loro codice bisogna però porre l'attenzione sul *ciclo for*. In questo esempio il contatore `i`, precedentemente illustrato non è presente. Al suo posto c'è la stringa:

```
obj in selection
```

tale stringa indica che, ad ogni ciclo ripetuto per quanti oggetti sono presenti nella selezione (`selection`), la variabile `obj` deve assumere il valore di una delle entità. Ecco che la riga:

```
obj.scale.x = incrsx
```

applica una trasformazione di scala lungo x ad ogni oggetto presente nella selezione. Il valore che quantifica la trasformazione è memorizzato nella variabile `incrsx.`



14 - Trasformazioni incrementali applicati a dei box ripetuti in serie (tratteggiati)

Il valore della variabile `incrsx` (come anche `incrrx` e `incrrz`) ad ogni ripetizione del ciclo viene cincrementata mediante la riga:

```
incrsx += 0.5
```

Un esempio un po' più complesso è rappresentato da due *cicli for* uno all'interno di un altro:

```
Cubo = box width:30 length:30 height:30
colonne = 10
righe = 7
for y = 0 to colonne do (
    for x = 0 to righe do (
         obj = copy Cubo
         obj.pos = [60*x,60*y,0]
    )
)
```

Tale script crea un cubo di lato 30 e lo memorizza all'interno della variabile `Cubo`; assegna alle variabili `colonne` e `righe` due valori numerici (dai quali dipenderà il numero di ripetizione dei due cicli); utilizza il *ciclo for* più interno:

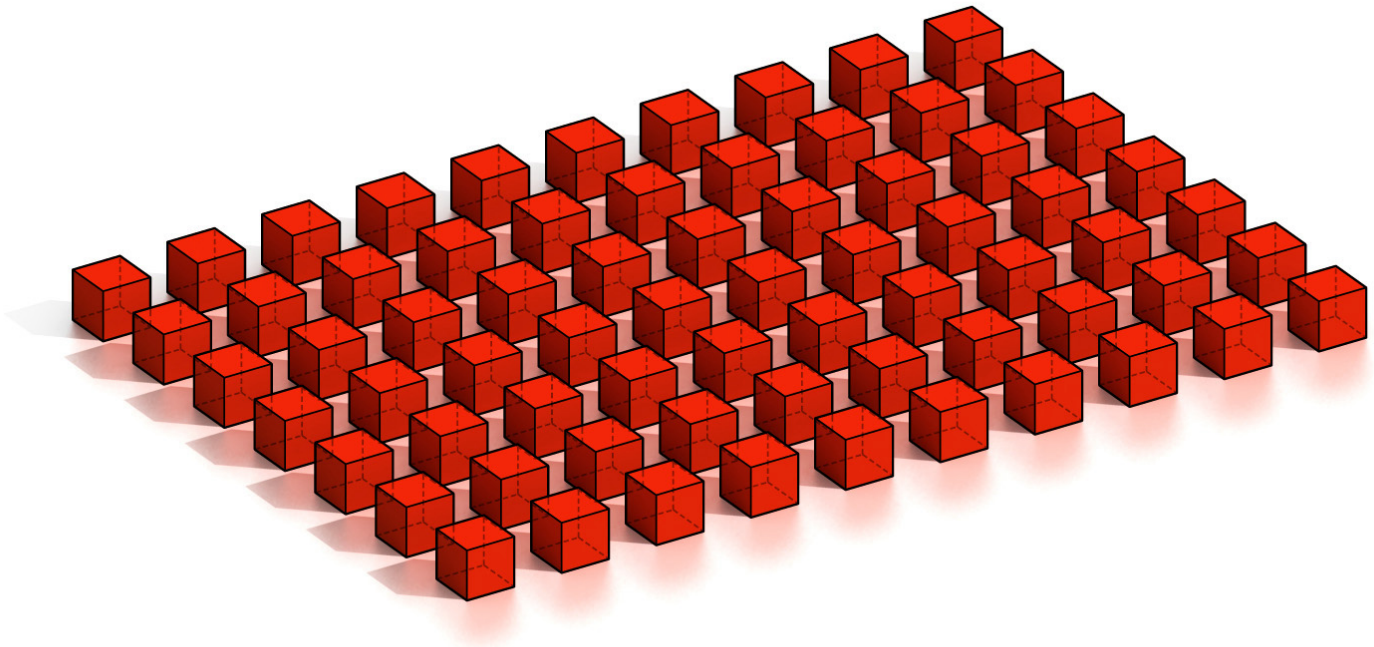
```
for x = 0 to righe do (
    obj = copy Cubo
    obj.pos = [60*x,60*y,0]
)
```

governato dal contatore `X`, per copiare il cubo creato e distanziarlo di 60 unità dalle altre copie.

Il *ciclo for* più esterno:

```
for x = 0 to colonne do
```

governato dal contatore Y, ripetendo il *ciclo* interno, copia su più righe la colonna appena creata.



15 - Serie creata mediante due *cicli for* nidificati

Fino ad ora abbiamo preso in esame delle operazioni basate su valori fissi o meglio su valori univoci. Mediante MaxScript abbiamo la possibilità di poter lavorare e scegliere anche dei valori casuali “pescati” in un *range* minimo e massimo stabilito da noi. La riga:

```
a = random 1 5
```

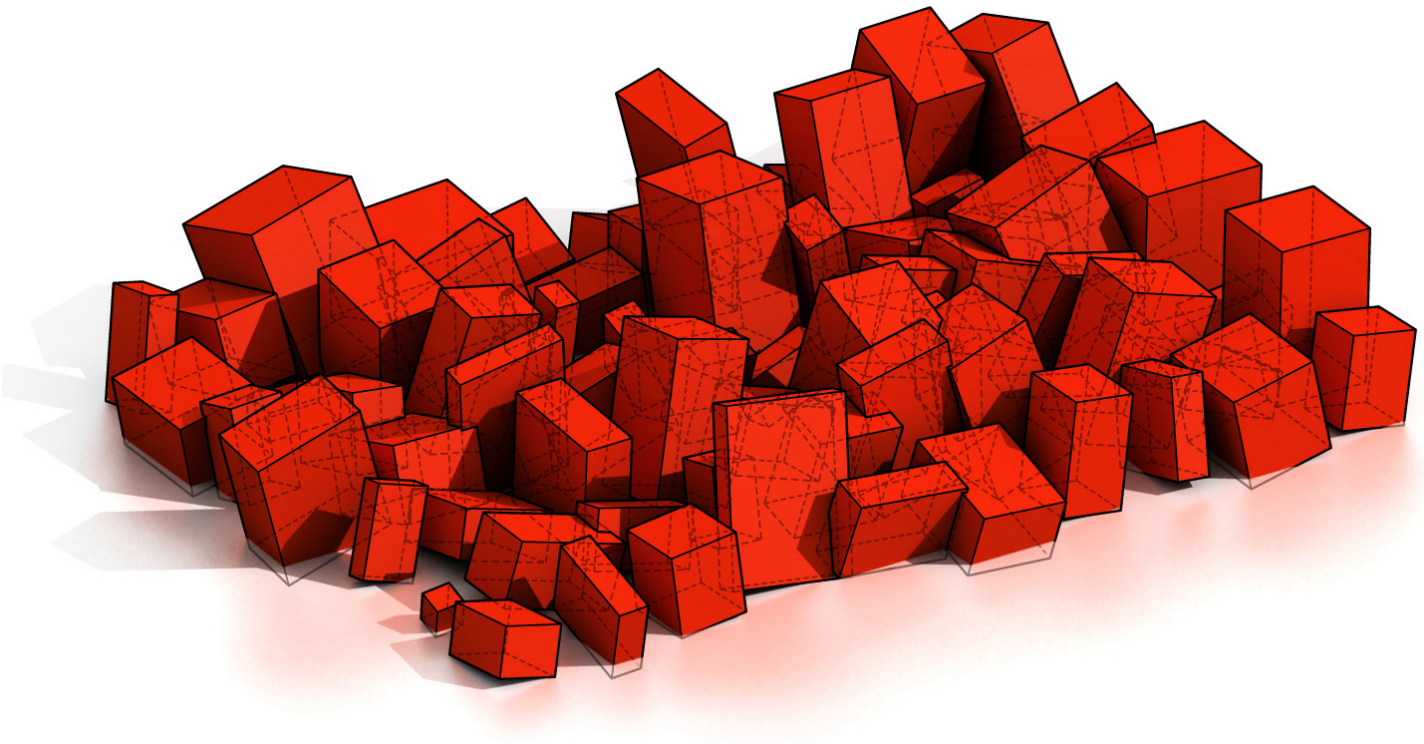
memorizza nella variabile a un valore scelto a caso tra 1 e 5; ogni volta che eseguiremo la riga, verrà assegnato ad a un valore diverso. Ecco che selezionando la serie di cubi creati con lo script precedente e mandando in esecuzione lo script:

```
for obj in selection do (
    scalx = random 0.9 5.0
    scaly = random 0.9 5.0
    scalz = random 0.9 7.0
    obj.scale = [scalx,scaly,scalz]
    obj.rotation.x_rotation = random -18.0 18.0
    obj.rotation.y_rotation = random -18.0 18.0
    obj.rotation.x_rotation = random -18.0 18.0
)
```

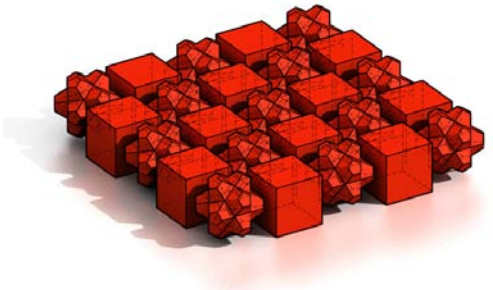
applicheremo delle trasformazioni (di scala e di rotazione) basate su valori casuali scelti in un *range* specifico per ogni tipo di trasformazione. Prendendo in esame le righe:

```
scalx = random 0.9 5.0
scaly = random 0.9 5.0
scalz = random 0.9 7.0
```

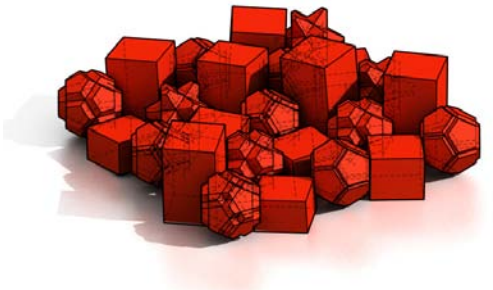
notiamo che le trasformazioni lungo gli assi X e Y saranno simili mentre quella lungo l’asse Z produrrà casualmente trasformazioni più grandi. Le configurazioni ottenibili da questo script sono infinite.



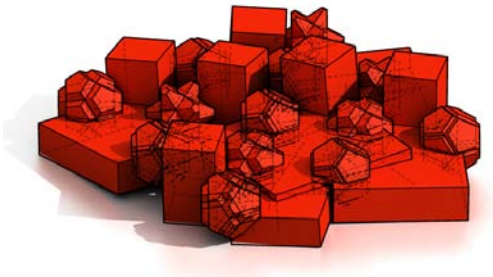
16 - Una delle infinite configurazioni ottenibili mediante l’applicazione di trasformazioni *random*



17 - Configurazione di partenza composta da primitive chiamate Box ed Hedra



18 - Una delle infinite configurazioni possibili ottenute mediante l'applicazione dello script che assegna trasformazioni differenti a seconda del tipo di geometria



19 - Una delle infinite configurazioni possibili ottenute mediante l'applicazione dello script che assegna trasformazioni differenti a seconda del tipo e dell'altezza della geometria.

Un'altra caratteristica del MaxScript, comune a tutti i linguaggi di programmazione, è quella di poter verificare delle condizioni che, se soddisfatte, eseguono del codice. Lo script:

```
for obj in selection do (
    if classof obj == box then (
          obj.height = random 25.0 90.0
          obj.rotation.z_rotation = random -90 90
          obj.rotation.x_rotation = random -5 5
          obj.rotation.y_rotation = random -5 5
    )     else (
          obj.q = random 0.0 1.0
    )
)
```

prende in esame tutti gli oggetti selezionati in Fig.17, mediante la riga:

```
if classof obj == box then
```

verifica che l'entità presa in esame dal *ciclo for* sia un cubo e, in caso affermativo, applica le trasformazioni presenti nelle parentesi. Nel caso l'entità non fosse un cubo vengono applicate le operazioni stabilite da `else`. Il risultato dell'applicazione di tale script è visibile in Fig. 18.

Si possono specificare più di una condizione e scegliere di applicare le trasformazioni solo e soltanto se tutte le condizioni vengono soddisfatte. Applicando lo scritp:

```
for obj in selection do (
    if classof obj == box and obj.height < 65 then
(
          obj.length = random 75.0 150.0
          obj.width = random 75.0 150
          obj.height = random 15.0 40
    )
)
```

alla configurazione visibile in Fig. 18, otteniamo la modifica casuale di tutti i cubi più bassi di 65 unità. Le due condizioni che determinano questa verifica sono:

```
classof obj == box
obj.height <= 65
```

unite mediante la parola *and*. Uno dei possibili risultati dell'applicazione di questo script è visibile in Fig. 19.

Quando il numero di variabili e il numero di operazioni presenti in uno script diventa eccessivo e quindi difficile da controllare mediante il solo testo si ricorre all'interfaccia.

Lo script d'esempio:

```
rollout RandTrans "Random" width:208 height:200 (
  spinner rotminx "Rotaz. Min X" range:[-50,50,-5]
  spinner rotmaxx "Rotaz. Max X" range:[-50,50,5]
  spinner rotminy "Rotaz. Min Y" range:[-50,50,-5]
  spinner rotmaxy "Rotaz. Max Y" range:[-50,50,5]
  spinner rotminz "Rotaz. Min Z" range:[-90,90,-90]
  spinner rotmaxz "Rotaz. Max Z" range:[-90,90,90]
  button applica "Ruota"

  on applica pressed do (
            for obj in selection do (
                  randx = random rotminx.value rotmaxx.value
                  randy = random rotminy.value rotmaxy.value
                  randz = random rotminz.value rotmaxz.value
                  obj.rotation.z_rotation = randx
                  obj.rotation.x_rotation = randy
                  obj.rotation.y_rotation = randz
            )
      )
)
createdialog RandTrans width:150
```

20 - Finestra creata dallo script

crea una finestra *pop-up* definita dalla riga:

```
rollout RandTrans "Random" width:208 height:200
```

dove `RandTrans` è la variabile che identifica la finestra, "`Random`" è il titolo e `width` e `height` stabiliscono la grandezza della finestra.

A questo punto lo script si divide in due parti:

- una che definisce gli elementi dell'interfaccia che permettono l'inserimento di valori e il lancio di comandi (*spinner* e *button*)
- una che contiene i comandi e le operazioni che leggono i valori inseriti mediante l'interfaccia e vengono eseguiti (*ciclo for*)

La finestra appena creata contiene degli elementi chiamati spinner e definiti da questo tipo di riga:

```
spinner rotminx "Rotaz. Min X" range:[-50,50,-5]
```

dove `spinner` indica il tipo di elemento grafico, `rotminx` indica la variabile all'interno della quale viene memorizzato il valore, "`rotaz. Min X`" è il testo che identifica il valore da inserire e `range` imposta, nell'ordine, valore minimo (-50), valore massimo (50) e valore di default (-5) dello spinner.

La riga:

```
button applica "Ruota"
```

definisce un pulsante dove `applica` è la variabile che indica l'elemento e `"Ruota"` è il testo che contraddistingue il pulsante.

La riga:

```
on applica pressed do
```

esegue i comandi presenti tra le parentesi quando il pulsante contraddistinto dalla variabile applica, viene premuto

Il *ciclo for* che viene eseguito quando il pulsante `"Ruota"` viene premuto, prende le geometrie presenti nella selezione e vi applica delle trasformazioni quantificate dai valori memorizzati all'interno delle variabili `randx`, `randy` e `randz`. Tali valori sono scelti casualmente in un *range* specificato dai valori presenti negli *spinner* mediante la riga d'esempio:

```
randx = random rotminx.value rotmaxx.value
```

Il popup viene creato dalla riga finale:

```
createdialog RandTrans width:150
```

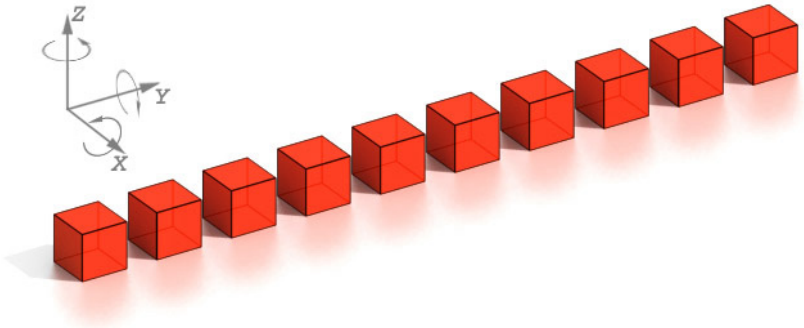
Se si volesse lanciare lo script di trasformazione appena si è cambiato un valore in uno spinner, si potrebbe inserire una riga del tipo:

```
on rotminx changed val do (applica.pressed())
```

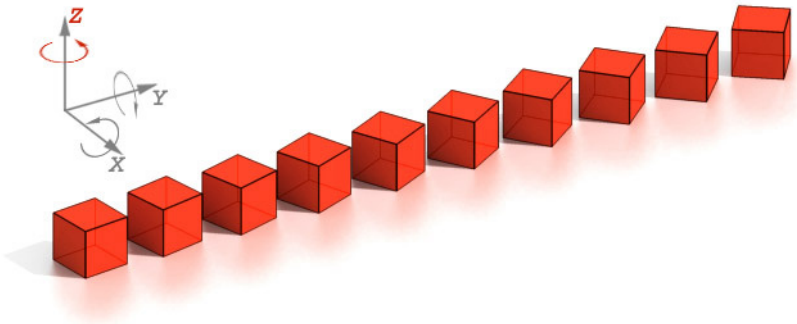
tale riga non fa altro che indicare a 3dsMax di premere il pulsante applica quando il valore all'interno dello *spinner* contraddistinto da `rotminx` viene cambiato.

Trasformazioni incrementali

Per trasformazione incrementale si intende un cambiamento di dimensione (scala o stretch) o di orientamento (rotazione) rispetto ad uno o più assi specificati, la cui quantità q, applicata ad un elemento, è pari a q sul primo elemento, a q*2 sul secondo, a q*3 sul terzo e così via.
Qui di seguito vi sono degli esempi che illustrano alcune trasformazioni incrementali applicate ad una serie di 10 elementi.



Nell'esempio sottostante è stata applicata una rotazione incrementale intorno all'asse Z.

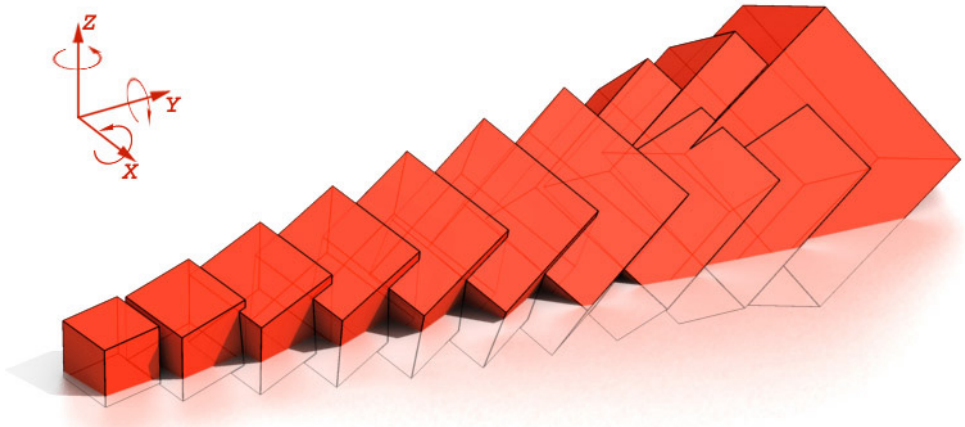


Nell'esempio sottostante è stata applicata una rotazione incrementale intorno all'asse Y ed intorno all'asse Z.

Nell'esempio sottostante è stata applicata una trasformazione di scala rispetto agli assi X e Z.

L'esempio sottostante mostra il risultato dell'applicazione simultanea delle trasformazioni rotatorie e dimensionali attorno agli assi indicati.

L'esempio sottostante mostra l'applicazione delle trasformazioni incrementali rotatorie e dimensionali rispetto a tutti e tre gli assi.



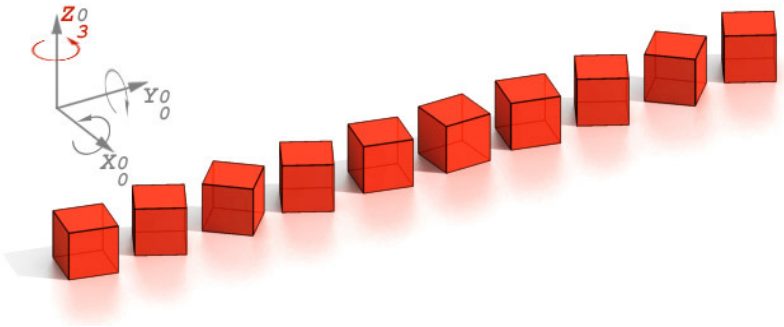
Trasformazioni incrementali dipendenti da un intervallo.

L'intervallo, specificato per ogni trasformazione (rotatoria e dimensionale) e per ogni asse, influenza l'applicazione della trasformazione incrementale.

L'intervallo i specifica il numero di oggetti al quale la quantità q viene aggiunta (incremento). Una volta raggiunto il numero di elementi specificati dall'intervallo i, la quantità q viene sottratta (decremento).

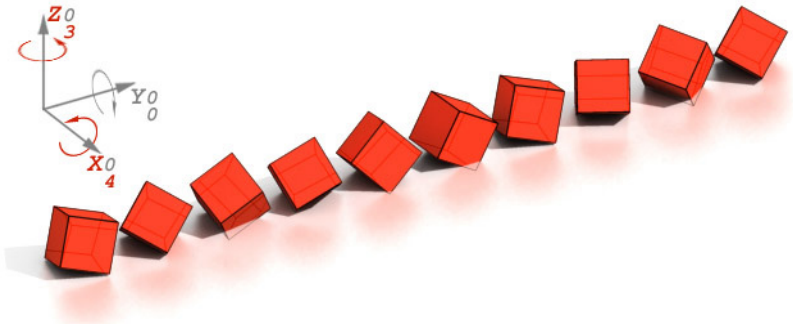
Qui di seguito vi sono alcuni esempi che illustrano delle trasformazioni incrementali applicate ad una serie di 10 elementi.

Nell'esempio sottostante è stata applicata una rotazione incrementale intorno all'asse Z. L'intervallo che specifica gli incrementi e i decrementi è uguale a 3.

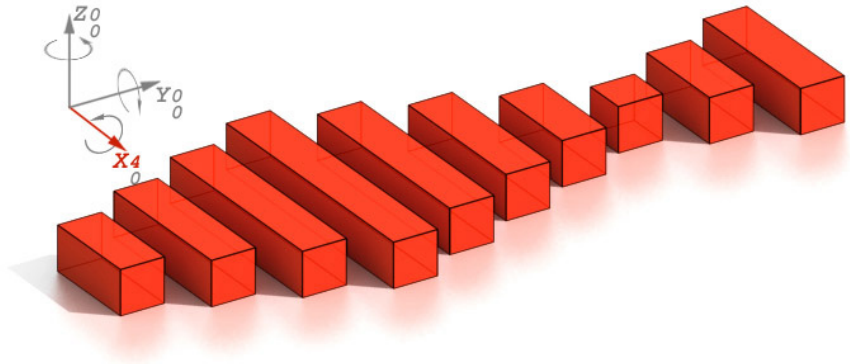


Nell'esempio sottostante è stata applicata una rotazione:

- intorno all'asse Z con un intervallo uguale a 3
- intorno all'asse X con un intervallo uguale a 4



Nell'esempio sottostante è stata applicato un cambiamento di scala rispetto all'asse X con un intervallo uguale a 4

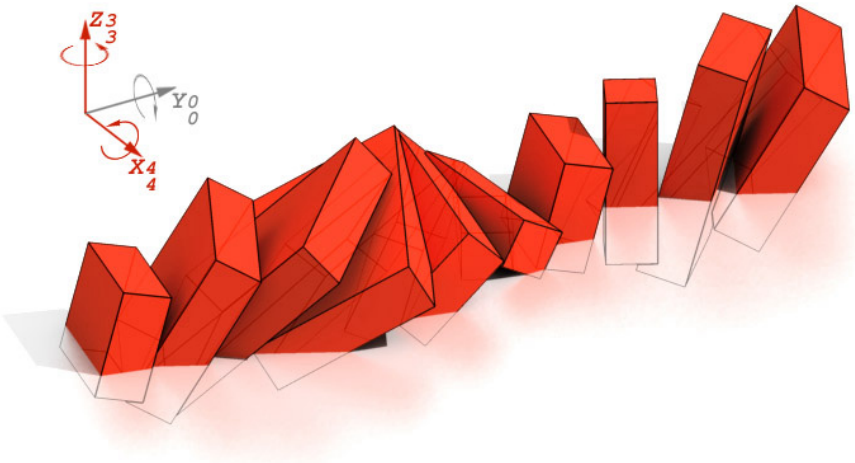


Nell'esempio sottostante è stata applicata una trasformazione di scala rispetto:

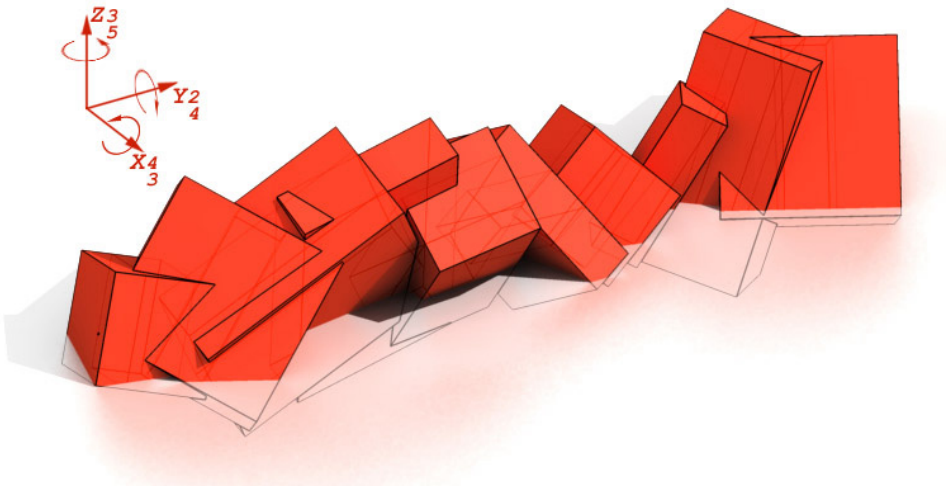
- all'asse X con un intervallo uguale a 4
- all'asse Z con un intervallo uguale a 3

e di rotazione rispetto:

- all'asse X con un intervallo uguale a 4
- all'asse Z con un intervallo uguale a 3



Esempio di applicazione delle trasformazioni incrementali (rotatorie e dimensionali) rispetto a tutti e tre gli assi con intervalli differenti per ogni asse.

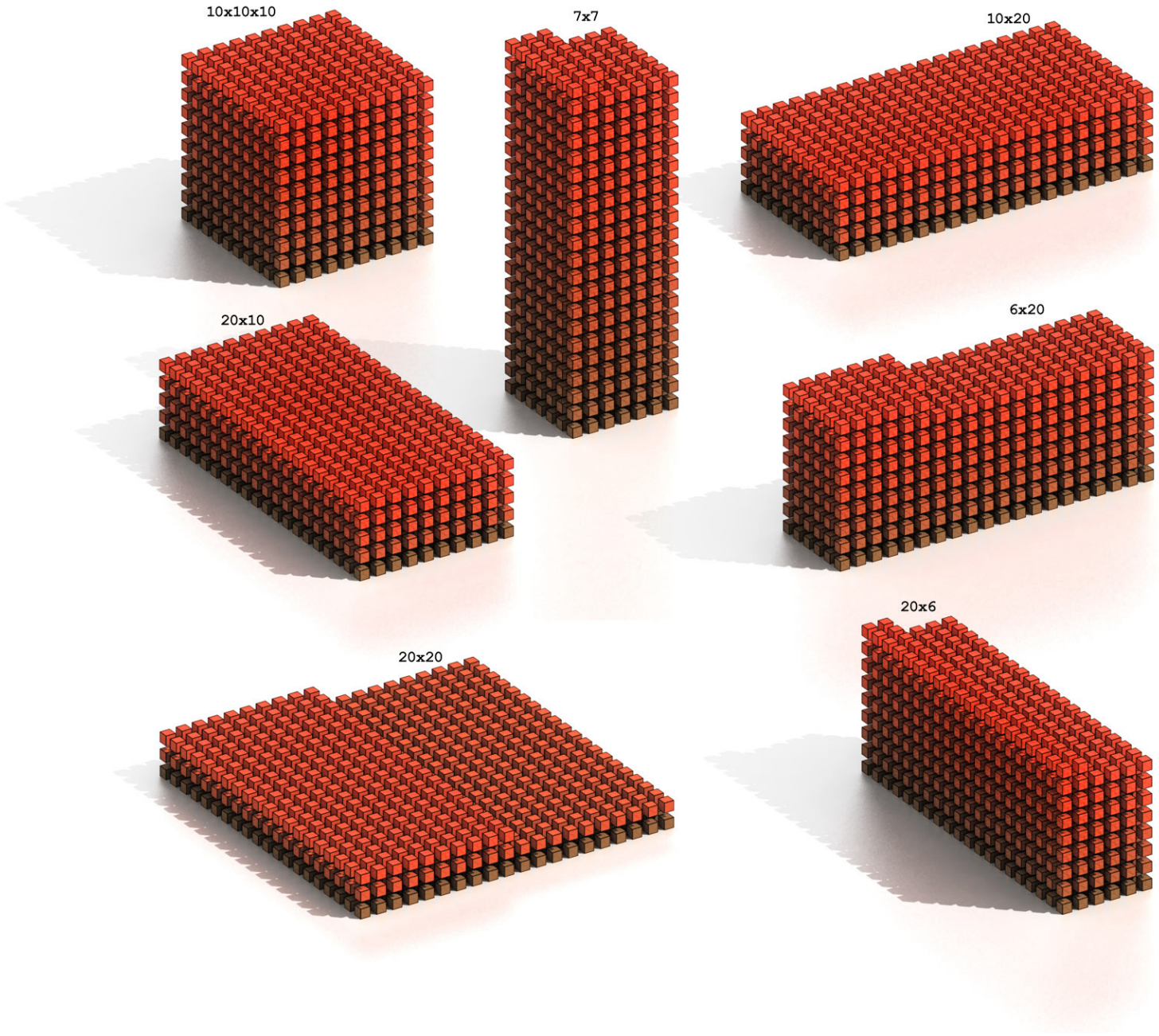


Ridistribuzione

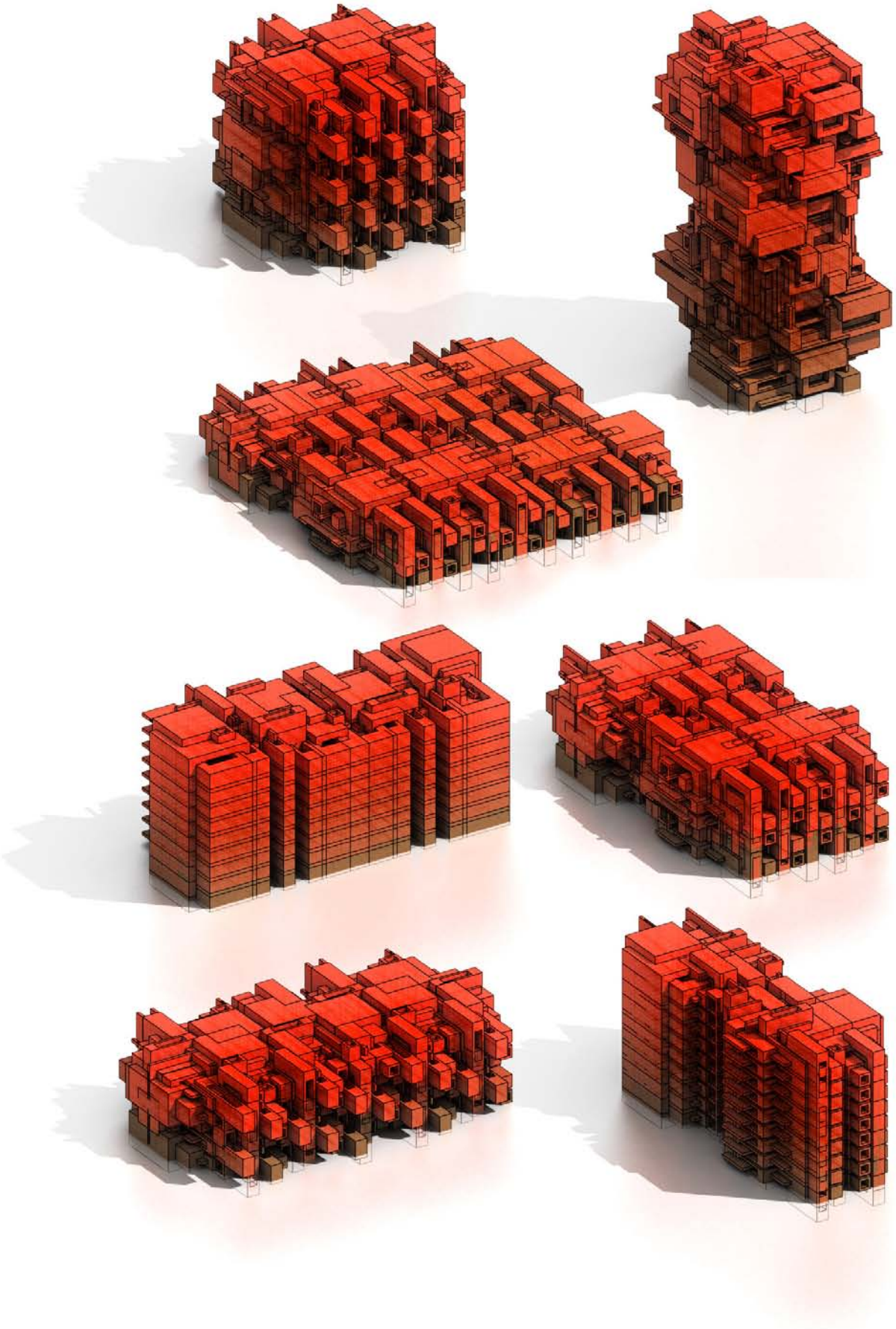
I mille elementi (10x10x10) duplicati e trasformati in maniera incrementale vengono ridistribuiti mantenendo le loro trasformazioni e cambiando soltanto la posizione rispetto al nuovo reticolo.

Fissate la larghezza e la profondità della griglia, l'altezza viene conseguentemente determinata dal numero finito degli elementi.

Nell'immagine sottostante sono riportati alcuni schemi ridistribuiti partendo dalla griglia (in alto a sx) 10x10x10.

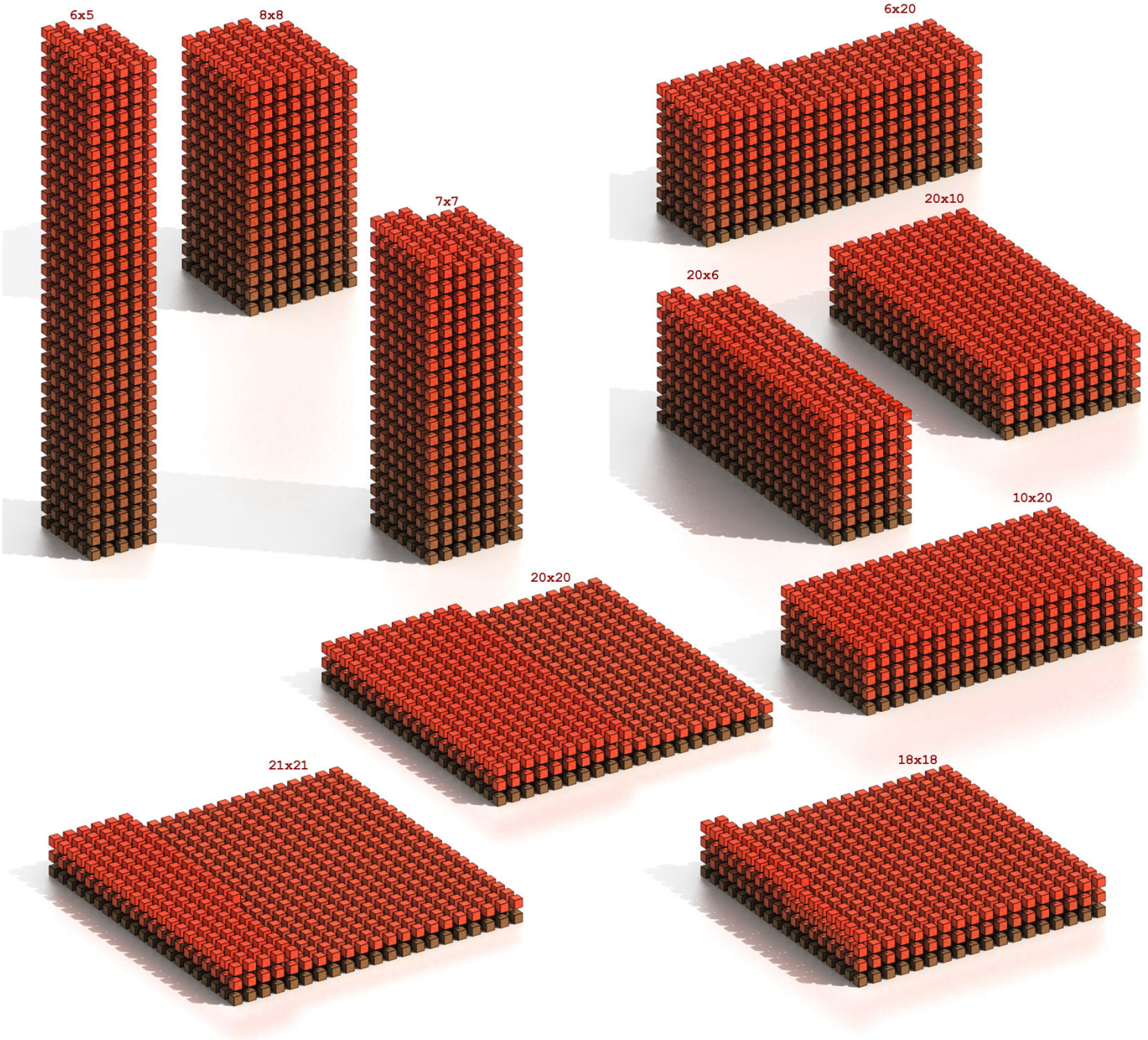


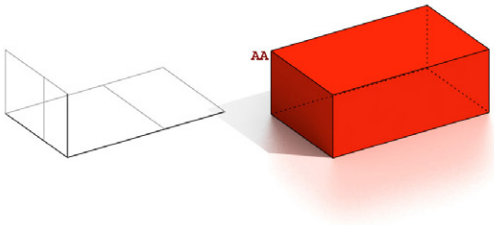
L’esempio sottostante mostra la configurazione D12 (vedere tavola n°4) ridistribuita negli schemi elencati precedentemente

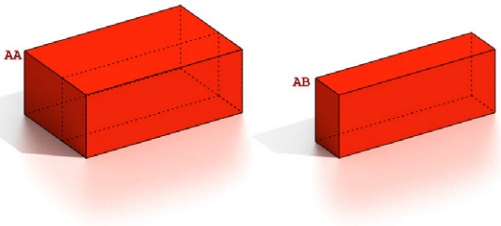


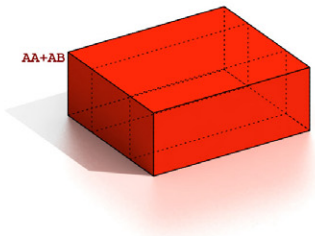
Memorie tipologiche

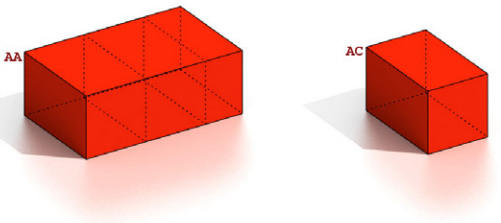
Sviluppo, mediante ridistribuzione, di alcune soluzioni accennate nell'esempio precedente.
Le soluzioni ricavate sono state ordinate in base a degli schemi che ricordano alcune tipologie edilizie quali la "torre", la "linea" e la "piastra".

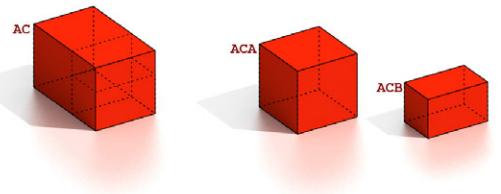


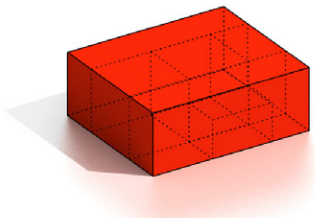








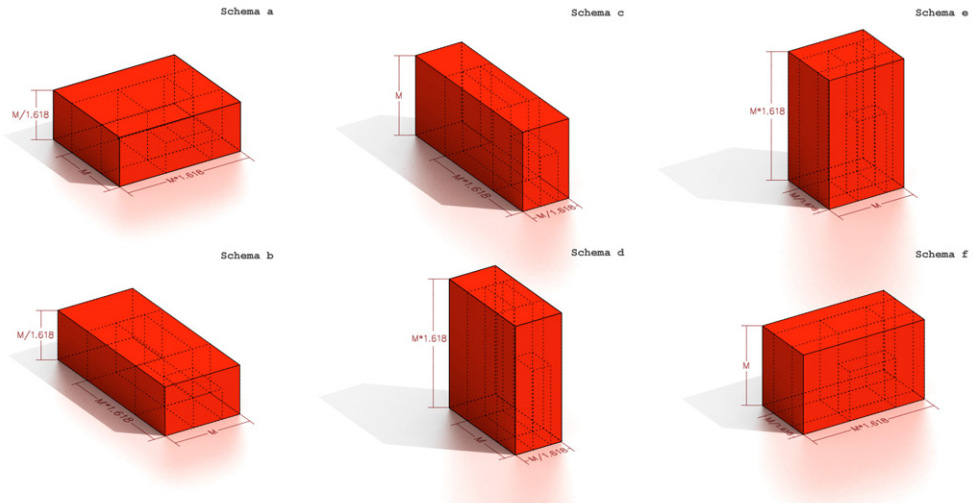


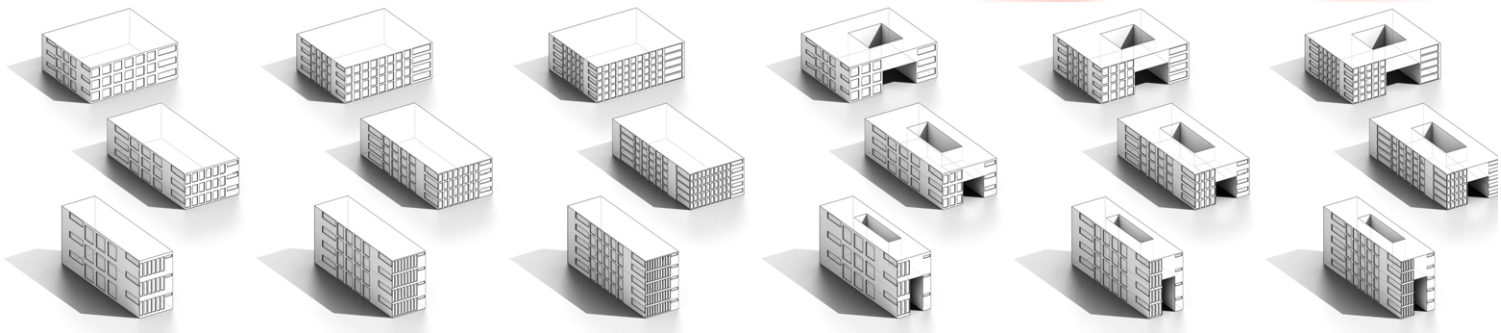


Oggetto ben formato per controllo dimensionale.

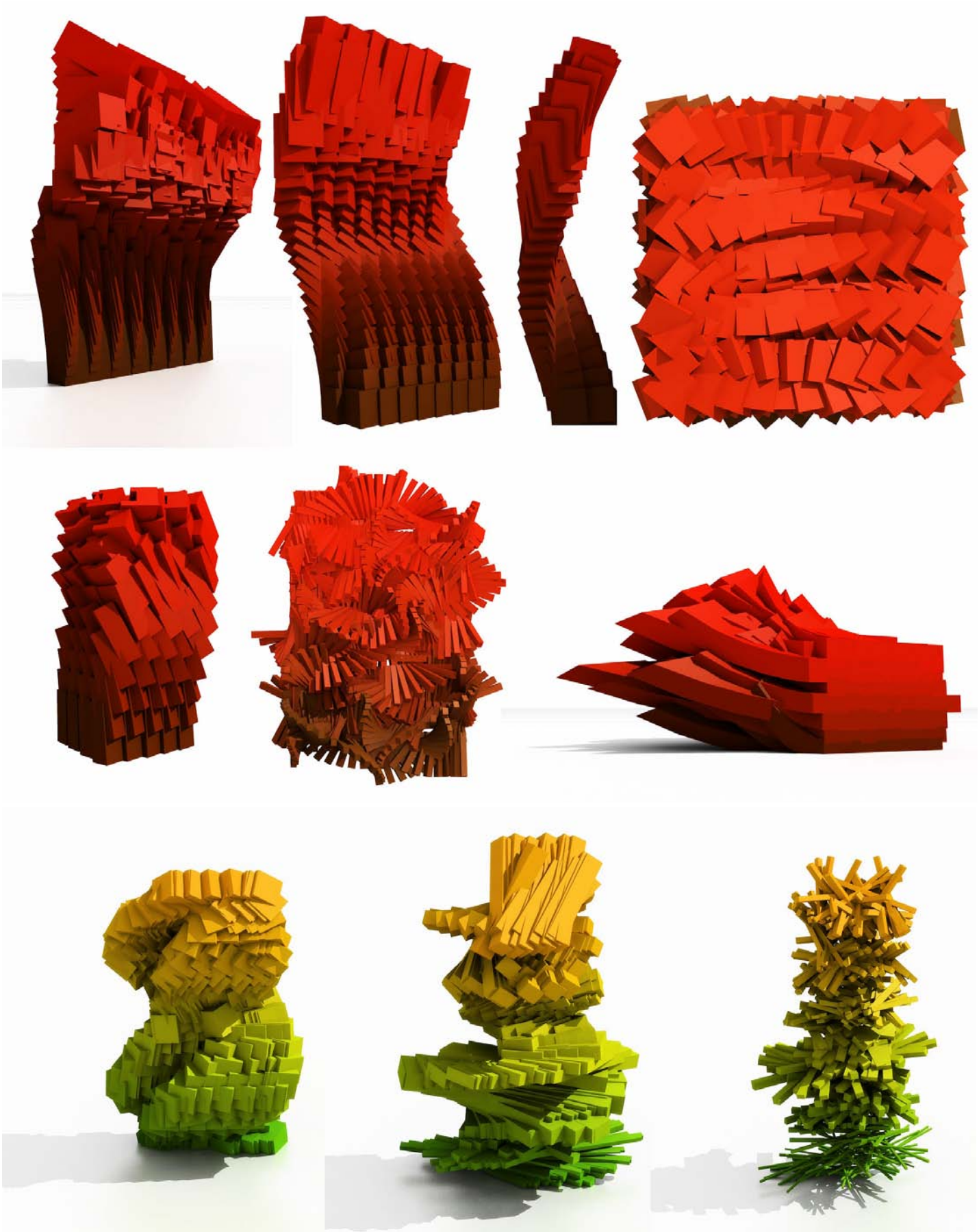
Le variazioni sperimentali fin qui esaminate hanno ruotato principalmente intorno al tema morfologico della ripetizione seguendo varie declinazioni.
L'ultima esercitazione sperimentale che viene proposta affronta un tema diverso, di notevole complessità, di possibili ed interessanti sviluppi.
Quello della costruzione di un oggetto ben formato a partire da procedimenti "automatici" (fortemente controllati) per calibratura dimensionale.
Partendo dalla misura M si ricava l'altezza (M/1.618) e la larghezza (M*1.618) con le quali si costruisce l'oggetto AA.
Dall'oggetto AA si ricava l'oggetto AB avente altezza e larghezza uguali ad AA e profondità pari a M/1,618.
Sempre dall'oggetto AA viene ricavato l'oggetto AC avente altezza e profondità pari ad AA e larghezza pari a M/1.618.
L'oggetto AC viene a sua volta diviso in due oggetti:
- ACA avente larghezza e altezza pari ad AC e profondità pari a M/1.618
- ACB avente larghezza pari ad AC, altezza pari a (M/1.618)/1.618 e profondità pari a M-(M/1.618).

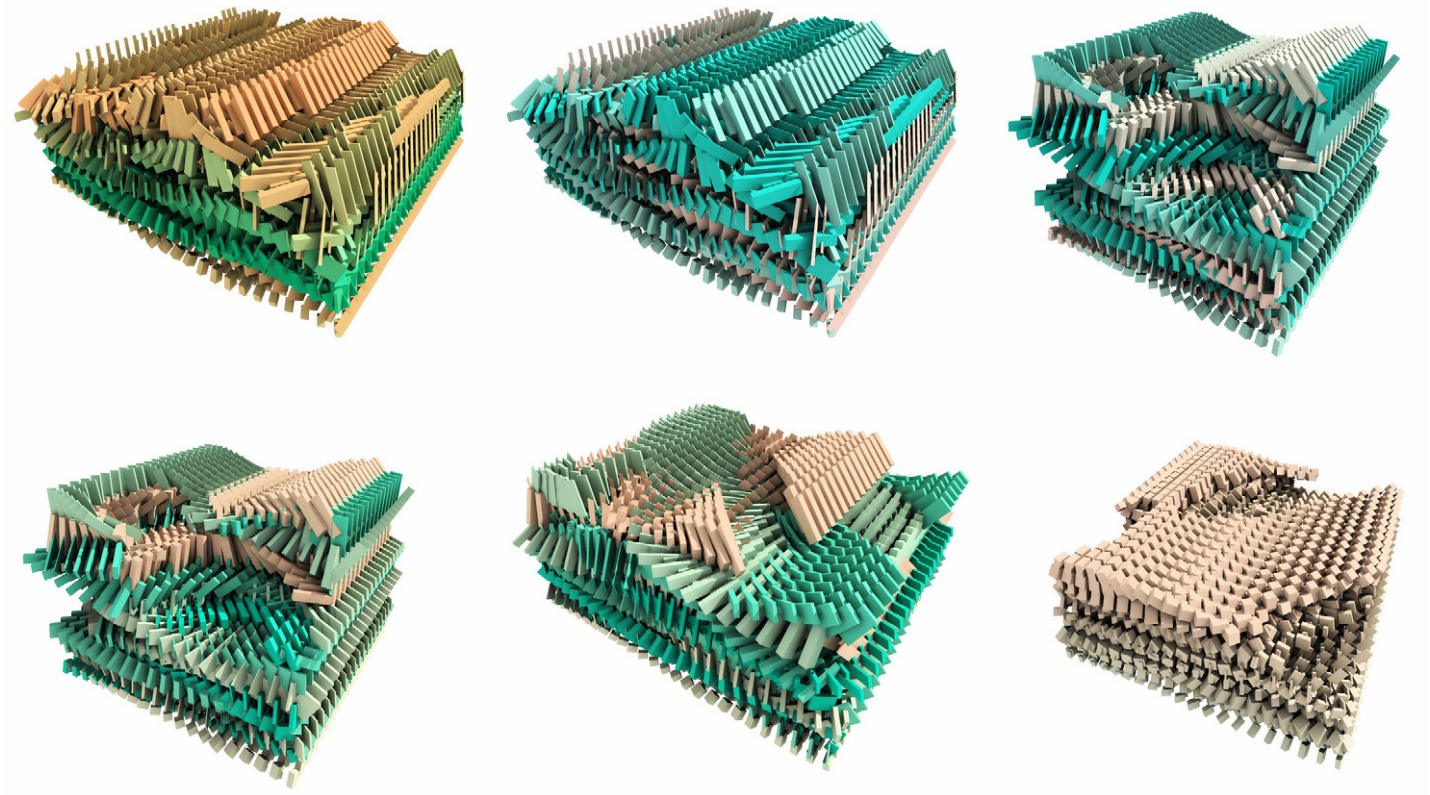
Dall'oggetto ottenuto si sono ricavati sei schemi che mostrano le possibili costruzioni del parallelepipedo assumendo M prima come larghezza, poi come profondita ed infine come altezza.
Gli schemi sono stati successivamente sviluppati presentando sei variazioni sul tema: tre, quattro e cinque piani senza corte e tre, quattro e cinque piani con corte.

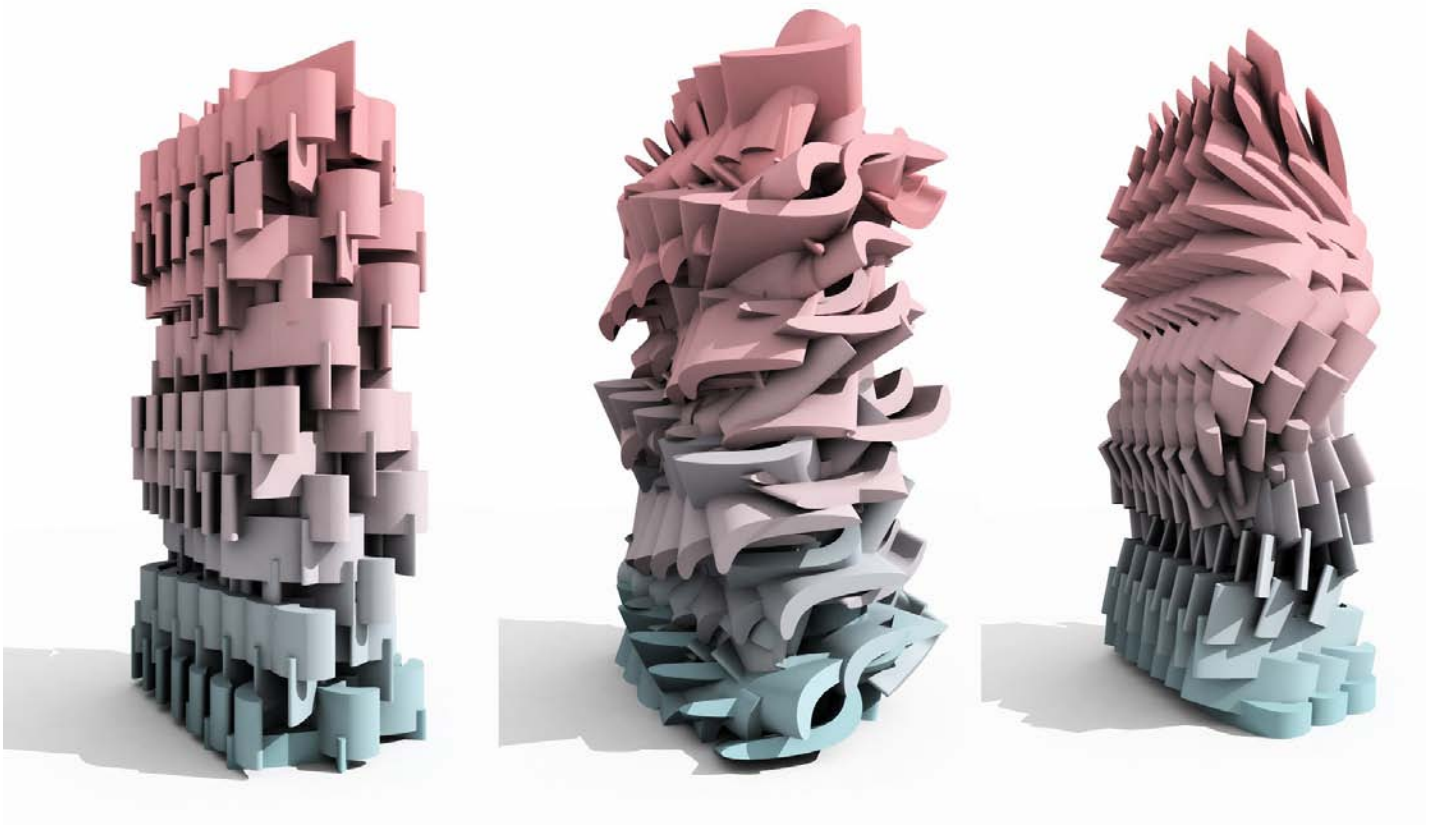


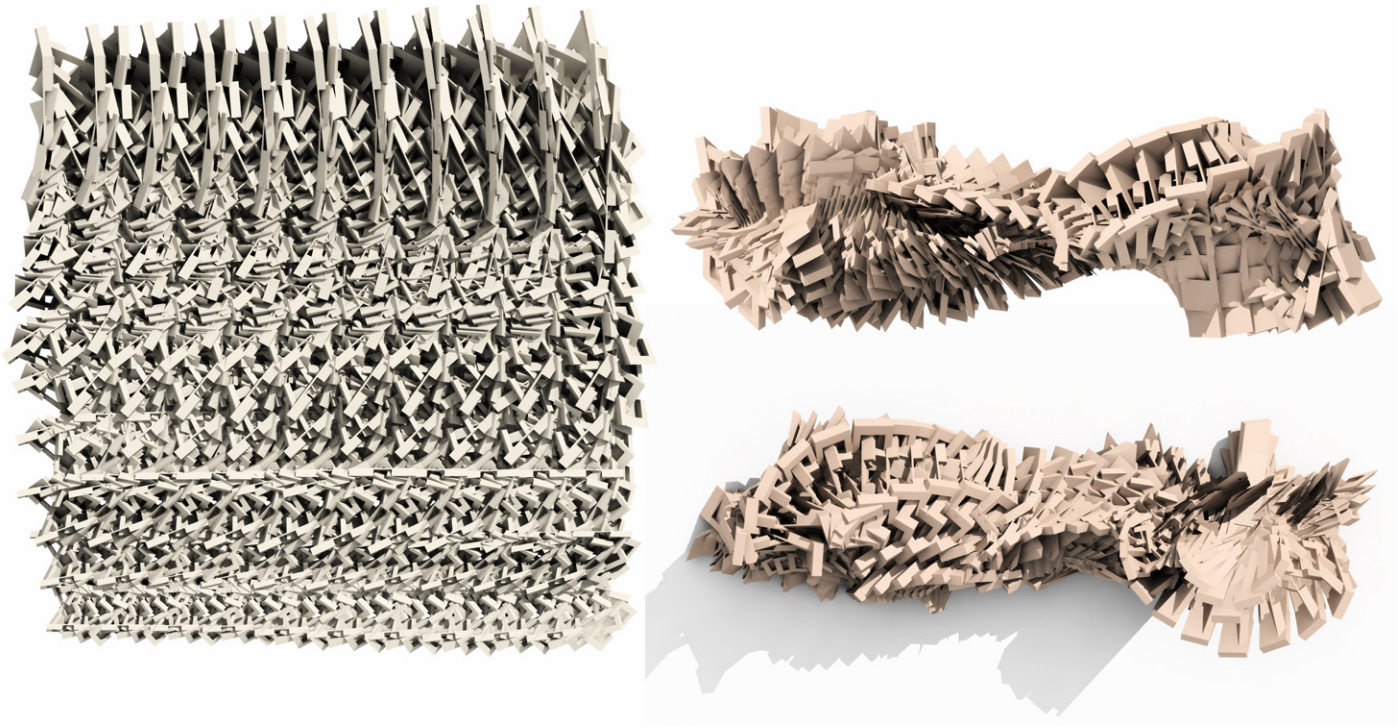


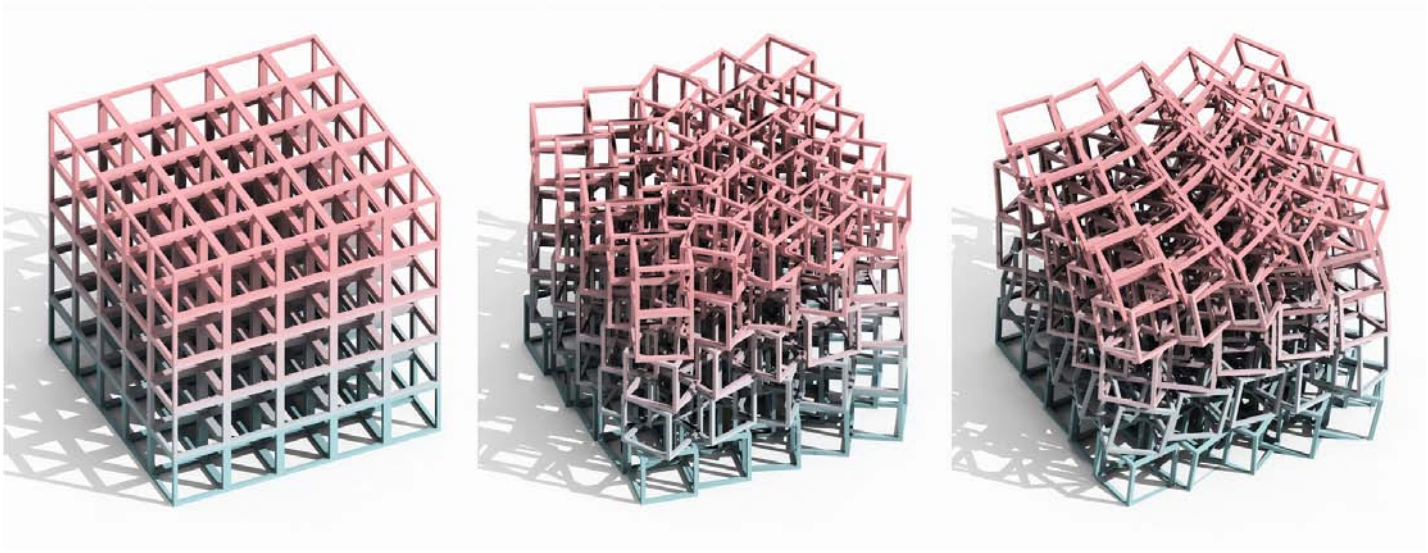
Modelli ottenuti durante lo studio e la realizzazione degli script.











Cenni sulle interfacce grafiche

La UI ovvero User Interface (interfacce utente) è la parte del computer che l'utente può vedere, sentire e toccare. È principalmente composta da due componenti: input e output. Con il primo l'utente immette i comandi tramite varie periferiche: tastiera, mouse, trackball, touch screen, comandi vocali etc. Con il secondo, mediante lo schermo, il computer mostra i risultati delle richieste dell'utente.

La UI può essere:

Testuale (linea di comando)

Grafica (GUI - Graphic User Interface) dove i componenti del sistema, i dati e i programmi compaiono sullo schermo sotto forma di oggetti.

Elementi che compongono una GUI

Il **desktop** è l'area di lavoro principale; contiene tutti gli elementi della GUI e funge da sfondo a tutte le operazioni svolte. È anche un'area dove vengono posizionati gli oggetti (solitamente icone) che si vuole utilizzare e raggiungere velocemente.

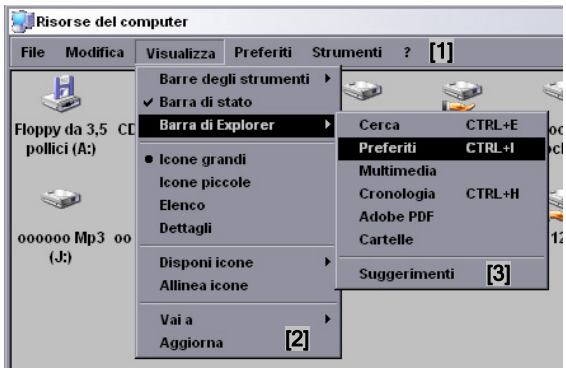
La **taskbar** si trova all'interno del desktop; è l'elemento d'accesso principale a tutte le informazioni, risorse e programmi del sistema. Contiene, sotto forma di bottoni, tutti i task (processi) e le finestre al momento attive; permette di passare velocemente da un task all'altro.
Nei sistemi operativi Windows la taskbar contiene:

- il **pulsante Start**, che apre un menù con i comandi per l'accesso a tutti i dati e le risorse del sistema;
- la **quick launch bar** contenente i comandi per accedere velocemente al web, alla mail e a tutte le risorse basilari del sistema.
- **status notification area**, posizionata nell'angolo in basso a destra del desktop; presenta, di default, l'orologio di sistema e viene utilizzata dai programmi residenti in memoria come area di accesso veloce alle opzioni.

Le **icone** sono la rappresentazione grafica di un elemento del sistema operativo (file, cartelle, programmi, etc).

La **finestra** è una porzione di schermo, solitamente rettangolare, delimitata da bordi, che può essere spostata nello schermo a piacimento e contenente una parte (astratta) del computer, dati di vario genere, un programma in esecuzione o un singolo messaggio.

L'utente si trova a manipolare dati provenienti da svariate sorgenti, sintetizzando, riassumendo e riorganizzando le informazioni. Questi processi non sono quasi mai svolti linearmente e continuativamente ma sono frammentati e interrotti da svariati eventi quali telefonate, distrazioni, consultazioni con i colleghi etc. In questo contesto la struttura "a finestre" aiuta l'utente ad organizzare il proprio lavoro, dandogli la possibilità di avere sott'occhio tutti i componenti del processo lavorativo e permettendogli di cambiare dinamicamente l'ordine delle incombenze.
Le finestre sono composte da una serie di elementi chiamati widgets; alcuni sono presenti in tutte le finestre, altri compaiono solo in alcuni tipi.



Frame (cornice): di solito è rettangolare e delimita la finestra permettendo di distinguerla dalle altre. I bordi possono essere di differente spessore e colore e la loro variazione è utilizzata per definire il tipo di finestra.

Title bar (barra del titolo): chiamata anche caption, caption bar o title area è la parte superiore della finestra che contiene un titolo, una descrizione o in generale una stringa di testo che ne identifica il contenuto. Spesso è utilizzata anche come area (da clickare e trascinare) per lo spostamento della finestra. All'estrema sinistra della title bar è presente la title bar icon che, clickata, fa comparire un pull-down memu (menù a cascata) contenente le opzioni relative alle proprietà della finestra.

Menu bar [1]: solitamente è posizionato sul bordo superiore della finestra, appena sotto la title bar e contiene una lista di voci che, una volta clickate, presentano un pull-down menu

Nelle finestre possono comparire diverse tipologie di menù:

pull-down [2] chiamati anche high level menu;

cascading [3] è un submenu derivante dall'high level menu, può contenere a sua volta altri scascading; è usato principalmente per ridurre la quantità di opzioni visibili in una volta sola e razionalizzarne l'ordine;

pop-up sono menù, ordinati verticalmente, che compaiono su richiesta dell'utente; spesso sono associati al click con tasto destro del mouse e permettono di accedere alle opzioni dell'oggetto clickato;

tear-off sono identici ai pull-down, solo che non spariscono quando si clicka altrove e possono essere posizionati ovunque sullo schermo;

iconic le voci, che negli altri menù sono sotto forma di testo, in questa particolare tipologia, si trovano sotto forma di icone.

Status bar (barra di stato): è la parte inferiore della finestra e vi si trovano informazioni inerenti al contenuto della finestra.

Scroll bar (barra di scorrimento): è utilizzata per lo scorrimento delle informazioni contenute nella finestra quando esse sono eccessive ed escono fuori dai bordi; possono essere sia orizzontali che verticali.

Split box: è un elemento che si trova sul bordo superiore della scroll bar e permette di dividere in due parti il contenuto della finestra; è spesso utilizzato nei programmi di videoscrittura o nei fogli di calcolo per controllare contemporaneamente due porzioni differenti del lavoro.

Size grip: si trova, di solito, nell'angolo in basso a destra e, una volta clickato, permette di ridimensionare la finestra.

Work Area: è la porzione di schermo, interna alla finestra, dove si trovano i dati e/o l'area di lavoro del software in esecuzione.

Buttons: sono dei rettangoli contenenti una descrizione (testuale o grafica) che ne indica la funzione. Sono realizzati con stili differenti, a seconda della loro funzione.

- Vi sono quelli che assomigliano a dei pulsanti, hanno forma rettangolare, sono posizionati all'interno delle finestre e hanno una descrizione testuale che indica il comando o l'azione svolta quando premuti.

- Vi sono quelli a forma quadrata o rettangolare con all'interno un'icona come descrizione. A loro sono assegnati dei comandi utilizzati frequentemente e per questo sono parte integrante delle toolbar.

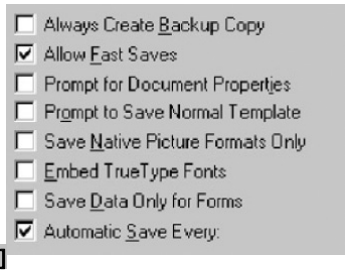
- Infine possiamo trovare quelli che hanno un simbolo come descrizione e sono utilizzati principalmente per accedere e regolare determinati parametri della finestra o dell'applicazione.

[4]

Toolbar (barra degli strumenti): è una porzione di finestra contenente i comandi (sotto forma di icone e bottoni) dell'applicazione che devono essere raggiunti velocemente. A seconda dell'applicazione possono essere personalizzabili, dragable (trascinabili) per mezzo di un apposita area chiamata grip area e ridimensionabili.

Text boxes (casella di testo): come dice il nome è una casella contenente del testo (a linea singola o multipla) che può essere immesso, modificato o già presente in modalità read only.

Radio buttons [4]: sono composti da due elementi: una forma geometrica (solitamente un cerchio) e una descrizione testuale. Sono utilizzati per la scelta di una tra diverse opzioni.



[5]

Check boxes [5]: sono anch'essi composti da due elementi: una forma geometrica (solitamente un quadrato) e una descrizione testuale. Sono utilizzati per l'attivazione o la disattivazione di una o più opzioni.

List boxes: è una casella di testo, apparentemente a singola riga, contenente una lista di opzioni (affiancata da scroll bar) che compare clickando su un bottone posto all'estrema destra della casella.

[6]

Spin boxes [6]: sono caratterizzati da una descrizione, una casella di testo contenente dei valori numerici e due pulsanti che, se premuti, permettono di aumentare o diminuire il valore del numero presente nel box.

Slider: è composta da una scala numerica, un bottone che, trascinato, permette di impostare il valore della scala numerica desiderato, un text box che indica il valore scelto o impostato. Può avere anche due bottoni per la regolazione del valore come negli spin boxes.

[7]

Tab [7]: è un metodo di divisione delle finestre. Si presentano come una sorta di cartelle contraddistinte da linguette riportanti la descrizione della sezione interessata.

Palette (tavolozza): una vera e propria tavolozza virtuale che elenca, graficamente, le opzioni (colori, campiture etc.) disponibili.

Note bibliografiche sulle interfacce:

- Wilber O. Galitz, *The essential Guide to User Interface Design,* ed. John Wiley & Sons, Inc. – 2002 USA
- AA.VV., *Fundamentals of Designing User Interaction,* 2002 Microsoft
- *http://en.wikipedia.org/wiki/Widget_%28computing%29*
- *http://en.wikipedia.org/wiki/Graphical_user_interface*
- *http://www.windoweb.it/edpstory_new/eg.htm*

Cronologia delle GUI

Nel 1970 i ricercatori della Xerox inventano il mouse (con relativo sistema “punta e seleziona”) come metodo di comunicazione uomo-macchina.

1970 Invenzione, da parte della Xerox, del mouse (con relativo sistema “punta e seleziona”) come metodo di comunicazione uomo-macchina

1973 I ricercatori della Xerox di Palo Alto mettono a punto gli elementi fondamentali della GUI

1981 La Xerox introduce sul mercato il mouse.

1983 La Apple rilascia Lisa, primo sistema operativo con pull-down menu e menu bars

1984 La Apple immette sul mercato il primo personal computer chiamato Machintosh con sistema operativo Lisa

1985 Uscita di Microsoft Windows 1.0

La Commodore commercializza l’Amiga 1000

1987 Apple introduce sul mercato il primo Macintosh a colori

1989 Nascita della GUI di UNIX

Uscita di Windows 3.0

1992 Nascita di OS/2 della IBM

Windows 3.1

1993 Windows NT

1995 Windows 95

1994 OS/2 Warp 4

1997 Mac OS 8

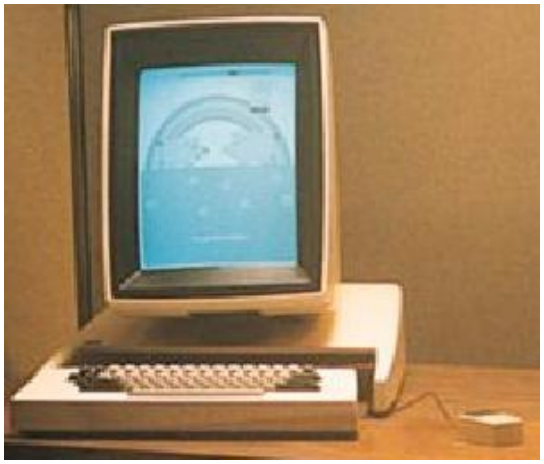
1998 Windows 98

1999 Mac OS X Server (basato sul sistema operativo UNIX)

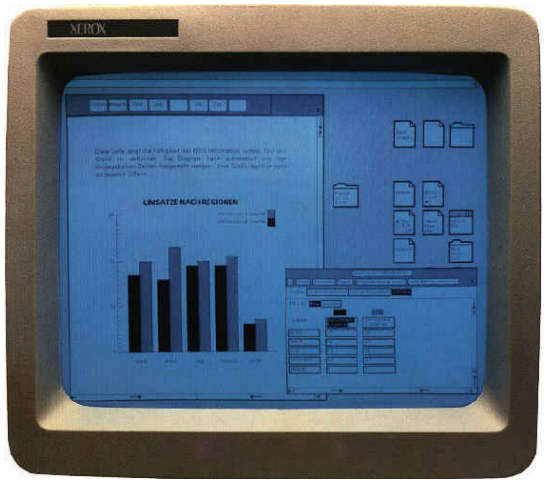
2000 Windows 2000 e Windows ME

2001 Windows XP

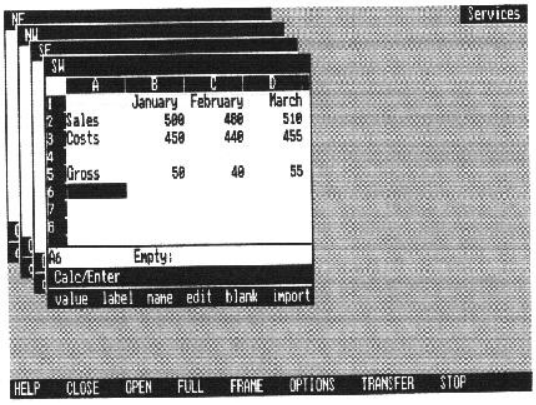
▼1973 i ricercatori della Xerox inventano il mouse (con relativo sistema “punta e seleziona”) come metodo di comunicazione uomo-macchina e mettono a punto il sistema **Alto**.



▼1981 La Xerox presenta **Star**, il successore di **Alto**.

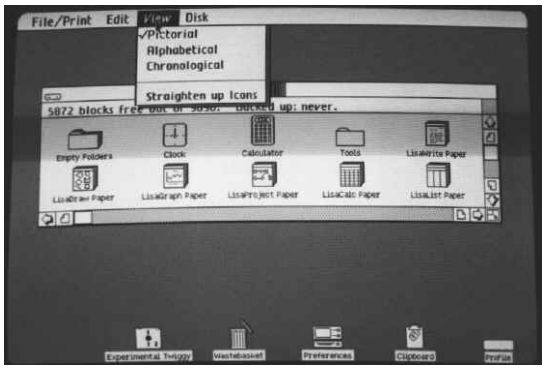


▼1983 Visi Corp rilascia **Visi On**, il primo ambiente software grafico integrato per il PC IBM.

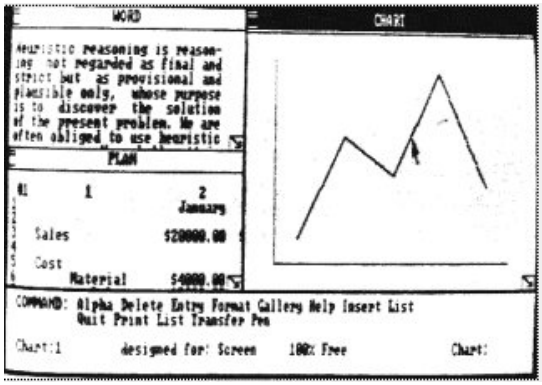


▼1980 La Three Rivers Computer Corporation presenta la workstation grafica **Perq**

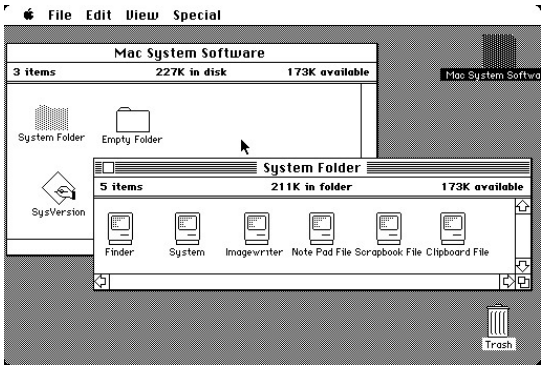
▼1983 La Apple presenta **Lisa**.



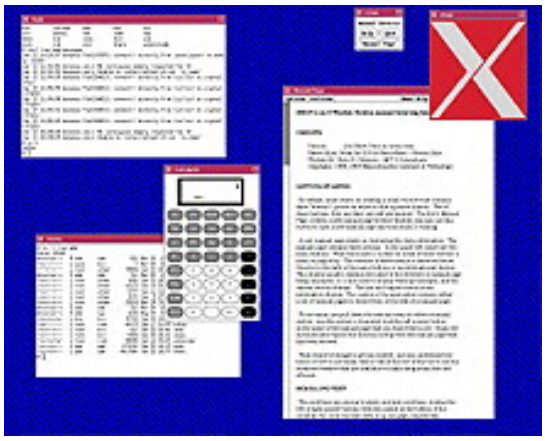
▼1983 Microsoft annuncia Windows



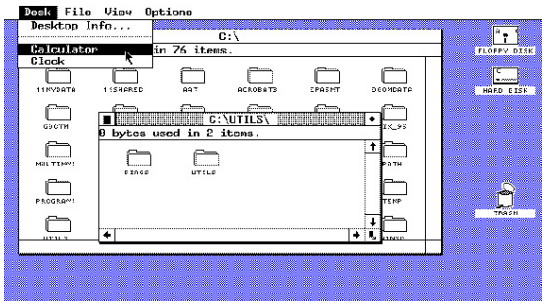
▼1984 A gennaio la Apple presenta il suo **Macintosh**



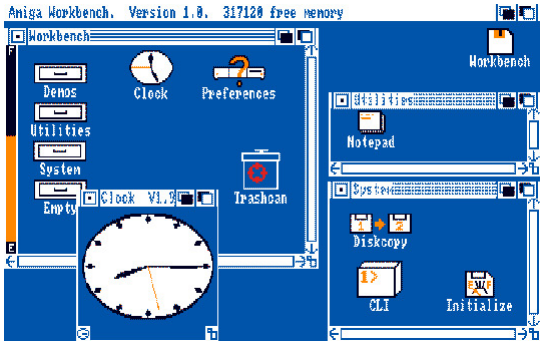
▼1984 al MIT viene presentato "**Window system X**"



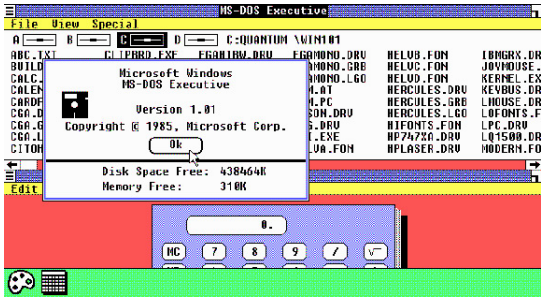
▼1984 La Digital Research annuncia il suo **GEM**



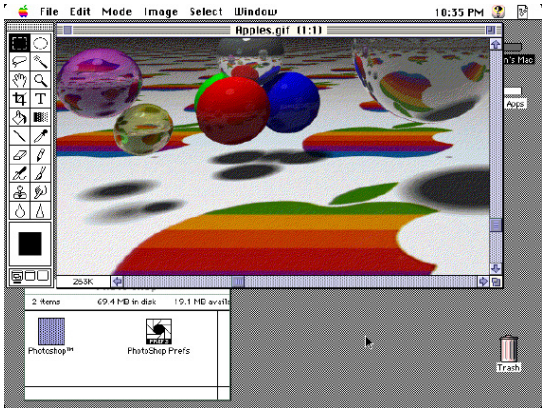
▼1985 Commodore lancia **l'Amiga1000**



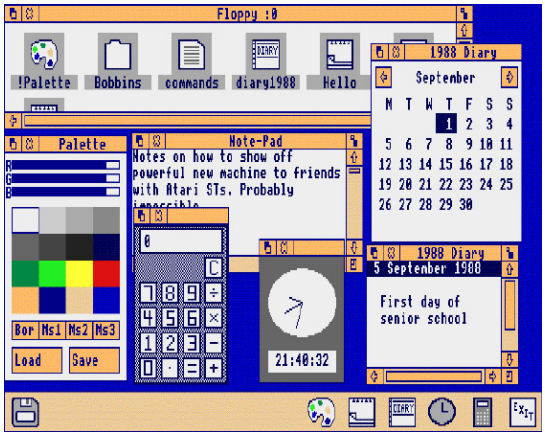
▼1985 Microsoft lancia **Windows 1.0**



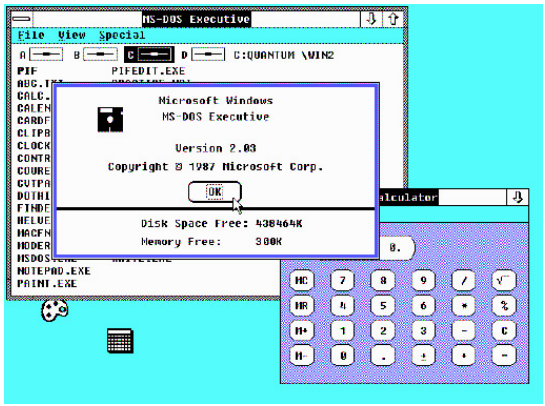
▼1987 **Macintosh II**



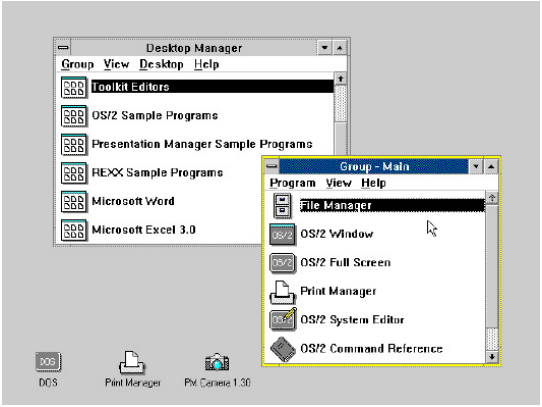
▼1987 La Acorn annuncia **Arthur**



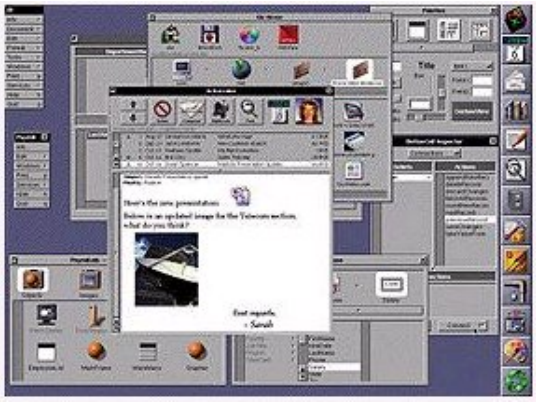
▼1987 **Windows 2**



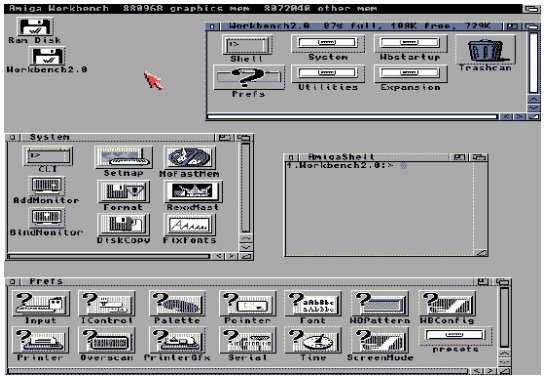
▼1988 IBM rilascia **OS/2 1.10 Standard Edition**



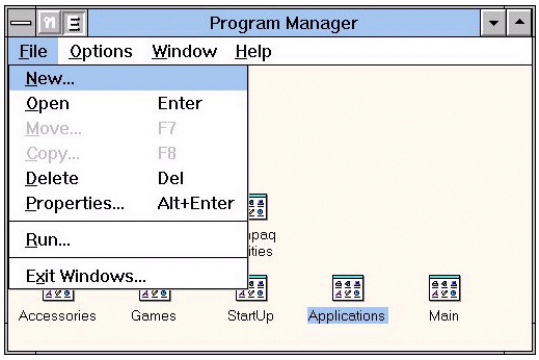
▼1988 **NExT**



▼1990 Commodore rilascia l'**Amiga Workbench 2** per

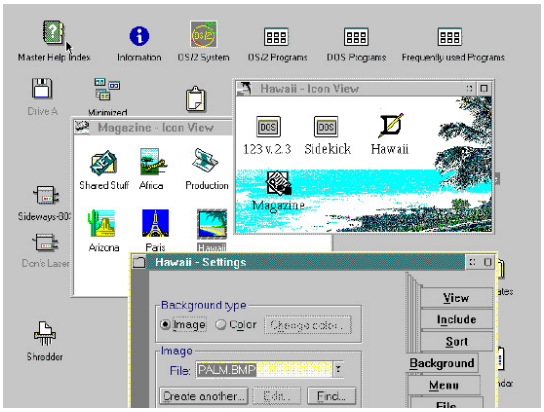


▼1990 La Microsoft rilascia **Windows 3.0**.

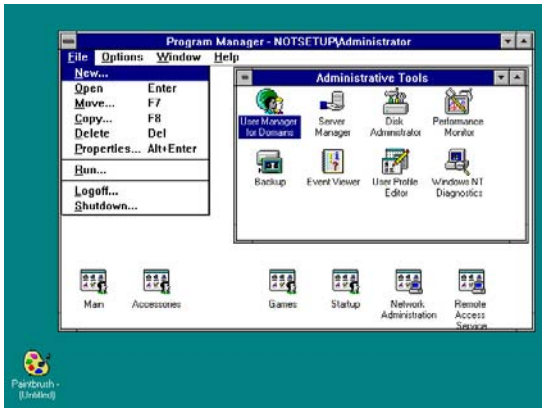


▼1992 **Amiga Workbench 3**

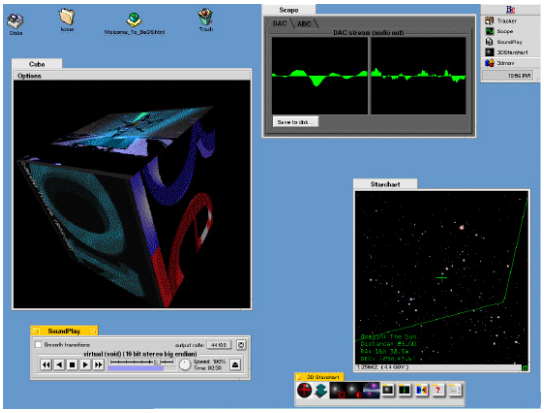
▼1992 la IBM presenta **OS/2 versione 2.0**



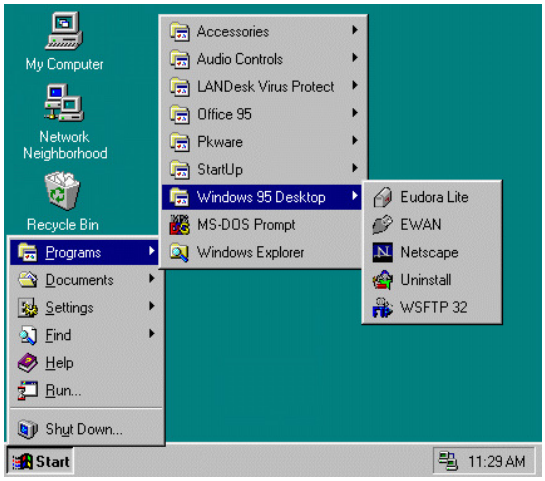
▼1993 **Windows NT**



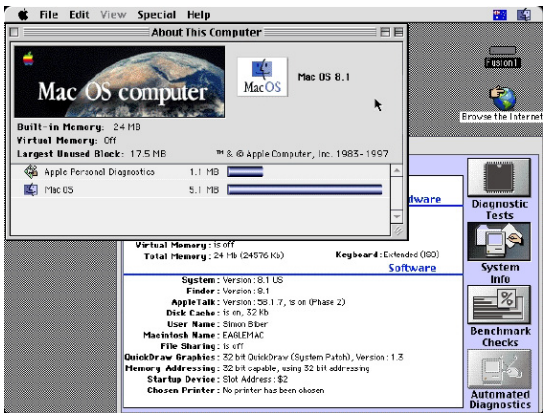
▼1995 **Beos**



▼1995 **Windows 95**

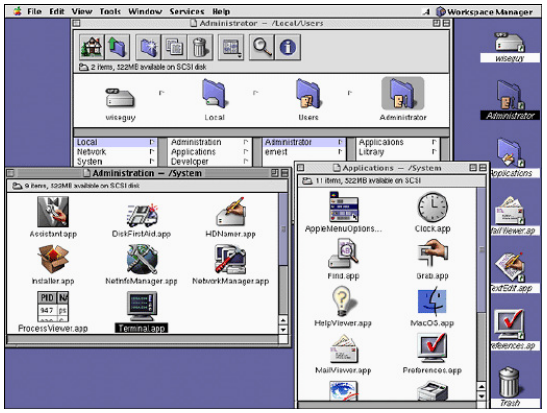


▼1997 **Mac OS 8**



▼1998 **Windows NT 4**

▼1999 **Mac OS X Server**

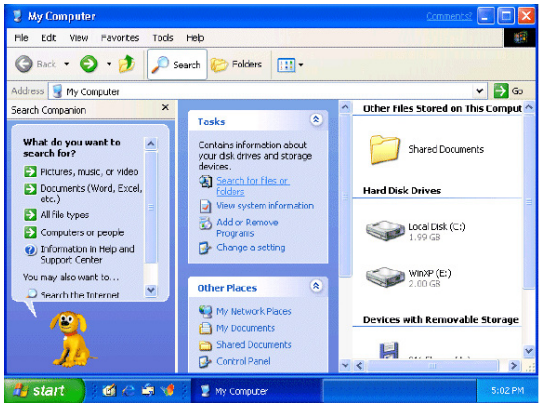


▼2000 **Windows 98**

▼2000 **Windows 2000**

▼2000 **Mac OS X Client** versione **Aqua**

▼2001 **Windows XP**

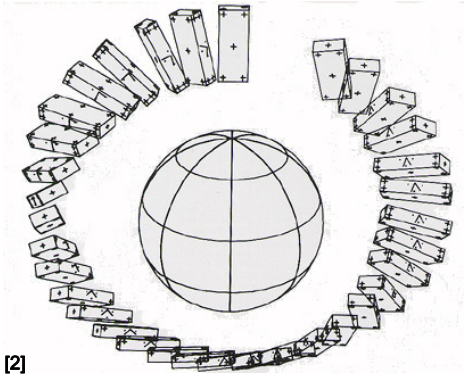


Appendice C

Cronologia degli eventi più importanti nella storia della computer grafica.

[1]

Il primo passo verso lo studio e lo sviluppo della computer grafica fu fatto nel 1961 da uno studente del MIT; il suo nome era Ivan Sutherland. Egli creò per la prima volta un programma per disegnare chiamato Sketchpad. Grazie all'uso di una penna ottica[1] era possibile disegnare forme semplici, come linee e cerchi, poggiandola direttamente sullo schermo. Ivan Sutherland riuscì a risolvere tutti i problemi, relativi al disegno digitale, che incontrò. Molte interfacce grafiche e strumenti che utilizziamo al giorno d'oggi nei programmi di grafica si basano sugli stessi concetti di "Sketchpad". Ad esempio se uno volesse disegnare un rettangolo egli non dovrebbe fare altro che specificare la posizione e la grandezza della forma, senza preoccuparsi di disegnare tutti e quattro i lati; ci penserà il software a disegnarli in base ai dati ricevuti. Le forme che Sutherland permise di disegnare non erano solamente punti su un monitor ma veri e propri oggetti che potevano essere modificati. Egli posò le basi per il disegno vettoriale.



[2]

Nel 1963 alcuni ricercatori iniziarono a creare simulazioni filmate. Il primo fu E. E. Zajac che creò "Simulation of a two-giro gravity attitude control system"[2]. In seguito Frank Sindon creò "Force, Mass and Motion illustratine Newton's laws of motion in operation", Nelson Max produsse "Flow of a Viscous Fluid".

Nel 1966 sempre Sutherland creò il primo display stereoscopico; lo chiamò "la spada di Damocle" per via delle ingenti risorse hardware richieste per il supporto.

Nel 1970 Ed. Catmull dell'Università dello Utah elaborò una delle prime animazione in 3d. l'animazione mostrava la sua mano mentre si apriva e si chiudeva.

Un grande passo avanti nella storia della computer grafica 3d fu fatto quando, sempre all'Università dello Utah, fu scritto il primo algoritmo per il calcolo e il disegno di superfici con linee nascoste. Successivamente si elaborarono strumenti per il rendering[15] e lo shading[16] di superfici ombreggiate piatte (Flat shading)[17] e nel 1971 Henri Gouraud presentò un

---

[15] Processo che trasforma un oggetto tridimensionale nella sua rappresentazione bidimensionale relativa al punto di vista dell'osservatore. (*http://clonline.infoservizi.it/daniel/gloss3d.html*)

[16] Processo con cui si applicano agli oggetti 3D effetti di illuminazione e ombreggiatura. (*http://clonline.infoservizi.it/daniel/gloss3d.html*)

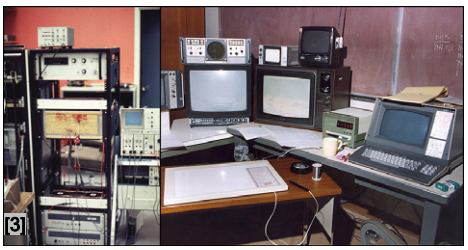
[17] Modello di ombreggiatura secondo cui ogni poligono viene rappresentato con un solo colore che rappresenta la combinazione della luce ambientale e delle caratteristiche cromatiche dell'oggetto. Il risultato è piuttosto scadente in quanto gli oggetti appaiono sfaccettati e lasciano

metodo per fondere tra loro, interplolandole, le tonalità delle facce ombreggiate.

Nel 1973 si ebbe la prima conferenza sulla computer grafica indetta da "Association of Computeng Machinery's" (ACM) "Special Interest Group on Computer Graphics" (SIGGRAPH)

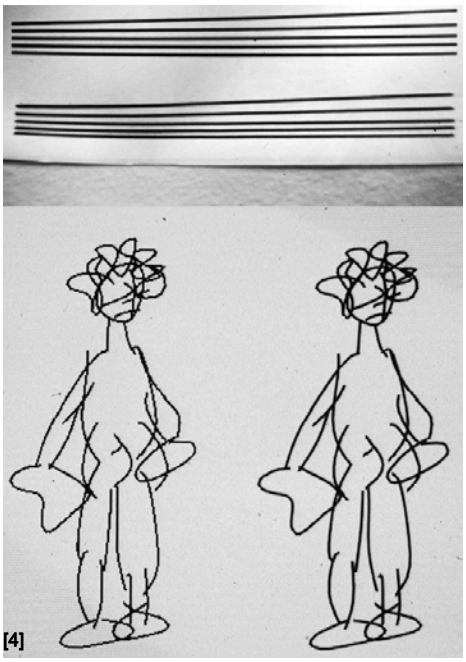
Nel 1974 Phong Bui-Toung corresse e migliorò l'algoritmo di Gourand inventando un nuovo shader. Questo nuovo metodo genera superfici più realistiche, "*calcolando l'interazione della luce ambientale con la superficie dell'oggetto su ciascun punto di ogni poligono*"[18], a discapito della velocità di elaborazione, cento volte più lenta.

Il matematico francese Benoit Mandelbrot, dopo circa venti anni di studio e ricerche, pubblicò nel 1975 un articolo chiamato "Gli oggetti Frattali". Egli elaborò una nuova geometria partendo dalle formule inventate dal matematico G. Julia e successivamente studiate dal metereologo Louis Ray Richardson. Al primo libro Mandelbrot fece seguire un secondo chiamato "La geometria frattale della natura" in cui mostra come i principi della geometria frattale possano essere applicati per la realizzazione di simulazioni realistiche della natura.



[3]

Negli stessi anni Richard Shoup iniziò a lavorare, grazie ai fondi stanziati dalla Xerox, a SuperPaint[3] una sorta di paint avanzato con capacità di acquisizione e manipolazione di immagini bitmap e la possibilità di disegnare linee comprese di effetto antialiasing[19] [4].



[4]

Sempre negli stessi anni Catmull inventò e mise a punto il Texture mapping[20], lo Z-buffering[21] e il rendering di superfici curve.

Sempre nel 1975 William Gates III assieme all'amico Paul Allen fondò una compagnia chiamata Microsoft; scrissero una versione del BASIC e la lanciarono sul mercato. Cinque anni dopo comprarono dalla Seattle Computer Products (SCP) un sistema operativo chiamato 86-DOS, lo riscrissero chiamandolo DOS e lo cedettero in licenza alla IBM che lo utilizzò nei suoi primi Personal Computer.

---

vedere la scomposizione in poligoni.
(*http://www.aessenet.org/glossario_definizione.php/272,0.htm*)

[18] *http://clonline.infoservizi.it/daniel/gloss3d.html*

[19] L'antialiasing approssima i bordi di forme e linee, che essendo composte da pixels apparirebbero scalettate, per farli apparire più lisci e smussati sullo schermo.

[20] Metodo utilizzato per conferire maggior realismo agli oggetti 3D. Consiste nell'applicazione di immagini bitmap dette texture sulle superfici degli oggetti tridimensionali.
(*http://www.aessenet.org/glossario_definizione.php/288,0.htm*)

[21] E' il processo che permette di stabilire in ogni momento quali sono gli oggetti in primo piano e di rimuovere dalla scena le linee nascoste.
(*http://www.aessenet.org/glossario_definizione.php/291,0.htm*)

Nel 1976 la ACM permise, per la prima volta, a dieci aziende, che lavoravano nell'ambito della computer grafica, di esporre i propri prodotti al SIGGRAPH.

Nello stesso anno James Blinn sviluppò un nuovo sistema di Texture Mapping con il quale era possibile generare superfici in rilievo partendo da immagini in scala di grigio; il suo sistema interpretava le aree bianche dell'immagine come sporgenze e le aree nere come solchi; aveva inventato il Bump Mapping.

Catmull scrisse, nel 1977, un tool chiamato "tween" che, partendo da due linee di diversa forma e diversa posizione, creava, per interpolazione[22], i passaggi intermedi; questo tool, assieme ad un altro tool per il disegno e l'acquisizione digitale di schizzi, venne venduto alla Disney. Nacque così il CAPS (Computer Animation Production System). Negli anni successivi Catmull continuò la sua ricerca nell'ambito del 3d fino ad approdare, nel 1979, alla Lucasfilm dove divenne vicepresidente e si mise a capo del dipartimento dedicato alla grafica computerizzata, una vera pietra miliare nella storia della computer grafica. Con la creazione del suddetto dipartimento i ricercatori ebbero a disposizione sia i fondi per la sperimentazione e la creazione di nuovi tools e nuovi sistemi, sia la possibilità di lavorare nell'ambiente cinematografico confrontandosi con problematiche, non più teoriche e speculative, ma reali.

Sempre nel 1977 vide la luce la prima rivista dedicata alla computer grafica chiamata "Computer Graphics World" e che, ancora oggi, rappresenta, con i suoi articoli e recensioni, il punto di riferimento nell'ambito della grafica computerizzata.

Nel 1980 al SIGGRAPH fu presentato un filmato "Vol Libre", realizzato da un programmatore della Boeing Company di Seattle, Loren Carpenter; il filmato mostrava un volo ad alta velocità attraverso montagne generate tramite frattali. Carpenter, che aveva letto il libro di Mandelbrot, era giunto a quel risultato cercando di realizzare degli scenari in cui gli aerei, realizzati da Carpenter per la Boeing, potessero volare. In seguito Carpenter scrisse per la Lucasfilm un motore di rendering chiamato REYES (Renders Everything You Ever Saw) che fu trasformato successivamente nel motore di rendering chiamato RENDERMAN della Pixar.

Steven Lisberger e Donald Kushner (1980) furono ingaggiati dalla Diseny per la realizzazione di Tron; primo film nella storia della cinematografia a contenere trenta minuti di grafica generata al computer.

---

[22] procedimento matematico che consiste nel ricavare nuovi dati da due dati iniziali. Il procedimento è rigoroso solo se si conosce come variano i dati, altresì i dati interpolati sono sì calcolati, ma in pratica "inventati". (*http://digilander.libero.it/ollecram/gloss.htm*)

Sempre nel 1980, anno in cui il personal computer della IBM iniziava a farsi strada nel mondo domestico e nelle piccole aziende, nacque la Silicon Graphics Inc. (SGI). Questa azienda focalizzò le sue risorse nella creazione di computer dalle altissime prestazioni da utilizzare nell'ambito della computer grafica.

Nel 1982 John Walker e Dan Drake assieme ad altri undici programmatori fondarono l'Autodesk Inc. e misero in commercio la prima versione di AutoCAD.

Contemporaneamente Lucas fonda la Industrial Light and Magic (ILM) divisione dedicata agli effetti speciali impiegati nel cinema e generati al computer.

Nel 1982 Tom Brigham presentò al SIGGRAPH un video dove una donna, modellata in 3d, si trasformava in lince. Nacque in questa maniera il Moprhing[23].

Jaron Lanier, nel 1983, assieme ad altri sviluppatori dell'Atari Research Center, inventò il DataGlove. Un guanto che, interpretando il movimento della mano, consentiva di manipolare oggetti 3d all'interno di una simulazione.

Nel 1984 nacque una nuova azienda chiamata Wavefront che iniziò a distribuire il primo programma commerciale per la realizzazione e l'animazione di scene tridimensionali. Infatti, fino ad allora, le compagnie, che lavoravano nell'ambito della grafica tridimensionale, erano state costrette ad avere un team di programmatori che scrivesse per loro i programmi.

Nello stesso anno Cindy Goral, Don Greenberg e altri programmatori della Cornell University pubblicarono "Modeling the Interaction of Light Between Diffuse Surfaces" in cui veniva descritto un nuovo metodo di calcolo della luce all'interno dei rendering chiamato Radiosity. Questo metodo utilizza le stesse formule con le quali viene calcolata la dispersione di calore all'interno degli ambienti e riesce a determinare la quantità di luce riflessa dalle superfici. I motori di rendering precedenti si limitavano al calcolo della sola luce incidente, ovvero calcolavano soltanto la quantità di luce che gli oggetti ricevevano dalla sorgente luminosa.

Sempre nel 1984 la Apple Computer rilascia sul mercato il Macintosh, primo personal computer ad utilizzare un sistema operativo con interfaccia grafica basata su icone e finestre.

---

[23] Passaggio continuo da una forma ad un'altra. Anziché disegnare ogni singolo fotogramma con gli strumenti dell'animazione tradizionale, il computer calcola automaticamente gli spostamenti degli oggetti e le variazioni delle forme da un fotogramma chiave ad un altro. (*http://www.byteman.it/gloss_m.htm*)

Nel 1985 la International Standards Organization (ISO) definisce il primo standard per il CD-ROM (Compat Discs with Read Only Memory) e lo chiama ISO 9660.

Nello stesso anno la Commodore lancia sul mercato l'Amiga, basato su un processore costruito dalla Motorola e compatibile, a livello hardware, con i personal computer della IBM.

Il canadese Daniel Langlois, nel 1986, fonda la Softimage e presenta l'omonimo programma di elaborazione e animazione 3d al SIGGRAPH nel 1988. Negli anni a venire Softimage diventerà lo standard dell'animazione tridimensionale computerizzata in Europa.

Nello stesso anno Jim Henson della Muppet assieme a Brad DeGraf della Digital Productions ebbero l'idea di creare una marionetta digitale, ovvero un modello virtuale in 3d comandato in tempo reale da un Waldo[24]. Nacque così il Motion Capture.

Sempre nel 1986 la Crystal Graphics immise sul mercato TOPAS, uno dei primi animatori 3d di qualità per personal computer.

Nel 1988 la Pixar rilascia la prima versione di RENDERMAN. Più che un motore di rendering è uno standard che descrive tutto quello che un computer ha bisogno di sapere prima di calcolare il rendering finale come gli oggetti, luci, punti di vista, effetti atmosferici etc.. Una volta convertita la scena nel formato RENDERMAN, questa potrà essere calcolata su qualsiasi piattaforma (Mac PC o SGI). In questo modo gli sviluppatori di programmi 3d potevano concentrarsi sullo sviluppo di potenti editor 3d eintegrare Renderman per il calcolo della scena finale. La novità più grande che Renderman introdusse furono gli Shaders, algoritmi che descrivevano e generavano textures complesse che potevano essere utilizzate sia negli effetti atmosferici sia nella definizione dei materiali. Nacquero le Texture procedurali[25].

Nel 1989 sempre la Pixar vince l'Oscar per il corto "Tin Toy", completamente realizzato al computer e renderizzato con Renderman.

Sempre nel 1989 l'Autodesk presenta al SIGGRAPH Autodesk Animator, potentissimo programma di painting e animazione 2d con il quale l'Autodesk iniziò il suo percorso nell'ambito dello sviluppo di programmi multimediali. Divenne in pochi anni uno standard per l'animazione e l'esecuzione di filmati su personal computer.

---

[24] Il Waldo, inventato pochi anni prima dagli scienziati della NASA, era un robot manipolatore comandato a distanza che prendeva il nome dallo scienziato che, in un romanzo di Rober Heinlein, per sopperire alle sue limitate capacità motorie, utilizzava delle protesi a forma di mano, comandate dallo stesso scienziato per mezzo di guanti.

[25] Mappa di texture generata da una funzione matematica, anziché da un'immagine bitmap reale proiettata sulla superficie di un oggetto. (*http://www.graphicax.com/glossario/texture.html*)

Nell'anno successivo l'Autodesk l'ancia il suo primo programma per l'animazione 3d computerizzata chiamato 3d Studio.

Questi sono gli anni in cui vengono prodotti per il cinema film come Abyss e Terminator 2 dove vengono, per la prima volta, realizzati personaggi composti da fluidi (acqua nel primo e metallo nel secondo); Jurassik Park dove vengono inseriti, all'interno di scene girate dal vivo, dinosauri generati al computer di un realismo mai visto prima; Toy story, primo lungometraggio totalmente realizzato al computer; The Mask dove, grazie al motion capture, viene animata realisticamente una maschera sul volto del protagonista.

Nel 1995 la ID software realizza e commercializza Quake, primo videogioco in 3d calcolato in tempo reale su personal computers dalle ridotte capacità elaborative. Da questo momento in poi si svilupperà, nell'ambito del personal computer, un mercato di schede 3d acceleratici destinate non più al mondo industriale e professionale ma a quello domestico.

ASSOCIATIVA, geometria

Sono associative tutte le entità che, una volta apportate modifiche al modello, non perdano eventuali riferimenti ad altri elementi grafici (ad es. parallelismo, coincidenza, riutilizzo di bordi di un solido, ecc.)

BUMP MAPPING

Il bump mapping è un metodo per simulare superfici scabrose, con rilievi (quindi tridimensionali), con textures bidimensionali. Per esempio: se realizziamo un oggetto che rappresenta un tronco di legno, le fenditure della corteccia, grazie al bump mapping, possono apparire tridimensionali anche se non sono realizzate realmente con poligoni. In questo modo, eventuali cambi di illuminazione possono far apparire tali solchi profondi anche se sono assolutamente piatti; grazie al bump mapping si possono, utilizzare molti meno poligoni per disegnare un oggetto 3D.

CAD

1) Computer Aided Design, cioè Progettazione Assistita da Elaboratore

In questa accezione, la più comune, CAD indica il settore dell'informatica volto all'utilizzo di tecnologie software e in particolare della computer grafica per supportare l'attivitá di progettazione (design) di manufatti. I sistemi di Computer Aided Design hanno come obiettivo la creazione di modelli, usualmente 3D, del manufatto. Ad esempio, un sistema Computer Aided Design può essere impiegato da un progettista meccanico nella creazione di un modello 3D di un albero a camme.

2) Computer Aided Drafting, cioè Disegno Tecnico Assistito da Elaboratore

In tale accezione indica il settore dell'informatica volto all'utilizzo di tecnologie software e specificamente della computer grafica per supportare l'attivitá di disegno (drafting). I sistemi di Computer Aided Drafting hanno come obietivo la creazione di un modello, tipicamente 2D, del disegno tecnico che descrive il manufatto, non del manufatto stesso. Ad esempio, un sistema Computer Aided Drafting può essere impiegato da un progettista nella creazione di un disegno tecnico 2D che rappresenta un albero a camme.

CEL SHADING

Tecnica attualmente utilizzata in molti prodotti, permette di "cartoonizzare" un oggetto 3D. Il processo di cartoonizzazione è abbastanza sem-

plice, in quanto si riduce a tracciare la silhouette del modello e illuminare le texture tramite sfumature a bande molto larghe.

FLAT SHADING

Tecnica di ombreggiatura obsoleta, dalla qualita' scadente, visualizza l'obreggiatura di un poligono con un solo colore che rappresenta la combinazione della luce ambientale e delle caratteristiche cromatiche dell'oggetto.

GRAFICA POLIGONALE

Rappresenta la grafica costituita da poligoni, come puo' essere la grafica di un engine 3D (dico puo', perche' in realta' un engine 3D puo' essere costituito anche da curve (quali le NURBS), vedi quake arena).

GUI (GRAPHIC USER INTERFACE)

Interfaccia grafica. Il termine si può riferire a un singolo comando o all'intera interfaccia di un software.

HIDDEN LINE

Visualizzazione wireframe che nasconde le linee in secondo piano. Questa visualizzazione sfrutta in realtà gli stessi poligoni dello shading.

KEYFRAME

I keyframe sono stati utilizzati per la prima volta nei laboratori della Walt Disney. Era un metodo per semplificare le animazioni : un team sviluppava solo i movimenti "chiave" dell'animazione, mentre un team a parte eseguiva le animazioni di passaggio. Nelle animazioni dei giochi 3D funziona in modo simile : i grafici creano solo i movimenti "chiave" mentre il computer gestisce le animazioni di intermezzo tramite interpolazione lineare, quadratica o cubica.

MACRO

Si tratta di una routine che esegue una serie di istruzioni in maniera automatica. Tali istruzioni sfruttano l'interfaccia del programma: una macro esegue una sequenza di comandi che anche un tente potrebbe eseguire. Per questo gli strumenti per realizzare macro prevedono anche le istruzioni di “record” e “play”.

Le macro consentono di ottenere una serie di operazioni con l'invio di un solo comando. Alcuni programmi hanno all'interno la capacità di registrare ed eseguire macro, in alternativa esistono software di automazione che consentono di realizzare macro a livello superiore, e quindi per tutti i

programmi che non le prevedono. Il concetto originario di macro non prevede l'uso di un linguaggio di programmazione, trattandosi di una semplice registrazione di una serie di comandi già disponibile all'interno del software. Evolvendosi nella forma attuale, le macro hanno acquisito un linguaggio di programmazione (es: VBA), con tanto di strutture condizionali (If... Then... Else...), subroutine, dialogo con l'utilizzatore attraverso finestre per l'introduzione di dati... raggiungendo un'efficienza molto maggiore, ma anche maggiore complessità nella loro stesura.

MESH

Un insieme di primitive che descrivono un'immagine tridimensionale.

MODELLATORI PARAMETRICI

I modellatori parametrici eseguono la rigenerazione del modello con un approccio di tipo procedurale, cioè catturano la storia delle operazioni di modellazione e le relazioni parametriche in ordine sequenziale. All'atto della rigenerazione, ricalcolano il modello usando la sequenza definita dei parametri.

MODELLATORI VARIAZIONALI

Nei modellatori variazionali i vincoli geometrici e non geometrici che caratterizzano il modello sono rappresentati da equazioni analitiche; la rigenerazione del modello perciò comporta la risoluzione di un sistema non lineare.

NURBS

(Non Uniform Rational B-Spline) Le NURBS sono potenti interpolazioni di curve che vengono usate nella modellazione per creare oggetti piu' realistici e naturali. Essendo spline garantiscono una continuità di secondo grado quando più NURBS vengono unite. Le NURBS sono interessanti da gestire, perche' grazie agli algoritmi di triangolizzazione permettono il LoD in tempo reale adattando il numero di poligoni alla potenza della machina in cui gira l'engine, ma attualmente vengono man mano sostituite da tecniche più managevoli come le subdivision surfaces.

PHONG SHADING

Il phong shading permette di realizzare un tipo di ombreggiatura altamente realistica; purtroppo data la natura dell'algoritmo piuttosto complessa (usa un'interpolazione non lineare) e' implementabile, in un engine 3D in real-time, solo tramite lightmap o, meglio ancora, pixel shader.

PIXEL

(PIcture ELement) Il pixel e' l'unita di misura delle immagini bitmap 2D ovvero l'elemento piu' piccolo a ciu si puo' fare riferimento. In uno schermo alla risoluzione di 640x480 pixel si troveranno 640 pixel per ogni riga e 480 pixel per ogni colonna, l'intera immagine sara formata da 307.200 pixel.

QUATERNIONI

Sono numeri complessi a tre dimensioni in cui un asse e' reale e gli altri due immaginari. Grazie ai quaternioni possono essere effettuati algortmi di rotazione molto precisi.

RADIOSITY

Sistema di rendering 3D avanzato per la gestione accurata delle luci in un ambiente. Tiene conto delle riflessioni e rifrazioni multiple e al giorno d'oggi richiede ancora un elevato

RENDERING

E', in ambito informatico, il processo di generazione di un' immagine a partire da una descrizione degli oggetti tridimensionali per mezzo di un programma per computer. La descrizione è data in un linguaggio o in una struttura dati definiti rigorosamente e deve contenere la geometria, il punto di vista, le informazioni sulla mappatura delle superfici visibili e sull' illuminazione.

SCRIPT

1) Nel linguaggio dei programmatori, uno script e' un programma o una sequenza di istruzioni che viene interpretata o portata a termine da un altro programma. Nel caso dei programmi grafici il programma che funge da interprete e da esecutore dello script è interno (embedded) al software grafico.

2) Una sequenza di comandi del sistema operativo che sono salvati in un file e che vengono eseguiti in sequenza dall'interprete del sistema quando il nome del file viene inserito come comando.

3) I programmi per sviluppare muldimedia chiamano script una sequenza di istruzioni che si inseriscono per indicare come le sequenze multimediali (o il file multimediale) devono essere presentate (la sequenza delle immagini, dei suoni, i loro tempi e le possibili interazioni dell'utente)

TOPOLOGIA CELLULARE

Caratterizza la possibilità di avere più solidi fra loro disgiunti all'interno dello stesso modello.

VIEWPORT

La viewport e' la "finestra" su cui viene renderizzata una scena 2D o 3D.

WIREFRAME

La codidetta modalita di rendering 3D a "fil di ferro", modalita' esclusivamente geometrica, dove gli oggetti sono rappresentati tramite vertici e angoli, senza fare uso di texture o illuminazione. Utilizzata soprattutto per il CAD.

Bibliografia

- Yuko Hasegawa, *Kazuyo Sejima + Ryue Nishizawa Sanaa*, Electa, Milano 2005.
- Peter Eisenman, *Giuseppe Terragni - Transformations Decompositions Critiques*, The Monacelli Press, New York 2003.
- AA. VV., *Catalogo della mostra: Le divine proporzioni*, a cura di Francesca Folicaldi, Nardini Editore, Firenze 2004.
- D'Arcy Thompson, *On growth and form, Cambridge University Press*, Cambridge 1968.
- Andrea Balzola, Anna Maria Monteverdi, *Le arti multimediali digitali*, Garzanti, Milano 2004.
- AA.VV., *Aut Aut*, n.289-290, rivista bimestrale, Milano 1999.
- AA.VV., *Il disegno dell'architettura fra tradizione e innovazione*, Strumenti del dottorato di ricerca in rilievo e rappresentazione dell'architettura e dell'ambiente, Università degli Studi di Roma La Sapienza, Gangemi editore, Roma 2002.
- AA.VV., *Flash math creativity*, Friends of ED, Birmingham 2002.

Sitografia

- Algorithmic art and Artificial Intelligence *http://iaaa.nl/cursusAA&AI/*
- Self-organising algorithmic architecture *http://angermann2.com/29*
- Armyofclerks *http://www.armyofclerks.net/*
- Flexible Architecture *http://iaaa.nl/rs/autrearcE.html*
- Shape Grammar *http://www.shapegrammar.org/*
- Math in the Media *http://www.ams.org/mathmedia/archive/10-2002-media.html#architecture*
- Michael Hansmeyer, Algorithms in Architecture *http://www.mh-portfolio.com/*
- Kostas Terzidis, Algorithmic Architecture http://kostas.bol.ucla.edu/algorithmicArchitecture.html
- Automi Cellulari *http://www.mat.uniroma1.it/people/mentrasti/automi-cellulari/it/main.html*
- Robert J. Krawczyk http://www.iit.edu/~krawczyk/
- Karl S. Chu *http://www.azw.at/otherprojects/soft_structures/karl_Chu/mainframe_chu.htm http://www.azw.at/otherprojects/soft_structures/allgemein/the_turing.htm*
- Fractals *http://astronomy.swin.edu.au/~pbourke/fractals/*
- L-System Tutorial *http://www.javaview.de/vgp/tutor/lsystem/PaLSystem.html*
- Jose Pinto Duarte, Malagueira *http://www.civil.ist.utl.pt/~jduarte/malag/*
- Artificial Life Links *http://www.alcyone.com/max/links/alife.html*
- Architectural Interpretation of Cellular Automata *http://www.iit.edu/~krawczyk/nks2003/index.html*
- Gallery of computation *http://www.complexification.net/*
- Genicap, Superformula papers *http://www.genicap.com/science/*