2 Digital Flowers

Quanto più a fondo esaminiamo la struttura della Natura, tanti più frattali scopriamo. In effetti la loro onnipresenza nel mondo naturale di cui facciamo farte è una delle ragtioni per le quali ci è così facile trovarli attraenti. Essi rappresentano la forma di computer art che ha colto il programma fondamentale di cui si sono servite le entità viventi per stabilire le proprie nicchie distinte nell'intero corso della storia evolutiva: la riproduzione autosimile di uno stesso motivo in differenti dimensioni.
-- John D. Barrow, L'Universo come opera d'arte.

Digital Flowers

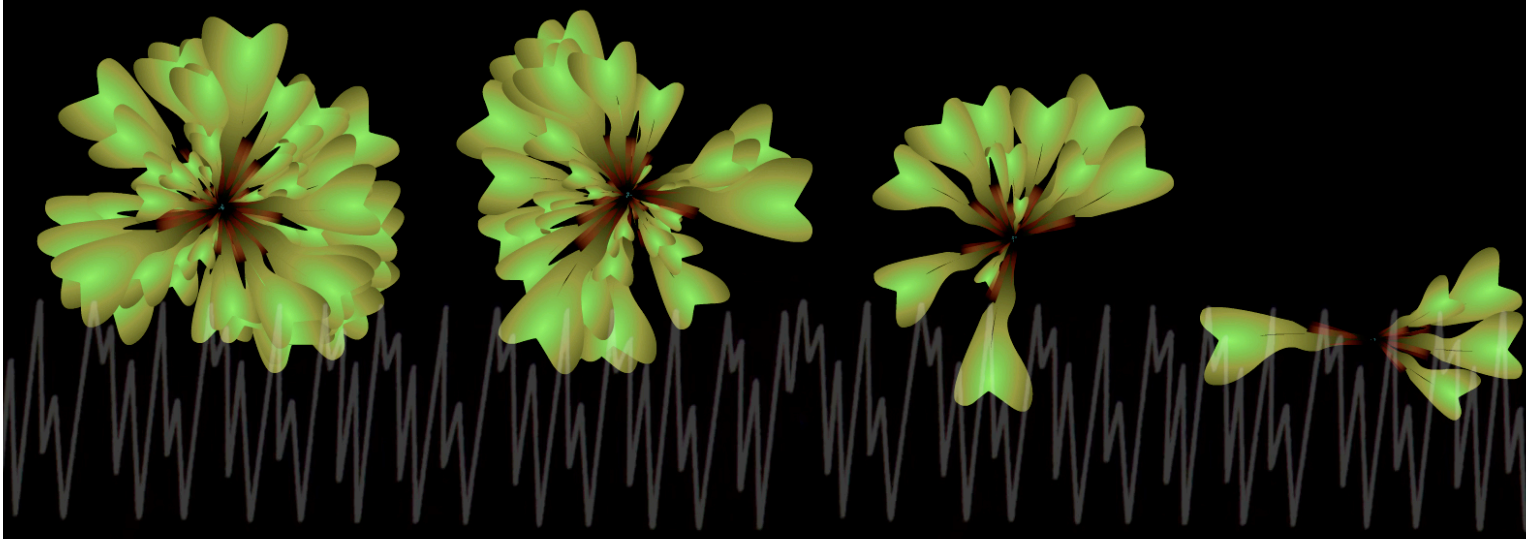
Passeggiare in campo fiorito è un'esperienza che chi sta leggendo ha vissuto almeno una volta nella vita. Indipendentemente dalla posizione geografica un filo rosso unisce i fiori di tutto il mondo, Il colore.
Perchè i fiori sono così colorati?
"*Fiori di differenti colori tendono ad attrarre diversi insetti*"[6] quindi il colore fa parte del meraviglioso ciclo della vita e noi ne siamo spesso inconsapevolmente coinvolti. Attrarre diversi insetti per un fiore è vitale alla sopravvivenza, quindi la Natura ha proceduto secondo un lento provare o repentino salto la sua evoluzione.
Il *digital flower* è la visualizzazione di alcuni dei processi che utilizza la Natura per i suoi stadi di evoluzione o adattamento.
La scrittura del Codice è come dipingere non l'apparenza della Natura ma i suoi processi interni. Svelarne alcuni è il primo passo verso una percezione della realtà più attenta e consapevole.
Quando nel 1999 insieme a due ragazzi torinesi - Luca Barbeni e Seba Vitale - inzio un lavoro di progettazione di ambienti digitali on line, la scrittura del codice era diventato un laboratorio di sperimentazione del rapporto arte, tecnologia e ambiente. Il gruppo di computer art chiamato 80/81 "*decide di confrontarsi con internet come se lo schermo di ogni utente fosse un palcoscenico affacciato su questa realtà in costruzione che è Island.8081, un progetto che intende sviluppare un ambiente in rete secondo le dinamiche (rappresentative e ambientali) dell'ipotetico ecosistema di un isola ideale*".[7]

6 The Color of Nature, Pat Murphy e Paul Doherty

7 WebCinema, Luca Barbeni

City Scape

Our Noise kills the Nature



Il secondo progetto è la scrittura del Codice ActionScript di un digital flower che risponde ai livelli di flusso sonoro perdendo i propri petali.

Our noise kills the Nature

La precedente installazione **Visualizing Urban Noise** mette in relazione il livello del flusso sonoro proveniente dal microfono collegato al competur con le vibrazioni di un visual pattern in un abiente urbano che abbiamo chiamato *City Scape*.

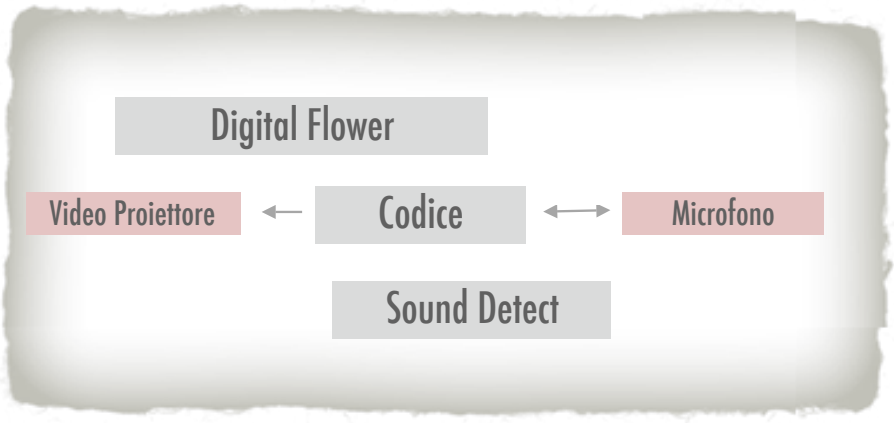
Il secondo progetto è la scrittura del Codice ActionScript di un *digital flower* che risponde ai livelli di flusso sonoro perdendo i propri petali.

Per scrivere il codice dell'installazione **Our noise kills the Nature** faremo un piccolo passo in quel ramo delle scienze informatiche che si dedicano allo sviluppo di *Intelligenze Artificiali*. Quando scriviamo il codice che permette ad una macchina di prendere autonomamente una decisione stiamo *implementando* un pezzo di intelligenza.
Dall'indagine del rapporto interattivo tra uomo e macchina emerge che la *computer art* continuamente ridefiniesce questo rapporto e lo mette in dimensione critica. La nascita del web all'inizio degli anni '90 ha visto un prima onda di **net-artisti** che hanno messo in crisi le nuove tecnologie esaltandone l'estetica dell'errore e del non sense. Nasce proprio in quel background culturale il progetto **Island.8081** proponendo una riflessione sul rapporto tecnologia e natura, cercando di allertare all'accelerazione in corso che vedeva e vede ancora oggi uno squilibrio tra l'utilizzo delle risorse del pianeta terra e la capacità di sopravvivenza di un ecosistema complesso e delicato come quello che regge la vita della Natura.

Se dobbiamo fare un raffronto con le arti del passato, le nuove tecnologie sono paragonabili al teatro greco, alla sua funzione di alfabetizzazione ed educazione ad un nuovo linguaggio che stimola percorsi esplorativi e processi di comprensione della reltà che ci circonda.

L'architettura tecnologica dell'installazione **Our noise kills the Nature** è simile a quella precedente.

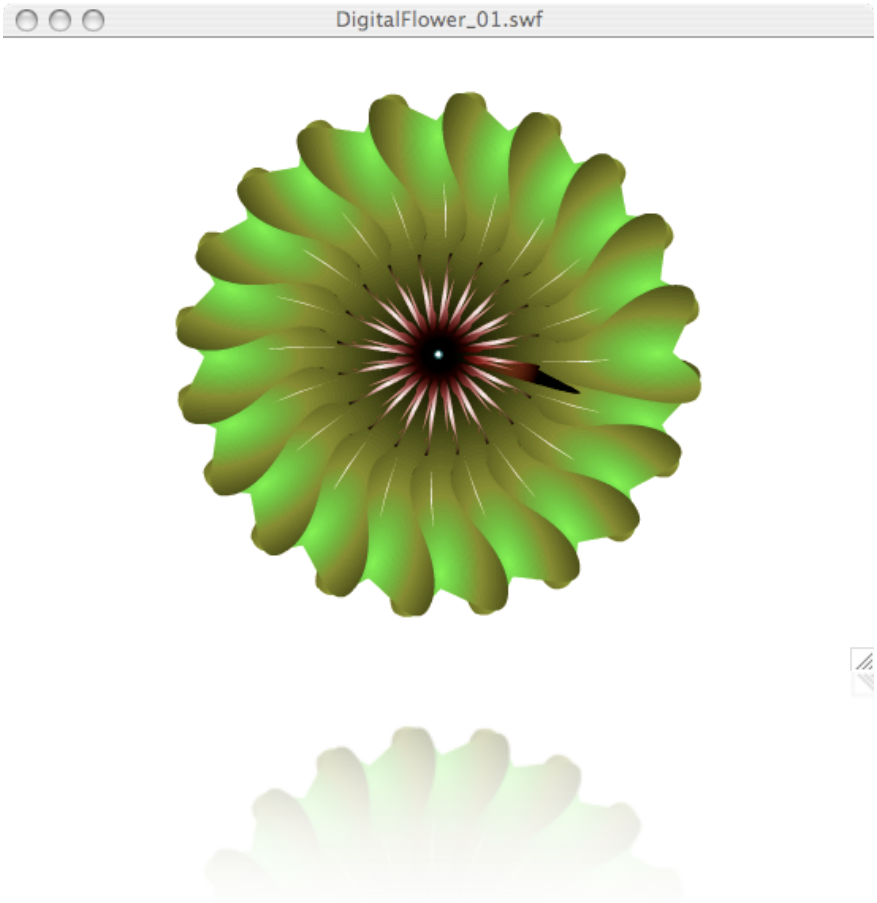
La struttura del Codice vede la scrittura delle funzioni che generano un *digital flower,* ovvero un insieme di petali composti secondo un dato ordine, e il *sound detect* che dovrà decidere quando far perder un petalo al fiore.



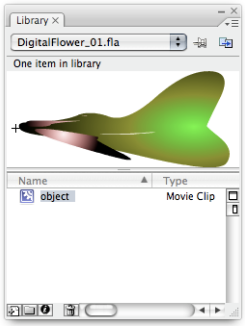
I primi due esercizi sono dedicati alla scrittura del digital flower, mentre nel terzo inizieremo a progettare piccole e semplici *intelligenze artificiali*.

DigitalFlower_01

rotazione + regolarità



Il problema Digital Flower



Disegniamo un petalo di dimensioni 50x250px e lo convertiamo in un simbolo *MovieClip*, istanziandolo con il nome *object*.
Come possiamo disegnare un certo numero di petali al centro dello Stage?

1. **Decidi** il numero di petali, ad esempio 20
2. **Calcola** l'angolo di incremento
3. **Copia** i pentali ruotandoli secondo l'angolo di incremento
4. Quando il digital flower è completo, **termina**.

In quattro passi siamo riusciti a descrivere le azioni necessarie per disegnare un digital flower. Passiamo ad analizzare il codice AS.

```
var stagex:Number = Stage.width;
var stagey:Number = Stage.height;
```

Dopo aver impostato le solite variabili globali, definiamo la variabile *max* che contiene il numero totale di oggetti (petali) che vogliamo disegnare.

```
var max:Number = 8;
```

Se vogliamo che i petali siano *equidistanti* l'angolo di incremento è di 360 gradi diviso *max*.

```
var inc_angle:Number = 360/max;
```

La funzione *digitalFlower()* utilizza un ciclo for per posizionare il numero max di petali. Ad ogni ciclo posiziona il nuovo petalo al centro dello schermo e lo ruota di inc_angle gradi ogni volta.

```
function digitalFlower() {
          for (var i = 0; i<=max; i++) {
                attachMovie("object","object_"+i,_root.getNextHighestDepth());
```

Dopo aver copiato il petalo *object* sullo *Stage* con la funzione *attachMovie,* lo posizioniamo al suo centro visivo che è la metà di *stagex* e *stagey*.

```
                this["object_"+i]._x = stagex/2; // center
                this["object_"+i]._y = stagey/2; // center
```

La proprietà *_rotation* ruota l'oggetto secondo l'angolo - espresso in gradi - assegnato. Il valore di *inc_angle* è il risulato del rapporto tra 360° e il numero max di petali impostato all'inizio. Se il numero di petali *max* è 8, allora l'incremento angolare *inc_angle* vale 45°. Questo valore è moltiplicato per l'indice *i* che va da zero al numero *max* di petali.

```
                this["object_"+i]._rotation = inc_angle*i; // angle
          }
}
```

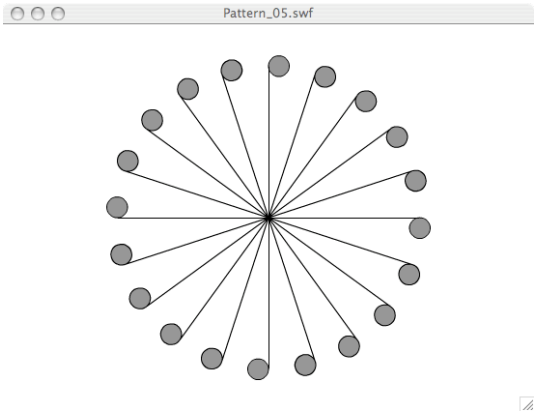
Per ottenere un effetto *realistico* è importante disegnare il petalo con il *punto di rotazione* (+) posizionato alla base del petalo (come mostrato in figura).

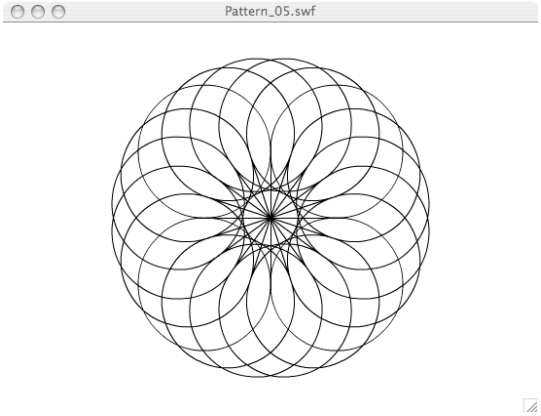
Non dimentichiamoci di chiamare la funzione alla fine del nostro codice altrimenti non succede nulla!

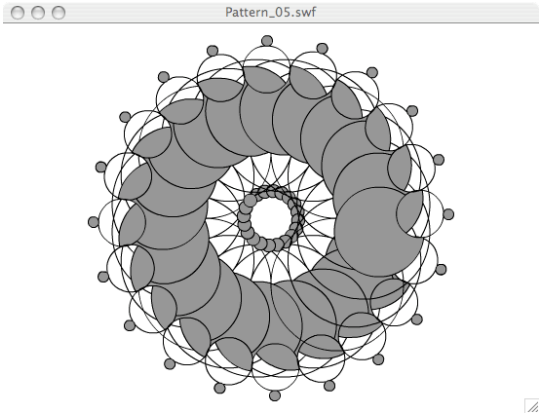
```
digitalFlower();
```

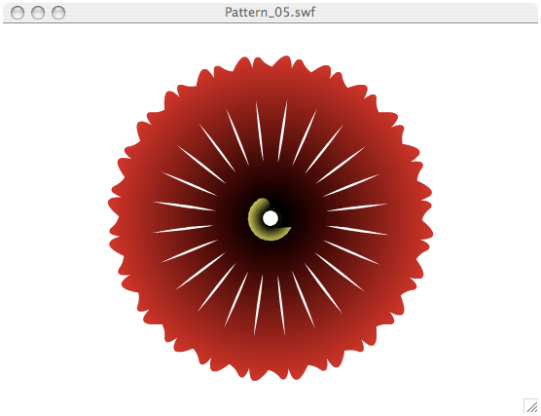
Esercizio per la mente

Sperimentare diverse quantità di petali, e diverse forme rispetto al punto di rotazione.


Pattern_05.swf

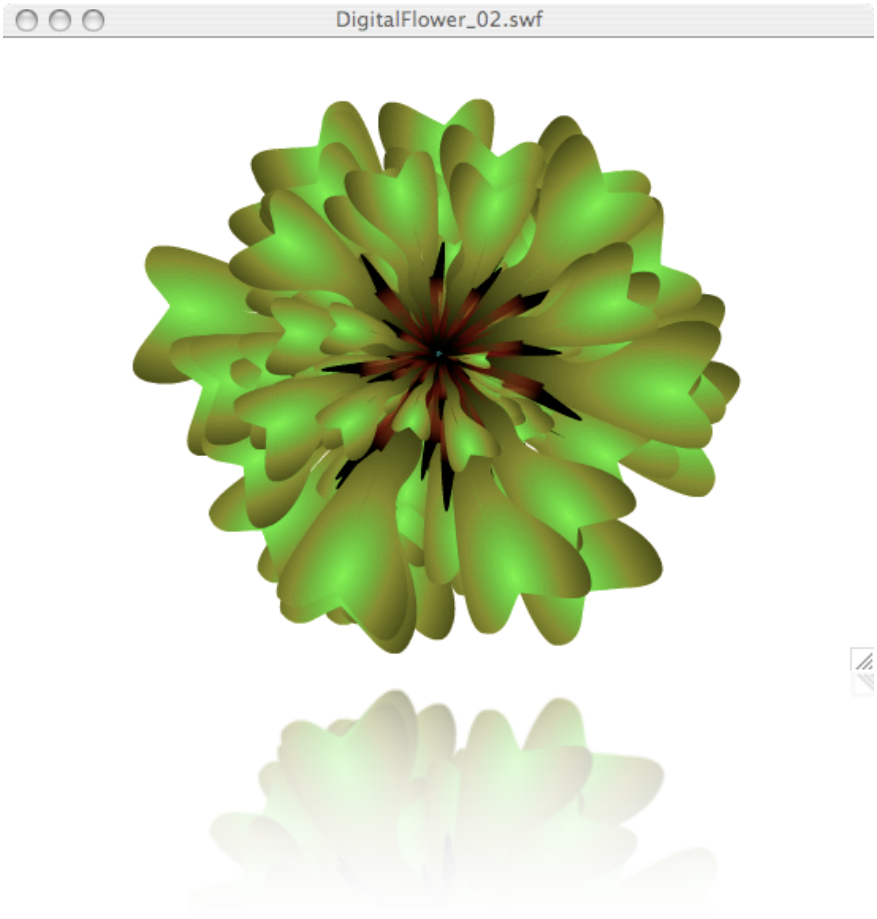

Pattern_05.swf


Pattern_05.swf


Pattern_05.swf

DigitalFlower_02

rotazione + casualità



Introducendo nel codice precedente delle funzioni che variano casualmente l'angolo di nascita e la dimensione di un petalo, otteniamo la *diversità delle forme* che caratterizza la bellezza dei fiori.

Il codice rimane identinco al precente salvo piccole modifiche nella funzione *digitalFlower*.

La funzione *digitalFlower()* posiziona ogni nuovo petalo secondo un angolo di rotazione *r* e un valore di scala *s* (stesso valore per *_xscale* e *_yscale*) generati casualmente.

```
function digitalFlower() {
          for (var i = 0; i<=max; i++) {
                attachMovie("object","object_"+i,_root.getNextHighestDepth());
                this["object_"+i]._x = stagex/2;// center
                this["object_"+i]._y = stagey/2;// center
```

La rotazione *r* di ogni nuovo petalo non è più legata ad un angolo di incremento unico - come nel precedente esercizio - ma ad ogni ciclo l'angolo è deciso dalla funzione *random()*.

```
// flowers irregular
var r = random(360);
this["object_"+i]._rotation = r;
```

Lo stesso vale per il fattore di scale *s* che scelto casualmente produrrà petali di diverse dimensioni. Assegnando lo stesso valore alle proprietà *_xscale* e *_yscale* si ottiene un ridimensionamento omogeno.

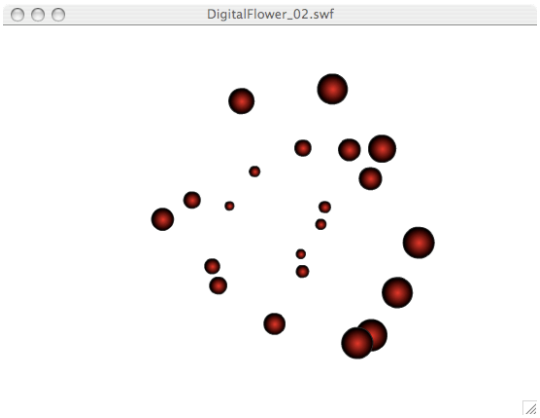
```
var s = random(100)+20;
this["object_"+i]._xscale = s;
this["object_"+i]._yscale = s;
```

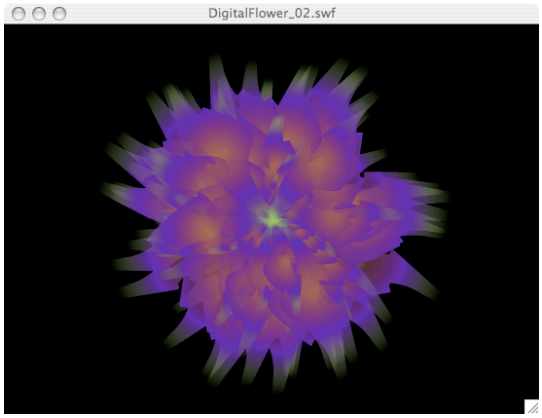
La funzione *swapDepths* pemettere di posizionare gli oggetti *MovieClip* su un dato livello virtuale.
Questo codice permette di mettere in primo piano i petali più piccoli e sullo sfondo i petali più grandi.

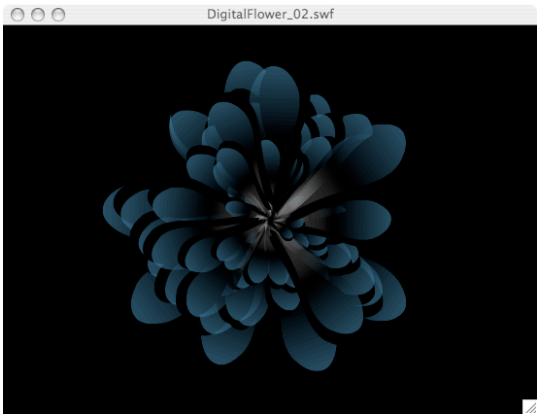
```
                this["object_"+i].swapDepths(2000-this["object_"+i]._xscale);
            }
    }
```

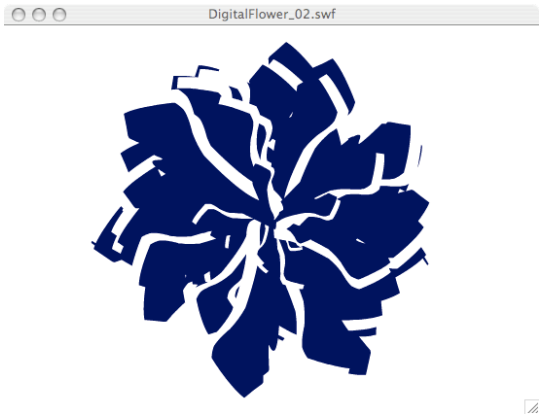
Esercizio per la mente

Sperimentare diverse quantità di petali, e diverse forme rispetto al punto di rotazione.


DigitalFlower_02.swf

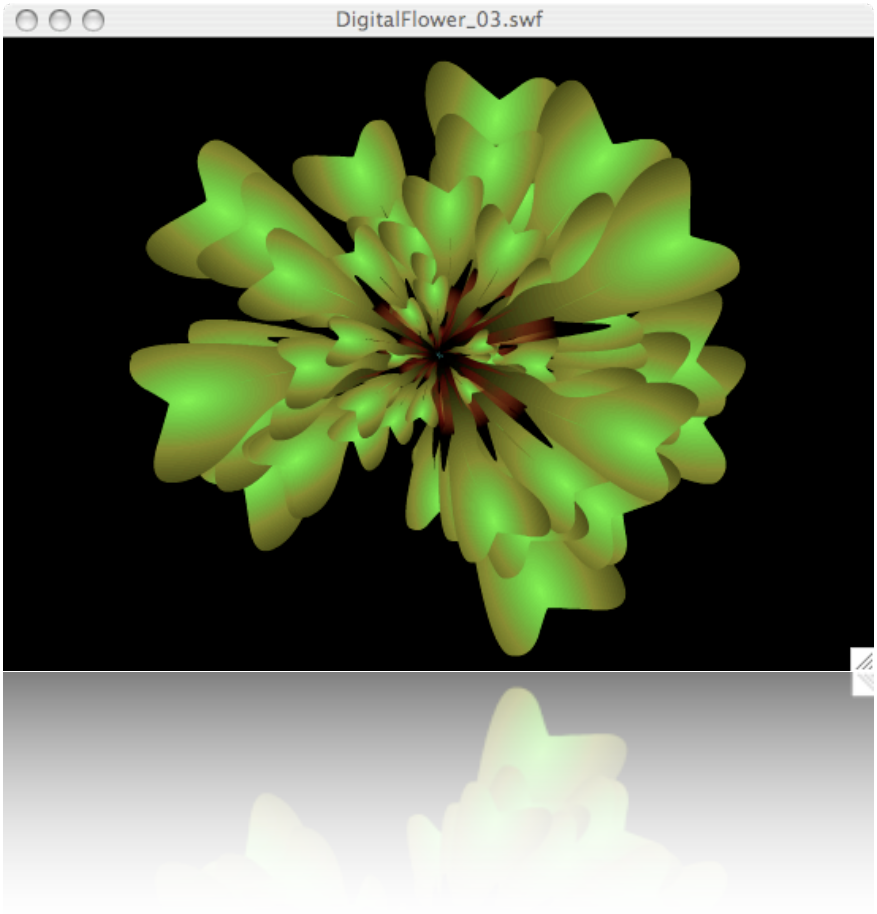

DigitalFlower_02.swf


DigitalFlower_02.swf


DigitalFlower_02.swf

DigitalFlower_03

rotazione + interattività



In questo esercizio concludiamo il codice dell'installazione **Our noise kills the Nature**. Fin qui abbiamo implementato la funzione *digitalFlower()* che una volta chiamata disegna un fiore digitale a partire da un dato numero di petali scelti da noi e posizionati secondo dimensioni e angoli di rotazione dipendenti dai numeri casuali generati dalla funzione *random()*.
La struttura dell'installazione è composta dalla funzione *digitalFlower()* che disegna un fiore digitale e dalla funzione *soundDetect()* che sceglie quando far perdere un petalo al nostro fiore.
Cosa succede quando il fiore ha perso tutti i petali?
Questa domanda ci porta a riflettere sul senso dell'installazione che va pensata come oggetto che non ha bisogno di assistenza continua, ma ha codificate le cosidette *condizioni di ritorno*, alla base dei primi esperimenti di Intelligenza Artificiale.
La risposta naturale alla domanda è quindi un ritorno allo stato iniziale con tutti i petali in cui l'installazione si rimette in ascolto del flusso sonoro e fa perdere un petalo quando si raggiunge un dato picco sonoro. Per implementare una condizione utilizziamo l'istruzione *if* che verifica una condizione e sceglie quale parte di codice eseguire.

if

```
if(condition) {
statement(s);
}
```

Valuta una condizione per determinare l'azione successiva da eseguire in un file SWF. Se la condizione restituisce true, Flash esegue le istruzioni che seguono la condizione all'interno delle parentesi graffe ({}). Se la condizione è false, Flash ignora le istruzioni all'interno delle parentesi graffe ed esegue le istruzioni dopo le parentesi. Utilizzare le istruzioni if, else e else if per creare una logica ad albero negli script.

Il Codice digitalFlower

La funzione *digitalFlower()* implementata nell'esercizio precedente è qui ripresa nella sua interezza. Il primo passo è disegnare il fiore digitale.

```
var stagex:Number = Stage.width;
var stagey:Number = Stage.height;
var max:Number = 100;

function digitalFlower() {
          for (var i = 0; i<=max; i++) {
                attachMovie("object","object_"+i,_root.getNextHighestDepth());
                this["object_"+i]._x = stagex/2;
                this["object_"+i]._y = stagey/2;


                // angolo casuale
                var r = random(360);
                this["object_"+i]._rotation = r;

                // dimensione casuale
                var s = random(100)+20;
                this["object_"+i]._xscale = s;
                this["object_"+i]._yscale = s;

                // il più piccolo in primo piano
                this["object_"+i].swapDepths(2000-this["object_"+i]._xscale);

          }
}

digitalFlower();
```

Il Codice soundDetect

Abbiamo affrontato il problema del *soundDetect* nella precedente installazione e abbiamo scoperto che ActionScript ha già implementata una Classe *Microphone* che aggancia e monitorizza il flusso sonoro proveniente da un microfono collegato al computer.
Ecco il codice per agganciare il microfono ad ActionScript.

```
var my_mic = Microphone.get();
_root.attachAudio(my_mic);
my_mic.setGain(90);
var level = 0;
```

Il metodo *this.onEnterFrame* avvia il monitoraggio della variabile *level* sulla proprietà *activityLevel* della classe AS *Microphone*.

```
this.onEnterFrame = function() {
          level = my_mic.activityLevel;
          soundDetect();
};
```

Nell'installazione precedente **Visualizing Urban Nois***e* la funzione *soundDetect* applicava la variabile sensibile *level* come moltiplicatore delle proprietà di scala, trasparenza e rotazione di ogni modulo del pattern visivo.
Nell'installazione **Our noise kill the Nature** vogliamo che quando il flusso sonoro raggiunge un certo livello di attività, il fiore digitale **perde un petalo**. Possiamo tradurre in codice AS l'azione di *perdere un petalo* con l'azione di mettere a zero la proprietà di trasparenza - *_alpha* - di un petalo *i* scelto a caso tra tutti i *max* petali disegnati dalla funzione *digitalFlower()*.

Definiamo la variabile *max_level* uguale ad un numero da 0 a 100 che indica il picco di intensità sonora.

```
var max_level:Number = 10;
```

La funzione soundDetect chiamata a run time verifica la *condizione if,* ovvero se la variabile *level* è maggiore del valore di *max_level* allora esegue il codice tra le parentesi graffe.

```
function soundDetect() {
          //
          if (level>10) {
```

Queste due righe di codice implementano l'azione di *perdere un petalo* semplicemente ponendo a zero la proprietà di trasparenza - *_alpha* - dell'oggetto *i* scelto a caso dalla funzione *random(max)*.

```
                var i = random(max);
                this["object_"+i]._alpha = 0;
          }
```

Infine la funzione *soundDetect* chiama la **condizione di ritorno**.

```
          returnCondition();
}
```

La condizione di ritorno

Abbiamo deciso che la condizione di ritorno per questa installazione è verificata quando tutti i petali sono scompasi. Quando la condizione è verificata riporta a 100 il valore della proprietà di trasparenza di tutti i *max* petali del fiore digitale.

La funzione *returnCondition()* utilizza la variabile *count* per contare il numero di petali trasparenti. Quando il numero di petali trasparenti è uguale al numero max dei petali del fiore digitale allora il fiore è scomparso e bisogna riaccendere tutti i petali.

```
function returnCondition() {
```

Definiamo la variabile count uguale a zero.

```
        var count = 0;
```

Con un ciclo for contiamo i petali spenti, ovvero con la proprietà di trasparenza uguale a zero.

```
        for (var i = 0; i<=max; i++) {
              if (this["object_"+i]._alpha == 0) {
                    count++;
              }
```

Utilizziamo il ciclo for applicare il valore di level come moltipicatore della proprietà di ortazione di ogni petalo, ottenendo un effetto di rotazione legato al flusso sonoro.

```
              this["object_"+i]._rotation += ((level/5)+0.5);
        }
```

Infine la *condizione if* verifica quando tutti petali sono scomparsi, attraverso il confronto tra il valore di *count,* risultato dal conteggio effettuato nel ciclo for precedente, e il valore *max* che contiene il numero totale di petali.

```
        if (count == max) {
```

Quando la condizione è soddisfatta un ciclo for scandisce tutti i petali e re-imposta la proprietà di trasparenza uguale a 100.

```
                for (var i = 0; i<=max; i++) {
                        this["object_"+i]._alpha = 100;
                }
        }

}
```

Non ci resta che mandare in esecuzione il filmato e immaginarlo proiettato su un'architettura urbana con i microfoni avvessi sull'ambiente sonoro circostante.

Esercizio per la mente

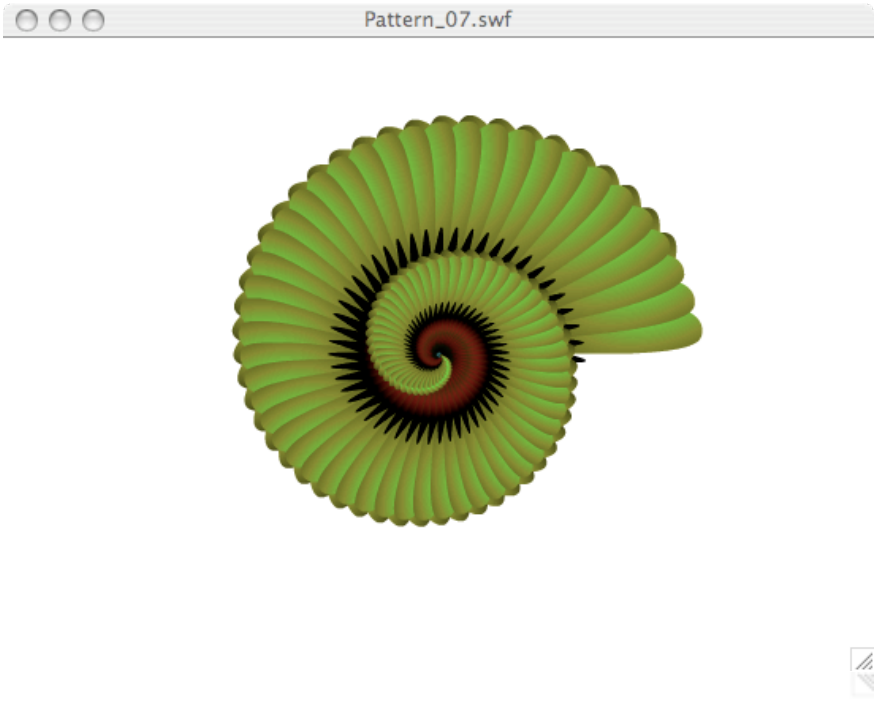
Sperimentare diverse condizioni di ritorno e sensibilità del flusso sonoro. Utilizzare il videoproiettore.


Our noise kills the Nature


Extended
Digital Flowers

Spirale

rotazione + incremento



Oltre alla casualità la natura ha uno strumento magico conosciuto come *incremento*, ovvero un valore di crescita applicabile sia all'angolo di rotazione che alla proprietà di scala.

Introduciamo nella funzione **makePatternRotational**() due nuove variabili s e r che iniziano da un valore nullo e crescono di un fattore inc_s e inc_r ad ogni ciclo.

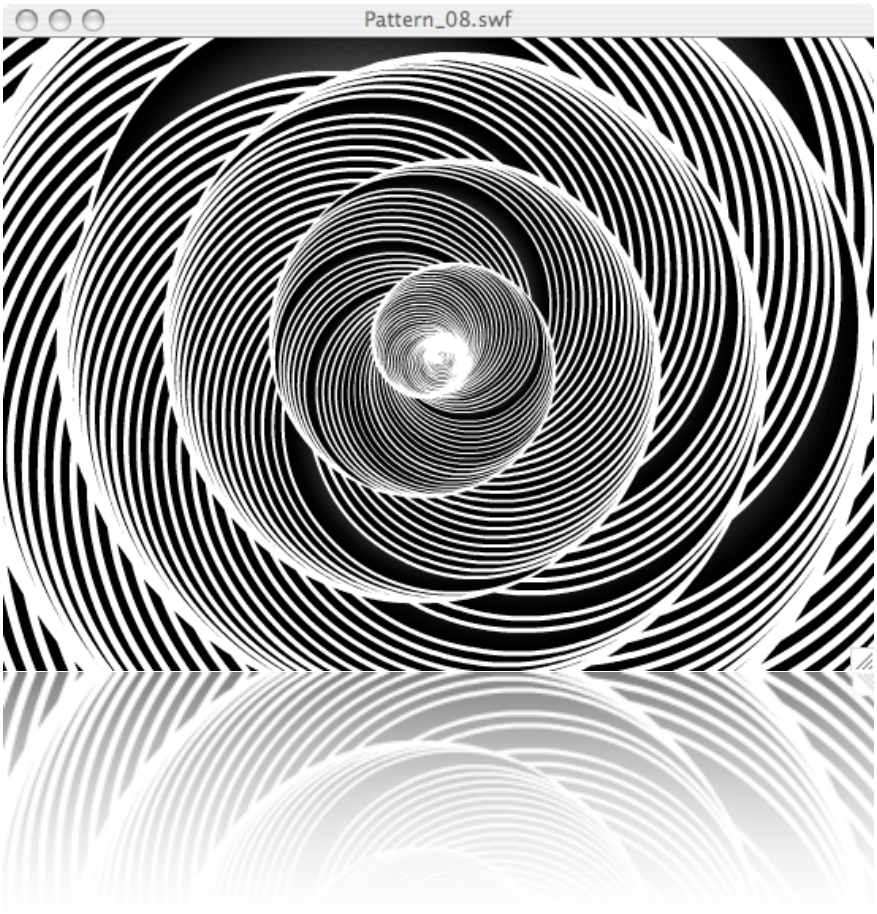
```
            var s = 0;
            var r = 0;
            var inc_r = 7;
            var inc_s = 1;
            for (var i = 0; i<=max; i++) {
                  attachMovie("object","object_"+i,_root.getNextHighestDepth());
                  this["object_"+i]._x = stagex/2;// center
                  this["object_"+i]._y = stagey/2;// center
                  this["object_"+i]._rotation = r += inc_r;
                  this["object_"+i]._xscale = this["object_"+i]._yscale=s += inc_s;

                  this["object_"+i].swapDepths(2000-this["object_"+i]._xscale);
            }
}
```

Esercizio per la mente

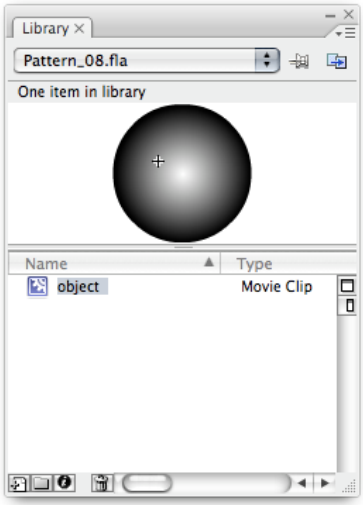
Sperimentare diversi valori di incremento e numero di petali.

Optical Art

rotazione + movimento



Se il nostro petalo è sostituito dal disegno di un cerchio otteniamo un effetto ottico che associato al movimento rotatorio ci riporta alle sperimentazioni di optical art dei primi del '900.
Aggiungiamo al codice precedente le funzioni che permettono di ruotare l'intera composizione ad una velocità stabilita definita dalla variabile *velocity*.



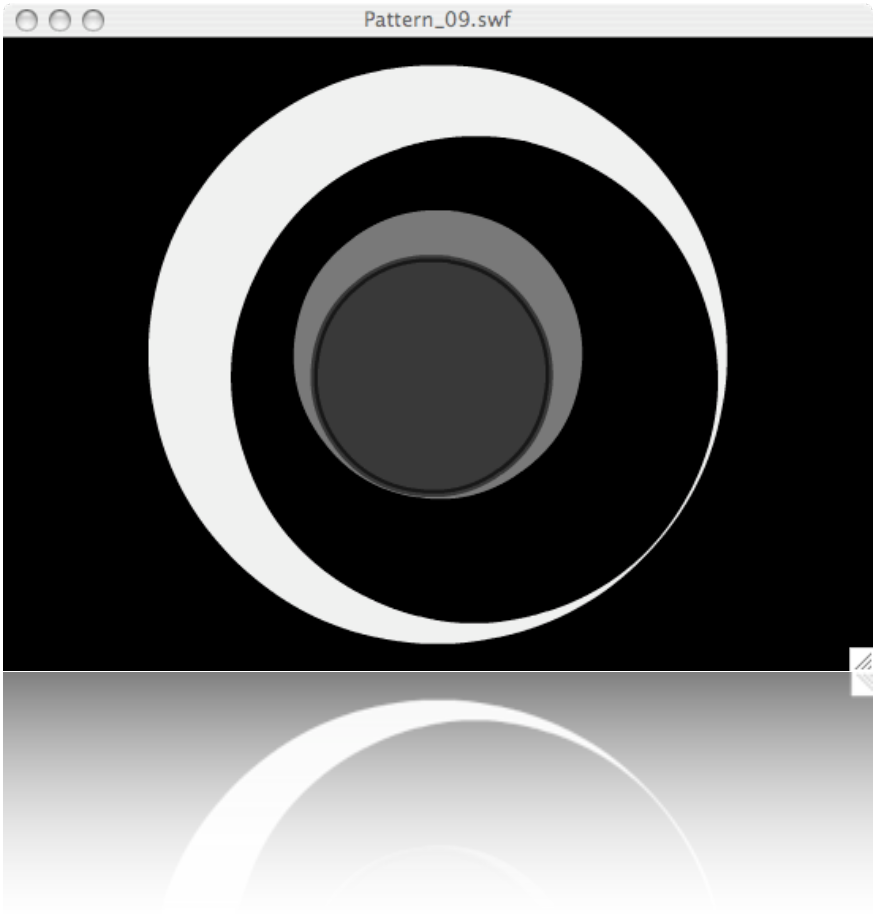

```
var velocity = 1;
function rotateObject(i) {
          this["object_"+i]._rotation +=
velocity;
}

function rotate() {
          for (var i = 0; i<=max; i++) {
                rotateObject(i);
          }
}

idInterval = setInterval(rotate, 1);
makePattern();
```

Esercizio per la mente

Sperimentare forme morbide e colorate dell'oggetto di base. Provare diverse velocità di rotazione variando il valore di *velocity* e i millisecondi della funzione **setInterval**.

Anemic Cinema 1.0

movimento + interattività



Questo esercizio è un omaggio al lavoro video di Marcel Duchamp, *Anemic Cinema* del 1926.

```
var stagex:Number = Stage.width;
var stagey:Number = Stage.height;
var max:Number = 10;
var inc_angle:Number = 360/max;
var velocity:Number = 1;

function makePattern() {
          for (var i = 0; i<max; i++) {
                attachMovie("object","object_"+i,_root.getNextHighestDepth());
                this["object_"+i]._x = stagex/2; // center
                this["object_"+i]._y = stagey/2; // center
```

Utilizziamo un incremento angolare regolare.

```
                this["object_"+i]._rotation = inc_angle*i; // angle
                rotateObject(i);
          }
}
```

All'interno dell'oggetto scriviamo la funzione **fadeOut**() che viene esegiuta ogni volta che il mouse passa sopra un oggetto. Utilizza la classe Tween per dissolvere e rimpicciolire ogni oggetto.

```
function fadeOut() {
          this.tween("_alpha",50,10,"easeOutSine");
          this.tween("_xscale",50,10,"easeOutSine");
          this.tween("_yscale",50,10,"easeOutSine");
}
this.onRollOver = fadeOut;
```

Esercizio per la mente

Provare diversi valori della funzione **fadeOut** (interna all'oggetto).