

COMPUTER

GRAPHICS

 & PUBLISHING

Keith Peters è fin troppo celebre per aver bisogno di una presentazione. Chiunque nel mondo abbia dedicato un po' del suo tempo alla ricerca di risorse gratuite su *Flash* nel Web conosce il suo prezioso sito BIT-101 (www.bit-101.com), dove sono raccolte le sue sperimentazioni quotidiane. Inoltre, moltissime persone, in America e non solo, hanno studiato su manuali parzialmente scritti da lui ed editi da Friends of ED (www.friendsofed.com).

Ciò che è meno noto è la sua storia, affascinante perché dimostra quanto il coraggio e la tenacia uniti al talento e alla passione portino alla realizzazione dei propri sogni. Keith, al di là dell'impressione che possono dare i complessi codici contenuti nei suoi lavori, è tutt'altro che lo stereotipo del

imparare più approfonditamente la programmazione al computer. Si procurò quindi un vero PC e un compilatore C++ Borland, ma nel giro di pochi mesi abbandonò l'argomento, in preda a una confusione interiore che lo accompagnò per alcuni anni. Non era molto sicuro di cosa volesse fare e in quel periodo turbinoso rinunciò all'arte e alla programmazione. Il suo interesse per il computer, però, rimase vivo e gli permise di continuare ad avventurarsi alla scoperta di qualche nuova tecnologia, a installare e provare nuovi programmi, a sistemare o aggiornare il sistema operativo e ovviamente ad aggiustare i computer di amici e colleghi.

L'incontro con Flash

Un giorno, nel 1999, un amico chiese a Keith se avesse mai sentito parlare di *Flash*. Keith immediatamente scaricò e provò la demo di *Flash*, allora giunto alla ver-

Keith Peters ha trovato in Flash lo strumento per instillare la vita negli oggetti grafici, utilizzando in maniera creativa la fisica e la matematica: ispirazioni e creazioni di un sostenitore dell'open source

Matematica creativa

programmatore: le sue opere dimostrano anzi che la matematica e le scienze esatte sono attività divertenti a servizio di chi voglia plasmare un prodotto creativo.

Un esordio da disegnatore

Keith è sempre stato bravo a disegnare. Da bambino copiava i fumetti di *MAD Magazine* o di altri libri, mentre da adolescente faceva i ritratti delle sue rock star preferite, utilizzando circa dieci tipi diversi di pennarelli. Tutto gli suggerirono di frequentare una scuola d'arte o di cercare di fare dell'arte il suo mestiere, ma per qualche ragione questa scintilla in Keith non scoccò. Il disegno rimase quindi per lui un hobby finché, quando aveva circa 25 anni, fu molto ispirato da alcuni disegni a penna e inchiostro che vide. Da allora, crebbe in lui un'autentica passione, che si tradusse anche in una professione: cominciò infatti a produrre illustrazioni per riviste.

All'inizio degli anni Novanta, cominciò a prendere confidenza con i computer, sperimentando programmi di 3D e di grafica, ma in quel tempo e a quell'età non poteva permettersi l'hardware e il software necessari per diventare un professionista (un hard disk da 20 MB costava 300 dollari). Keith quindi cominciò a scrivere semplici programmi in *Basic* sul suo Commodore Amiga. Per lui era davvero emozionante riuscire a creare figure e farle muovere e interagire. In quel periodo realizzò alcuni giochi molto carini e questo lo stimolò a



sione 4, ma non ne fu molto impressionato. Il suo amico, invece, ne era veramente entusiasta e in qualche modo era riuscito a farsi affidare lo sviluppo di un progetto in *Flash*, in cui coinvolse, nonostante tutto, Keith. A questo primo incarico ne seguirono molti altri, che furono una rampa di lancio. In quel periodo, Keith imparò moltissimo su *Flash*, ma per lo più si limitava ancora a realizzare semplici interpolazioni e azioni per i pulsanti. Il suo desiderio di andare più in profondità coincise fortunatamente con il rilascio del-

► Nelle immagini di queste pagine: alcuni esperimenti di Keith Peters visibili su www.bit-101.com



gliori flasher del mondo e di diventare famoso grazie alla possibilità che il Web offre a chiunque di esibire le proprie capacità. In questo modo, sperava, avrebbe ottenuto un lavoro solo grazie alla sua reputazione.

La tenacia e il suo impegno nello studiare la materia sono stati premiati: oggi Keith è davvero famoso, ha vinto la sezione sperimentale del Flash Forward Film Festival 2003, è Beta Tester ufficiale di *Flash* per la Macromedia, ed è coautore di diversi libri quali *Flash Math Creativity* (Friends of ED, 2002), *Flash MX Studio* (Friends of ED, 2002), *Fresh Flash* (Friends of ED, 2002), *Flash MX Most Wanted*, (Friends of ED, 2003), *Byte Size Flash: Adventures in Optimization* (Friends of ED, 2003) e *Flash MX Games Most Wanted* (Friends of ED, 2003). Inoltre, ha trovato un lavoro che gli piace molto presso un'azienda di design di Boston, <http://www.takeexit33.com>, dove programma in *Flash* tutto il giorno.

Persino quando diventa stressante, Keith dichiara senza esitazioni che è sempre il lavoro dei suoi sogni. Keith ama moltissimo programmare, attività che considera, senza ombra di dubbio,

creativa. Quando è concentrato sulla programmazione, la giornata trascorre senza che lui se ne renda conto.

I lavori di Keith dimostrano come le formule matematiche e le leggi fisiche, apparentemente noiose e astratte per i più, possano trovare applicazioni altamente stimolanti. Tutti gli studi che Keith ha fatto e continua a fare sulla programmazione in *Flash* sono pubblicati sul suo sito BIT-101, probabilmente la più grande raccolta di lavori *Flash* di un solo autore esistente sul Web. Nell'ideazione di BIT-101, Keith è stato ispirato soprattutto

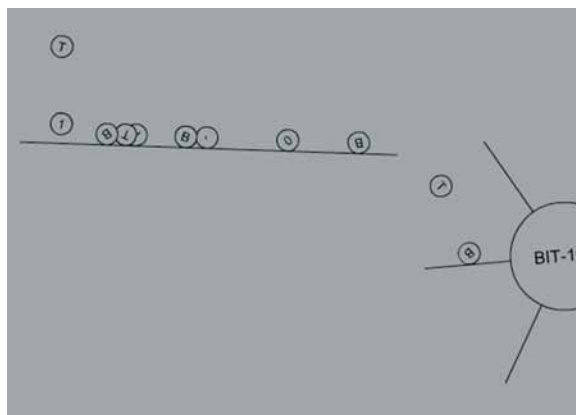
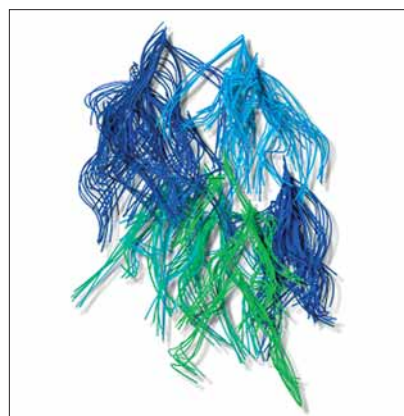
► segue a pagina 20

la versione 5 di *Flash*. Keith si accorse subito che l'*ActionScript* era davvero maturato e si cominciarono a vedere in Rete lavori davvero interessanti. Si sentì riportato ai giorni in cui programmava i giochi sull'Amiga e fu rapito dal desiderio d'imparare bene questo linguaggio diventato così potente. Comprò quindi *Foundation ActionScript*, manuale di Sham Bangal pubblicato da Friends of ED, e si sentì coinvolto come non lo era mai stato prima.

La nascita di BIT-101

Nel frattempo, il boom del "dotcom" ebbe fine. Tutti gli amici di Keith avevano abbandonato completamente *Flash* per dedicarsi ad al-

tro. Keith rimase disoccupato e, mentre cercava un lavoro, studiava e imparava tutto quel che poteva dell'*ActionScript*. Alla fine del 2001, rinunciò a trovare un lavoro e si lanciò negli esperimenti realizzati con l'*ActionScript*. Così nacque il suo sito, BIT-101 (www.bit-101.com). Dedicarsi completamente a questa attività lo convinse a non piegarsi ad alcuna sorta di lavoro noioso, che lo impegnasse l'intera giornata impedendogli di occuparsi di quel che più gli piaceva. L'unico tipo di mestiere che poteva pensare per se stesso doveva avere a che fare con i computer e la programmazione, ma Keith non aveva un titolo, una formazione formale o un'esperienza significativa nel mondo del lavoro. Decise quindi di dedicarsi con tutto il suo impegno all'*ActionScript*, coltivando il sogno di diventare uno dei mi-



ARM: tutorial di Barbara Sansone basato sul file di Keith Peters "031222 fla"

Per seguire questo tutorial (**Figura 1**), aprite il file 031222 fla, contenuto nella cartella Shared/Supporto (oppure "Materiali", tramite l'interfaccia) del CG&P-CD 14 allegato a questo numero della rivista (oppure scaricatelo alla pagina Internet www.iht.it/supporto2.htm). Tutto il codice è raccolto in uno script allegato al frame 1 dell'unico livello esistente sulla timeline.

Innanzitutto si deciderà di quanti pezzi dev'essere composto il braccio. Questo valore può essere cambiato in qualsiasi momento, per cui è conveniente conservarlo all'interno di una variabile dichiarata all'inizio dello script:

```
num = 10;
```

I pezzi non saranno tutti uguali come dimensioni rispetto al simbolo originale, si deve decidere un fattore di scala che sarà incrementato a mano a mano che l'ActionScript disegnerà le istanze. Anche questo valore, per consentire facili variazioni successive, dev'essere conservato in una variabile:

```
scale = 30;
```

Le azioni necessarie per disegnare le istanze e attribuire loro i movimenti, con qualche accorgimento, sono uguali per tutti i pezzi. È necessario quindi utilizzare un ciclo che si avvalga del valore di num:

```
for(var i=0;i<num;i++){
```

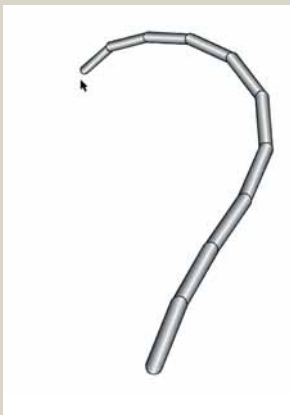
La prima azione del ciclo consiste nel creare l'istanza di un simbolo. La movieClip arm è stata opportunamente corredata di un ID necessario per l'esportazione ActionScript, come si può notare dalla proprietà Linkage reperibile dal menu delle opzioni della Library (**Figura 2**). Le istanze verranno nominate progressivamente "a0", "a1" e così via, aggiungendo alla stringa "a" il valore corrente di i:

```
arm = attachMovie("arm", "a"+i, i);
```

L'istanza creata nel ciclo si chiamerà per semplicità arm. Il valore di questa variabile sarà, a ogni ciclo, il nome dell'istanza appena creata. Ora all'istanza si attribuiscono fattori di scala x e y pari al valore di scale:

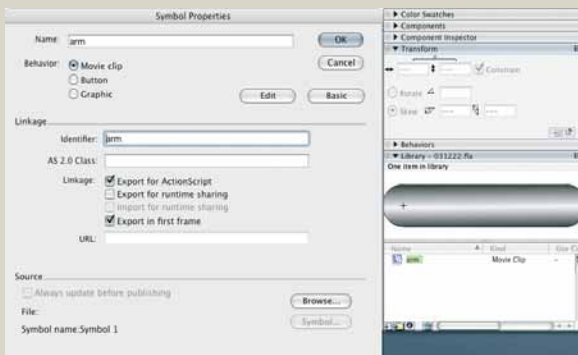
```
arm._xscale = arm._yscale = scale;
```

Nella variabile w all'interno dell'oggetto si conserverà un valore pari a una determinata percentuale della lunghezza dell'oggetto (qui è l'80%, ma



► **Figura 1:** il braccio che segue il puntatore nel tutorial Arm

► **Figura 2:** la proprietà Linkage di Arm



può essere modificata per verificarne l'effetto). Questo permetterà di regolare la distanza tra i vari pezzi del braccio:

```
arm.w = arm._width * .8;
```

Il fattore scale deve incrementarsi, per far sì che il pezzo creato al ciclo successivo sia più grande. Questo valore può essere modificato per creare oggetti dall'aspetto diverso:

```
scale +=3;
```

Tutte le azioni che regoleranno la posizione e la rotazione dell'istanza in oggetto possono essere raccolte in una funzione che sarà riportata successivamente nello script. Qui sarà sufficiente chiamarla:

```
arm.onEnterFrame = move;
```

I pezzi si dovranno condizionare a vicenda: il primo infatti seguirà il mouse, ma la sua posizione e rotazione influenzerà quella del pezzo successivo. È quindi utile creare la variabile parent (da non confondersi con _parent), che risiederà nell'istanza che avrà il nome con il numero inferiore e avrà il valore dell'istanza in corso. In parole più semplici, l'istanza "a0" conterrà la variabile parent che avrà valore "a1". In questo modo, durante le azioni relative a "a0", si potrà usare la variabile parent per far riferimento al pezzo successivo del braccio, "a12":

```
root["a"+(i-1)].parent = arm;
```

Chiudete il ciclo:

```
}
```

Alla fine del ciclo, l'ultimo valore di arm era "a9". Utilizzando ancora questa variabile (modificando il numero dei pezzi il nome dell'istanza sarebbe diverso), si impostano le coordinate del pezzo più grande al centro dello stage:

```
arm._x = 270;  
arm._y = 380;
```

Adesso si può scrivere la funzione move:

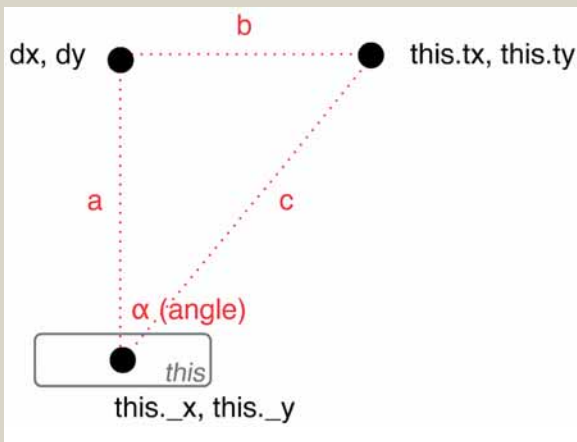
```
function move() {
```

Ora si tratta di determinare il punto d'attrazione. Questo dovrà essere definito dalle sue coordinate x e y, che saranno conservate rispettivamente nelle variabili tx e ty. Il punto d'attrazione deve chiaramente essere diverso per ogni pezzo del braccio. Il più piccolo, "a0", sarà attratto dalle coordinate del mouse. In base all'angolo che assumerà questa istanza dovrà essere determinato il punto d'attrazione del pezzo successivo, "a1", e così via. Il punto d'attrazione di "a0" sarà determinato alla fine dello script in un handler onEnterFrame, che permetterà di registrare continuamente le coordinate della posizione del puntatore. Il calcolo del punto d'attrazione del pezzo parent permetterà di utilizzare la stessa variabile con un nuovo valore al ciclo successivo. Innanzitutto bisogna trovare l'angolo d'inclinazione che arm deve assumere per rivolgersi verso il punto di attrazione. Per far questo, si immagini che il punto d'attrazione e il punto di registrazione dell'istanza disegnino un triangolo rettangolo immaginario (**Figura 3**).

I cateti di questo rettangolo possono essere calcolati sottraendo le coordinate rispettivamente orizzontali e verticali del punto d'attrazione e del punto di registrazione dell'oggetto:

```
var dx = this.tx - this._x;  
var dy = this.ty - this._y;
```

A questo punto, poiché sono noti i due valori di dx e dy, è possibile ottenere l'angolo α calcolando l'arco tangente:



```
var angle = Math.atan2(dy, dx);
```

Si supponga che l'oggetto sia stato ruotato. Si vuole che il punto d'attrazione per il pezzo successivo sia collocato all'80% della sua lunghezza, motivo per cui si era creata la variabile *w* in un passaggio precedente. Sono noti quindi un angolo e l'ipotenusa, ma non i due cateti, necessari per calcolare il nuovo valore di *tx* e *ty*. I cateti possono essere calcolati moltiplicando rispettivamente coseno e seno dell'angolo per l'ipotenusa (*this.w*). Il valore ottenuto va quindi sottratto ai valori precedenti di *this.tx* e *this.ty* per ottenere le coordinate del punto (Figura 4):

```
var tx = this.tx - Math.cos(angle)*this.w;
var ty = this.ty - Math.sin(angle)*this.w;
```

Ora è possibile attribuire i valori ottenuti alle variabili *tx* e *ty* residenti nell'oggetto *parent*, cioè il pezzo successivo nella catena (quello con il numero più alto). In questo modo, al prossimo ciclo, quando l'oggetto *parent* sarà diventato *this*, serviranno ulteriormente per calcolare i prossimi valori:

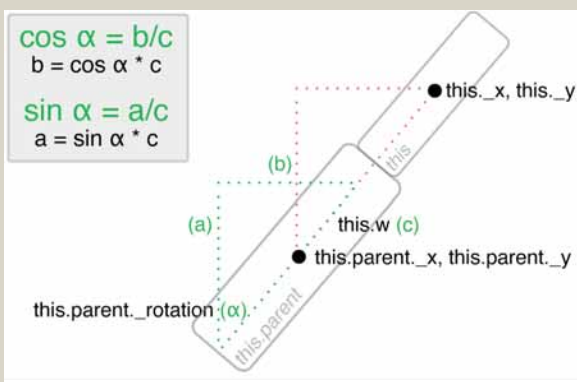
```
this.parent.tx = tx;
this.parent.ty = ty;
```

Il metodo *atan2* dell'oggetto *Math* restituisce un valore in radianti, mentre la proprietà *_rotation* di un oggetto è misurata in gradi. Quindi, poiché *angle* e il valore della proprietà *_rotation* dovranno essere trattati assieme, conviene uniformarne le unità di misura, portando *angle* in gradi:

```
var targetRotation = angle*180/Math.PI;
```

A *targetRotation* si deve sottrarre l'angolo attuale di rotazione dell'oggetto per ottenere l'angolo di variazione della rotazione che deve subire l'oggetto:

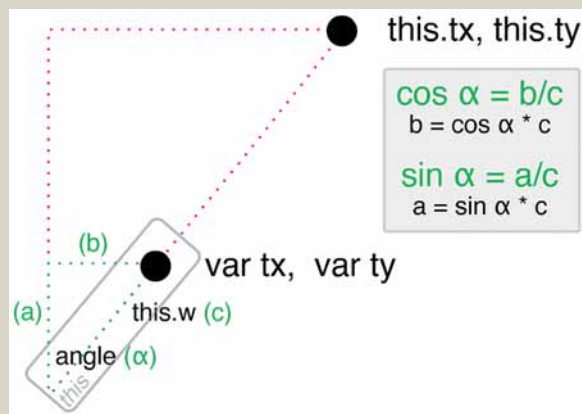
```
var diff = targetRotation-this._rotation;
```



Se il valore ottenuto è maggiore di 180, gli si deve sottrarre 360 (si usa *while* invece di *if* per far sì che si proceda finché la condizione è vera), mentre se il valore ottenuto è minore di -180, lo si deve sommare a 360. In questo modo si preverranno funzionamenti bizzarri quando, nel passaggio successivo, si attribuirà una velocità:

```
while (diff>180) {
    diff -= 360;
}
while (diff<-180) {
    diff += 360;
}
```

Per ammorbidire il movimento, meglio aggiungere un fattore d'attrito. Si tratta di un valore molto piccolo, per il quale verrà moltiplicato il fattore di rotazione (qui si propone 0.1, pari al 10% dell'angolo, ma è possibile provare a modificare questo valore per ottenere effetti diversi). A mano a mano che l'angolo diminuirà, diminuirà così anche la sua velocità, donando più organicità all'oggetto. Per i valori al di sotto di 2, questa variazione sarebbe così piccola da non essere visibile, per cui è possibile non farla calcolare av-



valendosi di una struttura condizionale:

```
if (Math.abs(diff)>2) {
    this.vr = diff*.1;
    this._rotation += this.vr;
}
```

Infine, si deve calcolare la posizione dell'oggetto, basandosi sulle coordinate del *parent*. Con la stessa formula di prima si ricaveranno i cateti del triangolo rettangolo basandosi sulla lunghezza *w* del *parent* e sulla sua rotazione. Quindi si aggiungeranno questi valori alle coordinate *x* e *y* del *parent*, per ottenere le nuove coordinate *x* e *y* dell'oggetto *this* (Figura 5):

```
this._x = this.parent._x +
    Math.cos(this.parent._rotation*Math.PI/180) * this.parent.w;
this._y = this.parent._y +
    Math.sin(this.parent._rotation*Math.PI/180) * this.parent.w;
```

Le coordinate *tx* e *ty* del punto d'attrazione relativo al primo pezzo del braccio, "a0", devono coincidere con le coordinate del puntatore. Come si diceva prima, il controllo deve avvenire continuamente e va quindi inserito in un handler *onEnterFrame*:

```
onEnterFrame = function () {
    a0.tx = _xmouse;
    a0.ty = _ymouse;
}
```

◀ **Figura 3:** il triangolo immaginario formato dalle coordinate dell'oggetto, quelle del punto d'attrazione e la loro differenza

◀ **Figura 4:** come calcolare le coordinate di un punto conoscendo l'angolo e l'ipotenusa del triangolo immaginario

◀ **Figura 5:** la stessa formula applicata a un confronto fra le proprietà di due oggetti

dai lavori di Josh Davis (www.prystation.com), Yugo Nakamura (www.yugop.com) e Jared Tarbell (www.levitated.net). Il sito è un giornale visuale on-line basato su esperimenti: qui gli studi di Keith vengono resi pubblici a mano a mano che nascono e durante le varie fasi di crescita. Alcuni lavori sono parti di studi di sperimentazioni più complesse, messi a disposizione della comunità perché ne tragga ispirazione o ne fornisca di ulteriori all'autore. BIT-101 contiene anche un interessante blog e una serie di tutorial su *Flash*, che sono sta-

ti tradotti in molte lingue e proposti in molti altri siti in tutto il mondo. Oltre ai tutorial, spiegati dettagliatamente in modo semplice e fruibile, tutti gli esperimenti raccolti per data sono disponibili nel formato .fla.

Il vantaggio collettivo dell'open source

Quando ha conosciuto *Flash*, Keith ha trascorso molto tempo a studiare su siti didattici, come Flashkit (www.flashkit.com) e Were-here (www.werehere.com). In questi luoghi esistono vere e proprie comunità, dove tutti si forniscono aiuto reciproco condividendo informazioni, ispirazioni, domande e risposte. Keith è stato molto colpito dallo spirito che animava questi gruppi d'utenti: il fatto di essere stato aiutato molto da persone che chiarivano i suoi dubbi ha alimentato il piacere che oggi ha di condividere e rendere pubbliche le sue scoperte e di rispondere a chi gli pone quesiti. L'open source è da sempre un aspetto molto significativo della comunità *Flash*, che si avvale di numerosissimi progetti, anche realizzati da grossi nomi, basati su questo principio. Keith pensa che non otterrebbe alcun vantaggio dal tenere la sua conoscenza tutta per sé, mentre il diffonderla e il confrontarla è fonte di stimoli continui e di sviluppi collettivi che arricchiscono ogni singolo individuo. Inoltre, curare un sito orientato al fornire e condividere informazioni genera un flusso di comunicazione continua, per lo più via e-mail, ma a volte anche via lettera o telefono, con persone che forniscono feedback o che chiedono consigli, che incoraggiano, stimolano, ispirano. Le energie che si mettono in moto in questo modo forniscono una dimensione più umana a un mezzo tanto capillare quanto pericolosamente impersonale quale è il Web.

Molti esperimenti di BIT-101 sono pesantemente basati sulla matematica e la fisica. Keith non aveva una particolare passione o preparazione riguardo a questi argomenti, ma l'uso dell'*ActionScript* gli suggerì come con qualche nozione di essi si possano creare degli effetti davvero grandiosi. Spesso gli utenti di BIT-101 chiedono a Keith se sia un professore di fisica o se abbia la laurea in matematica, altri gli dicono di aver imparato più dai suoi tutorial che dalle lezioni di fisica del liceo. In realtà l'educazione formale di Keith a proposito della matematica si è esaurita circa vent'anni fa sui banchi di scuola con un po' di algebra e basi di geometria. Le conoscenze che attualmente usa nel creare le sue sperimentazioni sono state raccolte negli ultimi anni da libri, siti e chiedendo alla gente, e probabilmente hanno trovato in lui anche una particolare attitudine. Spesso disegna prima il tipo di movimento che vuole creare e poi lavora semplicemente su variabili numeriche e formule finché non funziona bene, per poi dopo scoprire che quel movimento era già ben conosciuto, descritto e codificato, praticamente con le stesse formule intuitive da lui.

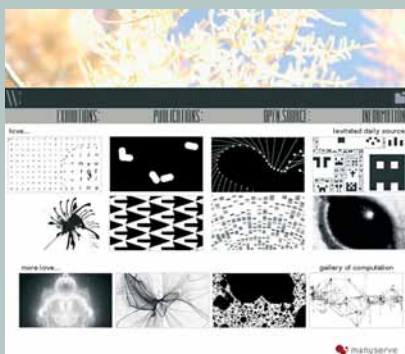
I software usati da Keith

Keith usa *PrimalScript 3.1* per l'editing dell'*ActionScript* e *Fireworks MX 2004* per la maggior parte dei la-

I siti consigliati da Keith

www.levitated.net,
www.complexification.net

Levitated è stato uno dei primi siti *Flash* che hanno davvero ispirato Keith ad andare avanti con *Flash* e creare un sito tutto suo. Jared Tarbell e Keith si sono conosciuti dapprima via e-mail, dopo aver entrambi scritto dei capitoli per il libro *Flash Math Creativity*. Ora si tengono regolarmente in contatto, per condividere idee e scambiarsi ispirazioni.



LEVITATED

www.prystation.com

Joshua Davis è stato uno dei pionieri della sperimentazione e dell'arte su *Flash*. Ora il sito ha un aspetto più commerciale, ma quello precedente è ancora raggiungibile nel portfolio Web.



JOSHUA DAVIS

www.yugop.com

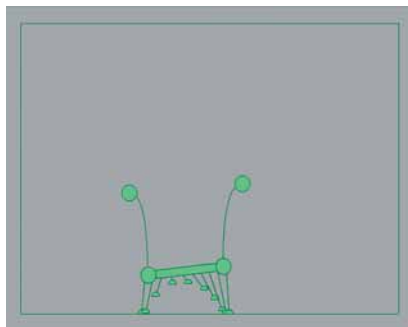
Anche Yugo Nakamura è stato uno dei pionieri della sperimentazione e dell'arte su *Flash*. Durante tutto il 2003 non aveva aggiunto ulteriori novità, ma si potevano ancora vedere tutti i suoi primi lavori. Ora è finalmente on-line la nuova versione del sito, da cui si può raggiungere anche l'archivio che conserva la opere che l'hanno reso giustamente famoso.



YUGO NAKAMURA

vori di grafica, perché è molto ben integrato con *Flash*. Queste applicazioni, assieme a *Outlook*, *Internet Explorer* e *WinAmp* sono le prime che avvia quando arriva al lavoro e generalmente rimangono aperte tutto il giorno. Ma il software che Keith usa di più è di gran lunga *Flash MX 2004*. *Flash* ha reso possibile anche per lo studente dotato di un piccolo computer domestico ciò per cui prima erano necessari macchine molto potenti ed esperti programmatori. Keith apprezza moltissimo l'*ActionScript 2*, l'evoluzione del linguaggio interno a *Flash* inaugurata con la versione *MX 2004*, che rende molto accessibile la programmazione orientata agli oggetti. Inoltre, secondo Keith, *Flash* ha reso il Web molto più dinamico. Gli altri modi di rendere dinamiche le pagine Web non sono altrettanto facili e affidabili: il *Java* è più ampio e difficile da programmare, il *DHTML* è troppo legato a quale browser o piattaforma si usi e richiede ulteriori plug-in per altri media. Molte persone attribuiscono a *Flash* la responsabilità del fatto che sul Web ci siano un sacco di lavori brutti realizzati con questo strumento. Ma Keith sottolinea giustamente che ci sono anche moltissimi pessimi lavori fatti in *HTML*, anche solo brutti graficamente. *Flash* è un programma molto potente: a seconda di come viene usato può dare origine a opere bellissime come a prodotti contrari a ogni canone possibile di estetica e di usabilità.

Attualmente, Keith non si occupa molto di design, ma soprattutto e solo di programmazione massiccia,



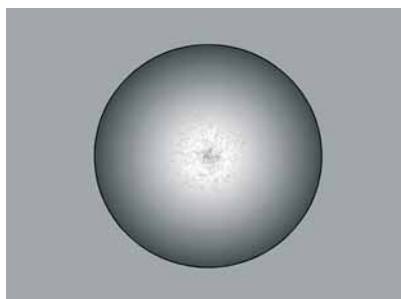
poiché l'azienda in cui lavora dispone di un valido team di designer. Queste competenze si relazionano in due modi possibili: a volte nasce prima il design, che poi viene consegnato a Keith perché lo faccia funzionare, altre volte prima crea qualcosa in *Flash* e poi i designer lo rivestono in modo che lui possa portare a compimento l'implementazione. Ultimamente, dal momento che i progetti di cui deve curare sviluppo e programmazione stanno diventando sempre più grandi e complessi, Keith sta cominciando a cogliere l'importanza di pianificare la programmazione di un progetto prima di realizzarlo. Si è messo quindi ancora una volta a studiare per cercare di stabilire delle regole entro le quali muoversi e per conoscere anche gli aspetti del lavoro di cui non deve occuparsi in prima persona, ma con i quali si deve relazionare. Si era accorto, infatti, che troppo spesso si buttava a capofitto in qualche lavoro e alla fine otteneva una struttura non propriamente flessibile. In fondo, in qualunque mestiere, nel suo in particolare, che si innesta in un terreno giovane e in rapida e inarrestabile evoluzione,

si deve sempre e continuamente imparare ed è una dote preziosa e irrinunciabile la capacità di riconoscere questa necessità.

Ingredienti di un sito Web

Nonostante non si occupi più di design, il suo passato di disegnatore e di flasher freelance ha dotato Keith di una certa competenza in merito, per cui gli abbiamo chiesto di dare ai lettori alcuni consigli per la realizzazione di siti Web efficaci.

In ogni genere di composizione visuale, l'obiettivo è passare un messaggio e usare gli elementi visuali allo scopo di dirigere l'attenzione dell'utente dove egli coglierà il messaggio. Come idea generale, Keith ritiene che convenga animare l'elemento più importante della



pagina, ciò che si vuole l'utente veda, oppure animare qualcosa che stia lì vicino, qualcosa che vada verso l'elemento chiave. L'animazione è

ormai un tipo di contributo ammissibile in qualsiasi contesto, ma non smette d'innervosire la gente il movimento continuo e ossessivo, come quello delle vecchie gif animate che si usavano sulle home page personali. Una volta canalizzata l'attenzione dell'utente, l'animazione deve interrompersi e permettere all'utente di leggere il messaggio.

A proposito del testo, è noto e comprensibile che la lettura su monitor non sia altrettanto gradevole quanto quella su carta. Per questo Keith pensa che su un sito Web sia necessario ridurre all'essenziale i contenuti testuali e corredarli di headline, puntatori e altri oggetti che la scansione visiva possa cogliere immediatamente. Inoltre, non si deve trascurare che il messaggio su una pagina Web ha talmente poco tempo a disposizione per essere colto, che è meglio che sia breve e a caratteri grandi, chiari e distinti. Per quanto riguarda il suono, Keith lo usa raramente. La sua scelta è dettata dalla consapevolezza di non saperne abbastanza e del fatto che, non usato nel modo giusto, l'audio è un elemento noioso che può rovinare l'atmosfera creata dalla grafica. In effetti, davvero pochi siti lo usano in maniera efficace: Praystation (www.praystation.com), per citarne uno, lo usa in modo superbo. Anche a proposito dell'uso del colore, Keith si rende conto che, per avvalersene con accortezza, bisognerebbe avere un bagaglio di conoscenze che lui non ha mai acquisito, motivo per cui nei suoi lavori usa molti grigi. Grafica pesante e suoni, inoltre, possono rendere i file molto lunghi da scaricare, mentre i lavori di Keith, semplici e gestiti quasi completamente da *ActionScript*, non risentono delle problematiche relative alla velocità di trasmissione. ■