RAPPRESENTAZIONE DIGITALE DELLE INFORMAZIONI (RIPASSO E PRECISAZIONI)

CODIFICA

- Una codifica esatta a *n* bit è possibile solo quando l'insieme delle informazioni da codificare è finito e di dimensione inferiore o uguale ***al massimo del valore che posso rappresentare con una parola di una determinata lunghezza***
- I calcolatori sono oggetti finiti che elaborano e memorizzano un numero finito di bit.
- Se l'insieme da codificare ha una contiene un numero di informazioni maggiore di 2^n se ne puo' dare solo una rappresentazione approssimata o parziale. Questa limitazione avviene in due modi:

➢ **Operazioni di limitazione**

➢ **Operazioni di partizionamento**

CODIFICA DEL TESTO

- Un testo e' una sequenza di caratteri alfabetici, separatori e caratteri speciali.
- Ad ogni carattere è associata una diversa configurazione di bit.
- E' un'insieme di informazioni finito e discreto ed è quindi rappresentabile in maniera completa.
- La stratificazione storica di vari tipi di codifica ha fatto in modo che non esiste una codifica univoca. Un programma che usa un testo deve sapere anche che tipo di codifica viene utilizzata.
- **Esempio**

NUMERI INTERI

- I numeri interi sono un insieme discreto illimitato.
- Per poter essere codificati devono essere limitati.
- Nel caso si vogliano rappresentare sia numeri positivi che negativi si usa *1* bit per rappresentare il segno e i restanti per rappresentare il modulo.
- Se è il **n** numero di bit che uso per rappresentare i numeri e **a** = n-1 l'ntervallo dei numeri rappresentabile andrà da -2^a *a* 2^a-1.
- Se invece voglio rappresentare solo l'insieme dei numeri positivi (i cosiddetti **unsigned**) l'intervallo andrà da **0** a 2^n.

NUMERI REALI

- I numeri reali sono un insieme continuo e illimitato.
- Per poterli rappresentare occorre limitarli (in modo simmetrico rispetto allo 0) e partizionarli.
- *Rappresentazione in virgola fissa:* Degli *n* bit della parola, *1* rappresenta il segno, *a* rappresentano le cifre prima della virgola e *b* le cifre dopo la virgola.
  - Il massimo numero rappresentabile è $(2^{n-1}-1)/2^b$
  - L'accuratezza assoluta è 2^{-b}
- *Rappresentazione in virgola mobile:* (Floating point) espressa nella forma → $s0.M\ B^{seE}$

CODIFICA DELLE IMMAGINI

- Le immagini sono informazioni continue in tre dimensioni: due spaziali ed una colorimetrica.
- Per codificarle occorre operare tre discretizzazioni.
  - Due discretizzazioni spaziali riducono l'immagine ad una matrice di punti colorati, detti **pixel**.
  - La terza discretizzazione limita l'insieme di colori che ogni pixel può assumere.

IMMAGINI A 256 COLORI

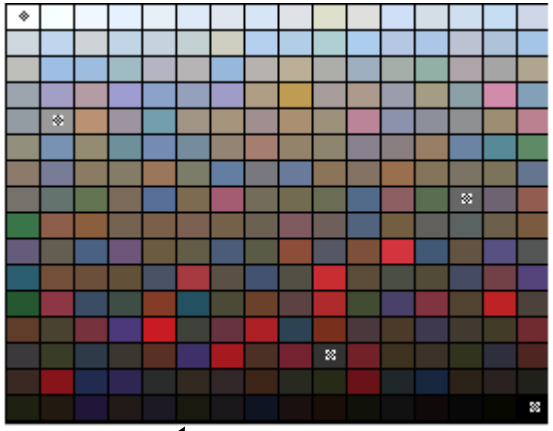
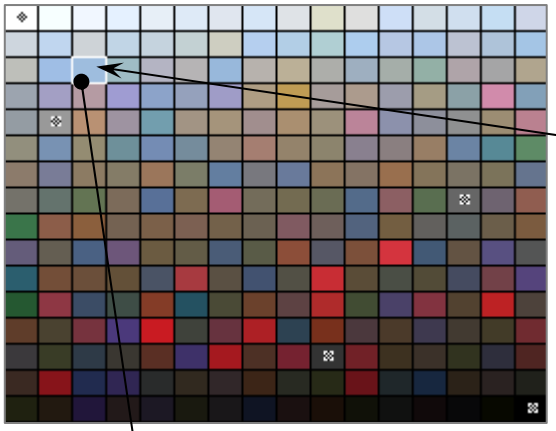


Tabella dei colori (palette) composta da 256 colori numerati da 0 a 255. Ogni colore viene definito per il suo contenuto di Rosso, Verde e Blu.

Immagine. Il colore di ogni punto (pixel) viene definito da un numero da 0 a 255 (8 bit). Viene utilizzato il colore definito nella palette all'indice corrispondente.

IMMAGINI A 256 COLORI



Il colore del pixel è definito dal numero **00100010** (34 decimale) che rappresenta l'indice della palette.

Ogni colore viene definito nella palette specificando i livelli dei tre colori fondamentali.

Indice	Rosso	Verde	Blu
00100010	10011110	10111101	11011110

IMMAGINI RGB

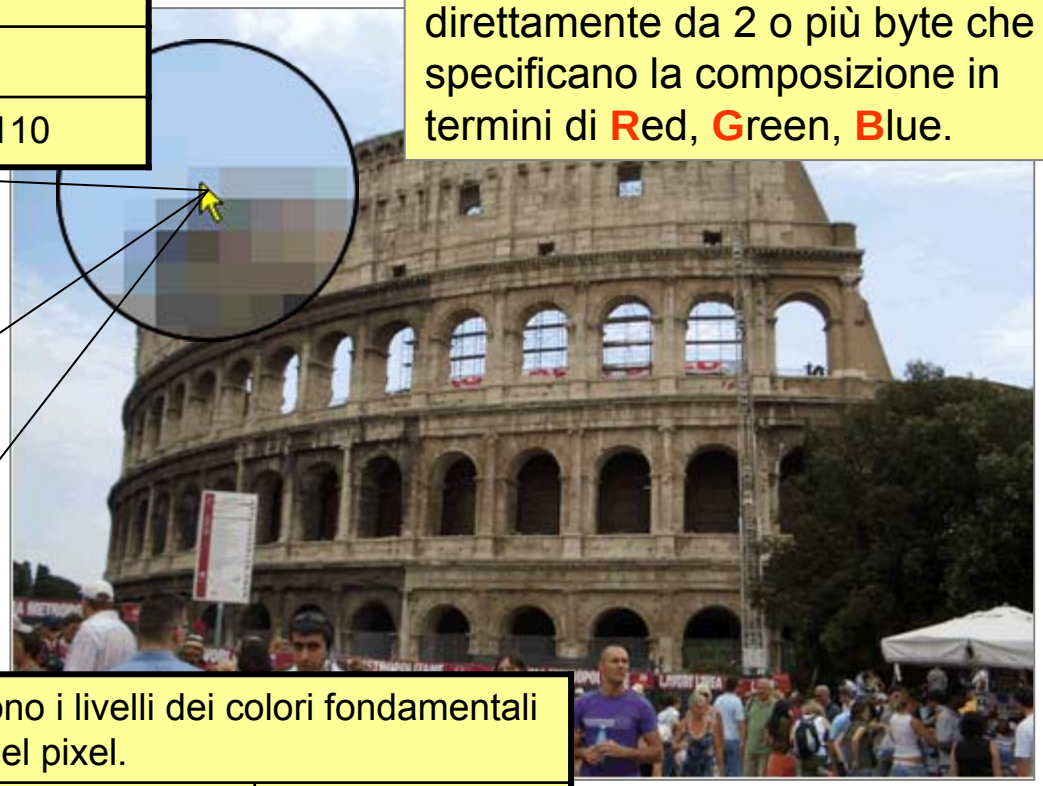
Nelle immagini a 24 bit tre byte definiscono i livelli dei colori fondamentali.

Rosso	Verde	Blu
10011110	10111101	11011110

Nelle immagini a 16 bit si usano 5 bit per definire i livelli dei colori fondamentali.

Rosso	Verde	Blu
10011	10111	11011

Il colore del pixel è definito direttamente da 2 o più byte che ne specificano la composizione in termini di **R**ed, **G**reen, **B**lue.



Nelle immagini a 32 bit tre byte definiscono i livelli dei colori fondamentali il quarto il livello di trasparenza (alpha) del pixel.

Rosso	Verde	Blu	Alpha
10011110	10111101	11011110	11111111

RAPPRESENTAZIONE ESADECIMALE DEI COLORI

Corrispondenza tra valori espressi in forma binaria, decimale ed esadecimale.							
Rosso		Verde		Blu		Alpha	
1001	1110	1011	1101	1101	1110	1111	1111
9	E	B	D	B	E	F	F
158		189		190		255	

9EBDBE

10403262

LA PROGRAMMAZIONE

- Abbiamo visto come le informazione vengono codificate per potere essere elaborate dal computer.
- Ora passeremo a vedere gli strumenti che abbiamo a disposizione per programmare il computer, per fare in modo, cioè, che il computer elabori le informazione che gli forniamo in modo utile per noi.

UN PO' DI STORIA - GLI ANNI '40

- All'inizio, negli anni '40, l'unico metodo per programmare era il **linguaggio macchina**.
- Il lavoro del programmatore consisteva nel settare ogni singolo bit a 1 o 0 su enormi computer che occupavano stanze intere e pesavano decine di tonnellate.
- I monitor non esistevano; i dati e i programma si fornivano al computer su schede perforate e il computer mandava i risultati su telescriventi.
- In questo modo gli scienziati riuscirono in pochi mesi a completare i calcoli per costruire la prima boma atomica, calcoli che se fatti a mano, con calcolatrici meccaniche avrebbero richiesto anni.

GLI ANNI '50

- Viene progettato il **FORtrAN** (FORmula trANslator), il cui utilizzo era ed è prettamente quello di svolgere in maniera automatica calcoli matematici e scientifici,
- Viene progettato l'**ALGOL** (ALGOrithmic Language), altro linguaggio per applicazioni scientifiche sviluppato da Backus (l'inventore del FORtrAN) e da Naur.
- Backus e Naur mettono a punto un metodo per rappresentare le regole dei vari linguaggi di programmazione che stavano nascendo.

GLI ANNI '60

- Nel **1960** venne presentato il **COBOL** (COmmon Business Oriented Language), ideato per applicazioni nei campi dell'amministrazione e del commercio, per l'organizzazione dei dati e la manipolazione dei file.
- Nel **1964** fa la sua comparsa il **BASIC**, il linguaggio di programmazione per i principianti, che ha come caratteristica fondamentale quella di essere molto semplice e, infatti, diventa in pochi anni uno dei linguaggi più utilizzati al mondo.

GLI ANNI '70

- Intorno al 1970, però, Niklus Wirth propone il **PASCAL** per andare incontro alle esigenze di apprendimento dei neo-programmatori, introducendo però la possibilità di creare programmi più leggeri e comprensibili di quelli sviluppati in basic.
- Pochi anni più tardi fa la sua comparsa il **C**.
- C e Pascal segnano una svolta importante in quanto sono linguaggi strutturati: struttura dei dati, struttura del codice.

GLI ANNI '80

- Negli anni ottanta comincia ad affermarsi la Object Oriented Programming; la programmazione orientata agli oggetti basata sul nuovo concetto di Classe.
- Il primo linguaggio a proporla ai programmatori professionisti è il C++ .

JAVA

- È il primo linguaggio progettato appositamente per la programmazione orientata agli oggetti (non è cioè l'adattamento di un linguaggio preesistente).
- Il programma ottenuto non è un programma destinato a *girare* in una ambiente specifico (Windows o Macintosh o Linux) ma gira su un computer virtuale, la *Java Virtual Machine*. Basterà installare la versione specifica della *Java Virtual Machine* per il sistema operativo specifico e lo stesso programma potrà girare in ambienti completamente diversi.

APPLICAZIONI MULTIMEDIALI

- Dal punto vista del nostro corso meritano una particolare attenzione gli strumenti per lo sviluppo di applicazioni multimediali.
- Possiamo distingue due tipi principale di applicazioni multimediali: quelle **on-line** e quelle **off-line**.

APPLICAZIONI ON-LINE

- Il veicolo obbligato per la distribuzione di qualsiasi applicazione on-line è il browser.
- Qualsiasi applicazione on-line passerà quindi anche attraverso l'uso del linguaggio **HTML** il linguaggio su cui si basa la composizione delle pagine su Internet.

HTML (HYPETEXT MARKUP LANGUAGE)

- Descrive come una pagina web debba essere mostrata da un browser.
- È un linguaggio a marcatori: tutte le istruzioni proprie del linguaggio sono inserite tra due segni specifici (i **marcatori** "**<**" e "**>**") e costituiscono i cosiddetti **tag**.
- I browser cercano di interpretare i **tag** come istruzioni mentre il resto viene mostrato come testo.
- I **tag** che il browser non riesce ad interpretare vengono semplicemente ignorati.

APPLICAZIONE AVANZATE

- **HTML** serve essenzialmente a descrivere come va composta una pagina e a definirne i collegamenti ipertestuali.
- Per costruire interattività più avanzate dobbiamo:
  - Utilizzare componenti che estendono le funzionalità di HTML (e che girano sul computer dell'utente o, come si dire dal **lato client**) oppure
  - Costruire applicazioni che utilizzino un architettura **client-server**.

OGGETTI PROGRAMMABILI

- Specifici **tag** consentono di inserire nella pagine web componenti che estendono le funzionalità del browser.
- I più importanti sono:
  - **Fogli di stile**
  - **Script**
  - **Applet**
  - **Object ed Embed**

FOGLI DI STILE

- Attraverso i fogli di stile si può ridefinire l'aspetto e il comportamento visuale degli elementi che costituiscono una pagina Internet
- La definizione degli stile può essere inglobata nella pagine utilizzando il tag **<style></style>**
- Oppure caricata da un file esterno di tipo **css** (**C**ascade **S**tyle **S**heet) utilizzando il tag **<link…/>**

SCRIPT

- E’ possibile inserire del codice che il browser è in grado di eseguire con un **interprete** integrato.
- I linguaggi a disposizione dello sviluppatore sono due: **VBSCRIPT** (che deriva dal Visual Basic e viene riconosciuto però solo da Internet Explorer) e **JAVASCRIPT** che deriva invece da JAVA ed è riconosciuto più o meno da tutti i browser.
- Il codice può essere inglobato nella pagina o caricato da un file esterno utilizzando il tag **<script></script>**

SCRIPT ESEMPIO

```
<script language="JavaScript" type="text/JavaScript">
<!--
function MM_preloadImages() { //v3.0
  var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
    var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0; i<a.length; i++)
    if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}}
}

function MM_swapImgRestore() { //v3.0
  var i,x,a=document.MM_sr; for(i=0;a&&i<a.length&&(x=a[i])&&x.oSrc;i++) x.src=x.oSrc;
}

function MM_findObj(n, d) { //v4.01
  var p,i,x;  if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
    d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
  if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++) x=d.forms[i][n];
  for(i=0;!x&&d.layers&&i<d.layers.length;i++) x=MM_findObj(n,d.layers[i].document);
  if(!x && d.getElementById) x=d.getElementById(n); return x;
}

function MM_swapImage() { //v3.0
  var i,j=0,x,a=MM_swapImage.arguments; document.MM_sr=new Array; for(i=0;i<(a.length-
2);i+=3)
   if ((x=MM_findObj(a[i]))!=null){document.MM_sr[j++]=x; if(!x.oSrc) x.oSrc=x.src; x.
src=a[i+2];}
}
//-->
</script>
```

APPLET

- Il tag **APPLET** consente di inserire in una pagina web un programma scritto in **JAVA** espressamente scritto per il web.
- Il programma dovrà infatti rispettare alcune condizioni di sicurezza.
- Il tag **APPLET** noi fa altro che offrire un collegamento tra il browser e la **Java Virtual Machine** che deve essere installata sul computer e fa *girare* l'applicazione.

APPLET ESEMPIO

```
<applet code="CellularAutomata3" archive="media/CellularAutomata3.jar" width="200" height="200">
</applet>
```

OBJECT-EMBED

- Il tag **OBJECT** (Internet Explorer su piattaforma Windows) insieme al tag **EMBED** (altri browser) consentono di inserire nelle pagine web oggetti programmabili di varia natura gestiti da estensioni dei browser o (in Windows) dai programmi gestiti direttamente dal sistema operativo (i cosiddetti ActiveX).
- A noi interessa perchò attraverso questo meccanismo il browser viene collegato all'oggetto **SHOCKWAVE FLASH** cioè, come è più corretto dire oggi alla **Virtual Machine** che è in grado di eseguire le applicazioni **FLASH**.

OBJECT-EMBED ESEMPIO

```
<OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
 codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=7,0,0,0"
 WIDTH="100%" HEIGHT="100%" ALIGN="">
 <PARAM NAME=movie VALUE="Loader.swf">
 <PARAM NAME=menu VALUE=true>
 <PARAM NAME=quality VALUE=high>
 <PARAM NAME=bgcolor VALUE=#FFFFFF>
 <EMBED src="Loader.swf" menu=false quality=high bgcolor=#FFFFFF  WIDTH="100%" HEIGHT="100%"
ALIGN=""  TYPE="application/x-shockwave-flash" PLUGINSPAGE =
"http://www.macromedia.com/go/getflashplayer"></EMBED>
</OBJECT>
```

ARCHITETTURA CLIENT-SERVER

- I tipi di pagine web di cui abbiamo fin qui parlato sono pagine statiche. Pagine, cioè, che vengono inviate al browser dal server web esattamente come sono state composte.
- Molti siti web oggi usano, invece, pagine dinamiche, pagine, cioè, il cui contenuto viene composto dal server al momento della richiesta.
- Le pagine che risiedono sul server sono programmi che vengono eseguiti ed elaborano informazioni sulla base dei parametri ricevuti. Quello che viene rispedito al client è il risultato di questa elaborazione.

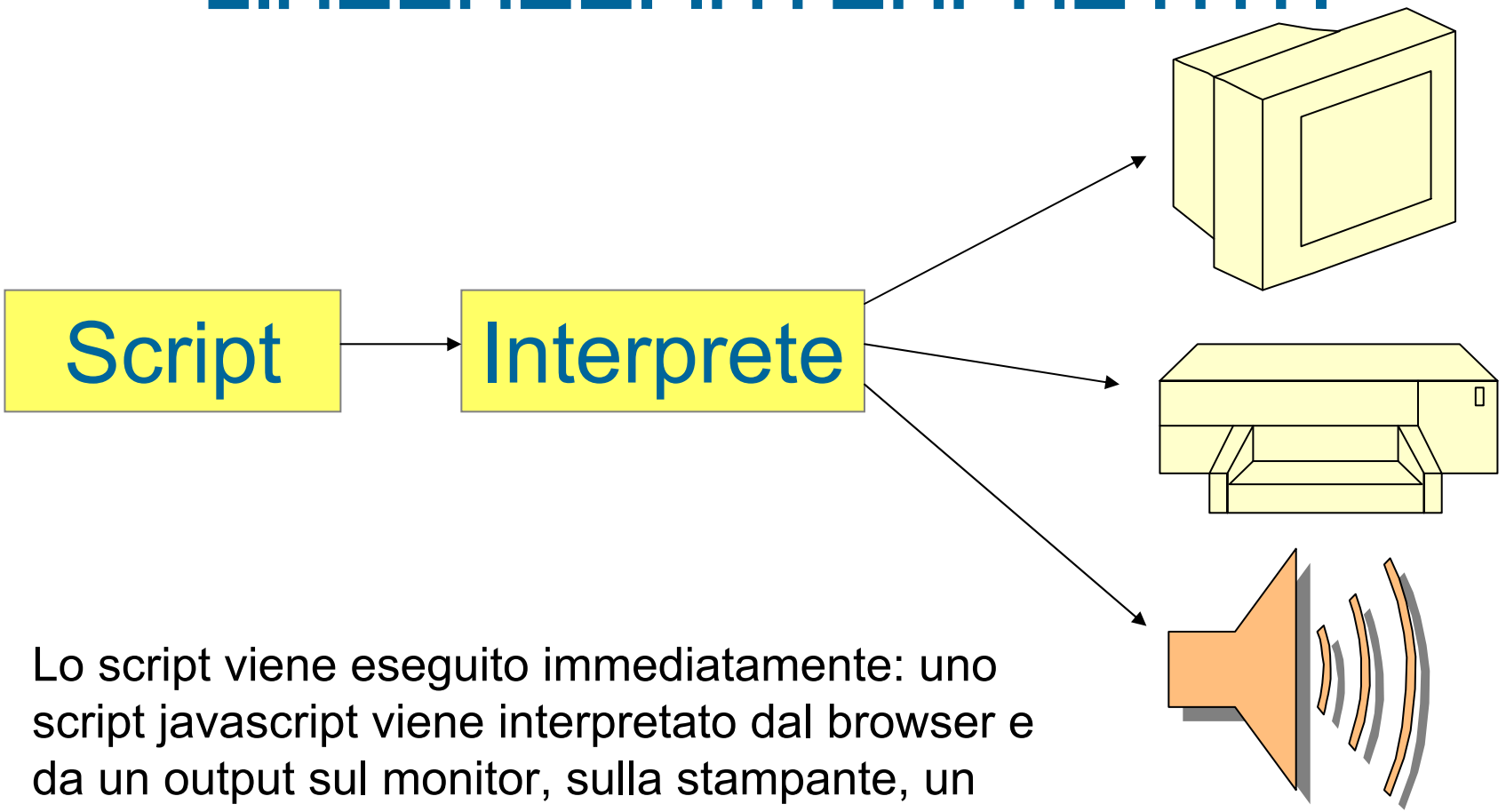
APPLICAZIONI MULTIMEDIALI OFF-LINE

- Per quanto riguarda la Creazione di Giochi gli sviluppatori tendono a usare (per ottenere la massima efficienza) strumenti il più vicino possibile al linguaggio dei processori quindi **C++** e **Assembler**.
- Per altre applicazioni (didattiche, pubblicitarie, ecc) e giochi non troppo complessi vengono usati **FLASH**, **DIRECTOR** e **JAVA**.
- Recentemente Adobe ha messo a disposizioni strumenti che consento di installare e far girare applicazioni ActionScript multipiattaforma off-line utilizzando la Virtual Machine basata su Flash Player (Adobe Air e Adobe Flex).

LINGUAGGI INTERPRETATI E COMPILATI

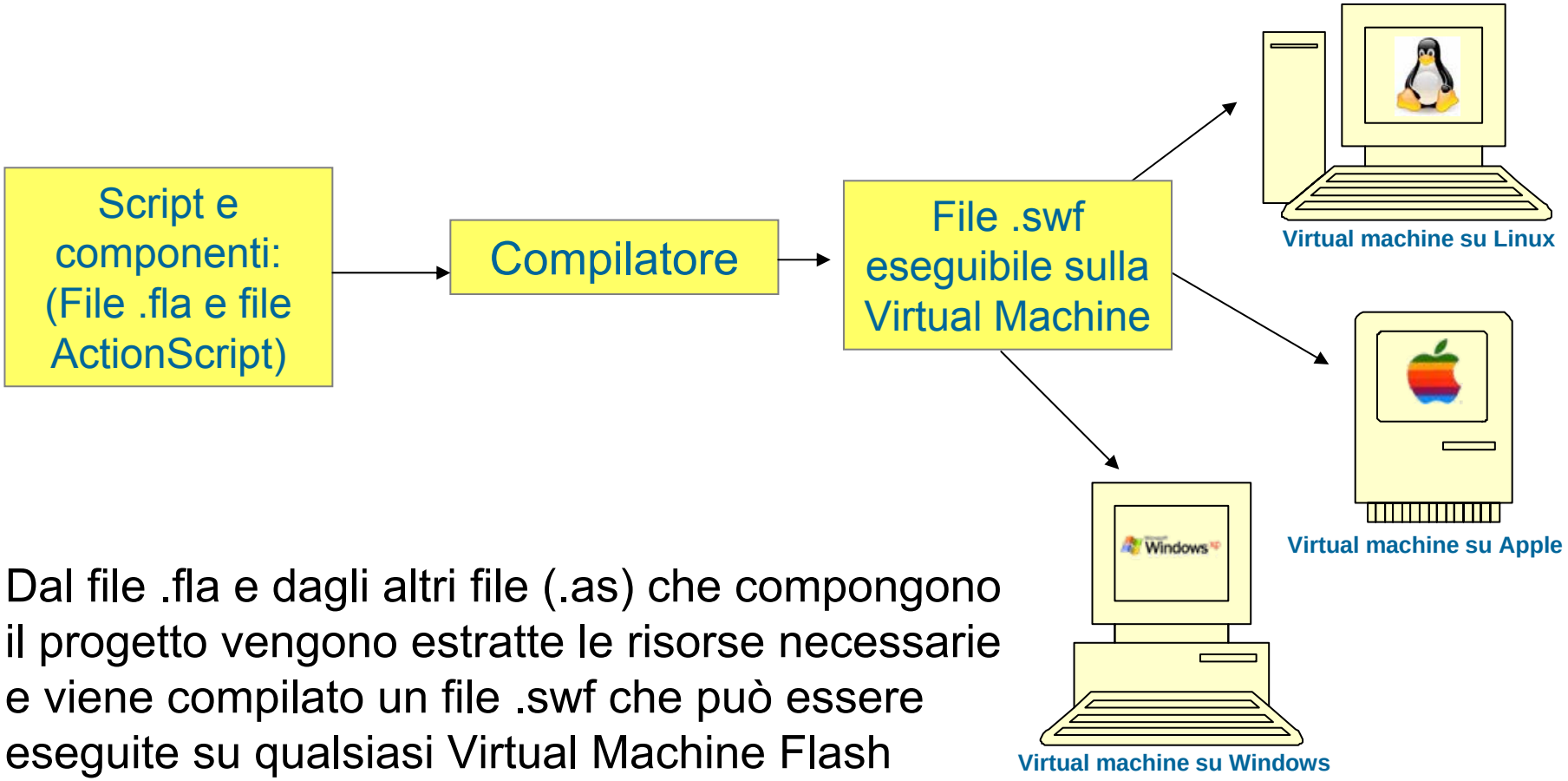
- Per quanto lo sviluppo di applicazioni possiamo distinguere i linguaggi di programmazioni indue grandi categorie;
  - I **linguaggi interpretati**: uno specifico modulo software (detto appunto interprete) esegue direttamente gli script così come li ho composti
  - I **linguaggi compilati**: prima dell'esecuzione uno specifico programma (detto compilatore) combine il codice ed eventualmente altre risorse in un nuovo file che può essere eseguito (o da un sistema operativo specifico o da una Virtual Mchne)

LINGUAGGI INTERPRETATI



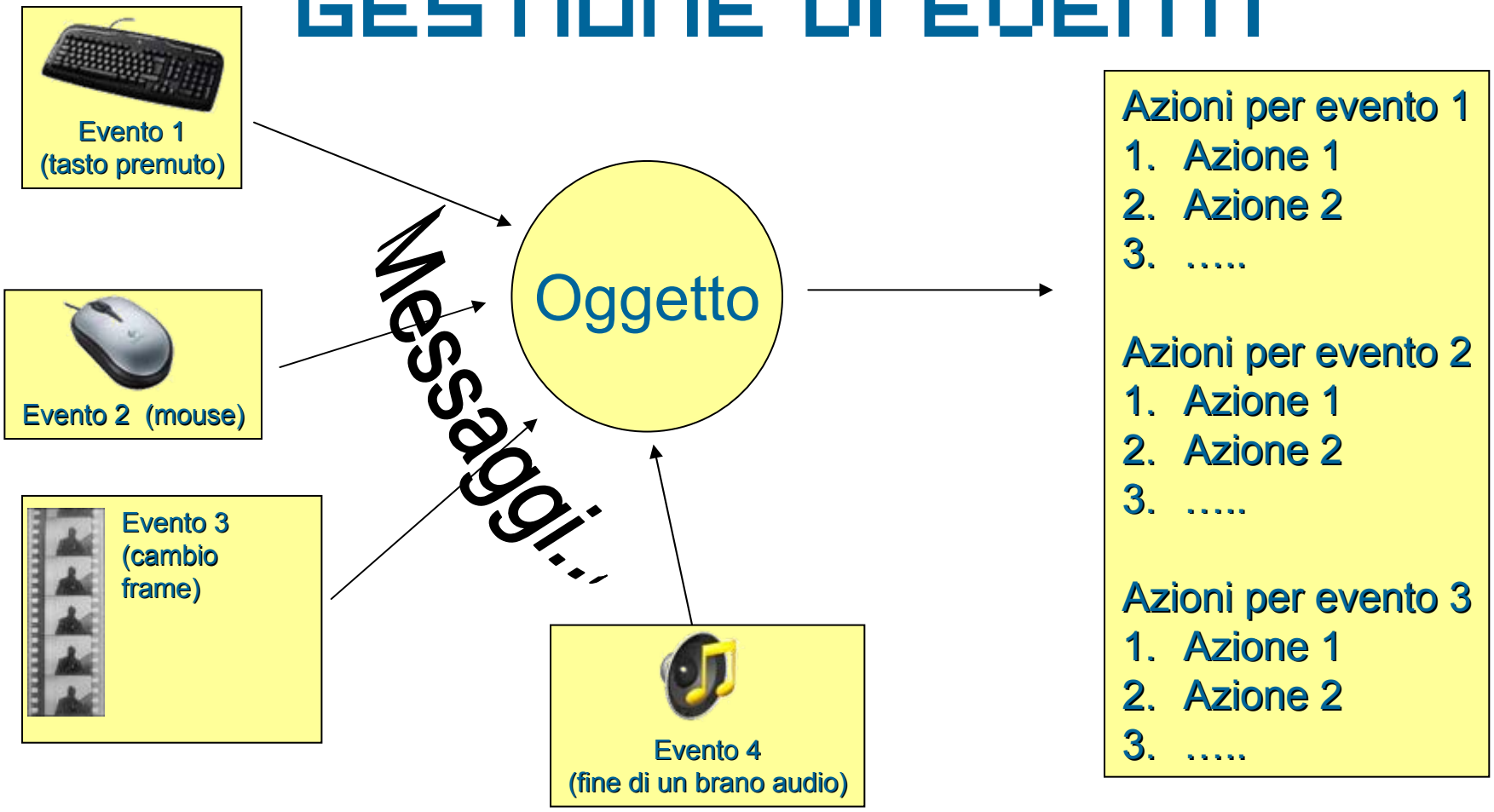
Lo script viene eseguito immediatamente: uno script javascript viene interpretato dal browser e da un output sul monitor, sulla stampante, un output audio, ecc.

LINGUAGGI COMPILATI (ESEMPIO ACTIONSCRIPT)



Dal file .fla e dagli altri file (.as) che compongono il progetto vengono estratte le risorse necessarie e viene compilato un file .swf che può essere eseguite su qualsiasi Virtual Machine Flash

PROGRAMMAZIONE È GESTIONE DI EVENTI



PROGRAMMA 1

```
stop();
prova_txt.text = “Ciao gente”;
```

PROGRAMMA 1

evento	azioni
Creazione del filmato	1. Crea un oggetto di tipo dinamico e assegnagli il nome **prova_txt** 2. Imposta le proprietà di **prova_txt** 3. Posizione **x** = … 4. Posizione **y** = … 5. Carattere = Arial 6. Dimensione Carattere = 20
Primo frame	1. Ferma il filmato su questo frame 2. Modifica la proprita testo di prova_txt: assegna alla proprietà il valore “Ciao gente”.

PROGRAMMA 2

```
var a:int;
var b:int;
a = parseInt(numero_txt.text);
b = a*a;
messaggio_txt.text = "Il
  quadrato di " + a.toString() +
  " è " + b.toString();
```

PROGRAMMA 2

evento	azioni
Creazione del filmato	1. Crea un oggetto di tipo testo input e assegnagli il nome **numero_txt** 2. Imposta le proprietà visive di **numero_txt** 3. Crea un oggetto di tipo testo dinamico e assegnagli il nome **messaggio_txt** 4. Imposta le proprietà visive di **messaggio_txt**
Primo frame *(il codice viene eseguito ogni volta che viene visualizzato il primo frame)*	1. Dichiara le varibili a e b come **int** (conterranno numeri interi) 2. Prendi il testo presente nella proprietà **text** di numero_txt convertilo in un numero intero e metti il risultato nella varibile **a**. 3. Calcola il quadrato di **a** e metti il risultato in **b**. 4. Converti a e b in testo (string), componi il messaggio che comunica il risultato e modifica la proprietà text di messaggio_txt in modo che mostri il messaggio.

GLI ELEMENTI DEL LINGUAGGIO

INTRODUZIONE

- **Costante**: quantità nota a priori che non dipende dall'input dell'utente e non cambia durante l'esecuzione del programma.
- **Variabile**: nome simbolico a cui è associato un valore che può dipendere dall'input dell'utente e cambiare durante l'esecuzione del programma.
- **Espressione**: sequenza di variabili, costanti, espressioni collegate tra loro da operatori.

Sintassi

- Ora prenderemo in esame questi elementi in termini grammaticali.
- Come punto di riferimento prenderemo il linguaggi con cui avremo a che fare in questo corso: ***ActionScript*** il linguaggio utilizzato da FLASH, ma la maggior parte delle cose di cui parleremo varranno, in generale, per tutti i linguaggi.

Elementi di un linguaggio

- Le unità semantiche di base di un linguaggio sono:
  - ***Parole chiave***
  - ***Operatori e separatori***
  - ***Letterali*** (o ***Costanti***)
  - ***Nomi*** (o ***Identificatori***)

Parole chiave

- Le parole chiave sono i termini (composti da caratteri alfanumerici), riservati al linguaggio di programmazione.
- Il creatore del linguaggio di programmazione stabilisce a priori quali termini riservare e quale sarà la loro funzione, il compito del programmatore è quello di impararle ed usarle in maniera appropriata.
- L'uso improprio di tali termini viene generalmente rilevato durante la fase di compilazione di un programma.

Parole chiave in ActionScript

and, break, case, catch, class, continue, default, delete, do, dynamic, else, extends, finally, for, function, get, if, implements, import, in, instanceof, interface, intrinsic, new, not, or, package, private, public, return, set, static, switch, this, throw, try, typeof, var, void, while, with

Parole chiave in JAVA

abstract, assert, boolean, break, byte, case, catch, char, class, const, continue, default, do, double, else, enum, extends, final, finally, float, for, goto, if, implements, import, instanceof, int, interface, long, native, new, package, private, protected, public, return, short, static, strictfp, super, switch, synchronized, this, throw, throws, transient, try, void, volatile, while

Operatori

- Gli operatori sono token composti di uno o più caratteri speciali che servono a controllare il flusso delle operazioni che dobbiamo eseguire o a costruire ***espressioni***
- Operatori usati sia in ***ActionScript*** che in ***JAVA***:

```
++  !   !=  !==  %   %=   &   &&   &=  (  )  -
*   *=  ,   .   ?:  /   //  /*  /=  []  ^   ^=
{}  |   ||  |=   ~   +   +=   <   <<   <<=
<=  <>   =   -=   ==   ===  >   >=   >>
>>=   >>>   >>>=
```

Proprietà degli operatori

- **Posizone**
  Un operatore si dice **prefisso** se viene posto prima degli operandi, **postfisso** se viene posto dopo e **infisso** se si trova tra gli operandi.
- **Arietà**
  L'arietà è il numero di argomenti di un operatore: 1, 2 o 3.

Proprietà degli operatori

- **Precedenza (o Priorità)**
  Indica l'ordine con il quale verranno eseguite le operazioni. Ad esempio in **4+7*5** verrà prima eseguita la moltiplicazione poi l'addizione.
- **Associtività**
  Un operatore può essere associativo a **sinistra** oppure associativo a **destra**. Indica quale operazione viene fatta prima a parità di priorità.

Separatori

- I separatori sono simboli di interpunzione che permettono di chiudere un'istruzione o di raggruppare degli elementi.
- Il separatore principale è lo *spazio* che separa i *termini* tra di loro quando non ci sono altri separatori. Gli altri separatori sono:

```
( ) { } , ;
```

Letterali (o costanti)

- Le *costanti* (o letterali) sono quantità note a priori il cui valore non dipende dai dati d'ingresso e non cambia durante l' esecuzione del programma.
- La sintassi con cui le costanti sono descritte dipende dal tipo di dati che rappresentano.

Costanti numeriche

- Le ***costanti numeriche*** iniziano sempre con un carattere numerico: il fatto che un *token* inizi con un numero basterà ad indicare al compilatore che si tratta di una costante numerica. Se il compilatore non potrà valutare quel *token* come numero segnalerà un errore.
- Il segno che separa la parte intera di un numero dalla parte decimale è il punto.
- È possibile inserire numeri in formato decimale, binario, ottale o esadecimale.
- Per segnalare al compilatore che un numero non è decimale si fa precedere il numero da un prefisso. Per i numeri esadecimali questo prefisso è 0x.
- Gli altri *termini* (*parole chiave* e *nomi*) NON possono iniziare con un numero.

Esempi di costanti numeriche

```
1
2433
1000000000
3.14
.33333333333
0.5
2345.675
0xFF0088
0x5500ff
0xff.00aa
```

Costanti stringa

- Una stringa è una sequenza di caratteri **UTF 16** ed permette di rappresentare testi. Un *costante* stringa è una sequenza (anche vuota) di caratteri racchiusi tra apici singoli o apici doppi.
- Per inserire ritorni a capo, tabulazioni, particolari caratteri o informazioni di formattazione si utilizzano speciali sequenze di caratteri dette *sequenze di escape*. Una sequenza di escape è formata da un carattere preceduto dal simbolo “\” (*backslash*). La sequenza di escape inserisce un carattere che non sarebbe altrimenti rappresentabile in un letterale stringa.

Principali sequenze di escape

\n nuova riga;

\r ritorno a capo;

\t tabulazione orizzontale;

\‘ apostrofo (o apice singolo);

\" doppio apice;

**** backslash(essendo un carattere speciale deve essere inserito con una sequenza di escape).

Esempi di costanti stringa

```
// Stringa racchiusa da apici singoli
'Ciao a tutti'
// Stringa racchiusa tra apici doppi
"Ciao"
/* La sequenza di escape risolve l’ambiguità tra l’apostrofo
inserito nella stringa e gli apici singoli che la
racchiudono */
'Questo è l\'esempio corretto'
/* In questo caso non c’è ambiguità perché la stringa è
racchiusa tra doppi apici */
"Anche questo è l'esempio corretto"
/* Per inserire un ritorno a capo si usano le sequenze
di escape */
"Questa è una stringa valida\rdi due righe"
```

Costanti booleane

- Le costanti booleane, poiché rappresentano valori logici, possono avere solo due valori: vero (rappresentato dal letterale ***true***) e falso (rappresentato dal letterale ***false***).

Costanti di tipo Array

- Il letterale ***Array*** è costituito da una serie di elementi separati da virgole compresa tra due parentesi quadre:

```
// array che contiene i mesi dell'anno
["January", "February", "March", "April"];
```

Costanti di tipo Object

- Il letterale ***Object*** è invece compreso tra parentesi graffe ed è costituito da una serie di coppie "**chiave:valore**" separate da virgole:

```
//record di una rubrica telefonica in formato Object
{name:"Irving",age:32,phone:"555-1234"};
```

Altre costanti

- I linguaggi normalmente definiscono anche altre costanti. Alcune rappresentano valori speciali che non possono essere rappresentati che da un valore simbolico (come abbiamo visto per ***true*** e ***false***) altre sono valori rappresentati da un letterale per comodità, ma possono essere anche rappresentate, a seconda dei casi, come stringe o come valori numerici.
- Per quanto riguarda il primo gruppo *ActionScript* definisce ***Infinity***, ***-Infinity***, ***undefined***, ***null*** e ***NaN.***

Identificatori (o Nomi)

- Un identificatore è un nome definito dal programmatore. Gli identificatori si usano per dare nomi alle variabili, alle funzioni e ai tipi di dati (o classi).

Regole per gli Identificatori

- il primo carattere deve essere una lettera o il simbolo “_” (ricordiamo che nel caso la prima lettera fosse un numero il compilatore tenterebbe di interpretare il nome come costante numerica);
- i caratteri successivi possono essere lettere, numeri o “_”.
- Gli identificatori non possono inoltre coincidere con le parole riservate del linguaggio.