Università di Pisa
Corso di laurea in Informatica Umanistica

Chameleonic web design

Progettazione e sviluppo della grafica di un sito sensibile a fattori esterni

RELATORI

Maria Simi
Theo van Boxel

CANDIDATO

Alessandra Bonacoscia

Anno Accademico 2009-2010

Indice

Capitolo 5 Il pannello di controllo

1. Come funziona il pannello di controllo

Conclusioni

Introduzione

Oggi stiamo vivendo l'era del *Web 2.0*, l'era di *Facebook*, di *Twitter*, di *Flickr*, in cui ogni singola persona che naviga la rete può contribuire ad arricchire i suoi contenuti, inserendo fotografie, commenti, link, articoli. Il navigatore, da semplice fruitore del web, ne diventa artefice.

Nell'ottobre 2009 si è tenuto il *Web 2.0 Summit*, in California, durante il quale è stato coniato il termine *Web Squared*. Si tratta di un'ipotesi di come si potrebbe sviluppare il web nell'immediato futuro.

Il *Web Squared* potrebbe essere il web di "*information shadow"* e cioè delle ombre che, da un numero sempre maggiore di elementi appartenenti al mondo reale, sono proiettate nel mondo virtuale. Queste ombre sono informazioni che, spesso, l'utente non è consapevole di fornire, o non fornisce volontariamente, come, ad esempio, la sua posizione geografica.

Il 7 e l'8 maggio a Pisa ha avuto luogo il IV Summit Italiano sull'Architettura dell'Informazione. Il prof. Amir Baldissera è intervenuto con: "Il Web incontra il Mondo. Come la rivoluzione mobile cambierà le nostre vite", intervento nel quale il professore ha parlato delle possibili conseguenze che questa rivoluzione, la rivoluzione del *Web Squared*, potrà avere sui contenuti del web e sulla loro organizzazione. Parla di come "Ora un nuovo livello di informazioni automatiche [...]" potrà arricchire l'insieme dei contenuti. Ma quali saranno gli sviluppi della grafica?

Questa tesi nasce dal progetto di costruire un sito internet che modelli proprio la sua grafica sull'ambiente reale che circonda chi lo visita.

Nel primo capitolo viene spiegato il ruolo della localizzazione geografica dell'utente nel cambiamento dell'aspetto grafico e vengono spiegati i mezzi grazie ai quali si riesce a venire a conoscenza di questa informazione e a utilizzarla per il nostro scopo.

Nel secondo capitolo viene spiegato l'uso del servizio meteorologico *Yahoo! Weather RSS Feed.*

Nel terzo capitolo viene esposto il ruolo del tempo cronologico nelle modifiche grafiche del sito, quali elementi influiscono e in che modo.

Il quarto capitolo tratta degli aspetti grafici che vengono modificati, in che modo e dell'uso dei CSS 3.

Nel quinto capitolo viene motivata la presenza di un pannello di controllo e vengono spiegate le sue funzionalità.

1.La localizzazione dell'utente

Per poter modellare la grafica del sito sull'ambiente che circonda il navigatore è necessario rintracciare, per prima cosa, la sua posizione geografica. Questa non influisce direttamente sugli elementi grafici del sito ma condiziona alcuni aspetti come il meteo, le stagioni e le quantità di ore di luce nella giornata.
La localizzazione dell'utente è molto usata sul web: ad esempio i motori di ricerca, come *Google* e *Yahoo*, le grandi multinazionali come *Apple*, *Ikea* e i siti *e-commerce,* come *eBay*, che spesso hanno una versione del sito per ogni stato o continente, reindirizzano automaticamente il navigatore alla versione della pagina corrispondente al suo paese di provenienza. Spesso anche i web *banners* utilizzano la localizzazione dell'utente per inserire nelle pagine visitate la pubblicità di prodotti o servizi appartenenti al paese dal quale proviene l'utente connesso.
In altri casi, nei quali non è necessario conoscere il paese di provenienza, i contenuti dei siti vengono modificati in base ad alcune informazioni sul sistema operativo dell'utente, ad esempio la lingua: se la lingua del sistema operativo è l'inglese il sito reindirizzerà direttamente l'utente alla versione inglese del sito.

1.1.La ricerca degli strumenti

Per localizzare l'utente i metodi utilizzabili erano due: il Global Positioning System (GPS) o il sistema di localizzazione per mezzo dell'indirizzo IP.
Se è scontato che quasi tutti i cellulari di ultima generazione, gli *smartphone*, abbiano il GPS integrato, non lo è altrettanto per i *laptop* o i *desktop*, soprattutto se non sono l'ultimo modello uscito sul mercato. Dunque la scelta del GPS avrebbe ristretto troppo il *target* di utenti a cui il sito avrebbe potuto essere destinato.
Per questo si è scelto di utilizzare il sistema di localizzazione geografica basato sull'analisi dell'indirizzo IP.
L'indirizzo IP è un numero che identifica ogni dispositivo collegato a una rete informatica. La versione attualmente in uso è la IPv4 a 32 bit ma a causa della smisurata crescita della rete è stato necessario aumentare gli indirizzi. Per questo è stata sviluppata la versione IPv6 a 128 bit che corrispondono a 32 cifre esadecimali, ma questa versione non è ancora pienamente in uso a causa di mappe di instradamento complesse.
Gli indirizzi IPv4, tranne rare eccezioni, sono globalmente univoci e non possono essere scelti in modo arbitrario. La strategia di assegnazione degli indirizzi internet è detta CIDR (*Classless Interdomain Routine*). L'indirizzo IP presenta la forma

decimale puntata *a.b.c.d/x* dove *x* indica il numero di bit della prima parte dell'indirizzo che costituiscono la porzione di rete dell'indirizzo e sono detti *prefisso di rete dell'indirizzo*. A un'organizzazione generalmente viene assegnato un intervallo d'indirizzi con prefisso comune per tutti gli indirizzi IP dei dispositivi che si trovano al suo interno. I *router* esterni alla rete dell'organizzazione considerano solo gli *x* bit del prefisso, saranno i *router* della rete interna a indirizzare i pacchetti verso il giusto dispositivo.
Prima dell'adozione di *CIDR* gli indirizzi venivano classificati in base alla lunghezza delle loro parti di rete: 8, 16, 24 bit che appartenevano rispettivamente alle classi A, B, o C. Questo sistema era noto come *classfull addressing.*
Per risalire al luogo di provenienza è necessario avere a disposizione dei *databases* che associno indirizzo IP a regione o provincia di provenienza.
I servizi che svolgono questo compito sono numerosi e offrono varie soluzioni.
Una delle possibilità è di scaricare il *database* e istallarlo sul proprio *server.* Ne esistono due versioni: una gratuita che, però, è molto poco precisa nella localizzazione della provenienza dell'indirizzo IP e spesso, per i paesi fuori dagli Stati Uniti, nelle tabelle non sono registrate le città; una a pagamento il cui prezzo può variare da circa $ 50 per un *database* che associa solo indirizzo IP a stato, fino ad arrivare a più di $ 1000 per uno che associa l'indirizzo IP alla città di provenienza e dà informazioni circa la latitudine, la longitudine, il fuso orario, codice postale.
La dimensione del *database* è elevata, quello a pagamento può arrivare fino a 1GB per la versione più completa, questo potrebbe comportare delle difficoltà nell'istallazione sui *server* e tempi lenti nella consultazione.
Per evitare l'istallazione sul server è possibile anche consultare il *database on line* per un numero limitato di richieste. Se le richieste sono inferiori a venticinque IP per una ricerca il servizio è gratuito, altrimenti è possibile comprare un abbonamento il cui prezzo varia in base a:

- accuratezza della versione del *database* che si vuole interrogare
- quantità delle interrogazioni necessarie.

Prima di scegliere questa soluzione è opportuno fare una stima del numero delle visite che si prevede potrebbe avere il sito sul quale vogliamo localizzare il visitatore e valutarne la convenienza.
Nelle tabelle che seguono sono riassunti alcuni dei servizi utilizzabili. La prima tabella elenca alcune delle diverse versioni dei *databases* scaricabili.
La seconda elenca i diversi servizi offerti usabili senza scaricare il database.

database	costo	quantità	accuratezza	dimensione
Servizio 1	$ 0	∞	scarsa	101MB
Servizio 2	$ 499	∞	buona	525MB

Tabella 1 Database a confronto

queries	costo	quantità	accuratezza	dimensione
Servizio 3	$ 0	25	buona	-
Servizio 4	$ 20	50000	buona	-
Servizio 5	$ 50	125000	buona	-

Tabella 2 Abbonamenti a confronto

Per questo progetto, dopo la valutazione dei diversi servizi offerti, è stato scelto www.maxmind.com che fornisce un risultato piuttosto preciso nella localizzazione: ad esempio l'indirizzo IP 93.47.2.199 viene localizzato a Viareggio mentre la sua posizione reale è Massa, l'errore è di circa 20Km. Altri servizi localizzano questo indirizzo IP a Livorno, a Prato o persino Milano.
Si è scelto di acquistare cinquantamila *queries* per 20$ poiché questo significa localizzare cinquantamila utenti che si collegheranno al sito.

1.2.Come avviene la localizzazione dell'utente.

Con la localizzazione vengono reperiti la città di provenienza dell'utente, la longitudine e la latitudine che verranno utilizzati successivamente per modificare le impostazioni grafiche del sito.
In questa prima fase il linguaggio di programmazione utilizzato è *PHP*.
Per ogni utente che si collega, il sito registra il suo indirizzo IP per mezzo della variabile *superglobale* `$_SERVER` che contiene tutte le informazioni legate all'ambiente di esecuzione dello *script* .

```
$ip=$_SERVER['REMOTE_ADDR'];
```

La variabile `$ip` viene passata come parametro al servizio di localizzazione, il quale interroga il database e restituisce un file *XML* con tutte le informazioni relative all'indirizzo IP inviato. Ad esempio per l'indirizzo IP 93.47.2.199 restituisce:

```
<?xml version="1.0" encoding="UTF-8"?>
<Locations>
  <Location id="0">
    <Ip>93.47.2.199</Ip>
    <Status>OK</Status>
    <CountryCode>IT</CountryCode>
    <CountryName>Italy</CountryName>
    <RegionCode>16</RegionCode>
    <RegionName>Toscana</RegionName>
    <City>Viareggio</City>
    <ZipPostalCode></ZipPostalCode>
    <Latitude>43.8667</Latitude>
    <Longitude>10.2333</Longitude>
    <TimezoneName>Europe/Rome</TimezoneName>
    <Gmtoffset>7200</Gmtoffset>
    <Isdst>1</Isdst>
  </Location>
</Locations>
```

Questo file *XML* viene analizzato da una funzione che costruisce un *array* che, per ogni elemento radice, conterrà un ulteriore *array* con tutti i suoi figli. Per accedere all'elemento desiderato basterà scorrerlo rispettando la struttura gerarchica del documento *XML*.

```
$city = $arr['Locations']['Location']['City'];
$latitudine=$arr['Locations']['Location']['Latitude'];
$longitude=$arr['Locations']['Location']['Longitude'];
```

Questa funzione sarà utilizzata anche in seguito per analizzare ogni documento *XML* presente nel progetto.
I dati ottenuti, nome della città, latitudine e longitudine, verranno utilizzati successivamente per reperire nuove informazioni necessarie per lo sviluppo della grafica.

2. Il servizio meteo

2.1. La ricerca degli strumenti

Poiché la grafica del sito sarà influenzata da agenti esterni, un fattore molto evidente e importante è il tempo meteorologico.
Quasi tutti i siti, che forniscono le previsioni meteorologiche, offrono la possibilità agli sviluppatori web di inserire gratuitamente all'interno del loro sito un modulo per le previsioni meteorologiche di una località. Alcuni di questi, ad esempio il sito www.ilmeteo.it, offre un abbonamento annuale che consente di ottenere dati numerici in xml che possono essere manipolati. I costi dell'abbonamento, riportati nella tabella che segue, variano in base alla quantità di località a cui siamo interessati.

Servizio	costo	durata
Dati meteo XML Giornalieri + Triorari per 7gg 1 località	€125,00+iva	1 anno
Dati meteo XML + mari e venti XML 1 località	€150,00+iva	1 anno
Dati meteo XML Giornalieri + Triorari 7gg TUTTE Province	€2500,00+iva	1 anno
Dati meteo XML Giornalieri + Triorari 7gg TUTTI Comuni	€9000,00+iva	1 anno

Tabella 3 Riassunto dei servizi meteorologici

Un servizio gratuito è quello offerto da *Yahoo Developer Network* che offre *Api (Application Programming Interface)* e servizi web.
Le Api sono un insieme di procedure messe a disposizione degli sviluppatori dai creatori di un programma o di un'applicazione che permettono di estendere le funzionalità di tale programma senza modificare l'intero codice.

2.2. Cos'è e come viene utilizzato GeoPlanet

GeoPlanet è una risorsa che gestisce tutti i nomi dei luoghi della terra, è stato ideato per facilitare l'interoperabilità spaziale e le scoperte geografiche.
Ogni luogo geografico del mondo è identificato da *GeoPlanet* con un codice numerico a 32-bit: *Where On Earth Identifiers* (WOEID). Una volta assegnato a un luogo, questo codice non viene mai cambiato o riciclato.
Prima di poter utilizzare il servizio, *Yahoo* richiede un'iscrizione dove è richiesta una breve descrizione del progetto nel quale verrà utilizzata questa risorsa.
All'utente verrà assegnato un codice identificativo che sarà inserito come parametro nella richiesta assieme alla città di cui vogliamo conoscere il WOEID.

```
$url2="http://where.yahooapis.com/v1/places.q(roma)?
appid=3bEXXDfV34GcM.NF4QdLOVRpmB5wELZaps6NskT3o556IqTIpE8ESsgj
2CG_vQg0.Wc0h8_i8ucQcW782P1iQYVSstvb";
```

Questa richiesta restituisce un file *XML*, riportato qui di seguito, da cui la funzione già utilizzata per la localizzazione dell'indirizzo IP ricava il WOEID.

```
<?xml version="1.0" encoding="UTF-8"?>
<places xmlns="http://where.yahooapis.com/v1/schema.rng"
xmlns:yahoo="http://www.yahooapis.com/v1/base.rng"
yahoo:start="0" yahoo:count="1" yahoo:total="32">
  <place yahoo:uri="http://where.yahooapis.com/v1/place/
721943" xml:lang="it-it">
    <woeid>721943</woeid>
    <placetypename code="7">Città</placetypename>
    <name>Roma</name>
    <country type="Paese" code="IT">Italia</country>
    <admin1 type="Regione" code="">Lazio</admin1>
    <admin2 type="Provincia" code="IT-RM">Roma</admin2>
    <admin3></admin3>
    <locality1 type="Citt&agrave;">Roma</locality1>
    <locality2></locality2>
    <postal></postal>
    <centroid>
      <latitude>41.903111</latitude>
      <longitude>12.495760</longitude>
    </centroid>
    <boundingbox>
      <southwest>
        <latitude>41.802910</latitude>
        <longitude>12.379420</longitude>
```

```
      </southwest>
      <northeast>
        <latitude>41.991852</latitude>
        <longitude>12.615890</longitude>
      </northeast>
    </boundingbox>
      <arearank>5</arearank>
      <poprank>13
      </poprank>
  </place>
</places>
```

2.3. Il servizio meteo Yahoo! Weather RSS Feed

Una volta che il codice WOEID è stato reperito si può procedere con l'invio della richiesta al servizio meteorologico di *Yahoo: Yahoo! Weather RSS Feed* .

```
$arr3 = xml2array("http://weather.yahooapis.com/forecastrss?
w=".$arr2['places']['place']['woeid']);
```

La richiesta ha come parametro il codice WOEID della città in cui si trova il visitatore e la risposta è ancora una volta un documento *XML* che viene nuovamente analizzato dalla stessa funzione `xml2array()` che trasforma la gerarchia del file *XML* in una serie di *array* annidati uno dentro l'altro.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<rss version="2.0" xmlns:yweather="http://
xml.weather.yahoo.com/ns/rss/1.0" xmlns:geo="http://
www.w3.org/2003/01/geo/wgs84_pos#">
<channel>
  <title>Yahoo! Weather - Sunnyvale, CA</title>
  <link>http://us.rd.yahoo.com/dailynews/rss/weather/
Sunnyvale__CA/*http://weather.yahoo.com/forecast/
USCA1116_f.html</link>
  <description>Yahoo! Weather for Sunnyvale, CA</description>
  <language>en-us</language>
  <lastBuildDate>Fri, 18 Dec 2009 9:38 am PST</lastBuildDate>
  <ttl>60</ttl>
  <yweather:location city="Sunnyvale" region="CA"
country="United States"/>
  <yweather:units temperature="F" distance="mi" pressure="in"
speed="mph"/>
  <yweather:wind chill="50"   direction="0"   speed="0" />
```

```
  <yweather:atmosphere humidity="94"  visibility="3"
pressure="30.27"  rising="1" />
  <yweather:astronomy sunrise="7:17 am"   sunset="4:52 pm"/>
  <image>
    <title>Yahoo! Weather</title>
    <width>142</width>
    <height>18</height>
    <link>http://weather.yahoo.com</link>
    <url>http://l.yimg.com/a/i/us/nws/th/main_142b.gif</url>
  </image>
  <item>
    <title>Conditions for Sunnyvale, CA at 9:38 am PST</title>
    <geo:lat>37.37</geo:lat>
    <geo:long>-122.04</geo:long>
    <link>http://us.rd.yahoo.com/dailynews/rss/weather/
Sunnyvale__CA/*http://weather.yahoo.com/forecast/
USCA1116_f.html</link>
    <pubDate>Fri, 18 Dec 2009 9:38 am PST</pubDate>
    <yweather:condition  text="Mostly Cloudy"  code="28"
temp="50"  date="Fri, 18 Dec 2009 9:38 am PST" />
    <description><![CDATA[
<img src="http://l.yimg.com/a/i/us/we/52/28.gif"/><br />
<b>Current Conditions:</b><br />
Mostly Cloudy, 50 F<BR />
<BR /><b>Forecast:</b><BR />
Fri - Partly Cloudy. High: 62 Low: 49<br />
Sat - Partly Cloudy. High: 65 Low: 49<br />
<br />
<a href="http://us.rd.yahoo.com/dailynews/rss/weather/
Sunnyvale__CA/*http://weather.yahoo.com/forecast/
USCA1116_f.html">Full Forecast at Yahoo! Weather</a><BR/><BR/>
(provided by <a href="http://www.weather.com" >The Weather
Channel</a>)<br/>
]]></description>
    <yweather:forecast day="Fri" date="18 Dec 2009" low="49"
high="62" text="Partly Cloudy" code="30" />
    <yweather:forecast day="Sat" date="19 Dec 2009" low="49"
high="65" text="Partly Cloudy" code="30" />
    <guid isPermaLink="false">USCA1116_2009_12_18_9_38_PST</
guid>
  </item>
</channel>
</rss>
```

Le informazioni che ci fornisce il servizio meteorologico sono numerose: oltre alle condizioni del tempo atmosferico, la longitudine, la latitudine, l'ora locale.

L’elemento del documento *XML* che contiene le condizioni atmosferiche è:

```
<yweather:condition  text="Mostly Cloudy"  code="28"
temp="50"  date="Fri, 18 Dec 2009 9:38 am PST" />
```

nell’attributo `text=””` è contenuta la descrizione delle condizioni del tempo, l’attributo `code=””` contiene il codice identificativo di tali condizioni. La corrispondenza tra codice e condizione atmosferica è riportata nella tabella seguente presente anche nella documentazione del servizio.

Codice	Descrizione
0	tornado
1	tropical storm
2	hurricane
3	severe thunderstorms
4	thunderstorms
5	mixed rain and snow
6	mixed rain and sleet
7	mixed snow and sleet
8	freezing drizzle
9	drizzle
10	freezing rain
11	showers
12	showers
13	snow flurries
14	light snow showers
15	blowing snow
16	snow
17	hail
18	sleet
19	dust
20	foggy
21	haze
22	smoky

Codice	Descrizione
23	blustery
24	windy
25	cold
26	cloudy
27	mostly cloudy (night)
28	mostly cloudy (day)
29	partly cloudy (night)
30	partly cloudy (day)
31	clear (night)
32	sunny
33	fair (night)
34	fair (day)
35	mixed rain and hail
36	hot
37	isolated thunderstorms
38	scattered thunderstorms
39	scattered thunderstorms
40	scattered showers
41	heavy snow
42	scattered snow showers
43	heavy snow
44	partly cloudy
45	thundershowers
46	snow showers
47	isolated thundershowers
3200	not available

Tabella 4 Corrispondenza fra codice e condizione atmosferica

Come si può notare, le descrizioni delle condizioni atmosferiche sono molto dettagliate, ad esempio in alcuni casi distinguono fra temporali e temporali isolati. Poiché per lo sviluppo del progetto non è necessaria tale precisione si è creduto utile raggruppare le condizioni meteorologiche in sette gruppi , contrassegnati da una

lettera dell'alfabeto, A o B o C o D o E o F o L, all'interno di una tabella in un *database*.
Quando dal documento *XML* viene estratto il codice numerico, riportato nell'immagine sopra, della condizione atmosferica, una funzione invia una *query* al database:

```
mysql_query("SELECT class,nome FROM weather WHERE id='$code'")
```

che seleziona il campo `class` che contiene la lettera del gruppo a cui appartiene la condizione meteorologica restituita dal servizio meteo di *Yahoo* e modificherà di conseguenza gli elementi della grafica che dipendono dal tempo atmosferico di cui si parlerà più avanti.

3.Il tempo cronologico

Il tempo cronologico è il fattore che influisce sulla grafica del sito in maniera più evidente ed è strettamente legato alla posizione geografica dell'utente, basti pensare ai fusi orari, alle diverse stagioni nei due emisferi. I dati cronologici necessari sono: la data, l'ora del giorno, la quantità di ore di luce nella giornata, l'ora in cui sorge il sole e l'ora in cui tramonta il sole.

3.1.La ricerca degli strumenti

I dati necessari sono tutti reperibili attraverso il servizio meteorologico di Yahoo, ma appoggiarsi totalmente a questa tecnologia avrebbe significato rinunciare a tutte le funzionalità del sito qual ora, a causa di qualsiasi problema tecnico, la connessione con il servizio non fosse disponibile.
La soluzione alternativa è stata quella di utilizzare un *JavaScript*, sviluppato da P. Lutus disponibile sul sito www.exptech.com. Un vantaggio notevole di questa soluzione è che i dati utilizzati per calcolare la quantità di ore di luce nella giornata, l'ora in cui sorge e tramonta il sole, possono essere inseriti manualmente attraverso un modulo che passa poi i parametri alle diverse funzioni.

3.2.Come funziona il modulo originale

Il primo dato che richiede il modulo è una posizione geografica.

E’ possibile scegliere una località geografica fra le opzioni proposte dal modulo stesso, ad esempio, una città degli Stati Uniti o una capitale del mondo o, ancora, un aeroporto. Quando l’utente sceglie un luogo del quale vuole sapere a che ora sorge e tramonta il sole, la funzione `list_pos()` prende i valori della longitudine e della latitudine contenuti nell’attributo `value=""` del *tag* `<option></option>`, li passa alle funzioni `doit()`, `dt_hms()` e `dd_dm()` che svolgono i calcoli.

```
<option value=" 5.288, 3.994">Abidjan</option>

function dd_dm(x,f) {
        var d,m,sgn;
        var r;
        sgn = (x>=0)?0:1;
        x = Math.abs(x)+.008333;
        d = Math.floor(x);
        x = (x-d) * 60;
        m = Math.floor(x);
        d = fix_places(d);
        m = fix_places(m);
        if(f == 0) {
                r = d;
        }
        if(f == 1) {
                r = m;
        }
        if(f == 2) {
                r = sgn;
        }
        return r;
}
function dt_hms(x,amdsp,dst) {
        var h,m,s;
        var ampm;
        var outstr;
        if((x > -100) && (x < 100)) {
                x = (x < 0)?0:x;
                h = Math.floor(x);
                x = (x-h) * 60;
                h = (h+dst) % 24; // daylight time adjustment
                m = Math.floor(x);
                x = (x-m) * 60;
                s = Math.floor(x);
                if(amdsp > 0) {
                        ampm = (h > 11)?" PM":" AM";
                        h = h % 12;
```

```
                        h = (h < 1)?12:h;
                }
                else {
                        ampm = "";
                }
                h = fix_places(h);
                m = fix_places(m);
                s = fix_places(s);
                outstr = h + ":" + m + ":" + s + ampm;
        }
        else {
                outstr = (x < 0)?"[Below]":"[Above]";
        }
        return outstr;
}

function list_pos(w) {
        var str,place,desc,lats,lngs,i;
        i = w.options.selectedIndex;
        with(document.all) { // reset all prior selections
                usa_city.options[0].selected = 1;
                usa_cap.options[0].selected = 1;
                world_city.options[0].selected = 1;
                world_cap.options[0].selected = 1;
                airports.options[0].selected = 1;
        }
        w.options[i].selected = 1; // restore current
selection
        with (w) {
                desc = options[0].text;
                str = options[options.selectedIndex].value;
                place = options[options.selectedIndex].text;
        }
        i = str.indexOf(",");

        lats = str.substring(0,i);
        lngs = str.substring(i+1,str.length);

        lat = parseFloat(lats);
        lng = parseFloat(lngs);

        if((lat != 0) && (lng != 0)) {
                dtime = -Math.floor((lng+7.5)/15);
                document.all.tz.options[12+dtime].selected =
1;
                doit("(" + desc + ") " + place);
```

```
        }
}

function doit(title)
    {
         var jd,mo,da,yr;

         var s = new pos;

         var r = new rts;

         if(title != "")
              {
                 document.all.placename.value = title;
              }

        if(title == "(map position)")
         {
               document.all.tz.options[12+m_dtime].selected =
1;

              lat = m_lat;

              lng = m_lng;

              document.all.placename.value =
show_mouse_pos(lat,lng,title);
         }

document.all.latd.value = dd_dm(lat,0);

document.all.latm.value = dd_dm(lat,1);

document.all.lats[dd_dm(lat,2)].checked = 1;

document.all.lngd.value = dd_dm(lng,0);

document.all.lngm.value = dd_dm(lng,1);

document.all.lngs[dd_dm(lng,2)].checked = 1;

dtime = - (12-document.all.tz.options.selectedIndex);

mo = month + 1;

da = day + 1;
```

```
yr = year;

jd = mdy_jd(mo,da,yr,0,0,0);

s = calcsun(jd,0,0,0);


r = rmstime(jd,lat,lng,dtime,SUNSET,s.lng,s.lat);

document.all.sunrise1.value = dt_hms(r.rise,ampm,dst);

document.all.suntransit.value = dt_hms(r.transit,ampm,dst);

document.all.sunset1.value = dt_hms(r.set,ampm,dst);

document.all.sunlen1.value = dt_hms(r.set-r.rise,0,0);
```

L'altra possibilità è quella, se la latitudine e la longitudine di un luogo sono già note, di inserirle manualmente all'interno dei campi di testo. In questo caso viene richiamata la funzione `man_pos()` che prende i valori dal campo di testo e non dall'attributo di `<option></option>` e li passa alle funzioni riportate sopra.

```
function man_pos()
  {
    lat = Math.abs(parseFloat(document.all.latd.value) +
         (parseFloat(document.all.latm.value)/60));
    if(document.all.lats[1].checked)
        {
          lat = -lat;
        }
    lng = Math.abs(parseFloat(document.all.lngd.value) +
(parseFloat(document.all.lngm.value)/60));
    if(document.all.lngs[1].checked)
       {
         lng = -lng;
       }
    dtime = -Math.floor((lng+7.5)/15);
    document.all.tz.options[12+dtime].selected = 1;
    doit("(manual entry)");
     return 1;
}
```

Un altro dato importante è la data. Di *default* viene impostata la data del giorno corrente:

```
function set_date_vars(title)
  {
    month = document.all.month.selectedIndex;
    day = document.all.day.selectedIndex;
    dst = ((month >= 2) && (month <= 9))?1:0;
    document.all.dst.checked = dst;
    doit(title);
  }

function set_default_date()
  {
    var now = new Date();
    var m = now.getMonth();
    var d = now.getDate();
    year = now.getYear();
    document.all.month.selectedIndex = m;
    document.all.day.selectedIndex = d - 1;
    set_date_vars("");
}
```

La funzione utilizza l'oggetto *JavaScript* `Date()` per prendere il mese

`var m = now.getMonth();`,

il giorno, `var d = now.getDate();`, e l'anno, `year = now.getYear();` e seleziona, poi, nell'option del form l'indice corrispondente al mese:

```
document.all.month.selectedIndex = m;
```

al giorno:

```
document.all.day.selectedIndex = d - 1;.
```

La funzione `set_date_vars();` assegna alle variabili `month` e `day` il valore selezionato nel modulo, questo permette all'utente di scegliere una data e di verificare, ad esempio, come cambiano le ore dell'alba e del tramonto durante l'anno in una data località.
Questo modulo verrà ripreso e modificato per calcolare i fattori che influenzano la grafica del sito

3.3. Com'è stato integrato e modificato il codice all'interno del progetto

Lo scopo dell'uso di questo codice *JavaScript* è quello di calcolare l'ora dell'alba, quella del tramonto e la quantità di ore di luce nella giornata della località in cui si trova il visitatore.
Per fare ciò è stato necessario modificare il codice, spiegato nel paragrafo precedente, in modo da eliminare il modulo che doveva essere compilato dal visitatore, poiché le informazioni necessarie ai calcoli sono fornite automaticamente dal codice *PHP* che rintraccia la latitudine e la longitudine della sua posizione. Questi dati, contenuti in variabili *PHP*, devono essere trasformati in variabili leggibili e manipolabili da *JavaScript*.
PHP è un linguaggio di programmazione *lato server,* questo significa che il codice programmato con questo linguaggio viene eseguito direttamente sul *server* e produce, come *output*, linguaggio *HTML. JavaScript,* invece, è un linguaggio di programmazione *lato client,* questo significa che lo *script* viene eseguito dal *browser.* Dunque per inserire il contenuto di una variabile dello *script PHP* in una variabile *JavaScript* è necessario scrivere in tale variabile *JavaScript* la riga di codice *PHP* che fa stampare a video il contenuto della variabile: "`<? php` " apre ogni frammento di codice *PHP*; "`echo`" è il comando usato per stampare a video; "`$latitudine;`" contiene il valore che deve essere passato alla variabile *JavaScript*; "`?>`" chiude ogni *script PHP*.

```
var latitudine="<? php echo $latitudine; ?>";
var longitudine="<? php echo $longitude; ?>";
```

Una volta calcolate ora dell'alba, del tramonto e quantità di ore di luce sulla base delle coordinate geografiche della posizione dell'utente, lo *script* passa questi parametri alle funzioni che modificheranno, di conseguenza, la grafica come verrà descritto in seguito. Invece, tutte le funzioni del codice che servivano per gestire i dati ricavati dal modulo sono state eliminate insieme al modulo stesso.

4. I cambiamenti della grafica

4.1.L'uso dei *CSS3* e di *Prototype*

I *CSS*, dall'inglese *Cascading Style Sheet*, sono un insieme di regole, emanate per la prima volta nel 1996 dal *W3C*(*World Wide Web Consortium*), che servono per definire la rappresentazione di documenti *HTML*, *XHTML*, *XML*. Come tutti i linguaggi di programmazione, anche i *CSS* sono in continua evoluzione: nel 1998 il *W3C* emanò le specifiche *CSS 2* e nel 2004 le specifiche *CSS 2.1*.
I *CSS 3* sono stati definiti per la prima volta dal W3C nel 1999 ma tutt'oggi non sono state ancora emanate le specifiche ufficiali, sono ancora in fase di sperimentazione, ma ciò non ha impedito ad alcuni *browser* quali Opera, Firefox e Safari di implementare alcune utili proprietà.
Per questo si è deciso di utilizzarli nello sviluppo del progetto che può essere definito sperimentale.
Le novità principali dei CSS 3 rispetto alle versioni precedenti sono i *moduli*, ovvero gruppi di proprietà affini e con scopi simili, che consentiranno un'evoluzione più flessibile e veloce delle specifiche.
Prototype è un *framework JavaScript* che ha lo scopo di facilitare lo sviluppo di applicazioni web dinamiche. Questa libreria è stata sviluppata da Sam Stephenson ed è apparsa per la prima volta nel 2005. Uno dei vantaggi che ha contribuito al suo successo è il facile controllo che essa può avere sugli elementi HTML della pagina e sui *CSS*

4.2. Le modifiche degli elementi grafici

Seguendo l'evoluzione del web, questo sito tende a far incontrare il mondo virtuale e il mondo reale cambiando la sua grafica in base alle condizioni atmosferiche della posizione geografica del visitatore e al tempo cronologico.

4.2.1.I cambiamenti grafici che dipendono dalle condizioni meteorologiche

Le condizioni meteorologiche agiscono sullo sfondo del div chiamato *"header_tutto"*.
Per ogni lettera alfabetica della tabella del *database* che raggruppa più condizioni atmosferiche, è stato creato uno sfondo che richiama gli agenti atmosferici, ad

esempio quando la *query* al database restituisce la classe A che raggruppa temporali, uragano e tornado viene caricato come sfondo nel *div#header_tutto* un'immagine che rappresenta delle nuvole nere e minacciose.

Quando il codice *PHP*, analizzato nel paragrafo 2.3 estrae dalla tabella del database il gruppo a cui appartiene la condizione meteorologica relativa alla posizione del visitatore questo risultato viene analizzato dalla seguente funzione:

```
switch (xml){

  case "A":
    $$
('#header_tutto.demo_sfondo'[0].style.backgroundImage="url('im
magini/temp_a.png')";
    $$('#header_tutto.demo_sfondo')
[0].style.backgroundRepeat="no-repeat";
    break;

  case "B":
    $$('#header_tutto.demo_sfondo')
  [0].style.backgroundImage="url('immagini/temp_b.png')";
    $$('#header_tutto.demo_sfondo')
[0].style.backgroundRepeat="no-   repeat";
    break;

  case "c":
    $$('#header_tutto.demo_sfondo')
[0].style.backgroundImage="url('immagini/temp_c.png')";
    $$('#header_tutto.demo_sfondo')
[0].style.backgroundRepeat="no-repeat";
    break;

  case "D":
    $$('#header_tutto.demo_sfondo')
[0].style.backgroundImage="url('immagini/temp_d.png')";
    $$('#header_tutto.demo_sfondo')
[0].style.backgroundRepeat="no-repeat";
    break;

  case "E":
    $$('#header_tutto.demo_sfondo')
[0].style.backgroundImage="url('immagini/temp_e.png')";
```

```
    $$('#header_tutto.demo_sfondo')
[0].style.backgroundRepeat="no-repeat";
    break;

  case "F":
    $$('#header_tutto.demo_sfondo')
[0].style.backgroundImage="url('immagini/temp_f.png')";
    $$('#header_tutto.demo_sfondo')
[0].style.backgroundRepeat="no-repeat";
    break;
  case "L":
    $$('#header_tutto.demo_sfondo')
[0].style.backgroundImage="url('immagini/temp_l.png')";
    $$('#header_tutto.demo_sfondo')
[0].style.backgroundRepeat="no-repeat";
    break;

    default:
      alert ("Siamo spiacenti ma le informazioni meteo sulla
tua località non sono disponibili");
}
```

Questa funzione analizza il contenuto della variabile *xml* che contiene la lettera associata al gruppo di agenti atmosferici nel database.
Ad ogni lettera associa il comando `$$('#header_tutto.demo_sfondo')[0].style.backgroundImage="url('immagini/temp_l.png')";` per cambiare l'immagine di sfondo dell'elemento interessato.
Questo comando utilizza il framework Prototype nella sua prima parte per selezionare il div della pagina HTML che come identificatore il nome "header_tutto" e come classe "demo_sfondo". È stato necessario inserire l'indice `[0]` perché il comando `$$()` può selezionare qualsiasi selettore CSS e restituisce un *array* che contiene tutti gli elementi HTML che corrispondono a esso. In questo caso, poiché l'attributo *id=""* dell'elemento HTML è un identificatore univoco può esistere un solo elemento che abbia `id="header_tutto"` e `class="demo_sfondo"` non è necessario utilizzare un *ciclo for* per poter modificare tutti gli elementi che sono stati inseriti dentro l'*array*.

4.2.2.I cambiamenti grafici che dipendono dal tempo cronologico

I cambiamenti grafici che dipendono dal tempo cronologico riguardano l'ombra degli elementi e la luminosità dei colori di sfondo.

Utilizzando la nuova versione sperimentale dei fogli di stile, i CSS 3, è possibile realizzare l'effetto ombra degli elementi, non più utilizzando un'immagine come accadeva nelle versioni precedenti, ma semplicemente con il comando:

```
-webkit-box-shadow:rgba(0, 0, 0, 0.7) 5px 0px 10px;
-moz-box-shadow: rgba(0, 0, 0, 0.7) 5px 0px 10px;
```

la prima riga serve per visualizzare l'effetto grafico nei browser Webkit Safari e Google Chrome, la seconda riga serve per visualizzare l'effetto nel browser Mozilla Firefox.
I parametri `rgba(0, 0, 0, 0.7)` del comando definiscono il colore dell'ombra e la sua opacità, mentre `5px 0px 10px` definiscono rispettivamente lo spostamento dell'ombra in senso orizzontale: se è positivo l'ombra viene visualizzata a destra, se è negativo a sinistra; il secondo definisce lo spostamento dell'ombra in senso verticale: se è positivo l'ombra viene visualizzata in basso, se è negativo in alto; il terzo definisce la sfocatura dell'ombra, più il valore sarà basso più il bordo dell'ombra sarà netto, se il valore aumenta il bordo dell'ombra sarà più sfumato.
I CSS 3, oltre ai metodi tradizionali già supportati dalle versioni precedenti, hanno introdotto anche nuovi modi per definire il colore: *HSL*(*Hue, Saturation, Lightness*) e *HSLA* (*Hue, Saturation, Lightness, Alpha channel*) che definiscono i colori impostando i valori della tonalità (*Hue*), della saturazione(*Saturation*), della luminosità (*Lightness*) e anche il canale *Alpha* che determina il grado di opacità, *RGBA*(*Red Green Blu Alpha Channel*).

4.2.3.Il cambiamento delle stagioni

Il passaggio delle stagioni viene sottolineato cambiando il colore dello sfondo della pagine, il colore del menu e dei titoli e le immagini decorative presenti nella pagina. Per ogni stagione sono state utilizzate delle tonalità che richiamassero i suoi colori caratteristici: colori freddi per l'inverno come l'azzurro chiaro, caldi per l'autunno come il giallo, il rosso, l'arancione, colori floreali per la primavera e frizzanti per l'estate.
La funzione `set_default_date();` visualizza la data del sistema operativo del browser dell'utente. La data viene utilizzata nella funzione `check_season()` per definire in quale stagione ci si trova. La funzione controlla, inoltre, se la latitudine della posizione dell'utente è positiva o negativa. Se è positiva significa che l'utente si trova nell'emisfero nord, se è negativa significa che l'utente si trova nell'emisfero sud e le stagioni sono invertite, quindi a dicembre sarà estate e ad agosto sarà inverno.

Per ogni stagione sono impostati il colore di sfondo che la pagina deve avere durante la stagione corrispondente, il valore minimo della luminosità e il suo valore massimo che vengono poi riutilizzati nella funzione `ombra_change()` che verrà spiegata nel prossimo paragrafo.

```
check_season= function()
  {
    if(lat>0)
      {
        if ( ( (month>=0)&&(month<2) ) ||
  ( (month==11)&&(day>=20) )||( (month==2)&&(day<20) ) )
      {
            colore_base="hsl(176, 20%, ";
            lum_I=53;
            lum_F=76;
            ombra_change(colore_base, lum_I, lum_F );
          }

        if ( ( (month>=3)&&(month<5) ) ||
( (month==2)&&(day>=20) ) || ( (month==5)&&(day<20) ) )
          {
            colore_base="hsl(62, 89%, ";
            lum_I=15;
            lum_F=38;
            background_url="(immagini/primavera.jpg) ";
            ombra_change(colore_base, lum_I, lum_F );
          }

      if ( ( (month>=6)&&(month<8) ) ||
( (month==5)&&(day>=20) ) ||( (month==8)&&(day<20) ) )
        {
          colore_base="hsl(337, 72%, ";
          lum_I=32;
          lum_F=55;
          ombra_change(colore_base, lum_I, lum_F );
        }

      if ( ( (month>=9)&&(month<11) ) ||
( (month==8)&&(day>=20) )     ||
(month==11)&&(day<20) ) )
        {
          colore_base="hsl(42, 100%, ";
          lum_I=32;
          lum_F=55;
          ombra_change(colore_base, lum_I, lum_F );
```

```
        }
  }
```

Nell’emisfero sud le stagioni sono invertite.

```
/*--------------------emisfero_sud-----------------*/
else
  {
    if ( ( (month>=0)&&(month<2) ) ||
( (month==11)&&(day>=20) )    || ( (month==2)&&(day<20) ) )
      {
        colore_base="hsl(337, 72%, ";
        lum_I=32;
        lum_F=55;
        ombra_change(colore_base, lum_I, lum_F );
      }

    if ( ( (month>=3)&&(month<5) ) ||
( (month==2)&&(day>=20) ) || ( (month==5)&&(day<20) ) )
      {
        colore_base="hsl(42, 100%, ";
        lum_I=32;
        lum_F=55;
        ombra_change(colore_base, lum_I, lum_F );
      }

    if ( ( (month>=6)&&(month<8) ) ||
( (month==5)&&(day>=20) )|| ( (month==8)&&(day<20) ) )
      {
        colore_base="hsl(176, 20%, ";
        lum_I=53;
        lum_F=76;
        ombra_change(colore_base,lum_I,lum_F );
      }

    if ( ( (month>=9)&&(month<11) ) ||
( (month==8)&&(day>=20) ) || ( (month==11)&&(day<20) ) )
      {
        colore_base="hsl(62, 89%, ";
        lum_I=15;
        lum_F=38;
        ombra_change(colore_base, lum_I, lum_F );

      }

    }
```

```
}
```

4.2.4.La gestione delle dimensioni della ombre e della luminosità dei colori

Potendo gestire le ombre e i colori in modo così preciso, grazie ai *CSS 3,* si è pensato di simulare il passaggio del tempo cronologico modificando proprio le ombre degli elementi e la luminosità dei colori in modo che potessero ricordare il cambiamento reale della luce durante il giorno e il movimento delle ombre causato dallo spostamento del sole.

Per simulare il cammino che il sole compie durante la giornata, l'ombra laterale degli elementi della pagina ha la sua estensione massima positiva all'alba, dieci *pixel*. Per ogni minuto che passa, questo valore viene decrementato fino a raggiungere zero *pixel* quando il sole ha raggiunto il mezzogiorno locale, dopodiché diventa negativo e l'ombra ricomincia a crescere nel lato sinistro degli elementi.

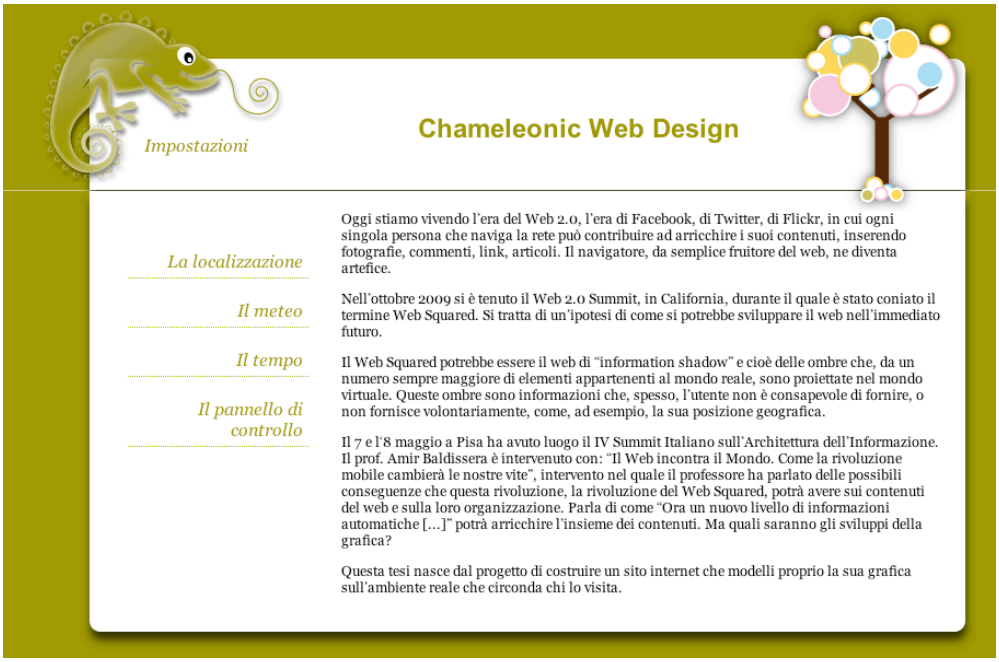
L'ombra verticale fa esattamente il contrario, cioè, la sua dimensione è zero *pixel* quando il sole sorge, raggiunge la sua estensione massima, ossia dieci *pixel*, quando il sole è nel punto più alto del suo cammino e decresce lentamente fino a tornare a essere zero *pixel* quando il sole tramonta.

Per quanto riguarda la luminosità viene usato lo stesso metodo: all'alba il suo valore è minimo e inizia ad aumentare fino a diventare massimo quando il sole raggiunge la sua massima altezza, a questo punto incomincia a diminuire il suo valore fino a tornare di nuovo minimo quando il sole è tramontato.

Durante la notte le ombre sono assenti e la luminosità del colore di sfondo è minima.

L'ombra laterale varia in tutto di venti *pixel*, dieci *pixel* la mattina e dieci *pixel* il pomeriggio, mentre la luminosità varia di ventitré punti la mattina e ventitré punti il pomeriggio per un totale di quarantasei punti.

1. Immagine del sito all'alba



2. Immagine del sito nel Mezzogiorno locale

3. Immagine del sito al tramonto

4. Immagine del sito nella notte

Per definire quanto variano ogni minuto i valori delle dimensioni delle ombre e le percentuali della luminosità del colore di sfondo, questi sono stati divisi per la quantità di minuti di luce in una giornata che sono stati calcolati dal codice analizzato nel paragrafo 2.2, insieme all'ora dell'alba e quella del tramonto anch'esse trasformate in minuti. Poiché l'ora dell'alba, del tramonto e la quantità di ore di luce sono riportate nel formato classico *hh:mm*, viene utilizzato il metodo `substring` dell'oggetto javascript `String` per inserire le ore in una variabile, trasformarle in minuti e sommarle ai minuti. Questa operazione viene fatta per tutti i formati delle ore.

```
//quantità di luce in minuti da alba a tramonto

var somma= parseFloat(h_m)+parseFloat(m_m);

// punti che si spostano ogni mintuto

var angolo=(20/somma);
var grad_lum=(46/somma);

var angolo_str= angolo+"";
var an_vir=angolo_str.indexOf(".");
var angolo_ok=parseFloat(angolo_str.substr(0, an_vir+3));

//minuto in cui sorge il sole
var alba_m= parseFloat(h_m_a)+parseFloat(m_m_a);

//minuto in cui tramonta il sole
var min_t=sunset.indexOf(":");

var tram_m= parseFloat(h_m_t)+parseFloat(m_m_t);
```

A questo punto sono necessari una serie di controlli di flusso che determinino se, quando l'utente si collega al sito, è mattina, è pomeriggio o è notte.
Se è mattina, la funzione prende le variabili che contengono la quantità di pixel che dovrebbe aumentare l'ombra verticale e diminuire l'ombra laterale col trascorrere dei minuti e la quantità di punti che dovrebbe aumentare la luminosità del colore e le moltiplica per i minuti trascorsi dall'alba, dopodiché aggiorna i *CSS*. All'interno del controllo di flusso viene anche richiamata la funzione `mattina()` ogni sessanta secondi grazie all'esecutore periodico: `var periodic = new PeriodicalExecuter(mattina, 60);`.

```
if ((minuto_corrente>alba_m)&&(minuto_corrente<mezzogiorno))
  {
    var min_partenza=parseFloat(minuto_corrente-alba_m);

    var partenza_r=parseFloat(10-(angolo_ok*min_partenza));

    var partenza_t=parseFloat(0+(angolo_ok*min_partenza));

    r_l=parseInt (partenza_r);

    t_b=parseInt (partenza_t);

    var ombra_riga=("rgba(0, 0, 0, 0.7) "+r_l+"px "+t_b+"px
10px");

    var ombra_riga_header=("rgba(0, 0, 0, 0.7) "+r_l+"px 0px
10px");

    for (  i = 0; i < $$(".auto_ombra").length; i++ )
      {

        $$(".auto_ombra")
[i].style.webkitBoxShadow=ombra_riga;

        $$(".auto_ombra")[i].style.mozBoxShadow=ombra_riga;

       }
  /*------------sfondo--------------*/

    var lum_partenza = parseFloat(lum_I+
(grad_lum*min_partenza));

    var lum = parseFloat(lum_partenza);

    var lum_str = lum+"";

    var lum_1 = lum_str.indexOf(".");

    var lum_2 = lum_str.substring(0, lum_1);

    var lum_colore=(colore_base+lum_2+"%)");

    for ( i = 0; i < $$(".auto_sfondo").length; i++ )
      {
```

```
        $$(".auto_sfondo")[i].style.background=lum_colore;

      }
    for ( i = 0; i < $$(".auto_color").length; i++ )
      {

        $$(".auto_color")[i].style.color=lum_colore;

        $$(".auto_color")[i].style.color=lum_colore;

       }

    $$("body")[0].style.background=lum_colore;

    var periodic = new PeriodicalExecuter(mattina, 60);

    }
```

La funzione `mattina()`, richiamata ogni minuto, effettua un ulteriore controllo per passare automaticamente alla funzione `pomeriggio()` quando l'ora ha superato il Mezzogiorno locale.
Se è ancora mattina la funzione prende i parametri modificati dai controlli riportati sopra e li aggiorna: prende le dimensioni delle ombre e gli aggiunge o toglie la quantità di pixel calcolata nella funzione `ombra_change()` e riassegna questi parametri ai fogli CSS.

```
mattina = function()
  {
    if(minuto_corrente>=mezzogiorno)
      {
        pomeriggio();
      }
    else
      {
        r_l=parseFloat (r_l-angolo_ok);

        var r_l_a=parseInt(r_l);

        t_b=parseFloat ([t_b+angolo_ok]);

        var t_b_a=parseInt(t_b);

        var ombra_riga=("rgba(0, 0, 0, 0.7) "+r_l_a+"px
"+t_b_a+"px 10px");
```

```
        var ombra_riga_header=("rgba(0, 0, 0, 0.7) "+r_l+"px
0px 10px");
        for ( i = 0; i < $$(".auto_ombra").length; i++ )
          {

            $$(".auto_ombra")
[i].style.webkitBoxShadow=ombra_riga;

            $$(".auto_ombra")
[i].style.mozBoxShadow=ombra_riga;

          }

        var lum_corrente = parseFloat(grad_lum*ind);

        lum = parseFloat(lum+grad_lum);

        var lum_str = lum+"";

        var lum_1 = lum_str.indexOf(".");

        var lum_2 = lum_str.substring(0, lum_1);

        var lum_colore=(colore_base+lum_2+"%)");

        for ( i = 0; i < $$(".auto_sfondo").length; i++ )

        {
          $$(".auto_sfondo")[i].style.background=lum_colore;

         }
        for ( i = 0; i < $$(".auto_color").length; i++ )
        {
          $$(".auto_color")[i].style.color=lum_colore;

          $$(".auto_color")[i].style.color=lum_colore;
        }

        $$("body")[0].style.background=lum_colore;

        ind++

        }
    }
```

La funzione che gestisce i parametri nel pomeriggio si comporta nello stesso modo ma la dimensione laterale dell'ombra degli elementi diventa negativa, quella dell'ombra verticale diminuisce, invece di aumentare come fa nella funzione `mattina()`, così come la luminosità.
Durante le ore della viene richiamata la funzione `notte()` che azzera le ombre degli elementi e assegna alla luminosità il suo valore minore.

```
var notte = function()
{
  var lum_colore=(colore_base+lum_I+"%)");

  var ombra_sintassi="rgba(0, 0, 0, 0.7) 0px 0px 5px";

  for ( i = 0; i < $$(".auto_ombra").length; i++ )
  {
    $$(".auto_ombra")
[i].style.webkitBoxShadow=ombra_sintassi;

    $$(".auto_ombra")[i].style.mozBoxShadow=ombra_sintassi;

  }
  for ( i = 0; i < $$(".auto_sfondo").length; i++ )
    {
      $$(".auto_sfondo")[i].style.background=lum_colore;

      $$(".auto_sfondo")[i].style.background=lum_colore;
    }
  for ( i = 0; i < $$(".auto_color").length; i++ )
    {
      $$(".auto_color")[i].style.color=lum_colore;

      $$(".auto_color")[i].style.color=lum_colore;
    }
  $$("body")[0].style.background=lum_colore;
  }

if((minuto_corrente<alba_m)||(minuto_corrente>tram_m))
  {
    notte();
    var periodic = new PeriodicalExecuter(pomeriggio, 60);

  }
```

Per compiere tutti i calcoli bisogna conoscere l'ora del sistema operativo del *browser* dell'utente visualizzata dall'oggetto JavaScript `Date()` e questa deve essere aggiornata ogni minuto.

```
set_hour = function()
  {
    var ora=new Date();

    var ora_now = ora.getHours();

    var min_now = ora.getMinutes();

    var ora_n=parseFloat(ora_now*60);

    var minuti_n=parseFloat(min_now);

    minuto_corrente=parseFloat(ora_n+minuti_n);
  }

var periodic_h = new PeriodicalExecuter(set_hour, 60);
```

4.3.Il cambiamento graduale dei colori con l'applicazione delle transizioni dei CSS 3

Solitamente, quando il valore di una proprietà del foglio di stile cambia, il risultato di questo cambiamento viene renderizzato istantaneamente e l'elemento a cui è assegnata tale proprietà viene immediatamente aggiornato.
Grazie a un nuovo modulo inserito nelle specifiche dei CSS3, chiamato *CSS Transition,* è possibile specificare la durata del cambiamento dal vecchio valore al nuovo.
Le transizioni possono agire su numerose proprietà degli elementi, sui colori, sull' opacità, sulle dimensioni. Queste trasformazioni, prima dell'inserimento del modulo potevano essere fatte solo usando *JavaScript* o una delle sue librerie.
Ammettiamo di voler cambiare il colore di sfondo del `<div id="header">` quando a questo elemento viene aggiunta la classe "colore", per fare ciò basterà aggiungere delle semplici righe di codice nel foglio di stile:

```
div#header
  {
```

```
    background:hsl(62, 89%, 15%);
    -webkit-transition-property: background;
    -webkit-transition-duration: 1s;
    -webkit-transition-timing-function: ease;
  }
```

Nella prima riga di codice viene dichiarato il colore dello sfondo, la seconda riga definisce la proprietà dell'elemento al quale viene applicata la transizione, la terza definisce la durata della transizione, la quarta definisce come vengono calcolati i valori intermedi della transizione.
Quando al `div#header` verrà assegnata la classe `.colore` il suo colore di sfondo passerà da verde a fucsia.

```
div#header.colore
  {
    background:hsl(176, 20%, 73%};
  }
```

Nel caso specifico di questo progetto le trasformazioni sono applicate al cambio del colore di sfondo degli elementi in base al passaggio delle stagioni e al cambio della luminosità del colore di sfondo in base al passaggio delle ore del giorno. Per ogni elemento che cambia le sue proprietà, sono state inserite nei fogli di stile le impostazioni per la transizione.

```
#header_tutto
  {
    background:hsl(62, 89%, 15%);
    -webkit-transition: background 1s linear;
    -moz-transition:background 1s linear;
  }

#slidedown_demo
  {
    background:hsla(223, 9%, 43%, 0.9);
    -webkit-transition:background 1s linear;
    -moz-transition:background 1s linear;
  }

#contenuto_ok
  {
    background:hsl(62, 89%, 15%);
    -webkit-transition: background 1s linear;
    -moz-transition: background 1s linear;
  }
```

```
a
  {
    color:hsl(62, 89%, 15%);
    -webkit-transition: color 1s linear;
    -moz-transition: color 1s linear;
  }

h1
  {
    color:hsl(62, 89%, 15%);
    -webkit-transition: color 1s linear;
    -moz-transition: color 1s linear;
  }

li.menu_v
  {
    color:hsl(62, 89%, 15%);
    -webkit-transition: color 1s linear;
    -moz-transition: color 1s linear;
  }
```

Nelle funzioni che gestiscono il cambiamento delle stagioni e della luminosità dei colori viene impostato il valore finale che deve raggiungere la transizione. La classe "auto_color" raggruppa tutti gli elementi che devono cambiare la luminosità del *background*. Il ciclo for permette di cambiare questo valore a ciascun elemento.

```
for ( i = 0; i < $$(".auto_color").length; i++ )
    {
        $$(".auto_color")[i].style.color=lum_colore;

    }
```

5. Il pannello di controllo

Il cambiamento della grafica dipende dai fattori reali: il tempo e lo spazio. Per dimostrare le numerose funzionalità del sito è stato necessario inserire un pannello di controllo che permetta all'utente di simulare di trovarsi in una certa posizione geografica e di scegliere una data e un'ora, altrimenti sarebbe difficile poter mostrare, per fini dimostrativi, lo spostamento dell'ombra, il cambiamento della luminosità del colore durante il giorno, il cambiamento delle impostazioni grafiche

che si modificano con il passare delle stagioni. Tutti questi mutamenti avrebbero un'esecuzione molto lenta.

5.Immagine del pannello di controllo

5.1.Come funziona il pannello di controllo

Il pannello di controllo è un modulo diviso in tre parti:

- spazio
- tempo
- risultati.

Le prime due, indipendenti l'una dall'altra, permettono all'utente di scegliere delle impostazioni, mentre la terza mostra i risultati del calcolo dell'ora dell'alba, del tramonto e la quantità di ore di luce nella giornata.

Le funzioni che agiscono sulle impostazioni grafiche sono le stesse che sono state spiegate nei paragrafi precedenti ma, invece di prendere i parametri da valutare in maniera automatica dal servizio di localizzazione, dall'orario del sistema operativo dell'utente, questi sono scelti direttamente dall'utente stesso.

Nella prima parte, quella che riguarda lo spazio, l'utente può scegliere una capitale presente all'interno del menu di selezione.

Ogni opzione del menu ha un attributo `value` che contiene le sue coordinate geografiche che vengono visualizzate nei relativi campi di testo e passate come parametri alle funzioni invocate:

- `set_hour_change();` che modifica l'ora visualizzata in base al fuso orario della città selezionata,

```
set_hour_change= function()
    {
        var d = new Date();
        var localTime = d.getTime();
        var timzn = d.getTimezoneOffset();
        var localOffset = d.getTimezoneOffset()*60000;
        var utc = localTime + localOffset;
        document.all.tz_text.value=document.all.tz.value;
        var gmt = document.all.tz.value;
        var gmt_1 = gmt.indexOf("T");
        var gmt_6 = gmt.indexOf(" ");
        var gmt_2 = gmt.substr(gmt_1+1,3);
        var offset = parseFloat(gmt_2);
        var cityTime = utc+(3600000*offset);
        var nd = new Date(cityTime);
        var nd_ora = nd.getHours();
        var nd_min = nd.getMinutes();
        var gg = nd.getDate();
        document.all.ora.selectedIndex = nd_ora;
        document.all.minuti.selectedIndex = nd_min;
    }
```

- `check_season();` (paragrafo 3.2.3) che controlla se la latitudine del luogo selezionato è positiva o negativa per impostare i parametri della stagione corrispondente alla data, e richiama a sua volta la funzione `ombra_change();` (paragrafo 3.2.4) che calcola la dimensione delle ombre e la luminosità del colore di sfondo in base all'ora del luogo scelto.
- `tempo_weather();` (paragrafo 1.3.1.2) che invia la richiesta delle informazioni meteorologiche a *Yahoo! Weather RSS Feed.*

Nella seconda parte ci sono le impostazioni che riguardano il tempo cronologico:

- un campo di testo nel quale viene visualizzato il fuso orario della città scelta nel modulo che riguarda lo spazio, il valore di *default* è GMT+00 (GMT)
- un menu di opzioni per il mese, uno per il giorno, per default vengono selezionati automaticamente il mese e la data correnti
- un menu di opzioni per l'ora e uno per i minuti

Modificando queste scelte le impostazioni grafiche del sito vengono cambiate in tempo reale. Ad esempio il visitatore, che si trova in Italia, può scegliere di visualizzare le impostazioni grafiche simulando di trovarsi a Tokio: a questo punto le funzioni calcolano la differenza di fuso orario fra l'Italia e il Giappone e impostano l'ora giusta in base alla quale verranno modificate le ombre e la luminosità dei colori relative a quell'orario.
Il pannello di controllo si trova all'interno di un *div* nascosto nell'*homepage*, quando il visitatore preme il bottone "*impostazioni"* viene invocata la funzione `slide()` che attiva l'effetto `Effect.SlideDown('nascosto', {duration: 0.5});` il quale abbassa gli altri *div* e mostra quello che contiene il pannello di controllo.
All'evento che mostra il *div* nascosto è legata la rimozione delle classi *.auto_ombra* e *.auto_sfondo,* assegnate a tutti gli elementi la cui grafica viene modificata dalle funzioni che, richiamate periodicamente, gestiscono il tempo cronologico in maniera automatica e vengono aggiunte le classi *.demo_ombra* e *.demo_sfondo* in modo da legare gli elementi alle modifiche grafiche attivate dalle funzioni invocate dal pannello di controllo.
Viene poi inizializzata la variabile `on=1;` in questo modo quando l'utente preme di nuovo il bottone *impostazioni* il *div* torna ad essere nascosto, vengono rimosse le classi *.demo_ombra* e *.demo_sfondo* e ripristinate le classi *auto_ombra* e *.auto_sfondo*, così che possano tornare in funzione le funzioni che modificano la grafica utilizzando i parametri automatici.

```
var on=0;
function slide()
  {
    if(on==0)
      {
        new
Ajax.Updater('slidedown_demo','dimostrazione_tempo_ok.php',{
        onComplete: function()
          {

            Effect.SlideDown('nascosto', {duration: 0.5});
            $('header').removeClassName('auto_ombra');
            $
('slidedown_demo').removeClassName('auto_ombra');
            $('contenuto').removeClassName('auto_ombra');
            $('header_tutto').removeClassName('auto_sfondo');
            $
('slidedown_demo').removeClassName('auto_sfondo');
            $('contenuto_ok').removeClassName('auto_sfondo');
```

```
            for ( i = 0; i < $$('a').length; i++ )
                {
                  $$('a')[i].removeClassName('auto_color');
                }

            for ( i = 0; i < $$('li').length; i++ )
                {
                  $$('li')[i].removeClassName('auto_color');
                }
            for ( i = 0; i < $$('h1').length; i++ )
                {
                  $$('h1')[i].removeClassName('auto_color');
                }

            $('header').addClassName('demo_ombra');
            $('slidedown_demo').addClassName('demo_ombra');
            $('contenuto').addClassName('demo_ombra');
            $('header_tutto').addClassName('demo_sfondo');
            $('slidedown_demo').addClassName('demo_sfondo');
            $('contenuto_ok').addClassName('demo_sfondo');
            for ( i = 0; i < $$('a').length; i++ )
                {
                  $$('a')[i].addClassName('demo_color');
                }

            for ( i = 0; i < $$('li').length; i++ )
                {
                  $$('li')[i].addClassName('demo_color');
                }
            for ( i = 0; i < $$('h1').length; i++ )
                {
                  $$('h1')[i].addClassName('demo_color');
                }
          },
        evalScripts:true
        } );

        on=1;
      }
    else
      {
        Effect.SlideUp('nascosto', { duration: 0.5});
        $('header').removeClassName('demo_ombra');
        $('slidedown_demo').removeClassName('demo_ombra');
        $('contenuto').removeClassName('demo_ombra');
        $('header_tutto').removeClassName('demo_sfondo');
```

```
        $('slidedown_demo').removeClassName('demo_sfondo');
        $('contenuto_ok').removeClassName('demo_sfondo');
        for ( i = 0; i < $$('a').length; i++ )
          {
            $$('a')[i].removeClassName('demo_color');
          }
        for ( i = 0; i < $$('li').length; i++ )
          {
            $$('li')[i].removeClassName('demo_color');
          }
        for ( i = 0; i < $$('h1').length; i++ )
          {
            $$('h1')[i].removeClassName('demo_color');
          }

          $('header').addClassName('auto_ombra');
          $('slidedown_demo').addClassName('auto_ombra');
          $('contenuto').addClassName('auto_ombra');
          $('header_tutto').addClassName('auto_sfondo');
          $('slidedown_demo').addClassName('auto_sfondo');
          $('contenuto_ok').addClassName('auto_sfondo');
          for ( i = 0; i < $$('a').length; i++ )
            {
              $$('a')[i].addClassName('auto_color');
            }
          for ( i = 0; i < $$('li').length; i++ )
            {
              $$('li')[i].addClassName('auto_color');
            }
          for ( i = 0; i < $$('h1').length; i++ )
                {
                  $$('h1')[i].addClassName('auto_color');
                }
        check_season();
        on=0;
        }
}
```

Conclusioni

Con lo sviluppo di questo progetto si è voluto mostrare una possibile evoluzione della grafica dei siti in conseguenza all'evoluzione del concetto di web.
Le ombre degli elementi simulano il movimento delle ombre reali prodotte dal sole in base all'ora dal luogo in cui si trova l'utente.
L'immagine di sfondo degli elementi riproducono il tempo meteorologico.
Il colore di sfondo suggerisce la stagione in corso.
La luminosità dei colori simula la quantità di luce nell'ora in cui il sito viene visitato.
Ma questo è solo un'ipotesi circa un possibile nuovo modo di progettare la grafica dei siti. Tutti gli elementi grafici, oltre ai contenuti, si modificheranno in base alle informazioni invisibili che vengono fornite dai *browser* sul mondo reale in cui si trova l'utente quando si connette, in base all'interazione dell'utente col sito stesso, ad esempio, *link* che sbiadiscono perché sono molto visitati e quindi si usurano, o al contrario spariscono pian piano se non vengono attivati. In un futuro non molto lontano, grazie anche al diffondersi delle nuove tecnologie che hanno fra le loro componenti bussole, navigatori satellitari, la grafica dei siti potrà modificarsi, per esempio, non solo in base alla posizione geografica del visitatore ma anche in base al suo orientamento rispetto ai punti cardinali.
La distanza che separa il mondo reale da quello virtuale sta diventando sempre minore.
"Il web incontra il mondo - questo è il Web Squared" (Special Report. Web Squared: Web 2.0 five years on - 2.0 Summit).

Bibliografia e sitografia

Bibliografia

Kurose James F. e Ross Keith W. *Internet e reti di calcolatori* Mc Grow-Hill

Munrphy Christopher e Persson Nicklas *HTML and CSS Web Standards Solutions: A Web Standardistas' Approach* Apress 2008

Sitografia

IV Summit Italiano Architettura dell'informazione
http://www.iasummit.it/2010/

CSS3.info
http://www.css3.info/

Full Web Buildings Tutorial
http://www.w3schools.com/

PHP user manual
http://us2.php.net/

Prototype JavaScript framework
http://www.prototypejs.org/

Web 2.0 Summit
http://www.web2summit.com/web2009/