Ingrid Arcas Sanz
Tutora: Victoria Sacco

Memòria tècnica
14 · 09 · 2012 · 2ª Convocatòria
AV / G · ESDi

Índex

[Introducció]

Partint dels coneixements adquirits en el marc teòric, aquesta memòria tècnica pretén mostrar el procés de materialització dels conceptes més rellevants en l'etapa de desenvolupament del projecte.

D'una banda tractaré de representar a la *multitud intel·ligent* que ha fet possible aquesta revolució com un conjunt de veus que ha unit les seves forces per un objectiu comú. Per altra banda també donaré veu a cada una de les persones que formen part d'aquest col·lectiu. Com aconseguir un relat a partir del *tot* i el *concret* quan es manifesten a la vegada? El repte del projecte és oferir a l'usuari una narració a partir de la visió d'aquestes dues parts convivint en un mateix espai–temps.

El fet d'utilitzar els textos de Twitter implicarà que els missatges que es mostrin en l'aplicació siguin breus i molt directes. A més a més, els textos que es mostraran seran els originals, sense cap interpretació. L'usuari veurà de primera mà els missatges de milers de persones que desitgen ser escoltades. Això farà que cada *tweet* adquireixi més força i ocasioni un impacte més gran en l'usuari que l'estarà llegint. Aquesta, per mi, és la principal raó per la qual la xarxa de microblogging Twitter té la importància que ha adquirit. Però a diferència de Twitter, jo no vull que l'usuari esdevingui espectador passiu. Vull que l'usuari tingui la llibertat d'escollir quins missatges llegir entre tots, a quins usuaris donar veu d'entre la *multitud*.

Per tant, l'aplicació dependrà de la interacció de l'usuari per descobrir aquests missatges, endinsant-lo dins del context on es van crear aquests. D'aquesta manera s'implicarà a l'usuari en la lectura del textos fent que tot plegat desemboqui en un experiència que el faci reflexionar sobre el moviment i la naturalesa del mateix a la xarxa Twitter.

[Marc Pràctic]

01_Recopilació *tweets*

Per desenvolupar aquest projecte el primer pas ha estat obtenir la matèria prima amb la que treballaré: tots aquells *tweets* que facin referència al Moviment #15M. Per dur a terme aquesta tasca abans cal determinar els límits d'aquesta cerca. Si bé aquest moviment no està aïllat a una duració concreta –avui dia segueix vigent–, el període que m'interessa abastar és el corresponent al sorgiment de la revolta i les acampades. El motiu és que en aquest període és quan va produir la primera manifestació de les *multituds intel·ligents* d'aquest moviment.

La data d'inici es pot establir el dia de la manifestació, el 15 de maig. No obstant, m'he decantat per començar la cerca el 13 de maig per tal de mostrar l'activitat de missatges que hi havia els dies previs a la manifestació i examinar com es començava a coure aquest esdeveniment.

Pel que fa a la data final vull incloure el període de les acampades, però no totes van acabar el mateix dia. Tenint en compte que la precursora de les acampades va ser la que es va instal·lar a la Porta del Sol de Madrid, l'*AcampadaSol*, he optat per escollir el dia que aquests van decidir marxar de la plaça, és a dir, el 12 de juny. No és la data on totes les acampades ja havien marxat, però sí que és la data d'inici d'aquesta decisió.

Per tant, el meu projecte mostrarà tots aquells missatges creats des del 13 de maig al 12 de juny.

L'altre criteri per a la cerca han estat els *hashtags* (temes clau). Com he descrit a la memòria teòrica, els *hashtags* manifesten –sota el meu parer– la intenció de parlar sobre un tema en concret. D'aquesta manera filtrant per *hashtag* es poden aconseguir el *tweets* que parlaven d'aquest moviment. No obstant, aquest moviment compta amb un mot clau genèric, #15m, però segons les accions o temes que anaven sorgint en creaven d'altres. Entre tots aquestes temes, l'aspecte que m'interessa ressaltar és el sorgiment del moviment i les acampades. Per al primer concepte, el *hashtag #tomalacalle* és el que es va fer servir per mobilitzar Twitter i convocar la manifestació. Per a les acampades, no va existir-ne un de genèric sinó que se'n van crear tants

com acampades van sorgir (al voltat d'uns 75[1]). Per tant, he triat el de *#acampadabcn*, corresponent a l'acampada de la Plaça de Catalunya de Barcelona per proximitat, ja que vaig participar activament en algunes accions que van fer; i perquè junt amb la de Madrid van ser les més importants i assenyalades.

D'aquesta manera el criteri de cerca a nivell de contingut, dins del període indicat anteriorment, l'he limitat de manera que sempre estigui present el *hashtag #15m* seguit de *#acampadabcn* o *#tomalacalle*.

Un cop triat els criteris de cerca, em trobo amb el primer problema: consultant la pàgina de Twitter per aconseguir els *tweets* que m'interessa guardar, m'adono que el seu motor de cerca està limitat. Twitter només mostra els últims 3.000 *tweets* d'un mateix usuari o en el cas dels *hashtags* els de l'últim mes. A més a més, el seu formulari de cerca no permet especificar més d'un criteri. Això és un problema greu, doncs la base del projecte es basa en la representació d'uns *tweets* que *a priori* no puc aconseguir.

Consulto també la API (funcions i mètodes que permeten consultar i obtenir dades de serveis online) que Twitter deixa a disposició de programadors i observo que aquest recurs pot mostrar qualsevol *tweet* donat un identificador, però no permet fer una cerca. Això em deixa en el mateix punt però amb una possible solució: si d'alguna manera puc obtenir aquest identificador, Twitter podria proporcionar-me les dades que necessito i, per tant, avançar en el projecte.

Després d'informar-me i buscar per Internet, trobo una possible solució: la pàgina web de *Topsy*[2]. Aquesta empresa es dedica a l'emmagatzematge de tots els *tweets* que es van publicant. El portal ofereix un servei per buscar *tweets* fins a dos anys d'antiguitat donat diversos criteris: data inici amb hora, data final amb hora, usuari, *hashtag* o que contingui (o no) un text especificat. És un servei gratuït que mostra 1.000 *tweets* com màxim en cada cerca. D'aquesta manera, fent cerques per hora donat un dia es poden obtenir 24.000 *tweets* per dia, més que suficient per al projecte. És així com a partir de realitzar les consultes adequades a *Topsy* podré resoldre el problema d'accedir als textos que es van escriure un any enrere.

[1] *Mi web*. (2011). Accès el 27 de juny de 2012, des de Grupo 15M - No nos vamos: http://www.mi-web.org/miembros/5-eric/textos/61260-listado-completo-de-acampadas-15-m-en-espana

[2] Topsy - Instant social insight. (2003). Accès el 13 de juny de 2012, des de Topsy: http://topsy.com

De nou em torno a trobar amb una dificultat durant aquest procés. Tot i haver trobat la pàgina web de *Topsy*, el problema radica en com obtenir tots aquests *tweets* sense que hagi de ser una tasca manual. Nathan Yau[3], en el seu llibre *Visualize This* explica com obtenir les dades que mostra una pàgina web com a resultat d'una cerca a partir d'un *script* fet amb el llenguatge de programació *Python*.

Per poder fer servir aquest *script* primer cal observar si la pàgina web càrrega les dades a partir de passar les variables de cerca per la URL. En el cas de *Topsy* es compleix, la seva URL conté variables de paginació, data inici i data final, paraules clau i l'ordre.

Per poder crear el *script* per a aquesta web, he pogut comptar amb la col·laboració de l'Albert Meroño, programador. Ell ha creat un petit programa anomenat *Topy-tweet-retriever* –que a més ha publicat a la pàgina web Github[4] per a la seva lliure utilització per part d'altres usuaris – on indicant a través d'un *terminal* les paraules clau i les dates d'inici i final, fa una cerca a la web de *Topsy* per hora i per dia guardant les *id* (número identificador) dels *tweets*. Posteriorment a través de la llibreria *python-twitter* es connecta a la API de Twitter[5] i per cada *id* obtingut anteriorment pot adquirir tota la informació que es requereixi –i que ofereixi aquest servei, per suposat– tant del *tweet* com de l'usuari que l'ha escrit. Tota aquesta informació es guarda en la base de dades que s'indiqui al *script*.

D'aquesta manera, executant adientment aquest *script* com a usuària he pogut construir una base de dades utilitzant el sistema de gestor de dades *MySQL* (gratuït, lliure, popular i molt eficient) amb 3.192 *tweets* guardant la següent informació:

Una taula amb informació del *Tweet*
· Data creació

[3] Yan, N. (2011). Visualize This: The FlowingData Guide to Design, Visualization, and Statistics. Indianapolis: Editorial Wiley.

[4] *Topsy Tweet Retriever*. (2012). Recuperado el 09 de setembre de 2012, de Github: https://github.com/albertmeronyo/topsy-tweet-retriever

[5] Twitter API. (2010). Accès el 02 de juliol de 2012, des de Twitter API: http://www.dev.twitter.com

· Identificador
· Text
· Pàgina des d'on s'ha publicat (Twitter, twitter mòbil o altres serveis)
· Identificador de l'usuari
· Número de *retweets* que ha tingut aquell missatge

Una taula amb informació dels usuaris dels *tweets* guardats
· Data creació del compte a Twitter
· Text que l'usuari pot introduir com a "descripció"
· Número de *tweets* que ha marcat com a "favorit"
· Número de seguidors
· Número d'usuaris a qui segueix
· Identificador
· Llengua que ha indicat per a la interfície de Twitter
· Si te activat la geolocalització
· La localització que ha indicat l'usuari
· Nom
· Aspectes de disseny con color de fons o color per als enllaços...etc
· URL de la imatge que mostra

El caire d'aquest *script* és genèric per tal d'acomodar-se a les diferents necessitats que pugin tenir els usuaris. És per això que molts dels camps especificats anteriorment no es requeriran per aquest projecte.

02_Tria de l'entorn de programació

Un cop obtinguts els missatges de Twitter, he investigat diferents entorns de programació per tal de decantar-me per un i desenvolupar el projecte.

En primer lloc reuneixo els llenguatges de programació que he après durant la carrera i que domino: Processing, ActionScript (Flash) i HTML, Javascript i CSS.

Per l'altra, llenguatges com Python, R o la llibreria en C++ OpenFramworks, que no he utilitzat mai o que tinc poca noció.

Tots aquests llenguatges tenen pros i contres. El criteri per seleccionar l'entorn s'ha regit segons la versatilitat que ofereix a nivell visual i la seva incorporació en un entorn web online. Sense dubte, els llenguatges que ofereixen més eines a nivell creatiu i flexibilitat a l'hora de ser publicats son ActionScript, Processing i la llibreria OpenFrameworks. Com amb aquest últim no tinc molta experiència, he decidit descartar-lo. ActionScript té molta força a nivell d'interfície per a creacions visual i interactives, però és un llenguatge ineficient quan ha de tractar amb moltes dades i avui dia està perdent molta força en l'àmbit web. En aquest sentit no pot ser una opció ja que per a la visualització requeriré treballar amb grans quantitats de dades.

Per tant, l'entorn que he decidit triar és Processing. És un llenguatge sobre el que tinc coneixements de nivell mitjà-alt i em trobo còmode desenvolupant en aquest entorn. Es tracta d'un programa lliure, de codi obert, molt versàtil i encarat a desenvolupar sobretot projectes visuals i interactius.

Compta amb tot un seguit de recursos i llibreries fetes per complementar les eines que ja ofereix. Amb poques línies de codi es poden aconseguir aplicacions molt interessants amb un alt component d'interacció.

L'únic punt en contra és que per poder obrir l'aplicació resultant, l'usuari ha de tenir instal·lat al seu sistema Java. Fa uns anys això podia ser un problema, no obstant avui dia pràcticament tots els ordinadors compten amb aquest complement de manera que no hauria de suposar un gran inconvenient. A més a més, com l'aplicació estarà incorporada dins d'una pàgina web, els

navegadors d'avui dia detecten si necessiten aquest *plug-in* i poden assistir a l'usuari en el procés d'instal·lació.

És totalment compatible amb els sistemes operatius de Mac, Windows i Linux. Al tractar-se de programari amb codi obert constantment evoluciona afegint noves funcionalitats d'acord amb les necessitats que els seus usuaris troben a faltar per als seus projectes.

També compta amb una gran comunitat d'usuaris, experts i programadors que interaccionen a la web de Processing i sempre pots rebre suport i ajuda a l'hora de desenvolupar el projecte.

Per últim, el seu sistema d'exportació és molt complet. Al ser un programa basat en el llenguatge de programació Java, tant pot ser una aplicació amb un executable com formar part d'una plataforma online. Qualsevol d'aquestes opcions poden ser escollides i el propi programa construeix l'estructura de cara a poder ser utilitzat un cop exportat sense haver de realitzar més accions.

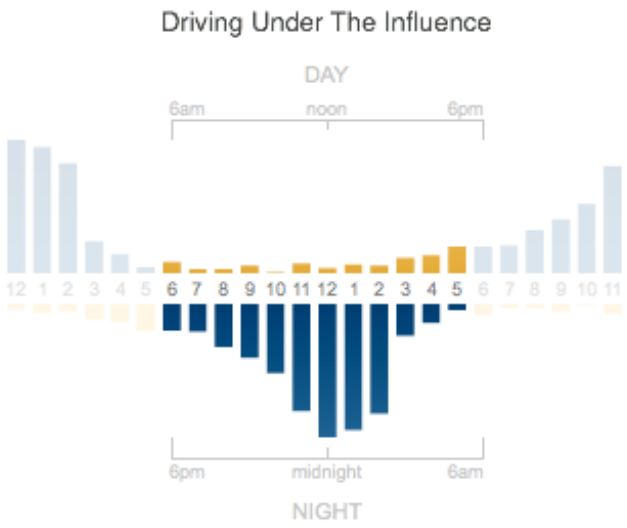
03_Visualització

La obtenció de més de 3.000 *tweets* sobre aquest moviment obre la qüestió de com representar-los per tal de complir amb els objectius marcats. Sense dubte la visualització de dades ofereix diverses solucions visuals que ajudaran a l'usuari a explorar i entendre aquesta gran quantitat de missatges.

Per escollir la manera en què es visualitzaran aquests *tweets* he estudiat diferents propostes segons els tipus d'eixos que utilitzen.

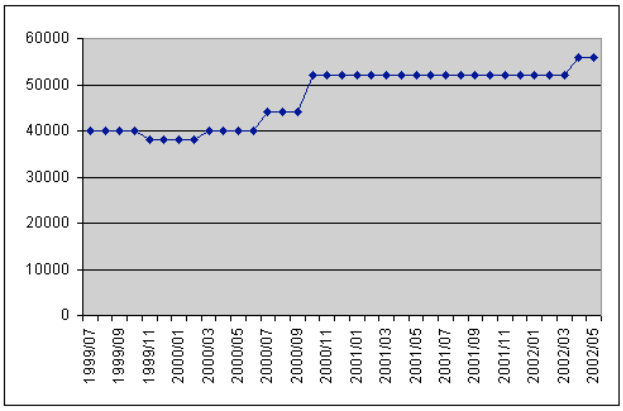
En primer lloc, dades situades en l'eix del temps. Aquests tipus de visualitzacions es caracteritzen per utilitzar com a mínim dos eixos: un de vertical i un d'horitzontal. Operen sempre amb una variable de temps (hores, dies, anys, etc.) i una segona variable que pot ser quasi de qualsevol tipus (temps, quantitat, percentatge, etc.). La interacció d'aquests dos eixos situen les dades en un temps concret donant a més a més informació sobre com evoluciona.

Aquests gràfic pot ser representat en barres, punts o continuïtat. Les barres ajuden a indicar normalment creixement o decreixement partint d'una base zero (*Imatge 1*).



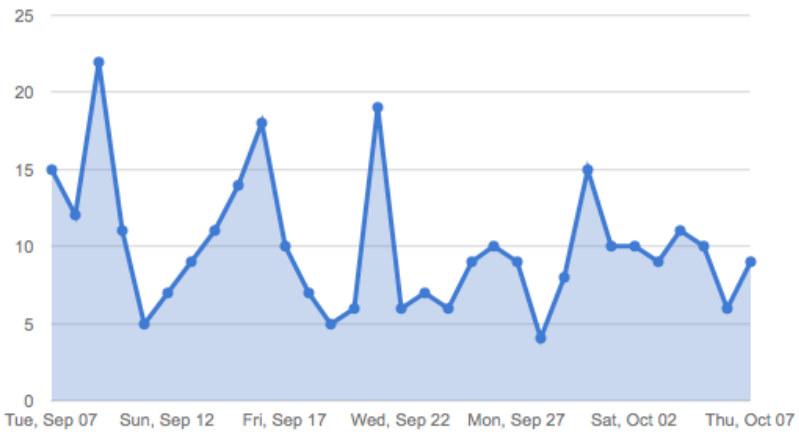
Imatge 1

Els punts (*Imatge 2*), a diferència de les barres, situen la data en un lloc concret del gràfic. Moltes vegades aquests punts s'uneixen mitjançant una línia per indicar una línia de progrés.



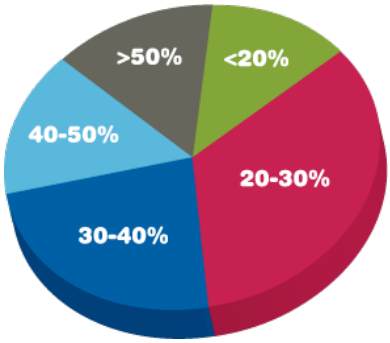
Imatge 2

Per últim, també es pot representar la continuïtat de les dades a partir de la connexió dels punts mitjançant una línia (*Imatge 3*). De la mateixa manera que amb les barres, les dades també han de partir d'una base zero per no distorsionar el resultat.



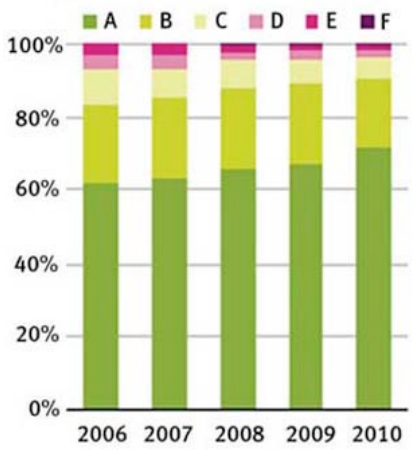
Imatge 3

En segon lloc, dades són representades segons la seva proporció. Les proporcions vénen donades per un mínim de tres valors: màxim, mínim i la distribució general. La seva representació pot ser de moltes maneres. La mes popular és a través d'un cercle dividit en tantes parts com dades hi hagin, i cada part tindrà una mida en proporció al màxim i mínim indicat.

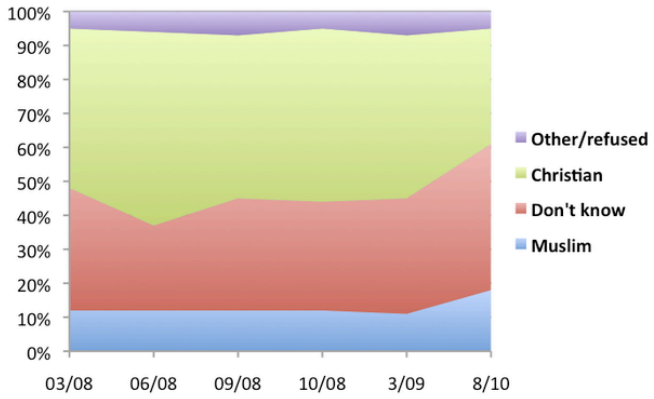


Imatge 4

Cal destacar que també es pot afegir la proporció a la representació segons l'eix del temps explicat anteriorment, concretament en els gràfics de barres (*Imatge 5*) i de continuïtat (*Imatge 6*). D'aquesta manera les barres i els gràfics de continuïtat també es poden dividir internament en proporcions.

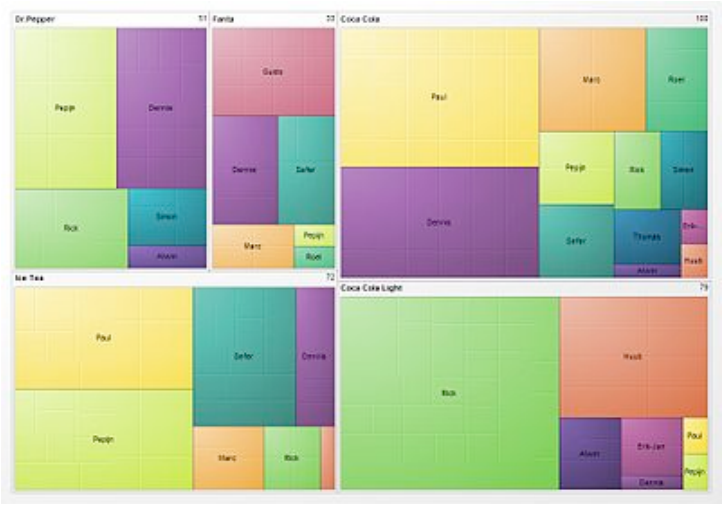


Imatge 5



Imatge 6

Dins de la representació mitjançant proporcions existeix el *Treemap* (*Imatge 7*). Un treemap, traduït com un mapa en arbre, és un tipus de visualització que consisteix en dividir una superfície quadrada o rectangular en tantes parts com dades tinguem. La mida de cada rectangle que en resulta representa un valor. El més interessant d'aquesta visualització és que permet ordenar els elements en capes gràcies al sistema de jerarquia que permet implementar. D'aquesta manera, molts rectangles poden funcionar com a categoria enlloc de representar la dada en sí i, per tant, donar un punt de vista més global o més concret. Aquest tipus de visualització resulta molt útil per observar, per exemple, l'espai en el disc que ocupen les carpetes de l'escriptori del nostre ordinador.



Imatge 7

Fins aquí, he desenvolupat les visualitzacions que donades les característiques del meu projecte resulten més adequades. Tot i així, també he observat altres visualitzacions que

tracten aspectes de representació de dades en mapes i representació de les relacions que hi ha entre dades, però en aquest cas no em resulten útils pel tipus de dades que tractaré. Les dades que tinc són dades textuals en el temps, i per tant, he descartat totes les visualitzacions basades en mapes i relacions. El contingut que m'interessa ressaltar de les *multituds* està generat en el temps i no en l'espai. Precisament ja hi ha projectes que exploren les dades en l'espai.

Partint doncs d'aquesta base, la meva proposta visual pretén situar cada un dels missatges segons el moment en què van ser escrits i enviats a Twitter. Aquesta representació es farà en dos nivells: genèric i concret. En el nivell genèric es mostraran tots els *tweets* recopilats durant la franja de temps que vull estudiar ubicats segons el dia i l'hora de l'enviament. Amb aquesta disposició vull que l'usuari pugui observar el flux d'activitat que hi va haver durant tot aquest període, deixant entreveure els moments clau d'aquest moviment donant un punt de vista generalista, de manera que l'usuari tingui una percepció global de la distribució dels missatges d'un sol cop d'ull.

Al tractar-se de text i tenir un nombre alt de *tweets* situar-lo directament en aquests eixos podria confondre a l'usuari i dificultar la lectura d'aquests missatges. És per això que he decidit "amagar" aquests missatges dins d'una imatge que representi d'alguna manera el tipus de text que conté. Aquestes imatges són els logotips originals de l'acampada de Barcelona i del moviment, que corresponen als dos *hashtags* que he seleccionat per buscar els *tweets*: *#acampadabcn* (*Imatge 8*) i *#tomalacalle* (*Imatge 9*). Cada un d'ells té un logotip associat i em sembla interessant que l'usuari pugui d'entrada diferenciar entre tots els missatges la intenció del *tweet*.

Imatge 8

Imatge 9

Aquestes dues imatges, per tant, apareixeran segons el *hashtag* que contingui al text, les vegades que siguin necessàries i es situaran segons el dia i l'hora en què el missatge que representen hagi estat escrit. Segons el número de *retweets* que tingui aquell *tweet*, la imatge es mostrarà més gran o més petita denotant així de forma gràfica la importància que té el seu *tweet* en comparació amb la resta i, sempre tindrà uns punts de transparència. Aquest últim aspecte ajudarà a ressaltar totes les "veus" de manera que no es tapin entre elles sinó que es sumin. Amb aquesta acció vull emfatitzar precisament el poder d'aquest moviment: moltes veus individuals que acaben formant-ne una de més gran i que les engloba a totes.

Quan l'usuari col·loca el cursor a sobre de qualsevol de les dues imatges, aquesta es transforma en la imatge de perfil de Twitter de l'autor que ha escrit el *tweet* i el text apareix en una de les cantonades de la imatge dins d'una caixa representant un globus de text. D'aquesta manera l'acció de l'usuari és el que dóna veu a l'autor del missatge que es revel·la a l'espectador només a l'hora d'emetre el seu missatge particular i només quan aquest el demana, establint un lligam entre l'usuari i aquella veu del moviment.

De la mateixa manera, l'usuari pot escollir entre veure tots els missatges a la vegada ,o bé per dia. La visualització comptarà amb un panell a la part superior on podrà fer clic sobre el dia que li interessi veure en detall o veure'ls tots. En el cas que l'usuari faci clic sobre un dia, els eixos de representació ja no seran de dies i hores, sinó d'hores i minuts. És així com es podrà observar les hores on hi ha més tendència d'enviar missatges ja sigui per un esdeveniment puntual d'aquest moviment o altres raons.

04_Sonorització

El so és un tipus d'estímul poc habitual en les plataformes web. Tot i això, té l'avantatge de cridar més l'atenció precisament per la novetat que li resulta a l'usuari rebre aquest tipus d'estímul. L'usuari, quan consulta pàgines web, parteix de la base que tot allò que li aparegui serà visual, tant si està estàtic com si té animació incorporada. És per això que per al meu projecte vull incorporar aquest aspecte. M'interessa introduir a l'usuari més enllà del textos del Moviment #15m, vull que es vegi envoltat del que realment ha suposat aquesta revolució: la força de les multituds.

En una primera instància he pensat en crear una melodia sintètica a partir de factors com el número de *retweets* o el *hashtag* que tinguessin. Aquesta opció, no obstant, em sembla artificial i amb un missatge molt pobre en comparació amb la força que tindrà la visualització.

També he pensat en sonoritzar l'aplicació a partir que l'usuari sentís allò que hi ha escrit al missatge. Però també ho he descartat doncs per una banda és redundant tenir un mateix missatge per dos canals, –textual-visual i sonor– i, per altra, la veu que llegís aquest text hagués estat una veu artificial donant una sensació artificial i freda. Jo vull tot el contrari, vull veus de veritat, que envoltin a l'usuari, i així ressaltar també el caràcter humà del moviment .

Finalment, m'he decantat per utilitzar els paisatges sonors que es van gravar gràcies a la iniciativa del Laboratori d'Art Sonor i l'assignatura Laboratori del Caos del Departament d'Escultura de la Facultat de Belles Arts de la Universitat de Barcelona sobre el Moviment #15m. Aquests paisatges van ser enregistrats durant el període de l'acampada de Barcelona i contenen accions com les cassolades, les assemblees, manifestacions o fins i tot el moment del desallotjament dels *indignats* de Plaça de Catalunya.

Per tant, la sonorització de l'aplicació constarà de sis fragments d'una hora de durada cada un aproximadament que sonaran d'entrada a la vegada i tindran per objectiu introduir a l'usuari dins de l'ambient que hi va haver durant aquell període. Seguidament, cada cinc segons sonarà un nou àudio, d'una durada molt curta –entre mig minut i tres minuts aproximadament. Aquest àudios a diferència dels de llarga durada, seran d'accions com les cassolades i les manifestacions per crear petits punts d'atenció a l'usuari i simular el context en que es van

crear els *tweets*. És així com l'usuari podrà tenir la sensació de trobar-se allà immers en l'ambient de l'acampada de Barcelona però també dins de les accions i activitats que es van realitzar quasi cada dia.

05_Interacció

La interacció és un dels punts que vull que estigui present d'alguna manera. El meu projecte tracta d'oferir una plataforma per a la lectura dels missatges de Twitter enviats durant el període esmentat. L'acció de llegir un text és passiva, de manera que l'usuari podria esdevenir espectador o lector.

Per tal que això no succeeixi, vull convidar a l'usuari que sigui la seva acció la que mostri el text. Amb aquest gest vull aconseguir que l'usuari es submergeixi dins l'aplicació i s'hi senti participant. Els testimonis de les milers de persones que han enviat *tweets* han quedat oblidades un any després degut al fet que Twitter no proporciona cap eina per recuperar missatges antics. L'usuari té al seu abast l'opció de ressuscitar aquests textos gràcies a la seva acció. D'aquesta manera també es ressuscita el fet que el moviment convidava a la participació activa els ciutadans; així, per a mi és molt important que el component interactiu també inciti a l'espectador a adherir-se i participar i no limitar-se a observar.

Per una banda, he decidit que aquests missatges es mostrin amb una acció molt bàsica però a la vegada funcional i còmode per l'usuari: el *Roll Over*. Es tracta doncs, d'una mínima actuació on el fet que l'usuari col·loqui el cursor a sobre d'una de les imatges fa que aquesta es transformi en la imatge de l'autor del *tweet* i mostri el text. Quan l'individu torna a estar fora d'aquesta superfície, el text desapareix i la imatge es torna a transformar en el logotip que hi havia inicialment.

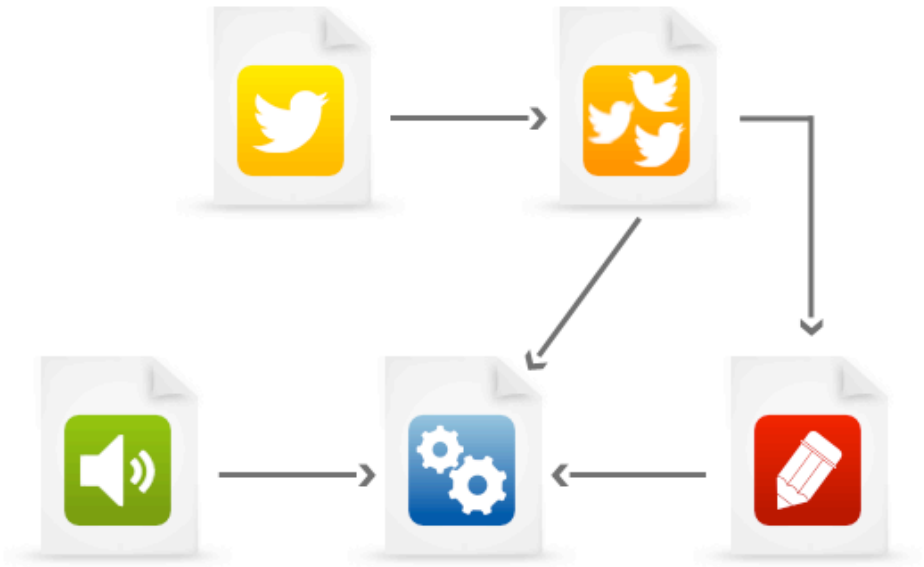
He escollit aquesta acció i no d'altres com el *clic* per tal de no distreure a l'usuari d'allò vertaderament important: l'aplicació. L'usuari no ha de parar especial atenció en moure el cursor per la superfície de l'aplicació. Si enlloc d'això hagués de fer *clic* cada vegada que volgués veure un text, aquesta acció es tornaria repetitiva i, fins i tot, pesada per a l'usuari.

Per altra banda, l'usuari també podrà decidir si vol veure representats tots els missatges enviats a la vegada, o els d'un dia en concret. Es tracta d'una altra acció mínima però, sota el meu parer, suficient per implicar a l'usuari en el projecte. L'aplicació conté estímuls sonors i força elements visuals que de manera implícita fan que l'usuari ja estigui interactuant amb ells encara que no estigui realitzant cap acció física. L'afegit del *Roll Over* i la opció de filtrar per dia complementen la seva experiència vers el projecte.

06_Arquitectura del programa

La programació de l'aplicació ha estat un tasca laboriosa però alhora molt satisfactòria pel resultat obtingut. Durant el procés de desenvolupament m'he vist en moltes ocasions ofuscada, és per això que en primer lloc vull agrair l'ajuda dels usuaris del fòrum de Processing[6] que m'han assistit en nombroses ocasions davant de dubtes i conceptes difícils de programació que m'han servit per poder estructurar l'aplicació d'una manera més eficient.

L'aplicació està composta pel programa principal i quatre classes: Tweet, Tweets, pintaTweet i SonoAudio. En el següent gràfic mostro quina dependència tenen cada una respecte de les altres.



Imatge 10

En la *imatge 10* es pot observar com les classes més importants són *Tweets* (taronja) i *PintaTweet* (vermell), ja que per elles dues es processa el 80% de l'aplicació, mentre que la resta (groc i verd) complementen a altres classes o funcionen independentment amb el programa principal (blau).

[6] Processing Forum. (2010). Accès el 29 d'agost de 2012, des de Processing Forum: http://forum.processing.org

A continuació desenvoluparé de cada una d'aquestes classes i el programa principal.

06_1_Classe Tweet

Un *tweet* no és només un text extret de Twitter. Quan s'obté un d'aquests missatges es pot observar com hi ha moltes més dades que no s'estan tenint en compte com: la data i hora de creació, l'usuari que l'ha escrit, el nombre de *retweets* que ha tingut, etc.

Partint d'aquí, he decidit crear una classe, *Tweet*, per poder englobar precisament totes aquestes dades. D'aquesta manera, sempre que cridi un *tweet* l'objecte em tornarà totes les dades que el formen.

La classe *Tweet*, doncs, està definida pels següents camps:

· Una variable `int rt_tw` per guardar el número de *retweets* d'aquell missatge.
· Una variable `String text_tw` per guardar el text del *tweet*.
· Una variable `String data_tw` per guardar la data de creació del text.
· Una variable `String id_user` per guardar el número identificador de l'autor que l'ha escrit.

Aquests són els principals camps els quals adquireixen valor dins del constructor. No obstant, també he utilitzat altres camps que deriven dels anteriors per formatar la data i hora, calcular les posicions *x* i *y* del *tweet* i la mida que tindrà la imatge que el representi.

La data de creació guardada al `String data_tw` està en format *datetime*, és a dir amb la següent estructura: "aaaa-mm-dd hh:mm:ss", que és un requisit de la base de dades *MySQL*. Per a la meva visualització necessito poder diferenciar la data i l'hora. És per això que dins del constructor, mitjançant el mètode `split()` i diverses variables temporals del tipus `String[ ]`, finalment he pogut obtenir un `String data` ordenat segons el sistema europeu (dd-mm-aaaa) i un `String hora` en format "hh:mm".

També he creat quatre variables més: `int dia`, `int hora2`, `int horaH` i `int horaM`. Aquestes em serviran per guardar el dia, l'hora en minuts, l'hora, i els minuts respectivament.

Les necessito tenir en aquests formats per més endavant poder calcular la posició del *tweet* en els dos eixos, segons si es tracta de la visualització completa o filtrada segons dia.

A l'hora de separar el `String data_tw` inicial he tingut problemes per transformar-lo a un enter, ja que quan hi havia un zero davant o darrera d'una xifra –03 o 20, per exemple– la funció `int()` eliminava aquest zero donant un valor erroni –3 o 2, seguint amb l'exemple anterior. Per solucionar això he fet servir la funció `charAt()` que em permet separar una cadena de text per caràcters. A partir d'una funció condicional, si el número en `string` que li passo conté un cero en algun dels seus caràcters, aquest li afegeix manualment.

Per tant, un cop guardats en variables aquest valors puc calcular la posició on anirà el *tweet*. Per calcular-ho he hagut de diferenciar quan es tracta la visualització global o filtrada per dia. D'aquesta manera he creat quatre variables: `float posX`, `float posY`, `float posXD` i `float posYD`. Les dues primeres corresponen a la posició en global i les dues restants a la posició filtrada per dia.

La posició global consta d'un eix horitzontal que representa el dia i un eix vertical que representa l'hora.

Per calcular el primer, em trobo amb el problema que el període triat no comença el "1" i acaba el "30" sinó que va del "13 de maig de 2011" al "12 de juny de 2011" i per tant, necessito una funció que reguli el canvi de mes. En el període seleccionat hi ha 31 dies, dels quals 18 dies són de maig i 12 de juny. Al no repetir-se cap dia en aquesta franja de temps i ser el primer dia el "13" i l'últim el "12", això em permet establir una separació just en aquest últim número, el "12". L'algoritme que calcula la posició per l'eix horitzontal és el següent:

```
posX = ((ampladaTotal – espaiMarge) / númeroDiesTotals) * (dia + margeEsquerra)
```

A través d'una funció condiciono si el dia és major o menor de 12. Si es tracta del primer cas, significarà que es tracta del mes de maig i per tant a la variable `dia` de l'equació anterior li resto 13 posicions per tal que adquireixi les primeres posicions. En cas contrari, li sumo a *dia* 18, corresponent a que es tracta del mes de juny.

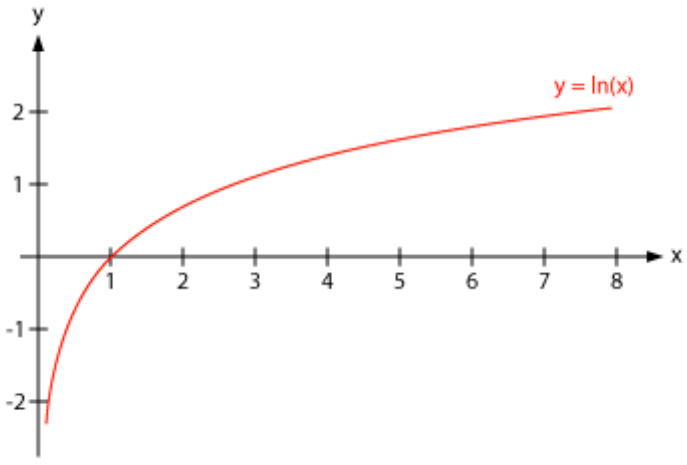
El càlcul de la posició en l'eix vertical respon a la següent equació:

```
posY = ((alturaTotal - espaiMarge) / 1440) * (horaEnMinuts + margeEsquerra)
```

El número “1440” respon al total de minuts que hi ha en 24 hores. A partir de la variable creada anteriorment `hora2` el càlcul resulta fàcil i sense complicacions com en el cas anterior.

Quan l’usuari decideix filtrar per dia, els eixos horitzontal i vertical son representats per `hora` i `minuts` respectivament. Per tant el càlcul dels dos eixos estan basats en l’algorisme utilitzat per calcular `posY`. En el cas de l’eix horitzontal `alturaTotal` es substitueix per `ampladaTotal` i el valor pel qual divideix és el total d’hores, és a dir, 24; el multiplicador es `horaH`. Per al segon cal realitzar una operació semblant, el divisor és el número total de minuts per hora, 60, i el multiplicador `horaM`.

Per últim, aquesta classe s’encarrega de calcular la mida de la imatge que representarà aquell *tweet*. El paràmetre a tenir en compte és el nombre de *retweets* guardat a la variable `rt_tw`. Hi ha molts *tweets* que no tenen cap *retweet*, bastant que tenen entre 1 i 50, i a partir d’aquí és menys probable però n’hi ha algun fins i tot amb més de 500. Si el càlcul fos lineal i a proporció, un *tweet* amb 0 *retweets* o amb 20 es veurien pràcticament iguals, ja que el valor màxim és massa gran i poc comú. Per tal de trobar una proporció que diferenciï més els valors petits i menys els grans he fet servir la funció logarítmica per representar aquests valors. Una funció logarítmica respon al següent esquema:



Imatge 11

En la *imatge 4* és pot observar com el creixement va des de –∞ a 1, després creix ràpid i en els últims valors es manté. Aquest és el comportament que utilitzaré per a escalar les imatges.

Per tant, l'algoritme que definirà la mida de la imatge respondrà a:

```
tamanyImg = log(rt_tw+2)*tamanyMínim
```

Tots aquests càlculs es realitzen dins del constructor. Fora d'ell he creat varis mètodes –"getters"– per accedir a cada un dels paràmetres que composen un *tweet* segons sigui necessari, ja que es considera una bona pràctica en els llenguatges orientats a objectes. Aquestes funcions són:

· `String getText()`. Retorna el text del *tweet*.
· `int getRT()`. Retorna el número de *retweets*.
· `String getData()`. Retorna la data (sense la hora) formada en el sistema europeu que va ser enviat el *tweet*.
· `String getHora()`. Retorna l'hora que va ser enviat el *tweet*.
· `float getPosXMes()`. Retorna la posició per l'eix horitzontal en la visualització global.
· `float getPosYMes()`. Retorna la posició per l'eix vertical en la visualització global.
· `float getPosXDia()`. Retorna la posició per l'eix horitzontal en la visualització filtrada per dia.
· `float getPosYDia()`. Retorna la posició per l'eix vertical en la visualització filtrada per dia.
· `float getTamanyImg()`. Retorna la mida de la imatge que representarà aquell *tweet*.
· `String getIdUser()`. Retorna el número identificador de l'autor del *tweet*.

06_2_Classe Tweets

La classe `Tweets` és qui s'encarregarà de crear tots els *tweets* a partir de la base de dades ja creada i també comprovarà si l'usuari ha posat el cursor a sobre d'una de les imatges que representes aquests *tweets*. *S*erà l'única classe que tindrà accés a la classe `Tweet` de manera directa.

Per poder accedir a la base de dades des de Processing, he utilitzat la llibreria `SQL`[7]. La connexió es realitza mitjançant la classe `MySQL` que proporciona. En primer lloc cal instanciar un nou objecte, en el meu cas anomenat `dbconnection`, al constructor passant-li el següents paràmetres: servidor, nom de la base de dades, usuari i contrasenya. Hi ha a més, un cinquè paràmetre que necessita la classe `MySQL` per funcionar: la referència a l'`Applet`. A l'estar dins d'una classe –`Tweets`– i no al programa principal, necessita aquesta referència. En aquest cas l'obtinc a través del constructor passant-li el paràmetre `PApplet`. En referència al constructor, en aquest cas he hagut de programar-ne dos. Un per guardar els *tweets* de la visualització global i l'altre per a la visualització filtrada per dia. Cadascun d'ells conté dins la *query* concreta per a obtenir els *tweets* que es necessitin en cada cas.

Un cop creat l'objecte `dbconnection` i es conecta mitjançant el mètode `connect()`, crido el mètode `query()` i a dins li especifico el text de la consulta. A cada constructor realitzo dues *querys*: una per obtenir el número total de *tweets* que em retornarà i l'altre els camps que necessitaré per crear tants objectes `Tweet` com *tweets* hagi obtingut de la primera consulta.

La primera consulta és necessària per poder indicar en el moment de la creació de la *array* d'objectes `Tweet` el número de camps que necessitarà. Un cop instanciada aquesta *array* realitzo la segona *query* que serà la que utilitzi després per crear cada objecte `Tweet`. La consulta que retorna la informació de la base de dades és la següent:

```
SELECT t.created_at, t.text, t.retweet_count, t.user  FROM tweets t WHERE ((t.text
LIKE '%#acampadabcn%' OR t.text LIKE '%#tomalacalle%') AND t.created_at >  '2011-05-13
00:00:00' AND t.created_at < '2011-06-12 23:59:59') ORDER BY t.created_at ASC
```

Seguidament creo un bucle *for* on donada una variable que és més petita que el nombre total de *tweets* adquirit en la primera consulta, s'incrementa i també s'executa el mètode `next()`. Aquesta funció serveix per passar a la fila següent del resultat de la consulta. Mentre no arribi a l'ultim *tweet*, a cada volta crea un nou objecte `Tweet[]` on hi guarda dins: el número de *retweets*, el text, la data de creació i el identificador de l'autor del *tweet*.

[7] SQLibrary. (2008). Accès el 16 de juliol de 2012, des de Bezier: http://bezier.de/processing/libs/sql

Com he dit abans, en aquesta classe he creat dos constructors. El primer és el que acabo d'explicar. El constructor per a la visualització filtrada parteix de base del constructor anterior però afegint alguns mètodes més. Aquest constructor requereix que a part de passar-li la referència a l'`Applet`, li arribi un enter que representa la data per la que l'usuari ha decidit filtrar.

Les *querys* de consulta son les mateixes excepte per un detall: en la part on especifico la data per filrar, he substituït el camp del dia i del mes per variables. La consulta resultant és la següent:

```
SELECT t.created_at, t.text, t.retweet_count, t.user  FROM tweets t WHERE ((t.text
LIKE '%#acampadabcn%' OR t.text LIKE '%#tomalacalle%') AND t.created_at >  '2011-
"+mes+"-"+data_consulta+" 00:00:00' AND t.created_at < '2011-"+mes2+"-"+data_final+"
23:59:59') ORDER BY t.created_at ASC
```

Les variables `mes`, `data_consulta`, *`mes2`* i `data_final` son calculades prèviament a la consulta. Les variables `data_consulta` i `data_final` prenen el valor del paràmetre `_data_consulta` que arriba a través del constructor. Seguidament mitjançant una funció *if* dono valor a les variables `mes` i `mes2` segons si el dia és més gran o igual a "13" i per tant, correspon al mes de maig o pel contrari, si és menor, es tracta de juny.

De la mateixa manera que passava amb el constructor anterior, després d'executar la consulta, s'inicia un bucle per omplir els objectes `Tweet[]`.

Fora del constructor, aquesta classe realitza altres funcions. Per una banda conté un mètode anomenar `Tweet getTweet(int index)` que retorna un *tweet* segons l'índex especificat. I per l'altra com ja avançava a l'inici de l'apartat, comprova si l'usuari es troba damunt d'una de les imatges que representen el *tweet*.

Per fer-ho, he hagut de crear dues funcions `Tweet mouseTocaImgMes(int x, int y)` i `Tweet mouseTocaImgDia(int x, int y)` que consulten una tercera funció anomenada `boolean overImg()`. Aquestes funcions són necessàries ja que Processing no proporciona cap mètode del tipus *RollOver()* com trobaríem a *Flash ActionScript*, per exemple. Per tant, he

creat questes funcions basant-me en l'exemple que Processing proporciona a través de la seva documentació online.

Les funcions `mouseTocaImgMes()` i `mouseTocaImgDia()` reben la posició *X* i *Y* del ratolí per paràmetre. La seva tasca és, mitjançant un bucle *while*, enviar a cada volta les posicions *X*, *Y* del cursor que ha rebut com a paràmetres, i la posició *X*, *Y* i la mida de la imatge del *tweet* a la funció `overImg()`. Tots dos són idèntics excepte pels paràmetres que envia cada un que es corresponen o als *tweets* de la visualització global o als de la visualització filtrada per dia. Mentre la funció `overImg()` no retorni *true* es continuarà executant fins que hagi recorregut tots els *tweets,* surti del bucle i retorni *null*. En cas afirmatiu surt del bucle i retorna el *tweet*.

La funció overImg(), rep els paràmetres esmentats anteriorment. Mitjançant una bloc *if* realitza la següent comparació:

```
mouseX >= (x- (width/2)) && mouseX <= (x- (width/2))+width &&
mouseY >= (y - (height/2)) && mouseY <= (y - (height/2))+height
```

Si el resultat és que el cursor es troba dins de la superfície d'una imatge retorna *true* a la funció que la cridava –`mouseTocaImgMes()` o `mouseTocaImgDia()`, si no *false*.

06_3_Classe pintarTweet

`PintarTweet` és la classe responsable de representar tot allò que faci referència a la visualització –botons, guies i ajudes visuals, no. Treballa mà per mà amb la classe `Tweets`, ja que és ella qui li proporcionarà la informació dels *tweets* que hagi de pintar a l'aplicació.

He creat aquesta classe amb l'objectiu de poder controlar de accions de representar les imatges o els textos segons convingui. Es tracta d'una classe que donats un cert nombre de *tweets* els mostra segons convingui.

El constructor d'aquesta classe és molt senzill, ja que les vertaderes accions les realitzarà en els diferents mètodes que en ella he creat. Per tant, el constructor requereix rebre per paràmetre els `Tweets` que ha de representar. Aquesta classe ja incorpora en l'apartat de

camps un objecte `Tweets`, de manera que pugui comunicar-se amb `Tweets` directament sense la necessitat de demanar la informació a través del programa principal –qui li passa els Tweets per paràmetre.

Un cop associat el paràmetre, rebut a través del constructor, amb l'objecte `Tweets` creada en aquesta classe; carrego les dues imatges que representaran els *tweets*. Creo tres objectes PImage abans d'entrar al constructor: un pel logotip associat al *hashtag #acampadabcn*, un altre per l'associat al *hashtag #tomalacalle* i l'últim per carregar-hi la imatge de perfil de l'autor del *tweet*.

En el constructor només dono valor als dos primers objectes `PImage` ja que sempre seran constants. El tercer objecte el definiré més endavant. A través del mètode `loadImage()`, puc carregar una imatge que es trobi dins de la carpeta `data`, o d'una URL[8].

Fins aquí les operacions del constructor. Com he dit abans, aquesta classe es caracteritza sobretot per les funcions que realitza de manera independent.

Per una banda, allò que primer mostra l'aplicació són dues imatges repetides moltes vegades però de diferent mida. La funció que s'encarrega d'això s'anomena `pintaImgsMes()`. Seguint la tònica de les classes anteriors, existeix una funció per a la visualització global i una altra per a la filtrada per dia, anomenada `pintaImgDia()`. Totes dues realitzen una operació senzilla – representar tots els tweets en la posició que reben de cada un, però amb un dificultat afegida: determinar si ha de mostrar la imatge referent a `#acampadabcn` o a la de `#tomalacalle`.

Per tant, necessito un algorisme que determini si dins del text del *tweet* s'hi troba el text *#acampadabcn* o *#tomalacalle*, ja que la imatge que s'associi amb el *tweet* dependrà de si conté un *hashtag* o l'altre.

Primer de tot he creat un bucle *for* que es repetirà tantes vegades com *tweets* hagi de pintar. Un cop a dins guardo dins d'una variable de tipus `Tweet` anomenada `tw`, el *tweet* que s'ha de

[8] Recordar que Processing no permet la consulta de pàgines externes al servidor on es trova allotjat. L'única manera de fer-ho seria verificant l'aplicació però això representaria un cost. Per aquest projecte les imatges es trobaran sempre al servidor.

representar cada vegada. Per poder saber si dins del text hi ha el *hashtag #acampadabcn* o *#tomalacalle*, necessito separar el text d'aquell *tweet* per paraules, fent servir el mètode split(). Un cop guardats les paraules en un *array* de tipus `String` anomenat `wordsTweet`, creo un nou bucle *for* dins de l'anterior.

Aquest segon bucle s'executarà tantes vegades com paraules contingui el *tweet*, fent ús de la propietat `length` a l'*array* `wordTweet`. Dins del bucle incorporo una nova funció de tipus condicional. Aquesta funció està definida per dues condicions. La primera comprova si la paraula que li arriba a través de `wordsTweet[]` coincideix amb "#acampadabcn", a través del mètode `equals()` que permet comparar dues cadenes `String`. Si es compleix, mitjançant el mètode `image()` pinta la imatge carregada anteriorment al constructor, a les posicions i amb la mida que li especifica el *tweet*. La imatge es representada amb `imageMode()` centrat i amb cert punt de transparència gràcies al mètode `tint()`.

Si aquesta primera condició no es compleix, passa a la segona on la comparació es fa en aquest cas amb "#tomalacalle". Si es compleix representarà la imatge de la mateixa manera que la condició anterior.

En cas de no complir-se cap de les condicions anteriors no es realitza cap acció i, per tant, si no ha arribat al final de les paraules o dels *tweet* es torna a executar tot.

Per una altra banda, he creat una funció `pintaTextMes(Tweet t)`, que de la mateixa manera que ha passat en anteriors casos, també existeix una segona funció pràcticament idèntica anomenada `pintaTextDia(Tweet t)` per a la visualització filtrada per dia. Totes dues funcions s'encarreguen de mostrar el text del `Tweet` que els arriba per paràmetre. Per fer-ho, a partir d'obtenir el *tweet*, la funció pot cridar la seva posició, el text, i la data i hora, gràcies als mètodes creats dins la classe `Tweet` que retornen precisament aquestes dades.

El text que es mostri ha d'aparèixer ubicat en la cantonada inferior dreta, en diagonal, de la imatge que representa el *tweet*. És per això que per poder calcular la posició requeriré el la mida de la imatge que puc obtenir a través del mètode `getTamanyImg()` de la classe `Tweet`. Per tant l'algoritme que determina la posició del text és:

```
posX_img = t.getPosXMes()+(t.getTamanyImg()/2)+marge;
```

```
posY_img = t.getPosYMes()+(t.getTamanyImg()/2)+marge;
```

Com les imatges es situen centrats en l'eix que els correspon, per calcular quan acaba la seva superfície haig de sumar-li la seva meitat a la posició original.

Un cop obtingudes les posicions respecte l'eix X i Y del text, determino la manera que es mostrarà aquest. El text es mostrarà en una petita caixa de 300 píxels d'amplada. Després de carregar la font des de la carpeta data, ajusta l'amplada de text a 300px i assignar-li una mida de lletra i un interlineat, em trobo amb el problema de no saber com calcular l'altura de la caixa.

Després d'investigar per Processing i els seus fòrums de discussió he trobat un usuari que compartia una possible solució a aquest problema. Es tracta de cridar a una funció fora de `pintaTextMes()` anomenada `calculateTextHeight()`. Aquesta funció requereix que se li passin per paràmetre les següents dades: el text, l'amplada del text, la mida de la font i l'interlineat.

La funció `calculateTextHeight()` crea una *array* `String []` anomenda `wordsArray`, i un `String` temporal buit. També crea una variable del tipus `int` anomenada `numLines` que servirà per comptar el número de línies que tindrà aquell text. Fent ús del mètode `split()` que permet separar una cadena de text segons un criteri, en aquest cas un espai en blanc, guardo a l'*array* `wordsArray` totes les paraules del text del *tweet* que li ha arribat per paràmetre.

Seguidament, a través d'un bucle *for* on donarà tantes voltes com paraules tingui –mesurat amb la propietat `length` de l'*array*. Dins d'aquest bucle hi ha una nova funció, però en aquest cas de condició. Si l'amplada del text –mesurada a través del mètode `textWidth()`– de la variable temporal sumada a la paraula de l'array wordsArray és inferior a 300 (l'amplada indicada del text), la variable temporal afegeix a la seva cadena aquesta última paraula i torna a començar. Si fos superior, l'acció es repetiria però la variable que compta les línies incrementa tantes vegades com la funció hi entra.

Quan ha arribat al final de totes les paraules del text s'incrementa la variable `numLines` un cop més per tenir en compte la primera línia que mai es compta a la funció. Per últim, l'altura total del text es calcula multiplicant `numLines` per la suma de `textDescent()` i `textAscent()`. Aquest últims mètodes retornen el punt més alt i el punt més baix de la lletra que sumades

donen l'altura d'aquella tipografia per una línia. Un cop calculat això, la funció retorna aquest valor.

Després d'obtenir l'altura del text, abans de passar a representar-lo, especifico mitjançant una funció condicional el comportament de la posició X i Y quan aquests es troben a prop de les cantonades. Com el text sempre es representa a la cantonada dreta inferior, en cas que la posició sumada a l'amplada o altura del text (segons si es tracta de la posició `X` o de la `Y`), el text es pintarà a la cantonada inferior esquerra o a la superior dreta respectivament.

Un cop determinat aquets comportament defineixo un rectangle amb `rect()` en la posició del text i amb mida del text. Seguidament a través del mètode `text()` li passo el text, la posició en el dos eixos, l'amplada i l'altura. Per últim crido a una funció independent que hi ha a la classe anomenada `pintaData()` que representa la data a sota del text del *tweet* en un altre color i mida de font, per diferenciar-lo, i tenint en compte l'altura calculada anteriorment. Com és un text d'una sola línia sempre, el rectangle te en compte aquesta altura fixa i li sumo en el moment que el creo.

Quan l'usuari es col·loca a sobre d'una de les imatges representades i, per tant, la funció `mouseTocaImgMes()` o `mouseTocaImgDia()` indiquen al programa principal que efectivament el cursor es troba a sobre d'un *tweet* es crida la funció `pintaTextMes()` o `pintaTextDia()` que acabo de desenvolupar. A més a més, també es crida una funció més dividida en les dues variant segons si es tracta de la visualització global o la filtrada per dia: `pintaImgMes()` o `pintaImgDia()`.

Aquesta funció és senzilla. Es tracta de pintar a sobre de la imatge del logotip, la imatge de perfil de l'autor del *tweet*. Dins la carpeta `data` estan totes les imatges de perfil dels usuaris dels *tweets* guardats a la base de dades.

Per poder mostrar aquestes imatges de perfil, he pogut tornar a comptar amb la col·laboració de l'Albert Meroño. La base de dades que he construït conté un camp a la taula 'users' on hi ha l'adreça web de la imatge. Processing per motius de seguretat no permet a l'aplicació consultar webs externes que no siguin les del propi hosting on està allotjada l'aplicació sinó està verificada. De manera que fer referència a aquestes imatges l'Albert va programar un *script*, anomenat *get-twitter-portraits*, que consultés aquestes URLs i es descarregués la imatge de

perfil. Aquest *script* també es pot trobar a Internet per a que altres usuaris el puguin fer servir[9]. Les imatges, per tant, no es troben a la base de dades ni a Twitter, sinó de manera local al servidor.

Aquestes imatges s'anomenen com l'identificador de l'usuari. D'aquesta manera a través de la funció `loadimage()`, utilitzada anteriorment per a carregar les dues imatges dels logotips, puc carregar la imatge que correspongui amb aquest *id*, gràcies a rebre per paràmetre de quin `Tweet` es tracta. Creo la imatge donant-li la mateixa posició i mida. D'aquesta manera simulo un *RollOver* d'una imatge que canvia segons si el cursor es troba a sobre o no.

06_4_Classe sonoAudio

Tal i com avançava al principi d'aquest bloc, sonoAudio és una classe que treballa independentment de la resta. Es comunica directament amb el programa principal proporcionant-li allò que necessiti. És la responsable de proveir so a la aplicació de dues maneres. Per una banda executa sis arxius a la vegada però un sol cop amb so que servirà per ambientar la aplicació. Per l'altra, executarà un arxiu triat aleatòriament cada cinc segons.

Per poder reproduir so a Processing, he utilitzat la llibreria `minim`[10]. Primer de tot creo un objecte `Minim` anomenat `minim`. Quan instancio un *nou* objecte al constructor, de la mateixa manera que passava amb la llibreria `SQL`, `Minim` necessita rebre per paràmetre la referència a l'aplicació principal mitjançant `PApplet`. Tot seguit, també creo un objecte `AudioPlayer` per als àudios que sonaran aleatòriament cada cinc segons. Per tant, aquest objecte serà una *array*, `AudioPlayer[] in`. Per als àudios que sonaran una vegada però tots junts, creo un objecte per a cada un d'ells: `AudioPlayer a, b, c, d, e` i `f`.

Dins del constructor també creo una nova instància de l'objecte `AudioPlayer []` indicant-li el número de camps que tindrà l'*array*. Per poder diferenciar els dos tipus de reproducció he creat una funció per a cada una.

[9] *Get Twitter Portraits*. (2012). Recuperado el 09 de setembre de 2012, de Github: https://github.com/albertmeronyo/get-twitter-portraits

[10] Minim. (2007). Accès el 27 de juliol de 2012, des de Code Compartmental: http://code.compartmental.net/tools/minim

La funció `playAudio()` regula la reproducció d'arxius cada cinc segons. Aquests arxius es troben al servidor fora de la carpeta `data`[11]. Per carregar l'àudio he utilitzat el mètode `loadFile()` de l'objecte `minim` i li assigno a l'`AudioPlayer[]` creat. Substitueixo el nom original de l'arxiu per un número, de manera que pugui cridar-los a través d'un `random(num)`. La reproducció dels àudios cada cinc segons es controla des del programa principal. A partir d'una funció creada en el `draw()`, quan ha passat el temps indicat crida la funció `playAudio()` que estic explicant ara. En el següent apartat explicaré el programa principal amb aquesta funció inclosa.

Per tant, aquesta funció, `playAudio()` es carrega des del programa principal en el moment que ho precisa. El principal problema amb el que m'he trobat és que quan l'aplicació porta varis minuts executant-se, comença a haver masses objectes `AudioPlayer[]` i es bloqueja o dóna un error indicant que la memòria està plena. Per evita això, he creat una funció on sempre hi hagi un màxim de cinc objecte `AudioPlayer[]` executant-se. Aquest *array* necessita, perpo, un manteniment específic doncs cal controlar en tot moment quin objecte `AudioPlayer` s'ha d'aturar, quin ha de reproduir so i quin ha de carregar nous fitxers d'àudio.

Per fer això he utilitzat el concepte *mòdul* per crear un comptador –de 0 a 4 en aquest cas. El *mòdul* és una operació matemàtica, representada pel signe "%", que dona per resultat el residu d'una divisió. En particular jo només vull tenir cinc objectes. D'aquesta manera, divint per 5 de 0 a infinit, sempre donarà restant zero quan es divideixi per ell mateix o per un múltiple i mentrestant, el residu anirà de 0 a 4.

Per tant, creo una variable `int count` que en el constructor li dono valor `0`. Dins la funció `playAudio()` un cop ha realitzat els càlculs i operacions necessaris, el comptador s'incrementa. Per controlar el número màxim –que serà el divisor– creo una altra variable `int max_N = 5`. A partir d'aquí el valor que crida l'objecte de l'*array* es el resultat d'aquesta

[11] Els arxius que es troben a la carpeta data s'inclouen en el .jar de l'aplicació quan s'exporta. Els àudios tenen un pes considerable. Si es troben dins de l'Applet, l'aplicació triga molt a executar-se des d'un navegador. Deixant-los fora, l'aplicació es carrega molt més ràpid i quan reprodueix un àudio no necessita estar pre-carregat abans, és a dir, a mesura que es descarrega es va reproduint.

operació: `count % max_N`. Això em permet mitjançant un bloc *if* tancar el primer objecte perquè després es torni a crear. Per saber quan ja s'han creat cinc objectes, faig que el bloc *if* comprovi si el resultat de `count / max_N` és major o igual a `1`. Cada cinc objectes el resultat de la divisió s'incrementa, sinó només canvia el residu. Per tant quan aquest número canvia significa que ja s'han creat de nou 5 objectes més.

Si això es compleix i entra dins la funció que comprova si ja s'han creat aquests cinc objectes, a través del mètode `close()` tanco l'objecte més antic. Després d'aquesta acció, surt de la funció i llavors mitjançant el mètode `play()` comença a reproduir l'arxiu carregat i sonarà a l'aplicació. Tot seguit s'incrementa el comptador per la propera vegada que sigui cridada aquesta operació.

La segona funció que he creat és per als àudios fixes que sonen només un cop i que es carreguen només executar l'aplicació. En aquesta cas les operacions de dins són senzilles: carrego a cada objecte `AudioPlayer` `a`, `b`, `c`, `d`, `e` i `f` el fitxer corresponent i després utilitzo el mètode `play()`.

06_5_Programa Principal

El programa principal és el que s'encarrega per una banda de representar la visualització de les imatges dels *tweets*, i la estructura de la interfície i per l'altra de cridar les funcions perquè soni l'àudio i preguntar a `Tweets` si l'usuari està tocant alguna de les imatges.

En primer lloc, creo instàncies dels objectes que faré servir. Dos objectes Tweets i dos objectes pintarTweet: un per mostrar els *tweets* de la visualització global i l'altre per a la filtrada. Un objecte sonoAudio per a reproduir l'audio; dos objectes Tweet; i un objecte controlP5.

En segon lloc creo l'estructura de la interfície. La aplicació tindrà una part on hi hagi representades les imatges i una altra on hi hagi una sèrie de botons per filtrar per dia o tornar a la visualització global. Aquests botons els he creats fent servir la llibreria `controlP5`[12] que

[12] Processing GUI, ControlP5 . (2007). Accès el 25 de juliol de 2012, des de Sojamo: http://www.sojamo.de/libraries/controlP5/

proporciona diversos mètodes per a la creació d'aquests tipus d'elements. Creo aquests botons i alguns textos de suport dins dels `setup()`. Aquesta funció s'executa una vegada a l'arrencar l'*Applet*, i s'encarrega de configurar totes les variables que seran necessàries durant l'execució

Per controlar si l'usuari fa clic sobre un botó, creo una funció `controlEvent()` on rep com a paràmetre un valor de tipus `ControlEvent`. A través d'aquest `Event` i els mètodes `getLabel()` o `getValue()`, puc saber si l'usuari a clicat sobre un botó, de quin es tracta i executar el `pintaImgs()` que correspongui –segons si es tracta de filtre per dia o tornar al global.

Dins del `setup()` creo també una nova instancia de l'objecte `Tweets` per a representar tots els *tweets*; un nou objecte `pintarTweet` fent referència a l'objecte anterior com a paràmetre; i un nou objecte `sonoAudio`. Un cop creats, crido les funcions `playAudio()`, `playAudioAmbient()` i `pintaImgsMes()` per que s'executin només començar des del `setup()`.

Dins del `draw()`, que s'executa un cop cada vegada que s'ha de pintar un *frame*, he creat dues funcions condicionals. Una controla el tipus de funció *RollOver* que ha d'executar i l'altre controla el `frameRate` per crear un àudio cada 5 segons com he indicat a l'apartat anterior.

La primera funció determina a través de tres casos si l'usuari ha clicat un botó per filtrar per data, si ha clicat el botó per tornar a veure tots els *tweets* o cap dels anteriors. Cada un dels mètodes que hi ha dins d'aquest condicionant determina quina funció per comprovar si hi ha *RollOver* s'executarà en cada cas.

Existeixen dues funcions *RollOver*: `RollOverTweetMes()` per a la visualització global i `RollOverTweetDia()`, per a la filtrada segons dia. Totes dues són idèntiques excepte pel tipus de mètodes que criden a dins que cada una es referirà o a la global o a la diària.

La funció en primer lloc crida la funció que es troba dins de `Tweets` `mouseTocasImg`[Mes o Dia]`()` i li passa com a paràmetres la posició del cursor mitjançant els mètodes `mouseX` i `mouseY`. Com ja he explicat quan he desenvolupat l'aparta `Tweets`, si es produeix aquest contacte la funció retorna el `Tweet`, en cas contrari `null`.

Per tant dins de la funció RollOver, he creat una funció condicional que té en compte tres casos. El primer contempla si la funció `mouseTocaImg` ha retornat `null`. En cas afirmatiu només s'executa el mètode `setupInicial` (un pel global i una altre pel diari) que esborra dibuixant un rectangle la zona de la visualització –fent que la zona dels botons del `setup()` no s'esborri com passaria amb `background()` –, pinta els eixos i pinta de nou els *tweets* del tipus que pertoqui (global o dia). Aquesta funció és necessària ja que no he trobat un mètode que elimini del *stage* un objecte en concret. És per això que aplico capes que van esborrant el que hi havia anteriorment i ho redibuixo tot, per tal de simular aquest comportament.

Si la funció retorna un `Tweet`, es contemplen dos casos: si prové d'un espai sense imatge o d'un *tweet* anterior. En el primer cas executa de nou el setupInicial que esborrà el que hi havia prèviament i cridarà les funcions `pintaImg()` i `pintaText()` que mostrarà la imatge de l'autor de *tweet* i el missatge respectivament. Recordar que el `Tweet` que retorna la funció anterior es passa com a paràmetre a aquests dos mètodes. L'últim cas fa el mateix però sense executar el mètode `setupInicial()`. En aquests dos casos es guarda sempre el Tweet dins la variable `tAnterior`, per saber si l'usuari prové o no ja d'un *tweet* en la següent comprovació.

La segona funció dins del `draw()` alenteix el número de *frames* per segon, sense haver de canviar-ho en el `frameRate()` del `setup()` –que distorsionaria altres funcions. Això ho he aconseguit fent servir de nou la operació *mòdul*. Fent servir el mètode `frameCount`, puc saber el *frame* que s'està executant en aquell moment. Si el divideixo per "150" –corresponent a 5 segons per cada 30 *frames* per segon– i el resultat el comparo amb "0", vol dir que el programa entrarà una vegada cada 5 segons a aquesta funció. Un cop a dins executa la funció `playAudio()`.

07_Aspectes de disseny

La part estètica de l'aplicació i, del projecte en general, deriva de la línia gràfica que proposa el col·lectiu del Moviment #15M. Aquest projecte vol ser una de les moltes aportacions que intenten oferir diferents punts de vista o lectures d'aquest fenomen. És per això que conservaré alguns dels elements propis del moviment, però alhora n'afegiré d'altres que tindran relació directa amb aquest projecte i que seran, per tant, un segell personal.

El Moviment #15m està composat per tot un seguit d'organitzacions com *Democracia Real Ya*, *Toma la Calle*, *SpanishRevolution* o cadascuna de les *Acampades*, entre moltes altres. En general cadascuna s'encarrega de la seva pròpia difusió gràfica. No obstant, hi ha dos elements que mantenen en comú en general: un és el color, groc ocre –tot i no tenir unificada la tonalitat– i l'altre la tipografia, de tipus *Stencil*, sense especificar-ne una de particular.

La imatge gràfica que he proposat manté el color groc escollint la tonalitat que proposa l'*Acampadabcn*. He triat aquesta tonalitat ja que em sembla un color molt viu amb un toc de taronja que suavitza el grau de lluminositat i li dona més cos. És tracta d'un color que en un context amb colors en escala de grisos crea un punt d'interès sobre l'usuari sense enlluernar-lo ni dificultant la seva lectura. L'elecció d'aquest color representa el pont de connexió amb aquest moviment per tal de mostrar el meu interès d'aportar aquest projecte com una eina que ofereixi un punt de vista alternatiu.

Per tant, el color m'ha semblat adient mantenir-lo ja que crec que és, junt amb l'element "#", un símbol generalitzat d'identitat d'aquest moviment. No està clar d'on prové aquest l'elecció del groc. Personalment penso que sorgeix arrel de l'*AcampadaSol* de Madrid. Quan es van instal·lar per segon cop, després que els desallotgessin el 16 de maig, ho van fer amb molta més força i van crear una imatge que representés aquesta acció.

Aquest "logotip" es composa per una tenda d'acampada, un os –símbol de Madrid– i el color groc –referent a Sol. Com a pioners que van ser, crec que les acampades que van sorgir després a moltes ciutats van mantenir el color groc per mostrar el seu recolzament amb la iniciativa de Madrid. Altres van crear la seva pròpia imatge al marge d'això.

Imatge 12

Imatge 13

Imatge 14

Imatge 15

Com es pot observar a les *Imatges 12* i *14* són pràcticament iguals, mentre que la *Imatge 13* i *15* varien la tonalitat del groc però mantenint l'element dinàmic del cercle i en el cas de l'última imatge incorpora una tipografia *Stencil*.

La tipografia *Stencil* és molt interessant ja que tradicionalment eren típiques d'un context militar. La seva principal característica és aquest trencat que hi ha sempre a les lletres. Aquest element, permet que pugui crear-se un estergit –de l'anglès *stencil*– i poder pintar sobre superfícies amb diferents materials el relleus que deixen els seus espais. Aquest moviment social s'ha apropiat d'aquest ús i n'ha fet una de les seves principals característiques. Visualment són molt potents ja que per poder-se pintar necessiten tenir prou espai buit en la plantilla (*Imatge 16*). A més aquests textos acostumen a ser representats lleugerament inclinats. D'aquesta manera emulen que han estat pintats a mà –pintar recte és molt difícil.

Imatge 16

En el meu cas, com tracto amb textos més llargs que un *eslògan*, necessito una tipografia que no tingui tant pes visual –per no carregar al lector– i que alhora sigui dinàmic amb un punt orgànic o natural que no doni la impressió de ser mecànica. Vull que el text d'alguna manera

estigui representat com si fos la mateixa veu de la persona i no com un text artificial produït per una màquina.

És per això que he escollit la tipografia *Terminal Dosis Light*. Aquesta tipografia és molt senzilla, fina i amb corbes que li donen un dinamisme natural. Una altra característica que m'agrada és el seu comportament quan canvia de mida: en grans proporcions no carrega la pantalla i és molt llegible.

Imatge 17

08_Creació del plafó resum

L'objectiu del plafó és representar el projecte de manera que funcioni com un resum visual conceptual. El principal concepte del meu projecte són les *multituds intel·ligents* que mitjançant la tecnologia de microbbloging Twitter creen el Moviment #15M.

Partint d'aquí, l'objectiu a nivell pràctic deriva en una aplicació capaç de narrar aquests esdeveniment a partir de les veus que la composen. Per tant, aquest és el primer element que he treballat en la creació de plafó.

Per una banda he buscat una imatge que pogués representar l'essència del moviment i alhora que mostres les *multituds*. Aquesta imatge la he pogut aconseguir a través de Flickr, on el fotògraf Julien Lagarde[13] hi tenia penjades vàries de l'acampada de Barcelona, entre elles una de perfecte per al plafó[14].

Es tracta d'una fotografia en blanc i negre on s'hi poden observar moltes persones amb les mans aixecades. El fet que en aquesta imatge la gent surti d'esquena i per tant no se'ls vegi la cara, és perfecte per representar el concepte de *multituds*.

L'altre punt és com mostrar que aquella *multitud* està composada per moltes veus. Per aquesta qüestió he aprofitat les imatges de perfil que he utilitzat per a l'Aplicació. Mitjançant el programa *AndreaMosaic*[15], creo un mosaic a partir de les imatges de perfil per representar la fotografia de les *multituds*.

[13] Julien Lagarde's Photostream. (2008). Accès el 30 d'agost de 2012, des de Flickr: http://flic.kr/p/9MhsHs

[14] Tot i tenir la fotografía una llicència Creative Comons, m'he posat en contacte amb l'autor per tal de tenir el seu permís explícit per a la seva manipulació, mantenint la seva autoria –per suposat.

[15] AndreaMosaic. (2004). Accès el 30 d'agost de 2012, des de AndreaMosaic Home Page: http://www.andreaplanet.com/andreamosaic

El resultat obtingut té un efecte molt interessant: des de lluny és pot apreciar la fotografia on es veuen moltes mans aixecades per part del col·lectiu, i si l'espectador s'apropa pot veure en detall com la imatge en realitat està composada per les imatges dels perfils d'usuari de Twitter de totes les veus que *a priori* no es veien.

El fet que la imatge estigui en blanc i negre li dona més dramatisme i també fa referència a un esdeveniment que ja ha passat, com és el període de les acampades.

En la part superior, fent servir la mateixa tipografia utilitzada en l'aplicació, he col·locat el títol en color blanc del projecte destacant en lletres més grans el terme "#15M". A més a més, he fet coincidir la posició del text amb el punt on hi ha les mans per representar aquest punt de connexió entre la *multitud* i el desig de ser escoltats a través del moviment.

Per trencar amb el blanc i negre del plafó, he dibuixat una franja del mateix color groc ocre[16] utilitzat en l'aplicació per a col·locar allà les meves dades, l'any de la promoció, etc. Aquesta franja està una mica inclinada per fer referència als *stencils*. Les dades, però, no es troben en una capa de text, sinó que estan "foradades" en aquesta franja, deixant veure el que hi ha a sota, fent al·lusió també als *stencils* tan utilitzats pel moviment. Tant a la franja com al text del títol li aplico uns punts de transparència per tal de no tapar cap de les veus que hi ha representades a sota.

[16] Pantone 116C (solid coated)

09_Desenvolupament plataforma web online

Després de programar l'aplicació per visualitzar el *tweets* del enviats sobre el Moviment #15M, queda l'últim aspecte a desenvolupar: la plataforma web.

Tractant-se d'un moviment sorgit a Internet el més raonable és, en aquesta primera versió *Beta*, desenvolupar un portal online on aquesta aplicació sigui accessible per tots aquells usuaris que vulguin trobar una eina per construir una narració alternativa.

La primera decisió ha estat el nom de domini. Aquest projecte l'he titulat: "*Més enllà dels Trending Topics: una visualització sonoritzada del #15M*". Per a mi és un títol que engloba tant els aspectes teòrics com pràctics treballats al llarg de les dues memòries. No obstant, no és el nom ideal per a un domini; és un títol massa llarg.

Per tal de no perdre la coherència ni la cohesió amb projecte he decidit fer una versió d'aquest títol: "més enllà del 15m". He tret el símbol "#" ja que en els dominis pot ocasionar problemes i confusió a l'hora d'escriure-ho. Per altra banda l'extensió del domino que he triat és el ".org". Aquesta extensió és la més utilitzada en portals que volen aportar contingut social i cultural. D'aquesta manera l'usuari, d'entrada veient el nom de domini "mesenlladel15m.org" pot endevinar que en el portal hi haurà informació sobre el Moviment #15M però aportant un aspecte cultural i social.

Per altra banda, aquest projecte l'he emmarcat sota la llicència de *Creative Commons reconeixement – compartir igual 3.0 unported*[17], el qual permet copiar, distribuir i comunicar públicament l'obra. Tanmateix és permet la transformació de l'obra i l'ús comercial, sempre i quan es mantingui l'autoria original i és distribueixi sota la mateixa llicència. D'aquesta manera aquest projecte és considerat un "treball per a la cultura lliure".

L'estructura d'aquesta plataforma l'he construïda a partir dels següents apartats: Projecte, Visualització, Codi, Agraïments, Contactar.

[17] Reconocimiento-CompartirIgual 3.0 Unported. (2010). Accès el 01 de setembre de 2012, des de Creative Commons: http://creativecommons.org/licenses/by-sa/3.0/deed.es_ES

L'apartat *Projecte* serà la pàgina d'inici de la web. Aquesta secció serà l'encarregada de posar a l'usuari en el context sota el que s'ha desenvolupat aquest projecte, referenciar que es tracta d'un projecte de final de carrera i les condicions de la llicència que li he aplicat.

La secció *Visualització* és on hi haurà l'aplicació. Allà també especifico que l'usuari haurà d'esperar uns minuts per a que carregui la visualització, i algunes indicacions en cas que no es carregui.

A *Codi*, l'usuari tindrà accés a veure els arxius de Processing que composen l'aplicació i que he desenvolupat en l'apartat "arquitectura del programa".

En l'apartat d'Agraïments vull ressaltar que és una aplicació que, tot i estar desenvolupada per mi, hi ha molta gent que m'ha ajudat de manera directa o indirecta.

Per últim, posaré a disposició de l'usuari una secció per si vol posar-se en contacte amb mi per qualsevol qüestió o problema que pugui tenir amb l'aplicació.

Pel que fa al disseny, he volgut que la plataforma integri el millor possible la aplicació. L'he dissenyat a partir de no carregar gens l'estructura i que sigui el màxim d'accessible i usable per l'usuari. La mida de la web és una mica més ample –1110píxels– de l'estipulat com a recomanable (900píxels aproximadament). Això és perquè per no carregar massa visualment l'aplicació l'he hagut de fer més ample. Sent una aplicació que requerirà Java d'entrada no serà compatible amb dispositius mòbils o tabletes. Partint d'això, la plataforma està pensada per obrir-se només en ordinadors.

La pàgina comença amb una capçalera sobre un fons quasi blanc que parteix de la proposta del plafó creat. Per tant l'usuari veurà la fotografia dels *indignats* aixecant les mans, construïda a partir de les imatges de perfil dels usuaris de Twitter utilitzats a l'aplicació, amb una petita franja groga amb el nom del portal web a dins.

Tot seguit hi ha el menú representat de manera molt net i clar amb cada una de les seccions esmentades anteriorment. Posteriorment amb un petit degradat creo l'efecte que la part on hi haurà el contingut estigui un pel ressaltat. D'aquesta manera emfatitzo més el contingut que allà hi haurà, centrant la mirada de l'usuari cap a aquella zona.

La zona per als continguts, per tant, queda ressaltada amb aquest degradat tant per dalt com per baix. Tot i que la capçalera delimita implícitament el marges laterals, aquests dos elements que degraden horitzontalment s'expandeixen per totes dues bandes fins arribar al límit de la finestra del navegador. D'aquesta manera no tanco l'espai sinó que el deixo obert i li dono a l'usuari més sensació d'amplitud.

L'aplicació ha estat fàcil incorporar-la a la web ja que Processing, quan s'exporta crea els arxius necessaris i el corresponent *.html* preparat per publicar *online*. A partir del disseny proposat, he incorporat els element que hi faltaven al document proporcionat pel programa.

Per últim he incorporat un peu amb el text "fet amb AMOR des d'Amsterdam" i la llicència esmentada anteriorment. L'objectiu d'aquest text és informar a l'usuari que aquest projecte ha estat desenvolupat fora del país, però amb la meva millor intenció. És tracta d'un gest perquè apropi l'usuari al projecte i no el vegi com un producte comercial sinó com una aportació altruista sota el marc d'un projecte acadèmic.

[Conclusions finals]

El desenvolupament de la part pràctica d'aquest projecte ha estat molt enriquidor tant a nivell personal com intel·lectual. He pogut adquirir una percepció més completa i profunda d'aquest moviment i això ho puc compartir a través de l'aplicació que he programat. En el moment que vaig plantejar aquest projecte, sabia que el Moviment #15M estava sent un moviment trencador i innovador per les eines i canals que estaven utilitzant. No només tracten de promoure un canvi, sinó que a més, ho fan aportant un nou model ètic i social basat en la cooperació i la cohesió social.

Durant aquest procés he recopilat els missatges que contenen els *hashtags* *#tomalacalle* (868 *tweets*) i *#acampadabcn* (2.324 *tweets*). Però addicionalment he realitzat també la cerca pels mots *#democraciarealya* (3.300 *tweets*), *#nolesvotes* (2.731 *tweets*), i *#spanishrevolution* (8.845 *tweets*). En total he pogut recopilar 14.351 *tweets* tot i que finalment només he utilitzat 3.192 missatges d'aquesta base de dades.

El motiu pel qual no he utilitzat tots els missatges adquirits és perquè l'aplicació no ha suportat tantes dades fent que la seva navegació s'alenteixi molt. El problema d'aquest alentiment no és a l'hora de representar els *tweets* en la pantalla, sinó en el moment de detectar si el ratolí es troba a sobre d'un d'ells. Aquesta acció li resulta molt costosa ja que per comprovar-ho ha de consultar tots els *tweets* fins que troba un que coincideix amb la posició del ratolí.

Abans de definir totalment com seria la visualització, vaig fer algunes proves treballant sobretot amb el text. En una primera aproximació vaig treballar entorn una narració lineal, on el textos apareixien automàticament com si es tractés d'una lectura assistida. Aquests textos tenien una mida de lletra en proporció al número de *retweets* que tenien. Aquesta primera prova era molt potent a nivell de text, però es tractava més d'una animació que no pas d'una visualització, ja que no hi havia diferència en llegir aquests textos a Twitter directament –vaig calcular que un usuari mirant la pantalla i llegint tots els missatges trigaria més de 12 hores.

Volia una lectura més ràpida i dinàmica. L'usuari en la prova anterior esdevenia espectador i per tant no prenia protagonisme en cap cas. És per això que vaig decidir amagar aquest text utilitzant un símbol en el seu per tal de poder mostrar el màxim de dades possibles en un espai

reduït. D'aquesta manera aconseguiria una lectura no-lineal, i podria involucrar a l'usuari en el projecte donant importància a la seva acció.

Després d'aquest període de proves, també he pogut comprovar que Processing té algunes carències per a mi importants. Per una banda, he trobat a faltar mètodes que ajudin a manipular millor els objectes creats a partir d'instàncies. M'he trobat que podia cridar un objecte en concret, però no detectar-lo ni eliminar-lo, sense recorre tots els objectes i comparar-los, o alterar l'espai on es troba.

Per altra banda, també he trobat que Processing dona pocs recursos per treballar amb text. Existeixen llibreries que ofereixen algunes funcionalitats extres amb text però tot i així, sota el meu parer, falten alguns mètodes bàsics com per exemple, per ressaltar textos amb negreta, cursiva o color.

Altres llenguatges com *ActionScript* de Flash, no tenen aquestes carències però són més ineficients en altres aspectes i també acostumen a donar més problemes. També entenc que no sóc programadora professional i, per tant, les aplicacions no estan programades de la manera més eficient possible.

Hi ha alguns aspectes sobre l'aplicació que no he pogut resoldre o que no funcionen del tot correcte. En primer lloc la visualització, quan mostra tots els *tweets* a través de les dues imatges simbòliques, en alguns casos al passar el ratolí per una zona on *a priori* no hi ha cap *tweet*, fa un *Roll Over* i mostra una imatge de perfil i un missatge. Tot i repassar el codi, no he estat capaç de solucionar això.

L'altre aspecte a solucionar és amb relació amb l'àudio. En algunes ocasions, després de tancar l'aplicació sembla ser que Java no es tanca del tot o es bloqueja i si posteriorment es vol escoltat àudio per un altre canal (pàgina web, *iTunes*, o altres aplicacions) l'àudio no funciona. He buscat i preguntat sobre aquest error, però sense èxit en les respostes. De moment només he descobert que es pot "solucionar" connectant i desconnectant un auricular de l'ordinador. Això em fa pensar que potser hi ha un problema amb Java quan requereix utilitzar recursos del sistema i que després no els hi torna.

En referència a la plataforma *online*, de moment tant l'aplicació com el portal estaran en català. Com els textos que mostro provenen de l'acampada que es va realitzar a Barcelona, tot i haver molts missatges en castellà, el seu context es troba a Catalunya i és per això que es tractarà primer d'un portal *Beta* en català a l'espera de poder seguir desenvolupant aquest projecte en altres aspectes, entre ells l'idioma castellà.

Per poder veure la visualització *online* també m'he trobat amb alguns problemes. Principalment relacionats amb Java. Per una banda, si l'usuari no disposa de Java i/o del *plug-in* pel navegador i requereix de la seva instal·lació. Aquest cas cada vegada és menys freqüent i, a més, tant el sistema com els navegadors assisteixen bastant bé als usuaris en aquest procés.

El principal problema l'he trobat en el sistema operatiu MacOsX concretament en les últimes versions –a partir de la versió 10.7. *Apple* va introduir un canvi en el comportament de Java que quan detecta que no es fa servir aquest complement, aquest s'inactiva. Això ocasiona que quan l'usuari accedeix a la web del meu projecte, observi el text "módulo inactivo". Per solucionar això, l'usuari ha de fer clic sobre aquest text, i fer clic a "activar" en el missatge que li apareixerà. Posteriorment haurà de reiniciar el navegador i d'aquesta manera podrà veure l'aplicació.

Certament això és un problema greu, doncs penso que l'usuari pot fer l'esforç d'instal·lar el *plug-in* si ho requereix el navegador i això només serà un cop i després ja no caldrà més; però activar Java cada vegada que es vulgui veure, em sembla demanar massa a l'usuari. De moment això és una qüestió d'*Apple* i jo no he trobat una solució a això, més que indicar a la web qué ha de fer si es troba l'usuari amb aquest problema i agrair-li l'esforç.

Tot i això, l'aplicació resultant compleix amb els objectius marcats: he aconseguit construir una base de dades amb els missatges d'aquest moviment; he estudiat diverses tècniques de visualització de dades que podien adequar-se amb les característiques del projecte; i finalment he programat una aplicació –que es troba en una plataforma *online*– que mostra els *tweets* del #15M ordenats cronològicament creant un relat a partir de tots aquest missatges on l'usuari és qui descobreix cada un dels textos a partir de la seva acció.

Aquest projecte no només tracta de solucionar un problema de lectura dels missatges de Twitter fent servir el Moviment #15M com a cas d'ús. En tot moment he buscat aquell

compromís social amb l'usuari –i amb mi mateixa– per tal de transportar-lo a aquell període i fer-lo reflexionar entorn el que ha suposat aquest moviment i la seva força a través de la xarxa social. M'ha motivat molt observar com moltes altres persones, que com jo, estan desenvolupant –o han desenvolupat– diferents projectes que intenten també aportar algun aspecte que ajudi a explicar i recordar millor aquest moviment.

Tot plegat ha estat una experiència amb un resultat molt gratificant però que no acaba aquí, tot el contrari. Aquest projecte és un punt de partida per seguir millorant l'aplicació i arribar a l'objectiu real: narrar la història d'aquest moviment a mesura que es va construint. Per suposat aquest és un objectiu que m'he marcat *a posteriori*. Vull seguir treballant en aquest projecte, millorar-lo i aconseguir intensificar encara més la interacció i l'experiència de l'usuari entorn aquest esdeveniment.

[Bibliografia]

Llibres

Fry, B. (2007). *Visualizing Data.* Sebastopol: O'Reilly.

Noble, J. (2009). *Programming Interactivity.* Sebastopol: O'Reilly Media.

Yan, N. (2011). *Visualize This: The FlowingData Guide to Design, Visualization, and Statistics.* Indianapolis: Editorial Wiley.

Documents electrònics i portals web

AndreaMosaic. (2004). Accès el 30 d'agost de 2012, des de AndreaMosaic Home Page: http://www.andreaplanet.com/andreamosaic

Get Twitter Portraits. (2012). Recuperado el 09 de setembre de 2012, de Github: https://github.com/albertmeronyo/get-twitter-portraits

Julien Lagarde's Photostream. (2008). Accès el 30 d'agost de 2012, des de Flickr: http://flic.kr/p/9MhsHs

Mi web. (2011). Accès el 27 de juny de 2012, des de Grupo 15M - No nos vamos: http://www.mi-web.org/miembros/5-eric/textos/61260-listado-completo-de-acampadas-15-m-en-espana

Minim. (2007). Accès el 27 de juliol de 2012, des de Code Compartmental: http://code.compartmental.net/tools/minim

Processing Forum. (2010). Accès el 29 d'agost de 2012, des de Processing Forum: http://forum.processing.org

Processing GUI, ControlP5 . (2007). Accès el 25 de juliol de 2012, des de Sojamo: http://www.sojamo.de/libraries/controlP5/

Reconocimiento-CompartirIgual 3.0 Unported. (2010). Accès el 01 de setembre de 2012, des de Creative Commons: http://creativecommons.org/licenses/by-sa/3.0/deed.es_ES

SQLibrary. (2008). Accès el 16 de juliol de 2012, des de Bezier: http://bezier.de/processing/libs/sql

Topsy - Instant social insight. (2003). Accès el 13 de juny de 2012, des de Topsy: http://topsy.com

Topsy Tweet Retriever. (2012). Recuperado el 09 de setembre de 2012, de Github: https://github.com/albertmeronyo/topsy-tweet-retriever

Twitter API. (2010). Accès el 02 de juliol de 2012, des de Twitter API: http://www.dev.twitter.com