Instal·lació multimèdia interactiva activada per control de moviment corporal

Victor Cruz Altamirano

Resum—L'àmbit de la comunicació entre els essers humans ha evolucionat i segueix patint canvis de diverses maneres a passos agegantats. L'aparició de les noves tecnologies ha facilitat aquest procés, i en aquest sentit volem anar més enllà amb la investigació i la composició de formes poc tradicionals d'interacció amb altres persones. Aquest projecte consisteix en la creació d'una instal·lació multimèdia que, mitjançant la captació dels moviments de diferents parts del nostre cos, pugui donar com a resultat la creació de efectes visuals i lluminosos. D'aquesta manera seran rebuts per un interlocutor, el qual podrà fer infinites interpretacions del que veu.
Per dur a terme aquest projecte, crearem una aplicació web que oferirà una interactivitat a l'usuari més enllà d'una experiència tradicional de comunicació.

Paraules clau—Detecció, javascript, opencv, js-objectdetect, haar cascade, pedestrian detection, interacció.

Abstract— The scope of communication between humans has evolved and continues to evolve in different ways. The emergence of new technologies has facilitated this process, in this sense we want to step aside and investigate untraditional forms of interaction with other people. This project is therefore to create a multimedia installation by capturing the movements of different parts of our body can result in the creation of visual and lighting effects that will be received by a partner, which may make infinite interpretations what he sees.
To carry out this project we will create a web application that the user will experience beyond traditional communication experience.

Index Terms— Detection javascript, opencv, js-objectdetect, haar cascade, pedestrian detection.

1 Introducció

La comunicació entre els essers humans mai ha estat limitada a un cert tipus d'interacció. Davant les limitacions físiques o del medi, sempre hem pogut fer-nos sentir.

Hem après a treure profit de les noves tecnologies per a que la distancia tampoc fos un impediment a l'hora de relacionar-nos. Però, no obstant, en la majoria dels casos seguim replicant les formes tradicionals, es a dir, una comunicació audiovisual objectiva que ens ofereix una única interpretació d'aquella informació que ens arriba.

En multitud de situacions, les nostres emocions no son visibles, o bé no volem que ho siguin. I es per aquest motiu que ens limitem nomes en l'intercanvi informació, i com a conseqüència, no mirem més enllà.

- *E-mail de contacte: victorcruzalt@gmail.com*
- *Menció realitzada: Enginyeria del Software.*
- *Treball tutoritzat per: Fernando Vilariño (Computació)*
- *Curs 2016/7*

Però, i si poguéssim transmetre totes aquestes paraules i emocions com a elements visuals cap al nostre interlocutor?.

Davant aquesta nova situació s'obren infinites possibilitats en les que dues o més persones fan diverses interpretacions del que veuen.

Ens trobem llavors, en un terreny poc explorat per part de les noves tecnologies, com es la comunicació artística. No només ens limitaríem a la part estètica, també volem produir diferents sentiments i sensacions en el nostre receptor, que aquest no es quedi indiferent davant el que veu, i que fins i tot, es pugui arribar a commoure.

El propòsit d'aquest document és el d'aconseguir aportar un interès general pel projecte mitjançant els resultats obtinguts.

S'exposaran els trets més rellevants del projecte. A partir d'aquesta introducció explicarem els objectius del treball, les raons per la qual es dur a terme la recerca del present projecte i la problemàtica que volem resoldre.

També, contindrà un apartat per indicar la contribució que pretén aportar el treball realitzat, la metodologia emprada i per últim, les conclusions finals de la recerca realitzada durant els darrers mesos.

2 Objectius

Per tal de dur a terme tot aquest procés d'intercanvi d'informació "visual", es imprescindible treballar mitjançant les eines pertinents per tal de processar aquesta informació, a més d'elements físics per a la visualització de les respostes rebudes per al nostre interlocutor.

L'objectiu final seria el de crear una instal·lació multimèdia interactiva basada en el moviment corporal que sigui capaç de transformar els nostres moviments en efectes visuals i/o lluminosos, amb l'objectiu de mantenir una comunicació bassada en la interpretació visual.

Els principals objectius d'aquest treball de fi de grau serien:

- Trobar les tecnologies necessàries per a dur a terme el projecte, documentar-se i familiaritzar-se amb les mateixes.
- Detectar per software les diferents part del cos, utilitzant com a hardware d'entrada una càmera web.
- Detectar els múltiples objectes per tal de poder establir posteriorment certa interacció.
- Tractar de manera digitalitzada els moviments, tenint en compte esdeveniments preestablerts en cas d'interacció.
- Generar efectes visuals o sonors en cas de que es produeixi interaccions.
- Establir una connexió amb un interlocutor, facilitant la consecució d'una comunicació visual a partir dels efectes creats per tots dos.

3 Estat de l'art

El procés d'investigació i documentació ens a portat a conèixer un gran nombre d'eines. S'han tingut en compte els següents tipus d'eines:

- Captura i detecció d'objectes
- Garantir una interacció amigable amb l'usuari a través d'una interfície familiar.
- Generació d'efectes visuals utilitzant com a dades d'entrada els nostres moviments.
- Comunicació amb altres persones utilitzant Internet.

3.1 Aplicació web

OpenCV es una llibreria que ens proporciona instruments per a la detecció d'objectes, fent servir c++ com a llenguatge natiu. Les primeres proves d'aquesta llibreria no eren res amigables per l'usuari, i la implementació d'una interfície gràfica no seria senzill.

Aquesta llibreria té diverses implementacions en JavaScript, que ens aporten un escenari ideal i més familiar: la utilització d'un navegador web per a la execució del nostre software.

3.2 Llibreria: js-objectdetect

Llibreria JavaScript escollida pel seu alt grau de deteccions, basat en el treball de Paul Viola and Rainer Lienhart.

3.3 Llibreria: JQuery

Llibreria JavaScript que ens permet controlar i manipular elements del DOM, fer animacions, controlar events i amb moltes més possibilitats.

La fusió de totes aquestes eines ens proporcionarà tot allò necessari per al desenvolupament del projecte.

4 Metodologia

A l'hora de fer el plantejament real del que es volia i del que consideràvem que es podria arribar a fer, ens varen reunir tant el tutor del projecte Fernando Vilariño, com l'artista Dan Norton, col·laborador habitual al Centre de Visió per Computador.

Per a la planificació s'han marcat una sèrie de fites setmanals clarament exposades en un diagrama de Gantt, on fèiem una valoració aproximada de el temps de dedicació que tindria cada part del projecte.

També, s'han realitzat reunions continues per part del tutor del projecte i reunions per videoconferència amb Dan Norton, on s'exposaven els últims avenços o possibles dubtes que sortien durant la realització del treball.

4.1 Planificació

Les fites programades a l'inici del projecte es poden veure en la següent taula, juntament amb el seu diagrama de Gantt.

	Tarea	Inicio	Duracion	Final
Definició del projecte	1	14/9/16	16	30/9/16
Lliurament Informe Inicial TFG	2	28/9/16	5	2/10/16
Estudi llibreria OpenCV	3	3/10/16	4	7/10/16
Primer software funcional	4	8/10/16	7	15/10/16
Detecció d'una persona en 2D	5	16/10/16	14	30/10/16
Lliurament Informe de Progrés I	6	31/10/16	6	6/11/16
Detecció de més d'una persona en 2D	7	7/11/16	10	17/11/16
Tractament digital dels moviments	8	18/11/16	23	11/12/16
Lliurament Informe de Progrés II	9	12/12/16	6	18/12/16
Interacció inter-site	10	19/12/16	27	15/1/17
Test unitari del software	11	8/10/16	110	15/1/17
Lliurament Proposta Informe Final	12	16/1/17	6	22/1/17
Lliurament Proposta presentació	13	1/2/17	4	5/2/17
Lliurament Finals al tutor	14	5/2/17	2	7/2/17
Lliurament Poster	15	10/2/17	2	12/2/17
Defensa TFG	16	10/2/17	7	17/2/17

Fig.1: Planificació

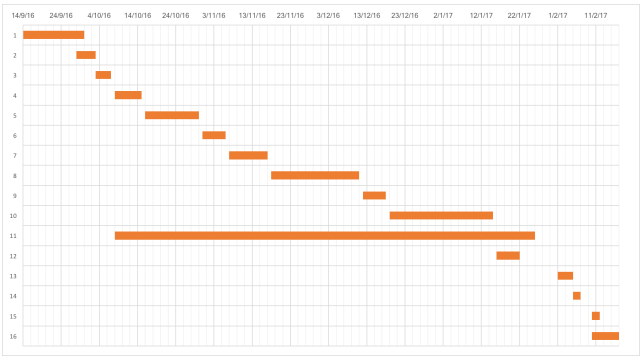

Fig.2: Diagrama de Gantt

4.2 Procés d'investigació

Una vegada establerts els objectius, el primer que calia fer era decidir quins serien els instruments que utilitzaríem per a la implementació del nostre sistema. A partir de la recomanació per part del tutor vam decidir que per a la detecció de moviments en dues dimensions treballaríem amb OpenCV (Open Source Computer Vision), que és una llibreria de codi obert de visió per computador i machine learning, que ens ajudarà a donar les primeres passes amb tot allò relacionat amb la detecció de cossos/objectes davant d'una càmera.

Aquesta eina disposa d'una extensa documentació i ha sigut adaptada a gran quantitat de plataformes. Originàriament va ser desenvolupada per a c++, per la qual cosa vam començar un primer procés d'aprenentatge en aquest llenguatge.

Finalment, i per idoneïtat en el nostre projecte, vam escollir una plataforma web, on tot el procés de gestió i càlcul dels algoritmes de tracking i detecció es desenvoluparien en JavaScript, amb html 5 com a base per a la presentació d'una interfície amigable al usuari sense necessitat de cap software específic, només el navegador.

Dins les diferents implementacions de OpenCV per a JavaScript, vam trobar tres candidats a utilitzar:

- tracking.js
- jsfeat
- js-objectdetect

Per la seva major fiabilitat en les captacions, bona velocitat de refresc (FPS) i una més extensa implementació de detecció d'objectes, ens vam decantar per js-objectdetect.

4.2 Js-objectdetect

Js-objectdetect es una llibreria JavaScript per a la detecció d'objectes en temps real.

Aquesta llibreria es basa en l'obra de Paul Viola i Rainer Lienhart i es compatible amb classificadors en cascada (Haar Cascade) utilitzats pel detector d'objectes OpenCV.

Els descriptors Wavwlet de Haar permeten definir de manera robusta classes d'objectes complexes, essent invariants a canvis de color i textura. S'utilitzen habitualment per a la descripció de persones. Presenten la possibilitat de codificar característiques, com poden ser canvis d'intensitats a diferents escales.

Aquest algoritme consisteix en recórrer la imatge utilitzant una petita finestra amb la possibilitat de trobar l'objecte desitjat (un rostre per exemple), en cas de no haver captat res a la primera passada, augmenten la mida de la finestra per si l'objecte es més gran que aquesta.

Podríem distingir tres aspectes fonamentals d'aquest algorisme:

- Utilització de trets de classificació
- Utilització d'una imatge integral
- Organització en cascada dels classificadors

A la figura 3 es mostren els tipus de formes utilitzades per l'algoritme per a detectar possibles trets d'un objecte. En el cas d'un rostre podem tenir una forma amb tres rectangles, a on el del mig serà el més clar i els dos dels costats seran més foscos, representant la zona del nas i els dos ulls, per exemple.

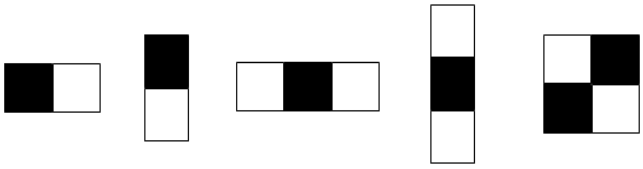

Fig.3: Trets per a la detecció d'objectes

Aquests trets poden ser nombrosos, per això necessitem un algoritme d'entrenament (AdaBoost) que ens aporti els millors trets de cada objecte a identificar. A partir d'aquest trets podrem calcular l'anomenat valor d'un tret, que s'obté de la posició dels vèrtex dels rectangles que els conformen. Tots aquest valors formen part d'un classificador.

Els classificadors els organitzarem en cascada, on cada finestra de la imatge analitzada passarà per diferents classificadors, fins que trobi algun que serà l'objecte que volem detectar. Aquest procés es pot veure simplificat a la figura 4.

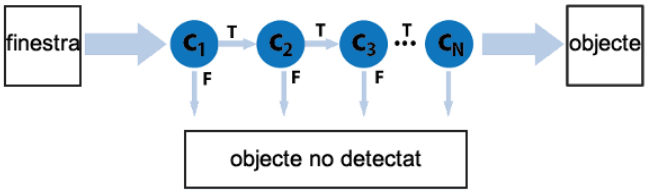

Fig.4: Distribució d'un classificador en cascada

En resum, aquest procés consisteix en recórrer la imatge amb una finestra a la que se li apliquen diversos classificadors en sèrie, cadascun d'ells més complex que l'anterior, els quals utilitzen les característiques per confirmar o descartar la hipòtesis de que es tracta de l'objecte buscat. Si la hipòtesis es rebutja en qualsevol nivell, el procés no continua, però si es confirmen tots els filtres significarà que s'ha detectat l'objecte desitjat. Els patrons es consideren girats en diversos possibles angles. A més, l'algoritme pot executar-se a diverses escales per obtenir objectes de diferents mides o de mida desconeguda.

5 ARQUITECTURA DEL SISTEMA

Passem a descriure l'arquitectura software actual del sistema, es a dir el de una aplicació web.

El primer que fa el sistema és demanar permís per accedir a la càmera web (multi_face_detection.js) i a partir d'aquí crearem un objecte detector (mètode pertanyent a objectdetect.js) que necessitarà de les dimensions del vídeo, així com un classificador que dependrà de l'objecte que volem detectar. En el cas de voler detectar la cara, li passarem l'arxiu objectdetect.frontalface_alt.js.

L'objecte detector processarà totes aquestes dades per retornar les coordenades, l'amplada i l'alçada de l'objecte trobat. Amb aquestes dades podem dibuixar en primer terme el rectangle que posicionarà l'objecte que ha detectat dins un element HTML anomenat canvas, que ens permet la generació de gràfics dinàmics.

Hem utilitzat el Model Vista Controlador (MVC), que és un patró d'arquitectura del software que separa les dades d'una aplicació, la interfície de l'usuari i la lògica de control en tres components. A la figura 5, podem veure exactament el funcionament d'aquest patró (en el cas de voler detectar múltiples cares), tenim com a controlador multi_face_detection, que rep els inputs que envia l'usuari, per després demanar-li dades al model, que serà objectdetect.js. Per últim, envia els resultats a index.html que s'encarrega de mostrar-los com nosaltres ho haguem determinat.

Aquest funcionament ens permet no barrejar codi, tot esta separat per a una millor gestió dels errors i reutilització posterior que es pugui fer del codi.

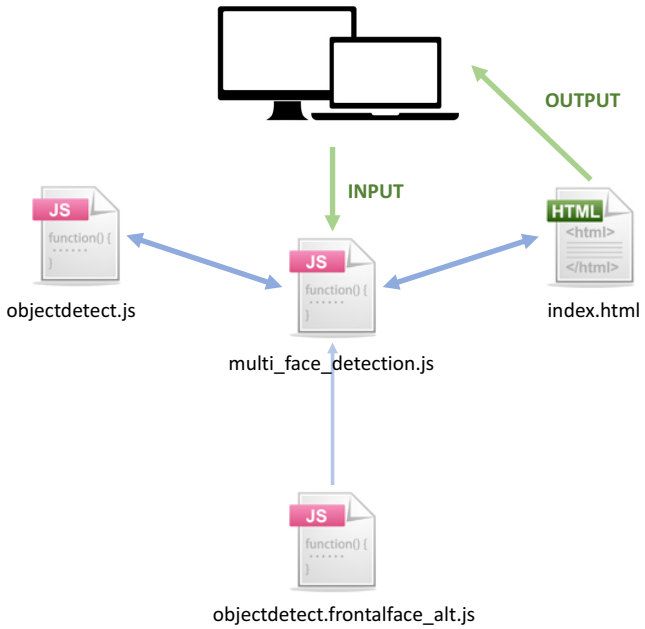

Fig.5: Arquitectura del sistema

A nivell de hardware, de moment només cal disposar d'un ordinador amb càmera web.

També, hem de definir el flux d'interacció de l'usuari amb l'aplicació web, i amb aquesta premissa podem començar a dissenyar la interfície per on es desplaçarà i interactuarà. A més de poder limitar, si cal, les possibilitats que es puguin tenir en un principi.

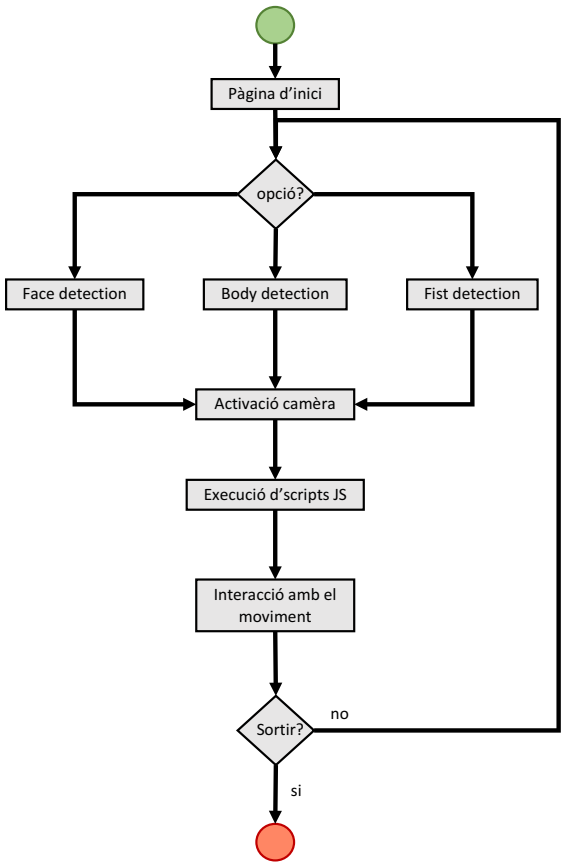

Fig.6: Flux de control

5.1 Events i efectes

Per tal de tenir un projecte modular, que ens facilitarà la seva reutilització, necessitarem separar les accions que puguin generar un efecte, del conjunt total d'efectes.

D'aquesta manera, podrien aplicar qualsevol efecte a diferents accions que puguem realitzar.

La implementació d'aquesta diferenciació ens ha permès afegir un grau de llegibilitat, que com a conseqüència li ha donat també certa complexitat al nostre codi.

5.2 Events

Els events son totes aquelles accions generades per la interacció de l'usuari amb el software.

Actualment disposem dels següents events:

- **Proximitat dels objectes.** A partir de l'amplada i llargada dels objectes detectats podem fer esti-

macions de la proximitat que hi ha entre la càmera i l'objecte.

- **Detecció de col·lisions.** Es guarden contínuament les ultimes dades de posició dels objectes segons un identificador diferent per a cadascun. A partir d'aquí podem comparar les posicions entre objectes i considerar una col·lisió en els casos on hi hagi encreuament de coordenades.
- **Detecció de salts.** Tenint en compte els canvis sobtats en la posició "y" dels objectes, podrem detectar els salts.

La codificació i programació d'aquests events es creen i es guarden dins d'un arxiu independent.

5.3 Efectes visuals

Els efectes son les diferents animacions visuals creades en JavaScript que es visualitzaran després de l'activació feta per algun event.

Aquests efectes s'han modificat fent servir la llibreria JQuery, que ens proporciona diverses dreceres per tal d'evitar tasques de codificació repetitives o de certa complexitat. Alguns exemples dels efectes utilitzats fins ara son els següents:

- **Explosió de partícules:** Creat per Aleksei Andreev, genera cercles de colors que s'escampen per la pàgina, activat per un clic de ratolí. En aquest cas s'ha modificat per a que comenci quan es detecta una col·lisió entre objectes.

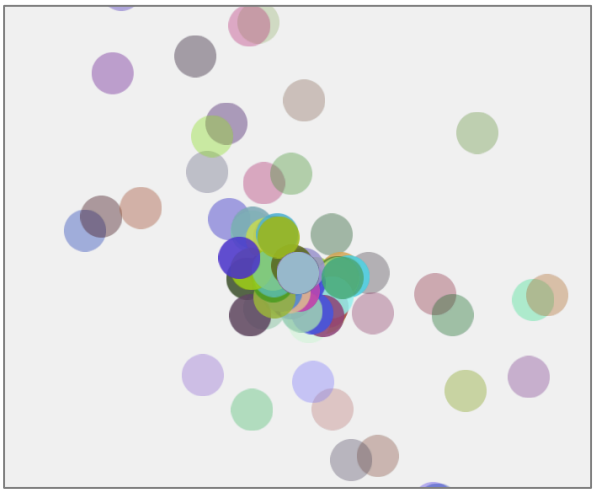

Fig.7: Explosió de boles

- **Partícules que orbiten al voltant d'un objecte:** Creat per Hakim El Hattab, produeix partícules que "orbiten" al voltant d'un centre determinat. En el nostre cas, alguna part del cos, seguint-lo i deixant una estela suau pel camí. També redueix o augmenta el seu radi depenent de la proximitat de l'objecte a la càmera. Es genera per a fins a dos objectes a la vegada.

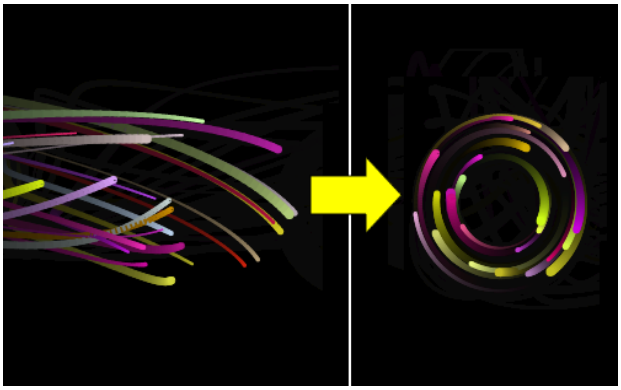

Fig.8: Partícules orbitant

- **Interacció entre partícules:** Creat per Elton Kamami, genera partícules que interaccionen si son del mateix color i es troben a prop. Pel que fa al nostre projecte, augmenten el nombre de partícules generades segons la nostra proximitat a la càmera web.

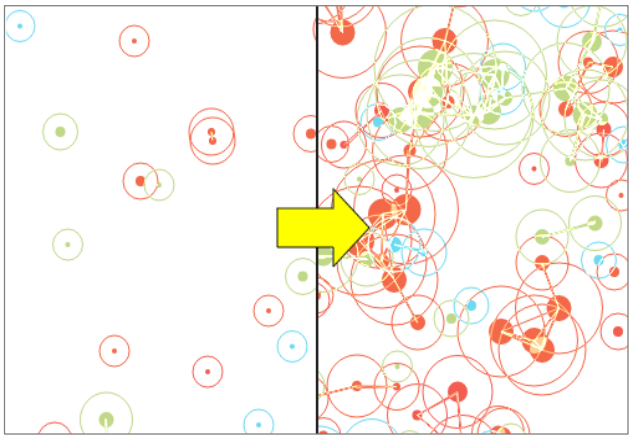

Fig.9: Interacció entre partícules

6 Detecció múltiple d'objectes

Fins ara, havíem treballat amb la detecció d'un sol objecte, però necessitem capturar interacció amb altres cossos. Es per aquest motiu, que a partir d'aquí s'ha implementat un mètode que es capaç de detectar múltiples objectes gracies a l'assignació a cadascun d'ells d'un identificador. Permeten així, poder diferenciar-los davant la gran quantitat de dades generades per la captura, ja que hem de tenir en compte que s'analitza cada fotograma del vídeo en stream que li arriba de la càmera.

El mètode detector de la llibreria js-objectdetect s'encarrega de guardar en un array d'objectes tot el que detecta, a partir d'aquí hem de processar aquestes dades.

L'array *detected_objects* conté totes les dades captura-

des, però per saber quants objectes ha detectat en aquell moment, simplement necessitaren la longitud d'aquest array. Per exemple, si identifica dos objectes `detected_objects.length = 2`. Hem de tenir en compte que aquest array s'actualitza a cada frame que li entra per la càmera, i en cas de voler tenir un històric, hauríem de crear un altre array on hi ficaríem aquestes dades.

Amb les coordenades proporcionades per `detected_objects` podem dibuixar un rectangle a sobre de cada objecte. En aquest cas no hem classificat les dades que ens arriba, es a dir, no diferencien un objecte d'un altre, tots els objectes detectats i dibuixats els tractarà per igual.

Ens interessa llavors diferenciar objectes, per això crearem un array d'objectes amb la següent estructura:

```
objects[i] = {
      x: x,
      y: y,
      w: width,
      id: i
};
```

Com veiem, ara podrem guardar la posició x, y, l'amplada i un identificador que vindrà determinat per la posició a l'array `detected_objects`.

Hem de tenir en compte que aquest nou array esta contínuament afegint dades i caldria tenir un array només amb les dades dels objectes actuals. Per tal de solucionarho, simplement cal afegir la següent línia:

```
objects.splice(objects.length-1,objects.length-
nobjects);
```

El que fem es esborrar totes les dades anteriors i ens quedem amb les actuals.

Ara ja som capaços de comparar les dades dels objectes, i de dibuixar rectangles diferents segons convingui, simplement assignant un color diferent que vindria determinat pel seu identificador.

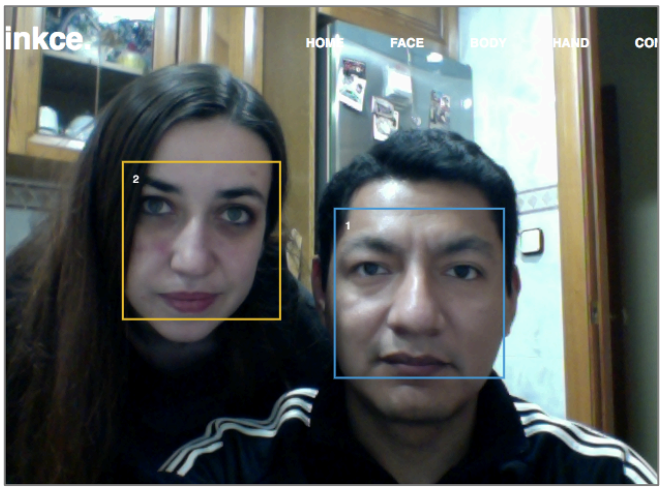

Fig.10: Detecció múltiple

7 INTERACCIONS I EFECTES VISUALS

Una vegada ja hem sigut capaços de gestionar, organitzar les dades transmeses i de treure la informació bàsica dels objectes detectats, estem preparats per començar a treballar amb la part més artística d'aquest projecte, es a dir centrar-nos en la captura d'events per tal de generar efectes en la pantalla, en aquest cas.

La prioritat d'aquesta part era la interacció entre persones. Necessitàvem doncs, poder detectar a més d'una persona per la càmera. Aconseguit aquest objectiu, ja podríem començar a pensar en totes les possibilitats que això ens podria permetre.

Quin tipus d'efectes podríem mostrar i a conseqüència de quines accions es generarien?. Amb l'ajuda del Dan vaig poder començar a treballar en aquest procés, ja que va aportar gran quantitats d'idees a desenvolupar que les podríem separar en tres parts:

- Interacció amb una resposta física i visual
- Interacció amb una resposta visual
- Interacció amb una resposta sonora

El primer punt serien els efectes generats a coses físiques, com pot ser un panel de LEDS, a on aquests es mostrarien per la nostra interacció a la webcam.

El segon son els efectes que mostraríem per la pantalla i generaríem des del navegador web, tot desenvolupat amb JavaScript.

I per últim, la generació de pulsacions binaurals que ens permet crear il·lusions auditives, utilitzant diferents freqüències a cada orella.

Com que la base del projecte es la web, s'ha decidit començar amb la generació d'efectes amb JavaScript.

8 INTERACCIÓ AMB RESPOSTA VISUAL

Començarem explicant les idees principals aportades per Dan:

- Generació progressiva d'un color de fons variant, o depenent de la nostra proximitat.
- Assignació de colors diferents per a cada persona.
- Aparició d'un rastre segons l'objecte, en aquest cas, quan nosaltres ens movem.
- Moviments en els eixos x, y, z, amb això modificaríem diferents objectes, com la llum.
- Creació d'efectes progressius. Aquest efectes es generaran després d'una seqüencia de moviments previs, com ara saltar tres cops per a crear un espècie d'explosió de colors.
- L'explosió anterior tindria conseqüències, com reassignació de colors als cossos.
- Col·lisions entre cossos, per a generar diferents efectes.

Abans de procedir a la implementació d'aquestes idees, ens hem d'assegurar de la viabilitat i la forma de presentar-se davant l'usuari, a on també haurem d'investigar les nostres idees perquè la interfície de l'aplicació pugui ser portada a terme.

Per una millor modularitat del projecte hem de distingir entre les accions que donen pas a un efecte i els efectes en si. Aquesta qüestió es tracta en el punt 5.1 d'aquest document.

Per tal d'obtenir feedback en el desenvolupament de l'aplicació, s'ha pujat el projecte a un servidor web, accessible des de la següent direcció: www.lalinkce.com.

9 RESULTATS

En aquest apartat avaluarem el resultat obtingut del projecte, raonarem sobre les objectius inicials i els canvis que hem hagut de fer.

A la figura 11 podem veure la pàgina inicial actual del projecte, on tenim una barra de navegació per escollir entre els objectes a detectar, la cara o el cos. Per motius de comoditat les proves s'han fet amb la detecció de cares. Treballar amb cossos sencers requeria de espai i teníem dificultats a l'hora de fer captures.

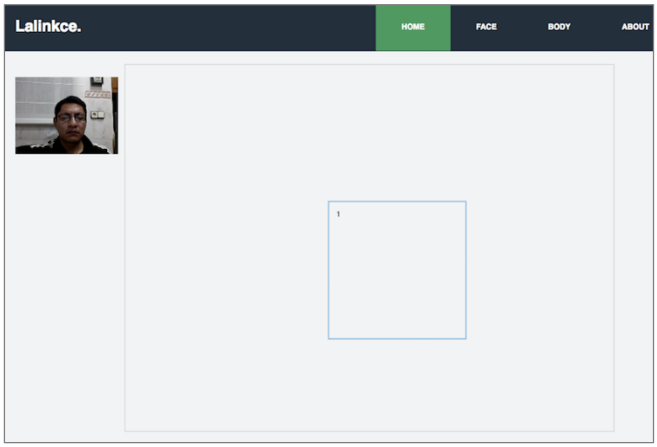

Fig.11: Pàgina inicial

Com podem veure, tenim la sortida de vídeo a la cantonada superior esquerra, per tal de donar el protagonisme al que es produïa dins el canvas.

Per tal de poder veure els efectes s'ha fixat un petit menú desplegable d'opcions (figura 12) per tal de escollirne un.

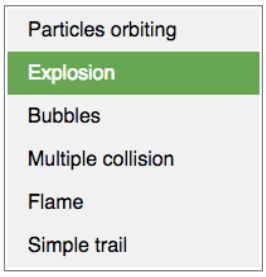

Fig.12: Menú d'efectes

Hem aplicat efectes que no depenen de cap event previ com podem ser els efectes que s'apliquen als objectes detectats, a la figura 13 veiem l'efecte de partícules orbitant sobre dues cares.

Fig.13: Efecte aplicat a dues cares

S'ha desenvolupat dos tipus d'accions que funcionen com a triggers (llançadors) dels efectes, el primer es la col·lisió entre dos objectes, on podem veure un exemple a la figura 14. El segon son els salts, on la intenció es que depenem del nombre de salts que fem provocar un efecte o un altre.

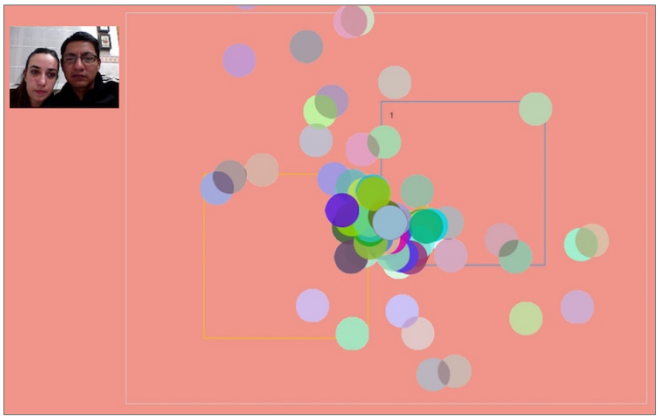

Fig.14: Efecte d'explosió provocat per una col·lisió

9.1 Rendiment de l'aplicació

Tant la detecció d'objectes com la seva posterior classificació han funcionat amb un alt grau de satisfacció, les imatges per segon mostrades eren estables, entorn als 25fps o superior.

Ara bé, durant el procés d'implementació dels primers efectes, van veure els primers problemes de rendiment. La majoria dels efectes preníem el ratolí i les seves coordenades (x, y) com a punt d'interacció. En el moment en que aquestes coordenades eren les dels objectes detectats, el rendiment es veia minvat. Era lògic, ja que les dades rebudes no proveníem d'una única x-y, sinó de desenes de coordenades per segon rebudes i emmagatzemades en un array. Per tal de pal·liar aquest fet, van aïllar només les dades del moment actual, esborrant tot l'anterior. La millora va ser important.

L'últim obstacle en quant l'execució fluida de la aplicació va ser l'execució a l'hora del stream de vídeo i dels efectes, que necessitaven de molta memòria i CPU, com a conseqüència el rendiment en els ordinadors més antics es veien ressentits.

Per tal de quantificar la utilització de recursos per part dels diversos scripts s'ha utilitzat l'eina pertanyent al navegador Google Chrome. Els resultats van ser els següents (figura 15 i 16).

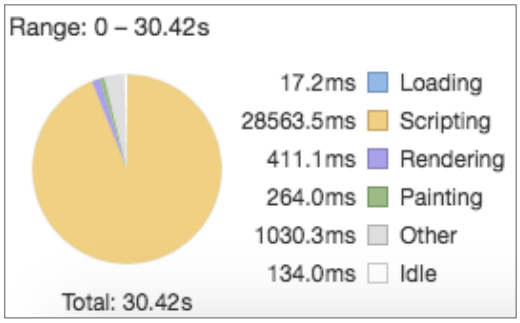

Fig.15: Temps per activitats

Heavy (Bottom Up)

Self Time		Total Time		Function
6794.7ms	67.01%	6795.4ms	67.02%	▶ detect
972.9ms	9.59%	972.9ms	9.59%	▶ fillRect
910.1ms	8.98%	910.1ms	8.98%	(program)
390.1ms	3.85%	390.1ms	3.85%	▶ stroke
372.0ms	3.67%	372.0ms	3.67%	▶ fill

Fig.16: CPU Profiles

A la figura 15 veiem el temps d'execució de les diferents parts de la web, es evident tot el temps que pren el funcionament dels scripts, on a la figura 16 es veu més clarament en quines funcions passem més temps en execució, en aquest cas es el detector d'objectes. En segon lloc fillRect, que s'encarrega d'omplir les imatges generades al canvas.

9.2 Possibles millores i solucions

Els problemes d'us excessiu de recursos sempre es veien agreujats per la utilització del videostream més els efectes. Una solució a aquest problema seria la separació de tots dos, utilitzant un intermediari per a compartir la informació generada, aquest intermediari podria ser una base de dades no relacional com Google Firebase, que ens permet realitzar canvis en temps real, per tal de veure reflexat els canvis generats per nosaltres en el mateix moment de fer-ho.

També, quan tracten la imatge que entra per la càmera, hem de tenir en compte que s'analitza cada fotograma, si poguessin gestionar millor aquest procés podrien seleccionar els fotogrames a analitzar.

10 Conclusions

L'objectiu principal d'aquest projecte, es el de transmetre informació, es a dir comunicar, el que possiblement el fa diferent es el com. En un primer moment no era fàcil veure un objectiu concret, eren molts conceptes i sempre intentava tornar cap a una direcció més tradicional, com podria ser utilitzar la detecció de moviment per fer coses més "normals" que no més senzilles, com per exemple el control remot de qualsevol artefacte.

Poc a poc i després de moltes xerrades amb el meu tutor vaig canviar el meu concepte inicial i eren més evidents les noves possibilitats que podria aportar aquest projecte de cara a una utilització futura en àmbits més artístics.

Com a primera aproximació del objectiu final esmentat abans, la utilització de JavaScript com a eina principal ens permet l'accessibilitat des de qualsevol punt connectat a Internet. Ja de nosaltres depèn de on volen fer la interpretació de les dades recollides, en el meu cas representat en llenguatge web.

Segurament existeixen formes més efectives o efectistes per fer arribar sensacions als altres, però que estigui disponible per qualsevol, des de on vulgui, només ho permet aquest medi, la xarxa.

Finalment, crec que he donat un primer pas en aquest procés de comunicar, però m'hagués agradat poder donar el següent i per tant concloure aquest cicle entre emissor i receptor. No obstant estic satisfet pel resultat obtingut.

Bibliografia

[1] Open Source Computer Vision Library (s.f.). OpenCV. Recuperat al setembre de 2016 de http://www.opencv.org

[2] Tschirsich, M. (2014). Js-objectdetect. Recuperat al novembre de 2016 de http://tinyurl.com/j35gtzt

[3] Equip de tracking.js (2014). Tracking.js. Recuperat al novembre de 2016 de http://www.trackingjs.com

[4] Salsita Software (2014). Javascripting.com. Recuperat al desembre de 2016 de http://www.javascripting.com

[5] W3Schools(1999). W3Schools Online Web Tutorials. Recuperat al setembre de 2016 de http://www.w3schools.com

[6] Ian Sommerville (2012). Ingeniería de software. Recuperat al setembre de 2016.

[7] Viola P. & Jones M. (2001). Robust Real-time Object Detection. Recuperat de http://tinyurl.com/hs69tzc

[8] del Toro Hernández E., Cabrera Sarmiento A. J., Sánchez Solano S. & Cabrera Aldaya, A. (2012). Impacto de la memoria cache en la aceleración de la ejecución de algoritmo de detección de rostros en sistemas empotrados. *Ingeniería Electrónica, Automática y Comunicaciones*, *33*(2), 57-71. Recuperat al desembre de 2016, de http://tinyurl.com/juy9ams

[9] Departament d'Enginyeria de la Informació i de les Comunicacions. Tema 4: Disseny d'aplicacions web. Material no publicat. Recuperat el desembre de 2016 de http://cv.uab.cat

[10] El Hattab H. (2012). Hakim El Hattab. Recuperat al gener de 2017 de http://hakim.se

[11] Kamanis E. (2012). Pixelgrid. Recuperat al gener de 2017 de https://github.com/pixelgrid

[12] Windle J. (2013). Sketch.js. Recuperat al gener de 20017 de https://github.com/soulwire/sketch.js

[13] Aleksei @alek (2014). Aleksei. Recuperat al gener de 2017 de http://codepen.io/alek

[14] Daniel Imms (2014). Growing with the web. Recuperat al desembre de 2016 de http://www.growingwiththeweb.com

[15] Equip de JQuery (2006). JQuery. Recuperat al octubre de 2016 de http://www.jquery.com

[16] Marc Downie (2004). Choreographing The Extended Agent: performance graphics for dance theater. Recuperat al gener de 2017 de https://tinyurl.com/hjun3j8

[17] Hervé Tullet (2016). Un Libro. Recuperat al setembre de 2016.