ILUMINACIÓN INTERACTIVA CON PROTOPIXEL

Piñeiro Viña, Alejandro Javier

Curs 2015-2016

Director: SEBASTIAN MEALLA CINCUEGRANI

GRAU EN ENGINYERIA DE SISTEMES AUDIOVISUALS

Universitat Pompeu Fabra Barcelona
Escola Superior Politècnica

Treball de Fi de Grau

ILUMINACIÓN INTERACTIVA CON PROTOPIXEL

Alejandro Javier Piñeiro Viña

TRABAJO DE FINAL DE GRADO

GRAU EN ENGINYERIA DE SISTEMES AUDIOVISUALS

ESCOLA SUPERIOR POLITÈCNICA UPF

2016

SEBASTIAN MEALLA CINCUENGRANI
Music Technology Group

Universitat Pompeu Fabra Barcelona

AGRADECIMIENTOS

Gracias a Sebastián Mealla por mostrarme y guiarme con Protopixel para mi crecimiento a nivel profesional y creativo; así como la confianza que depositó en mi para llevar a cabo este proyecto y su continuo apoyo durante el proceso. Gracias a Joan Morá por recomendarme a Sebastián como tutor para mi TFG. Gracias a Carles Fernández por su aguda vista y haberme ayudado a resolver todos los problemas técnicos que surgieron durante el trabajo.

Finalmente, gracias a mis padres por su completo apoyo y comprensión durante este largo camino que ha sido el Grado en Ingeniería de Sistemas Audiovisuales; también a mis amigos de la carrera, los cuales siempre me han ayudado, y a Gara por convencerme de que no era necesario agobiarme y que todo saldría bien.

ABSTRACT

Protopixel es un producto software y hardware desarrollado por la empresa homónima en Barcelona, el cual fue lanzado de forma beta para early-adopters a finales de 2015. Este dispositivo físico junto a una aplicación permite el mapeado de diferentes tipos de contenido visual a luminarias, siendo la reproducción de vídeo sobre una determinada tipología de LEDs su principal especialidad.

El propósito de este Trabajo de Fin de Grado es el desarrollo de una nueva funcionalidad para la aplicación Protopixel, la cual permita la generación de contenido lumínico basado en múltiples aspectos analizados de un vídeo.

Mediante el procesado de parámetros extraídos tales como la colorimetría o movimiento de un vídeo, se hace posible la creación de distintos contenidos dentro del mismo programa Protopixel, ya que cuenta con una opción de inclusión y ejecutado de scripts basados en su lenguaje nativo.

El objetivo principal es la creación de un sistema automático de iluminación que acompañe a vídeo, pudiendo tener otras muchas aplicaciones si se desea. El contenido generado en Protopixel consta de varios parámetros modificables que se traducen en diferentes disposiciones de luz en el espacio y en un lenguaje lumínico que ayuda al contenido video-gráfico a expresar más allá de los límites del marco en el que se muestra. La empresa Emex, proveedora de servicios audiovisuales tales como contenido y hardware, permitirá en su showroom la implementación y pruebas de campo, facilitando la focalización del trabajo en unos determinados *use-cases*.

La interacción se realiza entre el analizador de vídeo, con sus múltiples parámetros, y Protopixel, pudiendo ser en tiempo real o como en los casos tratados, previa a la reproducción simultánea de contenidos en el lugar de la instalación.

IN ENGLISH

Protopixel is a software and hardware product developed by a same named company in Barcelona. It was released in a beta way for early-adopters in the lately 2015. This device in addition to an application allows mapping different kind of visual content to luminaries, being the video playback over LED its main feature.

The aim of this Grade's Final Project is to develop a new functionality for the Protopixel application and it would be able to generate luminic content based on many features extracted from video analysis.

Using data as colour or movement information from a video, and data processing, is possible to generate content within Protopixel because of its scripts execution function.

The main goal of the project is the development of an automatic system for lighting-design, which could reinforce a video experience, although it could have many other uses. The content generated in Protopixel will have some parameters that will change the light disposition in the space, helping to create a light-language that allows to any video to express beyond the limits of the frame that contains it. A company named Emotion Experience, which provides audio-visual services like media content and hardware, would permit to use its showroom as the final implementation for the tests, also restricting the develop work to play y some determined use-cases.

The interaction will be played between the video analysis system and Protopixel's light-info generator, whether in real time or not, and the reproduction of both contents will be synced.

ÍNDICE

Página

Lista de figuras vii
1. INTRODUCCIÓN 1
1.1 Prólogo 1
1.2 El Dilema 2
1.3 La Respuesta 3
2. STATE OF THE ART 6
3. MOTIVATION 11
3.1 Definición del proyecto 12
3.1.1 Descripción del proyecto 12
3.1.2 “Goal” del proyecto 13
3.1.3 “Scope” del proyecto 13
3.1.4 Sponsors 14
3.1.5 Schedule 14
3.1.6 Riesgos y Costes 15
4. DISEÑO 17
4.1 Overview 17
4.2 Desarrollo de software 17
4.2.1 Framework 17
4.2.2 Arquitectura del sistema 18
4.2.3 Descripción de los módulos del sistema 21
4.2.3.1 Análisis de Vídeo 21
4.2.3.1.1 Color 21
4.2.3.1.2 Movimiento 26
4.2.3.1.3 Comunicación (información) 27
4.2.3.2 Generación de contenido gráfico para iluminación 29
5. METODOLOGÍA DE TRABAJO 33
6. RESULTADOS EXPERIMENTALES 35
6.1 Pruebas en la Demo Room 35
6.2 Discusión sobre los resultados 49
6.3 Future Work 51
6.3.1 Posibles mejoras 51
6.3.2 Otros posibles usos 52
7. CONCLUSIONES DEL PROYECTO 54
REFERENCIAS 55
APÉNDICE 56

LISTA DE FIGURAS

Página

Fig 1. Lighting design by Michal-Maciej 6
Fig 2. band Genesis's lighting design by Patrick Woodroffe 6
Fig 3. "New Angles" interactive installation by Super Nature.....7
Fig 4. Luminic interactive lamp "Infinity Light" by Scarpia 8
Fig 5. "Wasbah lights" interactive lighting design by Wasbah Lights 8
Fig 6. Diagrama de Boston: interactividad vs expresividad en el mercado 9
Fig 7. Project Schedule 14
Fig 8. Esquema de arquitectura del sistema 20
Fig 9. Arquitectura del análisis de vídeo 21
Fig 10. HUE Wheel 22
Fig 11. Cálculo de valores HUE que más aparecen 23
Fig 12. K-means clustering.....24
Fig 13. Cuantización de color.....25
Fig 14. Optical Flow a través de frames en el tiempo 26
Fig 15. Análisis de optical flow.....27
Fig 16. Datos de texto de la información de color.....28
Fig 17. Layouts del sistema generador de gráficos 30
Fig 18. Random layout sobre "dancing girls" 1 36
Fig 19. Random layout sobre "dancing girls" 2 36
Fig 20. Random layout sobre "dancing girls" 3 37
Fig 21. Random layout sobre "dancing girls" 4 37
Fig 22. Random layout sobre "ice cream" 1.....38
Fig 23. Random layout sobre "ice cream" 2.....38
Fig 24. Random layout sobre "ice cream" 3.....38
Fig 25. Random layout sobre "ice cream" 4.....38
Fig 26. Líneas verticales sobre "jungle", (4 divisiones).....40
Fig 27. Líneas verticales sobre "jungle", (12 divisiones).....41
Fig 28. Desplazamiento de líneas verticales sobre "paraguas" 1 ..43
Fig 29. Refresco de líneas verticales sobre "paraguas".....44
Fig 30. Desplazamiento de líneas verticales sobre "paraguas" 2 ..45

Fig 31. Tubos led en el suelo con movimiento de líneas verticales sobre “paraguas” 46
Fig 32. Tubos led colgantes con random sobre “dancing girls” 47
Fig 33. Tubos led colgantes con líneas verticales sobre “dancing girls” 48
Fig 34. Tubos led colgantes con líneas horizontales sobre “dancing girls” 48
Fig 35. Tubos led colgantes con disposición radial sobre “dancing girls” 49

.

1. INTRODUCCIÓN

1.1 PRÓLOGO

La luz es la principal razón por la cual el ser humano es capaz de observar lo que le rodea, procesarlo y actuar. Vemos gracias a la luz y siempre ha sido un elemento imprescindible para la vida, así como para infinidad de campos del desarrollo y la evolución del hombre.

Dado que es el principal elemento actor en la percepción visual, su variación permite cambiar la percepción del espacio, volumen, sentimientos, tiempo…

Esto se ha visto reflejado en la historia desde que en la antigua Grecia se manifestaron las primeras muestras de arte en las que la luz era partícipe y dotaba, en la época helenística, de volumen a las pinturas. Ambas luz y arte han ido cambiando a lo largo del tiempo estrechamente relacionadas, ya que el cambio de expresar o percibir una de ellas, lo supone también en el otro.

Nuevas maneras de obtener luz han proporcionado nuevas formas de entender realidades (físicas o imaginarias) que ya existían. Arquitectura, pintura, teatro, fotografía, cine, shows… todas han tomado nuevas dimensiones a la par que sucedían progresos en la luz: de ventanas y vidrieras colocadas estratégicamente en edificaciones a diseño de iluminación, de fotografía en blanco y negro a *lightpainting*, de focos en teatros a pantallas gigantes en estadios; todas consiguen que las sensaciones que pueden ofrecer lugares, canciones, movimientos, imágenes, puedan ser transformadas al influir en la persona que las capta.

1.2 EL DILEMA

En el tiempo actual, en el que el diseño de las experiencias de casi cualquier tipo tienden a ser multimodales –imagen y sonido se hacen prácticamente imprescindibles, siendo cada vez más habitual interactuar también con otros sentidos– el trato de la luz en sus diferentes funcionalidades se ha convertido en un elemento esencial ya que es el que aporta mayor influencia a la experiencia.

Sin embargo, a pesar de la progresión tecnológica, muchas de las herramientas no son todo lo adecuadas que se podría desear. En muchos casos su expresividad se ve limitada por la simplicidad de los sistemas en los que los parámetros de libre actuación son pocos, sobretodo si son interactivos; y a mayores, cuanto más eficaces a nivel inmersivo, también más costosos.

El grado de modalidades también influye, ya que cuantos más sentidos e intercambios de información hay entre el usuario y el elemento que transmite la experiencia, mejor es ésta última, pero también aumenta la dificultad.

Si además tenemos en cuenta que estos medios creados para influir en las percepciones tienen que estar muy medidos, cuidados y no son nada arbitrarios, nos encontramos con poca flexibilidad y un alto requisito de dedicación, con lo que no se puede simplificar y automatizar su diseño.

A razón de todo lo anterior, y desde el punto de vista de la luz, uno de los obstáculos que existe entre el artista o diseñador lumínico y la materialización de su idea, pasa por implementación técnica del mismo, las limitaciones del diseño, y también su coste.

Las vías usuales para conseguirlo conllevan adquirir profesionales y material programable o interactivo de alto precio, o la fabricación del mismo, siendo necesarios conocimientos y habilidades técnicas

avanzadas. Si además se quieren representar composiciones visuales en dichos elementos, hay que añadir más personal y software al proceso, que encarece el montaje final o multiplica su complejidad si tiene que ser desarrollado.

Todo esto hace que los mejores resultados solo puedan ser vistos en grandes proyectos artísticos o de investigación, siendo en ocasiones más importante el resultado técnico final, que no el influjo que la experiencia pueda ejercer en la forma de percibir realidades del usuario.

1.1. LA RESPUESTA

Volviendo al hecho de que la luz es el actor con más fuerza en la percepción humana, podría ser utilizada para potenciar montajes visuales sencillos, a modo de punto intermedio entre una muestra visual y una experiencia inmersiva. No quiere decir que no existan herramientas en las que se haya utilizado ya, pero para que suponga una superación de las barreras anteriormente descritas, tiene que ser eficaz y eficiente.

Para obtener un buen resultado la luz deberá interactuar con el contenido de algún modo, acompañando a lo que se enseña y reforzando tanto su mensaje como la sensación perceptiva que se desea en el espectador. La herramienta también será autónoma (en parte) y que permita un sencillo diseño de la interacción y/o espacio.

Es en este punto en el que entra en juego este Trabajo de Fin de Grado: diseñar una herramienta nueva o función mejorada de alguna existente, para cubrir este *gap* en el que diseñar y realizar experiencias con luz se vuelve difícil al requerir de profesionales cualificados y conocimientos técnicos específicos.

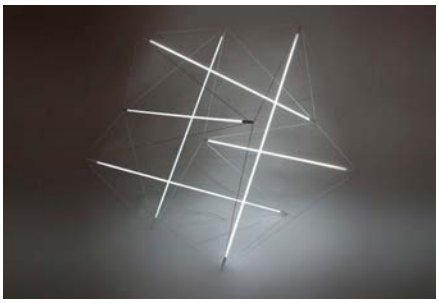
Como un punto de partida, empezar por la creación de un sistema de iluminación automática, basada en un contenido visual y que actúe a la par con el mismo, parece adecuado. Este sistema analizará el material visual, y generará un contenido lumínico representable en el espacio según se desee. Este desarrollo permitirá, con una sencilla selección de variables y un *footage*, generar de forma inmediata "luz" bajo unos criterios de diseño, sin requerir de grandes recursos en profesionales cada vez que se quiere proyectar una idea o experiencia al espacio real.

En este proyecto se planteará y llevará a cabo un limitado *primer stage*, acotado a un marco de trabajo y casos de uso abarcables en cuanto a tiempo, dificultad y recursos; sin embargo podrá ser mejorado para cubrir competencias mucho mayores como puede ser la interacción con más clases de contenido multimodal, así como tipos de iluminación.

2. STATE OF THE ART

El trato y diseño de la luz como centro de atención ha ido creciendo en la última década. Cada vez se tiene más en cuenta y se comprende su importancia, por lo que se le dedica más protagonismo. La expansión del *mapping* de superficies (sobre todo fachadas de edificaciones) y los visual-shows en eventos musicales por el mundo ha sido la catapulta para que el *lighting design* sea tomado en cuenta y primordial en el diseño de espacios y experiencias.

Sin embargo, este campo actualmente está muy limitado y menos desenvuelto que otros.
El diseño de iluminación de tipo decorativo o para eventos, la cual está programada (a tiempo real o no), se realiza por profesionales, a priori, y utilizando experiencia y herramientas software. Es muy agradable a la vista, pero son casos particulares, para los que se realiza un estudio y diseño personalizado, su interacción con el espectador u otro medio es mínima.



figuras 1, 2 – izq.: Lighting design by Michal-Maciej [1] *dcha.: band Genesis's lighting design by Patrick Woodroffe* [2].

Si nos acercamos al *interactive lighting*, nos encontramos unos peldaños más arriba en términos de riqueza de experiencia. Los elementos e instalaciones han sido pensadas especialmente para el

intercambio de información entre el usuario y el montaje, logrando que la experiencia sea más inmersiva e impresionante.

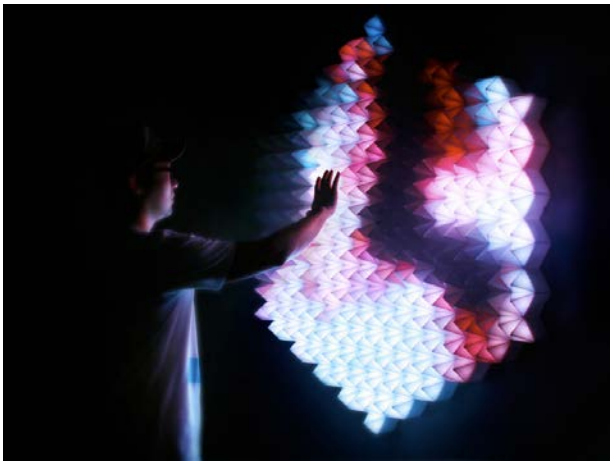


figura 3 – "New Angles" interactive installation by Super Nature [3].

En este caso podemos ver instalaciones lumínicas interactivas, las cuales solo se diferencian de las anteriores en que el contenido visual o de luz cambia o se genera según las acciones del usuario, pero sigue requiriendo un diseño específico y previo (y más complejo) (fig. 1); o también herramientas que si permiten el diseño de iluminación, pero cuya interactividad es el hecho de la operar con las luces en tiempo real (fig. 2). Se aproximarían al caso ideal sistemas como Light Act - Systems [4], que ofrecen el diseño de una instalación lumínica interactiva, pero se requiere una planificación y montaje específico para cada caso, con lo que se vuelve a la necesidad de software, hardware y profesionales.

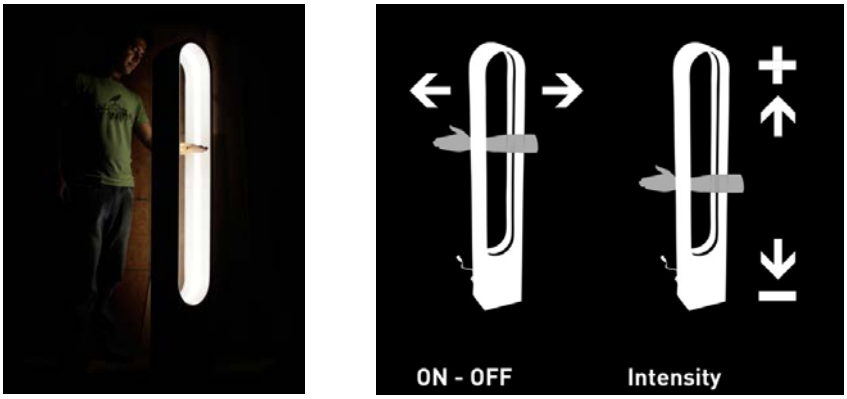


figura 4 – Luminic interactive lamp "Infinity Light" by Scarpia [5].

figura 5 – "Wasbah lights" interactive lighting design by Wasbah Lights [6].

En el mercado actual existe un compromiso entre expresividad e interactividad: aquellos sistemas o aplicaciones que cuentan con un alto grado de interacción, se ven reducidos a ser poco llamativos debido a los recursos necesarios para el flujo de datos. Por otro lado, los casos que ofrecen experiencias que pueden impresionar al espectador requieren de grandes recursos y es necesario un gran control, lo que limita la interactividad (fig. 6).

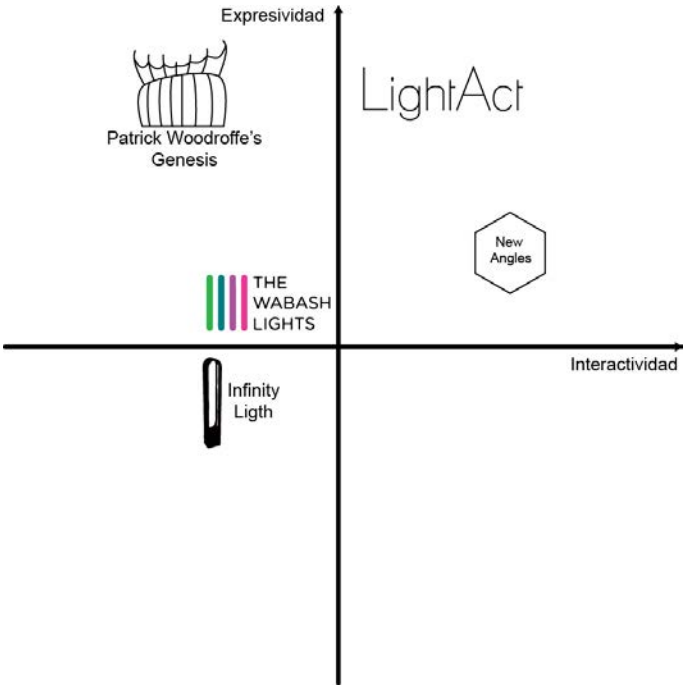


figura 6 – Diagrama de Boston: interactividad vs expresividad en el mercado.

Si nos adentramos en campo del automatismo, nos encontramos más restricciones. Normalmente la automatización de la iluminación responde a objetivos prácticos, tales como el ahorro de energía (sensores de presencia o programación horaria), dirección de la atención (escaparates y espacios); y no a un proceso de diseño.

3. MOTIVATION

Ante la falta de una oferta convincente para el diseño automático de luz en experiencias, este Trabajo de Fin de Grado tratará de crear una solución. El interés personal en la iluminación como forma narrativa para transformar y ensalzar experiencias junto con los conocimientos, tanto teóricos como de campo en el área, hacen de este proyecto el reto idóneo.

Pese a no haber encontrado ninguna herramienta con la que atacar directamente las deficiencias descritas, si será posible partir de un producto que tiene parte del camino hecho y cuya estructura permite el avance abierto en múltiples direcciones.

Si se había hablado anteriormente de que un gran obstáculo entre los diseñadores o artistas visuales que quisiesen tratar y diseñar con luz era la necesidad de conocimientos y habilidades técnicas (o el presupuesto para la subcontratación), con Protopixel [7] se rompe esa barrera.

Mediante la aplicación y el controlador hardware, de nombre Protopixel para ambos, es posible la comunicación con leds del tipo "WS28-". Éstos poseen un chip que permite su direccionamiento y la recepción de información independiente, lo que se conoce como "*pixel mapping*". Realizar anteriormente mapeado de píxeles requería de procedimientos que incluían muchas tareas tales como soldar, operar con tarjetas demultiplexadoras DMX, protocolos de comunicación, etc.

Sin embargo, la simple conexión de los leds al controlador y la configuración necesaria en la aplicación conceden al usuario la capacidad de transmitir colores, imágenes, vídeo y un sinfín de posibilidades por crear a la composición o estructura. Se consigue así que el artista, diseñador, o cualquiera que sea el usuario, tenga

que preocuparse solo de que aspecto tendrá al final su idea, en el mundo físico, y que mostrar o iluminar a través de ella.

¿Porqué utilizar Protopixel como punto de partida? Porque permite el desarrollo de módulos (scripts de código) por parte de cualquier usuario. En lenguaje *open-source* y con una pequeña *starter-guide*, pueden crearse aplicaciones dentro del programa que realicen cualquier tarea que se desee, siempre que exista la capacidad para implementarlas.
Esto hace a Protopixel el medio ideal para crear un generador automático de iluminación; y el elemento visual en el que basar la creación de contenido será el vídeo. Ya que los vídeos son en la actualidad una de las formas más básicas de comunicación visual -variable en el tiempo-, añadir una nueva dimensión mediante la luz para enriquecer la experiencia, desde un punto de vista artístico o de mercado es un primer paso abarcable.

3.1 DEFINICIÓN DEL PROYECTO

3.1.1 Descripción del proyecto

Este proyecto desarrollará una herramienta que permita la generación automática de contenido lumínico a partir de un contenido visual.

Se utilizará el producto Protopixel, el cual ofrece la posibilidad de mapear dicha información lumínica, que puede ser un vídeo o scripts nativos, a luminarias, actualmente led.

Se le añadirá una nueva funcionalidad, que será la encargada de producir el contenido a representar en las luminarias. Éste estará basado en información extraída de material videográfico y lo acompañará para extrapolar la experiencia más allá de observar algo que ocurre en un marco en dos dimensiones.

Constará de dos partes: análisis de vídeo y generación gráfica de contenido adaptable a iluminación mediante Protopixel. Ambas serán ideadas de manera autónoma y abierta, lo que permitirá su posterior inclusión en diferentes entornos, tan solo con la necesidad de adaptarlas. Ello también facilita el diseño, haciendo posible abstraerse y realizar solo una conversión en la fase de implementación.

Debido al curso de prácticas de empresa en Emotion Experience (Emex) [8], y su uso de vídeo contenido, así como la venta y alquiler de diferentes tipos de displays, se podrá realizar una última prueba del funcionamiento del proyecto acabado en su showroom. Además, se utilizarán contenidos relacionados con la temática del material que crea Emex. Esta prueba será un preámbulo a una posible implementación del analizador de vídeo dentro de su plataforma, que es la aplicación para gestión de contenidos que han desarrollado y utilizan.

3.1.2 “Goal” del proyecto

El principal objetivo del proyecto es la creación de una herramienta automática que, mediante el uso de luz, mejore la experiencia de presenciar un contenido en vídeo por el usuario.

Para lograr esto, y por consiguiente, es también objetivo el desarrollo de una nueva funcionalidad para Protopixel, generadora de contenido gráfico que mapear a luminarias, siendo éste creado a partir de información extraída mediante un previo análisis de vídeo, externo a Protopixel.

3.1.2 “Scope” del proyecto

- Extraer el mayor número de información y características de un vídeo, desde enfoque de utilidad para el proyecto.

- Generar un contenido visual automático que se adapte al uso que Protopixel necesita para gestionar la luz.
- Crear un sistema de comunicación entre el analizador de vídeo y Protopixel.
- Sincronización entre la reproducción del vídeo y la *perform* lumínica.
- Ensalzamiento de la experiencia visual.

3.1.4 Sponsors

Este proyecto esta esponsorizado por la Universitat Pompeu Fabra, Protopixel y Emotion Experience.

El proyecto está dirigido por Sebastián Mealla, CEO de Protopixel e Investigador en el MTG de la Universitat Pompeu Fabra.

3.1.5 Schedule

	4o trimestre 2015	1er trimestre 2016	2o trimestre 2016
Fase del Proyecto	Inicio del proyecto	Desarrollo de Software	Tests del sistema
	Constitución del Proyecto	Analizador de vídeo	Entrega TFG
	Toma de contacto con Protopixel	Generador de gráficos para Protopixel	Defensa del TFG

figura 7 – Project schedule

3.1.6 Riesgos y Costes

Riesgos:

- El desarrollo de software es esencial y una de las partes más delicadas, ya que incluirá diferentes lenguajes de programación y librerías, además de la comunicación entre distintos elementos.

Cualquier fallo o tiempo extra que haya que dedicarle a la implementación informática, fuera del plazo planeado para ello, retrasará las pruebas de campo y el finalizado del proyecto.
- Disponibilidad de los profesores y el espacio de pruebas técnicas.

Costes:

El coste del proyecto depende de las horas dedicadas y el material necesario.
El material requerido para llevar acabo el proyecto será:
- Versión actualizada de la aplicación Protopixel
- Controlador hardware Protopixel
- Luminarias y espacio para hacer las pruebas (DemoRoom Protopixel)

Todas las herramientas han sido proporcionadas por Protopixel.

4. DISEÑO

4.1 OVERVIEW

Para evaluar el incremento de la experiencia visual y la eficacia del automatismo del sistema, necesitamos llevar a cabo el análisis de diferentes tipos de vídeo, generar su contenido lumínico anexo en Protopixel y realizar experimentos con distintos casos.

Dado que Protopixel se encarga de toda la comunicación entre la información visual y las luminarias, el pilar principal del proyecto es el desarrollo de software. También es conveniente remarcar que aunque el hardware sea necesario solo en las pruebas, el diseño del proyecto debe tenerlo en cuenta ya que la luz es un elemento físico, y su percepción puede variar notablemente si no se tiene consideración.

4.2 DESARROLLO DE SOFTWARE

4.2.1 Framework

El primer paso ha sido elegir que herramientas utilizar. Debido a que es necesario analizar vídeo y desde un principio se deseó el uso de medios open-source, la librería OpenCV [9] se presenta como la mejor opción. Es libre, tiene un gran respaldo por la comunidad y es fácilmente integrable en múltiples lenguajes de programación (Java, C++, Python, etc.).

De forma análoga, para la generación de gráficos que mapear mediante Protopixel, el uso de OpenGL [10] será lo más adecuado.

Para la manera de proceder, y a consecuencia de que ambas herramientas son fácilmente portables entre lenguajes de programación, se ha escogido el entorno java “Processing”[11], por

familiaridad con el mismo. A causa de la experiencia previa en su utilización, que implementa nativamente OpenGL y que existe una librería que adapta los comandos más comunes de OpenCV, desarrollar el sistema dentro de Processing es la opción más rápida y eficaz. Más tarde, solo será necesario mover y adaptar cada parte a su lugar.

La cuestión de qué vídeos utilizar ha sido simplificada gracias a Emex. Las tipologías a día de hoy son innumerables, pero el hecho de que Emex trabaje mayormente con videos promocionales para marcas y empresas nos permite limitarnos a contenidos de tipo "spot".

4.2.2 Arquitectura del Sistema

La arquitectura general del sistema es sencilla:

- Módulo 1 = Analizar el vídeo, extraer características, transmitir la información
- Módulo 2 = Recibir la información, generar el contenido gráfico, mapear.
- Sincronizar y mostrar la experiencia enriquecida.

Sin embargo, debe decidirse que características se van a analizar, como se hará la comunicación de datos entre el analizador de vídeo y el generador, así como que forma o aspecto tendrán los gráficos que mapeará Protopixel.

Para el análisis de video se ha elegido la extracción de información de color y movimiento, que son características generales e inmediatas, y son dos de las informaciones primarias naturales que el ser humano procesa al ver un vídeo.

OpenCV también provee de herramientas para la detección de escenas, fondos y seguimiento de objetos; pero estas son útiles en

casos más específicos, con lo que añadirlas al video-análisis restringiría la usabilidad del sistema y además no son compaginables con el framework de este proyecto. Volveremos a tratar el tema más adelante, durante la descripción de cómo ha sido el desarrollo a nivel de programa.

La producción de gráficos, que más tarde serán traducidos y llevados por Protopixel a las luminarias, se basará en quads [12] de OpenGL. Aquí se ve reflejada una de las ventajas que se describieron de Protopixel: es posible diseñar un contenido adaptado a lo que la aplicación hará con él, para llevarlo al plano lumínico.

Las luminarias no son un display cuya resolución importa, si no que son puntos que emiten luz y, por consiguiente, solo es necesario asignar un "color" e intensidad a cada uno de ellos. Este valor de color variará en el tiempo en función de lo que el generador automático opere, pero podemos trabajar con una unidad tan básica dentro de OpenGL como es un cuadrado plano, con un color asignado.

Lo que si es decisivo en Protopixel (y en el espacio físico) es la posición de las luminarias, por ello el output del script tendrá diferentes disposiciones espaciales de la asignación de color en los quads, para así poder adaptarse a distintos casos o montajes.

La comunicación entre ambos módulos del sistema puede abordarse de varias formas. Dependerá de si ambos están situados en una misma máquina, de si se ejecutan en tiempo real o no…. El uso de OSC (open sound control) [13] podría ser una solución más que válida, pero debido a que en inclusión final en el sistema de Emex, el análisis será hecho previamente, la utilización de archivos de texto es suficiente, aunque pueda parecer extremadamente simple.

Estos archivos guardarán línea por línea la información que el analizador de vídeo tome a cada frame, siendo estos datos una marca de tiempo (para evitar problemas con frame-rates diferentes), la información de color y la información de movimiento. El segundo módulo tomará la información y representará el contenido gráfico en el mismo orden . El esquema de nuestro sistema sería el siguiente (fig. 7).

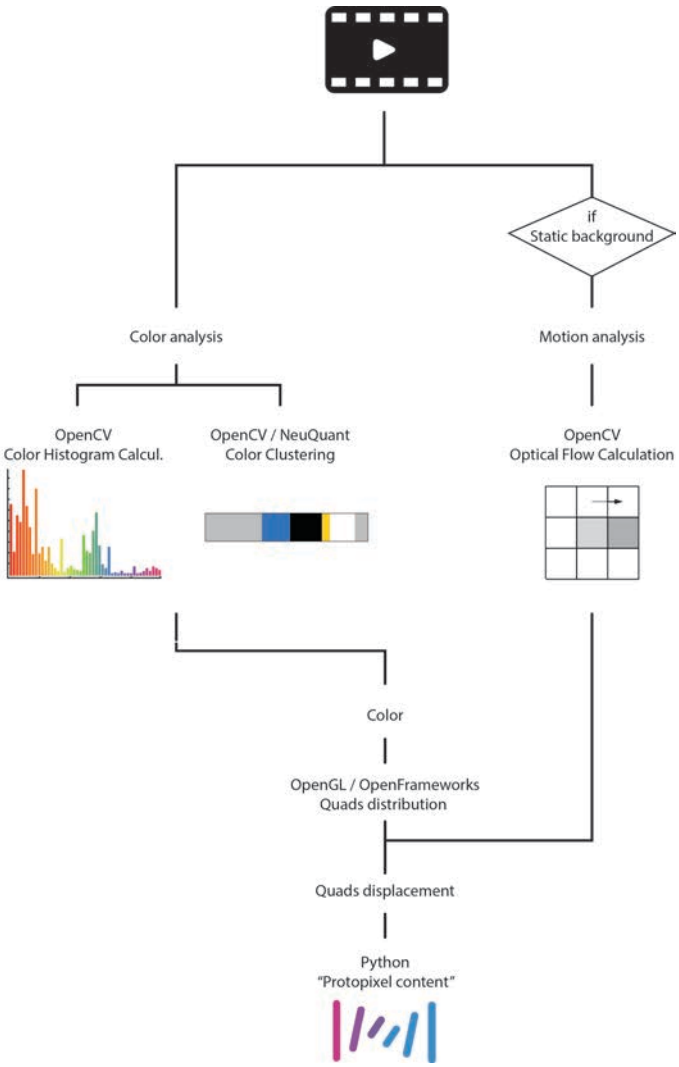


figura 8 – Esquema de arquitectura del sistema.

En el esquema aparecen referencias a elementos (como NeuQuant,, OpenFrameworks, optical flow, etc) de los que hablaremos a continuación.

4.2.3 Descripción de los módulos del sistema

Vamos a profundizar a continuación en el software que hemos desarrollado, lo cual es la parte mayoritaria de este proyecto.

Aquí hablaremos de la arquitectura interna de cada módulo y como ha sido programado, y aunque se muestre código, será de la forma más esquemática posible.

4.2.3.1 Análisis de Vídeo

4.2.3.1.1 Color

OpenCV es una herramienta muy potente y además, muy bien documentada. Con ella el análisis de vídeo en tiempo real se vuelve una tarea mucho más sencilla, si bien es necesario saber como emplearla.

Tras una labor de documentación previa necesaria, nuestro esquema inicial del análisis de vídeo en Processing lucirá así:

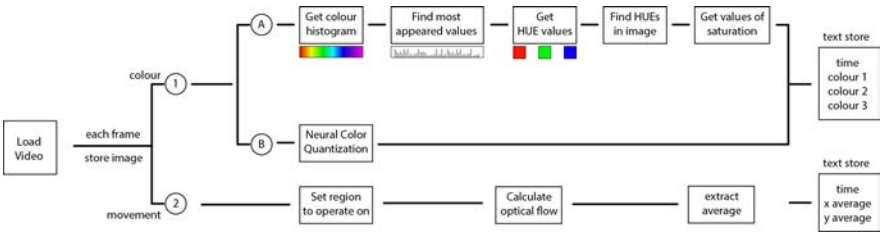


figura 9 – Arquitectura del análisis de vídeo

El acto de cargar el vídeo y obtener a cada frame la imagen del mismo es trivial. Es el procesado que realizamos de esa imagen, ya que el análisis y procesado de vídeo se basa en la información tratable en cada frame y la relación de los mismos a lo largo del tiempo.

Primero, y acerca de la detección de color, tenemos que en el camino A hacemos uso de los histogramas de la imagen, en concreto, del "*Hue histogram*".

Aquí se debe hacer un pequeño inciso: si bien el modelo de color RGB nos indica la composición de un color en función de las "cantidades" de rojo, azul y verde; el modelo HSV [14] ("Hue", "Saturation", "Value") lo hace según el ***hue***, la saturación y el brillo del mismo, respectivamente. ¿Qué es el hue? Es el tono del color en su estado puro. Es por ello que durante todo el diseño del análisis del color del vídeo se trabajará con su hue, ya que es la forma natural y más sencilla de hacerlo. Así, el tono del color dependerá solo de una variable, y no de la composición de tres. La saturación y el brillo indican la cantidad de "blanco" y "negro" que se mezcla con el color puro. Usualmente el hue se trata como una rueda en el que el valor varía a lo largo de 360 grados, y la saturación y brillo como valores de 0 a 1. Sin embargo, también es común la representación de las tres cantidades en un rango de 0 a 255 ya que a menudo se computan desde los valores RGB, y será este método con el que se trabaja en este proyecto.

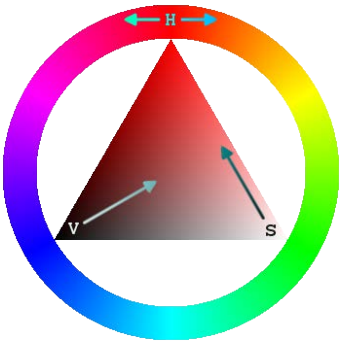


figura 10 – HUE wheel

Volviendo al diseño, se extrae mediante OpenCV el histograma del tono de color (hue), lo cual nos indica para cada valor de hue el número de veces que un pixel toma ese valor, obteniendo así "la

frecuencia de aparición de cada color". Una vez hecho esto, y a consecuencia de que el histograma solo nos ofrece información del tono del color, buscamos estos píxeles en la imagen y extraemos su brillo y saturación.

El resultado fue el correcto pero no el óptimo. En vídeos en los que había una gran presencia de, por un lado, pequeñas variaciones en el tono del mismo color; o por el otro, distintas cantidades de saturación y brillo dentro del mismo tono, ocurría que el output de colores que más aparecían eran siempre muy parecidos. Aunque hubiese otros colores muy presentes en la foto, la cantidad de uno de ellos eclipsaba a los demás, como puede apreciarse en la siguiente figura (fig10).

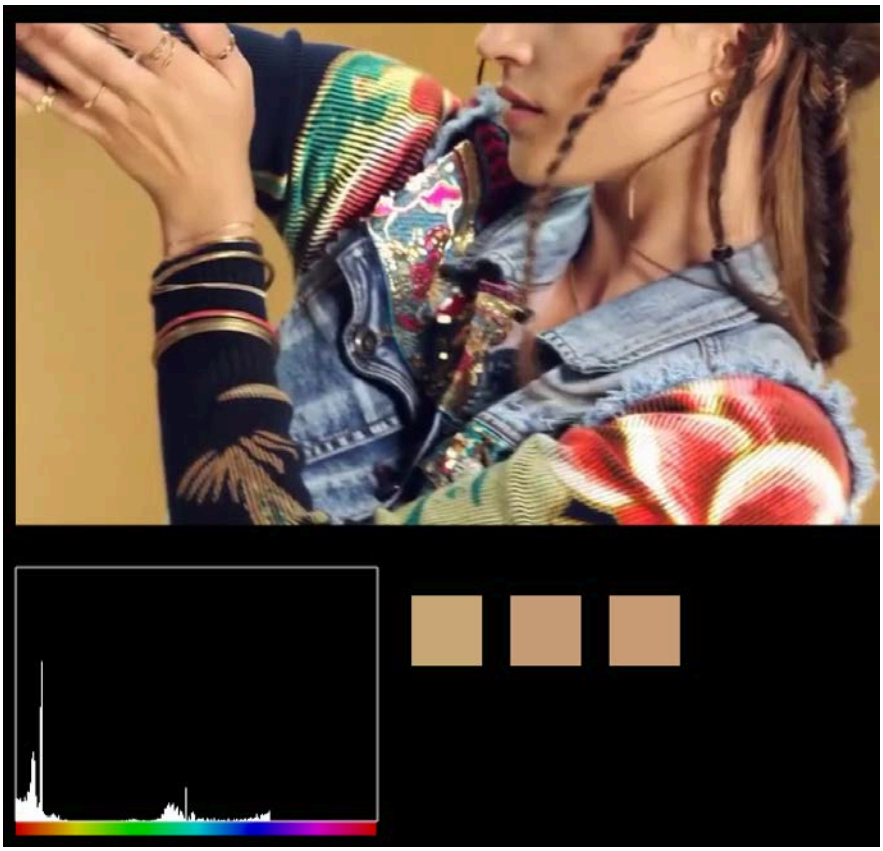


figura 11 – Cálculo de valores HUE que más aparecen.

Es entonces cuando hubo que recurrir al camino B: el "color quantization". Podría haberse encontrado una forma para que, en la detección de valores más comunes en el histograma, tuviese que existir una cierta diferencia entre ellos. Así podrían detectarse los diferentes picos en el histograma y que no se encontrasen todos alrededor del mismo punto. Esto en cambio, hubiese sido perjudicial si realmente el frame estuviese compuesto íntegramente de diferentes tonos de un solo color.

La cuantización –o segmentación- de color es un proceso en el número de distintos colores en una imagen, normalmente intentando que la nueva imagen cuantizada sea lo más parecida a la original. La aplicación directa más común es la reducción de información que requiere la imagen para ser mostrada (displays con limitaciones) o transmitida (memoria y comprensión). No obstante, el procedimiento será útil para encontrar los colores que más aparecen en el vídeo a cada frame.

Aquí es donde aparece el camino B, y en el se presentan dos opciones con las que realizar el color quantization. La primera es el clustering (o algoritmo de agrupamiento), el cual es un procedimiento para agrupar, vectores en principio, según un criterio de distancia o similitud [15], aunque se ha extrapolado a múltiples usos, tales como el que nos atañe. OpenCV integra el clustering de "K-means", que podemos resumir como la agrupación de elementos en "k" grupos, bajo la premisa que cada elemento está en el grupo con la media más cercana. En la figura 11 puede verse gráficamente como actúa, creando los grupos, asignando los elementos y haciendo re-cálculos con las medias.

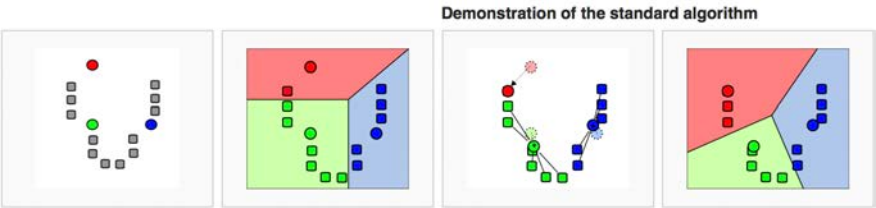


figura 12 – K-means clustering [15].

La segunda opción, la cual se ha utilizado en el proyecto debido a haber sido más fácil trabajar con ella (el k-means clustering ofrecía problemas en su adaptación en Java) es el "NeuQuant", o "Neural Image Cuantization" [16]. Este algoritmo hace uso del *machine learning*, y básicamente consta de una pequeña red de neuronas que se reorganiza mediante el aprendizaje para satisfacer la distribución de colores de la imagen. Gracias a esto podemos después extraer las agrupaciones de colores.

Aunque se haya desestimado el uso de k-means clustering en el trabajo, es importante dejar constancia de él, ya que una de las características principales de este proyecto es que sus módulos son portables a otros lenguajes, con lo que disponer tanto de k-means como de NeuQuant es interesante de cara al futuro. Cada uno puede ajustarse mejor al caso específico con el que se esté tratando, a nivel de implementación y desarrollo, como a nivel computacional, aunque no se hayan presentado carencias de recursos en su uso.

Desde el comienzo se decidió el empleo de los tres colores más comunes de un forma totalmente arbitraria, suficientes para tener variedad en la paleta cromática, pero no demasiados para que los grupos se pareciesen repetidamente. El sistema es totalmente susceptible de ser cambiado a dos, cuatro o los colores que se deseen. A continuación podemos ver los resultados de la cuantización de color (fig. 12).

figura 13 – Cuantización de color.

Como puede observarse, para el mismo frame, obtenemos colores mas distantes entre sí con la cuantización, que con el cálculo de histograma.

4.2.3.1.2 Movimiento

Para el análisis de movimiento contamos con una de los instrumentos más útiles de OpenCV: el "Optical Flow" (traducible como "flujo óptico") y que debemos entender como el movimiento aparente -y relativo- de los objetos respecto a un observador [17].

En el plano bidimensional que estamos manejando, podemos explicar simplificadamente, que el optical flow computa para cada pixel (o conjunto de pixeles) la dirección 2D que presenta a lo largo del tiempo. Este movimiento se calcula por medio de derivadas parciales entre frames consecutivos, respecto a espacio y tiempo; es decir, para cada punto se busca en los frames consecutivos cuánto se ha desplazado en el espacio. En la figura 13 puede verse un ejemplo ilustrado.

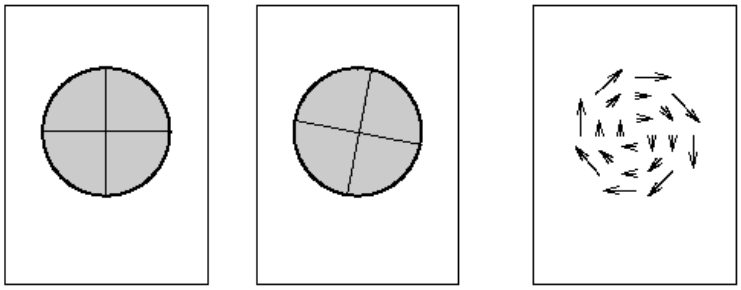


figura 14 – Optical Flow a través de frames en el tiempo [18].

La aplicación en nuestro sistema es sencillo: a cada frame del vídeo, se realiza el cálculo de optical flow para cada pixel. Como deseamos obtener el "movimiento general" que ocurre en el vídeo, seleccionamos una región central, de dimensiones inferiores al tamaño del mismo para descartar la información de los bordes. De aquí podemos extraer la media en forma de vector, teniendo así un valor de X y otro de Y para esa región. Esta operación puede realizarse a la par con el análisis de color, o por separado.

Es importante destacar que esta operación solo es útil si en el vídeo se presentan un fondo estático (o prácticamente en su totalidad) y un elemento que se mueve por delante de él. Dado que el cálculo se realiza a lo largo de los frames, si la totalidad de los pixeles cambia de uno a otro, no es posible derivar la continuidad del movimiento dentro de la imagen, ni a lo largo del tiempo (fig 14.). Por ello, vídeos en los que hay un cambio de escena no sirven, y del mismo modo, aquellos en los que todo se mueve no aportan una media eficiente.

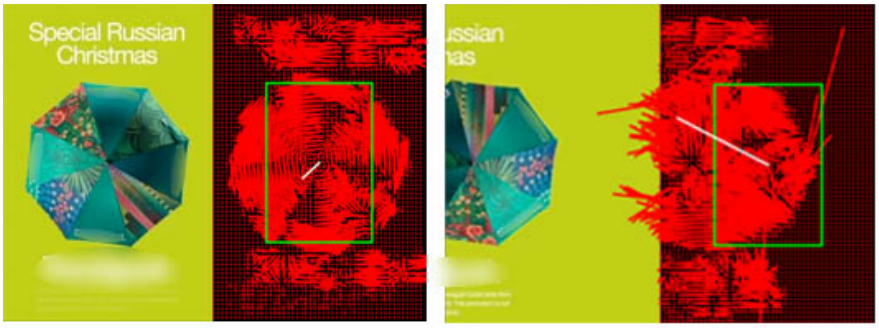


figura 15 – Análisis de optical flow.

Este hecho restringirá las pruebas a dos use-cases marcados, de los cuales se hablará más adelante.

4.2.3.1.3 Comunicación (información)

Sin ser lo que podríamos llamar módulo, se ha decidido tratar esta parte por separado porque es una pieza esencial, e igual que los elementos anteriores, puede cambiar y tratarse independientemente para adaptarla a las necesidades de la estructura que la integre. En ella discutiremos como hemos resuelto el guardado, traspaso y utilización de información por parte de los módulos previos; y que otras soluciones tendrían diferentes situaciones.

Como el caso práctico que manejaremos será bajo el marco de uso de Emotion Experience, el vídeo será analizado primero para guardar la información, y más tarde el contenido para iluminación

se generará en Protopixel a la par que se reproduce el vídeo. Esto nos permite almacenar la información en un archivo y utilizarla más tarde.

A pesar de su simplicidad, la escritura y lectura de archivos cumple perfectamente con lo deseado. Para el análisis de color, a cada frame guardamos en una línea el tiempo actual de reproducción del vídeo (en segundos y milisegundos, en que momento guardamos el valor) y los tres colores resultantes de la cuantización (nueve valores, 3 x RGB). Hacemos lo mismo con el análisis de movimiento para aquellos vídeos que servirán para las pruebas, escribiendo el tiempo, y los valores de X e Y de la media del optical flow. Se puede almacenar los datos por separado, o en un mismo archivo, los cuales lucen como sigue (fig. 15).

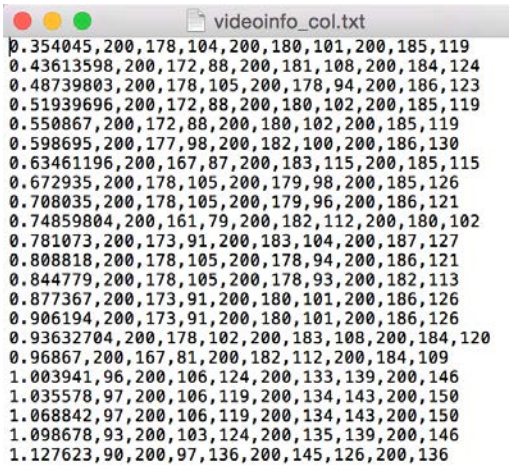


figura 16 – Datos de texto de la información de color.

Su lectura ha llevado un poco más de trabajo. De la misma manera que se escribieron línea a línea los datos en el archivo, así se leerán. Con dos pequeñas funciones que podrán verse en el apéndice, se divide el archivo en líneas y éstas se guardan por separado en un array, y más tarde puede extraerse cada valor numérico almacenado a tiempo real. Sin embargo, la lectura no puede realizarse frame a frame ya que existe la (muy alta) probabilidad de que el frame-rate de escritura no sea igual al de lectura.

Para solucionar la sincronía a la hora de la lectura, una pequeña función se encargará de leer y almacenar las anotaciones de tiempo del archivo. Después, tomará el punto temporal actual de ejecución del módulo (script) y buscar mediante la distancia euclidiana 1D, que línea será la que deba leerse en el frame en curso.

Una vez que sabemos a cada momento que línea leer para que la sincronización sea óptima, solo es necesario extraer los valores de color y/o movimiento y operar con ellos.

4.2.3.2 Generación de contenido gráfico para iluminación

El material visual que vamos a crear, tal como ya se ha discutido, se basará en quads de OpenGL. Este elemento es fácil de generar, tratar y requiere pocos recursos; además de adaptarse perfectamente a las necesidades del sistema.

Como Protopixel permite la distribución de luminarias en el espacio dentro de su editor, es posible realizar diferentes asignaciones de los valores de color de manera general, ya que más tarde se puede ajustar la luminaria al punto de luz y no al contrario, simplificando mucho la computación gráfica.

A pesar de las ventajas de mapeo disponibles, será conveniente crear diferentes *layouts,* en los que variará la disposición de colores de los quads para ayudar al mapping espacial. Las distribuciones elegidas serán aleatoria, líneas verticales, líneas horizontales, radiales y concéntricas –radiales- (fig 16). Además, se podrá seleccionar el tamaño del layout y el número de divisiones (cuanto de grandes son, por ejemplo, las líneas) para permitir una cierta flexibilidad a la hora de la configuración.

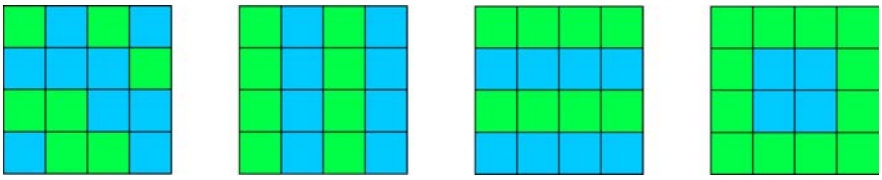


figura 17 – Layouts del sistema generador de gráficos.

Inicialmente se realizó un primer diseño en Processing, y más tarde se trasladó y terminó en un script Python, ya que es el lenguaje que utiliza Protopixel. Dentro de estos scripts, Protopixel integra Open Frameworks, un marco de programación con el que podemos utilizar diferentes herramientas, tales como OpenCV y OpenGL [19]. Describiremos ahora brevemente como se diseñó el sistema que permite la creación de gráficos en Protopixel.

En primer lugar, la muestra de quads es posible porque éstos se almacenan en una matriz. Esta matriz, junto a sus valores de tamaño "x" e "y", forman parte de las variables de una clase que llamamos "led_Matrix". Esta clase consta de varias funciones: 4 para la distribución de los quads en la matriz, 2 para dibujarla, 1 para el movimiento, y otras 4 para la lectura de la información de color y movimiento.

Para asignar los colores, se toma la información de los mismos y se asigna en la matriz variables tipo "ofColor", siguiendo un orden que depende de los layouts que ya se han comentado (random, vertical, horizonal y radial, una función para cada uno). Según se haya seleccionado el número de divisiones, se asignarán los colores a grupos mayores o menores de quads.

Para que el script muestre la matriz, la operación es tan sencilla como dibujar los cuadrados ("ofRect") en el espacio. Dependerá del input de tamaño, el número de pixeles que ocupará cada rectángulo. La otra función relacionada con el dibujado sirve para que el cambio a lo largo del tiempo de los colores no sea brusco, si no una interpolación, llamada en computación gráfica "lerp".

Para obtener la información de los colores utilizamos funciones para leer adecuadamente el archivo de texto que la contiene. Una lee el archivo y guarda las líneas en un array. La siguiente toma estas líneas y toma los valores numéricos separados por comas para transformar las variables, agruparlas y devolver los colores. Así también, hay una que realiza la misma operación para el movimiento. Finalmente, una función se encarga de comparar (mediante una distancia euclidiana) el tiempo actual de reproducción del vídeo con los tiempos anotados en el archivo, para que todo esté sincronizado.

A la hora de poner a funcionar el script, simplemente iniciamos las variables y ejecutamos a cada frame las funciones en orden correcto.

5. METODOLOGÍA DE TRABAJO

El flujo de trabajo a lo largo del proyecto ha sido una de las partes más importantes para su llevada a cabo. Debido a que el inicio supuso comenzar a programar desde cero, una buena organización y esquematización de lo que se deseaba fue imprescindible. Para ello, el apoyo de los profesores ha sido fundamental, ya que pasadas ciertas fases, la puesta en común de ideas y su visión más amplia de lo abarcado por el trabajo ha ayudado a centrar que dirección seguir.

Muchos de los objetivos alcanzados se han logrado siguiendo un exhaustivo método de documentación, investigación y sobretodo, prueba y error. Los diferentes intentos con resultados no satisfactorios son los que han conseguido florecer la autocrítica y autoanálisis para encontrar en que aspectos del proyecto había que hacer hincapié, cambiar y mejorar. Así ha sido posible que las opciones que ofrece el sistema final cubran un amplio abanico de necesidades y se puedan adaptar a diferentes casos.

A continuación veremos diferentes pruebas y experimentos realizados con distintos montajes para comprobar la eficacia de la nueva funcionalidad que hemos desarrollado.

6. RESULTADOS EXPERIMENTALES

Para probar que el sistema funciona y la nueva funcionalidad cumple con lo deseado, o por el contrario hacen falta cambios y mejores, son necesarias algunos experimentos.

Los tests mostrarán como se comporta la generación automática de gráficos en diferentes casos, siguiendo ciertas restricciones. Éstas dependerán de lo siguiente:

- Tipología del vídeo.
- Análisis del movimiento: sí o no.
- Descripción del vídeo.
- Disposición de las luminarias en el espacio.
- Parámetros:
    - Tamaño del layout
    - Tipo de layout
    - Divisiones del layout
- Resultados.

Dado que la eficacia de nuestro desarrollo depende de la expresividad y del enriquecimiento de una experiencia, el rendimiento es un tanto subjetivo, así que los resultados, en lugar de corresponder a mediciones, lo harán a una discusión de impresiones recogidas.

6.1 PRUEBAS EN LA DEMO ROOM

Todas las pruebas han sido realizadas en la Demo-Room de Protopixel, ya que cuentan con diferentes tipos de luminarias ya adaptadas para el uso con Protopixel además del apoyo in-situ por parte de los profesores.

Los vídeos utilizados han sido tomados de material para la realización de contenido de uso comercial (sin tratar). Se han elegido cuidadosamente aquellos que sacan el máximo partido al sistema y con las restricciones de use-cases debidas. El primer experimento se trata de una prueba del resultado visual que ofrece el uso de la funcionalidad desarrollada. Tras esto, se sucedieron las diferentes comparaciones para diferentes casos.

TEST “0”

Comprobaciones generales (disposición aleatoria).

Experimento para probar de forma genérica el uso de color por el sistema. Utilizaremos un vídeo que llamaremos “dancing girls” de aquí en adelante. En él, ocurre una sucesión de planos en el que unas chicas están realizando diferentes movimientos sobre un fondo plano. En cada plano hay un color dominante, junto a los colores que aportan las chicas y su ropa. No analizamos el movimiento.

Las luminarias utilizadas serán el foseado de la pared de la Demo Room, que aportará un marco de color al vídeo proyectado en la pared. El tamaño del layout ocupa todo el foseado, en un plano 2D.

El tipo de layout será el random, que distribuirá los 3 colores que más aparecen aleatoriamente por las luminarias. Debido al tamaño del layout, se asignará el color a varios leds juntos. La división será la original: 8x8 quads. A continuación pueden verse diferentes planos del vídeo con la respuesta de luz de Protopixel.

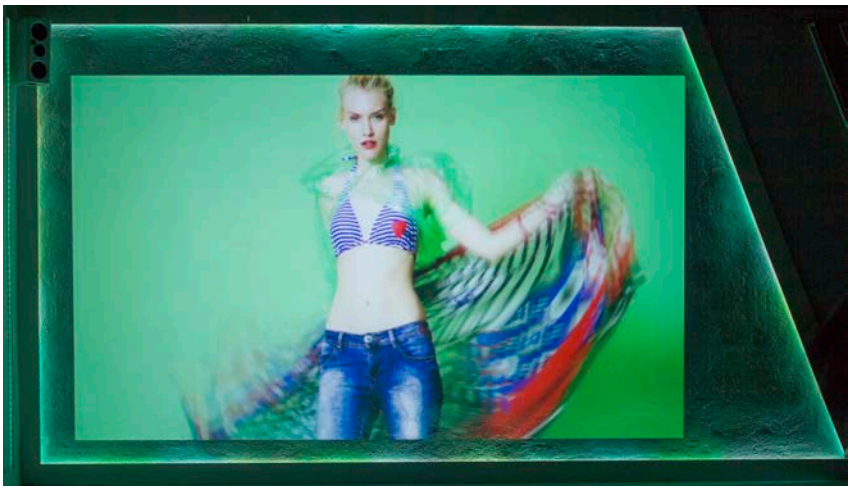


figura 18 – Random layout sobre "dancing girls" 1

.

figura 19 – Random layout sobre "dancing girls" 2

figura 20 – Random layout sobre "dancing girls" 3

figura 21 – Random layout sobre "dancing girls" 4

Resultados: Sobre el análisis y performance del color, se alcanzado un nivel satisfactorio. Los colores se corresponden y la luz acompaña al contenido. En persona se puede observar como la sincronización es aceptable, y los colores refuerzan la experiencia del visionado del vídeo.

En el script es posible cambiar la velocidad de refresco de los colores, pero cuanto más rápido es este, menor es la interpolación entre disposiciones de color; por lo que se han hecho previamente intentos hasta obtener una relación de velocidad – *smoothness* adecuada (pasos de interpolación de 0.03 a cada frame de un color a otro).

TEST 1

Velocidad narrativa del vídeo

En relación a la prueba inicial, haremos un pequeño test con un vídeo ("ice cream") en el que se suceden planos de las mismas características, pero a mayor velocidad, para comprobar en que medida esto afecta a la generación de gráficos y a la expresividad. No se analizará el movimiento, las luces seguirán siendo el foseado de la Demo Room y usaremos el layout de líneas verticales.

figuras 22, 23 – Random layout sobre "ice cream" 1 y 2 (izqd. y dcha.)

figuras 24, 25 – Random layout sobre "ice cream" 3 y 4 (izqd. y dcha.)

Resultados: El color vuelve a cumplir con las expectativas. Sin embargo, el cambio continuo de plano (se suceden 2 o 3 cada segundo) no permite apreciar bien la textura que genera la luz alrededor del vídeo, mermando la experiencia. Hay demasiadas variaciones para distinguir y admirar bien los efectos luminosos.

TEST 2

Divisiones del layout

Utilizaremos el layout de líneas verticales con un vídeo ("jungle") en el que unas chicas bailan delante de unas plantas. Compararemos la diferencia entre utilizar una mayor o menor cantidad de divisiones del layout, y que cambios a nivel visual y lumínico suponen.

Las luminarias serán las mismas que las de las pruebas anteriores, así como tampoco se analizará el movimiento. A continuación ambos resultados:

Figura 26 – Líneas verticales sobre "jungle", (4 divisiones)

Figura 27 – Líneas verticales sobre "jungle", (12 divisiones)

Resultados: El uso de color por el sistema sigue siendo correcto, como en los casos anteriores. Sobre las diferentes divisiones, puede observarse que en la figura 26 los cambios de color a lo largo de la línea horizontal están más marcados, algo que también acentúa el hecho de que exista mayor cantidad de pixeles blancos en el frame, al contrario que en los planos menos cerrados sobre las chicas. En la figura 25 la variación es más suave, y estéticamente es más agradable. Estos resultados se aplican también al uso del layout de líneas horizontales.

Podemos sacar de este test que las variaciones bruscas de color en vídeos con dos o más colores predominantes muy diferentes son menos atractivas que aquellos en los que hay un solo color, o variaciones tonales del mismo (fig. 20).

TEST 4

Movimiento

Como ya se había mencionado, el análisis de movimiento mediante optical flow restringe mucho la usabilidad del sistema, ya que es necesario que las acciones dentro del vídeo sucedan en un mismo plano; es decir, que no haya cambios bruscos de todos los píxeles, ya que así no es posible realizar un seguimiento de los mismos de un frame a otro. Es así que el análisis de movimiento lo hemos reducido a un único use-case dentro del framework que estamos abarcando, y es el caso de los vídeos en los que uno o varios elementos se desplazan frente a un fondo estático. Esto corresponde en el caso de Emex a promociones de retail en las que letras y algún objeto aparecen y desaparecen frente a un fondo plano o una textura inmóvil.

Para este experimento hemos tomado un vídeo en el que un paraguas y unas letras pasan sobre un background sólido que no se altera. El análisis de color se ha realizado como en todos los casos vistos, las luminarias serán de nuevo el foseado de la Demo Room, y el layout elegido ha sido líneas verticales, ya que permite reconocer el desplazamiento de los quads más fácilmente.

A continuación podemos ver una sucesión de fotografías tomadas en ráfaga, para ver el comportamiento del movimiento a través de los frames del vídeo. Consideramos en el plano 2D la dirección “X” positiva hacia la derecha, y la dirección positiva “Y” hacia arriba.

Figura 28 – Desplazamiento de líneas verticales sobre "paraguas" 1

Figura 29 – Refresco de líneas verticales sobre "paraguas"

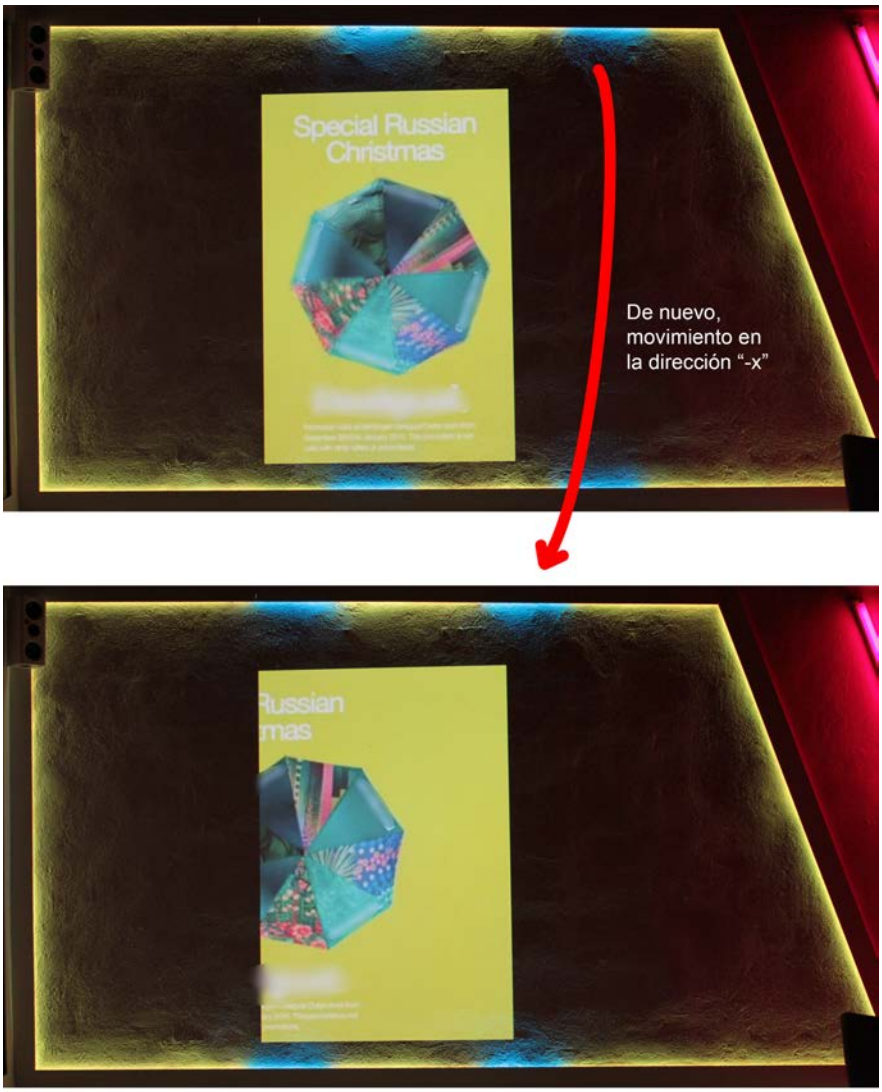


Figura 30 – Desplazamiento de líneas verticales sobre "paraguas" 2

Resultados: Si bien el movimiento no ha ocurrido en todos los frames en los que sí lo ha habido en el vídeo, se han presentado desplazamientos en la misma dirección y en el mismo momento. Se ha de tener en cuenta que en términos cuantitativos, el movimiento era poco (el paraguas se desplaza en un espacio equivalente a la mitad del ancho del vídeo, cada vez).

El análisis parece haber sido el correcto, así como la generación automática de contenido lumínico que responde al movimiento del vídeo; aunque esta última parte debería ser objetivo de algunas mejoras, como la cantidad de veces que se desplaza el layout en el tiempo que ocurre el movimiento.

TEST 5

Otras disposiciones lumínicas (pt. 1)

Dado que el foseado de la Demo Room solo nos proporciona un marco 2D de muy baja resolución, aunque muy útil debido a la iluminación de superficies que ofrece, es interesante probar otro tipo de luminarias.

En este caso, hemos utilizado unos tubos de fabricación casera por parte de Protopixel para sus early-performances, que en varios experimentos se han colocado de maneras distintas. En este en particular, se han puesto en el suelo, paralelamente a la vertical del vídeo. El vídeo será el paraguas en movimiento, en las mismas condiciones que el experimento anterior.

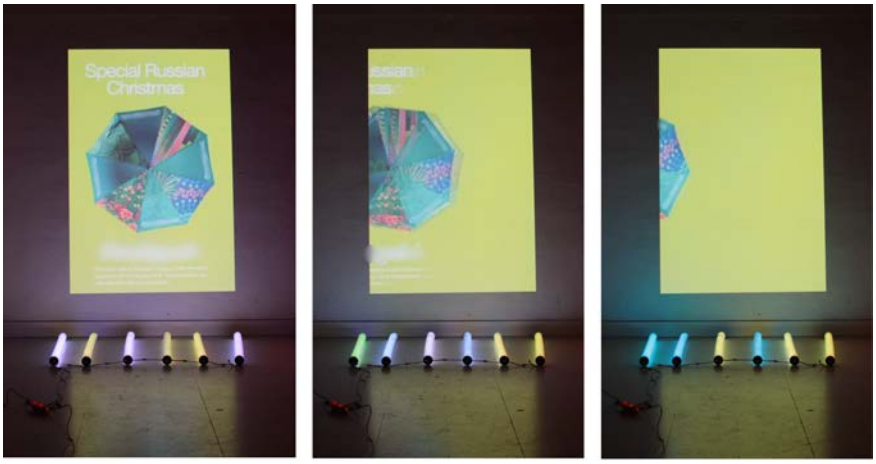


Figura 31 – Tubos led en el suelo con movimiento de líneas verticales sobre "paraguas"

Resultados: Lo que se muestra en la figura 30 es el paraguas desplazándose fuera del plano, como la iluminación también lo hace, y la par, se refresca el color de los tubos. La disposición de los tubos en el suelo es interesante, pareciendo que el vídeo flote sobre una plataforma. La opción de simular un reflejo también debe tener lugar, pero al realizarse un refresco del color sin criterios de posición del vídeo original, todavía no es posible.

TEST 6

Para finalizar, se han colgado los tubos y se han probado los cuatro distintos layouts con el vídeo inicial de "dancing girls". No ha habido análisis de movimiento, igual que en el primer experimento

Figura 32 – Tubos led colgantes con random sobre "dancing girls"

Figura 33 – Tubos led colgantes con líneas verticales sobre "dancing girls"

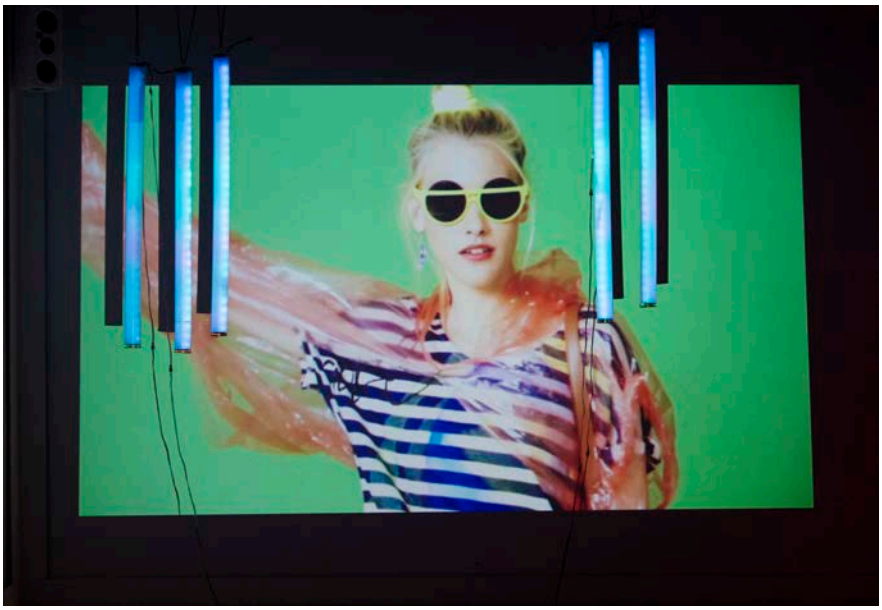


Figura 34 – Tubos led colgantes con líneas horizontales sobre "dancing girls"

Figura 35 – Tubos led colgantes con disposición radial sobre "dancing girls"

Resultados: Como puede observarse, los layouts son más identificables con los tubos que con el foseado. Los tubos ejercen menor luz, pero añaden una nueva dimensión (3D) al conjunto del contenido, si bien es cierto que lo ideal sería un mayor número de tubos y una disposición aleatoria a lo largo del espacio tridimensional por delante del vídeo.

6.2 DISCUSIÓN SOBRE LOS RESULTADOS

Tras todos los experimentos realizados, con sus correspondientes comparaciones y análisis de resultados, de manera muy genérica (ya que es subjetiva) podemos extraer:

- El sistema realiza un buen análisis y performance de la colorimetría. Si bien es posible adaptar el script para que se guarde la información de tres o más colores dominantes en el vídeo, tras las pruebas parece que lo más adecuado es tomar solo uno, o a lo sumo dos colores dominantes. La razón es que

los altos contrastes resultan más bruscos y menos agradables, por lo que en presencia de varios colores es mejor tomar el que más aparezca y acompañar así al vídeo. Por otro lado, si los colores resultan variaciones pequeñas de tono de uno mismo, esto añade expresividad. Facilitando la selección de que cantidad de colores extraer a la hora del análisis, este punto se podría mejorar.

- Si el contenido gráfico generado se representa en luminarias que proyectan la luz a una superficie, los layouts con menores divisiones ofrecen una variación de color mas suave a lo largo del área. Por otra parte, si los puntos de luz se ven, pasan a formar parte del contenido ("pintamos" en ellos con luz) y entonces puede ser más conveniente utilizar más resolución, aunque dependerá de cada caso en particular.

- Debe haber un compromiso entre la velocidad de *refresh* de los colores del layout y la interpolación. Si los colores se actualizan con mucha separación temporal, puede des-sincronizarse el contenido lumínico con el vídeo. Si por otra parte se realiza muy rápido, la interpolación entre colores (el paso intermedio entre ellos) se ve reducida y los cambios resultan bruscos y "alocados".

- El movimiento debería ajustarse para que se corresponda en duración temporal en el vídeo con el desplazamiento del layout, pero para los use-cases descritos cumple el objetivo principal.

6.3 FUTURE WORK

Este proyecto ha sido un punto de partida y un primer paso para lo que finalmente puede ser un sistema de diseño automático de iluminación. Una vez comprobado su funcionamiento, y antes de dar por terminado el proyecto, es momento de mirar hacia delante en el tiempo y pensar que mejoras o nuevos usos podrían partir del punto en el que se encuentra ahora el sistema.

6.3.1 Posibles mejoras

En primer lugar debería realizarse un estudio de resultados propiamente adecuado. Debido a la subjetividad de la evaluación de los experimentos, sería necesario realizar nuevas prueba con diferentes tipos de público, junto a unas encuestas (u otro tipo de estimaciones) para medir, de alguna forma un tanto más cuantitativa, la eficacia del sistema en cuanto a expresividad.

En segundo lugar, y de forma un tanto obvia para la generación de gráficos, se deberían añadir otros tipos de layouts que se ajusten mejor a diferentes disposiciones de luminarias.

Es necesario también explorar la implementación de este sistema (en parte, o todo entero) dentro de la empresa Emex para obtener un nuevo producto, resultado de la fusión entre el desarrollo propio de su plataforma de gestión de contenidos y este proyecto, obteniendo así un gestor de iluminación también.

Por último, implementar análisis de nuevos tipos de contenido, tales como audio y sensores, otorgaría nuevas posibilidades y enriquecimiento de experiencias, más allá del vídeo.

6.3.2 Otros posibles usos

Para pensar en otras formas de utilizar este sistema, debemos alejarnos de la imagen del vídeo con refuerzo lumínico. El sistema realiza un análisis de color y/o movimiento de un vídeo, y es posible que también extraiga información de sensores y audio en un futuro si así se mejora. También genera unos gráficos determinados.

El uso de estos datos y contenido gráfico ha sido enfocado en la finalidad que les da Protopixel, dentro de su framework, pero una de las ventajas del diseño en este proyecto es la autonomía del sistema. Tanto el análisis como el creado de contenido pueden ser portados a cualquier lenguaje y plataforma. Es entonces cuando podemos pensar en usos totalmente diferentes.

En primera instancia, y en un área cercana a la del proyecto, pueden tomarse, por ejemplo, datos en tiendas para generar contenido que acompañe a la música, a la gente que se encuentra en el interior, en el escaparate… y puede ser tratado como luz, como vídeo, como movimiento (uso de motores e instalaciones "kinetic"). Las posibilidades son muchas y muy dispares para hablar de todas ellas aquí, pero es conveniente saber cuánto de multifuncional puede ser la base de este proyecto.

7. CONCLUSIONES DEL PROYECTO

Una vez terminado el proyecto, es momento de evaluar los resultados y también repasar que se ha aprendido durante la elaboración del mismo.

Los resultados, en general, han sido satisfactorios. A pesar de ser una versión inicial, que dista mucho del aspecto que tendría un producto final, es positivo el hecho de que el funcionamiento es correcto, y realmente existe una diferencia notable entre ver simplemente los vídeos y verlos con el refuerzo lumínico. Así mismo, el sistema es totalmente automático (aparte de las variables que responden a criterios de diseño), lo que supone un avance en dirección a una solución que pueda cubrir esta carencia en el lighting-design de hoy en día. Pero aún queda mucho por andar.

Durante este proyecto también se ha aprendido que es necesario, desde el primer momento, trabajo de campo y pruebas prácticas, ya que la "prueba y error", en muchas ocasiones, es la mejor forma de llegar a un resultado óptimo. También han aparecido diferentes facetas de la luz, tanto en el espacio como en el desarrollo de la forma de mostrarla, algo que no se ve mucho en ingeniería; es decir, se ha alcanzado un equilibrio entre ingeniería y diseño estético, que resulta en una combinación que aporta mucho valor al proyecto.

La investigación y documentación, junto con la ayuda de los profesores, han mostrado una forma de trabajar muy eficiente, así como el aprendizaje de los lenguajes de programación con los que había poca familiaridad.

Por último, el uso de la luz en sistemas interactivos modernos ha ido aumentado últimamente y en este proyecto queda constancia de lo potente que puede ser como herramienta, por lo que es interesante desarrollar, no solo lo visto en este trabajo, si no cualquier otro modo de implementarla en el diseño de experiencias.

REFERENCIAS

[1] Tensegrity Lights by Michal Macieg. (2014, December 30). In Mocoloco. Retrieved June 15, 2016, from http://mocoloco.com/tensegrity-lights-by-michal-maciej/

[2] Patrick Woodroffe (lighting designer). (2016, April 24). In Wikipedia, The Free Encyclopedia. Retrieved June 15, 2016, from https://en.wikipedia.org/w/index.php?title=Patrick_Woodroffe_(lighting_designer)&oldid=716849748

[3] Super nature: ‘New Angles' interactive light installation. (2010, September 10). In Designboom. Retrieved June 15, 2016, from http://www.designboom.com/design/super-nature-new-angles-interactive-light-installation/

[4] Light Act - Systems. In Light Act - Systems. Retrieved June 15, 2016 from http://lightact-systems.com

[5] Interactive Illumination. (2014, February 6). In Yanko Design. Retrieved June 15, 2016, from http://www.yankodesign.com/2014/06/02/interactive-illumination/

[6] Wasbah Lights. (2015, July). In Wasbah Lights. Retrieved June 15, 2016, from ttp://www.thewabashlights.com

[7] Protopixel. (2015). In Protopixel. Retrieved June 15, 2016, from http://protopixel.net

[8] Emotion Experience. (2016). In Emotion Experience. Retrieved June 15, 2016, from http://emotionexperience.com

[9] OpenCV. In OpenCV. Retrieved June 15, 2016, from http://opencv.org/about.html

[10] OpenGL. In OpenGL. Retrieved June 15, 2016, from https://www.opengl.org/about/

[11] Processing. In Processing. Retrieved June 15, 2016, from https://processing.org

[12] Primitive. (2015, July 15). In OpenGL Wiki. Retreived June 15, 2016, from https://www.opengl.org/wiki/Primitive#Quads

[13] OpenSound Control. (2014, August 26). In Wikipedia, The Free Encyclopedia. Retrieved June 15, 2016, from https://es.wikipedia.org/w/index.php?title=OpenSound_Control&oldid=76590129

[14] Hue. (2016, May 24). In Wikipedia, The Free Encyclopedia. Retrieved June 15, 2016, from https://en.wikipedia.org/w/index.php?title=Hue&oldid=721861321

[15] K-means clustering. (2016, June 6). In Wikipedia, The Free Encyclopedia. Retrieved June 15, 2016, from https://en.wikipedia.org/w/index.php?title=K-means_clustering&oldid=723949335

[16] Anthony H. Dekker. (1994). Kohonen neural networks for optimal colour quantization. In Network: Computation in Neural Systems, Institute of Physics Publishing. Retrieved June 15, 2016, from http://members.ozemail.com.au/~dekker/NeuQuant.pdf

[17] Optical flow. (2016, June 4). In Wikipedia, The Free Encyclopedia. Retrieved June 15, 2016, from https://en.wikipedia.org/w/index.php?title=Optical_flow&oldid=723709683

[18] UIOWA. (1997, May 19). In Digital Image Processing, Image Analysis and Understanding, Chapter 14, Motion Analysis: Optical flow. Retrieved June 15, 2016, from http://user.engineering.uiowa.edu/~dip/lecture/Motion2.html

[19] OpenFrameworks. (2016, June 10). In Wikipedia, The Free Encyclopedia. Retrieved June 15, 2016, from https://en.wikipedia.org/w/index.php?title=OpenFrameworks&oldid=724622431

APÉNDICE

Se incluirán aquí partes del código que se ha desarrollado, con la intención de que aquellas personas que estén familiarizadas con la programación puedan ver (y utilizar) la implementación técnica.

Optical Flow en Processing

```
void draw() {
  background(0);
  opencv.loadImage(video);
  // get the frame
  img = opencv.getSnapshot();
  // calculate movement for each zone (pixel)
  opencv.calculateOpticalFlow();

  // draw the movement for each zone (pixel)
  image(video, 0, 0); translate(video.width,0); stroke(255,0,0);
  opencv.drawOpticalFlow();

 // get average flow
  PVector aveFlow =
opencv.getAverageFlowInRegion((video.width/2-video.width/4),
(video.height/2-video.height/4), (video.width/2), (video.height/2) );
  int flowScale = 50;

 // draw average flow
  stroke(255);  strokeWeight(2);
  line(video.width/2, video.height/2, video.width/2 +
aveFlow.x*flowScale, video.height/2 + aveFlow.y*flowScale);
```

```
  if (video.time() < video.duration())
  {   // Writting down the color info por each frame
          log.write(video.time()+","+aveFlow.x*flowScale+","+aveFl
ow.y*flowScale+"\n");
  }

 else if (video.time() >= video.duration())
    { log.close(); exit();}
```

Color Quantization en Processing

```
void setup() {

video = new Movie (this, "XXXX");
vidW = 288; vidH = 384;

cv = new OpenCV(this, vidW, vidH);
log = new Log("videoinfo_col.txt");
canvas = createGraphics(vidW/4, vidH/4, P3D);
PJOGL.profile = 1;
}

void draw() {

background (0);
int colorCount=3;

// load de video on a different canvas, so loadPixels has the same
pixels as the video
 canvas.beginDraw();
 canvas.background(0);
 canvas.image(img, 0, 0, vidW/4, vidH/4);
 canvas.loadPixels();
 pixels=canvas.pixels;
 nq=new NeuQuant(colorCount,false);
```

```
 nq.init();
 canvas.endDraw();
 image(canvas, width-vidW/4, (height-vidH/4)-20);

 for (int i=0;i<pixels.length;i++)
   {pixels[i]=nq.convert(pixels[i]);}

  int[] map=nq.getColorMap();
  int sz=width/map.length;
  for (int i=0;i<map.length;i++)
  {
    fill(map[i]);
    rect(i*sz,height-16,(i+1)*sz,height);
  }

  mostcolor1 = round(hue(color(map[0])));
  col1_s= round(saturation(color(map[0])));
  mostcolor2 = round(hue(color(map[1])));
  col2_s= round(saturation(color(map[1])));
  mostcolor3 = round(hue(color(map[2])));
  col3_s= round(saturation(color(map[2])));

// Store info

if (video.time() < video.duration())
  {
      log.write( video.time()+
                 ","+
                 int(red(color(mostcolor1, col1_s, 200)))+","+
                 int(green(color(mostcolor1, col1_s, 200)))+","+
                 int(blue(color(mostcolor1, col1_s, 200)))+","+
```

```
            int(red(color(mostcolor2, col2_s, 200)))+","+
            int(green(color(mostcolor2, col2_s, 200)))+","+
            int(blue(color(mostcolor2, col2_s, 200)))+","+

            int(red(color(mostcolor3, col3_s, 200)))+","+
            int(green(color(mostcolor3, col3_s, 200)))+","+
            int(blue(color(mostcolor3, col3_s, 200)))+"\n");
  }

else if (video.time() >= video.duration())
   { log.close(); exit(); }

}
```

Lectura de información en Python (funciones “read”)

```
# Split the text file in lines
def readVidFile (log):

   with open(log, "r") as ins:
      linesFrames = []
      for line in ins:
         linesFrames.append(line)

   return (linesFrames)

# Split the lines in values
def readVidFeaturesCol (listlog, frame):

   stringFeature = listlog[frame]

   tim = stringFeature.rsplit(',', 10)[0]
   c1r = stringFeature.rsplit(',', 10)[1]
   c1g = stringFeature.rsplit(',', 10)[2]
   c1b = stringFeature.rsplit(',', 10)[3]
```

```
    c2r = stringFeature.rsplit(',', 10)[4]
    c2g = stringFeature.rsplit(',', 10)[5]
    c2b = stringFeature.rsplit(',', 10)[6]
    c3r = stringFeature.rsplit(',', 10)[7]
    c3g = stringFeature.rsplit(',', 10)[8]
    c3b = stringFeature.rsplit(',', 10)[9]

    color1 = ofColor(int(c1r), int(c1g), int(c1b))
    color2 = ofColor(int(c2r), int(c2g), int(c2b))
    color3 = ofColor(int(c3r), int(c3g), int(c3b))

    return ([tim, color1, color2, color3])

def readVidFeaturesMov (listlog, frame):

    stringFeature = listlog[frame]

    tim = stringFeature.rsplit(',', 3)[0]
    xmov = stringFeature.rsplit(',', 3)[1]
    ymov = stringFeature.rsplit(',', 3)[2]


    return ([tim, xmov, ymov])

# compare current elapsed time with time-stamps from file
def findFrame(videoTime, listlog, a):

    # array the same size of frames
    frameTimes = [0.0 for i in range (len(listlog)-1)]
    # initialize the frame to return
    frametoget = 0
    # set de euclidian 1D distance to de maximun (first vs last frame)
    dist = a (listlog, (len(listlog)-1))[0]

    #store the frames
```

```
    for i in range (len(listlog)-1):
        frameTimes[i] = a (listlog, i)[0]

    for i in range (len(frameTimes)-1):
        # if the euclidian distance between current frame in the list is
smaller than the previous one
        # the new "closer time" frame is set
        if abs(float(frameTimes[i]) - videoTime) < dist:
            frametoget = i
            dist = abs(float(frameTimes[i])- videoTime)
        else:
            pass

    return frametoget
```

Clase: Matriz de quads en Python

```
class led_Matrix:
    def __init__(self, x, y):
        self.sizex = x
        self.sizey = y
        self.matrixcol = [[(ofColor.black) for i in range(self.sizex)] for
j in range(self.sizey)]

    # disposicion random de los quads
    def spreadColor(self, col1, col2, col3):

        colorchoose=0
        matrixc = [[(ofColor.black) for i in range(self.sizex)] for j in
range(self.sizey)]

        for i in range(self.sizex):
            for j in range(self.sizey):
```

```
            colorchoose = randint(1,3)

            if colorchoose == 1:
                matrixc[i][j] = ofColor(col1.r,col1.g,col1.b)
            if colorchoose == 2:
                matrixc[i][j] = ofColor(col2.r,col2.g,col2.b)
            if colorchoose == 3:
                matrixc[i][j] = ofColor(col3.r,col3.g,col3.b)

        return (matrixc)
```

def verticalLines(self, col1, col2, col3):

```
        colorchoose=0
        matrixc = [[(ofColor.black) for i in range(self.sizex)] for j in
range(self.sizey)]

        for i in range(0, self.sizex, int(round(self.sizex/matrixDvsn))):

            colorchoose = randint(1,3)

            for j in range(self.sizey):

                for k in range(int(round(self.sizex/matrixDvsn))):

                    if colorchoose == 1:
                        matrixc[i+k][j] = ofColor(col1.r,col1.g,col1.b)

                    if colorchoose == 2:
                        matrixc[i+k][j] = ofColor(col2.r,col2.g,col2.b)

                    if colorchoose == 3:
                        matrixc[i+k][j] = ofColor(col3.r,col3.g,col3.b)
```

```
        return (matrixc)

    def horizontalLines(self, col1, col2, col3):

        colorchoose=0
        matrixc = [[(ofColor.black) for i in range(self.sizex)] for j in
range(self.sizey)]

        for j in range(0, self.sizey, int(round(self.sizey/matrixDvsn))):

            colorchoose = randint(1,3)

            for i in range(self.sizex):

                for k in range(int(round(self.sizey/matrixDvsn))):

                    if colorchoose == 1:
                        matrixc[i][j+k] = ofColor(col1.r,col1.g,col1.b)

                    if colorchoose == 2:
                        matrixc[i][j+k] = ofColor(col2.r,col2.g,col2.b)

                    if colorchoose == 3:
                        matrixc[i][j+k] = ofColor(col3.r,col3.g,col3.b)


        return (matrixc)

    # disposicion radial de los colores, coloreando la matriz por
completa de fuera a dentro
    def radialColor(self, col1, col2, col3):

        colorchoose = 0
        matrixc = [[(ofColor.black) for i in range(self.sizex)] for j in
range(self.sizey)]
```

```
if radios == 1:

    radio1x = int(round(self.sizex/matrixDvsn))
    radio2x = int(round(self.sizex/(matrixDvsn/2)))
    radio1y = int(round(self.sizex/matrixDvsn))
    radio2y = int(round(self.sizex/(matrixDvsn/2)))

    # radio exterior
    colorchoose = randint(1,3)
    for i in range(self.sizex):
        for j in range(self.sizey):

            if colorchoose == 1:
                matrixc[i][j] = ofColor(col1.r,col1.g,col1.b)
            if colorchoose == 2:
                matrixc[i][j] = ofColor(col2.r,col2.g,col2.b)
            if colorchoose == 3:
                matrixc[i][j] = ofColor(col3.r,col3.g,col3.b)

    # radio medio
    colorchoose = randint(1,3)
    for i in range(0 + radio1x , self.sizex - radio1x):
        for j in range(0 + radio1y , self.sizey - radio1y):

            if colorchoose == 1:
                matrixc[i][j] = ofColor(col1.r,col1.g,col1.b)
            if colorchoose == 2:
                matrixc[i][j] = ofColor(col2.r,col2.g,col2.b)
            if colorchoose == 3:
                matrixc[i][j] = ofColor(col3.r,col3.g,col3.b)

    # radio interior
    colorchoose = randint(1,3)
    for i in range(0 + radio2x , self.sizex - radio2x):
```

```
        for j in range(0 + radio2y , self.sizey - radio2y):

            if colorchoose == 1:
                matrixc[i][j] = ofColor(col1.r,col1.g,col1.b)
            if colorchoose == 2:
                matrixc[i][j] = ofColor(col2.r,col2.g,col2.b)
            if colorchoose == 3:
                matrixc[i][j] = ofColor(col3.r,col3.g,col3.b)

elif radios == 2:

    radio1x = int(round(self.sizex/matrixDvsn))
    radio2x = int(round(self.sizex/(matrixDvsn/3)))
    radio3x = int(round(self.sizex/(matrixDvsn/2))+4)
    radio1y = int(round(self.sizex/matrixDvsn))
    radio2y = int(round(self.sizex/(matrixDvsn/3)))
    radio3y = int(round(self.sizex/(matrixDvsn/2))+4)


    # radio exterior
    colorchoose = randint(1,3)
    for i in range(self.sizex):
        for j in range(self.sizey):

            if colorchoose == 1:
                matrixc[i][j] = ofColor(col1.r,col1.g,col1.b)
            if colorchoose == 2:
                matrixc[i][j] = ofColor(col2.r,col2.g,col2.b)
            if colorchoose == 3:
                matrixc[i][j] = ofColor(col3.r,col3.g,col3.b)

    # radio medio
    colorchoose = randint(1,3)
    for i in range(0 + radio1x , self.sizex - radio1x):
        for j in range(0 + radio1y , self.sizey - radio1y):
```

```
            if colorchoose == 1:
                matrixc[i][j] = ofColor(col1.r,col1.g,col1.b)
            if colorchoose == 2:
                matrixc[i][j] = ofColor(col2.r,col2.g,col2.b)
            if colorchoose == 3:
                matrixc[i][j] = ofColor(col3.r,col3.g,col3.b)

    # radio interior
    colorchoose = randint(1,3)
    for i in range(0 + radio2x , self.sizex - radio2x):
        for j in range(0 + radio2y , self.sizey - radio2y):

            if colorchoose == 1:
                matrixc[i][j] = ofColor(col1.r,col1.g,col1.b)
            if colorchoose == 2:
                matrixc[i][j] = ofColor(col2.r,col2.g,col2.b)
            if colorchoose == 3:
                matrixc[i][j] = ofColor(col3.r,col3.g,col3.b)

    # radio interior 2
    colorchoose = randint(1,3)
    for i in range(0 + radio3x , self.sizex - radio3x):
        for j in range(0 + radio3y , self.sizey - radio3y):

            if colorchoose == 1:
                matrixc[i][j] = ofColor(col1.r,col1.g,col1.b)
            if colorchoose == 2:
                matrixc[i][j] = ofColor(col2.r,col2.g,col2.b)
            if colorchoose == 3:
                matrixc[i][j] = ofColor(col3.r,col3.g,col3.b)

elif radios == 3:

    radio1x = int(round(self.sizex/matrixDvsn))
```

```
radio2x = int(round(self.sizex/(matrixDvsn/4)))
radio3x = int(round(self.sizex/(matrixDvsn/3))+1)
radio4x = int(round(self.sizex/(matrixDvsn/2))+2)
radio1y = int(round(self.sizex/matrixDvsn))
radio2y = int(round(self.sizex/(matrixDvsn/4)))
radio3y = int(round(self.sizex/(matrixDvsn/3))+1)
radio4y = int(round(self.sizex/(matrixDvsn/2))+2)


# radio exterior
colorchoose = randint(1,3)
for i in range(self.sizex):
    for j in range(self.sizey):

        if colorchoose == 1:
            matrixc[i][j] = ofColor(col1.r,col1.g,col1.b)
        if colorchoose == 2:
            matrixc[i][j] = ofColor(col2.r,col2.g,col2.b)
        if colorchoose == 3:
            matrixc[i][j] = ofColor(col3.r,col3.g,col3.b)

# radio medio
colorchoose = randint(1,3)
for i in range(0 + radio1x , self.sizex - radio1x):
    for j in range(0 + radio1y , self.sizey - radio1y):

        if colorchoose == 1:
            matrixc[i][j] = ofColor(col1.r,col1.g,col1.b)
        if colorchoose == 2:
            matrixc[i][j] = ofColor(col2.r,col2.g,col2.b)
        if colorchoose == 3:
            matrixc[i][j] = ofColor(col3.r,col3.g,col3.b)

# radio interior
colorchoose = randint(1,3)
```

```
    for i in range(0 + radio2x , self.sizex - radio2x):
        for j in range(0 + radio2y , self.sizey - radio2y):

            if colorchoose == 1:
                matrixc[i][j] = ofColor(col1.r,col1.g,col1.b)
            if colorchoose == 2:
                matrixc[i][j] = ofColor(col2.r,col2.g,col2.b)
            if colorchoose == 3:
                matrixc[i][j] = ofColor(col3.r,col3.g,col3.b)

    # radio interior 2
    colorchoose = randint(1,3)
    for i in range(0 + radio3x , self.sizex - radio3x):
        for j in range(0 + radio3y , self.sizey - radio3y):

            if colorchoose == 1:
                matrixc[i][j] = ofColor(col1.r,col1.g,col1.b)
            if colorchoose == 2:
                matrixc[i][j] = ofColor(col2.r,col2.g,col2.b)
            if colorchoose == 3:
                matrixc[i][j] = ofColor(col3.r,col3.g,col3.b)

    colorchoose = randint(1,3)
    for i in range(0 + radio4x , self.sizex - radio4x):
        for j in range(0 + radio4y , self.sizey - radio4y):

            if colorchoose == 1:
                matrixc[i][j] = ofColor(col1.r,col1.g,col1.b)
            if colorchoose == 2:
                matrixc[i][j] = ofColor(col2.r,col2.g,col2.b)
            if colorchoose == 3:
                matrixc[i][j] = ofColor(col3.r,col3.g,col3.b)

return (matrixc)
```

```
def leprUpdate(self, matrix1, interpolStep):
    #interpolStep = 0.03

    for i in range(self.sizex):
        for j in range(self.sizey):
            oldcolor = self.matrixcol[i][j]
            nucolor = matrix1[i][j]
            btwncol = oldcolor.lerp(nucolor,interpolStep)
            self.matrixcol[i][j] = ofColor(btwncol.r, btwncol.g,
btwncol.b)
            #self.matrixcol[i][j] = oldcolor * (1-interpolStep) +
nucolor*interpolStep

def shiftMatrix(self, xdir, ydir, oldmatrix):

    xd = float(xdir)
    yd = float(ydir)

    selectcase = 0
    matrixc = [[(ofColor.black) for i in range(self.sizex)] for j in
range(self.sizey)]

    if xd < -10.0 and (yd < 10.0 and yd > -10.0):
        selectcase = 1
    elif xd < -10.0 and yd > -10.0:
        selectacase = 2
    elif (xd > -10.0 and xd < 10.0) and yd < -10.0:
        selectcase = 3
    elif xd > 10.0 and (yd < 10 and yd > -10.0):
        selectcase = 4
    elif xd > 10.0 and (yd < 10 and yd > -10.0):
        selectcase = 5
    elif xd > 10.0 and yd > 10.0:
        selectcase = 6
    elif (xd > -10.0 and xd < 10.0) and yd > 10.0:
```

```
    selectcase = 7
elif xd < -10.0 and yd > 10.0:
    selectcase = 8


print selectcase


if selectcase == 0:

    matrixc = oldmatrix

if selectcase == 1:

    for i in range(self.sizex):
        for j in range(self.sizey):

            if i < self.sizex-1:
                matrixc[i][j] = oldmatrix[i+1][j]
            if i == self.sizex-1:
                matrixc[i][j] = oldmatrix[0][j]

if selectcase == 2:

    for i in range(self.sizex):
        for j in range(self.sizey):

            if i < self.sizex-1 and j < self.sizey-1:
                matrixc[i][j] = oldmatrix[i+1][j+1]
            if i == self.sizex-1 and j < self.sizey-1:
                matrixc[i][j] = oldmatrix[0][j+1]
            if i < self.sizex-1 and j == self.sizey-1:
                matrixc[i][j] = oldmatrix[i+1][0]
            if i == self.sizex-1 and j == self.sizey-1:
                matrixc[i][j] = oldmatrix[0][0]
```

```
if selectcase == 3:

    for i in range(self.sizex):
        for j in range(self.sizey):

            if j < self.sizey-1:
                matrixc[i][j] = oldmatrix[i][j+1]
            if j == self.sizey-1:
                matrixc[i][j] = oldmatrix[0][j]

if selectcase == 4:

    for i in range(self.sizex):
        for j in range(self.sizey):

            if i > 0 and j < self.sizey-1:
                matrixc[i][j] = oldmatrix[i-1][j+1]
            if i == 0 and j < self.sizey-1:
                matrixc[i][j] = oldmatrix[self.sizex-1][j+1]
            if i > 0 and j == self.sizey-1:
                matrixc[i][j] = oldmatrix[i-1][0]
            if i == 0 and j == self.sizey-1:
                matrixc[i][j] = oldmatrix[self.sizex-1][0]

if selectcase == 5:

    for i in range(self.sizex):
        for j in range(self.sizey):

            if i < self.sizex and i > 0:
                matrixc[i][j] = oldmatrix[i-1][j]
            if i == 0:
                matrixc[i][j] = oldmatrix[self.sizex-1][j]
```

```
if selectcase == 6:

    for i in range(self.sizex):
        for j in range(self.sizey):

            if i > 0 and j > 0:
                matrixc[i][j] = oldmatrix[i-1][j-1]
            if i == 0 and j > 0:
                matrixc[i][j] = oldmatrix[self.sizex-1][j-1]
            if i > 0 and j == 0:
                matrixc[i][j] = oldmatrix[i-1][self.sizey-1]
            if i == 0 and j == 0:
                matrixc[i][j] = oldmatrix[self.sizex-1][self.sizey-1]


if selectcase == 7:

    for i in range(self.sizex):
        for j in range(self.sizey):

            if j < self.sizey and j > 0:
                matrixc[i][j] = oldmatrix[i][j-1]
            if j == 0:
                matrixc[i][j] = oldmatrix[i][self.sizey-1]

if selectcase == 8:

    for i in range(self.sizex):
        for j in range(self.sizey):

            if i < self.sizex-1 and j > 0:
                matrixc[i][j] = oldmatrix[i+1][j-1]
            if i == self.sizex-1 and j > 0:
                matrixc[i][j] = oldmatrix[i+1][self.sizey-1]
            if i < self.sizex-1 and j == 0:
```

```
            matrixc[i][j] = oldmatrix[0][j-1]
        if i == self.sizex-1 and j == 0:
            matrixc[i][j] = oldmatrix[0][self.sizey-1]

    return (matrixc)


def draw_led_Matrix(self, quadSize):
    for i in range(self.sizex):
        for j in range(self.sizey):
            ofSetColor(self.matrixcol[i][j])
            ofRect(i*quadSize, j*quadSize, quadSize, quadSize)
```